SIAM J. Matrix Anal. Appl.
Vol. 20, No. 1, pp. 1–13

POWER ADDITIVITY AND ORTHOGONALITY*

ROBERT E. HARTWIG† AND PETER ŠEMRL‡

Abstract. The relation between rank additivity and orthogonality is analyzed. The central role played by Cochran's theorem is illustrated and the properties of orthogonal and power-additive families are investigated.

Key words. power additivity, rank additivity, orthogonality, Cochran's theorem

AMS subject classifications. 15A21, 15A24

PII. S089547989731498X

1. Introduction.

Suppose we are given a family $\{A_i\}_{i=1}^{i=s}$ of complex $n \times n$ matrices and define $A = A_1 + \cdots + A_s$. We say that the family is *power additive* if

$$A^k = A_1^k + \cdots + A_s^k \quad \text{for all} \quad k = 1, 2, \ldots, \tag{1}$$

and that the family is *orthogonal* if

$$A_i A_j = 0 \quad \text{for} \quad i \neq j. \tag{2}$$

It is clear that the orthogonality implies power additivity, but the converse need not be true. Indeed, the latter may be seen from the example where

$$A_1 = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad A_3 = \begin{bmatrix} 0 & -1 \\ 0 & 0 \end{bmatrix}. \tag{3}$$

A family $\{A_i\}_{i=1}^{i=s}$ is rank additive relative to $A = A_1 + \cdots + A_s$ if

$$\operatorname{rank} A = \sum_{i=1}^{s} \operatorname{rank} A_i.$$

It is easy to find examples showing that rank additivity does not imply orthogonality and that orthogonality does not imply rank additivity. Extensions of the algebraic version of Cochran's statistical theorem (see [1, 4, 5]) play a crucial role in linking these two concepts. One part of this theorem states that if a family $\{A_i\}_{i=1}^{i=s}$ is rank additive and A is an idempotent, then this is an orthogonal family. Motivated by this statement, we say that a matrix A has property (C) if every rank additive family $\{A_i\}_{i=1}^{i=s}$, whose sum is A, is orthogonal.

How far can one generalize Cochran's theorem? To answer this we have to characterize the set of all matrices having property (C). We will show that a matrix A has property (C) if and only if it is a scalar multiple of an idempotent or a square-zero matrix.

[*] Received by the editors January 5, 1997; accepted for publication (in revised form) by G. P. Styan June 8, 1997; published electronically September 15, 1998.

http://www.siam.org/journals/simax/20-1/31498.html

[†] Department of Mathematics, North Carolina State University, Box 8205, Raleigh, NC 27695-8205 (hartwig@math.ncsu.edu).

[‡] Faculty of Mechanical Engineering, University of Maribor, Smetanova 17, 2000 Maribor, Slovenia (peter.semrl.uni_lj.si). The work of this author was supported by a grant from the Ministry of Science and Technology of Slovenia.

The second part of the algebraic version of Cochran's theorem states that an orthogonal family of idempotents is rank additive. Once again it seems natural to ask how far we can extend this part of Cochran's theorem. In other words, when does orthogonality imply rank additivity? To answer this question we have to examine the structure of orthogonal families of matrices. This is the main objective of this paper. When trying to obtain structural results for orthogonal families of matrices, we noticed that some of our proofs also work for power-additive families of matrices. The investigation of power-additive families of matrices might be of some independent interest because the concept of power additivity surfaces also in modular arithmetic.

Throughout this note, all matrices will be complex $n \times n$ matrices, and we shall denote the range space, the null space, the characteristic polynomial, the minimal polynomial, and the adjoint of A by $R(A)$, $N(A)$, Δ_A, ψ_A, and A^*, respectively. The algebraic multiplicity of λ as an eigenvalue of A is denoted by $n_A(\lambda)$. It is defined to be zero if λ is not an eigenvalue of A. We shall use A^D and A^- to, respectively, denote the Drazin inverse and an inner inverse (see [2, 3]). The core and nilpotent parts of A are defined and denoted by $C_A = A^2A^D$ and $N_A = A(I-AA^D)$, respectively. It is well known that these matrices are orthogonal polynomials in A and that $A^n = C_A^n$ is a group matrix; that is, a matrix which has a group inverse. From these observations, it is clear that $AB = 0$ if and only if $C_AB = 0$ and $N_AB = 0$ which is further equivalent to $C_AC_B = C_AN_B = N_AC_B = N_AN_B = 0$. Hence, if $\{A_i\}_{i=1}^{i=s}$ is orthogonal, then so are $\{C_{A_i}\}_{i=1}^{i=s}$ and $\{N_{A_i}\}_{i=1}^{i=s}$.

2. Orthogonality. We start by characterizing property (C).

Theorem 2.1. *A complex $n \times n$ matrix A has property (C) if and only if it is a scalar multiple of an idempotent or a square-zero matrix.*

Proof. If A is a scalar multiple of an idempotent, then it has property (C) by Cochran's theorem. Assume now that A is a square-zero matrix and that $\{A_i\}_{i=1}^{i=s}$ is a rank additive family with $A = A_1 + \cdots + A_s$. It follows [5, Lemma 6] that $R(A) = R(A_1) \oplus \cdots \oplus R(A_s)$. If $x \in R(A_i)$, $1 \le i \le s$, then $x \in R(A)$, and hence $Ax = 0 = A_1x + \cdots + A_sx$. As $R(A)$ is a direct sum of the $R(A_i)$'s, we have $A_1x = \cdots = A_sx = 0$. Hence, $\{A_i\}_{i=1}^{i=s}$ is an orthogonal family of square-zero matrices.

Assume now that A is neither a scalar multiple of an idempotent nor square-zero. We will show that there exist A_1 and $A_2 = A - A_1$ which are rank additive to A but are *not* orthogonal. The first case we will consider is that A has at least two nonzero eigenvalues. Rank additivity and orthogonality are invariant under similarity. So, applying the Jordan canonical form we can assume, with no loss of generality, that A is of the form

$$A = \begin{bmatrix} \lambda I + N_1 & 0 & 0 \\ 0 & \mu I + N_2 & 0 \\ 0 & 0 & A_3 \end{bmatrix},$$

where N_1 and N_2 are nilpotents and $0 \ne \lambda \ne \mu \ne 0$. Some bordering zeroes and A_3 may be absent. Obviously,

$$A_1 = \begin{bmatrix} \lambda I + N_1 & X & 0 \\ 0 & 0 & 0 \\ 0 & 0 & A_3 \end{bmatrix} \quad \text{and} \quad A_2 = \begin{bmatrix} 0 & -X & 0 \\ 0 & \mu I + N_2 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

is a rank additive splitting of A which is not orthogonal if $X \ne 0$. If A has exactly one nonzero eigenvalue but is not a scalar multiple of an idempotent, then it is similar to either

$$\sum_{i=1}^{k} \lambda E_{ii} + E_{12} + \sum_{i=2}^{n-1} \varepsilon_i E_{i,i+1}$$

or

$$\sum_{i=1}^{k} \lambda E_{ii} + E_{k+1,k+2} + \sum_{i=k+2}^{n-1} \varepsilon_i E_{i,i+1}.$$

Here, λ is a nonzero complex number, E_{ij} denotes the matrix having 1 in the (i,j)th position as the only nonzero entry, all the ε_i's are either 0 or 1, and $2 \leq k \leq n$ in the first case, while $1 \leq k \leq n-2$ in the second case. In the first case, we define $A_1 = \lambda E_{11} + E_{12}$ and $A_2 = A - A_1$ to obtain a nonorthogonal rank additive splitting of A. In the second case, we define A_1 by $\sum_{i=1}^{k} \lambda E_{ii} + E_{k+1,k}$ and $A_2 = A - A_1$ to show that A does not have property (C). It remains to consider the case where A is a nilpotent of index at least 3. Then A is similar to $E_{12} + E_{23} + \sum_{i=3}^{n-1} \varepsilon_i E_{i,i+1}$ with $\varepsilon_i \in \{0,1\}$. Matrices E_{12} and $A - E_{12}$ are rank additive to A, but not orthogonal, completing the characterization of (C). □

Let $\{A_i\}_{i=1}^{i=s}$ be a family of complex $n \times n$ matrices, and define $A = A_1 + \cdots + A_s$ and $B_i = A - A_i$, $i = 1, \ldots, s$. We will call a family $\{A_i\}_{i=1}^{k=s}$ left complementary orthogonal if $A_i B_i = 0$ for all $i = 1, \ldots, s$. It is clear that every orthogonal family is left complementary orthogonal. To show that the converse is not true, one can consider matrices

$$A_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad A_3 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

The next result is a structural theorem for a left complementary orthogonal family of matrices.

THEOREM 2.2. *Let $\{A_i\}_{i=1}^{i=s}$ be a left complementary orthogonal family of complex $n \times n$ matrices. Then there exists an invertible matrix P such that*

$$A_1 = P \begin{bmatrix} U_1 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 0 & \ldots & 0 & G_{12} \\ 0 & 0 & 0 & \ldots & 0 & G_{13} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 0 & G_{1s} \\ 0 & 0 & 0 & \ldots & 0 & N_1 \end{bmatrix} P^{-1},$$

$$A_2 = P \begin{bmatrix} 0 & 0 & 0 & \ldots & 0 & G_{21} \\ 0 & U_2 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 0 & \ldots & 0 & G_{23} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 0 & G_{2s} \\ 0 & 0 & 0 & \ldots & 0 & N_2 \end{bmatrix} P^{-1},$$

$$\vdots$$

$$A_s = P \begin{bmatrix} 0 & 0 & 0 & \dots & 0 & G_{s1} \\ 0 & 0 & 0 & \dots & 0 & G_{s2} \\ 0 & 0 & 0 & \dots & 0 & G_{s3} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & U_s & 0 \\ 0 & 0 & 0 & \dots & 0 & N_s \end{bmatrix} P^{-1},$$

$$A = \sum_{i=1}^{s} A_i = P \begin{bmatrix} U_1 & 0 & 0 & \dots & 0 & 0 \\ 0 & U_2 & 0 & \dots & 0 & 0 \\ 0 & 0 & U_3 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & U_s & 0 \\ 0 & 0 & 0 & \dots & 0 & N \end{bmatrix} P^{-1},$$

where $U_1, \dots, U_s$ are invertible matrices and $\{N_i\}_{i=1}^{i=s}$ is a left complementary orthogonal family of nilpotents.

Proof. Let us define idempotents $E_i = A_i^D A_i$, $i = 1, \dots, s$, and $E = A^D A$. Applying induction, we get

$$A_i^k A = A_i^{k+1} = A_i A^k \tag{4}$$

for all positive integers k and all $i = 1, \dots, s$. We can write A_i^D as $A_i^D = p_i(A_i)$, where p_i is a polynomial satisfying $p_i(0) = 0$. Consequently, $A_i^D(A - A_i) = 0$, or equivalently, $A_i^D A = E_i$. The core-nilpotent decomposition yields the existence of an invertible matrix P_i such that

$$A_i = P_i \begin{bmatrix} C_i & 0 \\ 0 & N_i \end{bmatrix} P_i^{-1}, \qquad i = 1, \dots, s,$$

with C_i being invertible and N_i being nilpotent. Hence,

$$A_i^D = P_i \begin{bmatrix} C_i^{-1} & 0 \\ 0 & 0 \end{bmatrix} P_i^{-1} \quad \text{and} \quad E_i = P_i \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} P_i^{-1}, \qquad i = 1, \dots, s.$$

From $E_i = A_i^D A$ and (4) it follows that $E_i E = E_i$. These two relations imply that

$$A = P_i \begin{bmatrix} C_i & 0 \\ S_i & T_i \end{bmatrix} P_i^{-1} \quad \text{and} \quad E = P_i \begin{bmatrix} I & 0 \\ Q_i & R_i \end{bmatrix} P_i^{-1},$$

for some matrices S_i, T_i, Q_i, R_i, which further yields that

$$A^D = P_i \begin{bmatrix} C_i^{-1} & 0 \\ W_i & Z_i \end{bmatrix} P_i^{-1}$$

for some matrices W_i and Z_i. It is now easy to conclude that

$$A_i^D = A_i^n (A^D)^{n+1} \tag{5}$$

for all $i = 1, \dots, s$. Postmultiplying $A_i^2 = A_i A$ by $(A^D)^2$, we get $A_i^2 (A^D)^2 = A_i A^D$. So, we can rewrite (5) as $A_i^D = A_i (A^D)^2$. Summing this relation over i and postmultiplying by A, we get

$$\sum_{i=1}^{s} E_i = E,$$

which by an extension of Cochran's theorem [1] forces $E_iE_j = 0$, or equivalently, $C_{A_i}C_{A_j} = 0$ for $i \neq j$.

We can assume now, without loss of generality, that

$$C_{A_1} = \begin{bmatrix} U_1 & 0 \\ 0 & 0 \end{bmatrix},$$

where U_1 is invertible. It then follows that

$$C_{A_i} = \begin{bmatrix} 0 & 0 \\ 0 & D_i \end{bmatrix}$$

for all $i > 1$. Once again we can assume, without loss of generality, that

$$D_2 = \begin{bmatrix} U_2 & 0 \\ 0 & 0 \end{bmatrix}$$

with U_2 being invertible. Repeating this procedure we arrive at

$$C_{A_i} = \mathrm{diag}(0, \ldots, 0, U_i, 0, \ldots, 0)$$

with U_i invertible, $i = 1, \ldots, s$. From $C_{A_i}^n A = A_i^n A = A_i^{n+1} = C_{A_i}^{n+1}$, we get

$$A = \begin{bmatrix} U_1 & 0 & \ldots & 0 & 0 \\ 0 & U_2 & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & U_s & 0 \\ M_1 & M_2 & \ldots & M_s & N \end{bmatrix}.$$

Now, $C_A = EAE = (\sum_{i=1}^s E_i)A(\sum_{i=1}^s E_i)$, and hence,

$$C_A = \mathrm{diag}(U_1, \ldots, U_s, 0).$$

As $N_AC_A = 0$ we have $M_1 = \cdots = M_s = 0$.

Let us now write A as

$$A = \begin{bmatrix} U_1 & 0 & 0 \\ 0 & D & 0 \\ 0 & 0 & N \end{bmatrix},$$

where $D = \mathrm{diag}(U_2, \ldots, U_s)$. From $A_1 = C_{A_1} + N_{A_1}$ and $C_{A_1}N_{A_1} = N_{A_1}C_{A_1} = 0$, we get

$$A_1 = \begin{bmatrix} U_1 & 0 & 0 \\ 0 & X & Y \\ 0 & Z & W \end{bmatrix}$$

for some matrices X, Y, Z, and W. Here, block matrices A and A_1 are partitioned conformally. Let us next define matrices X_k, Y_k, Z_k, W_k, $k = 1, 2, \ldots$, by

$$A_1^k = \begin{bmatrix} U_1^k & 0 & 0 \\ 0 & X_k & Y_k \\ 0 & Z_k & W_k \end{bmatrix}.$$

It follows from $A_1^{k+1} = A_1^k A$ that

$$\begin{bmatrix} X_{k+1} & Y_{k+1} \\ Z_{k+1} & W_{k+1} \end{bmatrix} = \begin{bmatrix} X_k & Y_k \\ Z_k & W_k \end{bmatrix} \begin{bmatrix} D & 0 \\ 0 & N \end{bmatrix} \tag{6}$$

for every positive integer k. Clearly, $X_n = 0$ and $Z_n = 0$. This yields, together with (6), that $X_{n-1} = 0$ and $Z_{n-1} = 0$. Repeating this procedure, we finally conclude that $X = 0$ and $Z = 0$. Hence, the matrix A_1 has the desired form. In the same way, we consider matrices $A_2, \dots, A_s$. This completes the proof. □

COROLLARY 2.3. *Let $\{A_i\}_{i=1}^{i=s}$ be a left complementary orthogonal family of complex $n \times n$ matrices, and let $A = A_1 + \cdots + A_s$. Then $C_A = C_{A_1} + \cdots + C_{A_s}$, and the family $\{C_{A_i}\}_{i=1}^{i=s}$ is orthogonal.*

Similar to a left complementary orthogonal family of matrices, one can define a right complementary orthogonal family of matrices. We will say that a family of matrices is complementary orthogonal if it is left and right complementary orthogonal. It is well known (and it is easy to prove) that if U is an invertible $k \times k$ matrix and M is a nilpotent $l \times l$ matrix, then the only solution of the matrix equation $UX = XM$ is $X = 0$. Applying this fact, together with Theorem 2.2, one can prove the following statement.

COROLLARY 2.4. *The family $\{A_i\}_{i=1}^{i=s}$ is complementary orthogonal if and only if there exists an invertible matrix P such that*

$$A_i = P \operatorname{diag}(G_i, N_i) P^{-1},$$

where $G_i = \operatorname{diag}(0, \dots, 0, U_i, 0, \dots, 0)$ with U_i invertible, and $\{N_i\}_{i=1}^{i=s}$ is a family of complementary orthogonal nilpotent matrices.

COROLLARY 2.5. *The family $\{A_i\}_{i=1}^{i=s}$ is orthogonal if and only if there exists an invertible matrix P such that*

$$A_i = P \operatorname{diag}(G_i, N_i) P^{-1}, \tag{7}$$

where $G_i = \operatorname{diag}(0, \dots, 0, U_i, 0, \dots, 0)$ with U_i invertible, and $\{N_i\}_{i=1}^{i=s}$ is a family of orthogonal nilpotent matrices.

A short direct proof of Corollary 2.5 can be obtained using the procedure that gives the block diagonal form of matrices $\{C_{A_i}\}_{i=1}^{i=s}$ in the proof of Theorem 2.2. Corollary 2.5 underlines the fact that if $\{A_i\}_{i=1}^{i=s}$ is orthogonal, then so are $\{C_{A_i}\}_{i=1}^{i=s}$ and $\{N_{A_i}\}_{i=1}^{i=s}$. From (7) it is clear that the general "orthogonality problem" reduces to the nilpotent case. Finally, it should be mentioned that the results from [5] dealing with algebraic and geometric multiplicities of eigenvalues of an orthogonal family of matrices follow trivially.

3. Power additivity.

Our first result in this section states that the concepts of power additivity and orthogonality are equivalent in the case of group matrices.

THEOREM 3.1. *For the family of group matrices $\{A_i\}_{i=1}^{i=s}$, the following are equivalent:*

(i) *$\{A_i\}_{i=1}^{i=s}$ is power additive,*
(ii) *$\{A_i\}_{i=1}^{i=s}$ is left complementary orthogonal,*
(iii) *$\{A_i\}_{i=1}^{i=s}$ is complementary orthogonal,*
(iv) *$\{A_i\}_{i=1}^{i=s}$ is orthogonal,*
(v) *$A_i = P \operatorname{diag}(0, \dots, 0, U_i, 0, \dots, 0) P^{-1}$, for some invertible P, with U_i being invertible, in which case A has a group inverse.*

Proof. In [4, Lemma 1] it was proved that (ii) implies (i) and that (i) implies (iv). Clearly, (iv) yields (iii) and (iii) yields (ii). Applying Corollary 2.5, one can easily prove that statements (iv) and (v) are equivalent. This completes the proof. □

In the next result we will show how the characteristic and the minimal polynomial of A are related to characteristic and minimal polynomials of A_i, $i = 1, \dots, s$, in the case that the family $\{A_i\}_{i=1}^{i=s}$ is power additive.

THEOREM 3.2. *Suppose that the family* $\{A_i\}_{i=1}^{i=s}$ *is power additive. Then*

(a) ψ_A *divides the least common multiple of* $\{\psi_{A_i} : i = 1, \dots, s\}$,

(b) $\lambda^{(s-1)n}\Delta_A(\lambda) = \prod_{i=1}^{s} \Delta_{A_i}$.

Proof. If $h = LCM(\psi_{A_i})$, then we can assume, with no loss of generality, that $h(0) = 0$. Otherwise, all matrices A_i would be invertible, which would imply by Theorem 3.1 that $s = 1$, in which case statement (a) is trivial. Power additivity yields

$$f(A) = \sum_{i=1}^{s} f(A_i)$$

for all polynomials $f(\lambda)$ with $f(0) = 0$. In particular, if $f = h$, then each term in the sum vanishes, and consequently, $h(\lambda)$ annihilates A. Applying the fact that ψ_A divides any polynomial that annihilates A we complete the proof of (a). In [4, Proof of Lemma 1] it was shown that the algebraic multiplicities of nonzero eigenvalues of a family of power additive matrices are related by

$$n_A(\lambda) = \sum_{i=1}^{s} n_{A_i}(\lambda), \qquad \lambda \neq 0.$$

From this (b) follows. □

THEOREM 3.3. *The family* $\{A_i\}_{i=1}^{i=s}$ *is power additive if and only if* $C_A = \sum_{i=1}^{s} C_{A_i}$, $N_A = \sum_{i=1}^{s} N_{A_i}$, *and* $\{C_{A_i}\}_{i=1}^{i=s}$ *and* $\{N_{A_i}\}_{i=1}^{i=s}$ *are power additive.*

Even before proving this theorem, we will give an example showing that the power additivity of families $\{C_{A_i}\}_{i=1}^{i=s}$ and $\{N_{A_i}\}_{i=1}^{i=s}$ does not imply the power additivity of $\{A_i\}_{i=1}^{i=s}$. If

$$A_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \quad \text{and} \quad A_2 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

then the families $\{C_{A_1}, C_{A_2}\}$, $\{N_{A_1}, N_{A_2}\}$ are power additive (in fact, even more is true; namely, both families are orthogonal), but $(A_1 + A_2)^2 \neq A_1^2 + A_2^2$. Thus, the assumptions of the core-nilpotent additivity $C_A = \sum_{i=1}^{s} C_{A_i}$ and $N_A = \sum_{i=1}^{s} N_{A_i}$ are essential in the above result.

Proof of Theorem 3.3. The sufficiency is clear as $C_B N_B = N_B C_B = 0$ for any B. So let us suppose that $\{A_i\}_{i=1}^{i=s}$ is power additive and set $C_{A_i} = C_i$ and $N_{A_i} = N_i$. Then in particular,

$$(C_A^n)^k = (A^n)^k = \sum_{i=1}^{s} (A_i^n)^k = \sum_{i=1}^{s} (C_i^n)^k$$

for all positive integers k. Each matrix C_i^n is a group matrix, and so, by Theorem 3.1 they must be orthogonal, which further implies that

$$C_i C_j = 0 \tag{8}$$

for $i \neq j$.

Next, we will show that $X = C_A - \sum_{i=1}^{s} C_i$ vanishes. To do this, we form $C_A^n X = C_A^n(C_A - \sum_{i=1}^{s} C_i) = C_A^{n+1} - (\sum_{i=1}^{s} C_i^n)(\sum_{i=1}^{s} C_i)$ which by (8) collapses to $C_A^{n+1} - \sum_{i=1}^{s} C_i^{n+1} = 0$. Hence, $R(X) \subset N(C_A)$. On the other hand, since rank is additive for the family $\{A_i^n\}_{i=1}^{i=s}$, we may deduce that $R(C_A) = R(A^n) = \oplus_{i=1}^{s} R(A_i^n) = \oplus_{i=1}^{s} R(C_i)$. Consequently, $R(X) \subset R(C_A) + R(\sum_{i=1}^{s} C_i) \subset R(C_A)$, which means that $R(X) \subset R(C_A) \cap N(C_A) = \{0\}$, and thus $X = 0$. Lastly, we may split the power additivity of the family $\{A_i\}_{i=1}^{i=s}$ as $C_A^k + N_A^k = \sum_{i=1}^{s}(C_i^k + N_i^k)$, $k \geq 1$, from which we may conclude that $N_A = \sum_{i=1}^{s} N_i$ and that the family $\{N_i\}_{i=1}^{i=s}$ is power additive, as desired. □

Corollary 3.4. *Let A_1, A_2, and $A = A_1 + A_2$ be $n \times n$ complex matrices. Then $A^k = A_1^k + A_2^k$ for every positive integer k if and only if there exist an invertible $n \times n$ matrix P, invertible matrices E and F, and nilpotent matrices N, N_1, and N_2 with $N^k = N_1^k + N_2^k$, $k = 1, 2, \dots$, such that*

$$A = P \begin{bmatrix} E & 0 & 0 \\ 0 & F & 0 \\ 0 & 0 & N \end{bmatrix} P^{-1}, \quad A_1 = P \begin{bmatrix} E & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & N_1 \end{bmatrix} P^{-1},$$

$$\text{and} \quad A_2 = P \begin{bmatrix} 0 & 0 & 0 \\ 0 & F & 0 \\ 0 & 0 & N_2 \end{bmatrix} P^{-1},$$

where the block matrices are partitioned conformally.

Proof. Applying Theorems 3.1 and 3.3, we can assume, with no loss of generality, that the core parts of A, A_1, and A_2 have the following matrix representations

$$C_A = \begin{bmatrix} E & 0 & 0 \\ 0 & F & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad C_{A_1} = \begin{bmatrix} E & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \text{and} \quad C_{A_2} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & F & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

where E and F are invertible matrices. Applying the fact that the core and nilpotent parts of any matrix are orthogonal matrices, we come to the following matrix representations

$$A = \begin{bmatrix} E & 0 & 0 \\ 0 & F & 0 \\ 0 & 0 & N \end{bmatrix}, \quad A_1 = \begin{bmatrix} E & 0 & 0 \\ 0 & U & V \\ 0 & W & N_1 \end{bmatrix}, \quad \text{and} \quad A_2 = \begin{bmatrix} X & 0 & Y \\ 0 & F & 0 \\ Z & 0 & N_2 \end{bmatrix},$$

but we have $A = A_1 + A_2$, and consequently, U, V, W, X, Y, and Z are zero matrices. This completes the proof. □

Example (3) shows that an analogue of the above statement does not hold for $s > 2$; that is, the power additivity does not guarantee that we may isolate the nilpotent parts from the core parts when $s > 2$.

4. The nilpotent case. Understanding the structure of orthogonal families of nilpotent matrices is, by Corollary 2.5, enough to understand the structure of an arbitrary orthogonal family of matrices. Similarly, Corollary 3.4 shows that the general "power additivity" problem in the case $s = 2$ reduces to the nilpotent case. In this section we will give some partial structural results on the families of orthogonal (power additive) nilpotent matrices. We will also give an example to stress that the general structural problem for power additive families of nilpotents might be very difficult.

THEOREM 4.1. *Let A_1 and A_2 be $n \times n$ complex matrices. Then A_1 and A_2 are orthogonal if and only if there exist an invertible $n \times n$ matrix P and matrices A_{ij}, $i = 1, 2$, $j = 1, \ldots, 4$, such that*

$$A_1 = P \begin{bmatrix} 0 & A_{11} & 0 & A_{12} \\ 0 & A_{13} & 0 & A_{14} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} P^{-1} \quad \text{and} \quad A_2 = P \begin{bmatrix} 0 & 0 & A_{21} & A_{22} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & A_{23} & A_{24} \\ 0 & 0 & 0 & 0 \end{bmatrix} P^{-1}, \tag{9}$$

where the block matrices are partitioned conformally.

Proof. We identify matrices A_1 and A_2 with endomorphisms acting on a finite-dimensional Hilbert space H. Assume that A_1 and A_2 are orthogonal. Let us denote the subspace $R(A_1) \cap R(A_2)$ by H_0. We choose subspaces $H_i \subset H$, $i = 1, 2$ such that $R(A_i) = H_0 \oplus H_i$. Finally, let us choose a subspace H_3 in H such that

$$H = H_0 \oplus H_1 \oplus H_2 \oplus H_3. \tag{10}$$

The restriction of A_i to the subspace $R(A_j) = H_0 \oplus H_j$, $i \neq j$ is a zero operator. It is now easy to see that A_1 and A_2 have the matrix representation (9) with respect to the direct sum decomposition (10). The converse statement is trivial. □

This result with its proof is an approach to the orthogonality problem that is completely different from the approach presented in the second section of this note. The disadvantage of this approach is that it cannot be generalized to the case $s > 2$. However, in case $s = 2$, the results from section 2 did not give any information about orthogonal nilpotent matrices. This special case is now solved by the matrix representation (9). Namely, matrices A_1 and A_2 given by (9) are nilpotents if and only if A_{13} and A_{23} are nilpotents.

It is easy to see that a family of orthogonal nilpotent matrices can be simultaneously triangulated. As the following result shows, even more is true. Once again we identify matrices with linear operators acting on a finite-dimensional Hilbert space.

THEOREM 4.2. *Let $\{A_i\}_{i=1}^{i=s}$ be a family of orthogonal nilpotents. Then there exists a direct sum decomposition*

$$H = H_1 \oplus \cdots \oplus H_s \tag{11}$$

such that $R(A_i) \subset H_1 \oplus \cdots \oplus H_i$, $i = 1, \ldots, s$, $H_1 \oplus \cdots \oplus H_{i-1} \subset N(A_i)$, $i = 2, \ldots, s$, and the restriction of A_i to H_i is injective for all $i = 2, \ldots, s$. So, with respect to the direct sum decomposition (11)*, we have the following matrix representation:*

$$A_1 = \begin{bmatrix} A^1_{11} & A^1_{12} & A^1_{13} & \ldots & A^1_{1s} \\ 0 & 0 & 0 & \ldots & 0 \\ 0 & 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & 0 \end{bmatrix}, \quad A_2 = \begin{bmatrix} 0 & A^2_{12} & A^2_{13} & \ldots & A^2_{1s} \\ 0 & A^2_{22} & A^2_{23} & \ldots & A^2_{2s} \\ 0 & 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & 0 \end{bmatrix}, \ldots,$$

$$A_s = \begin{bmatrix} 0 & 0 & 0 & \ldots & A^s_{1s} \\ 0 & 0 & 0 & \ldots & A^s_{2s} \\ 0 & 0 & 0 & \ldots & A^s_{3s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & A^s_{ss} \end{bmatrix}. \tag{12}$$

Proof. We will prove this statement by induction. In case $s = 2$, we decompose H as $H = N(A_2) \oplus R(A_2^*)$. From $A_2A_1 = 0$, it follows that $R(A_1) \subset N(A_2)$. This proves the assertion in the case $s = 2$. Assume now that the statement holds true for a positive integer $s > 2$, and consider an orthogonal family $\{A_i\}_{i=1}^{i=s+1}$ of nilpotent matrices. According to our induction hypothesis, H can be decomposed as

$$H = H_2 \oplus \cdots \oplus H_{s+1}$$

such that $R(A_i) \subset H_2 \oplus \cdots \oplus H_i$, $i = 2, \ldots, s+1$, $H_2 \oplus \cdots \oplus H_{i-1} \subset N(A_i)$, $i = 3, \ldots, s+1$, and the restriction of A_i to H_i is injective for all $i = 3, \ldots, s+1$. Let us denote the restriction of A_2 to H_2 by B. Then H_2 can be decomposed as $H_2 = N(B) \oplus R(B^*)$. The restrictions of operators $A_2, A_3, \ldots, A_{s+1}$ to subspaces $R(B^*), H_3, \ldots, H_{s+1}$, respectively, are injective. Thus, the orthogonality of the family $\{A_i\}_{i=1}^{i=s+1}$ implies that $R(A_1) \subset N(B)$. This completes the proof. □

In fact, we have proved that a family $\{A_i\}_{i=1}^{i=s}$ of nilpotent matrices has a matrix block representation (12) if and only if $A_iA_j = 0$ whenever $i > j$. It seems natural to try and use this result together with Weyr or Jordan canonical form for the inductive study of the structure of the family of orthogonal nilpotent matrices. This approach thus far has not been successful. However, if we combine orthogonality with rank additivity we can satisfactorily answer the question.

Theorem 4.3. *Let $\{N_i\}_{i=1}^{i=s}$ be a family of $n \times n$ complex matrices. Then the following are equivalent:*

(i) *$\{N_i\}_{i=1}^{i=s}$ is an orthogonal rank additive family of nilpotent matrices.*

(ii) *There exists an invertible matrix P such that*

$$N_1 = P \begin{bmatrix} M_1 & 0 & 0 & \ldots & 0 & B_1 \\ 0 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 0 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 0 & \ldots & 0 & 0 \end{bmatrix} P^{-1},$$

$$N_2 = P \begin{bmatrix} 0 & 0 & 0 & \ldots & 0 & 0 \\ 0 & M_2 & 0 & \ldots & 0 & B_2 \\ 0 & 0 & 0 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 0 & \ldots & 0 & 0 \end{bmatrix} P^{-1},$$

$$\vdots$$

$$N_s = P \begin{bmatrix} 0 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 0 & 0 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & M_s & B_s \\ 0 & 0 & 0 & \ldots & 0 & 0 \end{bmatrix} P^{-1}, \tag{13}$$

with M_i being nilpotent matrices, $[M_i, B_i]$ being full rank matrices satisfying $\operatorname{rank} M_i + \operatorname{rank} B_i = \operatorname{rank}[M_i, B_i]$ *for every $i \in \{1, \dots, s\}$, and*

$$\operatorname{rank} B = \operatorname{rank} \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_s \end{bmatrix} = \operatorname{rank} B_1 + \operatorname{rank} B_2 + \cdots + \operatorname{rank} B_s.$$

(iii) *Matrix* (13) *holds with M_i being nilpotents, $[M_i, B_i]$ being full rank matrices, and*

$$\operatorname{rank} \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_s \end{bmatrix} = \operatorname{rank} B_1 + \operatorname{rank} B_2 + \cdots + \operatorname{rank} B_s.$$

Proof. Assume first that $\{N_i\}_{i=1}^{i=s}$ is an orthogonal rank additive family of nilpotent matrices. Denote $N = N_1 + \cdots + N_s$. We have already mentioned that rank additivity is equivalent to range additivity $R(N) = R(N_1) \oplus \cdots \oplus R(N_s)$. It follows that there exists an invertible matrix Q such that

$$N_1 = Q \begin{bmatrix} M_1 & N_{12} & N_{13} & \dots & N_{1s} & B_1 \\ 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 \end{bmatrix} Q^{-1},$$

$$N_2 = Q \begin{bmatrix} 0 & 0 & 0 & \dots & 0 & 0 \\ N_{21} & M_2 & N_{23} & \dots & N_{2s} & B_2 \\ 0 & 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 \end{bmatrix} Q^{-1},$$

$$\vdots$$

$$N_s = Q \begin{bmatrix} 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ N_{s1} & N_{s2} & N_{s3} & \dots & M_s & B_s \\ 0 & 0 & 0 & \dots & 0 & 0 \end{bmatrix} Q^{-1},$$

with $[M_1, N_{12}, \dots, N_{1s}, B_1], \dots, [N_{s1}, N_{s2}, \dots, M_s, B_s]$ being full rank matrices. Applying orthogonality, together with the fact that the ith row of N_i is a full rank matrix,

$1 \leq i \leq s$, we get that $N_{ij} = 0$ for all $i \neq j$, $1 \leq i, j \leq s$. Multiplying the N_i's by

$$Q \begin{bmatrix} I & 0 & 0 & \dots & 0 & X_1 \\ 0 & I & 0 & \dots & 0 & X_2 \\ 0 & 0 & I & \dots & 0 & X_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & I & X_s \\ 0 & 0 & 0 & \dots & 0 & I \end{bmatrix} Q^{-1}$$

from the left, and by its inverse

$$Q \begin{bmatrix} I & 0 & 0 & \dots & 0 & -X_1 \\ 0 & I & 0 & \dots & 0 & -X_2 \\ 0 & 0 & I & \dots & 0 & -X_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & I & -X_s \\ 0 & 0 & 0 & \dots & 0 & I \end{bmatrix} Q^{-1}$$

from the right, we can assume after choosing appropriate blocks $X_1, \dots, X_s$, say $X_i = M_i^- B_i$, that $\operatorname{rank}[M_i, B_i] = \operatorname{rank} M_i + \operatorname{rank} B_i$ for every $i \in \{1, \dots, s\}$. Applying $R(N) = R(N_1) \oplus \dots \oplus R(N_s)$, one can now easily prove that

$$\operatorname{rank} \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_s \end{bmatrix} = \operatorname{rank} B_1 + \operatorname{rank} B_2 + \dots + \operatorname{rank} B_s.$$

This completes the proof that (i) implies (ii). Obviously, (ii) yields (iii). Finally, if (iii) is satisfied, then it follows at once that $\{N_i\}_{i=1}^{i=s}$ is an orthogonal family of nilpotent matrices. We next observe that the ranges of N_i, $i = 1, \dots, s$ form a direct sum. Clearly, $R(N) \subset \oplus_{i=1}^{s} R(N_i)$. For the converse inclusion, it suffices to show that $R(N_1) \subset R(N)$, but

$$R(N_1) \subset R\left(\begin{bmatrix} M_1 \\ 0 \end{bmatrix}\right) + R\left(\begin{bmatrix} B_1 \\ 0 \end{bmatrix}\right).$$

Obviously,

$$R\left(\begin{bmatrix} M_1 \\ 0 \end{bmatrix}\right) \subset R(N),$$

but because of rank additivity, we have

$$R\left(\begin{bmatrix} B_1 \\ 0 \end{bmatrix}\right) \subset R(B),$$

which trivially is contained in $R(N)$. This completes the proof. □

As far as power additivity for a nilpotent family $\{A_i\}_{i=1}^{i=s}$ is concerned, very little can be said in general. All we know is that $A = \sum_{i=1}^{s} A_i$ is nilpotent and that $\operatorname{index}(A) \leq \max_{i=1,\dots,s} \operatorname{index}(A_i)$. To see that the "power-additivity" problem for a family of nilpotents is difficult, one can consider matrices

$$\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}, \quad \text{and} \quad \begin{bmatrix} \frac{1}{2} + \frac{\sqrt{3}}{2}i & 1 \\ \frac{1}{2} - \frac{\sqrt{3}}{2}i & -\frac{1}{2} - \frac{\sqrt{3}}{2}i \end{bmatrix},$$

which are power-additive but are not simultaneously triangularizable. The same example shows that the power additivity of the complete set does not imply the power additivity of a subset.

In the case $s = 2$, we can say a little more. Indeed, $A_1A_2 = -A_2A_1$, and hence $p(A_1)A_2 = A_2p(-A_1)$ for every polynomial p. Moreover, $A_1A_2^k = -A_2A_1^k$ for all $k = 1, 2, \ldots$, and so,

$$A_1p(A_2) = -A_2p(A_1) \tag{14}$$

for every polynomial p with $p(0) = 0$. Applying Weyr form it is easy to see that A_1 and A_2 may be simultaneously triangulated. The general solution to $BX = -XB$ is obtained by finding a particular invertible solution to $BX = -XB$ and the general solution to $BY = YB$. If A_1 is a single Jordan block $A_1 = J$, then we know that $D = \mathrm{diag}(1, -1, 1, -1, \ldots,)$ is a particular solution to $JX = -XJ$, while at the same time $JX = XJ$ has a general solution of the form $T = p(J)$, where $p(\lambda) = a_0 + a_1\lambda + \cdots + a_m\lambda^m$ is any polynomial in λ; that is, T is Toeplitz. It follows that the general solution to $JY = -YJ$ has the form $Y = DT$. Needless to say, for this to be nilpotent we need that $a_0 = 0$. Power additivity now reduces to

$$[J + Dp(J)]^k = J^k + [Dp(J)]^k.$$

These equations can be further simplified using $Dp(J) = p(-J)D$ which yields $[Dp(J)]^{2m} = p(J)^m p(-J)^m$ and $[Dp(J)]^{2m+1} = D[p(J)]^{m+1}[p(-J)]^m$. Now, by (14) we have $Jf(Dp(J)) = -Dp(J)f(J)$ for all polynomials $f(\lambda)$ with $f(0) = 0$. For $n = 2, 3$, power additivity holds automatically for all A_2 of the form $Dp(J)$, while for $n = 4$ this is the case precisely when $a_1 = 0$ or $a_1 = 1$. For $n > 4$, the necessary and sufficient condition is that $a_1 = a_2 = \cdots = a_{n-3} = 0$.

When we have more than a single Jordan block, say, $A_1 = \mathrm{diag}(J_1, \ldots, J_m)$, then again $D = \mathrm{diag}(D_1, \ldots, D_m)$ solves $A_1D = -DA_1$. As such, to find the general solution we have to multiply D by the commutator $\{X : A_1X = XA_1\}$ which is well known. The easiest way to describe the commutator is via the Weyr form which has the advantage that the solutions are block upper triangular. The disadvantage of this approach is that there is no particular invertible solution of the equation $A_1X = -XA_1$ which is as simple as the matrix D. Using these methods and the inductive approach, one can solve the "power-additivity" problem for two nilpotents in the cases when n is small. This approach thus far has not lead to the general solution.

REFERENCES

[1] T. W. Anderson and G. P. H. Styan, *Cochran's theorem, rank additivity and tripotent matrices*, in Essays in Honor of C. R. Rao, G. Kallianpur, P. R. Krishnaiah, and J. K. Ghosh, eds., North–Holland, Amsterdam, 1982, pp. 1–23.

[2] A. Ben Israel and T. N. E. Greville, *Generalized Inverses, Theory and Applications*, John Wiley, New York, 1974.

[3] R. E. Hartwig, *More on the Souriau-Frame algorithm*, SIAM J. Appl. Math., 31 (1976), pp. 42–46.

[4] P. Šemrl, *On a matrix version of Cochran's statistical theorem*, Linear Algebra Appl., 237/238 (1996), pp. 477–487.

[5] G. P. H. Styan and A. Takemura, *Rank additivity and matrix polynomials*, in Studies in Econometrics, Time Series and Multivariate Statistics, S. Karlin, T. Amemiya, and L. A. Goodman, eds., Academic Press, New York, 1983, pp. 545–558.

SIAM J. MATRIX ANAL. APPL.
Vol. 20, No. 1, pp. 14–30

STRUCTURED TOTAL LEAST NORM FOR NONLINEAR PROBLEMS[*]

J. B. ROSEN[†], HAESUN PARK[‡], AND JOHN GLICK[§]

Abstract. An extension of the recently developed structured total least norm (STLN) problem formulation is described for solving a class of nonlinear parameter estimation problems. STLN is a problem formulation for obtaining an approximate solution to the overdetermined linear system $Ax \approx b$ preserving the given affine structure in A or $[A \mid b]$, where errors can occur in both the vector b and the matrix A. The approximate solution can be obtained to minimize the error in the L_p norm, where $p = 1, 2,$ or ∞. In the extension of STLN to nonlinear problems, the elements of A may be differentiable nonlinear functions of a parameter vector, whose value needs to be approximated. We call this extension structured nonlinear total least norm (SNTLN). The SNTLN problem is formulated and its solution by a modified STLN algorithm is described. Optimality conditions and convergence for the 2-norm case are presented.

Computational tests were carried out on an overdetermined system with Vandermonde structure and on two nonlinear parameter estimation problems. In these problems, both the coefficients and the unknown parameters were to be determined. The computational results demonstrate that the SNTLN algorithm recovers good approximations to the correct values of both the coefficients and parameters, in the presence of noise in the data and poor initial estimates of the parameters. It is also shown that the SNTLN algorithm with the 1-norm minimization is robust with respect to outliers in the data.

Key words. overdetermined linear systems, data fitting, least squares, linear prediction, parameter estimation, frequency estimation, outliers, total least norm, total least squares, Vandermonde matrix, 1-norm, 2-norm

AMS subject classifications. 65F20, 65F30, 65M10, 65Y20

PII. S0895479896301662

1. Introduction.

A new algorithm called structured total least norm (STLN) has recently been developed [24] for obtaining an approximate solution to the overdetermined linear system

$$Ax \approx b,$$

where errors may occur in both the vector b and in elements of the affinely structured $(m \times n)$ matrix A, where $m > n$. The STLN algorithm preserves the structure of A or $[A \mid b]$ and minimizes, in a suitable norm, the residual and the change in the elements of A. This minimization can be done using the L_p norm, where $p = 1, 2,$ or ∞.

[*] Received by the editors April 8, 1996; accepted for publication (in revised form) by S. Van Huffel June 20, 1997; published electronically September 15, 1998. The work of the first and third authors was supported in part by Air Force Office of Scientific Research grant AFOSR-91-0147 and the Minnesota Supercomputer Institute. The work of the first author was supported in part by ARPA/NIST grant 60NANB4D1615 and National Science Foundation grant CCR-9509085. The work of the second author was supported in part by National Science Foundation grants CCR-9209726 and CCR-9509085.

http://www.siam.org/journals/simax/20-1/30166.html

[†] Computer Science Department, University of Minnesota, Minneapolis, MN 55455, and Department of Computer Science and Engineering, University of California, San Diego, La Jolla, CA 92093 (rosen@cs.umn.edu, jbrosen@cs.ucsd.edu).

[‡] Computer Science Department, University of Minnesota, Minneapolis, MN 55455 (hpark@cs.umn.edu).

[§] Department of Mathematics and Computer Science, University of San Diego, San Diego, CA 92110 (glick@pwa.acusd.edu).

The STLN algorithm addresses the same class of problems as total least squares [30], but has the advantage that it preserves the structure of A or $[A \mid b]$ when they are perturbed in the solution. The theory, implementation, and some applications of the STLN algorithm are presented in several recent papers [24, 25, 21, 28, 20, 7].

In this paper, we show that the STLN algorithm can be extended to solve a related, but more difficult, structured approximation problem, in which the elements of A may be *nonlinear* differentiable functions of a parameter vector α. We call this extension structured nonlinear total least norm (SNTLN). The SNTLN algorithm will be described, and the optimality conditions and convergence properties of the algorithm are also given for the L_2 norm. Its relationship to the Gauss–Newton method is shown. The details of the SNTLN algorithm for the Vandermonde structured problems are shown. Some computational tests of the algorithm, using the L_1 and L_2 norms are summarized. These tests were carried out on overdetermined systems for which an exact solution with zero residual is known. That is, an exact parameter α_c, the right-hand side vector b_c, and the exact solution x_c are known so that

$$A(\alpha_c)x_c = b_c.$$

In typical applications, there is error in the vector b, and the parameter vector α_c is not known but is to be estimated. In the SNTLN algorithm, an initial estimate $\hat{\alpha}$ is needed to start the iterative solution. In some applications, an initial value $\hat{\alpha}$ is either given or can be estimated. In this case, the SNTLN algorithm is like the STLN algorithm in the sense that it solves the given overdetermined system preserving the given problem structure. The difference is that the SNTLN preserves nonlinear structures. In fact, when the SNTLN algorithm is applied to affinely structured problems, it becomes the STLN algorithm.

The computational results show that for both L_1 and L_2, the algorithm converged from a range of initial values $\hat{\alpha}$, to α_c for $b = b_c$, and to a value close to α_c for $\|b - b_c\|$ small. Furthermore, the robust behavior with respect to outliers in the vector b was demonstrated by the SNTLN algorithm using the L_1 norm. Such robust behavior has previously been observed by Barrodale and others [23, 4, 5, 17] for the simpler problem of finding an x such that the residual norm $\|b - Ax\|_1$ is minimized. The use of the L_1 norm for some nonlinear problems has also been investigated in [27, 31].

Before describing SNTLN we briefly summarize the earlier STLN formulation and algorithm. The formulation takes full advantage of the special structure of the given matrix A. In particular, when there are $q \leq mn$ elements of A which are subject to error, a $q \times 1$ vector α is used to represent the corresponding elements of the error matrix E, which gives

$$(A + E)x = b - r.$$

Note that for a sparse matrix, $q \ll mn$. Furthermore, if many elements of E must have the same value, then q is the number of *different* such elements. For example, in a Toeplitz or Hankel matrix, each diagonal or antidiagonal consists of elements with the same value, respectively, so $q \leq m + n - 1$.

The matrix E is specified by those elements of A which may be subject to error. Each different nonzero element of E corresponds to one of the α_k, $k = 1, \ldots, q$. Now, the residual vector $r = b - (A + E)x$ is a function of (α, x). Let $D \in \mathbf{R}^{q \times q}$ be a diagonal matrix that accounts for the repetition of elements of α in the matrix E. Then the STLN problem can be stated as follows.

```
Algorithm STLN.
Input – A Structured Total Least Norm problem (1.1), with matrices $A$, $D$, vector $b$, and tolerance $tol$
Output – Error matrix $E$, residual vector $r$, and vector $x$
    1. Set $\alpha = 0$, $E = 0$, compute $x$ from (1.4), construct $X$ from $x$, and set $r = b - Ax$.
    2. repeat
        (a) $\underset{\Delta x,\Delta\alpha}{\text{minimize}} \left\| \begin{pmatrix} X & A+E \\ D & 0 \end{pmatrix} \begin{pmatrix} \Delta\alpha \\ \Delta x \end{pmatrix} + \begin{pmatrix} -r \\ D\alpha \end{pmatrix} \right\|_p$.
        (b) Set $x := x + \Delta x$, $\alpha := \alpha + \Delta\alpha$.
        (c) Construct $X$ from $x$, and $E$ from $\alpha$. Compute $r = b - (A+E)x$.
       until ($\|\Delta x\| \le tol$ and $\|\Delta\alpha\| \le tol$)
```

$$\min_{\alpha,x} \left\| \begin{matrix} r(\alpha,x) \\ D\alpha \end{matrix} \right\|_p , \tag{1.1}$$

where $\|\cdot\|_p$ is the vector p-norm, for $p = 1, 2,$ or ∞.

In the iterative algorithm for solving the STLN problem, the vector Ex is represented in terms of α. This is accomplished by defining an $m \times q$ matrix X such that $X\alpha = Ex$. The matrix X consists of the elements of x, with suitable repetition, giving X a special structure. If E is Toeplitz and every diagonal in A is subject to error, then X can be arranged to be Toeplitz too. In fact, E and X have exactly the same number of nonzero elements.

In the minimization (1.1), a linear approximation to $r(\alpha, x)$ is used. Let Δx represent a small change in x, and ΔE a small change in the variable elements of E. Then we have $X(\Delta\alpha) = (\Delta E)x$, where $\Delta\alpha$ represents the corresponding small change in the elements of α. Neglecting the second-order terms in $\|\Delta\alpha\|$ and $\|\Delta x\|$,

$$r(\alpha + \Delta\alpha, x + \Delta x) = r(\alpha, x) - X\Delta\alpha - (A+E)\Delta x. \tag{1.2}$$

The linearization of (1.1) now becomes

$$\min_{\Delta\alpha,\Delta x} \left\| \begin{pmatrix} X & A+E \\ D & 0 \end{pmatrix} \begin{pmatrix} \Delta\alpha \\ \Delta x \end{pmatrix} + \begin{pmatrix} -r \\ D\alpha \end{pmatrix} \right\|_p . \tag{1.3}$$

To start the iterative algorithm, the initial values of $E = 0$ and the least norm value of $x = x_{ln}$ can be used, where x_{ln} is given by

$$\min_x \|b - Ax\|_p. \tag{1.4}$$

The STLN algorithm is summarized in Algorithm STLN.

2. SNTLN. We now show how the nonlinear extension is formulated, and how Algorithm STLN can be modified to solve this nonlinear problem. We describe the extension to the case in which the matrix A is a differentiable function $A(\alpha)$ of an $s \times 1$ parameter vector α and an approximate solution to the overdetermined system

$$A(\alpha)x \approx b \tag{2.1}$$

is to be obtained. Any number of the elements of A may depend on α, but important properties of the solution will depend on m, n, and s. This is discussed in section 3. The residual vector $r = r(\alpha, x)$ is now defined by

$$r(\alpha, x) = b - A(\alpha)x \tag{2.2}$$

and the parametric problem can be stated as

$$\min_{\alpha,x} \left\| \begin{array}{c} r(\alpha, x) \\ D(\alpha - \hat{\alpha}) \end{array} \right\|_p, \tag{2.3}$$

where $\hat{\alpha}$ is an initial estimate of the optimum parameter vector and D is a diagonal matrix of positive weights. For the assumptions on rank of A and Hessians of the elements of A, see section 3.

The nonlinear parameter estimation problem of minimizing the L_2-norm of $r(\alpha, x)$, where the problem is linear in x but nonlinear in α, has been investigated in many earlier papers [10, 12, 18, 6, 16]. It is often designated as "separable nonlinear least squares" [6, 10, 12, 18, 13]. The method of solution presented here, as an extension of the STLN algorithm and called the SNTLN algorithm, is different from those presented earlier.

The problem (2.3) clearly reduces to those considered earlier when $p = 2$ and $D = 0$. More importantly, the solution method given here, the SNTLN algorithm, applies directly to the problem (2.3) with $p = 1$ or ∞, in addition to $p = 2$. As shown in section 5, the use of $p = 1$ has important practical applications because of its robustness with respect to outliers in the data.

The most closely related of these earlier methods is variable projection [10, 12]. However, variable projection is only valid for $s \leq m - n$, whereas SNTLN, with positive D, is not similarly restricted. Furthermore, the SNTLN algorithm is valid for all three norms, $p = 1, 2, \infty$, while the variable projection method is limited to the L_2 norm. Starting with the same initial estimate $\hat{\alpha}$, the sequence of vectors $\{\alpha_k, x_k\}$ computed by SNTLN, with $p = 2$, will differ from those computed by the variable projection method. Specifically, variable projection x_k will always be the least squares solution of $A(\alpha_k)x \approx b$; this may not be true for SNTLN, except at termination. A complete statement on the increments $(\Delta\alpha, \Delta x)$ used in SNTLN is given in section 3.

The more general nonlinear model

$$\min_y \|f(y)\|_p,$$

$$f : \mathbf{C}^n \to \mathbf{C}^m, \quad m > n,$$

can also be solved by the SNTLN algorithm. With appropriate assumptions on $f(y)$, this is shown in [26]. In this paper, we limit consideration to the separable case

$$y = \begin{pmatrix} \alpha \\ x \end{pmatrix}, \qquad f = \begin{pmatrix} b - A(\alpha)x \\ D(\alpha - \hat{\alpha}) \end{pmatrix}.$$

Just as for the STLN algorithm, we compute the minimum solution to (2.3) iteratively by linearizing $r(\alpha, x)$:

$$r(\alpha + \Delta\alpha, x + \Delta x) = r(\alpha, x) - A(\alpha)\Delta x - J(\alpha, x)\Delta\alpha, \tag{2.4}$$

```
ALGORITHM SNTLN.
Input – Matrices $A(\alpha)$, $\nabla_\alpha a_i$, $1 \le i \le n$, $D$, vector $b$, initial estimate $\hat{\alpha}$, and tolerance $tol$.
Output – $\alpha$, residual vector $r$, and vector $x$
    1. Set $\alpha = \hat{\alpha}$, compute $x$ from (1.4) with $A = A(\alpha)$, $J(\alpha, x)$, and set $r = b - A(\alpha)x$.
    2. repeat
        (a) $\underset{\Delta x, \Delta\alpha}{\text{minimize}} \left\| \begin{pmatrix} A(\alpha) & J(\alpha,x) \\ 0 & D \end{pmatrix} \begin{pmatrix} \Delta x \\ \Delta\alpha \end{pmatrix} + \begin{pmatrix} -r \\ D(\alpha - \hat{\alpha}) \end{pmatrix} \right\|_p$.
        (b) Set $x := x + \Delta x$, $\alpha := \alpha + \Delta\alpha$.
        (c) Compute $J(\alpha, x)$, $A(\alpha)$, $r = b - A(\alpha)x$.
    until ($\|\Delta x\| \le tol$ and $\|\Delta\alpha\| \le tol$)
```

where $J(\alpha, x)$ is the Jacobian, with respect to α, of $A(\alpha)x$. Let $a_j(\alpha)$ represent the jth column of $A(\alpha)$. Then

$$J(\alpha, x) = \nabla_\alpha (A(\alpha)x) = \sum_{j=1}^{n} x_j \nabla_\alpha a_j(\alpha). \tag{2.5}$$

The algorithm for the nonlinear extension is identical to the STLN algorithm as given above, except that now the matrix X is replaced by $J(\alpha, x)$, and $A + E$ is replaced by $A(\alpha)$, wherever it is appropriate. The modification of STLN for nonlinear parameter estimation is given by Algorithm SNTLN. For the formulation of step 2(a) of Algorithm SNTLN for $p = 1$ and $p = \infty$ as a linear program, see [24, 26].

It should be noted that STLN is a special case of SNTLN, and Algorithm SNTLN becomes Algorithm STLN for affinely structured problems. Specifically, for affinely structured problems, in Algorithm STLN, we have

$$A(\alpha) = A + E(\alpha),$$

where $\hat{\alpha} = 0$ and $E(0) = 0$ according to step 1, which gives

$$A(\hat{\alpha}) = A(0) = A + E(0) = A.$$

Therefore,

$$A(\alpha)x = Ax + E(\alpha)x = Ax + X\alpha$$

and

$$J(\alpha, x) = \nabla_\alpha [A(\alpha)x] = X.$$

Note that using a small D in Step 2(a) of Algorithm STLN or Algorithm SNTLN makes only a small change in the condition of the problem. This is due to the fact that

$$\text{cond}\left(\begin{pmatrix} A & J \\ 0 & D \end{pmatrix} \right) \approx \text{cond}((A \quad J))$$

when D is small. Also, the stable algorithms for computing the QR decomposition, such as those based on Givens and Householder transformations, are not sensitive to having small rows $(0 \quad D)$ placed at the bottom of the matrix. For details, see [2].

Important applications of SNTLN include problems of estimating parameters. In some parameter estimation applications, it is assumed that the system, with no noise, can be represented by

$$y(t) = \sum_{j=1}^{n} x_j f_j(\alpha, t),$$

where the $f_j(\alpha, t)$ are specified functions of α and t. The functional dependence of the f_j on α is known, and it is desired to estimate the "best" values of α and x. Measurements of the system at $m > n + s$ points t_i, $i = 1, \ldots, m$, are taken, giving the $m \times 1$ vector b that represents y. Note that with the SNTLN algorithm presented here there is no requirement that the t_i be uniformly spaced.

This problem can immediately be put in the desired form by defining the elements of $A(\alpha)$ as

$$A(\alpha) = (a_{ij}) = f_j(\alpha, t_i),$$

which gives

$$\begin{pmatrix} f_1(\alpha,t_1) & f_2(\alpha,t_1) & \cdots & f_n(\alpha,t_1) \\ f_1(\alpha,t_2) & f_2(\alpha,t_2) & \cdots & f_n(\alpha,t_2) \\ \vdots & & & \vdots \\ f_1(\alpha,t_m) & f_2(\alpha,t_m) & \cdots & f_n(\alpha,t_m) \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \approx \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}. \tag{2.6}$$

This type of problem arises in a variety of signal processing applications, such as frequency and exponential decay estimation [14, 22, 1]. In another potential application, the matrix $A(\alpha)$ is large and sparse, with only a few elements depending on α. An example of this kind of problem occurs in fitting scattered data in three-dimensional space [11].

Another type of application involves solving an overdetermined system

$$A(\hat{\alpha})x \approx b$$

for x, where A is a nonlinear function of a vector α and the matrix $A(\hat{\alpha})$ is given. In this case, the SNTLN can be used to solve the system while preserving the given *nonlinear* structure in $A(\alpha)$. The only difference is that here the initial value $\hat{\alpha}$, for α, is given.

3. Optimality conditions and convergence.

In this section we generalize the STLN convergence results presented in [24] to the SNTLN algorithm. These results hold for the SNTLN algorithm using the L_2 norm, where the function being minimized is differentiable.

Some additional assumptions on $A(\alpha)$ are needed in order to obtain the convergence results. We assume that with no noise (or error) in $b = b_c$, there is a correct parameter vector α_c, and corresponding x_c, such that $A(\alpha_c)x_c = b_c$. A neighborhood Ω_c in the s-dimensional space that contains the initial estimate $\hat{\alpha}$ is assumed, with α_c at its center. Also, it is assumed that $A(\alpha)$ has full rank and that all the elements $a_{ij}(\alpha)$ have bounded Hessians for all $\alpha \in \Omega_c$.

The function (2.3) being minimized, when $p = 2$, is equivalent to

$$\varphi(\alpha, x) = \frac{1}{2} r^T r + \frac{1}{2}(\alpha - \hat{\alpha})^T D^2 (\alpha - \hat{\alpha}), \tag{3.1}$$

where $r = r(\alpha, x) = b - A(\alpha)x$. The first-order optimality conditions for a local optimum of $\varphi(\alpha, x)$ are the vanishing of the gradients $\nabla_\alpha\varphi$ and $\nabla_x\varphi$, where these gradients are given by

$$\begin{aligned}\nabla_\alpha\varphi &= -J^T(\alpha, x)r + D^2(\alpha - \hat{\alpha}),\\ \nabla_x\varphi &= -A(\alpha)^T r.\end{aligned} \tag{3.2}$$

At each iteration of the SNTLN algorithm with the L_2-norm, we compute the least squares solution to

$$\min_{\Delta x, \Delta\alpha} \left\| M \begin{pmatrix} \Delta\alpha \\ \Delta x \end{pmatrix} + \begin{pmatrix} -r \\ D(\alpha - \hat{\alpha}) \end{pmatrix} \right\|_2, \tag{3.3}$$

where

$$M = \begin{bmatrix} J(\alpha, x) & A(\alpha) \\ D & 0 \end{bmatrix}. \tag{3.4}$$

The least squares solution is given by

$$M^T M \begin{pmatrix} \Delta\alpha \\ \Delta x \end{pmatrix} = -M^T \begin{pmatrix} -r \\ D(\alpha - \hat{\alpha}) \end{pmatrix} = -\begin{pmatrix} \nabla_\alpha\varphi \\ \nabla_x\varphi \end{pmatrix}, \tag{3.5}$$

where the last equality follows from (3.2). Since M has full rank, by the assumption on $A(\alpha)$, $M^T M$ is nonsingular and $(\Delta\alpha, \Delta x)$ can only be zero when the gradient is zero. Therefore at termination, the algorithm gives (α, x), satisfying the first-order optimality conditions.

Now consider the Hessian of $\varphi(\alpha, x)$ with respect to (α, x). Then,

$$\begin{aligned} H(\alpha, x) &= \left(\nabla_{\alpha,x}(\nabla_\alpha\varphi(\alpha, x)) \quad \nabla_{\alpha,x}(\nabla_x\varphi(\alpha, x)) \right) && (3.6)\\ &= \left(\nabla_{\alpha,x}(-J^T(\alpha, x)r + D^2(\alpha - \hat{\alpha})) \quad \nabla_{\alpha,x}(-A(\alpha)^T r) \right). && (3.7)\end{aligned}$$

Since

(3.8)
$$\nabla_{\alpha,x}(-J^T(\alpha, x)r + D^2(\alpha - \hat{\alpha})) = \begin{pmatrix} J^T(\alpha, x)J(\alpha, x) - \sum_{i=1}^m r_i Q_i(\alpha, x) \\ A^T(\alpha)J(\alpha, x) - \sum_{i=1}^m r_i K_i^T(\alpha) \end{pmatrix} + \begin{pmatrix} D^2 \\ 0 \end{pmatrix}$$

and

$$\nabla_{\alpha,x}(-A(\alpha)^T r) = \begin{pmatrix} J^T(\alpha, x) \\ A^T(\alpha) \end{pmatrix} A(\alpha) + \begin{pmatrix} -\sum_{i=1}^m r_i K_i(\alpha) \\ 0 \end{pmatrix}, \tag{3.9}$$

we have

$$H(\alpha, x) = M^T M - \begin{bmatrix} H_1 & H_2 \\ H_2^T & 0 \end{bmatrix},$$

where

$$\begin{aligned} H_1 &= \textstyle\sum_{i=1}^m r_i Q_i(\alpha, x),\\ H_2 &= \textstyle\sum_{i=1}^m r_i K_i(\alpha),\end{aligned}$$

and

$$\begin{aligned} K_i^T(\alpha) &= \text{Jacobian of } i\text{th column of } A^T(\alpha),\\ Q_i(\alpha, x) &= \textstyle\sum_{j=1}^n x_j \nabla_\alpha^2 a_{ij}(\alpha).\end{aligned}$$

Since the Hessian of $a_{ij}(\alpha)$ are all assumed bounded, $\|H_1\|$ and $\|H_2\|$ are of $O(\|r\|)$. Therefore, provided the residual vector r is small, the positive definite matrix M^TM is a good approximation to $H(\alpha, x)$. The iteration (3.5) is therefore a good approximation to Newton's method, and in fact is equivalent to the Gauss–Newton method (see, for example, section 6.1 in [9]). Based on the known convergence properties of the Gauss–Newton method, we can say that the SNTLN algorithm with $p = 2$ will converge to α_c, provided $\|\hat{\alpha} - \alpha_c\|$ and the residual norm $\|r\|$ are both sufficiently small. Convergence of the SNTLN algorithm for $p = 1$ or ∞ is shown elsewhere [26]. To prove convergence, it is necessary to add a line search to the algorithm so that a strict decrease in the norm (2.3) can be shown at each iteration. The algorithm will then converge to a stationary point of (2.3). The computational results obtained so far show, however, that a line search is not required in order to obtain convergence of the algorithm to α_c, provided that the initial parameter estimate $\hat{\alpha}$ is reasonably close to α_c.

We now give explicit expressions for the increments $(\Delta\alpha, \Delta x)$ as computed at each iteration of the SNTLN algorithm. From (3.5) and (3.2) we have

$$\begin{aligned}(J^TJ + D^2)\Delta\alpha + J^TA\Delta x &= J^Tr - D^2(\alpha - \hat{\alpha}),\\ A^TJ\Delta\alpha + A^TA\Delta x &= A^Tr.\end{aligned} \tag{3.10}$$

Since $A(\alpha)$ has full rank, we can solve for $\Delta\alpha$ and Δx. These increments are given by

$$(J^TPJ + D^2)\Delta\alpha = J^TPr - D^2(\alpha - \hat{\alpha}), \tag{3.11}$$

$$\Delta x = (A^TA)^{-1}A^T(r - J\Delta\alpha), \tag{3.12}$$

where

$$P = I - A(A^TA)^{-1}A^T$$

is the projection onto the orthogonal complement of the range space of A. Note that Δx is given by the least squares solution to the linearized approximation to $r(\alpha + \Delta\alpha, x + \Delta x)$, as given by (2.4). That is

$$\min_{\Delta x} \|r - J\Delta\alpha - A\Delta x\|_2.$$

Also note that the first-order optimality conditions $\nabla_\alpha\varphi = \nabla_x\varphi = 0$ are satisfied when the increments $(\Delta\alpha, \Delta x)$ given by (3.11) and (3.12) are zero. This is seen from (3.2) and the fact that $A^Tr = 0$ implies $Pr = r$.

The need to choose positive diagonal elements for the diagonal matrix D, and its relationship to m, n, and s, will now be summarized. Since $\text{rank}(A) = n$, the $m \times m$ projection matrix P has rank $m - n$. Therefore, $\text{rank}(PJ) \le m - n$. Recall that $\Delta\alpha \in \mathbf{C}^s$, so if $s > \text{rank}(PJ)$ and $D = 0$, the value of $\Delta\alpha$ is not uniquely determined by (3.11). However, since J^TPJ is always positive semidefinite, the matrix $J^TPJ + D^2$ is always positive definite for any diagonal matrix D with positive diagonal elements, even when the diagonal elements of D are very small. Therefore, (3.11) will always give a unique $\Delta\alpha$ for $D > 0$. If J has full rank $(= s)$ and $s \le m - n$, then J^TPJ is positive definite, and $\Delta\alpha$ is uniquely determined even with $D = 0$. To illustrate the lack of uniqueness in α, when $s > m - n$ and $D = 0$, we give a simple example, where

$m = 3$, $n = 2$, and $s = 2$. Let $r(\alpha, x)$ be given by (2.2) and consider the minimization problem (2.3) with $D = 0$ and $p = 2$. Also, let

$$A(\alpha) = \begin{pmatrix} a_1(\alpha) \\ B \end{pmatrix}, \qquad b = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix},$$

where $a_1(\alpha) = \begin{pmatrix} a_{11} + \alpha_1 & a_{12} + \alpha_2 \end{pmatrix} \in \mathbf{R}^{1\times 2}$, $B \in \mathbf{R}^{2\times 2}$ is a nonsingular, constant matrix, and $\left(\begin{smallmatrix} b_2 \\ b_3 \end{smallmatrix}\right) \neq 0 \in \mathbf{R}^{2\times 1}$. Choosing $x^* = B^{-1}\left(\begin{smallmatrix} b_2 \\ b_3 \end{smallmatrix}\right)$ will make $r_2 = r_3 = 0$ and also give

$$r_1 = (b_1 - a_{11}x_1^* - a_{12}x_2^*) - (\alpha_1 x_1^* + \alpha_2 x_2^*).$$

Therefore, any values of α_1 and α_2 that give $r_1 = 0$ will produce a minimum solution to (2.3), with value zero.

The effect of the choice of D on the minimum solution to (2.3) is best understood by the requirement that $\nabla_\alpha \varphi = 0$, with $\nabla_\alpha \varphi$ given by (3.2). To simplify the discussion, let $D = \mu I$, with $\mu > 0$. First we observe that as μ is increased from zero, the term $D(\alpha - \hat{\alpha})$ increasingly dominates the minimization, so the norm $\|\bar{\alpha} - \hat{\alpha}\|$ will go to zero, where $\bar{\alpha}$ is the value of α obtained by the SNTLN algorithm. Therefore, relative large values of μ should be used only when the initial estimate $\hat{\alpha}$ is known to be reliable. The optimality conditions (3.2) require that

$$J^T(\bar{\alpha}, x)r = D^2(\bar{\alpha} - \hat{\alpha}) = \mu^2(\bar{\alpha} - \hat{\alpha}), \tag{3.13}$$

which gives

$$\|r\| \geq \frac{\mu^2}{\|J^T\|}\|\bar{\alpha} - \hat{\alpha}\|. \tag{3.14}$$

It follows from (3.14) that unless a very good initial estimate $\hat{\alpha}$ is known, a value of $\mu \ll \|J^T\|$ should be used.

For problems where $s > m - n$ and a reliable estimate $\hat{\alpha}$ is known, larger values for the elements of D may be chosen. Typical of such problems is STLN, as given by (1.1), where $\hat{\alpha} = 0$. For these cases the elements of D are positive integers giving the multiplicity of the occurrence of each α_i. Other examples include problems where $A(\alpha)$ has a structure (such as Vandermonde) to be preserved, and the values of its elements (possibly subject to error) are known.

4. SNTLN for Vandermonde matrices. In this section, we give a detailed description of the SNTLN algorithm for solving overdetermined systems with a Vandermonde structure,

$$A(\hat{\alpha})x \approx b, \tag{4.1}$$

where

$$A(\alpha) = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \alpha_1^2 & \alpha_2^2 & \cdots & \alpha_n^2 \\ \vdots & & & \\ \alpha_1^{m-1} & \alpha_2^{m-1} & \cdots & \alpha_n^{m-1} \end{pmatrix}, \qquad m \geq n, \tag{4.2}$$

and the parameter vector α is

$$\alpha = \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{pmatrix}.$$

The Vandermonde structure is one of the most frequently occurring nonlinear structures in applications [3, 8, 15, 22]. For example, in the exponential data modeling problems, m uniformly sampled data points y_i are given and are to be fitted to the following model function:

$$y_i \approx \sum_{j=1}^{n} x_j \alpha_j^i = \sum_{j=1}^{n} (a_j e^{\sqrt{-1}\phi_j}) e^{(-d_j + 2\pi\sqrt{-1} f_j) i \Delta t}, \qquad i = 0, \ldots, m-1,$$

where n is the model order and Δt is the constant sampling interval. The objective is to estimate the frequencies f_j, damping factors d_j, amplitudes a_j, and phases ϕ_j, $j = 1, \ldots, n$. The frequencies and damping factors can be found using one of several existing methods, e.g., the linear prediction method with the singular value decomposition [8] or the state–space-based method due to Kung et al. [8, 3, 15], which circumvents polynomial root finding and root selection. Improved versions of both methods, based on total least squares (TLS), are presented in [29]. In [21], we have shown that the STLN method further improves the accuracy of estimated frequencies and damping factors when it is used in the linear prediction method to preserve the Toeplitz structure. Once frequency and damping factors are found by using any of the methods mentioned above, they provide the estimate $\hat{\alpha}$ for the parameter vector α. Then the linear parameters x_j, which contain the amplitudes a_j and phases ϕ_j, $1 \leq j \leq n$, are estimated from solving the overdetermined Vandermonde system

$$A(\hat{\alpha})x \approx b, \qquad \text{where} \qquad b = \begin{pmatrix} y_0 \\ \vdots \\ y_{m-1} \end{pmatrix}.$$

We may also start the SNTLN iteration without using the methods mentioned above which provide the estimate $\hat{\alpha}$. However, as in any nonlinear problem, a good initial estimate is needed to obtain convergence.

In solving (4.1) using SNTLN, the perturbation on A will be found so that each $\hat{\alpha}_i$ is perturbed to $\alpha_i = \hat{\alpha}_i + h_i$ for some value h_i and the perturbed matrix $A(\alpha)$ keeps the Vandermonde structure. The solution vector x and the perturbation $h = (h_1 \cdots h_n)^T$ on the parameter $\hat{\alpha}$, which satisfy

$$A(\alpha)x = A(\hat{\alpha} + h)x = b - r,$$

will be found while minimizing $\| \left(\begin{smallmatrix} r \\ Dh \end{smallmatrix} \right) \|_p$.

For the Vandermonde matrix A, the Jacobian $J(\alpha, x)$ for $A(\alpha)x$ is

$$J(\alpha, x) = \begin{pmatrix} 0 & 0 & \cdots & 0 \\ x_1 & x_2 & \cdots & x_n \\ 2\alpha_1 x_1 & 2\alpha_2 x_2 & \cdots & 2\alpha_n x_n \\ \vdots & & & \\ (m-1)\alpha_1^{m-2} x_1 & (m-1)\alpha_2^{m-2} x_2 & \cdots & (m-1)\alpha_n^{m-2} x_n \end{pmatrix}. \tag{4.3}$$

Therefore, in Step 2(a) of Algorithm SNTLN, we need to solve a minimization problem with the $(m+n) \times 2n$ matrix $\left(\begin{smallmatrix} A(\alpha) & J \\ 0 & D \end{smallmatrix}\right)$, where J and $A(\alpha)$ are given in (4.3) and (4.2), respectively. Note that we have chosen not to perturb the first row of the matrix $A(\alpha)$.

In the next section, we present the numerical test results that compare the performance of SNTLN with $p = 2$ to that of LS and TLS for solving the Vandermonde overdetermined system. We also present results for two problems of the form (2.6) to show the effect of errors in the initial estimate $\hat{\alpha}$ and the ability of SNTLN with $p = 1$ to handle outliers in the data.

5. Computational test results. The SNTLN algorithm has been implemented in MATLAB in order to investigate its computational performance. We denote by SNTLN1 and SNTLN2 the SNTLN algorithm with $p = 1$ and $p = 2$, respectively. First, the accuracy of the solution computed by the SNTLN2 algorithm was compared to that of the LS and TLS methods in solving an overdetermined system $A(\hat{\alpha})x \approx b$, where $A(\hat{\alpha})$ is a Vandermonde matrix. In this test, we assume that the matrix $A(\hat{\alpha})$, i.e., the initial value $\hat{\alpha}$, is given, since this is required in applying the LS and TLS methods. Therefore, the SNTLN2 algorithm is used to solve a linear overdetermined system while preserving the nonlinear structure of the given matrix. Second, we present the computational test results that illustrate the effect on the convergence of the initial choice of $\hat{\alpha}$ and L_p norm on nonlinear parameter estimation problems.

5.1. Comparison of SNTLN2 to LS and TLS for Vandermonde overdetermined system. The computational tests were performed to compare the SNTLN2 solutions with the TLS and LS solutions for Vandermonde overdetermined systems. The initial value for $A(\alpha)$ is assumed to be given, and also it is assumed that there exists a "correct" Vandermonde matrix $A(\alpha_c)$ and vector b_c such that

$$A(\alpha_c)x_c = b_c \tag{5.1}$$

for some "correct" vector x_c. In other words, error-free values exist such that the overdetermined system has a solution x_c with zero residual. Since actual data contains noise, only the perturbed Vandermonde matrix $A(\alpha_c + \delta_\alpha) = A(\hat{\alpha})$ and the perturbed vector b_p are assumed to be known, instead of $A(\alpha_c)$ and b_c. The objective is to compare the three methods LS, TLS, and SNTLN2 in recovering the actual solution x_c by solving the perturbed system $A(\hat{\alpha})x_p \approx b_p$. Specifically, the error vector h for the parameter α, and residual vector r, are computed by solving

$$\min_{r,\alpha} \left\| \begin{pmatrix} r \\ Dh \end{pmatrix} \right\|_2 \qquad \text{such that} \qquad A(\alpha)x_p = A(\hat{\alpha} + h)x_p = b_p - r. \tag{5.2}$$

The test problems are constructed so that $A(\alpha_c)$, b_c, and x_c are known. Then random perturbations δ_α on α_c and δ_b on b_c are generated to give a Vandermonde matrix $A(\alpha_c + \delta_\alpha)$ and $b_p = b_c + \delta_b$, where the components of δ_α and δ_b are uniformly distributed random variables in a given interval. The matrix $A(\alpha)$ and r, x_p, satisfying (5.2) are then computed via LS, TLS, and SNTLN2. For LS, $\alpha = \hat{\alpha}$; i.e., $A(\alpha) = A(\hat{\alpha})$, since the matrix is not perturbed. For TLS, $A(\alpha) = A(\hat{\alpha}) + E$ for some matrix E since TLS does not preserve the structure or take the nonlinear dependence of A on α into account in computing the solution.

In Table 5.1, we present the test results of the following problem. Each data point shown represents the average of 100 solutions, each with different random values in

	(i) b is unperturbed			(ii) $\vert$ pert. in $b_i\vert \leq$1.0e-8			(iii) $\vert$ pert. in $b_i\vert \leq \gamma$		
γ	LS	TLS	SNTLN2	LS	TLS	SNTLN2	LS	TLS	SNTLN2
1.0e-8	4.8e-8	4.8e-8	4.9e-15	4.5e-8	4.5e-8	2.5e-8	4.6e-8	4.6e-8	2.5e-8
1.0e-6	4.5e-6	4.5e-6	2.2e-16	5.0e-6	5.0e-6	2.5e-8	4.6e-6	4.6e-6	2.5e-6
1.0e-4	4.9e-4	4.9e-4	1.7e-14	5.0e-4	5.0e-4	2.7e-8	4.6e-4	4.6e-4	2.3e-4
1.0e-3	5.0e-3	5.0e-3	3.5e-16	4.2e-3	4.2e-3	2.7e-8	4.7e-3	4.7e-3	2.5e-3
1.0e-2	4.5e-2	4.6e-2	2.1e-14	4.9e-2	4.9e-2	2.4e-8	4.3e-2	4.3e-2	2.7e-2
1.0e-1	4.2e-1	4.2e-1	5.1e-2	4.7e-1	5.7e-1	1.1e-1	5.1e-1	5.1e-1	3.4e-1

TABLE 5.1

Solution error $\frac{\|x_p - x_c\|_2}{\|x_c\|_2}$ *of* x_p *computed by LS, TLS, and SNTLN*2.

the range $[-\gamma, \gamma]$. In the test, $A(\alpha_c)$ is a 15×3 Vandermonde matrix where

$$\alpha_c = \begin{pmatrix} e^{-0.1+2\pi\sqrt{-1}*0.5} \\ e^{-0.2+2\pi\sqrt{-1}*0.4} \\ e^{-0.3+2\pi\sqrt{-1}*0.3} \end{pmatrix}, \qquad x_c = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \qquad b_c = A(\alpha_c)x_c.$$

Then α_c is perturbed by δ_α to give $\hat{\alpha} = \alpha_c + \delta_\alpha$, where the components of δ_α are uniformly distributed random variables in the interval $[-\gamma, \gamma]$. For the perturbation in b_c, we have tested three different cases: (i) when b_c is unperturbed, (ii) b_c is perturbed by uniformly distributed random variables in the interval [-1.0e-8, 1.0e-8], and (iii) b_c is perturbed by uniformly distributed random variables in the interval $[-\gamma, \gamma]$ like α_c. These three cases were tested to study how well the SNTLN2 recovers the correct solution x_c when the given data $\hat{\alpha}$ and b are affected by errors of different kind. The matrix D was chosen to be diag(1.0e-8) in this test.

The iteration in the SNTLN2 algorithm was continued until both

$$\|\Delta\alpha\| \leq \text{e-6 and } \|\Delta x\| \leq \text{e-6}$$

were satisfied. When this convergence test is not satisfied within 20 iterations, then the iteration was terminated and the result obtained at the 20th iteration is taken as the final solution. For most values of γ, this stopping criterion was always satisfied. However, when γ =1.0e-1, in 3 problems for case (i), 8 problems for case (ii), and 7 problems for case (iii), out of 100 test problems each, the iteration was terminated after 20 iterations since the convergence test was not satisfied. This explains the sudden deterioration of the performance of SNTLN2 in tests (i) and (ii) for γ =1.0e-1. However, the performance of SNTLN2 is still better than that of LS or TLS. When the average relative error of SNTLN2 was computed only for the cases converged within 20 iterations, the accuracy was increased to 8.6e-15 and 2.5e-8 in tests (i) and (ii), respectively, even for γ =1.0e-1.

5.2. Effect of errors in data and parameter estimates. As discussed in the previous section, the SNTLN algorithm will converge to the desired minimum vector α_c from an initial estimate $\hat{\alpha}$ if $\hat{\alpha}$ is sufficiently close to α_c. However, the practical question of how large a value of $\|\hat{\alpha} - \alpha_c\|$ will give convergence can only be answered by computational testing. The preliminary computational testing summarized in this section was carried out to explore this property of SNTLN. In addition, we wanted to study two related properties:

1. The effect of data errors on the computed estimate $\overline{\alpha}$ of the parameter vector. Specifically, the effect of the data errors on $\|\overline{\alpha} - \alpha_c\|$.
2. The robustness of the L_1-norm with respect to outliers in the data.

ϵ	Outlier Norm	$\|\overline{\alpha}-\alpha_c\|/\|\alpha_c\|$ from SNTLN2	$\|\overline{\alpha}-\alpha_c\|/\|\alpha_c\|$ from SNTLN1
0	5e-3	9.6e-3	0
5e-9	5e-3	9.6e-3	2.7e-7
5e-8	5e-3	9.6e-3	2.7e-6
5e-7	5e-3	9.6e-3	1.7e-5
5e-6	5e-3	9.6e-3	1.6e-4
5e-5	5e-3	1.1e-2	2.1e-3
5e-5	0	9.0e-3	1.9e-3

TABLE 5.2

Effect of errors in data on parameter estimate—type 1 *Signal.*

We now present computational results which show the ability of the SNTLN algorithm to determine good values of the parameter vector α and the coefficient vector x, in spite of noise in the data and relatively poor estimates of α_c. In order to carry out these computational tests, two different parameter estimation problems were used. For each test problem it is assumed that a noiseless signal $f(t)$ is of the form given below. The measured signal at m values of t is assumed to have the form

$$f_i = f(t_i) + \eta_i, \qquad i = 1, \ldots, m, \tag{5.3}$$

where the η_i represent noise or error in the measurement. The following two types of signal were chosen:

$$\begin{array}{lll} 1. & f(t) & = & \sum_{j=1}^{n} x_j e^{-\alpha_j t}, \\ 2. & f(t) & = & \sum_{j=1}^{n} x_j e^{-(t-\alpha_j)^2/\sigma^2}. \end{array}$$

Given the corresponding noisy data $f_i, i = 1, \ldots, m$, it is desirable to get the best estimate of the true parameter vector α_c and linear coefficient vector x_c, which determine the undistorted signal $f(t)$. We also know an initial estimate $\hat{\alpha}$ of α_c, which may also be in error.

The data for the signal of type 1 was obtained from that used by Osborne and Smyth [19]. Specifically, the values $\alpha_c = (\,0 \quad 4 \quad 7\,)^T$ and $x_c = (\,0.5 \quad 2 \quad -1.5\,)^T$ were used to give $f(t)$ over the interval $t \in [0, 1]$. A total of 30 points t_i were used so that $m = 30$, $n = 3$, and $s = 3$.

The sum of Gaussian functions, type 2 signal, is similar to that used in [13]. For this test, the values $\sigma^2 = 0.05$, $\alpha_c = (0.1\ 0.3\ 0.5\ 0.9)^T$, and $x_c = (1.0\ 0.5\ 2.0\ 0.25)^T$ were used. The values of α_j and x_j were to be determined (σ is assumed to be known). The 64 values of t_i were chosen to be equally spaced in $[0, 1]$ so that $m = 64$, $n = 4$, and $s = 4$. Since $n + s < m$ for both signal types, a small value (1.0e-8) was again chosen for the diagonal elements of D.

We used the type 1 signal to measure the effect of errors in the data vector f_i on the computed parameter estimate $\bar{\alpha}$. First, uniformly distributed random errors η_i, $i = 1, \ldots, m$, in the interval $[-\epsilon, \epsilon]$ were added to $f(t_i)$ to give f_i as in (5.3). With ϵ =5.0e-5, the relative error $\frac{\|\bar{\alpha}-\alpha_c\|}{\|\alpha_c\|}$ was 1.9e-3 using SNTLN1 and 9.0e-3 using SNTLN2. This is shown in the last row of Table 5.2. Additional tests with uniformly distributed random errors showed that the relative error in $\bar{\alpha}$ is proportional to ϵ for both $p = 1$ and $p = 2$ and that the relative error for $p = 1$ is somewhat less than the relative error for $p = 2$. For $\epsilon = 0$, as expected, we get $\bar{\alpha} = \alpha_c$. The initial estimate $\hat{\alpha} = \alpha_c$ was used for all these tests, so the error in $\bar{\alpha}$ was due entirely to the error in the data.

The next set of tests were to determine the effect of outliers in the data on the computed parameter estimate. In addition to the random errors η_i at each t_i, a single outlier was introduced at one of the points t_i. Its magnitude was 5.0e-3. The parameter estimate $\bar{\alpha}$ was then computed for a sequence of increasing values of ϵ. The value of ϵ was increased from 5.0e-9 to 5.0e-5. For each value of ϵ, the parameter estimate was computed using SNTLN with $p = 1$ and $p = 2$. The results are summarized in the first six rows of Table 5.2. It is seen that for $p = 2$, the effect of the outlier dominated the error in the parameter estimate. In contrast, the use of SNTLN1 produces a parameter estimate which ignores the outlier and depends only on the size of the random errors η_i. In particular, the parameter estimate error is essentially proportional to ϵ and is the same whether an outlier is present or not. Changing the location of the outlier had only a small effect on these results. Changing the magnitude of the outlier had no effect on the results with $p = 1$. This shows that SNTLN1 is very robust with respect to a single large error in the data.

Additional tests were made using more than one outlier with the type 1 signal. Outliers were introduced at randomly chosen time points t_i, with all other values of $f_i = f(t_i)$, that is, without error. Surprisingly, it was found that up to 10 outliers could be added, in some cases, with no adverse effect on the estimate $\overline{\alpha}$, using $p = 1$. That is, SNTLN1 almost always gave $\overline{\alpha} = \alpha_c$. However, when there were more than 10 outliers, the error $\|\overline{\alpha} - \alpha_c\|$ was comparable to the norm of the outliers.

The effect of error in the initial parameter estimate $\hat{\alpha}$ was investigated using a signal of type 2. In addition, the robustness, with respect to outliers, of SNTLN1 was further confirmed with this larger, and very different, type of test problem.

The ability of both SNTLN1 and SNTLN2 to converge to α_c from different initial parameter estimates $\hat{\alpha}$ was tested. This was done by choosing

$$\hat{\alpha} = \alpha_c + \delta, \tag{5.4}$$

where each δ_i, $i = 1, \ldots q$, is a uniformly distributed random variable in the interval $[-\gamma, \gamma]$. In addition, a uniformly distributed random error η_i was added to f_i, with $|\eta_i| \leq$e-7, for all cases. Convergence to α_c becomes more difficult as γ increases. For the purposes of this test we say SNTLN has converged if both $\|\Delta\alpha\| \leq$e-6 and $\|\Delta x\| \leq$e-6, in 10 iterations or less. The results for SNTLN1 and SNTLN2 are shown in Figure 5.1. Each data point shown represents the result of 20 solutions, each with different random values of the δ_i. The percentage of the solutions which converged is plotted as a function of the maximum value of $|\delta_i|$, as given by γ. Six values of γ were used, $\gamma = (0, .01, .02, .03, .05, .07)$, and since the smallest $\alpha_1 = 0.1$, the initial estimate error in α_1 could be as large as 70%. The upper plot (denoted by o) shows the results for no outliers and represents both SNTLN1 and SNTLN2. It is seen that the percentage of convergence is the same for both, and in fact, all cases converged for $\gamma \leq .02$. For $\gamma = .07$, the percentage of convergence drops to 75%. For no outliers, the error in the final computed estimate $\bar{\alpha}$, is determined by the random errors η_i in f_i and was never greater than 3.3e-6 for either SNTLN1 or SNTLN2.

In order to investigate the effect of outliers, the values of γ used above were repeated, with 10 and then 25 outliers. The outliers each had a value of ± 0.1 and were added to f_i (in addition to η_i) at randomly selected positions. Thus the total error in the signal f_i at t_i consisted of a small random perturbation η_i at every point t_i and a much larger outlier at either 10 or 25 of the 64 total points. The results are shown in the lower two curves in Figure 5.1. The SNTLN2 algorithm did not satisfy the convergence criterion in many cases so the results shown represent the SNTLN1 algorithm only.

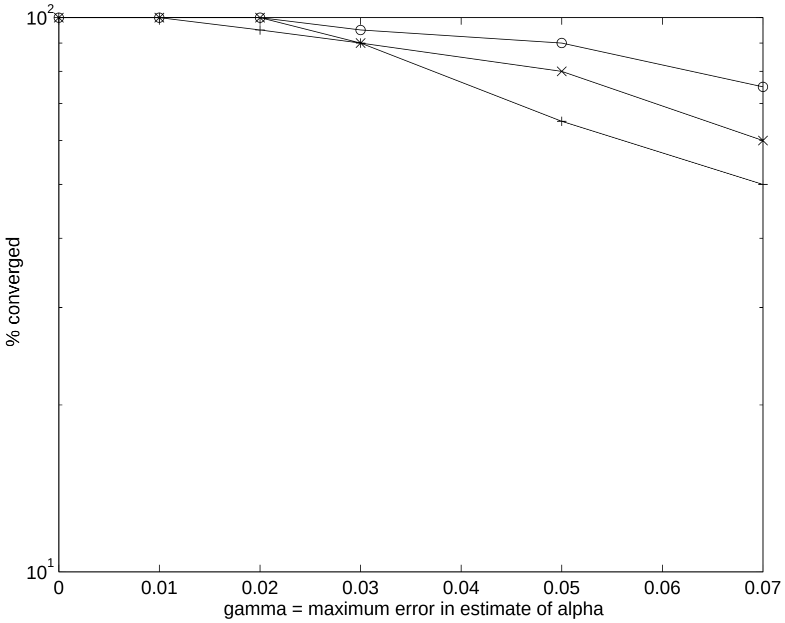


Fig. 5.1. *Effect of initial $\hat{\alpha}$ estimate error and number of outliers on convergence of SNTLN*1 *and SNTLN*2*. Random error in $f = 1.0e$-7, outlier magnitude* = 0.1*. Key:* o *(no outliers, L*1 *and L*2*),* × *(*10 *outliers, L_1 only),* + *(*25 *outliers, L_1 only). Maximum number of iterations for convergence: L_1 (*6 *iterations) and L_2 (*9 *iterations). Maximum error in computed $\alpha = 7.8e$-6, using SNTLN*1*.*

The results show that for 10 outliers, the percentage of convergence is 100% for $\gamma \leq .02$ and drops to 60% for $\gamma = .07$. For 25 outliers, the percentage of convergence is 100% for $\gamma \leq .01$, and it drops to 50% for $\gamma = .07$. It is important to note that the converged value of $\bar{\alpha}$ never had an error greater than 7.8e-6, even with 25 outliers. This means that when SNTLN1 converges, the error in the final estimate of α is determined completely by the values of the random errors η_i and is *unaffected by the outliers*. Furthermore, when convergence took place it never required more than 6 iterations of SNTLN1. Based on these computational results, it appears that SNTLN1 is very robust with respect to outliers, provided that the total number of outliers is somewhat less than $(m - n)/2$. The explanation of this robust behavior is being investigated, both theoretically and computationally [26].

Acknowledgments. We would like to thank Mr. Lei Zhang for carrying out the numerical tests, and the anonymous reviewers for valuable comments which made it possible to improve the paper.

REFERENCES

[1] T. J. Abatzoglou, J. M. Mendel, and G. A. Harada, *The constrained total least squares technique and its application to harmonic superresolution*, IEEE Trans. Signal Processing, 39 (1991), pp. 1070–1087.

[2] A. A. Anda and H. Park, *Self-scaling fast rotations for stiff least squares problems*, Linear Algebra Appl., 234 (1996), pp. 137–161.

[3] H. Barkhuijsen, R. De Beer, and D. Van Ormondt, *Improved algorithm for noniterative time-domain model fitting to exponentially damped magnetic resonance signals*, J. Mag-

netic Resonance, 73 (1987), pp. 553–557.

[4] I. Barrodale and A. Young, *Algorithms for best L_1 and L_∞ linear approximations on a discrete set*, Numer. Math., 8 (1966), pp. 295–306.

[5] I. Barrodale, *L_1 approximation and the analysis of data*, Appl. Statistics, 17 (1968), pp. 51–57.

[6] D. Bates and M. Lindstrom, *Nonlinear least squares with conditionally linear parameters*, in Proceedings of the Statistical Computing Section, American Statistical Association, Washington, DC, 1986, pp. 152–157.

[7] H. Chen, S. Van Huffel, and J. Vandewalle, *Exponential Data Fitting Using the Structured Total Least Norm Technique*, Technical report TR 95-18, Departement Elektrotechniek ESAT-SISTA, University of Leuven, Belgium, March, 1995.

[8] R. De Beer and D. Van Ormondt, *Analysis of NMR data using time-domain fitting procedures*, in In-vivo Magnetic Resonance Spectroscopy I: Probeheads, Radiofrequency Pulses, Spectrum Analysis, NMR Basic Principles and Progress 26, M. Rudin, ed., Springer-Verlag, Berlin, Heidelberg, 1992, pp. 201–248.

[9] R. Fletcher, *Practical Methods of Optimization*, John Wiley, New York, 1987.

[10] G. H. Golub and V. Pereyra, *The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate*, SIAM J. Numer. Anal., 10 (1973), pp. 413–432.

[11] T. A. Grandine, *Generating Surface Lofts to Scattered Data*, Engineering Computing and Analysis Technical report ECA-TR-157, Boeing Computer Services, Seattle, WA, 1991.

[12] L. Kaufman, *A variable projection method for solving separable nonlinear least squares problems*, BIT, 15 (1975), pp. 49–57.

[13] L. Kaufman and G. Sylvester, *Separable nonlinear least squares with multiple right-hand sides*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 68–89.

[14] R. Kumaresan and D. W. Tufts, *Estimating the parameters of exponentially damped sinusoids and pole-zero modeling in noise*, IEEE Trans. Acoust. Speech Signal Proc., 30 (1982), pp. 833–840.

[15] S. Y. Kung, K. S. Arun, and D. V. Bhaskar Rao, *State-space and singular value decomposition-based approximation methods for the harmonic retrieval problem*, J. Opt. Soc. Amer., 73 (1983), pp. 1799–1811.

[16] C. L. Lawson and R. J. Hanson, *Solving Least Squares Problems*, Prentice–Hall, Englewood Cliffs, NJ, 1974.

[17] Yuying Li, *Solving L_p-norm Problems and Applications*, Cornell Theory Center report CTC93TR122, Cornell University, Ithaca, NY, 1993.

[18] M. R. Osborne, *Some special nonlinear least squares problems*, SIAM J. Numer. Anal., 12 (1975), pp. 571–592.

[19] M. R. Osborne and G. K. Smyth, *A modified Prony algorithm for exponential function fitting*, SIAM J. Sci. Comput., 16 (1995), pp. 119–138.

[20] H. Park, J. B. Rosen, and J. Glick, *Structure preserving total least norm method and application to parameter estimation*, in Proceedings of the 1995 International Conference on Acoustics, Speech and Signal Processing, Vol. 2, 1995, pp. 1141–1144.

[21] H. Park, J. B. Rosen, and S. Van Huffel, *Structure preserving total least squares method and its application to parameter estimation*, SVD and Signal Processing, III: Algorithms, Architectures and Applications, M. Moonen and B. De Moor, eds., Elsevier, New York, 1995, pp. 399–406.

[22] M. A. Rahman and K. B. Yu, *Total least squares approach for frequency estimation using linear prediction*, IEEE Trans. Acous. Speech Signal Proc., 35 (1987), pp. 1440–1454.

[23] J. R. Rice and J. S. White, *Norms for smoothing and estimation*, SIAM Rev., 6 (1964), pp. 243–256.

[24] J. B. Rosen, H. Park, and J. Glick, *Total least norm formulation and solution for structured problems*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 110–128.

[25] J. B. Rosen, H. Park, and J. Glick, *Total least norm for linear and nonlinear structured problems*, in Recent Advances in Total Least Squares Techniques and Errors-in-Variables Modeling, S. Van Huffel, ed., SIAM, Philadelphia, 1996, pp. 203–214.

[26] J. B. Rosen, H. Park, J. Glick, and L. Zhang, *Accurate Solution to Overdetermined Systems with Errors, Using L_1 Norm Minimization*, Tech. report 98-7, Department UCSD, La Jolla, CA, January 13, 1988 (submitted for publication).

[27] H. Späth and G. A. Watson, *On orthogonal linear l_1 approximation*, Numer. Math., 51 (1996), pp. 531–543.

[28] S. Van Huffel, H. Park, and J. B. Rosen, *Formulation and solution of structured total least norm problems for parameter estimation*, IEEE Trans. Signal Process., (1996), pp. 2464–2474.

[29] S. Van Huffel, L. Aerts, J. Bervoets, J. Vandewalle, C. Decanniere, and P. Van Hecke, *Improved quantitative time-domain analysis of NMR data by total least squares*, in Signal Processing *VI*: Theories and applications, J. Vandewalle, R. Boite, M. Moonen and A. Oosterlinck, eds., Elsevier–North Holland, 1992, pp. III:1721–1724.

[30] S. Van Huffel and J. Vandewalle, *The Total Least Squares Problem, Computational Aspects and Analysis*, SIAM, Philadelphia, 1991.

[31] G. A. Watson and K. F. C. Yiu, *On the solution of the errors in variables problem using the l_1 norm*, BIT, 31 (1991), pp. 697–710.

SIAM J. Matrix Anal. Appl.
Vol. 20, No. 1, pp. 31–44

ASYMPTOTIC RESULTS ON THE SPECTRA OF BLOCK TOEPLITZ PRECONDITIONED MATRICES*

STEFANO SERRA[†]

Abstract. It is well known that the *generating* function $f \in L^1([-\pi,\pi],\mathbf{R})$ of a class of Hermitian Toeplitz matrices $A_n(f)$ describes very precisely the spectrum of each matrix of the class [U. Grenader and G. Szegö, *Toeplitz Forms and Their Applications*, 2nd ed., Chelsea, New York, 1984; E. E. Tyrtyshnikov, *Linear Algebra Appl.*, 232 (1996), pp. 1–43]. In this paper we consider $n \times n$ block Toeplitz matrices with $m \times m$ blocks generated by a Hermitian matrix-valued generating function $f \in L^1([-\pi,\pi],\mathbf{C}^{m\times m})$ and, in particular, we analyze the associated problem of *preconditioning.* Using previous results on this topic [P. Tilli and M. Miranda, *SIAM J. Matrix Anal. Appl.*, to appear], we extend some theorems to this case that were proved in the one-level Toeplitz case [F. Di Benedetto, G. Fiorentino, and S. Serra, *Comput. Math. Appl.*, 25 (1993), pp. 35–45; S. Serra, *SIAM J. Matrix Anal. Appl.*, 17 (1996), pp. 1007–1019; S. Serra, *Calcolo*, 32 (1995), pp. 153–176] as well as in the two-level Toeplitz case [S. Serra, *Linear Algebra Appl.*, 270 (1998), pp. 109–129]. This idea seems promising when dealing with linear systems arising from control theory and Markov chains theory [R. Preuss, *Workshop on Toeplitz matrices in Filtering and Control*, Santa Barbara, CA, August 1996; M. Neuts, *Structured Stochastic Matrices of M/G/1 Type and Their Applications*, Dekker, New York, 1989; E. Cinlar, *Introduction to Stochastic Processes*, Prentice–Hall, Englewood Cliffs, NJ, 1975].

Key words. Toeplitz matrix, generating function, preconditioning, conjugate gradient method

AMS subject classifications. 65F10, 65F15

PII. S0895479896310160

1. Introduction.

In this paper we are interested in the efficient solution of a linear system of the form

$$A_{n,m}(f)\mathbf{x} = \mathbf{b}, \tag{1}$$

where $A_{n,m}(f)$ is generated by an integrable Hermitian matrix-valued function $f \in L^1([-\pi,\pi],\mathbf{C}^{m\times m})$ defined on the fundamental interval $I = [-\pi,\pi]$ and extended periodically to the whole real axis. More precisely, the blocks of the matrix $A_{n,m}(f)$ along the kth diagonal are all given by

$$A_k \equiv A_k(f) = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(x)e^{-\mathbf{i}kx}dx, \quad \mathbf{i}^2 = -1, \quad k \in \mathbf{Z}. \tag{2}$$

For the scalar case the spectral properties of these matrices have been deeply studied in this century (see, for instance, [19, 41, 39]). In the case where $m > 1$, we obtain a kind of matrix which characterizes applications like the ones connected to industrial control theory [23] and Markov chains [22, 12, 7]. We restrict our attention to the case where $f(x)$ is a Hermitian matrix-valued function; in [36, 33] some results regarding the localization and asymptotic extreme behavior of the spectra are obtained. Moreover, in the following, we make use of a worthwhile theorem [36] which gives information about the asymptotical behavior of the spectra of the family

[*] Received by the editors October 7, 1996; accepted for publication (in revised form) by S. Van Huffel August 13, 1997; published electronically September 15, 1998.

http://www.siam.org/journals/simax/20-1/31016.html

[†] Dipartimento di Energetica, Via Lombroso 6/17, 50100 Firenze, Italy; Dipartimento di Informatica, Corso Italia 40, 56100 Pisa, Italy (serra@mail.dm.unipi.it).

$\{A_{n,m}(f)\}_{n,m}$ and which can be viewed as an elegant block version of the classical Szegö–Tyrtyshnikov theorem [19, 39].

THEOREM 1.1 (see [36]). *Let $f \in L^2([-\pi,\pi], \mathbf{C}^{m\times m})$ and let $\lambda_i^{(n,m)}$ be the eigenvalues of $A_{n,m}(f)$ (which are real since f is Hermitian matrix valued and the matrix $A_{n,m}(f)$ is Hermitian). Then, for any continuous function $F \in \mathbf{C}(\mathbf{R})$, with bounded support, the asymptotic formula*

$$\lim_{n\to\infty} \frac{1}{nm}\sum_{i=1}^{nm} F(\lambda_i^{(n,m)}) = \frac{1}{2\pi}\int_{-\pi}^{\pi} \frac{1}{m}\mathrm{trace}(F(f(x)))dx \tag{3}$$

holds.

When f is also nonnegative definite and is not essentially singular, the related matrix $A_{n,m}(f)$ is positive definite for any choice of the external dimension n and we can solve the associated systems by means of preconditioned conjugate gradient (PCG) methods. Since one of the most successful preconditioning strategies for the scalar case is the one based on (simple) Toeplitz matrices [6, 14, 10, 28, 27, 8, 21, 7, 29], in [33] we generalized this idea by obtaining "optimal" preconditioners where "optimal" is intended in the classical sense described, for instance, in [2].

The preconditioners will have the form of the original coefficient matrix: in effect, we consider the positive definite matrix $A_{n,m}(g)$ generated by a nonnegative definite, not essentially singular, matrix-valued function g and then, by following a known approach [14, 6], the preconditioned matrix takes the form $\mathcal{P}_{n,m}(f;g) = A_{n,m}^{-1}(g)A_{n,m}(f)$. In order to take advantage of this technique, we will choose the generating function in such a way that a generic system, having as coefficient matrix $A_{n,m}(g)$, is "simple to solve." If $g^{-1/2}fg^{-1/2}$ is associated with a well-conditioned quadratic form, that is, the m eigenvalues of $p = g^{-1}f$ range essentially in a positive interval $[r, R]$, then the preconditioned matrix $\mathcal{P}_{n,m}(f;g)$ has eigenvalues in (r, R) [33, 36]. Here, by making heavy use of Theorem 1.1 and some simple consequences, we show that the union of all the spectra of $\{\mathcal{P}_{n,m}(f;g)\}$ has a usual topological closure which contains the union of the essential ranges of the m eigenvalues of $g^{-1}f$ (see Theorem 3.6). This is a natural extension of the density results obtained previously for the scalar Toeplitz case [14, 27, 26]. Furthermore, by making use of generalized Rayleigh quotients associated with $\mathcal{P}_{n,m}(f;g)$, we obtain some partial results on the asymptotic behavior of the condition numbers of $A_{n,m}(f)$.

In addition, if $R/r \gg 1$, we expect a number of iterations [1] to reach the solution of (1), within an accuracy ϵ, less than

$$N1(f,g,\epsilon) = \frac{1}{2}\sqrt{\frac{R}{r}}\log\left(\frac{2}{\epsilon}\right) + 1. \tag{4}$$

Otherwise the estimate takes the following form:

$$N2(f,g,\epsilon) = \frac{\log\left(2\epsilon^{-1}\right)}{\log(\delta)} + 1, \tag{5}$$

with

$$\delta = \frac{1+\sqrt{\frac{r}{R}}}{1-\sqrt{\frac{r}{R}}}.$$

A consequence of the analysis performed in the following is the "unrigorous" confirmation of the tightness of the previous bounds (for a rigorous analysis see [18]).

Actually, owing to the "uniform" distribution of the eigenvalues of $A_{n,m}$ in the essential range of the eigenvalues of $g^{-1}f$ (Theorem 3.6), when $\bigcup_{i\leq m}\mathcal{ER}(\lambda_i(g^{-1}f))$ coincides with $[r,R]$, we expect that the previous upper bounds are tight. This fact is fully confirmed in the numerical experiments at the end of section 4.

The paper is organized as follows. In section 2 we summarize the known theory on this kind of unstructured block Toeplitz matrix. In section 3 we derive the density result for the preconditioned matrices and we introduce a tool to evaluate the asymptotic condition numbers of $A_{n,m}(f)$. Finally, in the last section, we discuss the applications to the related PCG methods for solving systems of the form (1).

2. Spectral properties of $\boldsymbol{A_{n,m}(f)}$. We denote by $A_{n,m}(f)$ the $n\times n$ block Toeplitz matrix with $m\times m$ blocks defined in equation (2).

If (j,k) indicates the block in the matrix $A_{n,m}=A_{n,m}(f)$ and (p,q) the position of the entry in the block, then we have

$$(A_{n,m})_{(j,k)(p,q)}=(A_{k-j})_{p,q}$$

for $j,k=1,\dots,n,\ p,q=1,\dots,m$.

Some properties of $A_{n,m}$ are easily obtained as stated in the following lemma.

LEMMA 2.1. *Let $f:I\to\mathbf{C}^{m\times m}$ be an L^1 Hermitian matrix-valued function in the sense that, for any couple (p,q), $(f(x))_{p,q}$ is an L^1 scalar complex-valued function and $(f(x))_{p,q}=(\overline{f(x)})_{q,p}$. Under these assumptions, $A_k=A^H_{-k}$ and therefore the matrix $A_{n,m}$ is globally Hermitian.*

In addition, if f is symmetric then $A_{n,m}$ is further specified in the sense that $A_{n,m}$ is a symmetric real matrix with symmetric blocks.

As in the scalar case, the spectrum is localized by considering the range of the generating function f.

THEOREM 2.2 (see [33]). *If $f:I\to\mathbf{C}^{m\times m}$ is a Hermitian matrix-valued function, then the eigenvalues of $A_{n,m}(f)$ lie in the interval $[m_f,M_f]$, where*

$$m_f=\mathrm{essinf}_I\min_{1,\dots,m}(\lambda(f(x)))$$

and

$$M_f=\mathrm{esssup}_I\max_{1,\dots,m}(\lambda(f(x))).$$

When $m=1$ there exists a stronger result [19] which assures that if $m_f<M_f$ then all the eigenvalues of $A_n(f)$ belong to the open interval (m_f,M_f).

In the scalar case, when $m_f=0$ and M_f is positive, the associated Toeplitz matrices are positive definite and ill conditioned. The positive definiteness is an important property because it allows us to define the basic Toeplitz preconditioners first considered in [6, 14] and then applied in modified ways to several different problems [10, 27, 28, 7, 25].

Unfortunately, the situation is a bit different in this block case as demonstrated in the following elementary example. Let $m=2$ and $f(x)$ be a constant Hermitian matrix-valued function B:

$$B=\begin{bmatrix}1&0\\0&0\end{bmatrix}.$$

Evidently, the associated matrices $A_{n,m}(f)$ have the simple form of block diagonal matrices, that is, $A_{n,m}(f)$ =diag$(B, B, \ldots, B)$. In this case $m_f = 0$ and $M_f = 1$ and the eigenvalues actually coincide with 0 or with 1.

The following is an important refinement of Theorem 2.2 and, actually, gives conditions in order to state that all the eigenvalues of $A_{n,m}(f)$ are in the open interval (m_f, M_f).

Theorem 2.3 (see [33]). *Let m_f and M_f be as in Theorem 2.2 and let us define $\hat{m}_f$ as* $\operatorname{esssup}_I \min_{1,\ldots,m}(\lambda(f(x)))$ *and $\hat{M}_f$ as* $\operatorname{essinf}_I \max_{1,\ldots,m}(\lambda(f(x)))$. *Then the following facts are true:*

1. *if $m_f < \hat{m}_f$ (that is, the minimal eigenvalue is not essentially constant when x varies in I), then all the eigenvalues λ lie in the interval $(m_f, M_f]$.*
2. *if $M_f > \hat{M}_f$ (that is, the biggest eigenvalue is not essentially constant when x varies in I), then all the eigenvalues λ lie in the interval $[m_f, M_f)$.*

3. The preconditioned Toeplitz matrices $\mathcal{P}_{n,m}(f; g)$. The aim of this section is to extend some results concerning Toeplitz preconditioning for scalar Toeplitz linear systems to the block case. For the scalar case, this strategy is very successful because of its great flexibility in applications to a large variety of different situations [6, 14, 26, 7, 10, 29, 27, 25].

Here and in the following, by *preconditioned Toeplitz* matrix, we define a matrix of the form $A_{n,m}^{-1}(g)A_{n,m}(f)$, which is also indicated by the shorter symbol $\mathcal{P}_{n,m}(f; g)$, where f and g are two L^1 Hermitian matrix-valued functions with g essentially nonnegative definite and not everywhere singular. We observe that, from the assumptions, the matrices $A_{n,m}(f)$ and $A_{n,m}(g)$ are well defined and $A_{n,m}(g)$ is positive definite (see Theorem 2.3); therefore the *preconditioned matrix* $A_{n,m}^{-1}(g)A_{n,m}(f)$ exists and is well defined.

The localization result is a very important one because it represents the main tool for defining Toeplitz preconditioners for Toeplitz linear systems, similar to the scalar case.

Theorem 3.1 (see [33]). *Let $\{A_{n,m}(g)\}_{n,m}$ and $\{A_{n,m}(f)\}_{n,m}$ be two sequences of Toeplitz matrices generated by two Lebesgue integrable Hermitian matrix-valued functions f and g, where g is essentially nonnegative definite and not identically singular. Let us suppose that, for any real value α, $m_{f-\alpha g} < \hat{m}_{f-\alpha g}$, $\hat{M}_{f-\alpha g} < M_{f-\alpha g}$. Then, for any positive integer n, the preconditioned matrix*

$$\mathcal{P}_{n,m}(f; g)$$

has eigenvalues in the set (r, R), where

$$r = \operatorname{essinf}_{x\in I} \min\{\lambda(g^{-1}f)\}, \quad R = \operatorname{esssup}_{x\in I} \max\{\lambda(g^{-1}f)\},$$

and $r < R$. Otherwise, if $r = R$ then the preconditioned matrix has the form $\mathcal{P}_{n,m}(f; g) = r \cdot I$.

Now we are going to prove a density result which allows one to evaluate very precisely a priori the number of iterations of the PCG method when a block Toeplitz preconditioning is used. For the sake of completeness, we introduce the definition of the essential range of a Lebesgue measurable function.

Definition 3.2. *Given a real-valued measurable function h defined on I, the essential range $\mathcal{ER}(h)$ of h is the closed set of all the real numbers y such that $\forall \epsilon > 0$, the Lebesgue measure of $\{x \in I : \; h(x) \in (y - \epsilon, y + \epsilon)\}$ is positive.*

Furthermore, in order to simplify the proof of the density result of Theorem 3.6, we premise three technical lemmas concerning the distribution of the eigenvalues of Toeplitz matrices.

LEMMA 3.3. *If $h \in L^2(I, \mathcal{C}^{m\times m})$ is a Hermitian-valued function and*

$$\sum_{j=1}^{m} m\{x \in I : \ \lambda_j(h(x)) = a\} = 0,$$

then

$$\lim_{n\to\infty} \frac{\#\{i : \ \lambda_i(A_{n,m}(h)) < a\}}{n} = \frac{1}{2\pi} \sum_{j=1}^{m} m\{x \in I : \ \lambda_j(h(x)) < a\}.$$

Proof. First observe that the relation we want to prove is the relation (3) with $F_a = Ch_{[0,a]}$, where Ch_X indicates the characteristic function of the set X. The difficulty is that F_a is not continuous as requested by Theorem 1.1. Therefore we consider two families of continuous approximations of F_a, namely, $\{F_\delta^+\}_\delta$ and $\{F_\delta^-\}_\delta$. The function F_δ^- is continuous piecewise linear and is identically 1 over $[0, a-\delta]$ and identically zero over $[a, \infty)$. The function F_δ^+ is continuous piecewise linear and is identically 1 over $[0, a]$ and identically zero over $[a+\delta, \infty)$. It follows that $F_\delta^- \leq F_a \leq F_\delta^+$. Now we use relation (3) with $F = F_\delta^-$ and with $F = F_\delta^+$ by obtaining that

$$\lim_{n\to\infty} \frac{\sum_i F_\delta^-(\lambda_i(A_{n,m}(h)))}{n} = \frac{1}{2\pi} \sum_{j=1}^{m} m\{x \in I : \ \lambda_j(h(x)) < a\} + k_1(\delta)$$

and

$$\lim_{n\to\infty} \frac{\sum_i F_\delta^+(\lambda_i(A_{n,m}(h)))}{n} = \frac{1}{2\pi} \sum_{j=1}^{m} m\{x \in I : \ \lambda_j(h(x)) < a\} + k_2(\delta),$$

with $k_1 \leq 0 \leq k_2$. Moreover, from the construction of $F_\delta^\pm$, it follows that

$$\lim_{\delta\to 0} k_i(\delta) = 0, \qquad i = 1, 2.$$

But from the inequality $F_\delta^- \leq F_a \leq F_\delta^+$, we easily deduce that

$$\frac{\sum_i F_\delta^-(\lambda_i(A_{n,m}(h)))}{n} \leq \frac{\#\{i : \ \lambda_i(A_{n,m}(h)) < a\}}{n} \leq \frac{\sum_i F_\delta^+(\lambda_i(A_{n,m}(h)))}{n}.$$

Since δ can be chosen arbitrarily small the claimed result follows. □

By using the same arguments, the following result is straightforward.

LEMMA 3.4. *If $h \in L^2(I, \mathcal{C}^{m\times m})$ is a Hermitian-valued function and*

$$\sum_{j=1}^{m} m\{x \in I : \ \lambda_j(h(x)) = a\} + \sum_{j=1}^{m} m\{x \in I : \ \lambda_j(h(x)) = b\} = 0,$$

then

$$\lim_{n\to\infty} \frac{\#\{i : \ \lambda_i(A_{n,m}(h)) \in (a,b)\}}{n} = \frac{1}{2\pi} \sum_{j=1}^{m} m\{x \in I : \ \lambda_j(h(x)) \in (a,b)\}.$$

When the assumption of the latter two results is not verified, i.e., when $\sum_{j=1}^{m} m\{x \in I : \ \lambda_j(h(x)) = a\} > 0$, it is interesting to remark that there exist infinitely many values a_k converging to a so that $\sum_{j=1}^{m} m\{x \in I : \ \lambda_j(h(x)) = a_k\} = 0$.

This fact is implied by the following lemma.

Lemma 3.5. *If $h \in L^2(I, \mathcal{C}^{m\times m})$ and $\sum_{j=1}^{m} m\{x \in I : \ \lambda_j(h(x)) = a\} > 0$, then, $\forall \delta > 0$, there exist two values a^+ and a^- so that*

$$a - \delta \leq a^- < a < a^+ \leq a + \delta$$

and

$$\sum_{j=1}^{m} m\{x \in I : \ \lambda_j(h(x)) = a^- \quad \text{or} \quad a^+\} = 0.$$

Proof. For notational simplicity, we prove the result under the assumption that $m = 1$. The general case follows from this. By contradiction, if the claimed thesis is not true, it means that there exist an uncountable number of points y belonging to $[a - \delta, a + \delta]$ such that $m\{x \in I : h(x) = y\} > 0$. Call W the set of these points and $B_y = \{x \in I : m\{x : h(x) = y\} > 0\}$. Then we have

- $B_y \subset I$,
- $B_y \cap B_w = \emptyset$ for $y \neq w$, $y, w \in W$,
- B_y is measurable.

From standard functional analysis techniques (see [24, pp. 88–89]), due to the uncountability of the set W, it follows that $m\{\bigcup_{y\in W} B_y\} = \sum_{y\in W} m\{B_y\} = \infty$ while $\bigcup_{y\in W} B_y$ is a subset of the interval I whose Lebesgue measure is 2π. □

Theorem 3.6. *Let $\{A_{n,m}(g)\}_{n,m}$ and $\{A_{n,m}(f)\}_{n,m}$ be two sequences of Toeplitz matrices generated by two square integrable Hermitian matrix-valued functions f and g, where g is essentially nonnegative definite, and such that the Lebesgue measure $m\{\cdot\}$ of the set $\{x \in [-\pi, \pi] : \lambda_1(g(x)) = 0\}$ is zero (λ_1 is understood to be the minimal eigenvalue of g). By taking into account the essential bounds r and R as in Theorem 2.3 and calling S the union of all the spectra of $\mathcal{P}_{n,m}(f; g)$, we obtain that $\bar{S}$ contains $\bigcup_{i\leq m} \mathcal{ER}(\lambda_i(g^{-1}f))$ and is contained in $[r, R]$.*

Proof. The inclusion $\bar{S} \subset [r, R]$ is a consequence of the localization Theorem 2.3. The nontrivial part is to prove that the topological closure $\bar{S}$ of S contains $\bigcup_{i\leq m} \mathcal{ER}(\lambda_i(g^{-1}f))$. Hereby we demonstrate this general result under the only hypothesis that g is essentially nonnegative and $m\{x \in I : \lambda_1(g(x)) = 0\} = 0$. Actually, the claimed thesis is equivalent to the following statement: $\forall \alpha \in \bigcup_{i\leq m} \mathcal{ER}(\lambda_i(g^{-1}f))$ and $\forall \epsilon > 0$,

$$\exists n \in \mathbf{N} \quad \text{and} \quad \lambda_\alpha \in S \quad \text{such that} \quad |\lambda_\alpha - \alpha| < \epsilon.$$

To prove the latter, for any n, we call I_{nm} the $nm \times nm$ identity matrix and we introduce some auxiliary matrices having the same inertia of the matrix $\mathcal{P}_{n,m}(f; g) - \alpha I_{nm}$, that is, having the same number of negative, positive, and zero eigenvalues as $\mathcal{P}_{n,m}(f; g) - \alpha I_{nm}$. The matrix $\mathcal{P}_{n,m}(f; g) - \alpha I_{nm}$ is similar to the Hermitian matrix $Z_{n,\alpha} = A_{n,m}^{-1/2}(g) A_{n,m}(f) A_{n,m}^{-1/2}(g) - \alpha I_{nm}$ since

$$Z_{n,\alpha} = A_{n,m}^{1/2}(g) \left(\mathcal{P}_{n,m}(f; g) - \alpha I_{nm}\right) A_{n,m}^{-1/2}(g).$$

Therefore $Z_{n,\alpha}$ and $\mathcal{P}_{n,m}(f;g) - \alpha I_{nm}$ have the same eigenvalues. Now, let $H_{n,\alpha} = A_{n,m}(f) - \alpha A_{n,m}(g)$. This matrix is obtained by $Z_{n,\alpha}$ in the following way:

$$H_{n,\alpha} = A_{n,m}^{1/2}(g) Z_{n,\alpha} A_{n,m}^{1/2}(g).$$

Owing to the positive definiteness of $A_{n,m}^{1/2}(g)$, we can conclude that the matrices $\mathcal{P}_{n,m}(f;g) - \alpha I_{nm}$, $Z_{n,\alpha}$, and $H_{n,\alpha}$ have the same inertia: therefore, if $H_{n,\alpha}$ is singular for some value n then there exists $\lambda_\alpha \in S$ such that $\lambda_\alpha = \alpha$ and there is nothing further to be proved.

Otherwise, $H_{n,\alpha}$ is nonsingular for any positive integer n. $H_{n,\alpha}$ is a block Toeplitz matrix generated by $c_\alpha(x) = f(x) - \alpha g(x)$. On the other hand, the assumption that $\alpha \in \bigcup_{i \le m} \mathcal{ER}(\lambda_i(g^{-1}f))$ implies the existence of at least an index $\bar{j}$ such that $\alpha \in \mathcal{ER}(\lambda_{\bar{j}}(g^{-1}f))$ and, therefore, it is understood that the matrix $g^{-1}f - \alpha I_m$ is *essentially singular*, in the sense that $0 \in \mathcal{ER}(\lambda_{\bar{j}}(g^{-1}f) - \alpha)$ or, equivalently, for any $\delta > 0$, the measure of $\{x \in I : \lambda_{\bar{j}}(g^{-1}f) - \alpha \in (-\delta, \delta)\}$ is positive.

Now, let us set

$$m_\epsilon^\alpha(i) = m\{x \in I : \lambda_i(g^{-1}f) - (\alpha + \epsilon) < 0\}$$

and

$$m_{-\epsilon}^\alpha(i) = m\{x \in I : \lambda_i(g^{-1}f) - (\alpha - \epsilon) < 0\}.$$

Since $0 \in \mathcal{ER}(\lambda_{\bar{j}}(g^{-1}f) - \alpha)$ and g is positive definite almost everywhere, it naturally follows that

$$m_\epsilon^\alpha(\bar{j}) > m_{-\epsilon}^\alpha(\bar{j})$$

and, for any $i \neq \bar{j}$, we have

$$m_\epsilon^\alpha(i) \geq m_{-\epsilon}^\alpha(i).$$

Moreover, in the light of the Sylvester inertia law [17], we observe that the matrices $g^{-1}f - (\alpha \pm \epsilon)I_m$, $g^{-1/2}fg^{-1/2} - (\alpha \pm \epsilon)I_m$ and $f - (\alpha \pm \epsilon)g$ have the same inertia: therefore, it follows that

$$m_\epsilon^\alpha(i) = m\{x \in I : \lambda_i(f - (\alpha + \epsilon)g) < 0\}$$

and

$$m_{-\epsilon}^\alpha(i) = m\{x \in I : \lambda_i(f - (\alpha - \epsilon)g) < 0\}.$$

The latter two relationships are the crucial ones in order to apply Lemma 3.4 with $h = c_{\alpha-\epsilon} = f - (\alpha - \epsilon)g$ and $h = c_{\alpha+\epsilon} = f - (\alpha + \epsilon)g$.

Therefore, we assume that

$$\sum_{i=1}^{m} m\{x \in I : \lambda_i(f - (\alpha + \epsilon)g) = 0\} + m\{x \in I : \lambda_i(f - (\alpha - \epsilon)g) = 0\} = 0,$$

and we invoke Lemma 3.4. Consequently we find that

$$\begin{aligned} &\#\{i : \lambda_i(A_{n,m}(c_{\alpha+\epsilon})) < 0\} = n\frac{\sum_{i=1}^m m_\epsilon^\alpha(i)}{2\pi} + o(n), \\ &\#\{i : \lambda_i(A_{n,m}(c_{\alpha-\epsilon})) < 0\} = n\frac{\sum_{i=1}^m m_{-\epsilon}^\alpha(i)}{2\pi} + o(n). \end{aligned} \tag{6}$$

By virtue of the relation $m_\epsilon^\alpha(\bar{j}) > m_{-\epsilon}^\alpha(\bar{j})$ it follows that, for n large enough, "many" eigenvalues of $A_{n,m}(c_z)$ move from positive values to negative values when the parameter z moves from $\alpha-\epsilon$ to $\alpha+\epsilon$. As a consequence, using a continuity argument, we have found $\lambda_\alpha(n) \in (\alpha-\epsilon, \alpha+\epsilon)$ such that the matrix $A_{n,m}(c_{\lambda_\alpha(n)}(x))$ is singular. But $A_{n,m}(c_{\lambda_\alpha(n)}(x))$ and $\mathcal{P}_{n,m}(f;g) - \alpha I_{nm}$ have the same inertia, i.e., $\lambda_\alpha(n)$ is a value for which $\mathcal{P}_{n,m}(f;g) - \alpha I_{nm}$ is singular. Finally, the latter is equivalent to write that $\lambda_\alpha(n)$ is an eigenvalue of the preconditioned matrix $\mathcal{P}_{n,m}(f;g)$, namely, $\lambda_\alpha(n) \in S$. Therefore the theorem is proved.

Notice that in equations (6) we assumed that $\sum_{i=1}^m m\{x \in I : \lambda_i(f-(\alpha+\epsilon))g = 0\} + m\{x \in I : \lambda_i(f-(\alpha-\epsilon)g) = 0\} = 0$. In the case where this assumption is not verified, by virtue of Lemma 3.5, we can choose s, $0 < s < \epsilon$, such that

$$\sum_{i=1}^m m\{x \in I : \lambda_i(f-(\alpha+s)g) = 0\} + m\{x \in I : \lambda_i(f-(\alpha-s)g) = 0\} = 0.$$

Therefore, by repeating the same proof with s in place of ϵ, we find a value of S belonging to the set $(\alpha-s, \alpha+s) \subset (\alpha-\epsilon, \alpha+\epsilon)$. The claimed thesis still holds. □

Observe that the previous result is a generalization of the density theorems proved for the scalar case [14, 27, 26], under the weaker assumption that f, g belong to L^1. This fact is a consequence of the Szegö formula, originally stated in L^∞, but recently extended to the L^1 case (see [37]).

However, it should be stressed that if an "ergodic" theorem like Theorem 1.1 is to be proved for functions ranging in L^1, then an extension of Theorem 3.6 with f and g in L^1 may be naturally obtained by repeating the same proof almost word for word. Finally we point out that the proof of Theorem 3.6 has been used very recently to find a more general result: in fact, in [34], a Szegö formula like the one displayed in (3) has been proved for the eigenvalues of the class of the preconditioned matrices $\{\mathcal{P}_{n,m}(f;g)\}_n$.

3.1. The asymptotic condition numbers of $A_{n,m}(f)$. First we define a relation of "asymptotical equivalence" between two nonnegative sequences (Definition 3.7) and between two nonnegative integrable functions (Definition 3.8). This allows us to state an equivalence theorem which furnishes a tool to evaluate the asymptotic behavior of the extreme eigenvalues of the block Toeplitz matrices considered here.

DEFINITION 3.7. *Let a_n and b_n be two nonnegative sequences; we say that a_n is asymptotical equivalent to b_n and we write $a_n \sim b_n$ if there exist two positive constant values c, C and an integer $\bar{n}$ such that*

$$ca_n \le b_n \le Ca_n \quad \forall n \ge \bar{n}.$$

DEFINITION 3.8. *Let $f(x)$ and $g(x)$ be two nonnegative definite integrable matrix-valued functions defined on I; we say that $f(x)$ is asymptotical equivalent to $g(x)$ and we write $f \sim g$ if there exist two positive constant values c and C such that $g^{-1}f$ has eigenvalues in $[c, C]$ almost everywhere on I.*

The following result holds true.

THEOREM 3.9. *Let $\mathcal{R}_f$ be the class of equivalence of all the L^1 functions g such that $g \sim f$ in the sense previously given. If g_1 and g_2 belong to $\mathcal{R}_f$ and if $\lambda_{\min}(i;n)$, $\lambda_{\max}(i;n)$, $k(i;n)$ are the minimal and the maximal eigenvalues and the Euclidean* *condition number of* $A_{n,m}(g_i)$*, then*

$$\lambda_{\min}(1;n) \sim \lambda_{\min}(2;n), \quad \lambda_{\max}(1;n) \sim \lambda_{\max}(2;n),$$

and therefore

$$k(1;n) \sim k(2;n).$$

Proof. By the hypothesis $g_1 \sim g_2$ and Theorem 3.1, we find that the quadratic form related to $g^{-1/2} f g^{-1/2}$ is well conditioned in the sense that there exist two positive constants r and R such that its numerical range is contained in (r, R). More precisely, given a nonzero vector $\mathbf{z} \in C^m$, we find

$$r < \frac{\mathbf{z}^H A_{n,m}(g_1)\mathbf{z}}{\mathbf{z}^H A_{n,m}(g_2)\mathbf{z}} < R. \tag{7}$$

Let $\mathbf{z}_i$ be the eigenvector of $A_{n,m}(g_i)$ related to the smallest eigenvalue $\lambda_{\min}(i;n)$, $i = 1, 2$. Then, by applying equation (7) with $\mathbf{z} = \mathbf{z}_2$ and with $\mathbf{z} = \mathbf{z}_1$, we find

$$\lambda_{\min}(2;n) > \frac{\mathbf{z}_2^H A_{n,m}(g_1)\mathbf{z}_2}{R} \geq \frac{\lambda_{\min}(1;n)}{R}$$

and

$$r\lambda_{\min}(2;n) \leq r\mathbf{z}_1^H A_{n,m}(g_2)\mathbf{z}_1 < \lambda_{\min}(1;n),$$

i.e., $\lambda_{\min}(1;n) \sim \lambda_{\min}(2;n)$. For the greatest eigenvalue the argument is very similar, while the statement for the Euclidean condition number is a simple consequence. □

For a practical application of this theorem, let us consider $f(x) \in C^{m \times m}$ having the eigenvalues $\lambda_1(f(x)) \sim |x - x_0|^2$ and $A \geq \lambda_j(f(x)) \geq a > 0$, $j = 2, \ldots, m$ a.e. in I. Therefore we have

$$Q(x)^H \mathrm{diag}(\lambda_1(f(x)), \ldots, \lambda_m(f(x)))Q(x)$$

with $Q(x)$ orthogonal matrix. Let us suppose, for simplicity, that $Q(x)$ and $\lambda_i(f(x))$ are continuous functions in $x = x_0$. Then, by choosing

$$g(x) = Q(x_0)^H \mathrm{diag}(|x - x_0|^2, 1, \ldots, 1)Q(x_0),$$

$\forall \mathbf{z} \in \mathbf{C}^2$ we have $\lim_{x \to x_0} \frac{\mathbf{z}^H f(x)\mathbf{z}}{\mathbf{z}^H g(x)\mathbf{z}} = \delta > 0$, while for any $\epsilon > 0$, for any $x \in I/B(x_0, \epsilon)$, the quadratic form associated with $g(x)$ is well separated from zero. Therefore we deduce that there exist two positive constants r and R for which, for any $\mathbf{z}$ and any $x \in I$,

$$r \leq \lambda(g^{-1} f) \leq R,$$

which means that $f \sim g$ ($g \in \mathcal{R}_f$). Now the study of the asymptotic behavior of the extreme eigenvalues of $A_{n,m}(f)$ is easily reduced to the same study regarding $A_{n,m}(g)$. Since $Q(x_0)$ is a constant $m \times m$ matrix, by direct calculation of the Fourier coefficients of g, we deduce that there exists a permutation matrix P for which

$$A_{n,m}(g) = I_n \otimes Q(x_0)^H P^T \mathrm{diag}(A_n(|x - x_0|^2), I_{n(m-1)})PI_n \otimes Q(x_0).$$

Owing to the fact that the condition number of the scalar Toeplitz matrix $A_n(|x-x_0|^2)$ behaves like n^2 [6, 31], we directly deduce the same thing for $A_{n,m}(f)$.

4. The preconditioning strategies. Concerning the preconditioning strategies, it seems quite natural to follow the same ideas as those for the scalar case. When f is positive definite, a simple block band Toeplitz (BBT) preconditioner is easily defined by $g(x) = \sum_{k=-q}^{q} A_k e^{\mathbf{i}kx}$: the cost of the solution of a system with coefficient matrix $A_{n,m}(g)$ is asymptotic to $O(nq^2m^3)$ by using a generalized band solver [17] or $O(nqm^3)$ by adapting the algorithm proposed in [15] to this kind of block Toeplitz matrix with $m \times m$ unstructured blocks. The matrix algebra approach (for instance, based on circulant τ, Hartley [9, 11, 20, 38, 4, 32, 5]) assures a good clustering (only $O(m)$ outliers) with an asymptotic cost per iteration of $O([n \log n]m^3)$ ops.

On the other hand, in the nonnegative case, the clustering properties of the matrix algebra preconditioned matrices are not so satisfactory. By following an argument such as that in [35], it follows that the clusters are weak, while the BBT preconditioning (when the preconditioner is easy to construct) is better since, in many cases, the optimality of the resulting PCG method is preserved.

4.1. Distribution and clustering. From the density result stated in Theorem 3.6, it follows that the spectrum of $\mathcal{P}_{n,m}(f;g)$ is asymptotically "distributed" in $E = \bigcup_{i \le m} \mathcal{ER}(\lambda_i(g^{-1}f))$ and is strictly contained in its convex hull. However, if $E = [a,b]$, $0 < a < b < \infty$, then the spectral condition number of the preconditioned matrix $\mathcal{P}_{n,m}(f;g)$ that is bounded by b/a describes very precisely the convergence rate of the PCG method owing to the distribution of the eigenvalues—a distribution which *does not allow the presence of a cluster neither proper nor general.* We stress that this situation is not academic but frequently occurs since, in general, the set E can be looked at as a nontrivial closed interval or as the union of a finite collection of closed nontrivial intervals.

As a consequence, if we want to devise PCG methods showing superlinear convergence, or, in other words, if we want to obtain a strong cluster around the unity, then we have to slightly modify the definition of the preconditioner by allowing the degree q to become dependent on n. In effect, it is sufficient to extend the preconditioning technique developed in [28, 32] for the scalar case. More precisely, the generating function of the preconditioner is defined as $g_n(x) = \sum_{k=-q}^{q} B_k e^{\mathbf{i}kx}$, $q = q(n)$, where

1. $\lim_{n\to\infty} q(n) = \infty$, g_n is positive definite a.e.
2. The cost of the solution of a system $A_{n,m}(g_n)\mathbf{y} = \mathbf{c}$ must be bounded by $O(m^3 n \log n)$ ops (that is, by the asymptotic cost of a product between $A_{n,m}(f)$ by a generic vector when a block fast Fourier transform is used [40]).
3. $\lim_{n\to\infty} \|g_n^{-\frac{1}{2}} f g_n^{-\frac{1}{2}} - I_m\|_2 = 0$.

To meet requirement (2), we can choose $q(n) = \sqrt{\log n}$ if we use a classic band solver [17] or $q(n) = \log n$ if we use a specialized multigrid method like the one described in [15, 16]. The requirement (3) is a delicate task when the minimal eigenvalue of $f(x)$ has essential zeros. In the strictly positive definite case, the Fourier choice given by $g_n(x) = \sum_{k=-q(n)}^{q(n)} A_k e^{\mathbf{i}kx}$, i.e., $B_k = A_k$, can be a suitable one when the function f is regular enough.

4.2. A numerical experiment. In this paragraph we perform some numerical tests in order to make evident the effectiveness of the theoretical scheme. Let us consider the following generating function (with $m = 2$):

$$f(x) = \left[\begin{array}{cc} 20\sin^2(x/2) & |x|^{5/2} \\ |x|^{5/2} & 20\sin^2(x/2) \end{array} \right].$$

We may prove that $0 = m_f < \hat{m}_f$, $\hat{M}_f < M_f = 20 + \pi^{\frac{5}{2}}$, and the eigenvalues, for x in a neighborhood of 0, are both asymptotical to $|x|^2$. Therefore $A_{n,2}(f)$ is positive definite and ill conditioned and, in light of Theorem 3.1, we choose

$$g(x) = \left[\begin{array}{cc} 20\sin^2(x/2) & 0 \\ 0 & 20\sin^2(x/2) \end{array} \right].$$

It is easy to verify that, for any $\mathbf{z}$ and any $x \in I$, the Rayleigh quotient related to f and g is contained in a positive bounded interval $[r, R]$, where $r = 1 - \pi^{\frac{5}{2}}/20$ and $R = 1 + \pi^{\frac{5}{2}}/20$. Then the eigenvalues of the preconditioned matrix $A_{n,2}^{-1}(g)A_{n,2}(f)$ are in the open set (r, R). Therefore, we expect that the number of iterations N_{it} in order to reach the solution of a linear system $A_{n,2}(f)\mathbf{x} = \mathbf{b}$, by using the PCG method with $A_{n,2}(f)$ as preconditioner, is constant with respect to n. In addition, since both the eigenvalues have a zero of order two and are positive for $x \neq 0$ we find that the condition number of $A_{n,2}(f)$, up to a positive constant (see Theorem 3.9), grows like n^2. Finally, in light of the powerful results of Axelsson and Lindskog [1], we expect that the conjugate gradient method, when applied with no preconditioning, requires $O(n)$ iterations in order to find the solution within a preassigned accuracy. All of these theoretical foresights are fully confirmed by the following numerical computations:

- For $n = 64$, $m = 2$ (dimension equal to 128), we have

$$k_2(A_{n,2}(f)) = 3395.5, \quad k_2(A_{n,2}^{-1}(g)A_{n,2}(f)) = 12.4.$$

- For $n = 256$, $m = 2$ (dimension equal to 512), we have

$$k_2(A_{n,2}(f)) = 51745.6, \quad k_2(A_{n,2}^{-1}(g)A_{n,2}(f)) = 14.2.$$

It is evident that the condition number of the preconditioned matrix is practically constant while the condition number of the original coefficient matrix grows like n^2 and, in fact, the ratio

$$\frac{k_2(A_{4*n,2}(f))}{k_2(A_{n,2}(f))}$$

for $n = 64$ is equal to 15.2, which is close, as predicted, to $4^2 = 16$. We point out that the preconditioning matrix is a block tridiagonal matrix whose related systems can be very efficiently solved by using classical band solvers [17] or multigrid schemes [15, 16]. Finally, Table 1 illustrates the optimality of the proposed PCG method in the sense of the invariance of the number of iterations with regard to the dimension n. On the other hand, for the nonpreconditioned system, as the quantity N_{it} behaves asymptotically as in $\sqrt{k_2(A_{n,2}(f))}$, we expect the related number of iterations to grow as in $\sqrt{k_2(A_{n,2}(f))} \sim cn$. This expectation also is fully confirmed in the subsequent table, where we report the number of iterations N_{it} to reach the solution within an accuracy $\epsilon = 10^{-7}$ with the zero vector as initial guess and the vector of all ones as data vector $\mathbf{b}$.

TABLE 1

Dim	Nonpreconditioned system	Preconditioned system
128	33	10
256	72	10
512	150	10

TABLE 2

Dim	Nonpreconditioned system	Preconditioned system
128	154	26
256	324	29
512	666	30

In light of Theorem 3.6, since the two eigenvalue functions of $g^{-1}f$ are continuous and cover the range $[r, R]$, we know that the eigenvalues of $\mathcal{P}_{n,2}(f; g)$ are distributed "uniformly" as per a sampling of $\lambda_1(g^{-1}f)$ and $\lambda_2(g^{-1}f)$ on a uniform mesh of $[0, \pi]$. Since the estimates (4) and (5) are derived [1] by minimizing the Chebyshev norm of a class of polynomials on the *continuous set* $[r, R]$, it is natural to think that in our case, wherein the eigenvalues are well distributed on $[r, R]$, these estimates are tight.

By calculating $N1$ and $N2$, we found 33.5 and 31.7, respectively.

Surprisingly enough, the actual number of iterations is, for all the considered dimensions, equal to 10, which is far from $N1$ and $N2$. However, the previous numerical experiment was done with a special choice of the data vector **b**. Actually, this vector has a "nongeneric" decomposition with respect to the frequency basis (eigenvectors of the circulant class [13] or eigenvectors of the τ algebra [3]), because it has strong components in low frequency and negligible components in high frequency.

Therefore, we considered as **b** a random vector which has substantial components in high frequency, and the results are in Table 2.

The observed number of iterations tends to the value indicated by the finer estimate $N2$, so this is also an unrigorous confirmation of the tightness of the Axelsson and Lindskog bounds when the eigenvalues are uniformly distributed.

Acknowledgments. I am thankful to Professor Buttazzo for illuminating conversations. Last but not least, this paper is dedicated to the mathematicians of my family: Giovanni, Ottavio, and Wanda.

REFERENCES

[1] O. AXELSSON AND G. LINDSKOG, *The rate of convergence of the preconditioned conjugate gradient method*, Numer. Math., 52 (1986), pp. 499–523.

[2] O. AXELSSON AND M. NEYTCHEVA, *The algebraic multilevel iteration methods—Theory and applications*, Proc. 2nd Int. Coll. on Numerical Analysis, D. Bainov, ed., Plovdiv, Bulgaria, August 1993, pp. 13–23.

[3] D. BINI AND M. CAPOVANI, *Spectral and computational properties of band symmetric Toeplitz matrices*, Linear Algebra Appl., 52/53 (1983), pp. 99–126.

[4] D. BINI AND F. DI BENEDETTO, *A new preconditioner for the parallel solution of positive definite Toeplitz linear systems*, Proc. 2nd SPAA, Crete, Greece, July 1990, pp. 220–223.

[5] D. BINI AND P. FAVATI, *On a matrix algebra related to the discrete Hartley transform*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 500–507.

[6] R. H. CHAN, *Toeplitz preconditioners for Toeplitz systems with nonnegative generating functions*, IMA J. Numer. Anal., 11 (1991), pp. 333–345.

[7] R. H. Chan and W. Ching, *Toeplitz–circulant preconditioners for Toeplitz systems and their applications to queueing network with batch arrivals*, SIAM J. Sci. Comput., 17 (1996), pp. 762–772.

[8] R. H. Chan and M. Ng, *Conjugate gradient method for Toeplitz systems*, SIAM Rev., 38 (1996), pp. 427–482.

[9] R. H. Chan and G. Strang, *Toeplitz equations by conjugate gradients with circulant preconditioner*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 104–119.

[10] R. H. Chan and P. Tang, *Fast band–Toeplitz preconditioners for Hermitian Toeplitz systems*, SIAM J. Sci. Comput., 15 (1994), pp. 164–171.

[11] T. F. Chan, *An optimal circulant preconditioner for Toeplitz systems*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 766–771.

[12] E. Cinlar, *Introduction to Stochastic Processes*, Prentice–Hall, Englewood Cliffs, NJ, 1975.

[13] P. Davis, *Circulant Matrices*, John Wiley and Sons, New York, 1979.

[14] F. Di Benedetto, G. Fiorentino, and S. Serra, *C.G. Preconditioning for Toeplitz Matrices*, Comput. Math. Appl., 25 (1993), pp. 35–45.

[15] G. Fiorentino and S. Serra, *Multigrid methods for Toeplitz matrices*, Calcolo, 28 (1991), pp. 283–305.

[16] G. Fiorentino and S. Serra, *Multigrid methods for symmetric positive definite block Toeplitz matrices with nonnegative generating functions*, SIAM J. Sci. Comput., 17 (1996), pp. 1068–1081.

[17] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, MD, 1983.

[18] A. Greenbaum, *Comparison of splittings used with the conjugate gradient algorithm*, Numer. Math., 33 (1979), pp. 181–194.

[19] U. Grenander and G. Szegö, *Toeplitz Forms and Their Applications*, 2nd Edition, Chelsea, New York, 1984.

[20] T. Huckle, *Circulant and skewcirculant matrices for solving Toeplitz matrix problems*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 767–777.

[21] X. Jin, *Fast iterative solvers for symmetric Toeplitz systems - A survey and an extension*, J. Comput. Appl. Math., 66 (1996), pp. 315–321.

[22] M. Neuts, *Structured Stochastic Matrices of M/G/1 Type and Their Applications*, Marcel Dekker, New York, 1989.

[23] R. Preuss, *Toeplitz matrices and control theory*, in Workshop on Toeplitz Matrices in Filtering and Control, Santa Barbara, CA, August 1996.

[24] W. Rudin, *Real and Complex Analysis*, McGraw–Hill, NY, 1985.

[25] S. Serra, *Preconditioning strategies for asymptotically ill–conditioned block Toeplitz systems*, BIT, 34 (1994), pp. 579–594.

[26] S. Serra, *Conditioning and solution of Hermitian (block) Toeplitz systems by means of preconditioned conjugate gradient methods*, in Proc. Advanced Signal Processing Algorithms, Architectures, and Implementations—SPIE conference, F. Luk, ed., San Diego, CA, July 1995, pp. 326–337.

[27] S. Serra, *Preconditioning strategies for Hermitian Toeplitz systems with nondefinite generating functions*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 1007–1019.

[28] S. Serra, *Optimal, quasi-optimal and superlinear preconditioners for asymptotically ill-conditioned positive definite Toeplitz matrices*, Math. Comp., 66 (1997), pp. 651–665.

[29] S. Serra, *New PCG based methods for Hermitian Toeplitz systems*, Calcolo, 32 (1995), pp. 153–176.

[30] S. Serra, *The extension of the concept of generating function to a class of preconditioned Toeplitz matrices*, Linear Algebra Appl., 267 (1997), pp. 139–161.

[31] S. Serra, *On the extreme eigenvalues of Hermitian (block) Toeplitz matrices*, Linear Algebra Appl., 270 (1998), pp. 109–129.

[32] S. Serra, *Superlinear PCG Methods for Symmetric Toeplitz Systems*, Math. Comp., to appear.

[33] S. Serra, *A Note on the Preconditioning for Block Toeplitz Systems having Nonnegative Definite Matrix-Valued Generating Functions*, Technical report 342, Dept. of Mathematics, Univ. of Genova, 1997.

[34] S. Serra, *Spectral and Computational Analysis of Preconditioned Block Toeplitz Matrices having Nonegative Definite Matrix-Valued Generating Functions*, manuscript.

[35] V. Strela and E. E. Tyrtyshnikov, *Which circulant preconditioner is better?*, Math. Comp., 65 (1996), pp. 137–150.

[36] P. Tilli and M. Miranda, *Asymptotical Spectrum of Hermitian Block Toeplitz Matrices and Preconditioning Results*, SIAM J. Matrix Anal. Appl., to appear.

[37] E. E. Tyrtyshnikov and N. L. Zamarashkin, *Spectra of multilevel Toeplitz matrices: Ad-*

vanced theory via simple matrix relationships, Linear Algebra Appl., 270 (1998), pp. 15–27.

[38] E. E. Tyrtyshnikov, *Optimal and superoptimal circulant preconditioners*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 459–473.

[39] E. E. Tyrtyshnikov, *A unifying approach to some old and new theorems on distribution and clustering*, Linear Algebra Appl., 232 (1996), pp. 1–43.

[40] C. Van Loan, *Computational Frameworks for the Fast Fourier Transform*, SIAM, Philadelphia, 1992.

[41] H. Widom, *Toeplitz matrices*, In Studies in Real and Complex Analysis, I. Hirshman Jr., Math. Assoc. Amer., 1965.

SIAM J. MATRIX ANAL. APPL.
Vol. 20, No. 1, pp. 45–59

BOUNDS FOR THE STRUCTURED BACKWARD ERRORS OF VANDERMONDE SYSTEMS [*]

JI-GUANG SUN[†]

Abstract. Consider the primal Vandermonde system $V(a)x = b$ and the dual Vandermonde system $V(a)^T x = b$, where $V(a)$ is the Vandermonde matrix defined in terms of the vector $a = (\alpha_1, \ldots, \alpha_n)^T$ with distinct scalars $\alpha_1, \ldots, \alpha_n \in \mathcal{C}$. In view of the special structure of the matrix $V(a)$, we define structured backward errors (SBEs) of the Vandermonde systems and describe a technique for obtaining upper and lower bounds for the SBEs. The results are illustrated by numerical examples.

Key words. stability, Vandermonde systems, backward perturbations

AMS subject classifications. 15A06, 15A60, 65F05, 65F35

PII. S0895479897314759

1. Introduction.

Let $\tilde{x}$ be an approximate solution to the linear system $Ax = b$. In general, there are many perturbations ΔA and Δb such that $\tilde{x}$ is the solution to the perturbed systems $(A + \Delta A)x = b + \Delta b$. It may well be asked, how close is the nearest system for which $\tilde{x}$ is the solution? It is known [5], [7], [8], [9] that there are various approaches to define backward errors (BEs) for measuring the distance between the perturbed systems and the original system, such as the BE $\eta(\tilde{x})$ defined by

$$\eta(\tilde{x}) = \min\left\{ \left\| \left(\frac{\Delta A}{\|A\|_F}, \frac{\Delta b}{\|b\|_2} \right) \right\|_F \,:\, (A + \Delta A)\tilde{x} = b + \Delta b \right\},$$

where $\|\cdot\|_F$ denotes the Frobenius norm and $\|\cdot\|_2$ defines the Euclidean vector norm. Note that a BE also can be called an optimal backward perturbation bound [9].

An algorithm for computing the solution x to the system $Ax = b$ is called backward stable if, for any A and b, it produces a computed $\tilde{x}$ with a small BE (see, e.g., [3], [7]). Consequently, to find an explicit expression of a BE may be very useful for testing the stability of practical algorithms. It has been proved [9, Remark 3.5] that the BE $\eta^{(\theta)}(\tilde{x})$ defined by

$$\eta^{(\theta)}(\tilde{x}) = \min\{\|(\Delta A, \theta\Delta b)\|_F \,:\, (A + \Delta A)\tilde{x} = b + \Delta b\} \tag{1.1}$$

has the expression

$$\eta^{(\theta)}(\tilde{x}) = \frac{\theta\|\tilde{x}\|_2}{\sqrt{1 + \theta^2\|\tilde{x}\|_2^2}} \cdot \frac{\|\hat{r}\|_2}{\|\tilde{x}\|_2} \quad \text{with} \quad \hat{r} = b - A\tilde{x}. \tag{1.2}$$

Taking $\theta = \|A\|_F/\|b\|_2 \equiv \theta_A$ in (1.1) and (1.2), from $\eta(\tilde{x}) = \eta^{(\theta_A)}(\tilde{x})/\|A\|_F$ we get an expression of $\eta(\tilde{x})$.

[*] Received by the editors January 2, 1997; accepted for publication (in revised form) by N. J. Higham September 19, 1997; published electronically September 15, 1998.
http://www.siam.org/journals/simax/20-1/31475.html

[†] Department of Computing Science, Umeå University, S-901 87 Umeå, Sweden (jisun@cs.umu.se).
The research of this author was supported by Swedish Natural Science Research Council contract M-AA/MA 06952-303 and by the Department of Computing Science, Umeå University.

In this paper, we consider the following Vandermonde systems:

$$\text{Primal}: \quad V(a)x = b; \tag{1.3}$$

and

$$\text{Dual}: \quad V(a)^T x = b, \tag{1.4}$$

where $V(a) \in \mathcal{C}^{n\times n}$ with $a = (\alpha_1, \ldots, \alpha_n)^T \in \mathcal{C}^n$ is the Vandermonde matrix defined by

$$V(a) = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \vdots & \vdots & & \vdots \\ \alpha_1^{n-1} & \alpha_2^{n-1} & \cdots & \alpha_n^{n-1} \end{pmatrix},$$

in which $\alpha_1, \ldots, \alpha_n$ are distinct scalars.

Let $\tilde{x}$ be an approximate solution to (1.3) or (1.4). Since the matrix $V(a)$ has a special structure, it is pertinent to define a BE by using a restricted class of perturbations ΔV that $V(a) + \Delta V$ are Vandermonde matrices too. We now define the SBE $\eta_{\mathrm{P}}^{(\theta)}(\tilde{x})$ of the system (1.3) by

$$\eta_{\mathrm{P}}^{(\theta)}(\tilde{x}) = \min\{\|(V(\tilde{a}) - V(a),\, \theta\Delta b)\|_F \;:\; V(\tilde{a})\tilde{x} = b + \Delta b\} \tag{1.5}$$

and the SBE $\eta_{\mathrm{D}}^{(\theta)}(\tilde{x})$ of the system (1.4) by

$$\eta_{\mathrm{D}}^{(\theta)}(\tilde{x}) = \min\{\|(V(\tilde{a}) - V(a),\, \theta\Delta b)\|_F \;:\; V(\tilde{a})^T\tilde{x} = b + \Delta b\}, \tag{1.6}$$

where the matrices $V(\tilde{a})$ are Vandermonde matrices defined in terms of the vectors $\tilde{a}$ and θ is a positive parameter. The parameter θ allows us some flexibility.

An algorithm for computing the solution x to the structured system (1.3) or (1.4) is called backward stable (or strongly stable [3]) if, for any $V(a)$ and b, it produces a computed $\tilde{x}$ with a small SBE. Consequently, finding explicit expressions or lower and upper bounds for the SBEs $\eta_{\mathrm{P}}^{(\theta)}(\tilde{x})$ and $\eta_{\mathrm{D}}^{(\theta)}(\tilde{x})$ may be useful for testing the stability of practical algorithms.

We start with the simplest case: $n = 2$. Let $\tilde{x}$ be an approximate solution to the system (1.3) or (1.4). For the system (1.3), by (1.5) and $n = 2$ we have

$$\eta_{\mathrm{P}}^{(\theta)}(\tilde{x}) = \min\{\|(\Delta a,\, \theta\Delta b)\|_F \;:\; V(a + \Delta a)\tilde{x} = b + \Delta b\}.$$

Write

$$\Delta a = \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \end{pmatrix}, \quad \Delta b = \begin{pmatrix} \delta_1 \\ \delta_2 \end{pmatrix}, \quad \tilde{x} = \begin{pmatrix} \xi_1 \\ \xi_2 \end{pmatrix}, \quad \hat{r} = b - V(a)\tilde{x} = \begin{pmatrix} \rho_1 \\ \rho_2 \end{pmatrix}.$$

Then

(1.7)

$$\begin{aligned}\eta_{\mathrm{P}}^{(\theta)}(\tilde{x}) &= \min\left\{\|(\epsilon_1,\epsilon_2,\theta\delta_1,\theta\delta_2)\|_2 \;:\; \begin{pmatrix} 0 & 0 \\ \epsilon_1 & \epsilon_2 \end{pmatrix}\begin{pmatrix} \xi_1 \\ \xi_2 \end{pmatrix} = \begin{pmatrix} \rho_1+\delta_1 \\ \rho_2+\delta_2 \end{pmatrix}\right\} \\ &= \left(\theta^2\rho_1^2 + \min_{\epsilon_1,\epsilon_2,\delta_2\in\mathcal{C}}\left\{\left\|\begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \theta\delta_2 \end{pmatrix}\right\|_2^2 \;:\; \left(\xi_1,\xi_2,-\frac{1}{\theta}\right)\begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \theta\delta_2 \end{pmatrix} = \rho_2\right\}\right)^{1/2} \\ &= \left(\theta\rho_1^2 + \left\|\left(\xi_1,\xi_2,-\frac{1}{\theta}\right)^{\dagger}\rho_2\right\|_2^2\right)^{1/2} = \frac{\theta\|\tilde{x}\|_2}{\sqrt{1+\theta^2\|\tilde{x}\|_2^2}}\sqrt{\frac{\|\hat{r}\|_2^2}{\|\tilde{x}\|_2^2}+\theta^2\rho_1^2}.\end{aligned}$$

Similarly, for the system (1.4), by (1.6) and $n=2$ we have

(1.8)

$$\begin{aligned}\eta_{\mathrm{D}}^{(\theta)}(\tilde{x}) &= \min\{\|(\Delta a,\,\theta\Delta b)\|_F \;:\; V(a+\Delta a)^T\tilde{x} = b+\Delta b\} \\ &= \min\left\{\|(\epsilon_1,\epsilon_2,\theta\delta_1,\theta\delta_2)\|_2 \;:\; \left(\xi_2 I_2,\, -\frac{1}{\theta}I_2\right)\begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \theta\delta_1 \\ \theta\delta_2 \end{pmatrix} = \begin{pmatrix} \rho_1 \\ \rho_2 \end{pmatrix}\right\} \\ &= \left\|\left(\xi_2 I_2,\, -\frac{1}{\theta}I_2\right)^{\dagger}\begin{pmatrix} \rho_1 \\ \rho_2 \end{pmatrix}\right\|_2 = \frac{\theta\|\tilde{x}\|_2}{\sqrt{1+\theta^2\xi_2^2}}\cdot\frac{\|\hat{r}_{\mathrm{D}}\|_2}{\|\tilde{x}\|_2},\end{aligned}$$

where $\hat{r}_{\mathrm{D}} = b - V(a)^T\tilde{x} = (\rho_1,\rho_2)^T$. From (1.2) and (1.7)–(1.8) we get

$$\eta^{(\theta)}(\tilde{x}) \le \eta_{\mathrm{P}}^{(\theta)}(\tilde{x}),\ \eta_{\mathrm{D}}^{(\theta)}(\tilde{x}) \le \sqrt{1+\theta^2\|\tilde{x}\|_2^2}\,\eta^{(\theta)}(\tilde{x}),$$

which shows that the SBEs $\eta_{\mathrm{P}}^{(\theta)}(\tilde{x})$ and $\eta_{\mathrm{D}}^{(\theta)}(\tilde{x})$ may be larger than the BE $\eta^{(\theta)}(\tilde{x})$ by a factor of about $\theta\|\tilde{x}\|_2$, and in some cases the scalar $\theta\|\tilde{x}\|_2$ may be extremely large.

Consequently, for the Vandermonde systems, it is necessary to study the SBEs $\eta_{\mathrm{P}}^{(\theta)}(\tilde{x})$ and $\eta_{\mathrm{D}}^{(\theta)}(\tilde{x})$. Note that if $n>2$, then the objective functions and the constraints in (1.5) and (1.6) are no longer linear, as in the symmetric case [9] or in the Toeplitz case [5], [10]; for this reason, the problem of finding explicit expressions of the SBEs $\eta_{\mathrm{P}}^{(\theta)}(\tilde{x})$ and $\eta_{\mathrm{D}}^{(\theta)}(\tilde{x})$ becomes very complicated.

For simplicity, we now linearize the objective functions in (1.5) and (1.6). Let $\tilde{x}$ be an approximate solution to (1.3). Define the linearized SBE $\beta_{\mathrm{P}}^{(\theta)}(\tilde{x})$ by

$$\beta_{\mathrm{P}}^{(\theta)}(\tilde{x}) = \min\{\|(\Delta a,\,\theta\Delta b)\|_F \;:\; V(a+\Delta a)\tilde{x} = b+\Delta b\}. \tag{1.9}$$

Similarly, the linearized SBE $\beta_{\mathrm{D}}^{(\theta)}(\tilde{x})$ for the system (1.4) is defined by

$$\beta_{\mathrm{D}}^{(\theta)}(\tilde{x}) = \min\{\|(\Delta a,\,\theta\Delta b)\|_F \;:\; V(a+\Delta a)^T\tilde{x} = b+\Delta b\}. \tag{1.10}$$

Obviously, the linearized SBE $\beta_{\mathrm{P}}^{(\theta)}(\tilde{x})$ (or $\beta_{\mathrm{D}}^{(\theta)}(\tilde{x})$) and the SBE $\eta_{\mathrm{P}}^{(\theta)}(\tilde{x})$ (or $\eta_{\mathrm{D}}^{(\theta)}(\tilde{x})$) have different meanings. But in the case of $n=2$, the SBEs $\beta_{\mathrm{P}}^{(\theta)}(\tilde{x})$ and $\beta_{\mathrm{D}}^{(\theta)}(\tilde{x})$ coincide with $\eta_{\mathrm{P}}^{(\theta)}(\tilde{x})$ and $\eta_{\mathrm{D}}^{(\theta)}(\tilde{x})$, respectively.

The problem of finding explicit expressions of the linearized SBEs $\beta_{\rm P}^{(\theta)}(\tilde{x})$ and $\beta_{\rm D}^{(\theta)}(\tilde{x})$ is also a difficult one. In this paper, we shall describe a technique for obtaining upper and lower bounds for the SBEs $\beta_{\rm P}^{(\theta)}(\tilde{x})$, $\beta_{\rm D}^{(\theta)}(\tilde{x})$, $\eta_{\rm P}^{(\theta)}(\tilde{x})$, and $\eta_{\rm D}^{(\theta)}(\tilde{x})$. The corresponding bounds will be given in sections 2 through 4, respectively, and the results will be illustrated by numerical examples in section 5.

Bartels and Higham [1] already considered linearized SBEs for Vandermonde systems, including for componentwise measures of the perturbations. In section 5 we shall give a comparison between our results and those of [1].

2. The linearized SBE $\beta_{\rm P}^{(\theta)}(\tilde{x})$.

In this section, we describe a technique for obtaining upper and lower bounds for the SBE $\beta_{\rm P}^{(\theta)}(\tilde{x})$. An upper bound for $\beta_{\rm P}^{(\theta)}(\tilde{x})$ is derived with an appropriate use of the Brouwer fixed-point theorem, and a lower bound for $\beta_{\rm P}^{(\theta)}(\tilde{x})$ is obtained using standard linear algebra technique. The same technique will be used for obtaining upper and lower bounds for the SBEs $\beta_{\rm D}^{(\theta)}(\tilde{x})$, $\eta_{\rm P}^{(\theta)}(\tilde{x})$, and $\eta_{\rm D}^{(\theta)}(\tilde{x})$ in sections 3 through 4, where we shall omit details.

2.1. An equivalent form of the constraint $V(a+\Delta a)\tilde{x} = b + \Delta b$.

Let

$$a = (\alpha_1, \ldots, \alpha_n)^T \in \mathcal{C}^n, \qquad \Delta a = (\epsilon_1, \ldots, \epsilon_n)^T \in \mathcal{C}^n, \tag{2.1}$$
$$\Delta V = V(a+\Delta a) - V(a), \qquad \hat{r} = b - V(a)\tilde{x}.$$

Then the constraint $V(a+\Delta a)\tilde{x} = b + \Delta b$ in (1.9) can be written as

$$\Delta V \tilde{x} = \hat{r} + \Delta b, \tag{2.2}$$

where

$$\Delta V = (\nu_{lj}) \quad \text{with} \quad \nu_{lj} = \begin{cases} 0, & l = 1, \\ \sum_{k=1}^{l-1} \begin{pmatrix} l-1 \\ k \end{pmatrix} \alpha_j^{l-1-k} \epsilon_j^k, & l > 1, \end{cases} \tag{2.3}$$

in which the coefficients $\begin{pmatrix} l-1 \\ k \end{pmatrix} = \frac{(l-1)!}{(l-1-k)!k!}$ for $k = 0, 1, \ldots, l-1$.

Let

(2.4)

$$\Delta b = (\delta_1, \ldots, \delta_n)^T, \quad \Delta b_2 = (\delta_2, \ldots, \delta_n)^T, \quad \hat{r} = (\rho_1, \ldots, \rho_n)^T, \quad \hat{r}_2 = (\rho_2, \ldots, \rho_n)^T,$$
$$\tilde{x} = (\xi_1, \ldots, \xi_n)^T, \quad \Delta a^{(j)} = (\epsilon_1^j, \ldots, \epsilon_n^j)^T, \ \ j = 2, \ldots, n-1,$$

and define the matrices $G_1, \ldots, G_{n-1} \in \mathcal{C}^{(n-1)\times n}$ and $T_{\rm P} \in \mathcal{C}^{(n-1)\times(2n-1)}$ by

(2.5)

$$G_1 = \begin{pmatrix} 1 & & & \\ & 2 & & \\ & & \ddots & \\ & & & n-1 \end{pmatrix} \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \vdots & \vdots & & \vdots \\ \alpha_1^{n-2} & \alpha_2^{n-2} & \cdots & \alpha_n^{n-2} \end{pmatrix} \begin{pmatrix} \xi_1 & & & \\ & \xi_2 & & \\ & & \ddots & \\ & & & \xi_n \end{pmatrix},$$

$$G_j = D_j^{(n-1)} \begin{pmatrix} 0 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 \\ 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \vdots & \vdots & & \vdots \\ \alpha_1^{n-1-j} & \alpha_2^{n-1-j} & \cdots & \alpha_n^{n-1-j} \end{pmatrix} \begin{pmatrix} \xi_1 & & & \\ & \xi_2 & & \\ & & \ddots & \\ & & & \xi_n \end{pmatrix}, \tag{2.6}$$

with the $(n-1) \times (n-1)$ diagonal matrices

$$D_j^{(n-1)} = \operatorname{diag}\left(0, \ldots, 0, \begin{pmatrix} j \\ j \end{pmatrix}, \begin{pmatrix} j+1 \\ j \end{pmatrix}, \ldots, \begin{pmatrix} n-1 \\ j \end{pmatrix}\right), \quad j = 2, \ldots, n-1,$$

and

$$T_{\mathrm{P}} = \left(G_1,\ -\frac{1}{\theta} I_{n-1}\right). \tag{2.7}$$

Then (2.2) is equivalent to

$$\begin{cases} \delta_1 = -\rho_1, \\ T_{\mathrm{P}} \begin{pmatrix} \Delta a \\ \theta \Delta b_2 \end{pmatrix} = \hat{r}_2 - (G_2 \Delta a^{(2)} + \cdots + G_{n-1} \Delta a^{(n-1)}). \end{cases} \tag{2.8}$$

2.2. Upper bounds for $\boldsymbol{\beta_{\mathrm{P}}^{(\theta)}(\tilde{x})}$. Let $t = \theta \Delta b_2$, and consider the nonlinear system

$$\begin{pmatrix} \Delta a \\ t \end{pmatrix} = T_{\mathrm{P}}^{\dagger} \left[\hat{r}_2 - (G_2 \Delta a^{(2)} + \cdots + G_{n-1} \Delta a^{(n-1)})\right], \tag{2.9}$$

where $\Delta a, \Delta b_2, \hat{r}_2, \Delta a^{(2)}, \ldots, \Delta a^{(n-1)}$, $G_2, \ldots, G_{n-1}$, and T_{P} are defined by (2.1) and (2.4)–(2.7). Since the $(n-1) \times (2n-1)$ matrix T_{P} is full rank, we have $T_{\mathrm{P}} T_{\mathrm{P}}^{\dagger} = I_{n-1}$, so multiplying (2.9) on the left by T_{P} yields the second equation of (2.8). This shows that any solution to the equation (2.9) is a solution to the second equation of (2.8). For this reason, if $(\widehat{\Delta a}^T, \hat{t}^T)^T$ is a solution to (2.9), then

$$\beta_{\mathrm{P}}^{(\theta)}(\tilde{x}) \le \sqrt{\theta^2 \rho_1^2 + \|(\widehat{\Delta a}^T,\ \hat{t}^T)\|_2^2}. \tag{2.10}$$

Let $z = (\Delta a^T, t^T)^T \in \mathcal{C}^{2n-1}$, and regard the elements of z as independent variables. Then the function $F(z)$, defined by

$$F(z) = T_{\mathrm{P}}^{\dagger} \left[\hat{r}_2 - (G_2 \Delta a^{(2)} + \cdots + G_{n-1} \Delta a^{(n-1)})\right], \tag{2.11}$$

can be regarded as a continuous mapping $\mathcal{M}_{\mathrm{P}} : \mathcal{C}^{2n-1} \to \mathcal{C}^{2n-1}$, and any fixed point of the mapping $\mathcal{M}_{\mathrm{P}}$ is a solution to the nonlinear system (2.9). Thus, the problem of finding an upper bound for $\beta_{\mathrm{P}}^{(\theta)}(\tilde{x})$ reduces to the problem of showing the existence of a fixed point of the continuous mapping $\mathcal{M}_{\mathrm{P}}$ and determining a bound on its size.

Let

$$\rho_{\mathrm{P}} = \|T_{\mathrm{P}}^{\dagger}\hat{r}_2\|_2, \quad \gamma_j = \|T_{\mathrm{P}}^{\dagger}G_j\|_2, \quad j = 2 : n-1, \quad \gamma = \gamma_2 + \cdots + \gamma_{n-1}. \tag{2.12}$$

It can be verified that if ρ_{P} satisfies

$$\rho_{\mathrm{P}} < 1/(4\gamma), \tag{2.13}$$

then the quadratic equation

$$\gamma\zeta^2 - \zeta + \rho_{\mathrm{P}} = 0 \tag{2.14}$$

has two positive roots, and the smallest positive root ζ_{P} can be expressed by

$$\zeta_{\mathrm{P}} = 2\rho_{\mathrm{P}}/(1 + \sqrt{1 - 4\gamma\rho_{\mathrm{P}}}). \tag{2.15}$$

We now define

$$\mathcal{S}_{\zeta_{\mathrm{P}}} = \{z \in \mathcal{C}^{2n-1} \; : \; \|z\|_2 \le \zeta_{\mathrm{P}}\}$$

and assume

$$\zeta_{\mathrm{P}} \le 1, \quad \text{i.e.,} \quad 2\rho_{\mathrm{P}}/(1 + \sqrt{1 - 4\gamma\rho_{\mathrm{P}}}) \le 1. \tag{2.16}$$

$\mathcal{S}_{\zeta_{\mathrm{P}}}$ is obviously a bounded closed convex set of $\mathcal{C}^{2n-1}$. Moreover, from (2.11) and (2.14) we see that if $z \in \mathcal{S}_{\zeta_{\mathrm{P}}}$, then

$$\|F(z)\|_2 \le \rho_{\mathrm{P}} + \gamma\|z\|_2^2 \le \rho_{\mathrm{P}} + \gamma\zeta_{\mathrm{P}}^2 = \zeta_{\mathrm{P}},$$

which shows that the continuous mapping $\mathcal{M}_{\mathrm{P}}$ expressed by (2.11) maps $\mathcal{S}_{\zeta_{\mathrm{P}}}$ into $\mathcal{S}_{\zeta_{\mathrm{P}}}$. Thus, by the Brouwer fixed-point theorem, the mapping $\mathcal{M}_{\mathrm{P}}$ has a fixed point in $\mathcal{S}_{\zeta_{\mathrm{P}}}$; i.e., under the hypotheses (2.13) and (2.16), the equation (2.9) has a solution $(\widehat{\Delta a}^T, \hat{t}^T)^T$ satisfying $\|(\widehat{\Delta a}^T, \hat{t}^T)^T\|_2 \le \zeta_{\mathrm{P}}$. Combining this fact with (2.10) gives

$$\beta_{\mathrm{P}}^{(\theta)}(\tilde{x}) \le \sqrt{\theta^2\rho_1^2 + \zeta_{\mathrm{P}}^2} \equiv u(\beta_{\mathrm{P}}^{(\theta)}). \tag{2.17}$$

Note that the relation

$$\rho_{\mathrm{P}} < \min\{1/(4\gamma), \; 1/2\} \tag{2.18}$$

implies (2.13) and (2.16). Consequently, we have proved that under the condition (2.18) the estimate (2.17) holds.

2.3. Lower bounds for $\boldsymbol{\beta_{\mathrm{P}}^{(\theta)}(\tilde{x})}$. Assume that the minimum $\beta_{\mathrm{P}}^{(\theta)}(\tilde{x})$ of (1.9) achieves at Δa_* and Δb_*. Then from (1.9) and (2.8),

$$\beta_{\mathrm{P}}^{(\theta)}(\tilde{x}) = \sqrt{\theta^2\rho_1^2 + \|(\Delta a_*^T,\, \theta\Delta b_{2*}^T)\|_2^2}, \tag{2.19}$$

where Δa_* and Δb_{2*} satisfy the second equation of (2.8). By the results of section 2.2,

$$\|\Delta a_*\|_2 \le \|(\Delta a_*^T,\, \theta\Delta b_{2*}^T)\|_2 \le \zeta_{\mathrm{P}}, \tag{2.20}$$

where ζ_{P} is expressed by (2.15).

Let T_{P} be the $(n-1)\times(2n-1)$ matrix defined by (2.7), and let

$$T_{\mathrm{P}} = U(\Sigma,\, 0)Q^H \quad \text{with} \quad \Sigma = \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_{n-1} \end{pmatrix}, \quad \sigma_1 \ge \cdots \ge \sigma_{n-1} > 0 \tag{2.21}$$

be a singular value decomposition of T_{P}, where U and Q are unitary. Substituting (2.21) and Δa_*, Δb_{2*} into the second equation of (2.8) and letting

$$Q^H \begin{pmatrix} \Delta a_* \\ \theta\Delta b_{2*} \end{pmatrix} = \begin{pmatrix} w \\ * \end{pmatrix} \quad \text{with} \quad w \in \mathcal{C}^{n-1}, \tag{2.22}$$

we get

$$w = \Sigma^{-1}U^H\hat{r}_2 - \Sigma^{-1}U^H(G_2\Delta a_*^{(2)} + \cdots + G_{n-1}\Delta a_*^{(n-1)}).$$

Combining it with (2.20)–(2.22) gives

$$\begin{aligned} \left\| \begin{pmatrix} \Delta a_* \\ \theta\Delta b_{2*} \end{pmatrix} \right\|_2 &\ge \|w\|_2 \\ &\ge \left\|\Sigma^{-1}U^H\hat{r}_2\right\|_2 - \left\|\Sigma^{-1}U^H\left(G_2\Delta a_*^{(2)} + \cdots + G_{n-1}\Delta a_*^{(n-1)}\right)\right\|_2 \\ &\ge \left\|T_{\mathrm{P}}^\dagger\hat{r}_2\right\|_2 - \left(\|T_{\mathrm{P}}^\dagger G_2\|_2\|\Delta a_*\|_2^2 + \cdots + \|T_{\mathrm{P}}^\dagger G_{n-1}\|_2\|\Delta a_*\|_2^{n-1}\right) \\ &\ge \rho_{\mathrm{P}} - (\gamma_2\zeta_{\mathrm{P}}^2 + \cdots + \gamma_{n-1}\zeta_{\mathrm{P}}^{n-1}) \equiv \omega_{\mathrm{P}}. \end{aligned} \tag{2.23}$$

Consequently, we have $\beta_{\mathrm{P}}^{(\theta)}(\tilde{x}) \ge \sqrt{\theta^2\rho_1^2 + \omega_{\mathrm{P}}^2} \equiv l(\beta_{\mathrm{P}}^{(\theta)})$ if $\omega_{\mathrm{P}} \ge 0$.

2.4. Estimates of $\boldsymbol{\beta_{\mathrm{P}}^{(\theta)}(\tilde{x})}$. Overall, we have the following results.

THEOREM 2.1. *Let $\tilde{x}$ be a computed solution to the primal Vandermonde system* (1.3)*, and let $\beta_{\mathrm{P}}^{(\theta)}(\tilde{x})$ be the SBE defined by* (1.9)*. Moreover, let $\gamma_2, \ldots, \gamma_{n-1}$ and ρ_{P} be defined by* (2.12)*. If ρ_{P} satisfies*

$$\rho_{\mathrm{P}} < \min\{1/(4\gamma),\ 1/2\} \quad \textit{with} \quad \gamma = \gamma_2 + \cdots + \gamma_{n-1}, \tag{2.24}$$

then

$$\beta_{\mathrm{P}}^{(\theta)}(\tilde{x}) \le \sqrt{\theta^2\rho_1^2 + \zeta_{\mathrm{P}}^2} \equiv u(\beta_{\mathrm{P}}^{(\theta)}), \tag{2.25}$$

where ζ_{P} *is defined by* (2.15)*. Moreover, let* ω_{P} *be defined by* (2.23)*. Then under the conditions* (2.24) *and* $\omega_{\mathrm{P}} \geq 0$*, we have*

$$\beta_{\mathrm{P}}^{(\theta)}(\tilde{x}) \geq \sqrt{\theta^2 \rho_1^2 + \omega_{\mathrm{P}}^2} \equiv l(\beta_{\mathrm{P}}^{(\theta)}). \tag{2.26}$$

The estimates (2.25) and (2.26) imply $\beta_{\mathrm{P}}^{(\theta)}(\tilde{x}) \approx \sqrt{\theta^2\rho_1^2 + \rho_{\mathrm{P}}^2}$ if $\rho_{\mathrm{P}} \ll 1$.

3. The linearized SBE $\boldsymbol{\beta_{\mathrm{D}}^{(\theta)}(\tilde{x})}$. In this section we present upper and lower bounds for the SBE $\beta_{\mathrm{D}}^{(\theta)}(\tilde{x})$.

Let ΔV be defined by (2.1). Then the constraint $V(a+\Delta a)^T\tilde{x} = b + \Delta b$ in (1.10) can be written as

$$\Delta V^T \tilde{x} = \hat{r}_{\mathrm{D}} + \Delta b \qquad \text{with} \quad \hat{r}_{\mathrm{D}} = b - V(a)^T\tilde{x}. \tag{3.1}$$

Let $\tilde{x}$ and $\Delta a^{(j)}$ $(j = 2, \ldots, n-1)$ be as in (2.4), and let

$$\begin{aligned} &H_j = \xi_{j+1} I_n + \begin{pmatrix} j+1 \\ j \end{pmatrix} \xi_{j+2} D_a + \cdots + \begin{pmatrix} n-1 \\ j \end{pmatrix} \xi_n D_a^{n-1-j} \\ &\text{with} \quad D_a = \operatorname{diag}(\alpha_1, \ldots, \alpha_n), \qquad j = 1, 2, \ldots, n-1, \end{aligned} \tag{3.2}$$

and

$$T_{\mathrm{D}} = \left(H_1, \ -\frac{1}{\theta} I_n \right). \tag{3.3}$$

Then by the expression (2.3), the equation (3.1) is equivalent to

$$T_{\mathrm{D}} \begin{pmatrix} \Delta a \\ \theta \Delta b \end{pmatrix} = \hat{r}_{\mathrm{D}} - (H_2 \Delta a^{(2)} + \cdots + H_{n-1} \Delta a^{(n-1)}). \tag{3.4}$$

Consider the nonlinear system

$$\begin{pmatrix} \Delta a \\ \theta \Delta b \end{pmatrix} = T_{\mathrm{D}}^{\dagger} [\hat{r}_{\mathrm{D}} - (H_2 \Delta a^{(2)} + \cdots + H_{n-1} \Delta a^{(n-1)})]. \tag{3.5}$$

Note that the $n \times 2n$ matrix T_{D} is full rank. By the same argument as above in section 2.2, if $(\widehat{\Delta a}^T, \theta\widehat{\Delta b}^T)^T$ is a solution to (3.5), then it is also a solution to (3.4), and so we have $\beta_{\mathrm{D}}^{(\theta)}(\tilde{x}) \leq \|(\widehat{\Delta a}^T, \theta\widehat{\Delta b}^T)^T\|_2$.

Using the technique described in sections 2.2 through 2.3, we get the following results.

THEOREM 3.1. *Let* $\tilde{x}$ *be a computed solution to* (1.4)*, and let* $\beta_{\mathrm{D}}^{(\theta)}(\tilde{x})$ *be the SBE defined by* (1.10)*. Define* ρ_{D} *and* η_j *by*

$$\rho_{\mathrm{D}} = \|T_{\mathrm{D}}^{\dagger} \hat{r}_{\mathrm{D}}\|_2, \qquad \eta_j = \|T_{\mathrm{D}}^{\dagger} H_j\|_2, \quad j = 2, \ldots, n-1, \tag{3.6}$$

where H_j *and* T_{D} *are the matrices defined by* (3.2) *and* (3.3)*. If* ρ_{D} *satisfies*

$$\rho_{\mathrm{D}} < \min\{1/(4\eta), \ 1/2\} \qquad \textit{with} \quad \eta = \eta_2 + \cdots + \eta_{n-1}, \tag{3.7}$$

then

$$\beta_{\mathrm{D}}^{(\theta)}(\tilde{x}) \leq 2\rho_{\mathrm{D}} / (1 + \sqrt{1 - 4\eta\rho_{\mathrm{D}}}) \equiv u(\beta_{\mathrm{D}}^{(\theta)}). \tag{3.8}$$

Moreover, under the condition (3.7)*, we have*

$$\beta_{\mathrm{D}}^{(\theta)}(\tilde{x}) \geq \rho_{\mathrm{D}} - \left(\eta_2 u(\beta_{\mathrm{D}}^{(\theta)})^2 + \cdots + \eta_{n-1} u(\beta_{\mathrm{D}}^{(\theta)})^{n-1} \right) \equiv l(\beta_{\mathrm{D}}^{(\theta)}). \tag{3.9}$$

The estimates (3.8) and (3.9) imply $\beta_{\mathrm{D}}^{(\theta)}(\tilde{x}) \approx \rho_{\mathrm{D}}$ if $\rho_{\mathrm{D}} \ll 1$.

4. The SBEs $\boldsymbol{\eta_{\mathrm{P}}^{(\theta)}(\tilde{x})}$ and $\boldsymbol{\eta_{\mathrm{D}}^{(\theta)}(\tilde{x})}$. In this section we present upper and lower bounds for the SBEs $\eta_{\mathrm{P}}^{(\theta)}(\tilde{x})$ and $\eta_{\mathrm{D}}^{(\theta)}(\tilde{x})$.

Let $V(a)$ and $V(\tilde{a})$ be the Vandermonde matrices, and let ΔV and Δa be as in (2.1). Then

$$\|(V(\tilde{a}) - V(a),\ \theta\Delta b)\|_F = \sqrt{\|\Delta V\|_F^2 + \theta^2\|\Delta b\|_2^2}. \tag{4.1}$$

By the expression (2.3), the matrix ΔV can be expressed by

$$\Delta V = \Delta V^{(1)} + \Delta V^{(2)} + \cdots + \Delta V^{(n-1)}, \tag{4.2}$$

where $\Delta V^{(j)}$ $(j = 1, 2, \ldots, n-1) \in \mathcal{C}^{n\times n}$ are defined by

$$\Delta V^{(j)} = A_j[\mathrm{diag}(\epsilon_1, \ldots, \epsilon_n)]^j \tag{4.3}$$

with

$$A_j = D_j^{(n)} \begin{pmatrix} 0 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 \\ 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \vdots & \vdots & & \vdots \\ \alpha_1^{n-1-j} & \alpha_2^{n-1-j} & \cdots & \alpha_n^{n-1-j} \end{pmatrix} \in \mathcal{C}^{n\times n}, \tag{4.4}$$

in which $D_j^{(n)}$ are $n \times n$ diagonal matrices expressed by

$$D_j^{(n)} = \mathrm{diag}\left(0, \ldots, 0, \begin{pmatrix} j \\ j \end{pmatrix}, \begin{pmatrix} j+1 \\ j \end{pmatrix}, \ldots, \begin{pmatrix} n-1 \\ j \end{pmatrix}\right).$$

Observe that (4.3) and (4.4) imply that

$$\|\Delta V^{(1)}\|_F = \|M\Delta a\|_2,$$

where

$$M = \mathrm{diag}(\mu_1, \ldots, \mu_n) \quad \text{with} \quad \mu_k = \sqrt{\sum_{l=1}^{n-1} (l|\alpha_k|^{l-1})^2} \geq 1, \quad k = 1, \ldots, n. \tag{4.5}$$

Moreover, from (4.3),

$$\|\Delta V^{(j)}\|_F \leq \|A_j M^{-j}\|_2 \|M\Delta a\|_2^j, \quad j = 2, \ldots, n-1.$$

Hence, we have

$$\begin{aligned} \sqrt{\|\Delta V\|_F^2 + \theta^2\|\Delta b\|_2^2} &\leq \sqrt{\left(\|\Delta V^{(1)}\|_F + \sum_{j=2}^{n-1} \|\Delta V^{(j)}\|_F\right)^2 + \theta^2\|\Delta b\|_2^2} \\ &\leq \sqrt{\|\Delta V^{(1)}\|_F^2 + \theta^2\|\Delta b\|_2^2} + \sum_{j=2}^{n-1} \|\Delta V^{(j)}\|_F \\ &\leq \|(M\Delta a,\ \theta\Delta b)\|_F + \sum_{j=2}^{n-1} \|A_j M^{-j}\|_2 \|(M\Delta a,\ \theta\Delta b)\|_F^j \end{aligned} \tag{4.6}$$

and

$$\sqrt{\|\Delta V\|_F^2 + \theta^2\|\Delta b\|_2^2} \geq \sqrt{\left(\|\Delta V^{(1)}\|_F - \sum_{j=2}^{n-1}\|\Delta V^{(j)}\|_F\right)^2 + \theta^2\|\Delta b\|_2^2}$$

$$\geq \sqrt{\|\Delta V^{(1)}\|_F^2 + \theta^2\|\Delta b\|_2^2} - \sum_{j=2}^{n-1}\|\Delta V^{(j)}\|_F \tag{4.7}$$

$$\geq \|(M\Delta a,\, \theta\Delta b)\|_F - \sum_{j=2}^{n-1}\|A_j M^{-j}\|_2\|(M\Delta a,\, \theta\Delta b)\|_F^j.$$

Define $\hat{\beta}_{\mathrm{P}}^{(\theta)}(\tilde{x})$ by

$$\hat{\beta}_{\mathrm{P}}^{(\theta)}(\tilde{x}) = \min\{\|(M\Delta a,\, \theta\Delta b)\|_F \;:\; V(a+\Delta a)\tilde{x} = b + \Delta b\}. \tag{4.8}$$

From (4.6)–(4.7) and (1.5) we see that if $l(\hat{\beta}_{\mathrm{P}}^{(\theta)}) \leq \hat{\beta}_{\mathrm{P}}^{(\theta)}(\tilde{x}) \leq u(\hat{\beta}_{\mathrm{P}}^{(\theta)})$, then

$$l(\hat{\beta}_{\mathrm{P}}^{(\theta)}) - \sum_{j=2}^{n-1}\|A_j M^{-j}\|_2 u(\hat{\beta}_{\mathrm{P}}^{(\theta)})^j \leq \eta_{\mathrm{P}}^{(\theta)}(\tilde{x}) \leq u(\hat{\beta}_{\mathrm{P}}^{(\theta)}) + \sum_{j=2}^{n-1}\|A_j M^{-j}\|_2 u(\hat{\beta}_{\mathrm{P}}^{(\theta)})^j,$$

where A_j and M are defined by (4.4) and (4.5), respectively.

We now consider the problem of finding upper and lower bounds $u(\hat{\beta}_{\mathrm{P}}^{(\theta)})$ and $l(\hat{\beta}_{\mathrm{P}}^{(\theta)})$ for $\hat{\beta}_{\mathrm{P}}^{(\theta)}(\tilde{x})$. By section 2.1, the constraint $V(a+\Delta a)\tilde{x} = b + \Delta b$ in (1.5) is equivalent to (2.8). Using the nonsingular diagonal matrix M defined by (4.5), we rewrite (2.8) as

$$\begin{cases} \delta_1 = -\rho_1, \\ \hat{T}_{\mathrm{P}}\begin{pmatrix} M\Delta a \\ \theta\Delta b_2 \end{pmatrix} = \hat{r}_2 - (\hat{G}_2 M^2 \Delta a^{(2)} + \cdots + \hat{G}_{n-1}M^{n-1}\Delta a^{(n-1)}), \end{cases} \tag{4.9}$$

where

$$\hat{T}_{\mathrm{P}} = T_{\mathrm{P}}\begin{pmatrix} M^{-1} & 0 \\ 0 & I_{n-1} \end{pmatrix}, \qquad \hat{G}_j = G_j M^{-j}, \quad j = 2, \ldots, n-1. \tag{4.10}$$

Comparing (4.8) with (1.9) and comparing (4.9) with (2.8), we see that the problem of finding upper and lower bounds for $\hat{\beta}_{\mathrm{P}}^{(\theta)}(\tilde{x})$ is quite similar to that for $\beta_{\mathrm{P}}^{(\theta)}(\tilde{x})$. Consequently, we can apply Theorem 2.1 to obtain corresponding bounds for $\hat{\beta}_{\mathrm{P}}^{(\theta)}(\tilde{x})$, and so we have the following result.

THEOREM 4.1. *Let $\tilde{x}$ be a computed solution to the primal Vandermonde system* (1.3)*, and let $\hat{\beta}_{\mathrm{P}}^{(\theta)}(\tilde{x})$ and $\eta_{\mathrm{P}}^{(\theta)}(\tilde{x})$ be defined by* (4.8) *and* (1.5)*, respectively. Moreover, let ρ_1 be the first element of the vector $\hat{r} = b - V(a)\tilde{x}$, the matrices A_j, M be defined by* (4.4)*–*(4.5)*, and let*

$$\hat{\rho}_{\mathrm{P}} = \|\hat{T}_{\mathrm{P}}^{\dagger}\hat{r}_2\|_2, \qquad \hat{\gamma}_j = \|\hat{T}_{\mathrm{P}}^{\dagger}\hat{G}_j\|_2, \quad j = 2, \ldots, n-1, \tag{4.11}$$

where $\hat{r}_2$ is defined by (2.4) *and $\hat{T}_{\mathrm{P}}, \hat{G}_j$ are defined by* (4.10)*. If $\hat{\rho}_{\mathrm{P}}$ satisfies*

$$\hat{\rho}_{\mathrm{P}} < \min\{1/(4\hat{\gamma}),\ 1/2\} \quad \textit{with} \quad \hat{\gamma} = \hat{\gamma}_2 + \cdots + \hat{\gamma}_{n-1}, \tag{4.12}$$

then

$$\hat{\beta}_{\mathrm{P}}^{(\theta)}(\tilde{x}) \le \sqrt{\theta^2\rho_1^2 + \hat{\zeta}_{\mathrm{P}}^2} \equiv u(\hat{\beta}_{\mathrm{P}}^{(\theta)})$$

and

$$\eta_{\mathrm{P}}^{(\theta)}(\tilde{x}) \le u(\hat{\beta}_{\mathrm{P}}^{(\theta)}) + \sum_{j=2}^{n-1} \|A_j M^{-j}\|_2 u(\hat{\beta}_{\mathrm{P}}^{(\theta)})^j \equiv u(\eta_{\mathrm{P}}^{(\theta)}), \tag{4.13}$$

where $\hat{\zeta}_{\mathrm{P}} = 2\hat{\rho}_{\mathrm{P}}/(1+\sqrt{1-4\hat{\gamma}\hat{\rho}_{\mathrm{P}}})$. *Moreover, let*

$$\hat{\omega}_{\mathrm{P}} = \hat{\rho}_{\mathrm{P}} - (\hat{\gamma}_2\hat{\zeta}_{\mathrm{P}}^2 + \cdots + \hat{\gamma}_{n-1}\hat{\zeta}_{\mathrm{P}}^{n-1}).$$

Then under the conditions (4.12) *and* $\hat{\omega}_{\mathrm{P}} \ge 0$, *we have*

$$\hat{\beta}_{\mathrm{P}}^{(\theta)}(\tilde{x}) \ge \sqrt{\theta^2\rho_1^2 + \hat{\omega}_{\mathrm{P}}^2} \equiv l(\hat{\beta}_{\mathrm{P}}^{(\theta)})$$

and

$$\eta_{\mathrm{P}}^{(\theta)}(\tilde{x}) \ge l(\hat{\beta}_{\mathrm{P}}^{(\theta)}) - \sum_{j=2}^{n-1} \|A_j M^{-j}\|_2 u(\hat{\beta}_{\mathrm{P}}^{(\theta)})^j \equiv l(\eta_{\mathrm{P}}^{(\theta)}). \tag{4.14}$$

Similarly, we define $\hat{\beta}_{\mathrm{D}}^{(\theta)}(\tilde{x})$ by

$$\hat{\beta}_{\mathrm{D}}^{(\theta)}(\tilde{x}) = \min\{\|(M\Delta a,\, \theta\Delta b)\|_F \ : \ V(a+\Delta a)^T\tilde{x} = b + \Delta b\}. \tag{4.15}$$

From (4.6)–(4.7) and (1.6) we see that if $l(\hat{\beta}_{\mathrm{D}}^{(\theta)}) \le \hat{\beta}_{\mathrm{D}}^{(\theta)}(\tilde{x}) \le u(\hat{\beta}_{\mathrm{D}}^{(\theta)})$, then

$$l(\hat{\beta}_{\mathrm{D}}^{(\theta)}) - \sum_{j=2}^{n-1} \|A_j M^{-j}\|_2 u(\hat{\beta}_{\mathrm{D}}^{(\theta)})^j \le \eta_{\mathrm{D}}^{(\theta)}(\tilde{x}) \le u(\hat{\beta}_{\mathrm{D}}^{(\theta)}) + \sum_{j=2}^{n-1} \|A_j M^{-j}\|_2 u(\hat{\beta}_{\mathrm{D}}^{(\theta)})^j.$$

We now consider the problem of finding upper and lower bounds $u(\hat{\beta}_{\mathrm{D}}^{(\theta)})$ and $l(\hat{\beta}_{\mathrm{D}}^{(\theta)})$ for $\hat{\beta}_{\mathrm{D}}^{(\theta)}(\tilde{x})$. By section 3.1, the constraint $V(a+\Delta a)^T\tilde{x} = b + \Delta b$ in (1.6) is equivalent to (3.4). Using the nonsingular diagonal matrix M defined by (4.5), we rewrite (3.4) as

$$\hat{T}_{\mathrm{D}} \begin{pmatrix} M\Delta a \\ \theta\Delta b \end{pmatrix} = \hat{r}_{\mathrm{D}} - (\hat{H}_2 M^2 \Delta a^{(2)} + \cdots + \hat{H}_{n-1} M^{n-1} \Delta a^{(n-1)}), \tag{4.16}$$

where $\hat{r}_{\mathrm{D}} = b - V(a)^T\tilde{x}$ and

$$\hat{T}_{\mathrm{D}} = T_{\mathrm{D}} \begin{pmatrix} M^{-1} & 0 \\ 0 & I_n \end{pmatrix}, \qquad \hat{H}_j = H_j M^{-j}, \quad j = 2, \ldots, n-1, \tag{4.17}$$

in which H_j and T_{D} are defined by (3.2)–(3.3). Comparing (4.15) with (1.10) and comparing (4.16) with (3.4), we see that the problem of finding upper and lower bounds for $\hat{\beta}_{\mathrm{D}}^{(\theta)}(\tilde{x})$ is quite similar to that for $\beta_{\mathrm{D}}^{(\theta)}(\tilde{x})$. Consequently, we can apply Theorem 3.1 to obtain corresponding bounds for $\hat{\beta}_{\mathrm{D}}^{(\theta)}(\tilde{x})$, and so we have the following result.

THEOREM 4.2. *Let $\tilde{x}$ be a computed solution to the dual Vandermonde system* (1.4)*, and let $\hat{\beta}_{\mathrm{D}}^{(\theta)}(\tilde{x})$ and $\eta_{\mathrm{D}}^{(\theta)}(\tilde{x})$ be defined by* (4.15) *and* (1.6)*, respectively. Moreover, let the matrices A_j, M be defined by* (4.4)*–*(4.5)*, and*

$$\hat{\rho}_{\mathrm{D}} = \|\hat{T}_{\mathrm{P}}^{\dagger}\hat{r}_{\mathrm{D}}\|_2, \quad \hat{\eta}_j = \|\hat{T}_{\mathrm{D}}^{\dagger}\hat{H}_j\|_2, \quad j = 2, \ldots, n-1, \tag{4.18}$$

where $\hat{r}_{\mathrm{D}} = b - V(a)^T\tilde{x}$, and $\hat{T}_{\mathrm{D}}, \hat{H}_j$ are defined by (4.17)*. If $\hat{\rho}_{\mathrm{D}}$ satisfies*

$$\hat{\rho}_{\mathrm{D}} < \min\{1/(4\hat{\eta}),\ 1/2\} \quad \textit{with} \quad \hat{\eta} = \hat{\eta}_2 + \cdots + \hat{\eta}_{n-1}, \tag{4.19}$$

then

$$\hat{\beta}_{\mathrm{D}}^{(\theta)}(\tilde{x}) \leq 2\hat{\rho}_{\mathrm{D}}/(1 + \sqrt{1 - 4\hat{\eta}\hat{\rho}_{\mathrm{D}}}) \equiv u(\hat{\beta}_{\mathrm{D}}^{(\theta)})$$

and

$$\eta_{\mathrm{D}}^{(\theta)}(\tilde{x}) \leq u(\hat{\beta}_{\mathrm{D}}^{(\theta)}) + \sum_{j=2}^{n-1} \|A_j M^{-j}\|_2 u(\hat{\beta}_{\mathrm{D}}^{(\theta)})^j \equiv u(\eta_{\mathrm{D}}^{(\theta)}). \tag{4.20}$$

Moreover, under the condition (4.19)*, we have*

$$\hat{\beta}_{\mathrm{D}}^{(\theta)}(\tilde{x}) \geq \hat{\rho}_{\mathrm{D}} - \left(\hat{\eta}_2 u(\hat{\beta}_{\mathrm{D}}^{(\theta)})^2 + \cdots + \hat{\eta}_{n-1} u(\hat{\beta}_{\mathrm{D}}^{(\theta)})^{n-1}\right) \equiv l(\hat{\beta}_{\mathrm{D}}^{(\theta)})$$

and

$$\eta_{\mathrm{D}}^{(\theta)}(\tilde{x}) \geq l(\hat{\beta}_{\mathrm{D}}^{(\theta)}) - \sum_{j=2}^{n-1} \|A_j M^{-j}\|_2 u(\hat{\beta}_{\mathrm{D}}^{(\theta)})^j \equiv l(\eta_{\mathrm{D}}^{(\theta)}). \tag{4.21}$$

5. Numerical examples. In this section, we use numerical examples to illustrate the estimates for the SBEs $\beta_{\mathrm{P}}^{(\theta)}(\tilde{x})$, $\beta_{\mathrm{D}}^{(\theta)}(\tilde{x})$, $\eta_{\mathrm{P}}^{(\theta)}(\tilde{x})$, and $\eta_{\mathrm{D}}^{(\theta)}(\tilde{x})$. All computations were performed using MATLAB, version 4.2c. The relative machine precision is 2.22×10^{-16}.

Consider the Vandermonde systems (1.3) and (1.4), where $V = V(a)$ is a 10×10 Vandermonde matrix with $a = (\alpha_1, \ldots, \alpha_{10})$ and $b = (\beta_1, \ldots, \beta_{10})^T$.

Example 5.1 (see [2]). $\alpha_i = 1/(i+2),\ \beta_i = 1/2^{i-1},\ i = 1 : 10.$

Example 5.2. $\alpha_i = 1/(i+2),\ \beta_i = (-1)^{i+1}/2^{i-1},\ i = 1 : 10.$

Example 5.3. $\alpha_i = 1 + i/10,\ \beta = i + 1/i,\ i = 1 : 10.$

Example 5.4. $\alpha_i = 1 + i/10,\ \beta = (-1)^{i+1}i + 1/i,\ i = 1 : 10.$

The computed solutions are obtained by using the Björck–Pereyra algorithms [2], [4, section 4.6], [6].

Taking $\theta = \|V(a)\|_F/\|b\|_2 \equiv \theta_V$ in (1.5), we get the SBE $\eta_{\mathrm{P}}(\tilde{x})$:

$$\eta_{\mathrm{P}}(\tilde{x}) \equiv \frac{\eta_{\mathrm{P}}^{(\theta_V)}(\tilde{x})}{\|V(a)\|_F} = \min\left\{\left\|\left(\frac{\|\Delta V\|_F}{\|V(a)\|_F}, \frac{\|\Delta b\|_2}{\|b\|_2}\right)\right\|_2 \ :\ (V(a) + \Delta V)\tilde{x} = b + \Delta b\right\}.$$

Taking $\theta = \|a\|_2/\|b\|_2 \equiv \theta_a$ in (1.9), we get the linearized SBE $\beta_{\mathrm{P}}(\tilde{x})$:

$$\beta_{\mathrm{P}}(\tilde{x}) \equiv \frac{\beta_{\mathrm{P}}^{(\theta_a)}(\tilde{x})}{\|a\|_2} = \min\left\{\left\|\left(\frac{\|\Delta a\|_2}{\|a\|_2}, \frac{\|\Delta b\|_2}{\|b\|_2}\right)\right\|_2 \ :\ V(a + \Delta a)\tilde{x} = b + \Delta b\right\}.$$

Table 1

Example	$l(\beta_{\rm P})$	$u(\beta_{\rm P})$	$l(\eta_{\rm P})$	$u(\eta_{\rm P})$	$\eta(\tilde{x})$
5.1	1.61e-09	1.61e-09	2.82e-10	2.82e-10	3.97e-18
5.2	2.48e-06	2.48e-06	—	—	9.36e-18
5.3	2.32e-10	2.32e-10	2.55e-10	2.55e-10	6.20e-18
5.4	1.24e-08	1.24e-08	1.37e-08	1.37e-08	6.41e-18

Similarly, we can define the SBE $\eta_{\rm D}(\tilde{x})$, the linearized SBE $\beta_{\rm D}(\tilde{x})$, and the BE $\eta(\tilde{x})$:

$$\eta_{\rm D}(\tilde{x}) \equiv \frac{\eta_{\rm D}^{(\theta_V)}(\tilde{x})}{\|V(a)\|_F}, \qquad \beta_{\rm D}(\tilde{x}) \equiv \frac{\beta_{\rm D}^{(\theta_a)}(\tilde{x})}{\|a\|_2}, \qquad \eta(\tilde{x}) \equiv \frac{\eta^{(\theta_V)}(\tilde{x})}{\|V(a)\|_F}.$$

Further, define $l(\eta_{\rm P})$, $u(\eta_{\rm P})$, $l(\beta_{\rm P})$, and $u(\beta_{\rm P})$ by

$$l(\eta_{\rm P}) = \frac{l(\eta_{\rm P}^{(\theta_V)})}{\|V(a)\|_F}, \quad u(\eta_{\rm P}) = \frac{u(\eta_{\rm P}^{(\theta_V)})}{\|V(a)\|_F}, \quad l(\beta_{\rm P}) = \frac{l(\beta_{\rm P}^{(\theta_a)})}{\|a\|_2}, \quad u(\beta_{\rm P}) = \frac{u(\beta_{\rm P}^{(\theta_a)})}{\|a\|_2},$$

respectively, where $l(\eta_{\rm P}^{(\theta_V)})$, $u(\eta_{\rm P}^{(\theta_V)})$, $l(\beta_{\rm P}^{(\theta_a)})$, and $u(\beta_{\rm P}^{(\theta_a)})$ can be computed by the formulas (4.14), (4.13), (2.26), and (2.25), respectively. Similarly, we can define and compute $l(\eta_{\rm D}), u(\eta_{\rm D})$, $l(\beta_{\rm D}), and\ u(\beta_{\rm D})$. By Theorems 4.1, 4.2, 2.1, and 3.1, we have

$$\begin{aligned} &l(\eta_{\rm P}) \le \eta_{\rm P}(\tilde{x}) \le u(\eta_{\rm P}), \qquad l(\eta_{\rm D}) \le \eta_{\rm D}(\tilde{x}) \le u(\eta_{\rm D}), \\ &l(\beta_{\rm P}) \le \beta_{\rm P}(\tilde{x}) \le u(\beta_{\rm P}), \qquad l(\beta_{\rm D}) \le \beta_{\rm D}(\tilde{x}) \le u(\beta_{\rm D}), \end{aligned} \tag{5.1}$$

respectively.

Some numerical results on lower and upper bounds for the SBEs $\beta_{\rm P}(\tilde{x})$, $\eta_{\rm P}(\tilde{x})$, $\beta_{\rm D}(\tilde{x})$, and $\eta_{\rm D}(\tilde{x})$, and on the BE $\eta(\tilde{x})$, are listed in Tables 1–2.

From the results listed in Table 1, we see the following facts. For any one of Examples 5.1–5.4, the Björck–Pereyra algorithm produces a computed $\tilde{x}$ with small SBEs $\beta_{\rm P}(\tilde{x})$ and $\eta_{\rm P}(\tilde{x})$ (except Example 5.2) and a very small BE $\eta(\tilde{x})$ (which is much smaller than $\beta_{\rm P}(\tilde{x})$ and $\eta_{\rm P}(\tilde{x})$); that is, $\tilde{x}$ is the solution to a Vandermonde system $V(\tilde{a})x = \tilde{b}$ and a linear system $\hat{A}x = \hat{b}$, where $\tilde{a}$, $V(\tilde{a})$ (except Example 5.2), and $\tilde{b}$ are relatively close to a, $V(a)$, and b, respectively, and $\hat{A}$ and $\hat{b}$ are relatively much closer to $V(a)$ and b. Note that for Example 5.2, the formulas (4.13)–(4.14) cannot be used to get upper and lower bounds for the SBE $\eta_{\rm P}^{(\theta)}(\tilde{x})$ because the condition (4.12) is violated.

Table 2

Example	$l(\beta_{\rm D})$	$u(\beta_{\rm D})$	$l(\eta_{\rm D})$	$u(\eta_{\rm D})$	$\eta(\tilde{x})$
5.1	1.50e-18	1.50e-18	3.27e-19	3.27e-19	3.65e-20
5.2	1.99e-19	1.99e-19	4.35e-20	4.35e-20	1.16e-20
5.3	7.05e-15	7.05e-15	7.51e-14	7.51e-14	5.85e-18
5.4	4.91e-15	4.91e-15	3.82e-14	3.82e-14	3.11e-18

The results listed in Table 2 show that for any one of Examples 5.1–5.4, the Björck–Pereyra algorithm produces a computed $\tilde{x}$, which is the solution to a linear system $\hat{A}x = \hat{b}$ and a Vandermonde system $V(\tilde{a})^T x = \tilde{b}$, where $\tilde{a}$ and $\hat{b}$, $\tilde{b}$ are relatively very close to a and b, respectively, and $\hat{A}$ and $V(\tilde{a})$ are relatively very close to $V(a)$.

Remark 5.5. It is worth pointing out two facts: (i) The assumptions (2.24), (3.7), (4.12), and (4.19) are somewhat harsh terms; e.g., the assumption (4.12) is violated for Example 5.2 (see Table 1), but by our technique, the assumptions are difficult to remove. (ii) For the dual problem, the expressions (3.8) (for $u(\beta_{\mathrm{D}}^{(\theta)})$), (3.9) (for $l(\beta_{\mathrm{D}}^{(\theta)})$), (4.20) (for $u(\eta_{\mathrm{D}}^{(\theta)})$), and (4.21) (for $l(\eta_{\mathrm{D}}^{(\theta)})$) can be computed in $O(n^2)$ flop operations, but in the primal case, the expressions (2.25) (for $u(\eta_{\mathrm{P}}^{(\theta)})$), (2.26) (for $l(\beta_{\mathrm{P}}^{(\theta)})$), (4.13) (for $u(\eta_{\mathrm{P}}^{(\theta)})$), and (4.14) (for $l(\eta_{\mathrm{P}}^{(\theta)})$) require $O(n^3)$ flop operations.

Finally, we give a comparison between our results and those of [1]. For simplicity and convenience, we only consider the dual system (1.4) and take the linearized SBE $\mathrm{be}_{\infty}^{\mathrm{dual}}(a,b,\tilde{x})$, defined by [1, Equation (3.1)],

$$\mathrm{be}_{\infty}^{\mathrm{dual}}(a,b,\tilde{x}) = \min\left\{ \left\| \begin{pmatrix} \widehat{\Delta a} \\ \widehat{\Delta b} \end{pmatrix} \right\|_{\infty} \ : \ V(a+\Delta a)^T \tilde{x} = b + \Delta b \right\}, \tag{5.2}$$

where $\tilde{x}$ is an approximate solution to (1.4), and

$$\widehat{\Delta a} = (\epsilon_1/|\alpha_1|, \ldots, \epsilon_n/|\alpha_n|)^T, \qquad \widehat{\Delta b} = (\delta_1/|\beta_1|, \ldots, \delta_n/|\beta_n|)^T, \tag{5.3}$$

in which α_i, β_i, ϵ_i, and δ_i are the components of $a, b, \Delta a$, and Δb, respectively. (Note that by [1, section 3], the denominators $|\alpha_i|$ and $|\beta_i|$ in (5.3) can be replaced by any nonnegative scalars.) If we linearize the constraint in (5.2), then we get the further linearized SBE $\mathrm{linbe}_{\infty}^{\mathrm{dual}}(a,b,\tilde{x})$, defined by [1, Equation (3.4)],

$$\mathrm{linbe}_{\infty}^{\mathrm{dual}}(a,b,\tilde{x}) = \min_i \left\{ \max\left\{ \frac{|\epsilon_i|}{|\alpha_i|}, \ \frac{|\delta_i|}{|\beta_i|} \right\} \ : \ \epsilon_i \tilde{\zeta}_i - \delta_i = \rho_i \right\},$$

where $(\rho_1, \ldots, \rho_n)^T = b - V(a)^T \tilde{x}$ and $(\tilde{\zeta}_1, \ldots, \tilde{\zeta}_n)^T = V'(a)^T \tilde{x}$, in which the matrix $V'(a)$ is defined by $V'(a) = \left(\frac{d}{d\alpha_j}(V(a))_{ij}\right)$. By [1, Equation (3.6)],

$$\mathrm{linbe}_{\infty}^{\mathrm{dual}}(a,b,\tilde{x}) = \max_i \left\{ \frac{|\rho_i|}{|\tilde{\zeta}_i||\alpha_i| + |\beta_i|} \right\} \equiv \mathrm{linbe}. \tag{5.4}$$

Moreover, define Δa^* and Δb^{**} by [1, section 3]

$$\Delta a^* = \left(\frac{\mathrm{sign}(\tilde{\zeta}_1)\rho_1|\alpha_1|}{|\tilde{\zeta}_1||\alpha_1| + |\beta_1|}, \ldots, \frac{\mathrm{sign}(\tilde{\zeta}_n)\rho_n|\alpha_n|}{|\tilde{\zeta}_n||\alpha_n| + |\beta_n|} \right)^T \equiv (\epsilon_1^*, \ldots, \epsilon_n^*)^T,$$

$$\Delta b^{**} = V(a + \Delta a^*)^T \tilde{x} - b \equiv (\delta_1^{**}, \ldots, \delta_n^{**})^T.$$

Then by [1, Equation (3.7)], we have

$$\mathrm{be}_{\infty}^{\mathrm{dual}}(a,b,\tilde{x}) \le \max\left\{ \max_i \frac{|\epsilon_i^*|}{|\alpha_i|}, \ \max_i \frac{|\delta_i^{**}|}{|\beta_i|} \right\} \equiv u_{\mathrm{BH}}(\mathrm{be}_{\infty}^{\mathrm{dual}}). \tag{5.5}$$

Note that by the definitions of $\mathrm{be}_{\infty}^{\mathrm{dual}}(a,b,\tilde{x})$ and $\beta_{\mathrm{D}}(\tilde{x})$ and using the relation $l(\beta_{\mathrm{D}}) \le \beta_{\mathrm{D}}(\tilde{x}) \le u(\beta_{\mathrm{D}})$ (see (5.1)), we get

$$\begin{aligned} &\mathrm{be}_{\infty}^{\mathrm{dual}}(a,b,\tilde{x}) \ge \tfrac{1}{\sqrt{2}} \min\left\{ \tfrac{\|a\|_2}{\sqrt{n}\max|\alpha_i|}, \ \tfrac{\|b\|_2}{\sqrt{n}\max|\beta_i|} \right\} l(\beta_{\mathrm{D}}) \equiv l(\mathrm{be}_{\infty}^{\mathrm{dual}}), \\ &\mathrm{be}_{\infty}^{\mathrm{dual}}(a,b,\tilde{x}) \le \max\left\{ \tfrac{\|a\|_2}{\min|\alpha_i|}, \ \tfrac{\|b\|_2}{\min|\beta_i|} \right\} u(\beta_{\mathrm{D}}) \equiv u(\mathrm{be}_{\infty}^{\mathrm{dual}}). \end{aligned} \tag{5.6}$$

TABLE 3

Example	linbe	$u_{\mathrm{BH}}(\mathrm{be}_\infty^{\mathrm{dual}})$	$l(\mathrm{be}_\infty^{\mathrm{dual}})$	$u(\mathrm{be}_\infty^{\mathrm{dual}})$
5.1	7.74e-14	1.66e-13	3.87e-19	8.85e-16
5.2	1.80e-10	1.58e-08	5.12e-20	1.18e-16
5.3	6.84e-11	1.15e-10	3.15e-15	7.11e-14
5.4	6.18e-08	3.16e-06	2.37e-15	6.44e-14

Some numerical results on $\mathrm{linbe}_\infty^{\mathrm{dual}}(a, b, \tilde{x})$ and lower and upper bounds for $\mathrm{be}_\infty^{\mathrm{dual}}(a, b, \tilde{x})$ are listed in Table 3.

Two comments: (i) From the results listed in Table 3, we see that the further linearized SBE $\mathrm{linbe}_\infty^{\mathrm{dual}}(a, b, \tilde{x})$ defined by [1, Equation (3.4)] (which linearizes not only the objective function but also the constraint!) is a rough measure. (ii) The SBE $\mathrm{be}_\infty^{\mathrm{dual}}(a, b, \tilde{x})$, as a *componentwise* SBE defined by [1], ought to give a more satisfactory measure of the distance between the perturbed systems $V(\tilde{a})^T\tilde{x} = b + \Delta b$ and the original system $V(a)^T x = b$. However, the results listed in Table 3 show that the upper bounds $u_{\mathrm{BH}}(\mathrm{be}_\infty^{\mathrm{dual}})$ computed by [1, Equation (3.7)] (see (5.5)) are conservative. (Note that $l(\mathrm{be}_\infty^{\mathrm{dual}})$ and $u(\mathrm{be}_\infty^{\mathrm{dual}})$ (see (5.6)) are lower and upper bounds for $\mathrm{be}_\infty^{\mathrm{dual}}(a, b, \tilde{x})$ obtained by applying our results.) The question of how to derive a sharper bound for $\mathrm{be}_\infty^{\mathrm{dual}}(a, b, \tilde{x})$ merits further investigation.

Acknowledgments. I am grateful to Nick Higham and the referees for constructive comments and suggestions. Nick Higham and one of the referees also pointed out that Bartels and D. J. Higham [1] already considered structured backward error for Vandermonde systems, including for componentwise measures of the perturbations.

REFERENCES

[1] S. G. BARTELS AND D. J. HIGHAM, *The structured sensitivity of Vandermonde-like systems,* Numer. Math., 62 (1992), pp. 17–33.

[2] Å. BJÖRCK AND V. PEREYRA, *Solution of Vandermonde systems of equations,* Math. Comp., 24 (1970), pp. 893–903.

[3] J. R. BUNCH, *The weak and strong stability of algorithms in numerical linear algebra,* Linear Algebra Appl., 88/89 (1987), pp. 49–66.

[4] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations,* 3rd ed., Johns Hopkins University Press, Baltimore, MD, and London, 1996.

[5] D. J. HIGHAM AND N. J. HIGHAM, *Backward error and condition of structured linear systems,* SIAM J. Matrix Anal. Appl., 13 (1992), pp. 162–175.

[6] N. J. HIGHAM, *Error analysis of the Björck-Pereyra algorithms for solving Vandermonde systems,* Numer. Math., 50 (1987), pp. 613–632.

[7] N. J. HIGHAM, *Accuracy and Stability of Numerical Algorithms,* SIAM, Philadelphia, PA, 1996.

[8] J. L. RIGAL AND J. GACHES, *On the compatibility of a given solution with the data of a linear system,* J. Assoc. Comput. Mach., 14 (1967), pp. 543–548.

[9] J.-G. SUN, *Optimal Backward Perturbation Bounds for Linear Systems and Linear Least Squares Problems,* UMINF 96.15, ISSN-0348-0542, Department of Computing Science, Umeå University, Umeå, Sweden, 1996.

[10] J. M. VARAH, *Backward error estimates for Toeplitz systems,* SIAM J. Matrix Anal. Appl., 15 (1994), pp. 408–417.

SIAM J. MATRIX ANAL. APPL.
Vol. 20, No. 1, pp. 60-77

A COMPRESSION ALGORITHM FOR PROBABILITY TRANSITION MATRICES[*]

WILLIAM M. SPEARS[†]

Abstract. This paper describes a compression algorithm for probability transition matrices. The compressed matrix is itself a probability transition matrix. In general the compression is not error free, but the error appears to be small even for high levels of compression.

Key words. probability transition matrix, transient behavior, compression, lumping, aggregation

AMS subject classifications. 15A51, 15A04

PII. S0895479897316916

1. Introduction.

Many discrete systems can be described by a Markov chain model in which each state of the Markov model is some discrete state of the dynamical system. If there are N states, then the Markov chain model is defined by an $N \times N$ matrix Q called the "1-step probability transition matrix," where $Q(i, j)$ is the probability of going from state i to state j in one step. The n-step behavior is described by the nth power of Q, Q^n. For many systems, the number of states is enormous and there is a computational advantage in reducing N.

Previous methods for reducing the number of states (referred to as *compression*, *aggregation*, or *lumping* methods) have focused on techniques that provide good estimations of the steady-state behavior of the Markov model. The focus of this paper, however, is on *transient* behavior, and the goal is to produce an algorithm for compressing Q matrices in a way that yields good estimates of the transient behavior of the Markov model. The algorithm described in this paper compresses a Q matrix into a smaller Q matrix with less states. In general, the compression will not be without error, so the goal is to provide an algorithm that compresses the original Q matrix without significant error. Although computing a compressed matrix might take some time, the savings resulting from using this compressed matrix in all subsequent computations can more than offset the compression time.

The organization of this paper is as follows. Section 2 introduces the compression algorithm, which compresses pairs of states by taking a *weighted* average of the row entries for those two states, followed by summing the two columns associated with those two states. Section 2 also introduces the important concepts of row and column *equivalence*, which are important for identifying pairs of states that can be compressed with no error. Section 3 provides mathematical justification for taking the weighted average of row entries and shows that the weights are simply column sums of probability mass. Section 4 proves that pairs of states that are row or column equivalent lead to perfect compression. Section 5 introduces an analysis of error and uses this to define a metric for row and column *similarity* which can be used to find pairs of states that yield almost perfect compression. Later sections illustrate the utility of the compression algorithm through experiments.

[*] Received by the editors February 25, 1997; accepted for publication (in revised form) January 7, 1998; published electronically September 15, 1998. This work was supported by ARPA Order D106/03.

http://www.siam.org/journals/simax/20-1/31691.html

[†] AI Center - Code 5514, Naval Research Laboratory, 4555 Overlook Avenue, Washington, DC 20375 (spears@aic.nrl.navy.mil).

2. The compression algorithm at a high level. The entries in the Q matrix, $p_{i,j} \equiv Q(i,j)$, represent the conditional probability that the system will transition to state j in one step, given that it currently is in state i.[1] Now suppose that states i and j have been chosen for compression. The new compressed state is referred to as state $\{i \vee j\}$. Compressing states i and j together means that the combined state represents being in either state i or state j. Since this is a disjunctive situation, the probability of transition from state k *into* the compressed state is simply the sum $p_{k,\{i \vee j\}} = p_{k,i} + p_{k,j}$. Stated another way, part of the compression algorithm is to sum columns of probability numbers.

However, in general, transitions *from* a compressed state are more complicated to compute. Clearly, the probability of transitioning from the compressed state to some other state $p_{\{i \vee j\},k}$ must lie somewhere between $p_{i,k}$ and $p_{j,k}$, depending on how much time is spent in states i and j. Thus a weighted average of row entries appears to be called for, where the weights reflect the amount of time spent in states i and j. Precisely how to do this weighted average is investigated in section 3.

The algorithm for compressing two states i and j together is as follows:[2]

```
Compress-states($i$,$j$)
        (a) Compute a weighted average of the $i$th and $j$th rows.
            Place the results in rows $i$ and $j$.
        (b) Sum the $i$th and $j$th columns.
            Place the results in column $i$. Remove row $j$ and column $j$.
```

The compression algorithm has two steps. It takes as input a matrix Q_u (an uncompressed Q matrix). Step (a) averages the row entries, producing an intermediate row-averaged matrix Q_r. Step (b) sums column entries to produce the final compressed matrix Q_c. Step (a) is the sole source of error, since in general it is difficult to estimate the amount of time spent in states i and j.

Now that the compression algorithm has been outlined, it is important to define what is meant by "perfect" compression. As mentioned before, analysis of n-step transition probabilities (i.e., *transient* behavior of the Markov chain) can be realized by computing Q^n. For large Q matrices this is computationally expensive. It would be less expensive to compress Q and to then raise it to the nth power. If the compression algorithm has worked well, then the nth power of the compressed matrix Q_c should be (nearly) identical to compressing the nth power of the uncompressed matrix Q_u. In other words, perfect compression has occurred if $(Q_u^n)_c = Q_c^n$.

It turns out that there are two situations under which perfect compression can be obtained. The first situation is referred to as "row equivalence," in which the two states i and j have identical rows (i.e., $\forall k,\ p_{i,k} = p_{j,k}$). In this case the weighted averaging cannot produce any error, since the weights will be irrelevant. The second situation is referred to as "column equivalence," in which state i has column entries that are a real multiple q of the column entries for state j (i.e., $\forall k,\ p_{k,i} = qp_{k,j}$). The intuition here is that when this situation occurs, the ratio of time spent in state i to state j is precisely q. The details of this can be found in section 4.

However, for arbitrary matrices, compressing an arbitrarily chosen pair of states will not necessarily lead to good results. Thus, the goal is to identify pairs of states i and j upon which the above compression algorithm will work well. It turns out that pairs of states that are row or column *similar* are good candidates for compression. The justification for these measures will be provided in section 5.

At a high level, of course, this simple compression algorithm must be repeated

[1] The notation $p_{i,j}^{(n)} \equiv Q^n(i,j)$ denotes the entries of the n-step probability transition matrix Q^n.

[2] The algorithm is written this way because it makes it amenable to mathematical analysis.

for many pairs of states if one wants to dramatically reduce the size of a Q matrix. The high-level compression algorithm is simply:

```
Compress()
    Repeat as long as possible
        (i) Find the pair of states $i$ and $j$ most similar to each other
        (ii) Compress-states($i$,$j$)
```

3. The compression algorithm in more detail. In the previous section the compression algorithm was described in two steps. Step (a) is where error can occur, and care must be taken to mathematically justify the weighted averaging of rows. This can be done by attempting to force $(Q_u^2)_c$ to be as similar as possible to Q_c^2 (later sections will generalize this to higher powers). This is mathematically difficult, but fortunately it suffices to force Q_u^2 to be as similar as possible to Q_uQ_r, which is much simpler and focuses on the row-averaged matrix Q_r explicitly. The intuition behind this is that if compression is done correctly, passage through the new compressed state should affect the 2-step transition probabilities as little as possible.[3] This will be shown with a 4×4 Q matrix and then generalized to an arbitrary $N \times N$ matrix. The result will be the weighted row averaging procedure outlined earlier. This particular presentation has been motivated by a concern for comprehension and hence is not completely formal. A completely formal presentation is in the appendix.

3.1. Weighted averaging with a 4 × 4 matrix. Consider a general uncompressed 4×4 matrix Q_u for a Markov chain model of 4 states, as well as the general intermediate matrix Q_r:

$$Q_u = \begin{bmatrix} p_{1,1} & p_{1,2} & p_{1,3} & p_{1,4} \\ p_{2,1} & p_{2,2} & p_{2,3} & p_{2,4} \\ p_{3,1} & p_{3,2} & p_{3,3} & p_{3,4} \\ p_{4,1} & p_{4,2} & p_{4,3} & p_{4,4} \end{bmatrix}, \quad Q_r = \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} & r_{1,4} \\ r_{2,1} & r_{2,2} & r_{2,3} & r_{2,4} \\ r_{3,1} & r_{3,2} & r_{3,3} & r_{3,4} \\ r_{4,1} & r_{4,2} & r_{4,3} & r_{4,4} \end{bmatrix}.$$

The notation $r_{i,j} \equiv Q_r(i,j)$ is used to prevent confusion with the $p_{i,j}$ in Q_u. Without loss of generality, the goal will be to compress the 3rd and 4th states (rows and columns) of this matrix. Since the 3rd and 4th states are being compressed, rows 1 and 2 of Q_r must be the same as Q_u (i.e., averaging rows 3 and 4 will not affect rows 1 and 2). Denoting $\{3 \vee 4\}$ to be the compressed state, the intermediate matrix is

$$Q_r = \begin{bmatrix} p_{1,1} & p_{1,2} & p_{1,3} & p_{1,4} \\ p_{2,1} & p_{2,2} & p_{2,3} & p_{2,4} \\ r_{\{3\vee4\},1} & r_{\{3\vee4\},2} & r_{\{3\vee4\},3} & r_{\{3\vee4\},4} \\ r_{\{3\vee4\},1} & r_{\{3\vee4\},2} & r_{\{3\vee4\},3} & r_{\{3\vee4\},4} \end{bmatrix}.$$

The $r_{\{3\vee4\},k}$ represent the weighted average of rows 3 and 4 of Q_u. Recall that step (a) of Compress-states(3,4) will place that average in both rows 3 and 4, which is why rows 3 and 4 of Q_r are the same. The trick now is to determine what $r_{\{3\vee4\},1}$, $r_{\{3\vee4\},2}$, $r_{\{3\vee4\},3}$, and $r_{\{3\vee4\},4}$ should be in order to produce a reasonable compression. This is done by considering Q_u^2 and Q_uQ_r:

$$Q_u^2 = \begin{bmatrix} p_{1,1}^{(2)} & p_{1,2}^{(2)} & p_{1,3}^{(2)} & p_{1,4}^{(2)} \\ p_{2,1}^{(2)} & p_{2,2}^{(2)} & p_{2,3}^{(2)} & p_{2,4}^{(2)} \\ p_{3,1}^{(2)} & p_{3,2}^{(2)} & p_{3,3}^{(2)} & p_{3,4}^{(2)} \\ p_{4,1}^{(2)} & p_{4,2}^{(2)} & p_{4,3}^{(2)} & p_{4,4}^{(2)} \end{bmatrix}, \quad Q_uQ_r = \begin{bmatrix} a_{1,1}^{(2)} & a_{1,2}^{(2)} & a_{1,3}^{(2)} & a_{1,4}^{(2)} \\ a_{2,1}^{(2)} & a_{2,2}^{(2)} & a_{2,3}^{(2)} & a_{2,4}^{(2)} \\ a_{3,1}^{(2)} & a_{3,2}^{(2)} & a_{3,3}^{(2)} & a_{3,4}^{(2)} \\ a_{4,1}^{(2)} & a_{4,2}^{(2)} & a_{4,3}^{(2)} & a_{4,4}^{(2)} \end{bmatrix}.$$

[3] More formally, it can be shown that if $Q_u^2 = Q_uQ_r$, then $(Q_u^2)_c = Q_c^2$ for row- or column-equivalent situations; see section 4.

The notation $a_{i,j}^{(2)}$ is used to prevent confusion with the $p_{i,j}^{(2)}$ in Q_u^2. Since the goal is to have $Q_u^2 = Q_u Q_r$, it is necessary to have $p_{i,j}^{(2)}$ be as similar as possible to $a_{i,j}^{(2)}$. The $p_{i,j}^{(2)}$ values can be computed using $p_{i,j}$ values, while the $a_{i,j}^{(2)}$ values require the unknowns $r_{\{3\vee 4\},1}$, $r_{\{3\vee 4\},2}$, $r_{\{3\vee 4\},3}$, and $r_{\{3\vee 4\},4}$.

For example, $p_{1,1}^{(2)}$ can be computed by multiplying Q_u by itself:

$$p_{1,1}^{(2)} = p_{1,1}p_{1,1} + p_{1,2}p_{2,1} + p_{1,3}p_{3,1} + p_{1,4}p_{4,1}.$$

However, $a_{1,1}^{(2)}$ is computed by multiplying Q_u and Q_r:

$$a_{1,1}^{(2)} = p_{1,1}p_{1,1} + p_{1,2}p_{2,1} + (p_{1,3} + p_{1,4})r_{\{3\vee 4\},1}.$$

In the ideal situation we would like both of these to be equal. This implies that

$$r_{\{3\vee 4\},1} = \frac{p_{1,3}p_{3,1} + p_{1,4}p_{4,1}}{p_{1,3} + p_{1,4}}.$$

But we can write another formula for $r_{\{3\vee 4\},1}$ by considering $p_{2,1}^{(2)}$ and $a_{2,1}^{(2)}$:

$$p_{2,1}^{(2)} = p_{2,1}p_{1,1} + p_{2,2}p_{2,1} + p_{2,3}p_{3,1} + p_{2,4}p_{4,1},$$

$$a_{2,1}^{(2)} = p_{2,1}p_{1,1} + p_{2,2}p_{2,1} + (p_{2,3} + p_{2,4})r_{\{3\vee 4\},1}.$$

Again, we would like both of these to be equal. This implies that

$$r_{\{3\vee 4\},1} = \frac{p_{2,3}p_{3,1} + p_{2,4}p_{4,1}}{p_{2,3} + p_{2,4}}.$$

Similarly, consideration of $p_{3,1}^{(2)}$ and $a_{3,1}^{(2)}$ yields

$$r_{\{3\vee 4\},1} = \frac{p_{3,3}p_{3,1} + p_{3,4}p_{4,1}}{p_{3,3} + p_{3,4}},$$

while consideration of $p_{4,1}^{(2)}$ and $a_{4,1}^{(2)}$ yields

$$r_{\{3\vee 4\},1} = \frac{p_{4,3}p_{3,1} + p_{4,4}p_{4,1}}{p_{4,3} + p_{4,4}}.$$

What has happened here is that the four elements in the first column of $Q_u Q_r$ lead to four expressions for $r_{\{3\vee 4\},1}$. In general, all four expressions for $r_{\{3\vee 4\},1}$ can not hold simultaneously (although we will investigate conditions under which they will hold later). The best estimate is to take a weighted average of the four expressions for $r_{\{3\vee 4\},1}$ (this is related to the concept of "averaging" probabilities — see appendix for more details). This yields

$$r_{\{3\vee 4\},1} = \frac{(p_{1,3} + p_{2,3} + p_{3,3} + p_{4,3})p_{3,1} + (p_{1,4} + p_{2,4} + p_{3,4} + p_{4,4})p_{4,1}}{(p_{1,3} + p_{2,3} + p_{3,3} + p_{4,3}) + (p_{1,4} + p_{2,4} + p_{3,4} + p_{4,4})}.$$

Note how the final expression for $r_{\{3\vee 4\},1}$ is a weighted average of the row entries $p_{3,1}$ and $p_{4,1}$, where the weights are column sums for columns 3 and 4. In general the elements of $Q_u Q_r$ in the kth column will constrain $r_{\{3\vee 4\},k}$:

$$r_{\{3\vee 4\},k} = \frac{(p_{1,3} + p_{2,3} + p_{3,3} + p_{4,3})p_{3,k} + (p_{1,4} + p_{2,4} + p_{3,4} + p_{4,4})p_{4,k}}{(p_{1,3} + p_{2,3} + p_{3,3} + p_{4,3}) + (p_{1,4} + p_{2,4} + p_{3,4} + p_{4,4})}.$$

Once again, note how the expression for $r_{\{3\vee 4\},k}$ is a weighted average of the row entries $p_{3,k}$ and $p_{4,k}$, where the weights are column sums for columns 3 and 4.

3.2. Weighted averaging with an $N \times N$ matrix. The previous results for a 4×4 matrix can be extended to an $N \times N$ matrix. Without loss of generality, compress states $N-1$ and N. Then the N elements of column k yield N expressions for each $r_{\{N-1 \vee N\},k}$. The best estimate is (see appendix for details)

$$r_{\{N-1\vee N\},k} = \frac{(p_{1,N-1} + \cdots + p_{N,N-1})p_{N-1,k} + (p_{1,N} + \cdots + p_{N,N})p_{N,k}}{(p_{1,N-1} + \cdots + p_{N,N-1}) + (p_{1,N} + \cdots + p_{N,N})}.$$

Note again how the weights are column sums for columns $N-1$ and N. Generalizing this to compressing two arbitrary states i and j yields

$$r_{\{i\vee j\},k} = \frac{(\sum_l p_{l,i})p_{i,k} + (\sum_l p_{l,j})p_{j,k}}{\sum_l p_{l,i} + \sum_l p_{l,j}}$$

or

$$r_{\{i\vee j\},k} = \frac{m_i p_{i,k} + m_j p_{j,k}}{m_i + m_j}, \tag{3.1}$$

where m_i and m_j are the sums of the probability mass in columns i and j of Q_u.

Equation (3.1) indicates how to compute the $r_{\{i\vee j\},k}$ entries in Q_r. Note how they are computed using the weighted average of the row entries in rows i and j. The weights are simply the column sums. This justifies the row averaging component of the compression algorithm described in the previous section. Intuitively stated, the column mass for columns i and j provide good estimates of the relative amount of time spent in states i and j. The estimates are used as weights to average the transitions from i to state k and from j to k, producing the probability of transition from the combined state $\{i \vee j\}$ to k.

3.3. Mathematical restatement of the compression algorithm. Now that the weighted averaging of rows i and j has been explained, it is only necessary to sum columns i and j in order to complete the compression algorithm. The whole algorithm can be expressed simply as follows. Assume that two states have been chosen for compression. Let S denote the set of all N states, and let the nonempty sets $S_1, \ldots, S_{N-1}$ partition S such that one S_i contains the two chosen states while each other S_i is composed of exactly one state. Let m_i denote the column mass of state i. Then the compressed matrix Q_c is

$$Q_c(x,y) = \frac{1}{\sum_{i\in S_x} m_i} \sum_{i \in S_x} \left[m_i \sum_{j\in S_y} p_{i,j} \right]. \tag{3.2}$$

This corresponds to taking a weighted average of the two rows corresponding to the two chosen states while summing the two corresponding columns. The other entries in the Q matrix remain unchanged. Consider an example in which states 2 and 3 are compressed. In that case $S_1 = \{1\}$ and $S_2 = \{2,3\}$. Q_c is described by

$$\begin{aligned} Q_c(1,1) &= p_{1,1}, \\ Q_c(1,2) &= p_{1,2} + p_{1,3}, \\ Q_c(2,1) &= \tfrac{1}{m_2+m_3}[m_2 p_{2,1} + m_3 p_{3,1}], \\ Q_c(2,2) &= \tfrac{1}{m_2+m_3}[m_2(p_{2,2}+p_{2,3}) + m_3(p_{3,2}+p_{3,3})]. \end{aligned}$$

Applying this to the following column equivalent matrix Q_u produces perfect results ($(Q_u^2)_c = Q_c^2$):

$$Q_u = \begin{bmatrix} .7 & .1 & .2 \\ .4 & .2 & .4 \\ .1 & .3 & .6 \end{bmatrix} \Rightarrow Q_u^2 = \begin{bmatrix} .55 & .15 & .30 \\ .40 & .20 & .40 \\ .25 & .25 & .50 \end{bmatrix} \Rightarrow (Q_u^2)_c = \begin{bmatrix} .55 & .45 \\ .30 & .70 \end{bmatrix},$$

$$Q_c = \begin{bmatrix} .7 & .3 \\ .2 & .8 \end{bmatrix} \Rightarrow Q_c^2 = \begin{bmatrix} .55 & .45 \\ .30 & .70 \end{bmatrix}.$$

In summary, this section has justified the use of column mass as weights in the row averaging portion of the compression algorithm. The whole compression algorithm is stated succinctly as a mathematical function which can compress any arbitrary pair of states. However, as stated earlier, compression of arbitrary pairs of states need not lead to good compression. The goal, then, is to identify such states. This is investigated in the next section and relies upon the concepts of row and column equivalence.

4. Special cases in which compression is perfect. If compression is working well, then the compressed version of Q_u^n should be (nearly) identical to Q_c^n. As suggested in section 2, there are two situations under which perfect compression will occur. The first situation is when two states are row equivalent. The intuition here is that the row average of two identical rows will not involve any error, and thus the compression will be perfect. The second situation is when two states are column equivalent. The intuition for this situation is that if the column $\mathbf{c}_i$ is equal to $q\mathbf{c}_j$, then the ratio of time spent in state i to state j is exactly q. Under these circumstances the weighted row average will also produce no error.

This section will prove that $(Q_u^n)_c = Q_c^n$ when the two states being compressed are either row equivalent or column equivalent. This will hold for any n and for any Q_u matrix of size $N \times N$. The method of proof will be to treat the compression algorithm as a linear transformation f and then to show that $f(Q_u^n) = (f(Q_u))^n$, where $f(Q_u) = Q_c$.

4.1. Row equivalence and the compression algorithm. This subsection will prove that when two states are row equivalent, compression of those states can be described by a linear transformation (matrix multiplication). The compression algorithm compresses an $N \times N$ matrix Q_u to an $(N-1) \times (N-1)$ matrix Q_c. However, for the sake of mathematical convenience, all of the matrix transformations will be with $N \times N$ matrices. Without loss of generality, it is assumed that states $N-1$ and N are being compressed. When it comes time to express the final compression, the Nth row and column will simply be ignored, producing the $(N-1) \times (N-1)$ compressed matrix. The "$\bullet$" notation is used to denote entries that are not important for the derivation.

Assume that states $N-1$ and N are row equivalent. Thus $\forall k,\ p_{N-1,k} = p_{N,k}$. Using (3.1) to compute the row averages yields

$$r_{\{N-1 \vee N\},k} = \frac{m_{N-1}p_{N-1,k} + m_N p_{N,k}}{m_{N-1} + m_N} = \frac{p_{N-1,k}(m_{N-1} + m_N)}{m_{N-1} + m_N} = p_{N-1,k},$$

and the compressed matrix should have the form

$$Q_c = \begin{bmatrix} p_{1,1} & \cdots & p_{1,N-2} & p_{1,N-1} + p_{1,N} \\ p_{2,1} & \cdots & p_{2,N-2} & p_{2,N-1} + p_{2,N} \\ \vdots & & \vdots & \vdots \\ p_{N-1,1} & \cdots & p_{N-1,N-2} & p_{N-1,N-1} + p_{N-1,N} \end{bmatrix}.$$

Theorem 4.1. *If states N and $N-1$ in Q_u are row equivalent, then $Q_c = TQ_uT$ and $TT = I$, where*

$$T = \left[\begin{array}{c|c} I & 0 \\ \hline 0 & \begin{matrix} 1 & 0 \\ 1 & -1 \end{matrix} \end{array}\right].$$

Proof. $Q_c = TQ_uT$ can be expressed as follows:

$$Q_c = T \begin{bmatrix} p_{1,1} & \cdots & p_{1,N-2} & p_{1,N-1} & p_{1,N} \\ p_{2,1} & \cdots & p_{2,N-2} & p_{2,N-1} & p_{2,N} \\ \vdots & & \vdots & \vdots & \vdots \\ p_{N-1,1} & \cdots & p_{N-1,N-2} & p_{N-1,N-1} & p_{N-1,N} \\ p_{N,1} & \cdots & p_{N,N-2} & p_{N,N-1} & p_{N,N} \end{bmatrix} \left[\begin{array}{c|c} I & 0 \\ \hline 0 & \begin{matrix} 1 & 0 \\ 1 & -1 \end{matrix} \end{array}\right]$$

$$= \left[\begin{array}{c|c} I & 0 \\ \hline 0 & \begin{matrix} 1 & 0 \\ 1 & -1 \end{matrix} \end{array}\right] \begin{bmatrix} p_{1,1} & \cdots & p_{1,N-2} & p_{1,N-1} + p_{1,N} & \bullet \\ p_{2,1} & \cdots & p_{2,N-2} & p_{2,N-1} + p_{2,N} & \bullet \\ \vdots & & \vdots & \vdots & \vdots \\ p_{N-1,1} & \cdots & p_{N-1,N-2} & p_{N-1,N-1} + p_{N-1,N} & \bullet \\ p_{N-1,1} & \cdots & p_{N-1,N-2} & p_{N-1,N-1} + p_{N-1,N} & \bullet \end{bmatrix}$$

$$= \begin{bmatrix} p_{1,1} & \cdots & p_{1,N-2} & p_{1,N-1} + p_{1,N} & \bullet \\ p_{2,1} & \cdots & p_{2,N-2} & p_{2,N-1} + p_{2,N} & \bullet \\ \vdots & & \vdots & \vdots & \vdots \\ p_{N-1,1} & \cdots & p_{N-1,N-2} & p_{N-1,N-1} + p_{N-1,N} & \bullet \\ \bullet & \cdots & \bullet & \bullet & \bullet \end{bmatrix}.$$

This is precisely what Q_c should be. Thus the compression of two row-equivalent states can be expressed simply as TQ_uT. The first T performs row averaging (which is trivial) and the second T performs column summing. The reader will also note that some elements of T do not appear to be important for the derivation that $Q_c = TQ_uT$. This is true; however, the purpose of these elements is to ensure that $TT = I$, since this fact will also be used to help prove that $(Q_u^n)_c = Q_c^n$:

$$TT = \left[\begin{array}{c|c} I & 0 \\ \hline 0 & \begin{matrix} 1 & 0 \\ 1 & -1 \end{matrix} \end{array}\right] \left[\begin{array}{c|c} I & 0 \\ \hline 0 & \begin{matrix} 1 & 0 \\ 1 & -1 \end{matrix} \end{array}\right] = I.$$

4.2. Column equivalence and the compression algorithm. This subsection will prove that when two states are column equivalent, compression of those states can be described by a linear transformation. Assume without loss of generality that states $N-1$ and N are column equivalent. Thus $\forall k,\ p_{k,N-1} = qp_{k,N}$, and $m_{N-1} = qm_N$. Using (3.1) to compute the row averages yields

$$r_{\{N-1 \vee N\},k} = \frac{m_{N-1}p_{N-1,k} + m_N p_{N,k}}{m_{N-1} + m_N} = \frac{qp_{N-1,k} + p_{N,k}}{q+1},$$

and the compressed matrix should have the form

$$Q_c = \begin{bmatrix} p_{1,1} & \cdots & p_{1,N-2} & p_{1,N-1} + p_{1,N} \\ p_{2,1} & \cdots & p_{2,N-2} & p_{2,N-1} + p_{2,N} \\ \vdots & & \vdots & \vdots \\ \frac{qp_{N-1,1}+p_{N,1}}{q+1} & \cdots & \frac{qp_{N-1,N-2}+p_{N,N-2}}{q+1} & \frac{qp_{N-1,N-1}+p_{N,N-1}+qp_{N-1,N}+p_{N,N}}{q+1} \end{bmatrix}.$$

Theorem 4.2. *If states N and $N-1$ in Q_u are column equivalent, then $Q_c = XQ_uY$ and $YX = I$, where*

$$Y = \left[\begin{array}{c|c} I & 0 \\ \hline 0 & \begin{matrix} 1 & \frac{1}{q} \\ 1 & -1 \end{matrix} \end{array}\right], \quad X = \left[\begin{array}{c|c} I & 0 \\ \hline 0 & \begin{matrix} \frac{q}{q+1} & \frac{1}{q+1} \\ \frac{q}{q+1} & -\frac{q}{q+1} \end{matrix} \end{array}\right].$$

Proof. $Q_c = XQ_uY$ can be expressed as follows:

$$Q_c = X \begin{bmatrix} p_{1,1} & \cdots & p_{1,N-2} & p_{1,N-1} & p_{1,N} \\ p_{2,1} & \cdots & p_{2,N-2} & p_{2,N-1} & p_{2,N} \\ \vdots & & \vdots & \vdots & \vdots \\ p_{N-1,1} & \cdots & p_{N-1,N-2} & p_{N-1,N-1} & p_{N-1,N} \\ p_{N,1} & \cdots & p_{N,N-2} & p_{N,N-1} & p_{N,N} \end{bmatrix} \left[\begin{array}{c|cc} I & \multicolumn{2}{c}{0} \\ \hline 0 & 1 & \frac{1}{q} \\ & 1 & -1 \end{array}\right]$$

$$= \left[\begin{array}{c|cc} I & \multicolumn{2}{c}{0} \\ \hline 0 & \frac{q}{q+1} & \frac{1}{q+1} \\ & \frac{q}{q+1} & -\frac{q}{q+1} \end{array}\right] \begin{bmatrix} p_{1,1} & \cdots & p_{1,N-2} & p_{1,N-1} + p_{1,N} & \bullet \\ p_{2,1} & \cdots & p_{2,N-2} & p_{2,N-1} + p_{2,N} & \bullet \\ \vdots & & \vdots & \vdots & \vdots \\ p_{N-1,1} & \cdots & p_{N-1,N-2} & p_{N-1,N-1} + p_{N-1,N} & \bullet \\ p_{N,1} & \cdots & p_{N,N-2} & p_{N,N-1} + p_{N,N} & \bullet \end{bmatrix}$$

$$= \begin{bmatrix} p_{1,1} & \cdots & p_{1,N-2} & p_{1,N-1} + p_{1,N} & \bullet \\ p_{2,1} & \cdots & p_{2,N-2} & p_{2,N-1} + p_{2,N} & \bullet \\ \vdots & & \vdots & \vdots & \vdots \\ \frac{qp_{N-1,1}+p_{N,1}}{q+1} & \cdots & \frac{qp_{N-1,N-2}+p_{N,N-2}}{q+1} & \frac{qp_{N-1,N-1}+p_{N,N-1}+qp_{N-1,N}+p_{N,N}}{q+1} & \bullet \\ \bullet & \cdots & \bullet & \bullet & \bullet \end{bmatrix}.$$

This is precisely what Q_c should be. Thus the compression of two column-equivalent states can be expressed simply as XQ_uY. X performs row averaging and Y performs column summing. The reader will note that some elements of X and Y are not important for the derivation that $Q_c = XQ_uY$ (e.g., T could be used instead of Y). This is true; however, the purpose of these elements is to ensure that $YX = I$, since this fact will be used to help prove that $(Q_u^n)_c = Q_c^n$ at the end of this section:

$$YX = \left[\begin{array}{c|cc} I & \multicolumn{2}{c}{0} \\ \hline 0 & 1 & \frac{1}{q} \\ & 1 & -1 \end{array}\right] \left[\begin{array}{c|cc} I & \multicolumn{2}{c}{0} \\ \hline 0 & \frac{q}{q+1} & \frac{1}{q+1} \\ & \frac{q}{q+1} & -\frac{q}{q+1} \end{array}\right] = I.$$

4.3. Some necessary lemmas. Before proving that $(Q_u^n)_c = Q_c^n$ for row- or column-equivalent states, it is necessary to prove some simple lemmas. The idea is to show that if Q_u is row or column equivalent, so is Q_u^n. This will allow the previous linear transformations to be applied to Q_u^n as well as Q_u.

Let square matrices A and B be defined as matrices of row and column vectors, respectively:

$$A = \begin{bmatrix} a_{1,1} & \cdots & a_{1,N} \\ \vdots & & \vdots \\ a_{N,1} & \cdots & a_{N,N} \end{bmatrix} = \begin{bmatrix} \mathbf{a}_1 \\ \vdots \\ \mathbf{a}_N \end{bmatrix},$$

$$B = \begin{bmatrix} b_{1,1} & \cdots & b_{1,N} \\ \vdots & & \vdots \\ b_{N,1} & \cdots & b_{N,N} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 & \cdots & \mathbf{b}_N \end{bmatrix}.$$

Then the matrix product AB can be represented using dot product notation:

$$AB = \begin{bmatrix} \mathbf{a}_1 \cdot \mathbf{b}_1 & \cdots & \mathbf{a}_1 \cdot \mathbf{b}_N \\ \vdots & & \vdots \\ \mathbf{a}_N \cdot \mathbf{b}_1 & \cdots & \mathbf{a}_N \cdot \mathbf{b}_N \end{bmatrix}.$$

LEMMA 4.3. *Row equivalence is invariant under postmultiplication.*

Proof. Suppose states i and j of A are row equivalent ($\mathbf{a}_i = \mathbf{a}_j$). Then $\forall k$, $\mathbf{a}_i \cdot \mathbf{b}_k = \mathbf{a}_j \cdot \mathbf{b}_k$. So, states i and j in AB must be row equivalent.

LEMMA 4.4. *Column equivalence is invariant under premultiplication.*

Proof. Suppose states i and j of B are column equivalent ($\mathbf{b}_i = q\mathbf{b}_j$). Then $\forall k,\ \mathbf{a}_k \cdot \mathbf{b}_i = q\mathbf{a}_k \cdot \mathbf{b}_j$. So, states i and j in AB must be column equivalent.

LEMMA 4.5. *Row and column equivalence are invariant under raising to a power.*

Proof. $Q^n = QQ^{n-1}$. Thus, if states i and j are row equivalent in Q, they are row equivalent in Q^n by Lemma 4.3. Similarly, $Q^n = Q^{n-1}Q$. Thus, if states i and j are column equivalent in Q, they are column equivalent in Q^n by Lemma 4.4.

Lemma 4.5 indicates that the previous linear transformations can be applied to Q_u^n to produce $(Q_u^n)_c$ when two states in Q_u are row or column equivalent.

4.4. Theorems for perfect compression. Given the previous theorems concerning the linear transformations and Lemma 4.5, it is now possible to state and prove the theorems for perfect compression. The Q matrix can be considered to be Q_u in these theorems.

THEOREM 4.6. *If Q is row equivalent, then $Q^n = QQ_r^{n-1}$ implies $(Q^n)_r = Q_r^n$, and $(Q^n)_c = Q_c^n$.*

Proof. If Q is row equivalent, then so is Q^n by Lemma 4.5. If $Q^n = QQ_r^{n-1}$, then $(Q^n)_r = (QQ_r^{n-1})_r = TQQ_r^{n-1} = Q_r^n$, and $(Q^n)_c = (QQ_r^{n-1})_c = T(QQ_r^{n-1})T = TQQ_c^{n-1} = TQTTQ_c^{n-1} = Q_cTQ_c^{n-1} = Q_c^n$.

THEOREM 4.7. *If Q is column equivalent, then $Q^n = QQ_r^{n-1}$ implies $(Q^n)_r = Q_r^n$, and $(Q^n)_c = Q_c^n$.*

Proof. If Q is column equivalent, then so is Q^n by Lemma 4.5. If $Q^n = QQ_r^{n-1}$, then $(Q^n)_r = (QQ_r^{n-1})_r = XQQ_r^{n-1} = Q_r^n$, and $(Q^n)_c = (QQ_r^{n-1})_c = X(QQ_r^{n-1})Y = XQQ_c^{n-1} = XQTTQ_c^{n-1} = Q_cTQ_c^{n-1} = Q_c^n$.

These two theorems illustrate the validity of trying to force Q_u^2 to be as similar as possible to Q_uQ_r in section 3.

THEOREM 4.8. *If Q is row equivalent, then $(Q^n)_c = Q_c^n$.*

Proof. If Q is row equivalent, then so is Q^n by Lemma 4.5. Then $(Q^n)_c = TQ^nT = TQ \cdots QT$. Since $TT = I$, then $(Q^n)_c = TQTTQ \cdots QTTQT = Q_c^n$.

THEOREM 4.9. *If Q is column equivalent, then $(Q^n)_c = Q_c^n$.*

Proof. If Q is column equivalent, then so is Q^n by Lemma 4.5. Then $(Q^n)_c = XQ^nY = XQ \cdots QY$. Since $YX = I$, then $(Q^n)_c = XQYXQ \cdots QYXQY = Q_c^n$.

These theorems hold for all n and for all row- or column-equivalent $N \times N$ Q matrices and highlight the importance of row and column equivalence. If two states are row or column equivalent, then compression of those two states is perfect (i.e., $(Q^n)_c = Q_c^n$).

5. Error analysis and a similarity metric. The previous sections have explained how to merge pairs of states and have explained that row- or column-equivalent pairs will yield perfect compression. Of course, it is highly unlikely that pairs of states will be found that are perfectly row equivalent or column equivalent. The goal then is to find a similarity metric that measures the row and column similarity (i.e., how close pairs of states are to being row or column equivalent). If the metric is formed correctly, those pairs of states that are more similar should yield less error when compressed. This section will derive an expression for error and then use this as a similarity metric for pairs of states.

We will use Q_uQ_r and Q_u^2 to estimate error. As mentioned before, it is desirable to have the entries in those two matrices be as similar as possible. Consider compressing two states i and j. Then the entries in Q_u^2 are

$$p_{x,y}^{(2)} = p_{x,i}p_{i,y} + p_{x,j}p_{j,y} + \sum_{k \neq i,j} p_{x,k}p_{k,y}.$$

The entries in Q_uQ_r are

$$a^{(2)}_{x,y} = (p_{x,i} + p_{x,j})r_{\{i\vee j\},y} + \sum_{k\neq i,j} p_{x,k}p_{k,y}.$$

Then the error associated with the (x, y)th element of Q_uQ_r is

$$Error_{i,j}(x, y) = a^{(2)}_{x,y} - p^{(2)}_{x,y} = (p_{x,i} + p_{x,j})r_{\{i\vee j\},y} - p_{x,i}p_{i,y} - p_{x,j}p_{j,y}.$$

Using (3.1) for $r_{\{i\vee j\},k}$ (and substituting y for k) yields

$$Error_{i,j}(x, y) = (p_{x,i} + p_{x,j})\left[\frac{m_ip_{i,y} + m_jp_{j,y}}{m_i + m_j}\right] - p_{x,i}p_{i,y} - p_{x,j}p_{j,y}.$$

Now denote $\alpha_{i,j}(y) = p_{i,y} - p_{j,y}$. This is a measure of the row similarity for rows i and j at column y (and will be explained further below). Then

$$Error_{i,j}(x, y) = (p_{x,i}+p_{x,j})\left[\frac{m_i(p_{j,y} + \alpha_{i,j}(y))+m_jp_{j,y}}{m_i+m_j}\right]-p_{x,i}(p_{j,y}+\alpha_{i,j}(y))-p_{x,j}p_{j,y}.$$

This simplifies to

$$Error_{i,j}(x, y) = \frac{(m_ip_{x,j} - m_jp_{x,i})\alpha_{i,j}(y)}{m_i + m_j}.$$

Denote $\beta_{i,j}(x) = (m_ip_{x,j} - m_jp_{x,i})/(m_i + m_j)$. Then

$$Error_{i,j}(x, y) = \beta_{i,j}(x)\alpha_{i,j}(y).$$

Now $\beta_{i,j}(x)$ can be considered to be a measure of column similarity for columns i and j at row x (this will be shown more explicitly further down). Since only the magnitude of the error is important, and not the sign, the absolute value of the error should be considered:

$$|Error_{i,j}(x, y)| = |\beta_{i,j}(x)\alpha_{i,j}(y)|.$$

Recall that $Error_{i,j}(x, y)$ is the error associated with the (x, y)th element of Q_uQ_r if states i and j are compressed. The total error of the whole matrix is

$$Error_{i,j} = \sum_x\sum_y |Error_{i,j}(x, y)| = \sum_x\sum_y |\beta_{i,j}(x)\alpha_{i,j}(y)|.$$

But this can be simplified to

$$Error_{i,j} = \left(\sum_x |\beta_{i,j}(x)|\right)\left(\sum_y |\alpha_{i,j}(y)|\right).$$

To understand this equation consider the situation where states i and j are row equivalent. Then $\forall y,\ p_{i,y} = p_{j,y}$. This indicates that $\forall y,\ \alpha_{i,j}(y) = 0$ and $Error_{i,j} = 0$. Thus there is no error associated with compressing row-equivalent states i and j, as has been shown in earlier sections.

Consider the situation where states i and j are column equivalent. Then $\forall x,\ p_{x,i} = qp_{x,j}$ and $m_i = qm_j$. It is trivial to show that $\forall x,\ \beta_{i,j}(x) = 0$, and as a consequence,

$Error_{i,j} = 0$. Thus there is no error associated with compressing column-equivalent states i and j, as has been shown in earlier sections.

Given this, a natural similarity metric is the expression for error:

$$Similarity_{i,j} = \left(\sum_x |\beta_{i,j}(x)| \right) \left(\sum_y |\alpha_{i,j}(y)| \right). \tag{5.1}$$

If the similarity is close to zero, then error is close to zero, and pairs of states can be judged as to the amount of error that will ensue if they are compressed.[4] The compression algorithm can now be written as follows:

```
Compress()
    Repeat as long as possible
        (i) Find pair of states $i$ and $j$ such that $Similarity_{i,j} < \epsilon$
        (ii) Compress-states($i$,$j$)
```

The role of ϵ is as a threshold. Pairs of states that are more similar than this threshold can be compressed. By raising ϵ one can compress more states, but with a commensurate increase in error.

The paper thus far has fully outlined the compression algorithm for pairs of states and identified situations under which compression is perfect — namely, when the pairs of states are row or column equivalent. By performing an error analysis, a natural measure of similarity was derived in which pairs of states that are row or column similar yield small amounts of error in the compression algorithm. The following section outlines some experiments showing the degree of compression that can be achieved in practice.

6. Some experiments. In order to evaluate the practicality of the compression algorithm, it was tested on some Markov chains derived from the field of genetic algorithms (GAs). In a GA, a population of individuals evolves generation by generation via Darwinian selection and perturbation operators such as recombination and mutation. Each individual in the population can be considered to be a point in a search space (see [8] for an overview of GAs).

Each different population of the GA is a state in the Markov chain, and $p_{i,j}$ is the probability that the GA will evolve from one population i to another j in one generation (time step). The number of states grows extremely fast as the size of the population increases and as the size of individuals increases. The details of the mapping of GAs to Markov chains can be found in [6]. Their use in examining transient behavior can be found in [2].[5]

6.1. Accuracy experiments. The first set of experiments examines the accuracy of the compressed Markov chains by using both Q_u^n and Q_c^n to compute the probability distribution $p^{(n)}$ over the states at time n. To answer such questions, Q_u^n must be combined with a set of initial conditions concerning the GA at generation 0. Thus, the a priori probability of the GA being in state i at time 0 is $p_i^{(0)}$.[6] Given this, the probability that the GA will be in a particular state j at time n is

$$p_j^{(n)} = \sum_i p_i^{(0)} \, p_{i,j}^{(n)}.$$

[4] It is useful to think of this as a "dissimilarity" metric.

[5] For the GA, Q has no zero entries and is thus ergodic.

[6] If states i and j have been compressed, then $p_{\{i \vee j\}}^{(0)} = p_i^{(0)} + p_j^{(0)}$.

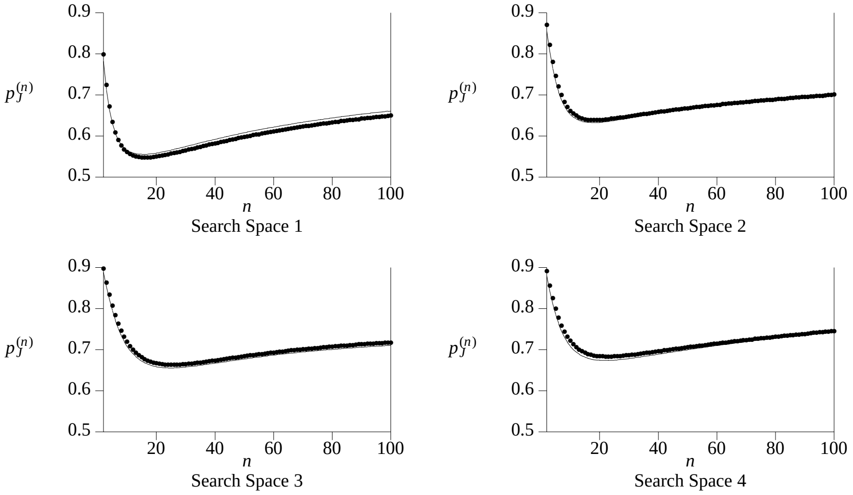


FIG. 6.1. $p_J^{(n)}$*, where* ϵ *is* 0.0 *and* 0.15 *for* $N = 455$*. The bold curves represent the exact values, while the nonbold curves represent the values computed from the compressed matrix.*

It is also possible to compute probabilities over a set of states. Define a predicate $Pred_J$ and the set J of states that make $Pred_J$ true. Then the probability that the GA will be in one of the states of J at time n is

$$p_J^{(n)} = \sum_{j \in J} p_j^{(n)}.$$

In this paper, J represents the set of all states which contain at least one copy of the optimum (i.e., the set of all populations which have at least one individual with the optimum function value). The Markov model is used to compute $p_J^{(n)}$, the probability of having at least one copy of the optimum in the population at time n.

The compression algorithm can thus be evaluated by using both Q_u^n (ground truth) and Q_c^n (the estimate) to compute $p_J^{(n)}$ for different values of n. The closer the estimate is to ground truth, the better the compression algorithm is working.

Since the goal is to compute probabilities involving states containing the optimum (the J set), J states should not be compressed with non-J states. Consequently, the compression algorithm is run separately for both sets of states. The algorithm is

```
Repeat until no new compressed states are created
    (a) For each state $i$ in the $J$ set of the current compressed model
        (i) find the most similar state $j$ in the $J$ set;
        (ii) if $Similarity_{i,j} < \epsilon$, Compress-states($i$,$j$).
    (b) For each state $i$ in the non-$J$ set of the current compressed model
        (i) find the most similar state $j$ in the non-$J$ set;
        (ii) if $Similarity_{i,j} < \epsilon$, Compress-states($i$,$j$).
```

In theory this compression algorithm could result in a two state model involving just J and non-J. In practice this would require large values of ϵ and unacceptable

TABLE 6.1
The percentage of states removed when $\epsilon = 0.15$.

	$N = 286$	$N = 455$	$N = 680$	$N = 969$
Search space 1	85%	88%	90%	92%
Search space 2	71%	76%	81%	84%
Search space 3	65%	73%	79%	82%
Search space 4	64%	73%	79%	82%

error in $p_J^{(n)}$ computations.

Four different search spaces were chosen for the GA. This particular set of four search spaces was chosen because experience has shown that it is hard to get a single compression algorithm to perform well on all. Also, in order to see how well the compression algorithm scales to larger Markov chains, four population sizes were chosen for the GA (10, 12, 14, and 16). These four choices of population size produced Markov chains of 286, 455, 680, and 969 states, respectively. Thus, the compression algorithm was tested on 16 different Markov chains.[7]

Naturally, the setting of ϵ is crucial to the success of the experiments. Experiments indicated that a value of 0.15 yielded good compression with minimal error for all 16 Markov chains. The results for $N = 455$ are shown in Figure 6.1. The results for the other experiments are omitted for the sake of brevity, but they are almost identical. The values $p_J^{(n)}$ are computed for n ranging from 2 to 100, for both the compressed and uncompressed Markov chains, and graphed as curves. The bold curves represent the exact $p_J^{(n)}$ values, while the nonbold curves represent the values computed from the compressed matrix.

The figures clearly indicate that the compressed matrix is yielding negligible error. To see how the amount of compression is affected by the size of the Markov chain, consider Table 6.1, which gives the percentage of states removed for each of the 16 chains. What is interesting is that, for these particular search spaces, the amount of compression is increasing as N increases (while still yielding negligible error). For $N = 969$, over 80% of the states have been removed, yielding Q_c matrices roughly 3% the size (in terms of memory requirements) of the original Q_u matrix. It is also interesting to note that different search spaces are consistently compressed to different degrees. For example, the third and fourth search spaces are consistently compressed less than the first search space. Further investigation into the nature of these search spaces may help characterize when arbitrary Markov chains are hard/easy to compress with this algorithm.

6.2. Timing experiments. It is now necessary to examine the computational cost of the compression algorithm. Our prior work, [2] and [9], focused heavily on the insights gained by actually examining Q_u^n, which involved computations on the order of N^3 (to multiply Q_u repeatedly). Thus the primary motivation for producing the compression algorithm was to gain the same insights more efficiently by dramatically reducing N. Since the second search space is quite representative in terms of the performance of the compression algorithm, we draw our timing results from the experiments with that particular search space. Table 6.2 gives the amount of CPU time (in minutes) needed to compute Q_u^n as n ranges from 2 to 100. Table 6.3 gives the amount of time needed to compress Q_u to Q_c as well as the time needed to compute

[7] See [2] for a definition of these search spaces.

TABLE 6.2
The time (in minutes) to compute Q_u^n for $n = 2$ to $n = 100$.

	$N = 286$	$N = 455$	$N = 680$	$N = 969$
Computation time	27	125	447	1289

TABLE 6.3
The time (in minutes) to compress Q_u and to compute Q_c^n for $n = 2$ to $n = 100$.

	$N = 286$	$N = 455$	$N = 680$	$N = 969$
Compression time	0.2	0.9	3.0	9.5
Computation time	2.4	7.6	17.9	38.1

Q_c^n as n ranges from 2 to 100.[8] Clearly, the compression algorithm achieves enormous savings in time when it is actually necessary to compute powers of Q_u.

Another common use of Q_u^n is to compute the probability distribution $p^{(n)}$ over the states at time n (as we did in the previous subsection). If the prior distribution $p^{(0)}$ is known in advance, however, this is more efficiently done by multiplying $p^{(0)}$ by Q_u repeatedly (i.e., this is repeated n times to produce $p^{(n)}$). The computation is of order N^2 instead of N^3.

Tables 6.4 and 6.5 give the amount of time needed to compute $p^{(n)}$ (from Q_u and Q_c, respectively). Despite the obvious benefits of computing $p^{(n)}$ from Q_c, the compression algorithm is not advantageous in this case since the time needed to compress Q_u exceeds the time to produce $p^{(n)}$ from Q_u. However, there are still occasions when compressing Q_u and then using Q_c to compute $p^{(n)}$ will in fact be more efficient. The first is when it is necessary to compute $p^{(n)}$ for a large number of different prior distributions (recall that Q_c does not depend on the prior information and hence need not be recomputed). The second occasion is when it is necessary to compute $p^{(n)}$ for large n (e.g., [10] indicates that times on the order of 10^8 are sometimes required). In both of these situations the cost of the compression algorithm is amortized. Finally, compression is also advantageous when the prior distribution is not known in advance.[9]

In summary, the compression algorithm is most advantageous when it is necessary to actually examine the powers of Q_u directly. For computing probability distributions over the states, the compression algorithm will be advantageous if the prior distribution is initially unknown, if a large number of prior distributions will be considered, or if the transient behavior over a long period of time is required.

7. Related work. The goal of this paper has been to provide a technique for compressing (or *aggregating*) discrete-time Markov chains (DTMCs) in a way that yields good estimates of the transient behavior of the Markov model. This section summarizes the work that is most closely related.

There is a considerable body of literature concerning the approximation of transient behavior in Markov chains. Techniques include the computation of matrix exponentials, the use of ordinary differential equations, and Krylov subspace methods [7]. However, all of these techniques are for continuous-time Markov chains (CTMCs), which use an infinitesimal generator matrix instead of a probability transition matrix. It is possible to discretize a CTMC to obtain a DTMC such that the stationary

[8] All timing results are on a Sun Sparc 20. The code is written in C and is available from the author.

[9] It is also important to emphasize that it is very likely that the compression algorithm can be extensively optimized, producing much better timing results.

Table 6.4
The time (in minutes) to compute $p^{(n)}$ for $n = 2$ to $n = 100$.

	$N = 286$	$N = 455$	$N = 680$	$N = 969$
Computation time	0.1	0.3	0.7	1.4

Table 6.5
The time (in minutes) to compress Q_u and to compute $p^{(n)}$ for $n = 2$ to $n = 100$.

	$N = 286$	$N = 455$	$N = 680$	$N = 969$
Compression time	0.2	0.9	3.0	9.5
Computation time	0.02	0.02	0.03	0.05

probability vector of the CTMC is identical to that of the DTMC. However, [10] notes that the transient solutions of DTMCs are not the same as those of the corresponding CTMCs, indicating that these techniques will be problematic for computing the transient behavior of DTMCs.

There is also considerable work in aggregation of DTMCs. Almost all theoretical analyses of aggregation (e.g., "block aggregation" [5]) utilize the same functional form:

$$f(Q_u) \;=\; Q_c \;=\; AQ_uB \quad (AB \;=\; I),$$

where A and B are matrices that determine the partitioning and the aggregation of the states [3], [4]. This functional form must satisfy two axioms: "linearity" and "state partitioning." Linearity implies that A and B do not depend explicitly on the entries in Q_u. State partitioning implies that the "aggregated" transition probabilities should depend only upon the probabilities associated with the aggregated states (e.g., the aggregation of states i and j should only depend on $p_{i,i}$, $p_{i,j}$, $p_{j,i}$, and $p_{j,j}$).

Neither axiom is true for compression of column-equivalent states in this paper. This is reflected in the fact that, in general, $AB = XY \neq I$. Instead, in this paper $BA = I$ for both row and column equivalence, yielding desirable properties with respect to the powers of Q_u. The current results indicate that the relevance of both axioms should be reexamined.

The aggregation technique most closely related to the work in this paper is described by [10], [11], and [12]. This aggregation technique partitions the set of states S into s nonempty sets $S_1, \ldots, S_s$. Denoting the steady-state probability of state i as π_i, then $\pi_y = \sum_{i \in S_y} \pi_i$ if

$$Q_c(x, y) = \frac{1}{\sum_{i \in S_x} \pi_i} \sum_{i \in S_x} \left[\pi_i \sum_{j \in S_y} p_{i,j} \right]. \tag{7.1}$$

If compression is performed in this manner, the steady-state behavior of the compressed system is the same as the original system. The aggregated matrix can be computed via the method of "stochastic complementation" or via "iterative aggregation/disaggregation" methods. The former will work on arbitrary matrices but is generally computationally expensive. The latter is most efficient for "nearly completely decomposable" (NCD) matrices (e.g., see [1]). However, the emphasis is always on steady-state behavior and not on transient behavior. This difference in emphasis can be seen by noting the difference in the choice of weights—the focus in this paper has been on column mass instead of steady-state values.

In a sense, the compression algorithm presented in this paper is a generalization of steady-state aggregation. The steady-state matrix is column equivalent for every pair of states, and the column masses, when renormalized, are the same as the steady-state probabilities. Thus the compression algorithm is a generalization of the aggregation formula to transient behavior.[10] This leads to the intriguing hypothesis that this new compression algorithm will be more accurate when describing transient behavior and less accurate for describing steady-state behavior. Preliminary results appear to confirm this hypothesis.

8. Summary and discussion. This paper has introduced a novel compression algorithm for probability transition matrices. The output from the algorithm is a smaller probability transition matrix with less states. The algorithm is designed to aggregate arbitrary (not necessarily NCD) probability transition matrices of DTMCs in order to obtain accurate estimations of transient behavior. Thus it appears to fill the gap between existing transient techniques (which focus on CTMCs) and existing aggregation techniques for DTMCs (which focus on steady-state behavior).

There are a number of potential avenues for further expansion of this research. The first possibility is to compress more than two states at once. Multiple-state compression may yield better results by allowing for a more accurate estimation of error. Another avenue is to derive estimates of how error propagates to higher powers of Q_c. The current similarity metric is not necessarily a good indicator of the error at higher powers of Q_c, although empirically the results are quite good. However, both of these avenues greatly increase the computational complexity of the algorithm.

The comparison with the related work indicates that this new compression algorithm can be considered to be a generalization of the more traditional aggregation formulas. This indicates yet a third avenue for research. If, in fact, column mass turns out to yield better weights for the weighted average during transient behavior, then it may be possible to smoothly interpolate between column mass and steady-state probabilities as the transient behavior approaches steady state. Of course, this presupposes the existence of the steady-state distribution, but efficient algorithms do exist to compute these distributions.

The current algorithm also quite deliberately ignores the roles of the priors $p_i^{(0)}$ in order to have as general an algorithm as possible. However, if priors are known, then it may be possible to use this information to improve the weighted averaging procedure (see appendix), thus once again reducing the error in some situations.

Finally, the amount of compression that can be achieved with negligible error is a useful indicator of whether the system is being modeled at the correct level of granularity. If the probability transition matrix is hard to compress, then the system is probably modeled at a reasonable level of granularity. However, ease of compression indicates that the system is being modeled in too much detail. In these cases monitoring the states that are chosen for compression by the similarity metric can yield important information about the characteristics of the system. This approach could be used to characterize systems that are defined by a probability transition matrix but are still not well understood at a higher level.

Appendix. This appendix formally computes $r_{\{i \vee j\},k}$. Let S_t be the random variable for the Markov chain, which can take on any of the N state values at time t. Then the shorthand notation $p_{i,j}$ is really $P[S_t = j | S_{t-1} = i]$, and $p_i^{(t)}$ is really $P[S_t = i]$. Recall the definition of conditional probability: $P[A|B] = P[A \wedge B]/P[B]$. Recall also the definition for "averaging" probabilities: $P[A] = \sum_l P[A \wedge B_l]$, where

[10] Note that Lemma 4.5 implies that if $\mathbf{b}_i = q\mathbf{b}_j$ for states i and j in Q_u, then $\pi_i = q\pi_j$.

the B_l's are mutually exclusive and exhaust the space. The computation of $r_{\{i\vee j\},k}$ is straightforward. By definition,

$$r_{\{i\vee j\},k} = P[S_t = k | S_{t-1} = (i \vee j)].$$

By definition of conditional probability and by expanding the disjunctions,

$$r_{\{i\vee j\},k} = \frac{P[S_t = k \wedge S_{t-1} = (i \vee j)]}{P[S_{t-1} = (i \vee j)]},$$

$$r_{\{i\vee j\},k} = \frac{P[S_t = k \wedge S_{t-1} = i] + P[S_t = k \wedge S_{t-1} = j]}{P[S_{t-1} = i] + P[S_{t-1} = j]}.$$

Expanding via the "averaging" of probabilities yields

$$r_{\{i\vee j\},k} = \frac{\sum_l P[S_t = k \wedge S_{t-1} = i \wedge S_{t-2} = l] + \sum_l P[S_t = k \wedge S_{t-1} = j \wedge S_{t-2} = l]}{\sum_l P[S_{t-1} = i \wedge S_{t-2} = l] + \sum_l P[S_{t-1} = j \wedge S_{t-2} = l]}.$$

Using the definition of conditional probability several times and the fact that the process is Markovian yields (in shorthand notation)

$$r_{\{i\vee j\},k} = \frac{p_{i,k} \sum_l p_{l,i} p_l^{(t-2)} + p_{j,k} \sum_l p_{l,j} p_l^{(t-2)}}{\sum_l p_{l,i} p_l^{(t-2)} + \sum_l p_{l,j} p_l^{(t-2)}}.$$

What is interesting to note is the time-dependence of this expression. Since the $p_l^{(t-2)}$ values are not known in advance, one can only make an assumption of "uniformity" (i.e., that the $p_l^{(t-2)}$ values are the same for all l). If this is done, the time-independent expression obtained is

$$r_{\{i\vee j\},k} = \frac{m_i p_{i,k} + m_j p_{j,k}}{m_i + m_j},$$

where m_i and m_j are the sums of the probability mass in columns i and j. This is what was obtained more intuitively in section 3.

Now clearly the uniformity assumption will be wrong in general, which explains why the averaging procedure can lead to errors in numerical computations. However, under conditions of row or column equivalence it is trivial to show that both the time-dependent and time-independent forms lead to the same time-independent answers. Thus, under row or column equivalence the uniformity assumption is irrelevant, and the averaging procedure yields no error. Under row and column similarity the uniformity assumption is nearly irrelevant and the time-independent expression is a good approximation for the time-dependent expression. The error of this approximation is computed in section 5.

Acknowledgments. I thank Diana Gordon for pointing out that a method for evaluating the compression algorithm was to show that $(Q_u^n)_c = Q_c^n$. Diana also pointed out sections that needed mathematical refinement. I also thank the anonymous reviewers for their very constructive comments.

REFERENCES

[1] T. Dayar and W. J. Stewart, *Quasi lumpability, lower-bounding coupling matrices and nearly completely decomposable Markov chains*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 482–498.

[2] K. De Jong, W. Spears, and D. F. Gordon, *Using Markov chains to analyze GAFOs*, in Foundations of GAs Workshop, Morgan-Kaufmann, San Francisco, 1994, pp. 115–137.

[3] E. C. Howe and C. R. Johnson, *Aggregation of Markov processes: Axiomatization*, J. Theoret. Probab., 2 (1989), pp. 201–208.

[4] E. C. Howe and C. R. Johnson, *Linear aggregation of input-output models*, SIAM J. Matrix Anal. Appl., 10 (1989), pp. 65–79.

[5] J. Kemeny and J. Snell, *Finite Markov Chains*, D. Van Nostrand, New York, 1960.

[6] A. E. Nix and M. D. Vose, *Modelling genetic algorithms with Markov chains*, Ann. Math. Artificial Intelligence, 5 (1992), pp. 79–88.

[7] R. Sidje and W. J. Stewart, *A survey of methods for computing large sparse matrix exponentials arising in Markov chains*, J. Comput. Statist. Data Anal., to appear.

[8] W. Spears, K. De Jong, T. Baeck, D. Fogel, and H. de Garis, *An overview of evolutionary computation*, in Proc. European Conference on Machine Learning, Springer-Verlag, Berlin, 1993, pp. 442–459.

[9] W. Spears and K. De Jong, *Analyzing GAs using Markov models with semantically ordered and lumped states*, in Foundations of GAs Workshop, Morgan-Kaufmann, San Francisco, 1996, pp. 85–100.

[10] W. J. Stewart, *Introduction to the Numerical Solution of Markov Chains*, Princeton University Press, Princeton, NJ, 1994.

[11] W. J. Stewart and W. Wu, *Numerical experiments with iteration and aggregation for Markov chains*, ORSA J. Comput., 4 (1992), pp. 336–350.

[12] M. Vose, *Modeling simple genetic algorithms*, Evolutionary Computation, 3 (1995), pp. 453–472.

SIAM J. MATRIX ANAL. APPL.
Vol. 20, No. 1, pp. 78–100

COMPUTATION OF DERIVATIVES OF REPEATED EIGENVALUES AND THE CORRESPONDING EIGENVECTORS OF SYMMETRIC MATRIX PENCILS[*]

ALAN L. ANDREW[†] AND ROGER C. E. TAN[‡]

Abstract. This paper presents and analyzes new algorithms for computing the numerical values of derivatives, of arbitrary order, and of eigenvalues and eigenvectors of $\mathbf{A}(\rho)\mathbf{x}(\rho) = \lambda(\rho)\mathbf{B}(\rho)\mathbf{x}(\rho)$ at a point $\rho = \rho_0$ at which the eigenvalues considered are multiple. Here $\mathbf{A}(\rho)$ and $\mathbf{B}(\rho)$ are hermitian matrices which depend analytically on a single real variable ρ, and $\mathbf{B}(\rho_0)$ is positive definite. The algorithms are valid under more general conditions than previous algorithms. Numerical results support the theoretical analysis and show that the algorithms are also useful when eigenvalues are merely very close rather than coincident.

Key words. multiple eigenvalues, close eigenvalues, eigenvalue and eigenvector sensitivities

AMS subject classifications. 65F15, 15A22

PII. S0895479896304332

1. Introduction.

Let $\mathbf{A}$ and $\mathbf{B}$ be mappings from the real field, $\mathbb{R}$, to the set of $n \times n$ (real or complex) matrices. In this paper, we study the dependence on $\rho \in \mathbb{R}$ of the eigenvalues $\lambda_i(\rho)$ and corresponding eigenvectors, $\mathbf{x}_i(\rho)$, of

$$\mathbf{A}(\rho)\mathbf{x}_i(\rho) = \lambda_i(\rho)\mathbf{B}(\rho)\mathbf{x}_i(\rho). \tag{1}$$

Except where stated otherwise, we consider only the case in which

(i) $\mathbf{A}(\rho)$, $\mathbf{B}(\rho)$, and the inverse $\mathbf{B}^{-1}(\rho)$ of $\mathbf{B}(\rho)$ are analytic functions of ρ throughout some open interval I_0 containing ρ_0;

(ii) $\mathbf{A}(\rho)$ and $\mathbf{B}(\rho)$ are hermitian for all $\rho \in I_0$ and $\mathbf{B}(\rho_0)$ is positive definite; and

(iii) (1) has an eigenvalue, $\lambda_1(\rho_0)$, of multiplicity r when $\rho = \rho_0$.

The specialization to the finite dimensional case of a known result on self-adjoint linear operators on Hilbert spaces [6, sect. 3.6.2, Thm. 1] ensures that under these hypotheses there exist r analytic functions $\lambda_1, \dots, \lambda_r : \mathbb{R} \to \mathbb{C}$ and r linearly independent analytic vector-valued functions $\mathbf{x}_1, \dots, \mathbf{x}_r : \mathbb{R} \to \mathbb{C}^n$, satisfying (1) throughout some open interval $I \subset I_0$ such that $\rho_0 \in I$ and $\lambda_1(\rho_0) = \cdots = \lambda_r(\rho_0)$. (For $\mathbf{B} = \mathbf{I}$ this was proved in [20].) In the case ($r = 1$) of simple eigenvalues, these sufficient conditions for the existence of analytic eigenvalue and eigenvector functions can be relaxed considerably [2]. The extent to which they can be relaxed when $r > 1$ is considered in section 2.

In several important problems in engineering, for example, the optimum design of structures [9], [10], and model updating [17], it is useful to know the derivatives, λ_i' and $\mathbf{x}_i'$, of the eigenvalues and eigenvectors of (1), and many papers (see [7], [9], [10], [25] for some references) are devoted to methods for the numerical computation

[*] Received by the editors May 28, 1996; accepted for publication (in revised form) December 2, 1997; published electronically September 15, 1998.

http://www.siam.org/journals/simax/20-1/30433.html

[†] School of Mathematics, La Trobe University, Bundoora, Victoria 3083, Australia (a.andrew@latrobe.edu.au).

[‡] Department of Mathematics and Programme in Computational Science, National University of Singapore, 10 Kent Ridge Crescent, Singapore 119260 (mattance@nus.edu.sg). Part of this work was done while this author was on sabbatical leave at La Trobe University.

of these derivatives. Some also consider the computation of higher derivatives which are used, for example, in reanalysis. Most of these papers deal only with the simple eigenvalue case ($r = 1$), but it is known [10], [19] that eigenvalues often coalesce for an optimum design, and even before optimizing, repeated eigenvalues may occur when a structure has certain symmetry properties [21]. First derivatives of repeated eigenvalues are relatively easy to compute [12], but, as we hope to discuss elsewhere (see also [16]), many methods which have been proposed for computing derivatives of the corresponding eigenvectors are flawed.

Since our hypotheses ensure that the eigenvalues are analytic in I, it follows that, for every pair of eigenvalues, λ_i, λ_j, either

(a) $\lambda_i(\rho) = \lambda_j(\rho)$ for all ρ in I and $\lambda_i^{(k)}(\rho_0) = \lambda_j^{(k)}(\rho_0)$ for all $k = 0, 1, 2, \ldots,$ where the superscript (k) denotes the kth derivative; or

(b) there is no open neighborhood of ρ_0 throughout which $\lambda_i(\rho) = \lambda_j(\rho)$, and there exists an integer k such that $\lambda_i^{(k)}(\rho_0) \neq \lambda_j^{(k)}(\rho_0)$.

In the first case, if, for example, $\lambda_1(\rho) = \cdots = \lambda_s(\rho)$ for all ρ in I, then, for $i = 1, \ldots, s$, $\mathbf{x}_i'(\rho)$ has an indeterminate component in $\mathrm{sp}\{\mathbf{x}_1(\rho_0), \ldots, \mathbf{x}_s(\rho_0)\}$, the space spanned by $\mathbf{x}_1(\rho_0), \ldots, \mathbf{x}_s(\rho_0)$, although it is still possible to compute derivatives of the corresponding invariant subspaces [8].

This paper is concerned only with the more common second case. Specifically, we consider the case in which $r > 1$, $\lambda_1(\rho_0) = \cdots = \lambda_r(\rho_0) \neq \lambda_i(\rho_0)$ for all $i > r$, but, for all $\rho \neq \rho_0$, in some neighborhood of ρ_0, the $\lambda_i(\rho)$ are all simple. In this case the corresponding eigenvectors are uniquely defined (to within a normalizing factor) for all $\rho \neq \rho_0$ in I, and hence, continuity requires a unique basis for the eigenspace corresponding to the repeated eigenvalue $\lambda_1(\rho_0) = \cdots = \lambda_r(\rho_0)$. This basis will not normally be the same as that computed by standard methods, which use values at ρ_0 only. Moreover, the uniqueness of the eigenvectors ensures that, in contrast to case (a), the only possible indeterminate component of $\mathbf{x}_i'(\rho)$ is the $\mathbf{x}_i(\rho_0)$ component, which depends on the normalizing condition. As in the case of simple eigenvalues [2], this condition may be chosen to define each $\mathbf{x}_i'(\rho_0)$ uniquely. Also since, for simple eigenvalues,

$$\mathbf{x}_i^*(\rho)\mathbf{B}(\rho)\mathbf{x}_j(\rho) = 0 \quad \text{for all } i \neq j, \tag{2}$$

continuity of the eigenvectors ensures that, in this case, (2) holds for all ρ in I.

Not surprisingly, derivatives of eigenvectors corresponding to multiple eigenvalues are easiest to compute when the derivatives of these eigenvalues are well separated. A good algorithm for first derivatives in this case is given in [15]. A (rather inefficient) extension of this method to some cases where repeated eigenvalues have repeated first derivatives, provided second derivatives are well separated, is given in [22], and a more efficient algorithm for these cases is given in [26]. An algorithm for the case where an eigenvalue of multiplicity $r(> 1)$ has first and second derivatives of multiplicity r and well-separated third derivatives was announced in [4]. Other approaches for special cases are given in [5] and [11]. In [5], [11], and [26] the symmetry requirement made here is relaxed, but the numerical stability problems arising in the nonsymmetric case are not considered.

In this paper, we present and analyze practical algorithms, which do not require the derivatives of any particular order of a repeated eigenvalue to be distinct and which compute the numerical values at ρ_0 of the derivatives (of any desired order) of the repeated eigenvalues and the corresponding suitably normalized eigenvectors. For simplicity we first examine an important special case in section 3, while in section 4 we show how our algorithm may be adapted to the general case and discuss practical implementation details. Some simple numerical examples are considered in section 5.

In numerical calculations the difference between equality and near equality is blurred. Even in the symmetric case, although eigenvalues are well conditioned, eigenvectors corresponding to very close eigenvalues are ill conditioned. As data are often uncertain, it is common in engineering practice to regard very close eigenvalues as equal and, instead of computing individual eigenvectors, to compute an orthonormal basis for the (well-conditioned) invariant subspace corresponding to a cluster of very close eigenvalues which are relatively well separated from the remaining eigenvalues. In this paper we propose a similar approach to the problem considered here. That is, very close eigenvalues are treated by the methods developed for repeated eigenvalues, and very close derivatives (including higher-order derivatives) are treated by the methods developed for repeated derivatives. For simplicity, section 3 deals with exact computation involving exactly repeated eigenvalues, while section 4.1 considers roundoff and the relationship between equality and near equality.

It is important to note that the problem considered here is *not* the same as the classical problem in which the only available data are $\mathbf{A}(\rho_0)$ and $\mathbf{B}(\rho_0)$. We are dealing here with matrix-valued functions. As pointed out in section 4.1, there are many important problems in engineering design in which $\mathbf{A}'(\rho_0)$, $\mathbf{B}'(\rho_0)$, and higher derivatives $\mathbf{A}^{(k)}(\rho_0)$ and $\mathbf{B}^{(k)}(\rho_0)$ are known with much the same accuracy as $\mathbf{A}(\rho_0)$ and $\mathbf{B}(\rho_0)$. It turns out that, although the eigenvectors of $\mathbf{A}(\rho_0)\mathbf{x}_i(\rho_0) = \lambda_i(\rho_0)\mathbf{B}(\rho_0)\mathbf{x}_i(\rho_0)$ corresponding to very close eigenvalues are ill conditioned, the problem may often be replaced by a well-conditioned one by regarding the close eigenvalues as equal and using information about $\mathbf{A}'(\rho_0)$ and $\mathbf{B}'(\rho_0)$ to determine the corresponding eigenvectors. Basically, this is possible if (and only if) those eigenvalues of (1) which are close at ρ_0 have derivatives which are well separated at ρ_0. Our approach could be regarded as a sort of regularization technique which, instead of putting a penalty on the magnitude of solutions as in the classical approach, uses information about the matrix derivatives. Similarly, the problem of computing the derivatives of eigenvectors corresponding to close eigenvalues is ill conditioned if the only data available are $\mathbf{A}(\rho_0)$, $\mathbf{B}(\rho_0)$, $\mathbf{A}'(\rho_0)$, and $\mathbf{B}'(\rho_0)$. However, if (and only if) the derivatives of these close eigenvalues are well separated, the problem becomes well conditioned when $\mathbf{A}''(\rho_0)$ and $\mathbf{B}''(\rho_0)$ are also specified. These questions are considered in more detail in section 4.1.

When close eigenvalues also have close derivatives, the eigenvectors can be made well conditioned by using also $\mathbf{A}''(\rho_0)$ and $\mathbf{B}''(\rho_0)$, and the derivatives of these eigenvectors can be made well conditioned by using also $\mathbf{A}'''(\rho_0)$ and $\mathbf{B}'''(\rho_0)$, provided, in each case, that the second derivatives of these close eigenvalues are well separated. In the (less common) case, in which both first and second derivatives of the eigenvalues are also close, the process can be continued, using successively higher derivatives of $\mathbf{A}$ and $\mathbf{B}$, until well-separated eigenvalue derivatives of some order are encountered. Algorithms 1 and 2 of section 3 involve a parameter m (defined just before Algorithm 1) which indicates the number of times we must go through this cycle. It is important to note that m is usually small in applications, with values of m greater than two being relatively rare, and that Algorithms 1 and 2 simplify considerably when m is small. The algorithm of [15] is included in the case $m = 1$ of Algorithm 1. However, the problem in which eigenvalues are merely very close rather than equal is not considered in [15]. Indeed, we are not aware of any of the large number of recent papers in the engineering literature on computation of derivatives of eigenvectors corresponding to repeated eigenvalues which deal adequately with close eigenvalues and the associated stability problems.

Our approach uses readily available data (the higher derivatives of $\mathbf{A}$ and $\mathbf{B}$) not utilized by classical methods to replace an important ill-conditioned problem by a well-conditioned problem and provides stable algorithms to solve that well-conditioned problem. Following the approach of [15] for repeated eigenvalues, we do not use the (ill-conditioned) eigenvectors corresponding to close eigenvalues as input for our algorithms. Instead we use an arbitrary basis of the corresponding (well-conditioned) invariant subspace. The eigenvectors are obtained as output from Algorithm 1, which uses derivatives of $\mathbf{A}$ and $\mathbf{B}$ in a way that ensures stability (see section 4.1). The theoretical predictions of section 4.1 are supported by our numerical results (summarized in section 5). For problems with repeated eigenvalues and a known closed form solution, our methods consistently computed these solutions with accuracy close to machine precision. Table 1 of section 5 demonstrates the clear superiority of our approach over classical methods when eigenvalues are very close (but not equal).

The contributions of this paper include the following. (i) Although methods for computing derivatives of eigenvectors corresponding to repeated eigenvalues with repeated derivatives have been proposed before, this paper appears to be the first to give complete proofs for any such method. (ii) We believe the method suggested here is more efficient than methods previously proposed for this problem. For example, the method of [11] requires computation of all eigenvalues and eigenvectors of (1), not just those required here. The method of [22] requires, among other things, the solution of an equation of the form $\mathbf{P}_1\mathbf{Z}'\mathbf{P}_2+\mathbf{P}_3\mathbf{Z}'\mathbf{P}_4+\mathbf{P}_5\mathbf{Z}'=\mathbf{P}_6$ (equation (15) of [22]) for the $n\times r$ matrix $\mathbf{Z}'$, where $\mathbf{P}_1$ $(n\times n)$, $\mathbf{P}_2$ $(r\times r)$, $\mathbf{P}_3$ $(n\times n)$, $\mathbf{P}_4$ $(r\times r)$, $\mathbf{P}_5$ $(n\times n)$, and $\mathbf{P}_6$ $(n\times r)$ are given. This is not required in our method. (Although there have been methods proposed for this problem which require even less computation than ours, all such methods known to us produce the wrong answers except in very special circumstances.) (iii) Apart from the special case considered in [4] (which refers to the present paper for proofs and generalizations), we are not aware of any previously published method for problems in which both first and second derivatives of repeated eigenvalues are repeated, though the possibility of such methods was mentioned, without any details, in [11] and [26]. By not requiring the derivatives of any specific order to be well separated, we have been able to give a more unified treatment of the problem than in previous papers. (iv) Unlike previous papers on computing derivatives of eigenvectors corresponding to repeated eigenvalues, this paper considers questions of numerical stability for this problem and the related problem in which eigenvalues, and perhaps also their derivatives, are merely very close rather than exactly equal. (v) Although several papers have been devoted to methods for computing second derivatives of eigenvectors corresponding to *simple* eigenvalues, our Algorithm 2 appears to be the first published method for computing these derivatives in the case of repeated eigenvalues (even when these eigenvalues have well-separated derivatives), and it also provides the first stable method for the case of very close eigenvalues. Moreover, Algorithm 2 remains valid when the repeated (or close) eigenvalues have repeated (or close) derivatives, and it allows the computation of eigenvalue and eigenvector derivatives of arbitrarily high order.

2. Existence of derivatives. Analytic eigenvalue and eigenvector functions exist in many cases not covered by the simple sufficient conditions of section 1. For example, if $\mathbf{A}(\boldsymbol{\rho})=\mathbf{S}(\boldsymbol{\rho})\ \mathbf{D}(\boldsymbol{\rho})\ \mathbf{S}^{-1}(\boldsymbol{\rho})$ and $\mathbf{B}$ is the identity matrix $\mathbf{I}$, where $\mathbf{S}(\boldsymbol{\rho})$, $\mathbf{D}(\boldsymbol{\rho})$, and $\mathbf{S}^{-1}(\boldsymbol{\rho})$ are analytic functions of the (real or complex) k-tuple $\boldsymbol{\rho}=\{\rho_1,\ldots,\rho_k\}$ and $\mathbf{D}(\boldsymbol{\rho})$ is diagonal, then, whatever their multiplicity, the eigenvalues of (1) will be the (analytic) diagonal elements of $\mathbf{D}(\boldsymbol{\rho})$ and the corresponding (linearly independent and analytic) eigenvectors will be the corresponding columns of $\mathbf{S}(\boldsymbol{\rho})$, although in this case, $\mathbf{A}(\boldsymbol{\rho})$ is not generally hermitian and the restriction made in section 1 that $\boldsymbol{\rho}$ be a single real number is not satisfied. Since, as pointed out in [13, p. 400], a hermitian matrix-valued analytic function of a complex variable must be constant, and since $\boldsymbol{\rho}$ is generally real (though not always scalar-valued) in engineering applications, we do not consider complex $\boldsymbol{\rho}$ further, except to mention that the Taylor series of an eigenvalue or eigenvector function which is analytic in a real neighborhood of $\boldsymbol{\rho_0}$ may be used to define a function analytic in a complex neighborhood of $\boldsymbol{\rho_0}$. (See also [6, sect. 3.5.3].) However, even when $\mathbf{B} = \mathbf{I}$, $\mathbf{A}(\boldsymbol{\rho})$ is hermitian, and $\boldsymbol{\rho}$ is a real k-tuple with $k > 1$, there are well-known examples [2], [6], [10], [20] in which neither the eigenvalues nor the eigenvectors are even once differentiable, although directional derivatives are known to exist in this case [10], and our methods can be used to compute partial derivatives. The case $k > 1$ is discussed further in [23].

The key requirement of our methods is that the eigenvalues and r corresponding linearly independent eigenvectors must be analytic (or at least sufficiently differentiable). The stronger conditions (i)–(iii) of section 1 are important, as they are sufficient conditions which are easy to check. Our methods can often be adapted for problems not satisfying conditions (i)–(iii) of section 1 provided this key differentiability requirement is guaranteed. However, even in the case when $\boldsymbol{\rho}$ is a real scalar, ρ, it is hard to find readily checked sufficient conditions much weaker than those mentioned in section 1 (although a result for $\mathbf{A}(\rho)$ normal and $\mathbf{B} = \mathbf{I}$ is given in [6, sect. 3.5.1 Thm. 1]). The effect of replacing the requirement that $\mathbf{B}(\rho_0)$ be positive definite by the weaker requirement that it be nonsingular is similar to the effect, in the case $\mathbf{B} = \mathbf{I}$, of allowing $\mathbf{A}(\rho)$ to be nonhermitian. For example, if, for some $\alpha > 0$,

$$\mathbf{A}(\rho) = \begin{bmatrix} 1 & 0 \\ 0 & \rho^\alpha \end{bmatrix} \quad \text{and} \quad \mathbf{B} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},$$

then at $\rho_0 = 0$, where (1) has a repeated eigenvalue, it has continuous linearly independent eigenvectors. The eigenvalues and eigenvectors (normalized, for example, by $\mathbf{x}^*\mathbf{x} = 1$) are analytic if and only if α is an even integer, while if α is an odd integer, they are differentiable only $(\alpha - 1)/2$ times. (Note the normalizations used in section 3 are appropriate only for $\mathbf{B}(\rho_0)$ positive definite.)

Although much progress [19] has been made in the development of methods for solving optimization problems in the nondifferentiable case, we believe that the class of problems where repeated eigenvalues and the corresponding eigenvectors are analytic is sufficiently large for the algorithms developed in this paper to be important.

3. The basic algorithms. Let $\boldsymbol{\Lambda} = \operatorname{diag}(\lambda_1, \ldots, \lambda_r)$, and let $\mathbf{X}$ be the $n \times r$ matrix function whose columns are $\mathbf{x}_1, \ldots, \mathbf{x}_r$, where the analytic eigenvalue functions, λ_i, and the linearly independent analytic eigenvector functions, $\mathbf{x}_i$, are those defined in section 1. Then, by (1), $\mathbf{AX} = \mathbf{BX\Lambda}$, and hence, since $\mathbf{A}, \mathbf{B}, \mathbf{X}, \boldsymbol{\Lambda}$ are analytic on I,

$$(\mathbf{AX} - \mathbf{BX\Lambda})^{(k)} = 0, \quad k = 0, 1, 2, \ldots. \tag{3}$$

This result plays a key role in the formulation of our algorithms.

To ensure uniqueness of the eigenvector derivatives, we require some normalizing condition for the eigenvectors. Two important examples are

$$\mathbf{x}_i^* \mathbf{B} \mathbf{x}_i = 1 \tag{4}$$

and

$$\mathbf{x}_i^*(\rho_0)\mathbf{B}(\rho_0)\mathbf{x}_i \;=\; 1. \tag{5}$$

In [2], where the relationship between (4) and (5) is discussed in some detail for $\mathbf{B} = \mathbf{I}$, the eigenvector functions produced by (4), which are favorite choices in the engineering literature, are called "uniformly normed," and those produced by (5), which are common in theoretical literature, are called "orthogonally constrained." The combination of either (4) or (5) with (2) gives

$$\mathbf{X}^*(\rho_0)\mathbf{B}(\rho_0)\mathbf{X}(\rho_0) \;=\; \mathbf{I}. \tag{6}$$

In the real case, (4) and (5) give the same result for $\mathbf{x}_i'(\rho_0)$ if $\mathbf{B}'(\rho_0) = \mathbf{0}$ (and, in particular, if $\mathbf{B}$ is constant), but, in general, (4) and (5) give different results for $\mathbf{x}_i^{(k)}(\rho_0)$. In the complex case, (4) does not determine $\mathbf{x}_i'(\rho_0)$ uniquely but leads (since $\mathbf{B}'(\rho)$ is hermitian) to the result $\operatorname{Re}\left(\mathbf{x}_i^*(\mathbf{B}'\mathbf{x}_i + 2\mathbf{B}\mathbf{x}_i')\right)(\rho_0) \;=\; 0$. It is customary to impose the stronger condition

$$\left(\mathbf{x}_i^*(\mathbf{B}'\mathbf{x}_i + 2\mathbf{B}\mathbf{x}_i')\right)(\rho_0) \;=\; 0. \tag{7}$$

Like (5), this specifies $\mathbf{x}_i$ to within a constant scalar factor of unit modulus (± 1 in the real case), which, in practice, is determined by the numerical process of computing $\mathbf{C}_0$ below. For higher derivatives the results for (4) become increasingly complicated, but (5) gives the simple result

$$(\mathbf{x}_i^*\mathbf{B}\mathbf{x}_i^{(k)})(\rho_0) = 0, \quad k = 1, 2, 3, \dots\ . \tag{8}$$

Together with (3), once $\mathbf{x}_i(\rho_0)$ has been fixed, (7) determines $\mathbf{x}_i'(\rho_0)$ uniquely and (8) determines each $\mathbf{x}_i^{(k)}(\rho_0)$ uniquely. Our Algorithm 1 gives the results obtained with both (7) and (8). For higher derivatives we give the result for (8) only.

Solution of (1) at $\rho = \rho_0$ by a standard computer package gives $\mathbf{\Lambda}(\rho_0)$ but, instead of $\mathbf{X}(\rho_0)$, gives an $n \times r$ matrix $\hat{\mathbf{X}}$ whose columns give a different basis for $\operatorname{sp}\{\mathbf{x}_1(\rho_0), \dots, \mathbf{x}_r(\rho_0)\}$, so that

$$\mathbf{X}(\rho_0) \;=\; \hat{\mathbf{X}}\mathbf{C}_0 \tag{9}$$

for some unknown $r \times r$ matrix $\mathbf{C}_0$. Often (especially when $\mathbf{B}(\rho_0) = \mathbf{I}$), $\hat{\mathbf{X}}$ also satisfies (6), and, if the package produces some other basis, we can easily use the Gram–Schmidt process, with the inner product induced by $\mathbf{B}(\rho_0)$, to obtain a basis with this property. Since this simplifies the analysis slightly, we assume (as in [15] and [22]) that this has been done; that is, we assume

$$\hat{\mathbf{X}}^*\mathbf{B}(\rho_0)\hat{\mathbf{X}} = \mathbf{I}.$$

It then follows from (6) that $\mathbf{C}_0$ is unitary. The two main tasks in the calculation of $\mathbf{X}'(\rho_0)$ that do not arise with simple eigenvalues are that we must calculate $\mathbf{C}_0$ and we must find the component of each $\mathbf{x}_i'(\rho_0)$ in $\operatorname{sp}\{\mathbf{x}_1(\rho_0), \dots, \mathbf{x}_r(\rho_0)\}$.

In this section we consider the case in which, for some positive integer m,

$$\begin{aligned} &\mathbf{\Lambda}^{(k)}(\rho_0) \;=\; \lambda^{(k)}(\rho_0)\mathbf{I}, && k = 0, \dots, m-1, \\ \text{but}\quad & \\ &\lambda_i^{(m)}(\rho_0) \neq \lambda_j^{(m)}(\rho_0) && \text{for all } i \neq j. \end{aligned} \tag{10}$$

This condition is satisfied in most applications involving repeated eigenvalues, usually, but by no means always, with $m = 1$. Our results for $m = 1$ include the results of [15], but many of our results (including Algorithm 2 and our observations concerning numerical stability) appear to be new even for $m = 1$. In applications involving multiple eigenvalues, the most common value of r is 2, and (10) is *always* satisfied when $r = 2$. (This is because when we have only two numbers to compare, either they are equal or they are not.) However, the situation can be more complicated when $r > 2$. For example, if $r = 3$ we may have $\lambda_1(\rho_0) = \lambda_2(\rho_0) = \lambda_3(\rho_0)$ and $\lambda_1'(\rho_0) = \lambda_2'(\rho_0) \neq \lambda_3'(\rho_0)$. (Geometrically, in this example, three eigenvalue curves intersect at ρ_0, where two of them are tangential to each other but not the third.) The modification to the algorithms presented in this section, required when (10) is not satisfied, is described in section 4.3. The parameter m defined in condition (10) plays a key role in Algorithms 1 and 2 below.

When $m > 1$, Algorithm 1 uses the fact that, since $(\mathbf{A} - \lambda\mathbf{B})\mathbf{X}' = (\lambda\mathbf{B}' - \mathbf{A}')\mathbf{X} + \mathbf{B}\mathbf{X}\mathbf{\Lambda}'$, it is easily shown (see Theorem 3.1) that, for each solution $\mathbf{V}_1$ of $(\mathbf{A} - \lambda\mathbf{B})(\rho_0)\mathbf{V}_1 = (\lambda\mathbf{B}' - \mathbf{A}')(\rho_0)\hat{\mathbf{X}} + \mathbf{B}(\rho_0)\hat{\mathbf{X}}\mathbf{\Lambda}'(\rho_0)$, there exists an $r \times r$ matrix $\mathbf{C}_1$ such that $\mathbf{X}'(\rho_0) = \mathbf{V}_1\mathbf{C}_0 + \mathbf{X}(\rho_0)\mathbf{C}_1$, where $\mathbf{C}_0$ is defined in (9). Unlike some previous methods, our method does *not* depend on making a special choice of $\mathbf{V}_1$, but a different $\mathbf{V}_1$ will produce a different $\mathbf{C}_1$. This is important for the stability analysis of section 4.1. Although $\mathbf{V}_1$ has an arbitrary component in $\mathrm{sp}\{\mathbf{x}_1(\rho_0), \ldots, \mathbf{x}_r(\rho_0)\}$, its component outside that space is well conditioned provided $|\lambda_1(\rho_0) - \lambda_i(\rho_0)|$ is not small for any $i > r$. Our methods also do not use the value of $\mathbf{A}$ or $\mathbf{B}$ or eigenvalues or eigenvectors of (1) at any point except ρ_0. In particular, numerical differentiation (a notoriously unstable process) is not used. Instead, Algorithm 1 uses $\hat{\mathbf{X}}$ and $\mathbf{V}_1$ and provides a method for computing $\mathbf{C}_0$ and $\mathbf{C}_1$, which is stable provided the derivatives, $\mathbf{A}^{(k)}(\rho_0)$ and $\mathbf{B}^{(k)}(\rho_0)$, $k = 1, \ldots, m+1$, can be computed with much the same accuracy as $\mathbf{A}(\rho_0)$ and $\mathbf{B}(\rho_0)$ (as is usual in applications) and the mth derivatives of $\lambda_1, \ldots, \lambda_r$ are well separated at ρ_0 (see section 4.1). The method of [15] cannot be used to compute $\mathbf{C}_0$ and $\mathbf{C}_1$ unless the derivatives of the repeated eigenvalues are well separated. We use the fact that condition (10) ensures that, for all $\mathbf{C}_1$, $\mathbf{\Lambda}^{(k)}(\rho_0)\mathbf{C}_1 = \mathbf{C}_1\mathbf{\Lambda}^{(k)}(\rho_0)$ for $k = 0, \ldots, m-1$ and that, if $\mathbf{\Lambda}^{(m)}(\rho_0)$ is known, the off-diagonal elements of $\mathbf{C}_1$ can easily be computed from the corresponding elements of $\mathbf{\Lambda}^{(m)}(\rho_0)\mathbf{C}_1 - \mathbf{C}_1\mathbf{\Lambda}^{(m)}(\rho_0)$. This is used in Algorithm 1 (step 7) to compute $\mathbf{C}_1$ by a generalization of a method used in [15] and in step 3 of the first algorithm of [24]. Although the primary purpose of Algorithm 1 is to compute $\mathbf{X}'$, it also gives the first $m + 1$ derivatives of $\mathbf{\Lambda}$ as a free bonus. In some applications, $\mathbf{\Lambda}''(\rho_0)$ is required but $\mathbf{X}'(\rho_0)$ is not. This requires only the first few steps of Algorithm 1. Of course, $\mathbf{\Lambda}^{(m)}(\rho_0)$ cannot be computed immediately using (3), as this would require knowledge of the eigenvector derivatives. A key idea of Algorithm 1 is that, instead of using $\mathbf{X}'(\rho_0), \mathbf{X}''(\rho_0), \ldots$, we use matrices $\mathbf{V}_1, \mathbf{V}_2, \ldots$, which may easily be computed sequentially (see step 3.3) and which are related to $\mathbf{X}'(\rho_0), \mathbf{X}''(\rho_0), \ldots$, as shown in Theorem 3.1 (iv).

Algorithm 1 computes $\mathbf{X}(\rho_0), \mathbf{\Lambda}'(\rho_0), \ldots, \mathbf{\Lambda}^{(m+1)}(\rho_0)$, and $\mathbf{X}'(\rho_0)$ when (10) is satisfied, using only $\lambda(\rho_0)$ $(= \lambda_1(\rho_0) = \cdots = \lambda_r(\rho_0))$, $\hat{\mathbf{X}}$, and $\mathbf{A}^{(k)}(\rho_0), \mathbf{B}^{(k)}(\rho_0)$, $k = 0, 1, \ldots, m+1$. Indeed, Theorem 3.1 shows that, for $k = 1, \ldots, m-1$, $\mathbf{\Lambda}^{(k)}(\rho_0)$ is computed at step 3.1, $\mathbf{\Lambda}^{(m)}(\rho_0)$ and $\mathbf{X}(\rho_0)$ are computed at step 4, $\mathbf{\Lambda}^{(m+1)}(\rho_0)$ is computed at step 6, and finally, $\mathbf{X}'(\rho_0)$ is computed at step 8. In order to keep its theoretical analysis as simple as possible, Algorithm 1 is written as though m in (10) is already known, but, as shown in section 4.2, when (10) is satisfied the algorithm is easily modified so that it calculates m automatically. This value of m is then available for use in Algorithm 2, which consequently needs no modification.

Notation. Throughout the rest of this section and throughout section 4, all functions are assumed to be evaluated at ρ_0 unless explicitly stated otherwise, and the summation operator $\sum_{k=j}^{i}$ is defined to be null if $j > i$.

ALGORITHM 1.

1. Set $\mathbf{T}_0 = \mathbf{0}$, $\mathbf{V}_0 = \hat{\mathbf{X}}$, and $\mu_0 = \lambda$, where $\mathbf{T}_0$ is $n \times r$.
2. Compute $\mathbf{Z}_0 = \mathbf{A}' - \lambda \mathbf{B}'$, $\mathbf{M}_1 = \hat{\mathbf{X}}^* \mathbf{Z}_0 \hat{\mathbf{X}}$, and $\mathbf{W}_0 = \mathbf{A} - \lambda \mathbf{B}$.
3. If $m > 1$, then, for $i = 1$ to $m - 1$, do
   - 3.1. Compute μ_i, the average of the diagonal elements of $\mathbf{M}_i$.
   - 3.2. Compute $\mathbf{W}_i = \mathbf{Z}_{i-1} - \mu_i \mathbf{B}$.
   - 3.3. Compute a solution $\mathbf{V}_i$ of

$$\mathbf{W}_0 \mathbf{V}_i \; = \; -\mathbf{T}_{i-1} - \mathbf{W}_i \mathbf{V}_0. \tag{11}$$

   (Existence of a solution is ensured by Theorem 3.1, and methods for computing $\mathbf{V}_i$ are discussed in section 4.1.)

   - 3.4. Compute

$$\mathbf{Z}_i = \mathbf{A}^{(i+1)} \; - \sum_{j=0}^{i} \binom{i+1}{j} \mu_j \mathbf{B}^{(i+1-j)} \quad \text{and}$$

$$\mathbf{T}_i = \sum_{j=1}^{i} \binom{i+1}{j} \mathbf{W}_{i+1-j} \mathbf{V}_j.$$

   - 3.5. Compute $\mathbf{M}_{i+1} = \hat{\mathbf{X}}^* (\mathbf{Z}_i \hat{\mathbf{X}} + \mathbf{T}_i)$.
4. Solve the eigenvalue problem for $\mathbf{M}_m$ (which is hermitian by Theorem 3.1 (v)). Using MATLAB, for example, this immediately gives a diagonal matrix $\boldsymbol{\Lambda}_m = \operatorname{diag}(\gamma_1, \ldots, \gamma_r)$, whose diagonal elements are the eigenvalues of $\mathbf{M}_m$, and a unitary matrix $\mathbf{U}$, whose columns are the corresponding normalized eigenvectors. Compute $\hat{\mathbf{X}}\mathbf{U}$. This is the accepted value of $\mathbf{X}$ and is denoted by $\mathbf{X}$ in the rest of the statement of this algorithm.
5. Compute a solution $\mathbf{V}_m$ of

$$\mathbf{W}_0 \, \mathbf{V}_m = -\mathbf{Z}_{m-1} \hat{\mathbf{X}} \; + \; \mathbf{B} \mathbf{X} \boldsymbol{\Lambda}_m \mathbf{U}^* \; - \; \mathbf{T}_{m-1}. \tag{12}$$

6. Compute

$$\begin{aligned} \mathbf{M}_{m+1} = \mathbf{X}^* \Bigg\{ & \left(\mathbf{A}^{(m+1)} \; - \sum_{i=0}^{m-1} \binom{m+1}{i} \mu_i \mathbf{B}^{(m+1-i)} \right) \mathbf{X} \\ & + \; (m+1)(-\mathbf{B}' \mathbf{X} \boldsymbol{\Lambda}_m \; - \; \mathbf{B} \mathbf{V}_1 \mathbf{U} \boldsymbol{\Lambda}_m \; + \; \mathbf{Z}_{m-1} \mathbf{V}_1 \mathbf{U}) \\ & + \sum_{i=2}^{m} \binom{m+1}{i} \mathbf{W}_{m+1-i} \mathbf{V}_i \mathbf{U} \Bigg\} . \end{aligned}$$

7. Compute the matrix $\mathbf{C}$, whose off-diagonal elements are given by

$$c_{ij} \; = \; \frac{m_{ij}}{[(m+1)(\gamma_j - \gamma_i)]}, \quad i \neq j, \quad i, j = 1, \ldots, r,$$

where c_{ij} and m_{ij} are the elements in the ith row and jth column of $\mathbf{C}$ and $\mathbf{M}_{m+1}$, respectively, and whose diagonal elements are the corresponding diagonal elements of

$$-\,(\mathbf{X}^*\mathbf{B}\mathbf{V}_1\mathbf{U}\,+\,\triangle),$$

where $\triangle = \mathbf{X}^*\mathbf{B}'\mathbf{X}/2$ if (7) is used and $\triangle = \mathbf{0}$ if (5) is used.

8. Compute $\mathbf{V}_1\mathbf{U}\,+\,\mathbf{X}\mathbf{C}$. This is the accepted value of $\mathbf{X}'$.

THEOREM 3.1. *Let* $\hat{\mathbf{X}}^*\mathbf{B}(\rho_0)\hat{\mathbf{X}} = \mathbf{I}$, *and let* (3), (6), (9), *and* (10) *be satisfied. Then, with the notation of Algorithm* 1, *there exist matrices* $\mathbf{V}_0,\ldots,\mathbf{V}_m$ *as required in Algorithm* 1, *and, for each choice of these, there exist matrices* $\mathbf{C}_0,\ldots,\mathbf{C}_m$ *satisfying* (iv) *below. Moreover,*

(i) $\mathbf{U} = (\mathbf{U}^{-1})^* = \mathbf{C}_0$, *and hence, step* 4 *defines the required matrix* $\mathbf{X}$;

(ii) $\mathbf{C}_1\;=\;\mathbf{C}$ *and* $\mathbf{X}'\;=\;\mathbf{V}_1\mathbf{C}_0\,+\,\mathbf{X}\mathbf{C}_1\;=\;\mathbf{V}_1\mathbf{U}+\mathbf{X}\mathbf{C}$;

(iii) $\mathbf{M}_i\;=\;\lambda^{(i)}\mathbf{I}$ *for* $i = 1,\ldots,m-1$;

(iv) $\mathbf{X}^{(i)}\;=\;\mathbf{V}_i\mathbf{C}_0\,+\,\sum_{p=1}^{i}\begin{pmatrix} i \\ p\end{pmatrix}\mathbf{X}^{(i-p)}\mathbf{C}_p$ *for* $i = 0,\ldots,m$;

(v) $\mathbf{\Lambda}_m = \mathbf{C}_0^*\mathbf{M}_m\mathbf{C}_0\;=\;\mathbf{\Lambda}^{(m)}$; *and*

(vi) $\mathbf{\Lambda}^{(m+1)}$ *is the matrix obtained from* $\mathbf{M}_{m+1}$ *by replacing its off-diagonal elements by zero.*

Proof. It follows from (3) and (10) that

$$(\mathbf{A}-\lambda\mathbf{B})\mathbf{X}^{(i)} = -\sum_{p=0}^{i-1}\begin{pmatrix} i \\ p\end{pmatrix}(\mathbf{A}-\lambda\mathbf{B})^{(i-p)}\mathbf{X}^{(p)}, \quad i = 0,1,\ldots,m-1, \tag{13}$$

and, from (3), (10), and (6), since also $\mathbf{X}^*\mathbf{A} = \mathbf{\Lambda}\mathbf{X}^*\mathbf{B}$, that

$$\begin{aligned}\mathbf{\Lambda}^{(i)} = \mathbf{X}^*\Bigg[&(\mathbf{A}^{(i)} - \sum_{j=0}^{i-1}\begin{pmatrix} i \\ j\end{pmatrix}\lambda^{(j)}\mathbf{B}^{(i-j)})\mathbf{X} \\ &+\sum_{j=1}^{i-1}\begin{pmatrix} i \\ j\end{pmatrix}(\mathbf{A}-\lambda\mathbf{B})^{(i-j)}\mathbf{X}^{(j)}\Bigg], \quad i = 0,1,\ldots,m.\end{aligned} \tag{14}$$

Make the inductive hypothesis that, for some integer $i \le m$, $\mathbf{V}_0,\ldots,\mathbf{V}_{i-1}$ as defined in Algorithm 1 exist, and, for each choice of $\mathbf{V}_0,\ldots,\mathbf{V}_{i-1}$, there exist $\mathbf{C}_0,\ldots,\mathbf{C}_{i-1}$, with $\mathbf{C}_0^* = \mathbf{C}_0^{-1}$, such that, for $j = 0,1,\ldots,i-1$,

$$\mathbf{X}^{(j)} = \mathbf{V}_j\mathbf{C}_0 + \sum_{p=1}^{j}\begin{pmatrix} j \\ p\end{pmatrix}\mathbf{X}^{(j-p)}\mathbf{C}_p \tag{15}$$

and

$$\mu_j = \lambda^{(j)}. \tag{16}$$

In the case $i = 1$ (when (15) and (16) reduce to $\mathbf{X} = \mathbf{V}_0\mathbf{C}_0$ and $\mu_0 = \lambda$), this is true by (9), (6), and the definitions in step 1 of Algorithm 1, since $\hat{\mathbf{X}}^*\mathbf{B}(\rho_0)\hat{\mathbf{X}} = \mathbf{I}$. A straightforward calculation, using the definitions of $\mathbf{T}_i, \mathbf{W}_i, \mathbf{Z}_i$, and $\mathbf{M}_i$ in step 3 of Algorithm 1, shows that (16) implies that (11) is equivalent to

$$(\mathbf{A}-\lambda\mathbf{B})\mathbf{V}_i = -\sum_{j=0}^{i-1}\begin{pmatrix} i \\ j\end{pmatrix}(\mathbf{A}-\lambda\mathbf{B})^{(i-j)}\mathbf{V}_j \tag{17}$$

and that

$$\mathbf{M}_i = \hat{\mathbf{X}}^* \left[\left(\mathbf{A}^{(i)} - \sum_{j=0}^{i-1} \binom{i}{j} \lambda^{(j)} \mathbf{B}^{(i-j)} \right) \hat{\mathbf{X}} + \sum_{j=1}^{i-1} \binom{i}{j} (\mathbf{A} - \lambda \mathbf{B})^{(i-j)} \mathbf{V}_j \right]. \tag{18}$$

Postmultiplying (17) by the nonsingular matrix $\mathbf{C}_0$ and subtracting (13), and later changing the order of summation and using standard combinatorial equalities, shows that (15) implies that (17) is equivalent to

$$\begin{aligned} (\mathbf{A} - \lambda \mathbf{B})(\mathbf{V}_i \mathbf{C}_0 - \mathbf{X}^{(i)}) &= \sum_{j=0}^{i-1} \binom{i}{j} (\mathbf{A} - \lambda \mathbf{B})^{(i-j)} (\mathbf{X}^{(j)} - \mathbf{V}_j \mathbf{C}_0) \\ &= \sum_{j=0}^{i-1} \binom{i}{j} (\mathbf{A} - \lambda \mathbf{B})^{(i-j)} \sum_{p=1}^{j} \binom{j}{p} \mathbf{X}^{(j-p)} \mathbf{C}_p \\ &= \sum_{p=1}^{i-1} \binom{i}{p} \sum_{j=p}^{i-1} \binom{i-p}{j-p} (\mathbf{A} - \lambda \mathbf{B})^{(i-j)} \mathbf{X}^{(j-p)} \mathbf{C}_p \\ &= \sum_{p=1}^{i-1} \binom{i}{p} \sum_{j=0}^{i-p-1} \binom{i-p}{j} (\mathbf{A} - \lambda \mathbf{B})^{(i-p-j)} \mathbf{X}^{(j)} \mathbf{C}_p \\ &= \sum_{p=1}^{i-1} \binom{i}{p} (\lambda \mathbf{B} - \mathbf{A}) \mathbf{X}^{(i-p)} \mathbf{C}_p, \end{aligned} \tag{19}$$

by (13) again. Since the right-hand side of (19) is clearly in the range of $(\mathbf{A} - \lambda \mathbf{B})$, and since the columns of $\mathbf{X}$ form a basis for the kernel of $(\mathbf{A} - \lambda \mathbf{B})$, it follows that (19), and hence also (11), has a solution and that, for each such solution, there exists a matrix $\mathbf{C}_i$ such that

$$\mathbf{X}^{(i)} = \mathbf{V}_i \mathbf{C}_0 + \sum_{p=1}^{i} \binom{i}{p} \mathbf{X}^{(i-p)} \mathbf{C}_p;$$

that is, (15) also holds for $j = i$. Similarly, (14), (18), (9), (15), (10), and (3) imply that

$$\begin{aligned} \mathbf{\Lambda}^{(i)} - \mathbf{C}_0^* \mathbf{M}_i \mathbf{C}_0 &= \mathbf{X}^* \sum_{j=1}^{i-1} \binom{i}{j} (\mathbf{A} - \lambda \mathbf{B})^{(i-j)} \sum_{p=1}^{j} \binom{j}{p} \mathbf{X}^{(j-p)} \mathbf{C}_p \\ &= \mathbf{X}^* \sum_{p=1}^{i-1} \binom{i}{p} \sum_{j=p}^{i-1} \binom{i-p}{i-j} (\mathbf{A} - \lambda \mathbf{B})^{(i-j)} \mathbf{X}^{(j-p)} \mathbf{C}_p \\ &= \mathbf{X}^* \sum_{p=1}^{i-1} \binom{i}{p} (\mathbf{A}\mathbf{X} - \mathbf{B}\mathbf{X}\mathbf{\Lambda})^{(i-p)} \mathbf{C}_p = \mathbf{0}. \end{aligned} \tag{20}$$

Since $\mathbf{C}_0^* = \mathbf{C}_0^{-1}$, it follows from (10) and (20) that if $i < m$, then (16) is also true when $j = i$. Hence, by induction, the required $\mathbf{V}_i$ and $\mathbf{C}_i$ all exist and (iii) and (iv) are true. Since we have also now proved (20) when $i = m$, (v) is also true. Since, by (10), the diagonal elements of the diagonal matrix $\mathbf{\Lambda}^{(m)}$ are distinct, (i) now follows from (v) and the definition of $\mathbf{U}$.

It follows from (i), (iii), (iv), (v), and the definition of $\mathbf{M}_{m+1}$ in step 6 of Algorithm 1 that

$$\begin{aligned}\mathbf{M}_{m+1} = \mathbf{X}^* \Biggl\{ & \left(\mathbf{A}^{(m+1)} - \sum_{i=0}^{m-1} \binom{m+1}{i} \lambda^{(i)} \mathbf{B}^{(m+1-i)} \right) \mathbf{X} \\ & + (m+1) \Biggl[-\mathbf{B}'\mathbf{X}\boldsymbol{\Lambda}^{(m)} - \mathbf{B}(\mathbf{X}' - \mathbf{X}\mathbf{C}_1)\boldsymbol{\Lambda}^{(m)} \\ & \qquad + \left(\mathbf{A}^{(m)} - \sum_{i=0}^{m-1} \binom{m}{i} \lambda^{(i)} \mathbf{B}^{(m-i)} \right) (\mathbf{X}' - \mathbf{X}\mathbf{C}_1) \Biggr] \\ & + \sum_{i=2}^{m} \binom{m+1}{i} (\mathbf{A} - \lambda\mathbf{B})^{(m+1-i)} \left(\mathbf{X}^{(i)} - \sum_{p=1}^{i} \binom{i}{p} \mathbf{X}^{(i-p)} \mathbf{C}_p \right) \Biggr\}.\end{aligned}$$

Also it follows from (3), (10), and (6) that

$$\begin{aligned}\boldsymbol{\Lambda}^{(m+1)} = \mathbf{X}^* \Biggl\{ & \left[\mathbf{A}^{(m+1)} - \sum_{i=0}^{m-1} \binom{m+1}{i} \lambda^{(i)} \mathbf{B}^{(m+1-i)} \right] \mathbf{X} \\ & - (m+1)\mathbf{B}'\mathbf{X}\boldsymbol{\Lambda}^{(m)} + (m+1) \left[\mathbf{A}^{(m)} - \sum_{i=0}^{m-1} \binom{m}{i} \lambda^{(i)} \mathbf{B}^{(m-i)} \right] \mathbf{X}' \\ & - (m+1)\mathbf{B}\mathbf{X}'\boldsymbol{\Lambda}^{(m)} + \sum_{i=2}^{m} \binom{m+1}{i} (\mathbf{A} - \lambda\mathbf{B})^{(m+1-i)} \mathbf{X}^{(i)} \Biggr\}.\end{aligned}$$

Hence, by (3) and an argument similar to that used in the proof of (20), taking care with the change of order of summation, it is easily shown that

$$\begin{aligned} & \boldsymbol{\Lambda}^{(m+1)} - \mathbf{M}_{m+1} \\ & = (m+1)\mathbf{X}^* \Biggl[\left(\mathbf{A}^{(m)} - \sum_{i=0}^{m-1} \binom{m}{i} \lambda^{(i)} \mathbf{B}^{(m-i)} \right) \mathbf{X}\mathbf{C}_1 \\ & \qquad + \sum_{i=2}^{m} \binom{m}{i-1} (\mathbf{A} - \lambda\mathbf{B})^{(m+1-i)} \mathbf{X}^{(i-1)} \mathbf{C}_1 \Biggr] - (m+1)\mathbf{C}_1\boldsymbol{\Lambda}^{(m)} \\ & \quad + \mathbf{X}^* \sum_{p=2}^{m} \binom{m+1}{p} \sum_{i=p}^{m} \binom{m+1-p}{i-p} (\mathbf{A} - \lambda\mathbf{B})^{(m+1-i)} \mathbf{X}^{(i-p)} \mathbf{C}_p \\ & = (m+1)(\boldsymbol{\Lambda}^{(m)}\mathbf{C}_1 - \mathbf{C}_1\boldsymbol{\Lambda}^{(m)}). \end{aligned} \tag{21}$$

Since $\boldsymbol{\Lambda}^{(m)} = \boldsymbol{\Lambda}_m = \operatorname{diag}(\gamma_1, \ldots, \gamma_r)$, the (i, j)th element of the right-hand side of (21) is $(m+1)(\gamma_i - \gamma_j)$ times the (i, j)th element of $\mathbf{C}_1$. In particular, the diagonal elements are zero. Since $\boldsymbol{\Lambda}^{(m+1)}$ is also diagonal, (vi) follows and it also follows that the off-diagonal elements of $\mathbf{C}_1$ are the corresponding elements of $\mathbf{C}$ defined in step 7 of Algorithm 1. The only constraint on the diagonal elements of $\mathbf{C}_1$ is the normalizing condition used for the eigenvectors, and it is readily deduced from (iv), (7), and (8) that the appropriate choice has been made. Hence, $\mathbf{C}_1 = \mathbf{C}$, and (ii) follows from (i) and (iv). □

Methods for computing higher derivatives of eigenvectors corresponding to simple eigenvalues are known [2] for much more general problems than (1), but even for

(1), we are not aware of any satisfactory methods in the literature for the general case of repeated eigenvalues. The following algorithm, which enables the successive computation of derivatives of arbitrarily high order, is intended to fill this gap. For arbitrary $k \in \mathbb{N}$, it computes $\mathbf{X}^{(k)}$ and $\mathbf{\Lambda}^{(m+k)}$ using the values of $\mathbf{X}, \mathbf{X}', \ldots, \mathbf{X}^{(k-1)}$ and $\mathbf{\Lambda}, \mathbf{\Lambda}', \ldots, \mathbf{\Lambda}^{(m+k-1)}$, as well as the quantities $\mathbf{Z}_0, \ldots, \mathbf{Z}_{m-1}$ and $\mathbf{W}_0, \ldots, \mathbf{W}_{m-1}$, computed in Algorithm 1, and the first $m+k$ derivatives of $\mathbf{A}$ and $\mathbf{B}$.

ALGORITHM 2.

1. For $i = k, \ldots, m+k-1$, compute a solution $\mathbf{V}_{ik}$ of

$$\mathbf{W}_0\mathbf{V}_{ik} = \sum_{j=0}^{k-1} \binom{i}{j} \left[\left(\sum_{p=0}^{i-j} \binom{i-j}{p} \mathbf{B}^{(i-j-p)} \mathbf{X}^{(j)} \mathbf{\Lambda}^{(p)} \right) - \mathbf{A}^{(i-j)} \mathbf{X}^{(j)} \right] - \sum_{j=k}^{i-1} \binom{i}{j} \mathbf{W}_{i-j} \mathbf{V}_{jk}.$$

2. Compute

$$\begin{aligned} \mathbf{M}_{m+k} = \mathbf{X}^* \Bigg\{ & \mathbf{A}^{(m+k)}\mathbf{X} - \sum_{p=0}^{m+k-1} \binom{m+k}{p} \mathbf{B}^{(m+k-p)} \mathbf{X} \mathbf{\Lambda}^{(p)} \\ & + \sum_{j=1}^{k-1} \binom{m+k}{j} \Bigg[\mathbf{A}^{(m+k-j)} \mathbf{X}^{(j)} \\ & \qquad - \sum_{p=0}^{m+k-j} \binom{m+k-j}{p} \mathbf{B}^{(m+k-j-p)} \mathbf{X}^{(j)} \mathbf{\Lambda}^{(p)} \Bigg] \\ & + \sum_{j=k}^{m+k-1} \binom{m+k}{j} \left(\mathbf{Z}_{m+k-j-1} \mathbf{V}_{jk} - \mathbf{B} \mathbf{V}_{jk} \mathbf{\Lambda}^{(m+k-j)} \right) \Bigg\}. \end{aligned}$$

3. Compute the matrix $\mathbf{C}_{kk}$, whose diagonal elements are the corresponding diagonal elements of $-\mathbf{X}^*\mathbf{B}\mathbf{V}_{kk}$ and whose remaining elements are given by

$$c_{ij;k} = m_{ij;k} \Big/ \left[\binom{m+k}{k} \left(\gamma_j - \gamma_i \right) \right], \quad i \neq j, \ \ i, j = 1, \ldots, r,$$

   where $c_{ij;k}$ and $m_{ij;k}$ are the elements in the ith row and jth column of $\mathbf{C}_{kk}$ and $\mathbf{M}_{m+k}$, respectively, and the γ_i are as in Algorithm 1.

4. Compute $\mathbf{V}_{kk} + \mathbf{X}\mathbf{C}_{kk}$ and $\mathbf{\Lambda}_{m+k} = \operatorname{diag}(m_{11;k}, \ldots, m_{rr;k})$. These are the accepted values of $\mathbf{X}^{(k)}$ and $\mathbf{\Lambda}^{(m+k)}$, respectively.

THEOREM 3.2. *Let the conditions of Theorem* 3.1 *be satisfied, and let the eigenvectors satisfy* (5). *Then*

(i) *for* $i = k, \ldots, m+k-1$ *the equation in step* 1 *of Algorithm* 2 *has a solution,* $\mathbf{V}_{ik}$, *and, for each choice of the* $\mathbf{V}_{ik}$, *there exist matrices* $\mathbf{C}_{pk}$, $p = k, \ldots, m+k-1$, *such that*

$$\mathbf{X}^{(i)} = \mathbf{V}_{ik} + \sum_{p=k}^{i} \binom{i}{p} \mathbf{X}^{(i-p)} \mathbf{C}_{pk}, \quad i = k, \ldots, m+k-1; \ \textit{and}$$

(ii) $\mathbf{C}_{kk}$ *is the matrix defined in step* 3 *of Algorithm* 2, *and step* 4 *gives the exact values of* $\mathbf{X}^{(k)}$ *and* $\mathbf{\Lambda}^{(m+k)}$.

Proof. The case $i = k$ of (i) is readily deduced from (3) and the general case is easily proved by induction (following closely the proof of Theorem 3.1 (iv)). Similarly, an argument exactly analogous to that used in the proof of (21) shows that it follows from (i) that

$$\begin{aligned}
&\mathbf{M}_{m+k} - \boldsymbol{\Lambda}^{(m+k)} \\
&= \mathbf{X}^* \sum_{j=k}^{m+k-1} \binom{m+k}{j} \sum_{p=k}^{j} \binom{j}{p} \Bigg[\Bigg(\sum_{s=0}^{m+k-j-1} \binom{m+k-j}{s} \lambda^{(s)} \mathbf{B}^{(m+k-j-s)} \\
&\qquad - \boldsymbol{\Lambda}^{(m+k-j)} \Bigg) \mathbf{X}^{(j-p)} \mathbf{C}_{pk} + \mathbf{B}\mathbf{X}^{(j-p)} \mathbf{C}_{pk} \boldsymbol{\Lambda}^{(m+k-j)} \Bigg] \\
&= \sum_{p=k}^{m+k-1} \binom{m+k}{p} \mathbf{X}^* \Bigg\{ (\mathbf{B}\mathbf{X}\boldsymbol{\Lambda} - \mathbf{A}\mathbf{X})^{(m+k-p)} \mathbf{C}_{pk} \\
&\qquad + \sum_{j=p}^{m+k-1} \binom{m+k-p}{j-p} \mathbf{B}\mathbf{X}^{(j-p)} \Big(\mathbf{C}_{pk} \boldsymbol{\Lambda}^{(m+k-j)} - \boldsymbol{\Lambda}^{(m+k-j)} \mathbf{C}_{pk} \Big) \Bigg\} \\
&= \binom{m+k}{k} \Big(\mathbf{C}_{kk} \boldsymbol{\Lambda}^{(m)} - \boldsymbol{\Lambda}^{(m)} \mathbf{C}_{kk} \Big), \text{ by (3), (6), and (10).}
\end{aligned}$$

The result now follows by the argument used in the last paragraph of the proof of Theorem 3.1. □

Note that Algorithm 2 simplifies considerably in the common case $m = 1$. In particular, only one equation needs to be solved in step 1, and the last term on the right-hand side of that equation vanishes since $i = k$.

4. Practical considerations. We now consider the removal of three key simplifying assumptions made in section 3, namely: (i) all arithmetic operations are carried out without roundoff error, and there is no uncertainty in the initial data, (ii) equation (10) is satisfied, and (iii) m in (10) is known a priori. The first of these is crucial. Because of roundoff and uncertain data, the true multiplicities of eigenvalues and their derivatives are often not known in practice. Hence, before (ii) and (iii) can be addressed, some mechanism is required for deciding whether two very close but inexactly computed quantities should be regarded as equal. Once this is available, then, as shown in sections 4.2 and 4.3, (ii) and (iii) are not too hard to deal with. Numerical stability problems associated with close eigenvalues were discussed briefly in [1] but have received remarkably little attention in the engineering literature on the computation of derivatives of eigenvalues and eigenvectors, even in those papers dealing with the nonsymmetric problem in which both eigenvectors and eigenvalues are often ill conditioned. Numerical stability and the question of when close values should be regarded as equal are discussed in section 4.1, which also considers methods for the solution of (11) and (12). The case of eigenvalues of multiplicity $r = 2$, when (10) is automatically satisfied, is considered in section 4.2, while section 4.3 concerns the more difficult (and less common) case $r > 2$.

4.1. Very close eigenvalues. Roundoff errors and uncertain data both effectively perturb $\mathbf{A}$ and $\mathbf{B}$ in (1) by small but unknown amounts. In engineering applications, $\mathbf{A}(\rho)$ and $\mathbf{B}(\rho)$ normally depend in a known way on ρ [10]. They may, for example, be polynomials in ρ, with the coefficients involving quantities subject to small errors in measurement. Because the matrix elements can be differentiated in closed form, the errors in $\mathbf{A}^{(i)}(\rho_0)$ and $\mathbf{B}^{(i)}(\rho_0)$ for $i > 0$ are usually of the same order of magnitude as those in $\mathbf{A}(\rho_0)$ and $\mathbf{B}(\rho_0)$. Roundoff errors have a similar effect. Hence, the numerical stability of this problem may be analyzed by methods similar to those used for the classical eigenvalue problem. The problem would have been much less tractable had we been forced to rely on approximate numerical values of $\mathbf{A}(\rho)$ and $\mathbf{B}(\rho)$ throughout a neighborhood of ρ_0 to estimate $\mathbf{A}^{(i)}(\rho_0)$ and $\mathbf{B}^{(i)}(\rho_0)$.

Consider first the solution of (11) (and of (12)). Since the solution has an arbitrary component in the r-dimensional kernel of W_0, we may expect different methods of solving (11) and (12) to give different solutions. However, as shown by Theorem 3.1, it does not matter which solution is obtained as long as the component of the solution outside the kernel of W_0 is computed accurately.

There are several possible methods for computing a solution of (11). We used a method described in [15], which is a generalization of a method described in [18] for the case $r = 1$, and found it to be very satisfactory. Another suitable method is the bordered matrix method whose numerical properties in the case $r = 1$ are analyzed in [2]. For example, since the columns of $\hat{\mathbf{X}}$ are assumed linearly independent, it is easily shown that the equation

$$\begin{bmatrix} \mathbf{A} - \lambda\mathbf{B} & -\mathbf{B}\hat{\mathbf{X}} \\ \hat{\mathbf{X}}^* & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{V}_1 \\ \mathbf{M} \end{bmatrix} = \begin{bmatrix} (\lambda\mathbf{B}' - \mathbf{A}')\hat{\mathbf{X}} \\ \mathbf{0} \end{bmatrix}$$

has a unique solution and that $\mathbf{V}_1$, given by that solution, is the particular solution of (11), with $i = 1$, which is orthogonal to $\operatorname{sp}\{\mathbf{x}_1, \ldots, \mathbf{x}_r\}$, and (with exact computation) the eigenvalues of $\mathbf{M}$ are the same as those of $\mathbf{M}_1$. This last observation gives a useful check on the accuracy of the calculations. Solutions of (11) may also be computed using generalized inverses, as described for $r = 1$ in [2] and [14]. A comparison of several methods for computing solutions of a system of equations with a singular coefficient matrix is made in a different context in [3].

Whichever method is used, the accuracy of the component of the computed solution outside the kernel of W_0 depends partly on the accuracy with which W_0 and the right-hand side of (11) are computed and partly on the ratio σ_1/σ_{n-r}, where $\sigma_1 \geq \cdots \geq \sigma_n$ are the singular values of W_0. Also σ_1/σ_{n-r} depends, in turn, on how close λ is to the nearest of the remaining $n - r$ eigenvalues of (1) at ρ_0. If λ is well separated from the other eigenvalues of (1), the error in the computed solution of (11) will normally be small, but if the uncertainty in the computed value of W_0 is such that its rank could be computed as $n - (r + 1)$, then the computed solution of (11) will usually be highly inaccurate. The same remarks apply to (12).

The other main sources of error in the computation of $\mathbf{X}$ and $\mathbf{X}'$ in Algorithm 1 are in the computation of $\mathbf{U}$ (in step 4) and $\mathbf{C}$ (in step 7). Both of these will be computed much more accurately if the eigenvalues of $\mathbf{M}_m$ are well separated, as the columns of $\mathbf{U}$ are eigenvectors of $\mathbf{M}_m$, and computation of the elements of $\mathbf{C}$ involves division by the differences between eigenvalues of $\mathbf{M}_m$. (Recall that the eigenvalues of $\mathbf{M}_m$ are $\lambda_1^{(m)}, \ldots, \lambda_r^{(m)}$.) Since $\mathbf{M}_m$ is hermitian, its eigenvalues are well conditioned and small changes in an eigenvalue of $\mathbf{M}_m$ will not significantly affect the final results unless it is close to another eigenvalue of $\mathbf{M}_m$. The accuracy of $\mathbf{X}^{(i)}$ $(i > 1)$ computed in Algorithm 2 is even more sensitive to both the closeness of the eigenvalues of $\mathbf{M}_m$ (as errors in the computation of lower $\mathbf{X}^{(j)}$ are fed back into the computation of the right-hand side in step 1) and the closeness of λ to other eigenvalues of (1) (as more equations with coefficient matrix W_0 must be solved in step 1).

Our numerical results suggest that the right-hand sides in Algorithms 1 and 2 can normally be evaluated with high relative accuracy and that roundoff and small inaccuracies in the data do not normally cause serious errors in the results unless either the repeated eigenvalues of (1) are very close to another eigenvalue or some of the repeated eigenvalues have very close mth derivatives. In the important case $\mathbf{B}$ = constant (or more generally when the required derivatives of $\mathbf{B}$ vanish at ρ_0), the right-hand sides simplify considerably, and this usually reduces roundoff errors. Moreover, although n (the order of $\mathbf{A}$ and $\mathbf{B}$) is often (though not always) quite large in applications, r (the multiplicity of the eigenvalue) and m (defined in (10)) are nearly always very small, and this substantially simplifies the calculation, especially for nonconstant $\mathbf{B}$, thus further reducing the scope for growth of roundoff errors in the right-hand sides.

These observations have important implications for the decision on whether two close derivatives, $\lambda_i^{(s)}$, $\lambda_j^{(s)}$, should be regarded as equal. It is because eigenvectors corresponding to very close eigenvalues are ill conditioned that it is common to regard sufficiently close eigenvalues as equal and to compute, instead, some arbitrary orthonormal basis of the corresponding invariant subspace. This replaces an ill-conditioned problem by a well-conditioned one. When (1) is known to have differentiable eigenvectors, and two very close eigenvalues have well-separated derivatives, then there is an additional reason for regarding the close eigenvalues as equal, since Algorithm 1, with $m = 1$, then enables us to compute the specific orthonormal basis which makes the eigenvectors differentiable. It may be useful to know this basis even when eigenvector derivatives are not required. When the eigenvalues and eigenvectors of (1) are sufficiently differentiable (and, in particular, when they are analytic), Algorithm 1, with $m > 1$, frequently allows us to compute $\mathbf{X}$ and $\mathbf{X}'$ stably, even when very close eigenvalues also have very close derivatives. Specifically, if, for some m, $\lambda_i^{(s)}(\rho_0)$ and $\lambda_j^{(s)}(\rho_0)$ are very close for $s = 0, \ldots, m-1$, but $\lambda_i^{(m)}(\rho_0)$ and $\lambda_j^{(m)}(\rho_0)$ are well separated, we advocate replacing both $\lambda_i^{(s)}(\rho_0)$ and $\lambda_j^{(s)}(\rho_0)$ by $[\lambda_i^{(s)}(\rho_0) + \lambda_j^{(s)}(\rho_0)]/2$ for $s = 0, \ldots, m-1$ and using Algorithm 1 to compute $\mathbf{x}_i$ and $\mathbf{x}_j$ and, if desired, $\mathbf{x}'_i$, $\mathbf{x}'_j$ and, for $s = 1, \ldots, m+1$, $\lambda_i^{(s)}$ and $\lambda_j^{(s)}$. This replaces the original (ill-conditioned) problem by a well-conditioned problem. Just as any regularization procedure changes the original problem, the new problem is not exactly the same as the original one. Indeed it cannot be so, as a well-conditioned problem cannot be identical to an ill-conditioned one. However, we claim that the computed solutions of the new (well-conditioned) problem are likely to be much closer to the true solution of the physical problem modelled by (1) than solutions obtained by classical methods will be. When the reliability of the data and the accuracy of the computation are such that the uncertainties in $\lambda_i^{(s)}$ and $\lambda_j^{(s)}$ are greater than $|\lambda_i^{(s)} - \lambda_j^{(s)}|$, no significant information is lost by replacing our estimates of $\lambda_i^{(s)}$ and $\lambda_j^{(s)}$ by the mean of these estimates. Unless the difference between eigenvalues can be computed with small relative error, classical methods cannot generally give good estimates of the corresponding eigenvectors. To compute derivatives of these eigenvectors by classical methods, we should also be able to compute the difference between the derivatives of the corresponding eigenvalues with low relative error. Our method makes no use of the (generally inaccurate) estimates of $|\lambda_i^{(s)} - \lambda_j^{(s)}|$ for $s < m$. Instead, it uses information from the higher derivatives of $\mathbf{A}$ and $\mathbf{B}$. When, for $s = 0, \ldots, m-1$, $\lambda_i^{(s)}$ and $\lambda_j^{(s)}$ are very close and $\lambda_i^{(s)} - \lambda_j^{(s)}$ is computed with a much greater relative error than $\lambda_i^{(m)} - \lambda_j^{(m)}$, our methods should perform much better than classical methods.

Trouble will also be experienced if Algorithm 1 is used with too small a value of m, as we will then encounter the problems described above, which arise when eigenvalues of $\mathbf{M}_m$ are very close. However, if the appropriate choice of m is used, then $\mathbf{M}_m$ will have no close eigenvalues and, consequently, Algorithm 1 will be stable. There remains the difficulty, noted in section 3, that, in general, the appropriate value of m is not known a priori, as it requires knowledge of higher derivatives of the eigenvalues. A modification of Algorithm 1, which allows m to be calculated, is described in section 4.2. It makes use of the fact that Algorithm 1 computes these higher derivatives. The more complicated (and rarer) problem, in which some, but not all, members of a set of very close eigenvalues have very close derivatives, is described in section 4.3. When (10) is not satisfied, the stability requirement that the eigenvalues of $\mathbf{M}_m$ are all well separated is replaced by the requirement that, for all i, j, s for which a decision is made not to regard $\lambda_i^{(s)}(\rho_0)$ and $\lambda_j^{(s)}(\rho_0)$ as equal, these two quantities should be well separated. There is just one class of problems for which our algorithms are not suitable. These are the problems for which, for some eigenvalues λ_i, λ_j, $|\lambda_i^{(s)} - \lambda_j^{(s)}|$ is very small for *all* $s = 0, 1, 2, \ldots$. Computationally, such problems are like those for which eigenvalues coincide throughout some open neighborhood of ρ_0.

The definition of μ_i in Algorithm 1 is made to minimize the effect of roundoff, as μ_i is the average of the eigenvalues of $\mathbf{M}_i$. When $\mathbf{\Lambda}^{(i)} = \lambda^{(i)}\mathbf{I}$ in exact arithmetic, this will provide a more stable measure of $\lambda^{(i)}$ in the presence of roundoff than will a single eigenvalue, or a single diagonal element, of $\mathbf{M}_i$. We recommend a similar approach whenever a decision is made to regard some set of close eigenvalues (or close eigenvalue derivatives) as equal. Each member of the set should be redefined as the mean of all the original members of the set. When we mention "multiple" eigenvalues of $\mathbf{M}_i$ in sections 4.2 and 4.3, this is taken to include close eigenvalues that have been redefined in this manner.

4.2. Computing m. We now consider the remaining points (ii) and (iii) mentioned at the beginning of this section. The common special case $r = 2$ is worth considering separately, as, in this case, (10) is always satisfied and the only modification required in Algorithm 1 is that a method for computing m, defined in (10), is needed. We can no longer ask that the loop in step 3 be done "for $i = 1$ to $m - 1$" since m is not known. The problem is overcome by selecting a small number $\epsilon > 0$ and replacing step 3 of Algorithm 1 by the following new step 3 starting with $i = 1$:

3.1. Compute μ_i, the average of the diagonal elements of $\mathbf{M}_i$. If

$$\|\mathbf{M}_i - \mu_i \mathbf{I}\| \geq \epsilon,$$

then set $m = i$ and go to step 4.

3.2, 3.3, 3.4, 3.5. (These are the same as in Algorithm 1.)

3.6. Increment i by 1, and go to 3.1.

Any convenient matrix norm may be used in step 3.1, and the choice of ϵ will depend on how willing the user is to regard close eigenvalues of $\mathbf{M}_i$ (ith derivatives of eigenvalues of (1)) as equal in cases of uncertainty. Our numerical results suggest that, when n is small and there is no uncertainty in the initial data, best results are obtained when ϵ is slightly less than the square root of the machine epsilon. Since errors are likely to increase as n increases, we suggest that a larger ϵ is appropriate when n is large or when the initial data are uncertain. Our methods are most accurate when there are no borderline values of $\|\mathbf{M}_i - \mu_i \mathbf{I}\|$ in step 3.1. For such problems, a wide range of values of ϵ will give the same result.

4.3. Problems with $r > 2$. When $r > 2$ we can no longer tell a priori whether (10) is satisfied, but in the majority of cases, (10) *is* satisfied and $m = 1$. These cases may be recognized by initially proceeding in Algorithm 1 as if $m = 1$, so that step 3 is omitted. Clearly $\mathbf{M}_1$ is hermitian, and it is well known (and easily proved) that $\mathbf{\Lambda}' = \mathbf{\Lambda}_1$, defined in step 4 with $m = 1$. Hence, when the eigenvalues of $\mathbf{M}_1$ are well separated, (10) is satisfied and Theorem 3.1 applies. The only modification required in Algorithm 1, in this case, is that step 3 is omitted and that in step 4 we must check whether the eigenvalues of $\mathbf{M}_1$ are sufficiently well separated to be regarded as distinct.

Consider now the remaining case in which $r > 2$ and some of the diagonal elements of $\mathbf{\Lambda}_1$ are sufficiently close to be regarded as equal. In this case, (10) is not usually satisfied and Algorithm 1 requires more substantial modification, although, of course, if (10) is satisfied, the above method may still be used. To illustrate the procedure when (10) is not satisfied, we describe below the method in the most common such case, that in which all repeated eigenvalues with repeated first derivatives have well-separated second derivatives.

First solve the eigenvalue problem for the $r \times r$ hermitian matrix $\mathbf{M}_1 = \hat{\mathbf{X}}^*(\mathbf{A}' - \lambda\mathbf{B}')\hat{\mathbf{X}}$; that is, compute $r \times r$ matrices $\mathbf{U}_1$ (unitary) and $\mathbf{\Lambda}_1$ (diagonal) such that $\mathbf{M}_1\mathbf{U}_1 = \mathbf{U}_1\mathbf{\Lambda}_1$. Again it is known that $\mathbf{\Lambda}' = \mathbf{\Lambda}_1$. Consider first the usual case in which $\mathbf{M}_1$ has only one multiple eigenvalue. Let its multiplicity be s. For convenience, we choose $\mathbf{U}_1$ and $\mathbf{\Lambda}_1$ so that the repeated eigenvalues of $\mathbf{M}_1$ occupy the bottom right $s \times s$ diagonal block of $\mathbf{\Lambda}_1$. Then the first $r - s$ columns of $\hat{\mathbf{X}}\mathbf{U}_1$ are columns of $\mathbf{X}$ which, for convenience, we label $\mathbf{x}_1, \ldots, \mathbf{x}_{r-s}$. Also there exists a unitary matrix $\mathbf{U}_2$, which differs from $\mathbf{I}$ only in the bottom right $s \times s$ block, such that $\mathbf{X} = \hat{\mathbf{X}}\mathbf{U}_1\mathbf{U}_2$.

Next compute $\mathbf{T}_1 = (\lambda\mathbf{B}' - \mathbf{A}')\hat{\mathbf{X}}\mathbf{U}_1 + \mathbf{B}\hat{\mathbf{X}}\mathbf{U}_1\mathbf{\Lambda}_1$. The block structure of $\mathbf{U}_2$ and $\mathbf{\Lambda}_1$ ensures that $\mathbf{U}_2\mathbf{\Lambda}_1 = \mathbf{\Lambda}_1\mathbf{U}_2$, and hence, $\mathbf{T}_1 = [(\lambda\mathbf{B}' - \mathbf{A}')\mathbf{X} + \mathbf{B}\mathbf{X}\mathbf{\Lambda}']\,\mathbf{U}_2^* = (\mathbf{A} - \lambda\mathbf{B})\mathbf{X}'\mathbf{U}_2^*$. Hence,

$$(\mathbf{A} - \lambda\mathbf{B})\mathbf{V}_1 = \mathbf{T}_1 \tag{22}$$

is solvable, and to each solution $\mathbf{V}_1$ of (22), there exists a matrix $\mathbf{C}$ such that

$$\mathbf{X}' = \mathbf{V}_1\mathbf{U}_2 + \mathbf{X}\mathbf{C}. \tag{23}$$

Compute a solution $\mathbf{V}_1$ of (22), using one of the methods described in section 4.1 or otherwise. Then compute

$$\mathbf{M}_2 = \mathbf{U}_1^*\hat{\mathbf{X}}^*\left\{(\mathbf{A}'' - \lambda\mathbf{B}'')\hat{\mathbf{X}}\mathbf{U}_1 + 2\left[(\mathbf{A}' - \lambda\mathbf{B}')\mathbf{V}_1 - \mathbf{B}\mathbf{V}_1\mathbf{\Lambda}_1 - \mathbf{B}'\hat{\mathbf{X}}\mathbf{U}_1\mathbf{\Lambda}_1\right]\right\}.$$

Since $\mathbf{\Lambda}' = \mathbf{\Lambda}_1$, $\mathbf{X} = \hat{\mathbf{X}}\mathbf{U}_1\mathbf{U}_2$, and $\mathbf{\Lambda}'\mathbf{U}_2 = \mathbf{U}_2\mathbf{\Lambda}'$, the argument used in the proof of Theorem 3.1 shows that

$$\mathbf{\Lambda}'' - \mathbf{U}_2^*\mathbf{M}_2\mathbf{U}_2 = 2(\mathbf{\Lambda}'\mathbf{C} - \mathbf{C}\mathbf{\Lambda}'). \tag{24}$$

Since the bottom right $s \times s$ block of the right-hand side is zero, it follows that $\mathbf{M}_2^-$, the bottom right $s \times s$ block of $\mathbf{M}_2$, is hermitian. Compute $s \times s$ matrices $\mathbf{U}_2^-$ (unitary) and $\mathbf{\Lambda}_2^-$ (diagonal) such that $\mathbf{M}_2^-\mathbf{U}_2^- = \mathbf{U}_2^-\mathbf{\Lambda}_2^-$. Since the second derivatives of the eigenvalues are well separated, it follows from (24) and the block structure of $\mathbf{U}_2$ that

$$\mathbf{U}_2 = \begin{bmatrix} \mathbf{I}_{r-s} & \mathbf{0} \\ \mathbf{0} & \mathbf{U}_2^- \end{bmatrix},$$

so that $\mathbf{X} = \hat{\mathbf{X}}\mathbf{U}_1\mathbf{U}_2$ can now be computed. To compute $\mathbf{X}'$ it remains only to find $\mathbf{C}$ in (23). The diagonal elements of $\mathbf{C}$ are computed using the normalizing condition as in step 7 of Algorithm 1. Since $\mathbf{\Lambda}''$ is diagonal and the diagonal elements of the right-hand side of (24) are zero, both $\mathbf{\Lambda}''$ and $(\mathbf{\Lambda}'\mathbf{C} - \mathbf{C}\mathbf{\Lambda}')$ are readily computed from (24), and the off-diagonal elements in the first $r - s$ rows and columns of $\mathbf{C}$ are computed from $\mathbf{\Lambda}'\mathbf{C} - \mathbf{C}\mathbf{\Lambda}'$, as in step 7 of Algorithm 1, since the first $r - s$ eigenvalues of $\mathbf{M}_1$ are simple. Since $\mathbf{X}, \mathbf{V}_1, \mathbf{U}_2, \mathbf{\Lambda}', \mathbf{\Lambda}''$, and $\mathbf{\Lambda}'\mathbf{C} - \mathbf{C}\mathbf{\Lambda}'$ are now all known, it is easy to compute

$$\begin{aligned}\mathbf{T}_2 = (\lambda\mathbf{B}'' - \mathbf{A}'')\mathbf{X} + 2(\lambda\mathbf{B}' - \mathbf{A}')\mathbf{V}_1\mathbf{U}_2 + 2(\mathbf{B}'\mathbf{X} + \mathbf{B}\mathbf{V}_1\mathbf{U}_2)\mathbf{\Lambda}' \\ +\mathbf{B}\mathbf{X}\mathbf{\Lambda}'' - 2\mathbf{B}\mathbf{X}(\mathbf{\Lambda}'\mathbf{C} - \mathbf{C}\mathbf{\Lambda}').\end{aligned}$$

It follows from (3) and (23) that $\mathbf{T}_2 = (\mathbf{A} - \lambda\mathbf{B})(\mathbf{X}'' - 2\mathbf{X}'\mathbf{C})$, and hence,

$$(\mathbf{A} - \lambda\mathbf{B})\mathbf{V}_2 = \mathbf{T}_2 \tag{25}$$

is solvable, and to each solution $\mathbf{V}_2$ of (25) there exists a matrix $\mathbf{C}_2$ such that

$$\mathbf{X}'' = \mathbf{V}_2 + 2\mathbf{X}'\mathbf{C} + \mathbf{X}\mathbf{C}_2. \tag{26}$$

Compute a solution $\mathbf{V}_2$ of (25) and then compute

$$\begin{aligned}\mathbf{M}_3 = \mathbf{X}^* \{(\mathbf{A}''' - \lambda\mathbf{B}''')\mathbf{X} + 3(\mathbf{A}'' - \lambda\mathbf{B}'')\mathbf{V}_1\mathbf{U}_2 + 3(\mathbf{A}' - \lambda\mathbf{B}')\mathbf{V}_2 \\ -3(\mathbf{B}''\mathbf{X} + 2\mathbf{B}'\mathbf{V}_1\mathbf{U}_2 + \mathbf{B}\mathbf{V}_2)\mathbf{\Lambda}' - 3(\mathbf{B}'\mathbf{X} + \mathbf{B}\mathbf{V}_1\mathbf{U}_2)\mathbf{\Lambda}''\}.\end{aligned}$$

By (23), the method used in the proof of Theorem 3.1 shows that

$$\begin{aligned}\mathbf{\Lambda}''' - \mathbf{M}_3 = 3\{[2\mathbf{X}^*(\mathbf{B}'\mathbf{X} + \mathbf{B}\mathbf{V}_1\mathbf{U}_2) + 2\mathbf{C}](\mathbf{\Lambda}'\mathbf{C} - \mathbf{C}\mathbf{\Lambda}') \\ +(\mathbf{\Lambda}''\mathbf{C} - \mathbf{C}\mathbf{\Lambda}'') + (\mathbf{\Lambda}'\mathbf{C}_2 - \mathbf{C}_2\mathbf{\Lambda}')\}.\end{aligned} \tag{27}$$

Now $\mathbf{C}$ and $(\mathbf{\Lambda}'\mathbf{C} - \mathbf{C}\mathbf{\Lambda}')$ have the block structure

$$\mathbf{\Lambda}'\mathbf{C} - \mathbf{C}\mathbf{\Lambda}' = \begin{bmatrix} \mathbf{D}_1 & \mathbf{D}_2 \\ \mathbf{D}_3 & \mathbf{0} \end{bmatrix} \quad \text{and} \quad \mathbf{C} = \begin{bmatrix} \mathbf{D}_4 & \mathbf{D}_5 \\ \mathbf{D}_6 & \mathbf{D}_7 \end{bmatrix},$$

where $\mathbf{D}_1$ and $\mathbf{D}_4$ are $(r - s) \times (r - s)$ and only $\mathbf{D}_7$ is unknown. In particular, the bottom right $s \times s$ corner of $\mathbf{C}(\mathbf{\Lambda}'\mathbf{C} - \mathbf{C}\mathbf{\Lambda}')$ is the known matrix $\mathbf{D}_6\mathbf{D}_2$. Since also the bottom right $s \times s$ corner of $\mathbf{\Lambda}'\mathbf{C}_2 - \mathbf{C}_2\mathbf{\Lambda}'$ is zero, the bottom right $s \times s$ corner of $\mathbf{\Lambda}''\mathbf{C} - \mathbf{C}\mathbf{\Lambda}''$ is now readily computed from (27). Since the diagonal elements of $\mathbf{\Lambda}''$ are well separated, the remaining elements of $\mathbf{C}$ (those in this bottom right $s \times s$ corner) are readily computed from this block of $\mathbf{\Lambda}''\mathbf{C} - \mathbf{C}\mathbf{\Lambda}''$ as in step 7 of Algorithm 1. The final simple step is to compute $\mathbf{X}'$ from (23).

A bonus is that, since $\mathbf{\Lambda}'''$ is diagonal and the diagonal elements of $\mathbf{\Lambda}'\mathbf{C}_2 - \mathbf{C}_2\mathbf{\Lambda}'$ are all zero, both $\mathbf{\Lambda}'''$ and $\mathbf{\Lambda}'\mathbf{C}_2 - \mathbf{C}_2\mathbf{\Lambda}'$ are readily computed from (27) now that all other quantities in (27) are known. The off-diagonal elements in the first $r - s$ rows and columns of $\mathbf{C}_2$ can then be computed as in step 7 of Algorithm 1 and the diagonal ones (which depend on the normalizing condition) as in step 3 of Algorithm 2. Hence, $\mathbf{x}_1'', \ldots, \mathbf{x}_{r-s}''$ can be computed from (26). Computation of second derivatives of the remaining eigenvectors requires an extra cycle of the process.

The above procedure is easily modified to deal with the case in which $\mathbf{M}_1$ has p (> 1) distinct multiple eigenvalues. Then, instead of one eigenvalue problem for

$\mathbf{M}_2^-$, p distinct eigenvalue problems must be solved, and $\mathbf{U}_2$ will be constructed from $(p+1)$ diagonal blocks, one for each of the p multiple eigenvalues and one for the simple eigenvalues. There are then p of the $p+1$ diagonal blocks of $\mathbf{C}$ whose values are determined from (27) rather than (24).

The related, but more complicated, method announced without complete analysis in [26] requires a special choice of $\mathbf{V}_1$ and $\mathbf{V}_2$ for its validity, but our method is valid for all solutions of (22) and (25) and is able to use the most efficient possible method for obtaining these solutions. Superficially, our method is easily generalized to the nonsymmetric case as in [26]. The main change is that instead of premultiplying by $\hat{\mathbf{X}}^*$, we premultiply by $\mathbf{Y}^*$, where $\mathbf{Y}^*\mathbf{A}(\rho_0) = \mathbf{\Lambda}(\rho_0)\mathbf{Y}^*\mathbf{B}(\rho_0)$ and $\mathbf{Y}^*\mathbf{B}(\rho_0)\hat{\mathbf{X}} = \mathbf{I}$, and instead of the unitary matrices $\mathbf{U}_i$, we compute matrices $\mathbf{U}_{Li}$ and $\mathbf{U}_{Ri}$ of left and right eigenvectors satisfying $\mathbf{U}_{Li}^*\mathbf{U}_{Ri} = \mathbf{I}$. It appears to us that similar orthogonality relations are required in the method of [26] if it is to cope with less special examples than those reported there. However, the real challenge of the nonsymmetric case is that questions of numerical stability and of existence of the higher order derivatives used in the calculations are more difficult than in the symmetric case. These questions are not considered in [11] or [26].

5. Numerical examples. We tested our algorithms on a number of simple examples with known closed form solution, and highly accurate results were obtained in *all* cases. In the first example, $\mathbf{B} = \mathbf{I}$ and $\mathbf{A} = \mathbf{SDS}^{-1}$, where

$$\sqrt{3}\mathbf{S} = \begin{bmatrix} \cos\rho & 1 & \sin\rho & -1 \\ -\sin\rho & -1 & \cos\rho & -1 \\ 1 & -\sin\rho & 1 & \cos\rho \\ -1 & \cos\rho & 1 & \sin\rho \end{bmatrix},$$

$\mathbf{D} = \mathrm{diag}(3\rho-1, 4\rho^2-3\rho, \delta(-\frac{1}{2}\rho^3+2\rho^2-\frac{3}{2}\rho+1)+(\rho^3+\rho^2-1), \rho^2-\rho+\frac{1}{2})$, and δ is a constant. Since $\mathbf{S}$ is orthogonal, $\mathbf{A}$ is symmetric. The eigenvalues of $\mathbf{A}$ (and hence, of (1) since $\mathbf{B} = \mathbf{I}$) are diagonal elements of $\mathbf{D}$, and the corresponding eigenvectors, normalized by (4), are the corresponding columns of $\mathbf{S}$. (Permuting the diagonal elements of $\mathbf{D}$ does not change $\mathbf{A}$ provided the columns of $\mathbf{S}$ are permuted the same way. Also (4) determines eigenvectors only to within a factor ±1.) When $\rho_0 = 1$ and $\delta = 0$ this example has a repeated eigenvalue ($\lambda_2 = \lambda_3$) satisfying condition (10) with $m = 3$. When the normalizing condition (5), with $\rho_0 = 1$, is used instead of (4), all the eigenvectors of this example are exactly $3/[\cos(\rho-1)+2]$ times those obtained with (4). For general δ, the eigenvalues at $\rho_0 = 1$ satisfy $\lambda_1 = 2,\ \lambda_2 = 1,\ \lambda_3 = 1+\delta,\ \lambda_4 = 1/2$; $\lambda_1' = 3,\ \lambda_2' = 5,\ \lambda_3' = 5+\delta,\ \lambda_4' = 1$; $\lambda_1'' = 0,\ \lambda_2'' = 8,\ \lambda_3'' = 8+\delta,\ \lambda_4'' = 2$; $\lambda_1''' = \lambda_2''' = \lambda_4''' = 0$, and $\lambda_3''' = 6-3\delta$. When δ is small but nonzero, there are no repeated eigenvalues. Consequently, eigenvector derivatives can theoretically be obtained by classical methods. However, our numerical results show that, for δ less than about 10^{-7} (roughly the square root of the machine epsilon used in our calculations, which were done with MATLAB on a PC), more accurate results are obtained by making the approximation $\lambda_2^{(k)} = \lambda_3^{(k)}$ for $k = 0, 1$, and 2 and using the new algorithms described in this paper.

First we describe the calculations with $\delta = 0$ (and $\rho_0 = 1$). For ease of display, all results shown are rounded to 3 decimal places. The eigenvectors of (1) corresponding to the repeated eigenvalue computed by MATLAB were already orthogonal, giving

$$\hat{\mathbf{X}} = \begin{bmatrix} 0.671 & -0.345 \\ -0.494 & -0.432 \\ -0.345 & -0.671 \\ 0.432 & -0.494 \end{bmatrix}.$$

With these starting data and with $\mathbf{V}_1$ computed by the method of [15] mentioned in section 4.1, Algorithm 1 gives

$$\mathbf{M}_1 = \begin{bmatrix} 5.000 & 0.000 \\ 0.000 & 5.000 \end{bmatrix}, \quad \mathbf{M}_2 = \begin{bmatrix} 8.000 & 0.000 \\ 0.000 & 8.000 \end{bmatrix}, \quad \mathbf{M}_3 = \begin{bmatrix} 0.296 & -1.300 \\ -1.300 & 5.704 \end{bmatrix},$$

$$\mathbf{\Lambda}_3 = \begin{bmatrix} -0.000 & 0 \\ 0 & 6.000 \end{bmatrix}, \qquad \mathbf{U} = \begin{bmatrix} 0.975 & -0.222 \\ 0.222 & 0.975 \end{bmatrix},$$

$$\mathbf{V}_1 = \begin{bmatrix} 0 & 0 \\ 0.160 & 0.433 \\ 0 & 0 \\ -0.433 & 0.160 \end{bmatrix}, \quad \mathbf{X} = \hat{\mathbf{X}}\mathbf{U} = \begin{bmatrix} 0.577 & -0.486 \\ -0.577 & -0.312 \\ -0.486 & -0.577 \\ 0.312 & -0.577 \end{bmatrix},$$

$$\mathbf{M}_4 = \begin{bmatrix} -0.000 & -7.592 \\ -7.592 & 0.000 \end{bmatrix}, \quad \text{and} \quad \mathbf{C} = \begin{bmatrix} 0.266 & -0.316 \\ 0.316 & 0.266 \end{bmatrix}.$$

Since $\mathbf{B}$ is constant, (4) and (5) give the same value of $\mathbf{C}$. The diagonal elements of $\mathbf{M}_4$ give $\lambda_2^{(4)}$ and $\lambda_3^{(4)}$, while the columns of

$$\mathbf{V}_1\mathbf{U} + \mathbf{X}\mathbf{C} = \begin{bmatrix} 0.000 & -0.312 \\ -0.000 & 0.486 \\ -0.312 & 0.000 \\ -0.486 & 0.000 \end{bmatrix}$$

give the derivatives of the corresponding eigenvectors, normalized by either (4) or (5).

Table 1 compares two methods of computing eigenvector derivatives (again with $\rho_0 = 1$) when $\delta \neq 0$. The first was the classical method of Nelson [18] which depends on the fact that the eigenvalues are not exactly equal. The second was Algorithm 1, with $m = 3$ and μ_0 chosen as the computed value of $[\lambda_2(\rho_0) + \lambda_3(\rho_0)]/2$. For each of the two methods, and for $i = 2$ and 3, Table 1 gives the value of $||\mathbf{x}_i'(\text{true}) - \mathbf{x}_i'(\text{app})||_2 / ||\mathbf{x}_i'(\text{true})||_2$, where $\mathbf{x}_i'(\text{true})$ is the true value of $\mathbf{x}_i'(\rho_0)$ and $\mathbf{x}_i'(\text{app})$ is the computed value. Calculations using another classical method (the bordered matrix method studied for example in [2]) gave much the same error as Nelson's method. Note that the error with our algorithm was approximately δ, whereas with the classical methods (with mantissa length of approximately 14 decimal digits), it was close to $10^{-14}/\delta$.

To test Algorithms 1 and 2 on an example with nonconstant $\mathbf{B}$, we considered the example $\mathbf{A} = \mathbf{S}\mathbf{D}_1\mathbf{S}^{-1}$, $\mathbf{B} = \mathbf{S}\mathbf{D}_2\mathbf{S}^{-1}$, where $\mathbf{D}_1 = \operatorname{diag}(2\rho+1, 4\rho^2 - 4\rho + 1, \rho^3 + \rho^2 - 1, \rho^2 - \rho + \frac{1}{2})$, $\mathbf{D}_2 = \operatorname{diag}(3\rho + 1, 1, \rho, \rho^2 + \frac{1}{2})$, and $\mathbf{S}$ is the same as before, so that the eigenvectors of (1) (and of $\mathbf{A}$ and of $\mathbf{B}$) are also the same as before. The eigenvalues are the diagonal elements of $\mathbf{D}_2^{-1}\mathbf{D}_1$. Again we took $\rho_0 = 1$, so that again (1) has

TABLE 1
Relative errors in computed eigenvector derivatives.

	Relative errors in computed $\mathbf{x}_2'$		Relative errors in computed $\mathbf{x}_3'$	
δ	Nelson's method	Algorithm 1	Nelson's method	Algorithm 1
10^{-13}	3.8×10^{-2}	8.4×10^{-14}	1.8×10^{-2}	1.5×10^{-13}
10^{-11}	1.8×10^{-4}	9.0×10^{-12}	2.7×10^{-4}	1.6×10^{-11}
10^{-9}	3.6×10^{-6}	9.1×10^{-10}	1.9×10^{-6}	1.6×10^{-9}
10^{-7}	2.5×10^{-8}	9.1×10^{-8}	2.1×10^{-8}	1.6×10^{-7}

repeated eigenvalues ($\lambda_2 = \lambda_3$) satisfying condition (10), this time with $m = 2$. In fact, $\lambda_2 = \lambda_3 = 1$, $\lambda_2' = \lambda_3' = 4$, $\lambda_2'' = 8 \neq \lambda_3'' = 0$. We first solved (1) at $\rho_0 = 1$ using the command EIG(A,B) of MATLAB. This produced a basis for the eigenspace corresponding to $\lambda_2 = \lambda_3$, which, although approximately orthogonal, was not exactly so. We then used the Gram–Schmidt process to obtain the orthonormal basis

$$\hat{\mathbf{X}} = \begin{bmatrix} 0.751 & 0.068 \\ -0.295 & 0.586 \\ -0.068 & 0.751 \\ 0.586 & 0.295 \end{bmatrix}.$$

Algorithm 1 then gave estimates for $\mathbf{x}_2'$ and $\mathbf{x}_3'$ with relative errors (as defined above) of 4.7×10^{-14} and 4.5×10^{-14}, respectively, using (4) and 4.7×10^{-14} and 5.7×10^{-14} using (5).

We next computed $\mathbf{x}_2''$ and $\mathbf{x}_3''$ for this example using Algorithm 2 with $m = 2$ and $k = 2$. It gave the results

$$\mathbf{V}_{22} = \begin{bmatrix} 0.577 & -0.486 \\ -0.577 & -0.312 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}, \quad \mathbf{V}_{32} = \begin{bmatrix} 0.955 & -0.312 \\ 0.200 & 0.486 \\ 0 & 0 \\ 0 & 0 \end{bmatrix},$$

$$\mathbf{M}_4 = \begin{bmatrix} 0.000 & 0.000 \\ -0.000 & -24.000 \end{bmatrix}, \quad \mathbf{C}_{22} = \begin{bmatrix} -0.667 & 0.000 \\ 0.000 & 0.333 \end{bmatrix},$$

and

$$\mathbf{V}_{22} + \mathbf{X}\mathbf{C}_{22} = \begin{bmatrix} 0.192 & -0.324 \\ -0.192 & -0.208 \\ 0.324 & 0.192 \\ -0.208 & 0.192 \end{bmatrix}.$$

The final relative errors for $\mathbf{x}_2''$ and $\mathbf{x}_3''$ were 1.1×10^{-12} and 3.4×10^{-12}, respectively, using (5). If (4) is used for second derivatives in the real case, the formula for the ith diagonal element in $\mathbf{C}_{22}$ given by Algorithm 2 is replaced by $-(\mathbf{x}_i^*\mathbf{B}\mathbf{v}_i + 2\mathbf{x}_i'^*\mathbf{B}'\mathbf{x}_i + \mathbf{x}_i'^*\mathbf{B}\mathbf{x}_i' + \mathbf{x}_i^*\mathbf{B}''\mathbf{x}_i/2)$, where $\mathbf{v}_i$ is the ith column of $\mathbf{V}_{22}$. With this normalization, we obtained relative errors of 9.0×10^{-13} and 2.0×10^{-12} in $\mathbf{x}_2''$ and $\mathbf{x}_3''$, respectively.

Numerical examples are included in several recent papers on the numerical computation of derivatives of multiple eigenvalues and the corresponding eigenvectors in the case where eigenvalue derivatives may also be repeated, but the only published example we have seen where our condition (10) is not satisfied is the second example of [26]. In that example, $\hat{\mathbf{X}}$ (which contains entries involving $\sqrt{2}$ and $\sqrt{3}$) appears not to have been calculated numerically using only $\mathbf{A}(\rho_0)$ and $\mathbf{B}(\rho_0)$, as must be done in real applications, but rather to have been constructed from the known solution in such a way that $\mathbf{U}_1 = \mathbf{I}$. However, when we computed $\hat{\mathbf{X}}$ using MATLAB, which uses only $\mathbf{A}(\rho_0)$ and $\mathbf{B}(\rho_0)$, we obtained accurate results for this problem by the method described here in section 4.3, although $\mathbf{U}_1 \neq \mathbf{I}$ with that $\hat{\mathbf{X}}$. More importantly, that example, in which $\mathbf{A}$ and $\mathbf{B}$ are real and $\mathbf{B}$ is a constant diagonal matrix, does not satisfy condition (ii) of section 1, as there is no neighborhood of ρ_0 throughout which $\mathbf{A}(\rho)$ is symmetric. The reason why the method of section 4.3 can be used without modification is that $\mathbf{A}(\rho_0)$, $\mathbf{A}'(\rho_0)$, $\mathbf{A}''(\rho_0)$, $\mathbf{M}_1$, and $\mathbf{M}_2^-$ are all symmetric, and the other important conditions (analytic eigenvalues and eigenvectors and distinct second derivatives of eigenvalues) are all satisfied.

The example considered in [26] is a little special because the columns of $\mathbf{V}_1$ corresponding to the eigenvalues with repeated derivatives are all zero. To test our method for less special problems not satisfying (10), we applied the method of section 4.3 to the example $\mathbf{A} = \mathbf{S}^{-1}\mathbf{D}_3\mathbf{S}$, $\mathbf{B} = \mathbf{I}$ at $\rho_0 = 2$, where $\mathbf{D}_3 = \mathrm{diag}(2\rho^2 - 3\rho + 1, 4\rho - 5, \rho^2 - 1, \rho)$ and $\mathbf{S}$ is the same as before. This example has $\lambda_1 = \lambda_2 = \lambda_3 = 3, \lambda_4 = 2$, $\lambda_1' = 5, \lambda_2' = \lambda_3' = 4$. With this example, we first computed a basis for the eigenspace corresponding to the multiple eigenvalue using MATLAB. Since this basis was not orthonormal we used the Gram–Schmidt method to obtain the orthonormal basis

$$\hat{\mathbf{X}} = \begin{bmatrix} -0.735 & 0.053 & 0.353 \\ 0.655 & -0.190 & 0.448 \\ 0.176 & 0.940 & -0.168 \\ -0.007 & 0.279 & 0.804 \end{bmatrix}.$$

The method of section 4.3 then gave

$$\mathbf{V}_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0.313 & -0.423 & 0.965 \end{bmatrix}, \quad \mathbf{U}_2 = \begin{bmatrix} 1.000. & 0 & 0 \\ 0 & -0.973 & -0.231 \\ 0 & 0.231 & -0.973 \end{bmatrix},$$

$$\mathbf{X} = \begin{bmatrix} -0.240 & -0.577 & -0.525 \\ -0.525 & 0.577 & 0.240 \\ 0.577 & 0.525 & -0.577 \\ -0.577 & 0.240 & -0.577 \end{bmatrix}, \quad \text{and } \mathbf{C} = \begin{bmatrix} 0.181 & -0.075 & -0.819 \\ 0.367 & -0.153 & 0.367 \\ 0.514 & 0.202 & -0.486 \end{bmatrix}.$$

It gave relative errors of 1.8×10^{-14}, 8.8×10^{-15}, and 1.3×10^{-14} in $\mathbf{x}_1'$, $\mathbf{x}_2'$, and $\mathbf{x}_3'$, respectively, using either (4) or (5), and gave estimates of the first three derivatives of λ_1, λ_2, and λ_3, all with errors less than 10^{-14}.

Acknowledgments. We thank the referees for comments which helped improve the presentation of our results.

REFERENCES

[1] A. L. Andrew, *Iterative computation of derivatives of eigenvalues and eigenvectors*, J. Inst. Math. Appl., 24 (1979), pp. 209–218.

[2] A. L. Andrew, K.-W. E. Chu, and P. Lancaster, *Derivatives of eigenvalues and eigenvectors of matrix functions*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 903–926.

[3] A. L. Andrew, K.-W. E. Chu, and P. Lancaster, *On the numerical solution of nonlinear eigenvalue problems*, Computing, 55 (1995), pp. 91–111.

[4] A. L. Andrew and R. C. E. Tan, *Computation of derivatives of repeated eigenvalues and the corresponding eigenvectors*, Z. Angew. Math. Mech., 76 (Suppl. 2) (1996), pp. 467–468.

[5] A. L. Andrew and R. C. E. Tan, *Computation of derivatives of repeated eigenvalues and corresponding eigenvectors by simultaneous iteration*, AIAA J., 34 (1996), pp. 2214–2216.

[6] H. Baumgärtel, *Analytic Perturbation Theory for Matrices and Operators*, Birkhäuser Verlag, Basel, Switzerland, 1985.

[7] J.-G Béliveau, S. Cogan, G. Lallement, and F. Ayer, *Iterative least-squares calculation for modal eigenvector sensitivity*, AIAA J., 34 (1996), pp. 385–391.

[8] K.-W. E. Chu, *On multiple eigenvalues of matrices depending on several parameters*, SIAM J. Numer. Anal., 27 (1990), pp. 1368–1385.

[9] R. T. Haftka and H. M. Adelman, *Recent developments in structural sensitivity analysis*, Structural Optimization, 1 (1989), pp. 137–151.

[10] E. J. Haug, K. K. Choi, and V. Komkov, *Design Sensitivity Analysis of Structural Systems*, Academic Press, New York, 1986.

[11] J. N. Juang, P. Ghaemmaghami, and C. B. Lim, *Eigenvalue and eigenvector derivatives of a nondefective matrix*, J. Guidance Control Dynam., 12 (1989), pp. 480–486.

[12] P. Lancaster, *On eigenvalues of matrices dependent on a parameter*, Numer. Math., 6 (1964), pp. 377–387.

[13] P. Lancaster and M. Tismenetsky, *The Theory of Matrices*, 2nd ed., Academic Press, New York, 1985.

[14] C. D. Meyer and G. W. Stewart, *Derivatives and perturbation of eigenvectors*, SIAM J. Numer. Anal., 25 (1988), pp. 679–691.

[15] W. C. Mills-Curran, *Calculation of eigenvector derivatives for structures with repeated eigenvalues*, AIAA J., 26 (1988), pp. 867–871.

[16] W. C. Mills-Curran, *Comment on "eigenvector derivatives with repeated eigenvalues"*, AIAA J., 28 (1990), p. 1846.

[17] J. E. Mottershead and M. I. Friswell, *Model updating in structural dynamics*: *A survey*, J. Sound Vibration, 167 (1993), pp. 347–375.

[18] R. B. Nelson, *Simplified calculation of eigenvector derivatives*, AIAA J., 14 (1976), pp. 1201–1205.

[19] M. L. Overton and R. S. Womersley, *Second derivatives for optimizing eigenvalues of symmetric matrices*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 697–718.

[20] F. Rellich, *Störungstheorie der Spektralzerlegung,* I: *Analytische Störung der isolierten Punkteigenwerte eines beschränkten Operators*, Math. Ann., 113 (1937), pp. 600–619.

[21] A. P. Seyranian, E. Lund, and N. Olhoff, *Multiple eigenvalues in structural optimization problems*, Structural Optimization, 8 (1994), pp. 207–227.

[22] J. Shaw and S. Jayasuriya, *Modal sensitivities for repeated eigenvalues and eigenvalue derivatives*, AIAA J., 30 (1992), pp. 850–852.

[23] J. G. Sun, *Multiple eigenvalue sensitivity analysis*, Linear Algebra Appl., 137/138 (1990), pp. 183–211.

[24] R. C. E. Tan and A. L. Andrew, *Computing derivatives of eigenvalues and eigenvectors by simultaneous iteration*, IMA J. Numer. Anal., 9 (1989), pp. 111–122.

[25] R. C. E. Tan, A. L. Andrew, and F. M. L. Hong, *Iterative computation of second-order derivatives of eigenvalues and eigenvectors*, Comm. Numer. Methods Engrg., 10 (1994), pp. 1–9.

[26] Y.-Q. Zhang and W.-L. Wang, *Eigenvector derivatives of generalized nondefective eigenproblems with repeated eigenvalues*, Trans. ASME J. Engrg. Gas Turbines Power, 117 (1995), pp. 207–212.

SIAM J. Matrix Anal. Appl.
Vol. 20, No. 1, pp. 101–116

AN ALGORITHM FOR COMPUTING THE DISTANCE TO INSTABILITY [*]

C. HE[†] AND G. A. WATSON[‡]

Abstract. An algorithm is developed for computing the distance to instability of an $n \times n$ matrix. It is aimed primarily at sparse matrices and can be used for any value of n provided that an eigenvalue problem for a $(2n) \times (2n)$ Hamiltonian matrix can be solved on the computer being used. The algorithm gives both a lower bound and an upper bound for the distance in guaranteed accuracy. The method is faster than other currently available methods.

Key words. stable matrices, distance to instability, inverse iteration, global minimum searching

AMS subject classifications. 65.68

PII. S0895479897314838

1. Introduction.

Let A be a real or complex $n \times n$ matrix. We call A a stable matrix if all eigenvalues of A are in the open left half of the complex plane. The stability of matrices has many important applications in physics, chemistry and engineering [14]. For example, stability is a fundamental concept in linear control systems having the form

$$\dot{x} = Ax + Bu, \tag{1}$$

where A is considered as a system matrix [12, 10, 11, 5]. The position of the eigenvalues of A may not give satisfactory information in practical computations, since a small perturbation may cause the eigenvalues of A to cross the imaginary axis if the eigenvalues are ill conditioned. A practical measure of the distance to instability was defined in [17] and is given by the following expression:

$$d_{us}(A) = \min_{s \in \mathcal{R}} \sigma_{\min}(A - s\mathrm{i}I),$$

where $\sigma_{\min}(A - s\mathrm{i}I)$ is the smallest singular value of $A - s\mathrm{i}I$ and $\mathrm{i} = \sqrt{-1}$. Define

$$f(s) = \sigma_{\min}(A - s\mathrm{i}I). \tag{2}$$

Then the problem of finding $d_{us}(A)$ is a one-variable optimization problem. Unfortunately, $f(s)$ is not convex, there may be local minima, and we require to find and identify the global minimum value. The distance defined here is, in fact, normally called the complex stability radius; recently a characterization of the real stability radius was given in [16].

An alternative way of looking at this problem is to consider Lyapunov stability. Let A^H denote the complex conjugate transpose of A, and let the separation of A to $-A^H$, sep(A), be defined as follows:

$$\mathrm{sep}(A) = \min\{\left\| A^H Y + YA \right\|, Y \in \mathcal{C}^{n \times n}, \|Y\| = 1\}.$$

[*] Received by the editors January 17, 1997; accepted for publication (in revised form) by A. Bunse-Gerstner December 30, 1997; published electronically September 15, 1998.
http://www.siam.org/journals/simax/20-1/31483.html

[†] Department of Mathematics, University of Kansas, Snow Hall 405, Lawrence, KS 66045 (che@zeus.cstp.umkc.edu).

[‡] Department of Mathematics, University of Dundee, Dundee, DD1 4HN, Scotland (gawatson@dundee.ac.uk).

Here $\| \quad \|$ denotes the 2-norm rather than the Frobenius norm which is conventionally used [7, 17]. The quantity $\operatorname{sep}(A)$ based on the 2-norm is easily computed [13]. In fact, if X is the positive definite solution of the Lyapunov equation $A^TX + XA = -I$, where I denotes the identity, then $\operatorname{sep}(A) = 1/\|X\|$ [13]. Moreover, the separation and the distance were proved to be equivalent in [8] in the sense that

$$\frac{1}{2}\operatorname{sep}(A) \le d_{us}(A) \le \frac{\sqrt{2}\,\|A\|\,\sqrt{\operatorname{sep}(A)}}{\sqrt{\pi}\sqrt{(\|A\| - \operatorname{sep}(A))}}.$$

Thus one solver of the Lyapunov equation yields rough bounds for the distance to instability. An algorithm has recently been given to compute the Lyapunov stability for sparse matrices under some restrictions [15].

The conventional method for computing $d_{us}(A)$ is the bisection method, which was presented in [4]. The method gives both a lower bound and an upper bound of $d_{us}(A)$ which are as tight as required. However, each step of the bisection method requires the solution of an eigenvalue problem of the Hamiltonian matrix

$$H(\alpha) = \begin{bmatrix} A & -\alpha I \\ \alpha I & -A^H \end{bmatrix} \tag{3}$$

for a positive number α. For matrices of small size, the method can be very efficient. However, when n increases much beyond 50, then the method can start to become quite expensive. A suitable variant of the bisection method was proved to have a quadratic convergence rate and can be extended to solve the H_∞-norm problem; see [2, 3].

Here a new method is developed for computing the distance to instability. The method consists of two major steps:

1. Use a method based on inverse iteration to descend to a stationary point of $f(s)$.
2. Solve an eigenvalue problem for $H(\alpha)$ to check whether the point reached is a global minimum or not. If it is not, an improved initial vector is found so that step 1 can be repeated to give movement away from the current point toward the global minimum value.

A similar two-level method has been developed for computing the numerical radius and shown to be successful [9, 18]. The motivation for using a method based on inverse iteration rather than relying primarily on eigenvalue computations is twofold. First, for a sparse matrix A, solving the linear system $Ax = b$ is much faster than solving the eigenvalue problem $Ax = \lambda x$. Second, the powerful checking step (step 2) not only checks for a global minimum, but also provides a good initial vector for inverse iteration to give further improvement if that is appropriate.

A better understanding of how the method can work is illustrated by the example shown in Figure 1, where the s axis is in a horizontal direction. If the algorithm is started from the initial point $P0 = (s_0, f(s_0))$, the inverse iteration method may descend to the local minimum value $P1 = (s_1, f(s_1))$. An improved point suggested by step 2 that is used as the initial approximation for the inverse iteration method can then lead to $P2 = (s_2, f(s_2))$ and from there to $P3 = (s_3, f(s_3))$, the global minimum point of the function $f(s)$.

The paper is organized as follows: the new method is given in section 2, and the numerical behavior of the method is discussed in section 3. In fact, the condition number of the eigenvalues of $H(\alpha)$ is explicitly given, and it is proved that the eigenvalues

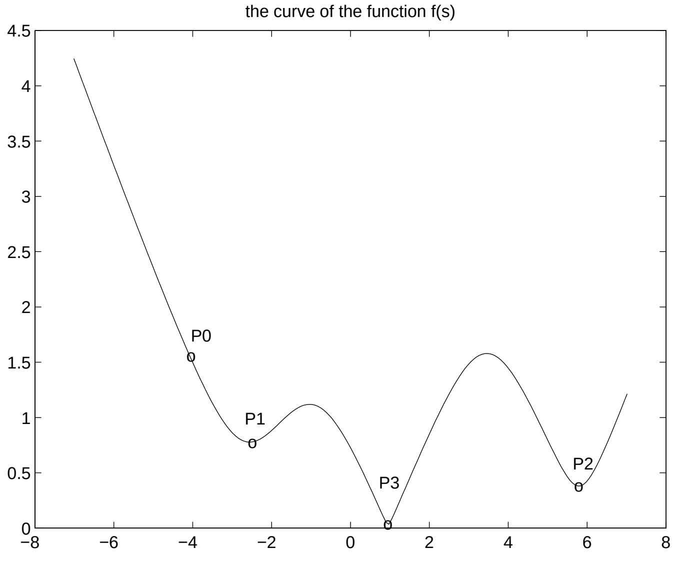


FIG. 1

of $H(\alpha)$ which are on the imaginary axis are at most perturbed by half precision if the original data are perturbed by full precision. Numerical examples are given in section 4 to demonstrate that the method is generally more efficient and faster than the quadratically convergent method.

2. The algorithm. Calculating $f(s)$ requires computing the smallest singular value of $A - s\mathrm{i}I$. This can be achieved by inverse iteration as follows:

```
choose an initial $x_0$;
for $j = 1, 2, \dots$ ,
$y_j = (A - s\mathrm{i}I)^{-1}(A - s\mathrm{i}I)^{-H}x_{j-1}$,
$x_j = y_j / \|y_j\|$ ,
```

where throughout the paper $\|.\|$ denotes the l_2-norm. Then we have $f(s) = 1/\sqrt{\|y_\infty\|}$, where y_∞ denotes a limit point of the iteration.

Methods other than inverse iteration can also be used for computing the smallest singular value, for example, Rayleigh quotient iteration, although inverse iteration is normally favored. Based on function values, any systematic method of adjusting s can now be used, for example, a direct search method based on bracketing a minimum and systematic reduction of its size. Because derivatives of f are normally available (see Theorem 2.1), more sophisticated methods are possible.

Another way of adjusting s is based on the observation that the problem

$$\min_s \|(A - s\mathrm{i}I)x_\infty\|$$

is solved by

$$s^+ = \mathrm{imag}(x_\infty^H A x_\infty). \tag{4}$$

This choice of s is in some ways a natural one (being locally the best possible). If this is used in conjunction with a computation for the smallest singular value, then because minima are being calculated in both parts of this process, the sequence of pairs of steps results in a descent process. It is, in fact, a special case of the alternating algorithm, and any limit point at which f is differentiable is a stationary point of f.

We choose here to use an iteration scheme which modifies the above idea in the following way. Rather than compute a sequence of accurate smallest singular values, combined with a separate adjustment of s at each iteration, we prefer to compute a sequence of pairs of values of x and s which achieves an accurate value of $f(s)$ simultaneously with satisfaction of the stationary point conditions only in the limit. This can be achieved, by analogy with the method of [3], by taking just one inverse iteration step in step 1. The modified iteration process is as follows:

(0) Choose x_0.

For $k = 1, 2, \ldots$ until convergence do:

(1) $s_{k-1} = \text{imag}(x_{k-1}^H A x_{k-1})$,

(2) $y_k = (A - s_{k-1}\mathrm{i}I)^{-1}(A - s_{k-1}\mathrm{i}I)^{-H} x_{k-1}$,

(3) $x_k = y_k / \|y_k\|$.

Experience suggests that this choice of $s_k,\ x_k$ is such that the sequence $\{\|(A - s_k\mathrm{i}I)x_k\|\}$ is descending. Of course this can always be ensured by including extra inverse iteration steps (without changing s_k) if necessary. Consider the problem of minimizing $\|(A - s\mathrm{i}I)^H x\|$ with respect to $s \in R$ and $x \in C^n$ with respect to $x^H x = 1$. The system of equations satisfied at a solution is

$$(A - s\mathrm{i}I)^H (A - s\mathrm{i}I)x \;=\; \lambda x, \quad x^H x = 1, \tag{5}$$

$$s \;=\; \text{imag}(x^H A x), \tag{6}$$

where λ is a Lagrange multiplier. The above iteration may therefore be interpreted as simple iteration (with renormalization) applied to (5) and (6). Further, let s^* be a fixed point of the iteration, and let $s_{k-1} = s^* + \epsilon_{k-1}$. Then

$$(A - s^*\mathrm{i}I)^H (A - s^*\mathrm{i}I) y_k = x_{k-1} + O(\epsilon_{k-1}).$$

Thus, to first order, a step of simple iteration is just a step of inverse iteration applied to the matrix $(A - s^*\mathrm{i}I)^H(A - s^*\mathrm{i}I)$. We would expect, therefore, the simple iteration process to inherit the convergence properties of inverse iteration in this case. We also have the following result.

THEOREM 2.1. *Let $s^*,\ x^*$ be a fixed point of the iteration. Then σ^* is a singular value of $(A - s^*\mathrm{i}I)$ such that*

$$\sigma^* \;=\; \|(A - s^*\mathrm{i}I)x^*\| \;=\; \min_{s \in \mathcal{R}} \|(A - s\mathrm{i}I)x^*\|. \tag{7}$$

If σ^* is the smallest singular value which is simple and greater than zero, then f is differentiable and

$$f'(s^*) \;= 0. \tag{8}$$

Proof. The relationship (7) is an immediate consequence of the iteration, so we prove (8).

For any s in a neighborhood of s^* such that the smallest singular value $\sigma > 0$ of $(A - s\mathrm{i}I)$ is simple, we can write

$$f(s) \;=\; v^H(A - s\mathrm{i}I)u,$$

where $v^Hv = 1,\ u^Hu = 1$, with v and u, respectively, corresponding to left and right singular vectors. Thus

$$\mathrm{real}((v')^Hv) = 0, \quad \mathrm{real}(u^Hu') = 0,$$

where the dash denotes differentiation with respect to s. Further,

$$f'(s) = (v')^H(A - s\mathrm{i}I)u - \mathrm{i}(v^Hu) + v^H(A - s\mathrm{i}I)u',$$

or equivalently (because $f'(s)$ is real),

$$\begin{aligned} f'(s) &= \mathrm{real}(\sigma(v')^Hv - \mathrm{i}(v^Hu) + \sigma u^Hu') \\ &= -\mathrm{real}(\mathrm{i}v^Hu). \end{aligned} \tag{9}$$

Now

$$\begin{aligned} \mathrm{i}v^Hu \;&=\; \frac{\mathrm{i}}{\sigma}u^H(A - s\mathrm{i}I)^Hu \\ &=\; \frac{\mathrm{i}}{\sigma}u^HA^Hu \;-\; \frac{1}{\sigma}s. \end{aligned}$$

Thus

$$\begin{aligned} f'(s^*) \;&=\; -\mathrm{real}\left(\frac{\mathrm{i}}{\sigma^*}(x^{*H}A^Hx^*) \;-\; \frac{1}{\sigma^*}s^*\right) \\ &=\; \frac{1}{\sigma^*}(-\mathrm{imag}(x^{*H}Ax^*) + s^*) \\ &=\; 0, \end{aligned}$$

and the result is proved. □

A fixed point of the iteration need not, of course, give $d_{us}(A)$ but may be at best a local minimum. It is necessary to make a check on this, and the following theorem given in [4] may be used for this purpose.

THEOREM 2.2. *$H(\alpha)$ defined by* (3) *has an eigenvalue whose real part is zero if and only if $\alpha \geq d_{us}(A)$.*

We now give a supplementary result which augments Theorem 2.2 and indicates how it can be used in practice.

THEOREM 2.3. *Let s_k, x_k be given by the modified inverse iteration procedure. Let $\alpha = \|(A - s_k\mathrm{i}I)x_k\| - tol$, where tol is a small positive number, and assume that $H(\alpha)$ has a pure imaginary eigenvalue $s\mathrm{i}$ with corresponding eigenvector $w \in \mathcal{C}^{2n}$, where w is partitioned so that its first n elements are represented by u and its last n elements are represented by v. Then we can normalize so that $\|u\| = \|v\| = 1$, and further,*

$$\|(A - s\mathrm{i}I)u\| \;<\; \|(A - s_k\mathrm{i}I)x_k\|. \tag{10}$$

Proof. We have, by assumption,

$$\begin{bmatrix} A & -\alpha I \\ \alpha I & -A^H \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = si \begin{bmatrix} u \\ v \end{bmatrix}. \tag{11}$$

It follows that

$$(A - s\mathrm{i}I)u = \alpha v$$

and

$$(A - s\mathrm{i}I)^H v = \alpha u.$$

Thus $\|u\| = \|v\|$, and further,

$$(A - s\mathrm{i}I)^H (A - s\mathrm{i}I)u = \alpha^2 u.$$

The result follows. □

Theorem 2.3 is important because (10) shows that an improved vector can be obtained for restarting the modified inverse iteration process, provided that $d_{us}(A)$ has not been found, as a by-product of the eigenvalue calculation. The most natural place to apply the result is, of course, when s_k and x_k are in fact limit points, but obviously it can be used as a restart procedure in other situations. We can now state the details of our algorithm.

```
Algorithm for computing $d_{us}(A)$.
INPUT: $A$; $tol$, a tolerance; and $x_0$, a random complex vector.
OUTPUT: $lb$ and $ub$, the lower and upper bounds of $d_{us}(A)$ satisfying $ub - lb \le tol$.
  0. Set $lb = 0$, $ub = \|A\|_1$, and $s_0 = \mathrm{imag}(x_0^H A x_0)/(x_0^H x_0)$.
  1. For $k = 1, 2, \ldots,$ do until convergence
       $y_k = (A - s_{k-1}\mathrm{i}I)^{-1}(A - s_{k-1}\mathrm{i}I)^{-H} x_{k-1}$,
       $x_k = y_k / \|y_k\|$,
       $s_k = \mathrm{imag}(x_k^H A x_k)$,
       $\sigma_k = 1/\sqrt{\|y_k\|}$.
  2. While $ub - lb > tol$, do
       Solve the eigenvalue problem of $H(\sigma - tol)$ using the ordinary QR method,
       where $\sigma$ is a limit point of $\{\sigma_k\}$ from step 1.
       If $H(\sigma - tol)$ does not have a pure imaginary eigenvalue, then stop and return
       $ub = \sigma$ and $lb = \sigma - tol$.
       Otherwise if $s\mathrm{i}$ is the pure imaginary eigenvalue of $H(\sigma - tol)$ and $u$ the vector,
       as in (11), then set $ub = \min\{ub, \sigma - tol\}$ and $x_0 = u$, $s_0 = s$, and repeat step
       1.
```

The storage required by the algorithm is roughly $(2n)^2$ floating point numbers, which is required by the QR method for solving the eigenvalue problem of the $(2n) \times (2n)$ Hamiltonian matrix $H(\alpha)$. Here the Hamiltonian matrix is treated as an unstructured matrix.

Let us discuss the computational aspects of our algorithm. Let A be a sparse matrix of size n, with number of nonzero elements n_0 with density η, where the density of a sparse matrix, η, is defined by $n_0 = \eta n^2$. We assume here that the computational cost of the sparse LU factorization of A is $2(\eta n/2)^2 n$ flops by a direct method; at least, this is true for a band matrix with lower bandwidth $0.5\eta n$ and upper bandwidth $0.5\eta n$ (see [7, p. 151]). The computational cost of the modified inverse iteration is dominated by that of the LU factorization of $A - s_k\mathrm{i}I$. Since the modified inverse iteration uses complex arithmetic which is 4 times as costly as real arithmetic, the total cost of N modified inverse iterations is $N * 4 * 2 * (\eta/2)^2 * n^3 = 2N\eta^2 n^3$ flops. Suppose that $\eta = 0.2$ (so that the density is 20%). Then the total cost of the N modified inverse iterations is $0.08Nn^3$ flops. Now the cost of one eigenvalue solver of a $(2n) \times (2n)$ matrix using the QR method is $10(2n)^3 = 80n^3$ flops (see [7, p. 380]). Thus, roughly speaking, 1000 modified inverse iterations for this kind of sparse matrix costs as much as one eigenvalue solver of a $(2n) \times (2n)$ Hamiltonian matrix. This number reduces to 30 if sparsity cannot be exploited.

We now compare the computational cost of our algorithm with that of the quadratically convergent algorithm for this problem. The latter is directly translated from the algorithm in [2] which was originally used to compute the L_∞-norm of a transfer function.

```
Boyd/Balakrishnan algorithm for computing $d_{us}(A)$.
INPUT: $A$; $tol$, a tolerance.
OUTPUT: $\gamma$, the computed distance within the tolerance $tol$.
    Until $l = 0$, do
  1. Find the frequency intervals $I_1, \ldots, I_l$, where $\gamma(1 - tol) < \sigma_{\min}(A - s\mathrm{i}I)$ for
     each $I_k$.
  2. Set $s_k = midpoint(I_k), k = 1, 2, \ldots, l$, and $\gamma = \max_k \sigma_{\min}(A - s_k\mathrm{i}I)$.
```

Any comparison clearly depends on how many steps of each method have to be taken. Let us assume that to achieve the same accuracy the new algorithm requires n_1 modified inverse iteration steps and n_2 eigenvalue solvers and the Boyd/Balakrishnan algorithm requires n_3 eigenvalue solvers. Then (ignoring extras and overheads) the new algorithm will be faster provided that

$$\frac{n_1}{1000} + n_2 \; < \; n_3.$$

For example, if $n_1 = 300$, $n_2 = 1$, and $n_3 = 4$, then based on these figures the new algorithm will be about 3 times as fast.

3. Computational considerations. Inverse iteration is one of the most powerful methods for computing the smallest modulus eigenvalue and the corresponding eigenvector of a matrix [7]. It is normally used to obtain the eigenvector when the eigenvalue is known, which requires one or two inverse iterations. It is also well known that if the eigenvalue to be computed lies in a cluster of eigenvalues, then this may result in a slowing down of the convergence of the inverse iterations. Let s^* be a limiting value of the sequence $\{s_k\}$. Then, as already observed, the convergence of the modified inverse iteration suggested here (step 1 of the Algorithm) is asymptotically the same as that of the usual inverse iteration procedure applied to the matrix $(A - s^*\mathrm{i}I)^H(A - s^*\mathrm{i}I)$ and therefore inherits properties of that iteration.

In the numerical computation, our stopping criterion is

$$|s_{k-1} - s_k| \le \min\{10^{-8}, tol * \|A\|_1\},$$

where 10^{-8} is based on the observation that the local minimizer s^* is usually half as accurate (measured by the number of decimal places) as the value $f(s^*)$. We will elaborate on that later. The use of $\|A\|_1$ is primarily for convenience of computation. To improve the convergence, a good initial value of s_0 can be chosen. Usually good choices for s_0 are the imaginary parts of the eigenvalues close to the imaginary axis.

The step which checks for a global minimum value (step 2 of the Algorithm) plays a vital role. Deciding whether $H(\alpha)$ for some α has a pure imaginary eigenvalue or not is crucial. The condition number of the pure imaginary eigenvalue will, in turn, play an important part in the criterion for that decision. Suppose for the moment that $\alpha = \sigma$, that λ is an eigenvalue of $H(\sigma)$, and x and y are the corresponding right and left eigenvectors, i.e.,

$$H(\sigma)x = \lambda x, y^H H(\sigma) = \lambda y^H.$$

Then the condition number of λ is defined in [19] as

$$\kappa = \frac{\|x\|\,\|y\|}{|y^H x|}. \tag{12}$$

THEOREM 3.1. *Suppose that $\sigma = f(s)$ is a simple singular value of $A - s\mathrm{i}I$ and u and v are the corresponding singular vectors such that $(A - s\mathrm{i}I)u = \sigma v$. If $s\mathrm{i}$ is a simple eigenvalue of $H(\sigma)$ and κ is defined in* (12), *then*

$$\kappa = \frac{1}{|f'(s)|}.$$

Proof. We start from the equation

$$H(\sigma)\begin{bmatrix} u \\ v \end{bmatrix} = s\mathrm{i}\begin{bmatrix} u \\ v \end{bmatrix}.$$

Since $H(\sigma)$ is a Hamiltonian matrix, we obtain

$$[v^H, -u^H]H(\sigma) = s\mathrm{i}[v^H, -u^H].$$

It follows from the fact that $s\mathrm{i}$ is simple that $[v^H, -u^H]$ is the left eigenvector of $H(\sigma)$. Thus

$$\kappa = \frac{\left\|\begin{bmatrix} u \\ v \end{bmatrix}\right\| \left\|\begin{bmatrix} v \\ -u \end{bmatrix}\right\|}{\left|[v^H, -u^H]\begin{bmatrix} u \\ v \end{bmatrix}\right|}.$$

From the argument following (11), we can normalize so that $\|u\| = 1$ and $\|v\| = 1$. Then we have

$$\begin{aligned} \kappa &= \frac{2}{|v^H u - u^H v|} \\ &= \frac{1}{|\mathrm{real}(\mathrm{i}v^H u)|}. \end{aligned} \tag{13}$$

The result follows using (9). □

The significance of Theorem 3.1 is that if $s\mathrm{i}$ is a simple eigenvalue of $H(\sigma)$, then it is safe to separate the eigenvalues on the imaginary axis from those off the imaginary axis numerically due to its robust condition number $\kappa = 1/|f'(s)|$. It is always the case that there are at least two pure imaginary eigenvalues of $H(\sigma)$ when $\sigma > d_{us}(A)$. The two pure imaginary eigenvalues are merged to be one when $\sigma = d_{us}(A)$. We are interested in the case that $\sigma = d_{us}(A)$ so that $H(\sigma)$ has a double eigenvalue whose real parts are zeros. Let us examine how $f'(s)$ behaves in a neighborhood of s^*, a local minimizer. In [2], the function $f(s)$ was proved to be twice differentiable at s^*; i.e., there is a constant c such that

$$f(s) = f(s^*) + c(s - s^*)^2 + o((s - s^*)^2).$$

Thus

$$(s - s^*) = \left(\frac{f(s) - f(s^*)}{c}\right)^{1/2} + o((s - s^*)).$$

This yields

$$\begin{aligned} f'(s) &= 2c(s - s^*) + o((s - s^*)) \\ &= 2(c(f(s) - f(s^*)))^{1/2} + o((s - s^*)). \end{aligned} \tag{14}$$

Suppose that $f(s) = f(s^*) + \epsilon$, where ϵ is the machine precision. Then $f'(s) \approx 2(c\epsilon)^{1/2}$ and Theorem 3.1 implies that

$$\kappa \approx \frac{1}{\sqrt{\epsilon}}.$$

Since

$$H(\sigma) = H(f(s^*)) + \begin{bmatrix} & -\epsilon \\ \epsilon & \end{bmatrix},$$

the perturbed eigenvalue of $H(\sigma)$ with respect to $s^*\mathrm{i}$ has at least half precision accuracy. In the case of $c = 0$, we may count the third term $(s - s^*)^3$. Then the perturbed eigenvalue has at least one-third precision accuracy.

This observation leads to a stopping criterion for the checking step. Suppose that $\lambda_j(H(\sigma)), j = 1, 2, \ldots, 2n$, are the eigenvalues of $H(\sigma)$ and λ_0 is the eigenvalue with the smallest absolute value of real part. Then if

$$|\mathrm{real}(\lambda_0)| \leq \sqrt{\epsilon}, \tag{15}$$

we say that the eigenvalue λ_0 is on the imaginary axis. Otherwise we say that there are no eigenvalues of $H(\sigma)$ on the imaginary axis. The standard perturbation theory for the eigenvalues corresponding to a nonlinear elementary divisor [19] also suggests the above criterion. (In the case of $c = 0$, we say that the eigenvalue λ_0 is on the imaginary axis if $|\mathrm{real}(\lambda_0)| \leq \epsilon^{2/3}$ and not on the the imaginary axis if $|\mathrm{real}(\lambda_0)| \geq \epsilon^{1/3}$.) The following example illustrates that the criterion works well.

EXAMPLE 1. *Let*

$$A = \begin{bmatrix} -0.4 + 6\mathrm{i} & 1 & & \\ 1 & -0.1 + \mathrm{i} & 1 & \\ & 1 & -1 - 3\mathrm{i} & 1 \\ & & 1 & -5 + \mathrm{i} \end{bmatrix}.$$

The eigenvalues of A are all in the left open half-plane so that A is stable. The function $f(s)$ is shown in Figure 1 with $d_{us}(A) = 0.03188701430309$. Table 2 shows

Table 1.

tol	$1.0e-3$	$1.0e-6$	$1.0e-9$	$1.0e-12$	$-1.0e-13$
$\lvert\mathrm{real}(\lambda_0)\rvert$	$1.8e-15$	$6.3e-14$	$3.0e-12$	$6.1e-11$	$3.0e-8$
tol	$1.0e-14$	$1.0e-15$	0	$-1.0e-15$	$-1.0e-14$
$\lvert\mathrm{real}(\lambda_0)\rvert$	$9.7e-8$	$1.0e-7$	$1.0e-7$	$1.0e-7$	$1.0e-7$
tol	$-1.0e-13$	$-1.0e-12$	$-1.0e-9$	$-1.0e-6$	$-1.0e-3$
$\lvert\mathrm{real}(\lambda_0)\rvert$	$1.4e-7$	$3.2e-7$	$9.7e-6$	$3.08e-4$	$9.6e-2$

that the smallest absolute value of the real parts of the eigenvalues of $H(\sigma)$, where $\sigma = d_{us}(A)+tol$ with varying *tol*. For the purposes of this experiment, *tol* simply plays the role of a perturbation of $d_{us}(A)$, and we allow it to take negative values. Numerical computations were carried out using MATLAB version 4.2a with $eps \approx 2.22e-16$. For this example,

$$\sqrt{eps} \approx 1.4901e-8.$$

From Table 1, the stopping criterion (15) correctly makes a decision that, for *tol* in $(10^{-12}, 10^{-3})$, $H(\sigma)$ has an eigenvalue on the imaginary axis and for *tol* in $(-10^{-3}, -10^{-13})$, $H(\sigma)$ has no eigenvalue on the imaginary axis.

For this example, and for most starting points, our algorithm typically converges in about 20 iterations to the global solution. Thus, treating A as a full matrix, and assuming 4 iterations of the Boyd/Balakrishnan method are required for comparable accuracy, the new method is about twice as fast.

Finally in this section, we discuss the "flat curve" of $f(s)$ in a neighborhood of the global minimizer s^*. When $\sigma > d_{us}(A)$, $H(\sigma)$ has at least two pure imaginary eigenvalues. Suppose that $s_a\mathrm{i}$ and $s_b\mathrm{i}$ are the two of those which bracket s^*. In [2], the quadratic convergence of the bisection method was proved if the next value of the parameter is chosen as the average of s_a and s_b. The same idea is applied here. The average of s_a and s_b, where s_a and s_b are specified as the first and second smallest imaginary parts of all pure imaginary eigenvalues of $H(\sigma)$, is chosen as the s_0 for reentering step 1 and repeating the modified inverse iteration procedure, but the vector x_0 remains the same as part of the eigenvector corresponding to $s_a\mathrm{i}$.

4. Further numerical examples.

Several stable matrices are chosen here to test the Algorithm. The second example is from [1], which was originally suggested by Y. Saad, the third example is from [14, 15]; and the fourth example from [6]. They are all sparse matrices. The numerical computations are performed in MATLAB on the machine Digital Alpha 2100 with 4 processors. The computation of the eigenvalues in our code is achieved via MATLAB command *schur*, which is more accurate than the command *eig* but takes a longer time. A comparsion of the CPU times of our code with the Boyd/Balakrishnan code is also reported here, though some caution is necessary in accepting these results, as MATLAB timings may be different due to different types of implementations. Usually a good (guaranteed) accuracy of the solution is given by the Boyd/Balakrishnan method after 4 iterations. So the CPU time for the Boyd/Balakrishnan code is specified for a maximum number of 4 iterations. Note that the restriction of $s^* \geq 0$ is removed in the Boyd/Balakrishnan code so that it can work for Example 3. Here *niter* denotes the number of inverse iteration steps.

Example 2. *This matrix is the discretized Jacobian of the Brusselator wave mode for reaction and transport interaction of chemical solutions in a tubular reactor. The dimension of the matrix is* 200*, and the number of nonzero elements is* 796*.*

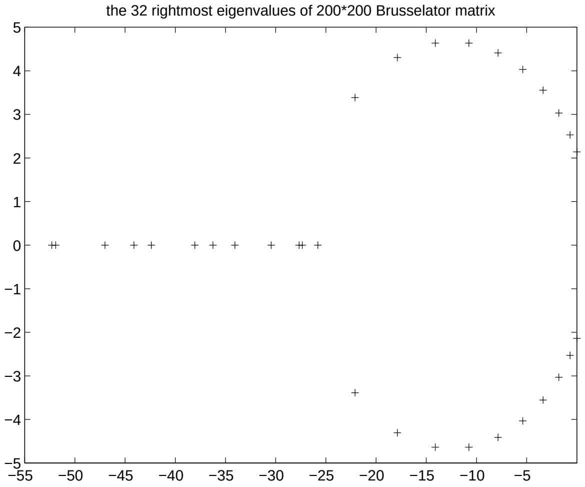


Fig. 2

Table 2
The CPU times for Example 2.

Code	Performance	CPU times
H/W	Step 1. $niter = 94$ Step 2.	$12.25s$ $63.70s$
B/B	4 iterations	$346.86s$

The spectrum of the matrix is shown in Figure 2.

The two eigenvalues closest to the imaginary axis are

$$-0.00001819987683 \pm 2.13949752207585\mathrm{i}.$$

Setting $tol = 10^{-10}$, it took the modified inverse iteration $niter = 94$ steps to compute the local minimizer $s = 2.13949748735869$. From the checking step the local minimizer is identified as being global, and the following lower bound and upper bound of $d_{us}(A)$ are returned:

$$lb = 8.240895352524869e - 06, \tag{16}$$

$$ub = 8.240995352524870e - 06. \tag{17}$$

The Boyd/Balakrishnan code after 4 iterations delivers an estimated distance which agrees with lb and ub in the first 10 digits and an estimated minimizer which agrees with s in the first 7 digits. (The estimated distance which is delivered by the Boyd/Balakrishnan code after 3 iterations agrees with lb and ub in the first 4 digits and the estimated minimizer agrees with s in the first 4 digits.) The CPU times for both codes are reported in Table 2.

Thus on the basis of these numbers, the new algorithm is 4.5 times as fast as the Boyd/Balakrishnan algorithm for this example.

Example 3. *The next example is the Orr–Sommerfeld operator for planar Poiseuille flow. Let* $L = \frac{d^2}{dx^2} - \alpha^2$, α *and* R *be positive parameters, and* $U = 1 - x^2$*; then the*

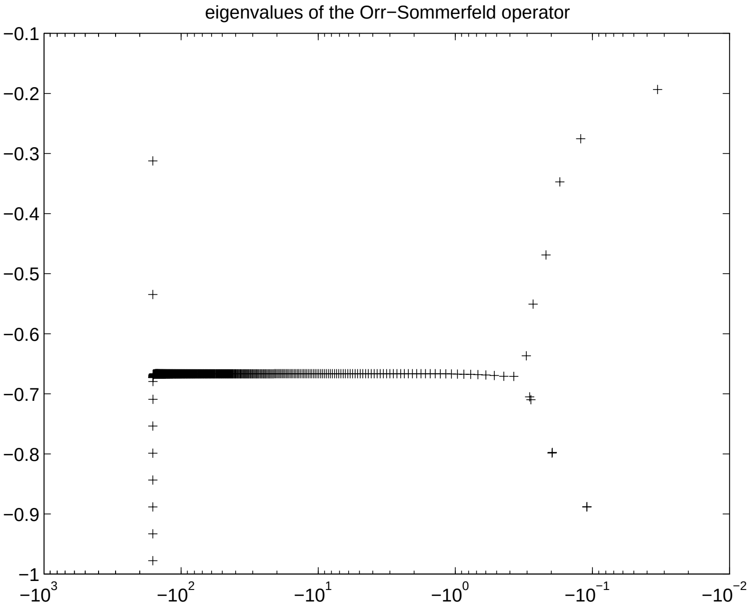


FIG. 3

Orr–Sommerfeld operator is given as follows:

$$\frac{1}{\alpha R}L^2 v - \mathsf{i}(UL - U'')v = \lambda L v,$$

where λ *is a spectral parameter and* v *is a function defined on* $[-1, 1]$*, with* $v(\pm 1) = v'(\pm) = 0$ (*see* [14, 15])*. We discretize the operator as in* [15]:

$x_k = -1 + k * h, k = 1, 2, \ldots, n$ *and* $h = \frac{2}{n+1}$.

$L_n = \frac{1}{h^2} tridiag(1, -(2 + \alpha^2 h^2), 1)$.

$U_n = diag(1 - x_1^2, \ldots, 1 - x_n^2)$.

The discretized problem is the following generalized eigenvalue problem:

$$B_n u_n = \mu L_n u_n,$$

where

$$B_n = \frac{1}{\alpha R} L_n^2 - \mathsf{i}(U_n L_n - 2I).$$

Let $\alpha = 1$*,* $R = 1000$*,* $n = 400$*, and* $A_n = L_n^{-1} B_n$.

Both B_n and L_n are sparse matrices. It follows that $(A_n - s\mathsf{i}I)x = b$ is easily converted to $(B_n - s\mathsf{i}L_n)x = L_n b$, which can be solved by calling sparse system solvers. The spectrum of A_n is shown in Figure 3. The eigenvalue of A_n closest to the imaginary line is

$$-0.03355288542026 - 0.19343672527540\mathsf{i}.$$

Since the imaginary parts of the eigenvalues close to the imaginary line are usually good choices for s_0, we start the algorithm with $tol = 10^{-6}$ and

Table 3
The CPU times for Example 3.

Code	Performance	CPU times
H/W	s_0	149.91s
	Step 1. $niter = 2404$	1566.60s
	Step 2.	4198.55s
B/B	4 iterations	16217.05

$s_0 = -0.19343672527043$. It takes 2404 inverse iteration steps to reach $s = -0.19975279875886$. and one checking step to get the

$$lb = 0.00197717232071,$$
$$ub = 0.00197817232071.$$

The Boyd/Balakrishnan code after 4 iterations delivered an estimated distance which agrees with ub in the first 10 digits and an estimated minimizer which agrees with s in the first 5 digits. Table 3 shows the CPU time (seconds) for both codes, and shows that based on these numbers, the new algorithm is about 2.7 times as fast as the Boyd/Balakrishnan algorithm for this problem. Note that here at the optimal s, $\|A - s\mathrm{i}I\|_1$ is about 160 and the 10 smallest singular values range from 0.74668... to 0.00197..., which may be considered clustered.

Example 4. *This matrix is called the Tolosa*92 *matrix and arises in the stability analysis of a model of an airplane in flight. The dimension of the matrix is* 1000. *It is a highly nonnormal matrix with* $\left\|AA^H - A^H A\right\|_1 = 2.1e + 12$. *The two eigenvalues of the Tolosa92 matrix closest to the imaginary line are*

$$-1.060000000000000e - 01 \pm 1.059999500000000e + 02\mathrm{i}.$$

Figure 4 shows the spectrum of the Tolosa92 matrix.

For the Tolosa92 matrix, the smallest singular values of $A - s\mathrm{i}I$ are clustered. For example, with $\sigma = 0.065$, $H(\sigma)$ has 128 pure imaginary eigenvalues, indicating that the singular value curves of $A - s\mathrm{i}I$ have at least 64 local minima. We start the algorithm with $tol = 10^{-6}$ and $s_0 = 1.0599995e + 02$. Then it took 1939 inverse iteration steps for s_k to reach $s = 1.059999220823240e{+}02$. One checking step returns lower bound and upper bound of $d_{us}(A)$ as

$$lb = 0.00199856927749,$$
$$ub = 0.00199956927749.$$

For this example, the CPU time for 1939 modified inverse iterations is 435.25 seconds, for computing s_0 is 113.68 seconds, and for the checking step is 7947.54 seconds. For this example, we have no comparison with the Boyd/Balakrishnan method because it exhausts the memory of the computer.

5. Concluding remarks. We have developed an algorithm for computing the distance of a stable matrix to instability which appears to improve computationally on previous methods for a range of medium to large matrices. It is based on the application of a modification of inverse iteration to descend to a stationary point of the function $f(s) = \sigma_{\min}(A - s\mathrm{i}I)$. This is then either identified as the value of $d_{us}(A)$ (to a prescribed tolerance) or a further sequence of modified inverse iteration steps is initiated, leading to an improved stationary point. A crucial part of the process

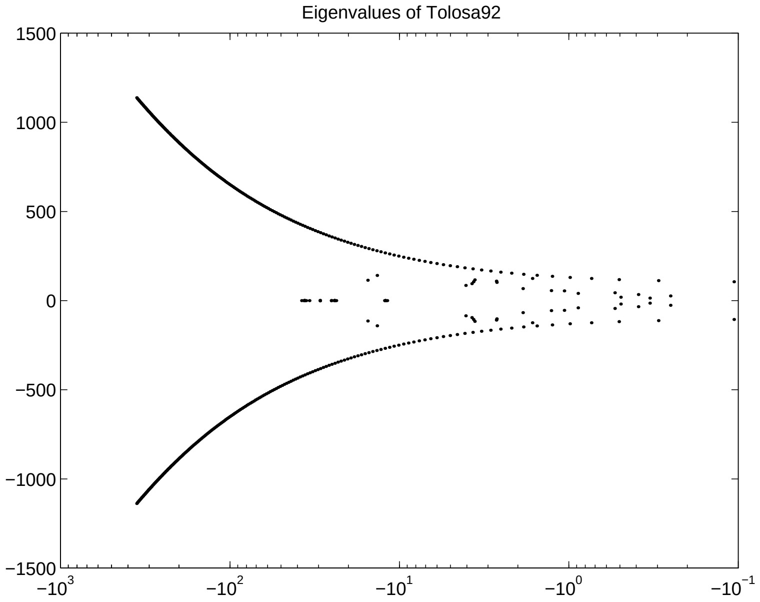


Fig. 4

is the checking stage, which involves computing the eigenvalues of a Hamiltonian matrix and making a judgement about which, if any of these, lie on the imaginary axis. Computational and theoretical considerations have shown that this is a perfectly reasonable computation which can be carried out effectively.

Any method which uses Theorem 2.2 has a limitation on the size of matrices being treated by the requirement for the eigenvalue calculation. However, approximations to the distance to instability can be obtained for very large matrices if the checking step is omitted. Although convergence to a limit point which is the global minimum is common, the possibility of running the method from different starting points would decrease the chances of failure in finding the correct value. However, this cannot be ruled out if Theorem 2.2 is not used.

The fact that, in practice, the global minimum is an attractor for the iteration process, coupled with the cheapness of the iterations, makes the present strategy an appealing one. However, the iteration process can be very slowly convergent. One modification is to limit the number of iterations before the checking step is taken, when the restart procedure gives a relatively large kick off in the right direction. This can reduce the total number of iteration but at the cost of more eigenvalue calculations, so that the total computational cost is not necessarily reduced. Another modification in the spirit of the present approach would be to work with a subset of the singular vectors which might help if the singular values are clustered. For example, one might be able to modify the calculation so that it is based on orthogonal iteration or Krylov subspaces. It is clear, however, that this would also bring additional computational cost to this part of the calculation which might offset any benefits.

It may be that completely different ways of finding a local minimum of f would be more effective, based on computing a sequence of values of $f(s)$ and applying an optimization method, possibly involving derivative values also. For any value of s, the calculation of $f(s)$ is the calculation of the smallest singular value of the matrix $A - siI$. If that singular value is simple, then $f'(s)$ is also available through (7). Based on these quantities, it would be possible to implement, say, the secant method to find a local minimum. However, the situation in which slow convergence of the present method occurs is precisely the situation in which we would expect slow convergence of inverse iteration, and although other possibilities exist, the slow convergence of the basic algorithm might be expected to effectively be passed on to any other method through difficulties in the calculation of values of $f(s)$ (and possibly $f'(s)$). Nevertheless, this may be an unnecessarily pessimistic view, and it would seem that such an optimization-based approach deserves further consideration.

To conclude, our main objective here has been to provide a method which improves existing methods and applies in a satisfactory way to a wide range of matrices. The attraction of the present approach is that it is simple and yet can be effective. However, there are issues concerning its performance that are of concern and should be addressed, and some of the ideas mentioned above might lead to improvement. This will be considered at a later date.

Acknowledgment. We are grateful to the referees for their constructive criticism, which included many helpful suggestions.

REFERENCES

[1] Z. Bai, R. Barrett, D. Day, J. Demmel, and J. Dongarra, *Test Matrix Collection (Non-Hermitian Eigenvalue Problem)*, manuscript, 1995.

[2] S. Boyd and V. Balakrishnan, *A regularity result for the singular values of a transfer matrix and a quadratically convergent algorithm for computing its L_∞-norm*, Systems Control Lett., 15 (1990), pp. 1–7.

[3] N. A. Bruinsma and M. Steinbuch, *A fast algorithm to compute the H_∞-norm of a transfer function matrix*, Systems Control Lett., 14 (1990), pp. 287–293.

[4] R. Byers, *A bisection method for measuring the distance of a stable matrix to the unstable matrices*, SIAM J. Sci. Stat. Comput., 9 (1988), pp. 875–881.

[5] J. W. Demmel, *On the condition numbers and the distance to the nearest ill-posed problem*, Numer. Math., 51 (1987), pp. 251–289.

[6] S. Godet-Thobie, *Eigenvalues of Large Highly Nonnormal Matrices.* Ph.D. thesis, Paris IX Dauphine University, Paris, 1993.

[7] G. Golub and C. Van Loan, *Matrix Computations*, 2nd ed., Johns Hopkins University Press, Baltimore, MD, 1989.

[8] C. He, *On the distance to uncontrollability and the distance to instability and their relation to some condition numbers in control*, Numer. Math., 76 (1997), pp. 463–477.

[9] C. He and G. A. Watson, *An algorithm for computing the numerical radius*, IMA J. Numer. Anal., 17 (1997), pp. 329–342.

[10] D. Hinrichsen and M. Motscha, *Optimization problems in the robustness analysis of linear state space systems.* Tech. Report 169, Institute of Dynamic Systems, Bremen University, Germany, 1987.

[11] D. Hinrichsen and A. J. Pritchard, *Stability radii of linear systems*, Systems Control Lett., 7 (1986), pp. 1–10.

[12] T. Kailath, *Linear Systems*, Prentice-Hall, Englewood Cliffs, N J, 1980.

[13] C. Kenney and G. Hewer, *The sensitivity of the algebraic and differential Riccati equations*, SIAM J. Control Optim., 28 (1990), pp. 50–69.

[14] W. Kerner, *Large-scale complex eigenvalue problem*, J. Comput. Phys., 85 (1989), pp. 1–85.

[15] A. N. Malyshev and M. Sadkane, *On the Lyapunov stability of large matrices*, manuscript, 1996.

[16] L. Qiu, B. Bernhardsson, A. Rantzer, E. J. Davison, P. M. Young, and J. C. Doyle, *A formula for computation of the real stability radius*, Automatica J. FAC, 31 (1995), pp. 879–890.

[17] C. F. Van Loan, *How near is a stable matrix to an unstable matrix?*, in Linear Algebra and Its Role in Systems Theory, Contemp. Math. 47, B. N. Datta, ed., American Mathematical Society, Providence, KL, 1985, pp. 465–478.

[18] G. A. Watson, *Computing the numerical radius*, Linear Algebra Appl., 234 (1996), pp. 163–172.

[19] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford University Press, London, 1965.

SIAM J. MATRIX ANAL. APPL.
Vol. 20, No. 1, pp. 117–130

CRITICAL GRAPHS FOR THE POSITIVE DEFINITE COMPLETION PROBLEM*

WAYNE W. BARRETT†, CHARLES R. JOHNSON‡, AND RAPHAEL LOEWY§

Abstract. Among various matrix completion problems that have been considered in recent years, the positive definite completion problem seems to have received the most attention. Indeed, in addition to being a problem of great interest, it is related to various applications as well as other completion problems. It may also be viewed as a fundamental problem in Euclidean geometry.

A partial positive definite matrix A is "critical" if A has no positive definite completion, though every proper principal submatrix does. The graph G is called critical for the positive definite completion problem if there is a critical partial positive definite matrix A, the graph of whose specified entries is G. Complete analytical understanding of the general positive definite completion problem reduces to understanding the problem for critical graphs. Thus, it is important to try to characterize such graphs. The first, crucial step toward that understanding is taken here. A novel and restrictive topological graph theoretic condition necessary for criticality is identified. The condition, which may also be of interest on pure graph theoretic grounds, is also shown to be sufficient for criticality of graphs on fewer than 7 vertices, and the authors suspect it to be sufficient in general. In any event, the condition, which may be efficiently verified, dramatically narrows the class of graphs for which completability conditions on the specified data are needed. The concept of criticality and the graph theoretic condition extend to other completion problems, such as that for Euclidean distance matrices.

Key words. critical graph, critical positive definite matrix, matrix completion problem, positive definite completion, ultraconnected graph

AMS subject classifications. 15A48, 05C50, 05C75

PII. S0895479897324573

1. Introduction.

A *partial matrix* is a rectangular array in which some entries are *specified*, while the remaining, *unspecified* entries are free to be chosen from an agreed-upon set. A *completion* of a partial matrix is a choice of values for the unspecified entries, resulting in a conventional matrix. Given a matricial property P, a matrix completion problem then asks which partial matrices have a completion with property P. It is convenient to describe the positions of the specified entries in a partial matrix $A = (a_{ij})$ with a graph G in which edges correspond to the specified entries: A is a G*-partial matrix* if a_{ij} is specified exactly when (i, j) is an edge of G. The focus then becomes, for each G, to determine conditions upon the specified entries in a G-partial matrix that characterize completability to a matrix with property P. For a wide variety of properties P (including positive definiteness), it may be shown [J2] that, for each graph G, there are finitely many conditions on the specified data (that depend upon G) that characterize completability to a matrix with property P.

*Received by the editors July 16, 1997; accepted for publication by R. Brualdi November 4, 1997; published electronically September 14, 1998.

http://www.siam.org/journals/simax/20-1/32457.html

†Department of Mathematics, Brigham Young University, Provo, UT 84602-6539 (wayne@math.byu.edu).

‡Department of Mathematics, The College of William and Mary, Williamsburg, VA 23187 (crjohnso@math.wm.edu). The work of this author was supported in part by National Science Foundation grant DMS-92-00899 and by Office of Naval Research contract N00014-90-J-1739.

§Department of Mathematics, Technion-Israel Institute of Technology, Haifa 32000, Israel (loewy@techunix.technion.ac.il). The work of this author was supported by the Fund for the Promotion of Research at the Technion. Part of the research was done while visiting Brigham Young University.

Although the concepts we introduce here are relevant to many properties P, we focus upon the real positive definite completion problem. Thus, we assume that A is square (n by n), has real specified entries, is *partial symmetric* (a_{ji} is specified, as a_{ij}, if and only if a_{ij} is specified), and is *partial positive definite* (all diagonal entries are specified, and all fully specified principal submatrices are positive definite). The positive definite completion problem asks when a partial symmetric matrix can be completed to a positive definite matrix. It has attracted a lot of attention in recent years. Indeed, in addition to being a problem of great interest, it is related to applications in probability and statistics, image enhancement, and systems engineering as well as other completion problems, like the Euclidean distance completion problem. One may also view the positive definite completion problem as a fundamental problem in Euclidean geometry, namely which potential geometric configurations of vectors are realizable in a Euclidean space. For the positive definite completion problem the graph G of the specified entries of A may be taken to be an undirected graph on n vertices, but we adopt the convention of omitting any explicit mention of self-edges at vertices. We note that there is no loss of generality in restricting attention to the real positive definite completion problem. By a variant of the conventional dimension doubling argument, understanding of positive definite completability conditions for each graph in the real case implies understanding for each graph in the complex case [K].

Our central purpose here is to show that to understand positive definite completability for all graphs, it suffices to understand it for a modest special subset of graphs. To this end we introduce the notion that a graph is critical for the positive definite completion problem. (An analogous concept for other completion problems could be defined similarly.)

An n-by-n G-partial matrix A is called *critical* if every $(n-1)$-by-$(n-1)$ principal submatrix of A has a positive definite completion but A itself does not. In this event A constitutes *critical data* for the graph G. If there exists critical data for G, then G is called a *critical graph* (for the positive definite completion problem). Importantly, we show that relatively few graphs are critical by identifying a novel graph theoretic condition (perhaps of interest by itself) they must have. The import of this is the following. If we wish to show that a given G-partial matrix A has a positive definite completion, it is an immediate logical consequence of the definition that we need only check for completability of the principal submatrices of A associated with maximal induced critical subgraphs of G. Thus, the problem of understanding positive definite completability for all graphs is reduced to that of critical graphs. This is the first major step in a natural program to fully mathematically understand the positive definite completion problem. The remaining two steps would be the efficient recognition of maximal critical subgraphs in a given graph and the understanding of completability conditions for critical graphs. (As preliminary evidence suggests that they fall into a modest number of topologically similar classes, conditions might be found for many critical graphs at once, e.g., for all cycles [BJT].) We note that computational approaches to the positive definite completion problem are also emerging [GHJT], [JKW].

In the next section we provide a review of relevant results and the graph theoretic background and notation necessary for the remainder of the discussion. Then, in section 3, we identify the 5-vertex critical graphs in order to motivate the graph theoretic condition for criticality we have discovered. This also affords an opportunity to make several useful preliminary observations. In section 4 we state and prove the main result that all critical graphs satisfy a graph theoretic condition we call "ultraconnected." We note, based upon the nature of the proof, that this graph theoretic condition is also necessary for criticality with respect to other completion problems, including the Euclidean distance completion problem. Finally, in the last section we discuss 6-vertex graphs and find that all ultraconnected graphs are critical, supporting the conjecture of the converse to our main result.

2. Background.

We first recall some concepts and set some notation from graph theory. Given a connected, undirected graph $G = (N, E)$ and $S \subseteq N$, we denote by G_S the subgraph of G induced by the vertices of S. The *edge deficiency* of $S, d_G(S) = d(S)$, is just the number of edges that must be added to G_S to obtain the complete graph on the vertices S. We say that S is a *clique* of G if $d(S) = 0$, and we call S a *k-clique* if $|S| = k$. A *maximal clique* is one that is not a proper subset of another clique. If G is connected, we denote the number of components of $G_{N-S} = G - S$ by $c(S)$. The vertex set S is called a *separator* if $c(S) \geq 2$; a *clique separator* is both a clique and a separator.

The *complete graph* on n vertices and the *cycle* on n vertices will be denoted as usual, by K_n and C_n, respectively, and the complement of $K_m \cup K_n$ by $K_{m,n}$, the "complete bipartite" graph, on m and n vertices. The graph G is called *chordal* if $C_k, k \geq 4$, does not occur as an induced subgraph of G. Given a graph G, we denote by G^+ the graph obtained from G via addition of a new vertex adjacent to all vertices of G. The *n-wheel* $W_n, n \geq 4$, is the graph C^+_{n-1}. Since $C_3 = K_3$, $W_4 = K_4$.

We abbreviate "positive definite" to PD and "positive semi-definite" to PSD and recall that $A[S]$ $(A(S))$ denotes the principal submatrix of the matrix A lying in (obtained by deleting) the rows and columns indicated by the index set S. A G-partial symmetric matrix A (with specified diagonal) is *partial* PD (PSD) if and only if $A[S]$ is PD (PSD) for every clique S of G. We refer to these as the *clique conditions.* The fundamental chordal result for the PD (PSD) completion problem is found in [GJSW].

THEOREM 2.1. *Every G-partial matrix satisfying the clique conditions has a PD (PSD) completion if and only if G is chordal.*

It is no accident that the set of graphs for the PSD and PD completion problems in Theorem 2.1 are the same. We note that, had we defined critical in terms of the PSD completion problem, the same graphs would result. The n-by-n partial matrix A is critical if and only if

$$\max_{\hat{A} \text{ completes } A} \lambda_{\min}(\hat{A}) \leq 0 < \min_{|\alpha|=n-1} \max_{\widetilde{A[\alpha]} \text{ completes } A[\alpha]} \lambda_{\min}(\widetilde{A[\alpha]}),$$

which, up to translation by a scalar multiple of I, just means that there is a gap between the left- and right-hand quantities. Thus, the critical graphs are those that permit such a gap. Criticality for the PSD problem also means that there is such a gap. Thus, as convenient, we may alternate between thinking of criticality for either background problem.

We say that the G-partial matrix A is PD (PSD)-*cycle completable* if and only if $A[S]$ has a PD (PSD)-completion whenever G_S is a cycle of G. Explicit PD (PSD)-completability conditions for a single cycle of arbitrary length are given in [BJT]. We refer to these as the *cycle conditions.*

Following [BJL], we call a graph G *cycle completable* if it has a chordal supergraph that contains no 4-clique not present in G. The cycle completable graphs include not only the chordal graphs and all cycles but much more [BJL, JM]. The main result of [BJL] is the following.

THEOREM 2.2. *Every G-partial matrix that satisfies the clique and cycle conditions has a PD (PSD) completion if and only if G is cycle completable.*

We note that if the graph G is not connected, it cannot be critical, as the positive definite completability of the principal submatrices of a G-partial matrix A associated with the components of G implies that A is completable by letting the remaining unspecified entries be 0. Thus, we restrict our attention to connected graphs throughout. On the other hand, the complete graphs $K_n, n \geq 2$, are all critical, as the n-by-n matrix

$$\begin{bmatrix} n-2 & -1 & \ldots & \ldots & -1 \\ -1 & n-2 & -1 & \ldots & -1 \\ & & \ddots & & \\ -1 & \ldots & \ldots & -1 & n-2 \end{bmatrix}$$

is not PSD, though every proper principal submatrix is PSD. Because of Theorem 2.1, no other chordal graphs can be critical.

3. The graphs on 5 vertices. We begin with some general observations about critical graphs. First, we note that C_n is critical. Clearly, C_3 is critical. For $n \geq 4$, C_n is nonchordal, so by Theorem 2.1 there is a C_n-partial matrix A satisfying the clique conditions that is not PD-completable. Since every proper induced subgraph of C_n is chordal, each proper principal submatrix of A is PD-completable by Theorem 2.1. Thus, A is critical data for G.

We call a graph G *minimally non-cycle completable* (minimally non-cc) if every proper induced subgraph of G is cycle completable but G itself is not.

THEOREM 3.1. *Any minimally non-cc graph is critical.*

Proof. Suppose G is minimally non-cc. Then by Theorem 2.2 there exists a G-partial matrix A satisfying the clique conditions and the cycle conditions that does not have a PD completion. Any proper principal submatrix B of A also satisfies the clique conditions and cycle conditions, and its graph is cycle completable, so B has a PD completion by Theorem 2.2. Thus, A is critical data for G. □

Our main result is a necessary condition for a graph G to be critical. Before stating and proving this result, it is illuminating to give an important special case.

THEOREM 3.2. *If G is critical, then G does not have a clique separator.*

Proof. Let $G = (N, E)$ be a graph with a clique separator S. Let V be one component of $G - S$, and let W be the union of the remaining components. We label the vertices N of G so that the vertices V come first, then those of S, and then those of W. Let A be a G-partial matrix for which every proper principal submatrix of A has a PD completion. We write A in partitioned form:

$$A = \begin{bmatrix} A[V] & A[V|S] & A[V|W] \\ A[S|V] & A[S] & A[S|W] \\ A[W|V] & A[W|S] & A[W] \end{bmatrix}.$$

Since S is a clique, $A[S]$ consists entirely of specified entries. But since there are no edges from V to W, $A[V|W]$ and $A[W|V]$ have only unspecified entries. Let $B_1[V \cup S]$ be a PD completion of $A[V \cup S]$, let $B_2[S \cup W]$ be a PD completion of $A[S \cup W]$, and let B be the partial matrix

$$B = \begin{bmatrix} B_1[V] & B_1[V|S] & ? \\ B_1[S|V] & A[S] & B_2[S|W] \\ ? & B_2[W|S] & B_2[W] \end{bmatrix}.$$

Then, if ij is any edge of G, $a_{ij} = b_{ij}$. Now, the graph of B has two maximal cliques, $V \cup S$ and $S \cup W$, so it is necessarily chordal. Since $B[V \cup S] = B_1[V \cup S]$ and $B[S \cup W] = B_2[S \cup W]$ are PD, B has a PD completion by Theorem 2.1, and this completion is a PD completion of A. This shows there are no critical data for G, so G is not critical. □

We now identify all critical graphs on $n \leq 5$ vertices. The graphs $K_2, K_3, K_4, K_5, C_4, C_5$ are critical. Any other critical graphs must be nonchordal. There are 5 of these, all with 5 vertices (Fig. 3.1).

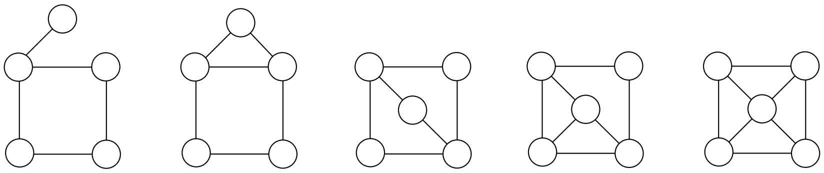


FIG. 3.1.

The first two of these have clique separators, so they are not critical by Theorem 3.2. (A cut vertex is trivially a clique separator.) The fourth and fifth graphs are minimally non-cc, and so are critical by Theorem 3.1. The remaining graph is $K_{2,3}$, which is a cycle completable graph. It follows from Theorem 2.2 that $K_{2,3}$ is not critical. Thus, the critical graphs on 5 vertices are K_5, C_5, and the last two graphs displayed above.

As $K_{2,3}$ does not have a clique separator, the converse of Theorem 3.2 is false. This example points the way to a necessary condition we believe is precise, which will be presented in the following section.

4. Main result. We begin by recalling a basic result on convex sets [E].

HELLY'S THEOREM. *Let $E_1, E_2, \ldots, E_m$ be convex sets in R^n, and assume that every collection of $n+1$ of the E_i's has a nonempty intersection. Then $\cap_{k=1}^m E_k \neq \phi$.*

DEFINITION. *We call the graph G* ultraconnected *if*

$$c(S) \leq d(S) + 1$$

for each subset S of the vertices of G.

Taking $S = \phi$ shows that $c(\phi) \leq 1$; i.e., G is connected. Moreover, if S is any clique of G, $c(S) \leq 0 + 1$, so an ultraconnected graph has no clique separator.

The graph $K_{2,3}$ is an instructive example (Fig. 4.1).

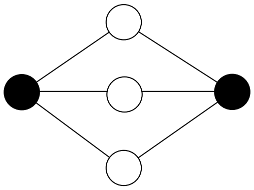


FIG. 4.1.

Let S be the set of darkened vertices in Figure 4.1. Then

$$c(S) - d(S) = 3 - 1 = 2$$

so $K_{2,3}$ is not ultraconnected.

It can be checked that each critical graph on 5 or fewer vertices is ultraconnected. Our main result is as follows.

THEOREM 4.1. *If G is a critical graph for the real positive definite completion problem, then G is ultraconnected.*

Proof. We prove the contrapositive of the statement. Let S be a set of vertices in G for which $c(S) \geq d(S) + 2$. Let $m = c(S)$ and $k = d(S)$. If $k = 0$, then S is a clique separator and G is not critical by Theorem 3.2. So we assume $k > 0$.

Let $e_1 = r_1s_1$, $e_2 = r_2s_2, \ldots, e_k = r_ks_k$ be the edges not present in G_S. Let $G_j = G_{N_j}$, $j = 1, \ldots, m$, be the components of $G - S$, and let $M_j = N_j \cup S$, $j = 1, \ldots, m$.

Let A be a G-partial matrix such that every proper principal submatrix of A has a PD completion. For each $x = (x_1, \ldots, x_k) \in R^k$, let A_x be the partial symmetric matrix obtained by replacing the r_is_i and s_ir_i entries of A by x_i, $i = 1, \ldots, k$. Now, let $E_j = \{x \in R^k : A_x[M_j]$ has a PD completion$\}$, $j = 1, \ldots, m$. Then each E_j is a convex set. Fix $\{i_1, \ldots, i_{k+1}\} \subset \{1, 2, \ldots, m\}$, and let B be a PD completion of the proper principal submatrix $A[S \cup N_{i_1} \cup N_{i_2} \cup \cdots \cup N_{i_{k+1}}]$ of A. (It is proper since $m \geq k + 2$.)

Let $x_i = b_{r_is_i}$, $i = 1, \ldots, k$, and let $x = (x_1, \ldots, x_k)$. Then $B[M_{i_j}]$ is a PD completion of $A_x[M_{i_j}]$, $j = 1, \ldots, k + 1$. That is,

$$E_{i_1} \cap E_{i_2} \cap \cdots \cap E_{i_{k+1}} \neq \phi.$$

Therefore, each collection of $k + 1$ sets of $\{E_1, \ldots, E_m\}$ has a nonempty intersection. By Helly's theorem $E_1 \cap E_2 \cap \cdots \cap E_m \neq \phi$.

Let $x \in E_1 \cap E_2 \cap \cdots \cap E_m$, and let B_j be a PD completion of $A_x[M_j]$, $j = 1, \ldots, m$. Define the n-by-n partial symmetric matrix B by

$$b_{rs} = \begin{cases} (A_x)_{rs} & \text{if } r, s \in S, \\ (B_i)_{rs} & \text{if } r, s \in M_i \text{ but } \{r, s\} \not\subset S, \\ ? & \text{otherwise.} \end{cases}$$

Since $N_1, \ldots, N_m$ are disjoint, the definition of B is unambiguous. Then the graph of B has m maximal cliques $M_1, M_2, \ldots, M_m$ with common intersection S. It follows that this graph is chordal and that B satisfies the clique conditions. By Theorem 2.1, B has a PD completion, which is a PD completion of A. This shows there are no critical data for G. Thus G is not critical, which completes the proof. □

We note that the proof of Theorem 4.1 hinges on three key facts about the property of positive definiteness: (1) any principal submatrix of a PD matrix is PD; (2) the fact that there is a chordal result (Theorem 2.1); and (3) the PD matrices form a convex set. Otherwise the proof is graph theoretic and is independent of the specific property of positive definiteness.

We can generalize Theorem 4.1 (including Theorem 3.2) as follows. We consider combinatorially symmetric partial matrices, i.e., a_{ji} is specified if and only if a_{ij} is specified, but we do not require them to be equal. Let P be a property that is defined for matrices of all orders, and let $D = \{t : t$ is an entry of some matrix with property $P\}$. We say that P is an *inherited* property if each principal submatrix of a matrix with property P has property P. Property P is said to be *chordal* if there is a chordal theorem for P, analogous to Theorem 2.1. We say that P is *implicitly convex* if there is a function f, $1 - 1$ on D, such that for each positive integer n, the set $\{(f(a_{ij})) : A = (a_{ij})$ is an n-by-n matrix with property $P\}$ is convex. Of course positive definiteness is inherited, chordal, and implicitly convex, but other properties are also. An n-by-n matrix $A = (a_{ij})$ is called a *Euclidean distance matrix* [B] if there exist points $p_1, p_2, \ldots p_n \in R^k$ (for some positive integer k) such that $a_{ij} = ||p_i - p_j||_2$, $i, j = 1, \ldots, n$. The property of being a Euclidean distance matrix is clearly inherited and is chordal [BJ]. If we set $f(x) = x^2$, which is $1-1$ on $D = [0, \infty)$, the property is seen to be implicitly convex because a matrix is a squared Euclidean distance matrix if and only if it has zero diagonal and is negative semidefinite on the orthogonal complement of the vector $(1, 1, \ldots, 1)$ [BJ]. Criticality for P is defined analogously.

Given $x \in R^k$, let $f(x) = (f(x_1), f(x_2), \ldots, f(x_k))$. Modifying the definition of E_j in the proof of Theorem 4.1 to $E_j = \{f(x) : A_x[M_j]$ has a completion with property $P\}$, yields the following result.

Observation. Let P be an inherited, chordal, implicitly convex property. If G is a critical graph for property P, then G is ultraconnected.

It follows that the critical graphs for the Euclidean distance completion problem are ultraconnected.

If we examine a list of the graphs on $n \leq 5$ vertices, we find that those that are ultraconnected are precisely the critical graphs for $n \leq 5$ identified in section 3. It is thus natural to conjecture that the converse of Theorem 4.1 holds. This is an open question; however, we examine the evidence for it in section 5.

5. Sufficiency for graphs on 6 vertices. Theorem 4.1 limits the search for critical graphs to the ultraconnected graphs. We furthermore conjecture that every ultraconnected graph is critical for the PD (PSD) completion problem. In this section we resolve this conjecture for $n = 6$.

A table of the 112 connected graphs on 6 vertices can be found in [CP]. Exactly 19 of these are ultraconnected; we list them and their numbers in Figure 5.1.

Any graph on 6 vertices that is critical for the PD completion problem must be one of these by Theorem 4.1.

We examine which of these is critical. Of course, the complete graph 1 and the cycle 106 are critical.

We next show that the graphs 28, 47, 51, 52, 69, 72, and 74 are critical. The main theorem (Theorem 3) in [BJL] gives 3 different graph theoretic conditions in order that a graph be cycle completable. Condition (1.1) of this theorem implies that each of these 7 graphs is not cycle completable, while it is straightforward to check that every proper induced subgraph of each of them is. Thus, each is minimally non-cc, and therefore critical by Theorem 3.1.

Criticality of two further graphs follows from the following general theorem.

THEOREM 5.1. *Let G be a connected graph on n vertices. Then G is critical if and only if G^+ is critical.*

Proof. Suppose first that G is critical. If $G = K_n$, then $G^+ = K_{n+1}$, so G^+ is critical. Thus, assume that $G \neq K_n$. Let B be a G-partial, critical matrix. Since $B(\{n\})$ is PD-completable, there exists $\alpha \in \mathbb{R}$ such that the G-partial matrix

$$A_{11} = B + \alpha^2 E_{nn}$$

is PD-completable.

Now define a G^+-partial matrix A by

$$A = \begin{bmatrix} A_{11} & \alpha e_n \\ \alpha e_n^t & 1 \end{bmatrix}.$$

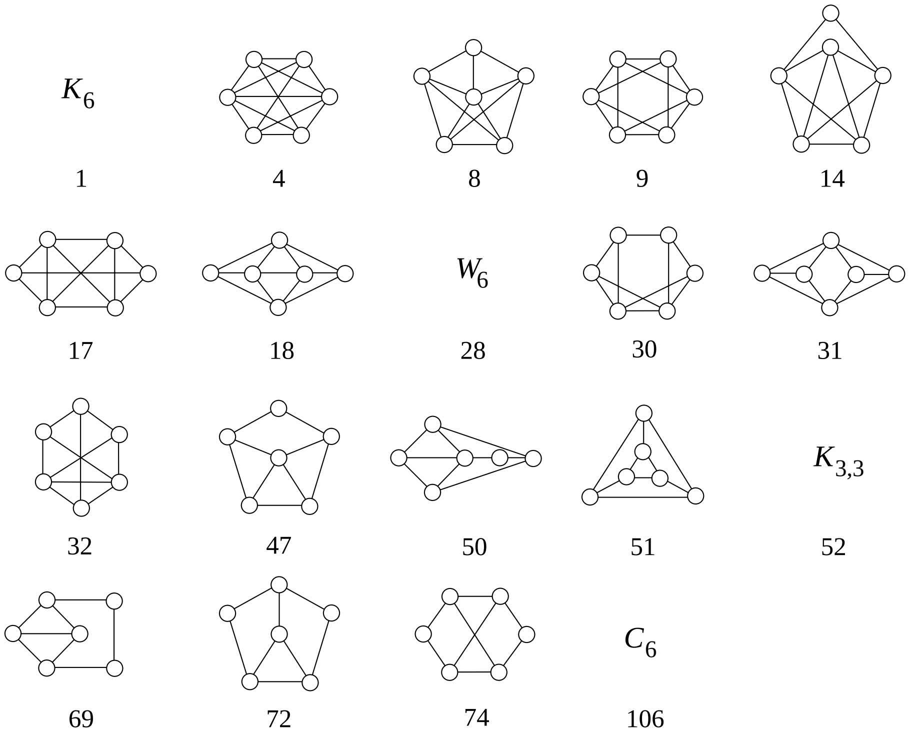


FIG. 5.1.

We claim that A is critical (with respect to G^+). Indeed, $A(\{n+1\}) = A_{11}$ is PD-completable by assumption. $A(\{n\}) = A_{11}(\{n\}) \oplus [1]$ is PD-completable because $A_{11}(\{n\})$ is. Finally let $1 \le k \le n-1$. The partial matrix $A(\{k\})$ is PD-completable because the Schur complement of its main diagonal entry in the bottom right-hand corner is equal to

$$A_{11}(\{k\}) - \alpha^2 E_{n-1,n-1} = B(\{k\}),$$

which is PD-completable by assumption.

We have shown that every n-by-n principal submatrix of A is PD-completable. Now, let

$$\tilde{A} = \begin{bmatrix} \tilde{A}_{11} & \alpha e_n \\ \alpha e_n^t & 1 \end{bmatrix}$$

be any completion of A. Then the Schur complement of the $(n+1, n+1)$ entry is

$$\tilde{A}_{11} - \alpha^2 E_{nn},$$

which is a completion of B and therefore is not positive definite. Thus A is not PD-completable, even though all its proper principal submatrices are. It follows that G^+ is critical.

Suppose now that $G = (N, E)$ is not critical, and let

$$A = \begin{bmatrix} B & c \\ c^t & d \end{bmatrix}$$

be any $(n+1)$-by-$(n+1)$ G^+-partial matrix, each of whose n-by-n principal submatrices is PD-completable. Motivated by Schur complements, we define a G-partial matrix S. Let $C = (c_{ij})$ be the G-partial matrix defined by

$$c_{ij} = \begin{cases} \frac{c_i c_j}{d} & \text{if } i = j \text{ or } ij \in E, \\ ? & \text{otherwise,} \end{cases}$$

and let $S = B - C$. (We are using the convention that the difference of two unspecified entries is unspecified.) For each k, $1 \leq k \leq n$, let

$$\hat{A}_k = \begin{bmatrix} B_k & c(\{k\}) \\ c(\{k\})^t & d \end{bmatrix}$$

be a PD completion of $A(\{k\})$. Here B_k is a completion of $B(\{k\})$ and $c(\{k\})$ is the vector obtained from c by removing its kth component. Then $B_k - \frac{1}{d}c(\{k\})c(\{k\})^t$ is a PD-completion of $S(\{k\})$. Thus, each $(n-1)$-by-$(n-1)$ principal submatrix of S is PD-completable. Since G is not critical S has a PD-completion $\hat{S}$. Then

$$\begin{bmatrix} \hat{S} + \frac{1}{d}cc^t & c \\ c^t & d \end{bmatrix}$$

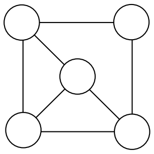
is a PD-completion of A, which shows there are no critical data for G^+. □

An immediate corollary follows.

COROLLARY 5.2. *The n-wheel W_n, $n \geq 4$, is critical.*

It also follows from Theorem 5.1 that the number of critical graphs on n vertices is a strictly increasing function of n. The rate of growth is probably exponential but likely much less than the rate of growth of the number of connected graphs.

Returning to our list of ultraconnected graphs on 6 vertices, we see that graph 4 is W_5^+ and graph 8 is $\hat{W}_4^+$, where $\hat{W}_4$ is the graph



Since W_5 and $\hat{W}_4$ are critical graphs on 5 vertices, graphs 4 and 8 are critical by Theorem 5.1.

At present we have no simple method for showing that the eight remaining ultraconnected graphs on 6 vertices are critical. Instead we have found critical data for each one individually.

Before presenting this data, we recall some elementary, but useful, results about completions.

Remark 5.1 (see [J1]). Let $a, d \in R$, $b, c \in R^{n-2}$, and let B be an $(n-2)$-by-$(n-2)$ PD matrix. Assume that

$$\begin{bmatrix} a & b^T \\ b & B \end{bmatrix}, \begin{bmatrix} B & c \\ c^T & d \end{bmatrix}$$

are PSD. Then the matrix

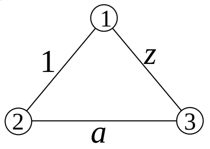
$$\begin{bmatrix} a & b^T & z \\ b & B & c \\ z & c^T & d \end{bmatrix}$$

is PSD if and only if $z \in [m-r, m+r]$, in which $m = b^T B^{-1} c$ and $r = \sqrt{(a - b^T B^{-1} b)(d - c^T B^{-1} c)}$.

Remark 5.2. The matrix

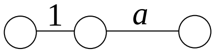
$$S = \begin{bmatrix} 1 & 1 & z \\ 1 & 1 & a \\ z & a & 1 \end{bmatrix},$$

$|a| \le 1$, is PSD if and only if $z = a$. If we represent the matrix S graphically



, $|a| \le 1,$

then this graph represents a PSD matrix if and only if $z = a$. In checking for PSD completions we will use this remark in the following form: if



, $|a| \le 1,$

occurs as an induced subgraph of a graph G corresponding to a G-partial matrix A, the entry on the remaining edge must be chosen to be a if A is to have a PSD completion.

Remark 5.3. The matrix

$$\begin{bmatrix} 1 & 1 & -\frac{1}{2} & -\frac{1}{2} \\ 1 & 1 & -\frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & -\frac{1}{2} & 1 & -\frac{1}{2} \\ -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & 1 \end{bmatrix}$$

is PSD.

Remark 5.4 (see [BJT]). Let $G = C_4$, and let A be a C_4-partial PSD matrix whose diagonal entries are all 1's. Let a be any specified entry of A. Then $|a| \le 1$, and there exists $0 \le \theta \le \pi$ such that $a = \cos\theta$. Doing likewise for any specified entry of A and arranging the θ's in decreasing order $\theta_1 \ge \theta_2 \ge \theta_3 \ge \theta_4$, the cycle conditions for completability of A to a PSD matrix are

$$\theta_1 \le \theta_2 + \theta_3 + \theta_4$$

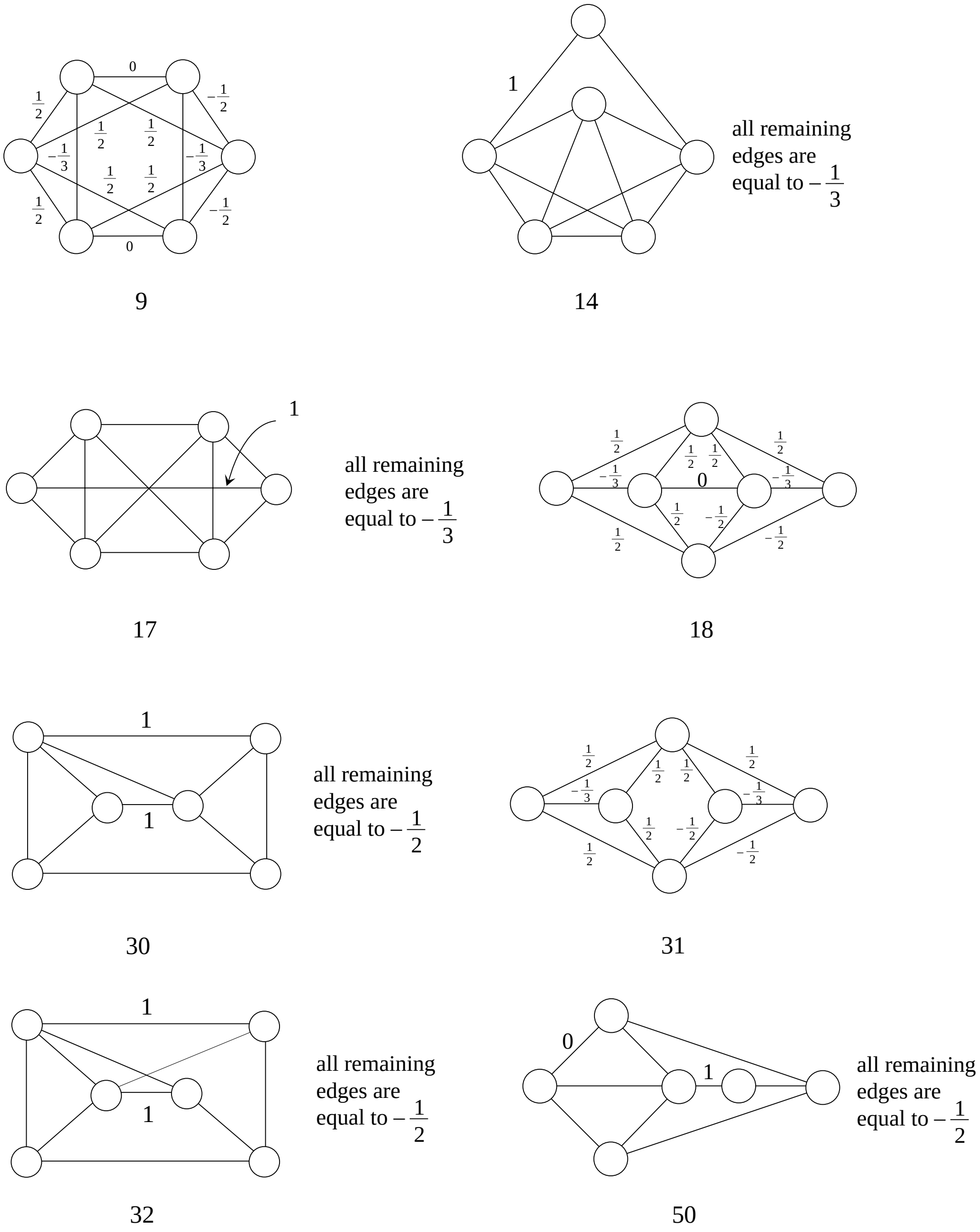
and

$$\theta_1 + \theta_2 + \theta_3 \le 2\pi + \theta_4.$$

THEOREM 5.3. *The displayed data in each of the eight graphs in Figure* 5.2 *is critical.*

Proof. We will verify that the data for graphs 14, 30, and 31 are critical. The data for the remaining graphs may be verified in a similar manner.

First consider graph 14 (Fig. 5.3). If the partial matrix A corresponding to the displayed data is PSD-completable, the 2,3- and 3,2-entries must be equal to $-\frac{1}{3}$ by Remark 5.2. But then the principal submatrix $A[\{2,3,4,5,6\}]$ has $-\frac{1}{3}$ as an eigenvalue. Therefore A has no PSD completion.



9

14

all remaining edges are equal to $-\frac{1}{3}$

17

all remaining edges are equal to $-\frac{1}{3}$

18

30

all remaining edges are equal to $-\frac{1}{2}$

31

32

all remaining edges are equal to $-\frac{1}{2}$

50

all remaining edges are equal to $-\frac{1}{2}$

FIG. 5.2.

We now show that each principal submatrix of order 5 is PSD-completable. The graphs $G-\{1\}$, $G-\{2\}$, and $G-\{3\}$ are chordal. Since $A(\{1\}), A(\{2\})$, and $A(\{3\})$ satisfy the clique conditions, each is PSD-completable by Theorem 2.1.

Now consider $A(\{4\})$. Again the 2,3- and 3,2-entries must be chosen to be $-\frac{1}{3}$,

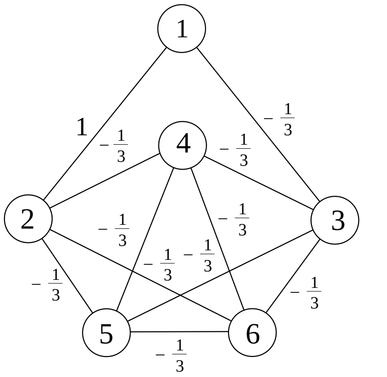


FIG. 5.3.

and with this choice $A[\{2,3,5,6\}]$ becomes

$$\begin{bmatrix} 1 & -\frac{1}{3} & -\frac{1}{3} & -\frac{1}{3} \\ -\frac{1}{3} & 1 & -\frac{1}{3} & -\frac{1}{3} \\ -\frac{1}{3} & -\frac{1}{3} & 1 & -\frac{1}{3} \\ -\frac{1}{3} & -\frac{1}{3} & -\frac{1}{3} & 1 \end{bmatrix},$$

which is PSD. It now follows that $A(\{4\})$ is completable by Theorem 2.1. The same argument shows that the matrices $A(\{5\})$ and $A(\{6\})$ are PSD-completable.

Now consider graph 30 (Fig. 5.4).

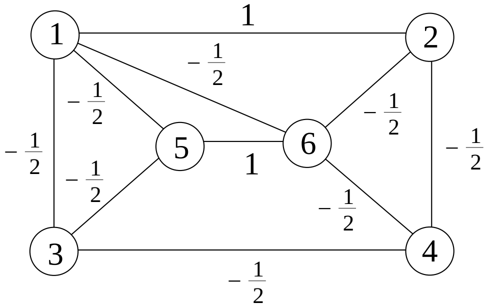


FIG. 5.4.

If the partial matrix A corresponding to the displayed data is PSD-completable, then all unspecified entries of A must be chosen to be $-\frac{1}{2}$. But then the principal submatrix $A[\{2,3,4,5\}]$ has $-\frac{1}{2}$ as an eigenvalue. Therefore A has no PSD-completion.

We now show that each principal submatrix of order 5 has a PSD completion. The graph $G-\{1\}$ is cycle completable. The partial matrix $A(\{1\})$ satisfies the clique conditions and the cycle conditions (Remark 5.4) since the θ-values corresponding to the cycle (3,4,6,5) are $\frac{2\pi}{3}, \frac{2\pi}{3}, \frac{2\pi}{3}, 0$. Therefore, $A(\{1\})$ is PSD-completable by Theorem 2.2. Now consider $A(\{2\})$. The 3,6- and 6,3-entries must be chosen to be $-\frac{1}{2}$; let A' be the matrix resulting from this choice. The graph of A' is chordal and A' satisfies the clique conditions (Remark 5.3); thus A' is PSD-completable by Theorem 2.1, and so is $A(\{2\})$. $G-\{3\}$ and $G-\{4\}$ are chordal, and $A(\{3\})$ and

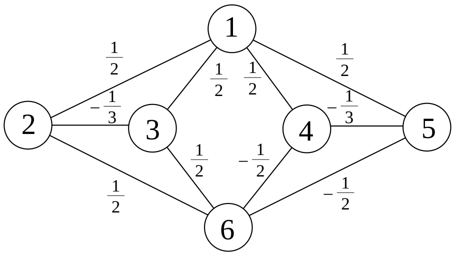


Fig. 5.5.

$A(\{4\})$ satisfy the clique conditions and so are PSD-completable. We now come to $A(\{5\})$. If we set the 1,4-and 4,1-entries equal to $-\frac{1}{2}$ we obtain a matrix whose graph is chordal and which satisfies the clique conditions. Therefore, $A(\{5\})$ is PSD-completable. Finally $G-\{6\}$ is cycle completable and $A(\{6\})$ satisfies the clique and cycle conditions and so is PSD-completable by Theorem 2.2. This completes the proof that graph 30 is critical.

Finally, we consider the graph 31 (Fig. 5.5). We show that the partial matrix A corresponding to the displayed data is not PSD-completable. By Remark 5.1, the partial matrix

$$\begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{2} & ? \\ \frac{1}{2} & 1 & -\frac{1}{3} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{3} & 1 & \frac{1}{2} \\ ? & \frac{1}{2} & \frac{1}{2} & 1 \end{bmatrix}$$

corresponding to $G_{\{1,2,3,6\}}$ becomes PSD if and only if $a_{16} = a_{61}$ is chosen to be in the interval $[\frac{1}{2}, 1]$ ($m = \frac{3}{4}$ and $r = \frac{1}{4}$). Now considering $G_{\{1,4,5,6\}}$, a similar argument shows that the corresponding partial matrix $A[\{1,4,5,6\}]$ becomes PSD if and only if $a_{61} = a_{16}$ is chosen in the interval $[-1, -\frac{1}{2}]$. This shows that A is not PSD-completable.

We now show that each principal submatrix of order 5 has a PSD completion. $G-\{1\}$ is chordal, and the corresponding partial matrix $A(\{1\})$ satisfies the clique conditions. Hence, by Theorem 2.1, $A(\{1\})$ is PSD-completable. A similar argument shows that $A(\{6\})$ is PSD-completable.

Now consider the partial matrix $A(\{2\})$. Letting $a_{16} = a_{61} = -\frac{1}{2}$, the (ordinary) matrix $A[\{1,4,5,6\}]$ is PSD, as indicated above. The graph corresponding to the new partial matrix A', obtained from $A(\{2\})$ by making this choice, is chordal, and since A' satisfies the clique conditions, it is PSD-completable by Theorem 2.1. Hence $A(\{2\})$ is PSD-completable. The same reasoning shows that $A(\{3\})$ is PSD-completable.

Now consider the partial matrix $A\{5\}$. Letting $a_{16} = a_{61} = \frac{1}{2}$, the (ordinary) matrix $A[\{1,2,3,6\}]$ is PSD, as indicated above. The graph corresponding to the new partial matrix A', obtained from $A(\{5\})$ by letting $a_{16} = a_{61} = \frac{1}{2}$, is chordal, and since A' satisfies the clique conditions, it is PSD-completable by Theorem 2.1. Hence $A(\{5\})$ is PSD-completable. The same reasoning shows that $A(\{4\})$ is PSD-completable, completing the proof that graph 31 is critical. □

Theorem 5.3 and the preceding remarks establish that the converse of Theorem 4.1 holds for $n = 6$. The report [PZ] treats this question for $n = 7$. Of the graphs on 7 vertices, 136 were found to be ultraconnected, and critical data were produced for many of them.

Acknowledgments. We thank Mohammad Omran for a chart he prepared of all connected graphs on 6 vertices with no clique separator, which was helpful in our initial investigation of this problem. Wayne Barrett gratefully acknowledges the support of Brigham Young University for travel to Williamsburg and Haifa, where part of this research was conducted. Raphael Loewy thanks Brigham Young University for its support.

REFERENCES

[BJ] M. Bakonyi and C. R. Johnson, *The Euclidean distance matrix completion problem*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 646–654.

[BJL] W. Barrett, C. R. Johnson, and R. Loewy, *The real positive definite completion problem: Cycle completability*, Mem. Amer. Math. Soc., 584 (1996), 71 pp.

[BJT] W. Barrett, C. R. Johnson, and P. Tarazaga, *The real positive definite completion problem for a simple cycle*, Linear Algebra Appl., 192 (1993), pp. 3–31.

[B] L. Blumenthal, *Theory and Applications of Distance Geometry*, 2nd ed., Chelsea, New York, 1970.

[CP] D. Cvetković and M. Petrić, *A table of connected graphs on six vertices*, Discrete Math., 50 (1984), pp. 37–49.

[E] H. G. Eggleston, *Convexity*, Cambridge University Press, London, 1958, p. 33.

[GHJT] W. Glunt, T. Mayden, C. R. Johnson, and P. Tarazaga, *Positive definite completions and determinant maximization*, Linear Algebra Appl., to appear.

[GJSW] R. Grone, C. R. Johnson, E. Sá, and H. Wolkowicz, *Positive definite completions of partial Hermitian matrices*, Linear Algebra Appl., 58 (1984), pp. 109–124.

[J1] C. R. Johnson, *Matrix completion problems: A survey*, in Proc. Sympos. Appl. Math., 40, AMS, Providence, RI, 1990, pp. 171–198.

[J2] C. R. Johnson, *On the Solvability of Matrix Completion Problems*, in preparation.

[JKW] C. R. Johnson, B. Kroschel, and H. Wolkowicz, *Interior point methods for approximate positive semidefinite completions*, Comput. Optim. Appl., 9 (1998), pp. 175–190.

[JM] C. R. Johnson and T. McKee, *Structural conditions for cycle completable graphs*, Discrete Math., 159 (1996), pp. 155–160.

[K] B. K. Kroschel, *Structured Eigenvectors, Interlacing, and Matrix Completions*, Ph.D. thesis, The College of William and Mary, Williamsburg, VA, 1996.

[PZ] S. Petrovic and S. Zepnick, *Critical Data and Ultra-Connected Graphs*, REU Report, The College of William and Mary, Williamsburg, VA, 1995. Advisor: C.R. Johnson.

SIAM J. MATRIX ANAL. APPL.
Vol. 20, No. 1, pp. 131–148

USING GENERALIZED CAYLEY TRANSFORMATIONS WITHIN AN INEXACT RATIONAL KRYLOV SEQUENCE METHOD*

R. B. LEHOUCQ† AND KARL MEERBERGEN‡

Abstract. The rational Krylov sequence (RKS) method is a generalization of Arnoldi's method. It constructs an orthogonal reduction of a matrix pencil into an upper Hessenberg pencil. The RKS method is useful when the matrix pencil may be efficiently factored. This paper considers approximately solving the resulting linear systems with iterative methods. We show that a Cayley transformation leads to a more efficient and robust eigensolver than the usual shift-invert transformation when the linear systems are solved inexactly within the RKS method. A relationship with the recently introduced Jacobi–Davidson method is also established.

Key words. generalized eigenvalue problem, rational Krylov sequence, Arnoldi method, eigenvalues, Cayley transformation

AMS subject classifications. 65F15, 65F50, 65N25, 65G05

PII. S0895479896311220

1. Introduction.

Suppose that a few eigenvalues, near a complex number μ, and possible corresponding eigenvectors of the generalized matrix eigenvalue problem

$$Ax = Bx\lambda \tag{1.1}$$

are needed. Assume that both A and B are large complex matrices of order n. Also, suppose that at least one of A or B is nonsingular so that equation (1.1) has n eigenvalues. Without loss of generality, assume that B is invertible. Following standard convention, we refer to (A, B) as a matrix pencil. For us, n is considered large when it is prohibitive to compute all the eigenvalues such as a dense algorithm in LAPACK [1] would attempt to do.

A standard approach is to perform inverse iteration [17, p. 386] with the matrix $A - \mu B$. The sequence of iterates

$$v,\ (A-\mu B)^{-1}Bv,\ [(A-\mu B)^{-1}B]^2 v,\ \dots \tag{1.2}$$

is produced. Under some mild assumptions, the sequence converges toward the desired eigenvector with eigenvalue closest to μ, and a Rayleigh quotient calculation gives an estimate of the eigenvalue. Another approach is to extract the approximate eigenpair by using the information from the subspace defined by joining together m iterates of the sequence (1.2). This leads to a straightforward extension [22] of the ideas introduced by Ericsson and Ruhe [13] for the spectral (shift-invert) transformation

*Received by the editors October 28, 1996; accepted for publication (in revised form) by A. Greenbaum February 16, 1998; published electronically September 23, 1998.
http://www.siam.org/journals/simax/20-1/31122.html

†Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439. Current address: Sandia National Laboratories, MS 1110, P.O. Box 5800, Albuquerque, NM 87185-1110 (rlehoucq@cs.sandia.gov). The work of this author was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Computational and Technology Research, U.S. Department of Energy, under contract W-31-109-Eng-38.

‡Rutherford Appleton Laboratory, Chilton, Didcot, Oxon, OX11 0QX, UK (K.A.Y.Meerbergen@rl.ac.uk). The work of this author was supported by the project *Iterative Methods in Scientific Computing*, contract HCM network CHRC-CT93-0420, coordinated by CERFACS, Toulouse, France.

Lanczos method. Starting with the vector v, Arnoldi's method [2] builds, step by step, an orthogonal basis for the *Krylov* subspace

$$\mathcal{K}_m(T^{SI}, v) \equiv \text{Span}\{v,\ T^{SI}v,\ \ldots,\ (T^{SI})^{m-1}v\} \quad \text{where} \quad T^{SI} = (A - \mu B)^{-1}B.$$

One improvement to the inverse iteration scheme given is to possibly vary the shift $\mu \equiv \mu_j$ at every step. For example, μ_j may be set to the Rayleigh quotient z^HAz/z^HBz, where z is a unit vector in the direction of $(T^{SI})^jv$. Ruhe [30, 32] elegantly shows how to build an orthogonal basis for the *rational Krylov* subspace

$$\text{Span}\{v,\ T_1^{SI}v,\ \ldots\ ,\ (T_{m-1}^{SI}\cdots T_1^{SI})v\}, \quad \text{where} \quad T_j^{SI} = (A - \mu_j B)^{-1}B.$$

The resulting algorithm is called a rational Krylov sequence (RKS) method and is a generalization of the shift-invert Arnoldi method, where the shift is possibly varied during each step.

All the methods considered require the solution of $(A - \mu B)x = By$ for x. This is typically accomplished by factoring $A - \mu B$. For example, when $A - \mu B$ is sparse, a direct method [7, 8, 9, 10, 12, 11] may be employed. If the shifts μ_j are not varied, then use of one of these direct methods in conjunction with ARPACK [21] is a powerful combination for computing a few solutions of the generalized eigenvalue problem (1.1).

However, for large eigenvalue problems ($n > 10,000$), direct methods using the RKS method may not provide an efficient solution because of the potentially prohibitive storage requirements. The motivation for the current study is to investigate the use of iterative methods for the linear systems of equations arising in the RKS method. One benefit is that for the many eigenvalue problems arising from a discretization of partial differential equations, an intelligent preconditioner may often be constructed. We shall call these methods *inexact* RKS methods because we no longer have a rational Krylov space. In particular, we shall demonstrate that a Cayley transformation $T_j^C \equiv (A - \mu_j B)^{-1}(A - \nu_j B)$ performs more robustly than a shift-invert transformation $T_j^{SI} \equiv (A - \mu_j B)^{-1}B$ when using iterative methods for the linear solves.

Before we continue, some remarks are in order. Although combining an eigensolver (using one of the methods discussed previously) with an iterative method for the linear solves is not a new (or even novel) idea, what is generally not appreciated is that residuals of the linear systems must be small. To be precise, the matrix vector product $v = T_j^{SI}u$ must be applied so that $\|Bu - (A - \mu_j B)v\| \approx \epsilon_M \|A - \mu_j B\|$, where v is the approximate solution of the linear system and ϵ_M is machine precision. *This is a necessary requirement for the correct representation of the underlying Krylov subspace.* If the linear systems are not solved with the above accuracy, there is no guarantee that a Krylov space for T_j^{SI} has been generated. For example, if Arnoldi's method is used, there is no reason to expect that the Hessenberg matrix generated represents the orthogonal projection of T_j^{SI} onto the Arnoldi vectors generated. If the above assumption of accuracy is violated (as is often the case), any results produced by such an eigensolver should be taken with caution.

Fittingly, the literature on approaches for finding a few solutions to the generalized eigenvalue problem (1.1), where only approximate solutions to the linear systems are available, is sparse. Bramble, Knyazev, and Pasciak [4], Knyazev [18], Knyazev and Skorokhodov [19], Morgan [25], and Szyld [42] each consider the situation where the matrix pencil is symmetric positive definite. (The papers [18, 19, 4] also contain numerous citations of the Russian literature.) Algorithms based on the Jacobi–Davidson method [38] introduced by Sleijpen and van der Vorst are discussed in [14, 39]. In a recent report, Sorensen [41] discusses methods based on truncating a QZ iteration. The recent paper by Meerbergen and Roose [23] provided motivation for the current article. They demonstrate the superior numerical performance of a Cayley transformation over that of a shift-invert transformation within an Arnoldi method when using an iterative linear solver.

Our paper is organized as follows. We introduce the RKS method in section 2. The inexact RKS method is introduced in section 3, along with its connection with inverse iteration, and some examples illustrating our ideas are presented. In section 4, we illustrate our method for a generalized eigenvalue problem. In section 5, we show that an appropriate (approximate) shift-invert transformation could be used. We compare inexact RKS and Jacobi–Davidson methods in section 6. We conclude the paper in section 7 with a summary of the main ideas and some remaining questions.

In this paper, matrices are denoted by uppercase Roman characters. Vectors are denoted by lowercase Roman characters. The range of the matrix V is denoted by $\mathcal{R}(V)$. The Hermitian transpose of the vector x is denoted by x^H. Specific notation will be introduced and employed in the next few sections. The norm $\|\cdot\|$ used is Euclidean.

2. The rational Krylov sequence method.

The method is outlined by the algorithm listed in Figure 2.1. For the practical RKS algorithm given in [32], Ruhe considers the shift-invert transformation $T_j^{SI} = (A - \mu_j B)^{-1} B$ rather than $T_j^C = (A - \mu_j B)^{-1}(A - \nu_j B)$. In exact arithmetic, both transformations lead to the same rational Krylov space because

$$T_j^C = I + (\mu_j - \nu_j) T_j^{SI}. \tag{2.1}$$

However, in finite-precision arithmetic and/or in conjunction with iterative methods for linear systems, substantial differences may exist (see [23] for examples). We call the μ_j's the poles, the ν_j's the zeros, and the $V_j t_j$'s the continuation vectors. A discussion of some possible choices is postponed until section 3.2. This section will discuss some relationships among quantities in steps 1–7 of Figure 2.1, the form of Gram–Schmidt orthogonalization we employ, and finally the computation of approximate eigenpairs and their convergence.

By eliminating w from steps 2–5, we obtain the relationship

$$(A - \mu_j B)^{-1}(A - \nu_j B) V_j t_j \equiv V_{j+1} \tilde{h}_j, \tag{2.2}$$

where $\tilde{h}_j = \begin{bmatrix} h_{1,j} & h_{2,j} & \cdots & h_{j+1,j} \end{bmatrix}^T$. Let $\tilde{t}_j = \begin{bmatrix} t_j^T & 0 \end{bmatrix}^T$. Rearranging equation (2.2) results in

$$\begin{aligned} (A - \mu_j B) V_{j+1} \tilde{h}_j &= (A - \nu_j B) V_{j+1} \tilde{t}_j, \\ A V_{j+1} (\tilde{h}_j - \tilde{t}_j) &= B V_{j+1} (\tilde{h}_j \mu_j - \tilde{t}_j \nu_j). \end{aligned}$$

By putting together the relations for $j = 1, \ldots, m$, we have that

$$A V_{m+1} (\tilde{H}_m - \tilde{T}_m) = B V_{m+1} (\tilde{H}_m M_m - \tilde{T}_m N_m), \tag{2.3}$$

where $\tilde{h}_j$ and $\tilde{t}_j$ are associated with the jth columns of $\tilde{H}_m$ and $\tilde{T}_m$, respectively, and $M_m = \mathrm{diag}(\mu_1, \ldots, \mu_m)$, $N_m = \mathrm{diag}(\nu_1, \ldots, \nu_m)$.

A final simplification is to rewrite equation (2.3) as

$$A V_{m+1} \tilde{L}_m = B V_{m+1} \tilde{K}_m, \tag{2.4}$$

```
• Choose a starting vector $v_1$ with $\|v_1\| = 1$ and set $V_1 \leftarrow [\ v_1\ ]$.
• For $j = 1, 2, \dots, m$
    1. Select a pole $\mu_j$, a zero $\nu_j \neq \mu_j$ and a vector $t_j \in \boldsymbol{R}^j$ with $\|t_j\| = 1$.
    2. Form $w = V_j t_j$ (continuation vector).
    3. Form $w \leftarrow (A - \mu_j B)^{-1}(A - \nu_j B)w$.
    4. Orthogonalize $w := w - V_j h_j$ with $h_j = V_j^H w$ (Gram–Schmidt).
    5. Set $V_{j+1} = \left[\ V_j \quad w/h_{j+1,j}\ \right]$ where $h_{j+1,j} = \|w\|$.
    6. Set $\tilde{l}_j = \begin{bmatrix} h_j \\ h_{j+1,j} \end{bmatrix} - \begin{bmatrix} t_j \\ 0 \end{bmatrix}$ and $\tilde{k}_j = \begin{bmatrix} h_j \\ h_{j+1,j} \end{bmatrix} \mu_j - \begin{bmatrix} t_j \\ 0 \end{bmatrix} \nu_j$.
    7. If $j > 1$ set $\tilde{L}_j = \begin{bmatrix} \tilde{L}_{j-1} & \tilde{l}_j \\ 0 & \end{bmatrix}$ and $\tilde{K}_j = \begin{bmatrix} \tilde{K}_{j-1} & \tilde{k}_j \\ 0 & \end{bmatrix}$.
    8. Compute approximate eigenpairs of interest.
    9. Check whether the approximate eigenpairs satisfy the convergence
       criterion.
```

Fig. 2.1. *Computing the Rational Krylov Sequence (RKS) for the matrix pencil (A,B).*

where

$$\tilde{L}_m \equiv \tilde{H}_m - \tilde{T}_m \ \text{ and } \ \tilde{K}_m \equiv \tilde{H}_m M_m - \tilde{T}_m N_m. \tag{2.5}$$

We remark that as long as the subdiagonal elements (the $h_{j+1,j}$'s) are nonzero, both $\tilde{H}_m$ and $\tilde{L}_m$ are unreduced upper Hessenberg (rectangular) matrices and hence are of full rank.

2.1. Orthogonalization. The orthogonalization of step 3 of the algorithm in Figure 2.1 is performed using an iterative classical Gram–Schmidt algorithm. This is the same approach used by Sorensen [40] based on the analysis [5] of reorthogonalization in the Gram–Schmidt algorithm.

2.2. Computing eigenvalue estimates. We now consider the calculation of approximate eigenpairs for the RKS method and first discuss how to compute Ritz pairs. The main purpose of this paper is to study the use of iterative linear system solvers in RKS and not the various ways to extract eigenvalues. Therefore, we use standard Ritz values throughout, although the theory can easily be extended to harmonic [32, 39] Ritz values.

Consider a matrix C and a subspace $\mathcal{R}(X)$, where $X \in \boldsymbol{C}^{n\times k}$ is of full rank. The pair $(\theta, y \equiv Xz)$ is called a Ritz pair of C with respect to the subspace $\mathcal{R}(X)$ if and only if

$$Cy - \theta y \perp \mathcal{R}(X)\ . \tag{2.6}$$

This is referred to as a Galerkin projection. Two important properties of a Galerkin projection are the following. First, if $\mathcal{R}(X) \equiv \boldsymbol{C}^n$, the Ritz pairs are exact eigenpairs of C. Second, if C is normal, the Ritz values lie in the convex hull of the eigenvalues of C. For example, if C is Hermitian, the Ritz values lie between the smallest and largest eigenvalue of C.

The following theorem shows how Ritz pairs may be computed from the RKS method outlined by the algorithm listed in Figure 2.1.

THEOREM 2.1. *$(\theta, y \equiv V_{m+1}\tilde{L}_m z)$ is a Ritz pair for $B^{-1}A$ with respect to $\mathcal{R}(V_{m+1}\tilde{L}_m)$ if and only if*

$$\tilde{L}_m^H \tilde{K}_m z = \theta \tilde{L}_m^H \tilde{L}_m z. \tag{2.7}$$

Proof. Following the definition (2.6) and equation (2.4), (θ, y) is a Ritz pair when

$$B^{-1}AV_{m+1}\tilde{L}_m z - \theta V_{m+1}\tilde{L}_m z = V_{m+1}(\tilde{K}_m - \theta\tilde{L}_m)z \perp \mathcal{R}(V_{m+1}\tilde{L}_m).$$

Thus, $(V_{m+1}\tilde{L}_m)^H V_{m+1}(\tilde{K}_m - \theta\tilde{L}_m)z = 0$, and the desired equivalence with (2.7) follows. □

We denote by $\theta_i^{(m)}$ the ith Ritz value available after m steps of the RKS algorithm of Figure 2.1. Unless otherwise stated, we assume that the Ritz values are in increasing distance from μ_m, that is, $|\theta_1^{(m)} - \mu_m| \le |\theta_2^{(m)} - \mu_m| \le \cdots \le |\theta_m^{(m)} - \mu_m|$. The associated Ritz vector is denoted by $y_i^{(m)}$. The sub- and superscripts are omitted whenever their meanings are clear from the context.

2.2.1. Computing Ritz pairs. The generalized eigenvalue problem (2.7) may be solved as a standard one. Since $\tilde{L}_m$ is an unreduced upper Hessenberg matrix, $\tilde{L}_m$ is of full rank, and hence $\tilde{L}_m^H\tilde{L}_m$ is invertible. Thus, the standard eigenvalue problem

$$\tilde{L}_m^\dagger \tilde{K}_m z = z\theta, \quad \text{where} \quad \tilde{L}_m^\dagger = (\tilde{L}_m^H\tilde{L}_m)^{-1}\tilde{L}_m,$$

is solved giving the $1 \le i \le m$ eigenpairs $(\theta_i^{(m)}, z_i^{(m)})$. We remark that $\tilde{L}_m^\dagger$ is the Moore–Penrose generalized inverse of $\tilde{L}_m$. The explicit formation of the inverse of $\tilde{L}_m^H\tilde{L}_m$ is not required. Instead, $\tilde{L}_m^\dagger\tilde{K}_m$ may be computed by least squares methods, for example, with the LAPACK [1] software. The Ritz vector is $y_i^{(m)} = V_{m+1}\tilde{L}_m z_i^{(m)}$, where $\|y_i^{(m)}\| = 1$. Sub- and superscripts are omitted when their meanings are clear in the context.

2.3. Stopping criterion. The accuracy of a Ritz pair $(\theta, y = V_{m+1}L_m z)$ is typically estimated by the residual norm $\|Ay - By\theta\|$. From equation (2.4), it follows that

$$f_i^{(m)} \equiv Ay_i^{(m)} - \theta_i^{(m)}By_i^{(m)} = BV_{m+1}(\tilde{K}_m - \theta\tilde{L}_m)z_i^{(m)} \equiv BV_{m+1}g_i^{(m)}, \tag{2.8}$$

where $g_i^{(m)} \equiv \tilde{K}_m z_i^{(m)} - \theta\tilde{L}_m z_i^{(m)}$. Thus, a simple check for convergence of a Ritz pair in the algorithm in Figure 2.1 is when

$$\|g_i^{(m)}\| \le \text{TOL}$$

for a user-defined error tolerance TOL.

For any Ritz pair (θ, y), it follows that $(A + E)y = By\theta$, where $E = -fy^H$. Hence if $\|B^{-1}E\| = \| - V_{m+1}gy^H\| = \|g\|$ is small relative to $\|B^{-1}A\|$, then (θ, y) is an eigenpair for a nearby problem. If θ is not a poorly conditioned eigenvalue of the matrix pencil and if $\|B^{-1}\|$ is not large, then the size of $\|g\|$ indicates the accuracy of the computed Ritz value.

This conclusion motivates us to say that the sequence of Ritz pairs $(\theta_i^{(m)}, y_i^{(m)})$, (fixed i) *converges* toward an eigenpair of equation (1.1) if and only if $\|g_i^{(m)}\|$ tends to zero as m increases toward n. Although this convergence is not rigorously defined (we necessarily have $\|g_i^{(n)}\| = 0$), it does allow us to track the progress of a Ritz pair after step m of the algorithm in Figure 2.1.

3. The inexact RKS method. At steps 3–5 of the RKS algorithm in Figure 2.1, the Cayley transformation

$$V_{j+1}\tilde{h}_j = (A - \mu_j B)^{-1}(A - \nu_j B)V_j t_j$$

is computed by a two-step process. First, the linear system

$$(A - \mu_j B)w = (A - \nu_j B)V_j t_j \tag{3.1}$$

is solved for w. Next, w is orthogonalized against V_j, and the solution $V_{j+1}\tilde{h}_j$ results. These two steps account for the largest source of errors arising when computing in floating-point arithmetic. Since our interest is in using a (preconditioned) iterative method for the solution of equation (3.1), we neglect the errors in the Gram–Schmidt orthogonalization phase (but we assume that the columns of V_{j+1} are orthogonal to machine precision).

Let us formally analyze the errors arising from the solution of equation (3.1). Let $x_j = V_{j+1}\tilde{h}_j$ denote the computed solution and let $s_j \equiv (A - \nu_j B)V_j t_j - (A - \mu_j B)x_j$ denote the associated residual. Thus,

$$(A - \mu_j B + s_j x_j^H/\|x_j\|^2)x_j = (A - \nu_j B)V_j t_j.$$

Here, $\|s_j x_j^H\|/\|x_j\|^2 = \|s_j\|/\|x_j\|$. If $\|s_j\|/(\|x_j\|\;\|A - \mu_j B\|)$ is a modest multiple of machine precision, we say that the direct method computes a *backward* stable solution. A robust implementation of a direct method gives a backward stable solution to a linear system. Note that even if a backward stable solution x_j is in hand, it may share few, if any, digits of accuracy with w. Moreover, achieving such a backward stable solution with an iterative method may be prohibitively expensive. Therefore, we shall study the situation where a *large* backward error is allowed for the solution of the linear system.

In order to give an indication of what we mean by *large*, a few words about iterative linear system solvers are needed. A linear system $Cx = b$ is said to be solved with a *relative residual tolerance* τ when the solution x satisfies $\|b - Cx\| \le \tau\|b\|$ for any b. Krylov methods [15, 34] are typically used. GMRES [35], BiCGSTAB(ℓ) [37], and QMR [16] are among those most widely used; see [3] for templates for all of these solvers. The performance of these solvers substantially improves when a suitable preconditioner is employed. Hence what we mean by a large error is that $10^{-8} \le \tau \le 10^{-2}$.

By putting together all s_j for $j = 1, \dots, m$ in $S_m \equiv \begin{bmatrix} s_1 & \cdots & s_m \end{bmatrix}$, we have

$$AV_{m+1}\tilde{L}_m = BV_{m+1}\tilde{K}_m - S_m, \tag{3.2}$$

which we call an inexact rational Krylov sequence (I-RKS) relation. This relation may be rewritten as

$$(A + E_m)V_{m+1}\tilde{L}_m = BV_{m+1}\tilde{K}_m \quad \text{with} \quad E_m = S_m\tilde{L}_m^\dagger V_{m+1}^H, \tag{3.3}$$

where $\tilde{L}_m^\dagger = (\tilde{L}_m^H\tilde{L}_m)^{-1}\tilde{L}_m^H$ is the generalized Moore–Penrose inverse. In other words, we have computed an exact RKS for the pencil $(A + E_m, B)$. We caution the reader not to confuse the E_m's with the unsubscripted E's of subsection 2.3.

Denote by $\sigma_{\min}^{-1}(\tilde{L}_m)$ the reciprocal of the minimum singular value of $\tilde{L}_m$. Therefore, if

$$\|E_m\| \le \|S_m\|\;\|\tilde{L}_m^\dagger\| = \|S_m\|\sigma_{\min}^{-1}(\tilde{L}_m)$$

```
• Choose a starting vector $v_1$ with $\|v_1\| = 1$ and set $V_1 \leftarrow [\ v_1\ ]$.
• For $j = 1, 2, \ldots, m$
    1. Select a pole $\mu_j$. If $j > 1$ set the zero $\nu_j = \theta_i^{(j-1)}$ and $t_j = \tilde{l}_{j-1}/\|\tilde{l}_{j-1}\|$.
       Otherwise, set $\nu_1 = 0$ and $t_1 = 1$.
    2. Compute the continuation vector $r^{(j-1)} = AV_j t_j - \nu_j BV_j t_j$.
    3. Form by solving $\left\{\begin{array}{ll} \text{(Cayley):} & (A - \mu_j B)x_j = r^{(j-1)} \\ \text{(Shift-invert):} & (A - \mu_j B)x_j = BV_j t_j \end{array}\right\}$ for $x_j$ and
       set $w = x_j$.
    4. See steps 4–6 of the IC-RKS method listed in Figure 3.2.
```

FIG. 3.1. *Inverse iteration via the inexact RKS method.*

is large, then the Ritz pairs from subsection 2.2.1 may not be those of a pencil near (A, B). This situation implies that even if we use a direct method for the linear systems, a nearly rank deficient $\tilde{L}_m$ might lead to inaccurate Ritz pairs. The matrix E_m incorporates the backward error of the linear solution and is the distance to the matrix pencil (A, B). We call the Ritz pairs for $(A + E_m, B)$ (from subsection 2.2.1) *inexact* Ritz pairs for (A, B).

We now define and discuss a few quantities that will prove helpful in the discussion that follows.

- *Cayley residual* s_j^C: this is the residual of the linear system (3.1) at step j of the rational Krylov method.
- *RKS residual* $f_i^{(j)}$: the RKS method computes a Ritz pair $(\theta_i^{(j)}, y_i^{(j)})$ with $y_i^{(j)} = V_{j+1}\tilde{L}_j z_i^{(j)}$ and $\|y_i^{(j)}\| = 1$ for $(A + E_j, B)$, and so the RKS residual satisfies

$$f_i^{(j)} \equiv BV_{j+1}(\tilde{K}_j - \theta_i^{(j)}\tilde{L}_j)z_i^{(j)} = (A + E_j)y_i^{(j)} - \theta_i^{(j)} B y_i^{(j)}.$$

- *True residual* $r_i^{(j)}$: this is the residual defined by $r_i^{(j)} = Ay_i^{(j)} - \theta^{(j)} B y_i^{(j)}$. (The sub- and superscript of the true residual are dropped with those of $f_i^{(j)}$, $y^{(j)}$, $z_i^{(j)}$, and $\theta_i^{(j)}$ when convenient.)

These three residuals may be linked via the relationships

$$r^{(j)} = f^{(j)} - S_j z^{(j)} = f^{(j)} - E_j y^{(j)} \quad \text{for} \quad j = 1, \ldots, m \tag{3.4}$$

that follow from equation (3.2) and the definition (3.3) of E_j. We present numerical evidence demonstrating that, although $\|f^{(j)}\|$ decreases in size for increasing j, $r^{(j)}$ does not decrease when an inexact shift-invert transformation is employed. However, when an inexact Cayley transformation is used instead, both $\|S_j z^{(j)}\|$ and $\|f^{(j)}\|$ decrease and the size of the true residual also decreases.

The continuation of this section is as follows. In subsection 3.1, we present a relationship with inverse iteration that includes a theorem showing the convergence for *inexact* inverse iteration. In subsection 3.2, we fix the various parameters of the RKS method, i.e., the poles, zeros, and continuation vectors. This selection makes a link with the generalized Davidson method [6, 25, 26]. In subsection 3.3, an informal argument is given for the convergence of the inexact Cayley rational Krylov sequence (IC-RKS) method, described in subsection 3.2, using the theoretical result from subsection 3.1. We also illustrate this by a numerical example.

3.1. Inverse iteration. We first exploit a direct relationship with inverse iteration that occurs with a special choice of the continuation vector when a Cayley transformation is used. An example is then presented that compares this choice with a shift-invert transformation. This subsection is concluded with a theorem is showing that the numerical behavior observed is not just a fortuitous event. Although the choice of continuation vector does not exploit the entire space of vectors, as in IC-RKS, the theorem justifies the superior properties of combining an approximate linear solve via a Cayley transformation.

From equation (2.2) and the matrix identity (2.1), it follows that

$$V_{j+1}\tilde{h}_j = V_j t_j + (\mu_j - \nu_j)(A - \mu_j B)^{-1} B V_j t_j.$$

Using (2.5) with $\tilde{l}_j = \tilde{L}_j e_j$, it follows that

$$V_{j+1}\tilde{l}_j = (\mu_j - \nu_j)(A - \mu_j B)^{-1} B V_j t_j, \tag{3.5}$$

and hence $V_{j+1}\tilde{l}_j$ is the linear combination of the columns of V_{j+1} obtained by performing one step of inverse iteration on the vector $V_j t_j$. An inductive argument easily establishes the following property.

LEMMA 3.1. *If $t_1 = 1$ and $t_j = \tilde{l}_{j-1}/\|\tilde{l}_{j-1}\|$ for $j > 1$, then*

$$V_{j+1}\tilde{l}_j = \zeta_j \prod_{i=1}^{j} \left((A - \mu_i B)^{-1} B\right) v_1,$$

where ζ_j is a scalar and v_1 is the starting vector of RKS.

Lemma 3.1 indicates how to compute an approximate eigenvalue. If we denote $\tilde{k}_j \equiv \tilde{K}_j e_j$, equation (2.4) gives the Rayleigh quotient

$$\theta^{(j)} = \frac{(V_{j+1}\tilde{l}_j)^H B^{-1} A (V_{j+1}\tilde{l}_j)}{(V_{j+1}\tilde{l}_j)^H (V_{j+1}\tilde{l}_j)} = \frac{\tilde{l}_j^H \tilde{k}_j}{\tilde{l}_j^H \tilde{l}_j} \tag{3.6}$$

as an estimate of an eigenvalue without need to explicitly apply $B^{-1}A$.

An algorithm for inverse iteration is given in Figure 3.1. The approximate eigenpair on iteration j is $(\theta^{(j)}, y^{(j)} = V_{j+1}\tilde{l}_j/\|\tilde{l}_j\|)$, so we can use the relationships (3.4) with $z^{(j)} = e_j/\|\tilde{l}_j\|$. Recall that we used $\nu_j = \theta^{(j-1)}$ and $V_j t_j = y^{(j-1)}$. The entries $\theta^{(0)}$ and v_1 determine the initial estimates for the eigenpair. We now compare inexact inverse iteration, computed via the RKS method using the shift-invert and Cayley transformations with an example.

Example 3.1. The Olmstead model [28] represents the flow of a layer of viscoelastic fluid heated from below. The equations are

$$\left\{ \begin{array}{rcl} \dfrac{\partial u}{\partial t} & = & (1-C)\dfrac{\partial^2 v}{\partial X^2} + C\dfrac{\partial^2 u}{\partial X^2} + Ru - u^3, \\ B\dfrac{\partial v}{\partial t} & = & u - v, \end{array} \right.$$

with boundary conditions $u(0) = u(1) = 0$ and $v(0) = v(1) = 0$. Here u represents the speed of the fluid and v is related to viscoelastic forces. The equation was discretized with central differences with gridsize $h = 1/(n/2)$. After the discretization, the equation may be written as $\dot{x} = f(x)$ with $x^T = [u_1, v_1, u_2, v_2, \ldots, u_{n/2}, v_{n/2}]$.

Table 3.1

Numerical results for inverse iteration in Example 3.1 *using inexact Cayley and shift-invert transformations. The table shows the norms of true residual* $r^{(j)}$, $S_j z^{(j)}$, *and the RKS residual* $f^{(j)}$. *The norm of* $\tilde{l}_j$ *is also displayed for the Cayley transformation.*

	Cayley				shift-invert		
j	$\Vert r^{(j)}\Vert$	$\Vert S_j z^{(j)}\Vert$	$\Vert f^{(j)}\Vert$	$\Vert\tilde{l}_j\Vert$	$\Vert r^{(j)}\Vert$	$\Vert S_j z^{(j)}\Vert$	$\Vert f^{(j)}\Vert$
1	$1{\cdot}10^{0}$	$8{\cdot}10^{-1}$	$7{\cdot}10^{-1}$	5.2	$4{\cdot}10^{0}$	$5{\cdot}10^{0}$	$4{\cdot}10^{-1}$
2	$1{\cdot}10^{1}$	$1{\cdot}10^{1}$	$5{\cdot}10^{-1}$	0.6	$7{\cdot}10^{-1}$	$7{\cdot}10^{-1}$	$7{\cdot}10^{-2}$
3	$1{\cdot}10^{0}$	$1{\cdot}10^{0}$	$2{\cdot}10^{-1}$	1.6	$4{\cdot}10^{-1}$	$4{\cdot}10^{-1}$	$3{\cdot}10^{-2}$
4	$2{\cdot}10^{-1}$	$2{\cdot}10^{-1}$	$8{\cdot}10^{-2}$	1.2	$5{\cdot}10^{-1}$	$5{\cdot}10^{-1}$	$1{\cdot}10^{-2}$
5	$1{\cdot}10^{-1}$	$9{\cdot}10^{-2}$	$4{\cdot}10^{-2}$	1.0	$5{\cdot}10^{-1}$	$5{\cdot}10^{-1}$	$7{\cdot}10^{-3}$
6	$5{\cdot}10^{-2}$	$5{\cdot}10^{-2}$	$2{\cdot}10^{-2}$	1.0	$5{\cdot}10^{-1}$	$5{\cdot}10^{-1}$	$4{\cdot}10^{-3}$
7	$2{\cdot}10^{-2}$	$2{\cdot}10^{-2}$	$8{\cdot}10^{-3}$	1.0	$5{\cdot}10^{-1}$	$5{\cdot}10^{-1}$	$2{\cdot}10^{-3}$
8	$9{\cdot}10^{-3}$	$9{\cdot}10^{-3}$	$3{\cdot}10^{-3}$	1.0	$5{\cdot}10^{-1}$	$5{\cdot}10^{-1}$	$1{\cdot}10^{-3}$
9	$4{\cdot}10^{-3}$	$4{\cdot}10^{-3}$	$1{\cdot}10^{-3}$	1.0	$5{\cdot}10^{-1}$	$5{\cdot}10^{-1}$	$5{\cdot}10^{-4}$
10	$2{\cdot}10^{-3}$	$1{\cdot}10^{-3}$	$4{\cdot}10^{-4}$	1.0	$5{\cdot}10^{-1}$	$5{\cdot}10^{-1}$	$3{\cdot}10^{-4}$
11	$6{\cdot}10^{-4}$	$6{\cdot}10^{-4}$	$2{\cdot}10^{-4}$	1.0	$5{\cdot}10^{-1}$	$5{\cdot}10^{-1}$	$1{\cdot}10^{-4}$
12	$2{\cdot}10^{-4}$	$2{\cdot}10^{-4}$	$6{\cdot}10^{-5}$	1.0	$5{\cdot}10^{-1}$	$5{\cdot}10^{-1}$	$7{\cdot}10^{-5}$
13	$8{\cdot}10^{-5}$	$8{\cdot}10^{-5}$	$2{\cdot}10^{-5}$	1.0	$5{\cdot}10^{-1}$	$5{\cdot}10^{-1}$	$4{\cdot}10^{-5}$
14	$3{\cdot}10^{-5}$	$3{\cdot}10^{-5}$	$8{\cdot}10^{-6}$	1.0	$5{\cdot}10^{-1}$	$5{\cdot}10^{-1}$	$2{\cdot}10^{-5}$
15	$1{\cdot}10^{-5}$	$1{\cdot}10^{-5}$	$3{\cdot}10^{-6}$	1.0	$5{\cdot}10^{-1}$	$5{\cdot}10^{-1}$	$1{\cdot}10^{-5}$

The size of the Jacobian matrix $A = \partial f/\partial x$ is $n = 100$. We consider the Jacobian for the parameter values $B = 2$, $C = 0.1$, and $R = 4.7$ for the trivial steady state $[u, v] = 0$. Thus, the interest is in the eigenvalue of largest real part.

We ran the algorithm in Figure 3.1. The linear systems were solved by 20 iterations of Gauss–Seidel starting with a zero initial vector. Since this solver is stationary, the relative residual norm is almost constant. The initial guess for the eigenvalue was $\theta^{(0)} = 0$. The initial vector for RKS was $v_1 = [\,1, \cdots, 1\,]^T/\sqrt{n}$. The poles μ_j were set equal to five for all j. The residuals $r^{(j)}$, $f^{(j)}$, and $S_j z^{(j)}$ are shown in Table 3.1. All three sequences decrease when the Cayley transform is used.

We repeated the experiments using the shift-invert transformation. The results are also shown in Table 3.1. Both $\Vert S_j z^{(j)}\Vert$ and $\Vert r^{(j)}\Vert$ stagnate near the same value. Note, however, that $\Vert f^{(j)}\Vert$ tends to zero.

Table 3.1 shows that the true residual decreases when the Cayley transformation is used but stagnates for the shift-invert transformation. The following result indicates what occurs under some mild conditions when performing inexact inverse iteration with either the shift-invert or the Cayley transformation.

Theorem 3.2. *Assume that there is an integer* $k \le m$ *and value* $\gamma > 0$ *such that* $\Vert\tilde{l}_j\Vert \ge \gamma$ *for* $j > k$. *Assume that* $\Vert f^{(j)}\Vert \le \rho \Vert f^{(j-1)}\Vert$ *for* $j \ge k$ *and* $\rho > 0$ *and that* τ *is the relative residual tolerance used for the linear solves* (*see equations* (3.12) *and* (3.13)).

If a Cayley transformation is used, then for $j \ge k + 1$,

$$\Vert r^{(j)}\Vert \le \left(\rho + \frac{\tau}{\gamma}\right)^{j-k} \Vert f^{(k)}\Vert + \left(\frac{\tau}{\gamma}\right)^{j-k} \Vert r^{(k)}\Vert, \tag{3.7}$$

and when a shift-invert transformation is used,

$$\Vert r^{(j)}\Vert \le \rho^{j-k} \Vert f^{(k)}\Vert + \left(\frac{\tau}{\gamma}\right) \Vert B\Vert\ . \tag{3.8}$$

Proof. With $z^{(j)} = e_j/\|\tilde{l}_j\|$, (3.4) becomes $r^{(j)} = f^{(j)} - s_j/\|\tilde{l}_j\|$. With $\|\tilde{l}_j\| \geq \gamma$, it follows that

$$\begin{aligned} \|r^{(j)}\| &\leq \|f^{(j)}\| + \|s_j\|/\|\tilde{l}_j\| \\ &\leq \|f^{(j)}\| + \|s_j\|/\gamma \ . \end{aligned} \tag{3.9}$$

For the Cayley transform, we prove (3.7) by induction on j. We clearly have that

$$\|r^{(k)}\| \leq \|f^{(k)}\| + \|r^{(k)}\| \ ,$$

which satisfies (3.7) for $j = k$. Suppose that (3.7) holds for some integer $j - 1 \geq k$. From the hypothesis of the theorem, we have that

$$\|f^{(j)}\| \leq \rho\|f^{(j-1)}\| \leq \cdots \leq \rho^{j-k}\|f^{(k)}\|.$$

Combining this with equations (3.9) and (3.12) results in

$$\|r^{(j)}\| \leq \|f^{(j)}\| + \|s_j\|/\gamma \leq \rho^{j-k}\|f^{(k)}\| + (\tau/\gamma)\|r^{(j-1)}\|.$$

Using our inductive hypothesis on $\|r^{(j-1)}\|$ gives

$$\begin{aligned} \|r^{(j)}\| &\leq (\rho^{j-k} + \tau/\gamma(\rho + \tau/\gamma)^{j-k-1})\|f^{(k)}\| + (\tau/\gamma)^{j-k}\|r^{(k)}\| \\ &\leq (\rho + \tau/\gamma)^{j-k}\|f^{(k)}\| + (\tau/\gamma)^{j-k}\|r^{(k)}\| \ , \end{aligned}$$

and (3.7) follows. For shift-invert, (3.8) follows from (3.9) and (3.13), which completes the proof. □

The theorem shows that if $\rho + \tau/\gamma < 1$, inexact inverse iteration computed via the Cayley transformation will produce a Ritz pair with a small direct residual. Since $\rho + \tau/\gamma \geq \rho$, inexact inverse iteration can do no better than exact inverse iteration. Although the term $\|f^{(j)}\|$ will decrease when using the shift-invert transformation, the size of the direct residual $\|r^{(j)}\|$ may stagnate. This occurs because the contribution from solving the linear systems inexactly (s_j^{SI}) to the true residual is constant. When a direct method is used for the linear system of equations, τ is a multiple of machine precision. Hence, whether a shift-invert or Cayley transformation is used, the true residual $\|r^{(j)}\|$ decreases at a rate proportional to ρ.

For the exact Cayley transformation, we have

$$V_{j+1}\tilde{h}_j = (A - \mu_j B)^{-1} f^{(j-1)}$$

and $\tilde{l}_j = \tilde{h}_j - \tilde{t}_j$ and $\|t_j\| = 1$. Hence, we have

$$1 - \delta_j \leq \|\tilde{l}_j\| \leq 1 + \delta_j \quad \text{where} \quad \delta_j = \|(A - \mu_j B)^{-1} f^{(j-1)}\| \ .$$

Thus, $\gamma = \max(0, \min_{j\geq k} 1 - \delta_j)$. If $\|(A - \mu_j B)^{-1} f^{(j-1)}\|$ converges to zero, then $1 \pm \delta_j$ tends to one for increasing j. Computation reveals that, quite often, $\|\tilde{l}_j\| \approx 1$ after a very small number of steps. This also holds for inexact inverse iteration, because it can be seen as exact inverse iteration applied to $(A + E_j, B)$, as Table 3.1 demonstrates. Hence, for large enough k, $\gamma \approx 1$ and the convergence rate of inverse iteration using the Cayley transform is approximately $\rho + \tau$. As the method progresses, ρ is easily estimated and thus the largest relative residual tolerance that may be used is also easily estimated.

3.2. Choosing a pole, zero, and continuation vector. A robust and efficient strategy for selecting the poles during the RKS method is a subject of research. The present situation is further complicated because we employ approximate methods for the linear solves. A fixed pole is used for the numerical experiments because our interest is in the Ritz pairs produced by $\tilde{K}_k$ and $\tilde{L}_k$.

The choice of the zero of the Cayley transformation is crucial for computing a Ritz pair with a small direct residual. This was demonstrated by the numerical examples in [23]. First, we formally analyze the choice of the zero and continuation vector and then give an example.

Suppose that $(\theta^{(j-1)}, y^{(j-1)})$ is an (inexact) Ritz pair computed during the j-1st step of an (inexact) RKS method. We select the zero $\nu_j = \theta^{(j-1)}$ and continuation vector $V_j t_j = y^{(j-1)}$ (or equivalently, $t_j = \tilde{L}_{j-1} z^{(j-1)}$) for some Ritz pair of interest. For a Cayley transformation, this leads to

$$(A - \mu_j B) w = r^{(j-1)} \tag{3.10}$$

while a shift-invert transformation gives

$$(A - \mu_j B) w = B y^{(j-1)} \tag{3.11}$$

as the linear systems to be solved. Although both transformations use the same continuation vector, the Cayley transformation also uses the Ritz value for its zero. The only difference in the two linear systems (3.10) and (3.11) is in their right-hand sides. When a preconditioner is used to solve the linear system (3.10), we have a generalization of Davidson's method [6, 26] for computing eigenvalues of a matrix pencil.

Denote the computed solutions to (3.10) and (3.11) by x_j^C and x_j^{SI}, respectively. If an iterative method with relative residual tolerance τ is used for the two linear systems, then the residuals of the linear systems satisfy

$$\|s_j^C\| \equiv \|r^{(j-1)} - (A - \mu_j B) x_j^C\| \le \tau \|r^{(j-1)}\| \tag{3.12}$$

$$\|s_j^{SI}\| \equiv \|B y^{(j-1)} - (A - \mu_j B) x_j^{SI}\| \le \tau \|B y^{(j-1)}\| \le \tau \|B\| \tag{3.13}$$

for the Cayley and shift-invert transformation, respectively. (We drop the superscripts that denote Cayley or shift-invert transformations when the context is clear.)

In view of the two bounds (3.12) and (3.13) on the computed solutions, a Cayley transformation is preferred over a shift-invert transformation. It appears that use of a Cayley transformation leads to better results with inexact linear solvers when the zero and continuation vector are chosen, as in (3.10). Our experimental results also support this conclusion. The algorithm in Figure 3.2 lists an inexact Cayley RKS method (IC-RKS). We now illustrate a few properties of this algorithm with an example that demonstrates: (1) the inexact rational Krylov method is not a Galerkin projection method; (2) the method can only compute one eigenvalue at a time, just as in Davidson methods.

Example 3.2. Consider the matrices $A = \operatorname{diag}(1, \cdots, 5)$ and $B = I$. The pencil (A, I) has eigenpairs (j, e_j), $j = 1, \ldots, 5$. The goal is to compute the smallest eigenvalue 1 and corresponding eigenvector e_1 with the IC-RKS method using a fixed pole $\mu_j = 0.7$. The starting vector is set equal to $v_1 = [\,1, \cdots, 1\,]^T/\sqrt{5}$. The Cayley system

$$(A - \mu_j I) x_j = r^{(j-1)}$$

```
• Choose a starting vector $v_1$ with $\|v_1\| = 1$ and set $V_1 \leftarrow [\ v_1\ ]$.
• For $j = 1, 2, \ldots, m$
    1. Select a pole $\mu_j$. If $j > 1$ set the zero $\nu_j = \theta_i^{(j-1)}$ and $t_j = \tilde{L}_{j-1} z^{(j-1)}$.
       Otherwise, set $\nu_1 = 0$ and $t_1 = 1$.
    2. Compute the continuation vector $r^{(j-1)} = AV_j t_j - \nu_j BV_j t_j$ (the true
       residual).
    3. Form $w \leftarrow (A - \mu_j B)^{-1} r^{(j-1)}$.
    4. See steps 4–7 of the RKS method listed in Figure 2.1.
    5. Solve the eigenvalue problem $\tilde{L}_j^{\dagger} \tilde{K}_j z = \theta z$ (see subsection 2.2.1).
    6. Check whether the approximate eigenpairs satisfy the convergence
       criterion.
```

FIG. 3.2. *Computing eigenvalues of the pencil (A, B) by the inexact Cayley rational Krylov sequence (IC-RKS) method.*

is solved as $x_j = M^{-1} r^{(j-1)}$, where

$$M^{-1} = \begin{bmatrix} (1-\mu_j)^{-1} & 10^{-2} & & & \\ 10^{-2} & \ddots & \ddots & & \\ & \ddots & \ddots & 10^{-2} & \\ & & 10^{-2} & (5-\mu_j)^{-1} \end{bmatrix}.$$

Note that M simulates a stationary iterative solver with residual tolerance $\tau = \|I - (A - \mu_j I) M^{-1}\| \approx 5 \cdot 10^{-2}$. We performed $m = n = 5$ iterations, so $\mathcal{R}(V_5 L_5) \equiv \boldsymbol{C}^n$, which implies that $f_i^{(5)} = 0$ for $i = 1, \ldots, 5$. Thus, the computed eigenpairs are exact eigenpairs of $A + E_5$. We found that

$$A + E_5 = \begin{bmatrix} 1.0000 & 0.0120 & -0.0697 & 0.3708 & -0.4728 \\ -0.0000 & 1.9987 & -0.5981 & 4.4591 & -5.6013 \\ -0.0001 & 0.1003 & 0.4666 & 17.1897 & -21.1757 \\ -0.0002 & 0.0340 & -4.4220 & 36.8172 & -40.7251 \\ -0.0002 & 0.0151 & -3.7375 & 26.8228 & -27.8127 \end{bmatrix}$$

and has eigenpairs

$$\begin{array}{rcccc} i = & 1 & 2 & 3 & 4 \ \& \ 5 \\ \theta_i^{(5)} = & 1.0000 & 2.0123 & 2.5340 & 3.4618 \pm 6.3095i \end{array}$$

$$y_i^{(5)} = \begin{bmatrix} 1.0000 & 0.0165 & -0.0177 & 0.0045 \pm 0.0068i \\ 0.0000 & 0.9812 & -0.1340 & 0.0249 \pm 0.0951i \\ 0.0000 & -0.1801 & 0.9229 & 0.0149 \pm 0.3758i \\ 0.0000 & -0.0602 & 0.3188 & -0.0826 \pm 0.7214i \\ 0.0000 & -0.0310 & 0.1681 & -0.1810 \pm 0.5374i \end{bmatrix}.$$

Since $f_i^{(5)} = 0$, the true residual has the form $r_i^{(5)} = -E_5 y_i^{(5)}$. For example, $\|r_1^{(5)}\| = 6 \cdot 10^{-5}$, but $1 \cdot 10^{-1} < \|r_i^{(5)}\| < 1 \cdot 10^{1}$ for $i > 1$.

This example shows that E_5 is nearly rank deficient and that the desired eigenvector of (A, I) is nearly its null vector. Therefore, the desired eigenvalue, in this case, $\lambda_1 = 1$, can be computed with a small true residual. It should be noted that the perturbation E_5 is small in the direction of only one eigenspace, hence IC-RKS is not able to compute several eigenvalues simultaneously. This is not the situation when the linear systems are solved more accurately with, for instance, a direct method.

In this example, IC-RKS computes the exact eigenpairs of $A + E_5$ after $m = 5$ steps. In general, however, $r_i^{(5)} \neq 0$, because the inexact Ritz pair is not computed from a Galerkin projection with A. We also remark that $\theta_4^{(5)}$ and $\theta_5^{(5)}$ are nonreal, and this would not be the case with a Galerkin projection because A is a real symmetric matrix. This is in contrast with other iterative eigenvalue solvers, such as Arnoldi and Jacobi–Davidson methods, where Galerkin projections with A are employed.

3.3. Inexact rational Krylov. We now informally discuss the algorithm listed in Figure 3.2, including a comparison with inexact inverse iteration of the previous section.

From (3.5), with the Ritz vector $y^{(i-1)}$ computed as in subsection 2.2.1, it follows that

$$V_{i+1}\tilde{l}_i = \zeta_i (A - \mu_i B)^{-1} B y^{(i-1)} \quad \text{with} \quad y^{(i-1)} = V_i \tilde{L}_{i-1} z^{(i-1)} \quad \text{for} \quad i = 1, \ldots, j \ .$$

Numerical experiments reveal that the jth component of $z^{(j)}$ is large relative to the initial $j-1$ components (see Table 3.2). This is because the best approximation of the desired eigenvector among the columns of $V_{j+1}\tilde{L}_j$ is given by $V_{j+1}\tilde{l}_j$—the improvement of the previous Ritz vector by one step of inverse iteration. Thus, using the continuation vector $V_i \tilde{L}_{i-1} z^{(i-1)}$ should give better results because information in the subspace $\mathcal{R}(V_{i+1}\tilde{L}_i)$ is used. Inexact inverse iteration only uses information in the space spanned by the last column of $V_{i+1}\tilde{L}_i$.

The inexact Ritz pairs $(\theta^{(i)}, y^{(i)})$ lead to true residuals $r^{(i)}$ if the Cayley transform is used. The Cayley residual on iteration i satisfies $\|s_i\| \leq \tau \|r^{(i-1)}\|$. The true residual on the jth iteration is decomposed as $r^{(j)} = f^{(j)} - S_j z^{(j)}$, where

$$\|S_j z^{(j)}\| \leq \sum_{i=1}^{j} \|s_i\| \, |e_i^T z^{(j)}| \leq \tau \sum_{i=1}^{j} \|r^{(i-1)}\| \, |e_i^T z^{(j)}|$$

gives an upper bound to $\|S_j z^{(j)}\|$. In the right-hand side, $\|s_i\|$ is independent of j and can be quite large for small i. However, because $|e_i^T z^{(j)}|$ typically forms a decreasing sequence for increasing j, we have a decreasing sequence $\|S_j z^{(j)}\|$.

Example 3.3. We now discuss an example for which $e_i^T z^{(j)}$ and $S_j z^{(j)}$ tend to zero in the IC-RKS method. The matrix arises from the same problem as in Example 3.1, but now $n = 200$. We ran algorithm IC-RKS from Figure 3.2 with fixed $\mu_j = 5$, starting with vector $v_1 = [1, \cdots, 1]^T/\sqrt{n}$. The linear systems were solved by GMRES preconditioned by ILU. The number of iterations of GMRES was determined by the relative error tolerance, which was selected as $\tau = 10^{-4}$. Table 3.2 shows the residual norm and the norm of the error term $S_j z^{(j)}$. Both $\|S_j z^{(j)}\|$ and $\|r^{(j)}\|$ tend to zero. For large j, $\|S_j z^{(j)}\| \approx \|r^{(j)}\|$. This is the case because $f^{(j)}$ converges more rapidly to zero than $S_j z^{(j)}$. Table 3.2 also illustrates the fact that $e_i^T z^{(j)}$ decreases for a fixed i and increasing j.

4. A numerical example. This example illustrates the use of inexact rational Krylov methods for the solution of a generalized eigenvalue problem. We also compare inexact inverse iteration with the Cayley transform and IC-RKS.

The simulation of flow of a viscous fluid with a free surface on a tilted plane leads, with a finite element approach, to an eigenvalue problem $Ax = Bx\lambda$ with

Table 3.2

Numerical results for the Olmstead model of Example 3.3. The table shows the order of accuracy for the residual norm of the rightmost Ritz pair, the norm of $S_j z^{(j)}$, and the first four components of $z^{(j)}$.

j	$\|r^{(j)}\|$	$\|S_j z^{(j)}\|$	$\|e_1^T z^{(j)}\|$	$\|e_2^T z^{(j)}\|$	$\|e_3^T z^{(j)}\|$	$\|e_4^T z^{(j)}\|$
1	$6{\cdot}10^{-1}$	$1{\cdot}10^{-1}$	$2{\cdot}10^{-1}$			
2	$2{\cdot}10^{0}$	$3{\cdot}10^{-1}$	$5{\cdot}10^{-1}$	$3{\cdot}10^{0}$		
3	$2{\cdot}10^{-2}$	$1{\cdot}10^{-2}$	$5{\cdot}10^{-3}$	$6{\cdot}10^{-1}$	1	
4	$2{\cdot}10^{-2}$	$1{\cdot}10^{-2}$	$3{\cdot}10^{-3}$	$6{\cdot}10^{-1}$	1	2
5	$8{\cdot}10^{-4}$	$7{\cdot}10^{-4}$	$1{\cdot}10^{-4}$	$4{\cdot}10^{-2}$	$6{\cdot}10^{-2}$	$3{\cdot}10^{-1}$
6	$3{\cdot}10^{-4}$	$3{\cdot}10^{-4}$	$2{\cdot}10^{-5}$	$2{\cdot}10^{-2}$	$3{\cdot}10^{-2}$	$3{\cdot}10^{-1}$
7	$2{\cdot}10^{-5}$	$2{\cdot}10^{-5}$	$2{\cdot}10^{-6}$	$1{\cdot}10^{-3}$	$2{\cdot}10^{-3}$	$2{\cdot}10^{-2}$
8	$5{\cdot}10^{-7}$	$5{\cdot}10^{-7}$	$7{\cdot}10^{-8}$	$2{\cdot}10^{-5}$	$5{\cdot}10^{-5}$	$1{\cdot}10^{-3}$
9	$3{\cdot}10^{-8}$	$3{\cdot}10^{-8}$	$4{\cdot}10^{-9}$	$2{\cdot}10^{-6}$	$2{\cdot}10^{-6}$	$6{\cdot}10^{-5}$

Table 4.1

Numerical results for the tilted plane problem from section 4. The methods used are inexact rational Krylov (IC-RKS) and inverse iteration with the Cayley transform. On iteration j, $\theta^{(j)}$ is the inexact Ritz value, s_j the Cayley residual, and $g^{(j)} = (\tilde{K}_j - \theta^{(j)}\tilde{L}_j)z^{(j)}$.

	IC-RKS (Figure 3.2)				Inverse Iteration (Figure 3.1)			
j	$(\theta_1^{(j)})^{-1}$	$\|s_j\|$	$\|r_1^{(j)}\|$	$\|g_1^{(j)}\|$	$(\theta_1^{(j)})^{-1}$	$\|s_j\|$	$\|r^{(j)}\|$	$\|g^{(j)}\|$
1	−9.40554	$5{\cdot}10^{-9}$	$1{\cdot}10^{-6}$	$3{\cdot}10^{-3}$	−9.40554	$5{\cdot}10^{-9}$	$1{\cdot}10^{-6}$	$2{\cdot}10^{-3}$
2	−9.48481	$1{\cdot}10^{-10}$	$3{\cdot}10^{-8}$	$5{\cdot}10^{-6}$	−9.49928	$1{\cdot}10^{-10}$	$2{\cdot}10^{-7}$	$4{\cdot}10^{-4}$
3	−9.48825	$1{\cdot}10^{-12}$	$1{\cdot}10^{-9}$	$8{\cdot}10^{-8}$	−9.48705	$2{\cdot}10^{-11}$	$4{\cdot}10^{-9}$	$6{\cdot}10^{-6}$
4	−9.48831	$6{\cdot}10^{-13}$	$2{\cdot}10^{-11}$	$1{\cdot}10^{-9}$	−9.48845	$3{\cdot}10^{-12}$	$4{\cdot}10^{-9}$	$8{\cdot}10^{-7}$
5	−9.48832	$2{\cdot}10^{-15}$	$6{\cdot}10^{-13}$	$6{\cdot}10^{-11}$	−9.48831	$3{\cdot}10^{-13}$	$5{\cdot}10^{-10}$	$8{\cdot}10^{-8}$
6	−9.48832	$5{\cdot}10^{-17}$	$2{\cdot}10^{-14}$	$1{\cdot}10^{-12}$	−9.48832	$5{\cdot}10^{-14}$	$5{\cdot}10^{-11}$	$8{\cdot}10^{-9}$
7					−9.48832	$4{\cdot}10^{-14}$	$5{\cdot}10^{-12}$	$6{\cdot}10^{-10}$
8					−9.48832	$5{\cdot}10^{-15}$	$4{\cdot}10^{-13}$	$3{\cdot}10^{-11}$
9					−9.48832	$3{\cdot}10^{-17}$	$3{\cdot}10^{-14}$	$6{\cdot}10^{-12}$

$A, B \in \boldsymbol{R}^{536\times 536}$ nonsymmetric and B a singular matrix. The computation of the eigenvalue nearest -10 is of interest. Since our theory is valid only for nonsingular B, we interchange the role of A and B by computing the eigenvalue $\gamma = \lambda^{-1}$ of $Bx = Ax\gamma$ nearest $\mu = -10^{-1}$.

The fact that B is singular implies that $\gamma = 0$ is an eigenvalue. It has been shown that the presence of this eigenvalue can disturb the calculation of a nonzero eigenvalue when either the shift-invert Arnoldi method [29, 24] or the rational Krylov method [36] is used. One way to reduce the impact of $\gamma = 0$ is to start the IC-RKS method with an initial vector v_1 that is poor in the eigenspace corresponding to $\gamma = 0$ [27]. This can be achieved by selecting $v_1 = (B - \mu A)^{-1}Bv$ with v arbitrary.

The eigenvalue γ nearest -0.1 was computed using IC-RKS (Figure 3.2) with fixed pole $\mu_j = -0.1$. The linear systems were solved by GMRES preconditioned with `ILUT(lfil=40,tol=1.e-3)` [33] with $\tau = 10^{-4}$. The initial vector v_1 was computed from the system $(B - \mu A)v_1 = Bv$ with $v = [\ 1, \cdots, 1\ ]^T$ using the GMRES-ILUT solver. The algorithm was stopped when $\|r_1^{(j)}\| \leq \text{TOL} = 10^{-13}$.

The numerical results are shown in Table 4.1 for inexact rational Krylov (IC-RKS) and inexact inverse iteration using the Cayley transform. First, note that $\|f^{(j)}\| \leq \|A\|\|g^{(j)}\|$, so $\|g^{(j)}\|$ does not measure the RKS residual (see also (2.8)). Also note that for both IC-RKS and inverse iteration, the sequences $\|r^{(j)}\|$, $\|s_j\|$, and

$\|g^{(j)}\|$ decrease. Both methods converge to $\lambda = \gamma^{-1} \approx -9.486$. Finally, note that IC-RKS is faster than inverse iteration.

5. A relation between inexact shift-invert and Cayley transforms. In the previous section, we showed that the inexact rational Krylov method can be used for the computation of eigenvalues of a matrix pencil. The example shows a substantial difference in convergence behavior between the shift-invert and Cayley transformations. In this section, we show that an appropriate shift-invert transformation may also be employed.

During each step of IC-RKS, the following relationship

$$(A - \mu_j B)x_j = (A - \nu_j B)y^{(j)} - s_j \tag{5.1}$$

results, where s_j is the residual of the linear system that is approximately solved. Rearranging (5.1) and adding $\mu_j B y^{(j)}$ to both sides gives the equivalent shift-invert system

$$(A - \mu_j B)(x_j - y^{(j)}) = (\mu_j - \nu_j)By^{(j)} - s_j.$$

Hence, if the zero vector is used as the initial guess for the iterative method for linear systems approximately solved via the Cayley transform, $-y^{(j)}$ should be used for the shift-invert transformation formulation.

Assume that τ is a constant and that IC-RKS converges to some eigenpair. From (3.13), it follows that when shift-invert is used, convergence to the same eigenpair is attained for decreasing τ (as j increases). In the context of inexact inverse iteration, Lai, Lin, and Lin [20] also observe that the approximate linear system solver requires an increasingly tighter tolerance on the residual (of the linear system) as the number of inverse iterations increases. In contrast, a Cayley transformation allows us to use a fixed tolerance on the linear system residual.

6. A connection with the Jacobi–Davidson method. We now show a connection between the Jacobi–Davidson [14, 39, 38] and RKS [32] methods.

Consider the linear system

$$(A - \mu_j B)w = (A - \tilde{\nu}_j B)y_j, \tag{6.1}$$

where $y_j = V_j \tilde{L}_{j-1} z^{(j-1)}$ is a Ritz vector of interest. This amounts to selecting the jth continuation vector $t_j = \tilde{L}_{j-1} z^{(j-1)}$ as in algorithm IC-RKS in Figure 3.2, with associated Ritz value

$$\tilde{\nu}_j = \frac{y_j^H A y_j}{y_j^H B y_j}.$$

The right-hand side in (6.1) is then the residual of the eigenpair $(\tilde{\nu}_j, y_j)$ and is orthogonal to y_j. Since we are interested in expanding our search space (the span of the columns of V_j), multiply both sides of equation (6.1) by the projector $I - By_j y_j^H/(y_j^H B y_j)$. The fact that $(A - \tilde{\nu}_j B)y_j \perp y_j$ results in

$$\left(I - \frac{By_j y_j^H}{y_j^H B y_j} \right) (A - \mu_j B)w = (A - \tilde{\nu}_j B)y_j.$$

Since $y_j \in \mathcal{R}(V_j)$, the component of w in the direction of y_j does not play a role when w is added to the subspace $\mathcal{R}(V_j)$. Thus, we are interested in finding only the component of w orthogonal to y_j, and so the linear system

$$\left(I - \frac{By_jy_j^H}{y_j^HBy_j}\right)(A - \mu_jB)\left(I - \frac{y_jy_j^H}{y_j^Hy_j}\right)w = (A - \tilde{\nu}_jB)y_j \tag{6.2}$$

is solved instead. The Jacobi–Davidson method calls equation (6.2) the *correction equation*. Suppose that x_j is a computed solution of equation (6.2) with residual s_j, given by

$$s_j = (A - \tilde{\nu}_jB)y_j - \left(I - \frac{By_jy_j^H}{y_j^HBy_j}\right)(A - \mu_jB)\left(I - \frac{y_jy_j^H}{y_j^Hy_j}\right)x_j, \tag{6.3}$$

where s_j is orthogonal to y_j. Rewrite (6.3) with $d \equiv (I - y_jy_j^H/(y_j^Hy_j))x_j \perp y_j$ as

$$(A - \mu_jB)d = \varepsilon_jBy_j + (A - \tilde{\nu}_jB)y_j - s_j = (A - (\tilde{\nu}_j - \varepsilon_j)B)y_j - s_j. \tag{6.4}$$

The orthogonality of y_j with d and s_j leads to

$$\varepsilon_j = \frac{y_j^H(A - \mu_jB)d}{y_j^HBy_j}. \tag{6.5}$$

Choosing the zero $\nu_j \equiv \tilde{\nu}_j - \varepsilon_j$ gives a relationship between the Jacobi–Davidson and RKS methods when Cayley transformations are used. When ε_j is computed, the solution of the Jacobi–Davidson correction equation $x_j = w$ can be inserted in the RKS method. Note that, although the Ritz vector y_j is orthogonal to the right-hand side of the Jacobi–Davidson correction equation (6.2), y_j is not orthogonal to the right-hand side of (6.4).

An advantage of the inexact rational Krylov method is that the matrices $\tilde{L}_j$ and $\tilde{K}_j$ do not require the explicit application of A and/or B as needed, as in the Jacobi–Davidson method. An efficient implementation of the Jacobi–Davidson method requires dot products (the first $j-1$ elements in the last row of $V_j^HAV_j$ and $V_j^HBV_j$).

We caution the reader to conclude that the Jacobi–Davidson method is an expensive variant of IC-RKS because it fits an IC-RKS framework. A detailed numerical comparison of the two methods requires examining the respective rates of convergence and ability to obtain relative residual reductions during the linear solves. This is the subject of future work.

7. Conclusions.

This paper studied the use of approximate linear solves within Ruhe's rational Krylov sequence method. The analysis of the convergence of inexact inverse iteration showed the importance of using the Cayley transformation instead of the usual shift-invert transformation, when the linear systems are solved with a given *relative* residual tolerance.

A theoretical link between the inexact rational Krylov method that uses generalized Cayley transformations and the Jacobi–Davidson methods was drawn resulting in a connection between the correction equation and Cayley transformation.

We called the eigenpairs computed by IC-RKS inexact Ritz pairs because they are Ritz pairs for a perturbed RKS method. The classical properties of Galerkin projection are lost due to this inexactness. Since IC-RKS solves a perturbed problem, the application of techniques developed for the RKS method (using approximate linear solves) may be employed. These techniques include the use of complex poles and zeros for real A and B [31], harmonic Ritz pairs, deflation and purging [32, 36], and the implicit application of a rational filter [36].

Acknowledgments. The authors thank Dirk Roose for the financial support that allowed the first author to visit the second author. This visit initiated the collaboration that led to this paper. The authors also thank Gorik De Samblanx, Gerard Sleijpen, and the referees for helpful comments and suggestions that improved the quality of the paper. In particular, one of the referees provided numerous constructive criticisms that improved the quality of the presentation.

REFERENCES

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen, *LAPACK Users' Guide*, 2nd ed., SIAM, Philadelphia, PA, 1995.

[2] W. E. Arnoldi, *The principle of minimized iterations in the solution of the matrix eigenvalue problem*, Quart. Appl. Math., 9 (1951), pp. 17–29.

[3] R. Barrett, M. Berry, T. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, Philadelphia, PA, 1993.

[4] J. H. Bramble, A. V. Knyazev, and J. E. Pasciak, *A subspace preconditioning algorithm for eigenvector/eigenvalue computation*, Adv. Comput. Math., 6 (1996), pp. 159–189.

[5] J. W. Daniel, W. B. Gragg, L. Kaufman, and G. W. Stewart, *Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization*, Math. Comp., 30 (1976), pp. 772–795.

[6] E. R. Davidson, *The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real symmetric matrices*, J. Comput. Phys., 17 (1975), pp. 87–94.

[7] T. A. Davis and I. S. Duff, *An unsymmetric-pattern multifrontal method for sparse LU factorization*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 140–158.

[8] T. A. Davis and I. S. Duff, *A combined unifrontal/multifrontal method for unsymmetric sparse matrices*, ACM Trans. Math. Software, to appear.

[9] J. W. Demmel, S. C. Eisenstat, J. R. Gilbert, X. S. Li, and J. W. H. Liu, *A supernodal approach to sparse partial pivoting*, SIAM J. Matrix Anal. Appl., to appear.

[10] I. S. Duff, *ME*28 *: A sparse unsymmetric linear equation solver for complex equations*, ACM Trans. Math. Software, 7 (1981), pp. 505–511.

[11] I. S. Duff and J. K. Reid, *The design of MA*48*, a code for direct solution of sparse unsymmetric linear systems of equations*, ACM Trans. Math. Software, 22 (1996), pp. 187–226.

[12] I. S. Duff and J. A. Scott, *The design of a new frontal code for solving sparse unsymmetric systems*, ACM Trans. Math. Software, 22 (1996), pp. 30–45.

[13] T. Ericsson and A. Ruhe, *The spectral transformation Lanczos method for the numerical solution of large sparse generalized symmetric eigenvalue problems*, Math. Comp., 35 (1980), pp. 1251–1268.

[14] D. R. Fokkema, G. L. G. Sleijpen, and H. A. van der Vorst, *Jacobi–Davidson style QR and QZ algorithms for the partial reduction of matrix pencils*, SIAM J. Sci. Comput., 20 (1999), pp. 94–125.

[15] R. W. Freund, G. H. Golub, and N. M. Nachtigal, *Iterative solution of linear systems*, in Acta Numerica, A. Iserles, ed., Cambridge University Press, Cambridge, UK, 1992, pp. 57–100.

[16] R. W. Freund and N. M. Nachtigal, *QMRPACK: A package of QMR algorithms*, ACM Trans. Math. Software, 22 (1996), pp. 46–77.

[17] G. Golub and C. Van Loan, *Matrix Computations*, 3rd ed., The Johns Hopkins University Press, Baltimore, MD, 1996.

[18] A. V. Knyazev, *Convergence rate estimates for iterative methods for mesh symmetric eigenvalue problem*, Soviet J. Numer. Anal. Math. Modelling, 2 (1987), pp. 371–396.

[19] A. V. Knyazev and A. L. Skorokhodov, *Preconditioned gradient-type iterative methods in a subspace for partial generalized symmetric eigenvalue problems*, SIAM J. Numer. Anal., 31 (1994), pp. 1226–1239.

[20] Y.-L. Lai, K.-Y. Lin, and W.-W. Lin, *An inexact inverse iteration for large sparse eigenvalue problems*, Numer. Linear Algebra Appl., 4 (1997), pp. 425–437.

[21] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK Users' Guide: Solution of Large Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*, SIAM, Philadelphia, PA, 1998.

[22] K. Meerbergen and D. Roose, *Matrix transformations for computing rightmost eigenvalues of real nonsymmetric matrices*, IMA J. Numer. Anal., 16 (1996), pp. 297–346.

[23] K. Meerbergen and D. Roose, *The restarted Arnoldi method applied to iterative linear system solvers for the computation of rightmost eigenvalues*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 1–20.

[24] K. Meerbergen and A. Spence, *Implicitly restarted Arnoldi with purification for the shift–invert transformation*, Math. Comp., 218 (1997), pp. 667–689.

[25] R. B. Morgan, *Davidson's method and preconditioning for generalized eigenvalue problems*, J. Comput. Phys., 89 (1990), pp. 241–245.

[26] R. B. Morgan, *Generalizations of Davidson's method for computing eigenvalues of large nonsymmetric matrices*, J. Comput. Phys., 101 (1992), pp. 287–291.

[27] B. Nour-Omid, B. N. Parlett, T. Ericsson, and P. S. Jensen, *How to implement the spectral transformation*, Math. Comp., 48 (1987), pp. 663–673.

[28] W. E. Olmstead, S. H. Davis, S. Rosenblat, and W. L. Kath, *Bifurcation with memory*, SIAM J. Appl. Math., 46 (1986), pp. 171–188.

[29] B. Philippe and M. Sadkane, *Improving the spectral transformation block Arnoldi method*, in Proc. Second IMACS Internat. Symposium on Iterative Methods in Linear Algebra, Blagoevgrad, Bulgaria, 1995, IMACS Series in Computational and Applied Mathematics, vol. 3, P. S. Vassilevski and S. D. Margenov, eds., pp. 57–63.

[30] A. Ruhe, *Rational Krylov sequence methods for eigenvalue computation*, Linear Algebra Appl., 58 (1984), pp. 391–405.

[31] A. Ruhe, *The rational Krylov algorithm for nonsymmetric eigenvalue problems,* III*: Complex shifts for real matrices*, BIT, 34 (1994), pp. 165–176.

[32] A. Ruhe, *Rational Krylov: A practical algorithm for large sparse nonsymmetric matrix pencils*, SIAM J. Sci. Comput., 19 (1998), pp. 1535–1551.

[33] Y. Saad, *SPARSKIT: A Basic Tool Kit for Sparse Matrix Computations*, Technical Report 90-20, Research Institute for Advanced Computer Science, NASA Ames Research Center, Moffet Field, CA, 1990.

[34] Y. Saad, *Iterative Methods for Sparse Linear Systems*. PWS, Boston, MA, 1996.

[35] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Stat. Comput., 7 (1986), pp. 856–869.

[36] G. De Samblanx, K. Meerbergen, and A. Bultheel, *The implicit application of a rational filter in the rks method*, BIT, 37 (1997), pp. 925–947.

[37] G. L. G. Sleijpen, D. R. Fokkema, and H. A van der Vorst, *BiCGstab(ℓ) and other hybrid Bi-CG methods*, Numer. Algorithms, 7 (1994), pp. 75–109.

[38] G. L. G. Sleijpen and H. A. van der Vorst, *A Jacobi–Davidson iteration method for linear eigenvalue problems*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 401–425.

[39] G. L. G. Sleijpen, J. G. L. Booten, D. R. Fokkema, and H. A. van der Vorst, *Jacobi-Davidson type methods for generalized eigenproblems and polynomial eigenproblems*, BIT, 36 (1996), pp. 595–633.

[40] D. C. Sorensen, *Implicit application of polynomial filters in a k-step Arnoldi method*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 357–385.

[41] D.C. Sorensen, *Truncated QZ Methods for Large Scale Generalized Eigenvalue Problems*, Technical Report TR98-01, Rice University, Houston, TX, 1998.

[42] Daniel B. Szyld, *Criteria for combining inverse and Rayleigh quotient iteration*, SIAM J. Numer. Anal., 25 (1988), pp. 1369–1375.

SIAM J. Matrix Anal. Appl.
Vol. 20, No. 1, pp. 149–158

THREE ABSOLUTE PERTURBATION BOUNDS FOR MATRIX EIGENVALUES IMPLY RELATIVE BOUNDS*

STANLEY C. EISENSTAT† AND ILSE C. F. IPSEN‡

Abstract. We show that three well-known perturbation bounds for matrix eigenvalues imply relative bounds: the Bauer–Fike and Hoffman–Wielandt theorems for diagonalizable matrices, and Weyl's theorem for Hermitian matrices. As a consequence, relative perturbation bounds are not necessarily stronger than absolute bounds, and the conditioning of an eigenvalue in the relative sense is the same as in the absolute sense.

We also show that eigenvalues of normal matrices are no more sensitive to perturbations than eigenvalues of Hermitian positive-definite matrices. The relative error bounds are invariant under congruence transformations, such as grading and scaling.

Key words. eigenvalues, relative perturbation bounds, conditioning

AMS subject classifications. 15A18, 15A42, 65F15

PII. S0895479897323282

1. Introduction.

Let A be a complex square matrix and let $A+E$ be a perturbation of A. We want to estimate the error in an eigenvalue $\hat{\lambda}$ of $A+E$ when it is viewed as an approximation to an eigenvalue of A.

Traditional perturbation results bound the absolute error in an eigenvalue. The Bauer–Fike theorem [2, Theorem IIIa], for instance, bounds the absolute distance between $\hat{\lambda}$ and a closest eigenvalue λ of a diagonalizable matrix A by

$$|\lambda - \hat{\lambda}| \leq \kappa(X)\, \|E\|,$$

where $\kappa(X) = \|X\|\,\|X^{-1}\|$ is the condition number of an eigenvector matrix X of A.

The simplest way to generate a relative perturbation bound is to divide an absolute error bound by a (nonzero) eigenvalue. In the case of the Bauer–Fike theorem we get

$$\frac{|\lambda - \hat{\lambda}|}{|\lambda|} \leq \kappa(X)\, \frac{\|E\|}{|\lambda|}.$$

Now the bound depends on λ. In particular, the bound is smaller for eigenvalues λ that are large in magnitude than for those that are small in magnitude.

However, this kind of relative perturbation bound may not be good enough because there are algorithms that compute all eigenvalues to high relative accuracy—even those of small magnitude. Among such algorithms are Jacobi methods for Hermitian positive-definite matrices [4, 13] and the dqds algorithm for certain tridiagonal

*Received by the editors June 23, 1997; accepted for publication (in revised form) by R. Bhatia March 3, 1998; published electronically September 23, 1998.

http://www.siam.org/journals/simax/20-1/32328.html

†Department of Computer Science, Yale University, P. O. Box 208285, New Haven, CT 06520-8285 (eisenstat-stan@cs.yale.edu). The research of this author was supported in part by NSF grant CCR-9400921.

‡Center for Research in Scientific Computation, Department of Mathematics, North Carolina State University, P.O. Box 8205, Raleigh, NC 27695-8205 (ipsen@math.ncsu.edu, http://www4.ncsu.edu/~ipsen/info.html). The research of this author was supported in part by NSF grant CCR-9400921.

matrices [7]. These algorithms have "genuine" relative error bounds that do not depend on the eigenvalues.

Our original motivation was to determine under which circumstances we can find genuine relative perturbation bounds that do not depend on the eigenvalues. In particular, does the existence of such a bound depend on the properties of the matrix (e.g., Hermitian positive-definite) or on the properties of the perturbation (e.g., relative componentwise)?

Our answer is that genuine relative perturbation bounds exist, whenever absolute bounds exist, for almost any matrix and for any perturbation. In particular, we show that three well-known absolute bounds imply genuine relative bounds. In this sense relative bounds are no stronger than absolute bounds. We also show that corresponding absolute and relative perturbation bounds have the same condition number.

1.1. Overview.

In section 2 we show that the Bauer–Fike theorem for diagonalizable matrices implies a large class of relative bounds. The condition number is the same for relative and absolute bounds. We conclude that the eigenvalues of a normal, nonsingular matrix are well conditioned in the absolute as well as in the relative sense.

In section 3 we derive a relative perturbation bound for normal matrices that suggests that the eigenvalues of a normal matrix are as well conditioned as the eigenvalues of its positive-definite polar factor. The bound is invariant under congruence transformations. Hence the eigenvalues of a graded, normal matrix are no more sensitive to perturbations than the eigenvalues of an "ungraded" Hermitian positive-definite matrix.

In section 4 we show that Weyl's perturbation theorem implies a relative bound.

In section 5 we extend the Hoffman–Wielandt theorem for diagonalizable matrices and show that it implies a relative bound.

1.2. Notation.

I is the identity matrix; $\|\cdot\|$ is the two-norm and $\|\cdot\|_F$ the Frobenius norm; A^* is the conjugate transpose of a matrix A; and $\kappa(X) \equiv \|X\|\,\|X^{-1}\|$ is the two-norm condition number of a matrix X with respect to inversion.

2. Two-norm bounds for diagonalizable matrices.

We show that the Bauer–Fike theorem implies a relative bound.

Let A be a diagonalizable matrix with eigendecomposition $A = X\Lambda X^{-1}$, where

$$\Lambda = \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix}$$

and λ_i are the eigenvalues of A. The Bauer–Fike theorem bounds the absolute error between a perturbed eigenvalue and a closest eigenvalue of A.

THEOREM 2.1 (Theorem IIIa in [2]). *If A is diagonalizable, then*

$$\min_i |\lambda_i - \hat{\lambda}| \leq \kappa(X)\,\|E\|,$$

where $\kappa(X) \equiv \|X\|\,\|X^{-1}\|$.

The Bauer–Fike theorem implies the relative bound below, provided A is nonsingular.

COROLLARY 2.2. *If A is diagonalizable and nonsingular, then*

$$\min_i \frac{|\lambda_i - \hat{\lambda}|}{|\lambda_i|} \leq \kappa(X)\,\|A^{-1}E\|.$$

Proof. Write $(A+E)\hat{x} = \hat{\lambda}\hat{x}$ as

$$(\bar{A}+\bar{E})\,\hat{x} = \hat{x}, \qquad \text{where} \qquad \bar{A} \equiv \hat{\lambda}A^{-1}, \quad \bar{E} \equiv -A^{-1}E.$$

Hence 1 is an eigenvalue of $\bar{A}+\bar{E}$. The matrix $\bar{A}$ has eigenvalues $\hat{\lambda}/\lambda_i$ and the same eigenvector matrix as A. Apply Theorem 2.1 to $\bar{A}$ and the eigenvalue 1 of $\bar{A}+\bar{E}$. □

We interpret the amplifier $\kappa(X)$ in the bounds as a condition number for the eigenvalues of A. Both absolute and relative perturbation bounds have the same condition number. The condition number indicates the sensitivity of an eigenvalue to absolute perturbations E and to relative perturbations

$$A^{-1}E = A^{-1}\left((A+E)-A\right).$$

Note, however, that an eigenvalue closest to $\hat{\lambda}$ in the absolute sense may be different from an eigenvalue closest to $\hat{\lambda}$ in the relative sense. Corollary 2.2 generalizes [16, Theorem 3.12].

2.1. A larger class of relative perturbations. So far we have expressed relative perturbations as $A^{-1}E$. But why confine A^{-1} to the left of E? Why not move it to the right? Or why not factor A^{-1} and distribute the factors on both sides of E? The bound below is a consequence of Corollary 2.2.

THEOREM 2.3. *Let A be diagonalizable and nonsingular. If $A = A_1A_2$, then*

$$\min_i \frac{|\lambda_i - \hat{\lambda}|}{|\lambda_i|} \leq \kappa(W)\,\|A_1^{-1}EA_2^{-1}\|,$$

where W is an eigenvector matrix of A_2A_1.

Proof. Define

$$\bar{A} \equiv A_2AA_2^{-1}, \qquad \bar{E} \equiv A_2EA_2^{-1}.$$

Since $\bar{A}$ is similar to A, it is diagonalizable with eigendecomposition $\bar{A} = W\Lambda W^{-1}$. Applying Corollary 2.2 to $\bar{A}$ and $\bar{A}+\bar{E}$ gives

$$\min_i \frac{|\lambda_i - \hat{\lambda}|}{|\lambda_i|} \leq \kappa(W)\,\|\bar{A}^{-1}\bar{E}\| = \kappa(W)\,\|A_1^{-1}EA_2^{-1}\|. \qquad \square$$

When A_1 and A_2 commute, the original condition number $\kappa(X)$ returns.

COROLLARY 2.4. *Let $A = A_1A_2$ be diagonalizable and nonsingular. If $A_1A_2 = A_2A_1$, then*

$$\min_i \frac{|\lambda_i - \hat{\lambda}|}{|\lambda_i|} \leq \kappa(X)\,\|A_1^{-1}EA_2^{-1}\|.$$

When $A_1 = A$ and $A_2 = I$, we recover Corollary 2.2. The choice $A_1 = I$ and $A_2 = A$ gives a similar bound.

COROLLARY 2.5. *Let A be diagonalizable and nonsingular. Then*

$$\min_i \frac{|\lambda_i - \hat{\lambda}|}{|\lambda_i|} \leq \kappa(X)\,\|EA^{-1}\|.$$

Another popular choice for A_1 and A_2 is a square root $A^{1/2}$ of A. A square root of a matrix [9, p. 54, Problem 7], [10, pp. 467 and 471] exists whenever the matrix is nonsingular [10, p. 468].

COROLLARY 2.6. *Let A be diagonalizable and nonsingular. Then*

$$\min_i \frac{|\lambda_i - \hat{\lambda}|}{|\lambda_i|} \leq \kappa(X)\, \|A^{-1/2} E A^{-1/2}\|.$$

3. Two-norm bounds for normal matrices.

We derive relative perturbation bounds for normal matrices that are invariant under congruence transformations such as grading and scaling.

When applied to normal matrices, Corollary 2.4 is simplified.

COROLLARY 3.1. *Let $A = A_1 A_2$ be normal and nonsingular. If $A_1 A_2 = A_2 A_1$, then*

$$\min_i \frac{|\lambda_i - \hat{\lambda}|}{|\lambda_i|} \leq \|A_1^{-1} E A_2^{-1}\|.$$

Therefore, eigenvalues of normal matrices are well conditioned in the absolute sense as well as in many relative senses.

Up to now we have chosen the following factorizations for A:

$$(A_1, A_2) = (A, I), \qquad (A_1, A_2) = (I, A), \qquad (A_1, A_2) = (A^{1/2}, A^{1/2}).$$

In each case, A_1 and A_2 commute and we retain the perfect conditioning of normal matrices. Normal matrices, however, admit another commuting factorization. It results from the polar factorization and the fact that polar factors of normal matrices commute.

Every nonsingular matrix A has a polar factorization $A = HU$, where $H \equiv (AA^*)^{1/2}$ is Hermitian positive-definite, and $U \equiv H^{-1}A$ is unitary [9, Corollary 7.3.3]. We use the following property of polar factors.

LEMMA 3.2. *Let A be normal and nonsingular with polar factorization $A = HU$. Then*

$$HU = UH = H^{1/2} U H^{1/2}.$$

Proof. The first equality follows from the fact that polar factors of a normal matrix commute [9, Theorem 7.3.4].

To prove the second equality, note that a Hermitian positive-definite matrix H has a unique Hermitian positive-definite square root $H^{1/2}$ [9, Theorem 7.2.6]. From the first equality, $HU = UH$ follows

$$H = UHU^* = (UH^{1/2}U^*)\,(UH^{1/2}U^*).$$

Thus $UH^{1/2}U^*$ is also a Hermitian positive-definite square root of H. However, uniqueness implies $H^{1/2} = UH^{1/2}U^*$. This means that $H^{1/2}U = UH^{1/2}$ and

$$A = HU = H^{1/2}\,(H^{1/2}U) = H^{1/2} U H^{1/2}. \qquad \square$$

The following bound is a consequence of Corollary 2.4.

THEOREM 3.3. *Let A be normal and nonsingular, with Hermitian positive-definite polar factor H. Then*

$$\min_i \frac{|\lambda_i - \hat{\lambda}|}{|\lambda_i|} \leq \|H^{-1/2} E H^{-1/2}\|.$$

Proof. $A = HU$ is a polar factorization of A, and Lemma 3.2 implies $A = H^{1/2} U H^{1/2}$. Set

$$A_1 \equiv H^{1/2} U, \qquad A_2 \equiv H^{1/2}.$$

Then

$$\|A_1^{-1} E A_2^{-1}\| = \|U^* H^{-1/2} E H^{-1/2}\| = \|H^{-1/2} E H^{-1/2}\|.$$

Since $A_2 A_1 = A_1 A_2$, we can apply Corollary 2.4 to get the desired bound. □

Therefore the eigenvalues of a normal matrix have the same relative error bound as the eigenvalues of its positive-definite polar factor, which suggests that they are as well conditioned as the eigenvalues of its positive-definite polar factor.

3.1. Grading. The advantage of Theorem 3.3 is that it is invariant under congruence transformations in the following sense.

COROLLARY 3.4. *Let A be normal and nonsingular, with Hermitian positive-definite polar factor H. If D is nonsingular and*

$$E = D E_1 D^*, \qquad H = D M_1 D^*,$$

then

$$\min_i \frac{|\lambda_i - \hat{\lambda}|}{|\lambda_i|} \leq \|M_1^{-1/2} E_1 M_1^{-1/2}\|.$$

Proof. Theorem 3.3 implies

$$\min_i \frac{|\lambda_i - \hat{\lambda}|}{|\lambda_i|} \leq \|H^{-1/2} E H^{-1/2}\|.$$

Since H is Hermitian positive-definite, $M_1 = D^{-1} H D^{-*}$ is also Hermitian positive-definite and has a Hermitian square root. Since

$$H = H^{1/2} H^{1/2} = D M_1^{1/2} \, (D M_1^{1/2})^*$$

are both "Cholesky factorizations" of H, they are related by a unitary matrix Q, i.e., $H^{1/2} = (D M_1^{1/2}) \, Q$. Hence by Theorem 3.3,

$$\min_i \frac{|\lambda_i - \hat{\lambda}|}{|\lambda_i|} \leq \|H^{-1/2} E H^{-1/2}\| = \|M_1^{-1/2} E_1 M_1^{-1/2}\|. \qquad □$$

Therefore the relative error bound is invariant under congruence transformations extracted from the perturbation and the positive-definite polar factor. Corollary 3.4 implies essentially that the eigenvalues of a graded, normal matrix are no more sensitive than the eigenvalues of the best "ungraded" positive-definite polar factor.

3.2. Relation to existing work. Slapničar and Veselić [15, section 2], [16, section 2] have obtained similar results. Their results are more general in the sense that they apply to the generalized eigenvalue problem $Ax = \lambda Bx$, where A is Hermitian and B is Hermitian positive-definite, and they bound the distance between the ith perturbed and exact eigenvalues. When $B = I$, their *absolute value of* A, $|A|$ turns out to be a Hermitian positive-definite polar factor of A. However, our results are more general in the sense that they apply to a larger class of matrices (normal as opposed to Hermitian), a larger class of perturbations (normwise as opposed to componentwise), and to a larger class of grading matrices (nonsingular as opposed to real diagonal).

To relate our results to those in [15, section 2], [16, section 2], we assume that the backward error is scaled in the same way as the matrix so that $\|E_1\| \le \epsilon \|M\|$, where ϵ is a small positive number and $A = DMD^*$. The following result is similar in spirit to [16, Theorem 2.13].

COROLLARY 3.5. *Let A be normal and nonsingular, with positive-definite polar factor H. Let D be nonsingular and*

$$A = DMD^*, \qquad E = DE_1D^*, \qquad H = DM_1D^*.$$

If $\|E_1\| \le \epsilon \|M\|$, then

$$\min_i \frac{|\lambda_i - \hat{\lambda}|}{|\lambda_i|} \le \epsilon \, \|M\| \|M_1^{-1}\|.$$

Proof. Corollary 3.4 implies

$$\min_i \frac{|\lambda_i - \hat{\lambda}|}{|\lambda_i|} \le \|M_1^{-1/2} E_1 M_1^{-1/2}\| \le \epsilon \, \|M_1^{-1/2}\|^2 \|M\|.$$

As a square root of the Hermitian positive-definite matrix M_1, $M_1^{1/2}$ is Hermitian. Therefore $\|M_1^{-1/2}\|^2 = \|M_1^{-1}\|$. □

Therefore the eigenvalues have small relative error when M and M_1^{-1} have small norm. Here, M is what is left over after extracting the grading from A, and M_1 is what is left over after extracting the grading from the positive-definite polar factor. One might argue that in Corollary 3.5 the polar factor of M would be preferable to the polar factor of A. However, then we would be comparing apples and oranges. Because A and H have the same eigenvalues (in magnitude), we have to compare the scaled version of A (which is M) to the scaled version of H (which is M_1).

In the special case where M is unitary we arrive at the same conclusion as [16, Theorem 2.37], namely, that the eigenvalues of A are well conditioned.

COROLLARY 3.6. *If, in addition to the assumptions of Corollary* 3.5*, D also commutes with the unitary polar factor of A and M is unitary, then*

$$\min_i \frac{|\lambda_i - \hat{\lambda}|}{|\lambda_i|} \le \epsilon.$$

Proof. $\|M\| = 1$ because M is unitary, and

$$A = UH = U \, DM_1D^* = D \, UM_1 \, D^*$$

because D and U commute. However, $A = DMD^*$ and the nonsingularity of D imply $M = UM_1$. Hence M_1 is unitary and $\|M_1^{-1}\| = 1$. □

In conclusion, our results only bound the error in a single eigenvalue of $A+E$ while other results bound the relative error in all eigenvalues of $A+E$ simultaneously. But this comes at a price. For instance, existing bounds for Hermitian matrices restrict the perturbation E so that $A+E$ is Hermitian and has the same inertia as A [1, Lemma 1], [16, Theorem 2.1], or they restrict the congruence transformation D and the size of the perturbation [8, Corollary 5]. Hence, we have traded simultaneous bounds for all eigenvalues against freedom in perturbations and applicability to a larger class of matrices.

4. Two-norm bounds for Hermitian matrices.

We show that Weyl's theorem implies a relative bound.

Let A and $A+E$ be Hermitian with respective eigenvalues

$$\lambda_n \leq \cdots \leq \lambda_1, \qquad \hat{\lambda}_n \leq \cdots \leq \hat{\lambda}_1.$$

Weyl's perturbation theorem bounds the worst-case absolute error between the ith exact and perturbed eigenvalues of Hermitian matrices in terms of the two-norm.

THEOREM 4.1 (Corollary III.2.6 in [3]). *If A and $A+E$ are Hermitian, then*

$$\max_{1\leq i\leq n} |\lambda_i - \hat{\lambda}_i| \leq \|E\|.$$

The absolute bound in Theorem 4.1 implies a relative bound, provided that the matrices are, in addition, positive-definite.

COROLLARY 4.2 (Theorem 2.3 in [14]). *If A and $A+E$ are Hermitian positive-definite, then*

$$\max_{1\leq i\leq n} \frac{|\lambda_i - \hat{\lambda}_i|}{|\lambda_i|} \leq \|A^{-1/2}EA^{-1/2}\|.$$

Proof. Fix an index i. Let $\hat{x}$ be an eigenvector of $A+E$ associated with $\hat{\lambda}_i$, i.e.,

$$(A+E)\hat{x} = \hat{\lambda}_i\hat{x}.$$

Multiplying $(\hat{\lambda}_i I - E)\hat{x} = A\hat{x}$ by A^{-1} gives

$$(\bar{A}+\bar{E})\, z = z, \qquad \text{where} \qquad \bar{A} \equiv \hat{\lambda}_i A^{-1}, \quad \bar{E} \equiv -A^{-1/2}EA^{-1/2}, \quad z \equiv A^{1/2}\hat{x}.$$

Hence 1 is an eigenvalue of $\bar{A}+\bar{E}$.

We will show that it is actually the $(n-i+1)$st eigenvalue. We argue as in the proof of [5, Theorem 2.1]. Since $\hat{\lambda}_i$ is the ith eigenvalue of $A+E$, 0 must be the ith eigenvalue of

$$(A+E) - \hat{\lambda}_i I = A^{1/2}\,(I - \bar{A} - \bar{E})\, A^{1/2}.$$

However, this is a congruence transformation because square roots of positive-definite matrices are Hermitian. Congruence transformations preserve the inertia. Hence 0 is the ith eigenvalue of $I - \bar{A} - \bar{E}$, and 1 is the $(n-i+1)$st eigenvalue of $\bar{A}+\bar{E}$.

Since $A+E$ is positive-definite, $\hat{\lambda}_i$ is positive and $\hat{\lambda}_i/\lambda_{n-j+1}$ is the jth eigenvalue of $\bar{A}$. Applying Theorem 4.1 to $\bar{A}$ and $\bar{A}+\bar{E}$ gives

$$\max_{1\leq j\leq n} \left| \frac{\hat{\lambda}_i}{\lambda_{n-j+1}} - \mu_j \right| \leq \|\bar{E}\| \leq \|A^{-1/2}EA^{-1/2}\|,$$

where μ_j are the eigenvalues of $\bar{A}+\bar{E}$. When $j = n-i+1$, then $\mu_{n-i+1} = 1$ and we get the desired bound. □

A slightly weaker bound with a restriction on $A+E$ appears in [11, Theorem 3.2].

5. Frobenius norm bounds for diagonalizable matrices. We show that a slightly stronger version of the Hoffman–Wielandt theorem for diagonalizable matrices implies a relative bound. The idea for this proof was inspired by the derivation of relative error bounds for multiplicative perturbations in [11].

Let A and $A+E$ be diagonalizable with eigendecompositions $A = X\Lambda X^{-1}$ and $A+E = \hat{X}\hat{\Lambda}\hat{X}^{-1}$, respectively. The eigenvalues are

$$\Lambda = \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix}, \qquad \hat{\Lambda} = \begin{pmatrix} \hat{\lambda}_1 & & \\ & \ddots & \\ & & \hat{\lambda}_n \end{pmatrix}.$$

The Hoffman–Wielandt theorem for diagonalizable matrices [6, Theorem 3.1] establishes a one-to-one pairing between exact and perturbed eigenvalues and bounds the sum of absolute errors in the Frobenius norm,

$$\sqrt{\sum_{i=1}^{n} |\lambda_i - \hat{\lambda}_{\tau(i)}|^2} \leq \kappa(\hat{X})\,\kappa(X)\ \|E\|_F \tag{1}$$

for some permutation τ. Note that $\kappa(X)$ and $\kappa(\hat{X})$ are expressed in the two-norm, rather than in the Frobenius norm. This makes the bound tighter because the two-norm never exceeds the Frobenius norm.

To demonstrate that an absolute Hoffman–Wielandt-type bound implies a relative version, we need an absolute bound that is slightly stronger than (1). We replace A by a product AC. The perturbed matrix is $AC+E$, where C must have the same eigenvector matrix as $AC+E$. The bound (1) is the special case where $C = I$. The eigendecomposition of C is

$$C = \hat{X}\Gamma\hat{X}^{-1}, \qquad \text{where} \qquad \Gamma = \begin{pmatrix} \gamma_1 & & \\ & \ddots & \\ & & \gamma_n \end{pmatrix}.$$

The eigendecompositions of A and the perturbed matrix remain the same,

$$A = X\Lambda X^{-1}, \qquad AC + E = \hat{X}\hat{\Lambda}\hat{X}^{-1}.$$

The stronger Hoffman–Wielandt theorem below bounds the sum of absolute errors in the products of the eigenvalues of A and C.

THEOREM 5.1. *Let A, C, and $AC+E$ be diagonalizable. There exists a permutation τ so that*

$$\sqrt{\sum_{i=1}^{n} |\lambda_i \gamma_{\tau(i)} - \hat{\lambda}_{\tau(i)}|^2} \leq \kappa(\hat{X})\,\kappa(X)\ \|E\|_F.$$

Proof. The proof is similar to that of [6, Theorem 3.1].

In $AC - \hat{X}\hat{\Lambda}\hat{X}^{-1} = -E$ decompose A and C,

$$X\Lambda X^{-1}\,\hat{X}\Gamma\hat{X}^{-1} - \hat{X}\hat{\Lambda}\hat{X}^{-1} = -E.$$

Multiply on the left by X^{-1} and on the right by $\hat{X}$ and set $Z \equiv X^{-1}\hat{X}$,

$$\Lambda Z \Gamma - Z\hat{\Lambda} = -X^{-1}E\hat{X}. \tag{2}$$

The (i,j)th element of this equation has absolute value

$$|z_{ij}|\,|\lambda_i\gamma_j-\hat\lambda_j| = \left|(X^{-1}E\hat X)_{ij}\right|.$$

The Frobenius norm is the sum of the squares of all these elements,

$$\sum_{i,j}|z_{ij}|^2\,|\lambda_i\gamma_j-\hat\lambda_j|^2 = \|X^{-1}E\hat X\|_F^2 \le \|X^{-1}\|^2\,\|\hat X\|^2\,\|E\|_F^2.$$

By [6, Main Theorem], there exists a doubly stochastic matrix $S=(s_{ij})$ so that

$$\frac{s_{ij}}{\|Z^{-1}\|^2}\le |z_{ij}|^2, \qquad 1\le i,j\le n.$$

Hence

$$\begin{aligned}\sum_{i,j}s_{ij}\,|\lambda_i\gamma_j-\hat\lambda_j|^2 &\le \|Z^{-1}\|^2\sum_{i,j}|z_{ij}|^2\,|\lambda_i\gamma_j-\hat\lambda_j|^2\\ &\le \kappa(X)^2\,\kappa(\hat X)^2\,\|E\|_F^2.\end{aligned}$$

Because S is doubly stochastic, Birkhoff's theorem [3, section II.2] implies that there exists a permutation τ with

$$\sum_i|\lambda_i\gamma_{\tau(i)}-\hat\lambda_{\tau(i)}|^2\le\sum_{i,j}s_{ij}\,|\lambda_i\gamma_j-\hat\lambda_j|^2.$$

Therefore

$$\sum_i|\lambda_i\gamma_{\tau(i)}-\hat\lambda_{\tau(i)}|^2\le\kappa(X)^2\,\kappa(\hat X)^2\,\|E\|_F^2. \qquad \square$$

The stronger absolute bound in Theorem 5.1 implies a relative bound, provided A is nonsingular. This relative bound is not new. It follows, for instance, from the multiplicative bound [12, Theorem 2.1′] with $D_1=I$ and $D_2=I+A^{-1}E$. However, the proof below demonstrates that the relative bound is no stronger than the absolute bound because it is implied by the absolute bound.

COROLLARY 5.2. *Let A and $A+E$ be diagonalizable. If A is also nonsingular, then there exists a permutation τ so that*

$$\sqrt{\sum_{i=1}^n\left(\frac{|\lambda_i-\hat\lambda_{\tau(i)}|}{|\lambda_i|}\right)^2}\le\kappa(\hat X)\,\kappa(X)\,\|A^{-1}E\|_F.$$

Proof. Since $A^{-1}(A+E)-A^{-1}E=I$, we can set

$$\bar A\equiv A^{-1}, \qquad C\equiv A+E, \qquad \bar E\equiv -A^{-1}E.$$

Then $\bar A$ is diagonalizable with eigenvector matrix X and eigenvalues λ_i^{-1}; C is diagonalizable with eigenvector matrix $\hat X$ and eigenvalues $\hat\lambda_i$; and $\bar AC+\bar E=\hat XI\hat X^{-1}$ is diagonalizable, where the eigenvalues are 1 and we can choose $\hat X$ as an eigenvector matrix. Applying Theorem 5.1 to $\bar A$, C, and $\bar E$ gives

$$\sum_{i=1}^n|\lambda_i^{-1}\hat\lambda_{\tau(i)}-1|^2\le\kappa(\hat X)^2\kappa(X)^2\|A^{-1}E\|_F^2. \qquad \square$$

REFERENCES

[1] J. Barlow and J. Demmel, *Computing accurate eigensystems of scaled diagonally dominant matrices*, SIAM J. Numer. Anal., 27 (1990), pp. 762–791.

[2] F. Bauer and C. Fike, *Norms and exclusion theorems*, Numer. Math., 2 (1960), pp. 137–141.

[3] R. Bhatia, *Matrix Analysis*, Springer-Verlag, New York, 1997.

[4] J. Demmel and K. Veselić, *Jacobi's method is more accurate than QR*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 1204–1245.

[5] S. Eisenstat and I. Ipsen, *Relative perturbation techniques for singular value problems*, SIAM J. Numer. Anal., 32 (1995), pp. 1972–1988.

[6] L. Elsner and S. Friedland, *Singular values, doubly stochastic matrices, and applications*, Linear Algebra Appl., 220 (1995), pp. 161–169.

[7] K. Fernando and B. Parlett, *Accurate singular values and differential qd algorithms*, Numer. Math., 67 (1994), pp. 191–229.

[8] M. Gu and S. Eisenstat, *Relative Perturbation Theory for Eigenproblems*, Research Report YALEU/DCS/RR-934, Department of Computer Science, Yale University, New Haven, CT, 1993.

[9] R. Horn and C. Johnson, *Matrix Analysis*, Cambridge University Press, Cambridge, 1985.

[10] R. Horn and C. Johnson, *Topics in Matrix Analysis*, Cambridge University Press, Cambridge, 1991.

[11] R. Li, *Relative perturbation theory:* I. *Eigenvalue and singular value variations*, SIAM J. Matrix Anal. Appl., 19 (1998), pp. 956–982.

[12] R. Li, *Relative perturbation theory.* III. *More bounds on eigenvalue variation*, Linear Algebra Appl., 266 (1997), pp. 337–345.

[13] R. Mathias, *Accurate eigensystem computations by Jacobi methods*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 977–1003.

[14] R. Mathias, *Spectral perturbation bounds for positive definite matrices*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 959–980.

[15] I. Slapničar, *Accurate Symmetric Eigenreduction by a Jacobi Method*, Ph.D. thesis, Fernuniversität Gesamthochschule Hagen, Germany, 1992.

[16] K. Veselić and I. Slapničar, *Floating-point perturbations of Hermitian matrices*, Linear Algebra Appl., 195 (1993), pp. 81–116.

SIAM J. Matrix Anal. Appl.
Vol. 20, No. 1, pp. 159–181

ON THE STABILITY OF A PARTITIONING ALGORITHM FOR TRIDIAGONAL SYSTEMS*

PLAMEN YALAMOV† AND VELISAR PAVLOV†

Abstract. Componentwise error analysis for a parallel partitioning algorithm for tridiagonal systems is presented. Bounds on the equivalent perturbations are obtained, depending on three constants. Then bounds on the forward error are presented as well, depending on two types of condition numbers. Estimates on the first constant come directly from the roundoff error analysis of the tridiagonal Gaussian elimination [N. Higham, *SIAM J. Matrix Anal. Appl.*, 11 (1990), pp. 521–530]. In the present paper, the second and third constants are bounded for some special classes of matrices, i.e., diagonally dominant (row or column), symmetric positive definite, M-matrices, and totally nonnegative.

One of the features of the analysis is that the exact forward and backward errors are bounded, not just their first order approximations, with respect to the machine precision. In all the bounds, the linear terms are given separately to show that the terms of higher order are small enough.

Key words. parallel partitioning algorithm, roundoff error analysis, tridiagonal matrices, diagonal dominance, symmetric positive definite matrix, M-matrix, nonnegative matrix

AMS subject classifications. 65G05, 65F05, 65Y05

PII. S0895479896314418

1. Introduction.

Large tridiagonal systems appear in many applications, such as finite elements, difference schemes to differential equations, power distribution systems, etc. Among the direct methods for solving such systems, we can mention the partition methods, which include very efficient parallel algorithms [1, 2, 3, 6, 7, 10, 14, 15, 16, 19, 20, 22, 24]. Concerning partition methods, a unifying approach for their derivation and study is given in [1, 3]. Stability analysis of other parallel algorithms for tridiagonal and bidiagonal systems can be found in [26, 27].

The aim of this paper is to obtain a stability analysis of the partitioning algorithm proposed by Wang in [24].

Let the linear system under consideration be denoted by

$$Ax = d, \tag{1}$$

where $A = \mathrm{tridiag}(a, b, c)$ is a tridiagonal matrix, a, b, c, d are vectors of size n,

$$a = (0, a_2, a_3, \ldots, a_n)^T, \qquad b = (b_1, b_2, \ldots, b_n)^T,$$
$$c = (c_1, c_2, \ldots, c_{n-1}, 0)^T, \qquad d = (d_1, d_2, \ldots, d_n)^T.$$

*Received by the editors December 27, 1996; accepted for publication (in revised form) by L. Reichel January 23, 1998; published electronically September 23, 1998. This research was supported by grants MM-707/97 and I-702/97 from the National Scientific Research Fund of the Bulgarian Ministry of Education and Science.

http://www.siam.org/journals/simax/20-1/31441.html

†Center of Applied Mathematics and Informatics, University of Rousse, 7017 Rousse, Bulgaria (yalamov@ami.ru.acad.bg, velisar@ami.ru.acad.bg).

Let us assume for simplicity that $n = ks - 1$ for some integer k, if s is the number of the parallel processors we want to use. We partition matrix A, the solution x, and the right-hand side d of the system (1) as follows:

$$A = \begin{pmatrix} B_1 & \bar{c}_1 & & & & & & & \\ a_k & b_k & c_k & & & & & & \\ & \bar{a}_2 & B_2 & \bar{c}_2 & & & & & \\ & & a_{2k} & b_{2k} & c_{2k} & & & & \\ & & & \ddots & \ddots & \ddots & & & \\ & & & & \bar{a}_{s-1} & B_{s-1} & \bar{c}_{s-1} & & \\ & & & & & a_{(s-1)k} & b_{(s-1)k} & c_{(s-1)k} & \\ & & & & & & \bar{a}_s & B_s & \end{pmatrix},$$

$$x = \left(X_1^T, x_k, X_2^T, x_{2k}, \ldots, X_{s-1}^T, x_{(s-1)k}, X_s^T\right)^T,$$
$$d = \left(D_1^T, d_k, D_2^T, d_{2k}, \ldots, D_{s-1}^T, d_{(s-1)k}, D_s^T\right)^T,$$

where $B_i \in \mathcal{R}^{(k-1)\times(k-1)},\ i = 1, 2, \ldots, s$ is a tridiagonal matrix,

$$B_i = \begin{pmatrix} b_{(i-1)k+1} & c_{(i-1)k+1} & & & \\ a_{(i-1)k+2} & b_{(i-1)k+2} & c_{(i-1)k+2} & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & c_{ik-2} \\ & & & a_{ik-1} & b_{ik-1} \end{pmatrix},$$

$\bar{a}_i \in \mathcal{R}^{(k-1)\times 1}, i = 2, \ldots, s,\ \bar{c}_i \in \mathcal{R}^{(k-1)\times 1}, i = 1, \ldots, s-1$ are vectors of the kind

$$\bar{a}_i = (a_{(i-1)k+1}, 0, \ldots, 0)^T, \quad \bar{c}_i = (0, \ldots, 0, c_{ik-1})^T,$$

and $X_i, D_i \in \mathcal{R}^{(k-1)\times 1}, i = 1, \ldots, s$ are vectors of the form

$$X_i = (x_{(i-1)k+1}, x_{(i-1)k+2}, \ldots, x_{ik-1})^T,$$
$$D_i = (d_{(i-1)k+1}, d_{(i-1)k+2}, \ldots, d_{ik-1})^T.$$

In this paper we present Wang's algorithm in a block form which is more appropriate for the following analysis. For this purpose we define the permutation

$$[1 : k-1, \ldots, (i-1)k+1 : ik-1, \ldots, (s-1)k+1 : sk-1, k, \ldots, ik, \ldots, (s-1)k]$$

of the numbers $[1, \ldots, sk-1]$ and denote the corresponding permutation matrix by $\mathcal{P}$. By applying this permutation to the rows and columns of matrix A, we obtain the system

$$\mathcal{A}\mathcal{P}x = \mathcal{P}d, \quad \text{with} \quad \mathcal{A} = \mathcal{P}A\mathcal{P}^T = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix},$$

where $A_{11} = \operatorname{diag}\{B_1, B_2, \ldots, B_s\} \in \mathcal{R}^{s(k-1)\times s(k-1)}$,

$$A_{12} = \begin{pmatrix} \bar{c}_1 & & & \\ \bar{a}_2 & \bar{c}_2 & & \\ & \ddots & \ddots & \\ & & \ddots & \bar{c}_{s-1} \\ & & & \bar{a}_s \end{pmatrix} \in \mathcal{R}^{s(k-1)\times(s-1)}, \tag{2}$$

$$A_{21} = \begin{pmatrix} 0 & \cdots & a_k & c_k & \cdots & 0 & & & & & & \\ & & & 0 & \cdots & a_{2k} & c_{2k} & \cdots & 0 & & & \\ & & & & \ddots & & \ddots & \ddots & & \ddots & & \\ & & & & & & 0 & \cdots & a_{(s-1)k} & c_{(s-1)k} & \cdots & 0 \end{pmatrix},$$

$A_{21} \in \mathcal{R}^{(s-1)\times s(k-1)}$, and $A_{22} = \operatorname{diag}(b_k, b_{2k}, \ldots, b_{(s-1)k}) \in \mathcal{R}^{(s-1)\times(s-1)}$.

We will distinguish between the two matrices A (original) and $\mathcal{A}$ (permuted). Evidently, the permutation does not influence the roundoff error analysis but it influences some properties of the matrices that we consider in section 3 (e.g., totally nonnegative matrices). For this reason we use these two notations. The permuted vectors $\mathcal{P}x$ and $\mathcal{P}d$ are not frequently used in the paper. We will stay with the same notation, i.e., x and d, and give explicitly its permuted components when necessary, or write $\mathcal{P}x$ and $\mathcal{P}d$ for the permuted vectors. Otherwise, we will need some error bounds on x with respect to the infinity norm, but it is clear that these bounds are not influenced by the permutation, and this will not lead to confusion.

The algorithm can be presented as follows.

Stage 1. Obtain the block LU factorization

$$\mathcal{A} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = LU = \begin{pmatrix} A_{11} & 0 \\ A_{21} & I_{s-1} \end{pmatrix} \begin{pmatrix} I_{s(k-1)} & R \\ 0 & S \end{pmatrix} \tag{3}$$

by the following steps:

1. Obtain the LU-factorization of $A_{11} = \mathcal{P}_1 L_1 U_1$ with partial pivoting, if necessary. Here $\mathcal{P}_1$ is a permutation matrix, L_1 is unit lower triangular, and U_1 is upper triangular.
2. Solve $A_{11}R = A_{12}$ using the LU factorization from the previous item, and compute $S = A_{22} - A_{21}R$, which is the Schur complement of A_{11} in $\mathcal{A}$.

Stage 2. Solve $Ly = d$ by using the LU factorization of A_{11} (Stage 1).

Stage 3. Solve $Ux = y$ by applying Gaussian elimination (with pivoting, if necessary) to the block S.

Because of the block diagonal structure of A_{11}, most of the computations are well parallelized. Let us note that the blocks L_1 and U_1 inherit the block diagonal structure of A_{11}. The block R is quite sparse and is also structured. If we take into account the structure of A_{12} in (2), then it is clear that

$$R = \begin{pmatrix} p^{(1)} & & & \\ q^{(2)} & p^{(2)} & & \\ & \ddots & \ddots & \\ & & \ddots & p^{(s-1)} \\ & & & q^{(s)} \end{pmatrix} \in \mathcal{R}^{s(k-1)\times(s-1)},$$

where

$$p^{(i)} = (p_{(i-1)k+1}, p_{(i-1)k+2}, \dots, p_{ik-1})^T \in \mathcal{R}^{(k-1)\times 1},$$
$$q^{(i)} = (q_{(i-1)k+1}, q_{(i-1)k+2}, \dots, q_{ik-1})^T \in \mathcal{R}^{(k-1)\times 1}.$$

So, most of the computations in Stage 3 are also well parallelized because of the block structure of submatrix R.

Let us note that matrix S (the so-called reduced matrix) is also tridiagonal. We shall need an explicit notation for its entries in the following. So, we assume that

$$S = \begin{pmatrix} v_1 & w_1 & & & \\ u_2 & v_2 & w_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & w_{s-2} \\ & & & u_{s-1} & v_{s-1} \end{pmatrix}, \tag{4}$$

where

$$u_i = -a_{ik}q_{ik-1},\ v_i = b_{ik} - a_{ik}p_{ik-1} - c_{ik}q_{ik+1},\ w_i = -c_{ik}p_{ik+1}.$$

In some publications, stability issues of this partitioning algorithm have been studied. More precisely, in [3] it is shown that the reduced matrix S is diagonally dominant, when matrix A is diagonally dominant. The same property holds when A is a symmetric positive definite (s.p.d.) matrix [4, p. 94], or an M-matrix [4, p. 209]. Another work concerning such a property is [23]. In this paper it is proved that if A is strictly diagonally dominant (in a more general sense than [3]), then the reduced matrix S is strictly diagonally dominant. So, in these three cases we can expect that the whole algorithm is stable because the Gaussian elimination at the separate stages is stable for diagonally dominant, s.p.d., and M-matrices. Also we should mention reference [2] in which some stability issues of the present algorithm are studied. It is shown that under small perturbations in a well conditioned matrix A, the errors in the reduced matrix S are small.

However, there is no full roundoff error analysis for general nonsingular matrices or for the whole algorithm (not only for separate stages of the algorithm). In this paper we present such an analysis. The main features of our analysis are that it is componentwise and that the exact expressions for the forward and backward errors are bounded, not just their first order approximation, with respect to the machine precision. The linear terms are given separately, to show that the terms of higher order are small enough.

The matrix inequalities are understood to be componentwise throughout the paper.

The outline of the paper is as follows. Section 2 presents backward and forward error analysis of the partitioning algorithm. In the next section we study some special classes of matrices. Finally, there are some numerical experiments in section 4.

2. Error analysis. Here and in the next sections we shall present the main results of this work. In the following, we denote the computed quantities by a hat. By δT we denote the error of the computation of an arbitrary matrix T, i.e., $\hat{T} = T + \delta T$. By ΔT we denote an equivalent perturbation in matrix T (see [9, p. 341]).

2.1. Previous results on the Gaussian elimination for tridiagonal systems. If $\hat{x}$ is the computed solution to a tridiagonal system $Ax = d$ obtained by Gaussian elimination, what is the "best" bound available for the error $x - \hat{x}$. This question is answered in [11] using backward error analysis, perturbation theory, and properties of the LU factorization of A. In [11] it is shown that if the LU factorization is used to solve a system $Ax = d$ by forward and back substitution, then it follows that the computed solution $\hat{x}$ satisfies

$$(A + \Delta A)\hat{x} = d, \quad |\Delta A| \leq f(\rho_0)|\hat{L}||\hat{U}|, \quad f(\rho_0) = 4\rho_0 + 3\rho_0^2 + \rho_0^3,$$

where ρ_0 is the roundoff unit. In the present paper we assume that $|\hat{L}||\hat{U}| \leq K|A|$, where K is a bound for the growth of elements. Then we obtain

$$|\Delta A| \leq K|A|f(\rho_0). \tag{5}$$

Let us note that the LU factorization of a given matrix A and bounds such as (5) do not always exist. In case they exist, the bounds can be large even for well-conditioned systems.

2.2. Analysis of the partitioning algorithm. The separate stages of the algorithm are analyzed in the following three lemmas, and the analysis for the whole algorithm is given in Theorem 2.4.

LEMMA 2.1. *If $L\hat{U} = \mathcal{A} + E$, then*

$$|E| \leq K_1|\mathcal{A}||N|f(\rho_0), \quad K_1 = \max\{k_1, 1\},$$

where k_1 is a bound for the growth of elements at Stage 1, and $N = \left(\begin{smallmatrix} 0 & \hat{R} \\ 0 & I_{s-1} \end{smallmatrix}\right)$.

Proof. Let us consider the matrix multiplication

$$L\hat{U} = \begin{pmatrix} A_{11} & 0 \\ A_{21} & I_{s-1} \end{pmatrix} \begin{pmatrix} I_{s(k-1)} & R + \delta R \\ 0 & \tilde{S} + \delta S \end{pmatrix},$$

where $\tilde{S} = A_{22} - A_{21}\hat{R}$. Here we denote by a tilde the exactly computed matrix from the data $A_{22}, A_{21}, \hat{R}$. Matrix L is without a hat because it consists of input entries only, and there is no computation involved. As a result of this multiplication we obtain

$$L\hat{U} = \begin{pmatrix} A_{11} & A_{12} + A_{11}\delta R \\ A_{21} & A_{21}\hat{R} + \tilde{S} + \delta S \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} + A_{11}\delta R \\ A_{21} & A_{22} + \delta S \end{pmatrix} = \mathcal{A} + E,$$

where the backward error from the calculation of the block LU factorization is denoted by

$$E = \begin{pmatrix} 0 & A_{11}\delta R \\ 0 & \delta S \end{pmatrix}. \tag{6}$$

Now we obtain bounds on the components of matrix E. Let us start with the entry $A_{11}\delta R$, where δR is a matrix. For simplicity, we shall bound a term of the following kind, $A_{11}\delta y$, where δy is the forward error when computing the solution y of the system $A_{11}y = z$, and z is a vector. The bound for $A_{11}\delta R$ is obtained in a way similar to the bound of $A_{11}\delta y$. From backward analysis (e.g., see [25]) we have

$$(A_{11} + \Delta A_{11})\hat{y} = (A_{11} + \Delta A_{11})(y + \delta y) = z. \tag{7}$$

After some manipulation from (7) we get

$$A_{11}\delta y = -\Delta A_{11}\hat{y}, \tag{8}$$

and, hence,

$$|A_{11}\delta y| \le |\Delta A_{11}||\hat{y}|. \tag{9}$$

Using the results obtained by Higham in [11] for the backward analysis of the Gaussian elimination for tridiagonal systems and (5), we have that $|\Delta A_{11}| \le k_1|A_{11}|f(\rho_0)$, where k_1 is a bound for the growth of elements in the Gaussian elimination at Stage 1. If we substitute this bound in (9) it follows that

$$|A_{11}\delta y| \le k_1|A_{11}||\hat{y}|f(\rho_0). \tag{10}$$

In our case we can substitute y by R and z by A_{12}. If we apply (10) for each column of the matrix R we obtain an analogous estimate

$$|A_{11}\delta R| \le k_1|A_{11}||\hat{R}|f(\rho_0). \tag{11}$$

Let us bound the term δS now. For the computation of the elements v_i of matrix S (see (4)), simple roundoff errors analysis (see [12, p. 89]) gives

$$\begin{aligned}\tilde{v_i} &= \{[b_{ik} - a_{ik}p_{ik-1}(1+\sigma_1)]\,(1+\sigma_2) - q_{ik+1}c_{ik}(1+\sigma_3)\}\,(1+\sigma_4)\\ &= b_{ik} - a_{ik}p_{ik-1} - q_{ik+1}c_{ik} + b_{ik}(\sigma_2+\sigma_4+\sigma_1\sigma_4)\\ &- a_{ik}p_{ik-1}(\sigma_1+\sigma_2+\sigma_4+\sigma_1\sigma_2+\sigma_1\sigma_4+\sigma_2\sigma_4+\sigma_1\sigma_2\sigma_4)\\ &- q_{ik+1}c_{ik}(\sigma_3+\sigma_4+\sigma_3\sigma_4),\end{aligned}$$

where $|\sigma_i| \le \rho_0, i = 1, \ldots, 4$. Hence,

$$\begin{aligned}|\delta v_i| &\le |b_{ik}|(2\rho_0+\rho_0^2) + |a_{ik}p_{ik-1}|(3\rho_0+3\rho_0^2+\rho_0^3) + |q_{ik+1}c_{ik}|(2\rho_0+\rho_0^2)\\ &\le (|b_{ik}| + |a_{ik}p_{ik-1}| + |q_{ik+1}c_{ik}|)(3\rho_0+3\rho_0^2+\rho_0^3).\end{aligned}$$

Analogous bounds are true for the other elements

$$|\delta u_i| \le |a_{ik}q_{ik-1}|\rho_0, \qquad |\delta w_i| \le |c_{ik}p_{ik+1}|\rho_0.$$

These are the errors from just one multiplication. In this way we obtain a bound for δS of the following kind:

$$|\delta S| \le (|A_{22}| + |A_{21}||\hat{R}|)g(\rho_0), \tag{12}$$

where $g(\rho_0) = 3\rho_0 + 3\rho_0^2 + \rho_0^3$. It is obvious that $g(\rho_0) \le f(\rho_0)$. Then from (11) and (12) we have

$$\begin{aligned}|E| &\le \begin{pmatrix} 0 & k_1|A_{11}||\hat{R}|f(\rho_0) \\ 0 & (|A_{22}| + |A_{21}||\hat{R}|)g(\rho_0) \end{pmatrix}\\ &\le K_1 \begin{pmatrix} |A_{11}| & 0 \\ |A_{21}| & |A_{22}| \end{pmatrix}\begin{pmatrix} 0 & |\hat{R}| \\ 0 & I_{s-1} \end{pmatrix} f(\rho_0)\\ &\le K_1|\mathcal{A}|\begin{pmatrix} 0 & |\hat{R}| \\ 0 & I_{s-1} \end{pmatrix} f(\rho_0)\\ &= K_1|\mathcal{A}||N|f(\rho_0),\end{aligned}$$

where $K_1 = \max\{k_1, 1\}$ and $N = \left(\begin{smallmatrix} 0 & \hat{R} \\ 0 & I_{s-1} \end{smallmatrix}\right)$. □

Next we analyze the solution of the system with the two block triangular matrices L and $\hat{U}$.

LEMMA 2.2. *If* $(L + \Delta L)\hat{y} = d$, *then* $|\Delta L| \leq K_1 |L| f(\rho_0)$.

Proof. When we solve a block triangular linear system with the matrix L for the equivalent perturbation ΔL, we have

$$|\Delta L| = \begin{pmatrix} |\Delta A_{11}| & 0 \\ |\Delta A_{21}| & |\Delta I_{s-1}| \end{pmatrix},$$

where ΔA_{11} is the perturbation coming from the Gaussian elimination at Stage 1 (solution of tridiagonal systems), and $\Delta A_{21}, \Delta I$ are the perturbations coming from the elimination of A_{21} (which is evidently equivalent to a solution of a unit triangular system). For the block diagonal matrix A_{11} with tridiagonal blocks from (5), again we have

$$|\Delta A_{11}| \leq k_1 |A_{11}| f(\rho_0). \tag{13}$$

The roundoff error analysis for the solution of a triangular system is well known (e.g., [12, p. 152]). In our case (with only two nonzero elements in each row of A_{21}) this analysis simplifies to

$$|\Delta A_{21}| \leq (2\rho_0 + \rho_0^2)|A_{21}|, \tag{14}$$

$$|\Delta I_{s-1}| \leq 2\rho_0 I_{s-1}. \tag{15}$$

Finally from (13), (14), and (15) we have

$$|\Delta L| \leq K_1 |L| f(\rho_0). \qquad \square$$

LEMMA 2.3. *If* $(\hat{U} + \Delta\hat{U})\hat{x} = \hat{y}$, *then*

$$|\Delta\hat{U}| \leq K_2 |\hat{U}| f(\rho_0), \quad K_2 = \max\{k_2, 1\},$$

where k_2 *is a bound for the growth of elements for the Gaussian elimination at Stage* 3 *(the solution of the reduced system with matrix* S*).*

Proof. The proof is similar to the proof of Lemma 2.2. $\square$

Now we combine the analysis of the separate stages to get the analysis for the whole algorithm. Before presenting the main result of this section, we need some notations.

Let us consider the jth component of the product $|N||\mathcal{P}\hat{x}|, \quad j = 1, \ldots, s(k-1)$. For this component it is true that

$$\begin{aligned}(|N||\mathcal{P}\hat{x}|)_j &\leq |\hat{R}_{j,i}||\hat{x}_{ik}| + |\hat{R}_{j,i+1}||\hat{x}_{(i+1)k}| \\ &\leq \|\hat{R}\|_\infty \max\left\{|\hat{x}_{ik}|, |\hat{x}_{(i+1)k}|\right\},\end{aligned}$$

for some $i = 1, \ldots, s-2$, because there are only two nonzero elements in each row of matrix R. Let us define the vector $\mathcal{P}x^*$ as follows:

$$\mathcal{P}x^* = [(x_1^*)^T \; (x_2^*)^T]^T,$$

where

$$x_1^* = (|\hat{x}_k| e, \max\{|\hat{x}_k|, |\hat{x}_{2k}|\} e, \ldots, \max\left\{|\hat{x}_{(s-2)k}|, |\hat{x}_{(s-1)k}|\right\} e)^T,$$

$$x_2^* = (|\hat{x}_k|, \ldots, |\hat{x}_{(s-1)k}|)^T,$$

and $e = (1, 1, \ldots, 1) \in \mathcal{R}^{1\times(k-1)}$. The vector x^* is clearly defined by

$$x^* = (|\hat{x}_k|e, |\hat{x}_k|, \max\{|\hat{x}_k|, |\hat{x}_{2k}|\}\, e, \ldots, |\hat{x}_{(s-1)k}|, \max\left\{|\hat{x}_{(s-2)k}|, |\hat{x}_{(s-1)k}|\right\} e)^T.$$

Then we have

$$|N||\mathcal{P}\hat{x}| \le \begin{pmatrix} \|R\|_\infty x_1^* \\ x_2^* \end{pmatrix} \le r\mathcal{P}x^*, \tag{16}$$

where $r = \max\{\|\hat{R}\|_\infty, 1\}$. Let us introduce the following condition number:

$$cond^*(\mathcal{A}, \mathcal{P}x^*) = \frac{\||\mathcal{A}^{-1}||\mathcal{A}|\mathcal{P}x^*\|_\infty}{\|\mathcal{P}\hat{x}\|_\infty}.$$

We would like to note that

$$cond^*(\mathcal{A}, x^*) \le cond(\mathcal{A}),$$

where $cond(\mathcal{A}) = \||\mathcal{A}^{-1}||\mathcal{A}|\|_\infty$.

Regarding the involvement of the original matrix A in the final bounds, we point out that

$$\begin{aligned} cond(\mathcal{A}) &= \||\mathcal{A}^{-1}||\mathcal{A}|\|_\infty = \|\mathcal{P}|A^{-1}|\mathcal{P}^T\mathcal{P}|A|\mathcal{P}^T\|_\infty \\ &= \|\mathcal{P}|A^{-1}||A|\mathcal{P}^T\|_\infty = \||A^{-1}||A|\|_\infty = cond(A), \end{aligned} \tag{17}$$

and also

$$cond^*(\mathcal{A}, \mathcal{P}x^*) = cond^*(A, x^*), \tag{18}$$

in a similar way.

We will also use Skeel's condition number [18]

$$cond(\mathcal{A}, \mathcal{P}\hat{x}) = \frac{\||\mathcal{A}^{-1}||\mathcal{A}||\mathcal{P}\hat{x}|\|_\infty}{\|\hat{x}\|_\infty},$$

but with the computed solution $\hat{x}$ instead of x. This fact allows us to obtain more precise estimates in the following theorem and to find computable bounds for the forward error. In a similar way to (17) and (18), we can get that

$$cond(\mathcal{A}, \mathcal{P}\hat{x}) = cond(A, \hat{x}). \tag{19}$$

The condition number $cond^*(A, x^*)$ is introduced to make the obtained bounds more realistic in some cases. As we shall see in the bounds of the forward error, the condition number $cond^*(A, x^*)$ is multiplied by the factor r (which can be large sometimes) while the condition number $cond(A, \hat{x})$ is not. So, when $cond^*(A, x^*)$ is small the influence of r should be negligible. An example of such a case is presented in section 5, which shows that our bounds are tight.

Theorem 2.4. *For the partitioning algorithm we have that* $(\mathcal{A} + \Delta\mathcal{A})\mathcal{P}\hat{x} = \mathcal{P}d$, *where*

$$|\Delta\mathcal{A}| \le |\mathcal{A}|\left[(K_1 + K_2)f(\rho_0) + h_1(\rho_0)\right] + |\mathcal{A}||N|\left[(3K_1 + 2K_2)f(\rho_0) + h_2(\rho_0)\right],$$

where

$$\begin{aligned} h_1(\rho_0) &= (K_1 + K_2)f(\rho_0)g(\rho_0) + K_1K_2f^2(\rho_0) + K_1K_2f^2(\rho_0)g(\rho_0), \\ h_2(\rho_0) &= (K_1 + K_2)f(\rho_0)g(\rho_0) + 2K_1K_2f^2(\rho_0) + K_1K_2f^2(\rho_0)g(\rho_0), \end{aligned}$$

are the terms of higher order in ρ_0, *and*

$$\begin{aligned}\frac{\|\delta x\|_\infty}{\|\hat{x}\|_\infty} = \frac{\|\hat{x}-x\|_\infty}{\|\hat{x}\|_\infty} &\\ &\leq cond(A,\hat{x})\left[(K_1+K_2)f(\rho_0)+h_1(\rho_0)\right] \\ &\quad + cond^*(A,x^*)r\left[(3K_1+2K_2)f(\rho_0)+h_2(\rho_0)\right].\end{aligned}$$

Proof. For the computed solution we have

$$(L+\Delta L)(\hat{U}+\Delta\hat{U})\mathcal{P}\hat{x} = \mathcal{P}d;$$

then

$$(L\hat{U}+\Delta L\hat{U}+L\Delta\hat{U}+\Delta L\Delta\hat{U})\mathcal{P}\hat{x} = \mathcal{P}d,$$

and from the fact that $L\hat{U} = \mathcal{A}+E$, we get

$$(\mathcal{A}+E+\Delta L\hat{U}+L\Delta\hat{U}+\Delta L\Delta\hat{U})\mathcal{P}\hat{x} = \mathcal{P}d.$$

Hence, we obtain $(\mathcal{A}+\Delta\mathcal{A})\mathcal{P}\hat{x} = \mathcal{P}d$, where

$$|\Delta\mathcal{A}| \leq |E|+|\Delta L||\hat{U}|+|L||\Delta\hat{U}|+|\Delta L||\Delta\hat{U}|. \tag{20}$$

From Lemmas 2.1–2.3 and equation (20) we get

$$\begin{aligned}|\Delta\mathcal{A}| &\leq K_1|\mathcal{A}||N|f(\rho_0)+K_1|L||\hat{U}|f(\rho_0) \\ &\quad +K_2|L||\hat{U}|f(\rho_0)+K_1K_2|L||\hat{U}|f^2(\rho_0) \\ &= K_1|\mathcal{A}||N|f(\rho_0)+(K_1f(\rho_0)+K_2f(\rho_0)+K_1K_2f^2(\rho_0))|L||\hat{U}|, \quad (21)\end{aligned}$$

but

$$\begin{aligned}|L||\hat{U}| &= \begin{pmatrix} |A_{11}| & 0 \\ |A_{21}| & I_{s-1}\end{pmatrix}\begin{pmatrix} I_{s(k-1)} & |\hat{R}| \\ 0 & |\hat{S}|\end{pmatrix} \\ &= \begin{pmatrix} |A_{11}| & |A_{11}||\hat{R}| \\ |A_{21}| & |A_{21}||\hat{R}|+|\hat{S}|\end{pmatrix} \\ &\leq \begin{pmatrix} |A_{11}| & |A_{11}||\hat{R}|+|A_{12}| \\ |A_{21}| & |A_{22}|+2|A_{21}||\hat{R}|+|\delta S|\end{pmatrix},\end{aligned}$$

where we have used the fact that $\hat{S} = A_{22}-A_{21}\hat{R}+\delta S$. Now from (12) we have

$$\begin{aligned}|L||\hat{U}| &\leq \begin{pmatrix} |A_{11}| & |A_{11}||\hat{R}|+|A_{12}| \\ |A_{21}| & |A_{22}|(1+g(\rho_0))+|A_{21}||\hat{R}|(2+g(\rho_0))\end{pmatrix} \\ &\leq |\mathcal{A}|(1+g(\rho_0))+\begin{pmatrix} 0 & |A_{11}||\hat{R}| \\ 0 & |A_{21}||\hat{R}|\end{pmatrix}(2+g(\rho_0)) \\ &\leq |\mathcal{A}|(1+g(\rho_0))+|\mathcal{A}|\begin{pmatrix} 0 & |\hat{R}| \\ 0 & I_{s-1}\end{pmatrix}(2+g(\rho_0)) \\ &= |\mathcal{A}|(1+g(\rho_0))+|\mathcal{A}||N|(2+g(\rho_0)). \quad (22)\end{aligned}$$

From (21) and (22) we obtain a bound of the equivalent perturbation $\Delta\mathcal{A}$ for the whole algorithm:

$$\begin{aligned} |\Delta\mathcal{A}| \le\ & K_1|\mathcal{A}||N|f(\rho_0) \\ & +(K_1 f(\rho_0)+K_2 f(\rho_0)+K_1K_2f^2(\rho_0)) \\ & [|\mathcal{A}|(1+g(\rho_0))+|\mathcal{A}||N|(2+g(\rho_0))] \\ =\ & |\mathcal{A}|\left[(K_1+K_2)f(\rho_0)+h_1(\rho_0)\right] \\ & +|\mathcal{A}||N|\left[(3K_1+2K_2)f(\rho_0)+h_2(\rho_0)\right], \end{aligned} \tag{23}$$

where

$$\begin{aligned} h_1(\rho_0) &= (K_1+K_2)f(\rho_0)g(\rho_0)+K_1K_2f^2(\rho_0)+K_1K_2f^2(\rho_0)g(\rho_0), \\ h_2(\rho_0) &= (K_1+K_2)f(\rho_0)g(\rho_0)+2K_1K_2f^2(\rho_0)+K_1K_2f^2(\rho_0)g(\rho_0). \end{aligned}$$

We have separated the terms of higher order in ρ_0 in the h_1 and h_2 functions. Evidently, h_1 and h_2 are small when there is no significant growth of K_1 and K_2. From the equality $(\mathcal{A}+\Delta\mathcal{A})\mathcal{P}\hat{x}=\mathcal{P}d$ we easily get the expression for the forward error $\mathcal{P}(\hat{x}-x)$:

$$\mathcal{P}(\hat{x}-x) = -\mathcal{A}^{-1}\Delta\mathcal{A}\mathcal{P}\hat{x}. \tag{24}$$

Then from (23) and (24) we get

$$\begin{aligned} |\mathcal{P}(\hat{x}-x)| \le\ & |\mathcal{A}^{-1}||\mathcal{A}||\mathcal{P}\hat{x}|\left[(K_1+K_2)f(\rho_0)+h_1(\rho_0)\right] \\ & + |\mathcal{A}^{-1}||\mathcal{A}||N||\mathcal{P}\hat{x}|\left[(3K_1+2K_2)f(\rho_0)+h_2(\rho_0)\right]. \end{aligned} \tag{25}$$

From (16) and (25) we have

$$\begin{aligned} \frac{\|\delta x\|_\infty}{\|\hat{x}\|_\infty} = \frac{\|\mathcal{P}(\hat{x}-x)\|_\infty}{\|\mathcal{P}\hat{x}\|_\infty} & \le \big\{ \||\mathcal{A}^{-1}||\mathcal{A}||\mathcal{P}\hat{x}|\|_\infty \left[(K_1+K_2)f(\rho_0)+h_1(\rho_0)\right] \\ & + \||\mathcal{A}^{-1}||\mathcal{A}|\mathcal{P}x^*\|_\infty r\left[(3K_1+2K_2)f(\rho_0)+h_2(\rho_0)\right]\big\} / \|\hat{x}\|_\infty \\ & = \mathit{cond}(\mathcal{A},\mathcal{P}\hat{x})\left[(K_1+K_2)f(\rho_0)+h_1(\rho_0)\right] \\ & + \mathit{cond}^*(\mathcal{A},\mathcal{P}x^*) r\left[(3K_1+2K_2)f(\rho_0)+h_2(\rho_0)\right], \end{aligned}$$

and the result follows from (18) and (19). □

Bounds on k_1 (respectively, K_1) for special classes of matrices are given in [11, 12]. Analogous bounds for k_2 (respectively, K_2) would be valid if matrix S (which is tridiagonal again) preserves the special properties of the original matrix. This topic and the bounding of $\|\hat{R}\|_\infty$ are discussed in the next section.

Let us also note that the bounds in the next section are some upper bounds for $\|\hat{R}\|_\infty$, and that it is not difficult to compute $\|\hat{R}\|_\infty$ in a relatively small number of steps. More precisely, we need $k-1$ parallel additions and $k+s$ parallel logical tests. This clearly does not increase the total computational time significantly because we need $17k+8s-41$ parallel steps (without taking into account the communication costs) for the whole algorithm.

3. Special classes of matrices. In this section we consider more precisely the case when the matrix A belongs to one of the following types: diagonally dominant, s.p.d., M-matrix, or totally nonnegative. We would like to mention that matrix $A \in \mathcal{R}^{n\times n}$ is a nonsingular M-matrix (see [5, p. 137]) if $a_{ij} \le 0$ for all $i \ne j$ and $A^{-1} \ge 0$. The matrix A is totally nonnegative [5, p. 57] if all its minors of any order are nonnegative. A well-known equivalent condition [21] for a nonsingular matrix A is that the inverse of a totally nonnegative matrix is a sign regular matrix, i.e.,

1. $a_{ij}^{(-1)} = (-1)^{i+j}\bar{a}_{ij}$, where $a_{ij}^{(-1)}$ are the entries of the inverse A^{-1}, and $\bar{a}_{ij} \geq 0$;
2. the matrix $\bar{A} = \{\bar{a}_{ij}\}_{i,j=1}^{n} = |A^{-1}|$ is totally nonnegative.

For diagonally dominant matrices A we assume row diagonal dominance in the sense that $|a_i| + |c_i| \leq |b_i|, \quad i = 1, 2, \ldots, n$. The definition of an s.p.d. matrix is well known (e.g., see [13, p. 250]), and we omit it here.

It is not difficult to see that the permuted matrix $\mathcal{A}$ is s.p.d., diagonally dominant, and an M-matrix, if the original matrix A is of that type. However, this is not the case with totally nonnegative matrices. The permutations may change the sign of some of the minors in such way that $\mathcal{A}$ is no longer totally nonnegative. More precisely, we will be interested not in the total nonnegativity of $\mathcal{A}$ but in the sign regularity of a part of $\mathcal{A}^{-1}$. As we will see later in this section, this is enough for our purposes.

For the following bounds of $\|\hat{R}\|_\infty$ and k_2, we need to analyze what is the type of the reduced matrix S if matrix A belongs to one of the above mentioned classes. First we analyze the type of S in exact arithmetic because we need this to bound $\|\hat{R}\|_\infty$. Then at the end of this section we consider the roundoff error implementation and comment on the growth of the constant k_2.

The next theorem is true not only for tridiagonal but for general dense matrices.

THEOREM 3.1. *Let $A \in \mathcal{R}^{n\times n}$. If matrix A is either*

- *symmetric positive definite, or*
- *a nonsingular M-matrix,*

then the reduced matrix S (the Schur complement) preserves the same property.

Proof. These properties are proved in [4, p. 94] and [4, p. 209], respectively. □

The case where A is row diagonally dominant has been presented in [3]. Here we give an alternative proof—the larger part of which will be used in one of the following theorems, where we find bounds on $\|\hat{R}\|_\infty$.

THEOREM 3.2. *Let $A \in \mathcal{R}^{n\times n}$ be nonsingular and tridiagonal. If matrix A is row diagonally dominant, then the reduced matrix S (the Schur complement) preserves the same property.*

Proof. As A is a row diagonally dominant matrix, we have

$$|a_i| + |c_i| \leq |b_i|, \tag{26}$$

for each $i = 1, 2, \ldots, n$. Let us consider the matrix $\mathcal{B}_i = (B_i, \bar{a}_i, \bar{c}_i)$ and prove that $\mathcal{B}_i$ preserves the property of row diagonal dominance when the Gaussian elimination is applied to invert matrix B_i. After the forward substitution, we get the following matrix:

$$\mathcal{B}_i^{(1)} = (B_i^{(1)}, \bar{a}_i^{(1)}, \bar{c}_i^{(1)}),$$

where $\bar{a}_i^{(1)} = (a_{i,1}^{(1)}, a_{i,2}^{(1)}, \ldots, a_{i,k-1}^{(1)})^T$, $\bar{c}_i^{(1)} = (0, 0, \ldots, 0, c_{i,k-1}^{(1)})^T$, and

$$B_i^{(1)} = \begin{pmatrix} 1 & c_{(i-1)k+1}^{(1)} & & & \\ & 1 & c_{(i-1)k+2}^{(1)} & & \\ & & \ddots & \ddots & \\ & & & 1 & c_{ik-2}^{(1)} \\ & & & & 1 \end{pmatrix}.$$

Here the new elements are obtained as follows:

$$
\begin{aligned}
a_{i,1}^{(1)} &= a_{i,1}/b_{(i-1)k+1},\\
c_{(i-1)k+1}^{(1)} &= c_{(i-1)k+1}/b_{(i-1)k+1},\\
b_{(i-1)k+j+1}^{(1)} &= b_{(i-1)k+j+1} - a_{(i-1)k+j+1}c_{(i-1)k+j}^{(1)}, && (27)\\
a_{i,j+1}^{(1)} &= -a_{(i-1)k+j+1}a_{i,j}^{(1)}/b_{(i-1)k+j+1}^{(1)}, && (28)\\
c_{(i-1)k+j+1}^{(1)} &= c_{(i-1)k+j+1}/b_{(i-1)k+j+1}^{(1)}, && (29)\\
j &= 1, 2, \dots, k-2.
\end{aligned}
$$

We consider the LU factorization with a unit diagonal in the U-factor for the sake of convenience. The proof is similar to the case where the L-factor has a unit diagonal.

We show now by induction on j that the diagonal dominance is preserved after the forward elimination. For the first row of matrix $\mathcal{B}_i$, the statement is evident because this is a scaled row of the original matrix A. Let us assume that

$$|a_{i,j}^{(1)}| + |c_{(i-1)k+j}^{(1)}| \le 1. \tag{30}$$

We will prove that $|a_{i,j+1}^{(1)}| + |c_{(i-1)k+j+1}^{(1)}| \le 1$. From (26)–(30) we have that

$$
\begin{aligned}
1 &= \frac{|b_{(i-1)k+j+1} - a_{(i-1)k+j+1}c_{(i-1)k+j}^{(1)}|}{|b_{(i-1)k+j+1}^{(1)}|}\\
&\ge \frac{|b_{(i-1)k+j+1}| - |a_{(i-1)k+j+1}|(1 - |a_{i,j}^{(1)}|)}{|b_{(i-1)k+j+1}^{(1)}|}\\
&= \frac{|b_{(i-1)k+j+1}| - |a_{(i-1)k+j+1}| + |a_{(i-1)k+j+1}||a_{i,j}^{(1)}|}{|b_{(i-1)k+j+1}^{(1)}|}\\
&\ge \frac{|c_{(i-1)k+j+1}|}{|b_{(i-1)k+j+1}^{(1)}|} + \frac{|a_{(i-1)k+j+1}||a_{i,j}^{(1)}|}{|b_{(i-1)k+j+1}^{(1)}|}\\
&= |c_{(i-1)k+j+1}^{(1)}| + |a_{i,j+1}^{(1)}|.
\end{aligned}
$$

Next we show the same fact for the backward substitution. As a result of this phase we have

$$\mathcal{B}_i^{(2)} = (I_{k-1}, q^{(i)}, p^{(i)}),$$

where

$$p^{(i)} = (p_{(i-1)k+1}, p_{(i-1)k+2}, \dots, p_{ik-1})^T,\ q^{(i)} = (q_{(i-1)k+1}, q_{(i-1)k+2}, \dots, q_{ik-1})^T.$$

Here the new elements are obtained in the following way:

$$
\begin{aligned}
q_{(i-1)k+j-1} &= a_{i,j-1}^{(1)} - q_{(i-1)k+j}c_{i,j-1}^{(1)}, && (31)\\
p_{(i-1)k+j-1} &= -p_{(i-1)k+j}c_{i,j-1}^{(1)}, && (32)\\
j &= k-1, k-2, \dots, 2.
\end{aligned}
$$

We shall use induction on j again. For the last row of $\mathcal{B}_i^{(2)}$, the diagonal dominance was proved in the forward substitution phase. Let us assume that

$$|q_{(i-1)k+j}| + |p_{(i-1)k+j}| \leq 1. \tag{33}$$

Then from (31), (32), and (33) we have that

$$\begin{aligned} |q_{(i-1)k+j-1}| &\leq |a_{i,j-1}^{(1)}| + |q_{(i-1)k+j}||c_{i,j-1}^{(1)}| \\ &\leq |a_{i,j-1}^{(1)}| + (1 - |p_{(i-1)k+j}|)|c_{i,j-1}^{(1)}| \\ &= |a_{i,j-1}^{(1)}| + |c_{i,j-1}^{(1)}| - |p_{(i-1)k+j}||c_{i,j-1}^{(1)}| \\ &\leq |a_{i,j-1}^{(1)}| + |c_{i,j-1}^{(1)}| - |p_{(i-1)k+j-1}|. \end{aligned}$$

Hence,

$$|q_{(i-1)k+j-1}| + |p_{(i-1)k+j-1}| \leq |a_{i,j-1}^{(1)}| + |c_{i,j-1}^{(1)}| \leq 1. \tag{34}$$

In this way, we obtain that matrix $\mathcal{B}_i^{(2)}$ preserves the property of row diagonal dominance. From this fact and equation (4), for the entries of S we obtain

$$\begin{aligned} |v_i| &\geq |b_{ik}| - |a_{ik}||p_{ik-1}| - |c_{ik}||q_{ik+1}| \\ &\geq |b_{ik}| - |a_{ik}|(1 - |q_{ik-1}|) - |c_{ik}|(1 - |p_{ik+1}|) \\ &\geq |c_{ik}| + |a_{ik}||q_{ik-1}| - |c_{ik}| + |c_{ik}||p_{ik+1}| \\ &\geq |u_i| + |w_i|. \end{aligned}$$

Hence, the reduced matrix S is row diagonally dominant. □

Now let us consider the case where A is totally nonnegative. We present a proof for this case because we could not find it in the existing literature.

THEOREM 3.3. *Let $A \in \mathcal{R}^{n\times n}$ be nonsingular and tridiagonal. If matrix A is totally nonnegative, then the reduced matrix S (the Schur complement) is*

- *an M-matrix when the blocks B_i are of odd order, i.e., k is even,*
- *totally nonnegative when the blocks B_i are of even order, i.e., k is odd.*

Proof. As the original matrix A is totally nonnegative, it follows from (3) that A_{11}, A_{22} are matrices of the same type, A_{11}^{-1} is sign regular, and $A_{12} \geq 0, A_{21} \geq 0$. Let us find the inverse of $\mathcal{A}$. From (3) we obtain

$$\begin{aligned} \mathcal{A}^{-1} = U^{-1}L^{-1} &= \begin{pmatrix} I_{s(k-1)} & -RS^{-1} \\ 0 & S^{-1} \end{pmatrix} \begin{pmatrix} A_{11}^{-1} & 0 \\ -A_{21}A_{11}^{-1} & I_{s-1} \end{pmatrix} \\ &= \begin{pmatrix} A_{11}^{-1} + H & -RS^{-1} \\ -S^{-1}A_{21}A_{11}^{-1} & S^{-1} \end{pmatrix}, \end{aligned} \tag{35}$$

where

$$H = RS^{-1}A_{21}A_{11}^{-1} = A_{11}^{-1}A_{12}S^{-1}A_{21}A_{11}^{-1}. \tag{36}$$

As A is a totally nonnegative matrix, we have that A^{-1} is a sign regular matrix which is permuted in such a way that the lower right corner of matrix $\mathcal{A}^{-1}$ (see (35)) coincides with S^{-1} and contains the entries of A^{-1} with indices

$$\begin{pmatrix} k,k & k,2k & \dots & k,(s-1)k \\ 2k,k & 2k,2k & \dots & 2k,(s-1)k \\ \vdots & \vdots & \dots & \vdots \\ (s-1)k,k & (s-1)k,2k & \dots & (s-1)k,(s-1)k \end{pmatrix}.$$

Taking into account the change of signs $+, -, +, \dots,$ in each row and column of matrix A^{-1} it follows that when k is even, then the matrix S^{-1} consists of nonnegative elements only (these elements have the above mentioned indices in A^{-1}), i.e., $S^{-1} \geq 0$. It remains to show that the off-diagonal entries of S are nonpositive. Let us write the signs for the matrix product $A_{21}A_{11}^{-1}A_{12}$ for one diagonal block of A_{11}^{-1}, taking into consideration only the signs of the entries (each entry with a $+$ or $-$ sign can also be equal to zero):

$$\begin{aligned} &\begin{pmatrix} + & 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & + \end{pmatrix} \begin{pmatrix} + & - & + & \dots & + \\ - & + & - & \dots & - \\ + & - & + & \dots & + \\ \vdots & \vdots & \vdots & & \vdots \\ + & - & + & \dots & + \end{pmatrix} \begin{pmatrix} + & 0 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \\ 0 & + \end{pmatrix} \\ &= \begin{pmatrix} + & + \\ + & + \end{pmatrix}. \end{aligned} \tag{37}$$

From (37) it is clear that all of the off-diagonal elements of $A_{21}A_{11}^{-1}A_{12}$ are nonnegative. But $S = A_{22} - A_{21}A_{11}^{-1}A_{12}$, where A_{22} is a diagonal matrix, and hence all of the off-diagonal elements of S are nonpositive. From the definition of a nonsingular M-matrix in the beginning of this section, it follows that S is an M-matrix.

The other case is where k is odd. From this fact, taking into account the change of signs in A^{-1}, it is not difficult to see that the elements with the above mentioned indices, i.e., the entries of S^{-1}, have signs which change in the same way as in A^{-1}. It remains to note that $|S^{-1}|$ is the lower right block of $|\mathcal{A}^{-1}|$, and the permutation matrix $\mathcal{P}$ is such that all the minors of $|S^{-1}|$ are minors also in $|A^{-1}|$. So, $|S^{-1}|$ is totally nonnegative. Then, S^{-1} is a sign regular matrix, and S is a totally nonnegative matrix. $\square$

As we saw in Theorem 2.4, the error bound depends not only on the growth factors K_1 and K_2, but also on the quantity r, which measures the growth in the matrix $\hat{R}$. Clearly, when some of the blocks B_i are ill conditioned (although the whole matrix A is well conditioned) the factor r can be large. This will lead to large errors even for well-conditioned matrices. So, we need bounds for r, or, equivalently $\|\hat{R}\|_\infty$. In the following we show that $\|\hat{R}\|_\infty$ is bounded by not large constants for the above mentioned four classes of matrices. For the next theorems, we need the following lemma.

LEMMA 3.4. *For the computed matrix $\hat{R}$, we have*

$$\|\hat{R}\|_\infty \leq \frac{\|R\|_\infty}{1 - k_1 cond(A_{11}) f(\rho_0)},$$

if $k_1 cond(A_{11}) f(\rho_0) < 1$.

Proof. For the quantity $\|\hat{R}\|_\infty$, we clearly have

$$\|\hat{R}\|_\infty \leq \|R\|_\infty + \|\delta R\|_\infty. \tag{38}$$

The bound of $\|\delta R\|_\infty$ is not so evident, and we present it below. Let us introduce the following notations: $|A_{11}^{-1}||A_{11}| = M = \{m_{it}\}_{i,t=1}^{s(k-1)}$, $M_i = i$th row of M, $l = s(k-1)$. From (8) we get

$$\delta y = -A_{11}^{-1} \Delta A_{11} \hat{y},$$

and, hence,

$$|\delta y^{(j)}| \le k_1 |A_{11}^{-1}||A_{11}||\hat{y}^{(j)}| f(\rho_0), \tag{39}$$

where $\hat{y}^{(j)}$ is the jth column of matrix $\hat{R}$. From (39) we have

$$\begin{aligned}
\|\delta R\|_\infty &= \max_{1\le i\le l} \sum_{j=1}^{s-1} |\delta y_i^{(j)}| \\
&\le \max_{1\le i\le l} \sum_{j=1}^{s-1} k_1 M_i |\hat{y}^{(j)}| f(\rho_0) \\
&= k_1 \max_{1\le i\le l} \Big[m_{i1}|\hat{y}_1^{(1)}| + \cdots + m_{il}|\hat{y}_l^{(1)}| + \cdots \\
&\quad + m_{i1}|\hat{y}_1^{(s-1)}| + \cdots + m_{il}|\hat{y}_l^{(s-1)}| \Big] f(\rho_0) \\
&= k_1 \max_{1\le i\le l} \left[m_{i1} \sum_{j=1}^{s-1} |\hat{y}_1^{(j)}| + m_{i2} \sum_{j=1}^{s-1} |\hat{y}_2^{(j)}| + \cdots + m_{il} \sum_{j=1}^{s-1} |\hat{y}_l^{(j)}| \right] f(\rho_0). \qquad (40)
\end{aligned}$$

It is evident that $\sum_{j=1}^{s-1} |\hat{y}_i^{(j)}| \le \|\hat{R}\|_\infty$, for each i. From this fact and from (40) we get

$$\begin{aligned}
\|\delta R\|_\infty &\le k_1 \|\hat{R}\|_\infty \max_{1\le i\le l} \sum_{j=1}^{l} m_{ij} f(\rho_0) \\
&= k_1 cond(A_{11}) \|\hat{R}\|_\infty f(\rho_0). \qquad (41)
\end{aligned}$$

Now from (38) and (41) we get

$$\|\hat{R}\|_\infty \le \|R\|_\infty + k_1 cond(A_{11}) \|\hat{R}\|_\infty f(\rho_0).$$

Then we have

$$\|\hat{R}\|_\infty \le \frac{\|R\|_\infty}{1 - k_1 cond(A_{11}) f(\rho_0)},$$

where it is necessary to suppose that $k_1 cond(A_{11}) f(\rho_0) < 1$. □

Because of Lemma 3.4, in the next theorems we need to find bounds only for the quantity $\|R\|_\infty$ with the exact matrix R. From Lemma 3.4, we see that the bound of $\|\hat{R}\|_\infty$ depends on how large $cond(A_{11})$ is. Therefore, we also find bounds on this condition number, and show that A_{11} is better conditioned than A, for the four classes of matrices.

THEOREM 3.5. *Let $A \in \mathcal{R}^{n\times n}$ be nonsingular and tridiagonal. If any of the following two conditions hold*

(a) *A is totally nonnegative,*

(b) *A is an M-matrix,*

then

$$\|\hat{R}\|_\infty \le \frac{cond(A)}{1 - k_1 cond(A_{11}) f(\rho_0)} \le \frac{cond(A)}{1 - k_1 cond(A) f(\rho_0)},$$

if $k_1 cond(A) f(\rho_0) < 1$.

Proof. (a) Let A be a totally nonnegative matrix. Then we have that $A_{12} \geq 0, A_{21} \geq 0$, and A_{11}^{-1} is sign regular. From Theorem 3.3, we obtain that when the blocks B_i are of odd order, then the reduced matrix S is an M-matrix, i.e., $S^{-1} \geq 0$, and when the blocks B_i are of even order, then the reduced matrix S is a totally nonnegative matrix, i.e., S^{-1} is sign regular. We analyze now the block diagonal entries $B_i^{-1} + H_{ii}$ of matrix $A_{11}^{-1} + H$ because the block off-diagonal entries of this matrix coincide with the block off-diagonal entries of matrix H. We show that H_{ii} satisfy the first condition of the definition of a sign regular matrix. In this way we can bound $\|R\|_\infty$ easily. For this purpose it is necessary to consider two cases.

In the first case, B_i is of odd order. In this case, when computing H_{ii} from (36) we take into account only the signs of the entries (each entry with a + or − sign can be also equal to zero). Let us write the signs of one block diagonal entry of $A_{12}S^{-1}A_{21}$ in the same way as in the proof of Theorem 3.3:

$$\begin{pmatrix} + & 0 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \\ 0 & + \end{pmatrix} \begin{pmatrix} + & + \\ + & + \end{pmatrix} \begin{pmatrix} + & 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & + \end{pmatrix}$$

$$= \begin{pmatrix} + & 0 & \dots & 0 & + \\ 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \dots & 0 & 0 \\ + & 0 & \dots & 0 & + \end{pmatrix}. \tag{42}$$

Now from (36) and (42) we have

$$\begin{aligned} H_{ii} &= \begin{pmatrix} + & - & + & \dots & + \\ - & + & - & \dots & - \\ + & - & + & \dots & + \\ \vdots & \vdots & \vdots & & \vdots \\ + & - & + & \dots & + \end{pmatrix} \begin{pmatrix} + & 0 & \dots & 0 & + \\ 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \dots & 0 & 0 \\ + & 0 & \dots & 0 & + \end{pmatrix} \begin{pmatrix} + & - & + & \dots & + \\ - & + & - & \dots & - \\ + & - & + & \dots & + \\ \vdots & \vdots & \vdots & & \vdots \\ + & - & + & \dots & + \end{pmatrix} \\ &= \begin{pmatrix} + & 0 & \dots & 0 & + \\ - & 0 & \dots & 0 & - \\ \vdots & \vdots & & \vdots & \vdots \\ - & 0 & \dots & 0 & - \\ + & 0 & \dots & 0 & + \end{pmatrix} \begin{pmatrix} + & - & + & \dots & + \\ - & + & - & \dots & - \\ + & - & + & \dots & + \\ \vdots & \vdots & \vdots & & \vdots \\ + & - & + & \dots & + \end{pmatrix} \\ &= \begin{pmatrix} + & - & + & \dots & + \\ - & + & - & \dots & - \\ + & - & + & \dots & + \\ \vdots & \vdots & \vdots & & \vdots \\ + & - & + & \dots & + \end{pmatrix}. \end{aligned}$$

Hence, we obtain that the blocks H_{ii} satisfy the first condition of the definition of a sign regular matrix.

In the second case B_i is of even order. Let us define the signs of one block diagonal entry of $A_{12}S^{-1}A_{21}$ in an analogous way:

$$\begin{pmatrix} + & 0 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \\ 0 & + \end{pmatrix} \begin{pmatrix} + & - \\ - & + \end{pmatrix} \begin{pmatrix} + & 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & + \end{pmatrix}$$

$$= \begin{pmatrix} + & 0 & \dots & 0 & - \\ 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \dots & 0 & 0 \\ - & 0 & \dots & 0 & + \end{pmatrix}. \tag{43}$$

Then from (36) and (43) we have

$$H_{ii} = \begin{pmatrix} + & - & + & \dots & - \\ - & + & - & \dots & + \\ + & - & + & \dots & - \\ \vdots & \vdots & \vdots & & \vdots \\ - & + & - & \dots & + \end{pmatrix} \begin{pmatrix} + & 0 & \dots & 0 & - \\ 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \dots & 0 & 0 \\ - & 0 & \dots & 0 & + \end{pmatrix} \begin{pmatrix} + & - & + & \dots & - \\ - & + & - & \dots & + \\ + & - & + & \dots & - \\ \vdots & \vdots & \vdots & & \vdots \\ - & + & - & \dots & + \end{pmatrix}$$

$$= \begin{pmatrix} + & 0 & \dots & 0 & - \\ - & 0 & \dots & 0 & + \\ \vdots & \vdots & & \vdots & \vdots \\ + & 0 & \dots & 0 & - \\ - & 0 & \dots & 0 & + \end{pmatrix} \begin{pmatrix} + & - & + & \dots & - \\ - & + & - & \dots & + \\ + & - & + & \dots & - \\ \vdots & \vdots & \vdots & & \vdots \\ - & + & - & \dots & + \end{pmatrix}$$

$$= \begin{pmatrix} + & - & + & \dots & - \\ - & + & - & \dots & + \\ + & - & + & \dots & - \\ \vdots & \vdots & \vdots & & \vdots \\ - & + & - & \dots & + \end{pmatrix}.$$

Hence, the blocks H_{ii} satisfy the first condition of the definition of a sign regular matrix again.

So, for both cases we get

$$|A_{11}^{-1} + H| = |A_{11}^{-1}| + |H|. \tag{44}$$

For the block diagonal entries, this fact follows because B_i^{-1} and H_{ii} have the same sign pattern, and for the block off-diagonal entries this is evident because these entries of A_{11}^{-1} are zero. Now, from (35) and (44) we obtain the following bound:

$$|\mathcal{A}^{-1}| = \begin{pmatrix} |B_{11}| & |B_{12}| \\ |B_{21}| & |B_{22}| \end{pmatrix} \geq \begin{pmatrix} |A_{11}^{-1} + H| & 0 \\ 0 & 0 \end{pmatrix} \geq \begin{pmatrix} |A_{11}^{-1}| & 0 \\ 0 & 0 \end{pmatrix}, \tag{45}$$

where the blocks B_{ij} correspond to the block partitioning defined in (3). From (45) it is clear that

$$|A_{11}^{-1}| \leq |B_{11}|, \tag{46}$$

and, hence,

$$\begin{aligned}\|R\|_\infty &\le \||A_{11}^{-1}||A_{12}|\|_\infty \le \||B_{11}||A_{12}|\|_\infty \le \||\mathcal{A}^{-1}||\mathcal{A}|\|_\infty\\ &= cond(\mathcal{A}) = cond(A),\end{aligned} \tag{47}$$

where we used equation (17) for the last equality. From Lemma 3.4 and (47) we obtain the first inequality of the theorem.

For the second inequality from (46) we have

$$|A_{11}^{-1}||A_{11}| \le |B_{11}||A_{11}|. \tag{48}$$

Taking the infinity norm in (48), we get

$$cond(A_{11}) \le \||B_{11}||A_{11}|\|_\infty \le cond(\mathcal{A}) = cond(A),$$

by (17) again.

(b) Suppose that A is an M-matrix. Then $A_{11}^{-1} \ge 0, \quad A_{12} \le 0, \quad A_{21} \le 0$. From Theorem 3.1 we obtain that S is also an M-matrix, i.e., $S^{-1} \ge 0$, and for the matrix H we get

$$H = A_{11}^{-1} A_{12} S^{-1} A_{21} A_{11}^{-1} \ge 0.$$

Then the equality (44) is valid again, and we prove part (b) in the same way. □

THEOREM 3.6. *Let $A \in \mathcal{R}^{n\times n}$ be nonsingular and tridiagonal. If matrix A is row diagonally dominant, then*

$$\|\hat{R}\|_\infty \le \frac{1}{1 - k_1 cond(A_{11}) f(\rho_0)} \le \frac{1}{1 - 2k_1 cond(A) f(\rho_0)}$$

if $2k_1 cond(A) f(\rho_0) < 1$.

Proof. It was proved in Theorem 3.2 that for each i, we have (see (34))

$$|p_i| + |q_i| \le 1.$$

As far as p_i and q_i are the only nonzero entries of R in the corresponding row, we have that

$$\|R\|_\infty \le 1. \tag{49}$$

The first inequality of the theorem follows now from Lemma 3.4.

Let us consider

$$|A_{11}^{-1}||A_{11}| = |A_{11}^{-1} + H - H||A_{11}| \le |A_{11}^{-1} + H||A_{11}| + |H||A_{11}|. \tag{50}$$

Let us take infinity norm for the first term in the right-hand side of (50):

$$\||A_{11}^{-1} + H||A_{11}|\|_\infty \le \||\mathcal{A}^{-1}||\mathcal{A}|\|_\infty = cond(\mathcal{A}) = cond(A), \tag{51}$$

as it was done in (47). For the second term in (50), we have

$$\begin{aligned}|H||A_{11}| &= |RS^{-1}A_{21}A_{11}^{-1}||A_{11}|\\ &\le |R||S^{-1}A_{21}A_{11}^{-1}||A_{11}| = |R||B_{21}||A_{11}|.\end{aligned} \tag{52}$$

Taking infinity norms in (52) from (49), we get

$$\| |H||A_{11}| \|_\infty \leq \| |B_{21}||A_{11}| \|_\infty \leq \| |\mathcal{A}^{-1}||\mathcal{A}| \|_\infty = cond(\mathcal{A}) = cond(A), \tag{53}$$

in a similar way. Then from (50), (51), and (53) we obtain

$$cond(A_{11}) \leq cond(A) + cond(A) = 2cond(A).$$

Thus the second inequality is proved as well. □

Remark. When A is a column diagonally dominant matrix, we have the following factorization:

$$\mathcal{A} = LU = \begin{pmatrix} I_{s(k-1)} & 0 \\ A_{21}A_{11}^{-1} & I_{s-1} \end{pmatrix} \begin{pmatrix} A_{11} & A_{12} \\ 0 & S \end{pmatrix}.$$

Then analogous bounds, as for row diagonally dominant matrices, are true and it is not difficult to propose a modification of the partitioning algorithm corresponding to this factorization. This modification needs the same number of time steps. We shall not discuss it here.

In the next theorem we use the fact that $\|\mathcal{A}^{-1}\|_2\|\mathcal{A}\|_2 = \|A^{-1}\|_2\|A\|_2$ which can be shown in a way similar to (17).

THEOREM 3.7. *Let $A \in \mathcal{R}^{n\times n}$ be tridiagonal. If A is a symmetric positive definite matrix, then*

$$\|\hat{R}\|_\infty \leq \frac{\sqrt{(s-1)cond_2(A)}}{1 - k_1 cond(A_{11})f(\rho_0)} \leq \frac{\sqrt{(s-1)cond_2(A)}}{1 - k_1(k-1)cond_2(A)f(\rho_0)},$$

if $k_1(k-1)cond_2(A)f(\rho_0) < 1$, where $cond_2(A) = \|A^{-1}\|_2\|A\|_2$.

Proof. Taking into account our Lemma 3.4, Lemma 10.12 from [12], and some simple norm relations we have

$$\begin{aligned}\|\hat{R}\|_\infty &\leq \frac{\|R\|_\infty}{1 - k_1 cond(A_{11})f(\rho_0)} \leq \frac{\sqrt{(s-1)}\|R\|_2}{1 - k_1 cond(A_{11})f(\rho_0)} \\ &= \frac{\sqrt{(s-1)}\|A_{11}^{-1}A_{12}\|_2}{1 - k_1 cond(A_{11})f(\rho_0)} \leq \frac{\sqrt{(s-1)cond_2(\mathcal{A})}}{1 - k_1 cond(A_{11})f(\rho_0)} \\ &= \frac{\sqrt{(s-1)cond_2(A)}}{1 - k_1 cond(A_{11})f(\rho_0)},\end{aligned}$$

which is exactly the first inequality of the statement.

The second inequality follows from simple norm relations:

$$\begin{aligned}cond(A_{11}) &= \| |A_{11}^{-1}||A_{11}| \|_\infty \\ &= \max_{1\leq i\leq s} \|B_i^{-1}\|_\infty \|B_i\|_\infty \\ &\leq \max_{1\leq i\leq s} \sqrt{k-1}\|B_i^{-1}\|_2 \sqrt{k-1}\|B_i\|_2 \\ &\leq (k-1)\|A_{11}^{-1}\|_2\|A_{11}\|_2 = (k-1)cond_2(A_{11}) \\ &\leq (k-1)cond_2(\mathcal{A}) = (k-1)cond_2(A). \qquad \square\end{aligned}$$

Theorems 3.5–3.7 show that $\|\hat{R}\|_\infty$ is bounded by not large constants for the four classes of matrices, if the whole matrix A is well conditioned. In order to bound

k_2 we can use Theorems 3.1–3.3 and the bounds already obtained for the Gaussian elimination in [11]. However, in practice we obtain the computed matrix $\hat{S}$ instead of the exact one. It is important to know what the distance is between S and $\hat{S}$. This question is answered in the following theorem.

THEOREM 3.8. *For the error $\Omega S = \hat{S} - S$ in the computed reduced matrix $\hat{S}$, it holds that*

$$\frac{\|\Omega S\|_\infty}{\|S\|_\infty} \le K_1 cond(A) r f(\rho_0).$$

Proof. For the computed reduced matrix $\hat{S}$, we have

$$\begin{aligned}\hat{S} &= A_{22} - A_{21}\hat{R} + \delta S = A_{22} - A_{21}(R + \delta R) + \delta S\\ &= A_{22} - A_{21}R - A_{21}\delta R + \delta S = S + \Omega S,\end{aligned}$$

where by $\Omega S = -A_{21}\delta R + \delta S$ we denote the total error in $\hat{S}$. From the proof of Lemma 2.1 (see (6)) we get

$$L^{-1}E = \begin{pmatrix} 0 & \delta R \\ 0 & \Omega S \end{pmatrix}. \tag{54}$$

On the other hand, we have

$$L^{-1}E = UU^{-1}L^{-1}E = U\mathcal{A}^{-1}E = \begin{pmatrix} I & R \\ 0 & S \end{pmatrix} \mathcal{A}^{-1}E. \tag{55}$$

From (54) and (55) for the block in the lower right corner, it follows that

$$\|\Omega S\|_\infty \le \|\mathcal{A}^{-1}E\|_\infty \|S\|_\infty. \tag{56}$$

From Lemma 2.1 we get

$$|\mathcal{A}^{-1}E| \le K_1 |\mathcal{A}^{-1}||\mathcal{A}||N| f(\rho_0),$$

and from the bound of $|N|$ in (16) we have

$$\|\mathcal{A}^{-1}E\|_\infty \le K_1 cond(\mathcal{A}) r f(\rho_0) = K_1 cond(A) r f(\rho_0). \tag{57}$$

Finally, from (56) and (57) we obtain

$$\frac{\|\Omega S\|_\infty}{\|S\|_\infty} \le K_1 cond(A) r f(\rho_0). \qquad \square$$

The theorems in this section show that $\|\hat{R}\|_\infty$ (and r, respectively) is not large for the four types of matrices when the original matrix A is well conditioned. So, the error in S is also bounded by a not large constant, if matrix A is well conditioned. Consequently, the constant k_2 is close to the theoretical constants from [11] (which are less than or equal to 3). For other types of matrices this conclusion may not be true, and the error ΩS may grow.

It is clear that when some of the blocks B_i are singular or ill conditioned, the algorithm can behave poorly. Iterative refinement [9, p. 126] is not of much help in general because it is convergent under certain restrictions on the conditioning of matrix B_i. We shall not discuss this problem in detail here. We just mention two

Table 1
The forward and backward error for the matrix A in Example 1, where $k = 6, s = 10$.

ε	1E–5	1E–10	1E–15
BE	8.73E–11	1.19E–5	0.42
FE	1.74E–10	2.38E–5	2.06

Table 2
The forward and backward error for the matrix A in Example 2, where $s = 10$.

k	6	56	256	556
BE	1.44E–15	1.11E–16	1.66E–16	1.14E–16
FE	3.33E–15	1.99E–15	1.31E–14	1.55E–15

references, [2] and [17], where two approaches to overcome this problem have been proposed. In [2] the authors do a QR-decomposition of each block B_i and introduce a new block partitioning, when necessary, in such a way that the new blocks are well conditioned. In [17] an experience with a perturbation approach is presented, and there is no limitation on the conditioning of B_i. Some theory for the latter approach and comparison of the two approaches will be given in a future work.

4. Numerical experiments.

The numerical experiments in this section are done in MATLAB, where the roundoff unit is $\rho_0 \approx 2.22\text{E}{-16}$. We measure two types of errors:

1. The relative forward error
$$FE = \frac{\|\hat{x} - x\|_\infty}{\|\hat{x}\|_\infty},$$
where $\hat{x}$ is the computed solution.
2. The componentwise backward error (see [12])
$$BE = \max_{1 \le i \le n} \frac{(|A\hat{x} - d|)_i}{(|A||\hat{x}| + |d|)_i}.$$

Let us consider the following examples.

Example 1. $A = \text{tridiag}(1, b, 1)$, where $b = (\varepsilon, \ldots, \varepsilon, 2)$. In this way A becomes very well conditioned. The exact solution is $x = (1, 1, \ldots, 1)^T$. We can notice how the backward and forward errors grow when $\varepsilon \to 0$, although the matrix A is very well conditioned and we use partial pivoting. This is because $\|\hat{R}\|_\infty$ grows infinitely when $\varepsilon \to 0$, a fact which is predicted by our theory. We report the results in Table 1 for different values of ε.

Example 2. A is the matrix from Example 1 with $\varepsilon = 1\text{E}{-16}$ (a number less than the roundoff unit). A is well conditioned again. The exact solution is
$$x = (1, \ldots, 1, 0; 1, \ldots, 1, 0; \ldots, 1, \ldots, 1, 0; 1, \ldots, 1)^T,$$
where $x_k = x_{2k} = \cdots = x_{(s-1)k} = 0$. We report the results of our example in Table 2, where $\varepsilon = 1\text{E}{-16}$, $s = 10$ for different values of k. This example shows why we have introduced the condition number $cond^*(A, x^*)$. Here we have a large r factor in Theorem 2.4 ($\approx 1E{+}16$) but, as can be seen from Table 2, the errors are very small. This is because $cond^*(A, x^*) \approx 0$ for this example, and the influence of r is not essential although the blocks B_i are almost singular. So, large r does not necessarily mean large errors as could be expected intuitively. The LU factorization of B_i is done

Table 3
The forward and backward error of Example 3, *where* $k = 5, s = 3, \varepsilon = .009$.

x	x_α	e	$rand$	$randn$
BE	4.67E–16	1.17E–16	1.59E–16	2.61E–16
FE	1.54E–9	2.21E–8	1.15E–7	6.95E–7
$cond(A, \hat{x})$	3.03E+7	8.99E+8	9.55E+8	4.72E+8
$cond^*(A, x^*)$	1.97E+8	8.99E+8	1.22E+9	1.71E+9
r	1	1	1	1

with partial pivoting again to make the constants k_1 and k_2 small. In this way we can see the importance of introducing the second condition number $cond^*(A, x^*)$.

Example 3. This is an example given in [11] (which was taken from [8]). The matrix is defined as follows:

$$a_i = \begin{cases} -\varepsilon/h^2, & 1 \le i \le m, \\ -\varepsilon/h^2 + (0.5 - ih)/h^2, & m+1 \le i \le n, \end{cases}$$

$$c_i = \begin{cases} -\varepsilon/h^2 - (0.5 - ih)/h^2, & 1 \le i \le m, \\ -\varepsilon/h^2, & m+1 \le i \le n, \end{cases}$$

and $b_i = -a_i - c_i, i = 1, \ldots, n$, where $m = \lfloor (n+1)/2 \rfloor, h = 1/(n+1), \varepsilon > 0$. Let us note that A is a nonsingular, row diagonally dominant M-matrix. The results we obtained are given in Table 3. The exact solutions here are chosen as $x_\alpha = (1, \alpha, \alpha^2, \ldots, 10^{-5})^T$, $\alpha = 10^{-5/(n-1)}$, $e = (1, 1, \ldots, 1)^T$, and "rand" and "randn" are exact solutions generated by the corresponding MATLAB functions. Again, as predicted by our theoretical results, the BE is small because A is row diagonally dominant and an M-matrix. The FE is larger because matrix A is not so well conditioned, as can be seen from Table 3. The forward error is almost equal to the theoretical bound from Theorem 2.4, which shows that our bounds cannot be improved essentially.

Acknowledgment. The authors are grateful to the anonymous referees for useful remarks and suggestions which improved the presentation of this paper.

REFERENCES

[1] P. Amodio and L. Brugnano, *Parallel factorizations and parallel solvers for tridiagonal linear systems*, Linear Algebra Appl., 172 (1992), pp. 347–364.

[2] P. Amodio and L. Brugnano, *The parallel QR factorization algorithm for tridiagonal linear systems*, Parallel Comput., 21 (1995), pp. 1097–1110.

[3] P. Amodio, L. Brugnano, and T. Politi, *Parallel factorizations for tridiagonal matrices*, SIAM J. Numer. Anal., 30 (1993), pp. 813–823.

[4] O. Axelsson, *Iterative Solution Methods*, Cambridge University Press, Cambridge, UK, 1994.

[5] A. Berman and R. J. Plemmons, *Nonnegative Matrices in the Mathematical Sciences*, SIAM, Philadelphia, PA, 1994.

[6] S. Bondeli, *Divide and Conquer: A New Parallel Algorithm for the Solution of a Tridiagonal Linear System of Equations*, Tech. Report 130, ETH Zürich, Department Informatik, Institut für Wissenschaftliches Rechnen, Zürich, Switzerland, 1990.

[7] L. Brugnano, *A parallel solver for tridiagonal linear systems for distributed memory parallel computers*, Parallel Comput., 17 (1991), pp. 1017–1023.

[8] F. Dorr, *An example of ill-conditioning in the numerical solution of singular perturbed problems*, Math. Comp., 25 (1971), pp. 271–283.

[9] G. Golub and C. van Loan, *Matrix Computations*, 3rd ed., The John Hopkins University Press, Baltimore, MD, 1996.

[10] I. Hajj and S. Skelboe, *A multilevel parallel solver for block tridiagonal and banded linear systems*, Parallel Comput., 15 (1990), pp. 21–45.

[11] N. J. Higham, *Bounding the error in Gaussian elimination for tridiagonal systems*, SIAM J. Matrix Anal. Appl., 11 (1990), pp. 521–530.

[12] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia, PA, 1996.

[13] R. Horn and C. Johnsson, *Matrix Analysis*, Cambridge University Press, Cambridge, UK, 1996.

[14] S. L. Johnsson, *Solving tridiagonal systems on ensemble architectures*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 354–392.

[15] A. Krenchel, H. Plum, and K. Stüben, *Parallelization and vectorization aspects of the solution of tridiagonal linear systems*, Parallel Comput., 14 (1990), pp. 31–49.

[16] J. Ortega, *Introduction to Parallel and Vector Solution of Linear Systems*, Plenum Press, New York, 1988.

[17] V. Pavlov and D. Todorova, *Stabilization and experience with the partitioning method for tridiagonal systems*, in Numerical Analysis and Its Applications, Lecture Notes in Comput. Sci. 1196, L. Vulkov, J. Wasniewski, and P. Yalamov, eds., Springer-Verlag, Berlin, 1997, pp. 380-388.

[18] R. Skeel, *Scaling for numerical stability in Gaussian elimination*, J. Assoc. Comput. Mach., 26 (1979), pp. 494–526.

[19] H. Stone, *An efficient parallel algorithm for the solution of a tridiagonal linear system of equations*, J. Assoc. Comput. Mach., 20 (1973), pp. 27–38.

[20] H. Stone, *Parallel tridiagonal solvers*, ACM Trans. Math. Software, 1 (1975), pp. 289–307.

[21] F. Stummel, *Perturbation theory for evaluation algorithms of arithmetic expressions*, Math. Comp., 37 (1981), pp. 435–473.

[22] H. van Der Vorst, *Large tridiagonal and block tridiagonal linear systems on vector and parallel computers*, Parallel Comput., 5 (1987), pp. 45–54.

[23] C. H. Walshaw, *Diagonal dominance in the parallel partition method for tridiagonal systems*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 1086–1099.

[24] H. H. Wang, *A parallel method for tridiagonal linear systems*, ACM Trans. Math. Software, 7 (1981), pp. 170–183.

[25] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, 1965.

[26] P. Yalamov, *Stability of a partitioning algorithm for bidiagonal systems*, Parallel Comput., 23 (1997), pp. 333–348.

[27] P. Yalamov, *On the stability of the cyclic reduction without back substitution for tridiagonal systems*, BIT, 34 (1994), pp. 428–447.

SIAM J. MATRIX ANAL. APPL.
Vol. 20, No. 1, pp. 182–195

POSITIVITY OF BLOCK TRIDIAGONAL MATRICES*

MARTIN BOHNER[†] AND ONDŘEJ DOŠLÝ[‡]

Abstract. This paper relates disconjugacy of linear Hamiltonian difference systems (LHdS) (and hence positive definiteness of certain discrete quadratic functionals) to positive definiteness of some block tridiagonal matrices associated with these systems and functionals. As a special case of a Hamiltonian system, Sturm–Liouville difference equations are considered, and analogous results are obtained for these important objects.

Key words. linear Hamiltonian difference system, Sturm–Liouville difference equation, block tridiagonal matrix, discrete quadratic functional

AMS subject classifications. 15A09, 15A63, 39A10, 39A12

PII. S0895479897318794

1. Introduction.

The aim of this paper is to relate disconjugacy of linear Hamiltonian difference systems (LHdS)

$$\Delta x_k = A_k x_{k+1} + B_k u_k, \quad \Delta u_k = C_k x_{k+1} - A_k^T u_k \tag{H}$$

and hence positivity of the discrete quadratic functional

$$\mathcal{F}(x,u) = \sum_{k=0}^{N} \left\{ u_k^T B_k u_k + x_{k+1}^T C_k x_{k+1} \right\} \tag{F}$$

to positive definiteness of a certain block tridiagonal symmetric matrix associated with (H) and (F). Here A, B, C are sequences of real $n \times n$ matrices such that

$$I - A_k \text{ are invertible and } B_k, C_k \text{ are symmetric for all } k \in \mathbb{N}_0. \tag{1}$$

To introduce our problem in more detail, we first recall the relation of disconjugacy of linear Hamiltonian systems (both differential and difference) to positivity of corresponding quadratic functionals and to solvability of the associated Riccati matrix equation. A statement of this kind is usually called a Reid roundabout theorem.

PROPOSITION 1.1 ([4, 6, 7]). *Consider the linear Hamiltonian differential system*

$$x' = A(t)x + B(t)u, \qquad u' = C(t)x - A^T(t)u, \tag{$\tilde{\text{H}}$}$$

where $A, B, C : I = [a,b] \to \mathbb{R}^{n \times n}$ are continuous real $n \times n$ matrix valued functions such that

$$B(t), C(t) \text{ are symmetric and } B(t) \text{ is positive semidefinite for all } t \in [a,b],$$

and suppose that this system is identically normal *in I, i.e., the only solution (x,u) of ($\tilde{\text{H}}$) for which $x \equiv 0$ on some nondegenerate subinterval of I is the trivial solution $(x,u) \equiv (0,0)$. Then the following statements are equivalent.*

*Received by the editors March 19, 1997; accepted for publication (in revised form) by U. Helmke March 9, 1998; published electronically September 23, 1998.

http://www.siam.org/journals/simax/20-1/31879.html

[†] San Diego State University, Department of Mathematics, 5500 Campanile Dr., San Diego, CA 92182-7200 (bohner@saturn.sdsu.edu). The research of this author was supported by the Alexander von Humboldt Foundation.

[‡] Masaryk University, Department of Mathematics, Janáčkovo Nám. 2A, CZ-66295 Brno, Czech Republic (dosly@math.muni.cz). The research of this author was supported by grant 201/96/0401 of the Czech Grant Agency.

(i) *System* $(\tilde{\mathrm{H}})$ *is* disconjugate *in* I*, i.e., the* $2n\times n$ *matrix solution* $\binom{X}{U}$ *of* $(\tilde{\mathrm{H}})$ *given by the initial condition* $X(a)=0$*,* $U(a)=I$ *has no* focal point *in* I*, i.e.,* $\det X(t)\neq 0$ *in* $(a,b]$.

(ii) *The quadratic functional*

$$(\tilde{\mathrm{F}})\qquad \tilde{\mathcal{F}}(x,u)=\int_a^b \left[u^T(t)B(t)u(t)+x^T(t)C(t)x(t)\right]\,\mathrm{d}t$$

is positive for every $x,u: I\to\mathbb{R}^n$ *satisfying* $x'=A(t)x+B(t)u$*,* $x(a)=0=x(b)$ *and* $x\not\equiv 0$ *in* I.

(iii) *There exists a symmetric solution* $Q: I\to\mathbb{R}^{n\times n}$ *of the Riccati matrix differential equation*

$$(\tilde{\mathrm{R}})\qquad Q'+A^T(t)Q+QA(t)+QB(t)Q-C(t)=0.$$

In the last decade, a considerable effort has been made to find a discrete analogue of this statement; see [1] and the references given therein. Finally, this problem was resolved in [2] and the discrete Roundabout Theorem reads as follows.

Proposition 1.2 ([1, 2]). *Assume* (1)*. Then the following statements are equivalent.*

(i) *System* (H) *is* disconjugate *in the discrete interval* $J:=[0,N]\cap\mathbb{N}_0$*,* $N\in\mathbb{N}$*, i.e., the* $2n\times n$ *matrix solution* $\binom{X}{U}$ *of* (H) *given by the initial condition* $X_0=0$*,* $U_0=I$ *has no* focal point *in* $J^*:=[0,N+1]\cap\mathbb{N}_0$*, i.e.,*

$$\operatorname{Ker} X_{k+1}\subset\operatorname{Ker} X_k \quad \textit{and}\quad D_k=X_kX_{k+1}^{\dagger}(I-A_k)^{-1}B_k\geq 0.$$

(*Here* Ker *and* † *stand for the kernel and the Moore–Penrose generalized inverse of the matrix indicated, respectively, and the matrix inequality* $\geq$ *stands for nonnegative definiteness.*)

(ii) *The discrete quadratic functional* $\mathcal{F}$ *is positive for every* $(x,u): J^*\to\mathbb{R}^n$ *satisfying* $\Delta x_k=A_kx_{k+1}+B_ku_k$*,* $x_0=0=x_{N+1}$ *and* $x\not\equiv 0$ *in* J^*.

(iii) *There exist symmetric matrices* $Q: J^*\to\mathbb{R}^{n\times n}$ *such that* $(I+BQ)$ *are invertible,* $(I+BQ)^{-1}B\geq 0$ *in* J*, and solve the discrete Riccati matrix difference equation*

$$(R)\qquad Q_{k+1}=C_k+(I-A_k^T)Q_k(I+B_kQ_k)^{-1}(I-A_k)^{-1}.$$

The main difference between continuous and discrete functionals $\tilde{\mathcal{F}}$ and $\mathcal{F}$ is that the space of x,u appearing in the discrete quadratic functional has finite dimension. This suggests investigating positivity of $\mathcal{F}$ not only via oscillation properties of (H) and solvability of (R) as given in the roundabout theorem (which are typical methods for an “infinite-dimensional” treatment), but also via linear algebra and matrix theory. The main idea of this approach can be illustrated in the case of the Sturm-Liouville equation

$$(2)\qquad -\Delta(r_k\Delta y_k)+p_ky_{k+1}=0$$

and the corresponding quadratic functional

$$(3)\qquad \mathcal{J}(y)=\sum_{k=0}^{N}\left\{r_k(\Delta y_k)^2+p_ky_{k+1}^2\right\}$$

as follows.

Expanding the differences in $\mathcal{J}$, for any $y = \{y_k\}_{k=0}^{N+1}$ satisfying

$$y_0 = 0 = y_{N+1}, \tag{4}$$

we have

$$\begin{aligned}\mathcal{J}(y) &= \sum_{k=0}^{N} \left\{ (r_k + p_k) y_{k+1}^2 - 2 r_k y_k y_{k+1} + r_k y_k^2 \right\} \\ &= \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix}^T \begin{pmatrix} \beta_0 & -r_1 & & \\ -r_1 & \beta_1 & \ddots & \\ & \ddots & \ddots & -r_{N-1} \\ & & -r_{N-1} & \beta_{N-1} \end{pmatrix} \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix},\end{aligned}$$

where $\beta_k = r_k + p_k + r_{k+1}$. Hence $\mathcal{J}(y) > 0$ for any nontrivial $y \in \mathbb{R}^{N+2}$ satisfying (4) iff the matrix

$$\mathcal{L} := \begin{pmatrix} \beta_0 & -r_1 & & \\ -r_1 & \beta_1 & \ddots & \\ & \ddots & \ddots & -r_{N-1} \\ & & -r_{N-1} & \beta_{N-1} \end{pmatrix}$$

is positive definite.

From the elementary course of linear algebra it is known that $\mathcal{L}$ is positive definite iff all its principal minors $\Delta_1 = \beta_0$, $\Delta_2 = \beta_0\beta_1 - r_1^2, \dots, \Delta_N = \det \mathcal{L}$ are positive. On the other hand, by (i) of Proposition 1.2 we have $\mathcal{J}(y) > 0$ for any nontrivial y satisfying (4) iff the solution $\tilde{y}$ of (2) given by the initial condition $\tilde{y}_0 = 0$, $\tilde{y}_1 = \frac{1}{r_0}$ satisfies

$$\tilde{y}_{k+1} \neq 0 \quad \text{and} \quad \delta_k := \frac{\tilde{y}_k}{r_k \tilde{y}_{k+1}} \geq 0, \qquad k = 0, \dots, N. \tag{5}$$

Using Laplace's rule for computation of determinants, we have the formula

$$\Delta_k = \beta_{k-1}\Delta_{k-1} - r_{k-1}^2 \Delta_{k-2}. \tag{6}$$

Expanding the forward differences in (2) we have

$$y_{k+2} = \frac{1}{r_{k+1}} \left[\beta_k y_{k+1} - r_k y_k \right].$$

This recurrent formula, coupled with (6) and the initial condition $\tilde{y}_0 = 0$, $\tilde{y}_1 = \frac{1}{r_0}$, gives

$$\tilde{y}_2 = \frac{\beta_0}{r_1 r_0} = \frac{\Delta_1}{r_1 r_0}, \qquad \tilde{y}_3 = \frac{1}{r_2 r_1 r_0} \left[\beta_0 \beta_1 - r_1^2 \right] = \frac{\Delta_2}{r_2 r_1 r_0},$$

and by induction

$$\tilde{y}_{k+1} = \frac{1}{r_k \dots r_2 r_1 r_0} \left[\beta_{k-1}\Delta_{k-1} - r_{k-1}^2 \Delta_{k-2} \right] = \frac{\Delta_k}{r_k \dots r_2 r_1 r_0}.$$

Consequently,

$$\delta_k = \frac{\tilde{y}_k}{r_k \tilde{y}_{k+1}} = \frac{\Delta_{k-1}}{\Delta_k}, \quad \Delta_0 := 1. \tag{7}$$

Now, by the Jacobi diagonalization method, there exists an $N \times N$ triangular matrix $\mathcal{M}$ such that

$$\mathcal{M}^T \mathcal{L} \mathcal{M} = \operatorname{diag}\{\delta_1, \dots, \delta_N\}.$$

From the last identity one may easily see why the quantities δ_k come to play in the definition of disconjugacy of (2).

In this paper we establish a similar identity relating the quadratic functional $\mathcal{F}$ and the matrices D_k from Proposition 1.2. This identity reveals why the matrices D_k appear in the definition of disconjugacy of (H). In particular, we find a block triangular matrix $\mathcal{M}$ such that

$$\mathcal{M}^T \mathcal{L} \mathcal{M} = \operatorname{diag}\{D_1, \dots, D_N\},$$

where $\mathcal{L}$ in this identity is the matrix representing the functional $\mathcal{F}$ (see Theorem 2.2 in the next section) and D_k are given in (i) of Proposition 1.2.

The paper is organized as follows. The next section is devoted to preliminary results. We recall some basic properties of solutions of (H), and we also show the relation between higher order Sturm–Liouville difference equations and LHdS (H). The main results of the paper, the equivalence of positive definiteness of a block tridiagonal matrix to nonnegative definiteness of certain matrices constructed via solutions of (H) (which reduce to δ_k in the scalar case) are given in section 3. In the last section we deal with LHdS (H) which correspond to higher order Sturm–Liouville equations; here the results of the previous section are simplified considerably. The statements of section 4 complement results of [3, 5].

2. Preliminary results. Subject to our general assumption (1) we consider a *linear Hamiltonian difference system* (H) and the corresponding *discrete quadratic functional* $\mathcal{F}$ defined by (F). Here, $x = \{x_k\}_{k\in\mathbb{N}_0}$ and $u = \{u_k\}_{k\in\mathbb{N}_0}$ are sequences of $\mathbb{R}^n$-vectors, and we say that such a pair (x, u) is *admissible* on J provided that

$$\Delta x_k = A_k x_{k+1} + B_k u_k \text{for all} k \in J$$

holds. An x is called admissible (on J) if there exists u such that the pair (x, u) is admissible (on J). The functional $\mathcal{F}$ is then said to be *positive definite* (and we write $\mathcal{F} > 0$) whenever

$$\begin{cases} \mathcal{F}(x,u) > 0 \text{ for all on } J \text{ admissible } (x,u) \\ \text{with } x_0 = x_{N+1} = 0 \text{ and } \begin{pmatrix} x_1 \\ \vdots \\ x_N \end{pmatrix} \neq 0 \end{cases}$$

holds. Throughout the paper we denote by (X, U) the *principal solution* of (H) (at 0), i.e., the solution introduced in Proposition 1.2(i). Concerning *Moore–Penrose inverses* we will need the following basic lemma which is proved, e.g., in [2, Remark 2(iii)].

Lemma 2.1. *For any two matrices V and W we have*

$$\mathrm{Ker}V \subset \mathrm{Ker}W \quad \text{iff} \quad W = WV^\dagger V \quad \text{iff} \quad W^\dagger = V^\dagger V W^\dagger.$$

It is the goal of this paper to relate the condition from Proposition 1.2(i) to a condition on certain block tridiagonal $kn \times kn$ matrices of the form

$$\mathcal{L}_k = \begin{pmatrix} T_0 & S_1 & & \\ S_1^T & T_1 & \ddots & \\ & \ddots & \ddots & S_{k-1} \\ & & S_{k-1}^T & T_{k-1} \end{pmatrix}, \qquad k \in \mathbb{N}_0,$$

where we put, for $k \in \mathbb{N}_0$,

$$T_k = C_k + (I - A_k^T)B_k^\dagger(I - A_k) + B_{k+1}^\dagger \quad \text{and} \quad S_k = -B_k^\dagger(I - A_k). \tag{8}$$

Let $\mathcal{L} = \mathcal{L}_N$. Our first result then is the following.

Theorem 2.2. *Let (x, u) be admissible on J with $x_0 = x_{N+1} = 0$. Then we have*

$$\mathcal{F}(x, u) = \begin{pmatrix} x_1 \\ \vdots \\ x_N \end{pmatrix}^T \mathcal{L} \begin{pmatrix} x_1 \\ \vdots \\ x_N \end{pmatrix}.$$

Proof. Let (x, u) be admissible on J so that

$$u_k^T B_k u_k = u_k^T B_k B_k^\dagger B_k u_k = \left(x_{k+1}^T(I - A_k^T) - x_k^T\right) B_k^\dagger \left((I - A_k)x_{k+1} - x_k\right)$$

holds for all $k \in J$. Then, if $x_0 = x_{N+1} = 0$, we have

$$\begin{aligned}
\mathcal{F}(x, u) &= \sum_{k=0}^{N} \left\{ x_{k+1}^T C_k x_{k+1} + \left(x_{k+1}^T(I - A_k^T) - x_k^T\right) B_k^\dagger \left((I - A_k)x_{k+1} - x_k\right) \right\} \\
&= \sum_{k=0}^{N} \left\{ x_{k+1}^T (T_k - B_{k+1}^\dagger) x_{k+1} + 2x_k^T S_k x_{k+1} + x_k^T B_k^\dagger x_k \right\} \\
&= \sum_{k=0}^{N} \left\{ x_{k+1}^T T_k x_{k+1} + 2x_k^T S_k x_{k+1} - \Delta(x_k^T B_k^\dagger x_k) \right\} \\
&= \sum_{k=0}^{N} \left\{ x_{k+1}^T T_k x_{k+1} + 2x_k^T S_k x_{k+1} \right\} - x_{N+1}^T B_{N+1}^\dagger x_{N+1} + x_0^T B_0^\dagger x_0 \\
&= \sum_{k=1}^{N-1} \left\{ x_{k+1}^T T_k x_{k+1} + 2x_k^T S_k x_{k+1} \right\} + x_1^T T_0 x_1 \\
&= \begin{pmatrix} x_1 \\ \vdots \\ x_N \end{pmatrix}^T \mathcal{L} \begin{pmatrix} x_1 \\ \vdots \\ x_N \end{pmatrix},
\end{aligned}$$

and hence our desired result is shown. □

By introducing the space

$$\mathcal{A} = \left\{ \begin{pmatrix} x_1 \\ \vdots \\ x_N \end{pmatrix} : \ x = \{x_k\}_{k \in \mathbb{N}_0} \text{ is admissible on } J \text{ with } x_0 = x_{N+1} = 0 \right\},$$

we can write an immediate consequence of Theorem 2.2.

COROLLARY 2.3. *$\mathcal{F} > 0$ iff $\mathcal{L} > 0$ on $\mathcal{A}$.*

Note that $\mathcal{L} > 0$ on $\mathcal{A}$, i.e., $\chi^T \mathcal{L} \chi > 0$ for all $\chi \in \mathcal{A} \setminus \{0\}$, is equivalent to

$$\mathcal{M}^T \mathcal{L} \mathcal{M} \geq 0 \quad \text{ and } \quad \text{Ker}\mathcal{M}^T \mathcal{L} \mathcal{M} \subset \text{Ker}\mathcal{M}$$

whenever $\mathcal{M}$ is a matrix with $\text{Im}\mathcal{M} = \mathcal{A}$. In the next section we will present such a matrix $\mathcal{M}$ for the general Hamiltonian case, and in the last section we will give this matrix $\mathcal{M}$ and further results for the Sturm–Liouville case. A *Sturm–Liouville difference equation*

$$\text{(SL)} \qquad \sum_{\mu=0}^{n} (-\Delta)^{\mu} \left\{ r_k^{(\mu)} \Delta^{\mu} y_{k+n-\mu} \right\} = 0, \qquad k \in \mathbb{N}_0$$

with reals $r_k^{(\mu)}$, $0 \leq \mu \leq n$, such that $r_k^{(n)} \neq 0$ for all $k \in \mathbb{N}_0$ is a special case of a linear Hamiltonian difference system. We define, for all $k \in \mathbb{N}_0$,

$$(9) \qquad \left\{ \begin{array}{l} A_k = \begin{pmatrix} 0 & 1 & & \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ & & & 0 \end{pmatrix}, \quad B_k = \begin{pmatrix} 0 & & & \\ & \ddots & & \\ & & 0 & \\ & & & \frac{1}{r_k^{(n)}} \end{pmatrix}, \\ \text{and} \qquad C_k = \begin{pmatrix} r_k^{(0)} & & & \\ & r_k^{(1)} & & \\ & & \ddots & \\ & & & r_k^{(n-1)} \end{pmatrix}. \end{array} \right.$$

Then assumption (1) is satisfied and the following result from [3, Lemma 4] holds.

LEMMA 2.4. *Suppose* (9). *Then x is admissible on J iff there exists a sequence $y = \{y_k\}_{0 \leq k \leq N+n-1}$ of reals such that*

$$x_k = \begin{pmatrix} y_{k+n-1} \\ \Delta y_{k+n-2} \\ \Delta^2 y_{k+n-3} \\ \vdots \\ \Delta^{n-1} y_k \end{pmatrix} \qquad \textit{for all} \quad 0 \leq k \leq N+1$$

holds, and in this case the functional defined by (F) *takes the form*

$$(10) \qquad \mathcal{F}(x, u) = \sum_{k=0}^{N} \sum_{\nu=0}^{n} r_k^{(\nu)} \left\{ \Delta^{\nu} y_{k+n-\nu} \right\}^2,$$

where u is such that (x, u) is admissible.

We conclude this section with two auxiliary results where we use the notation introduced in Proposition 1.2.

LEMMA 2.5. *Let $\tilde{A}_k := (I - A_k)^{-1}$ and $D_k = X_k X_{k+1}^{\dagger} \tilde{A}_k B_k$, $k \in \mathbb{N}_0$.*

(i) *Suppose* (1). *If* $\operatorname{Ker} X_{k+1} \subset \operatorname{Ker} X_k$, *then*

$$D_k \text{ is symmetric} \quad \text{and} \quad \operatorname{Ker} X_{k+1}^T \subset \operatorname{Ker} B_k \tilde{A}_k^T.$$

(ii) *Suppose* (9). *Then we have for all* $0 \le k \le n-1$

$$\operatorname{Ker} X_{k+1} \subset \operatorname{Ker} X_k, \quad \operatorname{rank} X_{k+1} = k+1, \quad \text{and} \quad D_k = 0.$$

Proof. While part (i) is shown in [2, Remark 2 (ii)], (ii) is the contents of [3, Lemma 4]. □

Lemma 2.6. *We have, for all* $k \in \mathbb{N}_0$,

$$X_{k+1}^T T_k X_{k+1} = \Delta \left\{ X_k^T (U_k + B_k^\dagger X_k) \right\} - X_k^T S_k X_{k+1} - X_{k+1}^T S_k^T X_k. \tag{11}$$

Proof. The calculation

$$\begin{aligned} X_{k+1}^T T_k X_{k+1} + X_k^T S_k X_{k+1} + X_{k+1}^T S_k^T X_k &= X_{k+1}^T \left\{ U_{k+1} - (I - A_k^T) U_k \right\} \\ &\quad + X_{k+1}^T (I - A_k^T) B_k^\dagger (I - A_k) X_{k+1} + X_{k+1}^T B_{k+1}^\dagger X_{k+1} \\ &\quad - X_k^T B_k^\dagger (I - A_k) X_{k+1} - X_{k+1}^T (I - A_k^T) B_k^\dagger X_k \\ &= X_{k+1}^T U_{k+1} - (X_k^T + U_k^T B_k) U_k + (X_k^T + U_k^T B_k) B_k^\dagger (X_k + B_k U_k) \\ &\quad + X_{k+1}^T B_{k+1}^\dagger X_{k+1} - X_k^T B_k^\dagger (X_k + B_k U_k) - (X_k^T + U_k^T B_k) B_k^\dagger X_k \\ &= \Delta \left\{ X_k^T U_k + X_k^T B_k^\dagger X_k \right\} \end{aligned}$$

shows that formula (11) holds. □

3. The linear Hamiltonian difference system. In this section we present a matrix $\mathcal{M}$ satisfying $\operatorname{Im} \mathcal{M} = \mathcal{A}$. We then give a proof of the following crucial result.

Theorem 3.1. *If* $\operatorname{Ker} X_{k+1} \subset \operatorname{Ker} X_k$ *holds on* J, *then we have*

$$\mathcal{M}^T \mathcal{L} \mathcal{M} = \operatorname{diag} \{ D_1, \dots, D_N \}.$$

To further motivate our investigations, first we would like to briefly consider the case that the matrices

$$B_k \text{ are invertible for all } k \in J. \tag{12}$$

Then we have $\mathcal{A} = \mathbb{R}^{Nn}$, and $\mathcal{F} > 0$ iff $\mathcal{L} > 0$. Then, if (X, U) is the principal solution of (H) at 0 such that X_k are invertible for all $1 \le k \le N+1$, and if we put

$$F_{k+1} = \begin{pmatrix} D_1 S_1 D_2 S_2 \dots D_k S_k \\ -D_2 S_2 \dots D_k S_k \\ \vdots \\ (-1)^{k-1} D_k S_k \\ (-1)^k I \end{pmatrix} \quad \text{for all } 0 \le k \le N,$$

it is easy to show that, for all $0 \le k \le N$,

$$D_{k+1}^{-1} = T_k - S_k^T D_k S_k, \qquad \mathcal{L}_{k+1} F_{k+1} D_{k+1} = \begin{pmatrix} 0 \\ (-1)^k I \end{pmatrix} \tag{13}$$

and

$$\mathcal{L}_{k+1}^{-1} = \begin{pmatrix} \mathcal{L}_k^{-1} & 0 \\ 0 & 0 \end{pmatrix} + F_{k+1} D_{k+1} F_{k+1}^T. \tag{14}$$

Theorem 3.2. *Assume* (12). *We put* $\mathcal{L}_0 := I$ *and* $\tilde{D}_0 := 0$. *Then we have recursively, for* $k = 0, 1, 2, \ldots, N$,

$$\mathcal{L}_{k+1} > 0 \Longleftrightarrow \mathcal{L}_k > 0 \quad \text{and} \quad \tilde{D}_{k+1} := \left\{ T_k - S_k^T \tilde{D}_k S_k \right\}^{-1} > 0. \tag{15}$$

Proof. Of course (15) is obvious for $k = 0$. However, if (15) holds for some $0 \le k \le N-1$, then (put $R_k = \binom{0}{S_k}$)

$$\mathcal{L}_{k+1} = \begin{pmatrix} \mathcal{L}_k & R_k \\ R_k^T & T_k \end{pmatrix} > 0 \Longleftrightarrow \mathcal{L}_k > 0 \quad \text{and} \quad T_k - R_k^T \mathcal{L}_k^{-1} R_k > 0.$$

However, by (14), $T_k - R_k^T \mathcal{L}_k^{-1} R_k = T_k - S_k^T D_k S_k$, where our D_k here are exactly the $\tilde{D}_k$ because of (13) and $D_0 = 0$. This proves our desired assertion. □

Now we turn our attention to the general case of an LHdS (H), where we assume (1). Let us define D_{ij} for $1 \le i \le j \le N$ matrices by

$$D_{ij} = X_i X_j^\dagger D_j.$$

Then we have

$$D_{ii} = X_i X_i^\dagger D_i = X_i X_i^\dagger X_i X_{i+1}^\dagger \tilde{A}_i B_i = X_i X_{i+1}^\dagger \tilde{A}_i B_i = D_i$$

and, by Lemma 2.1, if $\text{Ker} X_j \subset \text{Ker} X_i$,

$$D_{ij} = X_i X_j^\dagger D_j = X_i X_j^\dagger X_j X_{j+1}^\dagger \tilde{A}_j B_j = X_i X_{j+1}^\dagger \tilde{A}_j B_j.$$

We now define, for any $m \in J$, an $mn \times mn$ matrix $\mathcal{M}_m$ by

$$\mathcal{M}_m = \begin{pmatrix} D_{11} & D_{12} & \cdots & D_{1m} \\ 0 & D_{22} & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & D_{mm} \end{pmatrix}$$

and put $\mathcal{M} = \mathcal{M}_N$.

Theorem 3.3. *If* $\text{Ker} X_{k+1} \subset \text{Ker} X_k$ *holds on* J, *then* $\text{Im}\mathcal{M} = \mathcal{A}$.

Proof. We assume $\text{Ker} X_{k+1} \subset \text{Ker} X_k$ on J. First, let $\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \in \mathcal{A}$ and put

$$c_0 := 0, \quad c_{k+1} := c_k - X_{k+1}^\dagger \tilde{A}_k B_k (U_k c_k - u_k) \text{ for } k \in \mathbb{N}_0,$$

where $u = \{u_k\}_{k \in \mathbb{N}_0}$ is such that (x, u) is admissible on J. Let $d_k := U_k c_k - u_k$ for $k \in \mathbb{N}_0$. We have $X_0 c_0 = 0 = x_0$, and $X_k c_k = x_k$ for some $k \in J$ implies

$$\begin{aligned} X_{k+1} c_{k+1} &= X_{k+1} c_k - X_{k+1} X_{k+1}^\dagger \tilde{A}_k B_k (U_k c_k - u_k) \\ &= (\tilde{A}_k X_k + \tilde{A}_k B_k U_k) c_k - \tilde{A}_k B_k (U_k c_k - u_k) \\ &= \tilde{A}_k X_k c_k + \tilde{A}_k B_k u_k = \tilde{A}_k x_k + \tilde{A}_k B_k u_k = x_{k+1} \end{aligned}$$

because of Lemmas 2.5(i) and 2.1. Hence

$$X_k c_k = x_k \quad \text{for all} \quad 0 \le k \le N+1.$$

Next, we have for $j \in J$,

$$D_j d_j = X_j X_{j+1}^\dagger \tilde{A}_j B_j (U_j c_j - u_j) = X_j (c_j - c_{j+1}) = -X_j \Delta c_j$$

so that

$$D_{ij} d_j = X_i X_j^\dagger D_j d_j = -X_i X_j^\dagger X_j \Delta c_j = -X_i \Delta c_j$$

for all $0 \le i \le j \le N$ because of $\mathrm{Ker} X_j \subset \mathrm{Ker} X_i$ and Lemma 2.1. Therefore

$$\begin{aligned}\sum_{j=i}^{N} D_{ij} d_j &= -X_i \sum_{j=i}^{N} \Delta c_j = -X_i (c_{N+1} - c_i)\\ &= -X_i X_{N+1}^\dagger X_{N+1} c_{N+1} + X_i c_i = x_i\end{aligned}$$

for all $0 \le i \le N$ so that

$$\begin{pmatrix} x_1 \\ \vdots \\ x_N \end{pmatrix} = \mathcal{M} \begin{pmatrix} d_1 \\ \vdots \\ d_N \end{pmatrix} \in \mathrm{Im}\mathcal{M}.$$

Conversely, let $\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \in \mathrm{Im}\mathcal{M}$ and put $x_0 = x_{N+1} = 0$. We pick $d_1, \dots, d_N \in \mathbb{R}^n$ with

$$x_i = \sum_{j=i}^{N} D_{ij} d_j \quad \text{for all} \quad 1 \le i \le N.$$

Then we have

$$(I - A_0) x_1 - x_0 = \sum_{j=1}^{N} (I - A_0) X_1 X_j^\dagger D_j d_j = B_0 \sum_{j=1}^{N} X_j^\dagger D_j d_j \in \mathrm{Im} B_0,$$

$$(I - A_N) x_{N+1} - x_N = -x_N = -D_{NN} d_N = -D_{NN}^T d_N \in \mathrm{Im} B_N,$$

and, for $k \in J \setminus \{N\}$,

$$\begin{aligned}(I - A_k) x_{k+1} - x_k &= (I - A_k) \sum_{j=k+1}^{N} X_{k+1} X_j^\dagger D_j d_j - \sum_{j=k}^{N} X_k X_j^\dagger D_j d_j\\ &= \{(I - A_k) X_{k+1} - X_k\} \sum_{j=k+1}^{N} X_j^\dagger D_j d_j - D_k d_k\\ &= B_k U_k \sum_{j=k+1}^{N} X_j^\dagger D_j d_j - D_k d_k \in \mathrm{Im} B_k\end{aligned}$$

by Lemma 2.5(i) so that x is admissible on J and hence $\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \in \mathcal{A}$. $\square$

Our next result directly yields Theorem 3.1.

THEOREM 3.4. *If* $\mathrm{Ker}X_{k+1} \subset \mathrm{Ker}X_k$ *holds on* J, *then*

$$\mathcal{M}_{m+1}^T \mathcal{L}_{m+1} \mathcal{M}_{m+1} = \begin{pmatrix} \mathcal{M}_m^T \mathcal{L}_m \mathcal{M}_m & 0 \\ 0 & D_{m+1} \end{pmatrix}.$$

Proof. Let $k \in J$ and $m \in J \setminus \{N\}$. We define $kn \times n$ matrices P_k and R_k by

$$P_k = \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_k \end{pmatrix} \quad \text{and} \quad R_k = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ S_k \end{pmatrix}.$$

Then we have the recursions

$$\mathcal{M}_{m+1} = \begin{pmatrix} \mathcal{M}_m & P_m X_{m+1}^\dagger D_{m+1} \\ 0 & X_{m+1} X_{m+1}^\dagger D_{m+1} \end{pmatrix} \quad \text{and} \quad \mathcal{L}_{m+1} = \begin{pmatrix} \mathcal{L}_m & R_m \\ R_m^T & T_m \end{pmatrix}.$$

Hence by putting $\tilde{\mathcal{M}}_k = \binom{\mathcal{M}_k}{0}$,

$$\mathcal{M}_{m+1} = \begin{pmatrix} \tilde{\mathcal{M}}_m & P_{m+1} X_{m+1}^\dagger D_{m+1} \end{pmatrix}.$$

Therefore the matrix $\mathcal{M}_{m+1}^T \mathcal{L}_{m+1} \mathcal{M}_{m+1}$ turns out to be

$$\begin{pmatrix} \tilde{\mathcal{M}}_m^T \mathcal{L}_{m+1} \tilde{\mathcal{M}}_m & \tilde{\mathcal{M}}_m^T \mathcal{L}_{m+1} P_{m+1} X_{m+1}^\dagger D_{m+1} \\ D_{m+1} (X_{m+1}^\dagger)^T P_{m+1}^T \mathcal{L}_{m+1} \tilde{\mathcal{M}}_m & D_{m+1} (X_{m+1}^\dagger)^T P_{m+1}^T \mathcal{L}_{m+1} P_{m+1} X_{m+1}^\dagger D_{m+1} \end{pmatrix}$$
$$= \begin{pmatrix} \mathcal{M}_m^T \mathcal{L}_m \mathcal{M}_m & \Omega_m X_{m+1}^\dagger D_{m+1} \\ D_{m+1} (X_{m+1}^\dagger)^T \Omega_m^T & D_{m+1} (X_{m+1}^\dagger)^T \Lambda_m X_{m+1}^\dagger D_{m+1} \end{pmatrix}$$

with

$$\Omega_m = \tilde{\mathcal{M}}_m^T \mathcal{L}_{m+1} P_{m+1} \quad \text{and} \quad \Lambda_m = P_{m+1}^T \mathcal{L}_{m+1} P_{m+1}.$$

Hence our result follows directly from (ii) and (iii) of the following lemma. □

LEMMA 3.5. *If* $\mathrm{Ker}X_{k+1} \subset \mathrm{Ker}X_k$ *holds on* J, *then we have, for all* $k \in J \setminus \{N\}$,

(i) $\Lambda_k = X_{k+1}^T (U_{k+1} + B_{k+1}^\dagger X_{k+1})$;

(ii) $D_{k+1} (X_{k+1}^\dagger)^T \Lambda_k X_{k+1}^\dagger D_{k+1} = D_{k+1}$;

(iii) $\Omega_k = 0$.

Proof. First of all we have, by formula (11) of Lemma 2.6,

$$\Lambda_0 = X_1^T T_0 X_1 = X_1^T (U_1 + B_1^\dagger X_1)$$

and, if $\Lambda_{k-1} = X_k^T (U_k + B_k^\dagger X_k)$ already holds for some $k \in \{1, \ldots, N-1\}$, again by applying formula (11) we have

$$\begin{aligned} \Lambda_k &= \begin{pmatrix} P_k \\ X_{k+1} \end{pmatrix}^T \begin{pmatrix} \mathcal{L}_k & R_k \\ R_k^T & T_k \end{pmatrix} \begin{pmatrix} P_k \\ X_{k+1} \end{pmatrix} \\ &= P_k^T \mathcal{L}_k P_k + P_k^T R_k X_{k+1} + X_{k+1}^T R_k^T P_k + X_{k+1}^T T_k X_{k+1} \\ &= X_{k+1}^T (U_{k+1} + B_{k+1}^\dagger X_{k+1}). \end{aligned}$$

Hence (i) is shown, and by (i), for all $k \in J \setminus \{N\}$,

$$\begin{aligned}
&D_{k+1}(X_{k+1}^{\dagger})^T \Lambda_k X_{k+1}^{\dagger} D_{k+1} = D_{k+1}U_{k+1}X_{k+1}^{\dagger}D_{k+1} + D_{k+1}B_{k+1}^{\dagger}D_{k+1}\\
&= X_{k+1}X_{k+2}^{\dagger}\tilde{A}_{k+1}B_{k+1}U_{k+1}X_{k+1}^{\dagger}D_{k+1} + X_{k+1}X_{k+2}^{\dagger}\tilde{A}_{k+1}X_{k+1}X_{k+1}^{\dagger}D_{k+1}\\
&= X_{k+1}X_{k+2}^{\dagger}X_{k+2}X_{k+1}^{\dagger}D_{k+1} = X_{k+1}X_{k+1}^{\dagger}D_{k+1} = D_{k+1}
\end{aligned}$$

takes care of part (ii). Finally, $\Omega_0 = 0$, and if $\Omega_{k-1} = 0$ already holds for some $k \in \{1, \ldots, N-1\}$, then

$$\begin{aligned}
\Omega_k &= \begin{pmatrix} \mathcal{M}_k^T & 0 \end{pmatrix} \begin{pmatrix} \mathcal{L}_k & R_k \\ R_k^T & T_k \end{pmatrix} \begin{pmatrix} P_k \\ X_{k+1} \end{pmatrix}\\
&= \mathcal{M}_k^T(\mathcal{L}_k P_k + R_k X_{k+1})\\
&= \begin{pmatrix} \tilde{\mathcal{M}}_{k-1}^T \\ D_k(X_k^{\dagger})^T P_k^T \end{pmatrix} \left(\mathcal{L}_k P_k + \begin{pmatrix} 0 \\ S_k X_{k+1} \end{pmatrix} \right)\\
&= \begin{pmatrix} \Omega_{k-1} \\ D_k(X_k^{\dagger})^T \Lambda_{k-1} + D_k S_k X_{k+1} \end{pmatrix}\\
&= \begin{pmatrix} 0 \\ D_k(U_k + B_k^{\dagger}X_k) - D_k B_k^{\dagger}(X_k + B_k U_k) \end{pmatrix} = 0
\end{aligned}$$

because of part (i). Hence (iii) follows, and all our desired results are shown. □

4. The Sturm–Liouville difference equation. In this section we deal with the case where (H) and (F) correspond to a higher order Sturm–Liouville equation (SL) and its corresponding quadratic functional (10). The special structure of the matrices A, B, C enables us to simplify the results of the previous section.

We start with an identity which plays the crucial role in the proof of the main results of this section.

LEMMA 4.1. *Let* (X, U) *be the principal solution of LHdS corresponding to* (SL). *If* X_{k+1} *is nonsingular, then*

$$D_k = X_k X_{k+1}^{-1} \tilde{A}_k B_k = \operatorname{diag}\left\{ 0, \ldots, 0, \frac{\det X_k}{r_k^{(n)} \det X_{k+1}} \right\}.$$

Proof. Nonsingularity of X_{k+1} implies that $\operatorname{Ker} X_{k+1} \subset \operatorname{Ker} X_k$, hence the matrix $D_k = X_k X_{k+1}^{-1} \tilde{A}_k B_k$ is symmetric according to Lemma 2.5(i), and since $B_k = \operatorname{diag}\{0, \ldots, 0, (1/r_k^{(n)})\}$, the only nonzero entry of D_k is in the right lower corner. Using Laplace's rule, we have

$$\begin{aligned}
&\det X_k = \det\left[(I - A_k)X_{k+1} - B_k U_k\right]\\
&= \det\left[(I - A_k)X_{k+1} - \begin{pmatrix} 0 & \ldots & 0 \\ \vdots & & \vdots \\ 0 & \ldots & 0 \\ \frac{1}{r_k^{(n)}}(U_k)_{n,1} & \ldots & \frac{1}{r_k^{(n)}}(U_k)_{n,n} \end{pmatrix} \right]\\
&= \sum_{\nu=1}^{n} \left[((I - A_k)X_{k+1})_{n,\nu} - \frac{1}{r_k^{(n)}}(U_k)_{n,\nu} \right] (\operatorname{adj}\left[(I - A_k)X_{k+1}\right])_{\nu,n}\\
&= \sum_{\nu=1}^{n} ((I - A_k)X_{k+1} - B_k U_k)_{n,\nu} \left(\left[(I - A_k)X_{k+1}\right]^{-1} \right)_{\nu,n} \det\left[(I - A_k)X_{k+1}\right]
\end{aligned}$$

$$\begin{aligned}
&= \sum_{\nu=1}^{n} (X_k)_{n,\nu} \left(X_{k+1}^{-1} \tilde{A}_k \right)_{\nu,n} \det(I - A_k) \det X_{k+1} \\
&= \det X_{k+1} \left(X_k X_{k+1}^{-1} \tilde{A}_k \right)_{n,n} = r_k^{(n)} \det X_{k+1} \left(X_k X_{k+1}^{-1} \tilde{A}_k B_k \right)_{n,n} \\
&= r_k^{(n)} \det X_{k+1} \left(D_k \right)_{n,n} ,
\end{aligned}$$

so the desired result follows. □

In the next statement and its proof we suppose that the matrices $X, \mathcal{L}, \mathcal{M}$ are the same as in the previous section and that the matrices A, B, C in (H) are given by (9).

THEOREM 4.2. *Suppose that $X_n, \ldots, X_{N+1}$ are nonsingular and denote*

$$d_k := (D_k)_{n,n} = \frac{\det X_k}{r_k^{(n)} \det X_{k+1}}, \qquad k = n, \ldots, N. \tag{16}$$

Then there exists an $nN \times (N - n + 1)$ matrix $\mathcal{N}$ such that

$$\mathcal{N}^T \mathcal{L} \mathcal{N} = \operatorname{diag}\{d_n, \ldots, d_N\}.$$

Proof. Observe that the assumption of nonsingularity of $X_n, \ldots, X_{N+1}$ corresponds to the assumption $\operatorname{Ker} X_{k+1} \subset \operatorname{Ker} X_k$ in Theorem 3.1 because of Lemma 2.5 (ii). Denote by $\mathcal{M}^{[j]} \in \mathbb{R}^{nN}$, $j = 1, \ldots, nN$, the columns of the matrix $\mathcal{M}$ and let $\mathcal{N}$ be the $nN \times (N-n+1)$ matrix which results from $\mathcal{M}$ after omitting all zero columns, i.e.,

$$\mathcal{N} = \left[\mathcal{M}^{[n^2]} \ \mathcal{M}^{[n(n+1)]} \ldots \mathcal{M}^{[nN]} \right].$$

Since $D_1 = \cdots = D_{n-1} = 0$ for LHdS corresponding to (SL) by Lemma 2.5 (ii), we have $\mathcal{M}^T \mathcal{L} \mathcal{M} = \operatorname{diag}\{0, \ldots, 0, D_n, \ldots, D_N\}$ by Theorem 3.1. By Lemma 4.1, $D_j = \operatorname{diag}\{0, \ldots, 0, d_j\}$, $j = n, \ldots, N$, and the statement follows from the relation between $\mathcal{M}$ and $\mathcal{N}$. □

Sturm–Liouville equations may be investigated also directly, i.e., not as a special case of LHdS. Let $t_1, \ldots, t_n$ be n-dimensional vectors with entries

$$t_\nu^{(\mu)} = (-1)^{n-\nu} \binom{\mu - 1}{n - \nu}, \qquad \mu = 1, \ldots, n,$$

with the usual convention that $\binom{n}{k} = 0$ if $k > n$ or $k < 0$, and denote by T the $n \times n$ matrix whose columns are the vectors t_ν, i.e., $T = [t_1 \ldots t_n]$. Furthermore, let $\mathcal{P} = (p_{i,j})$ be the $nN \times (N - n + 1)$ matrix with n-vector entries $p_{i,j} \in \mathbb{R}^n$ given by $p_{i,j} = t_{n+j-i}$, with the convention that $t_l = 0$ if $l < 1$ or $l > n$.

It follows from Lemma 2.4 that if (H) corresponds to a Sturm–Liouville equation (SL), then (x, u) satisfying $x_0 = 0 = x_{N+1}$ is admissible iff there exists $y = \{y_k\}_{k=n}^{N}$ such that

$$x_k = T \begin{pmatrix} y_k \\ \vdots \\ y_{k+n-1} \end{pmatrix};$$

hence $(x_1^T,\dots,x_N^T)^T\in\mathcal{A}$ iff

$$\begin{pmatrix} x_1\\ \vdots\\ x_N\end{pmatrix}=\mathcal{P}\begin{pmatrix} y_n\\ \vdots\\ y_N\end{pmatrix}.$$

Consequently, if A,B,C are given by (9), T_k,S_k by (8), and $\mathcal{K}:=\mathcal{P}^T\mathcal{L}\mathcal{P}$, we have

$$\begin{aligned}\mathcal{F}(y)&=\sum_{k=0}^{N}\sum_{\nu=0}^{n}r_k^{(\nu)}(\Delta^\nu y_{k+n-\nu})^2=\sum_{k=0}^{N}\{x_{k+1}^TC_kx_{k+1}+u_k^TB_ku_k\}\\&=\begin{pmatrix} y_n\\ \vdots\\ y_N\end{pmatrix}^T\mathcal{P}^T\mathcal{L}\mathcal{P}\begin{pmatrix} y_n\\ \vdots\\ y_N\end{pmatrix}=\begin{pmatrix} y_n\\ \vdots\\ y_N\end{pmatrix}^T\mathcal{K}\begin{pmatrix} y_n\\ \vdots\\ y_N\end{pmatrix}.\end{aligned}$$

Expanding the differences in (10), it is easy to see that $\mathcal{K}$ is a $2n+1$-diagonal matrix. Our next computations extend the results of the first section by relating the quantities d_k from (16) (in section 1 for $n=1$ denoted by δ_k) to the principal minors of the matrix $\mathcal{K}$.

First we look for a relation between the $nN\times(N-n+1)$ matrices $\mathcal{P}$ and $\mathcal{N}$ and the corresponding representation for $(x_1^T,\dots,x_N^T)^T\in\mathcal{A}$. By the previous considerations and Theorem 3.3, $(x_1^T,\dots,x_N^T)^T\in\mathcal{A}$ iff there exists $c=(c_n,\dots,c_N)^T$ such that

$$\begin{pmatrix} x_1\\ \vdots\\ x_N\end{pmatrix}=\mathcal{N}\begin{pmatrix} c_n\\ \vdots\\ c_N\end{pmatrix}=\mathcal{P}\begin{pmatrix} y_n\\ \vdots\\ y_N\end{pmatrix}.$$

From the last equality we will find a relation between vectors $c=(c_n,\dots,c_N)^T$ and $y=(y_n,\dots,y_N)^T$. Note that these vectors are determined uniquely since the matrices $\mathcal{P}$ and $\mathcal{N}$ have full rank. Denote by d_i^j the last column of the matrix $X_iX_j^\dagger D_j$. Taking into account the form of the matrices $\mathcal{P}$ and $\mathcal{N}$, we get the system of equations

$$\text{(17)}\qquad\left\{\begin{array}{rcl} y_Nt_1&=&c_Nd_N^N,\\ y_{N-1}t_1+y_Nt_2&=&c_{N-1}d_{N-1}^{N-1}+c_nd_{N-1}^N,\\ &\vdots&\\ y_nt_1&=&c_nd_1^n+\ldots+c_Nd_1^N.\end{array}\right.$$

From this system of equations, we see that there exists an upper triangular matrix $\mathcal{B}=(\mathcal{B}_{i,j})\in\mathbb{R}^{(N-n+1)\times(N-n+1)}$ such that $y=\mathcal{B}c$ and that this is just the matrix which reduces the $2n+1$-diagonal matrix $\mathcal{K}=\mathcal{P}^T\mathcal{L}\mathcal{P}$ to the diagonal form. Indeed, we have

$$\begin{aligned}\mathcal{F}(y)&=\begin{pmatrix} y_n\\ \vdots\\ y_N\end{pmatrix}^T\mathcal{K}\begin{pmatrix} y_n\\ \vdots\\ y_N\end{pmatrix}=\begin{pmatrix} x_1\\ \vdots\\ x_N\end{pmatrix}^T\mathcal{L}\begin{pmatrix} x_1\\ \vdots\\ x_N\end{pmatrix}\\&=\begin{pmatrix} c_n\\ \vdots\\ c_N\end{pmatrix}^T\mathcal{N}^T\mathcal{L}\mathcal{N}\begin{pmatrix} c_n\\ \vdots\\ c_N\end{pmatrix}\end{aligned}$$

$$= \begin{pmatrix} y_n \\ \vdots \\ y_N \end{pmatrix}^T (\mathcal{B}^T)^{-1} \mathrm{diag}\{d_n, \dots, d_N\} \mathcal{B}^{-1} \begin{pmatrix} y_n \\ \vdots \\ y_N \end{pmatrix}$$

for any $y = (y_n, \dots, y_N)^T \in \mathbb{R}^{N-n+1}$; hence

$$\mathcal{B}^T \mathcal{K} \mathcal{B} = \mathrm{diag}\{d_n, \dots, d_N\}.$$

Observe also that the diagonal entries of the matrix $\mathcal{B}$ are $\mathcal{B}_{k,k} = (-1)^{n-1} d_{n+k-1}$, $k = 1, \dots, N-n+1$ (this follows directly from (17)), and that

$$(\mathcal{K}\mathcal{B})_{j,k} = (\mathcal{B}^T \mathcal{K})_{k,j} = \left(\mathrm{diag}\{d_n, \dots, d_N\} \mathcal{B}^{-1}\right)_{k,j} = \begin{cases} 0 & j = 1, \dots, k-1, \\ (-1)^{n-1} & j = k. \end{cases}$$

Here we used the information about diagonal entries of $\mathcal{B}$. The last expression may be regarded as a system of linear equations for $\mathcal{B}_{1,k}, \dots, \mathcal{B}_{k,k}$, and by Cramer's rule we have

$$\mathcal{B}_{k,k} = (-1)^{n-1} d_{n+k-1} = \frac{(-1)^{n-1} \Delta_{k-1}}{\Delta_k}.$$

Now we can summarize our previous computations and relate the quantities d_k from (16), $k = n, \dots, N$, to the principal minors Δ_k of $\mathcal{K}$. This statement may be viewed as a direct extension of (7) to higher order Sturm–Liouville difference equations. Here, similarly as in the previous theorem, (X, U) is the principal solution of (H) with A, B, C given by (9), and the assumption of nonsingularity of the matrices $X_n, \dots, X_{N+1}$ has the same meaning as in Theorem 4.2.

THEOREM 4.3. *Suppose $X_n, \dots, X_{N+1}$ are nonsingular and let Δ_k be the principal minors of the matrix $\mathcal{K}$. Then, for all $1 \le k \le N-n+1$,*

$$d_{n+k-1} = \frac{\Delta_{k-1}}{\Delta_k}, \quad \Delta_0 := 1.$$

REFERENCES

[1] C. D. AHLBRANDT AND A. C. PETERSON, *Discrete Hamiltonian Systems: Difference Equations, Continued Fractions, and Riccati Equations*, Kluwer Academic Publishers, Boston, MA, 1996.

[2] M. BOHNER, *Linear Hamiltonian difference systems: Disconjugacy and Jacobi-type conditions*, J. Math. Anal. Appl., 199 (1996), pp. 804–826.

[3] M. BOHNER, *On disconjugacy for Sturm-Liouville difference equations*, J. Difference Equations and Appl., 2 (1996), pp. 227–237.

[4] W. A. COPPEL, *Disconjugacy*, Lecture Notes in Math. 220, Springer-Verlag, Berlin, 1971.

[5] O. DOŠLÝ, *Factorization of disconjugate higher order Sturm-Liouville difference operators*, Comput. Math. Appl., to appear.

[6] W. T. REID, *Ordinary Differential Equations*, John Wiley, New York, 1971.

[7] W. T. REID, *Sturmian Theory for Ordinary Differential Equations*, Springer-Verlag, New York, 1980.

SIAM J. Matrix Anal. Appl.
Vol. 20, No. 1, pp. 196–227

ON THE DEGREE OF MIXED POLYNOMIAL MATRICES*

KAZUO MUROTA†

Abstract. The mixed polynomial matrix, introduced as a convenient mathematical tool for the description of physical/engineering dynamical systems, is a polynomial matrix of which the coefficients are classified into fixed constants and independent parameters. The valuated matroid, invented by Dress and Wenzel [*Appl. Math. Lett.*, 3 (1990), pp. 33–35], is a combinatorial abstraction of the degree of minors (subdeterminants) of a polynomial matrix. We discuss a number of implications of the recent developments in the theory of valuated matroids in the context of polynomial matrix theory. In particular, we apply the valuated matroid intersection theorem to the analysis of the degree of the determinant of a mixed polynomial matrix to obtain a novel duality identity together with an efficient algorithm.

Key words. combinatorial matrix theory, degree of determinant, mixed matrix, polynomial matrix, valuated matroid

AMS subject classifications. 05C50, 68Q40, 90C27

PII. S0895479896311438

1. Introduction.

Matrices consisting of polynomials or rational functions play fundamental roles in various branches in engineering (Gohberg, Lancaster, and Rodman [22]). For example, in dynamical system theory (Rosenbrock [51], Vidyasagar [59]), a linear time-invariant system is described by a polynomial matrix called the system matrix (the Laplace transform of the state-space equations) or by a rational function matrix called the transfer function matrix. Therefore, it is often the case that the degrees of minors (subdeterminants) of such a matrix have essential engineering significance (see section 2). The objective of this paper is to contribute to the combinatorial theory of matrices (Brualdi and Ryser [3], Edmonds [12]) by investigating the combinatorial aspects of the degree of minors of a polynomial/rational matrix using recent results on valuated matroids.

Let $A(s) = (A_{ij}(s))$ be an $m \times n$ rational function matrix with $A_{ij}(s)$ being a rational function in s with coefficients from a certain field $\boldsymbol{F}$ (typically the real number field $\mathbf{R}$). Denote by R and C the row set and the column set of A. In this paper we are interested in the highest degree of a minor (subdeterminant) of order k of $A(s)$:

$$\delta_k = \delta_k(A) = \max\{\deg_s \det A[I,J] \mid |I| = |J| = k\}, \tag{1.1}$$

where $A[I,J]$ denotes the submatrix of A with row set $I \subseteq R$ and column set $J \subseteq C$, and the degree of a rational function $f(s) = p(s)/q(s)$ (with $p(s)$ and $q(s)$ being polynomials) is defined by $\deg_s f(s) = \deg_s p(s) - \deg_s q(s)$. By convention we put $\deg_s(0) = -\infty$.

Define $\delta : 2^R \times 2^C \to \mathbf{Z} \cup \{-\infty\}$ and $\omega : 2^{R \cup C} \to \mathbf{Z} \cup \{-\infty\}$ by

$$\delta(I,J) = \deg_s \det A[I,J] \qquad (I \subseteq R, J \subseteq C), \tag{1.2}$$

$$\omega(B) = \delta(R \setminus B, C \cap B) \qquad (B \subseteq R \cup C), \tag{1.3}$$

[*] Received by the editors November 1, 1996; accepted for publication (in revised form) December 5, 1997; published electronically September 23, 1998.

http://www.siam.org/journals/simax/20-1/31143.html

[†] Research Institute for Mathematical Sciences, Kyoto University, Kyoto 606-8502, Japan (murota@kurims.kyoto-u.ac.jp).

where $\delta(\emptyset, \emptyset) = \omega(R) = 0$ and $\delta(I, J) = -\infty$ unless $|I| = |J|$. A combinatorial property of the function ω (equivalently, of δ) has been abstracted by Dress and Wenzel [10], [11] as the concept of *valuated matroid*. A valuated matroid is a pair $\mathbf{M} = (V, \omega)$ of a finite set V and a function $\omega : 2^V \to \mathbf{R} \cup \{-\infty\}$ such that $\mathcal{B} = \{B \subseteq V \mid \omega(B) \neq -\infty\}$ is nonempty and that the following exchange property holds:

(MV) For $B, B' \in \mathcal{B}$ and $u \in B - B'$, there exists $v \in B' - B$ such that $B - u + v \in \mathcal{B}$, $B' + u - v \in \mathcal{B}$, and

$$\omega(B) + \omega(B') \leq \omega(B - u + v) + \omega(B' + u - v).$$

The function ω of (1.3) arising from a rational function matrix $A(s)$ defines a valuated matroid $\mathbf{M} = (V, \omega)$ with $V = R \cup C$ and

$$\mathcal{B} = \{B \subseteq R \cup C \mid A[R \setminus B, C \cap B] \text{ is nonsingular}\}. \tag{1.4}$$

It has turned out that valuated matroids afford a nice combinatorial framework to which the optimization algorithms established for matroids generalize naturally (see Welsh [60], White [61] for matroid theory; Faigle [15], Fujishige [18], Lawler [32] for combinatorial optimization on matroids; and Iri [27], Murota [34], Recski [49] for application of matroids). Variants of greedy algorithms work for maximizing a matroid valuation, as has been shown by Dress and Wenzel [10] as well as by Dress and Terhalle [7, 8, 9] and Murota [39]. (These greedy-type algorithms are similar to, but not the same as, those in Korte, Lovász, and Schrader [31].) The weighted matroid intersection problem has been extended by Murota [41, 42] to the valuated matroid intersection problem with natural extensions of optimality criteria and algorithms. The essence of the present paper is an application of these results on the valuated matroid intersection problem to mixed polynomial matrices (or rather, it was the analysis of mixed polynomial matrices that had motivated the present author to investigate the valuated matroid intersection problem).

The concept of mixed polynomial matrix was introduced by Murota [34] (see also Murota and Iri [45]) as a convenient mathematical tool for the description of physical/engineering (linear time-invariant) dynamical systems. Let $\boldsymbol{K}$ be a subfield of a field $\boldsymbol{F}$. A polynomial matrix $A(s)$ over $\boldsymbol{F}$ (i.e., $A_{ij}(s) \in \boldsymbol{F}[s]$) is called a *mixed polynomial matrix* with respect to $\boldsymbol{F}/\boldsymbol{K}$ if

$$A(s) = Q(s) + T(s) = \sum_{k=0}^{N} s^k Q_k + \sum_{k=0}^{N} s^k T_k \tag{1.5}$$

for some integer $N \geq 0$, where

(MP-Q) Q_k $(k = 0, 1, \ldots, N)$ are matrices over $\boldsymbol{K}$, and

(MP-T) T_k $(k = 0, 1, \ldots, N)$ are matrices over $\boldsymbol{F}$ such that the set of their nonzero entries is algebraically independent over $\boldsymbol{K}$.

The assumption (MP-T) means that we can regard the nonzero entries of T_k $(k = 0, 1, \ldots, N)$ as independent parameters.

Example 1.1. For an illustration of the definition above, here is a concrete example of 2×3 mixed polynomial matrix:

$$A(s) = \begin{array}{c|ccc|} & c_1 & c_2 & c_3 \\ \hline r_1 & s^3 + 1 & s^2 + \alpha_1 & \alpha_2 s + 1 \\ r_2 & s^2 + \alpha_3 & s & 0 \\ \hline \end{array}$$

with

$$Q(s) = \boxed{\begin{matrix} s^3+1 & s^2 & 1 \\ s^2 & s & 0 \end{matrix}}\,, \quad T(s) = \boxed{\begin{matrix} 0 & \alpha_1 & \alpha_2 s \\ \alpha_3 & 0 & 0 \end{matrix}}\,.$$

Here we assume $\{\alpha_1, \alpha_2, \alpha_3\}$ to be algebraically independent over $\mathbf{Q}$ (field of rational numbers). Then we may take $\boldsymbol{K} = \mathbf{Q}$ and $\boldsymbol{F} = \mathbf{Q}(\alpha_1, \alpha_2, \alpha_3)$ (field of rational functions in α_1, α_2, α_3 over $\mathbf{Q}$). □

For a nonsingular mixed polynomial matrix $A(s) = Q(s) + T(s)$ we have the following identity:

$$\deg_s \det A = \max_{\substack{|I|=|J| \\ I \subseteq R, J \subseteq C}} \{\deg_s \det Q[I, J] + \deg_s \det T[R - I, C - J]\}, \tag{1.6}$$

which can be derived from (MP-Q), (MP-T), and the Laplace expansion of determinants (see Theorem 4.1 in section 4). The right-hand side of this identity involves a maximization over all pairs (I, J), the number of which is $\binom{|R|+|C|}{|R|}$, too large for an exhaustive search for maximization. Fortunately, however, the functions $\delta_Q(I, J) = \deg_s \det Q[I, J]$ and $\delta_T(I, J) = \deg_s \det T[I, J]$ enjoy a nice combinatorial property, each defining a valuated matroid, as explained above. Moreover, this maximization problem can be rewritten as a valuated matroid intersection problem, for which efficient (polynomial-time) algorithms have been developed.

This approach to the computation of $\deg_s \det A$ extends to the computation of $\delta_k(A)$ for a specified k. It is one of the main objectives of this paper to describe a concrete procedure for efficiently computing $\delta_k(A)$ for a mixed polynomial matrix A.

This paper is organized as follows. In section 2 we justify our interest in δ_k by explaining significances of δ_k in engineering context. In section 3 we summarize relevant facts on valuated matroids and indicate their implications in engineering applications. Sections 4 and 5 compose the main part of the paper. We derive new identities for δ_k and determine the coefficient of the highest degree term of $\det A$ with reference to the combinatorial canonical form of a layered mixed matrix. In section 5 we describe the algorithm for δ_k together with a worked-out example.

Remark 1.1. Already in [33, 34] (before the invention of valuated matroids), the present author was interested in the degree of the determinant of a mixed polynomial matrix and designed an efficient algorithm by making use of the results on the matroid intersection problem. It was possible to avoid valuated matroids because of a stronger assumption imposed on $Q(s)$:
(MP-Q$'$) Every nonvanishing subdeterminant of $Q(s)$ is a monomial in s over $\boldsymbol{K}$.
It was discussed at full length that this stronger assumption could be justified in engineering applications for a physical reason that can be categorized as a kind of dimensional analysis. See also an expository article [40]. □

Remark 1.2. The problem of computing $\delta_k(A)$ has attracted considerable research interest. See Bujakiewicz [4], Commault, Dion, and Perez [6], Hovelaque, Commault, and Dion [26], Reinschke [50], Suda, Wan, and Ueno [52], Svaricek [53, 54], and van der Woude [65] for graph-theoretic approaches; Murota [37, 38] and Iwata, Murota, and Sakuta [30] for combinatorial relaxation algorithms based on graph-theoretic methods; Murota and van der Woude [47] for a matroid-theoretic approach; and Dress and Terhalle [7, 8] and Murota [39] for valuated matroid-theoretic approaches. A recent paper of Iwata and Murota [29] affords a combinatorial relaxation algorithm based on the results of the present paper. □

2. Degree of subdeterminants. In this section we dwell on the significance of $\delta_k(A) = \max_{|I|=|J|=k} \deg_s \det A[I,J]$ in engineering context in order to motivate and justify our present interest in δ_k. The reader may go on to section 3 as the subsequent technical developments are independent of this section.

2.1. Kronecker form of a matrix pencil. A linear time-invariant dynamical system can be expressed most naturally in a descriptor form

$$E\dot{\boldsymbol{x}}(t) = F\boldsymbol{x}(t) + G\boldsymbol{u}(t), \quad \boldsymbol{y}(t) = H\boldsymbol{x}(t) \tag{2.1}$$

with "state" $\boldsymbol{x}(t) \in \mathbf{R}^N$, input $\boldsymbol{u}(t)$, and output $\boldsymbol{y}(t)$. When described in the frequency domain, the coefficient matrix is given by

$$\begin{pmatrix} F - sE & G \\ H & O \end{pmatrix}.$$

Here it is natural to assume that $sE - F$ is a regular pencil, i.e., that $\det(sE-F) \neq 0$, while E can be singular. Then $\delta_N(sE-F)$ represents the dynamical degree [25] (the number of independent initial conditions that can be imposed), which is equal to the dimension of the equivalent state-space equations.

For an $n \times n$ regular matrix pencil $A(s)$ in general, more detailed information can be obtained from the sequence $\delta_k(A)$ $(k = 1, 2, \ldots, n)$. As is well known (Gantmacher [19]), a regular pencil can be brought into the Kronecker form by means of "strict equivalence" as follows.

THEOREM 2.1 (Kronecker form). *Assume that $\boldsymbol{F}$ is an algebraically closed field of characteristic zero (e.g., $\boldsymbol{F} = \mathbf{C}$ (complex numbers)), and let $A(s)$ be an $n \times n$ regular pencil. There exist nonsingular constant matrices P and Q such that*

$$PA(s)Q = \text{block-diag}\left(sI_{m_0} + B; N_{m_1}(s), N_{m_2}(s), \ldots, N_{m_b}(s)\right), \tag{2.2}$$

where

$$m_1 \geq m_2 \geq \cdots \geq m_b \geq 1, \qquad m_0 + m_1 + \cdots + m_b = n,$$

B is an $m_0 \times m_0$ constant matrix, and $N_m(s)$ denotes an $m \times m$ bidiagonal matrix defined by

$$N_m(s) = \begin{pmatrix} 1 & s & & & \\ & 1 & s & & \\ & & \ddots & \ddots & \\ & & & \ddots & s \\ & & & & 1 \end{pmatrix}.$$

The matrices $N_{m_k}(s)$ $(k = 1, \ldots, b)$ are called the nilpotent blocks, and the number $m_1 = \max_{1 \leq k \leq b} m_k$ is the index of nilpotency.

Apart from the matrix B, the structural indices of the Kronecker form, i.e., the integers b, m_0, $m_1, \ldots, m_b$, can be determined from $\delta_k(A)$ $(k = 1, 2, \ldots, n)$ by

$$b = n - \max\{k \mid \delta_k(A) - \delta_{k-1}(A) = 1\} \tag{2.3}$$

(where $b = n$ if such k does not exist) and by

$$m_k = \begin{cases} \delta_n(A) & (k = 0), \\ \delta_{n-k}(A) - \delta_{n-k+1}(A) + 1 & (k = 1, \ldots, b). \end{cases} \tag{2.4}$$

Formulae (2.3) and (2.4) can be derived easily from (2.2) (cf. Murota [38]).

In the literature of numerical analysis a system of equations consisting of a mixture of differential and algebraic relations is often abbreviated to DAE. For a linear time-invariant DAE in general, say $A\boldsymbol{x} = \boldsymbol{b}$ with $A = A(s)$ being a nonsingular polynomial matrix in s, the index is defined by[1]

$$\nu(A) = \max_{i,j} \deg_s (A^{-1})_{ji} + 1.$$

Here it should be clear that each entry $(A^{-1})_{ji}$ of A^{-1} is a rational function in s. An alternative expression is

$$\nu(A) = \delta_{n-1}(A) - \delta_n(A) + 1.$$

When $\deg_s A_{ij} = 0$ or 1 for all (i, j) with $A_{ij} \neq 0$, as in (2.1), the index $\nu(A)$ agrees with the index of nilpotency of A as a matrix pencil; namely, we have $\nu(A) = m_1$.

The solution $\boldsymbol{x}$ to $A\boldsymbol{x} = \boldsymbol{b}$ is of course given by $\boldsymbol{x} = A^{-1}\boldsymbol{b}$, and therefore $\nu(A) - 1$ equals the highest order of the derivatives of the input $\boldsymbol{b}$ that can possibly appear in the solution $\boldsymbol{x}$. As such, a high index indicates the difficulty in numerical solution of the DAE, and sometimes even the inadequacy in mathematical modeling. See Brenan, Campbell, and Petzold [2], Gear [20, 21], Hairer and Wanner [23], Ungar, Kröner, and Marquardt [57] for more about the index of DAE.

2.2. Smith–McMillan form at infinity of a rational matrix. A rational function $f(s)$ is called proper if $\deg_s f(s) \leq 0$. We call a matrix a proper rational matrix if its entries are proper rational functions. A square proper rational matrix is called biproper if it is invertible and its inverse is a proper rational matrix. A proper rational matrix is biproper if and only if its determinant is of degree zero.

Since the proper rational functions form a Euclidean ring, any proper rational matrix can be brought into the Smith form (Newman [48]), which is sometimes referred to as the structure at infinity in the control literature. From this we see further that any rational matrix can be brought into the Smith–McMillan form at infinity, as stated below (Verghese and Kailath [58]).

THEOREM 2.2 (Smith–McMillan form at infinity). *Let $A(s)$ be a rational function matrix. There exist biproper matrices $U(s)$ and $V(s)$ such that*

$$U(s)A(s)V(s) = \begin{pmatrix} \Gamma(s) & O \\ O & O \end{pmatrix},$$

where

$$\Gamma(s) = \mathrm{diag}\,(s^{t_1}, \ldots, s^{t_r}),$$

$r = \mathrm{rank} A(s)$, *and $t_k = t_k(A)$ $(k = 1, \ldots, r)$ are integers with $t_1 \geq \cdots \geq t_r$. Furthermore, t_k can be expressed in terms of the minors of A as*

$$t_k(A) = \delta_k(A) - \delta_{k-1}(A) \qquad (k = 1, \ldots, r), \tag{2.5}$$

where $\delta_k(A)$ is defined by (1.1) *and $\delta_0(A) = 0$ by convention.*

The integers t_k $(k = 1, \ldots, r)$, uniquely determined by (2.5), are referred to as the contents at infinity (Verghese and Kailath [58]). If they are positive, t_k $(k = 1, \ldots, r)$

[1] The definition of index given here applies only to linear time-invariant DAE systems. Index can be defined for more general systems, and two kinds are distinguished in the literature, differential index and perturbation index, which coincide with each other for linear time-invariant DAE systems.

are the orders of the poles at infinity; if negative, $-t_k$ $(k = 1, \ldots, r)$ are the orders of the zeroes at infinity.

A (proper) rational function matrix typically appears as the transfer function matrix of a linear time–invariant dynamical system. The transfer function matrix of the descriptor system (2.1) is given by

$$P(s) = H(sE - F)^{-1}G, \tag{2.6}$$

provided that $\det(sE - F) \neq 0$ (while E can be singular). The Smith–McMillan form at infinity of $P(s)$ has control-theoretic significances (Commault and Dion [5], Hautus [24], Svaricek [54], Verghese and Kailath [58]).

In such a case it is desirable to express the Smith–McMillan form at infinity of $P(s)$ directly from the matrices E, F, G, and H, without referring to the entries of $P(s)$ explicitly. From the well-known formula

$$\det \begin{pmatrix} F - sE & G' \\ H' & O \end{pmatrix} = \det(F - sE) \cdot \det[-H'(F - sE)^{-1}G'],$$

where H' denotes a submatrix of H with k rows and G' is a submatrix of G with k columns, it follows that

$$\delta_k(P) = \delta_{N+k}(A; I_0, J_0) - \delta_N(F - sE), \tag{2.7}$$

where

$$A(s) = \begin{pmatrix} F - sE & G \\ H & O \end{pmatrix},$$

I_0 and J_0 are, respectively, the row and column sets corresponding to the $N \times N$ nonsingular submatrix $F - sE$, and

$$\delta_{N+k}(A; I_0, J_0) = \max\{\deg_s \det A[I, J] \mid I \supseteq I_0,\ J \supseteq J_0, |I| = |J| = N + k\}$$

means the highest degree of a minor of order $N + k$ that contains row set I_0 and column set J_0. Note that $\delta_{N+k}(A; I_0, J_0) = \delta_{N+k}(\widetilde{A}) - 2Nd$ for a sufficiently large integer d and

$$\widetilde{A}(s) = \begin{pmatrix} \mathrm{diag}\,(s^d, \ldots, s^d) & O \\ O & I \end{pmatrix} \begin{pmatrix} F - sE & G \\ H & O \end{pmatrix} \begin{pmatrix} \mathrm{diag}\,(s^d, \ldots, s^d) & O \\ O & I \end{pmatrix}.$$

2.3. Causal splitting for autoregressive models. In the behavioral approach of Willems [62, 63] to dynamical systems, no a priori distinction is made between inputs and outputs in the description of a dynamical system, but they are distinguished only a posteriori in view of the causality implied by the description. The maximum degree of determinants plays an important role in this connection.

To be specific, let $(w_j(t) \mid t = 0, 1, 2, \ldots)$ $(j = 1, \ldots, n)$ be n sequences, each indexed by $\mathbf{Z}_+ = \{t \in \mathbf{Z} \mid t \geq 0\}$. We consider here an autoregressive (AR) model, in which we assume that they are subject to a system of m homogeneous difference equations

$$\sum_{j=1}^{n} \sum_{k=1}^{N_{ij}} A_{ijk} w_j(t + k) = 0 \qquad (i = 1, \ldots, m)$$

with constant coefficients A_{ijk}. Denoting by s the backward time shift, i.e., $s \cdot w_j(t) = w_j(t+1)$, the above equation can be rewritten as $A(s)w(t) = 0$ with $A(s) = (\sum_{k=1}^{N_{ij}} A_{ijk}s^k \mid i = 1, \ldots, m; j = 1, \ldots, n)$ and $w(t) = (w_j(t) \mid j = 1, \ldots, n)$.

The variables w_j are called external variables, which are to be divided into two parts, inputs and outputs, so that (in the Z-domain) the outputs can be computed from the inputs as well as their initial values using proper transfer matrices. The number of outputs is equal to the rank of $A(s)$, say r, and consequently, that of inputs is $n - r$. Such a splitting of n external variables into inputs and outputs is named a causal splitting by van der Woude [66].

In case $r = m$, a causal splitting is tantamount to a splitting of the column set C_A of $A(s)$ into two disjoint sets B and $C_A - B$ in such a way that $A[R_A, B]$ is nonsingular and $A[R_A, B]^{-1}A[R_A, C_A - B]$ is proper, where R_A denotes the row set of $A(s)$. It is easy to see that $A[R_A, B]^{-1}A[R_A, C_A - B]$ is proper if and only if $\deg_s \det A[R_A, B]$ is maximized by B. Also in the general case, finding a causal splitting amounts to finding a submatrix $A[I, B]$ with $|I| = |B| = r$ that has the maximum value of $\deg_s \det A[I, B]$, as follows.

THEOREM 2.3 (van der Woude [64]). *$(B, C_A - B)$ is a causal splitting if and only if there exists $I \subseteq R_A$ such that $|I| = |B| = r$ and $\deg_s \det A[I, B] = \delta_r$, where δ_r is defined by* (1.1).

Let us call $B \subseteq C_A$ a dynamical base of a polynomial matrix $A(s)$ if $(B, C_A - B)$ is a causal splitting. In section 3.1 we will see that the family of dynamical bases of a given matrix possesses a nice combinatorial property, forming the basis family of a matroid.

3. Valuated matroid.

In this section we summarize relevant facts on valuated matroids and discuss their implications for polynomial/rational matrices.

3.1. Definition.

As is already mentioned in the introduction, a valuated matroid is a pair $\mathbf{M} = (V, \omega)$ of a finite set V and a function $\omega : 2^V \to \mathbf{R} \cup \{-\infty\}$ such that

$$\mathcal{B} = \{B \subseteq V \mid \omega(B) \neq -\infty\} \tag{3.1}$$

is nonempty and that the following exchange property holds:

(MV) For $B, B' \in \mathcal{B}$ and $u \in B - B'$, there exists $v \in B' - B$, such that $B - u + v \in \mathcal{B}$, $B' + u - v \in \mathcal{B}$, and

$$\omega(B) + \omega(B') \leq \omega(B - u + v) + \omega(B' + u - v).$$

If this is the case, $\mathcal{B}$ satisfies the following simultaneous exchange property:

(SE) For $B, B' \in \mathcal{B}$ and $u \in B - B'$, there exists $v \in B' - B$ such that $B - u + v \in \mathcal{B}$, $B' + u - v \in \mathcal{B}$,

and accordingly $\mathcal{B}$ forms the basis family of a matroid. Therefore, we can alternatively say that a valuated matroid is a triple $\mathbf{M} = (V, \mathcal{B}, \omega)$, where $(V, \mathcal{B})$ is a matroid (defined in terms of the basis family) and $\omega : \mathcal{B} \to \mathbf{R}$ is a function satisfying (MV).

Our interest in valuated matroids originates from the fact that a rational function matrix $A(s)$ defines a valuated matroid. Let $A(s)$ be an $m \times n$ matrix of rank m with each entry being a rational function in a variable s, and let $\mathbf{M}_A = (C_A, \mathcal{B}_A)$ denote the (linear) matroid [60, 61] defined on the column set C_A of $A(s)$ by the linear independence of the column vectors. Namely,

$$\mathcal{B}_A = \{B \subseteq C_A \mid A[R_A, B] \text{ is nonsingular}\}, \tag{3.2}$$

where R_A denotes the row set of A. Then $\omega_A : \mathcal{B}_A \to \mathbf{Z}$ defined by

$$\omega_A(B) = \deg_s \det A[R_A, B] \qquad (B \in \mathcal{B}_A) \tag{3.3}$$

satisfies the exchange axiom (MV) above (see Dress and Wenzel [11] for the proof). The valuated matroid explained in Introduction (cf., (1.3), (1.4)) is a variant of this construction.

3.2. Maximization in a valuated matroid. Let $\mathbf{M} = (V, \mathcal{B}, \omega)$ be a valuated matroid. For $B \in \mathcal{B}$, $u \in B$, and $v \in V - B$ we define

$$\omega(B, u, v) = \omega(B - u + v) - \omega(B). \tag{3.4}$$

The first lemma is most fundamental, showing the local optimality implies the global optimality.

LEMMA 3.1 (Dress and Wenzel [10, 11]). *Let $B \in \mathcal{B}$. Then $\omega(B) \geq \omega(B')$ for any $B' \subseteq V$ if and only if*

$$\omega(B, u, v) \leq 0 \quad \textit{for any } u \in B \textit{ and } v \in V - B. \tag{3.5}$$

When applied to a rational function matrix, this lemma yields the following. It is remarked that the second statement below is implicit in Willems [62] and van der Woude [64] (cf., Theorem 2.3).

LEMMA 3.2. *For $\mathbf{M}_A = (C_A, \mathcal{B}_A, \omega_A)$ associated with a rational matrix $A(s)$ of row-full rank, we have the following.*

(1) *For $B \in \mathcal{B}_A$,*

$$\omega_A(B, u, v) = \deg_s(A[R_A, B]^{-1} A[R_A, C_A - B])_{uv} \qquad (u \in B, v \in C_A - B).$$

(*The right-hand side designates the degree of the (u, v) entry of the rational matrix $A[R_A, B]^{-1} A[R_A, C_A - B]$.*)

(2) *$B \subseteq C_A$ maximizes ω_A if and only if $A[R_A, B]^{-1} A[R_A, C_A - B]$ is a proper rational matrix.*

For $p : V \to \mathbf{R}$ we define $\omega[p] : 2^V \to \mathbf{R} \cup \{-\infty\}$ (or $\mathcal{B} \to \mathbf{R}$) by

$$\omega[p](B) = \omega(B) + \sum \{p(u) \mid u \in B\}. \tag{3.6}$$

$\mathbf{M}[p] = (V, \mathcal{B}, \omega[p])$ is again a valuated matroid, called a similarity transformation of $\mathbf{M}$. For $\mathbf{M}_A = (C_A, \mathcal{B}_A, \omega_A)$ associated with a rational matrix $A(s)$, this operation corresponds to multiplying a diagonal matrix $\mathrm{diag}\,(s; p) = \mathrm{diag}\,(s^{p_1}, \ldots, s^{p_n})$ from the right.

The following theorem characterizes a valuated matroid as a family of matroids. The "only if" part is immediate from (MV), as observed by Dress and Wenzel [11].

THEOREM 3.3 (Murota [43, 44]). *Let $\omega : 2^V \to \mathbf{R} \cup \{-\infty\}$ be a function such that $\mathcal{B} = \{B \subseteq V \mid \omega(B) \neq -\infty\}$ forms the basis family of a matroid on V. Then $(V, \mathcal{B}, \omega)$ is a valuated matroid if and only if for any $p : V \to \mathbf{R}$ the set of the maximizers of $\omega[p]$ forms the basis family of a matroid on V.*

As explained in section 2.3, a causal splitting for $A(s)$ consists of finding a base $B \in \mathcal{B}_A$ such that $A[R_A, B]^{-1} A[R_A, C_A - B]$ is proper. We have named such B a dynamical base. Combination of Lemma 3.2 and Theorem 3.3 reveals that the set of dynamical bases indeed forms the basis family of a matroid on C_A. This observation suggests a promising research direction toward optimization on and enumeration of dynamical bases using general results obtained in matroid theory [15, 18, 32].

LEMMA 3.4. *For $\mathbf{M}_A = (C_A, \mathcal{B}_A, \omega_A)$ associated with a rational matrix $A(s)$ of row-full rank, the matroid defined by the maximizers of ω_A agrees with the linear matroid defined on C_A by a constant matrix*

$$A^*_B = \lim_{s\to\infty} A[R_A, B]^{-1} A[R_A, C_A],$$

where B is a maximizer of ω_A. (*The matroid defined on C_A by A^*_B is independent of the choice of B.*)

Proof. The proof is immediate from Lemma 3.2. □

Let R and C be disjoint finite sets and $\delta : 2^R \times 2^C \to \mathbf{R} \cup \{-\infty\}$ be a map such that

$$\delta(I, J) = \omega((R - I) \cup J) \qquad (I \subseteq R, J \subseteq C) \tag{3.7}$$

for some valuated matroid $(R \cup C, \omega)$ with $\omega(R) \neq -\infty$. Such triple (R, C, δ) is called a valuated bimatroid in [39]. Define

$$\begin{aligned}
\mathcal{S} &= \{(I, J) \mid |I| = |J|,\ I \subseteq R,\ J \subseteq C\},\\
r &= \max\{|I| \mid \exists (I, J) \in \mathcal{S} : \delta(I, J) \neq -\infty\},\\
\mathcal{S}_k &= \{(I, J) \mid |I| = |J| = k,\ I \subseteq R,\ J \subseteq C\} \qquad (0 \le k \le r),\\
\delta_k &= \max\{\delta(I, J) \mid (I, J) \in \mathcal{S}_k\} \qquad (0 \le k \le r),\\
\mathcal{M}_k &= \{(I, J) \in \mathcal{S}_k \mid \delta(I, J) = \delta_k\} \qquad (0 \le k \le r).
\end{aligned}$$

Note that $\delta(\emptyset, \emptyset) \neq -\infty$, and $\delta(I, J) \neq -\infty$ only if $(I, J) \in \mathcal{S}$.

THEOREM 3.5 (Murota [39]). *The sequence $(\delta_0, \delta_1, \ldots, \delta_r)$ is concave, i.e.,*

$$\delta_{k-1} + \delta_{k+1} \le 2\delta_k \qquad (1 \le k \le r - 1).$$

THEOREM 3.6 (Murota [39]). *For any $(I_k, J_k) \in \mathcal{M}_k$ with $1 \le k \le r - 1$, there exist $(I_l, J_l) \in \mathcal{M}_l$ $(0 \le l \le r,\ l \neq k)$ such that*

$$\begin{aligned}
(\emptyset =)\ I_0 \subseteq I_1 \subseteq \cdots \subseteq I_{k-1} \subseteq I_k \subseteq I_{k+1} \subseteq \cdots \subseteq I_r,\\
(\emptyset =)\ J_0 \subseteq J_1 \subseteq \cdots \subseteq J_{k-1} \subseteq J_k \subseteq J_{k+1} \subseteq \cdots \subseteq J_r.
\end{aligned}$$

Theorem 3.6 justifies the following incremental greedy algorithm for computing δ_k for $k = 0, 1, \ldots, r$. This algorithm involves $O(r|R|\,|C|)$ evaluations of δ to compute the whole sequence $(\delta_0, \delta_1, \cdots, \delta_r)$.

GREEDY ALGORITHM FOR δ_k $(k = 1, 2, \ldots)$.

```
  $I_0 := \emptyset$;  $J_0 := \emptyset$;
  for $k := 1, 2, \ldots$ do.
    Find $i \in R - I_{k-1}$, $j \in C - J_{k-1}$ that maximizes $\delta(I_{k-1}+i, J_{k-1}+j)$
    and put $I_k := I_{k-1} + i$, $J_k := J_{k-1} + j$ and $\delta_k := \delta(I_k, J_k)$.
```

The iteration stops when $\delta(I_k, J_k) = -\infty$, and then $r = k - 1$. See [39] for another algorithm to compute $(\delta_0, \delta_1, \ldots, \delta_r)$.

Given a rational matrix $A(s)$ we can naturally define a valuated bimatroid (R_A, C_A, δ_A) by

$$\delta_A(I, J) = \deg_s \det A[I, J] \qquad (I \subseteq R_A, J \subseteq C_A). \tag{3.8}$$

The associated valuated matroid in (3.7) is given by $\omega_{\widetilde{A}}$ of (3.3) for an $m \times (m+n)$ matrix $\widetilde{A} = (I_m \ A)$. In this special case the nesting property stated in Theorem 3.6 has been observed, though in a slightly weaker form, by Svaricek [53], [54, Satz 6.23] along with the greedy algorithm above.

3.3. Valuated independent assignment problem. The valuated independent assignment problem is defined as follows. Suppose we are given a bipartite graph $G = (V^+, V^-; E)$, valuated matroids $\mathbf{M}^+ = (V^+, \mathcal{B}^+, \omega^+)$ and $\mathbf{M}^- = (V^-, \mathcal{B}^-, \omega^-)$, and a weight function $w : E \to \mathbf{R}$.

VALUATED INDEPENDENT ASSIGNMENT PROBLEM.

Find a matching $M (\subseteq E)$ that maximizes

$$\Omega(M) \equiv w(M) + \omega^+(\partial^+ M) + \omega^-(\partial^- M) \tag{3.9}$$

subject to the constraint

$$\partial^+ M \in \mathcal{B}^+, \quad \partial^- M \in \mathcal{B}^-, \tag{3.10}$$

where $\partial^+ M$ (resp., $\partial^- M$) denotes the set of vertices in V^+ (resp., V^-) incident to M. A matching M satisfying the constraint (3.10) is called an independent assignment. Clearly the two matroids must have the same rank for the feasibility of this problem.

The above problem reduces to the independent assignment problem of Iri and Tomizawa [28] if the valuations are trivial with $\omega^\pm(B) = 0$ for $B \in \mathcal{B}^\pm$, and reduces further to the conventional assignment problem (cf., e.g., Lawler [32]) if the matroids are trivial or free with $\mathcal{B}^\pm = 2^{V^\pm}$.

The following theorem gives an optimality criterion in (1), referring to the existence of a "potential" function, whereas its reformulation in (2) reveals its duality nature. This is a natural extension of the corresponding result [28] for the ordinary (independent) assignment problem.

THEOREM 3.7 (Murota [41]). (1) *An independent assignment M in G is optimal for the valuated independent assignment problem* (3.9)–(3.10) *if and only if there exists a "potential" function $p : V^+ \cup V^- \to \mathbf{R}$ such that*

$$\text{(i)} \qquad w(a) - p(\partial^+ a) + p(\partial^- a) \begin{cases} \le 0 & (a \in E), \\ = 0 & (a \in M). \end{cases}$$

(ii) *$\partial^+ M$ is a maximum-weight base of $\mathbf{M}^+$ with respect to $\omega^+[p^+]$,*

(iii) *$\partial^- M$ is a maximum-weight base of $\mathbf{M}^-$ with respect to $\omega^-[-p^-]$, where $p^\pm$ is the restriction of p to $V^\pm$, and $\omega^+[p^+]$ (resp., $\omega^-[-p^-]$) is the similarity transformation defined in* (3.6); *namely,*

$$\omega^+[p^+](B^+) = \omega^+(B^+) + \sum\{p(u) \mid u \in B^+\} \qquad (B^+ \subseteq V^+),$$
$$\omega^-[-p^-](B^-) = \omega^-(B^-) - \sum\{p(u) \mid u \in B^-\} \qquad (B^- \subseteq V^-).$$

(2)

$$\begin{aligned} &\max_M \{\Omega(M) \mid M\text{: independent assignment}\} \\ &= \min_p \{\max(\omega^+[p^+]) + \max(\omega^-[-p^-]) \mid w(a) - p(\partial^+ a) + p(\partial^- a) \le 0 \ (a \in E)\}. \end{aligned}$$

(3) *If ω^+, ω^-, and w are all integer-valued, the potential p in* (1) *and* (2) *can be chosen to be integer-valued.*

(4) *Let p be a potential that satisfies* (i)–(iii) *above for some* (*optimal*) *independent assignment $M = M_0$. An independent assignment M' is optimal if and only if it satisfies* (i)–(iii) (*with M replaced by M'*).

Remark 3.1 Just as the weighted matroid intersection problem may be regarded as being equivalent to the independent assignment problem, the following problem is equivalent to the valuated independent assignment problem (see [41] for other equivalent problems).

Valuated matroid intersection problem.

> Given a pair of valuated matroids $\mathbf{M}^1 = (V, \mathcal{B}^1, \omega^1)$ and $\mathbf{M}^2 = (V, \mathcal{B}^2, \omega^2)$ defined on a common ground set V, and a weight function $w : V \to \mathbf{R}$, find a common base $B \in \mathcal{B}^1 \cap \mathcal{B}^2$ that maximizes $w(B) + \omega^1(B) + \omega^2(B)$.

The optimality criterion of Theorem 3.7, when adapted to this problem, gives a generalization of the well-known optimality criterion [13, 14, 16, 17, 32] for the weighted matroid intersection problem. □

The duality result in Theorem 3.7 above admits a linear algebraic interpretation for a triple matrix product as follows, although the author is not yet aware of its engineering applications.

Theorem 3.8. *Assume that a matrix product $A(s) = Q_1(s)T(s)Q_2(s)$ is nonsingular, where $Q_1(s)$ (resp., $Q_2(s)$) is a $k \times m$ (resp., $n \times k$) rational matrix over a field $\boldsymbol{K}$, and $T(s)$ is an $m \times n$ rational matrix over an extension field $\boldsymbol{F}$ ($\supseteq \boldsymbol{K}$) such that the set of the coefficients is algebraically independent over $\boldsymbol{K}$. Then there exist $k \times k$ nonsingular rational matrices $S_1(s)$, $S_2(s)$ and diagonal matrices* $\mathrm{diag}\,(s;p) = \mathrm{diag}\,(s^{p_1}, \ldots, s^{p_m})$, $\mathrm{diag}\,(s;q) = \mathrm{diag}\,(s^{q_1}, \ldots, s^{q_n})$ *with $p \in \mathbf{Z}^m$ and $q \in \mathbf{Z}^n$ such that*

$$\deg_s \det A = \deg_s \det S_1 + \deg_s \det S_2$$

and the matrices

$$\begin{aligned}\bar{Q}_1(s) &= S_1(s)^{-1}\, Q_1(s)\, \mathrm{diag}\,(s;p),\\ \bar{T}(s) &= \mathrm{diag}\,(s;-p)\, T(s)\, \mathrm{diag}\,(s;-q),\\ \bar{Q}_2(s) &= \mathrm{diag}\,(s;q)\, Q_2(s)\, S_2(s)^{-1}\end{aligned}$$

are all proper. Note that $S_1(s)^{-1}A(s)S_2(s)^{-1} = \bar{Q}_1(s)\bar{T}(s)\bar{Q}_2(s)$.

Proof. First, by the Cauchy–Binet formula, we have

$$\det A = \sum_{|I|=|J|=k} \pm \det Q_1[*, I] \cdot \det T[I, J] \cdot \det Q_2[J, *],$$

where $Q_1[*, I]$ designates the $k \times k$ submatrix of Q_1 with column set I and the whole row set, and similarly for $Q_2[J, *]$. There is no numerical cancellation in the summation above by virtue of the assumed algebraic independence of the coefficients in $T(s)$, and hence

$$\deg_s \det A = \max_{|I|=|J|=k} \{\deg_s \det Q_1[*, I] + \deg_s \det T[I, J] + \deg_s \det Q_2[J, *]\}.$$

Next, consider a valuated independent assignment problem defined as follows. The vertex sets V^+ and V^- are the row set and the column set of $T(s)$, respectively, and $E = \{(i, j) \mid T_{ij}(s) \neq 0\}$. The valuated matroids attached to V^+ and V^- are those defined by $Q_1(s)$ and $Q_2(s)$ as in (3.3), and the weight w_{ij} of an edge $(i, j) \in E$

is defined by $w_{ij} = \deg_s T_{ij}(s)$. Note that the maximum value of $\sum_{(i,j)\in M} w_{ij}$ over all matchings M with $I = \partial^+ M$ and $J = \partial^- M$ is equal to $\deg_s \det T[I,J]$.

Then we see from the above identity that $\deg_s \det A$ is equal to the maximum value of $\Omega(M)$ over all independent assignment M. Let M be an optimal independent assignment and put $I = \partial^+ M$ and $J = \partial^- M$. Let $\widehat{p} : V^+ \cup V^- \to \mathbf{Z}$ be the potential in Theorem 3.7, and define $p \in \mathbf{Z}^m$ and $q \in \mathbf{Z}^n$ by $p_i = \widehat{p}_i$ for $i \in V^+$ and $q_j = -\widehat{p}_j$ for $j \in V^-$. Define $S_1 = Q_1[*, I]\,\mathrm{diag}\,(s; p_I)$ and $S_2 = \mathrm{diag}\,(s; q_J)\,Q_2[J, *]$, where $p_I \in \mathbf{Z}^I$ is the restriction of p to I and similarly for $q_J \in \mathbf{Z}^J$.

The conditions (i), (ii) and (iii) in Theorem 3.7(1), coupled with Lemma 3.2, imply the properness of $\bar{T}(s)$, $\bar{Q}_1(s)$, and $\bar{Q}_2(s)$, respectively. □

Remark 3.2. The close relationship between the triple matrix product and the independent assignment problem through the Binet-Cauchy formula was first observed by Tomizawa and Iri [55, 56]. To be more precise, the rank of $A = Q_1 T Q_2$ was expressed in [55] as the maximum size of an independent matching, whereas the degree of the determinant of $A(s) = Q_1 T(s) Q_2$ with constant matrices Q_i $(i = 1, 2)$ was represented in [56] as the optimal value of an independent assignment. Our present contribution lies in an extension to the more general case with polynomial matrices $Q_i(s)$ $(i = 1, 2)$ by means of valuated matroids, and also in an explicit statement concerning the transformation into proper matrices. □

4. Degree of mixed polynomial matrix.

4.1. Mixed polynomial matrix.

Let $\boldsymbol{K}$ be a subfield of a field $\boldsymbol{F}$. A matrix A over $\boldsymbol{F}$ (i.e., $A_{ij} \in \boldsymbol{F}$) is called a *mixed matrix* with respect to $\boldsymbol{F}/\boldsymbol{K}$ if

$$A = Q + T, \tag{4.1}$$

where

(M-Q) Q is a matrix over $\boldsymbol{K}$ (i.e., $Q_{ij} \in \boldsymbol{K}$), and

(M-T) T is a matrix over $\boldsymbol{F}$ (i.e., $T_{ij} \in \boldsymbol{F}$) such that the set of its nonzero entries is algebraically independent over $\boldsymbol{K}$.

A mixed matrix A of (4.1) is called a *layered mixed matrix* (or an *LM matrix*) if the nonzero rows of Q and T are disjoint. In other words, A is an LM matrix if it can be put into the following form with a permutation of rows:

$$A = \begin{pmatrix} Q \\ T \end{pmatrix} = \begin{pmatrix} Q \\ O \end{pmatrix} + \begin{pmatrix} O \\ T \end{pmatrix}, \tag{4.2}$$

where Q and T satisfy (M-Q) and (M-T) above, respectively.

Though an LM-matrix is a special case of mixed matrix, the class of LM matrices is as general as the class of mixed matrices both in theory and in application. With an $m \times n$ mixed matrix $A = Q + T$ we associate a $(2m) \times (m+n)$ LM matrix

$$\widetilde{A} = \begin{pmatrix} \widetilde{Q} \\ \widetilde{T} \end{pmatrix} = \begin{pmatrix} I_m & Q \\ -\mathrm{diag}\,(t_1, \ldots, t_m) & T \end{pmatrix}, \tag{4.3}$$

where $\mathrm{diag}\,(t_1, \ldots, t_m)$ is a diagonal matrix with "new" variables $t_1, \ldots, t_m (\in \boldsymbol{F})$. Such transformation often works in the analysis of a mixed matrix by way of an LM-matrix. For example, if we are interested in the rank of A, we may instead compute the rank of $\widetilde{A}$ and use the relation $\mathrm{rank} A = \mathrm{rank} \widetilde{A} - m$.

A polynomial matrix $A(s)$ over $\boldsymbol{F}$ (i.e., $A_{ij}(s) \in \boldsymbol{F}[s]$) is called a *mixed polynomial matrix* with respect to $\boldsymbol{F}/\boldsymbol{K}$ if

$$A(s) = Q(s) + T(s) = \sum_{k=0}^{N} s^k Q_k + \sum_{k=0}^{N} s^k T_k \tag{4.4}$$

for some integer $N \geq 0$, where

(MP-Q) Q_k $(k = 0, 1, \ldots, N)$ are matrices over $\boldsymbol{K}$, and

(MP-T) T_k $(k = 0, 1, \ldots, N)$ are matrices over $\boldsymbol{F}$ such that the set of their nonzero entries is algebraically independent over $\boldsymbol{K}$.

A mixed polynomial matrix with respect to $\boldsymbol{F}/\boldsymbol{K}$ is a mixed matrix with respect to $\boldsymbol{F}(s)/\boldsymbol{K}(s)$. Also note that $A(s) = \sum_{k=0}^{N} s^k A_k$ with $A_k = Q_k + T_k$ and that A_k, for each k, is a mixed matrix with respect to $\boldsymbol{F}/\boldsymbol{K}$.

A mixed polynomial matrix $A(s)$ of (4.4) is called a *layered mixed polynomial matrix* (or an *LM-polynomial matrix*) if the nonzero rows of $Q(s)$ and $T(s)$ are disjoint, that is, if it looks like

$$A(s) = \begin{pmatrix} Q(s) \\ T(s) \end{pmatrix}, \tag{4.5}$$

where $Q(s)$ and $T(s)$ satisfy (MP-Q) and (MP-T) above, respectively. We denote by R_Q and R_T the row sets of $Q(s)$, and $T(s)$, respectively, whereas the column sets of $A(s)$, $Q(s)$, and $T(s)$, are identified and denoted by C. We put $m_Q = |R_Q|$, $m_T = |R_T|$, $n = |C|$. Obviously, an LM-polynomial matrix with respect to $\boldsymbol{F}/\boldsymbol{K}$ is an LM-matrix with respect to $\boldsymbol{F}(s)/\boldsymbol{K}(s)$.

The concepts of (layered) mixed (polynomial) matrices were introduced in Murota and Iri [45], Murota, Iri, and Nakamura [46], and Murota [34] as mathematical tools for the structural/combinatorial analysis of engineering systems. See [36] and [40] for surveys; the former deals with mathematical properties of (layered) mixed matrices while the latter explains engineering motivations.

4.2. Basic identities. We present basic identities concerning the degree of the determinant of (layered) mixed polynomial matrices, which are easy to derive from (MP-Q), (MP-T), and the Laplace expansion of determinants. They will be upgraded in section 5.1 to novel identities of deeper mathematical content.

Recall that R and C denote the row set and the column set of $A(s)$, respectively, and $Q[I, J]$, e.g., denotes the submatrix of Q with row set I and column set J.

THEOREM 4.1 (Murota [33]). *For a square mixed polynomial matrix $A(s) = Q(s) + T(s)$,*

$$\deg_s \det A = \max_{\substack{|I|=|J| \\ I \subseteq R, J \subseteq C}} \{\deg_s \det Q[I, J] + \deg_s \det T[R - I, C - J]\}. \tag{4.6}$$

(*For a singular matrix A both sides are equal to $-\infty$.*)

Proof. By the Laplace expansion [19] we see

$$\det A = \sum_{|I|=|J|} \pm \det Q[I, J] \cdot \det T[R - I, C - J].$$

Since the degree of a sum is bounded by the maximum degree of a summand, we obtain

$$\begin{aligned}\deg_s \det A &\le \max_{|I|=|J|} \deg_s(\det Q[I,J] \cdot \det T[R-I, C-J]) \\ &= \max_{|I|=|J|} \{\deg_s \det Q[I,J] + \deg_s \det T[R-I, C-J]\},\end{aligned}$$

where the inequality turns into an equality provided the highest-degree terms do not cancel one another. The algebraic independence of the nonzero coefficients in $T(s)$ ensures this. □

The above theorem immediately yields a similar identity for an LM-polynomial matrix $A(s) = \binom{Q(s)}{T(s)}$. Recall that R_Q and R_T denote the row sets of $Q(s)$ and $T(s)$, respectively, and C denotes the column sets of $A(s)$, $Q(s)$, and $T(s)$.

THEOREM 4.2. *For a square LM-polynomial matrix* $A(s) = \binom{Q(s)}{T(s)}$,

$$\deg_s \det A = \max_{J \subseteq C} \{\deg_s \det Q[R_Q, J] + \deg_s \det T[R_T, C-J]\}. \tag{4.7}$$

(*For a singular matrix A both sides are equal to $-\infty$.*)

In what follows we focus on an LM-polynomial matrix $A(s)$ and consider a variant of $\delta_k(A)$. Namely, for $A(s) = \binom{Q(s)}{T(s)}$ we define

$$\begin{aligned}\delta_k^{\mathrm{LM}}(A) = \max_{I,\,J} \{&\deg_s \det A[R_Q \cup I, J] \mid \\ &I \subseteq R_T,\ J \subseteq C, |I| = k, |J| = m_Q + k\},\end{aligned} \tag{4.8}$$

where $0 \le k \le \min(m_T, n - m_Q)$. It should be clear that $\delta_k^{\mathrm{LM}}(A)$ designates the highest degree of a minor of order $m_Q + k$ with row set containing R_Q, and that $\delta_k^{\mathrm{LM}}(A) = -\infty$ if there exists no (I, J) that satisfies the conditions on the right-hand side of (4.8). By substituting (4.7) into (4.8) we obtain

$$\begin{aligned}\delta_k^{\mathrm{LM}}(A) = \max_{I,\,J,\,B} \{&\deg_s \det Q[R_Q, B] + \deg_s \det T[I, J-B] \mid \\ &I \subseteq R_T,\ B \subseteq J \subseteq C, |I| = k, |J| = m_Q + k, |B| = m_Q\}.\end{aligned} \tag{4.9}$$

We prefer to work with $\delta_k^{\mathrm{LM}}(A)$ for an LM-polynomial matrix $A(s)$ rather than to deal directly with $\delta_k(A)$ for a mixed polynomial matrix $A(s) = Q(s) + T(s)$. This is because (i) any algorithm for δ_k^{LM} can be used to compute $\delta_k(A)$ for a general mixed polynomial matrix $A(s)$ (as explained below), and (ii) our algorithm description is much simpler for δ_k^{LM}.

The reduction of $\delta_k(A)$ for $A(s) = Q(s) + T(s)$ to $\delta_k^{\mathrm{LM}}(\widetilde{A})$ with an LM-polynomial matrix $\widetilde{A}(s)$ is similar to the transformation (4.3) of a mixed matrix to another LM-matrix. Given an $m \times n$ mixed polynomial matrix $A(s) = Q(s) + T(s)$ we consider a $(2m) \times (m + n)$ LM-polynomial matrix

$$\widetilde{A}(s) = \begin{pmatrix} \widetilde{Q}(s) \\ \widetilde{T}(s) \end{pmatrix} = \begin{pmatrix} \mathrm{diag}\,(s^{d_1}, \ldots, s^{d_m}) & Q(s) \\ -\mathrm{diag}\,(t_1 s^{d_1}, \ldots, t_m s^{d_m}) & T(s) \end{pmatrix}, \tag{4.10}$$

where $t_1, \ldots, t_m$ ($\in \boldsymbol{F}$) are "new" variables, and

$$d_i = \max_{j \in C_A} \deg_s Q_{ij}(s) \qquad (i \in R_A), \tag{4.11}$$

where R_A and C_A denote the row set and the column set of $A(s)$, and hence those of $Q(s)$. The following lemma reveals the relation between $\delta_k(A)$ and $\delta_k^{\mathrm{LM}}(\widetilde{A})$.

LEMMA 4.3. *Let $A(s)$ be a mixed polynomial matrix and $\widetilde{A}(s)$ be the associated LM-polynomial matrix defined by* (4.10)*. Then we have*

$$\delta_k(A) = \delta_k^{\mathrm{LM}}(\widetilde{A}) - \sum_{i=1}^{m} d_i.$$

Proof. Define

$$\widehat{A}(s) = \begin{array}{c} \\ R_Q \\ R_T \end{array} \begin{array}{c} \begin{array}{cc} \qquad R_A \qquad & \qquad C_A \end{array} \\ \begin{pmatrix} \mathrm{diag}\,(s^{d_1}, \ldots, s^{d_m}) & Q(s) \\ -\mathrm{diag}\,(s^{d_1}, \ldots, s^{d_m}) & T(s) \end{pmatrix} \end{array}, \tag{4.12}$$

which is obtained from $\widetilde{A}(s)$ by putting $t_i = 1$ $(i = 1, \ldots, m)$. It is obtained from $\widetilde{A}(s)$ also by dividing the $(m+i)$th row by t_i $(i = 1, \ldots, m)$ and redefining $T_{ij}(s)/t_i$ to be $T_{ij}(s)$. The latter fact implies $\delta_k^{\mathrm{LM}}(\widetilde{A}) = \delta_k^{\mathrm{LM}}(\widehat{A})$. (Here it should be clear that $\delta_k^{\mathrm{LM}}(\widehat{A})$ is defined similarly to (4.8), although this is a slight abuse of notation since $\widehat{A}(s)$ is not an LM-polynomial matrix.)

We denote the row sets (column sets) of A, Q, and T by R_A, R_Q, and R_T (by C_A, C_Q, and C_T), respectively, where R_Q and R_T (C_Q and C_T) have natural one-to-one correspondence with R_A (with C_A). Also we denote the row sets and the column sets of $\widehat{A}$, by $\widehat{R} = R_Q \cup R_T$ and $\widehat{C} = R_A \cup C_A$, as indicated in (4.12).

If $J \supseteq R_A$, we have

$$\deg_s \det \widehat{A}[R_Q \cup I, J] = \deg_s \det A[I, J \cap C_A] + \sum_{i=1}^{m} d_i \qquad (I \subseteq R_T).$$

Hence, taking the maximum of this expression over all I and J with $|I| = |J| - m_Q = k$ and $J \supseteq R_A$, we see that $\delta_k(A) + \sum_{i=1}^{m} d_i$ is equal to

$$\max\{\deg_s \det \widehat{A}[R_Q \cup I, J] \mid I \subseteq R_T,\ R_A \subseteq J \subseteq C, |I| = k, |J| = m_Q + k\}.$$

It remains to be shown that the extra constraint "$J \supseteq R_A$" can be removed without affecting the maximum value. Fix $I \subseteq R_T$ and let $J \subseteq R_A \cup C_A$ be a maximizer of $\deg_s \det \widehat{A}[R_Q \cup I, J]$ satisfying $J \supseteq R_A$. We claim that $\widehat{A}[R_Q \cup I, J]^{-1} \widehat{A}[R_Q \cup I, C_A \setminus J]$ is a proper rational matrix. This claim implies, by Lemma 3.2, that J is an optimum solution to the maximization problem without the constraint "$J \supseteq R_A$."

The claim can be proven as follows. Denoting by I_Q and I_A the copies of I in R_Q and R_A, respectively, we partition the matrix $\widehat{A}[R_Q \cup I, R_A \cup C_A]$ as

$$\widehat{A}[R_Q \cup I, R_A \cup C_A] = \begin{array}{c} \\ R_Q \cap I_Q \\ R_Q \setminus I_Q \\ R_T \cap I \end{array} \begin{array}{c} \begin{array}{cccc} R_A \cap I_A & R_A \setminus I_A & C_A \cap J & C_A \setminus J \end{array} \\ \begin{pmatrix} D_1 & O & Q_{11} & Q_{12} \\ O & D_2 & Q_{21} & Q_{22} \\ -D_1 & O & T_{11} & T_{12} \end{pmatrix} \end{array}$$

with the obvious shorthand notations D_1, Q_{11}, T_{11}, etc. for the relevant submatrices of diag $(s^{d_1}, \ldots, s^{d_m})$, $Q(s)$, $T(s)$, etc. By row transformations we obtain the following sequence of matrices:

$$\begin{aligned}
\begin{pmatrix} D_1 & O & Q_{11} & Q_{12} \\ O & D_2 & Q_{21} & Q_{22} \\ -D_1 & O & T_{11} & T_{12} \end{pmatrix} &\Rightarrow \begin{pmatrix} D_1 & O & Q_{11} & Q_{12} \\ O & D_2 & Q_{21} & Q_{22} \\ O & O & A_{11} & A_{12} \end{pmatrix} \quad (A_{ij} = Q_{ij} + T_{ij}) \\
&\Rightarrow \begin{pmatrix} D_1 & O & Q_{11} & Q_{12} \\ O & D_2 & Q_{21} & Q_{22} \\ O & O & I & {A_{11}}^{-1}A_{12} \end{pmatrix} \\
&\Rightarrow \begin{pmatrix} D_1 & O & O & Q_{12} - Q_{11}{A_{11}}^{-1}A_{12} \\ O & D_2 & O & Q_{22} - Q_{12}{A_{11}}^{-1}A_{12} \\ O & O & I & {A_{11}}^{-1}A_{12} \end{pmatrix} \\
&\Rightarrow \begin{pmatrix} I & O & O & {D_1}^{-1}[Q_{12} - Q_{11}{A_{11}}^{-1}A_{12}] \\ O & I & O & {D_2}^{-1}[Q_{22} - Q_{12}{A_{11}}^{-1}A_{12}] \\ O & O & I & {A_{11}}^{-1}A_{12} \end{pmatrix}.
\end{aligned}$$

This shows

$$\widehat{A}[R_Q \cup I, J]^{-1}\widehat{A}[R_Q \cup I, C_A \setminus J] = \begin{array}{c} \\ R_A \\ \\ C_A \cap J \end{array} \begin{array}{c} C_A \setminus J \\ \left(\begin{array}{c} B_1(s) \\ \\ B_2(s) \end{array}\right) \end{array}$$

with

$$\begin{aligned}
B_1(s) &= \mathrm{diag}\,(s^{-d_1}, \ldots, s^{-d_m})\{Q[R_Q, C_A \setminus J] - Q[R_Q, C_A \cap J]B_2(s)\}, \\
B_2(s) &= A[I, C_A \cap J]^{-1}A[I, C_A \setminus J].
\end{aligned}$$

Here $B_2(s)$ is a proper rational matrix (i.e., each entry has deg ≤ 0) by the choice of J, and $\mathrm{diag}\,(s^{-d_1}, \ldots, s^{-d_m})Q[R_Q, C_A]$ is also proper by the definition (4.11) of d_i. Therefore, $\widehat{A}[R_Q \cup I, J]^{-1}\widehat{A}[R_Q \cup I, C_A \setminus J]$ is a proper rational matrix. □

Example 4.1. For the 2×3 mixed polynomial matrix $A(s)$ in Example 1.1we have $d_1 = 3$, $d_2 = 2$, and

$$\widetilde{A}(s) = \begin{array}{c|cc|ccc|} & r_1 & r_2 & c_1 & c_2 & c_3 \\ \hline r_{Q1} & s^3 & & s^3+1 & s^2 & 1 \\ r_{Q2} & & s^2 & s^2 & s & 0 \\ \hline r_{T1} & -t_1 s^3 & & 0 & \alpha_1 & \alpha_2 s \\ r_{T2} & & -t_2 s^2 & \alpha_3 & 0 & 0 \\ \hline \end{array}.$$

It is easy to verify that

$$\begin{aligned}
\delta_1(A) &= \deg_s \det A[r_1, c_1] = 3, \\
\delta_2(A) &= \deg_s \det A[\{r_1, r_2\}, \{c_1, c_3\}] = 3,
\end{aligned}$$

whereas

$$\begin{aligned}
\delta_1^{\mathrm{LM}}(\widetilde{A}) &= \deg_s \det \widetilde{A}[\{r_{Q1}, r_{Q2}, r_{T1}\}, \{r_1, r_2, c_1\}] = 3 + 5, \\
\delta_2^{\mathrm{LM}}(\widetilde{A}) &= \deg_s \det \widetilde{A}[\{r_{Q1}, r_{Q2}, r_{T1}, r_{T2}\}, \{r_1, r_2, c_1, c_3\}] = 3 + 5. \quad □
\end{aligned}$$

4.3. Reduction to valuated independent assignment. We describe how the computation of $\delta_k^{\mathrm{LM}}(A)$ for an LM-polynomial matrix $A(s) = \binom{Q(s)}{T(s)}$ can be reduced to solving a valuated independent assignment problem. We denote by $\mathbf{M}_Q = (C_Q, \mathcal{B}_Q, \omega_Q)$ the valuated matroid associated with $Q(s)$ as (3.2) and (3.3); namely,

$$\mathcal{B}_Q = \{B \subseteq C_Q \mid \det Q[R_Q, B] \neq 0\}, \tag{4.13}$$

$$\omega_Q(B) = \deg_s \det Q[R_Q, B] \qquad (B \in \mathcal{B}_Q). \tag{4.14}$$

Here and henceforth $C_Q = \{j_Q \mid j \in C\}$ denotes a disjoint copy of the column set C of A (with $j_Q \in C_Q$ denoting the copy of $j \in C$), whereas R_Q and R_T mean, as before, the row sets of $Q(s)$ and $T(s)$, respectively; $|R_Q| = m_Q$, $|R_T| = m_T$, and $|C| = n$.

We consider a valuated independent assignment problem defined on a bipartite graph $G = (V^+, V^-; E)$ with $V^+ = R_T \cup C_Q$, $V^- = C$, and $E = E_T \cup E_Q$, where

$$E_T = \{(i,j) \mid i \in R_T, j \in C, T_{ij}(s) \neq 0\}, \quad E_Q = \{(j_Q, j) \mid j \in C\}.$$

The valuated matroids $\mathbf{M}^+ = (V^+, \mathcal{B}^+, \omega^+)$ and $\mathbf{M}^- = (V^-, \mathcal{B}^-, \omega^-)$ attached to V^+ and V^- are defined by

$$\mathcal{B}^+ = \{B^+ \subseteq V^+ \mid B^+ \cap C_Q \in \mathcal{B}_Q, |B^+ \cap R_T| = k\},$$
$$\mathcal{B}^- = \{B^- \subseteq V^- \mid |B^-| = m_Q + k\}$$

and

$$\omega^+(B^+) = \omega_Q(B^+ \cap C_Q) \qquad (B^+ \in \mathcal{B}^+),$$
$$\omega^-(B^-) = 0 \qquad (B^- \in \mathcal{B}^-).$$

The weight w_{ij} of an edge $(i,j) \in E$ is defined by

$$w_{ij} = \begin{cases} \deg_s T_{ij}(s) & ((i,j) \in E_T), \\ 0 & ((i,j) \in E_Q). \end{cases} \tag{4.15}$$

Note the dependence of $\mathbf{M}^\pm$ on k as well as the independence of G and w of k.

We then have the following characterization of $\delta_k^{\mathrm{LM}}(A)$ in terms of the optimal value of the valuated independent assignment problem. Recall the notation $\Omega(M)$ of (3.9) for the value of an independent assignment M.

THEOREM 4.4. *For an LM-polynomial matrix $A(s) = \binom{Q(s)}{T(s)}$ and an integer k with $0 \leq k \leq \min(m_T, n - m_Q)$, $\delta_k^{\mathrm{LM}}(A)$ of (4.8) coincides with the optimal value of the valuated independent assignment problem defined above. That is,*

$$\delta_k^{\mathrm{LM}}(A) = \max\{\Omega(M) \mid M\text{: } \textit{independent assignment}\}, \tag{4.16}$$

where the right-hand side is defined to be $-\infty$ if there exists no independent assignment M.

Proof. Define

$$\Delta(I, J, B) = \deg_s \det Q[R_Q, B] + \deg_s \det T[I, J - B], \tag{4.17}$$

which is the function to be maximized in the expression (4.9) for $\delta_k^{\mathrm{LM}}(A)$. By virtue of the algebraic independence of the nonzero coefficients in $T(s)$, the second term, $\deg_s \det T[I, J-B]$, is equal to the maximum weight (with respect to w_{ij}) of a matching of size $|I| = |J - B|$ in the bipartite graph $(R_T, C; E_T)$ that covers I and $J - B$.

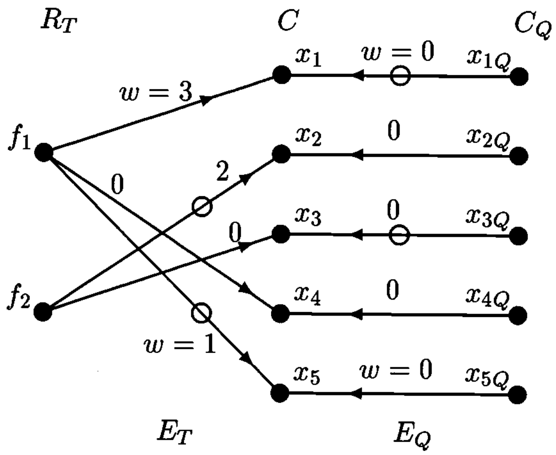


FIG. 1. *Graph* G (○*: arcs in* M, $B = \{x_{1Q}, x_{3Q}\}$).

Given (I, J, B) with $|I| = k$ and $\Delta(I, J, B) > -\infty$, we can construct an independent assignment M such that

$$I = \partial^+(M \cap E_T), \quad J = \partial^- M, \quad B = \partial^+(M \cap E_Q), \tag{4.18}$$

and that $M \cap E_T$ is a maximum weight k-matching in the graph $(R_T, C; E_T)$ that covers I and $J - B$. Note that $\det Q[R_Q, B] \neq 0$ and $|I| = k$ if and only if $B \cup I \in \mathcal{B}^+$. Moreover, $\omega^+(B \cup I) = \deg_s \det Q[R_Q, B]$ by the definition, and therefore we have $\Delta(I, J, B) = \Omega(M)$. Conversely, an independent assignment M with $\Omega(M)$ maximum determines (I, J, B), as above, for which $\Delta(I, J, B) = \Omega(M)$ holds true. Hence the maximum value of $\Delta(I, J, B)$ is equal to that of $\Omega(M)$. □

EXAMPLE 4.2. The valuated independent assignment problem associated with a 4×5 LM-polynomial matrix

$$A(s) = \begin{array}{c|ccccc|} & x_1 & x_2 & x_3 & x_4 & x_5 \\ \cline{2-6} & s^3 & 0 & s^3+1 & s^2 & 1 \\ & 0 & s^2 & s^2 & s & 0 \\ \cline{2-6} f_1 & -t_1 s^3 & 0 & 0 & \alpha_1 & \alpha_2 s \\ f_2 & 0 & -t_2 s^2 & \alpha_3 & 0 & 0 \\ \cline{2-6} \end{array} \tag{4.19}$$

with $k = 2$ is illustrated in Fig. 1. This matrix is essentially the same as $\widetilde{A}(s)$ in Example 4.1, but the columns and the rows are now indexed as $C = \{x_1, x_2, x_3, x_4, x_5\}$ and $R_T = \{f_1, f_2\}$ and accordingly $C_Q = \{x_{1Q}, x_{2Q}, x_{3Q}, x_{4Q}, x_{5Q}\}$. An optimal independent assignment $M = \{(f_1, x_5), (f_2, x_2), (x_{1Q}, x_1), (x_{3Q}, x_3)\}$ is marked by ○. We have $I = \partial^+(M \cap E_T) = \{f_1, f_2\}$, $J = \partial^- M = \{x_1, x_2, x_3, x_5\}$, $B = \partial^+(M \cap E_Q) = \{x_{1Q}, x_{3Q}\} \in \mathcal{B}_Q$, $\omega_Q(B) = 5$, $w(M) = 1 + 2 = 3$, and therefore $\Omega(M) = 5 + 3 = 8$, which agrees with $\delta_2^{\mathrm{LM}}(A) = 8$. In Section 5.2 we will come back to this example to explain the algorithm. □

4.4. Novel identities. The basic identities on the degree of subdeterminants presented in section 4.2 are upgraded here to novel identities of duality nature. They are obtained from the duality result (Theorem 3.7) on the valuated independent assignment problem, i.e., the optimality criterion involving a potential function. Besides this, the potential will be used extensively also in the algorithm in section 5, and furthermore it will play a crucial role in section 4.5 in determining the coefficient of the highest-degree term of $\det A(s)$.

Consider the valuated independent assignment problem associated with $A(s)$, introduced in section 4.3. Let M be an optimal independent assignment, (I, J, B) be defined by (4.18), and $\widehat{p} : R_T \cup C \cup C_Q \to \mathbf{Z}$ be a potential function guaranteed in Theorem 3.7.

We may assume that $\widehat{p}(j_Q) = \widehat{p}(j)$ for $j \in C$ (where $j_Q \in C_Q$ denotes the copy of $j \in C$). To see this, first note that $\widehat{p}(j_Q) \geq \widehat{p}(j)$ for $j \in C$ and the equality holds if $(j_Q, j) \in M$. For $j \in C$ with $(j_Q, j) \notin M$, we can redefine $\widehat{p}(j_Q)$ to $\widehat{p}(j)$, since (j_Q, j) is the only arc going out of j_Q and its weight $w_{j_Q j}$ is zero. Define $q \in \mathbf{Z}^{R_T}$ and $p \in \mathbf{Z}^C$ by

$$q_i = \widehat{p}(i) \quad (i \in R_T), \qquad p_j = -\widehat{p}(j) \quad (j \in C). \tag{4.20}$$

Then the conditions (i)–(iii) in Theorem 3.7(1) are expressed as follows:

$$\deg_s T_{ij}(s) \leq q_i + p_j \quad ((i,j) \in E_T), \tag{4.21}$$

$$\deg_s T_{ij}(s) = q_i + p_j \quad ((i,j) \in M \cap E_T), \tag{4.22}$$

$$\omega_Q[-p](B) = \max_{B' \in \mathcal{B}_Q} \omega_Q[-p](B'), \tag{4.23}$$

$$q(I) = \max_{|I'|=k} q(I'), \tag{4.24}$$

$$p(J) = \max_{|J'|=m_Q+k} p(J'), \tag{4.25}$$

where $q(I) = \sum_{i \in} q_i$ and $p(J) = \sum_{j \in J} p_j$. These conditions imply

$$\begin{aligned} \delta_k^{\mathrm{LM}}(A) &= \deg_s \det Q[R_Q, B] + \deg_s \det T[I, J - B] \\ &= \omega_Q(B) + q(I) + p(J - B) \\ &= \omega_Q[-p](B) + q(I) + p(J) \\ &= \max_{B' \in \mathcal{B}_Q} \omega_Q[-p](B') + \max_{|I'|=k} q(I') + \max_{|J'|=m_Q+k} p(J'). \end{aligned} \tag{4.26}$$

Thus we obtain the following theorem.

THEOREM 4.5. *For an LM-polynomial matrix $A(s) = \left(\begin{smallmatrix} Q(s) \\ T(s) \end{smallmatrix}\right)$ and an integer k such that $\delta_k^{\mathrm{LM}}(A) > -\infty$, the following identity holds true:*

$$\delta_k^{\mathrm{LM}}(A) = \min_{q_i + p_j \geq \deg_s T_{ij}} \left[\max_{|I|=k} q(I) + \max_{|J|=m_Q+k} p(J) + \max_{B \in \mathcal{B}_Q} \omega_Q[-p](B) \right], \tag{4.27}$$

where the minimum is taken over all $q \in \mathbf{Z}^{R_T}$, $p \in \mathbf{Z}^C$ satisfying $q_i + p_j \geq \deg_s T_{ij}$ for all (i, j).

Proof. Let (I, J, B) be as above. For any (q', p') with $q'_i + p'_j \geq \deg_s T_{ij}$ $(\forall (i,j))$, we have

$$\begin{aligned} \delta_k^{\mathrm{LM}}(A) &= \deg_s \det Q[R_Q, B] + \deg_s \det T[I, J - B] \\ &\leq \omega_Q(B) + q'(I) + p'(J - B) \\ &= \omega_Q[-p'](B) + q'(I) + p'(J) \\ &\leq \max_{B' \in \mathcal{B}_Q} \omega_Q[-p'](B') + \max_{|I'|=k} q'(I') + \max_{|J'|=m_Q+k} p'(J'), \end{aligned}$$

whereas the inequalities turn into equalities for $(q', p') = (q, p)$, as in (4.26). □

With q, p, and B above, we can transform the matrix $A(s)$ to another LM-polynomial matrix that is somehow canonical with respect to δ_k^{LM}. Let $S(s) = (Q[R_Q, B] \cdot \mathrm{diag}\,(s; -p_B))^{-1}$, where p_B is the restriction of p to B, and define

$$\widehat{A}(s) = \begin{pmatrix} \widehat{Q}(s) \\ \widehat{T}(s) \end{pmatrix} = \begin{pmatrix} S(s) & O \\ O & \mathrm{diag}\,(s; -q) \end{pmatrix} \cdot A(s) \cdot \mathrm{diag}\,(s; -p). \tag{4.28}$$

The conditions (4.21)–(4.23) mean that

$$\widehat{A}(s) = \begin{array}{c} \\ R_Q \\ I \\ R_T - I \end{array} \!\!\begin{array}{c} \begin{array}{ccc} B & J-B & C-J \end{array} \\ \begin{pmatrix} I_{m_Q} & Q_2'(s) & Q_3'(s) \\ T_1'(s) & T_2^{\bullet}(s) & T_3'(s) \\ T_1''(s) & T_2''(s) & T_3''(s) \end{pmatrix} \end{array} \tag{4.29}$$

is a proper matrix, in which $T_2^{\bullet}(s)$ admits a transversal consisting of entries of degree zero. Obviously,

$$\delta_k^{\mathrm{LM}}(\widehat{A}) = \max_{|B'|=m_Q} \deg_s \det \widehat{Q}[R_Q, B'] + \max_{|I'|=|J'|=k} \deg_s \det \widehat{T}[I', J'], \tag{4.30}$$

in which all the three terms are equal to zero. From this we obtain the following theorem.

Theorem 4.6. *For an LM-polynomial matrix $A(s) = \binom{Q(s)}{T(s)}$ and an integer k such that $\delta_k^{\mathrm{LM}}(A) > -\infty$, there exist $p \in \mathbf{Z}^C$ and $q \in \mathbf{Z}^{R_T}$ such that*

$$\begin{aligned} \bar{A}(s) &= \begin{pmatrix} I_{m_Q} & O \\ O & \mathrm{diag}\,(s; -q) \end{pmatrix} \cdot A(s) \cdot \mathrm{diag}\,(s; -p), \\ \bar{Q}(s) &= Q(s) \cdot \mathrm{diag}\,(s; -p), \\ \bar{T}(s) &= \mathrm{diag}\,(s; -q) \cdot T(s) \cdot \mathrm{diag}\,(s; -p) \end{aligned}$$

satisfy

$$\delta_k^{\mathrm{LM}}(\bar{A}) = \max_{|B|=m_Q} \deg_s \det \bar{Q}[R_Q, B] + \max_{|I|=|J|=k} \deg_s \det \bar{T}[I, J]. \tag{4.31}$$

Here we can impose on $\bar{A}$ an additional condition:

$$\delta_k^{\mathrm{LM}}(\bar{A}) = \delta_k^{\mathrm{LM}}(A) - \max_{|I|=k} q(I) - \max_{|J|=m_Q+k} p(J). \tag{4.32}$$

It should be clear that $\bar{A}(s) = \binom{\bar{Q}(s)}{\bar{T}(s)}$, which is also an LM-polynomial matrix.

Proof. The first identity (4.31) follows from (4.30). The second identity (4.32) is due to (4.26) combined with $\delta_k^{\mathrm{LM}}(\bar{A}) = \delta_k^{\mathrm{LM}}(\widehat{A}) + \deg_s \det S^{-1} = \omega_Q[-p](B)$. □

Example 4.3. We illustrate the above argument for the LM-polynomial matrix $A(s)$ of (4.19) with $k = 2$. The vectors $p \in \mathbf{Z}^C$ and $q \in \mathbf{Z}^{R_T}$ of (4.20) are given by $p = (-1, -1, -3, -4, -3)$ and $q = (4, 3)$. Accordingly we have

$$\bar{A}(s) = \begin{array}{c|ccccc|} & x_1 & x_2 & x_3 & x_4 & x_5 \\ \cline{2-6} & s^4 & 0 & s^6 + s^3 & s^6 & s^3 \\ & 0 & s^3 & s^5 & s^5 & 0 \\ \cline{2-6} f_1 & -t_1 & 0 & 0 & \alpha_1 & \alpha_2 \\ f_2 & 0 & -t_2 & \alpha_3 & 0 & 0 \\ \cline{2-6} \end{array}$$

for which (4.31) holds true with $\delta_2^{\mathrm{LM}}(\bar{A}) = 9 = 9 + 0$. Recall from Example 4.3 that $I = \{f_1, f_2\}$, $J = \{x_1, x_2, x_3, x_5\}$, $B = \{x_{1Q}, x_{3Q}\}$. The matrix $\widehat{A}(s)$ of (4.28) is equal to

$$\widehat{A}(s) = \begin{array}{c|cc|cc|c|} & x_1 & x_3 & x_5 & x_2 & x_4 \\ \cline{2-6} & 1 & 0 & \frac{1}{s} & \frac{-s^3-1}{s^3} & \frac{-1}{s} \\ & 0 & 1 & 0 & \frac{1}{s^2} & 1 \\ \cline{2-6} f_1 & -t_1 & 0 & \alpha_2 & 0 & \alpha_1 \\ f_2 & 0 & \alpha_3 & 0 & -t_2 & 0 \\ \cline{2-6} \end{array}. \tag{4.33}$$

In section 5.2 we will come back to this example and explain how the vectors p and q can be found (see the variable p in Fig. 4, in particular). $\square$

As corollaries to Theorem 4.6 we obtain the following two theorems, which should be compared to Theorem 4.2 and Theorem 4.1, respectively.

THEOREM 4.7. *For a nonsingular LM-polynomial matrix $A(s) = \binom{Q(s)}{T(s)}$, there exists $p \in \mathbf{Z}^C$ such that*

$$\bar{A}(s) = A(s) \cdot \mathrm{diag}\,(s; -p),\ \bar{Q}(s) = Q(s) \cdot \mathrm{diag}\,(s; -p),\ \bar{T}(s) = T(s) \cdot \mathrm{diag}\,(s; -p)$$

satisfy

$$\deg_s \det \bar{A} = \max_{|B|=|R_Q|} \deg_s \det \bar{Q}[R_Q, B] + \max_{|J|=|R_T|} \deg_s \det \bar{T}[R_T, J], \tag{4.34}$$

where it should be clear that $\bar{A}(s) = \binom{\bar{Q}(s)}{\bar{T}(s)}$, which is also an LM-polynomial matrix.

Proof. Apply Theorem 4.6 with $k = m_T = n - m_Q$ to obtain $p \in \mathbf{Z}^C$. The row transformation by $\mathrm{diag}\,(s; -q)$ is not necessary in the case of $k = m_T$. $\square$

THEOREM 4.8. *For a nonsingular mixed polynomial matrix $A(s) = Q(s) + T(s)$, there exist $p_R \in \mathbf{Z}^R$ and $p_C \in \mathbf{Z}^C$ such that*

$$\begin{aligned} \bar{A}(s) &= \mathrm{diag}\,(s; -p_R) \cdot A(s) \cdot \mathrm{diag}\,(s; -p_C), \\ \bar{Q}(s) &= \mathrm{diag}\,(s; -p_R) \cdot Q(s) \cdot \mathrm{diag}\,(s; -p_C), \\ \bar{T}(s) &= \mathrm{diag}\,(s; -p_R) \cdot T(s) \cdot \mathrm{diag}\,(s; -p_C) \end{aligned}$$

satisfy

$$\deg_s \det \bar{A} = \max_{\substack{|I|=|J| \\ I \subseteq R,\, J \subseteq C}} \deg_s \det \bar{Q}[I, J] + \max_{\substack{|I|=|J| \\ I \subseteq R,\, J \subseteq C}} \deg_s \det \bar{T}[R - I, C - J], \tag{4.35}$$

where it should be clear that $\bar{A}(s) = \bar{Q}(s) + \bar{T}(s)$, which is also a mixed polynomial matrix.

Proof. Apply Theorem 4.7 to the associated LM-polynomial matrix (4.10) to obtain $\widehat{p} \in \mathbf{Z}^{R \cup C}$. Denote by $\widehat{p}_R$ and $\widehat{p}_C$ the restrictions of $\widehat{p}$ to R and to C, respectively. Then put $p_R = d - \widehat{p}_R$ and $p_C = \widehat{p}_C$, where $d \in \mathbf{Z}^R$ is the vector of exponents in (4.10). $\square$

4.5. Leading coefficient. For a nonsingular LM-polynomial matrix $A(s)$ with respect to $\boldsymbol{F}/\boldsymbol{K}$, $\det A(s)$ is a polynomial in s with coefficients from $\boldsymbol{F}$. If we denote by $\mathcal{T}$ the set of nonzero coefficients in $T(s)$, we see that the coefficients in $\det A(s)$ are polynomials in $\mathcal{T}$ over $\boldsymbol{K}$. Recall that $\mathcal{T} \subseteq \boldsymbol{F}$ and $\mathcal{T}$ is algebraically independent over $\boldsymbol{K}$ (cf. (MP-T) in section 4.1).

In this section we are interested in the leading coefficient (= the coefficient of the highest-degree term) of $\det A(s)$, which we denote by $\eta(\mathcal{T}) \in \boldsymbol{K}[\mathcal{T}]$ (= ring of polynomials in $\mathcal{T}$ over $\boldsymbol{K}$). Namely,

$$\eta(\mathcal{T}) = \text{l.c.}(\det A(s)) = \lim_{s\to\infty} s^{-\delta_n(A)} \det A(s),$$

where l.c.$(\cdot)$ means the leading coefficient of a polynomial in s, and n is the size of the matrix. The following argument shows, among others, that we can determine which variables of $\mathcal{T}$ appear in $\eta(\mathcal{T})$ by means of arithmetic operations in $\boldsymbol{K}(s)$ without involving $\mathcal{T}$.

Recall $\widehat{A}(s)$ of (4.28), where $I = R_T$ and $J = C$, and define a constant matrix

$$\widehat{A}^* = \lim_{s\to\infty} \widehat{A}(s), \tag{4.36}$$

which is a nonsingular LM-matrix with respect to $\boldsymbol{F}/\boldsymbol{K}$. It is noted that the matrix $\widehat{A}^*$ depends on the choice of (q, p, B). Since

$$\det A(s) = \det Q[R_Q, B] \cdot \det \widehat{A}(s) \cdot s^{q(R_T)+p(C-B)}, \quad \det \widehat{A}(s) = \det \widehat{A}^* + o(1),$$

where $o(1)$ denotes a term that vanishes as $s \to \infty$, we see that

$$\eta(\mathcal{T}) = c \cdot \det \widehat{A}^*, \qquad c = \text{l.c.}(\det Q[R_Q, B]) \in \boldsymbol{K}. \tag{4.37}$$

For an LM-matrix in general, a block-triangular canonical form is known to exist (Murota [34], Murota, Iri, and Nakamura [46]). The canonical form is called the combinatorial canonical form (CCF) of the LM-matrix, and can be computed in $O(n^3 \log n)$ time with arithmetic operations in the subfield. Furthermore, it is known [35] that the factorization of the determinant of an LM-matrix is given through the irreducible diagonal components of its CCF. See [36] for a survey on the CCF.

Using these general results we see the following.

1. A variable $t \in \mathcal{T}$ appears in $\eta(\mathcal{T})$ if and only if t is contained in an irreducible diagonal block of the CCF of the LM-matrix $\widehat{A}^*$. Hence, once $\widehat{A}(s)$ is known, the set of variables of $\mathcal{T}$ that are contained in $\eta(\mathcal{T})$ can be computed in $O(n^3 \log n)$ time with arithmetic operations in $\boldsymbol{K}$.
2. The irreducible factors of $\eta(\mathcal{T})$, as a polynomial in $\mathcal{T}$ over $\boldsymbol{K}$, are given by the determinants of the irreducible components in the CCF of $\widehat{A}^*$. Hence, once $\widehat{A}(s)$ is known, the irreducibility of $\eta(\mathcal{T})$ in $\boldsymbol{K}[\mathcal{T}]$ can be determined in $O(n^3 \log n)$ time with arithmetic operations in $\boldsymbol{K}$.

Example 4.4. Consider the 4×4 submatrix of the LM-polynomial matrix $A(s)$ of (4.19) with column set $J = \{x_1, x_2, x_3, x_5\}$. We have $\mathcal{T} = \{t_1, t_2, \alpha_2, \alpha_3\}$. By direct calculation we see

$$\det A[R_A, J] = -\alpha_2 t_2 s^8 - t_1 t_2 s^7 - \alpha_2 \alpha_3 s^6 - t_1 \alpha_3 s^5,$$

and therefore $\eta(\mathcal{T}) = -\alpha_2 t_2$. On the other hand, the matrix $\widehat{A}^*$ (cf. (4.33)) and its CCF (which happens to be triangular) are given by

$$\widehat{A}^* = \begin{array}{c|cccc|} & x_1 & x_2 & x_3 & x_5 \\ \cline{2-5} & 1 & -1 & 0 & 0 \\ & 0 & 0 & 1 & 0 \\ \cline{2-5} f_1 & -t_1 & 0 & 0 & \alpha_2 \\ f_2 & 0 & -t_2 & \alpha_3 & 0 \\ \cline{2-5} \end{array}, \qquad \text{CCF} = \begin{array}{c|c|c|c|c|} & x_5 & x_1 & x_2 & x_3 \\ \cline{2-5} f_1 & \alpha_2 & -t_1 & 0 & 0 \\ \cline{2-5} & & 1 & -1 & 0 \\ \cline{2-5} f_2 & & & -t_2 & \alpha_3 \\ \cline{2-5} & & & & 1 \\ \cline{2-5} \end{array}.$$

We have $\det \widehat{A}^* = -\alpha_2 t_2$, in agreement with $\eta(\mathcal{T})$ (up to a constant factor). The variables, α_2 and t_2, appearing in the diagonal blocks agree with those variables contained in $\eta(\mathcal{T})$. □

5. Algorithm.

5.1. Algorithm description.

In section 4.3 we have explained how to reduce the computation of $\delta_k^{\mathrm{LM}}(A)$ to solving a valuated independent assignment problem. Here we will provide an algorithm for $\delta_k^{\mathrm{LM}}(A)$ by adapting the augmenting algorithm of Murota [42] for a general valuated independent assignment problem. Our algorithm computes $\delta_k^{\mathrm{LM}}(A)$ successively for $k = 0, 1, 2, \ldots, k_{\max}$, where $k_{\max}$ is the maximum k with $\delta_k^{\mathrm{LM}}(A) > -\infty$.

As described in section 4.3, the associated valuated independent assignment problem is defined on the bipartite graph $G = (V^+, V^-; E) = (R_T \cup C_Q, C; E_T \cup E_Q)$, where C_Q is a disjoint copy of C (with $j_Q \in C_Q$ denoting the copy of $j \in C$), and

$$E_T = \{(i,j) \mid i \in R_T, j \in C, T_{ij}(s) \neq 0\}, \quad E_Q = \{(j_Q, j) \mid j \in C\}.$$

The algorithm maintains a pair (M, B) of a matching $M \subseteq E_T \cup E_Q$ and a base $B \in \mathcal{B}_Q$ $(\subseteq 2^{C_Q})$ that maximizes

$$\Omega''(M,B) \equiv w(M) + \omega_Q(B) = w(M \cap E_T) + \omega_Q(B) \tag{5.1}$$

subject to the constraint that $\partial^+(M \cap E_Q) = B$ and M is of a specified size. We put

$$M_T = M \cap E_T, \qquad M_Q = M \cap E_Q.$$

With reference to (M, B) it constructs an auxiliary directed graph $\widetilde{G} = \widetilde{G}_{(M,B)} = (\widetilde{V}, \widetilde{E})$ with vertex set $\widetilde{V} = R_T \cup C_Q \cup C$ and arc set $\widetilde{E} = E_T \cup E_Q \cup E^+ \cup M^\circ$, where

$$\begin{aligned} E^+ &= \{(i_Q, j_Q) \mid i_Q \in B, j_Q \in C_Q - B, B - i_Q + j_Q \in \mathcal{B}_Q\}, \\ M^\circ &= \{\overline{a} \mid a \in M\} \qquad (\overline{a}\text{: reorientation of } a). \end{aligned}$$

It should be emphasized that the arcs in E^+ have both ends in C_Q and that the arcs in M° are directed from C to $R_T \cup C_Q$, i.e., $\partial^+ M^\circ \subseteq C$ and $\partial^- M^\circ \subseteq R_T \cup C_Q$. We put

$$\begin{aligned} M_T^\circ &= \{a \in M^\circ \mid \partial^- a \in R_T\} = \{\overline{a} \mid a \in M_T\}, \\ M_Q^\circ &= \{a \in M^\circ \mid \partial^- a \in C_Q\} = \{\overline{a} \mid a \in M_Q\}. \end{aligned}$$

We define the entrance $S^+ \subseteq \widetilde{V}$ and the exit $S^- \subseteq \widetilde{V}$ by

$$S^+ = R_T - \partial^+ M_T = R_T - \partial^- M_T^\circ, \qquad S^- = C - \partial^- M = C - \partial^+ M^\circ.$$

Note that no vertex in C_Q belongs to the entrance S^+.

We define the arc length $\gamma = \gamma_{(M,B)} : \widetilde{E} \to \mathbf{Z}$ by

$$\gamma(a) = \begin{cases} -\deg_s T_{ij}(s) & (a = (i,j) \in E_T), \\ \;\;\deg_s T_{ij}(s) & (a = (j,i) \in M_T^\circ), \\ -\omega_Q(B, i_Q, j_Q) & (a = (i_Q, j_Q) \in E^+), \\ 0 & (a \in E_Q \cup M_Q^\circ), \end{cases} \tag{5.2}$$

where $\omega_Q(B, i_Q, j_Q) = \omega_Q(B - i_Q + j_Q) - \omega_Q(B)$, compatibly with the notation (3.4). By Lemma 3.2 we can compute $\omega_Q(B, i_Q, j_Q)$ by means of pivoting operations on $Q(s)$, namely, for $P(s) = S(s)Q(s)$ with $S(s) = Q[R_Q, B]^{-1}$ we have $\omega_Q(B, i_Q, j_Q) = \deg_s P_{ij}(s)$.

Suppose there is a shortest path in $\widetilde{G}_{(M,B)}$ from the entrance S^+ to the exit S^- with respect to the arc length γ, and let L be (the set of arcs on) a shortest path from S^+ to S^- having the smallest number of arcs. Then we can update (M, B) to $(\overline{M}, \overline{B})$ by

$$\overline{M} = M - \{a \in M \mid \overline{a} \in L \cap M^\circ\} + (L \cap (E_T \cup E_Q)), \tag{5.3}$$

$$\overline{B} = B - \{\partial^+ a \mid a \in L \cap E^+\} + \{\partial^- a \mid a \in L \cap E^+\}. \tag{5.4}$$

In fact, $\overline{M}$ is obviously a matching with $\partial^+(\overline{M} \cap E_Q) = \overline{B}$ and $|\overline{M}| = |M| + 1$, and furthermore, it can be shown [42] that $\overline{B} \in \mathcal{B}_Q$ and $(\overline{M}, \overline{B})$ maximizes $\Omega''(\overline{M}, \overline{B})$ under these constraints.

Our algorithm for $\delta_k^{\mathrm{LM}}(A)$ repeats finding a shortest path and updating (M, B) as follows.

Outline of the algorithm.

```
Starting from a maximum-weight base $B \in \mathcal{B}_Q$ with respect to $\omega_Q$
and the corresponding matching $M = \{(j_Q, j) \mid j_Q \in B\}$, repeat
(i)–(ii) below:
(i) Find a shortest path $L$ having the smallest number of arcs (from
      $S^+$ to $S^-$ in $\widetilde{G}_{(M,B)}$ with respect to the arc length $\gamma_{(M,B)}$).
      [Stop if there is no path from $S^+$ to $S^-$.]
(ii) Update $(M, B)$ according to (5.3) and (5.4).
```

An initial base B of maximum value of ω_Q can be found by the greedy algorithm described in section 3.2. At each stage of this algorithm it holds that $\delta_k^{\mathrm{LM}}(A) = \Omega''(M, B)$ for $k = |M| - m_Q$ and that (I, J, B) defined by (4.18) gives the maximum in the expression (4.9) of $\delta_k^{\mathrm{LM}}(A)$.

Just as the primal-dual algorithm for the ordinary minimum-cost flow problem and the independent assignment problem, the algorithm outlined above can be made more efficient by the explicit use of a potential function on the auxiliary graph $\widetilde{G} = (\widetilde{V}, \widetilde{E})$. To this end we maintain $p : \widetilde{V} \to \mathbf{Z}$ that satisfies

$$-\deg_s T_{ij}(s) + p(i) - p(j) \geq 0 \quad ((i,j) \in E_T), \tag{5.5}$$

$$-\deg_s T_{ij}(s) + p(i) - p(j) = 0 \quad ((i,j) \in M_T), \tag{5.6}$$

$$p(j_Q) - p(j) \geq 0 \quad (j \in C), \tag{5.7}$$

$$p(j_Q) - p(j) = 0 \quad ((j_Q, j) \in M_Q), \tag{5.8}$$

$$-\omega_Q(B, i_Q, j_Q) + p(i_Q) - p(j_Q) \geq 0 \quad ((i_Q, j_Q) \in E^+), \tag{5.9}$$

$$p(i) - p(k) \geq 0 \quad (i \in R_T, k \in S^+), \tag{5.10}$$

$$p(k) - p(j) \geq 0 \quad (j \in C, k \in S^-). \tag{5.11}$$

It is remarked that the existence of such p implies the optimality of (M, B) with respect to Ω'' of (5.1). In fact, for any (M', B') with $|M'| = |M|$ and $\partial^+(M' \cap E_Q) = B'$ we have

$$\begin{aligned} w(M') + \omega_Q(B') &= w_p(M') + \omega_Q[p_Q](B') + p(\partial^+ M'_T) - p(\partial^- M') \\ &\leq \omega_Q[p_Q](B) + p(\partial^+ M_T) - p(\partial^- M) \\ &= w(M) + \omega_Q(B), \end{aligned}$$

where $M'_T = M' \cap E_T$, $w_p(a) = w(a) - p(\partial^+ a) + p(\partial^- a)$, and p_Q denotes the restriction of p to C_Q. Note that $w_p(M') \leq w_p(M) = 0$ by (5.5)–(5.8), $\omega_Q[p_Q](B') \leq \omega_Q[p_Q](B)$ by (5.9) and Lemma 3.1, $p(\partial^+ M'_T) \leq p(\partial^+ M_T)$ by (5.10), and $p(\partial^- M') \geq p(\partial^- M)$ by (5.11).

Initially, we have $M_T = \emptyset$, $S^+ = R_T$, and $S^- = C$, and therefore we can put

$$p(i) = \max_{k \in R_T, j \in C} \deg_s T_{kj}(s) \quad (i \in R_T), \qquad p(j) = p(j_Q) = 0 \quad (j \in C) \tag{5.12}$$

to meet the conditions (5.5)–(5.11). In general steps, p is updated to

$$\overline{p}(v) = p(v) + \Delta p(v) \qquad (v \in \widetilde{V}) \tag{5.13}$$

based on the length $\Delta p(v)$ of the shortest path from S^+ to v with respect to the modified arc length

$$\gamma_p(a) = \gamma(a) + p(\partial^+ a) - p(\partial^- a) \geq 0 \qquad (a \in \widetilde{E}), \tag{5.14}$$

where the nonnegativity of γ_p is due to (5.5)–(5.11). It can be shown [42] that $\overline{p}$ satisfies the conditions (5.5)–(5.11).

To compute $\omega_Q(B, i_Q, j_Q)$ we use two matrices (or two-dimensional arrays) $P = P(s)$ and $S = S(s)$, as well as a vector (or one-dimensional array) *base*. The array P represents an $m_Q \times n$ matrix of rational functions in s over $\boldsymbol{K}$, where $P = Q$ at the beginning of the algorithm (Step 1 below). The other array S is an $m_Q \times m_Q$ matrix of rational functions in s over $\boldsymbol{K}$, which is set to the unit (identity) matrix in Step 1. Variable *base* is a vector of size m_Q, which represents a mapping (correspondence): $R_Q \to C \cup \{0\}$. We also use a scalar (integer-valued) variable δ_Q to compute $\omega_Q(B)$.

The following algorithm computes $\delta_k^{\mathrm{LM}}(A)$ for $k = 0, 1, 2, \ldots, k_{\max}$, as well as the value of $k_{\max}$, where $k_{\max} = -1$ by convention, if $\mathrm{rank} Q(s) < m_Q$.

Algorithm for $\delta_k^{\mathrm{LM}}(A)$ $(k = 0, 1, 2, \ldots, k_{\max})$.

```
Step 1: (Initialize)
        $M := \emptyset$;   $B := \emptyset$;   $\delta_Q := 0$;
        $base[i] := 0$ $(i \in R_Q)$;   $P[i,j] := Q_{ij}$ $(i \in R_Q, j \in C)$;
        $S :=$ unit matrix of order $m_Q$;
        $p[i] := \max_{k \in R_T, j \in C} \deg_s T_{kj} (i \in R_T)$; $p[j] := p[j_Q] := 0$ $(j \in C)$ (cf. (5.12)).
Step 2: (Find $B \in \mathcal{B}_Q$ that maximizes $\omega_Q$)
        While $|B| < m_Q$ do
            {Find $(h, j)$ that maximizes $\deg_s P[h, j]$
             s.t. $base[h] = 0$, $j_Q \notin B$, and $P[h, j] \neq 0$;
             If there exists no such $(h, j)$, then stop with $k_{\max} := -1$;
             $B := B + j_Q$;   $\delta_Q := \delta_Q + \deg_s P[h, j]$;   $M := M + (j_Q, j)$;
             $base[h] := j$;   $w := 1/P[h, j]$;
             $P[h, l] := w \times P[h, l]$   $(l \in C)$;   $S[h, l] := w \times S[h, l]$   $(l \in R_Q)$;
             $P[m, l] := P[m, l] - P[m, j] \times P[h, l]$   $(h \neq m \in R_Q, l \in C)$;
             $S[m, l] := S[m, l] - P[m, j] \times S[h, l]$   $(h \neq m \in R_Q, l \in R_Q)$ };
        $k := 0$.
Step 3: (Construct the auxiliary graph $\widetilde{G}_{(M,B)}$)
        $\delta_k^{\mathrm{LM}}(A) := \delta_Q + \sum_{(i,j) \in M \cap E_T} \deg_s T_{ij}$ ;
        $S^+ := R_T - \partial^+(M \cap E_T)$;   $S^- := C - \partial^- M$;   $M^\circ := \{\overline{a} \mid a \in M\}$;
        $E^+ := \{(i_Q, j_Q) \mid h \in R_Q, j_Q \in B, P[h, j] \neq 0, i = base[h]\}$;
```

$$\gamma(a) := \begin{cases} -\deg_s T_{ij}(s) & (a = (i,j) \in E_T) \\ \deg_s T_{ij}(s) & (a = (j,i) \in M_T^\circ) \\ -\deg_s P[h,j] & (a = (i_Q, j_Q) \in E^+, base[h] = i) \\ 0 & (a \in E_Q \cup M_Q^\circ) \end{cases} \qquad \text{(cf. (5.2))}$$

where $M_T^\circ = \{\overline{a} \mid a \in M \cap E_T\}$, $M_Q^\circ = \{\overline{a} \mid a \in M \cap E_Q\}$;

$\gamma_p(a) := \gamma(a) + p(\partial^+ a) - p(\partial^- a) \qquad (a \in \widetilde{E}).$ (cf. (5.14))

```
Step 4: (Augment $M$ along a shortest path)
    For each $v \in \widetilde{V}$ compute the length $\Delta p(v)$ of the shortest path from $S^+$ to $v$
    in $\widetilde{G}_{(M,B)}$ with respect to the modified arc length $\gamma_p$;
    If there is no path from $S^+$ to $S^-$ (including the case where $S^+ = \emptyset$ or
    $S^- = \emptyset$), then stop with $k_{\max} := k$;
    Let $L$ ($\subseteq E$) be (the set of arcs on) a shortest path, having the smallest
    number of arcs, from $S^+$ to $S^-$ with respect to the modified arc length $\gamma_p$;
    $M := M - \{a \in M \mid \overline{a} \in L \cap M^\circ\} + (L \cap (E_T \cup E_Q))$ ;    $k := k + 1$;
    $p[v] := p[v] + \Delta p(v) \quad (v \in \widetilde{V})$;                                  (cf. (5.13))
    For all $(i_Q, j_Q) \in L \cap E^+$ (in the order from $S^+$ to $S^-$ along $L$) do the
    following:
        {Find $h$ such that $i = base[h]$;
        $B := B - i_Q + j_Q$;    $\delta_Q := \delta_Q + \deg_s P[h,j]$;
        $base[h] := j$;    $w := 1/P[h,j]$;
        $P[h,l] := w \times P[h,l] \quad (l \in C)$;    $S[h,l] := w \times S[h,l] \quad (l \in R_Q)$;
        $P[m,l] := P[m,l] - P[m,j] \times P[h,l] \quad (h \neq m \in R_Q, l \in C)$;
        $S[m,l] := S[m,l] - P[m,j] \times S[h,l] \quad (h \neq m \in R_Q, l \in R_Q)$ };
    Go to Step 3.    □
```

The shortest path in Step 4 can be found in time linear in the size of the graph $\widetilde{G}$, which is $O((|R| + |C|)^2)$, by means of the standard graph algorithms; see, e.g., [1].

For the updates of P in Steps 2 and 4, the algorithm assumes the availability of arithmetic operations on rational functions in a single variable s over the subfield $\boldsymbol{K}$. It is emphasized that no arithmetic operations are done on the T-part, so that no rational function operations involving coefficients in T (which are independent symbols) are needed.

The updates of P are the standard pivoting operations [19], the total number of which is bounded by $O(|R|^2 |C|\, k_{\max})$. Note that pivoting operations are required for each arc $(i_Q, j_Q) \in L \cap E^+$ (see Step 4). The sparsity of P should be taken into account in actual implementations of the algorithm.

The matrix $S(s)$ gives the inverse of $Q[R_Q, B]$, which is often useful (see, e.g., the proof of Theorem 4.6). When $S(s)$ is not needed, it may simply be eliminated from the computation without any side effect.

5.2. Example. The algorithm above is illustrated here for the 4×5 LM-polynomial matrix $A(s)$ of (4.19):

$$A(s) = \begin{array}{c|ccccc|} & x_1 & x_2 & x_3 & x_4 & x_5 \\ \cline{2-6} & s^3 & 0 & s^3+1 & s^2 & 1 \\ & 0 & s^2 & s^2 & s & 0 \\ \cline{2-6} f_1 & -t_1 s^3 & 0 & 0 & \alpha_1 & \alpha_2 s \\ f_2 & 0 & -t_2 s^2 & \alpha_3 & 0 & 0 \\ \cline{2-6} \end{array} . \tag{5.15}$$

We work with a 2×5 matrix $P(s)$, a 2×2 matrix $S(s)$, a vector *base* of size 2, and another vector p of size 12.

The flow of computation is traced below.

Step 1. $M := \emptyset$; $B := \emptyset$; $\delta_Q := 0$;

$$(base, P, S) := \begin{array}{c|c|} \cline{2-2} r_1 & 0 \\ r_2 & 0 \\ \cline{2-2} \end{array}\,, \quad \begin{array}{c|ccccc|} & x_1 & x_2 & x_3 & x_4 & x_5 \\ \cline{2-6} r_1 & s^3 & 0 & s^3+1 & s^2 & 1 \\ r_2 & 0 & s^2 & s^2 & s & 0 \\ \cline{2-6} \end{array}\,, \quad \begin{array}{|cc|} \hline 1 & 0 \\ 0 & 1 \\ \hline \end{array}\,;$$

$$p := \begin{array}{|cc|ccccc|ccccc|} f_1 & f_2 & x_1 & x_2 & x_3 & x_4 & x_5 & x_{1Q} & x_{2Q} & x_{3Q} & x_{4Q} & x_{5Q} \\ \hline 3 & 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline \end{array}\,.$$

Step 2. $(h, j) := (r_1, x_1)$; $B := \{x_{1Q}\}$, $\delta_Q := 3$; $M := \{(x_{1Q}, x_1)\}$;

$$(base, P, S) := \begin{array}{c|c|} \cline{2-2} r_1 & x_1 \\ r_2 & 0 \\ \cline{2-2} \end{array}\,, \quad \begin{array}{c|ccccc|} & x_1 & x_2 & x_3 & x_4 & x_5 \\ \cline{2-6} r_1 & 1 & 0 & \frac{s^3+1}{s^3} & \frac{1}{s} & \frac{1}{s^3} \\ r_2 & 0 & s^2 & s^2 & s & 0 \\ \cline{2-6} \end{array}\,, \quad \begin{array}{|cc|} \hline \frac{1}{s^3} & 0 \\ 0 & 1 \\ \hline \end{array}\,;$$

$(h, j) := (r_2, x_2)$; $B := \{x_{1Q}, x_{2Q}\}$, $\delta_Q := 5$; $M := \{(x_{1Q}, x_1), (x_{2Q}, x_2)\}$;

$$(base, P, S) := \begin{array}{c|c|} \cline{2-2} r_1 & x_1 \\ r_2 & x_2 \\ \cline{2-2} \end{array}\,, \quad \begin{array}{c|ccccc|} & x_1 & x_2 & x_3 & x_4 & x_5 \\ \cline{2-6} r_1 & 1 & 0 & \frac{s^3+1}{s^3} & \frac{1}{s} & \frac{1}{s^3} \\ r_2 & 0 & 1 & 1 & \frac{1}{s} & 0 \\ \cline{2-6} \end{array}\,, \quad \begin{array}{|cc|} \hline \frac{1}{s^3} & 0 \\ 0 & \frac{1}{s^2} \\ \hline \end{array}\,;$$

$k := 0$.

Step 3. $\delta_0^{\mathrm{LM}}(A) := 5$; $S^+ := \{f_1, f_2\}$; $S^- := \{x_3, x_4, x_5\}$;
$M^\circ := \{(x_1, x_{1Q}), (x_2, x_{2Q})\}$;
$E^+ := \{(x_{1Q}, x_{3Q}), (x_{1Q}, x_{4Q}), (x_{1Q}, x_{5Q}), (x_{2Q}, x_{3Q}), (x_{2Q}, x_{4Q})\}$;
γ and γ_p are given in $G^{(0)}$ of Fig. 2. (See $G^{(0)}$ in Fig. 2.)

Step 4.

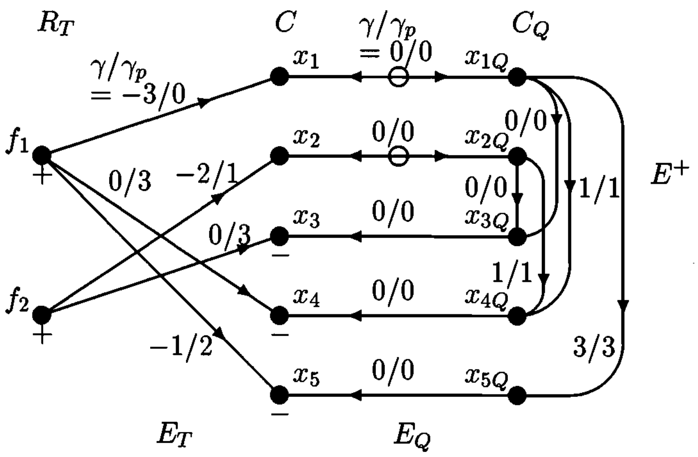
$$\Delta p := \begin{array}{|cc|ccccc|ccccc|} f_1 & f_2 & x_1 & x_2 & x_3 & x_4 & x_5 & x_{1Q} & x_{2Q} & x_{3Q} & x_{4Q} & x_{5Q} \\ \hline 0 & 0 & 0 & 1 & 0 & 1 & 2 & 0 & 1 & 0 & 1 & 3 \\ \hline \end{array}\,;$$

There exists a path from S^+ to S^-;
$L := \{(f_1, x_1), (x_1, x_{1Q}), (x_{1Q}, x_{3Q}), (x_{3Q}, x_3)\}$;
$M := \{(f_1, x_1), (x_{2Q}, x_2), (x_{3Q}, x_3)\}$; $k := 1$;

$$p := \begin{array}{|cc|ccccc|ccccc|} f_1 & f_2 & x_1 & x_2 & x_3 & x_4 & x_5 & x_{1Q} & x_{2Q} & x_{3Q} & x_{4Q} & x_{5Q} \\ \hline 3 & 3 & 0 & 1 & 0 & 1 & 2 & 0 & 1 & 0 & 1 & 3 \\ \hline \end{array}\,;$$

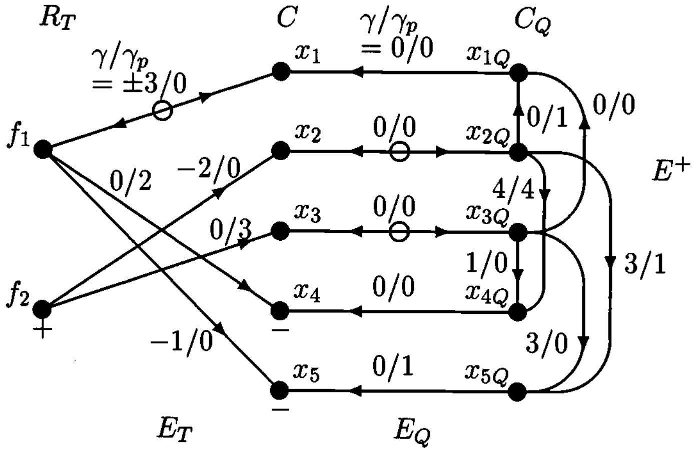
$(i_Q, j_Q) := (x_{1Q}, x_{3Q}) \in L \cap E^+$; $h := r_1$; $B := \{x_{3Q}, x_{2Q}\}$, $\delta_Q := 5$;

$$(base, P, S) := \begin{array}{c|c|} \cline{2-2} r_1 & x_3 \\ r_2 & x_2 \\ \cline{2-2} \end{array}\,, \quad \begin{array}{c|ccccc|} & x_1 & x_2 & x_3 & x_4 & x_5 \\ \cline{2-6} r_1 & \frac{s^3}{s^3+1} & 0 & 1 & \frac{s^2}{s^3+1} & \frac{1}{s^3+1} \\ r_2 & -\frac{s^3}{s^3+1} & 1 & 0 & \frac{1}{s(s^3+1)} & \frac{-1}{s^3+1} \\ \cline{2-6} \end{array}\,, \quad \begin{array}{|cc|} \hline \frac{1}{s^3+1} & 0 \\ \frac{-1}{s^3+1} & \frac{1}{s^2} \\ \hline \end{array}\,.$$



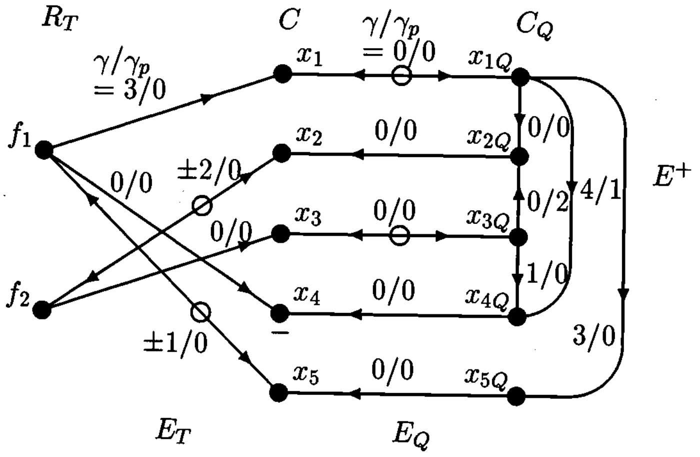
v	f_1	f_2	x_1	x_2	x_3	x_4	x_5	x_{1Q}	x_{2Q}	x_{3Q}	x_{4Q}	x_{5Q}
p	3	3	0	0	0	0	0	0	0	0	0	0
Δp	0	0	0	1	0	1	2	0	1	0	1	3

FIG. 2. *Graph* $G^{(0)}$ ($\bigcirc$: *arcs in* M, $B = \{x_{1Q}, x_{2Q}\}$, $S^+ = \{f_1, f_2\}$, $S^- = \{x_3, x_4, x_5\}$).



v	f_1	f_2	x_1	x_2	x_3	x_4	x_5	x_{1Q}	x_{2Q}	x_{3Q}	x_{4Q}	x_{5Q}
p	3	3	0	1	0	1	2	0	1	0	1	3
Δp	1	0	1	0	3	3	1	1	0	3	3	1

FIG. 3. *Graph* $G^{(1)}$ ($\bigcirc$: *arcs in* M, $B = \{x_{2Q}, x_{3Q}\}$, $S^+ = \{f_2\}$, $S^- = \{x_4, x_5\}$).



v	f_1	f_2	x_1	x_2	x_3	x_4	x_5	x_{1Q}	x_{2Q}	x_{3Q}	x_{4Q}	x_{5Q}
p	4	3	1	1	3	4	3	1	1	3	4	4

FIG. 4. *Graph* $G^{(2)}$ ($\bigcirc$: *arcs in* M, $B = \{x_{1Q}, x_{3Q}\}$, $S^+ = \emptyset$, $S^- = \{x_4\}$).

Step 3. $\delta_1^{\mathrm{LM}}(A) := 5 + 3 = 8$; $S^+ := \{f_2\}$; $S^- := \{x_4, x_5\}$;
$M^\circ := \{(x_1, f_1), (x_2, x_{2Q}), (x_3, x_{3Q})\}$;
$E^+ := \{(x_{2Q}, x_{1Q}), (x_{2Q}, x_{4Q}), (x_{2Q}, x_{5Q}), (x_{3Q}, x_{1Q}), (x_{3Q}, x_{4Q}), (x_{3Q}, x_{5Q})\}$;
γ and γ_p are given in $G^{(1)}$ of Fig. 3. (See $G^{(1)}$ in Fig. 3.)

Step 4.

$$\Delta p := \begin{array}{cc|ccccc|ccccc} f_1 & f_2 & x_1 & x_2 & x_3 & x_4 & x_5 & x_{1Q} & x_{2Q} & x_{3Q} & x_{4Q} & x_{5Q} \\ \hline 1 & 0 & 1 & 0 & 3 & 3 & 1 & 1 & 0 & 3 & 3 & 1 \end{array};$$

There exists a path from S^+ to S^-;
$L := \{(f_2, x_2), (x_2, x_{2Q}), (x_{2Q}, x_{1Q}), (x_{1Q}, x_1), (x_1, f_1), (f_1, x_5)\}$;
$M := \{(f_1, x_5), (f_2, x_2), (x_{1Q}, x_1), (x_{3Q}, x_3)\}$; $k := 2$;

$$p := \begin{array}{cc|ccccc|ccccc} f_1 & f_2 & x_1 & x_2 & x_3 & x_4 & x_5 & x_{1Q} & x_{2Q} & x_{3Q} & x_{4Q} & x_{5Q} \\ \hline 4 & 3 & 1 & 1 & 3 & 4 & 3 & 1 & 1 & 3 & 4 & 4 \end{array};$$

$(i_Q, j_Q) := (x_{2Q}, x_{1Q}) \in L \cap E^+$; $h := r_2$; $B := \{x_{3Q}, x_{1Q}\}$, $\delta_Q := 5$;

$$(base, P, S) := \begin{array}{c|c|} & \\ r_1 & x_3 \\ r_2 & x_1 \end{array}, \quad \begin{array}{c|ccccc|} & x_1 & x_2 & x_3 & x_4 & x_5 \\ r_1 & 0 & 1 & 1 & \frac{1}{s} & 0 \\ r_2 & 1 & \frac{-s^3-1}{s^3} & 0 & \frac{-1}{s^4} & \frac{1}{s^3} \end{array}, \quad \begin{array}{|cc|} 0 & \frac{1}{s^2} \\ \frac{1}{s^3} & \frac{-s^3-1}{s^5} \end{array}.$$

Step 3. $\delta_2^{\mathrm{LM}}(A) := 5 + 3 = 8$; $S^+ := \emptyset$; $S^- := \{x_4\}$;
$M^\circ := \{(x_5, f_1), (x_2, f_2), (x_1, x_{1Q}), (x_3, x_{3Q})\}$;
$E^+ := \{(x_{1Q}, x_{2Q}), (x_{1Q}, x_{4Q}), (x_{1Q}, x_{5Q}), (x_{3Q}, x_{2Q}), (x_{3Q}, x_{4Q})\}$;
γ and γ_p are given in $G^{(2)}$ of Fig. 4. (See $G^{(2)}$ in Fig. 4.)

Step 4. There exists no path from S^+ ($= \emptyset$) to S^-;
Stop with $k_{\max} := 2$. □

Acknowledgments. The author expresses his thanks to Satoru Iwata for discussion and to Vincent Hovelaque, Akiyoshi Shioura, and Jacob van der Woude for comments.

REFERENCES

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison–Wesley, Reading, MA, 1974.

[2] K. E. Brenan, S. L. Campbell, and L. R. Petzold, *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, North–Holland, New York, 1989.

[3] R. A. Brualdi and H. J. Ryser, *Combinatorial Matrix Theory*, Cambridge University Press, London, 1991.

[4] P. Bujakiewicz, *Maximum Weighted Matching for High Index Differential Algebraic Equations*, Doctoral thesis, Delft University of Technology, The Netherlands, 1994.

[5] C. Commault and J.-M. Dion, *Structure at infinity of linear multivariable systems: A geometric approach*, IEEE Trans. Automat. Control, AC-27 (1982), pp. 693–696.

[6] C. Commault, J.-M. Dion, A. Perez, *Disturbance rejection for structured systems*, IEEE Trans. Automat. Control, AC–36 (1991), pp. 884–887.

[7] A. W. M. Dress and W. Terhalle, *Well-layered maps and the maximum-degree $k \times k$-subdeterminant of a matrix of rational functions*, Appl. Math. Lett., 8 (1995), pp. 19–23.

[8] A. W. M. Dress and W. Terhalle, *Well-layered maps – A class of greedily optimizable set functions*, Appl. Math. Lett., 8 (1995), pp. 77–80.

[9] A. W. M. Dress and W. Terhalle, *Rewarding maps – On greedy optimization of set functions*, Adv. Appl. Math., 16 (1995), pp. 464–483.

[10] A. W. M. Dress and W. Wenzel, *Valuated matroid: A new look at the greedy algorithm*, Appl. Math. Lett., 3 (1990), pp. 33–35.

[11] A. W. M. Dress and W. Wenzel, *Valuated matroids*, Adv. Math., 93 (1992), pp. 214–250.

[12] J. Edmonds, *Systems of distinct representatives and linear algebra*, J. Res. Nat. Bur. Stand., 71B (1967), pp. 241–245.

[13] J. Edmonds, *Submodular functions, matroids and certain polyhedra*, in Combinatorial Structures and Their Applications, R. Guy, H. Hanani, N. Sauer, and J. Schönheim, eds., Gordon and Breach, New York, 1970, pp. 69–87.

[14] J. Edmonds, *Matroid intersection*, Ann. Discrete Math., 14 (1979), pp. 39–49.

[15] U. Faigle, *Matroids in combinatorial optimization*, in Combinatorial Geometries, N. White, ed., Cambridge University Press, London, 1987, pp. 161–210.

[16] A. Frank, *A weighted matroid intersection algorithm*, J. Algorithms, 2 (1981), pp. 328–336.

[17] S. Fujishige, *A primal approach to the independent assignment problem*, J. Oper. Res. Soc. Japan, 20 (1977), pp. 1–15.

[18] S. Fujishige, *Submodular Functions and Optimization*, Ann. Discrete Math. 47, North–Holland, Amsterdam, 1991.

[19] F. R. Gantmacher, *The Theory of Matrices*, Chelsea, New York, 1959.

[20] C. W. Gear, *Differential-algebraic equation index transformations*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 39–47.

[21] C. W. Gear, *Differential algebraic equations, indices, and integral algebraic equations*, SIAM J. Numer. Anal., 27 (1990), pp. 1527–1534.

[22] I. Gohberg, P. Lancaster, and L. Rodman, *Matrix Polynomials*, Academic Press, New York, 1982.

[23] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations* II, Springer-Verlag, Berlin, 1991.

[24] M. L. J. Hautus, *The formal Laplace transform for smooth linear systems*, in Mathematical Systems Theory, Lecture Notes in Econom. and Math. Systems 131, G. Marchesini and S. K. Mitter, eds., Springer-Verlag, Berlin, 1976, pp. 29–47.

[25] Y. Hayakawa, S. Hosoe, and M. Ito, *Dynamical degree and controllability for linear systems with intermediate standard form*, Trans. Inst. Electron. Comm. Engrs. Japan, J64A (1981), pp. 752–759 (in Japanese).

[26] V. Hovelaque, C. Commault, and J.-M. Dion, *Analysis of linear structured systems using primal-dual algorithm*, Systems Control Lett., 27 (1996), pp. 73–85.

[27] M. Iri, *Applications of matroid theory*, in Mathematical Programming – The State of the Art, A. Bachem, M. Grötschel, and B. Korte, eds., Springer-Verlag, Berlin, 1983, pp. 158–201.

[28] M. Iri and N. Tomizawa, *An algorithm for finding an optimal "independent assignment,"* J. Oper. Res. Soc. Japan, 19 (1976), pp. 32–57.

[29] S. Iwata and K. Murota, *Combinatorial Relaxation Algorithm for Mixed Polynomial Matrices*, RIMS Preprint 1113, Kyoto University, Kyoto, Japan, 1996.

[30] S. Iwata, K. Murota, and I. Sakuta, *Primal-dual combinatorial relaxation algorithms for the maximum degree of subdeterminants*, SIAM J. Sci. Comput., 17 (1996), pp. 993–1012.

[31] B. Korte, L. Lovász, and R. Schrader, *Greedoids*, Springer-Verlag, Berlin, 1991.

[32] E. L. Lawler, *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston, New York, 1976.

[33] K. Murota, *Use of the concept of physical dimensions in the structural approach to systems analysis*, Japan J. Appl. Math., 2 (1985), pp. 471–494.

[34] K. Murota, *Systems Analysis by Graphs and Matroids – Structural Solvability and Controllability*, Algorithms Combin. 3, Springer-Verlag, Berlin, 1987.

[35] K. Murota, *On the irreducibility of layered mixed matrices*, Linear and Multilinear Algebra, 24 (1989), pp. 273–288.

[36] K. Murota, *Mixed matrices – Irreducibility and decomposition*, in Combinatorial and Graph-Theoretical Problems in Linear Algebra, IMA Vol. Math. Appl. 50, R. A. Brualdi, S. Friedland, and V. Klee, eds., Springer-Verlag, New York, 1993, pp. 39–71.

[37] K. Murota, *Computing the degree of determinants via combinatorial relaxation*, SIAM J. Comput., 24 (1995), pp. 765–796.

[38] K. Murota, *Combinatorial relaxation algorithm for the maximum degree of subdeterminants: Computing Smith–McMillan form at infinity and structural indices in Kronecker form*, Appl. Algebra Engrg., Comm. Comput., 6 (1995), pp. 251–273.

[39] K. Murota, *Finding optimal minors of valuated bimatroids*, Appl. Math. Lett., 8 (1995), pp. 37–42.

[40] K. Murota, *Structural approach in systems analysis by mixed matrices – An exposition for index of DAE*, in ICIAM 95 (Proceedings of the Third International Congress on Industrial and Applied Mathematics held in Hamburg, Germany, July 3–7, 1995), Math. Res. 87, K. Kirchgässner, O. Mahrenholtz, and R. Mennicken, eds., Akademie Verlag, Berlin, 1996, pp. 257–279.

[41] K. Murota, *Valuated matroid intersection,* I: *Optimality criteria*, SIAM J. Discrete Math., 9 (1996), pp. 545–561.

[42] K. Murota, *Valuated matroid intersection,* II: *Algorithms*, SIAM J. Discrete Math., 9 (1996), pp. 562–576.

[43] K. Murota, *Convexity and Steinitz's exchange property*, Adv. Math., 124 (1996), pp. 272–311. Extended abstract in Integer Programming and Combinatorial Optimization (Proceedings of 5th International IPCO Conference, Vancouver, British Columbia, Canada, June 3–5, 1996), Lecture Notes in Comput. Sci. 1084, W. H. Cunningham, S. T. McCormick, and M. Queyranne, eds., Springer-Verlag, New York, 1996, pp. 260–274.

[44] K. Murota, *Characterizing a valuated delta-matroid as a family of delta-matroids*, J. Oper. Res. Soc. Japan, 40 (1997), pp. 565–578.

[45] K. Murota and M. Iri, *Structural solvability of systems of equations – A mathematical formulation for distinguishing accurate and inaccurate numbers in structural analysis of systems*, Japan J. Appl. Math., 2 (1985), pp. 247–271.

[46] K. Murota, M. Iri and M. Nakamura, *Combinatorial canonical form of layered mixed matrices and its application to block-triangularization of systems of linear/nonlinear equations*, SIAM J. Algebraic Discrete Meth., 8 (1987), pp. 123–149.

[47] K. Murota and J. W. van der Woude, *Structure at infinity of structured descriptor systems and its applications*, SIAM J. Control Optim., 29 (1991), pp. 878–894.

[48] M. Newman, *Integral Matrices*, Academic Press, London, 1972.

[49] A. Recski, *Matroid Theory and Its Applications in Electric Network Theory and in Statics*, Algorithms Combin. 6, Springer-Verlag, Berlin, 1989.

[50] K. J. Reinschke, *Multivariable Control, A Graph-Theoretic Approach*, Springer-Verlag, New York, 1988.

[51] H. H. Rosenbrock, *State-Space and Multivariable Theory*, Nelson, London, 1970.

[52] N. Suda, B. Wan, and I. Ueno, *The orders of infinite zeros of structured systems,* Trans. Soc. Instrument Control Engrs. Japan, 25 (1989), pp. 1062–1068 (in Japanese).

[53] F. Svaricek, *An improved graph theoretic algorithm for computing the structure at infinity of linear systems*, in Proceedings of the 29th Conference on Decision and Control, Honolulu, HI, December 1990, pp. 2923–2924.

[54] F. Svaricek, *Zuverlässige numerische Analyse linearer Regelungssysteme*, Teubner, Stuttgart, 1995.

[55] N. Tomizawa and M. Iri, *An algorithm for determining the rank of a triple matrix product AXB with applications to the problem of discerning the existence of the unique solution*

in a network, Trans. Inst. Electron. Comm. Engrs. Japan, 57A (1974), pp. 834–841 (in Japanese). English translation in Electron. Comm. Japan, 57A (1974), pp. 50–57.

[56] N. Tomizawa and M. Iri, *An algorithm for solving the "independent assignment" problem with application to the problem of determining the order of complexity of a network*, Trans. Inst. Electron. Comm. Engrs. Japan, 57A (1976), pp. 627–629 (in Japanese).

[57] J. Ungar, A. Kröner, and W. Marquardt, *Structural analysis of differential-algebraic equation systems: Theory and application*, Comput. Chemical Engrg., 19 (1995), pp. 867–882.

[58] G. C. Verghese and T. Kailath, *Rational matrix structure*, IEEE Trans. Automat. Control, AC-26 (1981), pp. 434–439.

[59] M. Vidyasagar, *Control System Synthesis: A Factorization Approach*, MIT Press, Cambridge, MA, 1985.

[60] D. J. A. Welsh, *Matroid Theory*, Academic Press, London, 1976.

[61] N. White, ed., *Theory of Matroids*, Cambridge University Press, London, 1986.

[62] J. C. Willems, *From time series to linear systems, Part* I: *Finite dimensional linear time invariant systems*, Automatica, 22 (1986), pp. 561–580.

[63] J. C. Willems, *Dynamical systems, controllability, and observability: A post-modern point of view*, in Mathematical System Theory, A. C. Antoulas, ed., Springer-Verlag, Berlin, 1991, pp. 17–37.

[64] J. W. van der Woude, *Causality relations in linear structured systems*, in Comm. Control Signal Processing, E. Arikan, ed., Elsevier, New York, 1990, pp. 772–778.

[65] J. W. van der Woude, *On the structure at infinity of a structured system*, Linear Algebra Appl., 148 (1991), pp. 145–169.

[66] J. W. van der Woude, *The generic dimension of a minimal realization of an AR-system*, Math. Control Signals Systems, 8 (1995), pp. 50–64.

SIAM J. Matrix Anal. Appl.
Vol. 20, No. 1, pp. 228–243

NONSYMMETRIC ALGEBRAIC RICCATI EQUATIONS AND HAMILTONIAN-LIKE MATRICES[*]

JONQ JUANG[†] AND WEN-WEI LIN[‡]

Abstract. We consider a nonsymmetric algebraic matrix Riccati equation arising from transport theory. The nonnegative solutions of the equation can be explicitly constructed via the inversion formula of a Cauchy matrix. An error analysis and numerical results are given. We also show a comparison theorem of the nonnegative solutions.

Key words. Hamiltonian, algebraic Riccati equation, M-matrices, nonnegative matrices, transport theory

AMS subject classifications. 15A24, 82C70

PII. S0895479897318253

1. Introduction.

In transport theory, a variation of the usual one-group neutron transport equation [2, 8, 10] is formulated as

$$\left\{(\mu+\alpha)\frac{\partial}{\partial x}+1\right\}\varphi(x,\mu)=\frac{c}{2}\int_{-1}^{1}\varphi(x,\omega)d\omega, \tag{1a}$$

$$\varphi(0,\mu)=f(\mu),\ \mu>-\alpha, |\mu|\le 1, \tag{1b}$$

$$\lim_{x\to\infty}\varphi(x,\mu)=0. \tag{1c}$$

Here φ is the neutron flux, α $(0\le\alpha<1)$ is an angular shift, and c is the average of the total number of particles emerging from a collision, which is assumed to be conservation; i.e., $0\le c\le 1$.

The scattering function (see, e.g., [15]) for particle transport (or radiative transfer) in the half-space can be derived from (1) via invariant embedding [2]. Such a scattering function satisfies the following integrodifferential equation (see the appendix in [15]):

$$\left(\frac{1}{\mu+\alpha}+\frac{1}{\nu-\alpha}\right)X(\mu,\nu)=c\left[1+\frac{1}{2}\int_{-\alpha}^{1}\frac{X(\omega,\nu)}{\omega+\alpha}d\omega\right]\left[1+\frac{1}{2}\int_{\alpha}^{1}\frac{X(\mu,\omega)}{\omega-\alpha}d\omega\right], \tag{2}$$

with $(\mu,\nu)\in[-\alpha,1]\times[\alpha,1]$. Here the function $X:[-\alpha,1]\times[\alpha,1]\to\mathbf{R}$ is called a scattering function. For the case in which $c=0$ or $\alpha=1$, (2) has a trivial solution. When $\alpha=0$, the existence of nonnegative solutions of (2) has been studied by many authors (see, e.g., [15] and the works cited therein). In fact, for this case, the two integrals in (2) are the usual Chandrasekhar H-function [5, 15].

Discretization of the integrodifferential equation of (2) yields an algebraic matrix Riccati equation. To see this, let $\{\omega_k\}_{k=1}^n$ and $\{c_k\}_{k=1}^n$ denote the sets of the Gauss–Legendre nodes and weights, respectively, on $[0,1]$ with

$$0<\omega_n<\cdots<\omega_2<\omega_1<1 \tag{3a}$$

[*] Received by the editors March 7, 1997; accepted for publication (in revised form) by V. Mehrmann January 28, 1998; published electronically September 23, 1998. This work was supported in part by the National Science Council of the R.O.C. on Taiwan.

http://www.siam.org/journals/simax/20-1/31825.html

[†] Department of Applied Mathematics, National Chiao Tung University, Hsinchu, Taiwan 31015, Republic of China (jjuang@math.nctu.edu.tw).

[‡] Institute of Applied Mathematics, National Tsing Hwa University, Hsinchu, Taiwan 30043, Republic of China (wwlin@am.nthu.edu.tw).

and

$$\sum_{k=1}^{n} c_k = 1,\ c_k > 0,\ k = 1, 2, \ldots, n. \tag{3b}$$

Transforming the Gauss–Legendre nodes and weights on $[0, 1]$ to the intervals $[-\alpha, 1]$ and $[\alpha, 1]$, respectively, we have the following relationships:

$$\omega_k^- = \{(1+\alpha)\omega_k - \alpha\} \in [-\alpha, 1], \quad c_k^- = c_k(1+\alpha), \tag{4a}$$

$$\omega_k^+ = \{(1-\alpha)\omega_k + \alpha\} \in [\alpha, 1], \quad c_k^+ = c_k(1-\alpha) \tag{4b}$$

for $k = 1, \ldots, n$. Let $X_{ij} = X(\omega_i^-, \omega_j^+),\ i, j = 1, \ldots, n$. Replacing μ, ν with ω_i^- and ω_j^+, respectively, in (2), the integrals in (2) can be approximated by

$$\int_{-\alpha}^{1} \frac{X(\omega, \omega_j^+)}{\omega + \alpha} d\omega \sim \sum_{k=1}^{n} \frac{c_k^- X_{kj}}{\omega_k^- + \alpha}$$

and

$$\int_{\alpha}^{1} \frac{X(\omega_i^-, \omega)}{\omega - \alpha} d\omega \sim \sum_{k=1}^{n} \frac{c_k^+ X_{ik}}{\omega_k^+ - \alpha}.$$

Consequently, the descretized version of (2) becomes

$$\begin{aligned} &\frac{1}{c(\omega_i^- + \alpha)} X_{ij} + \frac{1}{c(\omega_j^+ - \alpha)} X_{ij} \\ &= 1 + \frac{1}{2}\sum_{k=1}^{n} \frac{c_k^- X_{kj}}{\omega_k^- + \alpha} + \frac{1}{2}\sum_{k=1}^{n} \frac{c_k^+ X_{ik}}{\omega_k^+ - \alpha} + \frac{1}{4}\sum_{k=1}^{n}\sum_{l=1}^{n} \frac{X_{ik} c_k^+ c_l^- X_{lj}}{(\omega_k^+ - \alpha)(\omega_l^- + \alpha)}. \end{aligned} \tag{5}$$

Substituting (4) into (5) and writing the resulting equation in matrix form, we get an $n \times n$ nonsymmetric algebraic matrix Riccati equation in X:

$$B - AX - XD + XCX = 0, \tag{6}$$

where A, B, C, and D have the following structures:

$$A = \operatorname{diag}[\delta_1, \delta_2, \ldots, \delta_n] - eq^T, \tag{7a}$$

$$B = ee^T, \tag{7b}$$

$$C = qq^T, \tag{7c}$$

and

$$D = \operatorname{diag}[d_1, d_2, \ldots, d_n] - qe^T, \tag{7d}$$

where

$$\delta_i = \frac{1}{cw_i(1+\alpha)}, \qquad d_i = \frac{1}{cw_i(1-\alpha)}, \tag{8a}$$

and

$$e = [1, 1, \ldots, 1]^T, \qquad q = [q_1, q_2, \ldots, q_n]^T \text{ with } q_i = \frac{c_i}{2w_i}. \tag{8b}$$

In studying (6), we may assume that all the data are real and that $0 < c \leq 1,\ 0 \leq \alpha < 1$, and (3) are satisfied. Consequently, we may assume that

$$0 < \delta_1 < \delta_2 < \cdots < \delta_n \tag{9a}$$

and

$$0 < d_1 < d_2 < \cdots < d_n. \tag{9b}$$

In addition, we may assume that

$$d_i = \delta_i \text{ for } \alpha = 0,\ d_i > \delta_i \text{ for } \alpha \neq 0, \quad i = 1, 2, \ldots, n. \tag{9c}$$

Recently, the existence of nonnegative solutions (in the componentwise sense) of (6) was demonstrated via the degree theory by Juang [13]. Some iterative procedures [14] have been developed for finding the nonnegative solutions of (6). However, for the case in which $c \approx 1$ and $\alpha \approx 0$, the convergence rates of these procedures are very slow, which is unsatisfactory.

Now, let H denote a 2×2 block matrix of the form

$$H := \begin{bmatrix} D & -C \\ B & -A \end{bmatrix}, \tag{10}$$

where A, B, C, and D are as defined in (7). We call this matrix H a Hamiltonian-like matrix of (6). In this paper, we develop a different approach to constructing the nonnegative solutions of (6) based on computing the invariant subspaces of H corresponding to some specified eigenvalues. The inversion formula of a Cauchy matrix is also used to explicitly construct such solutions. Our approach gives a complete representation and bifurcation diagram, with respect to parameters c and α, for nonnegative solutions of (6). Furthermore, it provides a numerical algorithm for computing the nonnegative solutions of (6) and avoids the deficiencies inherent in the iterative procedures mentioned above.

Symmetric algebraic Riccati equations arising from linear-quadratic control problems are often solved by computing the "stable" invariant subspace of the corresponding Hamiltonian matrix $\tilde{H}$ (see, e.g., [17, p. 55]). Such equations have been treated at length in the literature (see, e.g., [17] and the works cited therein). Here $\tilde{H}$ is of the form $\tilde{H} = \left[\begin{smallmatrix} \tilde{A}^T & -\tilde{C} \\ \tilde{B} & -\tilde{A} \end{smallmatrix}\right]$, where $\tilde{B}, \tilde{C}$ are symmetric and $\tilde{A}$ is arbitrary. On the other hand, nonsymmetric Riccati equations (see, e.g., [7, 18]) are less well understood than their symmetric counterparts. Note that H, given in (10), is a Hamiltonian matrix only when $c = 1$ and $\alpha = 0$, which is why we call it a Hamiltonian-like matrix. Moreover, we are seeking a nonnegative solution of (6), as opposed to the positive semidefinite solutions found in linear-quadratic control problems or the nonsingular solutions found in polynomial factorizations.

This paper is organized as follows. In section 2, we analyze the eigenvalue distribution of H and characterize the components of the associated eigenvectors. In section 3, a complete representation and bifurcation diagram of the nonnegative solutions of (6) are established. In particular, we show that (6) has a unique nonnegative solution when $c = 1$ and $\alpha = 0$ and two nonnegative solutions otherwise. An error analysis and some numerical experiments for the computation of the nonnegative solutions are given in section 4. In section 5, some comparison results are derived. Specifically, we are able to show that the minimal solution of (6) is increasing in c and decreasing in α. Our concluding section primarily contains some thoughts regarding possible future research related to the results presented here. For completeness and ease of reference, we conclude this introductory section by recording some well-known results.

In what follows, we shall give the definition of the M-matrix and its properties (see, e.g., [19, p. 54]).

DEFINITION 1.1. *A real $n \times n$ matrix A is an M-matrix if there exists a nonnegative matrix B with a maximal eigenvalue r such that $A = cI_n - B$, where $c \geq r$.*

THEOREM 1.2. *Let matrix A be an $n \times n$ nonsingular real matrix with nonpositive off-diagonal elements. Then the following are equivalent:*
(i) *A is an M-matrix.* (ii) *Every real eigenvalue in A is nonnegative.* (iii) *A^{-1} is nonnegative.*

THEOREM 1.3. *Let two $n \times n$ matrices A_i, $i = 1, 2$, be decomposed into $A_i = D_i - B_i$, respectively, where D_i, $i = 1, 2$, are diagonal parts of A_i, $i = 1, 2$. Suppose A_1 is an invertible M-matrix, $D_1 \leq D_2$, and $B_1 \geq B_2 \geq 0$. A_2 is then an M-matrix and $A_2^{-1} \leq A_1^{-1}$.*

2. Properties of the Hamiltonian-like matrix H. In this section, we analyze the eigenvalue distribution of H given in (10) and characterize the components and properties of the associated eigenvectors.

LEMMA 2.1. *The matrix H, as defined in* (10), *has only real eigenvalues $\{-\mu_n, \ldots, -\mu_1, \lambda_1, \ldots, \lambda_n\}$, which are arranged in an ascending order and satisfy the following inequalities:*

$$-\delta_n < -\mu_n < -\delta_{n-1} < \ldots < -\delta_2 < -\mu_2 < -\delta_1 < -\mu_1 \leq 0, \tag{11a}$$

$$0 \leq \lambda_1 < d_1 < \lambda_2 < d_2 < \ldots < \lambda_n < d_n. \tag{11b}$$

Moreover, the following hold: (i) *$\mu_1 = 0$ only if $c = 1$.* (ii) *$\lambda_1 = 0$ only if $c = 1$ and $\alpha = 0$.* (iii) *$\mu_i = \lambda_i, i = 1, 2, \ldots, n$, for $\alpha = 0$.*

Proof. Let

$$D_1 = \operatorname{diag}[d_1, d_2, \ldots, d_n] \quad \text{and} \quad \Delta_1 = \operatorname{diag}[\delta_1, \delta_2, \ldots, \delta_n]. \tag{12}$$

We rewrite $H - \lambda I$ as

$$H - \lambda I = \begin{bmatrix} D_1 & 0 \\ 0 & -\Delta_1 \end{bmatrix} - \lambda I - \begin{bmatrix} q \\ -e \end{bmatrix} \begin{bmatrix} e^T, & q^T \end{bmatrix}. \tag{13}$$

The secular equation (see, e.g., [3, 6, 11]) of $H - \lambda I$ is given by

$$\begin{aligned} f(\lambda) &= 1 - (e^T, q^T) \begin{bmatrix} (D_1 - \lambda I)^{-1} & 0 \\ 0 & -(\Delta_1 + \lambda I)^{-1} \end{bmatrix} \begin{bmatrix} q \\ -e \end{bmatrix} \\ &= 1 - \sum_{i=1}^{n} \frac{q_i}{d_i - \lambda} - \sum_{i=1}^{n} \frac{q_i}{\delta_i + \lambda}. \end{aligned} \tag{14}$$

Since $\lambda = d_i$ and $-\delta_i$, $i = 1, \ldots, n$, are not eigenvalues of H, finding eigenvalues of H is equivalent to locating the roots of $f(\lambda)$. Using (14), we immediately have the following asymptotic properties:

$$\lim_{\lambda \to \pm\infty} f(\lambda) = 1, \quad \lim_{\lambda \to d_i^{\pm}} f(\lambda) = \pm\infty, \quad \lim_{\lambda \to -\delta_i^{\pm}} f(\lambda) = \mp\infty, \quad i = 1, 2, \ldots, n.$$

The intermediate value theorem indicates that $f(\lambda)$ must have at least one root in each of the intervals (d_i, d_{i+1}) and $(-\delta_{i+1}, -\delta_i)$, where $i = 1, 2, \ldots, n-1$. Thus, there are at least $2n-2$ roots in those intervals. We next examine the number of possible roots of $f(\lambda)$ in the interval $(-\delta_1, d_1)$. To this end, we evaluate $f(0)$ and its rate of change, $f^{'}(0)$.

From (14), (8), and (3b), it can be determined that

$$\begin{aligned} f(0) &= 1 - \sum_{i=1}^{n} \frac{c_i}{2\omega_i} c\omega_i(1-\alpha) - \sum_{i=1}^{n} \frac{c_i}{2\omega_i} c\omega_i(1+\alpha) \\ &= 1 - c \begin{cases} > 0 & \text{for } 0 \le c < 1, \\ = 0 & \text{for } c = 1 \end{cases} \end{aligned} \tag{15a}$$

and

$$\begin{aligned} f'(0) &= -\frac{1}{2}\sum_{i=1}^{n} c_i c^2 \omega_i (1-\alpha)^2 + \frac{1}{2}\sum_{i=1}^{n} c_i c^2 \omega_i (1+\alpha)^2 \\ &= 2\alpha c^2 \sum_{i=1}^{n} c_i \omega_i \begin{cases} > 0 & \text{for } \alpha > 0, \\ = 0 & \text{for } \alpha = 0. \end{cases} \end{aligned} \tag{15b}$$

Since $\lim_{\lambda \to -\delta_1^+} f(\lambda) = \lim_{\lambda \to d_1^-} f(\lambda) = -\infty$, we conclude, via (15), that, for $0 < c < 1$, $f(\lambda)$ has two other roots. Specifically, one is in $(-\delta_1, 0)$ and the other is in $(0, d_1)$. It follows from (15) that, for $c = 1$ and $0 < \alpha < 1$, one root of $f(\lambda)$ is zero and the other root is in $(0, d_1)$, and for $c = 1$ and $\alpha = 0$, $f(\lambda)$ has a zero root of multiplicity 2. We thus complete the proof of the lemma. □

The following results can be easily obtained by studying the secular equations of A and D. The proof of the lemma is thus omitted.

Lemma 2.2. *The eigenvalues of A and D are real and positive.*

We now turn our attention to the eigenvectors corresponding to the eigenvalues λ_k and μ_k of H for $k = 1, \ldots, n$.

Lemma 2.3. *Let $[x_{1,k}, \ldots, x_{2n,k}]^T$ and $[z_{1,k}, \ldots, z_{2n,k}]^T$ be the eigenvectors of H corresponding to λ_k and $-\mu_k$, respectively, for $k = 1, \ldots, n$. Then it holds that*

$$x_{i,k} = \frac{q_i(\delta_n + \lambda_k)}{d_i - \lambda_k} \quad \textit{and} \quad x_{n+i,k} = \frac{\delta_n + \lambda_k}{\delta_i + \lambda_k}, i = 1, \ldots, n, \tag{16}$$

$$z_{i,k} = \frac{q_i(\delta_n - \mu_k)}{d_i + \mu_k} \quad \textit{and} \quad z_{n+i,k} = \frac{\delta_n - \mu_k}{\delta_i - \mu_k}, i = 1, \ldots, n, \tag{17}$$

for $k = 1, \ldots, n$.

Proof. Let $x_k = [x_{1,k}, x_{2,k}, \ldots, x_{2n,k}]^T$ be the eigenvector corresponding to λ_k; i.e., $(H - \lambda_k I)x_k = 0$. Writing $H - \lambda_k I$ in the form of (13) and using the last component -1 in $\left[\begin{smallmatrix} q \\ -e \end{smallmatrix}\right]$ as a pivotal element to eliminate the other elements of $\left[\begin{smallmatrix} q \\ -e \end{smallmatrix}\right]$, we get

(18)

$$\begin{bmatrix} \tilde{d}_{1,k} & 0 & \cdots & 0 & 0 & \cdots & \cdots & 0 & -q_1\tilde{\delta}_{n,k} \\ 0 & \ddots & \ddots & & & & & \vdots & \vdots \\ \vdots & \ddots & \ddots & 0 & & & & \vdots & \vdots \\ \vdots & & \ddots & \tilde{d}_{n,k} & 0 & & & \vdots & -q_n\tilde{\delta}_{n,k} \\ \vdots & & & 0 & -\tilde{\delta}_{1,k} & \ddots & & \vdots & \tilde{\delta}_{n,k} \\ \vdots & & & & 0 & \ddots & \ddots & \vdots & \vdots \\ \vdots & & & & & \ddots & \ddots & 0 & \vdots \\ 0 & \cdots & \cdots & 0 & 0 & \cdots & 0 & -\tilde{\delta}_{n-1,k} & \tilde{\delta}_{n,k} \\ 1 & \cdots & \cdots & 1 & q_1 & \cdots & \cdots & q_{n-1} & -\tilde{\delta}_{n,k}+q_n \end{bmatrix} \begin{bmatrix} x_{1,k} \\ \vdots \\ \vdots \\ x_{n,k} \\ x_{n+1,k} \\ \vdots \\ \vdots \\ x_{2n-1,k} \\ x_{2n,k} \end{bmatrix} = 0.$$

Here, $\tilde{d}_{i,k} = d_i - \lambda_k$ and $\tilde{\delta}_{i,k} = \delta_i + \lambda_k$. Letting $x_{2n,k} = 1$, we see, via (18), that

$$x_{i,k} = \frac{q_i\tilde{\delta}_{n,k}}{\tilde{d}_{i,k}} = \frac{q_i(\delta_n + \lambda_k)}{d_i - \lambda_k} \quad \text{and } x_{n+i,k} = \frac{\tilde{\delta}_n}{\tilde{\delta}_i} = \frac{\delta_n + \lambda_k}{\delta_i + \lambda_k} \tag{19}$$

for $i = 1, \ldots, n$. The eigenvectors corresponding to $-\mu_k$ can be obtained in a similar way. □

PROPOSITION 2.4. *For $c = 1$ and $\alpha = 0, \lambda_1 = \mu_1 = 0$ is a zero eigenvalue of H that has an algebraic multiplicity of two and a geometric multiplicity of one.*

Proof. From Lemma 2.1 we see that the eigenvalues $\lambda_1 = \mu_1 = 0$ of H has algebraic multiplicity of two. To see the geometric multiplicity of $\lambda_1 = \mu_1 = 0$, by noting the leading principal $(2n-1)\times(2n-1)$ minor of the coefficient matrix in (18) with $\tilde{d}_{i,k} = d_i$ and $\tilde{\delta}_{i,k} = \delta_i$ is nonzero, we conclude that the geometric multiplicity of $\lambda_1 = \mu_1 = 0$ is one. □

Let

$$W = \left[\frac{1}{d_i - \lambda_j}\right]_{i,j=1}^{n} := [w_{i,j}], \tag{20}$$

where $\{d_i\}_{i=1}^n$ and $\{\lambda_j\}_{j=1}^n$ are given in Lemma 2.3. Then W is a Cauchy matrix. We next state a result of [9] which is very useful in proving our main results.

LEMMA 2.5. (i) (See *Theorem* 3.1 *of* [9].) *The matrix W defined in* (20) *is nonsingular, and its inverse is given by*

$$W^{-1} = D_1 W^T D_2, \tag{21}$$

where $D_1 = \operatorname{diag}(\alpha_1, \ldots, \alpha_n)$ and $D_2 = \operatorname{diag}(\beta_1, \ldots, \beta_n)$ with

$$\alpha_i = \frac{-\prod\limits_{j=1}^{n}(\lambda_i - d_j)}{\prod\limits_{j=1, j\neq i}^{n}(\lambda_i - \lambda_j)} \quad \textit{and} \quad \beta_i = \frac{\prod\limits_{j=1}^{n}(d_i - \lambda_j)}{\prod\limits_{j=1, j\neq i}^{n}(d_i - d_j)}, \tag{22}$$

for i=1,..., *n. Moreover, α_i, $\beta_i > 0$ for $1 \le i \le n$. For $n = 1$ the denominators in* (24) *is interpreted as 1.*

(ii) *Let $\alpha = [\alpha_1, \ldots, \alpha_n]^T$, $\beta = [\beta_1, \ldots, \beta_n]^T$. Then $W\alpha = e$ and $W^T\beta = e$. Here e is given in* (8b).

Proof. We need only to prove the second part of the lemma. To this end, let $W^T\beta = f = [f_1, f_2, \ldots, f_n]^T$. By (20) and (22),

$$f_i = \sum_{k=1}^{n} \frac{\prod\limits_{j=1,j\neq i}^{n}(d_k - \lambda_j)}{\prod\limits_{j=1,j\neq k}^{n}(d_k - d_j)} =: \sum_{k=1}^{n} r_k.$$

Let $\phi_i(\lambda) = \prod\limits_{j=1,j\neq i}^{n}(\lambda - \lambda_j)$, which is an $n-1$ order monic polynomial. Set

$$\psi(\lambda) := \sum_{k=1}^{n} r_k \prod_{j=1,j\neq k}^{n}(\lambda - d_j).$$

Then $\psi(\lambda)$ is the Lagrangian interplating polynomial of $\phi_i(\lambda)$ at the points $d_1, \ldots, d_n$. That is, $\phi_i(d_j) = \psi(d_j),\ j = 1, \ldots, n$. Because the order of $\psi(\lambda)$ is also $n-1$, we have $\phi_i(\lambda) \equiv \psi(\lambda)$. By comparing the first coefficient, we get $f_i = 1$. So we have $W^T\beta = e$. Finally, $W^{-1}e = D_1W^TD_2e = D_1W^T\beta = D_1e = \alpha$. We thus complete the proof of the lemma. □

Remark 2.1. Let α_i and β_i be as given in (22) except with λ_1 on the respective products replaced by $-\mu_k$. Denote such new α_i and β_i by $\alpha_{i,k}$ and $\beta_{i,k}$, respectively. Then the assertions in the second part of Lemma 2.5 still hold for the corresponding W since the interlace property remains true.

We next show the matrices X_1 and $Z_1^{(k)}$, given in (23), are invertible. Such assertions will be used in constructing the solution of the algebraic Riccati equation (6) in section 3.

THEOREM 2.6. *Let*

$$(23) \qquad X_1 = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,n} \\ \vdots & \vdots & & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,n} \end{bmatrix}, \ Z_1^{(k)} = \begin{bmatrix} z_{1,k} & x_{1,2} & \cdots & x_{1,n} \\ z_{2,k} & x_{2,2} & \cdots & x_{2,n} \\ \vdots & \vdots & & \vdots \\ z_{n,k} & x_{n,2} & \cdots & x_{n,n} \end{bmatrix},$$

where $x_{i,k}$ and $z_{i,k}$, $1 \le i, k \le n$, are defined in (16) *and* (17)*, respectively. Then X_1 and $Z_1^{(k)}$, $k = 1, 2, \ldots, n$, are invertible.*

Proof. From (16), decompose X_1 into

$$(24) \qquad X_1 = D_qWD_\delta,$$

where

$$(25a) \qquad D_q = \mathrm{diag}[q_1, q_2, \ldots, q_n],$$

$$(25b) \qquad D_\delta = \mathrm{diag}[\delta_n + \lambda_1, \delta_n + \lambda_2, \ldots, \delta_n + \lambda_n],$$

and W is defined as in (20). Thus, the nonsingularity of W, and therefore of X_1, follows immediately from Lemma 2.5. The assertion for $Z_1^{(k)}, k = 1, \ldots, n$, can be similarly obtained. □

COROLLARY 2.7. *Let $X_1^{-1} = [\widetilde{x}_{i,j}]_{i,j=1}^n$. Then it holds that*

$$(26) \qquad \widetilde{x}_{i,j} > 0 \quad \textit{for} \quad i \le j \qquad \textit{and} \quad \widetilde{x}_{i,j} < 0 \quad \textit{for} \quad i > j.$$

Similarly, let $(Z_1^{(k)})^{-1} = [\widetilde{z}_{i,j}^{(k)}]_{i,j}^n$. The corresponding elements $\widetilde{z}_{i,j}^{(k)}$, $i, j = 1, 2, \ldots, n$, then satisfy the relationship shown in (26).

Proof. The statement (26) follows immediately from (21), (22), and (24). □

3. Existence and multiplicity of nonnegative solutions. Our object in this section is to study the existence and multiplicity of the nonnegative solutions of (6). To derive the main results, we first write (6) in the form

$$\begin{bmatrix} D & -C \\ B & -A \end{bmatrix} \begin{bmatrix} I \\ X \end{bmatrix} = \begin{bmatrix} I \\ X \end{bmatrix} (D - CX). \tag{27}$$

It is easily seen that the Span $\{\begin{bmatrix} I \\ X \end{bmatrix}\}$ forms an invariant subspace of H corresponding to the matrix $D - CX$. We first recall the following well-known theorem (see, e.g., [17]).

THEOREM 3.1. *If the Span* $\{\begin{bmatrix} X_1 \\ X_2 \end{bmatrix}\}$ *forms an invariant subspace of* H *associated with the matrix* $\Lambda \in R^{n\times n}$ *and if* X_1 *is invertible, then* $X = X_2 X_1^{-1}$ *is a solution of* (6).

PROPOSITION 3.2. *If* X *is a nonnegative solution of* (6)*, then* $\{\lambda_2, \lambda_3, \ldots, \lambda_n\}$ *must be the eigenvalues of* $D - CX$*. Consequently,* (6) *has at most* $n+1$ *nonnegative solutions and at most* n *nonnegative solutions when* $c = 1$ *and* $\alpha = 0$.

Proof. Let X be a nonnegative solution of (6). Then,

$$D - CX = D_1 - q(e^T + q^T X) := D_1 - q\tilde{q}^T$$

with $\tilde{q}_i > 0$ for all i. The secular equation of $D_1 - q\tilde{q}^T$ is

$$s(\lambda) = 1 - \sum_{i=1}^{n} \frac{q_i \tilde{q}_i}{d_i - \lambda}.$$

Since $s(-\infty) > 0, s(d_1^-) < 0$, and $s(d_i^+)s(d_{i+1}^-) < 0$ for $i = 1, 2, \ldots, n-1$, we may conclude that $D - CX$ has n distinct real eigenvalues $\tilde{\lambda}_1, \tilde{\lambda}_2, \ldots, \tilde{\lambda}_n$. Moreover, there are at least $n-1$ positive eigenvalues, say, $\tilde{\lambda}_i > 0, i = 2, \ldots, n$. Since $\sigma(D - CX)$, the spectrum of $D - CX$, is contained in $\sigma\left(\begin{bmatrix} D & -C \\ B & -A \end{bmatrix}\right)$, it then follows from Lemma 2.1 that $\lambda_i = \tilde{\lambda}_i$ for $i = 2, 3, \ldots, n$. The assertions in the proposition then follow from Theorem 3.1 and Proposition 2.4. □

Remark 3.1. From Lemma 2.1, Theorem 3.1, and Proposition 2.4, we conclude that (6) has at most $\binom{2n}{n} - \binom{2n-2}{n-1}$ solutions for $c = 1$ and $\alpha = 0$ and at most $\binom{2n}{n}$ solutions otherwise.

We next prove the following main result.

THEOREM 3.3. *Let* $\begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$ *and* $\begin{bmatrix} Z_1^{(1)} \\ Z_2^{(1)} \end{bmatrix}$ *be the eigenvector matrices of* H *corresponding to* $\Lambda = \mathrm{diag}[\lambda_1, \lambda_2, \ldots, \lambda_n]$ *and* $\Gamma_1 = \mathrm{diag}[-\mu_1, \lambda_2, \ldots, \lambda_n]$*, respectively. Then* $X = X_2 X_1^{-1}$ *and* $Z = Z_2^{(1)}(Z_1^{(1)})^{-1}$ *are positive solutions of Riccati equation* (6)*. Moreover,* $Z \geq X > 0$.

Proof. We first prove that $X_2 X_1^{-1}$ is positive. Let $W_2 = \left[\frac{1}{\delta_i + \lambda_j}\right]$, D_δ, and D_q be given in (25). Using (16), we see that $X_2 = W_2 D_\delta$ and $X_1 = D_q W D_\delta$. Hence,

$$X = W_2 W^{-1} D_q^{-1} = W_2 D_1 W^T D_2 D_q^{-1},$$

where D_1 and D_2 are given in (21). Let $X = [\chi_{i,j}]$. We see that

$$\chi_{i,j} = \left\{ \sum_{\ell=1}^{n} \left(\frac{1}{\delta_i + \lambda_\ell} \right) \left(\frac{1}{d_j - \lambda_\ell} \right) \alpha_\ell \right\} \left(\frac{\beta_j}{q_j} \right).$$

Using the identity

$$\frac{1}{(\delta_i+\lambda_\ell)(d_j-\lambda_\ell)} = \frac{1}{\delta_i+d_j}\left(\frac{1}{\delta_i+\lambda_\ell}+\frac{1}{d_j-\lambda_\ell}\right)$$

and recognizing

$$\sum_{\ell=1}^{n}\frac{\alpha_\ell}{d_j-\lambda_\ell} = (W\alpha)_j = 1, \tag{28}$$

we see that

$$\chi_{i,j} = \frac{\beta_j}{q_j(\delta_i+d_j)}\left(1+\sum_{\ell=1}^{n}\frac{\alpha_\ell}{\delta_i+\lambda_\ell}\right) > 0. \tag{29}$$

The last equality in (28) is justified by Lemma 2.5 (ii). To complete the proof of the theorem, let $Z = Z_2^{(1)}(Z_1^{(1)})^{-1} = [\zeta_{i,j}]$. Here $Z_1^{(1)}$ is given in (23) and

$$Z_2^{(1)} = \begin{bmatrix} z_{n+1,1} & x_{n+1,2} & x_{n+1,3} & \cdots & x_{n+1,n} \\ z_{n+2,1} & x_{n+2,2} & x_{n+2,3} & \cdots & x_{n+2,n} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ z_{n+n,1} & x_{n+n,2} & x_{n+n,3} & \cdots & x_{n+n,n} \end{bmatrix},$$

where $z_{n+i,1}$ and $x_{n+i,k}$ are defined in (16) and (17), respectively.

To complete the proof of the theorem, it remains to show that $Z - X \geq 0$. To this end, we see that

$$\begin{aligned} Z - X &= (Z_2^{(1)} - XZ_1^{(1)})(Z_1^{(1)})^{-1} \\ &= \{(Z_2^{(1)} - X_2) + X(X_1 - Z_1^{(1)})\}(Z_1^{(1)})^{-1} \\ &=: F(Z_1^{(1)})^{-1}. \end{aligned}$$

Using (16), (17) and doing some direct calculations, we see that F must be of the form $F = ge_1^T$, where g is a nonnegative vector and $e_1^T = (1, 0, \ldots, 0)^T$. Note, via Corollary 2.7, that $e_1^T(Z_1^{(1)})^{-1} \geq 0$. Thus $Z - X = F(Z_1^{(1)})^{-1} = ge_1^T(Z_1^{(1)})^{-1} \geq 0$. The proof of the theorem is thus complete. □

Remark 3.2. Using Remark 2.1 and a procedure similar to that above, we have that $\zeta_{i,j}$ are defined as in (29) except with λ_1 in the respective products and summation taken as $-\mu_1$; i.e.,

$$\zeta_{i,j} = \frac{\beta_{j,1}}{q_j(\delta_i+d_j)}\left(1+\frac{\alpha_{1,1}}{\delta_i-\mu_1}+\sum_{\ell=2}^{n}\frac{\alpha_{\ell,1}}{\delta_i+\lambda_\ell}\right). \tag{30}$$

THEOREM 3.4. *Equation* (6) *has a unique nonnegative solution when* $c = 1$ *and* $\alpha = 0$*; otherwise, when* $0 < c < 1$ *and* $0 < \alpha < 1$*, it has two nonnegative solutions.*

Proof. From Proposition 3.2, it suffices to show, at this stage, that for $k = 2, \ldots, n$, letting $\left[\begin{smallmatrix} Z_1^{(k)} \\ Z_2^{(k)} \end{smallmatrix}\right]$ be the eigenvector matrices of H corresponding to $\{-\mu_k, \lambda_2, \ldots, \lambda_n\}$, respectively, results in $Z^{(k)} = Z_2^{(k)}(Z_1^{(k)})^{-1}$ being other than nonnegative. However, these assertions follow directly from Corollary 2.7 and (17). □

4. Error analysis and numerical experiments. In this section, we first provide a perturbation analysis of Riccati equation (6). For the case that $0 \leq c < 1$ and $0 < \alpha < 1$, one can apply the standard theory as discussed in Byers [4] and Kenney and Laub [16]. Let $X = X_2X_1^{-1}$ be the positive solution of (6). Let $||\cdot||$ denote the Frobenius norm and $P_X(\delta)$ be the set of perturbations with respect to X:

$$P_X(\delta) = \left\{ \frac{\|\Delta X\|}{\delta\|X\|} : \frac{\|\Delta A\|}{\|A\|} \leq \delta,\ \frac{\|\Delta D\|}{\|D\|} \leq \delta,\ \frac{\|\Delta C\|}{\|C\|} \leq \delta \right\}.$$

Here $\widetilde{A} = A+\Delta A,\ \widetilde{D} = D+\Delta D$, and $\widetilde{C} = C+\Delta C$ are some perturbed matrices in a δ-neighborhood of $A,\ D$, and C, respectively, and $\widetilde{X} = X+\Delta X$ is the corresponding perturbed solution. Note that $B = ee^T$ has no perturbation error. Following Rice [20], we define the (asymptotic) condition number of (6) as follows:

$$\nu_X(A,C,D) = \lim_{\delta\to 0}\sup\{P_X(\delta)\}. \tag{31}$$

The magnitude of $\nu_X(A,C,D)$ is used to measure the sensitivity of the solution of (6) to perturbations in the data. If $\nu_X(A,C,D)$ is "large," then small changes in the data make large changes in the solution. Consequently, the Riccati equation (6) is ill conditioned. If $\nu_X(A,C,D)$ is of "modest magnitude," then the small changes in the data make small changes in the solution. Hence, the corresponding Riccati equation (6) is well conditioned. Define

$$K_X(A,C,D) = \frac{\|\Theta_X\|\max\{\|A\|,\|D\|\} + \|\Pi_X\|\|C\|}{\|X\|}, \tag{32}$$

where Θ_X and Π_X are linear operators on $\mathbf{R}^{n\times n}$, respectively, given by

$$\Theta_X(V) = \Omega_X^{-1}(V^TX + XV) \ \text{ and } \ \Pi_X(V) = \Omega_X^{-1}(XVX)$$

with

$$\Omega_X(V) = (-XC+A)V + V(D-CX).$$

By a similar argument in [4], one can also show that

$$\frac{1}{9}K_X(A,C,D) \leq \nu_X(A,C,D) \leq 4K_X(A,C,D). \tag{33}$$

From (27), we have that the eigenvalues of Ω_X are of the form $\lambda_k + \mu_\ell$, where $\{\lambda_k\}_{k=1}^n$ and $\{\mu_\ell\}_{\ell=1}^n$, defined in Lemma 2.1, are eigenvalues of $(D-CX)$ and $(A-XC)$, respectively. Expressions (32) and (33) show that Riccati equation (6) is ill conditioned when $\|\Omega_X^{-1}\|$ is large. The quantity $\|\Omega_X^{-1}\|^{-1}$ is usually measured by $sep((D-CX),-(A-XC))$ [21]. From the definition of "*sep*" it follows that

$$\frac{1}{\min\limits_{1\leq k,\ell\leq n}|\lambda_k+\mu_\ell|} \leq \|\Omega_X^{-1}\| = [sep((D-CX),-(A-XC))]^{-1}. \tag{34}$$

We next apply the above perturbation analysis to the case that $c \approx 1$ and $\alpha \approx 0$. From Lemma 2.1, we see that $\lambda_1 \to 0^+$ and $-\mu_1 \to 0^-$ as $c \to 1^-$ and $\alpha \to 0^+$. In this case, $\frac{1}{|\lambda_1+\mu_1|}$; hence, $\nu_X(A,C,D)$ become very large. Therefore, the Riccati equation (6) for $c \approx 1$ and $\alpha \approx 0$ is very ill conditioned. This shows that the convergence rates of some iterative methods of [14] for solving Riccati equation (6) are very slow and unsatisfactory.

We now turn our attention to the formulae derived in (29) and (30). The nonnegative solutions $X = X_2 X_1^{-1}$ and $Z = Z_2^{(1)} Z_1^{(1)-1}$ are as in Theorem 3.4 and thus can be computed directly by (29), (30), and (22). In the following we give an error analysis on the method of (29) and (30). For simplicity, we suppose the given data $\{d_i, \delta_i\}_{i=1}^n$ have no error propagation in computation and let ϵ_{λ_i} be the relative error of λ_i caused by computation. By a standard technique of error analysis (see, e.g., [22, Chap. 1]), one can derive the relative errors for the computation of $\{\alpha_i\}_{i=1}^n$ and $\{\beta_i\}_{i=1}^n$ of (22) as follows:

$$\text{(35a)} \qquad \operatorname{rel}(\alpha_i) \equiv \epsilon_{\alpha_i} = \sum_{j=1}^{n} \frac{\lambda_i}{\lambda_i - d_j} \epsilon_{\lambda_i} - \sum_{j=1, j \neq i}^{n} \left(\frac{\lambda_i}{\lambda_i - \lambda_j} \epsilon_{\lambda_i} - \frac{\lambda_j}{\lambda_i - \lambda_j} \epsilon_{\lambda_j} \right)$$

and

$$\text{(35b)} \qquad \operatorname{rel}(\beta_i) \equiv \epsilon_{\beta_i} = \sum_{j=1}^{n} \frac{\lambda_j}{\lambda_j - d_i} \epsilon_{\lambda_j}.$$

Here rel(x) denotes the relative error for the computation of x. Let

$$\text{(36a)} \qquad \epsilon_{\max} := \max\{|\epsilon_{\lambda_i}|,\ i = 1, \ldots, n\},$$

$$\text{(36b)} \qquad \theta_{\max} := \max\left\{ \frac{|\lambda_i|}{|\lambda_i - d_i|},\ \frac{|\lambda_i|}{|\lambda_i - d_{i-1}|},\ i = 1, \ldots, n,\ d_0 = -\infty \right\},$$

$$\text{(36c)} \qquad \tilde{\theta}_{\max} := \max\left\{ \left| \frac{\lambda_i + \lambda_{i+1}}{\lambda_{i+1} - \lambda_i} \right|,\ i = 1, \ldots, n \right\},$$

$$\text{(36d)} \qquad d_{\min} := \min\{|d_i - d_{i+1}|,\ i = 1, \ldots, n\}.$$

From (35), (36), and (11) we can estimate $|\epsilon_{\alpha_i}|$ and $|\epsilon_{\beta_i}|$ as follows:

$$\begin{aligned} |\epsilon_{\alpha_i}| &\leq \Bigg[\left(\frac{|\lambda_i|}{|\lambda_i - d_i|} + \frac{|\lambda_i|}{|\lambda_i - d_{i-1}|} + \left(\sum_{j=1}^{i-2} + \sum_{j=i+1}^{n} \right) \frac{|\lambda_i|}{|\lambda_i - d_j|} \right) \\ &\qquad + \left(\frac{|\lambda_{i+1} + \lambda_i|}{|\lambda_{i+1} - \lambda_i|} + \frac{|\lambda_i + \lambda_{i-1}|}{|\lambda_i - \lambda_{i-1}|} + \left(\sum_{j=1}^{i-2} + \sum_{j=i+1}^{n} \right) \frac{|\lambda_i + \lambda_j|}{|\lambda_i - \lambda_j|} \right) \Bigg] \epsilon_{\max} \\ &\leq \left[2\left(\theta_{\max} + \tilde{\theta}_{\max} \right) + \frac{2d_i + d_n}{d_{\min}} \left(\ell n(4(i-2)(n-i)) \right) \right] \epsilon_{\max} \\ \text{(37)} \qquad &\leq \left[2\left(\theta_{\max} + \tilde{\theta}_{\max} \right) + \frac{6d_n}{d_{\min}} \ell n(2n) \right] \epsilon_{\max} \end{aligned}$$

and

$$\begin{aligned} |\epsilon_{\beta_i}| &\leq \left[\frac{|\lambda_i|}{|\lambda_i - d_i|} + \frac{|\lambda_{i+1}|}{|\lambda_{i+1} - d_i|} + \left(\sum_{j=1}^{i-2} + \sum_{j=i+1}^{n} \right) \frac{|\lambda_j|}{|\lambda_j - d_i|} \right] \epsilon_{\max} \\ &\leq \left[2\theta_{\max} + \frac{d_n}{d_{\min}} \left(\ell n(4(i-1)(n-i+1)) \right) \right] \epsilon_{\max} \\ \text{(38)} \qquad &\leq \left[2\theta_{\max} + \frac{2d_n}{d_{\min}} \ell n(2n) \right] \epsilon_{\max}. \end{aligned}$$

Table 4.1
($c = 0.999999,\ \alpha = 10^{-8}$).

	$n = 32$	$n = 64$	$n = 128$	$n = 256$
r_X	2.1807e-13	4.3211e-12	3.1650e-11	1.0723e-10
r_Z	2.1926e-13	4.4682e-12	3.1528e-11	1.2455e-10
λ_1	1.73206684765e-3	1.73206684059e-3	1.73206683757e-3	1.73206678707e-3
$-\mu_1$	–1.73203684762e-3	–1.73203684057e-3	–1.73203683731e-3	–1.73203678614e-3

Table 4.2
($c = 1.0,\ \alpha = 10^{-14}$).

	$n = 32$	$n = 64$	$n = 128$	$n = 256$
r_X	6.9022e-13	4.3476e-12	4.8312e-11	2.0916e-10
λ_1	1.72951930554e-15	4.14078493715e-15	2.15344021678e-15	2.61125671244e-15

Here ℓn denotes the natural logarithm.

Consequently, from (37) and (38) the relative error for the computation of $x_{i,j}$ is bounded by

$$|\mathrm{rel}(x_{i,j})| \leq \left[1 + 4\theta_{\max} + 2\tilde{\theta}_{\max} + \frac{8d_n}{d_{\min}}\ell n(2n)\right]\epsilon_{\max} \tag{39}$$

for $i,\ j = 1, \ldots, n$. From (4) and (8) we see that the distances between d_i and d_{i+1} are well separated. Moreover, using the fact that $\lambda_i \in (d_i, d_{i+1})$ and the secular equation $f(\lambda)$ in (14), we see that λ_i is well separated from the end points d_i and d_{i+1}. Therefore, the quantities $\theta_{\max}$ and $\tilde{\theta}_{\max}$ and $\frac{1}{d_{\min}}$ defined in (36) cannot become too large. Thus, the relative error of $x_{i,j}$ depends on the quantity of $\epsilon_{\max}$. Indeed, a bisection method combined with Newton's acceleration scheme can be applied to $f(\lambda)$ in (14) for computing the desired eigenvalues $\{\lambda_i\}_{i=1}^{n}$ accurately. Numerical stability for the computation $\{x_{i,j}\}_{i,j=1}^{n}$ and $\{z_{i,j}^{(1)}\}_{i,j=1}^{n}$ is guaranteed, even when the problem for solving Riccati equation (6) is ill posed for $c \approx 1$ and $\alpha \approx 0$.

In the following, we give the numerical results of our test examples. We compute the nonnegative solutions X and Z by using the formulae (29) and (30) with different matrix sizes, $n = 32$, 64, 128, and 256. In Table 4.1 we compute nonnegative solutions X and Z for $c = 0.999999$ and $\alpha = 10^{-8}$. In Table 4.2 we compute the unique nonnegative solution X for $c = 1.0$ and $\alpha = 10^{-14}$. Here r_X and r_Z denote, respectively, the 2-norm residuals of Riccati equation (6) for the computed solutions X and Z.

As mentioned in section 1 for the case in which $c \approx 1$ and $\alpha \approx 0$, iterative procedures [14] can cause numerical problems during the convergence process. Our numerical result shows that the residual r_X of the computed nonnegative solution is very satisfactory, even when the condition number $\nu_X(A, C, D)$ estimated by (32), (33), and (34) is very "large" for $c = 1$ and $\alpha = 10^{-14}$.

5. Comparison theorems for nonnegative solutions. Noting that only minimal solution is physically meaningful (see, e.g., [2]), we show in this section that the minimal nonnegative solution X of (6) is increasing in c and decreasing in α. The dependency of X on the parameter c is well known. However, the effect of the parameter α on X is less understood. Our assertions here provide a better picture as to how the roles of α are played.

To begin, we consider the following iteration:

$$AX^{(p+1)} + X^{(p+1)}(D - CX^{(p)}) = B = ee^T, \tag{40}$$

with $X^{(0)} = 0$. Let $X = X_2X_1^{-1}$ be as given in Theorem 3.3, and set

$$D - CX^{(p)} := \Lambda_p. \tag{41}$$

Let $X^{(p+1)} = [x_{i,j}^{(p+1)}]$. Equation (40) can be equivalently written as a linear system of the form

$$(A \otimes I + I \otimes \Lambda_p)[x_{11}^{(p+1)}, \ldots, x_{1n}^{(p+1)}, x_{21}^{(p+1)}, \ldots, x_{nn}^{(p+1)}] = [1, 1, \ldots, 1]^T. \tag{42}$$

Here $\otimes$ denotes the Kronecker product (see, e.g., [1]). To save notation, we shall write (42) as

$$(A \otimes I + I \otimes \Lambda_p)X^{(p+1)} = B. \tag{43}$$

We then prove the following lemmas.

Lemma 5.1. (i) *$\Lambda_0 = D$ is an M-matrix.* (ii) *$\overline{\Lambda} = X_1\Lambda X_1^{-1} = D - CX$ is an M-matrix.*

Proof. To see the first assertion of the lemma, we note, via Lemma 2.2, that the eigenvalues of D are real and nonnegative. Using the fact that off-diagonal elements of D are nonpositive, we conclude that Λ_0 is an M-matrix. The second assertion also follows from the fact that off-diagonal elements of $\overline{\Lambda}$ are nonpositive and its eigenvalues are $\{\lambda_i\}$, which are nonnegative. □

Lemma 5.2. (i) *$A \otimes I + I \otimes \overline{\Lambda}$ is an M-matrix.* (ii) *Let $p \in N \cup \{0\}$. For matrix $\Lambda_p = D - CX^{(p)}$, with $0 \le X^{(p)} \le X$, $A \otimes I + I \otimes \Lambda_p$ is an M-matrix, and $(A \otimes I + I \otimes \overline{\Lambda})^{-1} \ge (A \otimes I + I \otimes \Lambda_p)^{-1}$.*

Proof. We first note, via Lemma 2.2, that the eigenvalues of A are positive. It is then clear that the off-diagonal elements of $A \otimes I + I \otimes \overline{\Lambda}$ are nonpositive and its eigenvalues are nonnegative. Hence, $A \otimes I + I \otimes \overline{\Lambda}$ is an M-matrix. The second part of the lemma follows by applying Theorem 1.3 on $A \otimes I + I \otimes \overline{\Lambda}(= A_1)$ and $A \otimes I + I \otimes \Lambda_p(= A_2)$. □

We are now ready to prove the following theorem.

Lemma 5.3. *Let $X = X_2X_1^{-1}$ be as given in Theorem* 3.4. *Then $X^{(p)}$, as defined in* (40), *converge upward to X.*

Proof. We first prove that

$$0 \le X^{(p-1)} \le X^{(p)} \le X \qquad \text{for all } p \in \mathbf{N}.$$

To see this, we note that

$$0 = X^{(0)} \le X^{(1)}.$$

Moreover, using Lemma 5.2 (ii), we get

$$X = (A \otimes I + I \otimes \overline{\Lambda})^{-1}B \ge (A \otimes I + I \otimes \Lambda_0)^{-1}B = X^{(1)},$$

and so

$$0 = X^{(0)} \le X^{(1)} \le X.$$

Suppose (42) holds for $p = k$. Then we see, as in Lemma 5.2 (ii), that

$$(A \otimes I + I \otimes \Lambda_{k-1})^{-1} \le (A \otimes I + I \otimes \Lambda_k)^{-1} \le (A \otimes I + I \otimes \overline{\Lambda})^{-1}.$$

Using (41), we obtain that

$$0 \le X^{(k)} \le X^{(k+1)} \le X.$$

Therefore, we conclude, via an induction, that (42) holds as claimed. Let the limit of the sequence $\{X^{(p)}\}$ be denoted by $X^{(\infty)}$. Since $X^{(\infty)}$ is a nonnegative solution of (6) and $X^{(\infty)} \leq X$, it must be that $X^{(\infty)} = X$. $\square$

To emphasize the dependence of X on the parameters c and α, we write X as $X(c, \alpha)$. Likewise, all quantities are similarly written if necessary. We are now ready to state our comparison result.

THEOREM 5.4. *The solution* $X = X_2 X_1^{-1}$ *of* (6) *is increasing in* c *and decreasing in* α. *In particular,* $X(1, 0) \geq X(c, \alpha)$ *for all* c, α.

Proof. For fixed α and $c_1 \leq c_2$, suppose $X^{(p)}(c_1, \alpha) \leq X^{(p)}(c_2, \alpha)$, where $X^{(p)}(c_i, \alpha), i = 1, 2$, are as defined in (40). Then by applying Theorem 1.3 on

$$A_1 = A(c_2, \alpha) \otimes I + I \otimes \Lambda_p(c_2, \alpha) \qquad \text{and} \qquad A_2 = A(c_1, \alpha) \otimes I + I \otimes \Lambda_p(c_1, \alpha),$$

we get

$$(A(c_1, \alpha) \otimes I + I \otimes \Lambda_p(c_1, \alpha))^{-1} \leq (A(c_2, \alpha) \otimes I + I \otimes \Lambda_p(c_2, \alpha))^{-1}.$$

Here $\Lambda_p(c_i, \alpha) := D(c_i, \alpha) - CX^{(p)}(c_i, \alpha)$, $i = 1, 2$. It then follows from (41) that

$$X^{(p+1)}(c_1, \alpha) \leq X^{(p+1)}(c_2, \alpha).$$

Clearly,

$$0 = X^{(0)}(c_1, \alpha) \leq X^{(0)}(c_2, \alpha) = 0.$$

Hence, an induction yields that $X^{(p)}(c_1, \alpha) \leq X^{(p)}(c_2, \alpha)$ for all $p \in \mathbf{N}$. We conclude, via Lemma 5.3, that $X(c_1, \alpha) \leq X(c_2, \alpha)$.

To see $X(c, \alpha)$ is decreasing in α, we first note that

$$\delta_i + d_i = \frac{1}{cw_i(1+\alpha)} + \frac{1}{cw_i(1-\alpha)} = \frac{2}{cw_i(1-\alpha^2)}$$

are increasing in α. Therefore, for fixed c and $\alpha_1 \leq \alpha_2$, suppose $X^{(p)}(c, \alpha_1) \geq X^{(p)}(c, \alpha_2)$. Then by applying Theorem 1.3 on $A_1 = A(c, \alpha_1) \otimes I + I \otimes \Lambda_p(c, \alpha_1)$ and $A_2 = A(c, \alpha_2) \otimes I + I \otimes \Lambda_p(c, \alpha_2)$, we get

$$[A(c, \alpha_1) \otimes I + I \otimes \Lambda_p(c, \alpha_1)]^{-1} \geq [A(c, \alpha_2) \otimes I + I \otimes \Lambda_p(c, \alpha_2)]^{-1}.$$

Noting that

$$0 = X^{(0)}(c, \alpha_1) \geq X^{(0)}(c, \alpha_2) = 0,$$

we conclude, via an induction and Lemma 5.3, that

$$X(c, \alpha_1) \geq X(c, \alpha_2).$$

The proof of the theorem is thus complete. $\square$

6. Concluding remarks.

We conclude with a few suggestions for further related work.

First, the method of invariant embedding has been applied to transport problems (see, e.g., [10]) involving neutrons and gamma rays with realistic energy and angle-dependent cross-sections. It is therefore of interest to study a more general form of algebraic matrix Riccati equations encompassing those cases.

Next, we note that the simple transport model [8, 10] in an isotropically scattering plane-parallel layer of finite thickness would induce a differential Riccati equation of the form

$$X' = B - AX - XD + XCX, \tag{44a}$$

$$X(0) = 0. \tag{44b}$$

Here A, B, C, and D are as defined in (7). It would be worthwhile to pursue the asymptotic characteristics and stability of the nonnegative solutions of (6) with respect to this differential Riccati equation (44).

Finally, it would be desirable to generalize our techniques for solving the corresponding algebraic Riccati equation to infinitely dimensional cases [15].

Acknowledgments. We thank Professor Volker Mehrmann and referees for suggesting numerous improvements to the original draft. In particular, the inversion formula of a Cauchy matrix was brought to our attention, which leads to a much shorter proof of Theorem 3.4 and Lemma 2.5 (ii).

REFERENCES

[1] R. Bellman, *Introduction to Matrix Analysis*, 2nd ed., McGraw-Hill, New York, 1970.

[2] R. Bellman and G. M. Wing, *An Introduction to Invariant Embedding*, John Wiley, New York, 1975.

[3] J. R. Bunch, C. R. Nielsen, and D. C. Sorensen, *Rank-one modification of the symmetric eigenproblem*, Numer. Math., 31 (1978), pp. 31–48.

[4] R. Byers, *Numerical stability and instability in matrix sign function based in algorithms*, in Computational and Combined Methods in Systems Theory, C. I. Byrnes and A. Lindquist, eds., North-Holland, New York, 1986, pp. 185–200.

[5] S. Chandrasekhar, *Radiative Transfer*, Dover, New York, 1960.

[6] J. J. M. Cuppen, *A divide and conquer method for the symmetric tridiagonal eigenproblem*, Numer. Math., 36 (1981), pp. 177–195.

[7] D. J. Clements and B. D. O. Anderson, *Polynomial factorization via the Riccati equation*, SIAM J. Appl. Math., 31 (1976), pp. 179–205.

[8] F. Coron, *Computation of the asymptotic states for linear half space kinetic problem*, Transport Theory Statist. Phys., 19 (1990), pp. 89–114.

[9] T. Finck, G. Heinig, and K. Rost, *An inversion formula and fast algorithms for Cauchy-Vandermonde matrices*, Linear Algebra Appl., 183 (1993), pp. 179–197.

[10] B. D. Ganapol, *An investigation of a simple transport model*, Transport Theory Statist. Phys., 21 (1992), pp. 1–37.

[11] G. H. Golub, *Some modified matrix eigenvalue problems*, SIAM Rev., 15 (1973), pp. 318–334.

[12] G. H. Golub and J. H. Wilkinson, *Ill-conditioned eigensystems and the computations of the Jordan canonical form*, SIAM Rev., 18 (1976), pp. 578–619.

[13] J. Juang, *Existence of algebraic matrix Riccati equations arising in transport theory*, Linear Algebra Appl., 230 (1995), pp. 89–100.

[14] J. Juang and I-Der Chen, *Iterative solution for a certain class of algebraic matrix Riccati equations arising in transport theory*, Transport Theory Statist. Phys., 21 (1993), pp. 65–80.

[15] J. Juang and P. Nelson, *Global existence, asymptotic and uniqueness for the reflection kernel of the angularly shifted transport equation*, Math. Models Methods Appl. Sci., 5 (1995), pp. 239–251 .

[16] C. Kenney and A. J. Laub, *Condition estimation for matrix functions*, SIAM J. Matrix Anal. Appl., 10 (1989), pp. 191–209.

[17] V. L. Mehrmann, *The Autonomous Linear Quadratic Control Problem*, Springer-Verlag, Berlin, 1991.

[18] H. Meyer, *The matrix equation $AZ + B - ZCZ - ZD = 0$*, SIAM J. Appl. Math., 30 (1976), pp. 136-142.

[19] J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, 1970.

[20] J. R. Rice, *A theory of condition*, SIAM J. Numer. Anal., 3 (1966), pp. 287–310.

[21] G. W. Stewart, *Error and perturbation bounds for subspaces associated with certain eigenvalue problems*, SIAM Rev., 15 (1973), pp. 727–764.

[22] J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*, Springer-Verlag, New York, 1980.

SIAM J. MATRIX ANAL. APPL.
Vol. 20, No. 1, pp. 244–269

NEW FAST ALGORITHMS FOR STRUCTURED LINEAR LEAST SQUARES PROBLEMS[*]

MING GU[†]

Abstract. We present new fast algorithms for solving the Toeplitz and the Toeplitz-plus-Hankel least squares problems. These algorithms are based on a new fast algorithm for solving the Cauchy-like least squares problem. We perform an error analysis and provide conditions under which these algorithms are numerically stable. We also develop implementation techniques that significantly reduce the execution time. While no previous fast algorithm is known to be numerically stable for very ill conditioned problems, our numerical results indicate that these new algorithms are efficient and numerically stable for problems ranging from well conditioned to very ill conditioned to numerically singular.

Key words. displacement equation, error analysis, fast algorithm, iterative refinement, Toeplitz matrix

AMS subject classifications. 15A06, 65F05, 65G05

PII. S089547989529646X

1. Introduction.

1.1. Displacement equations and structured matrices.

The *Sylvester-type displacement equation* for a matrix $M \in \mathbf{C}^{m\times n}$ is

$$\Omega\, M - M\, \Lambda = \Delta, \tag{1.1}$$

where $\Omega \in \mathbf{C}^{m\times m}$ and $\Lambda \in \mathbf{C}^{n\times n}$, $\Delta \in \mathbf{C}^{m\times n}$ is called the *generator* of M with respect to Ω and Λ, and $r = \operatorname{rank}(\Delta) \le \min(m, n)$ is called the *displacement rank* of M with respect to Ω and Λ. M possesses a *displacement structure* with respect to Ω and Λ if $r \ll \min(m, n)$.

In general, there is no simple relationship between r and $\operatorname{rank}(M)$. For $r \ll \min(m, n)$, we usually factorize Δ as $\Delta = A\, B$ for matrices $A \in \mathbf{C}^{m\times r}$ and $B \in \mathbf{C}^{r\times n}$. This decomposition is not unique. For numerical stability reasons, we often choose A to be well conditioned.

The displacement equation (1.1) does not in general reflect the potential symmetry structure in M. The *symmetric Stein-type displacement equation* for a Hermitian matrix M is

$$M - \Omega^*\, M\, \Omega = \Theta\ , \tag{1.2}$$

where $\Omega \in \mathbf{C}^{m\times m}$; $\Theta \in \mathbf{C}^{m\times m}$ is Hermitian and is called the *generator* of M with respect to Ω.

For $r = \operatorname{rank}(\Theta) \ll m$, we usually factorize Θ as $\Theta = A\, J\, A^*$ for matrices $A \in \mathbf{C}^{m\times r}$ and $J \in \mathbf{C}^{r\times r}$ with J being Hermitian and both A and J being well conditioned.

[*] Received by the editors December 27, 1995; accepted for publication (in revised form) by L. Eldén April 23, 1997; published electronically September 29, 1998.

http://www.siam.org/journals/simax/20-1/29646.html

[†] Department of Mathematics, University of California, Los Angeles, CA 90095-1555 (mgu@math.ucla.edu). This work was supported in part by the Applied Mathematical Sciences Subprogram of the Office of Energy Research, U.S. Department of Energy, under contract DE-AC03-76SF00098.

The concept of displacement structure was introduced in Kailath, Kung, and Morf [34], the symmetric variant of which, the displacement equation of the form (1.2), appeared in Chun, Kailath, and Lev-Ari [14]. Displacement equations of the form (1.1) appeared in Heinig and Rost [31]. The most general form of displacement structure, which includes equations (1.1) and (1.2) as special cases, was introduced in Kailath and Sayed [36]. For a comprehensive discussion on the displacement structure theory and applications, see Kailath and Sayed [37].

A special case in (1.1) is when both Ω and Λ are diagonal. Let $C \in \mathbf{C}^{m\times n}$ satisfy

$$\Omega\, C - C\, \Lambda = A\, B \tag{1.3}$$

with $\Omega = \mathrm{diag}(\omega_1, \ldots, \omega_m)$, $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$, and

$$A = \begin{pmatrix} a_1^* \\ \vdots \\ a_m^* \end{pmatrix} \quad \text{and} \quad B = (b_1, \ldots, b_n)\,,$$

where ω_k, λ_j are scalars and a_k and b_j are r-dimensional vectors. C is called a *Cauchy-like* matrix. Usually we assume that $\omega_k \neq \lambda_j$ for all $1 \leq k \leq m$ and $1 \leq j \leq n$. In this case the (k, j) entry of C is $a_k^*\, b_j / \omega_k - \lambda_j$. In particular, C is a Cauchy matrix if $m = n$, $r = 1$, and $a_k = b_j = 1$ for all k and j. In the case where $\omega_k = \lambda_j$ for some pairs of (k, j), (1.3) requires that $a_k^*\, b_j = 0$ and allows the corresponding entries in C to be arbitrary. A Cauchy-like matrix C has the interesting property that any submatrix of C is again a Cauchy-like matrix. In addition, if C is a nonsingular square Cauchy-like matrix, then C^{-1} is a Cauchy-like matrix as well. Some symmetric/Hermitian Cauchy-like matrices satisfy[1] a displacement equation similar to (1.2):

$$M - \Omega^*\, M\, \Omega = A\, J\, A^*\ , \tag{1.4}$$

where

$$\Omega = \mathrm{diag}(\omega_1, \ldots, \omega_m), \quad A = \begin{pmatrix} a_1^* \\ \vdots \\ a_m^* \end{pmatrix},$$

and $J \in \mathbf{C}^{r\times r}$ is Hermitian. For $\omega_k^*\, \omega_j \neq 1$, the (k, j) entry of M is $a_k^*\, J\, a_j / 1 - \omega_k^*\, \omega_j$. For $\omega_k^*\, \omega_j = 1$, (1.4) requires that $a_k^*\, J\, a_j = 0$ and allows the corresponding entries in M to be arbitrary.

Other classes of structured matrices include the Toeplitz matrices and the Hankel matrices. A Toeplitz matrix T is a matrix whose entries are constant along every diagonal ($T = (t_{k-j})_{1\leq k\leq m, 1\leq j\leq n}$), and a Hankel matrix H is a matrix whose entries are constant along every antidiagonal ($H = (h_{k+j-2})_{1\leq k\leq m, 1\leq j\leq n}$). Toeplitz and Hankel matrices are included in the larger class of Toeplitz-plus-Hankel matrices, which are sums of Toeplitz and Hankel matrices. These matrices often arise from signal processing and control theory applications (see, for example, Bunch [9] and Nagy [41]). We will discuss the displacement equations that the Toeplitz matrix and the Toeplitz-plus-Hankel matrix satisfy in section 3.

[1] Assume that $\omega_k \neq 0$ for all k; then M is Cauchy-like since (1.4) can be rewritten in the form of (1.3) as $\Omega^{-*}\, M - M\, \Omega = (\Omega^{-*}\, A)\, (J\, A^*)$.

1.2. Fast algorithms for structured matrices. Our main goal is to develop new fast algorithms for solving the linear least squares problem

$$\min_x \|M\,x - h\|_2 \ , \tag{1.5}$$

where $M \in \mathbf{R}^{m\times n}$ is the Toeplitz or the Toeplitz-plus-Hankel matrix, and $h \in \mathbf{R}^m$ is a vector. We will also consider the case where M is a real or complex Cauchy-like matrix. Throughout this paper, we assume that $\operatorname{rank}(M) = n$ and that $m \geq n + r$, where r is the displacement rank of M. The problem (1.5) has a unique solution

$$x_M = (M^*\,M)^{-1}\ M^*\,h\ . \tag{1.6}$$

Fast algorithms for solving the least squares problem (1.5) when M is a Toeplitz matrix have been developed by Bojanczyk, Brent, and de Hoog [5], Chun, Kailath, and Lev-Ari [14], Cybenko [16, 17], Nagy [40], Park and Eldén [44], Qiao [45], and Sweet [46] that require $O(mn)$ floating point operations, as opposed to $O(mn^2)$ floating point operations normally required for solving general dense linear least squares problems. Fast parallel algorithms have also been developed by Bojanczyk and Brent [4] and Brent [7]. However, some of these algorithms have unknown stability properties (see Brent [7]) and others are known to be unstable (see Luk and Qiao [39]); most of these methods suffer from loss of accuracy for very ill conditioned problems.

A special case of the Toeplitz least squares problem is the Toeplitz linear system of equations. For discussions on some of the earlier fast and superfast methods (performing $O(n^2)$ and $O(n\log_2^2 n)$ floating point operations, respectively) for solving such equations, see Bojanczyk et al. [6], Bunch [9], Cybenko [15], Sweet [47], and the references therein.

Recently, Gohberg, Kailath, and Olshevsky [21] developed an algorithm, Algorithm GKO, for solving the Toeplitz system by transforming the Toeplitz matrix into a Cauchy-like matrix via fast Fourier or trigonometric transforms, and then solving the Cauchy-like linear system of equations via a fast variation of the straightforward Gaussian elimination with partial pivoting (GEPP) procedure; they have also demonstrated numerically that Algorithm GKO is stable. Their work is based on some earlier work of Heinig [30]. Sweet and Brent [48] have shown that the generator of the Cauchy-like matrix in Algorithm GKO could suffer large internal element growth, and Gu [28] has presented a modified procedure that avoids such internal element growth and that can perform a fast variation of Gaussian elimination with complete pivoting (GECP). Gohberg and Olshevsky [23, 24] and Kailath and Olshevsky [35] have developed fast variations of the Bunch–Kaufman pivoting procedure [8] for factorizing symmetric/Hermitian Cauchy-like matrices.

1.3. Main results. We present a new fast algorithm for solving the least squares problem (1.5) when M is a Cauchy-like matrix. This algorithm reduces the least squares problem to two Cauchy-like systems of linear equations and solves them by generalizing techniques of Gohberg, Kailath, and Olshevsky [21], Kailath and Olshevsky [35], Gu [28], and Heinig [30]. An error analysis performed by Gu [27] shows that this algorithm is backward stable if the L matrix in the LU factorization of M with fast partial/complete pivoting is well conditioned.

We also present a new fast algorithm for solving the least squares problem (1.5) when M is a Toeplitz or Toeplitz-plus-Hankel matrix. It transforms M into a Cauchy-like matrix via the fast Fourier or trigonometric transforms and solves the resulting Cauchy-like least squares problem using the fast algorithm above. Since the choices of transformations are not unique, we compare different choices in terms of efficiency and numerical accuracy in solving the Toeplitz and the Toeplitz-plus-Hankel least squares problems. An error analysis performed by Gu [27] shows that this algorithm is as backward stable as the algorithm for solving the resulting Cauchy-like least squares problem. Since the Hankel and Toeplitz matrices are simply related, this new fast algorithm is also a new fast algorithm for solving the Hankel least squares problem.

We develop implementation techniques that significantly reduce the execution time. We perform a large number of numerical experiments on the new fast Toeplitz and Toeplitz-plus-Hankel least squares problem solver and compare it with the straightforward QR-type least squares problem solver that ignores the Toeplitz and Toeplitz-plus-Hankel structures. Our numerical results indicate that this fast algorithm is indeed much faster than the straightforward solver and yet is essentially as accurate on problems ranging from well conditioned to very ill conditioned to numerically singular. To the best of our knowledge, no other fast Toeplitz or Toeplitz-plus-Hankel least squares problem solver is numerically stable for very ill conditioned least squares problems.

In section 2 we present the new fast algorithm for solving the Cauchy-like least squares problem. In section 3 we show how to transform Toeplitz and Toeplitz-plus-Hankel matrices into Cauchy-like matrices and compare different choices of transformations in terms of efficiency and numerical accuracy. In section 4 we develop implementation techniques that reduce the execution time and present numerical results. In section 5 we draw conclusions and discuss extensions.

1.4. Notation and conventions.

i is the unit imaginary number ($i^2 = -1$). For a matrix M, M^* denotes its complex conjugate; in case M is real, both M^* and M^T denote its transpose. $|M|$ is the matrix of moduli. We use the *max norm*, the ∞-*norm*, and the 2-*norm*:

$$\|M\|_{\max} = \max_{k,j} |M_{k,j}|, \ \|M\|_\infty = \max_k \sum_j |M_{k,j}|, \text{ and } \|M\|_2 = \max_{\|u\|_2=1} \|M\,u\|_2.$$

I_k is the $k \times k$ identity matrix. We use MATLAB-like notation to denote submatrices. $M_{p:q,s:k}$ is a submatrix of M that selects rows p to q of columns s to k; $M_{:,s:k}$ and $M_{s:k,:}$ select sth through kth rows and columns, respectively; and when $s = k$, we replace $s : k$ by s. When matrices Ω and Λ are diagonal, $\Omega_{s:k}$ and $\Lambda_{s:k}$ select both rows and columns s to k of Ω and Λ, respectively.

A *flop* is a real floating point operation $\alpha \circ \beta$, where α and β are real floating point numbers and $\circ$ is one of $+$, $-$, $\times$, and $\div$. Taking the absolute value or comparing two floating point numbers is also counted as a flop. We count a complex addition or subtraction as 2 flops, a complex multiplication as 6 flops, and a complex division as 11 flops.

2. The Cauchy-like least squares problem.

2.1. Reducing one least squares problem to two linear systems.

Let $C \in \mathbf{C}^{m \times n}$ be a Cauchy-like matrix satisfying (1.3). We assume that $\operatorname{rank}(C) = n$ and that the diagonal entries of Ω are distinct. So C is the unique solution to (1.3). In section 1.2 we consider the least squares problem (1.5) for $M = C$.

The expression in the solution (1.6) is not suitable for direct numerical computation for ill conditioned C. QR factorize C to get $C = Q\,R$, where $Q \in \mathbf{C}^{m \times n}$ is column unitary and $R \in \mathbf{C}^{n \times n}$ is upper triangular. The least squares solution is

$$x_C = (C^*\,C)^{-1}\,(C^*\,h) = (R^*\,Q^*\,Q\,R)^{-1}\,(R^*\,Q^*\,h) = R^{-1}\,(Q^*\,h)\,. \tag{2.1}$$

Although this scheme is backward stable, it is slow for large m and n. The QR factorization requires $O(mn^2)$ flops to compute in general, and no known algorithm can stably compute a QR factorization of a Cauchy-like matrix in $O(mn)$ flops.

Fortunately, this is not the only way to compute x_C. Partition

$$C = \begin{pmatrix} C_1 \\ C_2 \end{pmatrix}, \qquad h = \begin{pmatrix} h_{1:n} \\ h_{n+1:m} \end{pmatrix}, \tag{2.2}$$

where $C_1 = C_{1:n,1:n}$ and $C_2 = C_{n+1:m,1:n}$. We assume that C_1 is nonsingular and define

$$Z = C_2\, C_1^{-1} \in \mathbf{C}^{(m-n)\times n} \quad \text{and} \quad K = I_n + Z^*\, Z \in \mathbf{C}^{n\times n}\ . \tag{2.3}$$

The least squares solution x_C can now be rewritten as

$$\begin{aligned} x_C &= \left(C_1^*\ (I_n + Z^*\, Z)\ C_1\right)^{-1}\ C_1^* \left(\begin{pmatrix} I_n & Z^* \end{pmatrix} \begin{pmatrix} h_{1:n} \\ h_{n+1:m} \end{pmatrix} \right) \\ &= C_1^{-1}\, K^{-1}\ \left(h_{1:n} + Z^*\, h_{n+1:m}\right)\ . \end{aligned} \tag{2.4}$$

Equations like (2.4) were discussed by Noble in the context of generalized inverse computations [42]. In (2.4), C_1 is a square Cauchy-like matrix (see section 1.2). In Theorems 2.1 through 2.3 that follow, we show that for the types of Cauchy-like matrices C that interest us, both Z and K are Cauchy-like as well, with K being symmetric/Hermitian positive definite. Hence, x_C can be computed by solving the following two Cauchy-like linear systems of equations[2]

$$K\, g = h_{1:n} + Z^*\, h_{n+1:m} \quad \text{and} \quad C_1\, x_C = g. \tag{2.5}$$

THEOREM 2.1. *Let C be a Cauchy-like matrix satisfying* (1.3). *Then Z in* (2.3) *is a Cauchy-like matrix satisfying the displacement equation*

$$\Omega_{n+1:m}\, Z - Z\, \Omega_{1:n} = \left(A_{n+1:m,:} - C_2\, C_1^{-1}\, A_{1:n,:}\right)\, \left(B\, C_1^{-1}\right)\ . \tag{2.6}$$

Proof. Equation (1.3) can be rewritten as

$$\Omega_{1:n}\, C_1 - C_1\, \Lambda = A_{1:n,:}\, B \quad \text{and} \quad \Omega_{n+1:m}\, C_2 - C_2\, \Lambda = A_{n+1:m,:}\, B,$$

which can be further rewritten as

$$C_1^{-1}\, \Omega_{1:n} = \Lambda\, C_1^{-1} + (C_1^{-1}\, A_{1:n,:})\, (B\, C_1^{-1}) \quad \text{and} \quad \Omega_{n+1:m}\, C_2 = C_2\, \Lambda + A_{n+1:m,:}\, B.$$

On the other hand,

$$\Omega_{n+1:m}\, Z - Z\, \Omega_{1:n} = (\Omega_{n+1:m}\, C_2)\, C_1^{-1} - C_2\, (C_1^{-1}\, \Omega_{1:n}).$$

Plugging in the above two relations and simplifying, we obtain (2.6). □

The generator for Z in (2.6) can be rewritten as

$$Q\, W = \left(A_{n+1:m,:} - C_2\, C_1^{-1}\, A_{1:n,:}\right)\, \left(B\, C_1^{-1}\right)\ . \tag{2.7}$$

[2] Björck [3] observes that for $m - n \ll n$, the first equation in (2.5) can be solved in $O(n(m-n))$ flops. The fast algorithms in section 2.4 for factorizing K require $O(nm)$ flops and are designed to work best if $m \gg n$.

In section 2.2 we will discuss how to choose a well conditioned Q without explicitly computing either $A_{n+1:m,:} - C_2\, C_1^{-1}\, A_{1:n,:}$ or $B\, C_1^{-1}$.

Theorem 2.2. *Let C be a Cauchy-like matrix satisfying* (1.3). *Assume that all the matrices Ω, Λ, A, and B are real. Then K in* (2.3) *is a symmetric Cauchy-like matrix satisfying the displacement equation*

$$\Omega_{1:n}\, K - K\, \Omega_{1:n} = \mathcal{A}\, \mathcal{J}\, \mathcal{A}^T, \tag{2.8}$$

where $\mathcal{A} = \begin{pmatrix} Z^T Q & W^T \end{pmatrix}$ *and* $\mathcal{J} = \begin{pmatrix} 0 & -I_r \\ I_r & 0 \end{pmatrix}$ *for Q and W in* (2.7).

Proof.

$$\begin{aligned} \Omega_{1:n}\, K - K\, \Omega_{1:n} &= \Omega_{1:n}\, (I_n + Z^T Z) - (I_n + Z^T Z)\, \Omega_{1:n} \\ &= (Z\, \Omega_{1:n})^T\, Z - Z^T\, (Z\, \Omega_{1:n})\ . \end{aligned}$$

On the other hand, combining (2.6) and (2.7) gives

$$Z\, \Omega_{1:n} = \Omega_{n+1:m}\, Z - Q\, W.$$

Theorem 2.2 follows by plugging this relation into the above and simplifying. □

Remark 2.1. To compute $\mathcal{A}$ from a generator $Q\, W$ of Z, we need to form Z and perform the matrix–matrix product $Z^T\, Q$. The total cost is about $(4r+1)(m-n)n$ flops.

Theorem 2.3. *Let C be a Cauchy-like matrix satisfying* (1.3). *Assume that $|\omega_j| = 1$ for $1 \le j \le m$. Then K in* (2.3) *is a Cauchy-like matrix satisfying the displacement equation*

$$K - \Omega_{1:n}^*\, K\, \Omega_{1:n} = \mathcal{A}\, \mathcal{J}\, \mathcal{A}^*, \tag{2.9}$$

where $\mathcal{A} = \begin{pmatrix} W^* & Z^*\, \Omega_{n+1:m}^*\, Q \end{pmatrix}$ *and* $\mathcal{J} = \begin{pmatrix} -Q^*\, Q & I_r \\ I_r & 0 \end{pmatrix}$ *for Q and W in* (2.7).

Proof. By assumption, we have

$$\Omega_{1:n}^*\, \Omega_{1:n} = I_n \quad \text{and} \quad \Omega_{n+1:m}^*\, \Omega_{n+1:m} = I_{m-n}.$$

Hence,

$$\begin{aligned} K - \Omega_{1:n}^*\, K\, \Omega_{1:n} &= (I_n + Z^*\, Z) - \Omega_{1:n}^*\, (I_n + Z^*\, Z)\, \Omega_{1:n} \\ &= Z^*\, Z - (Z\, \Omega_{1:n})^*\, (Z\, \Omega_{1:n}). \end{aligned}$$

On the other hand, combining (2.6) and (2.7) gives

$$Z\, \Omega_{1:n} = \Omega_{n+1:m}\, Z - Q\, W.$$

Theorem 2.3 follows by plugging this relation into the above and simplifying. □

Remark 2.2. To compute $\mathcal{A}$ for a given generator $Q\, W$ of Z, we need to form the matrix Z and then perform the matrix–matrix product $Z^*\, \left(\Omega_{n+1:m}^*\, Q\right)$. The total cost is about $(16r+11)(m-n)n$ flops.

In sections 2.2 and 2.3, we will discuss fast algorithms for factorizing C_1 and computing the generator of Z, and in section 2.4 we will discuss fast algorithms for factorizing K.

2.2. Gaussian elimination for Cauchy-like matrices. Let C be a Cauchy-like matrix satisfying (1.3) and let the LU factorization of C be

$$C = L\,U, \tag{2.10}$$

where $L \in \mathbf{C}^{m\times n}$ is unit lower triangular and $U \in \mathbf{C}^{n\times n}$ is upper triangular. Given (2.10), the factorization for C_1 in (2.2) is simply $L_1\,U$ for $L_1 = L_{1:n,1:n}$.

While the generator of Z can be computed directly from formula (2.6) using factorization (2.10), numerical accuracy could be compromised for ill conditioned C_1. Theorems 2.4 and 2.5 below establish a procedure that computes the generator for Z during the course of Gaussian elimination on C. Define $C^{(1)} = C$ and set

$$\begin{aligned} C^{(k)} &= C_{k:m,k:n} - C_{k:m,1:k-1}\,C^{-1}_{1:k-1,1:k-1}\,C_{1:k-1,k:n} \\ &= \begin{pmatrix} \gamma^{(k)} & \left(u^{(k)}\right)^{*} \\ v^{(k)} & C^{(k)}_{2:m-k+1,2:n-k+1} \end{pmatrix}. \end{aligned} \tag{2.11}$$

$C^{(k)}$ is the Schur complement of $C_{1:k-1,1:k-1}$ for $k = 2,\dots,n$; and $\begin{pmatrix} \gamma^{(k)} \\ v^{(k)} \end{pmatrix}$ and $(\gamma^{(k)}\ \ (u^{(k)})^{*})$ are the first column and row of $C^{(k)}$, respectively. Theorem 2.4 below summarizes a few results of Gohberg, Kailath, and Olshevsky [21] and Gohberg and Olshevsky [23, 24]. Parts of it can also be found in [30, 36].

THEOREM 2.4. *Let the Cauchy-like matrix C in* (1.3) *have LU factorization* (2.10). *Then the kth row of U and kth column of L in* (2.10) *are determined by*

$$U_{k,k} = \gamma^{(k)}, \qquad U_{k,k+1:n} = \left(u^{(k)}\right)^{*}, \quad \textit{and} \quad L_{k+1:m,k} = \frac{v^{(k)}}{\gamma^{(k)}}.$$

For $1 \le k \le n-1$, the Schur complements $\{C^{(k+1)}\}_{k=1}^{n-1}$ satisfy the displacement equation

$$\Omega_{k+1:m}\,C^{(k+1)} - C^{(k+1)}\,\Lambda_{k+1:n} = A^{(k+1)}\,B^{(k+1)}, \tag{2.12}$$

where $\{A^{(k+1)}\}$ and $\{B^{(k+1)}\}$ satisfy the recursion: $A^{(1)} = A$, $B^{(1)} = B$, and

$$A^{(k+1)} = A^{(k)}_{2:m-k+1,:} - L_{k+1:m,k}\,A^{(k)}_{1,:} = A_{k+1:m,:} - C_{k+1:m,1:k}\,C^{-1}_{1:k,1:k}\,A_{1:k,:}, \tag{2.13}$$

$$B^{(k+1)} = B^{(k)}_{:,2:n-k+1} - B^{(k)}_{:,1}\,\frac{U_{k,k+1:n}}{U_{k,k}} = B_{:,k+1:n} - B_{:,1:k}\,C^{-1}_{1:k,1:k}\,C_{1:k,k+1:n}. \tag{2.14}$$

Hence, the kth column of L and kth row of U can be determined from the first column and row of $C^{(k)}$, which is recursively defined through (2.12), (2.13), and (2.14).

To compute the generator of Z, we define

$$Z^{(k)} = C_{k+1:m,1:k}\,C^{-1}_{1:k,1:k} \quad \text{for} \quad k = 1,2,\dots,n \tag{2.15}$$

so that $Z^{(n)} = Z$ (see (2.3)). Theorem 2.1 implies that every $Z^{(k)}$ is a Cauchy-like matrix of displacement rank at most r. In the following we introduce a simple recursion for computing $\{Z^{(k)}\}$ via $\{A^{(k+1)}\}$ and $\{B^{(k+1)}\}$.

First, by applying Theorem 2.1 to $Z^{(1)}$ we have

$$\Omega_{2:m,2:m}\,Z^{(1)} - Z^{(1)}\,\omega_1 = A^{(2)}\,Y^{(1)}, \quad \text{where } Y^{(1)} = B_{:,1}/\gamma.$$

To compute the generator for $Z^{(k)}$, we set

$$y^{(k)} = B^{(k)}_{:,1}/U_{k,k}, \qquad \left(z^{(k)}\right)^* = \left(A^{(k)}_{1,:}\,Y^{(k-1)}\right)\,(\omega_k I_{k-1} - \Omega_{1:k-1,1:k-1})^{-1}$$

and

$$Y^{(k)} = \left(Y^{(k-1)} - y^{(k)}\,\left(z^{(k)}\right)^* \quad y^{(k)}\right) \in \mathbf{C}^{r\times k}\,. \tag{2.16}$$

Theorem 2.5. *For $k = 2, \ldots, n$,*

$$\Omega_{k+1:m}\,Z^{(k)} - Z^{(k)}\,\Omega_{1:k} = A^{(k+1)}\,Y^{(k)}.$$

Proof. Combining Theorem 2.1 and (2.13), we see that the generator of $Z^{(k)}$ is simply $A^{(k+1)}\,B_{:,1:k}\,C^{-1}_{1:k,1:k}$ for all k. Hence, in the following we will show inductively that

$$Y^{(k-1)} = B_{:,1:k-1}\,C^{-1}_{1:k-1,1:k-1}. \tag{2.17}$$

Equation (2.17) is obviously true for $k = 2$ by construction. Assuming that it holds for some $k \geq 2$, we want to show it to hold for $k+1$. To this end, partition

$$C_{1:k,1:k} = \begin{pmatrix} C_{1:k-1,1:k-1} & C_{1:k-1,k} \\ C_{k,1:k-1} & C_{k,k} \end{pmatrix}.$$

It is well known (see, for example, Wilkinson [52, pp. 202–204]) that $U_{k,k}$ is the Schur complement of $C_{1:k-1,1:k-1}$,

$$U_{k,k} = C_{k,k} - C_{k,1:k-1}\,C^{-1}_{1:k-1,1:k-1}\,C_{1:k-1,k}.$$

Hence,

$$C^{-1}_{1:k,1:k} = \begin{pmatrix} C^{-1}_{1:k-1,1:k-1} + f\,e^*/U_{k,k} & -f/U_{k,k} \\ -e^*/U_{k,k} & 1/U_{k,k} \end{pmatrix},$$

where $f = C^{-1}_{1:k-1,1:k-1}\,C_{1:k-1,1:k}$ and $e^* = C_{k,1:k-1}\,C^{-1}_{1:k-1,1:k-1}$. Consequently

$$\begin{aligned} B_{:,1:k}\,C^{-1}_{1:k,1:k} &= (B_{:,1:k-1} \quad B_{:,k})\begin{pmatrix} C^{-1}_{1:k-1,1:k-1} + f\,e^*/U_{k,k} & -f/U_{k,k} \\ -e^*/U_{k,k} & 1/U_{k,k} \end{pmatrix} \\ &= \left(B_{:,1:k-1}\,C^{-1}_{1:k-1,1:k-1} - \widehat{f}\,e^*/U_{k,k} \quad \widehat{f}/U_{k,k}\right), \end{aligned} \tag{2.18}$$

where

$$\widehat{f} = B_{:,k} - B_{:,1:k-1}\,f = B_{:,k} - B_{:,1:k-1}\,C^{-1}_{1:k-1,1:k-1}\,C_{1:k-1,1:k},$$

which, according to (2.14), is simply $B^{(k)}_{:,1}$.

On the other hand, we observe from (2.15) that e^* is the first row of $Z^{(k-1)}$, which, by induction hypothesis (2.17) and Theorem 2.1, satisfies the displacement equation

$$\Omega_{k:m}\,Z^{(k-1)} - Z^{(k-1)}\,\Omega_{1:k-1} = A^{(k)}\,Y^{(k-1)}.$$

Hence,

$$e^* = \left(A^{(k)}_{1,:}\,Y^{(k)}\right)\,(\omega_k I_{k-1} - \Omega_{1:k-1})^{-1}\,.$$

It follows from (2.17) that $\widehat{f}/U_{k,k} = y^{(k)}$ and $e^* = \left(z^{(k)}\right)^*$. Now comparing these relations with (2.16), (2.17), and (2.18), we have $Y^{(k)} = B_{:,1:k}\,C^{-1}_{1:k,1:k}$. □

2.3. Pivoting and generator redecomposition. Factorization (2.10) does not always exists. One celebrated property of the Cauchy-like matrix C is that performing partial and complete pivoting on C does not change the Cauchy-like structure (see, for example, Gohberg and Olshevsky [22] and Heinig [30]). Let $P_m(j,k)$ and $Q_n(j,k)$ denote the permutations that interchange the jth and kth rows and columns of an $m \times n$ matrix, respectively. To perform pivoting on C, one finds a large magnitude entry $(k_{\max}, j_{\max})$ in C, permutes it to the $(1,1)$ entry to get $\widehat{C} \equiv P_m(1,k_{\max})\, C\, Q_n(1,j_{\max})$, and then applies the elimination step to $\widehat{C}$. Let C be a Cauchy-like matrix satisfying (1.3). Then for every $1 \le k \le m$ and $1 \le j \le n$,

$$(P_m(1,k)\, \Omega\, P_m(1,k))\ \widehat{C} - \widehat{C}\ (Q_n(1,j)\, \Lambda\, Q_n(1,j)) = (P_m(1,k)\, A)\ (B\, Q_n(1,j))\,.$$

It follows that $\widehat{C}$ is still a Cauchy-like matrix.

To perform partial pivoting, one chooses $j_{\max} = 1$, finds the largest magnitude entry $(k_{\max}, 1)$ in the first column of C, and permutes the $k\text{th}_{\max}$ and the first rows of C. Performing partial pivoting ensures that an LU factorization can always be computed. There is, however, a potential problem of *element growth.* Let

$$g_{PP} \equiv \max_{1 \le k \le n} \|C^{(k)}\|_{\max} / \|C\|_{\max}$$

be the *element growth factor.* It is well known that $g_{PP} \le 2^{n-1}$ for GEPP, and although very rare, this bound is attainable for certain dense matrices (see Golub and van Loan [25, pp. 115–116]). It is not clear whether this bound is attainable for Cauchy-like matrices with low displacement rank. When large element growth does occur, the computed LU factorizations can have a large backward error.

One way to reduce this element growth is to perform complete pivoting, whereby one chooses the largest magnitude entry in the entire matrix C and permutes it to the $(1,1)$ entry. This is an overall $O(mn^2)$ procedure. To reduce the cost, we adopt the fast variation of complete pivoting proposed in Gu [28] to find an entry that is sufficiently large in magnitude. Since this method involves redecomposing the generators of both $C^{(k)}$ and $Z^{(k-1)}$, in the following we show how it works on $C^{(k)}$ instead of on C.

In (2.12), we QR factorize $A^{(k)}$ to get $A^{(k)} = \mathcal{A}\, R$, where $\mathcal{A}$ is column unitary, and R is upper triangular. We then compute $\mathcal{B} = R\, B^{(k)}$ and $\mathcal{Y} = R\, Y^{(k-1)}$. It follows that

$$A^{(k)}\, B^{(k)} = \mathcal{A}\, \mathcal{B} \quad \text{and} \quad A^{(k)}\, Y^{(k-1)} = \mathcal{A}\, \mathcal{Y}.$$

In other words, we redecompose the generators of $C^{(k)}$ and $Z^{(k)}$ as $\mathcal{A}\, \mathcal{B}$ and $\mathcal{A}\, \mathcal{Y}$, respectively.

Since $\mathcal{A}$ is column unitary, the jth columns of $A^{(k)}\, B^{(k)}$ and $\mathcal{B}$ have the same 2-norm. Hence, we choose $j_{\max}$ by looking for the largest 2-norm column of $\mathcal{B}$. We then choose the $(k_{\max}, j_{\max})$ entry to be the largest magnitude entry in the $j\text{th}_{\max}$ column of $C^{(k)}$. The following lemma shows that $|C^{(k)}_{k_{\max}, j_{\max}}|$ is sufficiently large; it is a straightforward generalization of Lemma 2.1 in [28].

Lemma 2.1. *Let C be a Cauchy-like matrix satisfying equation* (1.3). *Then*

$$\|C^{(k)}\|_{\max} \le \sqrt{m}\, \psi\, |C^{(k)}_{k_{\max}, j_{\max}}|\,, \quad \textit{where} \quad \psi = \frac{\max_{k,j} |\omega_k| + |\lambda_j|}{\min_{k,j} |\omega_k - \lambda_j|}.$$

In addition to potential element growth in the LU factorization, Sweet and Brent show that the matrices $A^{(k)}$ and $B^{(k)}$, if updated as in (2.13) and (2.14), could also grow so that (see [48])

$$\left\| |A^{(k)}| \, |B^{(k)}| \right\|_2 \gg \left\| A^{(k)} \, B^{(k)} \right\|_2$$

for some k. And if this happens, the backward error in the LU factorization could be large. The same arguments show that if

$$\left\| |A^{(k)}| \, |Y^{(k-1)}| \right\|_2 \gg \left\| A^{(k)} \, Y^{(k-1)} \right\|_2 ,$$

then the backward error in the computed Z matrix could be large. However, such element growth goes away when redecomposition is performed; in fact it will not occur as long as $A^{(k)}$ is well conditioned (see Gu [27, 28]).

Algorithm 2.1 below computes an LU factorization of the form

$$C = P_m \, L \, U \, (Q_n)^T, \tag{2.19}$$

where $L \in \mathbf{C}^{m \times n}$ is unit lower triangular and $U \in \mathbf{C}^{n \times n}$ is upper triangular, and $P_m \in \mathbf{R}^{m \times m}$ and $Q_n \in \mathbf{R}^{n \times n}$ are permutation matrices. It also computes a generator of the matrix $Z = \widetilde{C}_2 \, \widetilde{C}_1^{-1}$, where

$$\widetilde{C} = (P_m)^T \, C \, Q_n = \begin{pmatrix} \widetilde{C}_1 \\ \widetilde{C}_2 \end{pmatrix} \quad \text{with} \quad \widetilde{C}_1 = \widetilde{C}_{1:n,1:n}, \ \widetilde{C}_2 = \widetilde{C}_{n+1:m,1:n}.$$

Rows of U and columns of L are computed according to Theorem 2.4, and the generator of Z is computed according to Theorem 2.5. Algorithm 2.1 also performs the variation of complete pivoting proposed in section 2.3 at every ζ step and partial pivoting at every step. Algorithm 2.1 assumes that the matrix A is initially column unitary. In practice we set $\zeta \gg r$.

Algorithm 2.1. LU factorization of Cauchy-like matrix C in (1.3) with pivoting.

```
$L := 0 \in \mathbf{C}^{m \times n}$; $U := 0 \in \mathbf{C}^{n \times n}$; $P_m := I_m$; $Q_n := I_n$;
for $k := 1$ to $n$ do
    if ($\mathbf{mod}(k, \zeta) = 0$) then
        $A_{k:m,:} := A_{k:m,:} \, R$ (QR factorizes $A_{k:m,:}$);
        $B_{:,k:n} := R \, B_{:,k:n}$; $Y_{:,1:k-1} := R \, Y_{:,1:k-1}$;
        $j_{\max} := \text{argmax}_{k \le j \le n} \|B_{:,j}\|_2$;
        if $j_{\max} > k$ then
            $Q_n := Q_n(k, j_{\max}) \, Q_n$; $\Lambda := Q_n(k, j_{\max}) \, \Lambda \, Q_n(k, j_{\max})$;
            $B := B \, Q_n(k, j_{\max})$; $U := U \, Q_n(k, j_{\max})$;
        endif
    endif
    $L_{k:m,k} := \left(\Omega_{k:m} - \lambda_k I_{m-k+1}\right)^{-1} A_{k:m,:} \, B_{:,k}$;
    $k_{\max} := \text{argmax}_{k \le j \le m} |L_{j,k}|$;
    if $k_{\max} > k$ then
        $P_m := P_m \, P_m(k, k_{\max})$; $\Omega := P_m(k, k_{\max}) \, \Omega \, P_m(k, k_{\max})$;
```

```
            $A := P_m(k, k_{\max})\, A$; $L := P_m(k, k_{\max})\, L$;
        **endif**
        $U_{k,k} := L_{k,k}$; $U_{k,k+1:n} := A_{k,:}\, B_{k+1:n}\, \left(\omega_k I_{n-k} - \Lambda_{k+1:n}\right)^{-1}$;
        $L_{k,k} := 1$; $L_{k+1:m,k} := L_{k+1:m,k}/U_{k,k}$;
        $A_{k+1:m,:} := A_{k+1:m,:} - L_{k+1:m,k}\, A_{k,:}$; $B_{:,k+1:n} := B_{:,k+1:n} - B_{:,k}\, U_{k,k+1:n}/U_{k,k}$;
        $z^* := A_{k,:}\, Y_{:,1:k-1}\, \left(\omega_k I_{k-1} - \Omega_{1:k-1}\right)^{-1}$; $y := B_{:,k}/U_{k,k}$;
        $Y_{:,1:k} := (Y_{:,1:k-1} - y\, z^* \quad y)$;
    **endfor**
```

Remark 2.3. If the input data A, B, Ω, and Λ are real, Algorithm 2.1 costs about $(4r+4)mn + (2r-1)n^2$ flops, plus an extra cost of about $(4m-n)nr^2/\zeta$ flops for redecomposing the generators; there is also potentially about $n^2(\zeta+1)/(2\zeta)$ swaps of memory locations. For a matrix transformed into a Cauchy-like matrix from a Toeplitz-plus-Hankel matrix (see sections 1 and 3), the displacement rank r is at most 4. If we choose $\zeta \gg 4$, then the cost of Algorithm 2.1 is about $(20m+7n)n$ flops.

Remark 2.4. If the input data are complex, Algorithm 2.1 costs about $(16r + 22)mn + 8rn^2$ flops, plus the cost for comparisons and the cost of about $4(4m-n)nr^2/\zeta$ flops for redecomposing the generators; there is also potentially about $n^2(\zeta+1)/\zeta$ swaps of memory locations. For a matrix transformed into a Cauchy-like matrix from a Toeplitz matrix (see sections 1 and 3), the displacement rank r is at most 2. If we choose $\zeta \gg 2$, and choose pivots that have the largest sum of real and imaginary parts in absolute value, the cost for comparisons is about $4mn - 2n^2$ flops, and the cost of Algorithm 2.1 is about $2(29m+7n)n$ flops.

Remark 2.5. If $r \gg \max(\zeta, 1)$, then the cost of redecomposing the generators dominates the computation of Algorithm 2.1. In this case we use QR updating techniques similar to those of Daniel et al. [18] (see also Golub and van Loan [25, section 12]) to perform the QR factorization at every step, bringing the total redecomposition cost down to $O(mnr)$.

To get an upper bound on the element growth factor for Algorithm 2.1, let

$$\mathcal{W}(k) = \left(k \prod_{s=2}^{k} s^{1/(s-1)}\right)^{1/2} = O\left(k^{\frac{1}{2}+\frac{1}{4}\ln k}\right).$$

This is Wilkinson's upper bound on the growth factor for GECP on a $k \times k$ general dense matrix. Although $\mathcal{W}(k)$ is not a polynomial in k, it does not grow very fast either [51]. The following theorem trivially generalizes a similar result in [28].

THEOREM 2.6. *Let C be a Cauchy-like matrix satisfying* (1.3), *and let* (2.19) *be the LU factorization generated by Algorithm* 2.1 *in exact arithmetic for $\zeta = 1$. Then the element growth factor $g_{CP} \equiv \max_{1\le k\le n} \|C^{(k)}\|_{\max}/\|C\|_{\max}$ satisfies*

$$g_{CP} \le \sqrt{n}\, \psi^{\,2+\sum_{k=1}^{n-1} 1/k}\, \mathcal{W}(n),$$

where ψ is defined in Lemma 2.1.

2.4. Factorizing a positive definite Cauchy-like matrix. In this section, we compute the Cholesky factorization of a symmetric/Hermitian positive definite Cauchy-like matrix K in (2.3):

$$K = \mathcal{L}\,\mathcal{D}\,\mathcal{L}^*, \tag{2.20}$$

where $\mathcal{L} \in \mathbf{C}^{n\times n}$ is unit lower triangular and $\mathcal{D} \in \mathbf{R}^{n\times n}$ is positive diagonal. We will assume that K satisfies either equation (2.8) or (2.9). In both cases, the displacement equation specifies all the off-diagonal entries of K; the diagonal entries of K have to be provided separately.

The first step of Cholesky factorization is to zero-out the first column and row of K below and above the diagonal entry:

$$K = \begin{pmatrix} \gamma & v^* \\ v & K_{2:n,2:n} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ l & I_{n-1} \end{pmatrix} \begin{pmatrix} \gamma & 0 \\ 0 & K^{(2)} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ l & I_{n-1} \end{pmatrix}^* . \tag{2.21}$$

Hence, $\mathcal{D}_{1,1} = \gamma$; $\mathcal{L}_{2:m,1} = l = v/\gamma$. $K^{(2)} = K_{2:m,2:n} - v\, v^*/\gamma$ is the Schur complement of γ.

2.4.1. Real Cholesky factorization. First assume that K satisfies (2.8). Then K is a real symmetric positive definite matrix. Similar to (2.12), $K^{(2)}$ is a Cauchy-like matrix. Theorem 2.7 below is a direct generalization of Theorem 2.4. More discussions on factorizing symmetric Cauchy-like matrices can be found in Gohberg and Olshevsky [23, 24] and Kailath and Olshevsky [35].

THEOREM 2.7. *Let K satisfy* (2.8). *Then $K^{(2)}$ in* (2.21) *satisfies the displacement equation*

$$\Omega_{2:n}\, K^{(2)} - K^{(2)}\, \Omega_{2:n} = \mathcal{A}^{(2)}\, \mathcal{J} \left(\mathcal{A}^{(2)}\right)^T ,$$

where $\mathcal{A}^{(2)} = \mathcal{A}_{2:n,:} - l\, \mathcal{A}_{1,:}$. The diagonal entries of $K^{(2)}$ satisfy

$$K^{(2)}_{k-1,k-1} = K_{k,k} - \frac{v_k^2}{\gamma} , \quad \textit{for} \quad k = 2, \ldots, n.$$

Hence, the first step of Cholesky factorization on K involves computing the first column of K from (2.8), computing l and $\mathcal{A}^{(2)}$, and computing diagonal entries of $K^{(2)}$. Cholesky factorization then proceeds by recursively applying this step to $K^{(2)}$, with the displacement equation in Theorem 2.7 replacing (2.8). Since K is symmetric positive definite, all the diagonal entries are positive and hence this procedure never breaks. At the end of this procedure, K is factored into (2.20) with $\mathcal{L} \in \mathbf{R}^{n\times n}$.

While Cholesky factorization on a general dense symmetric positive definite matrix is always backward stable, the above procedure may not be due to potential element growth in the generators similar to that in Algorithm 2.1 without generator redecomposition. To avoid it, we perform diagonal pivoting on K and redecompose the generators as well. Since K is symmetric positive definite, γ becomes the largest magnitude entry in K after diagonal pivoting. Based on Theorem 2.7, Algorithm 2.2 below computes a Cholesky factorization with diagonal pivoting:

$$K = Q_n\, \mathcal{L}\, \mathcal{D}\, \mathcal{L}^T \, (Q_n)^T ,$$

where $\mathcal{L} \in \mathbf{R}^{n\times n}$ is unit lower triangular and $\mathcal{D} \in \mathbf{R}^{n\times n}$ is positive diagonal. Algorithm 2.2 performs generator redecomposition every ζ step, and it assumes that on input, the diagonal entries of $\mathcal{D}$ are those of K and that $\mathcal{A}$ is well conditioned.

ALGORITHM 2.2. Real Cholesky factorization on a Cauchy-like matrix.

```
$\mathcal{L} := 0 \in \mathbf{R}^{n\times n}$; $Q_n := I_n$;
for $k := 1$ to $n$ do
```

```
        if ($\mathbf{mod}(k,\zeta)=0$) then
            $\mathcal{A}_{k:n,:} := \mathcal{A}_{k:n,:}\, R$ (QR factorizes $\mathcal{A}_{k:n}$); $\mathcal{J} := R\,\mathcal{J}\,R^T$;
        endif
        $k_{\max} := \mathrm{argmax}_{k\le j\le n}\,\mathcal{D}_{j,j}$;
        if $k_{\max} > k$ then
            $Q_n := Q_n\,Q_n(k,k_{\max})$; $\Omega_{1:n} := Q_n(k,k_{\max})\,\Omega_{1:n}\,Q_n(k,k_{\max})$;
            $\mathcal{D} := Q_n(k,k_{\max})\,\mathcal{D}\,Q_n(k,k_{\max})$;
            $\mathcal{A} := Q_n(k,k_{\max})\,\mathcal{A}$; $\mathcal{L}_{:,1:k} := Q_n(k,k_{\max})\,\mathcal{L}_{:,1:k}$;
        endif
        $\mathcal{L}_{k+1:n,k} := \left(\Omega_{k+1:n} - \omega_k I_{n-k}\right)^{-1}\,\mathcal{A}_{k+1:n,:}\,\mathcal{J}\,\mathcal{A}_{k,:}^T/\mathcal{D}_{k,k}$;
        $\mathcal{A}_{k+1:n,:} := \mathcal{A}_{k+1:n,:} - \mathcal{L}_{k+1:n,k}\,\mathcal{A}_{k,:}$;
        for $j := k+1$ to $n$ do
            $\mathcal{D}_{j,j} := \mathcal{D}_{j,j} - \mathcal{L}_{j,k}^2\,\mathcal{D}_{k,k}$.
        endfor
    endfor
```

Remark 2.6. Algorithm 2.2 costs about $(4r+2.5)n^2$ flops, where $2r$ is the displacement rank of K (see (2.8)), plus an extra cost of about $2n^2r^2/\zeta$ flops for redecomposing the generators; there is also potentially about $n^2/2$ swaps of memory locations. For a matrix transformed into a Cauchy-like matrix from a Toeplitz-plus-Hankel matrix (see sections 1 and 3), the displacement rank of K is $2r \le 8$. In this case, Algorithm 2.2 costs about $18.5n^2$ flops for $\zeta \gg 4$.

2.4.2. Complex Cholesky factorization. Now assume that K satisfies (2.9). Then K is a complex Hermitian positive definite matrix. Again $K^{(2)}$ in (2.21) is a Cauchy-like matrix. Theorem 2.8 below is a slight modification of Lemma 3.2 of Kailath and Olshevsky [35].

THEOREM 2.8. *Let K satisfy* (2.9). *Then $K^{(2)}$ in* (2.21) *satisfies the displacement equation*

$$K^{(2)} - \Omega_{2:n}^*\,K^{(2)}\,\Omega_{2:n} = \mathcal{A}^{(2)}\,\mathcal{J}\left(\mathcal{A}^{(2)}\right)^*$$

$$\textit{with}\quad \mathcal{A}^{(2)} = \mathcal{A}_{2:n,:} - (l - t/2)\,\mathcal{A}_{1,:}\,,\quad \textit{where}\quad t = \mathcal{A}_{2:n,:}\,\mathcal{J}\,\mathcal{A}_{1,:}^*/\gamma.$$

The diagonal entries of $K^{(2)}$ satisfy

$$K_{k-1,k-1}^{(2)} = K_{k,k} - \frac{|v_k|^2}{\gamma}\quad \textit{for}\quad k = 2,\ldots,n.$$

Similar to Algorithm 2.2, Algorithm 2.3 below is based on Theorem 2.8 and computes a Cholesky factorization with diagonal pivoting:

$$H = Q_n\,\mathcal{L}\,\mathcal{D}\,\mathcal{L}^*\,\left(Q_n\right)^T, \tag{2.22}$$

where $\mathcal{L} \in \mathbf{C}^{n\times n}$ is unit lower triangular and $\mathcal{D} \in \mathbf{R}^{n\times n}$ is positive diagonal. Algorithm 2.3 performs generator redecomposition every ζ step, and it assumes that on input, the diagonal entries of $\mathcal{D}$ are those of K and that $\mathcal{A}$ is well conditioned.

Algorithm 2.3. Complex Cholesky factorization.

```
$\mathcal{L} := 0 \in \mathbf{R}^{n\times n}$; $Q_n := I_n$;
for $k := 1$ to $n$ do
    if ($\mathbf{mod}(k, \zeta) = 0$) then
        $\mathcal{A}_{k:n,:} := \mathcal{A}_{k:n,:}\, R$ (QR factorizes $\mathcal{A}_{k:n,:}$); $\mathcal{J} := R\,\mathcal{J}\,R^*$;
    endif
    $k_{\max} := \mathrm{argmax}_{k\le j\le n}\, \mathcal{D}_{j,j}$;
    if $k_{\max} > k$ then
        $Q_n := Q_n\, Q_n(k, k_{\max})$; $\Omega_{1:n} := Q_n(k, k_{\max})\,\Omega_{1:n}\, Q_n(k, k_{\max})$;
        $\mathcal{D} := Q_n(k, k_{\max})\,\mathcal{D}\, Q_n(k, k_{\max})$;
        $\mathcal{A} := Q_n(k, k_{\max})\,\mathcal{A}$; $\mathcal{L}_{:,1:k} := Q_n(k, k_{\max})\,\mathcal{L}_{:,1:k}$;
    endif
    $t := \mathcal{A}_{k+1:n,:}\,\mathcal{J}\,\mathcal{A}^*_{k,:}/\mathcal{D}_{k,k}$; $\mathcal{L}_{k+1:n,k} := \left(I_{n-k} - \omega_k\,\bar{\Omega}_{k+1:n}\right)^{-1} t$;
    $\mathcal{A}_{k+1:n,:} := \mathcal{A}_{k+1:n,:} - (\mathcal{L}_{k+1:n,k} - t/2)\ \mathcal{A}_{k,:}$;
    for $j := k+1$ to $n$ do
        $\mathcal{D}_{j,j} := \mathcal{D}_{j,j} - |\mathcal{L}_{j,k}|^2\,\mathcal{D}_{k,k}$.
    endfor
endfor
```

Remark 2.7. Algorithm 2.3 costs about $(16r+14)n^2$ flops, where $2r$ is the displacement rank of K (see (2.9)), plus an extra cost of about $8n^2r^2/\zeta$ flops for redecomposing the generators; there is also potentially about n^2 swaps of memory locations. For a matrix transformed into a Cauchy-like matrix from a Toeplitz matrix (see sections 1 and 3), the displacement rank of K is $2r \le 4$. In this case, Algorithm 2.3 costs about $46n^2$ flops for $\zeta \gg 2$.

3. Toeplitz and Toeplitz-plus-Hankel least squares problems.

3.1. Toeplitz least squares problems. For any integer $k > 0$, define

$$T_k^{(\delta)} = \begin{pmatrix} 0 & 0 & \cdots & 0 & \delta \\ 1 & 0 & \cdots & \cdots & 0 \\ 0 & 1 & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & 0 \end{pmatrix} \in \mathbf{R}^{k\times k}.$$

Let $\Omega = T_m^{(1)}$ and $\Lambda = T_n^{(\delta)}$. As is noted in Gohberg, Kailath, and Olshevsky [21] and Heinig [30], it is easy to verify that every $m \times n$ Toeplitz matrix satisfies the displacement equation (1.1) with Δ having nonzero entries only in its first row and last column, and hence the displacement rank is $\mathrm{rank}(\Delta) \le 2$. To ensure that (1.1) always has a unique solution, we choose $\delta > 1$. Lemma 3.1 below is a generalization of Heinig [30].

Lemma 3.1. *Let $M \in \mathbf{C}^{m\times n}$ be a matrix satisfying the displacement equation*

$$T_m^{(1)}\, M - M\, T_n^{(\delta)} = A\, B, \tag{3.1}$$

where $A \in \mathbf{C}^{m\times r}$ and $B \in \mathbf{C}^{r\times n}$. Then $C \equiv \mathcal{F}_m\, M\, \mathcal{G}^{-1}\, \mathcal{F}_n^*$ is a Cauchy-like matrix:

$$\Omega\, C - C\,\Lambda = (\mathcal{F}_m\, A)\, \left(B\,\mathcal{G}^{-1}\,(\mathcal{F}_n)^*\right), \tag{3.2}$$

where $\mathcal{F}_m = \sqrt{\frac{1}{m}} \left(e^{\frac{2\pi i}{m}(k-1)(j-1)} \right)_{1 \le k,j \le m}$ *and* $\mathcal{F}_n = \sqrt{\frac{1}{n}} \left(e^{\frac{2\pi i}{n}(k-1)(j-1)} \right)_{1 \le k,j \le n}$ *are the normalized inverse discrete Fourier transform matrices,*

$$\mathcal{G} = \text{diag}(1, \widetilde{\delta}, \ldots, \widetilde{\delta}^{n-1}), \quad \textit{where} \quad \widetilde{\delta} = \delta^{\frac{1}{n}},$$

and

$$\Omega = \text{diag}\left(1, e^{\frac{2\pi i}{m}}, \ldots, e^{\frac{2\pi i}{m}(m-1)}\right), \quad \Lambda = \widetilde{\delta}\ \text{diag}\left(1, e^{\frac{2\pi i}{n}}, \ldots, e^{\frac{2\pi i}{n}(n-1)}\right).$$

Proof. It is well known that

$$T_m^{(1)} = (\mathcal{F}_m)^* \, \Omega \, \mathcal{F}_m.$$

On the other hand, $T_n^{(\delta)} = \widetilde{\delta}\, \mathcal{G}^{-1}\, T_n^{(1)}\, \mathcal{G}$ and hence

$$T_n^{(\delta)} = \mathcal{G}^{-1}\ (\mathcal{F}_n)^*\ \Lambda\, \mathcal{F}_n\, \mathcal{G}.$$

Equation (3.2) follows by plugging these relations into (3.1) and simplifying. ∎

A matrix M is called *Toeplitz-like* if it satisfies the displacement equation (3.1) with $r \ll n$ (cf. [21]). To solve the least squares problem (1.5) when M is a Toeplitz-like matrix, we transform M into a Cauchy-like matrix using Lemma 3.1. Setting $M = (\mathcal{F}_m)^*\ C\, \mathcal{F}^{(n)}\, \mathcal{G}$ in (1.6) and simplifying,

$$x_M = \mathcal{G}^{-1}\ (\mathcal{F}_n)^*\ (C^*\, C)^{-1}\ C^*\ (\mathcal{F}_m\, h) = \mathcal{G}^{-1}\ (\mathcal{F}_n)^*\ x_C,$$

where $x_C = (C^*\, C)^{-1}\ C^*\ (\mathcal{F}_m\, h)$ is the solution to the Cauchy-like least squares problem

$$\min_x \|C\, x - (\mathcal{F}_m\, h)\,\|_2.$$

The idea of transforming a Toeplitz linear system of equations into a Cauchy-like system of linear equations via the FFT was proposed by Heinig [30]. Related transformations were proposed by Fiedler [20], Gohberg and Olshevsky [22], and Pan [43].

We summarize the above into Algorithm 3.1, assuming that M satisfies (3.1) with A column orthogonal.

ALGORITHM 3.1. Solving the Toeplitz-like least squares problem.

1. Choose δ, compute Ω and Λ in Lemma 3.1;
2. compute $h := \mathcal{F}_m\, h$, $A := \mathcal{F}_m\, A$, and $B := B\, \mathcal{G}^{-1}\, (\mathcal{F}_n)^*$;
3. choose ζ_1, LU factorize C_1, and compute the generator for Z for C_1 and Z in (2.5) via Algorithm 2.1;
4. choose ζ_3, compute the initial generator for K in (2.5) via Theorem 2.3, and compute the Cholesky factorization of K via Algorithm 2.3;
5. compute x_C by solving the linear systems in (2.5) via forward and backward substitution;
6. compute $x_M := \mathcal{G}^{-1}\ (\mathcal{F}_n)^*\ x_C$.

Remark 3.1. The total cost of steps 1, 2, and 6 is $O(m \log_2 m)$ flops via the forward and backward FFTs; the cost of step 3 is about $(16r + 22)mn + 8rn^2$ flops, plus the cost for comparisons and the cost of about $4(4m - n)nr^2/\zeta_1$ flops for redecomposing the generators (see Remark 2.4); the cost of step 4 is about $(16r + 11)mn + 3n^2$ flops, plus an extra cost of about $8n^2r^2/\zeta_3$ flops for redecomposing the generators (see Remarks 2.2 and 2.7). Since the matrix Z is computed in step 4 (see Remark 2.2), the cost for step 5 is about $8(m+n)n$ flops. Hence, the total cost of Algorithm 3.1 is about $(32r+41)mn+(8r+11)n^2$ flops, plus the cost for comparisons and generator redecompositions. In particular, let M be a Toeplitz matrix; then $r \leq 2$. We modify Algorithm 2.1 to choose pivots that have the largest sum of real and imaginary parts in absolute value. Hence, the cost for comparisons in Algorithm 2.1 is about $4mn-2n^2$ flops (see Remark 2.4). We choose $\zeta_1 \gg 2$ and $\zeta_3 \gg 2$. Thus, the total cost of Algorithm 3.1 is about $(109m+25n)n$ flops.

3.2. Toeplitz-plus-Hankel least squares problems. For any integer $k > 0$, define

$$\mathcal{T}_k^{(\delta)} = \begin{pmatrix} 1 & 1 & 0 & \cdots & 0 \\ 1 & 0 & 1 & \cdots & \vdots \\ 0 & 1 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & 0 & 1 \\ 0 & \cdots & 0 & 1 & \delta \end{pmatrix} \in \mathbf{R}^{k\times k}.$$

Since $\mathcal{T}_k^{(\delta)}$ is symmetric, it has an eigendecomposition of the form

$$\mathcal{T}_k^{(\delta)} = \mathcal{Q}_k^{(\delta)}\,\mathcal{D}_k^{(\delta)}\,\left(\mathcal{Q}_k^{(\delta)}\right)^T,$$

where $\mathcal{Q}_k^{(\delta)}$ is orthogonal and $\mathcal{D}_k^{(\delta)}$ is diagonal. In particular,

$$\mathcal{Q}_k^{(1)} = \sqrt{\frac{2}{k}}\left(q_j \cos\frac{(2p-1)(j-1)\pi}{2n}\right)_{1\leq p,j\leq k}$$

and

$$\mathcal{Q}_k^{(-1)} = \sqrt{\frac{2}{k}}\left(\cos\frac{(2p-1)(2j-1)\pi}{4k}\right)_{1\leq p,j\leq k},$$

where $q_1 = \frac{1}{2}$ and $q_j = 1$ for $2 \leq j \leq k$, and the corresponding diagonal matrices are

$$\mathcal{D}_k^{(1)} = 2\ \mathrm{diag}\left(1, \cos\frac{\pi}{k}, \ldots, \cos\frac{(k-1)\pi}{k}\right)$$

and

$$\mathcal{D}_k^{(-1)} = 2\ \mathrm{diag}\left(\cos\frac{\pi}{2k}, \cos\frac{3\pi}{2k}, \ldots, \cos\frac{(2k-1)\pi}{2k}\right).$$

Let $\Omega = \mathcal{T}_m^{(\delta_1)}$ and $\Lambda = \mathcal{T}_n^{(-\delta_1)}$. As is noted in Gohberg, Kailath, and Olshevsky [21], it is easy to verify that every Toeplitz-plus-Hankel matrix satisfies the displacement equation (1.1) with Δ having nonzero entries only in its first and last row and column, thus the displacement rank of a Toeplitz-plus-Hankel matrix is $\mathrm{rank}(\Delta) \leq 4$. In particular, this result is true for every Toeplitz or Hankel matrix. Let $\gcd(m,n)$ be the *greatest common divider* of m and n.

LEMMA 3.2. *Let $M \in \mathbf{C}^{m\times n}$ be a matrix satisfying the displacement equation*

$$\mathcal{T}_m^{(\delta_1)}\, M - M\, \mathcal{T}_n^{(-\delta_1)} = A\, B, \tag{3.3}$$

where $A \in \mathbf{C}^{m\times r}$ and $B \in \mathbf{C}^{r\times n}$. Then $C \equiv \left(\mathcal{Q}_m^{(\delta_1)}\right)^T M\, \mathcal{Q}_n^{(-\delta_1)}$ is a Cauchy-like matrix:

$$\Omega\, C - C\, \Lambda = \left(\left(\mathcal{Q}_m^{(\delta_1)}\right)^T A\right) \left(B\, \mathcal{Q}_n^{(-\delta_1)}\right),$$

where $\Omega = \mathcal{D}_m^{(\delta_1)}$ and $\Lambda = \mathcal{D}_n^{(-\delta_1)}$. We choose $\delta_1 = 1$ if $m/\gcd(m,n)$ is odd, and $\delta_1 = -1$ otherwise. Then the diagonals of Ω and Λ satisfy

$$\min_{k,j} |\omega_k - \lambda_j| \geq 4 \sin^2 \frac{\gcd(m,n)\, \pi}{4m\, n} \quad \text{and} \quad \min_{k\neq j} |\omega_k - \omega_j| \geq 4 \sin^2 \frac{\pi}{2m}.$$

Proof. The displacement equation for C can be proved using arguments similar to those in the proof of Lemma 3.1.

Let $\widehat{m} = m/\gcd(m,n)$ and $\widehat{n} = n/\gcd(m,n)$. Then $\widehat{m}$ and $\widehat{n}$ are integers and $\gcd(\widehat{m}, \widehat{n}) = 1$. First assume that $\widehat{m}$ is odd. Then $\delta_1 = 1$. It follows that

$$\begin{aligned} |\omega_k - \lambda_j| &= 2 \left| \cos \frac{(k-1)\pi}{m} - \cos \frac{(2j-1)\pi}{2n} \right| \\ &= 4 \left| \sin\left(\frac{2\left((k-1)\,\widehat{n} - j\,\widehat{m}\right) + \widehat{m}}{4\widehat{m}\,\widehat{n}\, \gcd(m,n)}\, \pi \right) \sin\left(\frac{2\left((k-1)\,\widehat{n} + j\,\widehat{m}\right) - \widehat{m}}{4\widehat{m}\,\widehat{n}\, \gcd(m,n)}\, \pi \right) \right| \\ &\geq 4 \left| \sin\left(\frac{\pi}{4\widehat{m}\,\widehat{n}\, \gcd(m,n)} \right) \sin\left(\frac{\pi}{4\widehat{m}\,\widehat{n}\, \gcd(m,n)} \right) \right| = 4 \sin^2 \frac{\gcd(m,n)\, \pi}{4m\, n}\ , \end{aligned}$$

where we have used the fact that both $2\left((k-1)\,\widehat{n} - j\,\widehat{m}\right) + \widehat{m}$ and $2\left((k-1)\,\widehat{n} + j\,\widehat{m}\right) - \widehat{m}$ are odd integers and hence are never integer multiples of $4\widehat{m}\,\widehat{n}\,\gcd(m,n)$, the denominator.

On the other hand, if $\widehat{m}$ is even, then it follows that $\widehat{n}$ is odd and $\delta_1 = -1$. Similar arguments show that $|\omega_k - \lambda_j| \geq 4\sin^2 \gcd(m,n)\, \pi/4m\, n$. The bound on $\min_{k\neq j} |\omega_k - \omega_j|$ can be easily verified in both cases. □

A matrix M is called *Toeplitz-plus-Hankel-like* if it satisfies the displacement equation (3.3) with $r \ll n$ (cf. [21]). In the least squares problem (1.5), if the matrix M is real Toeplitz-plus-Hankel-like, we transform it into a real Cauchy-like matrix C using Lemma 3.2:

$$M = \mathcal{Q}_m^{(\delta_1)}\, C\, \left(\mathcal{Q}_n^{(-\delta_1)}\right)^T.$$

Plugging this relation into (1.6) and simplifying, we get $x_M = \mathcal{Q}_n^{(-\delta_1)}\, x_C$, where

$$x_C = \left(C^T\, C\right)^{-1}\, C^T\, \left(\left(\mathcal{Q}_m^{(\delta_1)}\right)^T\, h\right).$$

Similar to Algorithm 3.1, Algorithm 3.2 that follows solves the least squares problem with M satisfying (3.3). We assume that both M and h are real.

ALGORITHM 3.2. Solving the Toeplitz-plus-Hankel-like least squares problem.

1. Choose δ_1 and compute Ω and Λ in Lemma 3.2;
2. compute $h := \left(\mathcal{Q}_m^{(\delta_1)}\right)^T h$, $A := \left(\mathcal{Q}_m^{(\delta_1)}\right)^T A$, and $B := B\,\mathcal{Q}_n^{(-\delta_1)}$;
3. choose ζ_1, LU factorize C_1, and compute the generator for Z for C_1 and Z in (2.5) via Algorithm 2.1;
4. choose ζ_3, compute the initial generator for K in (2.5) via Theorem 2.2, and compute the Cholesky factorization of K via Algorithm 2.2;
5. compute x_C by solving the linear systems in (2.5) via forward and backward substitution;
6. compute $x_M := \mathcal{Q}_n^{(-\delta_1)}\, x_C$.

Remark 3.2. The total cost of steps 1, 2, and 6 is $O(m \log_2 m)$ flops via fast cosine transforms; the cost of step 3 is about $(4r+4)mn + (2r-1)n^2$ flops, plus the cost of about $(4m-n)nr^2/\zeta_1$ flops for redecomposing the generators (see Remark 2.3); the cost of step 4 is about $(4r+1)mn + 1.5n^2$ flops, plus an extra cost of about $2n^2r^2/\zeta_3$ flops for redecomposing the generators (see Remarks 2.1 and 2.6). Since the matrix Z is computed in Step 4 (see Remark 2.1), the cost for step 5 is about $2(m+n)n$ flops. Hence, the total cost of Algorithm 3.2 is about $(8r+7)mn + (2r+2.5)n^2$ flops, plus the cost for generator redecompositions. In particular, let M be a Toeplitz-plus-Hankel matrix; then $r \le 4$. We choose $\zeta_1 \gg 4$ and $\zeta_3 \gg 4$. Thus the total cost of Algorithm 3.2 is about $(39m + 10.5n)n$ flops.

3.3. Efficiency and accuracy considerations. Every Toeplitz matrix is a Toeplitz-plus-Hankel matrix, hence Algorithm 3.2 is an algorithm for solving real Toeplitz least squares problems as well. For such a problem, Algorithm 3.1 performs 2.7 times as many flops as Algorithm 3.2 and hence is much slower (see Remarks 3.1 and 3.2).

Gu [27] has performed an error analysis on both Algorithms 3.1 and 3.2. The methodology in this error analysis is similar to that of Chandrasekaran and Sayed [11, 12], Gu [28], and Sweet and Brent [48] for analyzing fast algorithms for solving various structured linear systems of equations. We summarize related results in Gu [27] in Theorem 3.1 below.

THEOREM 3.1. *Let $\widehat{x}_M$ be the computed solution of Algorithm* 3.1 *or* 3.2 *to the linear least squares problem* (1.5) *with $\zeta_1 = \zeta_2 = \zeta_3 = 1$, and let* $\widehat{L} = \begin{pmatrix} \widehat{L}_1 & 0 \\ \widehat{L}_2 & I_{m-n} \end{pmatrix}$ *be the $m \times m$ lower triangular matrix computed by the call to Algorithm* 2.1 *from Algorithm* 3.1 or 3.2. *Then $\widehat{x}_M$ is the exact solution to the linear least squares problem*

$$\min_x \|\,(M + \delta M)\; x - h\|_2\,, \quad \textit{with} \quad \|\delta M\|_2 \le \eta\,\left(m\,n\,(\psi+\phi)\,\kappa\left(\widehat{L}\right)\right)^2 \|M\|_2 + O(\epsilon^2),$$

where η is a small multiple of the machine precision ϵ, $\kappa\left(\widehat{L}\right) = \left\|\widehat{L}\right\|_\infty \left\|\widehat{L}^{-1}\right\|_\infty$, and

$$\psi = \frac{\max_{k,j} |\omega_k| + |\lambda_j|}{\min_{k,j} |\omega_k - \lambda_j|} \quad \textit{and} \quad \phi = \frac{2\,\max_k |\omega_k|}{\min_{k\ne j} |\omega_k - \omega_j|}.$$

$\Omega = \operatorname{diag}(\omega_1, \ldots, \omega_m)$ *and* $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$ *are defined in Lemma* 3.1 *for Algorithm* 3.1 *and in Lemma* 3.2 *for Algorithm* 3.2.

As commented in Gu [27], upper bounds similar to those in Theorem 3.1 hold for the case $\zeta_1 > 1$, $\zeta_2 > 1$, and $\zeta_3 > 1$ as well, provided that there is little element growth within the generators for Algorithms 2.1 through 2.3 (see section 2).

The upper bounds on g_{CP} (Theorem 2.6) and backward error (Theorem 3.1) suggest that the smaller ψ and ϕ are, the smaller the potential element growth and backward error will be. In our numerical experiments, we took $\delta = n$ in Algorithm 3.1. Hence, $\psi = O(n)$ and $\phi = O(m)$ for Algorithm 3.1, whereas $\psi = O(m^2 n^2 / \gcd^2(m,n))$ and $\phi = O(m^2)$ for Algorithm 3.2 (see Lemma 3.2). Hence, Theorem 3.1 suggests that Algorithm 3.1 could be more accurate than Algorithm 3.2, even though it is less efficient.

It is well known that $\kappa(\widehat{L})$ could be as large as $O(2^n)$ in the worst case if the Cauchy-like matrix C factorized in Algorithm 2.1 was a general dense matrix (see section 2.3). Consequently the upper bound on $\|\delta M\|_2$ could be as large as $O(4^n)$ in the worst case. It is not clear whether a sharper upper bound exists for Cauchy-like matrices with low displacement rank. On the other hand, even if $\kappa(\widehat{L}) \approx 1$, the upper bound on $\|\delta M\|_2$ in Theorem 3.1 is still much larger than that for the QR method, which is $O\left(\epsilon\, m\, n \|M\|_2\right)$ (see Demmel [19]). Hence, Algorithms 3.1 and 3.2 appear to be less accurate than QR.

Our numerical experiments indicate that Algorithm 3.2 is indeed, in general, less accurate than Algorithm 3.1, which in turn is, in general, less accurate than QR, but the lost accuracy can be recovered by one step of Björck iterative refinement [2], which costs about $21n^2$ flops for Algorithm 3.1 and $6n^2$ flops for Algorithm 3.2, respectively.

A number of fast algorithms has been developed over the years to solve the Toeplitz and Toeplitz-plus-Hankel least squares problems (see section 1.2). Among them, the algorithm proposed recently by Park and Eldén [44] appears to be more stable than others but is less efficient, requiring about $(9m + 18.5n)n$ flops. Algorithm 3.2 with Björck iterative refinement is about four times as expensive as the Park and Eldén algorithm. However, to the best of our knowledge, no previous fast algorithm is known to be numerically stable for very ill conditioned problems, whereas our new algorithms can solve problems ranging from well conditioned to very ill conditioned to numerically singular. See section 4 for more details.

4. Numerical experiments.

We have implemented Algorithms 2.1 through 3.2 in Fortran and have performed a large number of numerical experiments with them to investigate their behavior in finite arithmetic and to compare Algorithms 3.1 and 3.2 with other available algorithms. In this section we discuss some issues related to measuring backward errors and implementation, and report some of the numerical results. We chose $\zeta_1 = 10, \zeta_2 = \zeta_3 = m$, and $\delta = n$ in Algorithms 3.1 and 3.2. In other words, very few steps of complete pivoting and generator redecomposition were performed in our numerical experiments. As was also observed by Gohberg, Kailath, and Olshevsky [21], partial pivoting is usually sufficient to guarantee stability in factorizing Cauchy matrices, and generator redecomposition is usually needed only at the beginning (see Sweet and Brent [48]).

4.1. Implementation issues.

A natural way to implement Algorithm 2.1 is to keep P_m and Q_n in vectors and keep both L and U in a single matrix W by storing L in the strict lower triangular part of W and U in the upper triangular part of W. However, arrays are stored columnwise in Fortran. Since U is generated and stored row-by-row in Algorithm 2.1, columns of W have to be moved into and brought out of fast memory for most steps of elimination for large n. This causes a significant amount of memory movement between slow and fast levels in the memory hierarchy. For more detailed discussions on memory movement, see, for example, [19, section 2.6]. As in Gu [28], we reduce this memory movement by storing rows of U columnwise in Algorithm 2.1. Let $S_n \in \mathbf{R}^{n\times n}$ be the matrix that is 1 on the main antidiagonal and 0

Table 1
Fortran BLAS, large residuals.

Matrix type	Order		$\frac{1}{\kappa(T)}$	Execution time (seconds)				
	m	n		NEW-I	NEW-II	NEW-III	NEW-IV	QR
1	320	300	2×10^{-3}	2×10^{-1}	3×10^{-1}	4×10^{-1}	4×10^{0}	9×10^{-1}
	640	600	1×10^{-3}	8×10^{-1}	9×10^{-1}	2×10^{0}	2×10^{0}	6×10^{0}
	1280	1200	1×10^{-3}	3×10^{0}	3×10^{0}	7×10^{0}	7×10^{0}	5×10^{1}
	2560	2400	8×10^{-4}	1×10^{1}	1×10^{1}	3×10^{1}	3×10^{1}	4×10^{2}
2	320	300	6×10^{-17}	2×10^{-1}	2×10^{-1}	4×10^{-1}	4×10^{-1}	8×10^{-1}
	640	600	4×10^{-17}	8×10^{-1}	9×10^{-1}	2×10^{0}	2×10^{0}	6×10^{0}
	1280	1200	3×10^{-17}	3×10^{0}	3×10^{0}	6×10^{0}	7×10^{0}	5×10^{1}
	2560	2400	1×10^{-17}	1×10^{1}	1×10^{1}	3×10^{1}	3×10^{1}	4×10^{2}
3	320	300	4×10^{-9}	2×10^{-1}	2×10^{-1}	4×10^{-1}	4×10^{-1}	8×10^{-1}
	640	600	2×10^{-13}	9×10^{-1}	8×10^{-1}	2×10^{0}	2×10^{0}	6×10^{0}
	1280	1200	2×10^{-15}	3×10^{0}	3×10^{0}	6×10^{0}	7×10^{0}	5×10^{1}
	2560	2400	5×10^{-15}	1×10^{1}	1×10^{1}	3×10^{1}	3×10^{1}	4×10^{2}

Matrix type	Order		$\frac{1}{\kappa(T)}$	Backward error (τ)				
	m	n		NEW-I	NEW-II	NEW-III	NEW-IV	QR
1	320	300	2×10^{-3}	2×10^{1}	2×10^{-1}	1×10^{0}	1×10^{-1}	3×10^{-2}
	640	600	1×10^{-3}	8×10^{1}	3×10^{-1}	1×10^{0}	3×10^{-1}	2×10^{-2}
	1280	1200	1×10^{-3}	1×10^{3}	3×10^{-1}	2×10^{0}	3×10^{-1}	2×10^{-2}
	2560	2400	8×10^{-4}	3×10^{2}	1×10^{-1}	8×10^{0}	1×10^{-1}	1×10^{-2}
2	320	300	6×10^{-17}	1×10^{0}	9×10^{-1}	6×10^{-1}	6×10^{-1}	3×10^{-1}
	640	600	4×10^{-17}	2×10^{0}	1×10^{0}	5×10^{-1}	5×10^{-1}	3×10^{-1}
	1280	1200	3×10^{-17}	7×10^{-1}	5×10^{-1}	2×10^{0}	2×10^{0}	2×10^{-1}
	2560	2400	1×10^{-17}	2×10^{0}	2×10^{0}	3×10^{0}	3×10^{0}	2×10^{-1}
3	320	300	4×10^{-9}	6×10^{1}	5×10^{-1}	4×10^{0}	5×10^{-1}	1×10^{-1}
	640	600	2×10^{-13}	9×10^{1}	2×10^{2}	4×10^{0}	5×10^{-1}	1×10^{-1}
	1280	1200	2×10^{-15}	2×10^{1}	3×10^{1}	9×10^{0}	1×10^{1}	2×10^{-1}
	2560	2400	5×10^{-15}	3×10^{1}	4×10^{1}	2×10^{1}	3×10^{1}	1×10^{-1}

everywhere else. Then $\widetilde{U} \equiv S_n\, U^T\, S_n$ is an upper triangular matrix, whose kth column is the $(n-k+1)$st row of U in the reverse order. The backward substitution procedure for computing $U^{-1}\, y$ in Algorithm 2.1 can be rewritten as a forward substitution as $S_n\, ((\widetilde{U}^T)^{-1}\, (S_n\, y))$. Gu [28] shows that this technique significantly reduces memory movement.

We further reduce the memory movement by delaying permuting rows of L in Algorithms 2.1 through 2.3 until the factorization is completed. To be more specific, assume that at the kth step of elimination, we need to swap two rows of L in order to bring the pivot to the (k, k) position. We achieve this by only swapping the corresponding rows in the generator matrix. After the factorization is completed, we restore rows of L to their proper positions. This technique is also used to swap columns of U in Algorithm 2.1.

4.2. Numerical results. The computations were done on an IBM RS6000 workstation in double precision, where the machine precision is $\epsilon \approx 1.1 \times 10^{-16}$. We compared the following algorithms:[3]

[3] No software implementation of any of the fast algorithms for solving Toeplitz and Toeplitz-plus-Hankel least squares problems is available in the public domain. Hence, we only compared our new algorithms with the implementation of dense QR in LAPACK on netlib.

Table 2
Fortran BLAS, small residuals.

Matrix	Order		$\frac{1}{\kappa(T)}$	Execution time (seconds)				
type	m	n		NEW-I	NEW-II	NEW-III	NEW-IV	QR
1	320	300	2×10^{-3}	2×10^{-1}	3×10^{-1}	4×10^{-1}	5×10^{-1}	8×10^{-1}
	640	600	1×10^{-3}	8×10^{-1}	9×10^{-1}	2×10^{0}	2×10^{0}	6×10^{0}
	1280	1200	1×10^{-3}	3×10^{0}	3×10^{0}	6×10^{0}	7×10^{0}	5×10^{1}
	2560	2400	8×10^{-4}	1×10^{1}	1×10^{1}	3×10^{1}	3×10^{1}	4×10^{2}
2	320	300	6×10^{-17}	2×10^{-1}	2×10^{-1}	4×10^{-1}	5×10^{-1}	8×10^{-1}
	640	600	4×10^{-17}	8×10^{-1}	9×10^{-1}	2×10^{0}	2×10^{0}	6×10^{0}
	1280	1200	3×10^{-17}	3×10^{0}	3×10^{0}	7×10^{0}	7×10^{0}	5×10^{1}
	2560	2400	1×10^{-17}	1×10^{1}	1×10^{1}	3×10^{1}	3×10^{1}	4×10^{2}
3	320	300	4×10^{-9}	2×10^{-1}	2×10^{-1}	4×10^{-1}	4×10^{-1}	8×10^{-1}
	640	600	2×10^{-13}	8×10^{-1}	9×10^{-1}	2×10^{0}	2×10^{0}	6×10^{0}
	1280	1200	2×10^{-15}	3×10^{0}	3×10^{0}	6×10^{0}	7×10^{0}	5×10^{1}
	2560	2400	5×10^{-15}	1×10^{1}	1×10^{1}	3×10^{1}	3×10^{1}	4×10^{2}

Matrix	Order		$\frac{1}{\kappa(T)}$	Normalized residual (τ)				
type	m	n		NEW-I	NEW-II	NEW-III	NEW-IV	QR
1	320	300	2×10^{-3}	7×10^{2}	3×10^{0}	7×10^{0}	3×10^{0}	3×10^{-1}
	640	600	1×10^{-3}	4×10^{3}	8×10^{0}	1×10^{0}	8×10^{0}	5×10^{-1}
	1280	1200	1×10^{-3}	9×10^{4}	1×10^{1}	1×10^{1}	1×10^{1}	3×10^{-1}
	2560	2400	8×10^{-4}	4×10^{4}	5×10^{0}	5×10^{1}	5×10^{0}	3×10^{-1}
2	320	300	6×10^{-17}	2×10^{1}	9×10^{-1}	1×10^{0}	4×10^{-1}	3×10^{-1}
	640	600	4×10^{-17}	2×10^{2}	8×10^{-1}	3×10^{0}	4×10^{-1}	2×10^{-1}
	1280	1200	3×10^{-17}	5×10^{1}	5×10^{-1}	3×10^{0}	3×10^{0}	2×10^{-1}
	2560	2400	1×10^{-17}	2×10^{1}	2×10^{0}	4×10^{0}	3×10^{0}	2×10^{-1}
3	320	300	4×10^{-9}	4×10^{1}	1×10^{0}	2×10^{1}	1×10^{0}	3×10^{-1}
	640	600	2×10^{-13}	2×10^{2}	4×10^{2}	1×10^{1}	5×10^{0}	3×10^{-1}
	1280	1200	2×10^{-15}	5×10^{1}	2×10^{1}	1×10^{1}	2×10^{1}	3×10^{-1}
	2560	2400	5×10^{-15}	9×10^{1}	7×10^{1}	3×10^{1}	3×10^{1}	3×10^{-1}

- NEW-I: Algorithm 3.2 *without* iterative refinement; about $(39m + 10.5n)n$ flops.
- NEW-II: Algorithm 3.2 *with* iterative refinement; about $(39m+16.5n)n$ flops.
- NEW-III: Algorithm 3.1 *without* iterative refinement; about $(109m + 25n)n$ flops.
- NEW-IV: Algorithm 3.1 *with* iterative refinement; about $(109m+46n)n$ flops.
- QR: LAPACK [1] subroutine DGELS for solving a general dense linear least squares problem using the QR method; about $(2m - 2/3n)n^2$ flops.

We solved the Toeplitz(-plus-Hankel) linear least squares problem

$$\min_x \| (T + H)\ x - h\|_2$$

for the following types of Toeplitz matrices $T = (t_{k-j})_{1\leq k\leq m, 1\leq j\leq n}$ and Hankel matrices $H = (h_{k+j-2})_{1\leq k\leq m, 1\leq j\leq n}$:

- Type 1: $\{t_k\}$ and $\{h_j\}$ were randomly generated from uniform distribution on $(0, 1)$. A Type 1 matrix is usually well conditioned.
- Type 2: $h_j = 0$. $t_0 = 2\omega$ and $t_k = \frac{\sin(2\pi\omega k)}{\pi k}$ for $k \neq 0$, where $\omega = 0.25$. A Type 2 matrix is also called the Prolate matrix in Gohberg, Kailath, and Olshevsky [21] and Varah [49]; T is very ill conditioned.

Table 3
Optimized BLAS, large residuals.

Matrix type	Order		$\frac{1}{\kappa(T)}$	Execution time (seconds)				
	m	n		NEW-I	NEW-II	NEW-III	NEW-IV	QR
1	320	300	2×10^{-3}	2×10^{-1}	2×10^{-1}	4×10^{-1}	5×10^{-1}	3×10^{-1}
	640	600	1×10^{-3}	8×10^{-1}	8×10^{-1}	2×10^{0}	2×10^{0}	2×10^{0}
	1280	1200	1×10^{-3}	3×10^{0}	3×10^{0}	6×10^{0}	6×10^{0}	2×10^{1}
	2560	2400	8×10^{-4}	1×10^{1}	1×10^{1}	2×10^{1}	2×10^{1}	1×10^{2}
2	320	300	4×10^{-17}	2×10^{-1}	2×10^{-1}	4×10^{-1}	4×10^{-1}	3×10^{-1}
	640	600	4×10^{-17}	7×10^{-1}	8×10^{-1}	1×10^{0}	2×10^{0}	2×10^{0}
	1280	1200	2×10^{-17}	3×10^{0}	3×10^{0}	6×10^{0}	6×10^{0}	2×10^{1}
	2560	2400	7×10^{-18}	1×10^{1}	1×10^{1}	2×10^{1}	2×10^{1}	1×10^{2}
3	320	300	4×10^{-9}	2×10^{-1}	2×10^{-1}	4×10^{-1}	4×10^{-1}	3×10^{-1}
	640	600	2×10^{-13}	8×10^{-1}	8×10^{-1}	1×10^{0}	2×10^{0}	2×10^{0}
	1280	1200	2×10^{-15}	3×10^{0}	3×10^{0}	6×10^{0}	6×10^{0}	2×10^{1}
	2560	2400	5×10^{-15}	1×10^{1}	1×10^{1}	2×10^{1}	2×10^{1}	1×10^{2}

Matrix type	Order		$\frac{1}{\kappa(T)}$	Backward error (τ)				
	m	n		NEW-I	NEW-II	NEW-III	NEW-IV	QR
1	320	300	2×10^{-3}	2×10^{-1}	3×10^{-1}	4×10^{-1}	5×10^{-1}	8×10^{-1}
	640	600	1×10^{-3}	9×10^{1}	3×10^{-1}	1×10^{0}	3×10^{-1}	2×10^{-2}
	1280	1200	1×10^{-3}	1×10^{3}	3×10^{-1}	2×10^{0}	3×10^{-1}	2×10^{-2}
	2560	2400	8×10^{-4}	4×10^{2}	1×10^{-1}	8×10^{0}	1×10^{-1}	1×10^{-2}
2	320	300	4×10^{-17}	1×10^{0}	7×10^{-1}	8×10^{-1}	7×10^{-1}	2×10^{-1}
	640	600	4×10^{-17}	2×10^{0}	1×10^{0}	6×10^{-1}	7×10^{-1}	3×10^{-1}
	1280	1200	2×10^{-17}	9×10^{-1}	7×10^{-1}	2×10^{0}	2×10^{0}	2×10^{-1}
	2560	2400	7×10^{-18}	2×10^{0}	2×10^{0}	3×10^{0}	3×10^{0}	2×10^{-1}
3	320	300	4×10^{-9}	6×10^{1}	5×10^{-1}	4×10^{0}	5×10^{-1}	1×10^{-1}
	640	600	2×10^{-13}	2×10^{2}	8×10^{2}	4×10^{0}	5×10^{-1}	1×10^{-1}
	1280	1200	2×10^{-15}	3×10^{1}	5×10^{1}	1×10^{1}	2×10^{1}	1×10^{-1}
	2560	2400	5×10^{-15}	2×10^{1}	4×10^{1}	2×10^{1}	3×10^{1}	1×10^{-1}

- Type 3: A set of data was generated as

$$x_k = e^{-\alpha\, k} \sum_{j=1}^{\lfloor m/3 \rfloor} \frac{j}{\lfloor m/3 \rfloor + 1} \cos\left(\frac{j\, k\, \pi}{\lfloor m/3 \rfloor + 1} \right) + \beta\, r_k,$$

  where $\alpha = \pi/n$; r_k is taken from a normal distribution, $r_k \in N(0,1)$, and $\beta = 10^{-7}$, 10^{-11}, 10^{-15}, and 10^{-18}. The matrix $T \in \mathbf{R}^{m\times n}$ was constructed from $t_{k-s} = x_{k-s+n}$ and we set $H = 0$. This is a modification of Test V in Park and Eldén [44], which, in turn, is a modification of an example from Kolbel and Schafer [38]. T becomes more and more ill conditioned as β becomes smaller and smaller.

We chose two types of right-hand side vectors h:

- Components of h were randomly generated from uniform distribution on $(0,1)$. The linear least squares problem has a *large* residual.
- Components of x_M were randomly generated from uniform distribution on $(0,1)$ and $h = (T + H)\, x_M$. The problem has a *small* residual.

One way to measure the accuracy in a solution $\widehat{x}_M$ computed by either one of our new algorithms or QR is to compute the normwise smallest backward error δM such that $\widehat{x}_M$ is the exact solution to the linear least squares problem (see Theorem 3.1)

$$\min_x \| \left(M + \delta M \right) \, x - h\|_2. \tag{4.1}$$

To do so, we use the following theorem of Gu [26], which is a modification of a result of Waldén, Karlson, and Sun [50] and Higham [32, Chapter 19]. Let $M = Q \begin{pmatrix} D \\ 0 \end{pmatrix} W^T$ be the singular value decomposition of M, where $Q \in \mathbf{R}^{m\times m}$ and $W \in \mathbf{R}^{n\times n}$ are orthogonal, and $D \in \mathbf{R}^{n\times n}$ is nonnegative diagonal. Assume that $\hat{x}_M \neq 0$ and let

$$r = h - M\,\hat{x}_M = Q \begin{pmatrix} r_1 \\ r_2 \end{pmatrix} \quad \text{for} \quad r_1 \in \mathbf{R}^n\ , \quad r_2 \in \mathbf{R}^{m-n}, \quad \text{and} \quad \eta = \frac{\|r\|_2}{\|\hat{x}_M\|_2}.$$

THEOREM 4.1. *Define*

$$\tau = \frac{\min\left(\eta, \tilde{\sigma}\right)}{\sqrt{m}\,\|M\|_2\,\epsilon}, \quad \textit{where} \quad \tilde{\sigma} = \sqrt{\frac{r_1^T\,D^2\,\left(D^2+\eta^2 I\right)^{-1}\,r_1}{\|r_2\|_2^2/\eta^2 + \eta^2\,r_1^T\,\left(D^2+\eta^2 I_n\right)^{-2}\,r_1}}.$$

Then the Frobenius normwise smallest backward error δM such that $\hat{x}_M$ is the exact solution to the least squares problem (4.1) *satisfies*

$$\frac{\sqrt{5}-1}{2} \leq \frac{\|\delta M\|_F}{\sqrt{m}\,\tau\,\|M\|_2\,\epsilon} \leq 1.$$

According to Theorem 4.1, $\widehat{x}_M$ is a good approximate solution if and only if $\tau \leq O(1)$.

Numerical results are summarized in Tables 1 through 4. These results confirm that both Algorithms 3.1 and 3.2 are capable of solving problems ranging from well conditioned to ill conditioned to numerically singular, both for large residuals and small residuals, and that they are significantly faster than QR.

Algorithm 3.2 is the fastest algorithm, whereas QR is the slowest. Flop counts indicate that NEW-I and NEW-II break even with QR at $n \approx 40$. From our numerical experiments, for $m = 2560$ and $n = 2400$, Algorithm 3.2 is up to 30 times faster than QR in Fortran BLAS and up to 10 times faster in optimized BLAS; Algorithm 3.1 is up to 15 times faster than QR in Fortran BLAS and up to 5 times faster in optimized BLAS. One reason Algorithms 3.1 and 3.2 appear less efficient than flop counts indicate is that most of the computations in Algorithms 3.1 and 3.2 are in level-1 and level-2 BLAS, whereas most of the computations in QR are in level-3 BLAS. If these algorithms are implemented in systolic arrays, one might expect the speedups of Algorithms 3.1 and 3.2 to be more in line with what flop counts indicate.

Algorithm 3.2 is the least accurate, whereas QR is the most accurate. For $m = 2560$ and $n = 2400$, the backward error in Algorithm 3.2 is up to 10^5 times larger than that in QR. With iterative refinement, the backward error in Algorithm 3.2 is at most 200 times as large. While Algorithm 3.1 costs at least twice as much as Algorithm 3.2, it is significantly more accurate. With or without iterative refinement, the backward error in Algorithm 3.1 is at most 200 times larger than that in QR.

5. Conclusions and extensions. We have presented fast algorithms for solving the Toeplitz and Toeplitz-plus-Hankel linear least squares problems and shown them to be numerically stable under certain conditions. We have discussed implementation

Table 4
Optimized BLAS, small residuals.

Matrix type	Order m	Order n	$\frac{1}{\kappa(T)}$	Execution time (seconds) NEW-I	NEW-II	NEW-III	NEW-IV	QR
1	320	300	2×10^{-3}	2×10^{-1}	3×10^{-1}	4×10^{-1}	5×10^{-1}	3×10^{-1}
	640	600	1×10^{-3}	8×10^{-1}	8×10^{-1}	1×10^{0}	2×10^{0}	2×10^{0}
	1280	1200	1×10^{-3}	3×10^{0}	3×10^{0}	6×10^{0}	6×10^{0}	2×10^{1}
	2560	2400	8×10^{-4}	1×10^{1}	1×10^{1}	2×10^{1}	2×10^{1}	1×10^{2}
2	320	300	4×10^{-17}	2×10^{-1}	2×10^{-1}	4×10^{-1}	4×10^{-1}	3×10^{-1}
	640	600	4×10^{-17}	7×10^{-1}	8×10^{-1}	2×10^{0}	2×10^{0}	2×10^{0}
	1280	1200	2×10^{-17}	3×10^{0}	3×10^{0}	6×10^{0}	6×10^{0}	2×10^{1}
	2560	2400	7×10^{-18}	1×10^{1}	1×10^{1}	2×10^{1}	2×10^{1}	1×10^{2}
3	320	300	4×10^{-9}	2×10^{-1}	2×10^{-1}	4×10^{-1}	4×10^{-1}	3×10^{-1}
	640	600	2×10^{-13}	8×10^{-1}	8×10^{-1}	1×10^{0}	2×10^{0}	2×10^{0}
	1280	1200	2×10^{-15}	3×10^{0}	3×10^{0}	6×10^{0}	6×10^{0}	2×10^{1}
	2560	2400	5×10^{-15}	1×10^{1}	1×10^{1}	2×10^{1}	2×10^{1}	1×10^{2}

Matrix type	Order m	Order n	$\frac{1}{\kappa(T)}$	Backward error (τ) NEW-I	NEW-II	NEW-III	NEW-IV	QR
1	320	300	2×10^{-3}	2×10^{-1}	3×10^{-1}	4×10^{-1}	5×10^{-1}	8×10^{-1}
	640	600	1×10^{-3}	4×10^{3}	8×10^{0}	1×10^{0}	9×10^{0}	4×10^{-1}
	1280	1200	1×10^{-3}	7×10^{4}	1×10^{1}	1×10^{1}	1×10^{1}	5×10^{-1}
	2560	2400	8×10^{-4}	5×10^{4}	5×10^{0}	5×10^{1}	5×10^{0}	4×10^{-1}
2	320	300	4×10^{-17}	2×10^{1}	1×10^{0}	8×10^{-1}	4×10^{-1}	3×10^{-1}
	640	600	4×10^{-17}	3×10^{2}	7×10^{-1}	7×10^{-1}	7×10^{-1}	3×10^{-1}
	1280	1200	2×10^{-17}	7×10^{1}	7×10^{-1}	2×10^{0}	1×10^{0}	2×10^{-1}
	2560	2400	7×10^{-18}	4×10^{1}	2×10^{0}	4×10^{0}	4×10^{0}	2×10^{-1}
3	320	300	4×10^{-9}	4×10^{1}	1×10^{0}	2×10^{1}	1×10^{0}	4×10^{-1}
	640	600	2×10^{-13}	2×10^{2}	8×10^{2}	1×10^{1}	5×10^{0}	4×10^{-1}
	1280	1200	2×10^{-15}	3×10^{1}	5×10^{1}	2×10^{1}	2×10^{1}	4×10^{-1}
	2560	2400	6×10^{-15}	6×10^{1}	8×10^{1}	2×10^{1}	3×10^{1}	4×10^{-1}

techniques that further improve their efficiency. Numerical experiments indicate that they are both numerically stable and efficient in practice.

The algorithms presented in this paper can be modified to solve *mosaic Toeplitz* or *block Toeplitz* linear least squares problems (cf. [13, 21]).

As Theorem 3.1 indicates, these new algorithms could be numerically unstable if $\kappa(\widehat{L})$ is large. The best available upper bound on $\kappa(\widehat{L})$ is $O(2^n)$, which has never been seen in practice. It is not clear whether a sharper bound on $\kappa(\widehat{L})$ exists for Cauchy-like matrices with low displacement rank.

While the element growth factor for GECP is much smaller than that for GEPP, these factors remain largely theoretical. It would be interesting to know if there are cases where complete pivoting is significantly more accurate than partial pivoting. Another related issue is to study the impact of the different choices of ζ_j's on the accuracy of the numerical solution.

One way to reduce the upper bound on $\kappa(\widehat{L})$ is to perform a *rank-revealing LU* (RRLU) factorization on C instead of using GEPP/GECP (see, for example, Chan [10], Gu and Eisenstat [29], and Hwang, Lin, and Yang [33]). It would be interesting to see if fast RRLU factorization algorithms can be developed to guarantee that $\kappa(\widehat{L})$ is *always* modest.

Acknowledgments. The author is grateful to Profs. A. Björck and S. Chandrasekaran for helpful discussions, Prof. Golub for pointing out to him reference [42], and the anonymous referees for many helpful suggestions that significantly improved the presentation of this paper.

REFERENCES

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen, *LAPACK Users' Guide*, 2nd ed., SIAM, Philadelphia, PA, 1995.

[2] A. Björck, *Iterative refinement of linear least squares solutions* I, BIT, 7 (1967), pp. 257–278.

[3] A. Björck, *personal communication*, Dept. of Math., Linköping University, Linköping, Sweden, 1996.

[4] A. Bojanczyk and R. P. Brent, *Parallel solution of certain Toeplitz least-squares problems*, Linear Algebra Appl., 77 (1986), pp. 43–60.

[5] A. W. Bojanczyk, R. P. Brent, and F. de Hoog, *QR factorization of Toeplitz matrices*, Numer. Math., 49 (1986), pp. 81–94.

[6] A. W. Bojanczyk, R. P. Brent, F. R. de Hoog, and D. R. Sweet, *On the stability of the Bareiss and related Toeplitz factorization algorithms*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 40–57.

[7] R. P. Brent, *Parallel algorithms for Toeplitz systems*, in Numerical Linear Algebra, Digital Signal Processing and Parallel Algorithms, G. H. Golub and P. van Dooren, eds., Springer-Verlag, New York, 1990.

[8] J. Bunch and L. Kaufman, *Some stable methods for calculating inertia and solving symmetric linear systems*, Math. Comput., 31 (1977), pp. 163–179.

[9] J. R. Bunch, *Stability of methods for solving Toeplitz systems of equations*, SIAM J. Sci. Stat. Comput., 6 (1985), pp. 349–364.

[10] T. F. Chan, *On the existence and computation of LU-factorizations with small pivots*, Math. Comp., 42 (1984), pp. 535–547.

[11] S. Chandrasekaran and A. H. Sayed, *A fast stable solver for nonsymmetric Toeplitz and quasi-Toeplitz systems of linear equations*, SIAM J. Matrix Anal. Appl., 19 (1998), pp. 107–139.

[12] S. Chandrasekaran and A. H. Sayed, *Stabilizing the generalized Schur algorithm*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 950–983.

[13] J. Chun and T. Kailath, *Generalized displacement structure for block-Toeplitz, Toeplitz-block, and Toeplitz-derived matrices*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 114–128.

[14] J. Chun, T. Kailath, and H. Lev-Ari, *Fast parallel algorithms for QR and triangular factorization*, SIAM J. Sci. Stat. Comput., 8 (1987), pp. 899–913.

[15] G. Cybenko, *The numerical stability of the Levinson-Durbin algorithm for Toeplitz systems of equations*, SIAM J. Sci. Stat. Comput., 1 (1980), pp. 303–320.

[16] G. Cybenko, *A general orthorgonalization technique with applications to time series analysis and signal processing*, Math. Comp., 40 (1983), pp. 323–336.

[17] G. Cybenko, *Fast Toeplitz orthorgonalization using inner products*, SIAM J. Sci. Stat. Comput., 8 (1987), pp. 734–740.

[18] J. Daniel, W. B. Gragg, L. Kaufman, and G. W. Stewart, *Reorthogonalization and stable algorithms for updating the Gram–Schmidt QR factorization*, Math. Comp., 30 (1976), pp. 772–795.

[19] J. W. Demmel, *Applied Numerical Linear Algebra*, SIAM, Philadelphia, PA, 1997.

[20] M. Fiedler, *Hankel and Loewner matrices*, Linear Algebra Appl., 58 (1984), pp. 75–95.

[21] I. Gohberg, T. Kailath, and V. Olshevsky, *Fast Gaussian elimination with partial pivoting for matrices with displacement structure*, Math. Comp., 64 (1995), pp. 1557–1576.

[22] I. Gohberg and V. Olshevsky, *Complexity of multiplication with vectors for structured matrices*, Linear Algebra Appl., 202 (1994), pp. 163–192.

[23] I. Gohberg and V. Olshevsky, *Fast algorithm for matrix Nehari problem*, in Systems and Networks: Mathematical Theory and Applications, Proc. Internat. Symposium Mathematical Theory of Networks and Systems, (MTNS-93), U. Helmke, R. Mennicken, and J. Sauers, eds., Vol. 2, 1994, pp. 687–690.

[24] I. Gohberg and V. Olshevsky, *Fast state space algorithms for matrix Nehari and Nehari-Takagi interpolation problems*, Integral Equations Operator Theory, 20 (1994), pp. 44–83.

[25] G. Golub and C. van Loan, *Matrix Computations*, 3rd ed., The Johns Hopkins University Press, Baltimore, MD, 1996.

[26] M. Gu, *Backward perturbation bounds for linear least squares problems*, SIAM J. Matrix Anal. Appl., to appear.

[27] M. Gu, *New Fast Algorithms for Structured Linear Least Squares Problems*, LBL Report LBL-37878, Lawrence Berkeley National Laboratory, Berkeley, CA, 1995.

[28] M. Gu, *Stable and efficient algorithms for structured systems of linear equations*, SIAM J. Matrix Anal. Appl., 19 (1998), pp. 279–306.

[29] M. Gu and S. C. Eisenstat, *Efficient algorithms for computing a strong rank-revealing QR factorization*, SIAM J. Sci. Comput., 17 (1996), pp. 848–869.

[30] G. Heinig, *Inversion of generalized Cauchy matrices and other classes of structured matrices*, in Linear Algebra for Signal Processing, IMA Vol. Math. Appl. 69, Springer-Verlag, New York, 1995, pp. 95–114.

[31] G. Heinig and K. Rost, *Algebraic methods for Toeplitz-like matrices and operators*, Oper. Theory, 13 (1984), pp. 109–127.

[32] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia, PA, 1996.

[33] T.-M. Hwang, W.-W. Lin, and E. K. Yang, *Rank revealing LU factorizations*, Linear Algebra Appl., 175 (1992), pp. 115–141.

[34] T. Kailath, S. Kung, and M. Morf, *Displacement ranks of matrices and linear equations*, J. Math. Anal. Appl., 68 (1979), pp. 395–407.

[35] T. Kailath and V. Olshevsky, *Symmetric and Bunch-Kaufman Pivoting for Cauchy-like Matrices with Applications to Toeplitz-like Matrices*, manuscript.

[36] T. Kailath and A. H. Sayed, *Fast algorithms for generalized displacement structures*, in Recent Advances in Mathematical Theory of Systems, Control, Networks and Signal Processing, Vol. II, H. Kimura and S. Kodama, eds., Mita Press, Osaka, Japan, 1992, pp. 27–32.

[37] T. Kailath and A. H. Sayed, *Displacement structure: Theory and applications*, SIAM Rev., 37 (1995), pp. 297–386.

[38] W. Kolbel and H. Schafer, *Improvement and automation of the LPSVD algorithm by continuous regularization of the singular values*, J. Magnetic Resonance, 100 (1992), pp. 598–603.

[39] F. T. Luk and S. Qiao, *A fast but unstable orthogonal triangularization technique for Toeplitz matrices*, Linear Algebra Appl., 88/89 (1987), pp. 495–506.

[40] J. G. Nagy, *Fast inverse QR factorization for Toeplitz matrices*, SIAM J. Sci. Comput., 14 (1993), pp. 1174–1193.

[41] J. G. Nagy, *Applications of Toeplitz systems*, SIAM News, October 1995.

[42] B. Noble, *Computing the Moore-Penrose generalized inverse*, in Generalized Inverses and Applications, M. Z. Nashed, ed., Academic Press, New York, San Diego, 1976.

[43] V. Pan, *On computations with dense structured matrices*, Math. Comp., 55 (1990), pp. 179–190.

[44] H. Park and L. Eldén, *Stability Analysis and Fast Algorithms for Triangularization of Toeplitz Matrices*, Tech. Report Lith-Mat-R-95-16, Department of Mathematics, Linköping University, Linköping, Sweden, 1995.

[45] S. Qiao, *Hybrid algorithm for fast Toeplitz orthogonalization*, Numer. Math., 53 (1988), pp. 351–366.

[46] D. R. Sweet, *Fast Toeplitz orthogonalization*, Numer. Math., 43 (1984), pp. 1–21.

[47] D. R. Sweet, *The use of pivoting to improve the numerical performance of algorithms for Toeplitz matrices*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 468–493.

[48] D. R. Sweet and R. P. Brent, *Error analysis of a fast partial pivoting method for structured matrices*, in Advanced Signal Processing Algorithms, Proc. of SPIE, T. Luk, ed., Vol. 2363, 1995, pp. 266–280.

[49] J. M. Varah, *The Prolate matrix*, Linear Algebra Appl., 187 (1993), pp. 269–278.

[50] B. Waldén, R. Karlson, and J.-G. Sun, *Optimal backward perturbation bounds for the linear least square problem*, Numer. Linear Algebra Appl., 2 (1995), pp. 271–286.

[51] J. H. Wilkinson, *Error analysis of direct methods of matrix inversion*, J. Assoc. Comput. Mach., 10 (1961), pp. 281–330.

[52] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford University Press, Oxford, 1965.

SIAM J. Matrix Anal. Appl.
Vol. 20, No. 1, pp. 270–278

BLOCKWISE PERTURBATION THEORY FOR MARKOV CHAINS*

XUE JUNGONG† AND GAO WEIGUO†

Abstract. This paper is concerned with the relative sensitivity of the individual stationary probabilities of an irreducible Markov chain to small relative blockwise perturbations of the transition matrix. It is shown that this sensitivity depends on some quantities related to the stochastic complements of diagonal blocks of the transition matrix. For nearly uncoupled and nearly transient Markov chains, the stationary probabilities are sensitive to general perturbations in the relative sense. However, they are insensitive to small blockwise relative perturbations.

Key words. Markov chains, perturbation theory, nonnegative matrix, stationary distribution, transition matrix

AMS subject classifications. 65F35, 15A51, 60J10

PII. S0895479895294629

1. Introduction.

The sensitivity of the stationary distribution of an irreducible Markov chain has been addressed by many authors [9], [6], [4], [3]. Recently, much attention was focused on analyzing the relative sensitivity of the individual stationary probabilities. In [5], Ipsen and Meyer presented an important result: Let P and $\widetilde{P} = P + F$ be irreducible transition matrices with respective stationary distributions π^T and $\widetilde{\pi}^T$ satisfying

$$\pi^T P = \pi^T, \quad \widetilde{\pi}^T \widetilde{P} = \widetilde{\pi}^T, \quad \sum_i \pi_i = 1 = \sum_i \widetilde{\pi}_i.$$

Then

$$\frac{\pi_i - \widetilde{\pi}_i}{\pi_i} = \widetilde{\pi}^T F^{(i)} A_i^{-1} \mathbf{1} \tag{1.1}$$

and

$$\left| \frac{\pi_i - \widetilde{\pi}_i}{\pi_i} \right| \le \|F^{(i)}\| \cdot \|A_i^{-1}\|. \tag{1.2}$$

Here $F^{(i)}$ is the submatrix of F obtained by deleting the ith column and A_i is the principal submatrix of $A = I - P$ obtained by deleting the ith column and row. Throughout this paper, the norm $\| * \|$ is the one-norm for matrices and row vectors. We denote by $\mathbf{1}$ (a bold one) the vector of all ones. In later discussion, we give its dimension with a subscript (e.g., $\mathbf{1}_n$ for the vector with n entries) explicitly when necessary.

The relative sensitivity of the ith stationary probability to general perturbations is determined by $\|A_i^{-1}\|$. However, if F is an entrywise small relative perturbation to P, then $|\pi_i - \tilde{\pi}_i|/\pi_i$ must small. O'Cinneide's result [8] just states this: If $|f_{ij}| \le \eta |p_{ij}|$ for $i \ne j$, then

$$\left| \frac{\pi_i - \widetilde{\pi}_i}{\pi_i} \right| \le 2n\eta + O(\eta^2),$$

*Received by the editors November 10, 1995; accepted for publication (in revised form) by D. P. O'Leary October 22, 1997; published electronically September 29, 1998.

http://www.siam.org/journals/simax/20-1/29462.html

†Department of Mathematics, Fudan University, Shanghai 200433, People's Republic of China (jgxue@ms.fudan.edu.cn).

where n is the order of P. Similar results can also be found in [12] and [13]. However large $\|A_i^{-1}\|$ is, small relative errors in the entries of P result in small relative error in π_i.

The purpose of this paper is to extend O'Cinneide's result to the blockwise case. Let P and F be partitioned as

$$P = \begin{pmatrix} P_{11} & P_{12} & \cdots & P_{1t} \\ P_{21} & P_{22} & \cdots & P_{2t} \\ \vdots & \vdots & & \vdots \\ P_{t1} & P_{t2} & \cdots & P_{tt} \end{pmatrix} \quad \text{and} \quad F = \begin{pmatrix} F_{11} & F_{12} & \cdots & F_{1t} \\ F_{21} & F_{22} & \cdots & F_{2t} \\ \vdots & \vdots & & \vdots \\ F_{t1} & F_{t2} & \cdots & F_{tt} \end{pmatrix},$$

where P_{ii} and F_{ii} are square, of order m_i, with $n = \sum_{i=1}^t m_i$. The stationary distribution π^T is partitioned conformally with P, $\pi^T = (\pi_1^T, \dots, \pi_t^T)$. From now on, π_i^T is a row vector of order m_i and $\pi_i(j)$ is the jth entry of π_i^T. Similarly, $\tilde{\pi}^T$ can be partitioned as $\tilde{\pi}^T = (\tilde{\pi}_1^T, \dots, \tilde{\pi}_t^T)$. Under the condition that $\|F_{ii}\| \leq \eta$ for $i = 1, \dots, t$ and

$$\mathbf{1}^T |F_{ij}| \leq \eta \cdot \mathbf{1}^T P_{ij}, \quad i \neq j, \tag{1.3}$$

where the inequality and the absolute value are to be understood in a entrywise sense, we are to bound

$$\max_{1 \leq i \leq t,\ 1 \leq j \leq m_i} \left| \frac{\pi_i(j) - \widetilde{\pi}_i(j)}{\pi_i(j)} \right|. \tag{1.4}$$

In other words, we are to bound the relative error in each stationary probability when each column of the off-diagonal blocks of the transition matrix gets a small relative perturbation. We will find a bound that depends on some quantities related to the stochastic complements of diagonal blocks. The stochastic complement of P_{ii} [7] is

$$S_{ii} = P_{ii} + P_{i*}(I - P_i)^{-1} P_{*i},$$

where P_{i*} is an $m_i \times (n - m_i)$ matrix composed of the ith row of blocks of P with P_{ii} removed, P_{*i} is an $(n - m_i) \times m_i$ matrix composed of the ith column of blocks of P with P_{ii} removed, and P_i is the principal submatrix of P with the ith row and column of blocks removed.

Obviously, S_{ii} is also a stochastic matrix with stationary distribution $\pi_i^T / \|\pi_i^T\|$.

In section 3, we will discuss how to bound these quantities. For nearly uncoupled and nearly transient Markov chains, the stationary probabilities are sensitive to general perturbations in a relative sense. However, they are insensitive to small relative blockwise perturbations as in (1.3). This extends results in [1], [10], and [11].

2. Symbols and definitions.. Throughout this article $|B|$ denotes the matrix $|B| = (|b_{ij}|)$ and $B \leq C (< C)$ means $b_{ij} \leq c_{ij} (< c_{ij})$ for all i and j. We denote by $\bar{B}$, $\breve{B}$, and $\hat{B}$ the submatrices of B with the last column, the last row, and the last column and row removed, respectively. These symbols are also used for row vectors and columns. The jth column of the identity matrix I is denoted by e_j.

Let G be a tree of vertices $1, \dots, n$; i.e., G has $n-1$ edges and no cycles. A vertex of G that has one neighbor is called a leaf. If B is an $n \times n$ matrix, then we define

$$d_G(B) = \min\{\max(|a_{ij}|, |a_{ji}|) \ |\{i,j\} \text{ is an edge of } G\},$$

$$d(B) = \max_G d_G(B),$$

and

$$q(B) = \max\{\|C^{-1}\| \, C \text{ is a principal matrix of } B \text{ with order } n\}.$$

Writing $A_{ii} = I - P_{ii}$ and $B_{ii} = I - S_{ii}$, where S_{ii} is as in (1.5), we have

$$\pi^T B_{ii} = 0 \quad \text{and} \quad d(B_{ii}) = d(S_{ii}) \geq d(P_{ii}).$$

We set

$$q = \max_{1 \leq i \leq t} q(B_{ii}) \quad \text{and} \quad s = \max_{1 \leq i \leq t} s_i,$$

where

$$s_i = \frac{\max_{1 \leq j \leq m_i} \pi_i(j)}{\min_{1 \leq j \leq m_i} \pi_i(j)}.$$

If $m_i = 1$, we let $s = 1$ and $q = 0$.

3. Bounding q.

After computing π^T, we can get s at once. Now we discuss how to bound q.

LEMMA 3.1. *Let A be an $n \times n$ irreducible M-matrix. If $d(A) \geq c$ and A is column and row diagonally dominant, i.e., $A\mathbf{1} \geq 0$ and $\mathbf{1}^T A \geq 0$, then*

$$\|A_i^{-1}\| \leq \frac{n(n-1)}{2c}, \tag{3.1}$$

where A_i is the principal submatrix of A with the ith column and row deleted.

Proof. Without loss of generality, we prove (3.1) for $i = n$. Suppose that G is a tree and $d_G(A) = d(A)$. It is well known that a tree has at least two leaves and the subgraph of a tree with one leaf deleted is still a tree. So there exists a permutation of $1, \ldots, n-1$, which is denoted by $\alpha(1), \ldots, \alpha(n-1)$, such that $\alpha(j)$ is a leaf of the subgraph of G with the vertices $\alpha(1), \ldots, \alpha(j-1)$ deleted. In other words, for $1 \leq j \leq n-1$, we can find $j < k \leq n$ such that

$$a_{\alpha(k),\alpha j} \leq -c \quad \textit{or} \quad a_{\alpha(j),\alpha k} \leq -c; \tag{3.2}$$

here we set $\alpha(n) = n$. Without loss of generality, we assume that $\alpha(j) = j$. Thus after $n-2$ steps of Gaussian elimination on A, we have the LU factorization of A_n

$$L A_n U = D,$$

where $L = L_{n-2} \cdots L_1$ and $U = U_1 \cdots U_{n-1}$ with

$$L_j = \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & l_{j+1,j} & \ddots & \\ & & \vdots & & \\ & & l_{n-1,j} & & 1 \end{pmatrix}, \quad U_j = \begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & u_{j,j+1} & \cdots & u_{j,n-1} \\ & & & \ddots & & \\ & & & & \ddots & \\ & & & & & 1 \end{pmatrix},$$

and $D = \mathrm{diag}(d_1, \dots, d_{n-1})$. Because Gaussian elimination preserves the M-matrix structure and diagonal dominance, see [2], we have that $L_i \geq 0$, $U_i \geq 0$, and

$$\sum_{k=j+1}^{n-1} l_{k,j} \leq 1, \qquad \sum_{k=j+1}^{n-1} u_{j,k} \leq 1.$$

This gives

$$U_j \sum_{k=j+1}^{n-1} e_k \leq \sum_{k=j}^{n-1} e_k \quad \text{and} \quad \left(\sum_{k=j+1}^{n-1} e_k^T \right) L_j \leq \sum_{k=j}^{n-1} e_k^T. \tag{3.3}$$

From (3.3) and $U_j e_k = e_k$ for $k \leq j$, we get

$$U e_j \leq U_1 \cdots U_{j-1} e_j \leq U_1 \cdots U_{j-1} \sum_{k=j}^{n-1} e_k \leq \sum_{j=1}^{n-1} e_j.$$

Since U is upper triangular,

$$U \leq \begin{pmatrix} 1 & 1 & \cdots & 1 \\ & 1 & \ddots & \vdots \\ & & \ddots & \vdots \\ & & & 1 \end{pmatrix}.$$

Similarly,

$$L \leq \begin{pmatrix} 1 & & & \\ & 1 & 1 & \\ \vdots & \vdots & \ddots & \\ 1 & 1 & \cdots & 1 \end{pmatrix}.$$

Since the off-diagonal entries of A decrease under Gaussian elimination,

$$d_j \geq \max \left\{ \sum_{k=j+1}^{n} |a_{jk}|, \sum_{k=j+1}^{n} |a_{kj}| \right\} \geq c.$$

Thus

$$\|A_n^{-1}\| = \|UD^{-1}L\| \leq \frac{1}{c}\|UL\| \leq \frac{n(n-1)}{2c}. \qquad \square$$

Remark. Let

$$A = \begin{pmatrix} c & -c & & & \\ -c & 2c & -c & & \\ & \ddots & \ddots & \ddots & \\ & & -c & 2c & -c \\ & & & -c & c \end{pmatrix}.$$

We can easily get $A\mathbf{1} = 0$, $\mathbf{1}^T A = 0$, $d(A) = c$, and

$$\|A_n^{-1}\| = \frac{n(n-1)}{2c}.$$

This means that the bound (3.1) is tight.

The following theorem provides an upper bound for q.

THEOREM 3.1. *Let $m = \max_i m_i$ and $d = \min_i d(P_{ii})$. Then*

$$q \le \frac{m(m-1)s}{2d}.$$

Proof. Let $D_i = \operatorname{diag}(\pi_i(1), \ldots, \pi_i(m_i))$ and $C_{ii} = D_i B_{ii}$. We have that $\|D_i\| \le \max_{1 \le j \le m_i} \pi_i(j)$ C_{ii} is an M-matrix with $C_{ii}\mathbf{1} = 0$ and $\mathbf{1}^T C_{ii} = 0$ and

$$d(C_{ii}) \ge d(B_{ii}) \cdot \min_{1 \le j \le m_i} \pi_i(j) \ge d(P_{ii}) \cdot \min_{1 \le j \le m_i} \pi_i(j) \ge d \min_{1 \le j \le m_i} \pi_i(j).$$

From Lemma 3.1,

$$q(C_{ii}) \le \frac{m(m-1)}{2d \cdot \min_{1 \le j \le m_i} \pi_i(j)}.$$

Thus,

$$q(B_{ii}) \le \|D_i\| q(C_{ii}) \le \frac{m(m-1)s}{2d}. \qquad \square$$

4. The main result. In this section, we assess the relative sensitivity of the individual stationary probabilities. The following theorem is the main result of this paper.

THEOREM 4.1. *Let P and $\widetilde{P} = P + F$ be irreducible transition matrices with respective stationary distributions π^T and $\widetilde{\pi}^T$. Let P and F be partitioned as in* (1.3) *and π^T and $\widetilde{\pi}^T$ be partitioned conformally. Denote*

$$\delta = \max_{1 \le i \le t,\ 1 \le j \le m_i} \left| \frac{\pi_i(j) - \widetilde{\pi}_i(j)}{\pi_i(j)} \right|.$$

If for $i = 1, \ldots, t$ P_{ii} is irreducible and

$$\|F_{ii}\| \le \eta \quad \text{and} \quad \mathbf{1}^T |F_{ij}| \le \eta \cdot \mathbf{1}^T P_{ij} \ \text{for } i \ne j,$$

then

$$\delta \le \frac{2ts((s+1)q+1) \cdot \eta}{1 - 2ts((s+1)q+1) \cdot \eta}. \tag{4.1}$$

Proof. Without loss of generality, we assume that

$$\delta = \max_{1 \le i \le t,\ 1 \le j \le m_i} \left| \frac{\widetilde{\pi}_i(j) - \pi_i(j)}{\pi_i(j)} \right| = \left| \frac{\widetilde{\pi}_t(m_t) - \pi_t(m_t)}{\pi_t(m_t)} \right|.$$

From (1.1)

$$\begin{aligned}\delta = |\widetilde{\pi}^T \bar{F} \widehat{A}^{-1}\mathbf{1}| &\le \sum_{j=1}^{t} |\widetilde{\pi}_j^T (F_{j1}; \ldots ; F_{j,t-1}\ \bar{F}_{jt})\widehat{A}^{-1}\mathbf{1}| \\ &\le (1+\delta) \sum_{j=1}^{t} |\pi_j^T (F_{j1}; \ldots ; F_{j,t-1}\ \bar{F}_{jt})\widehat{A}^{-1}\mathbf{1}|. \qquad (4.2)\end{aligned}$$

Construct

$$X_j = \begin{pmatrix} I_{m_1} & & & & & & -\mathbf{1}_{m_1} & \\ & \ddots & & & & & \vdots & \\ & & I_{m_{j-1}} & & & & -\mathbf{1}_{m_{j-1}} & \\ & & & 0 & & & -\mathbf{1}_{m_j} & \bar{T}_{m_j} \\ & & & I_{m_{j+1}} & & & -\mathbf{1}_{m_{j+1}} & \\ & & & & \ddots & & \vdots & \\ & & & & & \breve{I}_{m_t} & -\mathbf{1}_{m_t-1} & \end{pmatrix}$$

and

$$Y_j = \begin{pmatrix} I_{m_1} & & & & & & \\ & \ddots & & & & & \\ & & I_{m_{j-1}} & & & & \\ & & & 0 & I_{m_{j+1}} & & \\ & & & & & \ddots & \\ -\frac{1}{\pi_t(m_t)}\pi_1^T & \cdots & \cdots & -\frac{1}{\pi_t(m_t)}\pi_j^T & \cdots & \cdots & -\frac{1}{\pi_t(m_t)}\bar{\pi}_t^T \\ & & & \breve{I}_{m_j} & & & \end{pmatrix}.$$

It is easily verified that

$$(F_{j1}; \ldots ; F_{j,t-1}; \bar{F}_{jt})X_j = (F_{j1}; \ldots ; F_{j,j-1}; F_{j,j+1}; \ldots ; F_{jt}; \bar{F}_{jj}),$$

$$Y_j\mathbf{1} = \mathbf{1} - \frac{1}{\pi_t(m_t)} e_{r_j}, \quad \text{where} \quad r_j = \sum_{i\neq j}^{t} m_i,$$

and

$$Y_j \widehat{A} X_j = H_j; \quad \text{i.e.,} \quad \widehat{A}^{-1} = X_j H^{-1} Y_j,$$

where

$$H_j = \begin{pmatrix} I - P_j & -\bar{P}_{*j} \\ -\breve{P}_{j*} & \widehat{A}_{jj} \end{pmatrix}.$$

Thus

$$\begin{aligned}\delta \le (1+\delta)\Bigg(& |\pi_t^T (F_{t1}; \ldots ; F_{t,t-1}\ \bar{F}_{tt})\widehat{A}^{-1}\mathbf{1}| \\ & + \sum_{j=1}^{t-1} \left| \pi_j^T (F_{j1}; \ldots ; F_{j,j-1}; F_{j,j+1}; \ldots ; F_{jt}; \bar{F}_{jj}) H_j^{-1} \left(\mathbf{1} + \frac{1}{\pi_t(m_t)} e_{r_j} \right) \right| \Bigg). \qquad (4.3)\end{aligned}$$

Now we bound

$$|\pi_t^T(F_{t1};\dots;F_{t,t-1};\bar{F}_{tt})\widehat{A}^{-1}\mathbf{1}|.$$

From $\mathbf{1}^T|F_{tj}| \le \eta\mathbf{1}^T P_{tj}$ for $j = 1,\dots,t-1$, it follows that

$$|\pi_t^T(F_{t1};\dots;F_{t,t-1})| \le s_t\eta\cdot\pi_t^T P_{t*} \le s\eta\cdot\pi_t^T P_{t*}.$$

Since $\pi^T A = 0$ implies

$$\pi_t^T(-P_{t*}\,;\,\bar{A}_{tt}) = -(\pi_1^T;\dots;\pi_{t-1}^T;0)\widehat{A},$$

i.e.,

$$\pi_t^T(-P_{t*}\,;\,\bar{A}_{tt})\widehat{A}^{-1} = -(\pi_1;\dots;\pi_{t-1};0)$$

and

$$\bar{\pi}_t^T \ge \pi_t^T\bar{A}_{tt} = \sum_{i-1}^{t-1}\pi_t^T\bar{P}_{it} \ge 0,$$

we have

$$\begin{aligned}|\pi_t^T(F_{t1};\dots;F_{t,t-1};\bar{F}_{tt})\widehat{A}^{-1}| &\le s\eta\cdot\pi_t^T(P_{t*};0)\widehat{A}^{-1} + \pi_t^T(0\,;|\bar{F}_{tt}|)\widehat{A}^{-1}\\ &= s\eta\cdot\pi_t^T(P_{t*}\;;-\bar{A}_{tt})\widehat{A}^{-1} + \pi_t^T(0\;;\,C)\widehat{A}^{-1}\\ &= s\eta\cdot(\pi_1^T;\dots;\pi_{t-1}^T;0) + \pi_t^T(0;C)\widehat{A}^{-1}.\end{aligned}$$

Here $C = s\eta\cdot\bar{I} + |\bar{F}_{tt}|$ with $\|C\| \le (s+1)\eta$. Writing

$$\widehat{A} = \begin{pmatrix} I-P_t & -\bar{P}_{*t}\\ -\breve{P}_{t*} & \widehat{A}_{tt}\end{pmatrix}$$

and noting $\pi^T A = 0$ implies $\bar{\pi}_t^T\breve{P}_{t*}(I-P_t)^{-1} \le (\pi_1^T;\dots;\pi_{t-1}^T)$, we get

$$\widehat{B}_{tt} = \widehat{A}_{tt} - \breve{P}_{t*}(I-P_t)^{-1}\bar{P}_{*t}$$

and

$$\begin{aligned}\pi_t^T(0\;;\;C)\widehat{A}^{-1} &= \pi_t^T C\widehat{B}_{tt}^{-1}(\breve{P}_{t*}(I-P_t)^{-1};\;I)\\ &\le s(s+1)q(B_{tt})\|C\|\eta\cdot\bar{\pi}_t^T(\breve{P}_{t*}(I-P_t)^{-1};\;I)\\ &\le s(s+1)q\eta\cdot(\pi_1^T;\cdots;\pi_{t-1}^T;\bar{\pi}_t^T).\end{aligned}$$

Thus

$$|\pi_t^T(F_{t1};\dots;F_{t,t-1};\bar{F}_{tt})\widehat{A}^{-1}| \le s((s+1)q+1)\eta\cdot(\pi_1^T;\cdots;\pi_{t-1}^T;\bar{\pi}_t^T). \tag{4.4}$$

Similarly, we have

$$|\pi_j^T(F_{j1};\dots;F_{j,j-1};F_{j,j+1};\dots;F_{jt};\bar{F}_{jj})H_j^{-1}|,$$

$$s((s+1)q+1)\eta\cdot(\pi_1^T;\dots;\pi_{j-1}^T;\pi_{j+1}^T;\dots;\pi_t;\bar{\pi}_j^T). \tag{4.5}$$

Substituting (4.4) and (4.5) into (4.3), we have

$$\delta \leq 2(1+\delta)st((s+1)q+t),$$

which gives (3.1). □

Remark. If $m_i = 1$ for $1 \leq i \leq t$, then $\delta \leq 2n\eta + O(\eta^2)$; this is just O'Cinneide's result. If s and q are not large, this theorem demonstrates that each stationary probability is insensitive to small relative blockwise perturbations. For some ill-conditioned Markov chains, such as nearly uncoupled and nearly transient Markov chains, $\max_i \|A_i^{-1}\|$ must be large, while s and q may be small. Consider the following example:

$$P = \begin{pmatrix} 0.5 & 0.5-\epsilon & \epsilon & 0 \\ 0.5-\epsilon & 0.5 & 0 & \epsilon \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{pmatrix} = \begin{pmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{pmatrix}$$

and

$$F = \begin{pmatrix} \eta\epsilon & 0 & 0 & -\eta\epsilon \\ 0 & \eta\epsilon & -\eta\epsilon & 0 \\ \frac{\eta}{4} & \frac{\eta}{4} & -\frac{\eta}{4} & -\frac{\eta}{4} \\ \frac{\eta}{4} & \frac{\eta}{4} & -\frac{\eta}{4} & -\frac{\eta}{4} \end{pmatrix} = \begin{pmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{pmatrix},$$

where P_{11}, P_{22}, F_{11}, and F_{22} are 2×2 matrices and ϵ is a small constant. Then

$$\pi^T = \frac{1}{2+4\epsilon}(1, 1, 2\epsilon, 2\epsilon),$$

and

$$\tilde{\pi}^T = \frac{1}{2+2\eta+4(1-\eta)\epsilon}(1+\eta, 1+\eta, 2(1-\eta)\epsilon, 2(1-\eta)\epsilon).$$

It is easy to show that $\|A_i^{-1}\| = O(\frac{1}{\epsilon}),\ i = 3, 4$ while $s = 1$ and $q \approx 2$. The entrywise relative error between π^T and $\tilde{\pi}^T$ is no more than $3\eta + O(\eta^2)$. However we cannot apply O'Cinneide's result to predict that the entrywise relative error is small, since some zero entries of P are perturbed to be nonzero. Applying Theorem 4.1, we have

$$\max_i \left| \frac{\pi_i - \tilde{\pi}_i}{\pi_i} \right| \leq 20\eta + O(\eta^2),$$

which shows that each stationary probability gets a small perturbation.

Acknowledgments. The authors are grateful to Prof. Jiang Erxiong for his careful reading of this paper and illuminating suggestions. We would like to thank two anonymous referees for their helpful comments.

REFERENCES

[1] J. L. Barlow, *Perturbation results for nearly uncoupled Markov chains with applications to iterative methods*, Numer. Math., 65 (1993), pp. 51–62.

[2] A. Berman and R. Plemmons, *Nonnegative Matrices in the Mathematical Science*, Academic Press, New York, 1979.

[3] R. E. Funderlic and C. D. Meyer, *Sensitivity of the stationary distribution vector for an ergodic Markov chain*, Linear Algebra Appl., 76 (1986), pp. 1–17.

[4] G. H. Golub and C. D. Meyer, *Using the QR factorization and group inversion to compute, differentiate, and estimate the sensitivity of stationary probabilities for Markov chains*, SIAM J. Algebraic Discrete Meth., 7 (1985), pp. 273–281.

[5] C. F. Ipsen and C. D. Meyer, *Uniform stability of Markov chains*, SIAM J. Matrix Anal. Appl., 4 (1994), pp. 1061–1074.

[6] C. D. Meyer, *The condition of a finite Markov chain and perturbation bounds for the limiting probabilities*, SIAM J. Algebraic Discrete Meth., 1 (1980), pp. 273–283.

[7] C. D. Meyer, *Stochastic complementation, uncoupling Markov chains, and the theory of nearly reducible systems,* SIAM Rev., 31 (1989), pp. 240–272.

[8] C. A. O'Cinneide, *Entrywise perturbation theory and error analysis for Markov chains*, Numer. Math., 65 (1993), pp. 190–120.

[9] P. J. Schweitzer, *Perturbation theory and finite Markov chains,* J. Appl. Prob., 5 (1968), pp. 401–413.

[10] G. Stewart and G. Zhang, *On a direct method for the solution of nearly uncoupled Markov chains,* Numer. Math., 59 (1991), pp. 1–11.

[11] G. Stewart, *On the perturbation of Markov chains with nearly transient states,* Numer. Math., 65 (1995), pp. 135–141.

[12] Y. Takahashi, *On the effects of small deviations in the transition matrix of a finite Markov chain,* J. Oper. Res. Soc. Japan, 16 (1973), pp. 104–129.

[13] R. L. Tweedie *Perturbation of countable Markov chains and processes,* Ann. Inst. Statist. Math., 32 (1980), pp. 283–290.

[14] G. Zhang, *On the sensitivity of the solution of nearly uncoupled Markov chains,* SIAM J. Matrix Anal. Appl., 4 (1993), pp. 1112–1123.

SIAM J. MATRIX ANAL. APPL.
Vol. 20, No. 2, pp. 279–294

NEWTON'S METHOD FOR DISCRETE ALGEBRAIC RICCATI EQUATIONS WHEN THE CLOSED-LOOP MATRIX HAS EIGENVALUES ON THE UNIT CIRCLE*

CHUN-HUA GUO†

Abstract. When Newton's method is applied to find the maximal symmetric solution of a discrete algebraic Riccati equation (DARE), convergence can be guaranteed under moderate conditions. In particular, the initial guess does not need to be close to the solution. The convergence is quadratic if the Fréchet derivative is invertible at the solution. When the closed-loop matrix has eigenvalues on the unit circle, the derivative at the solution is not invertible. The convergence of Newton's method is shown to be either quadratic or linear with the common ratio $\frac{1}{2}$, provided that the eigenvalues on the unit circle are all semisimple. The linear convergence appears to be dominant, and the efficiency of the Newton iteration can be improved significantly by applying a double Newton step at the right time.

Key words. discrete algebraic Riccati equations, Newton's method, maximal symmetric solution, convergence rate, matrix pencils

AMS subject classifications. 15A24, 65H10, 93B40

PII. S0895479897322999

1. Introduction.

Algebraic Riccati equations occur in many important applications [18], [20]. In a previous paper [11] we considered Newton's method for continuous algebraic Riccati equations (CAREs). In this paper we consider a DARE of the form

$$-X + A^TXA + Q - (C + B^TXA)^T(R + B^TXB)^{-1}(C + B^TXA) = 0, \tag{1.1}$$

where $A, Q \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^{n\times m}, C \in \mathbb{R}^{m\times n}, R \in \mathbb{R}^{m\times m}$, and $Q^T = Q, R^T = R$. We denote by $\mathcal{R}(X)$ the left-hand side of (1.1). The function $\mathcal{R}(X)$ and its derivatives are much more complicated than their CARE counterparts. Nevertheless, it will be shown that most analytical properties established in [11] for the CARE can be extended to the DARE. The analysis here is more involved, but the line of attack is the same.

Let $\mathcal{S}$ be the set of symmetric matrices in $\mathbb{R}^{n\times n}$. For any matrix norm (not necessarily multiplicative) $\mathcal{S}$ is a Banach space. Let $\mathcal{D} = \{X \in \mathcal{S} \mid R+B^TXB \text{ is invertible}\}$. We have $\mathcal{R} : \mathcal{D} \to \mathcal{S}$. The first Fréchet derivative of $\mathcal{R}$ at a matrix $X \in \mathcal{D}$ is a linear map $\mathcal{R}'_X : \mathcal{S} \to \mathcal{S}$ given by

$$\mathcal{R}'_X(S) = -S + \hat{A}^TS\hat{A}, \tag{1.2}$$

where $\hat{A} = A - B(R + B^TXB)^{-1}(C + B^TXA)$. Also the second derivative at $X \in \mathcal{D}$, $\mathcal{R}''_X : \mathcal{S} \times \mathcal{S} \to \mathcal{S}$, is given by

$$\mathcal{R}''_X(S_1, S_2) = -\hat{A}^TS_1HS_2\hat{A} - \hat{A}^TS_2HS_1\hat{A}, \tag{1.3}$$

where $H = B(R + B^TXB)^{-1}B^T$.

For $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times m}$, the pair (A, B) is said to be d-stabilizable if there is a $K \in \mathbb{R}^{m\times n}$ such that $A - BK$ is d-stable, i.e., all its eigenvalues are in the open

*Received by the editors June 16, 1997; accepted for publication (in revised form) by A. Ran January 15, 1998; published electronically October 7, 1998.

http://www.siam.org/journals/simax/20-2/32299.html

†Department of Mathematics and Statistics, University of Calgary, Calgary, AB T2N 1N4, Canada (guo@math.ucalgary.ca).

unit disk. For real symmetric matrices X and Y, we write $X \geq Y (X > Y)$ if $X - Y$ is positive semidefinite (definite). A symmetric solution X_+ of (1.1) is called maximal if $X_+ \geq X$ for every symmetric solution X. The following result is essentially the real version of Theorem 13.1.1 in [18]. See also [22].

THEOREM 1.1. *Let (A, B) be a d-stabilizable pair and assume that there is a symmetric solution $\tilde{X}$ of the inequality $\mathcal{R}(X) \geq 0$ for which $R + B^T\tilde{X}B > 0$. Then there exists a maximal symmetric solution X_+ of $\mathcal{R}(X) = 0$. Moreover, $R + B^T X_+ B > 0$ and all the eigenvalues of $A - B(R + B^T X_+ B)^{-1}(C + B^T X_+ A)$ lie in the closed unit disk.*

Remark 1.1. In Theorem 13.1.1 of [18], the matrix R is required to be invertible. This requirement is needed for some later developments in [18], but is not necessary for the conclusions of Theorem 13.1.1. The proof of that theorem should be slightly modified. We have only to replace expressions of the form $Q - C^T R^{-1} C + (L - R^{-1}C)^T R(L - R^{-1}C)$ by expressions of the form $Q + L^T RL - C^T L - L^T C$. That the invertibility of R is not necessary for the conclusions of Theorem 1.1 has also been noted in [2]. As noted in [3], the matrix R may well be singular in applications.

A symmetric solution X of (1.1) is called stabilizing (resp., almost stabilizing) if all the eigenvalues of $A - B(R + B^T XB)^{-1}(C + B^T XA)$ are in the open (resp., closed) unit disk. Such solutions play important roles in applications. Theorem 1.1 tells us that, under the given conditions, the maximal solution is at least almost stabilizing.

The Newton method for the solution of (1.1) is

$$X_i = X_{i-1} - (\mathcal{R}'_{X_{i-1}})^{-1}\mathcal{R}(X_{i-1}), \quad i = 1, 2, \ldots, \tag{1.4}$$

given that the maps $\mathcal{R}'_{X_i} (i = 0, 1, \ldots)$ are all invertible. The iteration (1.4) is closely related to the solution of the Stein equation described in the following classical result.

THEOREM 1.2 (cf. [18, p. 100]). *For any given matrices $A, B, \Gamma \in \mathbb{R}^{n\times n}$ the Stein equation $S - BSA = \Gamma$ has a unique solution (necessarily real) if and only if $\lambda_r \mu_s \neq 1$ for any $\lambda_r \in \sigma(A), \mu_s \in \sigma(B)$.*

It follows from Theorem 1.2 that, under the conditions of Theorem 1.1, $\mathcal{R}'_{X_+}$ is invertible if and only if $A - B(R + B^T X_+ B)^{-1}(C + B^T X_+ A)$ is d-stable.

When we apply Newton's method to the DARE (1.1) with (A, B) d-stabilizable, the initial matrix X_0 is taken such that $A - B(R + B^T X_0 B)^{-1}(C + B^T X_0 A)$ is d-stable. The usual way to generate such an X_0 is as follows. We choose $L_0 \in \mathbb{R}^{m\times n}$ such that $A_0 = A - BL_0$ is d-stable, and take X_0 to be the unique solution of the Stein equation

$$X_0 - A_0^T X_0 A_0 = Q + L_0^T RL_0 - C^T L_0 - L_0^T C. \tag{1.5}$$

In view of (1.2), the Newton iteration (1.4) can be rewritten as

$$X_i - A_i^T X_i A_i = Q + L_i^T RL_i - C^T L_i - L_i^T C, \quad i = 1, 2, \ldots, \tag{1.6}$$

where

$$L_i = (R + B^T X_{i-1} B)^{-1}(C + B^T X_{i-1} A) \tag{1.7}$$

and

$$A_i = A - BL_i. \tag{1.8}$$

THEOREM 1.3. *Under the same conditions as in Theorem* 1.1 *and for any* $L_0 \in \mathbb{R}^{m\times n}$ *such that* $A_0 = A - BL_0$ *is d-stable, starting with the symmetric matrix* X_0 *determined by* (1.5)*, the recursion* (1.6) *determines a sequence of symmetric matrices* $\{X_i\}_{i=0}^{\infty}$ *for which* $A - B(R + B^T X_i B)^{-1}(C + B^T X_i A)$ *is d-stable for* $i = 0, 1, \ldots,$ $X_0 \geq X_1 \geq \cdots,$ *and* $\lim_{i\to\infty} X_i = X_+$.

The maximal solution can thus be found by the Newton iteration with an initial guess not necessarily close to the solution. The proof of the above theorem can be found in [18, pp. 308–311] (with some slight modifications as pointed out in Remark 1.1). See also [13] and [22]. Note that an L_0 can be produced by automatic stabilizing procedures such as the one in [24]. It should also be noted that $X_0 \geq X_1$ is generally not true, if X_0 is not obtained from (1.5).

It is readily seen that $\mathcal{R}'_X$, as a function of X, is Lipschitz continuous on a closed ball centered at X_+ and contained in $\mathcal{D}$. Thus the well-known locally quadratic convergence of Newton's method (see [15], [21]), in combination with Theorem 1.3, yields the following result.

THEOREM 1.4. *If* $A - B(R + B^T X_+ B)^{-1}(C + B^T X_+ A)$ *is d-stable in Theorem* 1.3*, then for the sequence* $\{X_i\}_{i=0}^{\infty}$ *there is a constant* $c > 0$ *such that, for* $i = 0, 1, \ldots,$ $\|X_{i+1} - X_+\| \leq c\|X_i - X_+\|^2$*, where* $\|\cdot\|$ *is any given matrix norm.*

When the closed-loop matrix $A - B(R + B^T X_+ B)^{-1}(C + B^T X_+ A)$ has eigenvalues on the unit circle, $\mathcal{R}'_{X_+}$ is not invertible. This situation happens in some important applications (see [4], for example). We will show that the convergence of Newton's method is either quadratic or linear with common ratio $\frac{1}{2}$, provided that the eigenvalues on the unit circle are all semisimple (i.e., all elementary divisors corresponding to these eigenvalues are linear). The linear convergence appears to be dominant and, when this is the case, the efficiency of the Newton iteration can be improved significantly by applying a double Newton step at the right time. Numerical results are also given to illustrate these phenomena.

As in [11] we apply the following general formulation of Newton's method (see [5], [6], [7], [16], [17], [23]). Let F be a smooth map from a Banach space E into itself. Assume that there is an $x^* \in E$ such that $F(x^*) = 0$ and the Fréchet derivative at x^*, $F'(x^*)$, has a null space N of dimension d with $0 < d < \infty$. Also, it is assumed that $F'(x^*)$ has closed range M and that there is a *direct sum* decomposition $E = N \oplus M$. Then we may define P_N to be the projection onto N parallel to M and let $P_M = I - P_N$. Assume further that the following *regularity* condition holds: there is a $\phi_0 \in N$ such that the map B on N given by $B = P_N F''(x^*)(\phi_0, \cdot)$ is invertible. These ideas can now be used to formulate sufficient conditions for *local* convergence.

THEOREM 1.5 (cf. [16, Theorem 1.1]). *Let* $E = N \oplus M$*, let* ϕ_0 *be chosen so that* B *is invertible, and let* $N = \operatorname{span}\{\phi_0\} \oplus N_1$ *for some subspace* N_1*. Write* $\tilde{x} = x - x^*$ *and let*

$$W(\rho, \theta, \eta) = \{x \mid 0 < \|\tilde{x}\| < \rho, \|P_M \tilde{x}\| \leq \theta \|P_N \tilde{x}\|, \quad \|(P_N - P_0)\tilde{x}\| \leq \eta \|P_N \tilde{x}\|\}, \tag{1.9}$$

where P_0 *is the projection onto* $\operatorname{span}\{\phi_0\}$ *parallel to* $M \oplus N_1$*. If* $x_0 \in W(\rho_0, \theta_0, \eta_0)$ *for* ρ_0, θ_0, η_0 *sufficiently small, then the Newton sequence* $\{x_i\}$ *is well defined and* $\|F'(x_i)^{-1}\| \leq c\|\tilde{x}_i\|^{-1}$ *for all* $i \geq 1$ *and some constant* $c > 0$*. Moreover,*

$$\lim_{i\to\infty} \frac{\|\tilde{x}_{i+1}\|}{\|\tilde{x}_i\|} = \frac{1}{2}, \quad \lim_{i\to\infty} \frac{\|P_M \tilde{x}_i\|}{\|P_N \tilde{x}_i\|^2} = 0.$$

The regularity condition is very important for the above theorem. Without this condition, the behavior of Newton's method can be very erratic (see, e.g., [10]). Before we can apply Theorem 1.5 to the DARE (1.1), we need to check the direct sum condition and the regularity condition for the DARE. The direct sum condition will be discussed in sections 2 and 3. The regularity condition is satisfied for the DARE whenever the matrix pair (A, B) is d-stabilizable. This will be discussed in section 4.

If the matrix pair (A, B) is not d-stabilizable, a generalized Newton's method may be used for the solution of the DARE (1.1). For differential periodic Riccati equations without the stability condition, the convergence of a generalized Newton's method has been established in [12]. The ideas used in that paper can also be used for CAREs or DAREs without the stabilizability condition. In this paper, however, we restrict ourselves to the standard Newton's method and assume that the matrix pair (A, B) is d-stabilizable.

2. Interpretation of the direct sum condition for the DARE.

We now go back to the discussion of the DARE (1.1) and assume throughout that the conditions of Theorem 1.1 are satisfied. Let X_+ be the maximal solution of (1.1) with $\mathcal{R}'_{X_+}$ not invertible. Let $\mathcal{N} = \mathrm{Ker}\mathcal{R}'_{X_+}$, $\mathcal{M} = \mathrm{Im}\mathcal{R}'_{X_+}$. We have the following interpretation of the direct sum condition.

THEOREM 2.1. *$\mathcal{S} = \mathcal{N} \oplus \mathcal{M}$ if and only if all eigenvalues of*

$$A_+ = A - B(R + B^T X_+ B)^{-1}(C + B^T X_+ A)$$

on the unit circle are semisimple.

Proof. Let J be the real Jordan canonical form for A_+ with $P^{-1}A_+P = J$ and a real matrix P. We find that $K \in \mathcal{N}$ if and only if $K = P^{-T}LP^{-1}$ for some $L \in \mathcal{N}_J = \{Y \in \mathcal{S} \mid -Y + J^TYJ = 0\}$. Also $W \in \mathcal{M}$ if and only if $W = P^{-T}UP^{-1}$ for some $U \in \mathcal{M}_J = \{Y \in \mathcal{S} \mid Y = -V + J^TVJ \text{ for some } V \in \mathcal{S}\}$. Therefore, $\mathcal{S} = \mathcal{N} \oplus \mathcal{M}$ if and only if $\mathcal{S} = \mathcal{N}_J \oplus \mathcal{M}_J$.

If all eigenvalues of A_+ on the unit circle are semisimple, we gather the Jordan blocks of J in several groups:

$$J = \mathrm{diag}(G_1, G_2, G_3, \ldots, G_{p-1}, G_p). \tag{2.1}$$

Here $G_1 = -I_{r_1}, G_2 = I_{r_2}, G_p \in \mathbb{R}^{r_p \times r_p}$ consists of real Jordan blocks associated with eigenvalues in the open unit disk, and for $i = 3, \ldots, p-1$,

$$G_i = \mathrm{diag}\left(\begin{pmatrix} a_i & b_i \\ -b_i & a_i \end{pmatrix}, \ldots, \begin{pmatrix} a_i & b_i \\ -b_i & a_i \end{pmatrix}\right) \in \mathbb{R}^{r_i \times r_i}, \tag{2.2}$$

where $-1 < a_3 < \cdots < a_{p-1} < 1$, $b_i > 0$, and $a_i^2 + b_i^2 = 1$ for $i = 3, \ldots, p-1$. Using block matrix multiplications and applying Theorem 1.2 repeatedly, we can show that $\mathcal{S} = \mathcal{N}_J \oplus \mathcal{M}_J$. The detailed expressions for $\mathcal{N}_J$ and $\mathcal{M}_J$ will be given in Lemma 2.2 and will be needed in the sequel.

If A_+ has nonlinear elementary divisors corresponding to eigenvalues on the unit circle, we can arrange the Jordan blocks so that the first Jordan block J_1 has one of the following two forms:

$$\text{(i)}\ \ J_1 = \begin{pmatrix} a & 1 & & \\ & a & \ddots & \\ & & \ddots & 1 \\ & & & a \end{pmatrix}, \quad a = \pm 1.$$

$$\text{(ii)}\ \ J_1 = \begin{pmatrix} B & I & & \\ & B & \ddots & \\ & & \ddots & I \\ & & & B \end{pmatrix}, \quad B = \begin{pmatrix} a & b \\ -b & a \end{pmatrix}, \quad b > 0, \quad a^2 + b^2 = 1.$$

For the first case, $D_1 = \operatorname{diag}(0, \dots, 0, 1, 0, \dots, 0) \in \mathcal{N}_J \cap \mathcal{M}_J$, where the element 1 appears at the same position as the last diagonal element of J_1. Note that $D_1 = -V_1 + J^T V_1 J$ for

$$V_1 = \frac{1}{2}\operatorname{sign}(a) \begin{pmatrix} 0 & & & \\ & 0 & 1 & \\ & 1 & 0 & \\ & & & 0 \end{pmatrix},$$

where the 2×2 matrix in the center appears at the same position as the southeast corner of J_1. For the second case, $D_2 = \operatorname{diag}(0, \dots, 0, I, 0, \dots, 0) \in \mathcal{N}_J \cap \mathcal{M}_J$, where the 2×2 identity matrix I appears at the same position as the last diagonal block of J_1. Note that $D_2 = -V_2 + J^T V_2 J$ for

$$V_2 = \frac{1}{2b} \begin{pmatrix} 0 & & & \\ & 0 & T & \\ & -T & 0 & \\ & & & 0 \end{pmatrix} \text{with } T = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix},$$

where the 4×4 matrix in the center appears at the same position as the southeast corner of J_1. Therefore, $\mathcal{S} \neq \mathcal{N}_J \oplus \mathcal{M}_J$. □

In order to give an explicit construction of the spaces $\mathcal{N}_J$ and $\mathcal{M}_J$, we introduce, as in [11], the matrices

$$E_1 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad E_2 = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}, \quad E_3 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \quad E_4 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix},$$

and let $\mathcal{S}^k$ be the linear space of real symmetric matrices of order k. For $3 \leq j \leq p-1$, we define subspaces $\mathcal{S}_j, \mathcal{T}_j \subset \mathcal{S}^{r_j}$ by

$$\mathcal{S}_j = \left\{ X \otimes E_1 + Y \otimes E_2 \mid X \text{ symmetric, } Y \text{ antisymmetric; both have order } \frac{r_j}{2} \right\};$$

$$\mathcal{T}_j = \left\{ X \otimes E_3 + Y \otimes E_4 \mid X, Y \text{ symmetric of order } \frac{r_j}{2} \right\}.$$

Here, $\otimes$ denotes the Kronecker product (see p. 97 of [18], for example).

LEMMA 2.2. *If all eigenvalues of A_+ on the unit circle are semisimple, then*

$$\mathcal{N} = \{P^{-T} N P^{-1} \mid N \in \mathcal{N}_J\}, \quad \mathcal{M} = \{P^{-T} M P^{-1} \mid M \in \mathcal{M}_J\}$$

with

$$\begin{aligned} \mathcal{N}_J = \{N = \operatorname{diag}(N_1, \dots, N_p) \mid N_i \in \mathbb{R}^{r_i \times r_i}, 1 \leq i \leq p;\\ N_1^T = N_1, N_2^T = N_2, N_p = 0, N_i \in \mathcal{S}_i, 3 \leq i \leq p-1\},\\ \mathcal{M}_J = \{M = (M_{ij}) \mid M_{ij} \in \mathbb{R}^{r_i \times r_j}, M_{ij}^T = M_{ji}, 1 \leq i, j \leq p;\\ M_{11} = 0, M_{22} = 0, M_{ii} \in \mathcal{T}_i, 3 \leq i \leq p-1\}. \end{aligned}$$

Proof. The statement can be verified using block matrix multiplications and Theorem 1.2. □

3. Characterization of the direct sum condition via a matrix pencil. We have given in section 2 a characterization of the direct sum condition, in which the sought after solution X_+ appears. In order to give a characterization which is independent of X_+, we consider the matrix pencil $\lambda F_e - G_e$ with

$$F_e = \begin{pmatrix} I & 0 & 0 \\ 0 & A^T & 0 \\ 0 & -B^T & 0 \end{pmatrix}, \qquad G_e = \begin{pmatrix} A & 0 & B \\ -Q & I & -C^T \\ C & 0 & R \end{pmatrix}.$$

Matrix pencils of this type were first introduced in [8] and [25], but for a different purpose. See also [14].

Lemma 3.1. *If* (1.1) *has a Hermitian solution X, then*

$$(\lambda F_e - G_e)\begin{pmatrix} I & 0 & 0 \\ X & I & 0 \\ Z & 0 & I \end{pmatrix} = \begin{pmatrix} I & 0 & 0 \\ A^T X & I & Z^T \\ -B^T X & 0 & I \end{pmatrix}(\lambda M_e - N_e), \tag{3.1}$$

where $Z = -(R + B^T XB)^{-1}(C + B^T XA)$ and

$$M_e = \begin{pmatrix} I & 0 & 0 \\ 0 & (A+BZ)^T & 0 \\ 0 & -B^T & 0 \end{pmatrix}, \qquad N_e = \begin{pmatrix} A+BZ & 0 & B \\ 0 & I & 0 \\ 0 & 0 & R + B^T XB \end{pmatrix}.$$

Proof. It can be easily verified by direct computation. □

Note that, in contrast with Proposition 15.2.1 of [18], the equality (3.1) does not require the invertibility of R.

Corollary 3.2. *If* (1.1) *has a Hermitian solution X, then $\lambda F_e - G_e$ is a regular pencil. Moreover, $\alpha \neq 0$ is an eigenvalue of $A + BZ$ if and only if α and $\bar{\alpha}^{-1}$ are eigenvalues of $\lambda F_e - G_e$. A unimodular α is an eigenvalue of $A + BZ$ with algebraic multiplicity k if and only if it is an eigenvalue of $\lambda F_e - G_e$ with algebraic multiplicity $2k$.*

Proof. We have by Lemma 3.1 $\det(\lambda F_e - G_e) = (-1)^m \det(R + B^T XB)\det(\lambda I - (A + BZ))\det(\lambda(A + BZ)^T - I)$. If $\det(\lambda I - (A + BZ)) = (\lambda - \lambda_1)\cdots(\lambda - \lambda_n)$, we have $\det(\lambda(A+BZ)^T - I) = (\bar{\lambda}_1\lambda - 1)\cdots(\bar{\lambda}_n\lambda - 1)$. The conclusions in the corollary now follow easily. □

If all unimodular eigenvalues of $\lambda F_e - G_e$ are of algebraic multiplicity two, then all unimodular eigenvalues of $A+BZ$ are simple and the direct sum condition is satisfied. To give a complete characterization, we need to consider the relationship between the elementary divisors of $A + BZ$ and $\lambda F_e - G_e$.

Theorem 3.3. *Let α be a complex number with $|\alpha| = 1$ and X be a solution of* (1.1) *with $R + B^T XB > 0$. If*

$$\operatorname{rank}(\alpha I - A \;\; B) = n,$$

then the elementary divisors of $A+BZ$ corresponding to α have degrees $k_1, \ldots, k_s (1 \leq k_1 \leq \cdots \leq k_s \leq n)$ if and only if the elementary divisors of $\lambda F_e - G_e$ corresponding to α have degrees $2k_1, \ldots, 2k_s$.

Proof. Suppose the elementary divisors of $A + BZ$ corresponding to α have degrees $k_1, \ldots, k_s$. By the local Smith form (see [9], for example), we can find matrix polynomials $E_\alpha(\lambda)$ and $F_\alpha(\lambda)$ invertible at α such that

$$\lambda I - (A + BZ) = E_\alpha(\lambda) \begin{pmatrix} I & 0 \\ 0 & D \end{pmatrix} F_\alpha(\lambda), \tag{3.2}$$

where $D = \operatorname{diag}((\lambda - \alpha)^{k_1}, \ldots, (\lambda - \alpha)^{k_s})$. Replacing λ by $\bar{\lambda}^{-1}$ in (3.2), and then taking conjugate transpose (denoted by $*$), we get

$$(A + BZ)^T - \lambda^{-1} I = K_\alpha(\lambda) \begin{pmatrix} I & 0 \\ 0 & D \end{pmatrix} L_\alpha(\lambda), \tag{3.3}$$

where $K_\alpha(\lambda)$ and $L_\alpha(\lambda) = (E_\alpha(\bar{\lambda}^{-1}))^*$ are rational matrix functions invertible at α. For any rational matrix functions $F(\lambda)$ and $G(\lambda)$, we will write $F(\lambda) \sim G(\lambda)$ if there are rational matrix functions $K(\lambda)$ and $L(\lambda)$ invertible at α such that $F(\lambda) = K(\lambda)G(\lambda)L(\lambda)$.

Now, in view of Lemma 3.1, we have

$$\lambda F_e - G_e \sim \begin{pmatrix} \lambda I - (A + BZ) & 0 & -B \\ 0 & (A + BZ)^T - \lambda^{-1} I & 0 \\ 0 & -B^T & -(R + B^T X B) \end{pmatrix}.$$

By (3.2) and (3.3) we further have (for λ in a neighborhood of α)

$$\lambda F_e - G_e \sim \begin{pmatrix} I & 0 & 0 & 0 & B_{11} & B_{12} \\ 0 & D & 0 & 0 & B_{21} & B_{22} \\ 0 & 0 & I & 0 & 0 & 0 \\ 0 & 0 & 0 & D & 0 & 0 \\ 0 & 0 & C_{11} & C_{12} & S_{11} & S_{12} \\ 0 & 0 & C_{21} & C_{22} & S_{21} & S_{22} \end{pmatrix},$$

where we have written

$$-(E_\alpha(\lambda))^{-1} B = (B_{ij}), \quad -((E_\alpha(\bar{\lambda}^{-1}))^{-1} B)^* = (C_{ij}), \quad -(R + B^T X B) = (S_{ij}).$$

Since $\operatorname{rank}(\alpha I - A \;\; B) = n$, $\operatorname{rank}(\lambda I - (A + BZ) \;\; -B) = n$ at $\lambda = \alpha$. Therefore, at $\lambda = \alpha$,

$$\operatorname{rank} \begin{pmatrix} I & 0 & B_{11} & B_{12} \\ 0 & D & B_{21} & B_{22} \end{pmatrix} = n$$

and thus $\operatorname{rank}(B_{21} \;\; B_{22}) = s$. Note also that $E_\alpha(\bar{\lambda}^{-1}) = E_\alpha(\lambda)$ at $\lambda = \alpha$. We may then assume that B_{21} and C_{12} are invertible in a neighborhood of α. Now we obtain by block elimination

$$\lambda F_e - G_e \sim W(\lambda) = \begin{pmatrix} I & 0 & 0 & 0 & 0 & 0 \\ 0 & D & 0 & 0 & I & 0 \\ 0 & 0 & I & 0 & 0 & 0 \\ 0 & 0 & 0 & D & 0 & 0 \\ 0 & 0 & 0 & I & V_{11} & V_{12} \\ 0 & 0 & 0 & 0 & V_{21} & V_{22} \end{pmatrix},$$

where

$$V(\lambda) = (V_{ij}) = \begin{pmatrix} C_{12}^{-1} & 0 \\ -C_{22}C_{12}^{-1} & I \end{pmatrix} (S_{ij}) \begin{pmatrix} B_{21}^{-1} & -B_{21}^{-1}B_{22} \\ 0 & I \end{pmatrix}$$

is a rational matrix function with $-V(\alpha) > 0$ (we have used $R + B^TXB > 0$ here). It is clear that no principal minors of $V(\lambda)$ are zero at α.

All nonzero minors of order i for $W(\lambda)$ have the form $(\lambda - \alpha)^l q(\lambda)$, where $l \geq 0$ and α is neither a zero nor a pole of the rational function $q(\lambda)$. For $2n+m-s+1 \leq i \leq 2n+m$, the smallest l turns out to be $l_i = \Sigma_{j=1}^{s+i-2n-m} 2k_j$. For $1 \leq i \leq 2n+m-s$, the smallest l is $l_i = 0$. By the Binet–Cauchy formula (see [19], for example), we can see that $(\lambda - \alpha)^{l_i}$ is also the greatest common divisor (of the form $(\lambda - \alpha)^l$) of all minors of order i for $\lambda F_e - G_e$. Thus the elementary divisors of $\lambda F_e - G_e$ corresponding to α are $(\lambda - \alpha)^{2k_1}, \ldots, (\lambda - \alpha)^{2k_s}$. This proves the "only if" part of the theorem. The "if" part follows readily from the "only if" part. $\square$

Corollary 3.4. *If the conditions of Theorem* 1.1 *are satisfied and* $\mathcal{R}'_{X_+}$ *is not invertible, then* $\mathcal{S} = \mathcal{N} \oplus \mathcal{M}$ *if and only if all the elementary divisors of* $\lambda F_e - G_e$ *corresponding to the eigenvalues on the unit circle are of degree two.*

A previous result of the same nature as Theorem 3.3 can be found in [26]. That result is applicable to the DARE (1.1) with $C = 0$, $R > 0$, and $Q \geq 0$.

4. Convergence rate of the Newton method. When $\mathcal{S} = \mathcal{N} \oplus \mathcal{M}$, we let $P_{\mathcal{N}}$ denote the projection onto $\mathcal{N}$ parallel to $\mathcal{M}$ and let $P_{\mathcal{M}} = I - P_{\mathcal{N}}$. For the DARE (1.1), we start the Newton iteration with the symmetric matrix X_0 obtained from the Stein equation (1.5). By Theorem 1.3, the Newton sequence is well defined and converges to X_+. The following result shows that there is some possibility of quadratic convergence.

Lemma 4.1. *For any fixed* $\theta > 0$*, let* $Q = \{i \mid \|P_{\mathcal{M}}(X_i - X_+)\| > \theta\|P_{\mathcal{N}}(X_i - X_+)\|\}$. *Then there exist an integer* i_0 *and a constant* $c > 0$ *such that* $\|X_i - X_+\| \leq c\|X_{i-1} - X_+\|^2$ *for all* i *in* Q *for which* $i \geq i_0$.

Proof. Let $\tilde{X}_i = X_i - X_+$, $i = 0, 1, \ldots$, and let $L_+ = (R + B^TX_+B)^{-1}(C + B^TX_+A)$, (thus $A_+ = A - BL_+$). We have (see [18, p. 314])

$$\tilde{X}_i - A_+^T\tilde{X}_iA_+ = (L_+ - L_i)^T(R + B^TX_iB)(L_+ - L_i)$$

and $\|L_+ - L_i\| = O(\|\tilde{X}_{i-1}\|)$. We also have

$$\begin{aligned} L_+ - L_{i+1} &= \{(R + B^TX_+B)^{-1} - (R + B^TX_iB)^{-1}\}(C + B^TX_+A) \\ &\quad -(R + B^TX_iB)^{-1}B^T\tilde{X}_iA \\ &= (R + B^TX_iB)^{-1}B^T\tilde{X}_iBL_+ - (R + B^TX_iB)^{-1}B^T\tilde{X}_iA \\ &= -(R + B^TX_+B)^{-1}B^T\tilde{X}_iA_+ + O(\|\tilde{X}_i\|^2), \end{aligned}$$

where we have written $O(\|\tilde{X}_i\|^2)$ for a term $W(X_i)$ satisfying $\|W(X_i)\| = O(\|\tilde{X}_i\|^2)$. Now, in view of (1.1) and (1.7),

$$\begin{aligned} \mathcal{R}(X_i) &= \mathcal{R}(X_i) - \mathcal{R}(X_+) \\ &= -\tilde{X}_i + A^T\tilde{X}_iA - (C + B^TX_iA)^TL_{i+1} + (C + B^TX_+A)^TL_+ \\ &= -\tilde{X}_i + A_+^T\tilde{X}_iA_+ - A_+^T\tilde{X}_iA_+ + A^T\tilde{X}_iA \\ &\quad -\{(C + B^TX_iA)^T - (C + B^TX_+A)^T\}L_{i+1} \\ &\quad +(C + B^TX_+A)^T(L_+ - L_{i+1}) \end{aligned}$$

$$\begin{aligned}&= O(\|\tilde{X}_{i-1}\|^2) - A_+^T \tilde{X}_i A_+ + A^T \tilde{X}_i A\\ &\quad -A^T \tilde{X}_i B L_+ + O(\|\tilde{X}_i\|^2) - (BL_+)^T \tilde{X}_i A_+\\ &= O(\|\tilde{X}_{i-1}\|^2) + O(\|\tilde{X}_i\|^2).\end{aligned}$$

Thus for i large enough,

$$\|\mathcal{R}(X_i)\| \le c_1 \|\tilde{X}_{i-1}\|^2 + c_2 \|\tilde{X}_i\|^2 \tag{4.1}$$

for some constants c_1 and c_2.

On the other hand, for i in Q and large enough, we have as in [23]

$$\|\mathcal{R}(X_i)\| \ge (c_3(\theta^{-1}+1)^{-1} - c_4\|\tilde{X}_i\|)\|\tilde{X}_i\| \tag{4.2}$$

for some constants c_3 and c_4. Since $X_i \neq X_+$ for any i, we have by (4.1) and (4.2)

$$c_3(\theta^{-1}+1)^{-1} - c_4\|\tilde{X}_i\| \le c_2\|\tilde{X}_i\| + c_1\|\tilde{X}_{i-1}\|^2/\|\tilde{X}_i\|.$$

Therefore, we can find an i_0 such that $\|\tilde{X}_i\| \le c\|\tilde{X}_{i-1}\|^2$ for all i in Q for which $i \ge i_0$. □

COROLLARY 4.2. *Assume that, for given $\theta > 0$, $\|P_{\mathcal{M}}(X_i - X_+)\| > \theta\|P_{\mathcal{N}}(X_i - X_+)\|$ for all i large enough. Then $X_i \to X_+$ quadratically.*

The condition in Corollary 4.2 appears to be not easily satisfied. In fact, quadratic convergence has never been observed in our numerical experiments. We do not know if there are any examples of quadratic convergence in our setting. The next result describes what will happen if the convergence of the Newton iteration is not quadratic.

THEOREM 4.3. *Assume $\mathcal{S} = \mathcal{N} \oplus \mathcal{M}$. If the convergence of the Newton sequence $\{X_i\}$ is not quadratic, then $\|(\mathcal{R}'_{X_i})^{-1}\| \le c\|X_i - X_+\|^{-1}$ for all $i \ge 1$ and some constant $c > 0$. Moreover,*

$$\lim_{i\to\infty} \frac{\|X_{i+1} - X_+\|}{\|X_i - X_+\|} = \frac{1}{2}, \quad \lim_{i\to\infty} \frac{\|P_{\mathcal{M}}(X_i - X_+)\|}{\|P_{\mathcal{N}}(X_i - X_+)\|^2} = 0.$$

The proof of this theorem will be an application of Theorem 1.5. We first establish some preliminary results.

LEMMA 4.4. *Let J and P be as in the proof of Theorem 2.1. Then*

$$\operatorname{rank}(\lambda I - J \;\; P^{-1}B(R + B^T X_+ B)^{-1} B^T P^{-T}) = n$$

for every complex number λ with $|\lambda| \ge 1$.

Proof. In view of Theorem 4.5.6(b) of [18], we need only to show that the pair $(J, P^{-1}B(R + B^T X_+ B)^{-1} B^T P^{-T})$ is d-stabilizable, or equivalently,

$$(A - BL_+, B(R + B^T X_+ B)^{-1} B^T) \text{ is d-stabilizable.} \tag{4.3}$$

Since (A, B) is d-stabilizable and $\operatorname{Im}(B(R + B^T X_+ B)^{-1} B^T) = \operatorname{Im}B$, (4.3) follows from Lemma 4.5.3 of [18]. □

LEMMA 4.5 (see [11, Lemma A.3]). *Let W be a Hermitian positive semidefinite matrix. If the determinant of a principal submatrix of W is zero, then the rows of W containing this submatrix must be linearly dependent.*

We now set out to check the regularity condition needed in Theorem 1.5. For fixed $Z \in \mathcal{N}$, we consider the map $\mathcal{B}_Z : \mathcal{N} \to \mathcal{N}$ defined by

$$\mathcal{B}_Z(Y) = P_{\mathcal{N}} \mathcal{R}''_{X_+}(Z, Y).$$

By Lemma 2.2, we can write $Y = P^{-T}Y_JP^{-1}, Z = P^{-T}Z_JP^{-1}$ with $Y_J, Z_J \in \mathcal{N}_J$. Let $H_+ = B(R+B^TX_+B)^{-1}B^T$. We have by (1.3)

$$\begin{aligned}\mathcal{B}_Z(Y) &= -P_{\mathcal{N}}(A_+^TZH_+YA_+ + A_+^TYH_+ZA_+)\\ &= -P^{-T}P_{\mathcal{N}_J}(J^TZ_JD_+Y_JJ + J^TY_JD_+Z_JJ)P^{-1},\end{aligned}$$

where $D_+ = P^{-1}B(R+B^TX_+B)^{-1}B^TP^{-T}$, and $P_{\mathcal{N}_J}$ is the projection onto $\mathcal{N}_J$ parallel to $\mathcal{M}_J$. Let $Z_J = \mathrm{diag}(Z_1, \ldots, Z_p), Y_J = \mathrm{diag}(Y_1, \ldots, Y_p)$ and $\mathrm{diag}(D_1, \ldots, D_p)$ be the block diagonal of D_+. Let $\mathcal{S}_i = \mathcal{S}^{r_i}$ for $i = 1, 2$. We have further

$$\mathcal{B}_Z(Y) = -P^{-T}\mathrm{diag}(\mathcal{F}_{Z_1}(Y_1), \mathcal{F}_{Z_2}(Y_2), \ldots, \mathcal{F}_{Z_{p-1}}(Y_{p-1}), 0)P^{-1}, \tag{4.4}$$

where we define linear transformations $\mathcal{F}_{Z_i} : \mathcal{S}_i \to \mathcal{S}_i$ by

$$\begin{aligned}&\mathcal{F}_{Z_i}(Y_i) = Z_iD_iY_i + Y_iD_iZ_i, \quad i = 1, 2,\\ &\mathcal{F}_{Z_i}(Y_i) = P_{\mathcal{S}_i}(G_i^T(Z_iD_iY_i + Y_iD_iZ_i)G_i), \quad i = 3, \ldots, p-1\end{aligned}$$

with $P_{\mathcal{S}_i}$ being the projection onto $\mathcal{S}_i$ parallel to $\mathcal{T}_i$. The matrices G_i were defined in (2.1) and (2.2).

For $k = 1, 2, \ldots, p-1$, let

$$\mathcal{U}_k = \{Z_k \in \mathcal{S}_k \mid \mathcal{F}_{Z_k} : \mathcal{S}_k \to \mathcal{S}_k \text{ is not invertible } \}.$$

LEMMA 4.6. *For $k = 1, 2, \ldots, p-1$, the set $\mathcal{U}_k$ has measure zero in $\mathcal{S}_k$.*

Proof.

Case 1. $k = 1, 2$. We prove the result for $k = 1$, since the proof for $k = 2$ is similar. As in [11], we can show that $\mathcal{U}_1$ has measure zero in $\mathcal{S}_1$ unless $\mathrm{det}D_1 = 0$. Note that $D_+ = P^{-1}B(R+B^TX_+B)^{-1}B^TP^{-T}$ is symmetric positive semidefinite. If $\mathrm{det}D_1 = 0$, the first r_1 rows of D_+ would be linearly dependent by Lemma 4.5. Thus $\mathrm{rank}(-I - J \quad D_+) < n$, which contradicts Lemma 4.4.

Case 2. $k = 3, \ldots, p-1$. We will first find a more explicit expression for $\mathcal{F}_{Z_k}(Y_k)$. It is easily seen that

$$G_k = a_kI \otimes E_1 + b_kI \otimes E_2. \tag{4.5}$$

By Lemma 2.2, we can write

$$Y_k = M_s \otimes E_1 + M_a \otimes E_2, \quad Z_k = N_s \otimes E_1 + N_a \otimes E_2, \tag{4.6}$$

where M_s and N_s are symmetric; M_a and N_a are antisymmetric. Let

$$\begin{aligned}&D_k = (D_{ij})_{i,j=1}^{r_k/2} \text{ with } D_{ij} = \begin{pmatrix} d_1^{ij} & d_3^{ij} \\ d_4^{ij} & d_2^{ij} \end{pmatrix},\\ &Q_s = (q_{ij}^s)_{i,j=1}^{r_k/2} \text{ with } q_{ij}^s = \frac{1}{2}(d_1^{ij} + d_2^{ij}),\\ &Q_a = (q_{ij}^a)_{i,j=1}^{r_k/2} \text{ with } q_{ij}^a = \frac{1}{2}(d_3^{ij} - d_4^{ij}).\end{aligned}$$

Then

$$D_k = Q_s \otimes E_1 + Q_a \otimes E_2 + R_s \otimes E_3 + T_s \otimes E_4, \tag{4.7}$$

where Q_s, R_s and T_s are symmetric; Q_a is antisymmetric. Using (4.5)–(4.7) to expand $G_k^T(Z_kD_kY_k + Y_kD_kZ_k)G_k$, we find that each map $\mathcal{F}_{Z_k}$ has the same form as in the CARE case (see [11]). Thus, as in [11], each $\mathcal{U}_k$ has measure zero in $\mathcal{S}_k$ unless $\det(Q_s + iQ_a) = 0$.

To complete the proof, we need to show $\det(Q_s + iQ_a) \neq 0$. By Lemma 4.4 we have $\operatorname{rank}((a_k+b_ki)I - J \quad D_+) = n$. Let $E(i, j(m))$ be the elementary matrix obtained from I by adding m times row j to row i. Let $t_k = r_1 + \cdots + r_{k-1}$ and

$$U = E(t_k + r_k - 1, (t_k + r_k)(-i)) \cdots E(t_k + 3, (t_k + 4)(-i))E(t_k + 1, (t_k + 2)(-i)).$$

Then

$$\operatorname{rank}(U((a_k + b_ki)I - J) \quad UD_+U^*) = n.$$

Since the (t_k+1)th, (t_k+3)th, $\ldots$, (t_k+r_k-1)th rows of the matrix $U((a_k+b_ki)I-J)$ are all zero, the corresponding rows of the Hermitian positive semidefinite matrix UD_+U^* must be linearly independent. By Lemma 4.5, the principal submatrix (of order $r_k/2$) of UD_+U^* contained in these rows must have a nonzero determinant. The principal submatrix is exactly $2(Q_s + iQ_a)$. Therefore, $\det(Q_s + iQ_a) \neq 0$. □

LEMMA 4.7. *If $\mathcal{S} = \mathcal{N} \oplus \mathcal{M}$, then*

$$\mathcal{U} = \{Z \in \mathcal{N} \mid \mathcal{B}_Z : \mathcal{N} \to \mathcal{N} \textit{ is not invertible } \}$$

has measure zero in $\mathcal{N}$. In particular, the regularity condition holds.

Proof. The result follows from (4.4) and Lemma 4.6, as in [11]. □

Proof of Theorem 4.3. Note that the map $\mathcal{R}$ can be extended to a smooth map on $\mathcal{S}$ without changing its values on a closed ball centered at X_+ and contained in $\mathcal{D}$. Now, as in [11], the proof can be completed by applying Theorem 1.3, Theorem 1.5, Corollary 4.2, and Lemma 4.7. □

When all elementary divisors of the closed-loop matrix corresponding to the eigenvalues on the unit circle are linear, we know from Theorem 4.3 that the convergence of the Newton iteration is either quadratic or linear with rate $\frac{1}{2}$. Quadratic convergence, however, has not been observed in numerical experiments when the closed-loop matrix has eigenvalues on the unit circle. The convergence has been observed to be linear with rate $\frac{1}{\sqrt[p]{2}}$, where p is the highest degree of elementary divisors associated with eigenvalues on the unit circle. The next example gives a little theoretical support for the observation. A general theory for the case $p > 1$ would be a topic for future research.

Example 4.1. Consider the DARE (1.1) with $n = 2, m = 1$ and

$$A = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad C = 0, \quad Q = 0, \quad R = 1.$$

Clearly (A, B) is d-stabilizable and $X_+ = 0$ (0 is the unique almost stabilizing solution in this case. See Theorem 13.5.2 of [18], for example). Note that $(\lambda - 1)^2$ is the only elementary divisor of $A_+ = A$. The Newton sequence $\{X_i\}$ is well defined and we write for $i = 0, 1, \ldots,$

$$X_i = \begin{pmatrix} a_i & c_i \\ c_i & b_i \end{pmatrix}.$$

Since $A - B(R + B^TX_iB)^{-1}(C + B^TX_iA)$ is d-stable, we can deduce that $c_i \neq 0$. Since $X_i \geq 0$, we also have $a_i, b_i > 0$.

By (1.6)–(1.8), we find for $i = 0, 1, \dots$

$$a_{i+1} = \frac{2a_i^2 + 3a_ic_i + 2c_i}{(2a_i - c_i + 4)a_i}, \tag{4.8}$$

$$b_{i+1} = \frac{((2 + a_i)a_{i+1} - a_i)c_i}{2(1 + a_i)^2}, \tag{4.9}$$

$$c_{i+1} = \frac{(1 + a_{i+1})c_i}{2(1 + a_i)}. \tag{4.10}$$

Since $X_i \to 0$, we get from (4.10)

$$\lim_{i\to\infty} \frac{c_{i+1}}{c_i} = \frac{1}{2}. \tag{4.11}$$

It follows from (4.8) that

$$\lim_{i\to\infty} \frac{c_i}{a_i} = 0. \tag{4.12}$$

It then follows from (4.9), (4.11), and (4.12) that $\lim_{i\to\infty} b_i/a_i = 0$. If the convergence of the Newton iteration is linear with rate μ, then $\lim_{i\to\infty} a_{i+1}/a_i = \mu$. Now by (4.8) and (4.12),

$$\lim_{i\to\infty} \frac{a_{i+1}}{a_i} = \frac{1}{2}\left(1 + \lim_{i\to\infty} \frac{c_i}{a_i^2}\right). \tag{4.13}$$

If $\lim_{i\to\infty} c_i/a_i^2 = 0$, we would have $\lim_{i\to\infty} a_{i+1}/a_i = 1/2$ by (4.13) and further $\lim_{i\to\infty} c_i/a_i^2 = \infty$ by (4.11), which is a contradiction. Therefore, $\lim_{i\to\infty} c_i/a_i^2 \neq 0$. Thus we get from (4.11) that $\mu = 1/\sqrt{2}$.

The above example can also serve to show that $X_0 \geq X_1$ is generally not true if X_0 is not determined by (1.5). Take

$$X_0 = \begin{pmatrix} \epsilon^\alpha & \epsilon \\ \epsilon & \delta \end{pmatrix}$$

with $\alpha > 1, 0 < \epsilon < 1$, and δ real. It is easily checked that $A - B(R + B^TX_0B)^{-1}(C + B^TX_0A)$ is d-stable. We see from (4.8) that $a_1 \sim 0.5\epsilon^{1-\alpha}$ as $\epsilon \to 0$. Thus $X_0 \geq X_1$ cannot be true for small ϵ. As ϵ and δ go to zero, we have $\|X_0 - X_+\| \to 0$, but $\|X_1 - X_+\| \to \infty$.

5. Using the double Newton step. We have shown that the convergence of Newton's method is either quadratic or linear with rate $\frac{1}{2}$, provided that the unimodular eigenvalues of the closed-loop matrix are all semisimple. Quadratic convergence has not been observed in our numerical experiments. Therefore, we should always be prepared for linear convergence. In this section we will show that the efficiency of the Newton iteration (when it is linearly convergent) can be improved significantly if a double Newton step is used at the right time. However, since the second derivative of the Riccati function is no longer constant, the improvement will not be as dramatic as in the CARE case.

LEMMA 5.1. *In the setting of Theorems* 1.1 *and* 1.3, *assume that* X_k *is close enough to* X_+ *with* $X_k - X_+ \in \mathcal{N}$ *and that* $\|(\mathcal{R}'_{X_k})^{-1}\| \leq c\|X_k - X_+\|^{-1}$ *with* c *independent of* k. *If* $Y_{k+1} = X_k - 2(\mathcal{R}'_{X_k})^{-1}\mathcal{R}(X_k)$, *then* $\|Y_{k+1} - X_+\| \leq c_1\|X_k - X_+\|^2$ *for some constant* c_1 *independent of* k.

Proof. By Taylor's theorem,

$$\mathcal{R}(X_k) = \frac{1}{2}\mathcal{R}''_{X_+}(X_k - X_+, X_k - X_+) + O(\|X_k - X_+\|^3),$$

and then

$$\begin{aligned}\mathcal{R}'_{X_k}(X_k - X_+) &= \mathcal{R}''_{X_+}(X_k - X_+, X_k - X_+) + O(\|X_k - X_+\|^3)\\ &= 2\mathcal{R}(X_k) + O(\|X_k - X_+\|^3).\end{aligned}$$

Thus

$$X_k - X_+ = 2(\mathcal{R}'_{X_k})^{-1}\mathcal{R}(X_k) + O(\|X_k - X_+\|^2). \qquad \square$$

When the direct sum condition is satisfied and the convergence of the Newton sequence $\{X_k\}$ is not quadratic, we have $\|(\mathcal{R}'_{X_k})^{-1}\| \le c\|X_k - X_+\|^{-1}$ for all k (cf. Theorem 4.3). Moreover, the error $X_k - X_+$ will be dominated by its $\mathcal{N}$-component for large k. A much better approximate solution can then be obtained by applying the double Newton step. More precisely, we have the following result.

THEOREM 5.2. *Assume $\mathcal{S} = \mathcal{N} \oplus \mathcal{M}$ and the convergence of the Newton iteration is not quadratic. If for some k, $\|X_k - X_+\|$ is small enough and $\|P_{\mathcal{M}}(X_k - X_+)\| \le \epsilon\|P_{\mathcal{N}}(X_k - X_+)\|$ with ϵ sufficiently small, and $Y_{k+1} = X_k - 2(\mathcal{R}'_{X_k})^{-1}\mathcal{R}(X_k)$, then $\|Y_{k+1} - X_+\| \le c_1\epsilon + c_2\|X_k - X_+\|^2$ for some constants c_1 and c_2 independent of ϵ and k.*

Proof. The result follows from Lemma 5.1 and the argument used in the proof of [11, Theorem 3.2]. $\square$

In contrast to the CARE case, the error estimate for Y_{k+1} contains the term $c_2\|X_k - X_+\|^2$. For a problem which produces a large c_2, the error $\|Y_{k+1} - X_+\|$ will be small only when $\|X_k - X_+\|$ is already sufficiently small. In this case the double Newton step will be useful only at a very late stage of the iteration.

In the CARE case (as described in [11]), the iterate produced by the double Newton step is at least almost stabilizing (see the discussions in [2]). For the DARE case, however, it can happen that the matrix Y_{k+1} in Theorem 5.2 is neither stabilizing nor almost stabilizing.

Example 5.1 (cf. [18, Example 13.2.1]). Consider the DARE (1.1) with $Q = C = 0$ and $A = B = R = I$. Clearly (A, B) is d-stabilizable and $X_+ = 0$. All eigenvalues of the closed-loop matrix are on the unit circle and semisimple. For $L_0 = I$, the Newton iterates are found to be

$$X_k = \frac{1}{2^{k+1} - 1}I, \quad k = 0, 1, \ldots.$$

Thus, the convergence is linear with rate 1/2. If we compute Y_{k+1} as in Theorem 5.2, we get

$$Y_{k+1} = -\frac{1}{(2^{k+1} - 1)(2^{k+2} - 1)}I.$$

Although Y_{k+1} is much more accurate than X_{k+1} for large k, it is neither stabilizing nor almost stabilizing.

The double Newton step is useful in that it can significantly improve the accuracy of the current Newton iterate and thus find more correct digits of the exact solution.

The potential problem of getting a slightly nonstabilizing approximate solution is not our concern here. Even if an exact solution with an infinite number of decimals is known, we will probably get a slightly nonstabilizing approximate solution by keeping only a finite number of decimals.

Theorem 5.2 suggests the following modification of the Newton method.

Algorithm (modified Newton method for DARE).

```
1. Choose a matrix $L_0$ for which $A - BL_0$ is d-stable.
2. Find $X_0$ from (1.5).
3. For $k = 0, 1, \ldots$ do:
     Solve $\mathcal{R}'_{X_k}(H) = \mathcal{R}(X_k)$;
     Compute $X_{k+1} = X_k - 2H$;
     If $\|\mathcal{R}(X_{k+1})\| < \epsilon$, stop;
     Otherwise, compute $X_{k+1} = X_k - H$;
     If $\|\mathcal{R}(X_{k+1})\| < \epsilon$, stop.
```

In the above algorithm, $\|\cdot\|$ is an easily computable matrix norm (e.g., 1-norm) and ϵ is a prescribed accuracy. The equation $\mathcal{R}'_{X_k}(H) = \mathcal{R}(X_k)$ can be rewritten as a Stein equation $H - A_{k+1}^T H A_{k+1} = -\mathcal{R}(X_k)$, which can be solved efficiently by a variation of the Bartels/Stewart algorithm [1] (also see [20]). According to Theorem 5.2, the double Newton step will be efficient only when the current iterate is already reasonably close to the solution. This is a major difference between the CARE and DARE cases. We may try the double Newton step only when the norm of the residual is small enough (less than $\sqrt{\epsilon}$, say) and save a little more computational work. In the above algorithm, all iterates except the last one are identical to those produced by the original Newton method. Thus all good properties of the Newton method are retained.

6. Numerical results.

In this section we give two simple examples to illustrate the performance of the modified Newton method.

Example 6.1. We consider the DARE (1.1) with $n = m = 2$ and

$$A = \begin{pmatrix} 0 & -1 \\ 0 & 2 \end{pmatrix}, \; B = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, \; C = 0, \; Q = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \; R = \begin{pmatrix} 4 & 2 \\ 2 & 1 \end{pmatrix}.$$

Note that A and R are both singular. It can be easily verified that $X_+ = \mathrm{diag}(1, 0)$ is the only solution of the DARE and the closed-loop eigenvalues are 0 and 1. We take $L_0 = \mathrm{diag}(0, 2)$ so that $A_0 = A - BL_0$ is d-stable, and apply the modified Newton method with $\epsilon = 10^{-10}$. The numerical results are recorded in Table 6.1. The last iterate is produced by the double Newton step.

Example 6.2. We consider the DARE (1.1) with $n = m = 8$ and

$$A = \mathrm{diag}\left(\begin{pmatrix} -1 & & \\ & 1 & \\ & & 1 \end{pmatrix}, \begin{pmatrix} \frac{\sqrt{3}}{2} & \frac{1}{2} \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} \end{pmatrix}, \begin{pmatrix} \frac{1}{2} & 1 & \\ & \frac{1}{2} & 1 \\ & & \frac{1}{2} \end{pmatrix} \right),$$

$$B = \begin{pmatrix} 1 & & & \\ 1 & 1 & & \\ & \ddots & \ddots & \\ & & 1 & 1 \end{pmatrix}, \quad C = 0, \quad Q = 0, \quad R = I.$$

For this example, $X_+ = 0$ and the closed-loop eigenvalues are those of A. The unimodular eigenvalues are all semisimple. We take $L_0 = \mathrm{diag}(-1, 1, 1, 1, 1, 0.1, 0.1, 0.1)$

TABLE 6.1
Performance of the modified Newton method for Example 6.1.

k	$\Vert X_k - X_+\Vert_1$	$\Vert\mathcal{R}(X_k)\Vert_1$
0	$0.5000D+01$	$0.4545D+01$
1	$0.4167D+00$	$0.1894D+00$
2	$0.1471D+00$	$0.3342D-01$
3	$0.6410D-01$	$0.7284D-02$
4	$0.3012D-01$	$0.1711D-02$
5	$0.1462D-01$	$0.4153D-03$
6	$0.7205D-02$	$0.1023D-03$
7	$0.3577D-02$	$0.2540D-04$
8	$0.1782D-02$	$0.6328D-05$
9	$0.3170D-05$	$0.2009D-10$

TABLE 6.2
Performance of the modified Newton method for Example 6.2.

k	$\Vert X_k - X_+\Vert_1$	$\Vert\mathcal{R}(X_k)\Vert_1$
0	$0.2344D+02$	$0.2327D+02$
1	$0.2273D+01$	$0.1855D+01$
2	$0.3733D+00$	$0.1766D+00$
3	$0.1419D+00$	$0.2444D-01$
4	$0.6291D-01$	$0.6681D-02$
5	$0.2987D-01$	$0.1611D-02$
6	$0.1458D-01$	$0.3826D-03$
7	$0.7204D-02$	$0.9472D-04$
8	$0.3581D-02$	$0.2357D-04$
9	$0.1785D-02$	$0.5877D-05$
10	$0.8914D-03$	$0.1467D-05$
11	$0.4454D-03$	$0.3666D-06$
12	$0.2226D-03$	$0.9161D-07$
13	$0.3986D-07$	$0.1312D-10$

so that $A_0 = A - BL_0$ is d-stable, and apply the modified Newton method with $\epsilon = 10^{-10}$. The results are recorded in Table 6.2. Again, the last iterate is produced by the double Newton step.

In both examples, the convergence of the Newton method is linear and the final double Newton step reduces the error significantly. We have by (4.1) that $\Vert\mathcal{R}(X_k)\Vert \leq c\Vert X_k - X_+\Vert^2$, where X_k are the Newton iterates. The last iterate, Y_l, is produced by the double Newton step and $\Vert\mathcal{R}(Y_l)\Vert \leq c\Vert Y_l - X_+\Vert^2$ is not necessarily true. Typically, for l large enough, the error $\Vert Y_l - X_+\Vert$ is comparable to $\Vert\mathcal{R}(X_{l-1})\Vert$.

Acknowledgments. The author wishes to thank Dr. Peter Lancaster for his many helpful suggestions. Careful comments by Dr. Peter Benner are also gratefully acknowledged.

REFERENCES

[1] R. H. BARTELS AND G. W. STEWART, *Solution of the matrix equation* $AX + XB = C$, Comm. ACM, 15 (1972), pp. 820–826.

[2] P. BENNER, *Contributions to the Numerical Solution of Algebraic Riccati Equations and Related Eigenvalue Problems*, Logos-Verlag, Berlin, 1997.

[3] P. BENNER, A. J. LAUB, AND V. MEHRMANN, *A Collection of Benchmark Examples for the Numerical Solution of Algebraic Riccati Equations* II: *Discrete-Time Case*, Tech. report SPC 95-23, Fakultät für Mathematik, Technische Universität Chemnitz-Zwickau, Germany, 1995.

[4] S. W. Chan, G. C. Goodwin, and K. S. Sin, *Convergence properties of the Riccati difference equation in optimal fitering of nonstabilizable systems*, IEEE Trans. Automat. Control, 29 (1984), pp. 110–118.

[5] 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.

[6] D. W. Decker and C. T. Kelley, *Newton's method at singular points* I, SIAM J. Numer. Anal., 17 (1980), pp. 66–70.

[7] D. W. Decker and C. T. Kelley, *Convergence acceleration for Newton's method at singular points*, SIAM J. Numer. Anal., 19 (1982), pp. 219–229.

[8] A. Emami-Naeini and G. F. Franklin, *Comments on "On the numerical solution of the discrete-time algebraic Riccati equation,"* IEEE Trans. Automat. Control, 25 (1980), pp. 1015–1016.

[9] I. Gohberg, P. Lancaster, and L. Rodman, *Matrix Polynomials*, Academic Press, New York, 1982.

[10] A. Griewank and M. R. Osborne, *Analysis of Newton's method at irregular singularities*, SIAM J. Numer. Anal., 20 (1983), pp. 747–773.

[11] C.-H. Guo and P. Lancaster, *Analysis and modification of Newton's method for algebraic Riccati equations*, Math. Comp., 67 (1998), pp. 1089–1105.

[12] V. Hernández and A. Pastor, *On the Kleinman iteration for periodic nonstabilizable systems*, in Proc. European Control Conference ECC97, No. 946, 1997.

[13] G. A. Hewer, *An iterative technique for the computation of the steady-state gains for the discrete optimal regulator*, IEEE Trans. Automat. Control, 16 (1971), pp. 382–384.

[14] V. Ionescu and M. Weiss, *On computing the stabilizing solution of the discrete-time Riccati equation*, Linear Algebra Appl., 174 (1992), pp. 229–238.

[15] L. V. Kantorovich and G. P. Akilov, *Functional Analysis in Normed Spaces*, Pergamon, New York, 1964.

[16] C. T. Kelley, *A Shamanskii-like acceleration scheme for nonlinear equations at singular roots*, Math. Comp., 47 (1986), pp. 609–623.

[17] 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.

[18] P. Lancaster and L. Rodman, *Algebraic Riccati Equations*, Clarendon Press, Oxford, 1995.

[19] P. Lancaster and M. Tismenetsky, *The Theory of Matrices*, 2nd ed., Academic Press, Orlando, FL, 1985.

[20] V. L. Mehrmann, *The Autonomous Linear Quadratic Control Problem*, Lecture Notes in Control and Inform. Sci. 163, Springer-Verlag, Berlin, 1991.

[21] J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, 1970.

[22] A. C. M. Ran and R. Vreugdenhil, *Existence and comparison theorems for algebraic Riccati equations for continuous- and discrete-time systems*, Linear Algebra Appl., 99 (1988), pp. 63–83.

[23] G. W. Reddien, *On Newton's method for singular problems*, SIAM J. Numer. Anal., 15 (1978), pp. 993–996.

[24] V. Sima, *An efficient Schur method to solve the stabilizing problem*, IEEE Trans. Automat. Control, 26 (1981), pp. 724–725.

[25] P. Van Dooren, *A generalized eigenvalue approach for solving Riccati equations*, SIAM J. Sci. Comput., 2 (1981), pp. 121–135.

[26] H. K. Wimmer, *Normal forms of symplectic pencils and the discrete algebraic Riccati equation*, Linear Algebra Appl., 147 (1991), pp. 411–440.

SIAM J. MATRIX ANAL. APPL.
Vol. 20, No. 2, pp. 295–302

THE EQUATIONS $A^TX \pm X^TA = B$[*]

H. W. BRADEN[†]

Abstract. We give necessary and sufficient conditions for the matrix equations

$$A^TX \pm X^TA = B$$

to admit solutions and give their general solution. Because A may be singular, the solution involves the generalized inverse of A. The $(-)$ equation underlies the modern R-matrix approach to completely integrable Hamiltonian systems. This paper provides a new and algorithmic approach to constructing such R-matrices.

Key words. generalized inverse, R-matrix

AMS subject classifications. Primary, 15A06, 15A09, 15A24, 58F07; Secondary, 70G99

PII. S0895479897323270

1. Introduction.

The purpose of this article is to analyze the matrix equations

$$A^TX \pm X^TA = B, \tag{1}$$

where B is an arbitrary square matrix. We shall give necessary and sufficient conditions for (1) to admit solutions, together with their general solutions. Before describing how the equations arose, let us describe the result. Because A need not be either square or invertible, we introduce a generalized inverse G satisfying

$$AGA = A. \tag{2}$$

Such a generalized inverse always exists, and further restrictions are possible. (Accounts of generalized inverses may be found in [3, 7, 11, 12].) For example, if G satisfies (2), then $\bar{G} = GAG$ satisfies both

$$A\bar{G}A = A \qquad \text{and} \qquad \bar{G}A\bar{G} = \bar{G}. \tag{3}$$

Indeed the Moore–Penrose inverse—which is unique and always exists—further satisfies $(AG)^\dagger = AG$, $(GA)^\dagger = GA$. We assume here only that (2) is satisfied. Given such a G we define the projection operators $P_1 = GA$ and $P_2 = AG$ which satisfy

$$AP_1 = P_2A = A. \tag{4}$$

With these quantities defined, we may now state our result.

THEOREM 1. *The matrix equation* (1) *has solutions if and only if*

$$\mathrm{B}^\mathrm{T} = \pm\mathrm{B}, \tag{C1}$$

$$(1 - \mathrm{P}_1^\mathrm{T})\mathrm{B}(1 - \mathrm{P}_1) = 0, \tag{C2}$$

[*] Received by the editors June 23, 1997; accepted for publication (in revised form) by R. Bhatia March 12, 1998; published electronically October 20, 1998.
http://www.siam.org/journals/simax/20-2/32327.html

[†] Department of Mathematics and Statistics, The University of Edinburgh, Edinburgh EH9 3JZ, UK (hwb@ed.ac.uk).

in which case the general solution is

$$X = \frac{1}{2} G^T B P_1 + G^T B (1 - P_1) + (1 - P_2^T) Y + (P_2^T Z P_2) A, \tag{5}$$

where Y is arbitrary and Z is constrained only by the symmetry requirement that

$$\left(P_2^T Z P_2\right)^T = \mp P_2^T Z P_2.$$

Corollary 1. *Let X be specified as in the theorem, and suppose that $\bar{X}$ is any other solution of* (1)*, not necessarily constructed from the same generalized inverse. Then*

$$\bar{X} = X + (1 - P_2^T)(\bar{X} - X) \mp P_2^T G^T (\bar{X}^T - X^T) P_2 A. \tag{6}$$

In particular, the corollary shows that the general solution of (1) does not depend on the choice of generalized inverse G made. Further, (6) enables us to relate the parameters Y and Z of (5) appearing in different solutions.

It remains to place (1) in context. Unlike the matrix equation

$$AX - XC = B, \tag{7}$$

which has been studied extensively since Sylvester's time (see [4] for a comprehensive review and bibliography), our equation seems to have received less attention. The appearance of the transpose in (1) leads to rather different behavior from that arising in Lyapunov's equation

$$AX + XA^\dagger = B.$$

Being linear, (1) may be treated by standard techniques: in the notation of [9] we are solving

$$\left(I \otimes A^T \pm \sum_{ij} A E_{ij} \otimes E_{ij}^T \right) \mathrm{vec}(X) = \mathrm{vec}(B).$$

Our theorem gives an explicit solution. The only study of (1) I have been able to find is in [8], where the equation is treated for the particular case of A invertible and where the matrices considered are taken to be over a finite field.

Corollary 2 (Hodges). *If* $\det A \neq 0$, *then*

$$X = P^T \left(Z + \frac{1}{2} Q^T B Q \right) Q^{-1}$$

is the general solution of (1)*, where $Z = \mp Z^T$ is arbitrary and $PAQ = I$.*

This follows immediately from the theorem and observing $G = QP$.

Recent interest in (the minus form of) (1) has its origins in Hamiltonian dynamics, and I shall describe this application in the next section. Section 3 will be devoted to the proofs, and section 4 will conclude with a brief discussion.

2. An application. The motivation for our investigation of (1) arises from the modern R-matrix treatment of completely integrable mechanical systems. These are Hamiltonian mechanical systems in, say, $2N$ variables (x_i, p_i) which possess N-functionally independent conserved quantities that Poisson commute with each other [1]. A theorem of Liouville [10, 1] then says that the solutions of Hamilton's equation may be obtained by quadrature: that is, by algebraic operations, the inversion of functions, and the integration of specified functions. Historically the discovery of completely integrable systems has relied upon the considerable ingenuity of the discoverer. The last decade and a half, however, has seen a simple conceptual framework developed that unifies most known completely integrable systems. This unification has its origin in soliton bearing partial differential equations and inverse-scattering theory, which have at their heart a zero-curvature or Lax equation.

The modern approach has two ingredients:

1. Encode the equations of motion for the system under consideration in terms of a Lax pair,

$$\dot{L} = [L, M]. \tag{8}$$

   Here L, M are matrices (the Lax pair) taking values in some representation E of a Lie algebra $\mathfrak{g}$, the left-hand side is a matrix commutator, and the consistency of (8) is the equations of motion for the mechanical system.

2. Construct an $E \otimes E$ valued R-matrix [13] such that

$$\{L \overset{\otimes}{,} L\} = [R, L \otimes 1] - [R^T, 1 \otimes L]. \tag{9}$$

   The notation employed here is further elaborated below.

Upon using the cyclicity of the trace and the fact that the trace of any commutator vanishes, we see that (8) entails that the quantities $\operatorname{Tr}_E L^k$ are conserved while (9) shows that these quantities Poisson commute after noting

$$\{\operatorname{Tr}_E L^k, \operatorname{Tr}_E L^m\} = \operatorname{Tr}_{E\otimes E}\{L^k \overset{\otimes}{,} L^m\} = km \operatorname{Tr}_{E\otimes E} L^{k-1} \otimes L^{m-1}\{L \overset{\otimes}{,} L\} = 0.$$

Thus the Lax pair and R-matrix together provide us with a large number of Poisson commuting conserved quantities. The complete integrability of the system will then follow if we have sufficient functionally independent quantities amongst these. The Lax matrix L is usually constructed so that there are manifestly sufficient functionally independent quantities.

The R-matrix approach to completely integrable systems then rests upon finding solutions to (8) and (9). It has been proven [2] that the eigenvalues of L Poisson commute if and only if one can construct a matrix R satisfying (9), so the R-matrix necessarily exists for an integrable Lax pair. If we are to be honest, the ingenuity of the original discoverer of an integrable system has now been replaced by the equally arcane art of constructing Lax pairs (8). The merit of the approach, however, is that it is able to unify many disparate examples. Our interest here is on the construction of the R-matrix satisfying (9). We shall now see that this is a particular example of (1). This observation, alongside that of Theorem 1, reduces the construction of R-matrices to straightforward algebraic operations. Condition (C1) is automatically satisfied in this setting, and consequently condition (C2) gives us a straightforward necessary test for the integrability of a Lax matrix L. We remark that the known ambiguities in R-matrices correspond to the possible choices of generalized inverse. Our corollary shows that any choice of generalized inverse suffices to construct the R-matrix.

We now identify (9) with the minus form of (1) and in the process amplify the notation being employed. Let T_μ denote a basis for the (finite dimensional) Lie algebra $\mathfrak{g}$ with $[T_\mu, T_\nu] = c^{\lambda}_{\mu\nu}\, T_\lambda$ defining the structure constants of $\mathfrak{g}$; let $\phi(T_\mu) = X_\mu$ yield the representation E of the Lie algebra $\mathfrak{g}$ under consideration. We now expand L and R in terms of the resulting basis of E and $E \otimes E$, respectively. With $L = \sum_\mu L^\mu X_\mu$ the left-hand side of (9) becomes

$$\{L \overset{\otimes}{,} L\} = \sum_{\mu,\nu} \{L^\mu, L^\nu\} X_\mu \otimes X_\nu.$$

If $R = R^{\mu\nu} X_\mu \otimes X_\nu$, then R^T denotes the transpose $R^T = R^{\nu\mu} X_\mu \otimes X_\nu$; the right-hand side of (9) then yields

$$\begin{aligned}[R, L\otimes 1] - [R^T, 1 \otimes L] &= R^{\mu\nu}([X_\mu, L] \otimes X_\nu - X_\nu \otimes [X_\mu, L]) \\ &= R^{\mu\nu} L^\lambda ([X_\mu, X_\lambda] \otimes X_\nu - X_\nu \otimes [X_\mu, X_\lambda]) \\ &= (R^{\tau\nu} c^{\mu}_{\tau\lambda} L^\lambda - R^{\tau\mu} c^{\nu}_{\tau\lambda} L^\lambda) X_\mu \otimes X_\nu.\end{aligned}$$

Upon setting $A^{\mu\nu} = c^{\nu}_{\mu\lambda} L^\lambda = -\mathrm{ad}(L)^{\nu}_{\mu}$, $B^{\mu\nu} = \{L^\mu, L^\nu\}$, and $X^{\mu\nu} = R^{\mu\nu}$ we see that (9) is an example of the minus equation of (1). Further, the antisymmetry of the Poisson bracket means that (C1) is satisfied for this problem. The construction of the R-matrix thus reduces to finding the generalized inverse to $\mathrm{ad}(L)$ and verifying the (now) necessary and sufficient condition (C2). Elsewhere [6] I have shown how to construct a generalized inverse to $\mathrm{ad}(L)$ for generic $L \in \mathfrak{g}$ together with an explicit form for the Moore–Penrose inverse. That result together with our present theorem gives us the R-matrix for generic L and an algorithm to compute it.

We conclude the section with an example to make matters concrete. Although the example is trivially integrable, our treatment is new.

2.1. An example. Consider matrices L, M given by

$$L = \begin{pmatrix} p_1 & f_\alpha \\ f_{-\alpha} & p_2 \end{pmatrix}, \qquad M = \begin{pmatrix} 0 & f'_\alpha \\ f'_{-\alpha} & 0 \end{pmatrix}. \tag{10}$$

Here $f_\alpha = f_\alpha(x_1 - x_2)$ and $f_{-\alpha} = f_{-\alpha}(x_2 - x_1)$ are two arbitrary functions, and we view L and M as being in the natural representation of $\mathfrak{g} = gl_2$. One finds for these matrices that the consistency of (8) is equivalent to the equations of motion for the Hamiltonian system (of particles on the line) with Hamiltonian

$$H = \frac{1}{2}\mathrm{Tr} L^2 = \frac{1}{2}(p_1^2 + p_2^2) + f_\alpha(x_1 - x_2) f_{-\alpha}(x_2 - x_1).$$

Of course this system is separable and thus integrable because the potential depends only on the difference of coordinates, yet nonetheless it provides an instructive example. Here the conserved quantity $\mathrm{Tr}L$ corresponds to the centre of mass momentum.

We shall now show how our theorem provides the R-matrix to this problem,

$$R = \frac{f'_\alpha}{f_{-\alpha}} E_{12} \otimes E_{12} + \frac{f'_{-\alpha}}{f_\alpha} E_{21} \otimes E_{21} + \begin{pmatrix} p & q \\ -\Lambda^{-1} v p - F u & -\Lambda^{-1} v q - F \Lambda^T \end{pmatrix}. \tag{11}$$

Here the final term represents the ambiguity in the R-matrix: p, q are arbitrary 2×2 matrices coming from the Y term of (5), while F is a symmetric 2×2 matrix built from Z; the remaining matrices are fixed by the problem and specified below. Even in this simple problem the complete R-matrix (11) appears new.

To apply our theorem we must first compute the matrices $A = -(\mathrm{ad}L)^T$ and B. We order our basis of gl_2 as $\{E_{11}, E_{22}, E_{12}, E_{21}\}$, where E_{ij} is the matrix with 1 in the ijth position and zero elsewhere. Then

$$\mathrm{ad}(L) = \begin{pmatrix} 0 & 0 & -f_{-\alpha} & f_\alpha \\ 0 & 0 & f_{-\alpha} & -f_\alpha \\ -f_\alpha & f_\alpha & p_1 - p_2 & 0 \\ f_{-\alpha} & -f_{-\alpha} & 0 & p_2 - p_1 \end{pmatrix} = \begin{pmatrix} 0 & u^T \\ v & \Lambda \end{pmatrix}$$

and (upon using $\{p_i, x_j\} = \delta_{ij}$)

$$B = \begin{pmatrix} 0 & 0 & f'_\alpha & -f'_{-\alpha} \\ 0 & 0 & -f'_\alpha & f'_{-\alpha} \\ -f'_\alpha & f'_\alpha & 0 & 0 \\ f'_{-\alpha} & -f'_{-\alpha} & 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & -\mu^T \\ \mu & 0 \end{pmatrix} = -B^T.$$

With our choice of basis we find a block structure to ad$L$ and B; this enables us to define the various 2×2 matrices u, v, Λ, and μ. Provided $p_1 \neq p_2$ (a generic condition) the generalized inverse of $\mathrm{ad}(L)$ is found to be $\begin{pmatrix} 0 & 0 \\ 0 & \Lambda^{-1} \end{pmatrix}$; the generalized inverse of A is thus minus the transpose of this. The projection operators defined earlier are then

$$P_1 = \begin{pmatrix} 0 & 0 \\ \Lambda^{-1T}u & 1 \end{pmatrix}, \qquad P_2 = \begin{pmatrix} 0 & v^T\Lambda^{-1T} \\ 0 & 1 \end{pmatrix}.$$

For the case at hand the constraint (C2) takes the form

$$\text{(C2)} \qquad 0 = \mu^T\Lambda^{-1T}u - u^T\Lambda^{-1}\mu,$$

which is readily verified to be true. Applying our main result (5) then shows the general R-matrix to be of the form

$$\text{(12)} \qquad R = \begin{pmatrix} 0 & 0 \\ -\Lambda^{-1}\mu & 0 \end{pmatrix} + \begin{pmatrix} p & q \\ -\Lambda^{-1}vp - Fu & -\Lambda^{-1}vq - F\Lambda^T \end{pmatrix},$$

where

$$\text{(13)} \qquad -\Lambda^{-1}\mu = \frac{1}{p_1 - p_2} \begin{pmatrix} f'_\alpha & -f'_\alpha \\ f'_{-\alpha} & -f'_{-\alpha} \end{pmatrix}.$$

The second term of (12) characterizes the ambiguity in R. We have parameterized the matrices Y, Z in (5) by $Y = \begin{pmatrix} p & q \\ r & s \end{pmatrix}$ and $Z = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$. The matrices p, q are arbitrary, while the entries of Z are such that

$$F = \Lambda^{-1}vav^T\Lambda^{-1T} + d + \Lambda^{-1}vb + cv^T\Lambda^{-1T}$$

is symmetric. These are the entries of (11) mentioned above.

We remark that (in view of (13)) the R-matrix given by the first term of (12) depends on all of the dynamical variables. However, by choosing $p = q = 0$ and

$$F = \frac{-c}{p_1 - p_2} \begin{pmatrix} f'_\alpha/f_{-\alpha} & 0 \\ 0 & -f'_{-\alpha}/f_\alpha \end{pmatrix} + \frac{1-c}{p_1 - p_2} \begin{pmatrix} 0 & f'_\alpha/f_\alpha \\ f'_\alpha/f_\alpha & (f'_\alpha f_{-\alpha} + f'_{-\alpha}f_\alpha)/f_\alpha^2 \end{pmatrix}$$

we obtain the one parameter family of momentum independent R-matrices

$$R = c\,\frac{f'_\alpha}{f_{-\alpha}} E_{12}\otimes E_{12} + (1-c)\,\frac{f'_\alpha}{f_\alpha}\left(E_{12}\otimes E_{21} - E_{21}\otimes E_{12}\right) + \frac{(1-c)f'_\alpha f_{-\alpha} + f'_{-\alpha} f_\alpha}{f_\alpha^2} E_{21}\otimes E_{21}.$$

In particular we note the particularly simple form (when $c = 1$)

$$R = \frac{f'_\alpha}{f_{-\alpha}} E_{12}\otimes E_{12} + \frac{f'_{-\alpha}}{f_\alpha} E_{21}\otimes E_{21}$$

that we have chosen for (11) above. Finally, observe that in the situation where $f_\alpha(z) = -f_\alpha(-z) = w_\alpha(z)$ is any (odd) function, we recover the result of [5] with $c = 0$:

$$R = \frac{w'_\alpha}{w_\alpha}\left(E_{12}\otimes E_{21} - E_{21}\otimes E_{12}\right). \tag{14}$$

In fact this present work extends that of [5] for the present theorem characterizes the full ambiguity of the R-matrix, while the earlier cited work gives only a subspace of this.

3. Proof of the theorem. Suppose that solutions of (1) exist. By taking the transpose of both sides of this equation it is immediate that B must be (skew-) symmetric (C1). Similarly, upon multiplying both sides of (1) by $1 - P_1^T$ on the left and $1 - P_1$ on the right and then employing the projection properties (4), we obtain the constraint (C2). The constraints (C1), (C2) are therefore necessary. We will establish their sufficiency by showing that the general solution is as stated.

Suppose that X is given by (5). Then

$$A^T X = \frac{1}{2} P_1^T B P_1 + P_1^T B(1 - P_1) + A^T Z A$$

and

$$X^T A = \frac{1}{2} P_1^T B^T P_1 + (1 - P_1^T) B^T P_1 + A^T Z^T A.$$

Upon using the (anti)symmetry (C1) of B and the appropriate symmetry of $A^T Z A$ we have

$$A^T X \pm X^T A = P_1^T B P_1 + P_1^T B(1 - P_1) + (1 - P_1^T) B P_1 = B - (1 - P_1^T) B (1 - P_1).$$

Now employing (C2) shows that the final term of this expression vanishes and consequently that X given by (5) satisfies (1). This has established the sufficiency of the constraints for a solution to exist, and it remains to show that (5) is indeed the general solution.

Let $\bar{X}$ be any solution of (1) and let X_0 be the solution given by

$$X_0 = \frac{1}{2} G^T B P_1 + G^T B(1 - P_1).$$

Set $Y = \bar{X} - X_0$. Then by linearity

$$A^T Y \pm Y^T A = 0. \tag{15}$$

Further, the matrix Y may be decomposed using the projector P_2 as

$$Y = (1 - P_2^T)Y + P_2^TY. \tag{16}$$

Utilizing (15) we may rewrite the final term in this decomposition as

$$P_2^TY = P_2^TP_2^TY = P_2^TG^TA^TY = \mp P_2^TG^TY^TA = \mp P_2^T\left(G^TY^T\right)P_2A = P_2^TZP_2A,$$

where we have defined $Z = \mp G^TY^T = \mp G^T(\bar{X}^T - X_0^T)$. Further use of (15) also yields

$$(P_2^TZP_2)^T = \mp P_2^TYGP_2 = \mp P_2^TG^TA^TYGP_2 = P_2^TG^TY^TP_2P_2 = \mp(P_2^TZP_2).$$

Putting these results together shows

$$\bar{X} = X_0 + (1 - P_2^T)Y + P_2^TZP_2A \tag{17}$$

with Y and Z as given in Theorem 1, thus establishing that (5) is the general solution of (1). Further, (17) establishes the result of the corollary (6).

4. Discussion. This paper has been devoted to the matrix equations (1): we have given necessary and sufficient conditions for them to possess solutions and presented general solutions when these hold. The solutions are constructed from any generalized inverse of the matrix A, with differing generalized inverses yielding the same general solution. A particular case of the (–) matrix equation is (9), which arises in the modern R-matrix approach to completely integrable equations. In this setting (C2) becomes the necessary and sufficient condition for an R-matrix to exist. The paper thus provides a new and algorithmic approach to constructing R-matrices without the usual recourse of guessing a suitable ansatz; applications and extensions have been presented elsewhere [6]. The equation

$$AX - X^TC = B$$

is a natural generalization of both (7) and the equations studied here. I don't know of a simple explicit solution to this equation at present.

Acknowledgments. I thank the referee for suggested improvements to the presentation of this paper.

REFERENCES

[1] V.I. Arnold, *Mathematical Methods of Classical Mechanics*, Springer-Verlag, New York, 1978.

[2] O. Babelon and C.M. Viallet, *Hamiltonian structures and Lax equations*, Phys. Lett. B, 237 (1990), pp. 411–416.

[3] A. Ben-Israel and T.N.E. Greville, *Generalized Inverses: Theory and Applications*, Krieger, Huntington, NY, 1974.

[4] R. Bhatia and P. Rosenthal, *How and why to solve the operator equation $AX - XB = Y$*, Bull. London Math. Soc., 29 (1997), pp. 1–21.

[5] H.W. Braden and T. Suzuki, *R-matrices for the $n = 2, 3$ elliptic Calogero-Moser Models*, Phys. Lett. A, 192 (1994), pp. 17–21.

[6] H.W. Braden, *R-matrices and generalized inverses*, J. Phys. A, 30 (1997), pp. L485–L493.

[7] S.R. Caradus, *Generalized Inverses and Operator Theory*, Queen's Papers in Pure and Appl. Math. 50, Queen's University, Kingston, ON, 1978.

[8] J.H. Hodges, *Some matrix equations over a finite field*, Ann. Mat. Pura Appl.(4), 44 (1957), pp. 245–250.

[9] R.A. Horn and C.R. Johnson, *Topics in Matrix Analysis*, Cambridge University Press, Cambridge, 1991.

[10] J. Liouville, *Note sur les équations de la dynamique*, J. Math. Pures Appl., 14 (1855), pp. 137–138.

[11] R.M. Pringle and A.A. Rayner, *Generalized Inverse Matrices with Applications to Statistics*, Griffin's Statistical Monographs and Courses 28, Charles Griffin, London, 1971.

[12] C.R.A Rao and S.K. Mitra, *Generalized Inverse of Matrices and its Applications,* John Wiley and Sons, New York, 1971.

[13] M.A. Semenov-Tian-Shansky, *What is a classical r-matrix?*, Funct. Anal. Appl., 17 (1983), pp. 17–33.

SIAM J. MATRIX ANAL. APPL.
Vol. 20, No. 2, pp. 303–353

THE GEOMETRY OF ALGORITHMS WITH ORTHOGONALITY CONSTRAINTS*

ALAN EDELMAN†, TOMÁS A. ARIAS‡, AND STEVEN T. SMITH§

Abstract. In this paper we develop new Newton and conjugate gradient algorithms on the Grassmann and Stiefel manifolds. These manifolds represent the constraints that arise in such areas as the symmetric eigenvalue problem, nonlinear eigenvalue problems, electronic structures computations, and signal processing. In addition to the new algorithms, we show how the geometrical framework gives penetrating new insights allowing us to create, understand, and compare algorithms. The theory proposed here provides a taxonomy for numerical linear algebra algorithms that provide a top level mathematical view of previously unrelated algorithms. It is our hope that developers of new algorithms and perturbation theories will benefit from the theory, methods, and examples in this paper.

Key words. conjugate gradient, Newton's method, orthogonality constraints, Grassmann manifold, Stiefel manifold, eigenvalues and eigenvectors, invariant subspace, Rayleigh quotient iteration, eigenvalue optimization, sequential quadratic programming, reduced gradient method, electronic structures computation, subspace tracking

AMS subject classifications. 49M07, 49M15, 53B20, 65F15, 15A18, 51F20, 81V55

PII. S0895479895290954

1. Introduction.

Problems on the Stiefel and Grassmann manifolds arise with sufficient frequency that a unifying investigation of algorithms designed to solve these problems is warranted. Understanding these manifolds, which represent orthogonality constraints (as in the symmetric eigenvalue problem), yields penetrating insight into many numerical algorithms and unifies seemingly unrelated ideas from different areas.

The optimization community has long recognized that linear and quadratic constraints have special structure that can be exploited. The Stiefel and Grassmann manifolds also represent special constraints. The main contribution of this paper is a framework for algorithms involving these constraints, which draws upon ideas from numerical linear algebra, optimization, differential geometry, and has been inspired by certain problems posed in engineering, physics, and chemistry. Though we do review the necessary background for our intended audience, this is not a survey paper. This paper uses mathematics as a tool so that we can understand the deeper geometrical structure underlying algorithms.

In our first concrete problem we minimize a function $F(Y)$, where Y is constrained to the set of n-by-p matrices such that $Y^TY = I$ (we call such matrices orthonormal), and we make the further homogeneity assumption that $F(Y) = F(YQ)$, where Q is

*Received by the editors August 28, 1995; accepted for publication (in revised form) by M. L. Overton June 20, 1997; published electronically October 20, 1998.
http://www.siam.org/journals/simax/20-2/29095.html

†Department of Mathematics Room 2-380, Massachusetts Institute of Technology, Cambridge, MA 02139 (edelman@math.mit.edu). This research was supported by a fellowship from the Alfred P. Sloan Foundation and NSF grants 9501278-DMS and 9404326-CCR.

‡Department of Physics, Massachusetts Institute of Technology, Cambridge, MA 02139 (muchomas@mit.edu). This research was supported by an NSF/MRSEC Seed Project grant from the MIT Center for Material Science and Engineering.

§MIT Lincoln Laboratory, 244 Wood Street, Lexington, MA 02173 (stsmith@ll.mit.edu). This research was sponsored by DARPA under Air Force contract F19628-95-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the author and are not necessarily endorsed by the United States Air Force.

any p-by-p orthogonal matrix. In other words, the objective function depends only on the subspace spanned by the columns of Y; it is invariant to any choice of basis. The set of p-dimensional subspaces in $\mathbf{R}^n$ is called the Grassmann manifold. (Grassmann originally developed the idea in 1848, but his writing style was considered so obscure [1] that it was appreciated only many years later. One can find something of the original definition in his later work [48, Chap. 3, Sec. 1, Article 65].) To the best of our knowledge, the geometry of the Grassmann manifold has never been explored in the context of optimization algorithms, invariant subspace computations, physics computations, or subspace tracking. Useful ideas from these areas, however, may be put into the geometrical framework developed in this paper.

In our second problem we minimize $F(Y)$ without the homogeneity condition $F(Y) = F(YQ)$ mentioned above, i.e., the optimization problem is defined on the set of n-by-p orthonormal matrices. This constraint surface is known as the Stiefel manifold, which is named for Eduard Stiefel, who considered its topology in the 1930s [82]. This is the same Stiefel who in collaboration with Magnus Hestenes in 1952 originated the conjugate gradient algorithm [49]. Both Stiefel's manifold and his conjugate gradient algorithm play an important role in this paper. The geometry of the Stiefel manifold in the context of optimization problems and subspace tracking was explored by Smith [75]. In this paper we use numerical linear algebra techniques to simplify the ideas and algorithms presented there so that the differential geometric ideas seem natural and illuminating to the numerical linear algebra and optimization communities.

The first author's original motivation for studying this problem came from a response to a linear algebra survey [30], which claimed to be using conjugate gradient to solve large dense eigenvalue problems. The second and third authors were motivated by two distinct engineering and physics applications. The salient question became: What does it mean to use conjugate gradient to solve eigenvalue problems? Is this the Lanczos method? As we shall describe, there are dozens of proposed variations on the conjugate gradient and Newton methods for eigenvalue and related problems, none of which are Lanczos. These algorithms are not all obviously related. The connections among these algorithms have apparently not been appreciated in the literature while in some cases numerical experiments have been the only basis for comparison when no theoretical understanding was available. The existence of so many variations in so many applications compelled us to ask for the big picture: What is the mathematics that unifies all of these apparently distinct algorithms? This paper contains our proposed unification.

We summarize by itemizing what is new in this paper.

1. Algorithms for Newton and conjugate gradient methods on the Grassmann and Stiefel manifolds that naturally use the geometry of these manifolds. In the special cases that we are aware of, our general algorithms are competitive up to small constant factors with the best known special algorithms. Conjugate gradient and Newton on the Grassmann manifold have never been explicitly studied before. Stiefel algorithms have been studied before [75], but the ideas here represent considerable simplifications.

2. A geometrical framework with the right mix of abstraction and concreteness to serve as a foundation for any numerical computation or algorithmic formulation involving orthogonality constraints, including the symmetric eigenvalue problem. We believe that this is a useful framework because it connects apparently unrelated ideas; it is simple and mathematically natural. The framework provides new insights into

existing algorithms in numerical linear algebra, optimization, signal processing, and electronic structures computations, and it suggests new algorithms. For example, we connect the ideas of geodesics and the cubic convergence of the Rayleigh quotient iteration, the CS decomposition, and sequential quadratic programming. We also interpret the ill-conditioning of eigenvectors of a symmetric matrix with multiple eigenvalues as the singularity of Stiefel and Grassmann coordinates.

3. Though geometrical descriptions of the Grassmann and Stiefel manifolds are available in many references, ours is the first to use methods from numerical linear algebra emphasizing computational efficiency of algorithms rather than abstract general settings.

The remainder of this paper is organized into three sections. The geometrical ideas are developed in section 2. This section provides a self-contained introduction to geometry, which may not be familiar to some readers, while deriving the new geometrical formulas necessary for the algorithms of section 3, and the insights of section 3 provide descriptions of new algorithms for optimization on the Grassmann and Stiefel manifolds. Concrete examples of the new insights gained from this point of view are presented in section 4. Because we wish to discuss related literature in the context developed in sections 2 and 3, we defer discussion of the literature to section 4, where specific applications of our theory are organized.

2. Differential geometric foundation for numerical linear algebra.

A geometrical treatment of the Stiefel and Grassmann manifolds appropriate for numerical linear algebra cannot be found in standard differential geometry references. For example, what is typically required for practical conjugate gradient computations involving n-by-p orthonormal matrices are algorithms with complexity of order np^2. In this section we derive new formulas that may be used in algorithms of this complexity in terms of standard operations from numerical linear algebra. These formulas will be used in the algorithms presented in the following section. Because we focus on computations, our approach differs from the more general (and powerful) coordinate-free methods used by modern geometers [18, 47, 54, 62, 79, 87]. Boothby [8] provides an undergraduate level introduction to the coordinate-free approach.

For readers with a background in differential geometry, we wish to point out how we use extrinsic coordinates in a somewhat unusual way. Typically, one uses a parameterization of the manifold (e.g., $x = \cos u \sin v$, $y = \sin u \sin v$, $z = \cos v$ for the sphere) to derive metric coefficients and Christoffel symbols in terms of the parameters (u and v). Instead, we only use extrinsic coordinates subject to constraints (e.g., (x, y, z) such that $x^2 + y^2 + z^2 = 1$). This represents points with more parameters than are intrinsically necessary, but we have found that the simplest (hence computationally most useful) formulas for the metric and Christoffel symbol are obtained in this manner. The choice of coordinates does not matter abstractly, but on a computer the correct choice is essential.

We now outline this section. After defining the manifolds of interest to us in section 2.1, we take a close look at the Stiefel manifold as a submanifold of Euclidean space in section 2.2. This introduces elementary ideas from differential geometry and provides the geometric structure of the orthogonal group (a special case of the Stiefel manifold), which will be used throughout the rest of the paper. However, the Euclidean metric is not natural for the Stiefel manifold, which inherits a canonical metric from its definition as a quotient space. Therefore, we introduce the quotient space point of view in section 2.3. With this viewpoint, we then derive our formulas for geodesics and parallel translation for the Stiefel and Grassmann manifold in

Table 2.1
Representations of subspace manifolds.

Space	Symbol	Matrix rep.	Quotient rep.
Orthogonal group	O_n	Q	–
Stiefel manifold	$V_{n,\,p}$	Y	O_n/O_{n-p}
Grassmann manifold	$G_{n,\,p}$	None	$\left\{\begin{array}{c} V_{n,\,p}/O_p \\ \text{or} \\ O_n/\left(O_p \times O_{n-p}\right) \end{array}\right\}$

sections 2.4 and 2.5. Finally, we describe how to incorporate these formulae into conjugate gradient and Newton methods in section 2.6.

2.1. Manifolds arising in numerical linear algebra. For simplicity of exposition, but for no fundamental reason, we will concentrate on real matrices. All ideas carry over naturally to complex matrices. Spaces of interest are as follows:

1. The orthogonal group O_n consisting of n-by-n orthogonal matrices;
2. The Stiefel manifold $V_{n,\,p}$ consisting of n-by-p "tall-skinny" orthonormal matrices;
3. The Grassmann manifold $G_{n,\,p}$ obtained by identifying those matrices in $V_{n,\,p}$ whose columns span the same subspace (a quotient manifold).

Table 2.1 summarizes the definitions of these spaces. Our description of $G_{n,\,p}$ is necessarily more abstract than O_n or $V_{n,\,p}$. $G_{n,\,p}$ may be defined as the set of all p-dimensional subspaces of an n-dimensional space.

We shall benefit from two different yet equivalent modes of describing our spaces: concrete representations and quotient space representations. Table 2.2 illustrates how we store elements of $V_{n,\,p}$ and $G_{n,\,p}$ in a computer. A point in the Stiefel manifold $V_{n,\,p}$ is represented by an n-by-p matrix. A point on the Grassmann manifold $G_{n,\,p}$ is a linear subspace, which may be specified by an arbitrary orthogonal basis stored as an n-by-p matrix. An important difference here is that, unlike points on the Stiefel manifold, the choice of matrix is not unique for points on the Grassmann manifold.

The second mode of representation, the more mathematical, is useful for obtaining closed-form expressions for the geometrical objects of interest. It is also the "proper" theoretical setting for these manifolds. Here, we represent the manifolds as quotient spaces. Points in the Grassmann manifold are equivalence classes of n-by-p orthogonal matrices, where two matrices are equivalent if their columns span the same p-dimensional subspace. Equivalently, two matrices are equivalent if they are related by right multiplication of an orthogonal p-by-p matrix. Therefore, $G_{n,\,p} = V_{n,\,p}/O_p$. On the computer, by necessity, we must pick a representative of the equivalence class to specify a point.

TABLE 2.2
Computational representation of subspace manifolds.

SPACE	DATA STRUCTURE	REPRESENTS	TANGENTS Δ
Stiefel manifold	Y	one point	$Y^T\Delta = \text{skew-symmetric}$
Grassmann manifold	Y	entire equivalence class	$Y^T\Delta = 0$

The Stiefel manifold may also be defined as a quotient space but arising from the orthogonal group. Here, we identify two orthogonal matrices if their first p columns are identical or, equivalently, if they are related by right multiplication of a matrix of the form $\left(\begin{smallmatrix} I & 0 \\ 0 & Q \end{smallmatrix}\right)$, where Q is an orthogonal $(n-p)$-by-$(n-p)$ block. Therefore, $V_{n,\,p} = O_n/O_{n-p}$. With the Stiefel manifold so represented, one has yet another representation of the Grassmann manifold, $G_{n,\,p} = O_n/(O_p \times O_{n-p})$.

2.2. The Stiefel manifold in Euclidean space. The Stiefel manifold $V_{n,\,p}$ may be embedded in the np-dimensional Euclidean space of n-by-p matrices. When $p = 1$, we simply have the sphere, while when $p = n$, we have the group of orthogonal matrices known as O_n. These two special cases are the easiest and arise in numerical linear algebra the most often.

Much of this section, which consists of three subsections, is designed to be a painless and intuitive introduction to differential geometry in Euclidean space. Section 2.2.1 is elementary. It derives formulas for projections onto the tangent and normal spaces. In section 2.2.2, we derive formulas for geodesics on the Stiefel manifold in Euclidean space. We then discuss parallel translation in section 2.2.3.

In the two special cases when $p = 1$ and $p = n$, the Euclidean metric and the canonical metric to be discussed in section 2.4 are the same. Otherwise they differ.

2.2.1. Tangent and normal space. Intuitively, the tangent space at a point is the plane tangent to the submanifold at that point, as shown in Figure 2.1. For d-dimensional manifolds, this plane is a d-dimensional vector space with origin at the point of tangency. The normal space is the orthogonal complement. On the sphere, tangents are perpendicular to radii, and the normal space is radial. In this subsection, we will derive the equations for the tangent and normal spaces on the Stiefel manifold. We also compute the projection operators onto these spaces.

An equation defining tangents to the Stiefel manifold at a point Y is easily obtained by differentiating $Y^TY = I$, yielding $Y^T\Delta + \Delta^TY = 0$, i.e., $Y^T\Delta$ is skew-symmetric. This condition imposes $p(p+1)/2$ constraints on Δ, or, equivalently, the vector space of all tangent vectors Δ has dimension

$$np - \frac{p(p+1)}{2} = \frac{p(p-1)}{2} + p(n-p). \tag{2.1}$$

Both sides of (2.1) are useful for the dimension counting arguments that will be employed.

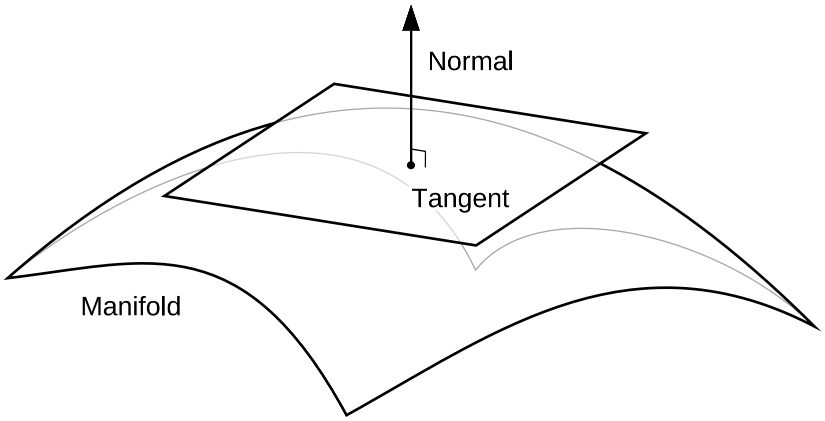


Fig. 2.1. *The tangent and normal spaces of an embedded or constraint manifold.*

The normal space is defined to be the orthogonal complement of the tangent space. Orthogonality depends upon the definition of an inner product, and because in this subsection we view the Stiefel manifold as an embedded manifold in Euclidean space, we choose the standard inner product

$$g_e(\Delta_1, \Delta_2) = \operatorname{tr} \Delta_1^T \Delta_2 \tag{2.2}$$

in np-dimensional Euclidean space (hence the subscript e), which is also the Frobenius inner product for n-by-p matrices. We shall also write $\langle \Delta_1, \Delta_2 \rangle$ for the inner product, which may or may not be the Euclidean one. The normal space at a point Y consists of all matrices N which satisfy

$$\operatorname{tr} \Delta^T N = 0$$

for all Δ in the tangent space. It follows that the normal space is $p(p+1)/2$ dimensional. It is easily verified that if $N = YS$, where S is p-by-p symmetric, then N is in the normal space. Since the dimension of the space of such matrices is $p(p+1)/2$, we see that the normal space is exactly the set of matrices $\{\, YS \,\}$, where S is any p-by-p symmetric matrix.

Let Z be any n-by-p matrix. Letting $\operatorname{sym}(A)$ denote $(A + A^T)/2$ and $\operatorname{skew}(A) = (A - A^T)/2$, it is easily verified that at Y

$$\pi_N(Z) = Y \operatorname{sym}(Y^T Z) \tag{2.3}$$

defines a projection of Z onto the normal space. Similarly, at Y,

$$\pi_T(Z) = Y \operatorname{skew}(Y^T Z) + (I - YY^T)Z \tag{2.4}$$

is a projection of Z onto the tangent space at Y (this is also true of the canonical metric to be discussed in section 2.4). Equation (2.4) suggests a form for the tangent space of $V_{n,\,p}$ at Y that will prove to be particularly useful. Tangent directions Δ at Y then have the general form

$$\Delta = YA + Y_\perp B \tag{2.5}$$

$$ = YA + (I - YY^T)C, \tag{2.6}$$

where A is p-by-p skew-symmetric, B is $(n-p)$-by-p, C is n-by-p, B and C are both arbitrary, and $Y_\perp$ is any n-by-$(n-p)$ matrix such that $YY^T + Y_\perp Y_\perp{}^T = I$; note that $B = Y_\perp{}^T C$. The entries in the matrices A and B parameterize the tangent space at Y with $p(p-1)/2$ degrees of freedom in A and $p(n-p)$ degrees of freedom in B, resulting in $p(p-1)/2 + p(n-p)$ degrees of freedom as seen in (2.1).

In the special case $Y = I_{n,p} \equiv \left(\begin{smallmatrix} I_p \\ 0 \end{smallmatrix}\right)$ (the first p columns of the n-by-n identity matrix), called the origin, the tangent space at Y consists of those matrices

$$X = \begin{pmatrix} A \\ B \end{pmatrix}$$

for which A is p-by-p skew-symmetric and B is $(n-p)$-by-p arbitrary.

2.2.2. Embedded geodesics. A geodesic is the curve of shortest length between two points on a manifold. A straightforward exercise from the calculus of variations reveals that *for the case of manifolds embedded in Euclidean space* the acceleration vector at each point along a geodesic is normal to the submanifold so long as the curve is traced with uniform speed. This condition is necessary and sufficient. In the case of the sphere, acceleration for uniform motion on a great circle is directed radially and therefore normal to the surface; therefore, great circles are geodesics on the sphere. One may consider embedding manifolds in spaces with arbitrary metrics. See Spivak [79, Vol. 3, p. 4] for the appropriate generalization.

Through (2.3) for the normal space to the Stiefel manifold, it is easily shown that the geodesic equation for a curve $Y(t)$ on the Stiefel manifold is defined by the differential equation

$$\ddot{Y} + Y(\dot{Y}^T\dot{Y}) = 0. \tag{2.7}$$

To see this, we begin with the condition that $Y(t)$ remains on the Stiefel manifold

$$Y^TY = I_p. \tag{2.8}$$

Taking two derivatives,

$$Y^T\ddot{Y} + 2\dot{Y}^T\dot{Y} + \ddot{Y}^TY = 0. \tag{2.9}$$

To be a geodesic, $\ddot{Y}(t)$ must be in the normal space at $Y(t)$ so that

$$\ddot{Y}(t) + Y(t)S = 0 \tag{2.10}$$

for some symmetric matrix S. Substitute (2.10) into (2.9) to obtain the geodesic equation (2.7). Alternatively, (2.7) could be obtained from the Euler–Lagrange equation for the calculus of variations problem

$$d(Y_1, Y_2) = \min_{Y(t)} \int_{t_1}^{t_2} (\operatorname{tr} \dot{Y}^T\dot{Y})^{1/2}\, dt \quad \text{such that } Y(t_1) = Y_1,\, Y(t_2) = Y_2. \tag{2.11}$$

We identify three integrals of motion of the geodesic equation (2.7). Define

$$C = Y^TY, \qquad A = Y^T\dot{Y}, \qquad S = \dot{Y}^T\dot{Y}. \tag{2.12}$$

Directly from the geodesic equation (2.7),

$$\begin{aligned} \dot{C} &= A + A^T, \\ \dot{A} &= -CS + S, \\ \dot{S} &= [A, S], \end{aligned}$$

where

$$[A, S] = AS - SA \tag{2.13}$$

is the Lie bracket of two matrices. Under the initial conditions that Y is on the Stiefel manifold ($C = I$) and $\dot{Y}$ is a tangent (A is skew-symmetric), then the integrals of the motion have the values

$$\begin{aligned} C(t) &= I, \\ A(t) &= A(0), \\ S(t) &= e^{At} S(0) e^{-At}. \end{aligned}$$

These integrals therefore identify a constant speed curve on the Stiefel manifold. In most differential geometry books, the equation of motion for geodesics is written in intrinsic coordinates in terms of so-called Christoffel symbols which specify a quadratic form of the tangent vectors. In our formulation, the form $\Gamma_e(\dot{Y}, \dot{Y}) = Y \dot{Y}^T \dot{Y}$ is written compactly in extrinsic coordinates.

With these constants of the motion, we can write an integrable equation for the final geodesic,[1]

$$\frac{d}{dt}\left(Ye^{At},\ \dot{Y}e^{At}\right) = \left(Ye^{At},\ \dot{Y}e^{At}\right)\begin{pmatrix} A & -S(0) \\ I & A \end{pmatrix},$$

with integral

$$Y(t) = \left(Y(0),\ \dot{Y}(0)\right)\ \exp t \begin{pmatrix} A & -S(0) \\ I & A \end{pmatrix}\ I_{2p,p} e^{-At}.$$

This is an exact closed form expression for the geodesic on the Stiefel manifold, but we will not use this expression in our computation. Instead we will consider the non-Euclidean canonical metric on the Stiefel manifold in section 2.4.

We mention in the case of the orthogonal group ($p = n$), the geodesic equation is obtained simply from $A = Q^T \dot{Q} =$ constant, yielding the simple solution

$$Q(t) = Q(0) e^{At}. \tag{2.14}$$

From (2.14) it is straightforward to show that on connected components of O_n,

$$d(Q_1, Q_2) = \left(\sum_{k=1}^{n} \theta_k^2\right)^{1/2}, \tag{2.15}$$

where $\{e^{i\theta_k}\}$ are the eigenvalues of the matrix $Q_1^T Q_2$ (cf. (2.67) and section 4.3).

2.2.3. Parallel translation. In Euclidean space, we move vectors parallel to themselves simply by moving the base of the arrow. On an embedded manifold, if we move a tangent vector to another point on the manifold by this technique, it is generally not a tangent vector. One can, however, transport tangents along paths on the manifold by infinitesimally removing the component of the transported vector in the normal space.

[1] We thank Ross Lippert [56] for this observation.

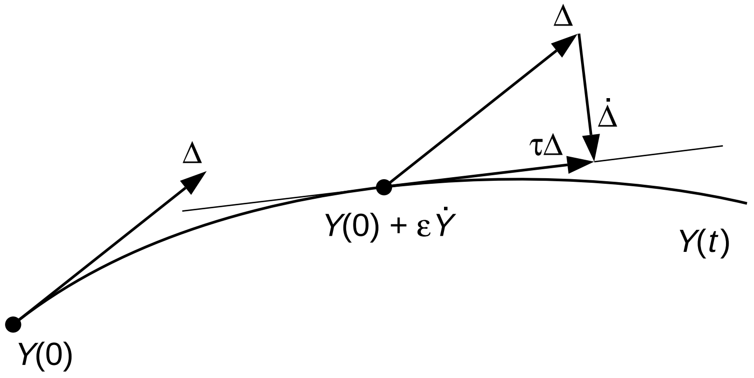


Fig. 2.2. *Parallel transport in a submanifold of Euclidean space (infinitesimal construction).*

Figure 2.2 illustrates the following idea: Imagine moving a tangent vector Δ along the curve $Y(t)$ in such a manner that every infinitesimal step consists of a parallel displacement of Δ in the Euclidean np-dimensional space, which is then followed by the removal of the normal component. If we move from $Y(0) = Y$ to $Y(\epsilon)$ then to first order, our new location is $Y + \epsilon\dot{Y}$. The equation for infinitesimally removing the component generated in the normal space as we move in the direction $\dot{Y}$ is obtained by differentiating (2.3) as follows:

$$\dot{\Delta} = -Y(\dot{Y}^T\Delta + \Delta^T\dot{Y})/2. \tag{2.16}$$

We are unaware of any closed form solution to this system of differential equations along geodesics.

By differentiation, we see that parallel transported vectors preserve the inner product. In particular, the square length of Δ ($\operatorname{tr}\Delta^T\Delta$) is preserved. Additionally, inserting $\dot{Y}$ into the parallel transport equation, one quickly sees that a geodesic always parallel transports its own tangent vector. This condition may be taken as the definition of a geodesic.

Observing that $\operatorname{tr}\Delta^T\Delta$ is the sum of the squares of the singular values of Δ, we conjectured that the individual singular values of Δ might also be preserved by parallel transport. Numerical experiments show that this is not the case.

In the case of the orthogonal group ($p = n$), however, parallel translation of Δ along the geodesic $Q(t) = Q(0)e^{At}$ is straightforward. Let $\Delta(t) = Q(t)B(t)$ be the solution of the parallel translation equation

$$\dot{\Delta} = -Q(\dot{Q}^T\Delta + \Delta^T\dot{Q})/2,$$

where $B(t)$ is a skew-symmetric matrix. Substituting $\dot{\Delta} = \dot{Q}B + Q\dot{B}$ and $\dot{Q} = QA$, we obtain

$$\dot{B} = -\frac{1}{2}[A, B], \tag{2.17}$$

whose solution is $B(t) = e^{-At/2}B(0)e^{At/2}$; therefore,

$$\Delta(t) = Q(0)e^{At/2}B(0)e^{At/2}. \tag{2.18}$$

These formulas may be generalized to arbitrary connected Lie groups [47, Chap. 2, Ex. A.6].

So as to arrive at the general notion of parallel transport, let us formalize what we did here. We saw that the geodesic equation may be written

$$\ddot{Y} + \Gamma_e(\dot{Y}, \dot{Y}) = 0,$$

where in the Euclidean case

$$\Gamma_e(\Delta_1, \Delta_2) = Y(\Delta_1^T \Delta_2 + \Delta_2^T \Delta_1)/2.$$

Anticipating the generalization, we interpret Γ as containing the information of the normal component that needs to be removed. Knowing the quadratic function $\Gamma(\Delta, \Delta)$ is sufficient for obtaining the bilinear function $\Gamma(\Delta_1, \Delta_2)$; the process is called polarization. We assume that Γ is a symmetric function of its arguments (this is the so-called torsion-free condition), and we obtain

$$4\Gamma(\Delta_1, \Delta_2) = \Gamma(\Delta_1 + \Delta_2, \Delta_1 + \Delta_2) - \Gamma(\Delta_1 - \Delta_2, \Delta_1 - \Delta_2).$$

For the cases we study in this paper, it is easy in practice to guess a symmetric form for $\Gamma(\Delta_1, \Delta_2)$ given $\Gamma(\Delta, \Delta)$.

We will give a specific example of why this formalism is needed in section 2.4. Let us mention here that the parallel transport defined in this manner is known to differential geometers as the Levi–Civita connection. We also remark that the function Γ when written in terms of components defines the Christoffel symbols. Switching to vector notation, in differential geometry texts the ith component of the function $\Gamma(v, w)$ would normally be written as $\sum_{jk} \Gamma^i_{jk} v_j w_k$, where the constants Γ^i_{jk} are called Christoffel symbols. We prefer the matrix notation over the scalar notation.

2.3. Geometry of quotient spaces.

Given a manifold whose geometry is well understood (where there are closed form expressions for the geodesics and, perhaps also, parallel transport), there is a very natural, efficient, and convenient way to generate closed form formulas on quotient spaces of that manifold. This is precisely the situation with the Stiefel and Grassmann manifolds, which are quotient spaces within the orthogonal group. As just seen in the previous section, geodesics and parallel translation on the orthogonal group are simple. We now show how the Stiefel and Grassmann manifolds inherit this simple geometry.

2.3.1. The quotient geometry of the Stiefel manifold.

The important ideas here are the notions of the horizontal and vertical spaces, the metric, and their relationship to geodesics and parallel translation. We use brackets to denote equivalence classes. We will define these concepts using the Stiefel manifold $V_{n,p} = O_n/O_{n-p}$ as an example. The equivalence class $[Q]$ is the set of all n-by-n orthogonal matrices with the same first p columns as Q. A point in the Stiefel manifold is the equivalence class

$$[Q] = \left\{ Q \begin{pmatrix} I_p & 0 \\ 0 & Q_{n-p} \end{pmatrix} : Q_{n-p} \in O_{n-p} \right\}; \tag{2.19}$$

that is, a point in the Stiefel manifold is a particular subset of the orthogonal matrices. Notice that in this section we are working with equivalence classes rather than n-by-p matrices $Y = QI_{n,p}$.

The vertical and horizontal spaces at a point Q are complementary linear subspaces of the tangent space at Q. The vertical space is defined to be vectors tangent to the set $[Q]$. The horizontal space is defined as the tangent vectors at Q orthogonal to the vertical space. At a point Q, the vertical space is the set of vectors of the form

$$\Phi = Q\begin{pmatrix} 0 & 0 \\ 0 & C \end{pmatrix}, \tag{2.20}$$

where C is $(n-p)$-by-$(n-p)$ skew-symmetric, and we have hidden postmultiplication by the isotropy subgroup $\left(\begin{smallmatrix} I_p & \\ & O_{n-p} \end{smallmatrix}\right)$. Such vectors are clearly tangent to the set $[Q]$ defined in (2.19). It follows that the horizontal space at Q is the set of tangents of the form

$$\Delta = Q\begin{pmatrix} A & -B^T \\ B & 0 \end{pmatrix} \tag{2.21}$$

(also hiding the isotropy subgroup), where A is p-by-p skew-symmetric. Vectors of this form are clearly orthogonal to vertical vectors with respect to the Euclidean inner product. The matrices A and B of (2.21) are equivalent to those of (2.5).

The significance of the horizontal space is that it provides a representation of tangents to the quotient space. Intuitively, movements in the vertical direction make no change in the quotient space. Therefore, the metric, geodesics, and parallel translation must all be restricted to the horizontal space. A rigorous treatment of these intuitive concepts is given by Kobayashi and Nomizu [54] and Chavel [18].

The canonical metric on the Stiefel manifold is then simply the restriction of the orthogonal group metric to the horizontal space (multiplied by 1/2 to avoid factors of 2 later on). That is, for Δ_1 and Δ_2 of the form in (2.21),

$$\begin{aligned} g_c(\Delta_1, \Delta_2) &= \frac{1}{2}\operatorname{tr}\left(Q\begin{pmatrix} A_1 & -B_1^T \\ B_1 & 0 \end{pmatrix}\right)^T Q\begin{pmatrix} A_2 & -B_2^T \\ B_2 & 0 \end{pmatrix} \\ &= \tfrac{1}{2}\operatorname{tr} A_1^T A_2 + \operatorname{tr} B_1^T B_2, \end{aligned} \tag{2.22}$$

which we shall also write as $\langle \Delta_1, \Delta_2 \rangle$. It is important to realize that this is *not* equal to the Euclidean metric g_e defined in section 2.2 (except for $p = 1$ or n), even though we use the Euclidean metric for the orthogonal group in its definition. The difference arises because the Euclidean metric counts the independent coordinates of the skew-symmetric A matrix twice and those of B only once, whereas the canonical metric counts all independent coordinates in A and B equally. This point is discussed in detail in section 2.4.

Notice that the orthogonal group geodesic

$$Q(t) = Q(0)\exp t\begin{pmatrix} A & -B^T \\ B & 0 \end{pmatrix} \tag{2.23}$$

has horizontal tangent

$$\dot{Q}(t) = Q(t)\begin{pmatrix} A & -B^T \\ B & 0 \end{pmatrix} \tag{2.24}$$

at every point along the curve $Q(t)$. Therefore, they are curves of shortest length in the quotient space as well, i.e., geodesics in the Grassmann manifold are given by the simple formula

$$\text{Stiefel geodesics} = [Q(t)], \tag{2.25}$$

where $[Q(t)]$ is given by (2.19) and (2.23). This formula will be essential for deriving an expression for geodesics on the Stiefel manifold using n-by-p matrices in section 2.4.

In a quotient space, parallel translation works in a way similar to the embedded parallel translation discussed in section 2.2.3. Parallel translation along a curve (with everywhere horizontal tangent) is accomplished by infinitesimally removing the vertical component of the tangent vector. The equation for parallel translation along the geodesics in the Stiefel manifold is obtained by applying this idea to (2.17), which provides translation along geodesics for the orthogonal group. Let

$$\mathfrak{A} = \begin{pmatrix} A_1 & -B_1^T \\ B_1 & 0 \end{pmatrix} \quad \text{and} \quad \mathfrak{B} = \begin{pmatrix} A_2 & -B_2^T \\ B_2 & 0 \end{pmatrix} \tag{2.26}$$

be two horizontal vectors t $Q = I$. The parallel translation of $\mathfrak{B}$ along the geodesic $e^{\mathfrak{A}t}$ is given by the differential equation

$$\dot{\mathfrak{B}} = -\frac{1}{2}[\mathfrak{A}, \mathfrak{B}]_H, \tag{2.27}$$

where the subscript H denotes the horizontal component (lower right block set to zero). Note that the Lie bracket of two horizontal vectors is not horizontal and that the solution to (2.27) is not given by the formula $(e^{-\mathfrak{A}t/2}\mathfrak{B}(0)e^{\mathfrak{A}t/2})_H$. This is a special case of the general formula for reductive homogeneous spaces [18, 75]. This first order linear differential equation with constant coefficients is integrable in closed form, but it is an open question whether this can be accomplished with $O(np^2)$ operations.

2.3.2. The quotient geometry of the Grassmann manifold. We quickly repeat this approach for the Grassmann manifold $G_{n,p} = O_n/(O_p \times O_{n-p})$. The equivalence class $[Q]$ is the set of all orthogonal matrices whose first p columns span the same subspace as those of Q. A point in the Grassmann manifold is the equivalence class

$$[Q] = \left\{ Q \begin{pmatrix} Q_p & 0 \\ 0 & Q_{n-p} \end{pmatrix} : Q_p \in O_p,\ Q_{n-p} \in O_{n-p} \right\}, \tag{2.28}$$

i.e., a point in the Grassmann manifold is a particular subset of the orthogonal matrices, and the Grassmann manifold itself is the collection of all these subsets.

The vertical space at a point Q is the set of vectors of the form

$$\Phi = Q \begin{pmatrix} A & 0 \\ 0 & C \end{pmatrix}, \tag{2.29}$$

where A is p-by-p skew-symmetric and C is $(n-p)$-by-$(n-p)$ skew-symmetric. The horizontal space at Q is the set of matrices of the form

$$\Delta = Q \begin{pmatrix} 0 & -B^T \\ B & 0 \end{pmatrix}. \tag{2.30}$$

Note that we have hidden postmultiplication by the isotropy subgroup $\left(\begin{smallmatrix} O_p & \\ & O_{n-p} \end{smallmatrix}\right)$ in (2.29) and (2.30).

The canonical metric on the Grassmann manifold is the restriction of the orthogonal group metric to the horizontal space (multiplied by 1/2). Let Δ_1 and Δ_2 be of the form in (2.30). Then

$$g_c(\Delta_1, \Delta_2) = \operatorname{tr} B_1^T B_2. \tag{2.31}$$

As opposed to the canonical metric for the Stiefel manifold, this metric is in fact equivalent to the Euclidean metric (up to multiplication by 1/2) defined in (2.2).

The orthogonal group geodesic

$$Q(t) = Q(0) \exp t \begin{pmatrix} 0 & -B^T \\ B & 0 \end{pmatrix} \tag{2.32}$$

has horizontal tangent

$$\dot{Q}(t) = Q(t) \begin{pmatrix} 0 & -B^T \\ B & 0 \end{pmatrix} \tag{2.33}$$

at every point along the curve $Q(t)$; therefore,

$$\text{Grassmann geodesics} = [Q(t)], \tag{2.34}$$

where $[Q(t)]$ is given by (2.28) and (2.32). This formula gives us an easy method for computing geodesics on the Grassmann manifold using n-by-p matrices, as will be seen in section 2.5.

The method for parallel translation along geodesics in the Grassmann manifold is the same as for the Stiefel manifold, although it turns out the Grassmann manifold has additional structure that makes this task easier. Let

$$\mathfrak{A} = \begin{pmatrix} 0 & -A^T \\ A & 0 \end{pmatrix} \quad \text{and} \quad \mathfrak{B} = \begin{pmatrix} 0 & -B^T \\ B & 0 \end{pmatrix} \tag{2.35}$$

be two horizontal vectors at $Q = I$. It is easily verified that $[\mathfrak{A}, \mathfrak{B}]$ is in fact a vertical vector of the form of (2.29). If the vertical component of (2.17) is infinitesimally removed, we are left with the trivial differential equation

$$\dot{\mathfrak{B}} = 0. \tag{2.36}$$

Therefore, the parallel translation of the tangent vector $Q(0)\mathfrak{B}$ along the geodesic $Q(t) = Q(0)e^{\mathfrak{A}t}$ is simply given by the expression

$$\tau\mathfrak{B}(t) = Q(0)e^{\mathfrak{A}t}\mathfrak{B}, \tag{2.37}$$

which is of course horizontal at $Q(t)$. Here, we introduce the notation τ to indicate the transport of a vector; it is not a scalar multiple of the vector. It will be seen in section 2.5 how this formula may be computed using $O(np^2)$ operations.

As an aside, if H and V represent the horizontal and vertical spaces, respectively, it may be verified that

$$[V, V] \subset V, \quad [V, H] \subset H, \quad [H, H] \subset V. \tag{2.38}$$

The first relationship follows from the fact that V is a Lie algebra, the second follows from the reductive homogeneous space structure [54] of the Grassmann manifold, also possessed by the Stiefel manifold, and the third follows the symmetric space structure [47, 54] of the Grassmann manifold, which the Stiefel manifold does not possess.

2.4. The Stiefel manifold with its canonical metric.

2.4.1. The canonical metric (Stiefel). The Euclidean metric

$$g_e(\Delta, \Delta) = \operatorname{tr} \Delta^T \Delta$$

used in section 2.2 may seem natural, but one reasonable objection to its use is that it weighs the independent degrees of freedom of the tangent vector unequally. Using the representation of tangent vectors $\Delta = YA + Y_\perp B$ given in (2.5), it is seen that

$$\begin{aligned} g_e(\Delta, \Delta) &= \operatorname{tr} A^T A + \operatorname{tr} B^T B \\ &= 2 \sum_{i<j} a_{ij}^2 + \sum_{ij} b_{ij}^2. \end{aligned}$$

The Euclidean metric counts the $p(p+1)/2$ independent coordinates of A twice. At the origin $I_{n,p}$, a more equitable metric would be $g_c(\Delta, \Delta) = \operatorname{tr} \Delta^T (I - \frac{1}{2} I_{n,p} I_{n,p}^T)\Delta = \frac{1}{2} \operatorname{tr} A^T A + \operatorname{tr} B^T B$. To be equitable at all points in the manifold, the metric must vary with Y according to

$$g_c(\Delta, \Delta) = \operatorname{tr} \Delta^T (I - \tfrac{1}{2} Y Y^T)\Delta. \tag{2.39}$$

This is called the *canonical metric* on the Stiefel manifold. This is precisely the metric derived from the quotient space structure of $V_{n,p}$ in (2.22); therefore, the formulas for geodesics and parallel translation for the Stiefel manifold given in section 2.3.1 are correct if we view the Stiefel manifold as the set of orthonormal n-by-p matrices with the metric of (2.39). Note that if $\Delta = YA + Y_\perp B$ is a tangent vector, then $g_c(\Delta, \Delta) = \frac{1}{2} \operatorname{tr} A^T A + \operatorname{tr} B^T B$, as seen previously.

2.4.2. Geodesics (Stiefel). The path length

$$L = \int g_c(\dot{Y}, \dot{Y})^{1/2}\, dt \tag{2.40}$$

may be minimized with the calculus of variations. Doing so is tedious but yields the new geodesic equation

$$\ddot{Y} + \dot{Y}\dot{Y}^T Y + Y\big((Y^T \dot{Y})^2 + \dot{Y}^T \dot{Y}\big) = 0. \tag{2.41}$$

Direct substitution into (2.41) using the fact that

$$(I - I_{n,p} I_{n,p}^T) X (I - I_{n,p} I_{n,p}^T) = 0,$$

if X is a skew-symmetric matrix of the form

$$X = \begin{pmatrix} A & -B^T \\ B & 0 \end{pmatrix},$$

verifies that the paths of the form

$$Y(t) = Q e^{Xt} I_{n,p} \tag{2.42}$$

are closed form solutions to the geodesic equation for the canonical metric.

We now turn to the problem of computing geodesics with algorithms of complexity $O(np^2)$. Our current formula $Y(t) = Q \exp t \left(\begin{smallmatrix} A & -B^T \\ B & 0 \end{smallmatrix}\right) I_{n,p}$ for a geodesic is not useful. Rather we want to express the geodesic $Y(t)$ in terms of the current position $Y(0) = Y$

and a direction $\dot{Y}(0) = H$. For example, $A = Y^T H$ and we have $C := B^T B = H^T(I - YY^T)H$. In fact the geodesic only depends on $B^T B$ rather than B itself. The trick is to find a differential equation for $M(t) = I_{n,p}^T \exp t\left(\begin{smallmatrix} A & -B^T \\ B & 0 \end{smallmatrix}\right) I_{n,p}$.

The following theorem makes clear that the computational difficulty inherent in computing the geodesic is the solution of a constant coefficient second order differential equation for $M(t)$. The answer is obtained not by a differential equation solver but rather by solving the corresponding quadratic eigenvalue problem.

THEOREM 2.1. *If $Y(t) = Qe^{t\left(\begin{smallmatrix} A & -B^T \\ B & 0 \end{smallmatrix}\right)} I_{n,p}$, with $Y(0) = Y$ and $\dot{Y}(0) = H$, then*

$$Y(t) = YM(t) + (I - YY^T)H \int_0^t M(t)\,dt, \tag{2.43}$$

where $M(t)$ is the solution to the second order differential equation with constant coefficients

$$\ddot{M} - A\dot{M} + CM = 0; \qquad M(0) = I_p, \quad \dot{M}(0) = A, \tag{2.44}$$

$A = Y^T H$ is skew-symmetric, and $C = H^T(I - YY^T)H$ is nonnegative definite.

Proof. A direct computation verifies that $M(t)$ satisfies (2.44). By separately considering $Y^T Y(t)$ and $(I - YY^T)Y(t)$, we may derive (2.43). □

The solution of the differential equation (2.44) may be obtained [25, 88] by solving the quadratic eigenvalue problem

$$(\lambda^2 I - A\lambda + C)x = 0.$$

Such problems are typically solved in one of three ways: (1) by solving the generalized eigenvalue problem

$$\begin{pmatrix} C & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} x \\ \lambda x \end{pmatrix} = \lambda \begin{pmatrix} A & -I \\ I & 0 \end{pmatrix} \begin{pmatrix} x \\ \lambda x \end{pmatrix},$$

(2) by solving the eigenvalue problem

$$\begin{pmatrix} 0 & I \\ -C & A \end{pmatrix} \begin{pmatrix} x \\ \lambda x \end{pmatrix} = \lambda \begin{pmatrix} x \\ \lambda x \end{pmatrix},$$

or (3) any equivalent problem obtained by factoring $C = K^T K$ and then solving the eigenvalue problem

$$\begin{pmatrix} A & -K^T \\ K & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \lambda \begin{pmatrix} x \\ y \end{pmatrix}.$$

Problems of this form arise frequently in mechanics, usually with A symmetric. Some discussion of physical interpretations for skew-symmetric matrices may be found in the context of rotating machinery [21]. If X is the p-by-$2p$ matrix of eigenvectors and Λ denotes the eigenvalues, then $M(t) = Xe^{\Lambda t}Z$, and its integral is $\int M(t)\,dt = Xe^{\Lambda t}\Lambda^{-1}Z$, where Z is chosen so that $XZ = I$ and $X\Lambda Z = A$.

Alternatively, the third method along with the matrix exponential may be employed.

COROLLARY 2.2. *Let Y and H be n-by-p matrices such that $Y^T Y = I_p$ and $A = Y^T H$ is skew-symmetric. Then the geodesic on the Stiefel manifold emanating from Y in direction H is given by the curve*

$$Y(t) = YM(t) + QN(t), \tag{2.45}$$

where

$$QR := K = (I - YY^T)H \tag{2.46}$$

is the compact QR-decomposition of K (Q n-by-p, R p-by-p) and $M(t)$ and $N(t)$ are p-by-p matrices given by the matrix exponential

$$\begin{pmatrix} M(t) \\ N(t) \end{pmatrix} = \exp t \begin{pmatrix} A & -R^T \\ R & 0 \end{pmatrix} \begin{pmatrix} I_p \\ 0 \end{pmatrix}. \tag{2.47}$$

Note that (2.47) is easily computed by solving a $2p$-by-$2p$ skew-symmetric eigenvalue problem, which can be accomplished efficiently using the SVD or algorithms specially tailored for this problem [86].

2.4.3. Parallel translation (Stiefel). We now develop a notion of parallel transport that is consistent with the canonical metric. The geodesic equation takes the form $\ddot{Y} + \Gamma(\dot{Y}, \dot{Y}) = 0$, where, from (2.41), it is seen that the Christoffel function for the canonical metric is

$$\Gamma_c(\Delta, \Delta) = \Delta\Delta^T Y + Y\Delta^T(I - YY^T)\Delta. \tag{2.48}$$

By polarizing we obtain the result

$$\begin{aligned} \Gamma_c(\Delta_1, \Delta_2) = \tfrac{1}{2}(\Delta_1\Delta_2^T + \Delta_2\Delta_1^T)Y + \tfrac{1}{2}Y\big(\Delta_2^T(I - YY^T)\Delta_1 \\ +\Delta_1^T(I - YY^T)\Delta_2\big). \end{aligned} \tag{2.49}$$

Parallel transport is given by the differential equation

$$\dot{\Delta} + \Gamma_c(\Delta, \dot{Y}) = 0, \tag{2.50}$$

which is equivalent to (2.27). As stated after this equation, we do not have an $O(np^2)$ method to compute $\Delta(t)$.

2.4.4. The gradient of a function (Stiefel). Both conjugate gradient and Newton's method require a computation of the gradient of a function, which depends upon the choice of metric. For a function $F(Y)$ defined on the Stiefel manifold, the gradient of F at Y is defined to be the tangent vector ∇F such that

$$\operatorname{tr} F_Y^T\Delta = g_c(\nabla F, \Delta) \equiv \operatorname{tr}(\nabla F)^T(I - \tfrac{1}{2}YY^T)\Delta \tag{2.51}$$

for all tangent vectors Δ at Y, where F_Y is the n-by-p matrix of partial derivatives of F with respect to the elements of Y, i.e.,

$$(F_Y)_{ij} = \frac{\partial F}{\partial Y_{ij}}. \tag{2.52}$$

Solving (2.51) for ∇F such that $Y^T(\nabla F) =$ skew-symmetric yields

$$\nabla F = F_Y - YF_Y^TY. \tag{2.53}$$

Equation (2.53) may also be derived by differentiating $F(Y(t))$, where $Y(t)$ is the Stiefel geodesic given by (2.45).

2.4.5. The Hessian of a function (Stiefel). Newton's method requires the Hessian of a function, which depends upon the choice of metric. The Hessian of a function $F(Y)$ defined on the Stiefel manifold is defined as the quadratic form

$$\operatorname{Hess} F(\Delta, \Delta) = \left.\frac{d^2}{dt^2}\right|_{t=0} F\bigl(Y(t)\bigr), \tag{2.54}$$

where $Y(t)$ is a geodesic with tangent Δ, i.e., $\dot{Y}(0) = \Delta$. Applying this definition to $F(Y)$ and (2.45) yields the formula

$$\begin{aligned}\operatorname{Hess} F(\Delta_1, \Delta_2) = F_{YY}(\Delta_1, \Delta_2) &+ \tfrac{1}{2}\operatorname{tr}\bigl((F_Y^T\Delta_1 Y^T + Y^T\Delta_1 F_Y^T)\Delta_2\bigr)\\ &-\tfrac{1}{2}\operatorname{tr}\bigl((Y^T F_Y + F_Y^T Y)\Delta_1^T \Pi \Delta_2\bigr),\end{aligned} \tag{2.55}$$

where $\Pi = I - YY^T$, F_Y is defined in (2.52), and the notation $F_{YY}(\Delta_1, \Delta_2)$ denotes the scalar $\sum_{ij,\,kl}(F_{YY})_{ij,\,kl}(\Delta_1)_{ij}(\Delta_2)_{kl}$, where

$$(F_{YY})_{ij,\,kl} = \frac{\partial^2 F}{\partial Y_{ij}\partial Y_{kl}}. \tag{2.56}$$

This formula may also readily be obtained by using (2.50) and the formula

$$\operatorname{Hess} F(\Delta_1, \Delta_2) = F_{YY}(\Delta_1, \Delta_2) - \operatorname{tr} F_Y^T \Gamma_c(\Delta_1, \Delta_2). \tag{2.57}$$

For Newton's method, we must determine the tangent vector Δ such that

$$\operatorname{Hess} F(\Delta, X) = \langle -G, X\rangle \quad \text{for all tangent vectors } X, \tag{2.58}$$

where $G = \nabla F$. Recall that $\langle\, ,\rangle \equiv g_c(\, ,)$ in this context. We shall express the solution to this linear equation as $\Delta = -\operatorname{Hess}^{-1} G$, which may be expressed as the solution to the linear problem

$$F_{YY}(\Delta) - Y\operatorname{skew}(F_Y^T\Delta) - \operatorname{skew}(\Delta F_Y^T)Y - \frac{1}{2}\Pi\Delta Y^T F_Y = -G, \tag{2.59}$$

$Y^T\Delta =$ skew-symmetric, where $\operatorname{skew}(X) = (X - X^T)/2$ and the notation $F_{YY}(\Delta)$ means the unique tangent vector satisfying the equation

$$F_{YY}(\Delta, X) = \langle F_{YY}(\Delta), X\rangle \quad \text{for all tangent vectors } X. \tag{2.60}$$

Example problems are considered in section 3.

2.5. The Grassmann manifold with its canonical metric. A quotient space representation of the Grassmann manifold was given in section 2.3.2; however, for computations we prefer to work with n-by-p orthonormal matrices Y. When performing computations on the Grassmann manifold, we will use the n-by-p matrix Y to represent an entire equivalence class

$$[Y] = \{YQ_p : Q_p \in O_p\}, \tag{2.61}$$

i.e., the subspace spanned by the columns of Y. Any representative of the equivalence class will do.

We remark that an alternative strategy is to represent points on the Grassmann manifold with projection matrices YY^T. There is one such unique matrix corresponding to each point on the Grassmann manifold. On first thought it may seem foolish to use n^2 parameters to represent a point on the Grassmann manifold (which has dimension $p(n-p)$), but in certain ab initio physics computations [43], the projection matrices YY^T that arise in practice tend to require only $O(n)$ parameters for their representation.

Returning to the n-by-p representation of points on the Grassmann manifold, the tangent space is easily computed by viewing the Grassmann manifold as the quotient space $G_{n,\,p} = V_{n,\,p}/O_p$. At a point Y on the Stiefel manifold then, as seen in (2.5), tangent vectors take the form $\Delta = YA + Y_\perp B$, where A is p-by-p skew-symmetric, B is $(n-p)$-by-p, and $Y_\perp$ is any n-by-$(n-p)$ matrix such that $(Y,\ Y_\perp)$ is orthogonal. From (2.61) it is clear that the vertical space at Y is the set of vectors of the form

$$\Phi = YA; \tag{2.62}$$

therefore, the horizontal space at Y is the set of vectors of the form

$$\Delta = Y_\perp B. \tag{2.63}$$

Because the horizontal space is equivalent to the tangent space of the quotient, the tangent space of the Grassmann manifold at $[Y]$ is given by all n-by-p matrices Δ of the form in (2.63) or, equivalently, all n-by-p matrices Δ such that

$$Y^T\Delta = 0. \tag{2.64}$$

Physically, this corresponds to directions free of rotations mixing the basis given by the columns of Y.

We already saw in section 2.3.2 that the Euclidean metric is in fact equivalent to the canonical metric for the Grassmann manifold. That is, for n-by-p matrices Δ_1 and Δ_2 such that $Y^T\Delta_i = 0$ $(i = 1,\, 2)$,

$$\begin{aligned} g_c(\Delta_1, \Delta_2) &= \operatorname{tr} \Delta_1^T(I - \tfrac{1}{2}YY^T)\Delta_2, \\ &= \operatorname{tr} \Delta_1^T\Delta_2, \\ &= g_e(\Delta_1, \Delta_2). \end{aligned}$$

2.5.1. Geodesics (Grassmann). A formula for geodesics on the Grassmann manifold was given via (2.32); the following theorem provides a useful method for computing this formula using n-by-p matrices.

THEOREM 2.3. *If* $Y(t) = Qe^{t\left(\begin{smallmatrix}0 & -B^T\\ B & 0\end{smallmatrix}\right)}I_{n,\,p}$, *with* $Y(0) = Y$ *and* $\dot{Y}(0) = H$, *then*

$$Y(t) = \begin{pmatrix} YV & U \end{pmatrix} \begin{pmatrix} \cos \Sigma t \\ \sin \Sigma t \end{pmatrix} V^T, \tag{2.65}$$

where $U\Sigma V^T$ *is the compact singular value decomposition of* H.

Proof 1. It is easy to check that either formulation for the geodesic satisfies the geodesic equation $\ddot{Y} + Y(\dot{Y}^T\dot{Y}) = 0$, with the same initial conditions. □

Proof 2. Let $B = (U_1,\ U_2)\binom{\Sigma}{0}V^T$ be the singular value decomposition of B (U_1 n-by-p, U_2 p-by-$(n-p)$, Σ and V p-by-p). A straightforward computation involving the partitioned matrix

$$\begin{pmatrix} 0 & -B^T \\ B & 0 \end{pmatrix} = \begin{pmatrix} V & 0 & 0 \\ 0 & U_1 & U_2 \end{pmatrix} \begin{pmatrix} 0 & -\Sigma & 0 \\ \Sigma & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} V^T & 0 \\ 0 & U_1^T \\ 0 & U_2^T \end{pmatrix} \tag{2.66}$$

verifies the theorem. □

A subtle point in (2.65) is that if the rightmost V^T is omitted, then we still have a representative of the same equivalence class as $Y(t)$; however, due to consistency conditions along the equivalent class $[Y(t)]$, the tangent (horizontal) vectors that we use for computations must be altered in the same way. This amounts to postmultiplying everything by V, or, for that matter, any p-by-p orthogonal matrix.

The path length between Y_0 and $Y(t)$ (distance between subspaces) is given by [89]

$$d\big(Y(t), Y_0\big) = t\|H\|_F = t\left(\sum_{i=1}^{p} \sigma_i^2\right)^{1/2}, \tag{2.67}$$

where σ_i are the diagonal elements of Σ. (Actually, this is only true for t small enough to avoid the issue of conjugate points, e.g., long great circle routes on the sphere.) An interpretation of this formula in terms of the CS decomposition and principal angles between subspaces is given in section 4.3.

2.5.2. Parallel translation (Grassmann). A formula for parallel translation along geodesics of complexity $O(np^2)$ can also be derived as follows.

THEOREM 2.4. *Let H and Δ be tangent vectors to the Grassmann manifold at Y. Then the parallel translation of Δ along the geodesic in the direction $\dot{Y}(0) = H$ (see (2.65)) is*

$$\tau\Delta(t) = \left((\,YV \quad U\,) \begin{pmatrix} -\sin \Sigma t \\ \cos \Sigma t \end{pmatrix} U^T + (I - UU^T) \right) \Delta. \tag{2.68}$$

Proof 1. A simple computation verifies that (2.68) and (2.65) satisfy (2.16). □

Proof 2. Parallel translation of Δ is given by the expression

$$\tau\Delta(t) = Q \exp t \begin{pmatrix} 0 & -A^T \\ A & 0 \end{pmatrix} \begin{pmatrix} 0 \\ B \end{pmatrix}$$

(which follows from (2.37)), where $Q = (Y,\ Y_\perp)$, $H = Y_\perp A$, and $\Delta = Y_\perp B$. Decomposing $\left(\begin{smallmatrix} 0 & -A^T \\ A & 0 \end{smallmatrix}\right)$ as in (2.66) (note well that A has replaced B), a straightforward computation verifies the theorem. □

2.5.3. The gradient of a function (Grassmann). We must compute the gradient of a function $F(Y)$ defined on the Grassmann manifold. Similarly to section 2.4.4, the gradient of F at $[Y]$ is defined to be the tangent vector ∇F such that

$$\operatorname{tr} F_Y^T \Delta = g_c(\nabla F, \Delta) \equiv \operatorname{tr}(\nabla F)^T \Delta \tag{2.69}$$

for all tangent vectors Δ at Y, where F_Y is defined by (2.52). Solving (2.69) for ∇F such that $Y^T(\nabla F) = 0$ yields

$$\nabla F = F_Y - YY^T F_Y. \tag{2.70}$$

Equation (2.70) may also be derived by differentiating $F(Y(t))$, where $Y(t)$ is the Grassmann geodesic given by (2.65).

2.5.4. The Hessian of a function (Grassmann). Applying the definition for the Hessian of $F(Y)$ given by (2.54) in the context of the Grassmann manifold yields the formula

$$\operatorname{Hess} F(\Delta_1, \Delta_2) = F_{YY}(\Delta_1, \Delta_2) - \operatorname{tr}(\Delta_1^T \Delta_2 Y^T F_Y), \tag{2.71}$$

where F_Y and F_{YY} are defined in section 2.4.5. For Newton's method, we must determine $\Delta = -\operatorname{Hess}^{-1} G$ satisfying (2.58), which for the Grassmann manifold is expressed as the linear problem

$$F_{YY}(\Delta) - \Delta(Y^T F_Y) = -G, \tag{2.72}$$

$Y^T\Delta = 0$, where $F_{YY}(\Delta)$ denotes the unique tangent vector satisfying (2.60) for the Grassmann manifold's canonical metric.

Example problems are considered in section 3.

2.6. Conjugate gradient on Riemannian manifolds. As demonstrated by Smith [75, 76], the benefits of using the conjugate gradient algorithm for unconstrained minimization can be carried over to minimization problems constrained to Riemannian manifolds by a covariant translation of the familiar operations of computing gradients, performing line searches, the computation of Hessians, and carrying vector information from step to step in the minimization process. In this section we will review the ideas in [75, 76], and then in the next section we formulate concrete algorithms for conjugate gradient on the Stiefel and Grassmann manifolds. Here one can see how the geometry provides insight into the true difference among the various formulas that are used in linear and nonlinear conjugate gradient algorithms.

Figure 2.3 sketches the conjugate gradient algorithm in flat space and Figure 2.4 illustrates the algorithm on a curved space. An outline for the iterative part of the algorithm (in either flat or curved space) goes as follows: at the $(k-1)$st iterate x_{k-1}, step to x_k, the minimum of f along the geodesic in the direction H_{k-1}, compute the gradient $G_k = \nabla f(x_k)$ at this point, choose the new search direction to be a combination of the old search direction and the new gradient

$$H_k = G_k + \gamma_k \tau H_{k-1}, \tag{2.73}$$

and iterate until convergence. Note that τH_{k-1} in (2.73) is the parallel translation of the vector H_{k-1} defined in section 2.2.3, which in this case is simply the direction of the geodesic (line) at the point x_k (see Figure 2.4). Also note the important condition that x_k is a minimum point along the geodesic

$$\langle G_k, \tau H_{k-1} \rangle = 0. \tag{2.74}$$

Let us begin our examination of the choice of γ_k in flat space before proceeding to arbitrary manifolds. Here, parallel transport is trivial so that

$$H_k = G_k + \gamma_k H_{k-1}.$$

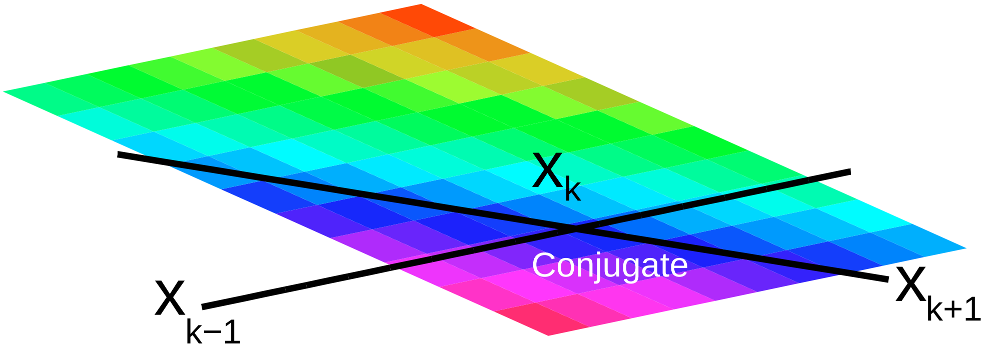


Fig. 2.3. *Conjugate gradient in flat space.*

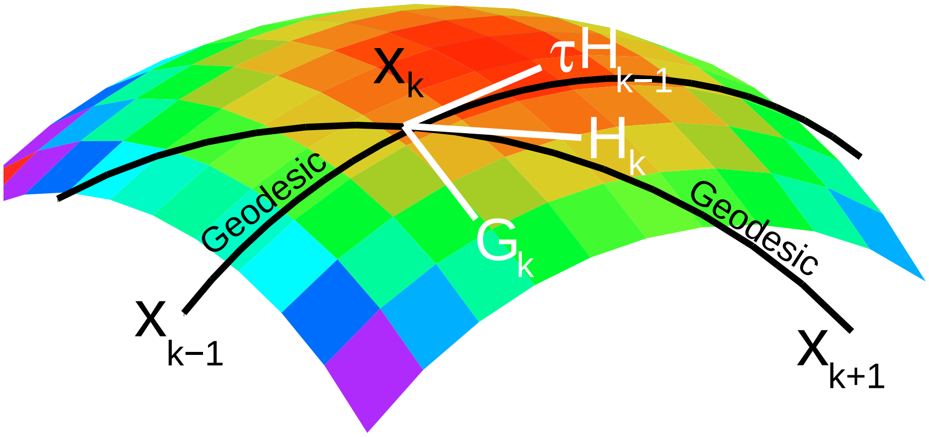


Fig. 2.4. *Conjugate gradient in curved space.*

In both linear and an idealized version of nonlinear conjugate gradient, γ_k may be determined by the exact conjugacy condition for the new search direction

$$f_{xx}(H_k, H_{k-1}) = 0,$$

i.e., the old and new search direction must be conjugate with respect to the Hessian of f. (With $f_{xx} = A$, the common notation [45, p. 523] for the conjugacy condition is $p_{k-1}^T A p_k = 0$.) The formula for γ_k is then

$$\text{(2.75)} \qquad \text{Exact Conjugacy:} \quad \gamma_k = -f_{xx}(G_k, H_{k-1})/f_{xx}(H_{k-1}, H_{k-1}).$$

The standard trick to improve the computational efficiency of linear conjugate gradient is to use a formula relating a finite difference of gradients to the Hessian times the direction ($r_k - r_{k-1} = -\alpha_k A p_k$ as in [45]). In our notation,

$$\langle G_k - G_{k-1}, \cdot \rangle \approx \alpha f_{xx}(\cdot, H_{k-1}), \tag{2.76}$$

where $\alpha = \|x_k - x_{k-1}\| / \|H_{k-1}\|$.

The formula is exact for linear conjugate gradient on flat space, otherwise it has the usual error in finite difference approximations. By applying the finite difference formula (2.76) in both the numerator and denominator of (2.75), and also applying (2.74) twice (once with k and once with $k-1$), one obtains the formula

$$\text{Polak–Ribière:} \quad \gamma_k = \langle G_k - G_{k-1}, G_k \rangle / \langle G_{k-1}, G_{k-1} \rangle. \tag{2.77}$$

Therefore, the Polak–Ribiére formula is the exact formula for conjugacy through the Hessian, where one uses a difference of gradients as a finite difference approximation to the second derivative. If $f(x)$ is well approximated by a quadratic function, then $\langle G_{k-1}, G_k \rangle \approx 0$, and we obtain

$$\text{Fletcher–Reeves:} \quad \gamma_k = \langle G_k, G_k \rangle / \langle G_{k-1}, G_{k-1} \rangle. \tag{2.78}$$

For arbitrary manifolds, the Hessian is the second derivative along geodesics. In differential geometry it is the second covariant differential of f. Here are the formulas

$$\text{Exact Conjugacy:} \quad \gamma_k = -\operatorname{Hess} f(G_k, \tau H_{k-1}) / \operatorname{Hess} f(\tau H_{k-1}, \tau H_{k-1}), \tag{2.79}$$

$$\text{Polak–Ribière:} \quad \gamma_k = \langle G_k - \tau G_{k-1}, G_k \rangle / \langle G_{k-1}, G_{k-1} \rangle, \tag{2.80}$$

$$\text{Fletcher–Reeves:} \quad \gamma_k = \langle G_k, G_k \rangle / \langle G_{k-1}, G_{k-1} \rangle \tag{2.81}$$

which may be derived from the finite difference approximation to the Hessian,

$$\langle G_k - \tau G_{k-1}, \cdot \rangle \approx \alpha \operatorname{Hess} f(\cdot, \tau H_{k-1}), \quad \alpha = d(x_k, x_{k-1}) / \|H_{k-1}\|.$$

Asymptotic analyses appear in section 3.6.

3. Geometric optimization algorithms.

The algorithms presented here are our answer to the question: What does it mean to perform the Newton and conjugate gradient methods on the Stiefel and Grassmann manifolds? Though these algorithms are idealized, they are of identical complexity up to small constant factors with the best known algorithms. In particular, no differential equation routines are used. It is our hope that in the geometrical algorithms presented here, the reader will recognize elements of any algorithm that accounts for orthogonality constraints. These algorithms are special cases of the Newton and conjugate gradient methods on general Riemannian manifolds. If the objective function is nondegenerate, then the algorithms are guaranteed to converge quadratically [75, 76].

3.1. Newton's method on the Grassmann manifold.

In flat space, Newton's method simply updates a vector by subtracting the gradient vector premultiplied by the inverse of the Hessian. The same is true on the Grassmann manifold (or any Riemannian manifold for that matter) of p-planes in n-dimensions with interesting modifications. Subtraction is replaced by following a geodesic path. The gradient is the usual one (which must be tangent to the constraint surface), and the Hessian is obtained by twice differentiating the function along a geodesic. We show in section 4.9 that this Hessian is related to the Hessian of the Lagrangian; the two Hessians arise from the difference between the intrinsic and extrinsic viewpoints. It may be suspected that following geodesics may not be computationally feasible, but because we exploit the structure of the constraint surface, this operation costs $O(np^2)$, which is required even for traditional algorithms for the eigenvalue problem—our simplest example.

Let $F(Y)$ be a smooth function on the Grassmann manifold, i.e., $F(Y) = F(YQ)$ for any p-by-p orthogonal matrix Q, where Y is an n-by-p matrix such that $Y^TY = I_p$. We compute formulas for F_Y and $F_{YY}(\Delta)$ using the definitions given in section 2.5.4. Newton's method for minimizing $F(Y)$ on the Grassmann manifold is as follows.

Newton's Method for Minimizing $F(Y)$ on the Grassmann Manifold

- Given Y such that $Y^TY = I_p$,
    - ○ Compute $G = F_Y - YY^TF_Y$.
    - ○ Compute $\Delta = -\operatorname{Hess}^{-1} G$ such that $Y^T\Delta = 0$ and

    $$F_{YY}(\Delta) - \Delta(Y^TF_Y) = -G.$$
- Move from Y in direction Δ to $Y(1)$ using the geodesic formula

  $$Y(t) = YV\cos(\Sigma t)V^T + U\sin(\Sigma t)V^T,$$

  where $U\Sigma V^T$ is the compact singular value decomposition of Δ (meaning U is n-by-p and both Σ and V are p-by-p).
- Repeat.

The special case of minimizing $F(Y) = \frac{1}{2}\operatorname{tr} Y^TAY$ (A n-by-n symmetric) gives the geometrically correct Newton method for the symmetric eigenvalue problem. In this case $F_Y = AY$ and $F_{YY}(\Delta) = (I - YY^T)A\Delta$. The resulting algorithm requires the solution of a Sylvester equation. It is the idealized algorithm whose approximations include various forms of Rayleigh quotient iteration, inverse iteration, a number of Newton style methods for invariant subspace computation, and the many variations of Davidson's eigenvalue method. These ideas are discussed in sections 4.1 and 4.8.

3.2. Newton's method on the Stiefel manifold. Newton's method on the Stiefel manifold is conceptually equivalent to the Grassmann manifold case. Let Y be an n-by-p matrix such that $Y^TY = I_p$, and let $F(Y)$ be a smooth function of Y without the homogeneity condition imposed for the Grassmann manifold case. Compute formulas for F_Y and $F_{YY}(\Delta)$ using the definitions given in section 2.4.5. Newton's method for minimizing $F(Y)$ on the Stiefel manifold is as follows.

Newton's Method for Minimizing $F(Y)$ on the Stiefel Manifold

- Given Y such that $Y^TY = I_p$,
  - Compute $G = F_Y - YF_Y^TY$.
  - Compute $\Delta = -\operatorname{Hess}^{-1} G$ such that $Y^T\Delta =$ skew-symmetric and
    $$F_{YY}(\Delta) - Y\,\text{skew}(F_Y^T\Delta) - \text{skew}(\Delta F_Y^T)Y - \tfrac{1}{2}\Pi\Delta Y^TF_Y = -G,$$
    where $\text{skew}(X) = (X - X^T)/2$ and $\Pi = I - YY^T$.
- Move from Y in direction Δ to $Y(1)$ using the geodesic formula
  $$Y(t) = YM(t) + QN(t),$$
  where QR is the compact QR decomposition of $(I - YY^T)\Delta$ (meaning Q is n-by-p and R is p-by-p), $A = Y^T\Delta$, and $M(t)$ and $N(t)$ are p-by-p matrices given by the $2p$-by-$2p$ matrix exponential
  $$\begin{pmatrix} M(t) \\ N(t) \end{pmatrix} = \exp t \begin{pmatrix} A & -R^T \\ R & 0 \end{pmatrix} \begin{pmatrix} I_p \\ 0 \end{pmatrix}.$$
- Repeat.

For the special case of minimizing $F(Y) = \frac{1}{2}\operatorname{tr} Y^TAYN$ (A n-by-n symmetric, N p-by-p symmetric) [75], $F_Y = AYN$ and $F_{YY}(\Delta) = A\Delta N - YN\Delta^TAY$. Note that if N is not a multiple of the identity, then $F(Y)$ does not have the homogeneity condition required for a problem on the Grassmann manifold. If $N = \text{diag}(p, p-1, \dots, 1)$, then the optimum solution to maximizing F over the Stiefel manifold yields the eigenvectors corresponding to the p largest eigenvalues.

For the orthogonal Procrustes problem [32], $F(Y) = \frac{1}{2}\|AY - B\|_F^2$ (A m-by-n, B m-by-p, both arbitrary), $F_Y = A^TAY - A^TB$ and $F_{YY}(\Delta) = A^TA\Delta - Y\Delta^TA^TAY$. Note that $Y^TF_{YY}(\Delta) =$ skew-symmetric.

3.3. Conjugate gradient method on the Grassmann manifold. Conjugate gradient techniques are considered because they are easy to implement, have low storage requirements, and provide superlinear convergence in the limit. The Newton equations may be solved with finitely many steps of linear conjugate gradient; each nonlinear conjugate gradient step, then, approximates a Newton step. In flat space, the nonlinear conjugate gradient method performs a line search by following a direction determined by conjugacy with respect to the Hessian. On Riemannian manifolds, conjugate gradient performs minimization along geodesics with search directions defined using the Hessian described above [75, 76]. Both algorithms approximate Hessian conjugacy with a subtle formula involving only the gradient directions, resulting in an algorithm that captures second derivative information by computing only first derivatives. To "communicate" information from one iteration to the next, tangent vectors must parallel transport along geodesics. Conceptually, this is necessary because, unlike flat space, the definition of tangent vectors changes from point to point.

Using these ideas and formulas developed in section 3.1, the conjugate gradient method on the Grassmann manifold is as follows.

Conjugate Gradient for Minimizing $F(Y)$ on the Grassmann Manifold

- Given Y_0 such that $Y_0^TY_0 = I$, compute $G_0 = F_{Y_0} - Y_0Y_0^TF_{Y_0}$ and set $H_0 = -G_0$.
- For $k = 0, 1, \ldots,$
  - ○ Minimize $F(Y_k(t))$ over t where
  $$Y(t) = YV\cos(\Sigma t)V^T + U\sin(\Sigma t)V^T$$
  and $U\Sigma V^T$ is the compact singular value decomposition of H_k.
  - ○ Set $t_k = t_{\min}$ and $Y_{k+1} = Y_k(t_k)$.
  - ○ Compute $G_{k+1} = F_{Y_{k+1}} - Y_{k+1}Y_{k+1}^TF_{Y_{k+1}}$.
  - ○ Parallel transport tangent vectors H_k and G_k to the point Y_{k+1}:
  $$\tau H_k = (-Y_kV\sin\Sigma t_k + U\cos\Sigma t_k)\Sigma V^T, \tag{3.1}$$
  $$\tau G_k = G_k - \big(Y_kV\sin\Sigma t_k + U(I-\cos\Sigma t_k)\big)U^TG_k. \tag{3.2}$$
  - ○ Compute the new search direction
  $$H_{k+1} = -G_{k+1} + \gamma_k\tau H_k, \qquad \text{where} \qquad \gamma_k = \frac{\langle G_{k+1} - \tau G_k, G_{k+1}\rangle}{\langle G_k, G_k\rangle}$$
  and $\langle \Delta_1, \Delta_2\rangle = \operatorname{tr}\Delta_1^T\Delta_2$.
  - ○ Reset $H_{k+1} = -G_{k+1}$ if $k+1 \equiv 0 \bmod p(n-p)$.

3.4. Conjugate gradient method on the Stiefel manifold. As with Newton's method, conjugate gradient on the two manifolds is very similar. One need only replace the definitions of tangent vectors, inner products, geodesics, gradients, and parallel translation. Geodesics, gradients, and inner products on the Stiefel manifold are given in section 2.4. For parallel translation along geodesics on the Stiefel manifold, we have no simple, general formula comparable to (3.2). Fortunately, a geodesic's tangent direction is parallel, so a simple formula for τH_k comparable to (3.1) is available, but a formula for τG_k is not. In practice, we recommend setting $\tau G_k := G_k$ and ignoring the fact that τG_k will not be tangent at the point Y_{k+1}. Alternatively, setting $\tau G_k := 0$ (also not parallel) results in a Fletcher–Reeves conjugate gradient formulation. As discussed in the next section, neither approximation affects the superlinear convergence property of the conjugate gradient method.

The conjugate gradient method on the Stiefel manifold is as follows.

Conjugate Gradient for Minimizing $F(Y)$ on the Stiefel Manifold

- Given Y_0 such that $Y_0^TY_0 = I$, compute $G_0 = F_{Y_0} - Y_0F_{Y_0}^TY_0$ and set $H_0 = -G_0$.
- For $k = 0, 1, \ldots,$
    - ○ Minimize $F(Y_k(t))$ over t where

      $$Y_k(t) = Y_kM(t) + QN(t),$$

      QR is the compact QR decomposition of $(I - Y_kY_k^T)H_k$, $A = Y_k^TH_k$, and $M(t)$ and $N(t)$ are p-by-p matrices given by the $2p$-by-$2p$ matrix exponential appearing in Newton's method on the Stiefel manifold in section 3.2.
    - ○ Set $t_k = t_{\min}$ and $Y_{k+1} = Y_k(t_k)$.
    - ○ Compute $G_{k+1} = F_{Y_{k+1}} - Y_{k+1}F_{Y_{k+1}}^TY_{k+1}$.
    - ○ Parallel transport tangent vector H_k to the point Y_{k+1}:

      $$\tau H_k = H_kM(t_k) - Y_kR^TN(t_k). \tag{3.3}$$

      As discussed above, set $\tau G_k := G_k$ or 0, which is *not* parallel.
    - ○ Compute the new search direction

      $$H_{k+1} = -G_{k+1} + \gamma_k\tau H_k, \qquad \text{where} \qquad \gamma_k = \frac{\langle G_{k+1} - \tau G_k, G_{k+1}\rangle}{\langle G_k, G_k\rangle}$$

      and $\langle \Delta_1, \Delta_2\rangle = \operatorname{tr}\Delta_1^T(I - \frac{1}{2}YY^T)\Delta_2$.
    - ○ Reset $H_{k+1} = -G_{k+1}$ if $k + 1 \equiv 0 \bmod p(n-p) + p(p-1)/2$.

3.5. Numerical results and asymptotic behavior.

3.5.1. Trace maximization on the Grassmann manifold. The convergence properties of the conjugate gradient and Newton's methods applied to the trace maximization problem $F(Y) = \operatorname{tr} Y^TAY$ are shown in Figure 3.1, as well as the convergence of an approximate conjugate gradient method and the Rayleigh quotient iteration for comparison. This example shows trace maximization on $G_{5,3}$, i.e., three-dimensional subspaces in five dimensions. The distance between the subspace and the known optimum subspace is plotted versus the iteration number, where the distance in radians is simply the square root of the sum of squares of the principal angles between the subspaces. The dimension of this space equals $3(5-3) = 6$; therefore, a conjugate gradient algorithm with resets should at least double in accuracy every six iterations. Newton's method, which is cubically convergent for this example (this point is discussed in section 4.1), should triple in accuracy every iteration. Variable precision numerical software is used to demonstrate the asymptotic convergence properties of these algorithms.

The thick black curve (CG-1) shows the convergence of the conjugate gradient algorithm using the Polak–Ribière formula. The accuracy of this algorithm is at least doubled between the first and sixth and the seventh and twelfth iterations,

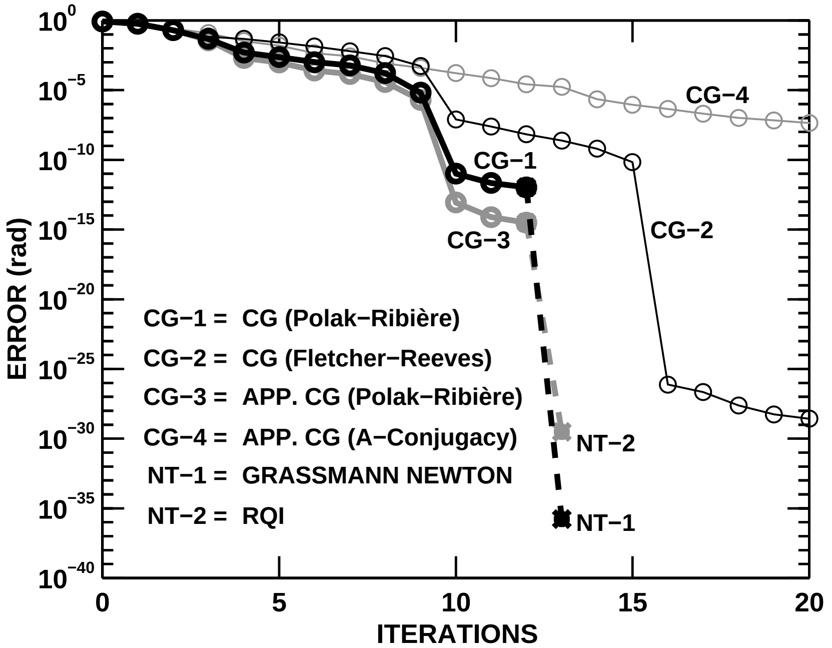


FIG. 3.1. *Convergence of the conjugate gradient and Newton's method for trace maximization on the Grassmann manifold* $G_{5,3}$. *The error* (*in radians*) *is the arc length distance between the solution and the subspace at the ith iterate* ((2.67) *and section* 4.3). *Quadratic convergence of conjugate gradient is evident, as is cubic convergence of Newton's method, which is a special property of this example.*

demonstrating this method's superlinear convergence. Newton's method is applied to the twelfth conjugate gradient iterate, which results in a tripling of the accuracy and demonstrates cubic convergence of Newton's method, shown by the dashed thick black curve (NT-1).

The thin black curve (CG-2) shows conjugate gradient convergence using the Fletcher–Reeves formula

$$\gamma_k = \langle G_{k+1}, G_{k+1}\rangle / \langle G_k, G_k\rangle. \tag{3.4}$$

As discussed below, this formula differs from the Polak–Ribière formula by second order and higher terms, so it must also have superlinear convergence. The accuracy of this algorithm more than doubles between the first and sixth, seventh and twelfth, and thirteenth and eighteenth iterations, demonstrating this fact.

The algorithms discussed above are actually performed on the constraint surface, but extrinsic approximations to these algorithms are certainly possible. By perturbation analysis of the metric given below, it can be shown that the conjugate gradient method differs from its flat space counterpart only by cubic and higher terms close to the solution; therefore, a flat space conjugate gradient method modified by projecting search directions to the constraint's tangent space will converge superlinearly. This is basically the method proposed by Bradbury and Fletcher [9] and others for the single eigenvector case. For the Grassmann (invariant subspace) case, we have performed line searches of the function $\phi(t) = \operatorname{tr} Q(t)^T AQ(t)$, where $Q(t)R(t) := Y + t\Delta$

is the compact QR decomposition and $Y^T\Delta = 0$. The QR decomposition projects the solution back to the constraint surface at every iteration. Tangency of the search direction at the new point is imposed via the projection $I - YY^T$.

The thick gray curve (CG-3) illustrates the superlinear convergence of this method when the Polak–Ribière formula is used. The Fletcher–Reeves formula yields similar results. In contrast, the thin gray curve (CG-4) shows convergence when conjugacy through the matrix A is used, i.e., $\gamma_k = -(H_k^T A G_{k+1})/(H_k^T A H_k)$, which has been proposed by several authors [67, Eq. (5)], [19, Eq. (32)], [36, Eq. (20)]. This method cannot be expected to converge superlinearly because the matrix A is in fact quite different from the true Hessian on the constraint surface. This issue is discussed further in section 4.4.

To compare the performance of Newton's method to the Rayleigh quotient iteration (RQI), which approximates Newton's method to high order (or vice versa), RQI is applied to the approximate conjugate gradient method's twelfth iterate, shown by the dashed thick gray curve (NT-2).

3.5.2. Orthogonal procrustes problem on the Stiefel manifold. The orthogonal Procrustes problem [32]

$$\min_{Y \in V_{n,\,p}} \|AY - B\|_F \quad A,\, B \text{ given matrices,} \tag{3.5}$$

is a minimization problem defined on the Stiefel manifold that has no known analytical solution for p different from 1 or n. To ensure that the objective function is smooth at optimum points, we shall consider the equivalent problem

$$\min_{Y \in V_{n,\,p}} \frac{1}{2}\|AY - B\|_F^2. \tag{3.6}$$

Derivatives of this function appear at the end of section 3.2. MATLAB code for Newton's method applied to this problem appears below. Convergence of this algorithm for the case $V_{5,\,3}$ and test matrices A and B is illustrated in Figure 3.2 and Table 3.1. The quadratic convergence of Newton's method and the conjugate gradient algorithm is evident. The dimension of $V_{5,3}$ equals $3(3-1)/2 + 6 = 9$; therefore, the accuracy of the conjugate gradient should double every nine iterations, as it is seen to do in Figure 3.2. Note that the matrix B is chosen such that a trivial solution $\hat{Y} = I_{n,\,p}$ to this test optimization problem is known.

MATLAB Code for Procrustes Problem on the Stiefel Manifold

```
n = 5; p = 3;

A = randn(n);
B = A*eye(n,p);
Y0 = eye(n,p);        % Known solution Y0
H = 0.1*randn(n,p);   H = H - Y0*(H'*Y0);  % small tangent vector H at Y0
Y = stiefgeod(Y0,H);  % Initial guess Y (close to know solution Y0)

% Newton iteration (demonstrate quadratic convergence)
d = norm(Y-Y0,'fro')
while d > sqrt(eps)
   Y = stiefgeod(Y,procrnt(Y,A,B));
   d = norm(Y-Y0,'fro')
end
```

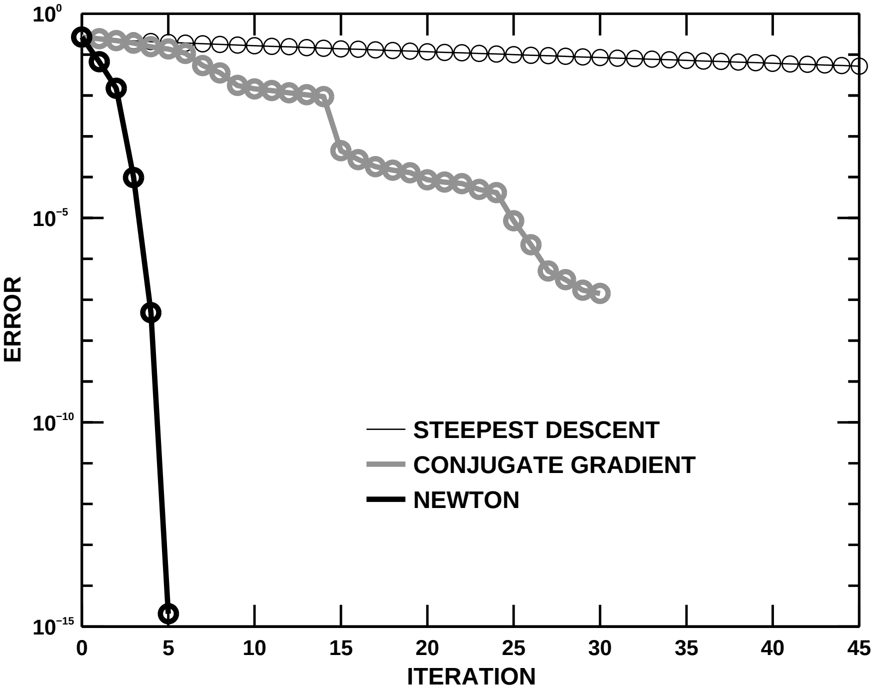


FIG. 3.2. *Convergence of the conjugate gradient and Newton's method for the orthogonal Procrustes problem on the Stiefel manifold* $V_{5,\,3}$*. The error is the Frobenius norm between the ith iterate and the known solution. Quadratic convergence of the conjugate gradient and Newton methods is evident. The Newton iterates correspond to those of Table* 3.1.

function stiefgeod

```
function [Yt,Ht] = stiefgeod(Y,H,t)
%STIEFGEOD Geodesic on the Stiefel manifold.
%       STIEFGEOD(Y,H) is the geodesic on the Stiefel manifold
%       emanating from Y in direction H, where Y'*Y = eye(p), Y'*H =
%       skew-hermitian, and Y and H are n-by-p matrices.
%
%       STIEFGEOD(Y,H,t) produces the geodesic step in direction H scaled
%       by t.  [Yt,Ht] = STIEFGEOD(Y,H,t) produces the geodesic step and the
%       geodesic direction.
[n,p] = size(Y);

if nargin < 3, t = 1; end
A = Y'*H;  A = (A - A')/2;  % Ensure skew-symmetry
[Q,R] = qr(H - Y*A,0);
MN = expm(t*[A,-R';R,zeros(p)]);  MN = MN(:,1:p);

Yt = Y*MN(1:p,:) + Q*MN(p+1:2*p,:);  % Geodesic from (2.45)
if nargout > 1, Ht = H*MN(1:p,:) - Y*(R'*MN(p+1:2*p,:)); end
                                      % Geodesic direction from (3.3)
```

Table 3.1

Newton's method applied to the orthogonal Procrustes problem on the Stiefel manifold using the MATLAB code given in this section. The matrix A is given below the numerical results, and $B = AI_{5,3}$. The quadratic convergence of Newton's method, shown by the Frobenius norm of the difference between Y_i and $\hat{Y} = I_{5,3}$, is evident. This convergence is illustrated in Figure 3.2. It is clear from this example that the difference $Y_i - \hat{Y}$ approaches a tangent vector at $\hat{Y} = I_{n,p}$, i.e., $\hat{Y}^T(Y_i - \hat{Y}) \to$ skew-symmetric.

Iterate i	$\lVert Y_i - \hat{Y}\rVert_F$	Y_i
0	2.68×10^{-01}	$\begin{pmatrix} 0.98341252163956 & -0.09749309852408 & -0.06630579165572 \\ 0.08482117605077 & 0.99248149019173 & -0.02619408666845 \\ 0.08655810575052 & 0.02896396566088 & 0.98816425471159 \\ 0.01388126419090 & 0.00902267322408 & 0.00728525462855 \\ 0.13423928340551 & 0.06749272129685 & -0.13563090573981 \end{pmatrix}$
1	6.71×10^{-02}	$\begin{pmatrix} 0.99954707914921 & 0.01554828497046 & 0.00423211303447 \\ -0.01656743168179 & 0.99905154070826 & 0.01216605832969 \\ -0.00306529752246 & -0.01070234416262 & 0.99915251911577 \\ -0.00910501510207 & -0.01286811040265 & 0.00924631200657 \\ -0.02321334579158 & -0.03706941336228 & 0.03798454294671 \end{pmatrix}$
2	1.49×10^{-02}	$\begin{pmatrix} 0.99993878247585 & 0.00296823825310 & 0.00486487784745 \\ -0.00301651579786 & 0.99998521441661 & 0.00192519989544 \\ -0.00479673956404 & -0.00191288709538 & 0.99996440819180 \\ -0.00311307788732 & -0.00157358730922 & 0.00121316839587 \\ -0.00897953054292 & -0.00382429023234 & 0.00650669969719 \end{pmatrix}$
3	9.77×10^{-05}	$\begin{pmatrix} 0.99999999888990 & 0.00000730457866 & -0.00003211124313 \\ -0.00000730341460 & 0.99999999951242 & 0.00000603747062 \\ 0.00003210887572 & -0.00000603508216 & 0.99999999682824 \\ 0.00000457898008 & -0.00001136276061 & 0.00002209393458 \\ 0.00003339025497 & -0.00002750041840 & 0.00006919392999 \end{pmatrix}$
4	4.81×10^{-08}	$\begin{pmatrix} 1.00000000000000 & 0.00000000813187 & 0.00000001705718 \\ -0.00000000813187 & 1.00000000000000 & 0.00000000613007 \\ -0.00000001705718 & -0.00000000613007 & 1.00000000000000 \\ -0.00000001001345 & -0.00000000397730 & 0.00000000429327 \\ -0.00000002903373 & -0.00000000827864 & 0.00000002197399 \end{pmatrix}$
5	2.07×10^{-15}	$\begin{pmatrix} 1.00000000000000 & 0.00000000000000 & 0.00000000000000 \\ 0.00000000000000 & 1.00000000000000 & 0.00000000000000 \\ 0.00000000000000 & 0.00000000000000 & 1.00000000000000 \\ 0.00000000000000 & 0.00000000000000 & 0.00000000000000 \\ 0.00000000000000 & 0.00000000000000 & 0.00000000000000 \end{pmatrix}$

$$A =$$

$$\begin{pmatrix} 0.59792470347241 & -1.60148995048070 & 1.29611959631725 & 0.00742708895676 & -0.09653196026400 \\ -0.34991267564713 & 1.03005546700300 & 0.38145454055699 & 0.14195063498923 & -0.16309797180034 \\ 0.16783050038338 & 0.51739189509778 & -0.42204935150912 & 1.75394028742695 & -0.63865179066515 \\ 0.24927536521443 & -1.34694675520019 & 0.92362255783368 & 0.62648865033822 & -0.31561702752866 \\ -0.24846337483192 & -0.44239067350975 & -1.52598136000449 & 0.89515519875598 & 0.87362106204727 \end{pmatrix}$$

function procrnt

```
function H = procrnt(Y,A,B)
%PROCRNT Newton Step on Stiefel Manifold for 1/2*norm(A*Y-B,'fro')^2.
%       H = PROCRNT(Y,A,B) computes the Newton step on the Stiefel manifold
%       for the function 1/2*norm(A*Y-B,'fro')^2, where Y'*Y = eye(size(Y,2)).

[n,p] = size(Y);
AA = A'*A;  FY = AA*Y - A'*B;  YFY = Y'*FY;  G = FY - Y*YFY';

% Linear conjugate gradient to solve a Newton step
dimV = p*(p-1)/2 + p*(n-p);             % == dim Stiefel manifold
```

```
% This linear CG code is modified directly from Golub and Van Loan [45]
H = zeros(size(Y));  R1 = -G;  P = R1;  P0 = zeros(size(Y));
for k=1:dimV
   normR1 = sqrt(stiefip(Y,R1,R1));
   if normR1 < prod(size(Y))*eps, break; end
   if k == 1, beta = 0; else, beta = (normR1/normR0)^2; end
   P0 = P;  P = R1 + beta*P;  FYP = FY'*P;  YP = Y'*P;
   LP = AA*P - Y*(P'*AA*Y) ...    % Linear operation on P
        - Y*((FYP-FYP')/2) - (P*YFY'-FY*YP')/2 - (P-Y*YP)*(YFY/2);
   alpha = normR1^2/stiefip(Y,P,LP);  H = H + alpha*P;
   R0 = R1;  normR0 = normR1;  R1 = R1 - alpha*LP;
end
```

function stiefip

```
function ip = stiefip(Y,A,B)
%STIEFIP Inner product (metric) for the Stiefel manifold.
%       ip = STIEFIP(Y,A,B) returns trace(A'*(eye(n)-1/2*Y*Y')*B),
%       where Y'*Y = eye(p), Y'*A & Y'*B = skew-hermitian, and Y, A,
%       and B are n-by-p matrices.

ip = sum(sum(conj(A).*(B - Y*((Y'*B)/2))));  % Canonical metric from (2.39)
```

3.6. Convergence rates of approximate methods. The algorithms presented in the previous sections are idealized in that geometrically natural ideas such as geodesics and parallel translation are used in their definitions. However, approximations can yield quadratic rates of convergence. In the limit, the Riemannian algorithms approach their Euclidean counterparts in the tangent plane of the solution point. A perturbation analysis shows which terms are necessary and which terms are not necessary to achieve quadratic convergence. The following argument holds for any Riemannian manifold and, therefore, applies to either the Grassmann or Stiefel manifold case.

Consider the conjugate gradient method applied to a function $F(Y)$ starting at a point Y within distance ϵ (small) of the solution $\hat{Y}$. For a manifold of dimension d, we must perform a sequence of d steps that take us within distance $O(\epsilon^2)$ of the solution $\hat{Y}$. The Riemannian conjugate gradient method

$$\begin{aligned} H_{\text{new}} &= -G_{\text{new}} + \gamma\tau H_{\text{old}}, \quad \gamma = \frac{\langle G_{\text{new}} - \tau G_{\text{old}}, G_{\text{new}}\rangle}{\|G_{\text{old}}\|^2}; \\ Y_{\text{new}} &= Y(t_{\min}), \quad Y(0) = Y_{\text{old}}, \quad \dot{Y}(0) = H_{\text{new}} \end{aligned}$$

does this, but we wish to approximate this procedure. Within a ball of size $O(\epsilon)$ around $\hat{Y}$, these quantities have sizes of the following orders:

Order	Quantity
$O(1)$	$t_{\min}$, γ
$O(\epsilon)$	G, H (new or old)
$O(\epsilon^2)$	$\|G\|^2$, $\|H\|^2$ (new or old)
$O(\epsilon^3)$	$\langle \tau G_{\text{old}}, G_{\text{new}}\rangle$

Also, by perturbation analysis of the Riemannian metric [18], [79, Vol. 2, Chap. 4, Props. 1 and 6], we have

$$\begin{aligned} Y(\epsilon) &= Y(0) + \epsilon\Delta + O(\epsilon^3), \\ \tau G(\epsilon) &= G + O(\epsilon^2), \\ \langle\,,\rangle &= I + O(\epsilon^2), \end{aligned}$$

where $Y(\epsilon)$ is a geodesic in direction Δ, $\tau G(\epsilon)$ is parallel translation of G along $Y(\epsilon)$, and the last approximation is valid for an orthonormal basis of the tangent plane at $Y(\epsilon\Delta)$ and I is the identity.

Inserting these asymptotics into the formulas for the conjugate gradient method shows that near the solution, eliminating the Riemannian terms gives $O(\epsilon^3)$ perturbations of the conjugate gradient sequence and, therefore, does not affect the quadratic rate of convergence. Furthermore, it can also be seen that eliminating the Polak–Ribière term $-\langle\tau G_{\mathrm{old}}, G_{\mathrm{new}}\rangle/\|G_{\mathrm{old}}\|^2$, yielding the Fletcher–Reeves algorithm, perturbs the conjugate gradient sequence by $O(\epsilon^2)$ terms, which does not affect the quadratic rate of convergence. Thus the approximate conjugate gradient methods discussed in section 3.5.1 converge quadratically.

4. Examples: Insights and applications. In this section, we consider ideas from the literature as applications of the framework and methodology developed in this paper. It is our hope that some readers who may be familiar with the algorithms presented here will feel that they now really see them with a new deeper but ultimately clearer understanding. It is our further hope that developers of algorithms that may somehow seem new will actually find that they also already fit inside of our geometrical framework. Finally, we hope that readers will see that the many algorithms that have been proposed over the past several decades are not just vaguely connected to each other, but are elements of a deeper mathematical structure. The reader who sees the depth and simplicity of section 4.10, say, has understood our message.

4.1. Rayleigh quotient iteration. If A is a symmetric matrix, it is well known that RQI is a cubically convergent algorithm. It is easy to derive formulas and show that it is true; here, we will explain our view of *why* it is true. Let $r(x)$ denote the Rayleigh quotient x^TAx, and, abusing notation, let $r(\theta)$ denote the Rayleigh quotient on a geodesic with $\theta = 0$ corresponding to an eigenvector of A.

Here is the intuition. Without writing down any formulas, it is obvious that $r(\theta)$ is an even function of θ; hence $\theta = 0$ is an extreme point. Newton's optimization method, usually quadratically convergent, converges cubically on nondegenerate even functions. Keeping in mind that $A - r(x)I$ is the second covariant derivative of the Rayleigh quotient, inverting it must amount to applying Newton's method. Following this intuition, RQI must converge cubically. The intuition is that simple.

Indeed, along a geodesic, $r(\theta) = \lambda\cos^2\theta + \alpha\sin^2\theta$ (we ignore the degenerate case $\alpha = \lambda$). The kth step of Newton's method for the univariate function $r(\theta)$ is readily verified to be

$$\theta_{k+1} = \theta_k - \tfrac{1}{2}\tan(2\theta_k) = -\tfrac{4}{3}\theta_k^3 + O(\theta_k^5).$$

We think of updating θ as moving along the circle. If we actually moved tangent to the circle by the Newton update $-\frac{1}{2}\tan(2\theta_k)$ and then projected to the circle, we would have the RQI

$$\theta_{k+1} = \theta_k - \arctan\bigl(\tfrac{1}{2}\tan(2\theta_k)\bigr) = -\theta_k^3 + O(\theta_k^5).$$

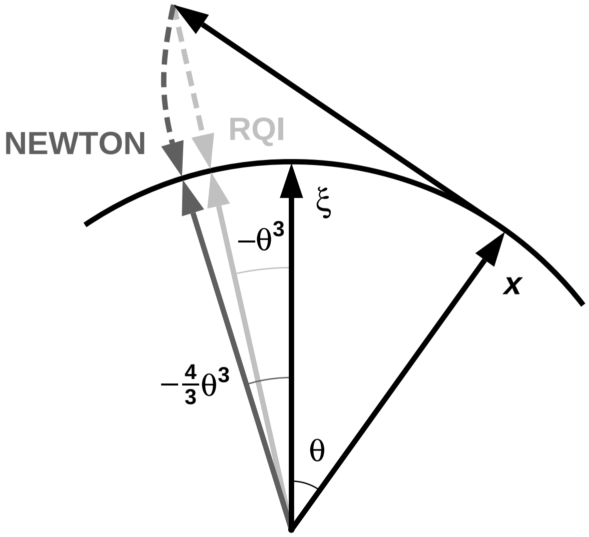


FIG. 4.1. *Cubic convergence of RQI and Newton's method applied to Rayleigh's quotient. The vector ξ is an eigenvector.*

This is the mechanism that underlies RQI. It "thinks" Newton along the geodesic, but moves along the tangent. The angle from the eigenvector goes from θ to $-\theta^3$ almost always. (Readers comparing with Parlett [65, Eq. (4-7-3)] will note that only positive angles are allowed in his formulation.)

When discussing the mechanism, we only need one variable: θ. This is how the mechanism should be viewed because it is independent of the matrix, eigenvalues, and eigenvectors. The algorithm, however, takes place in x space. Since $A - r(x)I$ is the second covariant derivative of $r(x)$ in the tangent space at x, the Newton update δ is obtained by solving $\Pi(A - r(x)I)\delta = -\Pi Ax = -(A - r(x)I)x$, where $\Pi = I - xx^T$ is the projector. The solution is $\delta = -x + y/(x^Ty)$, where $y = (A - r(x)I)^{-1}x$. The Newton step along the tangent direction is then $x \to x + \delta = y/(x^Ty)$, which we project to the unit sphere. This is exactly an RQI step. These ideas are illustrated in Figure 4.1.

One subtlety remains. The geodesic in the previous paragraph is determined by x and the gradient rather than x and the eigenvector. The two geodesics converge to each other by the inverse iteration process (almost always) allowing the underlying mechanism to drive the algorithm.

One trivial example where these issues arise is the generalization and derivation of Davidson's method [74, 26, 22]. In this context there is some question as to the interpretation of $D-\lambda I$ as a preconditioner. One interpretation is that it preconditions the eigenproblem by creating better eigenvalue spacings. We believe that there is a more appropriate point of view. In linear conjugate gradient for $Ax = b$, preconditioners are used to invert M which is an approximation to A (the Hessian of $\frac{1}{2}x^TAx - x^Tb$) against the gradient. This is an approximate Newton step. In nonlinear conjugate gradient, there is no consensus as to whether inverting the Hessian (which is approximated by $D - \lambda I$!) would constitute the ideal preconditioner, but it is a Newton step. Therefore, with the link between nonlinear conjugate gradient preconditioning and approximate Newton step, we see that Davidson's method is deserving of being called a preconditioner from the conjugate gradient point of view.

4.2. Coordinate singularities of symmetric matrices. An important open problem in numerical linear algebra is the complete understanding of the influence of singularities on computations [52, 17]. In this section we shall describe the singularity associated with multiple eigenvalues of symmetric matrices in terms of coordinate singularities, i.e., the breakdown of the coordinate representation. In section 4.10, we will describe how understanding this coordinate singularity underlies a regularization approach to eigenvalue optimization.

Matrix factorizations are nothing more than changes in variables or coordinate changes. In the plane, Cartesian and polar coordinates both give orthogonal systems, but polar coordinates have a coordinate singularity at the origin. A small perturbation near the origin can violently change the angle coordinate. This is ill-conditioning. If the r coordinate goes through the origin we have a singularity of the form $|r|$.

Consider traceless, symmetric, 2-by-2 matrices as follows:

$$A = \begin{pmatrix} x & y \\ y & -x \end{pmatrix}.$$

The positive eigenvalue is $r = \sqrt{x^2 + y^2}$, and one of the orthogonal eigenvectors is $\binom{\cos \frac{1}{2}\theta}{\sin \frac{1}{2}\theta}$, where $\tan \theta = y/x$. The conversion between matrix elements and the eigendecomposition is exactly the conversion from Cartesian to polar coordinates. Whatever ill-conditioning one associates with a symmetric matrix with two close eigenvalues, it is the same numerical difficulty associated with the origin in polar coordinates. The larger eigenvalue behaves like $|r|$ at the origin, and the eigenvector behaves like θ changing violently when perturbed. If one wants to think about all 2-by-2 symmetric matrices, add z as the trace, and the resulting interpretation is cylindrical coordinates.

We now generalize. Let S_n be the space of n-by-n symmetric matrices. Suppose that the largest p eigenvalues $\lambda_1, \ldots, \lambda_p$ coalesce. The corresponding eigenvectors are not uniquely determined, but the invariant subspace is. Convenient parameterizations are

$$\begin{aligned} S_n &\equiv \quad \text{Symmetric Matrices} \quad &&= \mathbf{R}^p \times V_{n,\,p} \times S_{n-p}, \\ S_{n,\,p} &\equiv \{\, S_n : \lambda_1 \text{ has multiplicity } p \,\} &&= \mathbf{R} \times G_{n,\,p} \times S_{n-p}. \end{aligned}$$

That is, any symmetric matrix may be parameterized by its p largest eigenvalues, the corresponding eigenvectors in order, and the $(n-p)$-by-$(n-p)$ symmetric operator on the space orthogonal to these eigenvectors. To parameterize a symmetric matrix with eigenvalue λ of multiplicity p, we must specify the invariant subspace corresponding to this eigenvalue and, once again, the $(n-p)$-by-$(n-p)$ symmetric operator on the orthogonal subspace. It is worth mentioning that the parameters in these decompositions give an orthonormal system (surfaces with constant parameters intersect orthogonally). The codimension of $S_{n,\,p}$ in S_n is $p(p+1)/2 - 1$, obtained by adding $p-1$ (corresponding to $\lambda_2, \ldots, \lambda_p$) to $p(p-1)/2$ (the codimension of $G_{n,\,p}$ in $V_{n,\,p}$).

Another interpretation of the well-known fact that when eigenvalues coalesce, eigenvectors, but not invariant subspaces, are ill-conditioned, is that the Stiefel manifold collapses to the Grassmann manifold. As with polar coordinates we have a coordinate singularity corresponding to ill-conditioning near $S_{n,\,p}$. Near this set, a small perturbation will violently move the Stiefel component. The singularity associated with the coalescing of eigenvalues is very much the singularity of the function $f(x) = |x|$.

4.3. The CS decomposition. The CS decomposition [45] should be recognized as the geodesic between two points on the Grassmann manifold. Any n-by-n orthogonal matrix Q may be written as

$$Q = \begin{pmatrix} V & 0 \\ 0 & U \end{pmatrix} \begin{pmatrix} C & -S & 0 \\ S & C & 0 \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} \tilde{V} & 0 \\ 0 & \tilde{U} \end{pmatrix}^T \tag{4.1}$$

for some p-by-p orthogonal matrices V and $\tilde{V}$ and $(n-p)$-by-$(n-p)$ orthogonal matrices U and $\tilde{U}$, and p angles θ_i where $C = \operatorname{diag}(\cos\theta_1, \dots, \cos\theta_p)$ and $S = \operatorname{diag}(\sin\theta_1, \dots, \sin\theta_p)$. Comparing this with the geodesic formula (2.65) and letting $\theta_i = t\sigma_i$ $(i = 1, \dots, p)$ where σ_i are the diagonal elements of Σ, we see that the first p columns of the CS decomposition traverse a geodesic emanating from $Y(0) = \binom{I}{0}$ (the origin). The next p columns give an orthogonal basis for the velocity vector along the geodesic (in fact, they are the orthogonal component of its polar decomposition).

As is well known, the θ_i are the principal angles between the subspaces spanned by the first p columns of Q and the origin. In general, let θ_i $(i = 1, \dots, p)$ be the principal angles between the two subspaces spanned by the columns of n-by-p orthonormal matrices Y_1 and Y_2, i.e., $U(\cos\Theta)V^T$ is the singular value decomposition of $Y_1^T Y_2$, where Θ is the diagonal matrix of principal angles. Also let θ and $\sin\theta$ represent the p-vectors formed by the θ_i and $\sin\theta_i$. These principal angles provide several different definitions of the distance between two subspaces as follows:

1. **arc length:** $d(Y_1, Y_2) = \|\theta\|_2$,
2. **Fubini-Study:** $d_{\mathrm{FS}}(Y_1, Y_2) = \arccos|\det Y_1^T Y_2| = \arccos(\prod_i \cos\theta_i)$,
3. **chordal 2-norm:** $d_{c2}(Y_1, Y_2) = \|Y_1 U - Y_2 V\|_2 = \|2\sin\frac{1}{2}\theta\|_\infty$,
4. **chordal Frobenius-norm:** $d_{cF}(Y_1, Y_2) = \|Y_1 U - Y_2 V\|_F = \|2\sin\frac{1}{2}\theta\|_2$,
5. **projection 2-norm** [45]**:** $d_{p2}(Y_1, Y_2) = \|Y_1 Y_1^T - Y_2 Y_2^T\|_2 = \|\sin\theta\|_\infty$,
6. **projection F-norm:** $d_{pF}(Y_1, Y_2) = 2^{-1/2}\|Y_1 Y_1^T - Y_2 Y_2^T\|_F = \|\sin\theta\|_2$.

The arc length distance is derived from the intrinsic geometry of the Grassmann manifold. The chordal 2-norm and Frobenius-norm distances are derived by embedding the Grassmann manifold in the vector space $\mathbf{R}^{n\times p}$, then using the 2- and Frobenius-norms, respectively, in these spaces. Note that these distances may be obtained from the minimization problems

$$d_{c2 \text{ or } cF}(Y_1, Y_2) = \min_{Q_1, Q_2 \in O_p} \|Y_1 Q_1 - Y_2 Q_2\|_{2 \text{ or } F}.$$

The projection matrix 2-norm and Frobenius-norm distances are derived by embedding the Grassmann manifold in the set of n-by-n projection matrices of rank p, then using the 2- and Frobenius-norms, respectively. The Fubini-Study distance is derived via the Plücker embedding of $G_{n,p}$ into the projective space $\mathbf{P}(\bigwedge^p(\mathbf{R}^n))$ (by taking wedge products between all columns of Y), then using the Fubini-Study metric [54].[2] Note that all metrics except the chordal and projection matrix 2-norm distances are asymptotically equivalent for small principal angles, i.e., these embeddings are isometries, and that for $Y_1 \neq Y_2$ we have the strict inequalities

$$d(Y_1, Y_2) > d_{\mathrm{FS}}(Y_1, Y_2), \tag{4.2}$$

[2] We thank Keith Forsythe for reminding us of this distance.

$$d(Y_1,Y_2) > d_{cF}(Y_1,Y_2) > d_{pF}(Y_1,Y_2), \tag{4.3}$$

$$d(Y_1,Y_2) > d_{cF}(Y_1,Y_2) > d_{c2}(Y_1,Y_2), \tag{4.4}$$

$$d(Y_1,Y_2) > d_{pF}(Y_1,Y_2) > d_{p2}(Y_1,Y_2). \tag{4.5}$$

These inequalities are intuitively appealing because by embedding the Grassmann manifold in a higher dimensional space, we may "cut corners" in measuring the distance between any two points.

4.4. Conjugate gradient for the eigenvalue problem. Conjugate gradient algorithms to minimize $\frac{1}{2}y^TAy$ (A symmetric) on the sphere ($p = 1$) is easy and has been proposed in many sources. The correct model algorithm for $p > 1$ presented in this paper is new. We were at first bewildered by the number of variations [2, 9, 33, 34, 3, 39, 35, 36, 69, 70, 38, 67, 19, 46, 93], most of which propose "new" algorithms for conjugate gradient for the eigenvalue problem. Most of these algorithms are for computing extreme eigenvalues and corresponding eigenvectors. It is important to note that none of these methods are equivalent to Lanczos [31]. It seems that the correct approach to the conjugate gradient algorithm for invariant subspaces ($p > 1$) has been more elusive. We are only aware of three papers [2, 70, 36] that directly consider conjugate gradient style algorithms for invariant subspaces of dimension $p > 1$. None of the proposed algorithms are quite as close to the new idealized algorithms as the $p = 1$ algorithms are. Each is missing important features which are best understood in the framework that we have developed. We discuss these algorithms below.

The simplest nontrivial objective function on the Grassmann manifold $G_{n,p}$ is the quadratic form

$$F(Y) = \frac{1}{2}\operatorname{tr} Y^TAY,$$

where A is a symmetric n-by-n matrix. It is well known that the solution to the minimization of F is the sum of the p smallest eigenvalues of A, with an optimal Y providing a basis for the invariant subspace corresponding to the p smallest eigenvalues.

To solve the eigenvalue problem, one may use the template directly from section 3.3 after deriving the gradient

$$\nabla F(Y) = AY - Y(Y^TAY)$$

and the second covariant derivative of $F(Y)$

$$\operatorname{Hess} F(\Delta_1,\Delta_2) = \operatorname{tr}\bigl(\Delta_1^TA\Delta_2 - (\Delta_1^T\Delta_2)Y^TAY\bigr).$$

The line minimization problem may be solved as p separate two-by-two problems in parallel, or it may be solved more completely by solving the $2p$-by-$2p$ eigenvalue problem. This does not follow the geodesic directly, but captures the main idea of the block Lanczos algorithm which in some sense is optimal [23, 24].

If one is really considering the pure linear symmetric eigenvalue problem, then pure conjugate gradient style procedures must be inferior to Lanczos. Every step of all proposed nonpreconditioned conjugate gradient algorithms builds vectors inside the same Krylov space in which Lanczos gives an optimal solution. However, exploring conjugate gradient is worthwhile. When the eigenvalue problem is nonlinear or the matrix changes with time, the Lanczos procedure is problematic because it stubbornly remembers past information that perhaps it would do well to forget. (Linear conjugate gradient, by contrast, benefits from the memory of this past information.) Applications towards nonlinear eigenvalue problems or problems that change in time drive us to consider the conjugate gradient method. Even the eigenvalue problem still plays a worthy role: it is the ideal model problem that allows us to understand the procedure much the way the Poisson equation on the grid is the model problem for many linear equation solvers.

Conjugate gradient on the sphere ($p = 1$) computes the smallest eigenvalue of a symmetric matrix A. Two papers [67, 19] consider imposing conjugacy through A. This is an unfortunate choice by itself because A is quite different from the Hessian $A - r(x)I$, where $r(x)$ is the Rayleigh quotient. A few authors directly consider conjugacy through the unconstrained Hessian [39, 93]. Others attempt to approximate conjugacy through the Hessian by using Polak–Ribiére or Fletcher–Reeves [9, 33, 34, 3, 35, 38, 46, 93, 69]. It is quite possible that most of these variations might well be competitive with each other and also our idealized algorithm, but we have not performed the numerical experiments because ultimately the $p = 1$ case is so trivial. A comparison that may be of more interest is the comparison with restarted Lanczos. We performed an informal numerical experiment that showed that the conjugate gradient method is always superior to two step Lanczos with restarts (as it should be since this is equivalent to the steepest descent method), but is typically slightly slower than four step Lanczos. Further experimentation may be needed in practice.

Turning to the $p > 1$ case, the three papers that we are aware of are [2, 70, 36]. The algorithm proposed in Alsén [2], has a built-in extra feature not in the idealized algorithm. Though this may not be obvious, it has one step of orthogonal iteration built in. This may be viewed as a preconditioning procedure giving the algorithm an advantage. The Sameh–Wisniewski [70] algorithm begins with many of the ideas of an idealized Grassmann algorithm, including the recognition of the correct tangent on the Grassmann manifold (though they only mention imposing the Stiefel constraint). Informal experiments did not reveal this algorithm to be competitive, but further experimentation might be appropriate. The more recent Fu and Dowling algorithm [36] imposes conjugacy through A and, therefore, we do not expect it to be competitive.

4.5. Conjugate gradient for the generalized eigenvalue problem. It is well known that the generalized eigenvalue problem $Ax = \lambda Bx$ may also be posed as a constrained optimization problem. Now we must find

$$\min \operatorname{tr} Y^TAY$$

subject to the constraint that

$$Y^TBY = I_p.$$

With the change of variables

$$\bar{Y} = B^{1/2}Y, \tag{4.6}$$

$$\bar{\Delta} = B^{1/2}\Delta, \tag{4.7}$$

$$\bar{A} = B^{-1/2}AB^{-1/2} \tag{4.8}$$

the problem becomes

$$\min \operatorname{tr} \bar{Y}^T\bar{A}\bar{Y} \quad \text{subject to} \quad \bar{Y}^T\bar{Y} = I_p.$$

The numerical algorithm will be performed on the nonoverlined variables, but the algorithm will be mathematically equivalent to one performed on the overlined variables.

Notice that the condition on tangents in this new coordinate system is that

$$\Delta^T BY = 0.$$

It is readily checked that the gradient of the trace minimization problem becomes

$$G = (B^{-1} - YY^T)AY$$

(note that $G^T BY = 0$).

Geodesics may be followed in any direction Δ for which $\Delta^T BY = 0$ by computing a compact variation on the SVD of Δ as follows:

$$\Delta = U\Sigma V^T, \quad \text{where } U^T BU = I.$$

For simplicity, let us assume that Δ has full rank p. The V vectors are the eigenvectors of the matrix $\Delta^T B\Delta$, while the U vectors are the eigenvectors of the matrix $\Delta\Delta^T B$ corresponding to the nonzero eigenvalues. There is also a version involving the two matrices

$$\begin{pmatrix} 0 & 0 & \Delta \\ B & 0 & 0 \\ 0 & \Delta^T & 0 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 0 & 0 & B \\ \Delta^T & 0 & 0 \\ 0 & \Delta & 0 \end{pmatrix}.$$

This SVD may be expressed in terms of the quotient SVD [45, 27].

Given the SVD, we may follow geodesics by computing

$$Y(t) = \begin{pmatrix} YV & U \end{pmatrix} \begin{pmatrix} C \\ S \end{pmatrix} V^T.$$

All the Y along this curve have the property that $Y^T BY = I$. For the problem of minimizing $\frac{1}{2}\operatorname{tr} Y^T AY$, line minimization decouples into p two-by-two problems just as in the ordinary eigenvalue problem.

Parallel transport, conjugacy, and the second covariant derivative may all be readily worked out.

4.6. Electronic structures computations. In this section, we briefly survey a research area where conjugate gradient minimization of nonquadratic but smooth functions on the Stiefel and Grassmann manifolds arise, the ab initio calculation of electronic structure within the local density approximation. Such approaches use only the charge and mass of electrons and atomic nuclei as input and have greatly furthered understanding of the thermodynamic properties of bulk materials [12], the structure and dynamics of surfaces [51, 61], the nature of point defects in crystals [60], and the diffusion and interaction of impurities in bulk materials [84]. Less than ten years ago, Car and Parrinello [13] in a watershed paper proposed minimization through simulated annealing. Teter and Gillan [42, 83] later introduced conjugate gradient based schemes and demonstrated an order of magnitude increase in the convergence rate. These initial approaches, however, ignored entirely the effects of curvature on the choice of conjugate search directions. Taking the curvature into *partial* account using a generalization of the Riemannian projection led to a further improvement in computation times by over a factor of three under certain conditions [5].

Our ability to compute ab initio, using only the charge and mass of electrons and atomic nuclei as input, the behavior of systems of everyday matter has advanced greatly in recent years. However, the computational demands of the approach and the attendant bounds on the size of systems which may be studied (several hundred atoms) have limited the direct impact of the approach on materials and chemical engineering. Several ab initio applications which will benefit technology tremendously remain out of reach, requiring an order of magnitude increase in the size of addressable systems. Problems requiring the simultaneous study of thousands of atoms include defects in glasses (fiber optics communications), complexes of extended crystalline defects (materials' strength and processing), and large molecules (drug design).

The theoretical problem of interest is to find the smallest eigenvalue E_0 of the Schrödinger equation in the space of $3N$-dimensional skew-symmetric functions,

$$H\psi = E_0\psi,$$

where the Hamiltonian operator H is defined by

$$H = \sum_{1\le n\le N}\left(-\frac{1}{2}\nabla_n^2 + V_{ion}(r_n)\right) + \frac{1}{2}\sum_{1<n\ll m\le N}\frac{1}{\|r_n - r_m\|^2}.$$

Here, N is the number of electrons in the system under study, now typically on the order of several hundred, r_i is the position of the ith electron, $V_{ion}(r)$ is the potential function due to the nuclei and inner electrons, and the second summation is recognized as the usual Coulomb interactions. Directly discretizing this equation at M gridpoints in space would lead to absurdly huge eigenvalue problems where the matrix would be M^N-by-M^N. This is not just a question of dense versus sparse methods, a direct approach is simply infeasible.

The fundamental theorems which make the ab initio approach tractable come from the density functional theory of Hohenberg and Kohn [50] and Kohn and Sham [55]. Density functional theory states that the ground states energy of a quantum mechanical system of interacting electrons and ions is equal to the solution of the problem of minimizing an energy function over all possible sets of N three-dimensional functions (electronic orbitals) obeying the constraints of orthonormality. Practical calculations generally use a finite basis to expand the orbitals, but for purposes of discussion, we may discretize the problem onto a finite spatial grid consisting of M points. The Kohn–Sham minimization then becomes

$$\begin{aligned} E_0 &= \min_{X^TX=I_N} E(X) \\ &\equiv \min_{X^TX=I_N} \operatorname{tr}(X^THX) + f\big(\rho(X)\big), \end{aligned} \tag{4.9}$$

where each column of X is a different electronic orbital sampled on the spatial grid, ρ is the vector $\rho_i(X) \equiv \sum_n |X_{in}|^2$, H is an M-by-M matrix (single-particle Hamiltonian), and f is a function which we leave unspecified in this discussion. In full generality the X are complex, but the real case applies for physical systems of large extent that we envisage for this application [66], and we, accordingly, take X to be real.

Recent advances in computers have enabled such calculations on systems with several hundreds of atoms [4, 11]. Further improvements in memory and performance will soon make feasible computations with upwards of a thousand atoms. However, with growing interest in calculations involving larger systems has come the awareness that as the physical length of systems under study increases, the Hessian about the minimum of (4.9) becomes increasingly ill-conditioned and nonconjugate minimization approaches exhibit a critical slowing down [83]. This observation prompted workers [42, 83] to apply conjugate gradient concepts to the problem, and now dozens of researchers have written papers using some form of the conjugate gradient method. In particular, one has a Grassmann problem when the number of electrons in each state is constant (i.e., two one spin up and one spin down). This is what happens in calculations on semiconductors and "closed shell" atoms and molecules. Otherwise, one has a Stiefel problem such as when one has metals or molecules with partially filled degenerate states.

The framework laid out in this paper may be of practical use to the ab initio density-functional community when the inner product computation through the Hessian of $E(X)$ is no more computationally complex to evaluate than calculating the energy function $E(X)$ or maintaining the orthonormality constraints $X^T X = I_N$. A suitable form for this inner product computation is

$$\begin{aligned}(4.10)\quad \frac{1}{2}\sum_{in,\,jm} Y_{in}\frac{\partial^2 E}{\partial X_{in}\partial X_{jm}} Z_{jm} &= \operatorname{tr}\bigl(Y^T(H+V)Z\bigr) + \sum_{ij}\sigma_i\left(2\frac{\partial^2 f}{\partial\rho_i\partial\rho_j}\right)\tau_j \\ &\quad - \operatorname{tr}\bigl(X^T(H+V)(XY^TZ)\bigr),\end{aligned}$$

where V is the diagonal matrix defined by $V_{ij} = (\partial f/\partial\rho_i)\delta_{ij}$, $\sigma_i \equiv \sum_n Y_{in}X_{in}$, $\tau_i \equiv \sum_n Z_{in}X_{in}$. Written this way, the first two terms of (4.10) have the same form and may be evaluated in the same manner as the corresponding terms in (4.9), with σ and τ playing roles similar to ρ. The third term, coming from the curvature, may be evaluated in the same way as the first term of (4.10) once given the object XY^TZ, which is no more computationally complex to obtain than the Gram–Schmidt orthonormalization of an object like X.

4.7. Subspace tracking. The problem of computing the principal invariant subspace of a symmetric or Hermitian matrix arises frequently in signal processing applications, such as adaptive filtering and direction finding [64, 72, 6, 73, 68]. Frequently, there is some time-varying aspect to the signal processing problem, and a family of time-varying principal invariant subspaces must be tracked. The variations may be due to either the addition of new information as in covariance matrix updating, a changing signal environment, or both. For example, compute the principal invariant subspace of either of the covariance matrices

$$(4.11)\qquad R_k = R_{k-1} + x_k x_k^T \qquad k = 1,\,2,\,\ldots,\text{ and } x_k \text{ is given,}$$

$$(4.12)\qquad R(t) = \text{a continuous function of } t$$

at every iteration or at discrete times. Equation (4.11) typically arises from updating the sample covariance matrix estimate; (4.12), the more general case, arises from a time-varying interference scenario, e.g., interference for airborne surveillance radar [85, 77]. Solving this eigenvalue problem via the eigenvalue or singular value decompositions requires a large computational effort. Furthermore, only the span of the first few principal eigenvectors may be required, whereas decomposition techniques compute all eigenvectors and eigenvalues, resulting in superfluous computations. Approaches to this problem may be classified as standard iterative methods [44], methods exploiting rank 1 updates [64, 53, 73, 94, 58, 81, 14, 57], i.e., (4.11), Lanczos based methods [20, 91, 90], gradient based methods [64, 92, 10], conjugate gradient based methods [38, 19, 71, 93, 75, 36, 78], which are surveyed by Edelman and Smith [31], Rayleigh–Ritz based methods [37, 20], and methods that exploit covariance matrix or array structure [68, 91, 90].

If the subspace does not change quickly over (discrete or continuous) time, then the desired solution will be close to the previously computed solution, and an iterative gradient-based algorithm such as the conjugate gradient algorithm may be computationally attractive for the subspace tracking problem. Thus the subspace tracking problem is treated as a time-varying optimization problem. Other conjugate gradient methods for computing principal invariant subspaces in a signal processing context have appeared [19, 71, 93, 36]; however, these conjugate gradient techniques do not exploit the structure of the subspace constraint (see section 4.4). Instead, we employ the conjugate gradient method on the Grassmann manifold, or an approximation of it discussed in section 3.5. Comon and Golub [20] describe and compare a wide variety of different algorithms for the problem of exponential covariance matrix updates, with particular emphasis on Lanczos and gradient-based algorithms. Yang, Sarkar, and Arvas [93] survey some conjugate gradient algorithms applied to computing the principal invariant subspace of a fixed symmetric matrix. We adopt the general assumption that the matrix may change arbitrarily over time, but that it must vary "slowly enough" so that using a conjugate gradient based approach is computationally efficient. This last constraint is, of course, dependent upon the application. For the example of space-time adaptive processing for airborne radar with a rotating antenna, Smith [78] shows that this method is capable of tracking the principal invariant subspace of clutter interference; however, when the interference dimension p is increased to account for new interference eigenvalues, one does better to compute the eigendecomposition from scratch and use it to initiate a new subspace track.

4.8. Newton's method for invariant subspace computations. Methods for refining estimates for invariant subspace computations have been proposed by Chatelin [15, 16], Dongarra, Moler, and Wilkinson [29], and Stewart [80]. Demmel [28, Sect. 3] proposes a unified approach by showing that they are all solutions to a Riccati equation.

These algorithms, when applied to symmetric matrices, are all variations on our geometrical Newton algorithm and may be understood in this context. There is nothing special about the eigenvalue problem; Newton's method for any function on the Grassmann manifold yields a Sylvester equation in the tangent space. The reason a Riccati equation arises rather than a Sylvester equation is that the previous algorithms formulate the problem in an affine space with arbitrary constraints. Previous researchers knew the quadratic term in the Riccati equation belonged there and knew that it somehow is related to the orthogonality constraints, but we now see that it is an artifact of a flat space derivation.

Let us take a closer look. Previous researchers proposed algorithms for invariant subspaces by asking for a solution to the matrix equation

$$AY - YB = 0$$

made nondegenerate by imposing the affine constraint

$$Z^T Y = I$$

for some arbitrary choice of Z. In the Dongarra et al. case, Z may be obtained by inverting and transposing an arbitrary p-by-p minor of the n-by-p matrix Y. In Moler's Matlab notation `Z=zeros(n,p); Z(r,:)=inv(Y(r,:))'`, where `r` denotes a p-vector of row indices. For Stewart, $Z = Y(Y^T Y)^{-1}$.

A mathematically insightful approach would require no arbitrary choice for Z. We would simply specify the problem by performing Newton's method on the function $F(Y) = \frac{1}{2}\operatorname{tr} Y^TAY$ on the Grassmann manifold. The stationary points of $F(Y)$ are the invariant subspaces. There is no need to specify any further constraints, and there are no degeneracies. (Notice that asking for the solution to $AY = Y(Y^TAY)$ subject to $Y^TY = I$ is a degenerate problem.)

Newton's method requires the solution Δ to the Sylvester equation

$$\Pi\bigl(A\Delta - \Delta(Y^TAY)\bigr) = -\Pi AY,$$

where $\Pi = (I - YY^T)$ denotes the projection onto the tangent space of the Grassmann manifold and $G = \Pi AY$ is the gradient. The solution is $\Delta = -Y + Z(Y^TZ)^{-1}$, where Z is the solution to the Sylvester equation $AZ - Z(Y^TAY) = Y$. Y may be chosen so that Y^TAY is diagonal, yielding simultaneous RQIs. If we move along the tangent and project rather than the geodesic we have the iteration sending Y to the Q factor in the QR decomposition of Z.

4.9. Reduced gradient methods, sequential quadratic programming, and Lagrange multipliers. In this section, we generalize beyond the Stiefel and Grassmann manifolds to show how the language and understanding of differential geometry provides insight into well-known algorithms for general nonlinear constrained optimization. We will show the role that geodesics play in these algorithms. In the next subsection, we will then apply the geometrical intuition developed here to directly formulate regularized sequential quadratic programs as is needed in eigenvalue optimization.

Here we study sequential quadratic programming (SQP) and reduced gradient methods (RGM). By SQP we mean the algorithm denoted as Newton SQP by Boggs and Tolle [7, p. 14], SQP by Nash and Sofer [59, p. 512], and QP-based projected Lagrangian by Gill, Murray, and Wright [41, p. 238, Eq. (6.41)]. By RGM, we specifically mean the method sometimes denoted as the reduced Hessian method [7, p. 25], other times simply denoted RGM [59, p. 520], and yet other times considered an example of an RGM [41, p. 221, Eq. (6.17)]. The difference is that RGM is derived based (roughly) on the assumption that one starts at a feasible point, whereas SQP does not.

We begin by interpreting geometrically the Lagrangian function as it is used in constrained optimization. Consider the optimization problem

$$\min_{x\in\mathbf{R}^n} f(x) \quad \text{given the constraint that} \quad h(x) = 0 \in \mathbf{R}^p. \tag{4.13}$$

For simplicity we consider the case where the level surfaces $h(x) = c$ are manifolds ($\partial h/\partial x$ has full rank everywhere) and we work with the Euclidean metric. In the Euclidean case, the formulations are routine in the optimization community, but we have not seen the geometric intuition (particularly geometric interpretations away from the optimization point and the role that geodesics play "behind-the-scenes") in the optimization references that we have consulted. Numerical Lagrange multiplier issues are discussed in [40] and [41], for example. In this paper, we give the new interpretation that the Hessian of the Lagrangian is the correct matrix for computing second derivatives along geodesics at every point, not only as an approximation to the result at the optimal point.

At every point $x \in \mathbf{R}^n$, it is possible to project the gradient of f onto the tangent space of the level surface through x. This defines a sort of flattened vector field.

In terms of formulas, projection onto the tangent space (known as computing least-squares Lagrange multiplier estimates) means finding λ that minimizes the norm of

$$\mathfrak{L}_x = f_x - \lambda \cdot h_x, \tag{4.14}$$

i.e.,

$$\lambda = f_x h_x^T (h_x h_x^T)^{-1}. \tag{4.15}$$

At every point $x \in \mathbf{R}^n$ (not only the optimal point) Lagrange multipliers are the coordinates of f_x in the normal space to a level surface of the constraint, i.e., the row space of h_x. (Our convention is that f_x is a 1-by-n row vector, and h_x is a p-by-n matrix whose rows are the linearizations of the constraints.)

If $x(t)$ is any curve starting at $x(0) = x$ that is constrained to the level surface at x, then $\mathfrak{L}_x \dot{x}$ computes the derivative of f along the curve. (In other words, $\mathfrak{L}_x$ is the first covariant derivative.) The second derivative of f along the curve is

$$\frac{d^2}{dt^2} f\bigl(x(t)\bigr) = \dot{x}^T \mathfrak{L}_{xx} \dot{x} + \mathfrak{L}_x \ddot{x}. \tag{4.16}$$

At the optimal point $\mathfrak{L}_x$ is 0, and, therefore, $\mathfrak{L}_{xx}$ is a second-order model for f on the tangent space to the level surface. The vanishing of the term involving $\mathfrak{L}_x$ at the optimal point is well known.

The idea that we have not seen in the optimization literature and that we believe to be new is the geometrical understanding of the quantity at a nonoptimal point: at any point at all, $\mathfrak{L}_x$ is tangent to the level surface while $\ddot{x}(t)$ is normal when x is a geodesic. The second term in (4.16) conveniently vanishes here too because we are differentiating along a geodesic! Therefore, the Hessian of the Lagrangian has a natural geometrical, meaning it is the second derivative of f along geodesics on the level surface, i.e., it is the second covariant derivative in the Euclidean metric.

We now describe the RGM geometrically. Starting at a point x on (or near) the constraint surface $h(x) = 0$, the quadratic function

$$\mathfrak{L}_x \dot{x} + \tfrac{1}{2} \dot{x}^T \mathfrak{L}_{xx} \dot{x}$$

models f (up to a constant) along geodesics emanating from x. The $\dot{x}$ that minimizes this function is the Newton step for the minimum for f. Intrinsic Newton would move along the geodesic in the direction of $\dot{x}$ a length equal to $\|\dot{x}\|$. Extrinsically, we can move along the tangent directly from x to $x + \dot{x}$ and then solve a set of nonlinear equations to project back to the constraint surface. This is RGM. It is a static constrained Newton method in that the algorithm models the problem by assuming that the points satisfy the constraints rather than trying to dynamically move from level surface to level surface as does the SQP.

In SQP, we start on some level surface. We now notice that the quadratic function

$$\mathfrak{L}_x \dot{x} + \tfrac{1}{2} \dot{x}^T \mathfrak{L}_{xx} \dot{x} \tag{4.17}$$

can serve as a model not only for the first and second covariant derivative of f on the level surface through x but also on level surfaces for points near x. The level surface through x is specified by the equation $h_x \dot{x} = 0$. Other parallel level surfaces are $h_x \dot{x} + c = 0$. The right choice for c is $h(x)$, which is a Newton step towards the level surface $h(x) = 0$. Therefore, if the current position is x, and we form the problem of minimizing $\mathfrak{L}_x\dot{x} + \frac{1}{2}\dot{x}^T\mathfrak{L}_{xx}\dot{x}$ subject to the constraint that $h_x\dot{x} + h(x) = 0$, we are minimizing our model of f along geodesics through a level surface that is our best estimate for the constraint $h(x) = 0$. This is the SQP method.

Practicalities associated with implementing these algorithms are discussed in the aforementioned texts. Generalizations to other metrics (non-Euclidean) are possible, but we do not discuss this in detail. Instead we conclude by making clear the relationship between Lagrange multipliers and the Christoffel symbols of differential geometry.

To derive the geodesic equation, let $f(x) = x_k$, the kth coordinate of x. From (4.15), the Lagrange multipliers are $h_{x_k}^T(h_x h_x^T)^{-1}$. Since $f_{xx} = 0$ we then have that the geodesic equations are $\ddot{x}_k = \dot{x}^T\mathfrak{L}_{xx}^k\dot{x}$ ($k = 1, \ldots, n$), where $\mathfrak{L}_{xx}^k$ denotes, $-h_{x_k}^T(h_x h_x^T)^{-1} \cdot h_{xx}$, the Hessian of the Lagrangian function of x_k. The matrix $\Gamma^k = -\mathfrak{L}_{xx}^k$ is the Christoffel symbol of differential geometry.

4.10. Eigenvalue optimization. The geometric approach allows the formulation of sequential quadratic programming problems when the Lagrange multiplier formalism breaks down due to coordinate singularities. Specifically, the geometric insight from the previous subsection is that during the execution of a sequential quadratic program there are three types of directions. The first direction is towards the constraint manifold. SQP performs a Newton step in that direction. The second family of directions is parallel to the constraint manifold. SQP forms a quadratic approximation to the objective function in the parallel level surface obtained from the Newton step. The remaining directions play no role in an SQP and should be ignored.

Consider the problem of minimizing the largest eigenvalue of $A(x)$, an n-by-n real symmetric matrix-valued function of $x \in \mathbf{R}^m$ when it is known that at the minimum, exactly p of the largest eigenvalues coalesce. Overton and Womersley [63] formulated SQPs for this problem using Lagrange multipliers and sophisticated perturbation theory. The constraint in their SQP was that the p largest eigenvalues were identical. Here, we will consider the case of $m > p(p+1)/2$. One interesting feature that they observed was the nondifferentiability of the largest eigenvalue at the optimum. Following the geometry of the previous section, a new algorithm without Lagrange multipliers may be readily devised. There will be no Lagrange multipliers because there will be no consideration of the third directions mentioned above.

We will write A for $A(x)$. Let $\Lambda = Y^TAY$, where the orthonormal columns of Y span the invariant subspace for the p largest eigenvalues of A, $\lambda_1, \ldots, \lambda_p$. We let $F(A) = \lambda_1$ and $\mathfrak{L}(A) = \operatorname{tr}(\Lambda) = \lambda_1 + \cdots + \lambda_p$. Unlike the function $F(A)$, $\mathfrak{L}(A)$ is a differentiable function at the optimal point. One might have guessed that this $\mathfrak{L}(A)$ was the right $\mathfrak{L}(A)$, but here is how one can logically deduce it.

The trick is to rely not on the Lagrange multiplier formalism of constraint functions, but rather on the geometry. Geometry has the power to replace a long complicated derivation with a short powerful one. Once the techniques are mastered, geometry provides the more intuitive understanding. There is no convenient $h(A)$ to express the constraint of multiple eigenvalues; artificially creating one leads to unnecessary complications due to the coordinate singularity when one moves from the level surface $h(A) = 0$ to another level surface. The right way to understand the coordinate singularity was described in section 4.2. The direction of the Newton step must be the first order constraint of the coallescing of the eigenvalues. Using the notation of section 4.2, the parallel directions are the tangent vectors of $S_{n,p}$. All other directions play no role. The natural level surfaces are thereby obtained by shifting the p largest eigenvalues by a constant and developing the orthogonal eigenvector matrix $Q(0)$ as in (2.32).

The message from section 4.9 is that whatever function we are interested in, we are only interested in the component of the gradient in the direction parallel to $S_{n,p}$. The very construction of a Lagrangian $\mathfrak{L}$ then may be viewed as the construction of an appropriate function with the property that $\mathfrak{L}_x$ is parallel to the tangent vectors of $S_{n,p}$. Of course the tangent space to $S_{n,p}$ (see section 4.2) includes projection matrices of the form $\sum_{i=1}^{p} \alpha_i y_i y_i^T$, where y_i is the eigenvector corresponding to λ_i, only when the α_i are all equal. This corresponds to an identical shift of these eigenvalues. Therefore, to form the correct gradient of the objective function $F(A) = \lambda_1$ everywhere, we should replace the true gradient, which is well known to be the spectral projector $y_1 y_1^T$, with its component in the direction YY^T, which is an $S_{n,p}$ tangent vector. Integrating, we now see that the act of forming the Lagrangian, which we now understand geometrically to mean replacing $y_1 y_1^T$ with YY^T (projecting the gradient to the surface of uniform shifts), amounts to nothing more than changing the objective function from $F(x)$ to $\mathfrak{L}(x) = \mathrm{tr}(\Lambda) = \mathrm{tr}\, Y^T AY$. While one might have guessed that this was a convenient Langrangian, we deduced it by projecting the gradient of $f(x)$ on the tangent space of a level surface. The components of $f(x)$ that we removed implicitly would have contained the Lagrange multipliers, but since these components are not well defined at the coordinate singularity, it is of little value to be concerned with them.

Now we must explicitly consider the dependence of $\mathfrak{L}$ on x. Our optimization step is denoted Δx, and $\dot{A}$ and $\ddot{A}$, respectively, denote $[A_x \Delta x]$ and $[A_{xx} \Delta x \Delta x]$ (notation from [63]). It is easy to verify that

$$\mathfrak{L}_x = \mathrm{tr}\, Y^T \dot{A} Y, \tag{4.18}$$

$$\mathfrak{L}_{xx} = \mathrm{tr}(Y^T \ddot{A} Y + Y^T \dot{A} \dot{Y} + \dot{Y}^T \dot{A} Y), \tag{4.19}$$

where $\dot{Y}$ is the solution to

$$\dot{Y}\Lambda - (I - YY^T)A\dot{Y} = (I - YY^T)\dot{A}Y \tag{4.20}$$

that satisfies $Y^T \dot{Y} = 0$. The resulting sequential quadratic program over Δx is then

$$\min \mathfrak{L}_x + \frac{1}{2}\mathfrak{L}_{xx}, \tag{4.21}$$

subject to the linear constraint (on Δx) that

$$Y^T \dot{A} Y + \Lambda = \alpha I, \tag{4.22}$$

where the scalar α is arbitrary.

Let us explain all of these steps in more detail. The allowable $\dot{Y}$ are Grassmann directions, $Y^T \dot{Y} = 0$. Otherwise, we are not parallel to the constraint surface. Equation (4.18) is the derivative of $Y^T AY$. Noting that $AY = Y\Lambda$ and $Y^T \dot{Y} = 0$, two terms disappear. Equation (4.19) is trivial but we note the problem that we do not have an explicit expression for $\dot{Y}$, we only have A, Y and $\dot{A}$. Fortunately, the perturbation theory for the invariant subspace is available from (4.20). It may be derived by differentiating $AY = Y\Lambda$ and substituting $\dot{\Lambda} = Y^T \dot{A} Y$.[3] The solution to (4.20) is unique so long as no other eigenvalue of A is equal to any of $\lambda_1, \ldots, \lambda_p$.

[3] Alert readers may notice that this is really the operator used in the definition of "sep" in numerical linear algebra texts. The reader really understands the theory that we have developed in this paper if he or she can now picture the famous "sep" operator as a Lie bracket with a Grassmann tangent and is convinced that this is the "right" way to understand "sep."

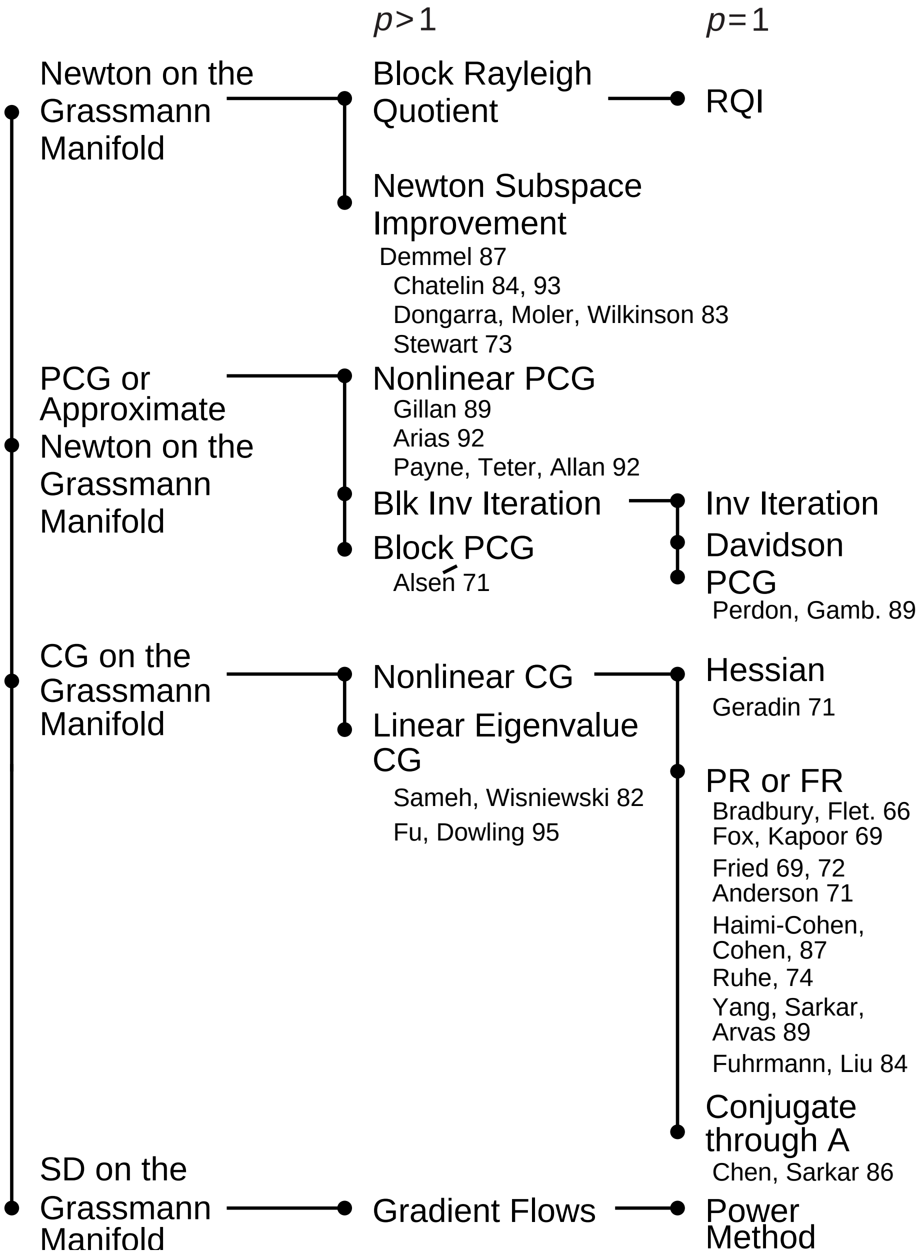


FIG. 4.2. *Taxonomy of algorithms defined from the Grassmann manifold.*

The linear constraint on Δx is the one that infinitesimally moves us to the constraint surface. It is the condition that moves us to a diagonal matrix. Therefore, $\dot{\Lambda} = Y^T \dot{A} Y$ when added to Λ must be a scalar multiple of the identity. This is a linear condition on $\dot{A}$ and, therefore, on Δx. The α does not explicitly appear in the constraint.

5. Conclusions. This paper offers a new approach to the algorithms in numerical analysis involving orthogonality constraints. We have found that these algorithms should be understood as optimization algorithms in the correct geometrical setting; however, they rarely are.

As a concluding example of the insight gained, we propose a Grassmann based taxonomy for problems related to the symmetric eigenproblem. This taxonomy allows us to view algorithms not as isolated entities, but as objects with a coherent mathematical structure. It is our hope that developers of new algorithms and perturbation theories will benefit from the analytical approach that lead to our taxonomy.

In this taxonomy, algorithms are viewed as either restrictions or approximations of their parent. Ultimately, we have Newton's method on arbitrary Riemannian manifolds as the root. One can then restrict to a particular manifold such as the Stiefel manifold or, as we illustrate in Figure 4.2, the Grassmann manifold. Along the vertical axis in the left column we begin with Newton's method which may be approximated first with preconditioned conjugage gradient (PCG) or approximate Newton methods, then pure conjugate gradient, and finally steepest descent. Moving from left to right the idealized algorithms are replaced with more practical versions that specialize for particular problems. The second column contains block algorithms, while the third contains single eigenvector related algorithms. This abstraction would not be possible without geometry.

Acknowledgments. The first author would like to thank Jim Demmel, Velvel Kahan, and Beresford Parlett who have stressed the importance of geometry in numerical linear algebra. We also thank Scott Axelrod, Victor Guillemin, and Shoshichi Kobayashi for a number of interesting conversations concerning differential geometry and Roger Brockett for the idea of investigating conjugate gradient methods on symmetric spaces. We further wish to thank Dianne O'Leary, Mike Todd, Mike Overton, and Margaret Wright for equally interesting conversations in the area of numerical optimization. We are indebted to the San Diego crowd consisting of Scott Baden, Beth Ong, Ryoichi Kawai (University of Alabama), and John Weare for working together towards understanding the electronic structure problem. In particular the first author thanks John Weare for his hospitality during visits in San Diego, where we explored the issue of conjugate gradient minimization. Steve Vavasis asked the penetrating question of how this relates to Lagrange multipliers which we answer in section 4.9. Furthermore, Ross Lippert has made a number of valuable suggestions that are acknowledged in this paper. We are indebted to Gene Golub for inviting Steve Smith to the 1993 Householder Symposium in Lake Arrowhead, California which serendipitously launched this collaboration.

We would especially like to thank our editor Mike Overton and the reviewers who went far beyond the call of duty with many very valuable suggestions that enhanced the readability and relevance of this paper.

REFERENCES

[1] D. Abbott, ed., *The Biographical Dictionary of Scientists; Mathematicians*, P. Bedrick Books, New York, 1986.

[2] B. M. Alsén, *Multiple Step Gradient Iterative Methods for Computing Eigenvalues of Large Symmetric Matrices*, Tech. report UMINF-15, University of Umeå, Sweden, 1971.

[3] I. Andersson, *Experiments with the Conjugate Gradient Algorithm for the Determination of Eigenvalues of Symmetric Matrices*, Tech. report UMINF-4.71, University of Umeå, Sweden, 1971.

[4] T. A. Arias and J. D. Joannopoulos, Ab initio *theory of dislocation interactions: From close-range spontaneous annihilation to the long-range continuum limit*, Phys. Rev. Lett., 73 (1994), pp. 680–683.

[5] T. A. Arias, M. C. Payne, and J. D. Joannopoulous, Ab initio *molecular dynamics: Analytically continued energy functionals and insights into iterative solutions*, Phys. Rev. Lett., 71 (1992), pp. 1077–1080.

[6] G. Bienvenu and L. Kopp, *Optimality of high resolution array processing using the eigensystem approach*, IEEE Trans. Acoust., Speech, Signal Processing, ASSP-31 (1983), pp. 1235–1248.

[7] P. T. Boggs and J. W. Tolle, *Sequential quadratic programming*, Acta Numerica, (1995), pp. 199–242.

[8] W. M. Boothby, *An Introduction to Differentiable Manifolds and Riemannian Geometry*, 2nd ed., Academic Press, New York, 1986.

[9] W. W. Bradbury and R. Fletcher, *New iterative methods for solutions of the eigenproblem*, Numer. Math., 9 (1966), pp. 259–267.

[10] R. W. Brockett, *Dynamical systems that learn subspaces*, in Mathematical Systems Theory: The Influence of R. E. Kalman, Springer-Verlag, Berlin, 1991.

[11] K. Brommer, M. Needels, B. E. Larson, and J. D. Joannopoulos, Ab initio *theory of the Si*(111)-(7×7) *surface reconstruction: A challenge for massively parallel computation*, Phys. Rev. Lett., 68 (1992), pp. 1355–1358.

[12] F. Buda, R. Car, and M. Parrinello, *Thermal expansion of* c-*Si via* ab initio *molecular dynamics.*, Phys. Rev. B, 41 (1990), pp. 1680–1683.

[13] R. Car and M. Parrinello, *Unified approach for molecular dynamics and density-functional theory*, Phys. Rev. Lett., 55 (1985), pp. 2471–2474.

[14] B. Champagne, *Adaptive eigendecomposition of data covariance matrices based on first-order perturbations*, IEEE Trans. Signal Processing, 42 (1994), pp. 2758–2770.

[15] F. Chatelin, *Simultaneous newton's iteration for the eigenproblem*, Comput. Suppl., 5 (1984), pp. 67–74.

[16] F. Chatelin, *Eigenvalues of Matrices*, John Wiley & Sons, New York, 1993.

[17] F. Chatelin and V. Frayssé, *Lectures on Finite Precision Computations*, SIAM, Philadelphia, PA, 1996.

[18] I. Chavel, *Riemannian Geometry—A Modern Introduction*, Cambridge University Press, London, 1993.

[19] H. Chen, T. K. Sarkar, S. A. Dianat, and J. D. Brulé, *Adaptive spectral estimation by the conjugate gradient method*, IEEE Trans. Acoust., Speech, Signal Processing, ASSP-34 (1986), pp. 272–284.

[20] P. Comon and G. H. Golub, *Tracking a few extreme singular values and vectors in signal processing*, Proc. IEEE, 78 (1990), pp. 1327–1343.

[21] S. H. Crandall, *Rotordynamic software*, in Rotating Machinery Dynamics, Proceedings of the Third International Symposium on Transport Phenomena and Dynamics of Rotating Machinery (ISROMAC-3), J. H. Kim and W.-J. Wang, eds., Hemisphere Publishing, New York, 1992, pp. 3–21.

[22] M. Crouzeix, B. Philippe, and M. Sadkane, *The Davidson method*, SIAM J. Sci. Comput., 15 (1994), pp. 62–76.

[23] J. K. Cullum, *The simultaneous computation of a few of the algebraically largest and smallest eigenvalues of a large, sparse, symmetric matrix*, BIT, 18 (1978), pp. 265–275.

[24] J. K. Cullum and R. A. Willoughby, *Lanczos Algorithms for Large Symmetric Eigenvalue Computations*, Vol. 1, Birkhäuser, Stuttgart, 1985.

[25] B. Datta, *Numerical Linear Algebra and Applications*, Brooks/Cole, Pacific Grove, CA, 1995.

[26] E. R. Davidson, *The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real symmetric matrices*, J. Comput. Phys., 17 (1975), pp. 87–94.

[27] B. De Moor, *The Riemannian singular value decomposition*, in Proc. 3rd Internat. Workshop on SVD and Signal Processing, Vol. 3, M. Moonen and B. D. Moor, eds., Elsevier Science, 1995, pp. 61–78.

[28] J. W. Demmel, *Three methods for refining estimates of invariant subspaces*, Computing, 38 (1987), pp. 43–57.

[29] J. J. Dongarra, C. B. Moler, and J. H. Wilkinson, *Improving the accuracy of computed eigenvalues and eigenvectors*, SIAM J. Numer. Anal., 20 (1983), pp. 23–45.

[30] A. Edelman, *Large dense numerical linear algebra in* 1993: *The parallel computing influence*, J. Supercomputing Applications, 7 (1993), pp. 113–128.

[31] A. Edelman and S. T. Smith, *On conjugate gradient-like methods for eigen-like problems*, BIT, 36 (1996), pp. 494–508. See also *Proc. Linear and Nonlinear Conjugate Gradient-*

Related Methods, Loyce Adams and J. L. Nazareth, eds., SIAM, Philadelphia, PA, 1996.

[32] L. Eldén, *Algorithms for the regularization of ill-conditioned least-squares problems*, BIT, 17 (1977), pp. 134–145.

[33] R. L. Fox and M. P. Kapoor, *A miminimization method for the solution of the eigenproblem arising in structural dynamics*, in Proc. 2nd Conf. Matrix Methods in Structural Mechanics, L. Berke, R. M. Bader, W. J. Mykytow, J. S. Przemieniecki, and M. H. Shirk, eds., Wright-Patterson Air Force Base, OH, 1969, pp. 271–306.

[34] I. Fried, *Gradient methods for finite element eigenproblems*, AIAA J., 7 (1969), pp. 739–741.

[35] I. Fried, *Optimal gradient minimization scheme for finite element eigenproblems*, J. Sound Vibration, 20 (1972), pp. 333–342.

[36] Z. Fu and E. M. Dowling, *Conjugate gradient eigenstructure tracking for adaptive spectral estimation*, IEEE Trans. Signal Processing, 43 (1995), pp. 1151–1160.

[37] D. R. Fuhrmann, *An algorithm for subspace computation, with applications in signal processing*, SIAM J. Matrix Anal. Appl., 9 (1988), pp. 213–220.

[38] D. R. Fuhrmann and B. Liu, *An iterative algorithm for locating the minimal eigenvector of a symmetric matrix*, in Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing, 1984, pp. 45.8.1–4.

[39] M. Geradin, *The computational efficiency of a new minimization algorithm for eigenvalue analysis*, J. Sound Vibration, 19 (1971), pp. 319–331.

[40] P. E. Gill and W. Murray, *The computation of Lagrange multipier estimates for constrained minimization*, Math. Programming, 17 (1979), pp. 32–60.

[41] P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*, 2nd ed., Academic Press, New York, 1981.

[42] M. J. Gillan, *Calculation of the vacancy formation energy in aluminium*, J. Physics, Condensed Matter, 1 (1989), pp. 689–711.

[43] S. Goedecker and L. Colombo, *Efficient linear scaling algorithm for tight-binding molecular dynamics*, Phys. Rev. Lett., 73 (1994), pp. 122–125.

[44] G. Golub and D. O'Leary, *Some history of the conjugate gradient and Lanczos methods*, SIAM Rev., 31 (1989), pp. 50–102.

[45] G. H. Golub and C. F. V. Loan, *Matrix Computations*, 2nd ed., Johns Hopkins University Press, Baltimore, MD, 1989.

[46] R. Haimi-Cohen and A. Cohen, *Gradient-type algorithms for partial singular value decomposition*, IEEE Trans. Pattern. Anal. Machine Intell., PAMI-9 (1987), pp. 137–142.

[47] S. Helgason, *Differential Geometry, Lie Groups, and Symmetric Spaces*, Academic Press, New York, 1978.

[48] H. G. Grassmann, *Die Ausdehnungslehre*, Enslin, Berlin, 1862.

[49] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Res. National Bureau of Standards, 49 (1952), pp. 409–436.

[50] P. Hohenberg and W. Kohn, *Inhomogeneous electron gas*, Phys. Rev., 136 (1964), pp. B864–B871.

[51] S. Ihara, S. L. Ho, T. Uda, and M. Hirao, Ab initio *molecular-dynamics study of defects on the reconstructed Si*(001) *surface*, Phys. Rev. Lett., 65 (1990), pp. 1909–1912.

[52] W. Kahan and J. Demmel, *Personal communication*, 1994–96.

[53] J. Karhunen, *Adaptive algorithms for estimating eigenvectors of correlation type matrices*, in Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing, 1984, pp. 14.6.1–4.

[54] S. Kobayashi and K. Nomizu, *Foundations of Differential Geometry*, Vols. 1 and 2, Wiley, New York, 1969.

[55] W. Kohn and L. J. Sham, *Self-consistent equations including exchange and correlation effects*, Phys. Rev., 140 (1965), pp. A1133–A1138.

[56] R. A. Lippert, *Personal communication* (*see also* http://www.mit.edu/people/ripper/grass/grassmann.html), 1995.

[57] G. Mathew, V. U. Reddy, and S. Dasgupta, *Adaptive estimation of eigensubspace*, IEEE Trans. Signal Processing, 43 (1995), pp. 401–411.

[58] M. Moonen, P. Van Dooren, and J. Vandewalle, *A singular value decomposition updating algorithm for subspace tracking*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 1015–1038.

[59] S. G. Nash and A. Sofer, *Linear and Nonlinear Programming*, McGraw–Hill, New York, 1995.

[60] M. Needels, J. D. Joannopoulos, Y. Bar-Yam, and S. T. Pantelides, *Oxygen complexes in silicon*, Phys. Rev. B, 43 (1991), pp. 4208–4215.

[61] M. Needels, M. C. Payne, and J. D. Joannopoulos, *High order reconstructions of the Ge*(100) *surface*, Phys. Rev. B, 38 (1988), pp. 5543–5546.

[62] B. O'Neill, *Semi-Riemannian Geometry with Applications to Relativity*, Academic Press,

New York, 1983.

[63] M. L. Overton and R. S. Womersley, *Second derivatives for optimizing eigenvalues of symmetric matrices*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 698–718.

[64] N. L. Owsley, *Adaptive data orthogonalization*, in Proc. IEEE Internat. Conf. Acoust., Speech, Signal Processing, 1978, pp. 109–112.

[65] B. Parlett, *The Symmetric Eigenvalue Problem*, Prentice–Hall, Englewood Cliffs, NJ, 1980.

[66] M. C. Payne, M. P. Teter, D. C. Allan, T. A. Arias, and J. D. Joannopoulos, *Iterative minimization techniques for ab initio total-energy calculations: Molecular dynamics and conjugate gradients*, Rev. Mod. Phys, 64 (1992), pp. 1045–1097.

[67] A. Perdon and G. Gambolati, *Extreme eigenvalues of large sparse matrices by Rayleigh quotient and modified conjugate gradients*, Comput. Methods Appl. Mech. Engrg., 56 (1986), pp. 251–264.

[68] R. Roy and T. Kailath, *ESPRIT—estimation of signal parameters via rotational invariance techniques*, IEEE Trans. Acoust., Speech, Signal Processing, 37 (1989), pp. 984–995.

[69] A. Ruhe, *Iterative eigenvalue algorithms for large symmetric matrices*, in Numerische Behandlung von Eigenwertaaufgaben Oberwolfach 1972, Internat. Series Numerical Math., Vol. 24, 1974, pp. 97–115.

[70] A. H. Sameh and J. A. Wisniewski, *A trace minimization algorithm for the generalized eigenvalue problem*, SIAM J. Numer. Anal., 19 (1982), pp. 1243–1259.

[71] T. P. Sarkar and N. Sangruji, *An adaptive nulling system for a narrow-band signal with a look direction constraint utilizing the conjugate gradient method*, IEEE Trans. Antennas, Propagation, 37 (1989), pp. 940–944.

[72] R. O. Schmidt, *Multiple emitter location and signal parameter estimation*, IEEE Trans. Antennas, Propagation, AP-34 (1986), pp. 276–280. Reprinted from Proc. RADC Spectrum Estimation Workshop, Griffiss Air Force Base, NY, 1979.

[73] R. Schreiber, *Implementation of adaptive array algorithms*, IEEE Trans. Acoust., Speech, Signal Processing, ASSP-34 (1986), pp. 1038–1045.

[74] G. L. G. Sleijpen and H. A. Van der Vorst, *A Jacobi–Davidson iteration method for linear eigenvalue problems*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 401–425.

[75] S. T. Smith, *Geometric Optimization Methods for Adaptive Filtering*, Ph.D. thesis, Harvard University, Cambridge, MA, 1993.

[76] S. T. Smith, *Optimization techniques on Riemannian manifolds*, in Fields Institute Communications, Vol. 3, AMS, Providence, RI, 1994, pp. 113–146.

[77] S. T. Smith, *Space-Time clutter covariance matrix computation and interference subspace tracking*, in Proc. 29th Asilomar Conf. Signals, Systems, Computers, Vol. 2, 1995, pp. 1193–1197.

[78] S. T. Smith, *Subspace tracking with full rank updates*, 31st Asilomar Conf. Signals, Systems, Computers, Vol. 1, 1997, pp. 793–797.

[79] M. Spivak, *A Comprehensive Introduction to Differential Geometry*, Vols. 1–3, 2nd ed., Publish or Perish, Houston, TX, 1979.

[80] G. W. Stewart, *Error and perturbation bounds for subspaces associated with certain eigenvalue problems*, SIAM Rev., 15 (1973), pp. 727–764.

[81] G. W. Stewart, *An updating algorithm for subspace tracking*, IEEE Trans. Signal Processing, 40 (1992), pp. 1535–1541.

[82] E. Stiefel, *Richtungsfelder und fernparallelismus in n-dimensionalem mannig faltigkeiten*, Commentarii Math. Helvetici, 8 (1935–1936), pp. 305–353.

[83] M. P. Teter, M. C. Payne, and D. C. Allan, *Solution of Schrödinger's equation for large systems*, Phys. Rev. B, 40 (1989), pp. 12255–12263.

[84] C. G. Van de Walle, Y. Bar-Yam, and S. T. Pantelides, *Theory of hydrogen diffusion and reactions in crystalline silicon*, Phys. Rev. Lett., 60 (1988), pp. 2761–2764.

[85] J. Ward, *Space-Time adaptive processing for airborne radar*, Tech. report 1015, DTIC No. ESC-TR-94-109, MIT Lincoln Laboratory, December 1994. See also Proc. IEEE Internat. Conf. Acoust., Speech, Signal Processing, Detroit, MI, May 1995, Vol. 5, pp. 2809–2812.

[86] R. C. Ward and L. J. Gray, *Eigensystem computation for skew-symmetric matrices and a class of symmetric matrices*, ACM Trans. Math. Software, 4 (1978), pp. 278–285.

[87] F. W. Warner, *Foundations of Differentiable Manifolds and Lie Groups*, Springer-Verlag, New York, 1983.

[88] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, England, 1965.

[89] Y.-C. Wong, *Differential geometry of Grassmann manifolds*, Proc. Nat. Acad. Sci. U.S.A., 57 (1967), pp. 589–594.

[90] G. Xu and T. Kailath, *Fast estimation of principal eigenspace using Lanczos algorithm*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 974–994.

[91] G. Xu and T. Kailath, *Fast subspace decomposition*, IEEE Trans. Signal Processing, 42 (1994), pp. 539–551.

[92] J.-F. Yang and M. Kaveh, *Adaptive eigenspace algorithms for direction or frequency estimation and tracking*, IEEE Trans. Acoust., Speech, Signal Processing, 36 (1988), pp. 241–251.

[93] X. Yang, T. P. Sarkar, and E. Arvas, *A survey of conjugate gradient algorithms for solution of extreme eigen-problems of a symmetric matrix*, IEEE Trans. Acoust., Speech, Signal Processing, 37 (1989), pp. 1550–1556.

[94] K.-B. Yu, *Recursive updating the eigenvalue decomposition of a covariance matrix*, IEEE Trans. Signal Processing, 39 (1991), pp. 1136–1145.

SIAM J. MATRIX ANAL. APPL.
Vol. 20, No. 2, pp. 354–362

A STABLE AND EFFICIENT ALGORITHM FOR THE INDEFINITE LINEAR LEAST-SQUARES PROBLEM*

S. CHANDRASEKARAN†, M. GU‡, AND A. H. SAYED§

Abstract. We develop an algorithm for the solution of indefinite least-squares problems. Such problems arise in robust estimation, filtering, and control, and numerically stable solutions have been lacking. The algorithm developed herein involves the QR factorization of the coefficient matrix and is provably numerically stable.

Key words. backward stability, error analysis, indefinite least-squares problems

AMS subject classifications. 15A06, 65F05, 65G05

PII. S0895479896302229

1. Introduction.

Many optimization criteria have been used for parameter estimation, starting with the standard least-squares formulation of Gauss (ca. 1795) and moving to more recent works on total least-squares (TLS) and robust (or H$^\infty$) estimation (see, e.g., [4, 5, 7, 8, 10, 11]). The latter formulations have been motivated by an increasing interest in estimators that are less sensitive to data uncertainties and measurement errors. They can both be shown to require the minimization of indefinite quadratic forms, where the standard inner product of two vectors, say a^Tb, is replaced by an indefinite inner product of the form a^TJb for a given *signature matrix*

$$J = \begin{pmatrix} I_p & 0 \\ 0 & -I_q \end{pmatrix},$$

where I_p and I_q are the identity matrices of dimensions p and q, respectively.

In this paper, we consider the *indefinite least-squares problems* of the form

$$\min_x \; (A\,x - b)^T \; J \; (A\,x - b)\,, \tag{1.1}$$

where $A \in R^{m\times n}$ is a given matrix with $m \geq n$; $b \in R^m$ is a given vector; and $p + q = m$. This problem reduces to the standard linear least-squares problem when $q = 0$. This is a characteristic of the so-called Krein spaces [5, 10].

Contrary to standard least-squares problems that always have solutions, the introduction of J with both positive and negative inertia can lead to minimization problems that are not necessarily solvable. Under certain solvability conditions, however, they lead to normal equations with positive-definite coefficient matrices. In this paper, we propose an algorithm for the solution of (1.1). We show that it is backward stable.

In section 2 we discuss situations where problem (1.1) might arise. In section 3 we solve problem (1.1). In section 4 we perform an error analysis.

*Received by the editors April 17, 1996; accepted for publication (in revised form) by N. J. Higham July 8, 1997; published electronically October 20, 1998.

http://www.siam.org/journals/simax/20-2/30222.html

†Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106 (shiv@ece.ucsb.edu).

‡Department of Mathematics, University of California, Los Angeles, CA 90095 (mgu@math.ucla.edu).

§Department of Electrical Engineering, University of California, Los Angeles, CA 90095 (sayed@ee.ucla.edu).

Throughout this paper, a *flop* is a real floating-point operation $\alpha \circ \beta$, where α and β are real floating-point numbers and $\circ$ is one of the operations $+$, $-$, $\times$, or $\div$. In our error analysis, we assume the following model for floating-point arithmetic:

$$\mathbf{fl}(\alpha \circ \beta) = (\alpha\,(1+\eta_1)) \circ (\beta\,(1+\eta_2))\,,$$

where $\mathbf{fl}(\alpha \circ \beta)$ is the floating-point result of the operation $\circ$, and $|\eta_i| \leq \epsilon$ with ϵ being the machine precision. For simplicity, we ignore the possibility of overflow and underflow.

2. Motivation of indefinite quadratic forms. We briefly indicate in this section how indefinite quadratic forms arise in the context of TLS and robust estimation methods.

Let $A \in R^{m\times n}$ be a given matrix with $m \geq n$, and let $b \in R^m$ be a given vector, which are assumed to be linearly related via an unknown vector of parameters $x \in R^n$,

$$b = A\,x + v. \tag{2.1}$$

The vector $v \in R^m$ explains the mismatch between Ax and the given vector (or observation) b.

2.1. The TLS problem. The TLS method has been devised to deal with data errors in both A and b; it incorporates possible errors in the matrix A into the problem formulation. More specifically, given (A, b) and assuming that both data quantities are noisy, the TLS problem seeks a matrix $\widehat{A}$ and a vector $\widehat{x}$ that minimize the following optimization problem (defined in terms of the Frobenius norm):

$$\min_{\widehat{A},\widehat{x}} \left\| \begin{bmatrix} \widehat{A} - A & \widehat{A}\,\widehat{x} - b \end{bmatrix} \right\|_F^2 \iff \min_{\widehat{A},\widehat{b}\in\mathcal{R}(\widehat{A})} \left\| \begin{bmatrix} A & b \end{bmatrix} - \begin{bmatrix} \widehat{A} & \widehat{b} \end{bmatrix} \right\|_F^2 . \tag{2.2}$$

The optimal solution $\widehat{A}$ is regarded as an approximation for A, which in turn is used to determine an $\widehat{x}$ that guarantees $\widehat{b} \in \mathcal{R}(\widehat{A})$. The solution of the TLS problem is known to be given by the following construction [7, p. 36].

Assume A is $m \times n$ with $m > n$ (i.e., A is a nonsquare matrix). Let $\{\sigma_1, \ldots, \sigma_n\}$ denote the singular values of A, with $\sigma_1 \geq \cdots \geq \sigma_n \geq 0$. Also, let $\{\bar{\sigma}_1, \ldots, \bar{\sigma}_n, \bar{\sigma}_{n+1}\}$ denote the singular values of the extended matrix $\begin{bmatrix} A & b \end{bmatrix}$. If $\bar{\sigma}_{n+1} < \sigma_n$, then the unique solution $\widehat{x}$ of (2.2) is given by

$$\widehat{x} = \left(A^T\,A - \bar{\sigma}_{n+1}^2 I_n\right)^{-1}\,A^T\,b.$$

This form of the solution shows that the TLS solution can also be obtained by minimizing the *indefinite* quadratic cost function

$$\min_x \left[\,\|b - A\,x\|_2^2 \;-\; \bar{\sigma}_{n+1}^2\|\,x\,\|_2^2\,\right].$$

The cost function can be rewritten in the form

$$\min_x \left(\begin{bmatrix} b \\ 0 \end{bmatrix} - \begin{bmatrix} A \\ \bar{\sigma}_{n+1} \end{bmatrix} x\right)^T \begin{bmatrix} I_n & 0 \\ 0 & -I_n \end{bmatrix} \left(\begin{bmatrix} b \\ 0 \end{bmatrix} - \begin{bmatrix} A \\ \bar{\sigma}_{n+1} \end{bmatrix} x\right),$$

where I_n denotes the identity matrix of size $n \times n$. This is a special case of the indefinite quadratic cost function to be studied in this paper (see (1.1)). A similar cost function also arises in the solution of a least-squares problem with bounded errors-in-variables [3].

2.2. Robust or H$^\infty$-smoothing. In recent years there has been an interest in (suboptimal) min-max estimation, with the belief that the resulting so-called robust or H$^\infty$ algorithms will be more robust and less sensitive to modeling assumptions (e.g., [8, 11]). In this section, we review the H$^\infty$-smoothing formulation, which can be shown to include as a special case the standard least-squares solution. The application to parameter estimation given in this section follows [5, 10].

Consider again the model (2.1). Assume that an arbitrary vector $\widehat{x}$ is picked as an estimate for the unknown x. Then, no matter what the given (A, b) are, it is always possible to find a vector $\widehat{v}$ that matches (2.1), i.e., that satisfies

$$b = A\,\widehat{x} + \widehat{v}\,.$$

The particular choice $\widehat{x}$ induces an error norm $\|x - \widehat{x}\|_2$ and a noise norm $\|\widehat{v}\|_2$. But since $\widehat{x}$ has been picked arbitrarily, these norms may be arbitrarily large or small. That is, the estimate may be good or bad, and one would like to develop a procedure that picks an estimate that always guarantees a certain level of performance.

To clarify this point even further, consider the case when the norm of the original perturbation v in (2.1) is small. In this case, the data vector b is only a slight perturbation apart from Ax. So one expects in this situation to be able to come up with a better estimate for x than in the case when the noise v is large. In other words, one would like to define a procedure that picks an $\widehat{x}$ in such a way that if the original perturbation v is small, then so will be the resulting error $(x - \widehat{x})$.

This idea can be formalized and leads to a so-called robust estimation problem. In this context, one seeks an estimate $\widehat{x}$ (affine in b, say $\widehat{x} = Kb + k$ for some $K \in R^{n\times m}, k \in R^n$) in order to guarantee that the following bound holds irrespective of the nature of the noise component v:

$$\text{find } \widehat{x} \text{ such that} \qquad \max_{v\neq 0} \frac{\|\, x - \widehat{x}\,\|_2^2}{\|\, v\,\|_2^2} \;\leq\; \gamma^2 \tag{2.3}$$

for a specified value of γ (say $\gamma = 1$ or some other value). The resulting estimate $\widehat{x}$, when it exists (and this depends on the value of γ), will guarantee that the maximum 2-norm gain from the disturbance v to the estimation error $(x - \widehat{x})$ will always be less than γ^2; hence the qualification "robust" estimate since it guarantees that if the disturbance v is small, then so will be the estimation error.

It is not difficult to see that, since $v = b - Ax$, an alternative way of requiring expression (2.3) to hold is to equivalently require the *indefinite* quadratic cost function

$$\mathcal{J} = \|\, A\,x - b\,\|_2^2 \;- \gamma^{-2}\,\|\, x - \widehat{x}\,\|_2^2$$

to be nonnegative for all x. That is, the optimization problem (2.3) over v is now an optimization problem over x. We shall not pursue in detail the complete solution of the robust smoothing problem here (instead, see [2, 5, 10]). We only note that $\mathcal{J}$ can be rewritten in the form

$$\mathcal{J} = \left(\begin{bmatrix} c \\ 0 \end{bmatrix} - \begin{bmatrix} A \\ \gamma^{-1} \end{bmatrix} z\right)^T \begin{bmatrix} I_n & 0 \\ 0 & -I_n \end{bmatrix} \left(\begin{bmatrix} c \\ 0 \end{bmatrix} - \begin{bmatrix} A \\ \gamma^{-1} \end{bmatrix} z\right),$$

where $z = x - \widehat{x}$ and $c = b - A\widehat{x}$. This is again a special case of (1.1).

3. Solution of the indefinite least-squares problem.

It is known that (see [5]) problem (1.1) has a unique solution if and only if

$$A^T J A \quad \text{is symmetric positive-definite.} \tag{3.1}$$

If this condition does not hold, then (1.1) can either have no solution or infinitely many solutions. We shall assume throughout this paper that condition (3.1) holds and, therefore, that problem (1.1) has a unique solution. In particular, condition (3.1) implies that $p \geq n$.

To solve (1.1), we first note that the quadratic cost function can be rewritten as

$$\begin{aligned}(A\,x-b)^T\;J\;(A\,x-b) &= x^T\;\left(A^T\;J\;A\right)\;x-2\left(A^T\;J\;b\right)^T\;x+b^T\;J\;b\\ &= (x-x_s)^T\;\left(A^T\;J\;A\right)\;(x-x_s)\\ &\quad +b^T\;J\;b-\left(A^T\;J\;b\right)^T\;\left(A^T\;J\;A\right)^{-1}\;\left(A^T\;J\;b\right),\end{aligned}$$

where x_s is the unique solution of the linear system of equations

$$\left(A^T\;J\;A\right)\;x_s=\left(A^T\;J\;b\right). \tag{3.2}$$

It follows from condition (3.1) that x_s is the unique solution to (1.1). Parallel to the usual least-squares problem, we refer to (3.2) as the *normal equation* associated with (1.1).

One straightforward approach to solving (3.2) is to directly form the coefficient matrix and the right-hand side, and then solve the equation by computing the Cholesky factorization of the coefficient matrix. However, this approach is in general *not* backward stable even for the usual least-squares problem, where J is the identity matrix (see [4, Chap. 5]).

In the following, we derive a new stable algorithm for computing x_s. We first compute the QR factorization of the matrix A, say

$$A=\begin{pmatrix} Q_1\\ Q_2\end{pmatrix}R=Q\;R\;,$$

where $R \in R^{n\times n}$ is upper triangular; $Q_1 \in R^{p\times n}$ and $Q_2 \in R^{q\times n}$; and $Q = (Q_1^T \quad Q_2^T)^T$ is column orthogonal, i.e.,

$$Q^T\;Q=Q_1^T\;Q_1+Q_2^T\;Q_2=I_n.$$

It follows from condition (3.1) that R is nonsingular. For our purposes, we explicitly compute the matrix Q as well. The cost of this decomposition is about $4n^2m-\frac{4}{3}n^3$ flops.

Now substituting the QR factorization of A into (3.2) and simplifying, we obtain

$$\left(Q^T\;J\;Q\right)\;R\,x_s=\left(Q_1^T\;Q_1-Q_2^T\;Q_2\right)R\,x_s=Q^T\;J\;b. \tag{3.3}$$

We remark that the fact that Q is orthogonal is *not* needed to get (3.3). In fact, this equation is equivalent to (3.2) for any factorization $A = QR$ as long as $R \in R^{n\times n}$ is nonsingular. This fact will be very important for our error analysis in section 4.

We also remark that condition (3.1) implies that the matrix

$$2\,Q_1^T\;Q_1-I_n=Q_1^T\;Q_1-Q_2^T\;Q_2$$

is symmetric positive-definite. Hence the singular values of Q_1 are all between $1/\sqrt{2}$ and 1. In other words, Q_1 is very well-conditioned, even though the matrix $Q_1^T Q_1 - Q_2^T Q_2$ itself could be very ill-conditioned.

To solve the linear system of equations (3.3), we form the matrix $Q_1^T Q_1 - Q_2^T Q_2$ explicitly and then compute its Cholesky factorization

$$Q_1^T\, Q_1 - Q_2^T\, Q_2 = L\, L^T,$$

where $L \in R^{n\times n}$ is lower triangular. The cost of this decomposition is about $n^2 m + \frac{1}{3} n^3$ flops. To compute x_s, we then compute the right-hand side in (3.3) and perform one forward and two backward substitutions. These computations cost about $O(mn)$ flops. Hence the total cost for computing x_s is about $(5m-n)n^2$ flops. We shall establish in section 4.2 that the proposed algorithm is backward stable.

Remark. A referee pointed out that an alternative method to solve the indefinite least-squares problem (1.1) can be derived by using hyperbolic Householder transforms (see Berry and Cybenko [1] and Rader and Steinhardt [9]). This alternative is potentially less expensive than the one in section 3, although it might not be backward stable.

4. Error analysis.

We now perform an error analysis for the proposed solution to the indefinite least-squares problem. We begin with some definitions and well-known results.

4.1. Preliminaries.

We use the definition of stability in [12, pp. 75–76]. Let $\mathcal{F}(\mathcal{X})$ be a function of the input data $\mathcal{X}$. We say that an algorithm for computing $\mathcal{F}(\mathcal{X})$ is *backward stable* if its output is exactly $\mathcal{F}(\bar{\mathcal{X}})$, where $\bar{\mathcal{X}}$ is a small perturbation of $\mathcal{X}$.

Let $A \in R^{m\times n}$ and $B \in R^{n\times l}$. When the matrix–matrix product $A\cdot B$ is computed in the straightforward way, the computed product $\mathbf{fl}\,(A\cdot B)$ satisfies (see [4, pp. 66–68])

$$\mathbf{fl}\,(A\cdot B) = A\cdot B + O\left(\epsilon \cdot \|A\|_2\, \|B\|_2\right).$$

Let $b \in R^m$. When the matrix–vector product $A\cdot b$ is computed in the straightforward way, the computed product satisfies (see [6, Chap. 3])

$$\mathbf{fl}\,(A\cdot b) = (A + \delta A)\cdot b,$$

where $\|\delta A\|_2 = O(\epsilon \cdot \|A\|_2)$. Let

$$A + \delta\widehat{A} = \widehat{Q}\, \widehat{R}$$

be the computed QR factorization of A (say, by Householder transformations) with $\delta\widehat{A} \in R^{m\times n}$, $\widehat{Q} \in R^{m\times n}$, and $\widehat{R} \in R^{n\times n}$. Then $\widehat{R}$ is upper triangular. The computed QR factorization is stable in that (see [6, Chap. 18])

$$\widehat{Q}^T\, \widehat{Q} = I_n + \Delta_1 \quad \text{with} \quad \Delta_1 = \Delta_1^T = O(\epsilon) \in R^{n\times n} \quad \text{and} \quad \|\delta\widehat{A}\|_2 = O(\epsilon\cdot\|A\|_2).$$

Let $M \in R^{n\times n}$ be a symmetric positive-definite matrix, and assume that a numerical Cholesky factorization of M can be successfully computed

$$M + \delta M = \widehat{L}\widehat{L}^T,$$

where $\widehat{L} \in R^{n\times n}$ is lower triangular. Then δM is symmetric and satisfies $\|\delta M\|_2 = O(\epsilon \cdot \|M\|_2)$. For details see [6, Chap. 10].

Let $R \in R^{n\times n}$ be a nonsingular upper (or lower) triangular matrix; let $b \in R^n$ be a vector; and let $\widehat{x}$ be the computed solution via backward (or forward) substitution to the linear system of equations

$$Rx = b.$$

Then $\widehat{x}$ satisfies

$$(R + \delta R)\,\widehat{x} = b, \tag{4.1}$$

where (see [6, Chap. 8])

$$|\delta R| \leq \frac{n\,\epsilon}{1 - n\,\epsilon} \cdot |R|.$$

Here $|\delta R|$ and $|R|$ are matrices of moduli of δR and R, respectively, and the inequality is meant entrywise. It follows that $\delta R = O(\epsilon \cdot \|R\|_2)$.

4.2. Analysis of the indefinite least-squares solution. Let

$$A + \delta\widehat{A} = \begin{pmatrix} \widehat{Q}_1 \\ \widehat{Q}_2 \end{pmatrix} \widehat{R} = \widehat{Q}\,\widehat{R}$$

be the *numerical* QR factorization of A, with $\widehat{R} \in R^{n\times n}$ upper triangular. It follows from section 4.1 that $\|\delta\widehat{A}\|_2 = O(\epsilon \cdot \|A\|_2)$ and that $\widehat{Q}$ is numerically column orthogonal, i.e.,

$$\widehat{Q}^T\,\widehat{Q} = \widehat{Q}_1^T\,\widehat{Q}_1 + \widehat{Q}_2^T\,\widehat{Q}_2 = I_n + \Delta_1, \tag{4.2}$$

where $\Delta_1 = \Delta_1^T = O(\epsilon) \in R^{n\times n}$.

We first assume that the matrix $\widehat{Q}_1^T\widehat{Q}_1 - \widehat{Q}_2^T\widehat{Q}_2$ has been computed and successfully Cholesky factorized, so that

$$\widehat{Q}_1^T\,\widehat{Q}_1 - \widehat{Q}_2^T\,\widehat{Q}_2 + \Delta_2 = \widehat{L}\,\widehat{L}^T. \tag{4.3}$$

It follows from section 4.1 that $\Delta_2 \in R^{n\times n}$ is symmetric and $\|\Delta_2\|_2 = O(\epsilon)$.

Let $\widehat{x}_s$ be the computed solution to (3.3). For simplicity we assume that $\widehat{R}$ is nonsingular and $\widehat{x}_s \neq 0$. According to section 4.1, $\widehat{x}_s$ satisfies

$$\left(\widehat{L} + \delta\widehat{L}_1\right)\left(\widehat{L} + \delta\widehat{L}_2\right)^T\left(\widehat{R} + \delta\widehat{R}\right)\,\widehat{x}_s = \left(\widehat{Q} + \delta\widehat{Q}\right)^T\,J\,b, \tag{4.4}$$

where

$$\|\delta\widehat{R}\|_2 = O(\epsilon \cdot \|\widehat{R}\|_2)\,, \quad \|\delta\widehat{Q}\|_2 = O(\epsilon), \quad \text{and} \quad \|\delta\widehat{L}_i\|_2 = O(\epsilon \cdot \|\widehat{L}\|_2) \quad \text{for} \quad i = 1, 2.$$

Since $\widehat{R}$ is a nonsingular upper triangular matrix, it follows from (4.1) that $\widehat{R} + \delta\widehat{R}$ is also nonsingular, and hence

$$\widehat{y} \overset{\text{def}}{=} \left(\widehat{R} + \delta\widehat{R}\right)\,\widehat{x}_s \neq 0.$$

We will write some of the round-off errors in (4.4) into $\widehat{Q}_1$ and b. To this end, define

$$\Delta_3 = v\,\widehat{y}^T + \widehat{y}\,v^T, \quad \text{where} \quad v = \frac{I - \dfrac{\widehat{y}\,\widehat{y}^T}{2\,\|\widehat{y}\|_2^2}}{\|\widehat{y}\|_2^2} \cdot \left(\delta\widehat{L}_1\,\widehat{L}^T + \widehat{L}\,\delta\widehat{L}_2^T + \delta\widehat{L}_1\,\delta\widehat{L}_2^T\right)\,\widehat{y}.$$

It is easy to verify that $\Delta_3 \in R^{n\times n}$ is symmetric and satisfies

$$\left(\widehat{L} + \delta\widehat{L}_1\right)\,\left(\widehat{L} + \delta\widehat{L}_2\right)^T\,\left(\widehat{R} + \delta\widehat{R}\right)\widehat{x}_s = \left(\widehat{L}\widehat{L}^T + \Delta_3\right)\,\left(\widehat{R} + \delta\widehat{R}\right)\widehat{x}_s.$$

Furthermore,

$$\begin{aligned}\|\Delta_3\|_2 &\le 2\,\|v\|_2\,\|\widehat{y}\|_2 \\ &\le 2\,\frac{\left\|I - \dfrac{\widehat{y}\,\widehat{y}^T}{2\,\|\widehat{y}\|_2^2}\right\|_2}{\|\widehat{y}\|_2^2} \cdot \left\|\delta\widehat{L}_1\,\widehat{L}^T + \widehat{L}\,\delta\widehat{L}_2^T + \delta\widehat{L}_1\,\delta\widehat{L}_2^T\right\|_2\,\|\widehat{y}\|_2\,\|\widehat{y}\|_2 \\ &= O\left(\epsilon\cdot\|\widehat{L}\|_2^2\cdot\left\|I - \frac{\widehat{y}\,\widehat{y}^T}{2\,\|\widehat{y}\|_2^2}\right\|_2\right) = O\left(\epsilon\right),\end{aligned}$$

where we have used the fact that

$$\|\widehat{L}\|_2^2 = \left\|\widehat{Q}_1^T\,\widehat{Q}_1 - \widehat{Q}_2^T\,\widehat{Q}_2 + \Delta_2\right\|_2 \le 1 + O(\epsilon) \quad \text{and} \quad \left\|I - \frac{\widehat{y}\,\widehat{y}^T}{2\,\|\widehat{y}\|_2^2}\right\|_2 \le 1.$$

Combining the above with equations (4.3) and (4.4), we have

$$\left(\widehat{Q}_1^T\widehat{Q}_1 - \widehat{Q}_2^T\widehat{Q}_2 + \Delta_2 + \Delta_3\right)\left(\widehat{R} + \delta\widehat{R}\right)\widehat{x}_s = \left(\widehat{Q} + \delta\widehat{Q}\right)^T\,J\,b.$$

Similar to section 3, relations (4.2) and (4.3) imply that the singular values of $\widehat{Q}_1$ are all between $1/\sqrt{2} + O(\epsilon)$ and $1 + O(\epsilon)$. Hence $\widehat{Q}_1$ is very well-conditioned.

In the following we shall rewrite $\Delta_2 + \Delta_3$ as a perturbation to $\widehat{Q}_1$. Let $\widehat{P} \in R^{p\times p}$ be the unique symmetric positive-definite matrix such that

$$\widehat{P}^2 = I_p + \widehat{Q}_1\,\left(\widehat{Q}_1^T\,\widehat{Q}_1\right)^{-1}\,(\Delta_2 + \Delta_3)\,\left(\widehat{Q}_1^T\,\widehat{Q}_1\right)^{-1}\,\widehat{Q}_1^T,$$

and let $(\widehat{P} + I)^{\frac{1}{2}}$ be the unique symmetric positive-definite square root of $\widehat{P} + I$. It follows that

$$\widehat{P} - I_p = \left(\widehat{P} + I\right)^{-\frac{1}{2}}\cdot\widehat{Q}_1\,\left(\widehat{Q}_1^T\,\widehat{Q}_1\right)^{-1}\,(\Delta_2 + \Delta_3)\,\left(\widehat{Q}_1^T\,\widehat{Q}_1\right)^{-1}\,\widehat{Q}_1^T\cdot\left(\widehat{P} + I\right)^{-\frac{1}{2}},$$

and that

$$\begin{aligned}\|\widehat{P} - I_p\|_2 &\le \left\|\left(\widehat{P} + I\right)^{-\frac{1}{2}}\right\|_2\cdot\left\|\widehat{Q}_1\,\left(\widehat{Q}_1^T\,\widehat{Q}_1\right)^{-1}\,(\Delta_2 + \Delta_3)\,\left(\widehat{Q}_1^T\,\widehat{Q}_1\right)^{-1}\,\widehat{Q}_1^T\right\|_2 \\ &\quad\cdot\left\|\left(\widehat{P} + I\right)^{-\frac{1}{2}}\right\|_2 \\ &\le \left\|\widehat{Q}_1\,\left(\widehat{Q}_1^T\,\widehat{Q}_1\right)^{-1}\,(\Delta_2 + \Delta_3)\,\left(\widehat{Q}_1^T\,\widehat{Q}_1\right)^{-1}\,\widehat{Q}_1^T\right\|_2 = O(\epsilon)\,.\end{aligned}$$

Now define

$$\widetilde{Q} = \begin{pmatrix} \widehat{P}\,\widehat{Q}_1 \\ \widehat{Q}_2 \end{pmatrix} = \widehat{Q} + O(\epsilon).$$

Since both $\widehat{P}$ and $\widehat{Q}_1$ are very well-conditioned, it follows that $\widetilde{Q}$ is itself very well-conditioned. Hence

$$\begin{aligned} \widetilde{Q}^T\, J\, \widetilde{Q} &= \left(\widehat{Q}_1^T\, \widehat{P}\right) \left(\widehat{Q}_1^T\, \widehat{P}\right)^T - \widehat{Q}_2^T\, \widehat{Q}_2 \\ &= \widehat{Q}_1^T\, \left(I_p + \widehat{Q}_1\, \left(\widehat{Q}_1^T\, \widehat{Q}_1\right)^{-1}\, (\Delta_2 + \Delta_3)\, \left(\widehat{Q}_1^T\, \widehat{Q}_1\right)^{-1} \widehat{Q}_1^T\right)\, \widehat{Q}_1 - \widehat{Q}_2^T\, \widehat{Q}_2 \\ &= \widehat{Q}_1^T\, \widehat{Q}_1 - \widehat{Q}_2^T\, \widehat{Q}_2 + \Delta_2 + \Delta_3. \end{aligned}$$

Hence we can now rewrite equation (4.4) simply as

$$\left(\widetilde{Q}^T\, J\, \widetilde{Q}\right)\, \left(\widehat{R} + \delta\widehat{R}\right)\, \widehat{x}_s = \left(\widehat{Q} + \delta\widehat{Q}\right)^T\, J\, b. \tag{4.5}$$

In the following, we write the round-off errors on the right-hand side as an error in b.

$$\begin{aligned} \left(\widehat{Q} + \delta\widehat{Q}\right)^T\, J\, b &= \left(\widetilde{Q}^T + \widetilde{Q}^T\, \widetilde{Q}\, \left(\widetilde{Q}^T\, \widetilde{Q}\right)^{-1}\, \left(\delta\widehat{Q} + \widehat{Q} - \widetilde{Q}\right)^T\right)\, J\, b \\ &= \widetilde{Q}^T\, J\, \left(b + \delta b\right), \end{aligned}$$

where

$$\delta b = J\, \widetilde{Q}\, \left(\widetilde{Q}^T\, \widetilde{Q}\right)^{-1}\, \left(\delta\widehat{Q} + \widehat{Q} - \widetilde{Q}\right)^T\, J\, b,$$

and hence $\|\delta b\|_2 = O(\epsilon \cdot \|b\|_2)$. Equation (4.5) can now be rewritten as

$$\left(\widetilde{Q}^T\, J\, \widetilde{Q}\right)\, \left(\widehat{R} + \delta\widehat{R}\right)\, \widehat{x}_s = \widetilde{Q}^T\, J\, \left(b + \delta b\right). \tag{4.6}$$

Finally we define the perturbation in A as

$$\delta A = \delta\widehat{A} + \left(\widetilde{Q} - \widehat{Q}\right)\, \widehat{R} + \widetilde{Q}\, \delta\widehat{R}.$$

Then it follows that $\|\delta A\|_2 = O(\epsilon \cdot \|A\|_2)$. It can be easily checked that

$$A + \delta A = \widetilde{Q}\left(\widehat{R} + \delta\widehat{R}\right). \tag{4.7}$$

With these backward errors, we note that (4.6) is exactly the equation (3.3) for the perturbed indefinite least-squares problem

$$\min_x\ \left((A + \delta A)\, x - (b + \delta b)\right)^T\, J\, \left((A + \delta A)\, x - (b + \delta b)\right).$$

Hence the new algorithm in section 3 is *backward stable*. Note that the matrix $\widetilde{Q}$ is in general not orthogonal, and hence the factorization (4.7) is in general *not* a QR factorization.

Now we consider the case where one fails to numerically compute the Cholesky factorization (4.3). This can happen only if the matrix

$$\widehat{Q}_1^T\,\widehat{Q}_1 - \widehat{Q}_2^T\,\widehat{Q}_2 + \Delta_2$$

is not symmetric positive-definite for a symmetric $\Delta_2 \in R^{n\times n}$ with a small 2-norm. With the techniques developed above, it is straightforward to show that this implies that there exists a $\delta A \in R^{m\times n}$ such that the matrix $(A+\delta A)^T J(A+\delta A)$ is not symmetric positive-definite. In other words, the indefinite least-squares problem (1.1) does not have a unique solution for a slightly perturbed A. Such a problem cannot be expected to have a numerically meaningful solution in general.

5. Conclusion. In this paper we proposed a stable and efficient algorithm for solving the indefinite least-squares problem. Our error analysis shows that this algorithm is backward stable.

Acknowledgments. The authors thank the anonymous referees for their helpful comments and suggestions.

REFERENCES

[1] M. Berry and G. Cybenko, *Hyperbolic Householder algorithms for factoring structured matrices*, SIAM J. Matrix Anal. Appl., 11 (1990), pp. 499–520.

[2] S. Chandrasekaran, G. H. Golub, M. Gu, and A. H. Sayed, *Parameter estimation in the presence of bounded data uncertainties*, SIAM J. Matrix Anal. Appl., 19 (1998), pp. 235–252.

[3] S. Chandrasekaran, G. H. Golub, M. Gu, and A. H. Sayed, *An efficient algorithm for a bounded errors-in-variables model*, SIAM J. Matrix Anal. Appl., to appear.

[4] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed., The Johns Hopkins University Press, Baltimore, MD, 1996.

[5] B. Hassibi, A. H. Sayed, and T. Kailath, *Linear estimation in Krein spaces - Part* I*: Theory*, IEEE Trans. Automat. Control, 41 (1996), pp. 18–33.

[6] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia, PA, 1996.

[7] S. Van Huffel and J. Vandewalle, *The Total Least Squares Problem: Computational Aspects and Analysis*, SIAM, Philadelphia, PA, 1991.

[8] P. Khargonekar and K. M. Nagpal, *Filtering and smoothing in an H^∞-setting*, IEEE Trans. Automat. Control, AC-36 (1991), pp. 151–166.

[9] C. M. Rader and A. O. Steinhardt, *Hyperbolic Householder transforms*, SIAM J. Matrix Anal. Appl., 9 (1988), pp. 269–290.

[10] A. H. Sayed, B. Hassibi, and T. Kailath, *Fundamental Inertia Conditions for the Minimization of Quadratic Forms in Indefinite Metric Spaces*, Oper. Theory: Adv. Appl., Birkhauser, Cambridge, MA, 1996.

[11] U. Shaked and Y. Theodor, *H^∞-optimal estimation: A tutorial*, in Proc. IEEE Conference on Decision and Control, Tucson, AZ, 1992, pp. 2278–2286.

[12] G. W. Stewart, *Introduction to Matrix Computations*, Academic Press, New York, 1973.

SIAM J. MATRIX ANAL. APPL.
Vol. 20, No. 2, pp. 363–372

BACKWARD PERTURBATION BOUNDS FOR LINEAR LEAST SQUARES PROBLEMS*

MING GU†

Abstract. Recently, Higham, Waldén, Karlson, and Sun have provided formulas for computing the best backward perturbation bounds for the linear least squares problem. In this paper we provide several backward perturbation bounds that are easier to compute and optimal up to a factor less than 2. We also show that any least squares algorithm that is stable in the sense of Stewart is necessarily a backward stable algorithm. Our results make it possible to measure numerically the amount of accuracy in any alleged solution of a least squares problem.

Key words. linear least squares problem, perturbation, stability

AMS subject classifications. 15A06, 65F05, 65G05

PII. S0895479895296446

1. Introduction.

Given a matrix $M \in \mathbf{R}^{m\times n}$ with $m \geq n$ and a vector $h \in \mathbf{R}^m$, the linear least squares problem is

$$\min_x \|M\, x - h\|_2. \tag{1.1}$$

Assume that M has full column rank, then the unique solution to (1.1) is

$$x_M = \left(M^T\, M\right)^{-1}\, M^T\, h.$$

Let $\mathcal{A}$ be an algorithm for solving (1.1) and let $\hat{x}_M \in \mathbf{R}^n$ be the numerical solution computed by $\mathcal{A}$ in machine precision ϵ. We say that $\mathcal{A}$ is numerically *stable* if for any such M and h, there exist small perturbation matrices and vectors $\delta M \in \mathbf{R}^{m\times n}$, $\delta h \in \mathbf{R}^m$, and $\delta\hat{x}_M \in \mathbf{R}^n$ such that (see Stewart [18, pp. 75–76])

$$\left\|(M+\delta M)\;(\hat{x}_M+\delta\hat{x}_M)-(h+\delta h)\right\|_2 = \min_x \left\|(M+\delta M)\; x-(h+\delta h)\right\|_2\,, \tag{1.2}$$

where $\|\delta M\|_2 \leq \tau\; \|M\|_2$, $\|\delta h\|_2 \leq \tau\; \|h\|_2$, and $\|\delta\hat{x}_M\|_2 \leq \tau\; \|\hat{x}_M\|_2$, with $\tau > 0$ being a small multiple of ϵ.

It is well known that if M is ill conditioned, then $\hat{x}_M$ can be very different from the exact solution x_M (see Higham [11, Chapter 19]). We will call $\hat{x}_M$ a *stable solution* to (1.1) if it satisfies (1.2).

One straightforward method for solving (1.1) is to compute the QR-factorization of M using Householder transformations (see, for example, Businger and Golub [5]). This method is stable since it produces a numerical solution $\hat{x}_M$ which satisfies (1.2) with $\delta h = 0$ and $\delta\hat{x}_M = 0$ (see, for example, [11, Chapter 19]). However, if M is a structured matrix such as the Toeplitz matrix, then this method can be inefficient. There is a vast literature on fast methods for solving structured linear least squares problems in $O(mn)$ floating point operations or less, as opposed to $O(mn^2)$ floating

*Received by the editors December 27, 1995; accepted for publication (in revised form) by Z. Strakos June 24, 1997; published electronically October 20, 1998. This work was supported in part by NSF Career Award CCR-9702866 and by the Applied Mathematical Sciences Subprogram of the Office of Energy Research, U.S. Department of Energy under contract DE-AC03-76SF00098.

http://www.siam.org/journals/simax/20-2/29644.html

†Department of Mathematics, University of California, Los Angeles, CA 90095-1555 (mgu@math.ucla.edu).

point operations normally required by the above QR-factorization method. Some of these fast methods produce numerical solutions that satisfy (1.2) with nonzero δh and $\delta \hat{x}_M$ (see, for example, Gu [9]), while others can be numerically unstable (see Bojanczyk and Brent [2, 4], Bojanczyk, Brent, and de Hoog [3], Brent [4], Chun, Kailath, and Lev-Ari [6], Cybenko [7, 8], Luk and Qiao [14], Nagy [15], Park and Eldén [16], Qiao [17], and Sweet [19]).

These fast methods vary greatly for different matrix structures. In order to be able to provide a general procedure to verify numerically the stability properties of these methods, it is necessary to solve the following problem:

$(\mathcal{P})$ *Given a vector $\hat{x}_M \in \mathbf{R}^n$, verify whether it is a stable solution to* (1.1).

In this paper, we will solve Problem $(\mathcal{P})$ by finding out whether there exist small perturbations δM, δh, and $\delta \hat{x}_M$, for which $\hat{x}_M + \delta \hat{x}_M$ satisfies (1.2). For simplicity, we assume throughout this paper that $M \neq 0$ and $h \neq 0$.

1.1. Backward perturbation bounds. A restricted version of Problem $(\mathcal{P})$ is the problem of whether there exists a small perturbation $\widehat{\delta M} \in \mathbf{R}^{m \times n}$ for which

$$\left\| (M + \widehat{\delta M})\, \hat{x}_M - h \right\|_2 = \min_x \|(M + \widehat{\delta M})\, x - h\|_2, \tag{1.3}$$

where $\|\widehat{\delta M}\|_2 \leq \tau\, \|M\|_2$, with $\tau > 0$ being a small multiple of ϵ. We will call $\hat{x}_M$ a *backward stable solution* to (1.1) if it satisfies (1.3). A backward stable solution is clearly a stable solution. As mentioned above, solving (1.1) via the QR-factorization of M produces an $\hat{x}_M$ that satisfies (1.3). Hence, this method is backward stable.

In general, the matrix $\widehat{\delta M}$ in (1.3) is not uniquely defined. Recently, Waldén, Karlson, and Sun [20] and Higham [11, Chapter 19] have provided the following formula for computing the smallest $\|\widehat{\delta M}\|_F$ among all the possible matrices $\widehat{\delta M}$ that satisfy (1.3).

THEOREM 1.1. *Let $r = h - M\, \hat{x}_M$. Then the optimal normwise backward error in F-norm is*

$$\begin{aligned} \mathcal{E}\,(\hat{x}_M) &:= \min\left\{ \|\widehat{\delta M}\|_F, \ \ \textit{where } \widehat{\delta M} \textit{ is a solution to } (1.3) \right\} \\ &= \begin{cases} 0 & \textit{if } r = 0, \\ \dfrac{\|M^T\, h\|_2}{\|h\|_2} & \textit{if } \hat{x}_M = 0, \\ \min\{\eta, \sigma_{\min}([M \quad \eta\, C])\} & \textit{otherwise,} \end{cases} \end{aligned}$$

where $\eta = (\|r\|_2)/\|\hat{x}_M\|_2$, $C = I - (r\, r^T)/r^T\, r$, and $\sigma_{\min}([M \quad \eta\, C])$ is the smallest singular value of $[M \quad \eta\, C]$.

According to Theorem 1.1, $\hat{x}_M$ is a backward stable solution (and hence a stable solution) if $\mathcal{E}\,(\hat{x}_M)$ is small. However, Theorem 1.1 does not say whether $\hat{x}_M$ is a stable solution if $\mathcal{E}\,(\hat{x}_M)$ is not small. Although Waldén, Karlson, and Sun [20] have also considered perturbations in h, they have not solved Problem $(\mathcal{P})$.

Another problem with Theorem 1.1 is that while $\mathcal{E}\,(\hat{x}_M)$ is optimal, it is not very straightforward to compute for large m. Since the matrix $[M \quad \eta\, C]$ is m-by-$(m+n)$, computing its smallest singular value with a dense singular value decomposition (SVD) routine such as `xGESVD` in LAPACK [1] requires $O(m^3)$ floating point operations. Furthermore, since η can be very large for $\hat{x}_M \approx 0$, there could be some numerical difficulties in computing $\mathcal{E}\,(\hat{x}_M)$ accurately as well.

1.2. Main results. We provide an alternative F-norm bound on $\widehat{\delta M}$ that is easier to compute and that differs from $\mathcal{E}\left(\hat{x}_M\right)$ by at most a factor less than 2. To solve Problem $(\mathcal{P})$ completely, we further show that a stable solution in the sense of (1.2) is necessarily a backward stable solution in the sense of (1.3). Hence, any stable least squares algorithm is necessarily backward stable, and a numerical solution $\hat{x}_M$ is a stable solution in the sense of (1.2) if and only if $\mathcal{E}\left(\hat{x}_M\right)$ is small.

The aforementioned alternative F-norm bound on $\widehat{\delta M}$ requires $O(mn^2)$ floating point operations to compute. For $m \gg n$, this cost is much less than the $O(m^3)$ operations required for the SVD of $[M \quad \eta\, C]$ in Theorem 1.1. This bound was used extensively in our numerical comparison of various fast methods for solving structured linear least squares problems (see Gu [9]). On the other hand, we note that our new bound still costs much more than $O(mn)$ operations, hence it may not be suitable for run-time stability verification of fast methods for the linear least squares problem.

2. Alternative backward perturbation bounds. In this section, we express our results in terms of the SVD of M. While it is possible to express these results without the SVD, the resulting expressions are more complicated.

Let $M = Q \left(\begin{smallmatrix} D \\ 0 \end{smallmatrix} \right) W^T$ be the SVD of M, where $Q \in \mathbf{R}^{m\times m}$ and $W \in \mathbf{R}^{n\times n}$ are orthogonal, and $D \in \mathbf{R}^{n\times n}$ is a non-negative diagonal. Rewrite

$$h = Q \begin{pmatrix} h_1 \\ h_2 \end{pmatrix} \quad \text{and} \quad r = h - M\, \hat{x}_M = Q \begin{pmatrix} r_1 \\ r_2 \end{pmatrix},$$

where h_1 and $r_1 = h_1 - D\ (W^T\ \hat{x}_M) \in \mathbf{R}^n$ and $h_2 = r_2$. It is well known that $\gamma := \|r_2\|_2 = \|h - M\, x_M\|_2$, $h_1 = D\ (W^T\ x_M)$ and that $r_1 = 0$ if $\hat{x}_M = x_M$.

THEOREM 2.1. *Let* $\eta = (\|r\|_2)/\|\hat{x}_M\|_2$ *and define*[1]

$$\tilde{\sigma} = \sqrt{\frac{r_1^T\ D^2\ \left(D^2 + \eta^2 I\right)^{-1}\ r_1}{\gamma^2/\eta^2 + \eta^2\ r_1^T\ \left(D^2 + \eta^2 I\right)^{-2}\ r_1}} \quad \textit{and} \quad \tilde{\mathcal{E}}\left(\hat{x}_M\right) = \begin{cases} \dfrac{\|D\, r_1\|_2}{\|r\|_2} & \textit{if } \hat{x}_M = 0, \\ \min\left(\eta, \tilde{\sigma}\right) & \textit{otherwise.} \end{cases}$$

Then

$$1 \le \frac{\tilde{\mathcal{E}}\left(\hat{x}_M\right)}{\mathcal{E}\left(\hat{x}_M\right)} \le \frac{\sqrt{5}+1}{2}.$$

Proof. Theorem 2.1 obviously holds for $\hat{x}_M = 0$. Hence in the following we assume that $\hat{x}_M \neq 0$. By definition, $\sigma_{\min}\left([M \quad \eta\, C]\right)$ is the smallest non-negative σ such that

$$f(\sigma) := \det\Big(\left([M \quad \eta\, C]\right)\ \left([M \quad \eta\, C]\right)^T - \sigma^2 I\Big) = 0.$$

Replacing M by its singular value decomposition and simplifying,

$$f(\sigma) = \det\left(M\ M^T + \eta\, C^2 - \sigma^2 I\right) = \det\left(M\ M^T + (\eta^2 - \sigma^2)\ I - \eta^2\, \frac{r\, r^T}{\|r\|_2^2}\right)$$

[1] It is easy to check that $\tilde{\mathcal{E}}\left(\hat{x}\right)$ is continuous at $\hat{x} = 0$:

$$\tilde{\mathcal{E}}\left(0\right) = \lim_{\hat{x}\to 0} \tilde{\mathcal{E}}\left(\hat{x}\right) = \frac{\|M^T\ h\|_2}{\|h\|_2}\ .$$

$$= \left(\eta^2 - \sigma^2\right)^{m-n-1} \det\left(\begin{pmatrix} D^2 & 0 \\ 0 & 0 \end{pmatrix} + \left(\eta^2 - \sigma^2\right) I - \frac{\eta^2}{\|r\|_2^2} \begin{pmatrix} r_1 \\ \gamma \end{pmatrix} \begin{pmatrix} r_1^T & \gamma \end{pmatrix}\right)$$
$$= \left(\eta^2 - \sigma^2\right)^{m-n} \det\left(D^2 + \left(\eta^2 - \sigma^2\right) I\right)$$
$$\cdot \left(1 - \frac{\eta^2\, \gamma^2}{\|r\|_2^2\, (\eta^2 - \sigma^2)} - \frac{\eta^2}{\|r\|_2^2}\, r_1^T\, \left(D^2 + \left(\eta^2 - \sigma^2\right) I\right)^{-1}\, r_1\right).$$

Hence, $\sigma_{\min}\left([M \quad \eta\, C]\right)$ is the smallest non-negative $\sigma < \eta$ such that

$$1 - \frac{\eta^2\, \gamma^2}{\|r\|_2^2\, (\eta^2 - \sigma^2)} - \frac{\eta^2}{\|r\|_2^2}\, r_1^T\, \left(D^2 + \left(\eta^2 - \sigma^2\right) I\right)^{-1}\, r_1 = 0. \tag{2.1}$$

This equation can be rewritten as

$$1 - \frac{\eta^2\, \gamma^2}{\|r\|_2^2\, \eta^2} - \frac{\eta^2}{\|r\|_2^2}\, r_1^T\, \left(D^2 + \eta^2\, I\right)^{-1}\, r_1$$
$$= \frac{\eta^2\, \gamma^2}{\|r\|_2^2\, (\eta^2 - \sigma^2)} - \frac{\eta^2\, \gamma^2}{\|r\|_2^2\, \eta^2} + \frac{\eta^2}{\|r\|_2^2}\, r_1^T\, \left(D^2 + \left(\eta^2 - \sigma^2\right) I\right)^{-1}\, r_1$$
$$-\frac{\eta^2}{\|r\|_2^2}\, r_1^T\, \left(D^2 + \eta^2\, I\right)^{-1}\, r_1.$$

Since $\|r\|_2^2 = \|r_1\|_2^2 + \gamma^2$, the above equation can be simplified, after some algebra, into

$$\sigma^2 = \frac{r_1^T\, D^2\, \left(D^2 + \eta^2 I\right)^{-1}\, r_1}{\gamma^2/(\eta^2 - \sigma^2) + \eta^2\, r_1^T\, \left(D^2 + (\eta^2 - \sigma^2) I\right)^{-1}\, \left(D^2 + \eta^2 I\right)^{-1}\, r_1}. \tag{2.2}$$

We note that the expression on the right-hand side is $\tilde{\sigma}^2$ if $\sigma = 0$. Since $\gamma^2/\eta^2 \leq \gamma^2/(\eta^2 - \sigma^2)$ and

$$r_1^T\, \left(D^2 + \eta^2 I\right)^{-2}\, r_1 \leq r_1^T\, \left(D^2 + (\eta^2 - \sigma^2) I\right)^{-1}\, \left(D^2 + \eta^2 I\right)^{-1}\, r_1$$

for $\sigma \leq \eta$, (2.2) implies that $\sigma_{\min}\left([M \quad \eta\, C]\right) \leq \tilde{\sigma}$. It follows that $\mathcal{E}\left(\hat{x}_M\right) \leq \tilde{\mathcal{E}}\left(\hat{x}_M\right)$.

We now assume that $\tilde{\sigma} > \eta$. In this case we have $\tilde{\mathcal{E}}\left(\hat{x}_M\right) = \eta$. We claim that

$$\sigma_{\min}\left([M \quad \eta\, C]\right) \geq \beta\, \eta, \quad \text{where} \quad \beta = \frac{\sqrt{5} - 1}{2}. \tag{2.3}$$

We show this by contradiction. Assume that this was false, so that $\sigma_{\min}\left([M \quad \eta\, C]\right) < \beta\, \eta$. We note that $\gamma^2/\eta^2 > (1 - \beta^2)\, \gamma^2/(\eta^2 - \sigma^2)$ and that

$$r_1^T\, \left(D^2 + \eta^2 I\right)^{-2}\, r_1 > (1 - \beta^2)\, r_1^T\, \left(D^2 + (\eta^2 - \sigma^2) I\right)^{-1}\, \left(D^2 + \eta^2 I\right)^{-1}\, r_1$$

for $\sigma < \beta\, \eta$. Equation (2.2) now implies that

$$\tilde{\sigma} < \frac{\sigma_{\min}\left([M \quad \eta\, C]\right)}{1 - \beta^2} < \frac{\beta\, \eta}{1 - \beta^2} = \eta,$$

which is a contradiction. Hence, relation (2.3) is indeed valid and we have

$$\beta\, \eta \leq \mathcal{E}\left(\hat{x}_M\right) \leq \eta.$$

So Theorem 2.1 holds in this case.

We further consider the case where $\tilde{\sigma} \leq \eta$. In this case we have $\tilde{\mathcal{E}}\left(\hat{x}_M\right) = \tilde{\sigma}$ and $\beta := \sigma_{\min}\left([M \quad \eta\, C]\right)/\tilde{\sigma} \leq 1$. Similar to the above we have

$$\tilde{\sigma} \leq \frac{\sigma_{\min}\left([M \quad \eta\, C]\right)}{1-\beta^2} \leq \frac{\tilde{\sigma}\,\beta}{1-\beta^2},$$

which simplifies to

$$1-\beta^2 \leq \beta \quad \text{or} \quad \beta \geq \frac{\sqrt{5}-1}{2}.$$

It follows that

$$\frac{\sqrt{5}-1}{2}\,\tilde{\sigma} \leq \mathcal{E}\left(\hat{x}_M\right) \leq \tilde{\sigma}.$$

So Theorem 2.1 holds in this case as well. □

Hence $\tilde{\mathcal{E}}\left(\hat{x}_M\right)$ differs from the smallest possible backward perturbation $\mathcal{E}\left(\hat{x}_M\right)$ by a factor of at most $(\sqrt{5}+1)/2 < 2$. To compute $\tilde{\mathcal{E}}\left(\hat{x}_M\right)$, we only need to compute D (the singular values of M) and $Q^T\; r$, and we do not need to compute Q and W explicitly. This computation can be done, for example, by using the subroutines `xGESVD` in LAPACK [1]. Since the matrix M is m-by-n, this computation requires $O(mn^2)$ floating point operations, which is much cheaper than $O(m^3)$ for $m \gg n$.

Equation (2.1) provides an efficient way to compute $\sigma_{\min}\left([M \quad \eta\, C]\right)$ (and hence $\mathcal{E}\left(\hat{x}_M\right)$) as well. In fact, (2.1) is similar to the *secular equations* solved in Gu and Eisenstat [10] and Li [13], and their methods can be easily modified to compute $\sigma_{\min}\left([M \quad \eta\, C]\right)$. Both computing the SVD of M and solving (2.1) can be done reliably.

Recently, Karlson and Waldén [12] discussed a method for estimating $\mathcal{E}\left(\hat{x}_M\right)$ that requires $O(mn)$ floating point operations. While their method is more efficient than ours, it can be unreliable in that its estimates can differ from $\mathcal{E}\left(\hat{x}_M\right)$ by an arbitrary factor [12]. In contrast, our estimate $\tilde{\mathcal{E}}\left(\hat{x}_M\right)$ is guaranteed to differ from $\mathcal{E}\left(\hat{x}_M\right)$ by a factor less than 2.

In the rest of this section we analyze $\mathcal{E}\left(\hat{x}_M\right)$ for several special cases.

COROLLARY 2.2. *Assume that* $\|r_1\|_2 \leq \alpha\,\gamma$. *Define*

$$\tilde{\sigma}_1 = \frac{\sqrt{r_1^T\; D^2\; \left(D^2+\eta^2 I\right)^{-1}\; r_1}}{\|\hat{x}_M\|_2}.$$

Then

$$\frac{1}{\sqrt{1+\alpha^2}} \leq \frac{\tilde{\sigma}_1}{\mathcal{E}\left(\hat{x}_M\right)} \leq \frac{\sqrt{5}+1}{2}.$$

Proof. Since $\|r\|_2^2 = \|r_1\|_2^2 + \gamma^2$ and $\eta^2 = \|r\|_2^2/\|\hat{x}_M\|_2^2$, the assumption implies that

$$\begin{aligned}\frac{\|\hat{x}_M\|_2^2}{1+\alpha^2} \leq \frac{\gamma^2}{\eta^2} &\leq \frac{\gamma^2}{\eta^2} + \eta^2\, r_1^T\; \left(D^2+\eta^2 I\right)^{-2}\; r_1 \\ &\leq \frac{\gamma^2}{\eta^2} + \frac{\eta^2\, r_1^T\; r_1}{\eta^4} = \frac{\|r\|_2^2}{\eta^2} \\ &= \|\hat{x}_M\|_2^2.\end{aligned}$$

We also have

$$\frac{\tilde{\sigma}_1}{\tilde{\sigma}} = \frac{\sqrt{\frac{\gamma^2}{\eta^2} + \eta^2\, r_1^T\, \left(D^2 + \eta^2 I\right)^{-2}\, r_1}}{\|\hat{x}_M\|_2}.$$

Consequently,

$$\frac{1}{\sqrt{1+\alpha^2}} \le \frac{\tilde{\sigma}_1}{\tilde{\sigma}} \le 1.$$

Corollary 2.2 follows by combining the above relations with Theorem 2.1 and the fact that $\tilde{\sigma}_1 \le \eta$. □

The least squares problem (1.1) has a small residual if $\gamma = \|h - M\, x_M\|_2 \approx 0$ and large residual otherwise; and $\hat{x}_M = x_M$ if and only if $r_1 = 0$. Since a good approximate solution $\hat{x}_M$ always makes r_1 small, Corollary 2.2 implies that, for large residual problems, $\hat{x}_M$ is a backward stable solution if and only if $\tilde{\sigma}_1$ is small.

Corollary 2.3 below gives a backward perturbation bound for small residual problems.

COROLLARY 2.3. *Assume that*

$$\|r_1\|_2 \ge \alpha\, \gamma \quad \textit{and} \quad \eta \le \sigma_{\min}(M).$$

Then

$$\frac{\sqrt{5}-1}{2}\sqrt{\frac{2\alpha^2}{4+\alpha^2}}\,\eta \le \mathcal{E}\left(\hat{x}_M\right) \le \eta.$$

Proof. Let $\beta = 2\alpha^2/(4+\alpha^2)$ be a scalar. Then,

$$\begin{aligned}\tilde{\sigma}^2 - \beta\,\eta^2 &= \frac{r_1^T\, D^2\, \left(D^2+\eta^2 I\right)^{-1}\, r_1}{\gamma^2/\eta^2 + \eta^2\, r_1^T\, \left(D^2+\eta^2 I\right)^{-2}\, r_1} - \beta\,\eta^2 \\ &= \frac{r_1^T\, D^2\, \left(D^2+\eta^2 I\right)^{-1}\, r_1 - \beta\,\gamma^2 - \beta\,\eta^4\, r_1^T\, \left(D^2+\eta^2 I\right)^{-2}\, r_1}{\gamma^2/\eta^2 + \eta^2\, r_1^T\, \left(D^2+\eta^2 I\right)^{-2}\, r_1}.\end{aligned}$$

Since $\eta \le \sigma_{\min}(M) = \sigma_{\min}(D)$, we have

$$r_1^T\, D^2\, \left(D^2+\eta^2 I\right)^{-1}\, r_1 \ge \frac{\|r_1\|_2^2}{2} \quad \text{and} \quad \eta^4\, r_1^T\, \left(D^2+\eta^2 I\right)^{-2}\, r_1 \le \frac{\|r_1\|_2^2}{4}.$$

Combining these relations and simplifying,

$$\tilde{\sigma}^2 - \beta\,\eta^2 \ge \frac{\|r_1\|_2^2/2 - \beta\,\gamma^2 - \beta\,\|r_1\|_2^2/4}{\gamma^2/\eta^2 + \eta^2\, r_1^T\, \left(D^2+\eta^2 I\right)^{-2}\, r_1} \ge \frac{\alpha^2\,\gamma^2/2 - \beta\,\gamma^2 - \beta\,\alpha^2\,\gamma^2/4}{\gamma^2/\eta^2 + \eta^2\, r_1^T\, \left(D^2+\eta^2 I\right)^{-2}\, r_1} = 0.$$

It follows that $\tilde{\sigma}^2 \ge \beta\,\eta^2$ and that

$$\sqrt{\beta}\,\eta \le \tilde{\mathcal{E}}\left(\hat{x}_M\right) \le \eta.$$

Corollary 2.3 follows by combining this relation with Theorem 2.1. □

3. A stable solution is a backward stable solution. In this section we solve Problem ($\mathcal{P}$) by showing that a stable solution in the sense of (1.2) is a backward stable solution in the sense of (1.3). Hence a numerical solution $\hat{x}_M$ is a stable solution in the sense of (1.2) if and only if $\mathcal{E}\,(\hat{x}_M)$ is small. Theorem 3.1 below, together with Theorem 2.1, formed the basis of our study on the numerical accuracy of the various methods for solving structured linear least squares problems (see Gu [9]).

To prove Theorem 3.1 below, we need a certain amount of notation. In (1.2), let the SVD of $M + \delta M$ be $\hat{Q}\ (\begin{smallmatrix} \hat{D} \\ 0 \end{smallmatrix})\ \hat{W}^T$. Define $\widehat{\delta x}_M = \hat{W}^T\ \delta\hat{x}_M$,

$$\hat{Q}^T\ h = \begin{pmatrix} \hat{h}_1 \\ \hat{h}_2 \end{pmatrix}, \quad \delta\hat{h} = \hat{Q}^T\ \delta h = \begin{pmatrix} \delta\hat{h}_1 \\ \delta\hat{h}_2 \end{pmatrix}, \quad \text{and} \quad \hat{r} = h - (M + \delta M)\ \hat{x}_M,$$

where $\hat{h}_1$ and $\delta\hat{h}_1 \in \mathbf{R}^n$ and $\hat{h}_2$ and $\delta\hat{h}_2 \in \mathbf{R}^{(m-n)}$. We also define

$$\hat{\eta} = \frac{\|\hat{r}\|_2}{\|\hat{x}_M\|_2} \quad \text{and} \quad \hat{r}_1 = \hat{D}\ \widehat{\delta x}_M - \delta\hat{h}_1.$$

We adopt the convention that $\|\delta\hat{x}_M\|_2/\|\hat{x}_M\|_2 = 0$ if $\delta\hat{x}_M = 0$ and $\hat{x}_M = 0$.

THEOREM 3.1. *In* (1.2) *let δM, δb, and $\delta\hat{x}_M$ be small perturbations of M, b, and $\hat{x}_M$, respectively. Then there exists a matrix $\widehat{\delta M} \in \mathbf{R}^{m\times n}$ satisfying* (1.3) *with*

$$\frac{\|\widehat{\delta M}\|_2}{\|M\|_2} \le \frac{\|\delta M\|_2}{\|M\|_2} + 2\left(1 + \frac{\|\delta M\|_2}{\|M\|_2}\right)\left(\frac{\|\delta\hat{x}_M\|_2}{\|\hat{x}_M\|_2} + \frac{\|\delta h\|_2}{\|h\|_2}\right)\Bigg/\left(1 - 2\,\frac{\|\delta h\|_2}{\|h\|_2}\right).$$

Proof. We prove this theorem by applying backward perturbation bounds in section 2 to $M + \delta M$. We note that

$$(h + \delta h) - (M + \delta M)\ (\hat{x}_M + \delta\hat{x}_M) = \hat{Q}\ \begin{pmatrix} 0 \\ \hat{h}_2 + \delta\hat{h}_2 \end{pmatrix}$$

and that

$$(M + \delta M)\ \delta\hat{x}_M - \delta h = \hat{Q}\ \begin{pmatrix} \hat{D}\ \widehat{\delta x}_M - \delta\hat{h}_1 \\ -\delta\hat{h}_2 \end{pmatrix},$$

where we have used the fact that $\hat{x}_M + \delta\hat{x}_M$ is the exact solution to the perturbed least squares problem (1.2). Hence, we can rewrite $\hat{r}$ as

$$\begin{aligned} \hat{r} &= ((h + \delta h) - (M + \delta M)\ (\hat{x}_M + \delta\hat{x}_M)) + ((M + \delta M)\ \delta\hat{x}_M - \delta h) \\ &= \hat{Q}\ \begin{pmatrix} \hat{D}\ \widehat{\delta x}_M - \delta\hat{h}_1 \\ \hat{h}_2 \end{pmatrix}. \end{aligned}$$

In the following we derive an upper bound on $\mathcal{E}\,(\hat{x}_M)$ with $M + \delta M$ as the coefficient matrix in the least squares problem.

We first assume that $\|\hat{h}_2\|_2 \le \|\hat{r}_1\|_2$. By Theorem 2.1,

$$\begin{aligned} \mathcal{E}\,(\hat{x}_M) \le \hat{\eta} = \frac{\|\hat{r}\|_2}{\|\hat{x}_M\|_2} &\le \sqrt{2}\ \frac{\|\hat{r}_1\|_2}{\|\hat{x}_M\|_2} \le \sqrt{2}\ \frac{\|\hat{D}\ \widehat{\delta x}_M\|_2 + \|\delta\hat{h}_1\|_2}{\|\hat{x}_M\|_2} \\ &\le \sqrt{2}\left(\|\hat{D}\|_2\ \frac{\|\widehat{\delta x}_M\|_2}{\|\hat{x}_M\|_2} + \frac{\|\delta\hat{h}_1\|_2}{\|\hat{x}_M\|_2}\right). \end{aligned} \tag{3.1}$$

Since $\hat{x}_M + \delta\hat{x}_M$ is the exact solution to (1.2), it follows that

$$\hat{D}\ W^T\ (\hat{x}_M + \delta\hat{x}_M) = \hat{h}_1 + \delta\hat{h}_1,$$

and hence

$$\|\hat{h}_1 + \delta\hat{h}_1\|_2 \leq \|\hat{D}\|_2\ \|\hat{x}_M + \delta\hat{x}_M\|_2 \leq \|\hat{D}\|_2\ \|\hat{x}_M\|_2\ \left(1 + \frac{\|\delta\hat{x}_M\|_2}{\|\hat{x}_M\|_2}\right). \tag{3.2}$$

On the other hand,

$$\hat{Q}^T\ h = \begin{pmatrix} \hat{h}_1 \\ \hat{h}_2 \end{pmatrix} = \begin{pmatrix} \hat{h}_1 + \delta\hat{h}_1 \\ 0 \end{pmatrix} + \begin{pmatrix} -\delta\hat{h}_1 \\ \hat{h}_2 \end{pmatrix}.$$

Taking 2-norms on both sides, and noting that $\|\delta\hat{h}_1\|_2 \leq \|\delta h\|_2$,

$$\begin{aligned}\|h\|_2 &\leq \left\|\hat{h}_1 + \delta\hat{h}_1\right\|_2 + \left\|\delta\hat{h}_1\right\|_2 + \left\|\hat{h}_2\right\|_2 \leq \left\|\hat{h}_1 + \delta\hat{h}_1\right\|_2 + \|\delta h\|_2 + \|\hat{r}_1\|_2 \\ &= \left\|\hat{h}_1 + \delta\hat{h}_1\right\|_2 + \|\delta h\|_2 + \left\|\hat{D}\ \widehat{\delta x}_M - \delta\hat{h}_1\right\|_2.\end{aligned}$$

Plugging in the 2-norm upper bound (3.2) and simplifying, we get

$$\frac{\|h\|_2}{\|\hat{x}_M\|_2} \leq \|\hat{D}\|_2\ \left(1 + 2\ \frac{\|\delta\hat{x}_M\|_2}{\|\hat{x}_M\|_2}\right) \Big/ \left(1 - 2\ \frac{\|\delta h\|_2}{\|h\|_2}\right).$$

In (3.1) we have

$$\begin{aligned}\mathcal{E}\left(\hat{x}_M\right) &\leq \sqrt{2}\ \left(\|\hat{D}\|_2\ \frac{\|\widehat{\delta x}_M\|_2}{\|\hat{x}_M\|_2} + \frac{\|\delta\hat{h}_1\|_2}{\|h\|_2}\ \frac{\|h\|_2}{\|\hat{x}_M\|_2}\right) \\ &\leq \sqrt{2}\ \left(\|\hat{D}\|_2\ \frac{\|\widehat{\delta x}_M\|_2}{\|\hat{x}_M\|_2} + \frac{\|\delta h\|_2}{\|h\|_2}\ \frac{\|h\|_2}{\|\hat{x}_M\|_2}\right).\end{aligned}$$

Plugging in the upper bound on $(\|h\|_2)/\|\hat{x}_M\|_2$ and simplifying, we obtain

$$\begin{aligned}\mathcal{E}\left(\hat{x}_M\right) &\leq \sqrt{2}\ (\|M\|_2 + \|\delta M\|_2)\ \left(\frac{\|\delta\hat{x}_M\|_2}{\|\hat{x}_M\|_2} + \frac{\|\delta h\|_2}{\|h\|_2}\right) \Big/ \left(1 - 2\ \frac{\|\delta h\|_2}{\|h\|_2}\right) \\ &\leq 2\ (\|M\|_2 + \|\delta M\|_2)\ \left(\frac{\|\delta\hat{x}_M\|_2}{\|\hat{x}_M\|_2} + \frac{\|\delta h\|_2}{\|h\|_2}\right) \Big/ \left(1 - 2\ \frac{\|\delta h\|_2}{\|h\|_2}\right), \qquad (3.3)\end{aligned}$$

where we have used the facts that

$$\|\hat{D}\|_2 = \|M + \delta M\|_2 \leq \|M\|_2 + \|\delta M\|_2 \quad \text{and} \quad \|\widehat{\delta x}_M\|_2 = \|\delta\hat{x}_M\|_2.$$

Now we assume that $\|\hat{h}_2\|_2 \geq \|\hat{r}_1\|_2$. By Corollary 2.2 we have $\mathcal{E}\left(\hat{x}_M\right) \leq \sqrt{2}\ \tilde{\sigma}_1$, where

$$\tilde{\sigma}_1 = \frac{\sqrt{\hat{r}_1^T\ \hat{D}^2\ \left(\hat{D}^2 + \hat{\eta}^2 I\right)^{-1}\ \hat{r}_1}}{\|\hat{x}_M\|_2}.$$

Since $\hat r_1 = \hat D\,\widehat{\delta x}_M - \delta\hat h_1$, it follows that

$$\begin{aligned}\tilde\sigma_1 &\le \frac{\sqrt{\left(\hat D\,\widehat{\delta x}_M\right)^T\,\hat D^2\,\left(\hat D^2+\hat\eta^2 I\right)^{-1}\,\left(\hat D\,\widehat{\delta x}_M\right)}}{\|\hat x_M\|_2} + \frac{\sqrt{\delta\hat h_1^T\,\hat D^2\,\left(\hat D^2+\hat\eta^2 I\right)^{-1}\,\delta\hat h_1}}{\|\hat x_M\|_2}\\ &\le \|\hat D\|_2\,\frac{\|\widehat{\delta x}_M\|_2}{\|\hat x_M\|_2} + \frac{\sqrt{\delta\hat h_1^T\,\hat D^2\,\left(\hat D^2+\hat\eta^2 I\right)^{-1}\,\delta\hat h_1}}{\|\hat x_M\|_2}.\end{aligned}$$

Since $\|\hat r\|_2 = \hat\eta\,\|\hat x_M\|_2$, it follows from the above relation that

$$\tilde\sigma_1 \le \|\hat D\|_2\,\frac{\|\delta\hat x_M\|_2}{\|\hat x_M\|_2} + \frac{\|\delta\hat h_1\|_2}{\|\hat x_M\|_2} \quad\text{and}\quad \tilde\sigma_1 \le \|\hat D\|_2\,\frac{\|\delta\hat x_M\|_2}{\|\hat x_M\|_2} + \frac{\|D\,\delta\hat h_1\|_2}{\|\hat r\|_2}.$$

Combining these with relation (3.2) we obtain

$$\begin{aligned}\tilde\sigma_1 &\le \|\hat D\|_2\,\frac{\|\delta\hat x_M\|_2}{\|\hat x_M\|_2} + \min\left(\frac{\|\delta h\|_2}{\|\hat h_1+\delta\hat h_1\|_2}, \frac{\|\delta h\|_2}{\|\hat r\|_2}\right)\,\|\hat D\|_2\,\left(1+\frac{\|\delta\hat x_M\|_2}{\|\hat x_M\|_2}\right)\\ &= \|\hat D\|_2\,\frac{\|\delta\hat x_M\|_2}{\|\hat x_M\|_2} + \frac{\|\delta h\|_2}{\max\left(\|\hat h_1+\delta\hat h_1\|_2, \|\hat r\|_2\right)}\,\|\hat D\|_2\,\left(1+\frac{\|\delta\hat x_M\|_2}{\|\hat x_M\|_2}\right).\end{aligned}$$

Since $\|\hat r\|_2 \ge \|\hat h_2\|_2$, it follows that

$$\begin{aligned}\max\left(\|\hat h_1+\delta\hat h_1\|_2, \|\hat r\|_2\right) &\ge \max\left(\|\hat h_1+\delta\hat h_1\|_2, \|\hat h_2\|_2\right) \ge \frac{1}{\sqrt 2}\,\sqrt{\|\hat h_1+\delta\hat h_1\|_2^2+\|\hat h_2\|_2^2}\\ &= \frac{1}{\sqrt 2}\,\left\|\begin{pmatrix}\hat h_1+\delta\hat h_1\\ \hat h_2\end{pmatrix}\right\|_2 \ge \frac{1}{\sqrt 2}\,(\|h\|_2-\|\delta h\|_2).\end{aligned}$$

Consequently,

$$\tilde\sigma_1 \le \|\hat D\|_2\,\frac{\|\delta\hat x_M\|_2}{\|\hat x_M\|_2} + \frac{\sqrt 2\,\|\delta h\|_2}{\|h\|_2-\|\delta h\|_2}\,\|\hat D\|_2\,\left(1+\frac{\|\delta\hat x_M\|_2}{\|\hat x_M\|_2}\right).$$

From this relation we get

$$\begin{aligned}\mathcal E\left(\hat x_M\right) &\le 2\left(\|M\|_2+\|\delta M\|_2\right)\,\left(\frac{\|\delta\hat x_M\|_2}{\|\hat x_M\|_2}+\frac{\|\delta h\|_2}{\|h\|_2}\right)\Big/\left(1-\frac{\|\delta h\|_2}{\|h\|_2}\right)\\ &\le 2\left(\|M\|_2+\|\delta M\|_2\right)\,\left(\frac{\|\delta\hat x_M\|_2}{\|\hat x_M\|_2}+\frac{\|\delta h\|_2}{\|h\|_2}\right)\Big/\left(1-2\,\frac{\|\delta h\|_2}{\|h\|_2}\right),\end{aligned}$$

which is identical to (3.3).

In both cases, there exists a matrix $\widehat{\delta M}_1 \in \mathbf R^{m\times n}$ with $\|\widehat{\delta M}_1\|_F = \mathcal E\left(\hat x_M\right)$ such that

$$\|(M+\delta M+\widehat{\delta M}_1)\,\hat x_M - h\|_2 = \min_x \|(M+\delta M+\widehat{\delta M}_1)\,x-h\|_2.$$

Now we define $\widehat{\delta M} = \delta M + \widehat{\delta M}_1$. It follows that

$$\|(M+\widehat{\delta M})\,\hat x_M - h\|_2 = \min_x \|(M+\widehat{\delta M})\,x-h\|_2 \quad\text{and}\quad \|\widehat{\delta M}\|_2 \le \|\delta M\|_2 + \mathcal E\left(\hat x_M\right).$$

The theorem follows immediately by plugging the upper bound (3.3) into this relation. □

Acknowledgments. The author is grateful to the anonymous referees for many suggestions that improved the presentation of the paper and for pointing out the paper [12] to him.

REFERENCES

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen, *LAPACK Users' Guide*, 2nd ed., SIAM, Philadelphia, PA, 1994.

[2] A. Bojanczyk and R. P. Brent, *Parallel solution of certain Toeplitz least-squares problems*, Linear Algebra Appl., 77 (1986), pp. 43–60.

[3] A. W. Bojanczyk, R. P. Brent, and F. de Hoog, *QR factorization of Toeplitz matrices*, Numer. Math., 49 (1986), pp. 81–94.

[4] R. P. Brent, *Parallel algorithms for Toeplitz systems*, in Numerical Linear Algebra, Digital Signal Processing and Parallel Algorithms, G. H. Golub and P. Van Dooren, eds., Springer, 1990.

[5] P. A. Businger and G. H. Golub, *Linear least squares solutions by Householder transformations*, Numer. Math., 7 (1965), pp. 269–276.

[6] J. Chun, T. Kailath, and H. Lev-Ari, *Fast parallel algorithms for QR and triangular factorization*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 899–913.

[7] G. Cybenko, *A general orthorgonalization technique with applications to time series analysis and signal processing*, Math. Comp., 40 (1983), pp. 323–336.

[8] G. Cybenko, *Fast Toeplitz orthorgonalization using inner products*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 734–740.

[9] M. Gu, *New fast algorithms for structured linear least squares problems*, SIAM J. Matrix Anal. Appl., 20 (1998), pp. 244–269.

[10] M. Gu and S. C. Eisenstat, *A divide-and-conquer algorithm for the bidiagonal SVD*, SIAM J. Matrix Anal. Appl., 16 (1995), pp. 79–92.

[11] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia, PA, 1996.

[12] R. Karlson and B. Waldén, *Practical estimation of optimal backward perturbation bounds for the linear least square problem*, BIT, 1996, submitted.

[13] R.-C. Li, *Solving Secular Equations Stably and Efficiently*, manuscript, October 1992.

[14] F. T. Luk and S. Qiao, *A fast but unstable orthogonal triangularization technique for Toeplitz matrices*, Linear Algebra Appl., 88/89 (1987), pp. 495–506.

[15] J. G. Nagy, *Fast inverse QR factorization for Toeplitz matrices*, SIAM J. Sci. Statist. Comput., 14 (1993), pp. 1174–1183.

[16] H. Park and L. Eldén, *Stability Analysis and Fast Algorithms for Triangularization of Toeplitz Matrices*, Tech. report Lith-Mat-R-95-16, Department of Mathematics, Linköping University, Sweden, May 1995.

[17] S. Qiao, *Hybrid algorithm for fast Toeplitz orthogonalization*, Numer. Math., 53 (1988), pp. 351–366.

[18] G. W. Stewart, *Introduction to Matrix Computations*, Academic Press, New York, 1973.

[19] D. R. Sweet, *Fast Toeplitz orthogonalization*, Numer. Math., 43 (1984), pp. 1–21.

[20] B. Waldén, R. Karlson, and J.-G. Sun, *Optimal backward perturbation bounds for the linear least square problem*, Numer. Linear Alg. Appl., 2 (1995), pp. 271–286.

SIAM J. Matrix Anal. Appl.
Vol. 20, No. 2, pp. 373–399

ACCURATELY COUNTING SINGULAR VALUES OF BIDIAGONAL MATRICES AND EIGENVALUES OF SKEW-SYMMETRIC TRIDIAGONAL MATRICES[*]

K. V. FERNANDO[†]

This paper is dedicated to Gene H. Golub and William Kahan.

Abstract. We have developed algorithms to count singular values of a real bidiagonal matrix which are greater than a specified value. This requires the transformation of the singular value problem to an equivalent symmetric eigenvalue problem. The counting of singular values is paramount in the design of bisection- and multisection-type algorithms for computing singular values on serial and parallel machines.

The algorithms are based on the eigenvalues of BB^t, B^tB, and the $2n \times 2n$ zero–diagonal tridiagonal matrix which is permutationally equivalent to the Jordan–Wielandt form $\left[\begin{smallmatrix} 0 & B \\ B^t & 0 \end{smallmatrix}\right]$, where B is an $n \times n$ bidiagonal matrix. The two product matrices, which do not have to be formed explicitly, lead to the progressive and stationary qd algorithms of Rutishauser. The algorithm based on the zero–diagonal matrix, which we have named the Golub–Kahan form, may be considered as a combination of both the progressive and stationary qd algorithms.

We study important properties such as the backward error analysis, the monotonicity of the inertia count and the scaling of data which guarantee the accuracy and integrity of these algorithms. For high relative accuracy of tiny singular values, the algorithm based on the Golub–Kahan form is the best choice. However, if such accuracy is not required or requested, the differential progressive and differential stationary qd algorithms with certain modifications are adequate and more efficient.

We also show how to transform the real skew-symmetric tridiagonal eigenvalue problem to a real bidiagonal singular value problem. Thus, the eigenvalues of a skew-symmetric matrix can be readily counted using algorithms developed here for bidiagonal matrices.

Key words. bisection, multisection, singular values, eigenvalues, monotonic arithmetic, error analysis, parallel algorithms, qd algorithms, bidiagonal matrices, symmetric tridiagonal matrices, skew-symmetric tridiagonal matrices

AMS subject classifications. 15A18, 34L15, 65F15

PII. S089547989631175X

1. Introduction.

We have developed algorithms to count the number of singular values of a bidiagonal matrix which are greater (or less) than a specified value. This requires the transformation of the singular value problem to an equivalent symmetric eigenvalue problem. The counting of singular values (or equivalent eigenvalues) is paramount in the design of bisection- and multisection-type algorithms for computing singular values. The algorithms are embarrassingly parallel and, hence, suitable for multiprocessor environments.

Bisection is one of the most powerful methods available for computing eigenvalues of a real symmetric matrix. It has been in use on serial machines since the discovery by Givens in the 1950s [11] and the bisection/multisection approach is becoming the algorithm of choice on parallel machines. Early implementations of bisection algorithms were based on the Sturm counts obtained by computing the principal minors of the shifted tridiagonal symmetric matrix $T - \tau I$ where T is a real symmetric tridiagonal matrix and τ is the shift. Chapter 5 of Wilkinson [28] and section 8.4.1 of Golub and

[*] Received by the editors November 8, 1996; accepted for publication (in revised form) by A. Edelman November 12, 1997; published electronically October 20, 1998. The preliminary work was carried out while the author was visiting Professor James Demmel and his (Sca)LAPACK group at the Electrical Engineering and Computer Science Department of the University of California at Berkeley. The work of this author was partially supported by NSF grant ASC-9005933 and ONR contract N000014-90-J-1372.

http://www.siam.org/journals/simax/20-2/31175.html

[†] NAG Ltd., Wilkinson House, Jordan Hill, Oxford OX2 8DR, UK (vince@nag.co.uk).

Van Loan [14] give further details. However, the computation of the principal minors is susceptible to overflow/underflow and other numerical problems. More recent implementations depend upon the application of Sylvester–Jacobi inertia theorem[1] to the LDL^t factorization of the shifted matrix. There are inexpensive ways to overcome overflow/underflow and other numerical problems in the factorization method as indicated by Barth, Martin, and Wilkinson [3] and unequivocally by Kahan [17]. Kahan [18] has also shown that IEEE arithmetic [15] provides a natural framework for the implementation of the continued fractions associated with these algorithms. Demmel and Gragg [5] have extended the classes of matrices which are easily amenable to bisection-type algorithms. Problems associated with bisection on parallel computers have recently been studied by Demmel, Dhillon, and Ren [7].

In this report, we study the counting mechanism to locate singular values of an $n \times n$ real bidiagonal matrix B. This is achieved by transforming the bidiagonal singular value problem to an equivalent symmetric tridiagonal eigenvalue problem. In fact, there are at least three principal ways to count singular values of a bidiagonal matrix. The first technique considers the eigenvalues of the Jordan–Wielandt form of B

$$\begin{bmatrix} 0 & B \\ B^t & 0 \end{bmatrix}, \tag{1}$$

which are given by the singular values and the negated singular values of B. Alternatively, it is possible to consider the eigenvalues of BB^t or B^tB, which are given by the squared singular values of B. The last two approaches lead to many variants of the qd algorithms of Rutishauser [10], [21], [22]. Fortunately, it is not required to form the numerically undesired matrix products BB^t and B^tB explicitly to count the eigenvalues.

We also show that eigenvalues of a real skew-symmetric tridiagonal matrix can be counted by transforming that problem to an equivalent bidiagonal singular value problem. This derived bidiagonal matrix is (approximately) half the size of the original skew-symmetric tridiagonal matrix and, hence, this is a rather economical way to count eigenvalues of a skew-symmetric tridiagonal matrix. It is well known that singular values of bidiagonal matrices can be computed to high relative precision if a suitable algorithm is used [6], [2], [10]. Because of this relationship between skew-symmetric tridiagonal and bidiagonal matrices, the high relative accuracy is also inherited by skew-symmetric tridiagonal matrices.

This report is organized as follows. In section 2, the notation is developed and the transformations between the bidiagonal singular value problem and equivalent eigenproblems are studied. The formal connection between the skew-symmetric eigenproblem and the bidiagonal singular value problem is also discussed.

In section 3, the three main algorithms for computing singular values are presented. The first and the second are the stationary qd algorithm and the progressive qd algorithm of Rutishauser. The third is based on a $2n \times 2n$ zero–diagonal tridiagonal matrix which we have named the *Golub–Kahan* form to honor their discovery and the use of this matrix in [13]. This matrix is permutationally equivalent to (1). The qd algorithms are less expensive than the algorithm for the Golub–Kahan form as there are only half the number of divisions, (which is the most expensive out of the four basic arithmetic operations). The connections between these algorithms are also highlighted. In particular, the continued fraction expansion associated with the

[1] Kahan [17] attributes this LDL^t factorization approach to Boris Davison, 1959.

Golub–Kahan form can be separated into odd and even continued fractions.[2] The differential stationary algorithm represents the odd branch while the differential progressive algorithm matches the even branch.

Section 4 studies effects due to finite precision. The Rutishauser's differential forms of the two qd algorithms and the algorithm for the Golub–Kahan form are conformable with backward error analysis with relative perturbations on the elements. We also study a modified form of differential algorithms which are backward stable with absolute perturbations. We are unaware of any singular value or eigenvalue algorithms, which furnish more accurate singular values than the algorithms we have analyzed in this section. In particular, the algorithm developed for the Golub–Kahan form provides the highest accuracy.

Section 5 is devoted to the analysis of the monotonicity of the computed inertia count. In exact arithmetic, the number of negative eigenvalues of a real symmetric matrix never decreases monotonically with the shift. However, in finite precision arithmetic, the monotonicity of the count can break down. Violation of monotonicity can lead to spurious and missing singular values. There are ways to overcome this problem on serial computers as it was for the routine xSTEBZ in LAPACK [1]. However, it is rather difficult to cope with nonmonotonic algorithms on parallel computers. Following the work of Kahan [17], we show that the algorithm for the Golub–Kahan form retains the monotonicity of the count if monotonic (e.g., IEEE) arithmetic is used. Unfortunately, the Rutishauser's differential qd algorithms do not possess this property although the differential qd algorithms with some modifications do give monotonic counts.

Finally, in section 6, we show how to scale bidiagonal matrices to maximize the high relative accuracy of the algorithms. The qd algorithms are more difficult to scale than the algorithm for the Golub–Kahan form.

The Rutishauser's differential qd algorithms have a backward error analysis with relative error terms but do not have the monotonicity property, whereas the modified differential qd algorithms have the monotonicity property but relative accuracy for small singular values is not present. However, the algorithm for the Golub–Kahan form has a backward error analysis in a relative sense and the monotonicity property as well. Furthermore, the Golub–Kahan form is easy to scale. Thus the algorithm based on the Golub–Kahan form is the overall winner if the accuracy of the computed singular values is paramount. However, if the cost (dominated by the division operation) is more important than the accuracy of the tiny singular values, then the differential qd algorithms are recommended.

Table 1 gives a summary of our analysis. The stationary qd (sqd), progressive qd (pqd), differential stationary (dsqd), and differential progressive (dpqd) are the four main qd algorithms of Rutishauser which were originally designed for computing eigenvalues of tridiagonal matrices. The *sqd* and *pqd* probably give eigenvalues with absolute accuracy but no error analyses are provided in this paper. The four classical qd algorithms and the two extensions described in this table are defined in the relevant sections.

In our opinion, the algorithm associated with the Golub–Kahan form is the primary algorithm for computing singular values of a bidiagonal matrix. The odd/even decomposition of the associated continued fraction gives the differential stationary and progressive qd algorithms. Algebraic manipulations to cut down the cost of computing lead to the nondifferential qd algorithms and the modified qd algorithms. This

[2] See Wall [26], Lorentzen and Waadeland [19] for details of the odd/even decomposition of a continued fraction.

Table 1
The main results.

Algorithm	Monotonicity	Accuracy	Divisions	Loop count	Prefix
sqd	yes	absolute ?	$n-1$	$n-1$	**s** = stationary
pqd	yes	absolute ?	$n-1$	$n-1$	**p** = progressive
dsqd	?	relative	$n-1$	$n-1$	**d** = differential
dpqd	?	relative	$n-1$	$n-1$	
mdsqd	yes	absolute	$n-1$	$n-1$	**m** = modified
mdpqd	yes	absolute	$n-1$	$n-1$	
for G-K	yes	relative	$2n-1$	$2n-1$	Golub–Kahan form

hierarchal classification is illustrated in Table 2 where B is taken as an upper bidiagonal matrix. However, it should be noted that this classification does not represent the historical order of discovery or development.

Table 2
Classification of algorithms.

Primary	Secondary	Tertiary	Continued fraction	Matrix
for G-K				G-K form of B
	dsqd		odd	B^tB
		sqd	odd	B^tB
		mdsqd	odd	B^tB
	dpqd		even	BB^t
		pqd	even	BB^t
		mdpqd	even	BB^t

Recently, Fernando [9], [8] developed new algorithms to compute eigenvectors of a tridiagonal matrix by solving a problem posed by Wilkinson. By using the Golub–Kahan form of a bidiagonal matrix, the algorithms in [9], [8] can be used to obtain singular vectors of a bidiagonal matrix. Similarly, the qd algorithms, which are derived from the Golub–Kahan form, can be extended to compute singular vectors. This topic is studied elsewhere.

2. Preliminaries.

2.1. Notation. Uppercase Roman letters denote matrices while lowercase Roman and Greek letters denote scalars. The singular values of an $n \times n$ real matrix C are arranged in monotone decreasing order and denoted by $\sigma_1, \sigma_2, \ldots, \sigma_n$; their union is $\sigma[C]$. The matrix C^t indicates the transpose of C.

We shall be concerned mainly with bidiagonal matrices and we take them to be upper bidiagonal. The diagonal elements of the bidiagonal matrix B are denoted by a_i and the off-diagonals by b_i:

$$B = \begin{bmatrix} a_1 & b_1 & & & & \\ & a_2 & b_2 & & & \\ & & \cdot & \cdot & & \\ & & & \cdot & \cdot & \\ & & & & a_{n-1} & b_{n-1} \\ & & & & & a_n \end{bmatrix}.$$

We often express the matrix B in terms of the qd variables q_i and e_i where $q_i = a_i^2$, $i = 1, \ldots, n$ and $e_i = b_i^2$, $i = 1, \ldots, n-1$. For notational convenience we assume that

$b_0 = b_n = 0$, $e_0 = e_n = 0$. Without loss of generality, we assume that none of the off-diagonal elements b_i are zero; otherwise, the matrix B splits into two submatrices.

2.2. Inertia. We make frequent references to the LDL^t factorization of a shifted symmetric tridiagonal matrix $T - \tau I$ as the product of a lower bidiagonal matrix L, a diagonal matrix D, and the transpose of L where τ is a real scalar value and I is the identity matrix. The factors are unique if the factorization exists. We use ν to denote the number of negative elements of D, which according to the Sylvester–Jacobi inertia theorem, gives the number of negative eigenvalues of $T - \tau I$.

2.3. Monotonicity of the inertia count. Let τ_1 and τ_2 be two shifts such that $\tau_2 > \tau_1$. In exact arithmetic, the outputs given by any reliable algorithm for computation of inertia should satisfy $\nu(\tau_2) \geq \nu(\tau_1)$. However, if the monotonicity property is not present then a routine could indicate that $\nu(\tau_2) < \nu(\tau_1)$, which would lead to the inconsistent conclusion that there are a negative number of eigenvalues (equal to $\nu(\tau_2) - \nu(\tau_1)$) in the half-open interval $[\tau_1, \tau_2)$. Demmel, Dhillon, and Ren [7] have designed a 2×2 matrix example to show the nonmonotonicity of the *bisect* routine in EISPACK [23]. The highly efficient parallel-prefix bisection algorithms of Mathias [20] also suffer from nonmonotonicity. The well-known technical report by Kahan [17], is a prerequisite for researchers who are seriously interested in monotonicity and other arithmetic issues in bisection algorithms and LDL^t factorization.

2.4. Characterization of singular values via eigenvalues. There are many ways to convert a singular value problem to an equivalent eigenvalue problem as indicated in section 8.3 of Golub and Van Loan [14]. It is well known that the eigenvalues of C^tC are equal to the eigenvalues of CC^t and that the eigenvalues are the squared singular values of the real square matrix C, an observation which dates back to Beltrami [4],

$$\lambda_i[C^tC] = \lambda_i[CC^t] = \sigma_i^2[C], \;\; i = 1, \ldots n.$$

The second approach, which was first exploited by Jordan [16], is to define a $2n \times 2n$ symmetric matrix,

$$A = \begin{bmatrix} 0 & C \\ C^t & 0 \end{bmatrix}.$$

The eigenvalues of A are then given by the union of the singular values of C and the negated singular values of C, $\lambda[A] = \{\sigma[C]\} \cup \{-\sigma[C]\}$ as presented in Theorem 4.2 of Stewart and Sun [25]. For historical details, refer to Stewart [24].

If C is bidiagonal (that is, $C = B$), then Golub and Kahan discovered that A can be condensed to a tridiagonal matrix with zero diagonal entries by using a permutational similarity transformation equivalent to a perfect shuffle. In recognition of this seminal work [13], we call this tridiagonal matrix, which we denote by T_0, the *Golub–Kahan* form. We identify the related skew-symmetric form by S_0:

$$T_0 = \begin{bmatrix} 0 & a_1 & & & & & \\ a_1 & 0 & b_1 & & & & \\ & b_1 & 0 & a_2 & & & \\ & & a_2 & 0 & . & & \\ & & & . & 0 & . & \\ & & & & . & 0 & a_n \\ & & & & & a_n & 0 \end{bmatrix}, \quad S_0 = \begin{bmatrix} 0 & a_1 & & & & & \\ -a_1 & 0 & b_1 & & & & \\ & -b_1 & 0 & a_2 & & & \\ & & -a_2 & 0 & . & & \\ & & & . & 0 & . & \\ & & & & . & 0 & a_n \\ & & & & & -a_n & 0 \end{bmatrix}.$$

The skew-symmetric form S_0 is less well known than the symmetric form T_0. However, Ward and Gray [27] proposed an algorithm for computing the eigenvalues of skew-symmetric forms S_0, which may be considered as an analogue of the Golub algorithm [12] for computing eigenvalues of the symmetric form.

2.5. Eigenvalues of skew-symmetric tridiagonal matrices. It is known [27] that eigenvalues of a $2n \times 2n$ skew-symmetric matrix can be determined by computing the singular values of an $n \times n$ matrix. If the dimension n of the skew-symmetric matrix is odd, we pad it with a zero row and a column to make it even and, hence, our approach applies to any general real skew-symmetric matrix. Note that any real skew-symmetric matrix can be reduced to tridiagonal form using orthogonal similarity transformations. The basis for our algorithm for computing eigenvalues of S_0 is contained in the following theorem, which is easy to prove.

THEOREM 2.1. *The eigenvalues of the complex Hermitian form $\sqrt{-1}\, S_0$ are given by $\{\sigma_i[B]\} \cup \{-\sigma_i[B]\}$.*

Remark. Thus, by computing the singular values of the equivalent real bidiagonal matrix B of the associated complex Hermitian form, the eigenvalues of a skew-symmetric matrix can be determined. Since the algorithms for computing eigenvalues of S_0 are identical to that of computing T_0, we do not make any further comments about skew-symmetric tridiagonal matrices.

3. The algorithms.

3.1. Inertia of $B^tB - \tau I$ via the sqd algorithm. We study the inertia of the shifted symmetric tridiagonal matrix $B^tB - \tau I$ where τ is an eigenvalue shift. The inertia of this matrix $B^tB - \tau I$ could be determined by knowing the signs of the diagonal pivot matrix $D(\tau) = \operatorname{diag}(\tilde{q}_1, \tilde{q}_2, \ldots, \tilde{q}_n)$ of the LDL^t factorization of $B^tB - \tau I$. It is easy to establish that Gaussian elimination (without pivoting) gives the following algorithm for $\tilde{q}_i(\tau)$:

$$\tilde{q}_i = \alpha_i - \frac{\beta_{i-1}}{\tilde{q}_{i-1}} - \tau \quad \text{for} \quad i = 1, \ldots, n, \tag{2}$$

where the diagonal elements α_i and the off-diagonal elements $\sqrt{\beta_i}$ of the symmetric tridiagonal matrix B^tB are given by

$$\alpha_i = a_i^2 + b_{i-1}^2 = q_i + e_{i-1} \quad \text{for} \quad i = 1, \ldots, n,$$

$$\beta_i = a_i^2 b_i^2 = q_i e_i \quad \text{for} \quad i = 1, \ldots, n-1$$

with $\beta_0 = 0$. Note that the LDL^t factorization exists if $\tilde{q}(\tau) \neq 0$ for $i = 1, \ldots, n-1$. Readers who are familiar with qd algorithms might recognize (2) as the (nondifferential) *sqd* algorithm of Rutishauser [21], [22].

Following Rutishauser, we define an auxiliary variable t_i as $t_i = q_i - \tilde{q}_i$, then (2) can be transformed to a recurrence for t_i,

$$t_i = \tau + e_{i-1}\frac{t_{i-1}}{\tilde{q}_{i-1}} \quad \text{for} \quad i = 2, \ldots, n \quad \text{with} \quad t_1 = \tau. \tag{3}$$

Once a particular t_i is computed, the corresponding pivot $\tilde{q}_i$ is given by $\tilde{q}_i = q_i - t_i$. Equation (3) corresponds to the *differential* form of the *sqd* algorithm of Rutishauser. This requires $n-1$ additional multiplications compared with (2). The incentive for this extra work is the low errors due to round-off in floating-point arithmetic as demonstrated in a later section. Now we state the basic algorithm for counting the negative eigenvalues of $B^tB - \tau I$ based on (3). On exit, the algorithm outputs $\nu(\tau)$, the number of eigenvalues of B^tB which are less than τ.

ALGORITHM 1. *dsqd*

```
$\nu := 0$
$t := \tau$
$\tilde{q} := q_1 - t$
*if* $\tilde{q} < 0$ *then* $\nu := 1$
*for* $i = 2, n$
    $t := \tau + e_{i-1} * t/\tilde{q}$
    $\tilde{q} := q_i - t$
    *if* $\tilde{q} < 0$ *then* $\nu := \nu + 1$
*endfor*
```

In modern architecture, the appearance of a conditional can slow down the operations in a pipeline. Thus, in a production version of the algorithm, the conditional $\tilde{q} < 0$ should be removed from the inner loop. As an example, it is possible to replace the conditional by $\nu := \nu + [0.5 - \text{sign}(0.5, \tilde{q})]$, where the sign function is as defined in Fortran. However, we do not study such implementation issues in this report.

We now rearrange terms in (3) to give a marginally different qd algorithm. This modified version has the advantage of having the monotonicity property as demonstrated in a later section. However, by gaining monotonicity, the modified algorithm loses out on round-off errors.

DEFINITION 3.1 (modified differential stationary qd (mdsqd)).
The mdsqd *is of the form*

$$t_i = \tau + e_{i-1}\left[\frac{q_{i-1}}{\tilde{q}_{i-1}} - 1\right], \quad \tilde{q}_i = q_i - t_i. \tag{4}$$

3.2. Inertia of $BB^t - \tau I$ via the pqd algorithm. Instead of using the matrix $B^tB - \tau I$, it is possible to use the symmetric tridiagonal matrix $BB^t - \tau I$ to count the singular values of B. As previously, Gaussian elimination (without pivoting) can be used to obtain the diagonal pivots of the LDL^t transformation of $BB^t - \tau I$. In this case we denote the pivots (which are the diagonal elements of $D(\tau)$) as $\hat{q}_i$. They can be computed using the recursion

$$\hat{q}_i = \alpha_i - \frac{\beta_{i-1}}{\hat{q}_{i-1}} - \tau \ \text{ for } \ i = 1, \dots, n, \tag{5}$$

where the diagonal elements α_i and the off-diagonal elements $\sqrt{\beta_i}$ of the symmetric tridiagonal matrix BB^t are given by

$$\alpha_i = a_i^2 + b_i^2 = q_i + e_i \ \text{ for } \ i = 1, \dots, n,$$

$$\beta_i = a_{i+1}^2 b_i^2 = q_{i+1} e_i \ \text{ for } \ i = 1, \dots, n-1$$

with $\beta_0 = 0$. The LDL^t factorization of $BB^t - \tau I$ exists if $\hat{q}_i(\tau) \neq 0$ for $i = 1, \dots, n-1$. The above recursion may be recognized as the (nondifferential) *pqd* algorithm of Rutishauser, if we define an auxiliary variable s_i as $s_i = \hat{q}_i - e_i$. Then (5) can be rewritten as

$$s_i = s_{i-1}\frac{q_i}{\hat{q}_{i-1}} - \tau \ \text{ for } \ i = 2, \dots, n \ \text{ with } \ s_1 = q_1 - \tau. \tag{6}$$

The recursion (6) requires $n-1$ additional multiplications compared with (5) and it is the *differential* form of the *pqd* algorithm of Rutishauser. This qd algorithm,[3] with the given name *dqds*, was extensively studied by Fernando and Parlett [10]. Following Rutishauser, Fernando and Parlett used the notation d_i to denote s_i of (6). In this report we do not use d_i as it can be confused with the diagonal pivots of the LDL^t factorization. Now we state the basic algorithm for counting the negative eigenvalues of $BB^t - \tau I$ based on (6). On exit, Algorithm 2 outputs $\nu(\tau)$, the number of eigenvalues of BB^t which are less than τ.

ALGORITHM 2. *dpqd*

```
$\nu := 0$
$s := q_1 - \tau$
$\hat{q} := q_1 + e_1$
if $\hat{q} < 0$ then $\nu := 1$
for $i = 2, n$
    $s := q_i * s/\hat{q} - \tau$
    $\hat{q} := s + e_i$
    if $\hat{q} < 0$ then $\nu := \nu + 1$
endfor
```

Instead of (6), it is possible to define a modified version to acquire the monotonicity property.

DEFINITION 3.2 (modified differential progressive qd (mdpqd)).
The mdpqd *is of the form*

$$s_i = \left[1 - \frac{e_{i-1}}{\hat{q}_{i-1}}\right] q_i - \tau, \quad \hat{q}_i = s_i + e_i. \tag{7}$$

3.3. Inertia of the Golub–Kahan form T_0. In this section we study the $2n \times 2n$ zero–diagonal symmetric tridiagonal matrix T_0. As stated in an earlier section, the eigenvalues of T_0 are given by the singular values of B and the negated singular values of B.

The diagonal matrix $D(\rho)$ of the LDL^t factorization of the tridiagonal matrix $T_0 - \rho I$ can be obtained by Gaussian elimination (without pivoting). In this case, we denote the pivots as d_i and they are given by

$$d_i = -\rho - \frac{z_{i-1}}{d_{i-1}} \text{ for } i = 2, \ldots, 2n \text{ with } d_1 = -\rho, \tag{8}$$

where $z_i = a_i^2 = q_i$ if i is odd and $z_i = b_i^2 = e_i$ if i is even with $z_0 = b_0 = 0$. The LDL^t factorization of $T_0 - \rho I$ exists if $d_i(\rho) \neq 0$ for $i = 1, \ldots, 2n-1$. On exit, Algorithm 3 gives $\nu(\rho)$, the number of eigenvalues of $T_0 - \rho I$ which are less than ρ.

ALGORITHM 3. *Inertia of the Golub–Kahan form*

```
$\nu := 0$
$d := -\rho$
if $d < 0$ then $\nu := 1$
for $i = 1, 2n - 1$
```

[3] Rutishauser intended this algorithm for computing eigenvalues of a tridiagonal matrix. Fernando and Parlett "misused" it for computing accurate singular values of a bidiagonal matrix.

```
        $d := -\rho - z_i/d$
        *if* $d < 0$ *then* $\nu := \nu + 1$
    *endfor*
```

Algorithms 1 and 2 have n fewer divisions than Algorithm 3 but $n-1$ more multiplications. Since divisions are more expensive than multiplications in modern architecture (perhaps by a factor of six on a typical workstation), the cost is nearly doubled in the third algorithm. The modified algorithms *mdsqd* and *mdpqd* have the same number of divisions as the Rutishauser's differential algorithms.

3.4. Relationships between the algorithms. Our intention is to show that Algorithm 3 contains information given by both the *dpqd* algorithm and the *dsqd* algorithm for nonzero ρ. In the previous section, we derived the recursion for the Golub–Kahan form T_0 as

$$d_i = -\rho - z_{i-1}/d_{i-1}, \quad i = 1, \ldots, 2n \tag{9}$$

with $z_0 = 0$. We also recall that $Z = (a_1^2, b_1^2, a_2^2, \ldots, b_{n-1}^2, a_n^2) = (q_1, e_1, q_2, \ldots, e_{n-1}, q_n)$. The recursion (9) can be rewritten in the form

$$x_i = -\rho - b_{i-1}^2/y_{i-1}, \quad y_i = -\rho - a_i^2/x_i, \;\; i = 1, \ldots, n,$$

where $(d_1, d_2, d_3, \ldots, d_{2n}) = (x_1, y_1, x_2, \ldots, y_n)$.

THEOREM 3.3. *If the LDL^t factorization of $T_0 - \rho I$ exists for $\tau = \rho^2 > 0$ then the recursion for the Golub–Kahan form*

$$x_i = -\rho - e_{i-1}/y_{i-1}, \;\; i = 2, \ldots, n \;\; \textit{with} \;\; x_1 = -\rho, \tag{10}$$

$$y_i = -\rho - q_i/x_i, \;\; i = 1, \ldots, n \tag{11}$$

is related to the dsqd *algorithm*

$$t_i = \tau + e_{i-1}t_{i-1}/(q_{i-1} - t_{i-1}) \;\; \textit{for} \;\; i = 2, \ldots, n \;\; \textit{with} \;\; t_1 = \tau, \tag{12}$$

and the dpqd *algorithm*

$$s_i = s_{i-1}q_i/(s_{i-1} + e_{i-1}) - \tau \;\; \textit{for} \;\; i = 2, \ldots, n \;\; \textit{with} \;\; s_1 = q_1 - \tau \tag{13}$$

via the relationships $t_i = -x_i\rho$ and $s_i = y_i\rho$.

Proof. By eliminating y_{i-1} in (10) by substituting (11) we get

$$x_i = -\rho + \frac{e_{i-1}}{\rho + q_{i-1}/x_{i-1}}$$

and multiplying both sides by ρ

$$(\rho x_i) = -\rho^2 + \frac{q_{i-1}}{q_{i-1} + (\rho x_{i-1})}(\rho x_{i-1}). \tag{14}$$

By comparing the preceding equation with (12) and noting that $\rho^2 = \tau$ we obtain the claimed result $t_i = -\rho x_i$. Similarly, it is possible to show that $s_i = y_i\rho$. □

Remark. The algebraic manipulations in the preceding proof are equivalent to decomposing a continued fraction expansion into its odd and even parts [19], [26].

COROLLARY 3.4. *For nonzero ρ,*

$$t_i = \tau\frac{\hat{q}_{i-1}}{s_{i-1}}, \tag{15}$$

$$\begin{tikzcd}[column sep=huge, row sep=huge]
Z \arrow[r, "\textbf{dsqd}", "\text{computed}"'] \arrow[d, "\begin{array}{l}\text{change each}\\ q_i,\ i \neq 1 \neq n,\ \text{by 1 ULP}\\ e_i\ \text{by 3 ULPs}\end{array}"'] & \tilde{Z} \\
\bar{Z} \arrow[r, "\textbf{dsqd}", "\text{exact}"'] & \breve{Z} \arrow[u, "\begin{array}{l}\text{change each}\\ \breve{q}_i,\ i \neq 1 \neq n,\ \text{by 2 ULPs}\\ \breve{q}_1,\ \breve{q}_n\ \text{by 1 ULP}\end{array}"']
\end{tikzcd}$$

FIG. 1. *Effects of round-off in* dsqd.

$$s_i = \tau \frac{\tilde{q}_i}{t_i}. \tag{16}$$

Proof. By multiplying (10) by ρ we obtain $(\rho x_i) = -\rho^2 + e_{i-1}\rho^2/(\rho y_{i-1})$. Since $\rho x_i = -t_i$, $\rho y_{i-1} = q_i$, and $\rho^2 = \tau$, we get $t_i = \tau(1 + e_{i-1}/s_{i-1})$. We confirm (15) by noting that $\hat{q}_{i-1} = e_{i-1} + s_{i-1}$. A similar reasoning leads to (16). □

4. Backward error analyses.

4.1. Round-off error. Our model of arithmetic is such that the floating-point result of a basic arithmetic operation $\circ$ satisfies

$$\mathrm{fl}(x \circ y) = (x \circ y)(1 + \varsigma) = \frac{(x \circ y)}{(1 + \delta)},$$

where ς and δ depend on x, y, $\circ$, and the arithmetic unit but satisfy $|\varsigma| < \epsilon$, $|\delta| < \epsilon$ for a given ϵ that depend only on the arithmetic unit. Typically, ϵ is less than or equal to the machine precision of the machine. We shall choose freely the form (ς or δ) that suits the analysis. We omit the phrase *at most* in front of the number of units in the last place held (ULPs) in our error theorems. The notation fl(.) indicates computed values in floating-point arithmetic.

4.2. The differential stationary qd. Data for the *dsqd* algorithm consists of a representable positive qd array $Z = [q_1, e_1, \ldots, q_n]$ and a positive representable eigenvalue shift τ. The algorithm computes intermediate quantities $\{t_i, \tilde{q}_i\}$ and finally $\nu(\tau)$, the number of negative $\tilde{q}_i(\tau)$.

THEOREM 4.1. *In the absence of underflow (including denormalization), overflow or infinity, the* dsqd *algorithm computes a ν that is exact for a positive qd array $\bar{Z}$, where $\bar{q}_i$ differs from q_i by* 1 ULP *and $\bar{e}_i$ differs from e_i by* 3 ULP*s (except that $\bar{q}_1 = q_1$ and $\bar{q}_n = q_n$). See Figure* 1.

Proof. Let t_i and $\tilde{q}_i$ denote the representable quantities computed by the *dsqd* algorithm. We write down the exact relations that connect the q, e, and t quantities and then interpret these relations so that they reveal an exact *dsqd* algorithm acting on perturbed data. Let $\breve{Z} = [\breve{q}_1, \breve{e}_1, \breve{q}_2, \breve{e}_2, \ldots, \breve{e}_{n-1}, \breve{q}_n]$ be the output of the *dsqd* algorithm in exact arithmetic with input $\bar{Z}$. However, in this algorithm, $\breve{e}_i$ is not computed. Recall that for $i = 1, \ldots, n$

$$t_i = \mathrm{fl}\left(\tau + e_{i-1} * \frac{t_{i-1}}{\tilde{q}_{i-1}}\right), \quad \tilde{q}_i = \mathrm{fl}\left(q_i - t_i\right).$$

The t_i suffer from possible errors in the three operations $/$, $*$, and $+$, which we choose to account for in the following way. For $i = 1, \ldots, n$

$$t_i = \frac{\left[\tau + e_{i-1}(1+\epsilon_*)(1+\epsilon_/)t_{i-1}/\tilde{q}_{i-1}\right]}{(1+\epsilon_+^i)}. \tag{17}$$

However, $t_1 = \tau$ as no arithmetic operation takes place and, hence, $\epsilon_+^1 = 0$. We have not bothered to put the superscript i on ϵ_* or $\epsilon_/$ because they play a straightforward role in the analysis. Similarly, we write

$$\tilde{q}_i = \frac{(q_i - t_i)}{(1+\epsilon_-^i)}. \tag{18}$$

The constraint that drives our choices is that τ should be unperturbed each time it appears. Consequently, we invoke the (unrepresentable) quantities

$$\bar{t}_i = t_i(1+\epsilon_+^i) \tag{19}$$

instead of the t_i themselves. However, $\bar{t}_1 = t_1, \epsilon_+^1 = 0$. Now we can write down an exact instance of the *dsqd* algorithm. For $i = 2, \ldots n$

$$\begin{aligned}
\bar{t}_i &= t_i(1+\epsilon_+^i), \text{ by (19)} \\
&= \tau + \frac{e_{i-1}(1+\epsilon_*)(1+\epsilon_/)t_{i-1}(1+\epsilon_-^{i-1})}{q_{i-1} - t_{i-1}}, \text{ by (17) and (18)} \\
&= \tau + \frac{\bar{e}_{i-1}t_{i-1}}{q_{i-1} - t_{i-1}},
\end{aligned}$$

where

$$\bar{e}_{i-1} = e_{i-1}(1+\epsilon_*)(1+\epsilon_/)(1+\epsilon_-^{i-1}). \tag{20}$$

We also define

$$\bar{q}_i = q_i(1+\epsilon_+^i), \;\; i = 1, \ldots, n \quad \text{where } \; \epsilon_+^1 = \epsilon_+^n = 0, \tag{21}$$

so that $\bar{t}_{i-1}/(\bar{q}_{i-1} - \bar{t}_{i-1}) = t_{i-1}/(q_{i-1} - t_{i-1})$ and thus an exact instance of the *dsqd* algorithm in terms of the unrepresentable quantities is given by $\bar{t}_i = \tau + \bar{e}_{i-1}\bar{t}_{i-1}/(\bar{q}_{i-1} - \bar{t}_{i-1})$. Then,

$$\begin{aligned}
\breve{q}_i &= \bar{q}_i - \bar{t}_i, \text{ by definition} \\
&= (1+\epsilon_+^i)(q_i - t_i), \text{ by (19) and (21)} \\
&= (1+\epsilon_+^i)(1+\epsilon_-^i)\tilde{q}_i, \text{ by (18).}
\end{aligned} \tag{22}$$

Provided that no underflow (including denormalization), overflow, zero, or infinity occurs then (22) shows that $\breve{q}_i < 0$ if and only if $\tilde{q}_i < 0$. Thus $\nu(\tau)$, the number of negative $\tilde{q}_i(\tau)$, is also exact for the qd array $\bar{Z}$ defined by (20), (21), and eigenvalue shift τ as claimed. □

$$\begin{tikzcd}[column sep=huge, row sep=huge]
Z \arrow[r, "\mathbf{dpqd}", "\text{computed}"'] \arrow[d, "\begin{array}{l}\text{change each}\\ q_i,\ i \neq 1 \text{ by 3 ULPs}\\ e_i \text{ by 1 ULP}\end{array}"'] & \hat{Z} \\
\bar{Z} \arrow[r, "\mathbf{dpqd}", "\text{exact}"'] & \breve{Z} \arrow[u, "\begin{array}{l}\text{change each}\\ \breve{q}_i,\ i \neq 1 \text{ by 2 ULPs}\\ \breve{q}_1 \text{ by 1 ULP}\end{array}"']
\end{tikzcd}$$

FIG. 2. *Effects of round-off in* dpqd.

4.3. The dpqd. Data for the *dpqd* algorithm consists of a representable positive qd array $Z = [q_1, e_1, \ldots, q_n]$ and a nonnegative representable eigenvalue shift τ. The algorithm computes intermediate quantities $\{s_i, \hat{q}_i\}$ and finally $\nu(\tau)$, the number of negative $\hat{q}_i(\tau)$.

THEOREM 4.2. *In the absence of underflow (including denormalization), overflow, or infinity, the* dpqd *algorithm computes a ν that is exact for a positive qd array $\bar{Z}$ where $\bar{q}_i$ differs from q_i by* 3 ULP*s and $\bar{e}_i$ by differs from e_i by* 1 ULP *(except that $\bar{q}_1 = q_1$ and $\bar{q}_n = q_n$. See Figure* 2.

Proof. Let $\hat{q}_i$ and s_i denote the representable quantities computed by the *dpqd.* We write down the exact relations that connect the q, e, p, and s quantities and then interpret these relations so that they reveal an exact *dpqd* algorithm acting on perturbed data. Let $\breve{Z} = [\breve{q}_1, \breve{e}_1, \breve{q}_2, \breve{e}_2, \ldots, \breve{e}_{n-1}, \breve{q}_n]$ be the output of the *dpqd* algorithm in exact arithmetic with input $\bar{Z}$. However, in this algorithm $\breve{e}_i$ are not required as information on inertia is contained in $\breve{q}_i$. Recall that

$$s_i = \mathrm{fl}\left(s_{i-1} * \frac{q_i}{\hat{q}_{i-1}} - \tau\right), \quad i = 1, \ldots, n,$$
$$\hat{q}_i = \mathrm{fl}\left(s_i + e_i\right), \quad i = 1, \ldots, n-1, \quad \hat{q}_n = s_n.$$

The s_i suffer from possible errors in the three operations $/$, $*$, and $-$, which we choose to account for in the following way. For $i = 1, \ldots, n$,

$$s_i = \frac{\left[s_{i-1}(1+\epsilon_*)(1+\epsilon_/)q_i/\hat{q}_{i-1} - \tau\right]}{(1+\epsilon_-^i)}. \tag{23}$$

However, $s_1 = (q_1 - \tau)/(1 + \epsilon_-^1)$ where $\epsilon_* = \epsilon_/ = 0$ in this initial step. We write

$$\hat{q}_i = \frac{(s_i + e_i)}{(1 + \epsilon_+^i)}, \;\; i = 1, \ldots, n-1 \;\text{ with }\; \hat{q}_n = s_n, \quad \epsilon_+^n = 0. \tag{24}$$

The constraint that drives our choices is that τ should be unperturbed each time it appears. Consequently, we invoke the (unrepresentable) quantities

$$\bar{s}_i = (1 + \epsilon_-^i) s_i \tag{25}$$

instead of the s_i themselves. Now we can write down an exact instance of the *dpqd* algorithm. For $i = 1, \ldots, n$,

$$\begin{aligned}\bar{s}_i &= s_i(1+\epsilon_-^i)\\ &= \frac{s_{i-1}q_i(1+\epsilon_*)(1+\epsilon_/)}{\hat{q}_{i-1}} - \tau \text{ , by (23)}\\ &= \frac{s_{i-1}q_i(1+\epsilon_*)(1+\epsilon_/)(1+\epsilon_+^{i-1})}{s_{i-1}+e_{i-1}} - \tau, \text{ by (24)}\\ &= \frac{s_{i-1}(1+\epsilon_-^{i-1})q_i(1+\epsilon_*)(1+\epsilon_/)(1+\epsilon_+^{i-1})}{s_{i-1}(1+\epsilon_-^{i-1})+e_{i-1}(1+\epsilon_-^{i-1})} - \tau\\ &= \frac{\bar{s}_{i-1}\bar{q}_i}{\bar{s}_{i-1}+\bar{e}_{i-1}} - \tau,\end{aligned}$$

where

$$\bar{e}_{i-1} = e_{i-1}(1+\epsilon_-^{i-1}), \tag{26}$$

$$\begin{gathered}\bar{q}_i = q_i(1+\epsilon_*)(1+\epsilon_/)(1+\epsilon_+^i), \quad \bar{q}_1 = q_1,\\ \bar{Z} = [\bar{q}_1, \bar{e}_1, \bar{q}_2, \bar{e}_2, \ldots, \bar{e}_{n-1}, \bar{q}_n]\,.\end{gathered} \tag{27}$$

By definition,

$$\breve{q}_i = \bar{s}_i + \bar{e}_i, \quad i = 1, \ldots, n-1 \text{ with } \breve{q}_n = \bar{s}_n. \tag{28}$$

It follows from (28), (25), (26), and (24) that

$$\breve{q}_i = (1+\epsilon_-^i)(s_i+e_i) = (1+\epsilon_+^i)(1+\epsilon_-^i)\hat{q}_i. \tag{29}$$

Provided that no underflow (including denormalization), overflow, zero, or infinity occurs, then (29) shows that $\breve{q}_i < 0$ if and only if $\hat{q}_i < 0$. Thus $\nu(\tau)$, the number of negative $\hat{q}_i(\tau)$, is also exact for the qd array $\bar{Z}$ defined by (26), (27), and the eigenvalue shift τ as claimed. □

4.4. The factorization of the Golub–Kahan form. Data for Algorithm 3 consists of a representable positive qd array

$$Z = [q_1, e_1, q_2, e_2, \ldots, e_{n-1}, q_n] = [z_1, z_2, \ldots, z_{2n-2}, z_{2n-1}]$$

and a representable singular value shift ρ. The algorithm computes intermediate quantities $\{d_i\}$and finally $\nu(\rho)$, the number of negative $d_i(\rho)$.

THEOREM 4.3. *In the absence of underflow (including denormalization), overflow, or infinity, Algorithm* 3 *computes a* ν *that is exact for a positive qd array* $\bar{Z}$*, where* $\bar{q}_i$ *differs from* q_i *by* 2 ULP*s (except that* $\bar{q}_1$ *differs from* q_1 *by* 1 ULP*) and* $\bar{e}_i$ *differs from* e_i *by* 2 ULP*s. See Figure* 3.

Proof. Let d_i denote the representable quantities computed by Algorithm 3. We write down the exact relations that connect the q, e, and d quantities and then interpret these relations so that they reveal an exact algorithm acting on perturbed data. Recall that for $i = 2, \ldots, 2n$, $d_i = -\mathrm{fl}(\rho + z_{i-1}/d_{i-1})$. Initially, $d_1 = -\rho$. So we set $\breve{d}_1 = d_1$. The remaining d_i suffer from possible errors in the two operations /, and +, which we choose to account for in the following way:

$$d_i = \frac{-\left[\rho + z_{i-1}(1+\epsilon_/)/d_{i-1}\right]}{(1+\epsilon_+^i)}. \tag{30}$$

$$\begin{tikzcd}[column sep=huge, row sep=huge]
Z \arrow[r, "LDL^t", "\text{computed}"'] \arrow[d, "{\begin{array}{l}\text{change each}\\ z_i,\, i \neq 1 \text{ by 2 ULPs}\\ z_1 \text{ by 1 ULP}\end{array}}"'] & d_i \\
\bar{Z} \arrow[r, "LDL^t", "\text{exact}"'] & \breve{d}_i \arrow[u, "{\begin{array}{l}\text{change each}\\ \breve{d}_i, i \neq 1 \text{ by 1 ULP}\end{array}}"']
\end{tikzcd}$$

FIG. 3. *Effects of round-off in Algorithm* 3.

The constraint that drives our choices is that ρ should be unperturbed each time it appears. Consequently, we invoke the (unrepresentable) quantities

$$\breve{d}_i = d_i(1+\epsilon_+^i), \quad \epsilon_+^1 = 0 \tag{31}$$

instead of the d_i themselves. Now we can write down an exact instance of Algorithm 3 for the Golub–Kahan form. For $i = 2, \dots, 2n$

$$\begin{aligned}
\breve{d}_i &= d_i(1+\epsilon_+^i) \\
&= -\rho - \frac{z_{i-1}(1+\epsilon_/)}{d_{i-1}}\ , \text{ by (30)} \\
&= -\rho - \frac{z_{i-1}(1+\epsilon_/)(1+\epsilon_+^{i-1})}{d_{i-1}(1+\epsilon_+^{i-1})} \\
&= -\rho - \frac{\bar{z}_{i-1}}{\breve{d}_{i-1}}, \ \text{ by (31),}
\end{aligned}$$

where

$$\bar{z}_{i-1} = z_{i-1}(1+\epsilon_/)(1+\epsilon_+^{i-1}). \tag{32}$$

Provided that no underflow (including denormalization), overflow, zero, or infinity occurs, then (31) shows that $\breve{d}_i < 0$ if and only if $d_i < 0$. Thus $\nu(\rho)$, the number of negative $d_i(\rho)$, is also exact for the qd array $\bar{Z}$ defined by (32) and singular value shift ρ as claimed. □

4.5. The msdqd. The dsqd algorithm is based on the LDL^t factorization of $B^tB - \tau I$. We now state without proof the algorithm we get if $B^tB - \Delta - \tau I$ is factorized where Δ is a diagonal matrix.

LEMMA 4.4. *Let the diagonal pivots (as given by the diagonal matrix D) of the LDL^t factorization of $B^tB + \Delta - \tau I$ are given by $\tilde{q}_i$, $i = 1, \dots, n$ where $\Delta = diag(\delta_1, \dots, \delta_n)$. The $\tilde{q}_i$ are then given by $t_i = \tau - \delta_i + e_{i-1} * (q_{i-1}/\tilde{q}_{i-1} - 1), \tilde{q}_i = q_i - t_i$.*
Our intention is to show that the *mdsqd* algorithm, in floating point arithmetic, is equivalent to the factorization of a matrix $\bar{B}^t\bar{B} + \Delta - \tau I$, where Δ is a diagonal matrix which contains perturbational terms and $\bar{B}$ is a relatively perturbed version of B,

$$\begin{tikzcd}
Z \arrow[r, "\textbf{mdsqd}", "\text{computed}"'] \arrow[d, "\begin{array}{l}\text{change each}\\ q_i,\ i \neq 1 \neq n \text{ by 1 ULP}\\ e_i \text{ by 4 ULPs}\end{array}"'] & \tilde{Z} \\
\bar{Z} \arrow[r, "\begin{array}{c}LDL^t \text{ of}\\ \bar{B}^t\bar{B} + \Delta - \tau I\end{array}", "\text{exact}"'] & \breve{Z} \arrow[u, "\begin{array}{l}\text{change each}\\ \breve{q}_i,\ i \neq 1 \neq n \text{ by 2 ULPs}\\ \breve{q}_1, \breve{q}_n \text{ by 1 ULP}\end{array}"']
\end{tikzcd}$$

FIG. 4. *Effects of round-off in* mdsqd.

where the relative perturbations are comparable to that induced by the Rutishauser's differential stationary algorithm.

THEOREM 4.5. *In the absence of underflow (including denormalization), overflow, or infinity, the* mdsqd *algorithm computes a ν that is exactly given by the LDL^t factorization of $\bar{B}^t\bar{B} + \Delta - \tau I$, where $\bar{q}_i$ differs from q_i by* 1 ULP *and $\bar{e}_i$ differs from e_i by* 4 ULP*s (except that $\bar{q}_1 = q_1$ and $\bar{q}_n = q_n$) and $\Delta = diag(\delta_1, \dots, \delta_n)$, δ_i is approximately equal to $2\bar{e}_{i-1}$ times the machine precision. See Figure* 4.

Proof. Let t_i and $\tilde{q}_i$ denote the representable quantities computed by the LDL^t factorization. We write down the exact relations that connect the q, e, and t quantities and then interpret these relations so that they reveal an exact LDL^t factorization (as in Lemma 4.4) acting on perturbed data. Let $\breve{Z} = [\breve{q}_1, \breve{e}_1, \breve{q}_2, \breve{e}_2, \dots, \breve{e}_{n-1}, \breve{q}_n]$ be the output of the LDL^t factorization of $\bar{B}^t\bar{B} + \Delta - \tau I$ algorithm in exact arithmetic with input $\bar{Z}$. However, in this algorithm, $\breve{e}_i$ are not computed. Recall that for $i = 1, \dots, n$,

$$t_i = \mathrm{fl}\left(\tau + e_{i-1} * \left(\frac{q_{i-1}}{\tilde{q}_{i-1}} - 1\right)\right), \quad \tilde{q}_i = \mathrm{fl}\left(q_i - t_i\right).$$

The t_i suffer from possible errors in the three operations /, $*$, and $+$, which we choose to account for in the following way. For $i = 2, \dots, n$

$$t_i = \frac{\left\{\tau + e_{i-1}(1+\epsilon_*)(1+\epsilon_-)\left[(1+\epsilon_/)^{-1} q_{i-1}/\tilde{q}_{i-1} - 1\right]\right\}}{(1+\epsilon_+^i)}. \tag{33}$$

Similarly, we write

$$\tilde{q}_i = (q_i - t_i)(1+\epsilon_-^i). \tag{34}$$

The constraint that drives our choices is that τ should be unperturbed each time it appears. Consequently, we invoke the (unrepresentable) quantities

$$\bar{t}_i = t_i(1+\epsilon_+^i) \tag{35}$$

instead of the t_i themselves. Now we can write down an exact instance of the LDL^t factorization (see Lemma 4.4). For $i = 2, \dots, n$

$\bar{t}_i = t_i(1+\epsilon_+^i)$, by (35)

$$
\begin{aligned}
&= \tau + e_{i-1}(1+\epsilon_*)(1+\epsilon_-)\left[\frac{(1+\epsilon_/)^{-1}(1+\epsilon_-^{i-1})^{-1}q_{i-1}}{q_{i-1}-t_{i-1}} - 1\right] \text{, by (33) and (34)}\\
&= \tau + e_{i-1}(1+\epsilon_*)(1+\epsilon_-)(1+\epsilon_/)^{-1}(1+\epsilon_-^{i-1})^{-1}\\
&\quad \left[\frac{t_{i-1}}{q_{i-1}-t_{i-1}} - (\epsilon_/ + \epsilon_-^{i-1} + \epsilon_/\epsilon_-^{i-1})\right]\\
&= \tau + \frac{\bar{e}_{i-1}t_{i-1}}{q_{i-1}-t_{i-1}} - \delta_i,
\end{aligned}
$$

where

$$
\begin{aligned}
\bar{e}_{i-1} &= e_{i-1}(1+\epsilon_*)(1+\epsilon_-)(1+\epsilon_-^{i-1})^{-1}(1+\epsilon_/)^{-1},\\
\delta_i &= \bar{e}_{i-1}(\epsilon_/ + \epsilon_-^{i-1} + \epsilon_/\epsilon_-^{i-1}).
\end{aligned} \tag{36}
$$

With an eye on (34) we define

$$
\bar{q}_{i-1} = q_{i-1}(1+\epsilon_+^{i-1}) \tag{37}
$$

so that$\bar{t}_{i-1}/(\bar{q}_{i-1} - \bar{t}_{i-1}) = t_{i-1}/(q_{i-1} - t_{i-1})$ and thus an exact instance of the LDL^t factorization in terms of the unrepresentable quantities is given by $\bar{t}_i = \tau + \bar{e}_{i-1}\bar{t}_{i-1}/(\bar{q}_{i-1} - \bar{t}_{i-1}) - \delta_i$. Then,

$$
\begin{aligned}
\breve{q}_i &= \bar{q}_i - \bar{t}_i \ \text{ by definition}\\
&= (1+\epsilon_+^i)(q_i - t_i), \ \text{ by (35) and (37)}\\
&= (1+\epsilon_+^i)(1+\epsilon_-^i)^{-1}\tilde{q}_i, \ \text{ by (34)} \qquad (38)
\end{aligned}
$$

with $\epsilon_+^1 = 0$. Provided that no underflow (including denormalization), overflow, zero, infinity occurs, then (38) shows that $\breve{q}_i < 0$ if and only if $\tilde{q}_i < 0$. Thus $\nu(\tau)$, the number of negative $\tilde{q}_i(\tau)$, is given by the pivots of the exact factorization of $\bar{B}^t\bar{B} + \Delta - \tau I$ as claimed. □

4.6. The mpdqd. The analysis in this case is similar to that of the mdsqd algorithm and, hence, we state the following result without proof.

THEOREM 4.6. *In the absence of underflow (including denormalization), overflow, or infinity, the* mdpqd *algorithm computes a* ν *that is exactly given by the* LDL^t *factorization of* $\bar{B}\bar{B}^t + \Delta - \tau I$*, where* $\bar{q}_i$ *differs from* q_i *by* 4 ULP*s and* $\bar{e}_i$ *differs from* e_i *by* 1 ULP *(except that* $\bar{q}_1 = q_1$*) and* $\Delta = diag(\delta_1, \ldots, \delta_n)$*,* δ_i *is approximately equal to* $2\bar{q}_i$ *times the machine precision (except that* $\delta_1 = 0$*). See Figure* 5.

4.6.1. Rutishauser's differential algorithms. In the *dsqd* algorithm the off-diagonal elements of B, on average, are more affected than the diagonal elements while in *dpqd* algorithm the opposite is true. Thus, if the diagonal elements are more dominant than the off-diagonals then one will be tempted to use the *dsqd* algorithm. However, if the off-diagonals are dominant then the *dpqd* algorithm should be selected. In the algorithm for the Golub–Kahan form, the errors are equal for both diagonals and off-diagonals and hence, as far as the errors are concerned, it is the algorithm of choice if information regarding diagonal dominance is not available. Table 3 shows the errors sustained in each algorithm. The following extends Theorems 4.1, 4.2, and 4.3.

COROLLARY 4.7. *In the absence of underflow (including denormalization), overflow, or infinity, Algorithms* 1, 2, *and* 3 *compute singular values to relatively high*

$$\begin{tikzcd}
Z \arrow[r, "\textbf{mdpqd}", "\text{computed}"'] \arrow[d, "\begin{array}{l}\text{change each}\\ q_k,\ k \neq 1 \text{ by 4 ULPs}\\ \\ e_k \text{ by 1 ULP}\end{array}"'] & \hat{Z} \\
\bar{Z} \arrow[r, "\begin{array}{c}LDL^t \text{ of}\\ \bar{B}\bar{B}^t + \Delta - \tau I\end{array}", "\text{exact}"'] & \breve{Z} \arrow[u, "\begin{array}{l}\text{change each}\\ \breve{q}_i,\ i \neq 1, \text{ by 2 ULPs}\\ \breve{q}_1 \text{ by 1 ULP}\end{array}"']
\end{tikzcd}$$

FIG. 5. *Effects of round-off in* mdpqd.

TABLE 3
ULP *changes on qd variables.*

	q_1	q_i	q_n	e_i	Total
dsqd	0	1	0	3	$4n-5$
dpqd	0	3	3	1	$4n-4$
Golub–Kahan	1	2	2	2	$4n-3$

accuracy. In particular, the bidiagonal matrix $\bar{B}$ associated with $\bar{Z}$ (that is, $\bar{a}_i = \sqrt{\bar{q}_i}$, $\bar{b}_i = \sqrt{\bar{e}_i}$) satisfies

$$\sigma_i[\bar{B}] = \sigma_i[B]exp\{(2n-k)\epsilon\}, \quad i = 1, \ldots, n,$$

where $k = 5/2$, 2, and 3/2 for Algorithms 1, 2, and 3, respectively.

Proof. This follows from Theorem 2 in Demmel and Kahan [6].

Remark. We are unaware of any other algorithm for computing singular values (or eigenvalues) of a matrix which is more accurate than the three primary algorithms described in this report. Note that any singular value of B, even tiny ones, can be determined to within $2n$ ULPs if the stipulated conditions are satisfied. However, the tiny error bounds are still pessimistic and often the computed singular values are within a couple of ULPs of the exact value.

4.6.2. mdqd algorithms. The modified algorithms contain the diagonal perturbational term Δ in the LDL^t factorization in addition to the relative perturbations on the elements of B. However, the relative perturbations are comparable to the Rutishauser's differential algorithms and, hence, we do not make further comments on the effects due to these relative perturbations.

The *mdsqd* algorithm is based on the LDL^t factorization of $(\bar{B}^t\bar{B} + \Delta) - \tau I$ rather than $\bar{B}^t\bar{B} - \tau I$ and, hence, this algorithm counts the number of eigenvalues of the perturbed matrix $(\bar{B}^t\bar{B} + \Delta)$. Since $\delta_i = \bar{e}_{i-1}(\epsilon_/ + \epsilon_-^{i-1} + \epsilon_/\epsilon_-^{i-1})$, $||\Delta||_2$ is (approximately) bounded by $2\max_{2\leq i\leq n}\bar{e}_{i-1}$ times the machine precision. From Weyl's theorem, we know that the diagonal perturbation cannot disturb the eigenvalues of $\bar{B}^t\bar{B}$ by more than $||\Delta||_2$ and, hence, the worst case perturbation of eigenvalues due to $||\Delta||_2$ are essentially determined by the maximal $\bar{e}_i$, which is within four ULPs of $\max_{2\leq i\leq n} e_{i-1}$. If the eigenvalue approximated by τ is greater than (or approximately equal to) $\max_{2\leq i\leq n} e_{i-1}$, then the Δ perturbation is tiny and, hence, the modified algorithm gives relative accuracy; otherwise, it provides eigenvalues with absolute accuracy.

A similar analysis is possible for the *mdpqd* algorithm. It can be shown that the worst case eigenvalue perturbation due to Δ in the LDL^t factorization of $\bar{B}\bar{B}^t+\Delta-\tau I$ is (approximately) bounded by $2\max_{2\leq i\leq n}\bar{q}_i$ times the machine precision. Thus, if the eigenvalue approximated by τ is greater than (or approximately equal to) $\max_{2\leq i\leq n} q_i$, then the Δ perturbation is relatively small. Hence, the modified algorithm provides relative accuracy for such τ.

5. Monotonicity of the inertia count.

5.1. Monotonic arithmetic. The integer $\nu(\tau)$ counts the eigenvalues of BB^t or B^tB which are less than τ. In the context of the *Golub–Kahan* form, $\nu(\rho)$ counts the eigenvalues of T_0 which are less than ρ. So by definition, ν is monotone increasing in τ or ρ, but that is in the context of exact arithmetic. Our aim is to prove that in floating-point arithmetic the computed values of ν are also monotonic in τ or ρ provided that the arithmetic unit obeys certain reasonable axioms. Such arithmetic units are said to provide monotonic arithmetic operations [17].

Axioms. For any machine representable normalized numbers a, b, and c:
$(+)$ if $a<\tilde{a}$ and $\mathrm{fl}(a+b)>\mathrm{fl}(\tilde{a}+c)$, then $b>c$;
$(\oplus)$ if $a\leq\tilde{a}$ and $\mathrm{fl}(a+b)>\mathrm{fl}(\tilde{a}+c)$, then $b>c$;
$(\times)$ if $a>0$ and $\mathrm{fl}(a*b)>\mathrm{fl}(a*c)$, then $b>c$;
$(\div)$ if $a>0$, $b\neq 0$, $c\neq 0$ and $\mathrm{fl}(\frac{a}{b})>\mathrm{fl}(\frac{a}{c})$, then $\begin{cases} c>b & \text{if } \mathrm{sign}\{b\}=\mathrm{sign}\{c\}, \\ c<b & \text{if } b>0\ ,\ c<0, \end{cases}$
if underflow (including denormalization), overflow, or infinity does not occur.

For a particular algorithm we use either the $(+)$ axiom or the $(\oplus)$ axiom. IEEE arithmetic [15], [18] always satisfies these axioms.

The monotonicity of the bisection algorithm for symmetric tridiagonal matrices was first established by Kahan in a well-known unpublished report [17]. Since the Golub–Kahan form is a symmetric tridiagonal matrix, albeit with a zero diagonal, the analysis of Kahan is readily applicable to the Golub–Kahan form. However, in our analysis we use the weaker axiom $(+)$ rather than $(\oplus)$. The monotonicity proofs of the modified differential algorithms are new.

5.2. The Golub–Kahan form. The integer $\nu(\rho)$ counts the number of eigenvalues of a given matrix that are less than ρ. So by definition, $\nu(\rho)$ is monotone increasing in ρ but that is in the context of exact arithmetic. In the present context $\nu(\rho)$ counts the number of negative values among $\{d_1,\ldots,d_n\}$, where d_i satisfies

$$d_i=-\rho-\frac{z_{i-1}}{d_{i-1}},\quad i=1,\ldots,n \tag{39}$$

and $d_0=1$ and $z_0=0$. Recall that $Z=(a_1^2,b_1^2,a_2^2,\ldots,b_{n-1}^2,a_n^2)=(q_1,e_1,q_2,\ldots,e_{n-1},q_n)$ and (39) comes from Algorithm 3.

We prove that the computed values of $\nu(\rho)$ are also monotonic in ρ provided that the arithmetic unit obeys the axioms. Our argument imitates that of Kahan [17] but expresses it differently. In exact arithmetic each $d_k(\rho)$, $k=2,\ldots,n$ is a rational function, monotonic decreasing in ρ for all ρ except for the poles. In practice we need appropriate definitions of poles and zeros.

Next-after. For each representable number x let $\vec{x}$ denote the next representable number exceeding x.

Pole. $[p,\vec{p}]$ is a pole of d_k if $d_k(p)<d_k(\vec{p})$. The signs may or may not differ.

Zero. $[z, \vec{z}]$ is a zero of d_k if $d_k(\vec{z}) \leq -\theta < 0 < \theta \leq d_k(z)$, where θ is a carefully chosen positive threshold below which computed absolute values of d_k are not permitted to fall.

Threshold. The recurrence (39) is supplemented with

$$\text{if} \quad -\theta \leq d_k < \theta, \quad \text{then} \quad d_k = -\theta \tag{40}$$

and $\theta \geq \eta$, where η is the smallest representable normalized positive number.

Remark. The cost of the above threshold operation *per se* is minute. However, in advanced architecture, such conditionals can become a bottleneck in the processing of pipelined instructions. If the system fully supports IEEE binary arithmetic (with signed zeros and infinities), then it is unnecessary to implement the threshold rule [5], [18]. We do not study such implementation issues in this report.

We begin the proof with two simple observations. For $k \geq 2$, between two distinct zeros of d_k, there must be a pole which changes sign but there could also be several poles that do not change sign. However, $d_1(\rho) = -\rho$ has a single, rather fat, zero at $[-\theta, \theta]$, because of the threshold, but no poles. The proof that computed $\nu_k(\rho)$ is actually monotone in ρ is composed of three lemmata of which the last two will provide a contradiction.

Lemma 5.1. *If $[p, \vec{p}]$ is a pole of d_k, $k > 1$, then $[p, \vec{p}]$ is either a zero or a same-sign pole of d_{k-1}.*

Proof. Since $[p, \vec{p}]$ is a pole of d_k,

$$d_k(p) = \text{fl}\left\{-p - \frac{z_{k-1}}{d_k} - 1(p)\right\} < d_k(\vec{p}) = \text{fl}\left\{-\vec{p} - \frac{z_{k-1}}{d_k} - 1(\vec{p})\right\}.$$

By the (+) axiom, $\text{fl}\{z_{k-1}/d_{k-1}(p)\} > \text{fl}\{z_{k-1}/d_{k-1}(\vec{p})\}$. There are two cases. If $\text{sign}\{d_{k-1}(p)\} = \text{sign}\{d_{k-1}(\vec{p})\}$ then, by the ($\div$) axiom $d_{k-1}(p) < d_{k-1}(\vec{p})$ and, by definition, $[p, \vec{p}]$ is a pole of d_{k-1}. On the other hand, since $z_i > 0$, the only way the signs can differ is if $d_{k-1}(p) \geq \theta > 0$ and $d_{k-1}(\vec{p}) \leq -\theta < 0$. By definition, $[p, \vec{p}]$ is a zero of d_k. □

Lemma 5.2. *If $\nu_i(\rho), i = 1, \ldots, k-1$ are each monotonically nondecreasing functions of ρ and if $[p, \vec{p}]$ is a pole of d_k, then $\nu_{k-1}(\vec{p}) - \nu_{k-1}(p) \geq 1$.*

Proof. Apply Lemma 5.1 repeatedly with $j = k, k-1, \ldots, 2,\ 1$, until there occurs a d_j for which $[p, \vec{p}]$ is a zero and not a pole. There must be such a j because d_1 has no poles. Let j now indicate that index; that is, $[p, \vec{p}]$ is a zero of d_j and a same-sign pole of d_i for $i = j+1, \ldots, k-1$. Clearly, $1 \leq j \leq k-1$. Now consider ν_j. Since $d_j(p) > 0 > d_j(\vec{p})$,

$$\begin{aligned} \nu_j(\vec{p}) &= \nu_{j-1}(\vec{p}) + 1, \text{ because } d_j(\vec{p}) < 0, \\ &\geq \nu_{j-1}(p) + 1, \text{ by the nondecreasing assumption on } \nu_{j-1}, \\ &= \nu_j(p) + 1, \text{ because } d_j(p) > 0. \end{aligned}$$

Thus,

$$\nu_j(\vec{p}) - \nu_j(p) \geq 1. \tag{41}$$

Since $[p, \vec{p}]$ is a same-sign pole of d_i for $i = j+1, \ldots, k-1$ we have

$$\nu_i(\vec{p}) - \nu_{i-1}(\vec{p}) = \nu_i(p) - \nu_{i-1}(p), \quad \text{sign}\{d_i(\vec{p})\} = \text{sign}\{d_i(p)\}.$$

Thus,

$$\nu_i(\vec{p}) - \nu_i(p) = \nu_{i-1}(\vec{p}) - \nu_{i-1}(p), \quad i = j+1, \ldots, k-1. \tag{42}$$

On adding all the (42) relations to (41) we find

$$\nu_{k-1}(\vec{p}) - \nu_{k-1}(p) = \nu_j(\vec{p}) - \nu_j(p) \geq 1. \qquad \square \tag{43}$$

Next we establish a consequence of the failure of monotonicity for ν_k.

LEMMA 5.3. *Suppose that $\nu_i(\rho)$ fails to be monotonic nondecreasing in ρ for some i and let k be the smallest such index. Then d_k has a pole $[y, \vec{y}]$ and $\nu_{k-1}(\vec{y}) - \nu_{k-1}(y) = 0$.*

Proof. If ν_k is not monotonic, then there is a y with $\nu_k(y) > \nu_k(\vec{y})$. Rewrite this as

$$[\nu_k(y) - \nu_{k-1}(y)] - [\nu_k(\vec{y}) - \nu_{k-1}(\vec{y})] > \nu_{k-1}(\vec{y}) - \nu_{k-1}(y). \tag{44}$$

The last inequality is a consequence of k's minimality. A careful consideration of $\text{sign}\{d_k(y)\}$ and $\text{sign}\{d_k(\vec{y})\}$ shows that the only way (44) can hold is if $d_k(y) < 0, d_k(\vec{y}) > 0$ and, by definition, $[y, \vec{y}]$ is, therefore, a pole of d_k. Since $\nu_{k-1}(\vec{y}) - \nu_{k-1}(y) < 1$, it must vanish as claimed. $\square$

THEOREM 5.4. *If axioms $(+)$ and $(\div)$ hold, then each computed function $\nu_k(\rho)$ is monotone nondecreasing in ρ in Algorithm* 3.

Proof. The conclusion of Lemma 5.3 contradicts the conclusion to Lemma 5.2. Consequently, there is no smallest index k for which ν_k fails to be monotonic. $\square$

5.3. Nondifferential qd algorithms. It is possible to prove the monotonicity property for nondifferential versions of qd algorithms by essentially following the proof given for Algorithm 3.

The *pqd* algorithm may be written in the form

$$\hat{q}_i = \alpha_i - \left(\tau + \frac{\beta_{i-1}}{\hat{q}_{i-1}}\right), \quad i = 1, \ldots, n, \tag{45}$$

where $\alpha_i = q_i + e_i, \beta_i = q_{i+1}e_i$. We count the negative values of $\hat{q}_i$, $\nu(\tau)$ for a given (eigenvalue) shift τ. The recursion for the *sqd* algorithm is identical to (45), except that the αs and the βs are different. For the *sqd* algorithm, they are given by $\alpha_i = q_i + e_{i-1}, \beta_i = q_i e_i$.

LEMMA 5.5. *If $[p, \vec{p}]$ is a pole of $\hat{q}_k$, then $[p, \vec{p}]$ is either a zero or a same-sign pole of $\hat{q}_{k-1}$.*

Proof. Since $[p, \vec{p}]$ is a pole of $\hat{q}_k$,

$$\hat{q}_k(p) = \text{fl}\left\{\alpha_k - \left[p - \frac{\beta_{k-1}}{\hat{q}_{k-1}}(p)\right]\right\} < \hat{q}_k(\vec{p}) = \text{fl}\left\{\alpha_k - \left[\vec{p} - \frac{z_{k-1}}{\hat{q}_{k-1}}(\vec{p})\right]\right\}.$$

As a consequence of the $(\oplus)$ axiom, $\text{fl}\{p + \beta_{k-1}/\hat{q}_{k-1}(p)\} > \text{fl}\{\vec{p} + \beta_{k-1}/\hat{q}_{k-1}(\vec{p})\}$. By applying the same axiom again, $\text{fl}\{\beta_{k-1}/\hat{q}_{k-1}(p)\} > \text{fl}\{\beta_{k-1}/\hat{q}_{k-1}(\vec{p})\}$. There are two cases. If $\text{sign}\{\hat{q}_{k-1}(p)\} = \text{sign}\{\hat{q}_{k-1}(\vec{p})\}$, then by the $(\div)$ axiom $\hat{q}_{k-1}(p) < \hat{q}_{k-1}(\vec{p})$ and, by definition, $[p, \vec{p}]$ is a pole of $\hat{q}_{k-1}$. On the other hand, since $\beta_i > 0$, the only way the signs can differ is if $\hat{q}_{k-1}(p) \geq \theta > 0$ and $\hat{q}_{k-1}(\vec{p}) \leq -\theta < 0$. By definition, $[p, \vec{p}]$ is a zero of $\hat{q}_k$. $\square$

We assume that $|\tilde{q}_k|$ below the threshold θ are set to $-\theta$ in *sqd*:

$$\text{if} \quad -\theta \leq \tilde{q}_k < \theta \quad \text{then} \quad \tilde{q}_k = -\theta. \tag{46}$$

For $\hat{q}_k$ in *pqd*, a similar threshold rule is used. The rest of the proof is identical to that of Algorithm 3 and, hence, we state the main result of this section without further ado.

THEOREM 5.6. *If axioms $(\oplus)$ and $(\div)$ hold, then each computed function $\nu_k(\tau)$ is monotone nondecreasing in τ for the (nondifferential) pqd and the (nondifferential) sqd.*

5.4. The msdqd algorithm. In this section we study the monotonicity of the *mdsqd* algorithm

$$t_i(\tau) = \tau + e_{i-1}\left\{\frac{q_{i-1}(\tau)}{\tilde{q}_{i-1}(\tau)} - 1\right\} \text{ for } i = 2, \ldots, n \text{ with } t_1 = \tau,$$

$$\tilde{q}_i(\tau) = q_i - t_i(\tau) \text{ for } i = 1, \ldots, n.$$

The following is the critical lemma which we have to prove.

LEMMA 5.7. *If $[p, \vec{p}]$ is a pole of $\tilde{q}_k$, $k > 1$, then $[p, \vec{p}]$ is either a zero or a same-sign pole of $\tilde{q}_{k-1}$.*

Proof. Since $[p, \vec{p}]$ is a pole of $\tilde{q}_k$,

$$\begin{aligned}\tilde{q}_k(p) &= \mathrm{fl}\left\{q_k - \left[p + e_{k-1}\left(\frac{q_{k-1}}{\tilde{q}_{k-1}}(p) - 1\right)\right]\right\} < \tilde{q}_k(\vec{p}) \\ &= \mathrm{fl}\left\{q_k - \left[\vec{p} + e_{k-1}\left(\frac{q_{k-1}}{\tilde{q}_{k-1}}(\vec{p}) - 1\right)\right]\right\},\end{aligned}$$

By invoking the ($\oplus$) axiom, $\mathrm{fl}\{p+e_{k-1}(q_{k-1}/\tilde{q}_{k-1}(p)-1)\} > \mathrm{fl}\{\vec{p}+e_{k-1}(q_{k-1}/\tilde{q}_{k-1}(\vec{p})-1)\}$. Using the same axiom again, $\mathrm{fl}\{e_{k-1}(q_{k-1}/\tilde{q}_{k-1}(p)-1)\} > \mathrm{fl}\{e_{k-1}(q_{k-1}/\tilde{q}_{k-1}(\vec{p})-1)\}$. By calling the ($\times$) axiom followed by the ($\oplus$) axiom, $\mathrm{fl}\{q_{k-1}/\tilde{q}_{k-1}(p)\} > \mathrm{fl}\{q_{k-1}/\tilde{q}_{k-1}(\vec{p})\}$. There are two cases. If $\mathrm{sign}\{\tilde{q}_{k-1}(p)\} = \mathrm{sign}\{\tilde{q}_{k-1}(\vec{p})\}$ then, by ($\div$) axiom, $\tilde{q}_{k-1}(p) < \tilde{q}_{k-1}(\vec{p})$ and, by definition, $[p, \vec{p}]$ is a pole of $\tilde{q}_{k-1}$. On the other hand, since $q_k > 0$, the only way the signs can differ is if $\tilde{q}_{k-1}(p) \geq \theta > 0$ and $\tilde{q}_{k-1}(\vec{p}) \leq -\theta < 0$. By definition, $[p, \vec{p}]$ is a zero of $\tilde{q}_k$. □

The rest of the proof is similar to the one previously described and, hence, we state our main result without further details.

THEOREM 5.8. *If axioms ($\oplus$), ($\times$), and ($\div$) hold then each computed function $\nu_k(\tau)$ is monotone non-decreasing in τ as given by the* mdsqd *algorithm.*

5.5. The mpdqd Algorithm. In this section we study the monotonicity of the *mdsqd* algorithm

$$s_i(\tau) = q_i\left\{1 - \frac{e_{i-1}}{\hat{q}_{i-1}}(\tau)\right\} - \tau \text{ for } i = 2, \ldots, n \text{ with } s_1 = q_1 - \tau,$$

$$\hat{q}_i(\tau) = e_i + s_i(\tau) \text{ for } i = 1, \ldots, n.$$

The following is essential to establish monotonicity.

LEMMA 5.9. *If $[p, \vec{p}]$ is a pole of $\hat{q}_k$, $k > 1$ then $[p, \vec{p}]$ is either a zero or a same-sign pole of $\hat{q}_{k-1}$.*

Proof. Since $[p, \vec{p}]$ is a pole of $\hat{q}_k$,

$$\begin{aligned}\hat{q}_k(p) &= \mathrm{fl}\left\{e_k + \left[q_k\left(1 - \frac{e_{k-1}}{\hat{q}_{k-1}}(p)\right) - p\right]\right\} < \hat{q}_k(\vec{p}) \\ &= \mathrm{fl}\left\{e_k + \left[q_k\left(1 - \frac{e_{k-1}}{\hat{q}_{k-1}}(\vec{p})\right) - \vec{p}\right]\right\},\end{aligned}$$

By invoking the ($\oplus$) axiom, $\mathrm{fl}\{q_k(1-e_{k-1}/\hat{q}_{k-1}(p))-p\} < \mathrm{fl}\{q_k(1-e_{k-1}/\hat{q}_{k-1}(\vec{p}))-$▮ $\vec{p}\}$. Using the ($\times$) axiom and followed by the ($\oplus$) axiom, $\mathrm{fl}\{e_{k-1}/\hat{q}_{k-1}(p)) - p\} > \mathrm{fl}\{e_{k-1}/\hat{q}_{k-1}(\vec{p})) - \vec{p}\}$. By calling the ($\oplus$) axiom again, $\mathrm{fl}\{e_{k-1}/\hat{q}_{k-1}(p))\} >$

Table 4
Parameters of IEEE machines.

	Single precision	Double precision
β	2	2
Ω	$(1-2\epsilon)*2^{128}$	$(1-2\epsilon)*2^{1024}$
$\hat{\Omega}$	2^{127}	2^{1023}
η	2^{-126}	2^{-1022}
ϵ	2^{-24}	2^{-53}

$\mathrm{fl}\{e_{k-1}/\hat{q}_{k-1}(\vec{p})\}$. There are two cases. If $\mathrm{sign}\{\hat{q}_{k-1}(p)\} = \mathrm{sign}\{\hat{q}_{k-1}(\vec{p})\}$ then by the $(\div)$ axiom, $\hat{q}_{k-1}(p) < \hat{q}_{k-1}(\vec{p})$ and, by definition, $[p, \vec{p}]$ is a pole of $\hat{q}_{k-1}$. On the other hand, since $e_{k-1} > 0$, the only way the signs can differ is if $\hat{q}_{k-1}(p) \geq \theta > 0$ and $\hat{q}_{k-1}(\vec{p}) \leq -\theta < 0$. By definition, $[p, \vec{p}]$ is a zero of $\hat{q}_k$. □

The rest of the proof is similar to the previously described and, hence, we state our main result for the *mdpqd* algorithm without further details.

Theorem 5.10. *If axioms* $(\oplus)$, $(\times)$, *and* $(\div)$ *hold then each computed function* $\nu_k(\tau)$ *is monotone nondecreasing in* τ *as given by the mdpqd algorithm.*

5.6. Rutishauser's differential qd algorithms. It does not seem to be possible to establish monotonicity for the Rutishauser's differential algorithms based solely on the axioms of monotonic arithmetic. In particular, there are no results corresponding to Lemmas 5.7 or 5.9. However, it might be possible to obtain weaker versions of these lemmata which could be adequate in practice.

6. Coping with underflow and overflow. In this section we propose ways to avoid overflow and reduce the occurrences of underflow of the variables in the three main algorithms. When underflow occurs, we try to minimize the adverse effects. These objectives are achieved by scaling a_i and b_i of the bidiagonal matrix. Our aim is to compute the singular values, even the tiny ones, to relatively high accuracy.

6.1. Machine parameters. To avoid being pedantic we assume that β, the base of the machine, is two. The following notation is used for other machine parameters:

Ω is the overflow threshold: the largest positive real number which can be represented on the machine;

$\hat{\Omega}$ is an approximation to Ω such that $\hat{\Omega} = 2^j \leq \Omega$ where the integer j is maximal;

η is the underflow threshold: the smallest positive real number which can be represented without denormalization on the machine;

$\epsilon = \beta^k = 2^k$ is the machine precision: k is the largest integer such that $\mathrm{fl}(1+\epsilon) = 1$.

Machine parameters for systems which conform to the IEEE data format [15] are displayed in Table 4. In practice, it is often possible to assume that $\hat{\Omega} = 1/\eta$ for machines which conform to the IEEE format. Also note that $\eta^{1/2} \ll \epsilon$ for IEEE standard machines.

Theorem 6.1. *If g and h are two machine representable normalized positive numbers, then underflow in the subtraction $g-h$ can occur only if* $\max(g,h) < \eta/\epsilon$.

Proof. If the computed value $g-h$ underflows then the exact $g-h$ satisfies $|g-h| < \eta$. By definition $g-h \neq 0$ (if $g=h$ then it is exact cancellation not underflow). Since g and h are two unequal representable numbers $|g-h| \geq \epsilon \max(g,h)$. Hence, $\epsilon \max(g,h) \leq |g-h| < \eta$ and thus $\max(g,h) < \eta/\epsilon$. □

6.2. The Golub–Kahan form. In the model of arithmetic used for error analysis we did not allow for underflow. We now show that underflow is not possible except for very tiny shifts. The proof is trivial.

Corollary 6.2. *If $\rho > 0$ then underflow in the subtraction $d_i = fl(-\rho - z_{i-1}/d_{i-1})$ can occur only if* $\max(\rho, -z_{i-1}/d_{i-1}) < \eta/\epsilon$ *with $d_{i-1} < 0$.*

Remark. In IEEE double precision, the bound for underflow is $\rho < \eta/\epsilon = 2^{-969}$ and in IEEE single precision $\rho < \eta/\epsilon = 2^{-102}$.

6.2.1. Systems without IEEE ∞. If IEEE signed ∞ is not supported, then to avoid overflow and to decrease the chance of underflow the matrix B has to be scaled. We also assume that there is no gradual underflow and, hence, there is no machine representable nonzero number between $-\eta$ and η.

Kahan [17] has studied this problem for symmetric tridiagonal matrices and our scaling is influenced by his work. Basically, the scaling factor is taken as 2^m, where m is an integer so that scaling does not introduce any round-off errors. The integer m is chosen such that it is maximal and satisfies $2^m \max(\max_i a_i, \max_i b_i) \leq \{\gamma\hat{\Omega}\}^{1/2}$ where γ satisfies $\epsilon > \gamma = 2^l \geq \eta$ and l is an integer.

Equivalently,

$$2^{2m} \max(\max_i q_i, \max_i e_i) \leq \gamma\hat{\Omega}. \tag{47}$$

The new scaled qd parameters are then given by

$$q_i \leftarrow (2^m a_i)^2, \;\; e_i \leftarrow (2^m b_i)^2. \tag{48}$$

If any scaled e_i underflows then the matrix will split into two submatrices. Similarly, if any q_i is zero, the matrix should be processed to extract this zero singular value. It is easily seen from (48) that any a_i or b_i, which is less than $\sqrt{\eta}2^{-m}$, forces the corresponding scaled q_i or e_i to underflow to zero.

If it is required to compute tiny singular values to high relative accuracy then γ should be assigned to a high value to avoid underflow of scaled q_i and e_i. This is particularly important if the magnitudes of the elements of the matrix B vary very widely. In general, $\sqrt{\eta}$ is a reasonable choice for γ. Note that $\epsilon > \sqrt{\eta} > \eta$ on machines conforming to the IEEE data representation.

We recall the threshold setting (40): if $-\theta \leq d_i < \theta$ then $d_i \leftarrow -\theta$. If $z_i = \gamma\hat{\Omega}$ and $|d_i| = \gamma$ then $z_i/|d_i| = \hat{\Omega}$. Hence, to avoid overflow of the quotient z_i/d_i, the threshold θ can be set to γ.

In [17], effects due to the threshold setting is accounted for by perturbing the ith diagonal value of the tridiagonal matrix. However, in our problem, the tridiagonal matrix has a zero diagonal and we are keen to respect this structural property. We now consider the worst case due to threshold setting.

Theorem 6.3. *Suppose that if $d_i = \theta$, then d_i is set to $-\theta$. If $\bar{z}_i$ is the perturbed value of z_i which will give $d_i = \theta$ without invoking the threshold rule, then $\theta/\rho = (\bar{z}_{i-1} - z_{i-1})/(\bar{z}_{i-1} + z_{i-1})$.*

Proof. This follows directly from the fact that $\theta = -\rho - z_{i-1}/d_{i-1}$ and $-\theta = -\rho - \bar{z}_{i-1}/d_{i-1}$. By removing d_{i-1} from the above two equations we get the claimed result. □

As a consequence of the above theorem we get the following bound for the shift ρ, which will lead to low relative errors when the threshold rule is invoked for d_i.

Corollary 6.4. *If $\rho \geq 2\theta/\epsilon$ then the required relative perturbation on z_{i-1} to account for the threshold rule is no more than one* ULP.

The Gershgorin theorem tells us that the eigenvalue with the largest magnitude of the scaled matrix T_0 is bounded by $\max_i(|a_i| + |b_i|)$. Thus, it is meaningless (in exact arithmetic) to use a shift ρ which has a magnitude greater than $2\sqrt{\gamma\Omega}$ for the scaled matrix. That is,

$$\rho \leq 2\sqrt{\gamma\Omega}. \tag{49}$$

TABLE 5
Scaling of the Golub–Kahan form.

Variable	Bounds	Remarks	See
q_i	$q_i \leq \gamma\hat{\Omega}$	typically $\gamma = \sqrt{\eta}$	(47)
e_i	$e_i \leq \gamma\hat{\Omega}$		(47)
d_i	$\gamma \leq \lvert d_i \rvert \leq \hat{\Omega}$		
ρ	$\rho \leq 2\sqrt{\gamma\hat{\Omega}}$	Gershgorin bound	(49)
ρ	$\rho \geq \eta/\epsilon$	no underflow	Corollary 6.2
ρ	$\rho \geq 2\theta/\epsilon$	tiny threshold error typically $\theta = \gamma$	Corollary 6.4

The bounds for the scaled variables are tabulated in Table 5.

We have included overflow control to Algorithm 3 and present it as Algorithm 4 where we have assumed that the singular value shift ρ is positive.

ALGORITHM 4. *Inertia of the Golub–Kahan form with overflow control for* $\rho > 0$

```
$d := -\rho$
$\nu := 1$
for $i = 1, 2n-1$
    if $-\theta \leq d < \theta$ then $d := -\theta$
    $d := -\rho - z_i/d$
    if $d < 0$ then $\nu := \nu + 1$
endfor
```

6.3. The differential stationary algorithm. For this algorithm we can show also that underflow is not possible except for very tiny shifts. The next result follows from Theorem 6.1.

COROLLARY 6.5. *If* $\tau > 0$, *then underflow due to addition in*

$$t_i = fl\{\tau + e_{i-1}\left(\frac{t_{i-1}}{\tilde{q}_{i-1}}\right)\} \tag{50}$$

can occur only if $\max\{\tau, e_{i-1}(t_{i-1}/\tilde{q}_{i-1})\} < \eta/\epsilon$ *with* $\tilde{q}_{i-1} < 0$.

Remark. In IEEE double precision, the bound for underflow is $\tau < \eta/\epsilon = 2^{-969}$ and in IEEE single precision $\tau < \eta/\epsilon = 2^{-102}$. Thus underflow is not possible except for very tiny shifts. Note that for the Golub–Kahan form, the bounds are expressed in terms of the singular value shift $\sigma(= \sqrt{\tau})$.

The scaling for the *mdsqd* algorithm is identical to the unmodified algorithm and, hence, it is not described separately.

6.3.1. Systems without IEEE ∞. If $\tilde{q}_i$ is tiny then the quotient $t_{i-1}/\tilde{q}_{i-1}$ in (50) can overflow. It is not difficult to establish that the maximum value of $|t_{i-1}/\tilde{q}_{i-1}|$ is when $|\tilde{q}_{i-1}|$ is minimal since $t_{i-1}/\tilde{q}_{i-1} = q_{i-1}/\tilde{q}_{i-1} - 1$. Suppose that the threshold rule $\tilde{q}_{i-1} \leftarrow -\theta$ if $-\theta \leq \tilde{q}_{i-1} < \theta$ is invoked with $\theta = \gamma$. Then we could choose the maximal integer m such that

$$2^{2m}(\max_{i=1,n} q_i) \leq \gamma\hat{\Omega}. \tag{51}$$

However, $e_{i-1}(t_{i-1}/\tilde{q}_{i-1})$ in (50) can also overflow. This calamity can be avoided by having the condition

$$2^{4m}(\max_{i=1,n-1} q_i e_i) \leq \gamma\hat{\Omega}. \tag{52}$$

Table 6
Scaling of the dsqd algorithm.

Variable	Bounds	Remarks	See
q_i	$q_i \leq \gamma\hat{\Omega}$	typically $\gamma = \sqrt{\eta}$	(51)
$q_i e_i$	$q_i e_i \leq \gamma\hat{\Omega}$		(51)
e_i	$e_i \leq \zeta\hat{\Omega}$	typically $\zeta = \gamma$	(53)
$\tilde{q}_i$	$\gamma \leq \lvert\tilde{q}_i\rvert \leq \hat{\Omega}$		
τ	$\tau \leq (\gamma + \zeta)\hat{\Omega} + \sqrt{\gamma\hat{\Omega}}$	Gershgorin bound	(54)
τ	$\tau \geq \eta/\epsilon$	no underflow	Corollary 6.5
q_i	$q_i \geq 2\theta/\epsilon$	tiny threshold error typically $\theta = \gamma$	Corollary 6.7

Table 7
Scaling of the dpqd algorithm.

Variable	Bounds	Remarks
q_i	$q_i \leq \zeta\hat{\Omega}$	typically $\gamma = \sqrt{\eta}$, $\zeta = \gamma$
$q_{i+1}e_i$	$q_{i+1}e_i \leq \gamma\hat{\Omega}$	
e_i	$e_i \leq \gamma\hat{\Omega}$	
$\hat{q}_i$	$\gamma \leq \lvert\hat{q}_i\rvert \leq \hat{\Omega}$	
τ	$\tau \leq (\gamma + \zeta)\hat{\Omega} + \sqrt{\gamma\hat{\Omega}}$	Gershgorin bound
τ	$\tau \geq \eta/\epsilon$	no underflow
e_i	$e_i \geq 2\theta/\epsilon$	tiny threshold error typically $\theta = \gamma$

To bound the origin of the largest Gershgorin circle, we also impose the extra condition

$$2^{2m}(\max_{i=1,n-1} e_i) \leq \zeta\hat{\Omega}, \tag{53}$$

where typically $\zeta = \gamma$. However, with extra care, values greater than γ may be possible for ζ. The Gershgorin upper limit for the eigenvalues of the scaled matrix B^tB is then given by

$$\lambda_{\max}[B^tB] \leq q_i + e_{i-1} + \sqrt{q_i e_i} + \sqrt{q_{i-1}e_{i-1}} \leq (\gamma + \zeta)\hat{\Omega} + 2\sqrt{\gamma\hat{\Omega}}. \tag{54}$$

We now study the effects due to the threshold on $\tilde{q}_i$.

Theorem 6.6. *Suppose that when $\tilde{q}_i = \theta$ then $\tilde{q}_i$ is set to $-\theta$. If $\bar{q}_i$ is the perturbed value of q_i, which will force $\tilde{q}_i$ to be $-\theta$ without the threshold rule, then $2\theta/q_i = (q_i - \bar{q}_i)/q_i$.*

Proof. This follows directly from the fact that $\theta = q_i - t_i$ and $-\theta = \bar{q}_i - t_i$. By removing t_i from the above two equations we get the claimed result. □

As a consequence of the above theorem we get the following bound for q_i, which will lead to low relative errors if the threshold rule is used for $\tilde{q}_i$.

Corollary 6.7. *If $q_i \geq 2\theta/\epsilon$ then the required relative perturbation on q_i to account for the threshold rule is no more than one* ULP.

Bounds are tabulated in Table 6.

6.4. The differential progressive algorithm.

6.4.1. Systems without IEEE ∞. Scaling of the pqd algorithms is similar to the sqd algorithm and, hence, details are omitted; Table 7 gives a summary.

Acknowledgments. Invigorating discussions with Professor Gene Golub are greatly appreciated. Jeremy Du Croz and Neil Swindells commented on drafts of the paper. Arithmetic issues were discussed with Michael Pont. The author wishes to thank Professor Gene Golub and Jeremy Du Croz for the Ward and Gray reference.

REFERENCES

[1] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen, *LAPACK Users' Guide, Release* 2.0, SIAM, Philadelphia, PA, 1995.

[2] J. Barlow and J. Demmel, *Computing accurate eigensystems of scaled diagonally dominant matrices*, SIAM J. Numer. Anal., 27 (1990), pp. 762–791.

[3] W. Barth, R. S. Martin, and J. H. Wilkinson, *Calculation of the eigenvalues of a symmetric tridiagonal matrix by the method of bisection*, in Handbook for Automatic Computation, Vol. II: Linear Algebra, J. H. Wilkinson and C. Reinsch, eds., Springer-Verlag, Berlin, 1971, pp. 249–256.

[4] E. Beltrami, *Sulle funzioni bilineari*, Giornale di matematiche ud uso Degli Studenti Delle Universita, 11 (1873), pp. 98–106.

[5] J. Demmel and W. Gragg, *On computing accurate singular values and eigenvalues of acyclic matrices*, Linear Algebra Appl., 185 (1993), pp. 203–218.

[6] J. Demmel and W. Kahan, *Accurate singular values of bidiagonal matrices*, SIAM J. Sci. Statist. Comput., 11 (1990), pp. 873–912.

[7] J. W. Demmel, I. Dhillon, and H. Ren, *On the correctness of some bisection-like parallel algorithms in floating point arithmetic*, Electron. Trans. Numer. Anal., 3 (1995), pp. 116–149.

[8] K. V. Fernando, *Accurate BABE Factorization of Tridiagonal Matrices for Eigenproblems*, Tech. report TR5/95, Numerical Algorithms Group Ltd., Wilkinson House, Jordan Hill, Oxford OX2 8DR, 1995.

[9] K. V. Fernando, *Computing an eigenvector of a tridiagonal matrix: Part* I: *Basic results*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 1013–1034.

[10] K. V. Fernando and B. N. Parlett, *Accurate singular values and differential qd algorithms*, Numer. Math., 67 (1994), pp. 191–229.

[11] W. Givens, *Numerical Computation of the Characteristic Values of a Real Symmetric Matrix*, Tech. report ORNL-1574, Oak Ridge National Laboratory, Oak Ridge, TN, 1954.

[12] G. H. Golub, *Least squares, singular values and matrix approximations*, Aplikace Matematiky, 13 (1987), pp. 44–51.

[13] G. H. Golub and W. Kahan, *Calculating the singular values and pseudo-inverse of a matrix*, SIAM J. Numer. Anal., 2 (1965), pp. 205–224.

[14] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, MD, 1989.

[15] Institute for Electrical and Electronics Engineers, *Standard for Binary Floating Point Arithmetic*, Vol. Standard 754-1985, ANSI/IEEE, New York, 1985.

[16] C. Jordan, *Mèmoire sur les formes bilinéaires*, J. Math. Pures Appl., 19 (1874), pp. 35–54.

[17] W. Kahan, *Accurate Eigenvalues of a Symmetric Tri-Diagonal Matrix*, Tech. report CS 41, Computer Science Department, Stanford University, CA, 1966.

[18] W. Kahan, *Lecture Notes on the Status of the IEEE Standard* 754 *for Binary Floating-Point Arithmetic*, in preparation, Electrical Engineering and Computer Science Department, University of California, Berkeley, CA, May 1996.

[19] L. Lorentzen and H. Waadeland, *Continued Fractions with Applications*, North–Holland, Amsterdam, 1992.

[20] R. Mathias, *The stability of parallel prefix matrix multiplication*, SIAM J. Sci. Comput., 16 (1996), pp. 956–973.

[21] H. Rutishauser, *Vorlesungen über numerische Mathematik*, Birkhäuser-Verlag, Basel, 1976.

[22] H. Rutishauser, *Lectures on Numerical Mathematics*, Birkhäuser, Boston, 1990.

[23] B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow, Y. Ikebe, V. C. Klema, and C. B. Moler, *Matrix Eigensystem Routines – EISPACK Guide*, Lecture Notes in Comput. Sci. 6, Springer-Verlag, Berlin, 1976.

[24] G. W. Stewart, *On the early history of the singular value decomposition*, SIAM Rev., 35 (1993), pp. 551–566.

[25] G. W. Stewart and J.-G. Sun, *Matrix Perturbation Theory*, Academic Press, San Diego, CA, 1990.

[26] H. S. Wall, *Analytic Theory of Continued Fractions*, Van Nostrand, New York, 1948.

[27] R. C. Ward and L. J. Gray, *Eigensystem computation for skew-symmetric matrices and a class of symmetric matrices*, ACM Trans. Math. Software, 4 (1978), pp. 278–285.

[28] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, 1965.

SIAM J. Matrix Anal. Appl.
Vol. 20, No. 2, pp. 400–419

THE CHEBYSHEV POLYNOMIALS OF A MATRIX[*]

KIM-CHUAN TOH[†] AND LLOYD N. TREFETHEN[‡]

Abstract. A Chebyshev polynomial of a square matrix A is a monic polynomial p of specified degree that minimizes $\|p(A)\|_2$. The study of such polynomials is motivated by the analysis of Krylov subspace iterations in numerical linear algebra. An algorithm is presented for computing these polynomials based on reduction to a semidefinite program which is then solved by a primal-dual interior point method. Examples of Chebyshev polynomials of matrices are presented, and it is noted that if A is far from normal, the lemniscates of these polynomials tend to approximate pseudospectra of A.

Key words. matrix polynomial, Chebyshev polynomial of a matrix, semidefinite programming, Krylov subspace iteration

AMS subject classifications. 15A60, 41A99, 65F10, 90C99

PII. S0895479896303739

1. Introduction.

Let A be an $N \times N$ matrix and n a nonnegative integer. The degree n *Chebyshev polynomial of* A is the unique monic polynomial p_n^* of degree n such that

$$\|p_n^*(A)\| = \text{minimum}, \tag{1}$$

where $\|\cdot\|$ denotes the matrix 2-norm. To be precise, p_n^* is unique provided that n is less than or equal to the degree of the minimal polynomial of A; otherwise we have $p_n^*(A) = 0$, and the problem ceases to be interesting.

This notion of the polynomial that minimizes $\|p(A)\|$ seems so simple and natural that one would expect it to be a standard one. We suspect it may have been considered before, perhaps decades ago in the literature of approximation theory. Nevertheless, we have been unable to find any literature on this problem before our 1994 paper with Greenbaum [7]. In that paper, Chebyshev polynomials of matrices are defined, and it is proved that they exist (obvious by compactness) and that they are unique under the condition just mentioned (not obvious).

Even if they are not discussed explicitly, Chebyshev polynomials of matrices are never far away from any discussion of convergence of Krylov subspace iterations in numerical linear algebra. For these iterations, convergence depends on certain vector norms $\|p(A)q\|$ being as small or nearly as small as possible, where q is a starting vector. Most of the convergence properties of applied interest do not depend too strongly on q, and thus it is the near-minimality of $\|p(A)\|$ that is often the heart of the matter [22]. For finding eigenvalues, the principal iterative method in this category is the Arnoldi iteration, which becomes the Lanczos iteration if A is real and symmetric. For solving systems of equations, the analogous methods include GMRES, biconjugate gradients, CGS, QMR, and Bi-CGSTAB in the general case and conjugate gradients if A is symmetric positive definite [6]. (For systems of equations, the notion of a Chebyshev polynomial of A should be normalized differently by the

[*] Received by the editors May 17, 1996; accepted for publication (in revised form) by R. Freund August 13, 1997; published electronically October 26, 1998. This research was supported by NSF grant DMS-9500975CS and DOE grant DE-FG02-94ER25199.

http://www.siam.org/journals/simax/20-2/30373.html

[†] Department of Mathematics, National University of Singapore, 10 Kent Ridge Crescent, Singapore 119260 (mattohkc@math.nus.sg).

[‡] Oxford University Computing Laboratory, Wolfson Building, Parks Road, Oxford OX1 3QD, UK (Nick.Trefethen@comlab.ox.ac.uk).

condition $p(0) = 1$ instead of the condition that p is monic. In [7], a Chebyshev polynomial of a matrix is called an *ideal Arnoldi polynomial,* and its analogue with this other normalization is called an *ideal GMRES polynomial.*)

The motivation for the term "Chebyshev polynomial of a matrix" is as follows. All readers will be familiar with the classical Chebyshev polynomials $\{T_n\}$, which are 2^{n-1} times monic polynomials of minimal $\|\cdot\|_\infty$-norm on the interval $[-1, 1]$. This notion was generalized by Faber in 1920 to the idea of the *Chebyshev polynomials of* S, where S is a compact set in the complex plane $\mathbb{C}$: the monic polynomials of minimal $\|\cdot\|_\infty$-norm over S [5], [24]. Now suppose that A is a hermitian or, more generally, a normal matrix, having a complete set of orthogonal eigenvectors. Then by a unitary reduction to diagonal form, it is easily shown that the nth Chebyshev polynomial of A as defined by (1) is precisely the nth Chebyshev polynomial of S in this latter sense, where S is the spectrum of A. Such a polynomial can be computed, for example, by generalizations of the Remez algorithm [15].

Chebyshev polynomials of normal matrices, then, are trivial; the matrix problem reduces to a scalar problem. But what if A is an arbitrary square matrix, with nonorthogonal eigenvectors or perhaps no complete set of eigenvectors? This is the subject of this paper, and our purpose is twofold.

First, we describe an algorithm for computing Chebyshev polynomials of matrices. The optimization problem implicit in (1) is far from smooth, and unless the degree is very small, these problems are quite difficult if approached by general methods of unconstrained optimization. The algorithm we describe, which we believe is the first to have been developed for this problem, is based instead on interior point methods for semidefinite programming. With this algorithm, we can reliably compute Chebyshev polynomials for matrices of order ≤ 50 in less than a minute on workstations available in 1996. No parameters are involved that must be tuned. We should mention, however, that although our algorithm is reasonably fast, it is not fast enough to easily handle matrix dimensions of the order of 1,000 or more.

Second, we present computed examples, the first we know of to have been published. A few numerical coefficients are listed for possible comparison by later authors, but our main aim is to give insight into the behavior of Chebyshev polynomials of matrices, largely with the aid of pictures. A natural question is, how are the coefficients of the polynomials affected by the degree and nature of the nonnormality of A? For a partial answer, we plot lemniscates $|p_n^*(z)| = \text{constant}$ of our polynomials and find that in many cases they approximate pseudospectra of A.

2. Reduction to a semidefinite program. Let $\{B_0, B_1, \ldots, B_n\}$ be a linearly independent set of matrices in $\mathbb{C}^{N\times N}$. The Chebyshev problem (1) is a special case of a norm minimization problem involving linear functions of matrices:

$$\min_{x\in\mathbb{C}^n} \left\| \sum_{k=1}^{n} x_k B_k - B_0 \right\|. \tag{2}$$

For our special case, $B_0 = A^n$ and

$$B_k \;=\; A^{k-1}, \quad k = 1, \ldots, n,$$

and the numbers x_k are the coefficients (actually their negatives) of the Chebyshev polynomial of A.

It is well known that (2) can be expressed as a semidefinite program [11], [23]. We shall not show in detail how this is done. One difference between our work and what has been done before is that the existing literature, as far as we are aware, considers only real matrices.

THEOREM 1. *The norm minimization problem* (2) *is equivalent to the following semidefinite program involving hermitian matrices:*

$$(3)\qquad \begin{aligned} &-\max_{x\in\mathbb{C}^n,\ \lambda\in\mathbb{R}} \ \lambda \\ &s.t.\ \ \sum_{k=1}^{n} (\alpha_k\, A_k \ + \ \beta_k\, A_{n+k}) \ + \ \lambda\, A_{2n+1} \ + \ Z \ = \ A_0, \\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad Z \ \geq \ 0, \end{aligned}$$

where $\alpha_k = \mathrm{Re}(x_k)$*,* $\beta_k = \mathrm{Im}(x_k)$*,*

$$A_{2n+1} \ = \ \begin{pmatrix} I & 0 \\ 0 & I \end{pmatrix}, \quad A_0 \ = \ \begin{pmatrix} 0 & B_0 \\ B_0^* & 0 \end{pmatrix},$$

$$(4)\qquad A_k \ = \ \begin{pmatrix} 0 & B_k \\ B_k^* & 0 \end{pmatrix}, \quad A_{n+k} \ = \ \begin{pmatrix} 0 & iB_k \\ -iB_k^* & 0 \end{pmatrix},$$

$k = 1, \dots, n$*, and* $Z \geq 0$ *means that* Z *is positive semidefinite.*

Proof. Problem (2) is equivalent to the problem of minimizing $-\lambda$ such that

$$(5)\qquad \left\| \sum_{k=1}^{n} x_k B_k \ - \ B_0 \right\| \ \leq \ -\lambda.$$

Using the fact that for any $M \in \mathbb{C}^{N\times N}$,

$$\|M\| \ = \ \lambda_{\max}\left(\begin{bmatrix} 0 & M \\ M^* & 0 \end{bmatrix} \right),$$

where $\lambda_{\max}(\cdot)$ denotes the maximum eigenvalue, (5) can be rewritten as

$$\lambda_{\max}\left(\begin{bmatrix} 0 & B(x) \\ B(x)^* & 0 \end{bmatrix} \right) \ \leq \ -\lambda,$$

where $B(x) = \sum_{k=1}^{n} x_k B_k - B_0$. But this is equivalent to

$$\begin{pmatrix} 0 & B(x) \\ B(x)^* & 0 \end{pmatrix} \ + \ \lambda \begin{pmatrix} I & 0 \\ 0 & I \end{pmatrix} + \ Z \ = \ 0, \quad Z \geq 0.$$

By writing this equation out in full, we get (3). □

3. Transformation to a better-conditioned basis. Before we discuss how the semidefinite program (3) can be solved by interior point methods, we must address the issue of change of basis in (2), as the numerical stability of these algorithms depends on the conditioning of the basis $\{B_1, \dots, B_m\}$. This is an essential point for the computation of Chebyshev polynomials of matrices. The power basis is usually highly ill conditioned, as can be seen by considering the special case of a diagonal matrix, where we get the phenomenon of ill-conditioning of the basis of monomials $\{x^k\}$, familiar in approximation theory. For numerical stability in most cases, the power basis must be replaced by a better-conditioned alternative.

Suppose $\{\widehat{B}_0, \widehat{B}_1, \dots, \widehat{B}_n\}$ is another linearly independent set of matrices in $\mathbb{C}^{N\times N}$ related linearly to $\{B_0, B_1, \dots, B_n\}$ by

$$\begin{aligned} [\, B_1 \,|\, \cdots \,|\, B_n \,] &= \left[\, \widehat{B}_1 \,|\, \cdots \,|\, \widehat{B}_n \,\right] T, \\ B_0 &= c\, \widehat{B}_0 \ - \ \left[\, \widehat{B}_1 \,|\, \cdots \,|\, \widehat{B}_n \,\right] t, \end{aligned}$$

where T is an $n \times n$ nonsingular matrix, t is an n-vector, and c is a nonzero scalar. (The notation here means that $B_k = T_{1k}\widehat{B}_1 + \cdots + T_{nk}\widehat{B}_n$ for $k \geq 1$ and $B_0 = c\widehat{B}_0 - (t_1\widehat{B}_1 + \cdots + t_n\widehat{B}_n)$.) The following theorem describes how (2) is modified by this change of basis. The proof is straightforward, and we shall omit it.

THEOREM 2. *The minima*

$$\min_{x \in \mathbb{C}^n} \left\| \sum_{k=1}^{n} x_k B_k \; - \; B_0 \right\|$$

and

$$|c| \min_{\hat{x} \in \mathbb{C}^n} \left\| \sum_{k=1}^{n} \hat{x}_k \widehat{B}_k \; - \; \widehat{B}_0 \right\|$$

are the same, and the unique vectors x and $\hat{x}$ that achieve them are related by

$$\hat{x} \; = \; \frac{1}{c}\,(Tx + t).$$

We are aware of three choices of basis that are particularly attractive for practical computations.

Scaled power basis. Suppose $B_0, \ldots, B_n$ are given as in (2). A simple way to get a better conditioned basis is to scale the norm of A to 1. With such a scaling, we have $\widehat{B}_0 = B_0/\alpha_0$ and

$$\widehat{B}_k \; = \; B_k/\alpha_k, \quad k = 1, \ldots, n,$$

where $\alpha_0 = \|A\|^n$ and $\alpha_k = \|A\|^{k-1}$, $k = 1, \ldots, n$. Hence $T = \operatorname{diag}(\alpha_1, \ldots, \alpha_n)$, $t = 0$, and $c = \alpha_0$ in (2).

Faber polynomial basis. Even the best-scaled power basis is often highly ill conditioned. A more powerful idea is to consider the basis $\{F_0(A), \ldots, F_n(A)\}$ defined by the Faber polynomials $F_0, \ldots, F_n$ associated with some region Ω in the complex plane containing the spectrum of A. The Faber polynomials $\{F_n\}$ are the natural analogues for a general region Ω in $\mathbb{C}$ of the monomials $\{z^n\}$ for the unit disk or the Chebyshev polynomials $\{T_n\}$ for $[-1, 1]$; see [2]. In most cases, $\{F_n(A)\}$ will be far better conditioned than any power basis.

For the Faber basis, the matrix T in (2) is upper triangular, with columns containing the coefficients of $F_0, \ldots, F_{n-1}$. The scalar c is the positive number $\operatorname{cap}(\Omega)^n$, where $\operatorname{cap}(\Omega)$ is the logarithmic capacity of Ω. The vector t is the vector of coefficients of the expansion of the degree $n-1$ polynomial $cF_n(z) - z^n$ in terms of $F_0(z), \ldots, F_{n-1}(z)$.

Of course, one must choose a region Ω for which the associated Faber polynomials can be obtained either analytically or numerically. If Ω is chosen to be an ellipse or an interval, then the Faber polynomials are simply the scaled Chebyshev polynomials $\{T_n\}$. More generally, if Ω is chosen to be a polygonal domain, the Faber polynomials can be computed numerically via Schwarz–Christoffel mapping. We have used the MATLAB Schwarz–Christoffel toolbox for this purpose, due to Driscoll [4].

Orthonormal basis. Finally, our third idea is a more elementary one, but powerful in practice. One may simply orthonormalize the power basis $\{I, A, \ldots, A^n\}$ with respect to the "trace inner product" $\langle A, B\rangle = \operatorname{tr}(AB^*)$ in $\mathbb{C}^{N \times N}$ to obtain a basis $\{Q_1, Q_2, \ldots, Q_{n+1}\}$ that is typically well conditioned even in the 2-norm. This can be done by a modified Gram–Schmidt procedure similar to that used in the Arnoldi iteration:

```
$Q_1 = N^{-1/2} I$
for $k = 1 : n$
    $V = AQ_k$
    for $j = 1 : k$
        $h_{jk} = \langle V, Q_j \rangle$
        $V = V - h_{jk} Q_j$
    $h_{k+1,k} = \langle V, V \rangle^{1/2}$
    $Q_{k+1} = V / h_{k+1,k}$.
```

To obtain the matrix T in (2), we note that there is a unique $(n+1) \times (n+1)$ upper triangular matrix R such that

$$[\, I \,|\, A \,|\, \cdots \,|\, A^n \,]\; R = [\, Q_1 \,|\, Q_2 \,|\, \cdots \,|\, Q_{n+1} \,],$$

and the columns of R can be computed from the following recurrence relation (in MATLAB notation):

$$\begin{aligned} R(1,1) &= 1/\sqrt{N}, \\ h_{k+1,k}\,\mathbf{r_{k+1}} &= \begin{bmatrix} 0 \\ \mathbf{r_k} \end{bmatrix} - \begin{bmatrix} R(1:k, 1:k)\,\mathbf{h_k} \\ 0 \end{bmatrix}, \end{aligned}$$

where $\mathbf{r_k} = R(1:k, k)$ and $\mathbf{h_k} = (h_{1k}, \ldots, h_{kk})^T$, for $k = 1, \ldots, n$. It is now easy to see that

$$T \;=\; R^{-1}(1:n, 1:n), \quad c \;=\; R^{-1}(n+1, n+1), \quad t \;=\; R^{-1}(1:n, n+1),$$

again in MATLAB notation.

For simplicity, we use the orthonormal basis in the examples reported in this paper. Although it is more expensive to compute than the other two bases, the amount of time taken remains small compared to the time required for solving (3).

We note that transformation to a better-conditioned basis does not eliminate any ill-conditioning that is inherent in the Chebyshev minimization problem itself.

4. Solution by primal-dual interior point method. Assuming a suitable basis has been chosen, we now turn to the problem of how (3) can be solved by interior point methods similar to those in linear programming, specifically, by Mehrotra-type primal-dual predictor-corrector algorithms. Extensive research has been done on both the algorithms and the theory of semidefinite programming (SDP). We refer the reader to [1], [9], [10], [11], [12], [16], [23], and [25] for details.

A general SDP has the form

$$\begin{aligned} &\text{(D)}: \quad \max_{y \in \mathbb{R}^n} \; b^T y \\ &\text{s.t.} \quad \sum_{k=1}^{n} y_k A_k \;+\; Z \;=\; C, \qquad Z \;\geq\; 0, \end{aligned} \tag{6}$$

where C, Z, A_k, $k = 1, \ldots, n$, are $N \times N$ hermitian matrices and $b \in \mathbb{R}^n$. The idea behind an interior point method is to use a suitable barrier function, $-\log\det(Z)$ in the case of SDP, to transform the semidefinite constrained convex problem (D) into a parametrized family (by μ) of equality constrained convex problems whose optimal solutions $(X(\mu), y(\mu), Z(\mu))$ satisfy the optimality conditions

$$\begin{aligned} \sum_{k=1}^{n} y_k A_k \ + \ Z &= C, \\ \langle X,\, A_k\rangle &= b_k, \quad k = 1, \ldots, n, \\ XZ &= \mu I, \end{aligned} \tag{7}$$

where X and Z are hermitian positive definite. The parameter $\mu > 0$ is to be driven explicitly to zero (as fast as possible), and in the limit $\mu \to 0$, an optimal solution of (6) is obtained.

Mehrotra-type primal-dual predictor-corrector algorithms essentially consist of a sequence of modified Newton iterations. Usually, one step of Newton's iteration is applied to (7) for each new μ.

It is readily shown that application of Newton's method to (7) gives rise to the equations

$$\begin{aligned} \sum_{k=1}^{n} (\Delta y)_k\, A_k \ + \ \Delta Z \ &= \ C \ - \ Z - \sum_{k=1}^{n} y_k\, A_k, \\ \langle \Delta X,\, A_k\rangle \ &= \ b_k \ - \ \langle X,\, A_k\rangle, \quad k = 1, \ldots, n, \end{aligned} \tag{8}$$

$$\Delta X\, Z \ + \ X\, \Delta Z \ = \ \mu I \ - \ XZ. \tag{9}$$

In order to keep ΔX hermitian (this is desirable since the fundamental objects in an SDP are hermitian matrices), (9) is usually symmetrized with respect to an invertible matrix P, whereupon it becomes

$$P(\Delta X\, Z + X\, \Delta Z)P^{-1} \ + \ P^{-*}(Z\, \Delta X + \Delta Z\, X)P^{*} \ = \ R, \tag{10}$$

where

$$R \ = \ 2\mu I \ - \ P(XZ)P^{-1} \ - \ P^{-*}(ZX)P^{*}. \tag{11}$$

Different choices of P give rise to different Newton steps. For example, $P = I$ gives rise to what is known as the Alizadeh–Haeberly–Overton (AHO) direction [1]; $P = Z^{1/2}$ gives rise to the Monteiro direction [10]; and $P = W^{-1/2}$, where $W = Z^{-1/2}(Z^{1/2}XZ^{1/2})^{1/2}Z^{-1/2}$, gives rise to the Nesterov–Todd (NT) direction [12].

The general algorithmic framework of a Mehrotra-type predictor-corrector method is as follows.

Algorithm. Given an initial iterate (X^0, y^0, Z^0) with X^0, Z^0 positive definite, for $k = 0, 1, \ldots,$

(Let the current and the next iterate be (X, y, Z) and (X^+, y^+, Z^+), respectively.)

1. Predictor step. Compute the Newton step $(\delta X, \delta y, \delta Z)$ from (8) and (10) with $\mu = 0$ in (11).
2. Determine the real parameter $\mu = \sigma\, \langle X,\, Z\rangle / n$, where

$$\sigma \ = \ \frac{\langle X + \alpha\, \delta X,\, Z + \beta\, \delta Z\rangle^2}{\langle X,\, Z\rangle^2}.$$

Here α and β are suitable steplengths chosen to ensure that $X + \alpha\,\delta X$ and $Z + \beta\,\delta Z$ are positive definite. Generally, α and β have the form

$$\alpha \;=\; \min\left(1, \frac{-\tau}{\lambda_{\min}(X^{-1}\delta X)}\right), \quad \beta \;=\; \min\left(1, \frac{-\tau}{\lambda_{\min}(Z^{-1}\delta Z)}\right), \tag{12}$$

where $\tau < 1$ is a control parameter.

3. Corrector step. Compute the Newton step $(\Delta X, \Delta y, \Delta Z)$ from (8) and (10) with the right-hand side matrix R given by

$$R \;=\; 2\mu I \;-\; P(XZ + \delta X\delta Z)P^{-1} \;-\; P^{-*}(ZX + \delta Z\delta X)P^{*}.$$

4. Update (X, y, Z) to (X^{+}, y^{+}, Z^{+}) by

$$X^{+} \;=\; X \;+\; \alpha\,\Delta X, \quad y^{+} \;=\; y \;+\; \beta\,\Delta y, \quad Z^{+} \;=\; Z \;+\; \beta\,\Delta Z,$$

where α and β are defined by (12) with δX, δZ replaced by ΔX, ΔZ.

We shall not discuss implementation details of the above algorithm—for example, how to solve efficiently for the search directions $(\delta X, \delta y, \delta Z)$ and $(\Delta X, \Delta y, \Delta Z)$ from the linear systems of $2N^2 + n$ equations (8) and (10); we refer the reader to [16] for such details. Instead, we just note that the search directions are typically computed via a Schur complement equation. For such an implementation, each iteration has a complexity of $O(nN^3) + O(n^2N^2)$, which is equal to $O(nN^3)$ for our Chebyshev approximation problem since $n < N$. Computations have shown that careful implementations of the predictor-corrector algorithm that use a Schur complement equation can typically reduce the duality gap of an SDP to about $\epsilon_{\mathrm{mach}}^{2/3}$ for the three search directions mentioned above, namely, the AHO, Monteiro, and NT directions. For these three directions, each iteration has a complexity of at most $12nN^3$, and the number of iterations needed to reduce the duality gap by a factor of 10^{10} seldom exceeds 20.

In all of our computations we use the NT direction for the following reasons. Although the orders of complexity for computing these three directions are the same, computing the AHO direction is about twice as expensive as computing the Monteiro or NT directions. Of the latter two, the NT direction has the virtue of being primal-dual symmetric. This implies that primal-dual predictor-corrector algorithms based on the NT direction are likely to be more robust than those based on the Monteiro direction, in the sense that the problems of stagnation, such as taking very small steplengths, are less likely to occur.

5. The special case when *A* is normal. It is worth setting down the form our algorithm takes in the special case where A is normal, i.e., unitarily diagonalizable. As we have already mentioned in the introduction, we may assume in this case that A is diagonal, so that the Chebyshev problem (1) reduces to the classical Chebyshev approximation problem on the spectrum $\Lambda(A)$ of A, i.e.,

$$\|p_n^*(A)\| \;=\; \|p_n^*\|_{\Lambda(A)} \;=\; \text{minimum}.$$

For this special case, the Chebyshev polynomials of A can be computed cheaply by the predictor-corrector algorithm discussed in the last section by exploiting the block diagonal structure present in the associated SDP problem.

As in the general case, we consider the norm minimization problem (2), but the matrices B_k, $k = 0, \ldots, n$, are now diagonal: $B_k = \mathrm{diag}(d_k)$ for each k. Since the

2-norm of a diagonal matrix is the $\|\cdot\|_\infty$-norm of its diagonal vector, (2) is equivalent to the minimax problem

$$\min_{x\in\mathbb{C}^n}\max_{1\le l\le N}\left|\sum_{k=1}^{n} x_k d_k^{(l)} - d_0^{(l)}\right|, \tag{13}$$

where $d_k^{(l)}$ denotes the lth component of the N-vector d_k. As before, (13) can be expressed as an SDP.

Theorem 3. *The minimax problem* (13) *is equivalent to the following SDP involving block diagonal hermitian matrices:*

$$\begin{aligned} & -\max_{x\in\mathbb{C}^n,\ \lambda\in\mathbb{R}} \lambda \\ \text{(14)}\qquad & s.t.\ \sum_{k=1}^{n}(\alpha_k A_k + \beta_k A_{n+k}) + \lambda A_{2n+1} + Z = A_0, \qquad Z \ge 0, \end{aligned}$$

where $\alpha_k = \mathrm{Re}(x_k)$, $\beta_k = \mathrm{Im}(x_k)$,

$$A_{2n+1} = \mathrm{diag}\left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}_{l=1}^{N}\right), \quad A_0 = \mathrm{diag}\left(\begin{bmatrix} 0 & d_0^{(l)} \\ \overline{d_0^{(l)}} & 0 \end{bmatrix}_{l=1}^{N}\right),$$

$$A_k = \mathrm{diag}\left(\begin{bmatrix} 0 & d_k^{(l)} \\ \overline{d_k^{(l)}} & 0 \end{bmatrix}_{l=1}^{N}\right), \quad A_{n+k} = \mathrm{diag}\left(\begin{bmatrix} 0 & i\,d_k^{(l)} \\ \overline{-i\,d_k^{(l)}} & 0 \end{bmatrix}_{l=1}^{N}\right), \tag{15}$$

$k = 1, \ldots, n$. *The matrices* A_k *consist of* N *blocks of* 2×2 *matrices on the diagonal.*

A proof of the above theorem is similar to that of Theorem 1, based on the observation that for any complex number a we have

$$|a| = \lambda_{\max}\left(\begin{bmatrix} 0 & a \\ \bar{a} & 0 \end{bmatrix}\right).$$

We omit the details.

Also, the process of transformation to a better-conditioned basis for (14) is exactly the same as for the general case. However, note that (14) cannot be obtained as a direct consequence of Theorem 1 by specializing the matrices B_k to diagonal matrices.

If the initial iterate (X^0, Z^0) is chosen to have the same block diagonal structure as the matrices A_k, then this structure is preserved throughout for (X^k, Z^k). By exploiting this block diagonal structure, the work for each iteration of the predictor-corrector algorithm is reduced to $O(n^2N)$ flops as opposed to $O(nN^3)$ for nonnormal matrices. In practice, we can compute the degree-25 Chebyshev polynomial of a normal matrix of dimension 1000 in Matlab in about 12 minutes on a Sun Ultra Sparcstation.

It would be interesting to know how this special case of our algorithm for normal matrices compares with other methods for linear complex Chebyshev approximation, such as the Remez semiinfinite programming methods discussed in [15], but we have not investigated this.

6. Computed examples.

We turn now to computed examples of Chebyshev polynomials of matrices. Our aim is to demonstrate the effectiveness of our algorithm and to give some insight into the behavior of these polynomials. This is not a subject we fully understand, but the experimental observations are fascinating.

Most of our experimental results will be presented as plots. To "plot" a polynomial p_n^*, we show its roots in the complex plane and also the boundary of a region that we call the *Chebyshev lemniscate*[1] for that polynomial and the given matrix A. This region is defined by the equation

$$\mathcal{L}_n(A) \;=\; \{z \,:\, |p_n^*(z)| \;\le\; \|p_n^*(A)\|\}.$$

The Chebyshev lemniscates characterize where in the complex plane the Chebyshev polynomials of A "live," just as the spectrum or the pseudospectra characterize (though not precisely, unless A is normal) where in the complex plane A itself "lives." As a minimum, since $\|p_n^*\|_{\Lambda(A)} \le \|p_n^*(A)\|$, we know that the Chebyshev lemniscate contains the spectrum

$$\Lambda(A) \,\subset\, \mathcal{L}_n(A). \tag{16}$$

In each example we present, the dimension of the matrix A is 48×48 or 100×100, though we typically print only its 5×5 or 6×6 analogue. For each example, we give plots showing the Chebyshev lemniscates (solid curves) of A, typically of degrees $n = 8$ and $n = 16$. The zeros of the Chebyshev polynomials are shown as small circles, and the eigenvalues of A are shown as solid dots.

For comparison with the Chebyshev lemniscate, each of our plots also shows a dotted curve. This is the boundary of an ϵ-pseudospectrum of A. The value of ϵ has been chosen "by hand" to make the match with the Chebyshev lemniscate a good one. (The ϵ-pseudospectrum of A is the set $\Lambda_\epsilon(A) = \{z\colon\ \|(zI-A)^{-1}\| \ge \epsilon^{-1}\}$ in the complex plane; see [14] and [21].)

For all of these examples, the Chebyshev polynomials were computed in MATLAB by the methods described in the previous sections.

Primal-dual predictor-corrector algorithms are highly efficient and robust for solving SDPs. For the set of examples we present here, it takes an average of 12 iterations to reduce the duality gap by a factor of 10^{10}. (This number is rather insensitive to the dimension of A; it would be essentially the same for matrices of dimensions 5×5 or 200×200. This insensitivity to problem size is one of the remarkable features of primal-dual interior point methods.) For a 48×48 real matrix, each iteration takes about 5 and 7 seconds for $n = 8$ and $n = 16$, respectively, on a Sun Ultra Sparcstation. The corresponding numbers for a 48×48 complex matrix are about 30 seconds and 45 seconds.

Here are our examples. Omitted entries are all zero.

Example 1. *Diagonal.*

$$A \;=\; \mathrm{diag}(d) \qquad (100\times100),$$

where d is a vector whose first entry is 1 and the rest of whose entries are distributed uniformly in the interval $[-1, 0.8]$. Thus the spectrum of A consists of points that densely fill the interval $[-1, 0.8]$ and an outlier at $z = 1$.

[1] Properly speaking, the word lemniscate refers to the boundary of $\mathcal{L}_n$, and $\mathcal{L}_n$ itself is a *lemniscatic region*, but this expression is cumbersome, and we shall avoid it.

Example 2. *Bidiagonal.*

$$A \;=\; \begin{pmatrix} d_1 & 0.2 & & & \\ & d_2 & 0.2 & & \\ & & \ddots & \ddots & \\ & & & \ddots & 0.2 \\ & & & & d_N \end{pmatrix} \qquad (100 \times 100),$$

where the vector d is the same as that in Example 1. The spectrum is the same as in Example 1.

Example 3. *Grcar* [21].

$$A \;=\; \begin{pmatrix} 1 & 1 & 1 & 1 & & \\ -1 & 1 & 1 & 1 & 1 & \\ & -1 & 1 & 1 & 1 & 1 \\ & & -1 & 1 & 1 & 1 \\ & & & -1 & 1 & 1 \\ & & & & -1 & 1 \end{pmatrix} \qquad (48 \times 48).$$

Example 4. *Ellipse.*

$$A \;=\; \begin{pmatrix} 0 & 3 & & & \\ 2 & 0 & 3 & & \\ & 2 & 0 & 3 & \\ & & 2 & 0 & 3 \\ & & & 2 & 0 \end{pmatrix} \qquad (48 \times 48).$$

Example 5. *Bull's head* [14].

$$A \;=\; \begin{pmatrix} 0 & 0 & 1 & .7 & & \\ 2i & 0 & 0 & 1 & .7 & \\ & 2i & 0 & 0 & 1 & .7 \\ & & 2i & 0 & 0 & 1 \\ & & & 2i & 0 & 0 \\ & & & & 2i & 0 \end{pmatrix} \qquad (48 \times 48).$$

Example 6. *Lemniscate*1 [14].

$$A \;=\; \begin{pmatrix} 1 & 1 & & & & \\ & -1 & 1 & & & \\ & & 1 & 1 & & \\ & & & -1 & 1 & \\ & & & & 1 & 1 \\ & & & & & -1 \end{pmatrix} \qquad (48 \times 48).$$

Example 7. *Lemniscate*2 [20].

$$A \;=\; \begin{pmatrix} 1 & \alpha & & & & \\ & 5 & \alpha & & & \\ & & 5 & \alpha & & \\ & & & 1 & \alpha & \\ & & & & 5 & \alpha \\ & & & & & 5 \end{pmatrix} \qquad (48 \times 48),$$

where $\alpha = (256/27)^{1/3}$.

Example 8. *Gauss–Seidel* [14].

$$A \;=\; \begin{pmatrix} 0 & \frac{1}{2} & & & & \\ 0 & \frac{1}{4} & \frac{1}{2} & & & \\ 0 & \frac{1}{8} & \frac{1}{4} & \frac{1}{2} & & \\ 0 & \frac{1}{16} & \frac{1}{8} & \frac{1}{4} & \frac{1}{2} & \\ 0 & \frac{1}{32} & \frac{1}{16} & \frac{1}{8} & \frac{1}{4} & \frac{1}{2} \\ 0 & \frac{1}{64} & \frac{1}{32} & \frac{1}{16} & \frac{1}{8} & \frac{1}{4} \end{pmatrix} \qquad (48 \times 48).$$

This is the matrix that corresponds to a Gauss–Seidel iteration applied to the standard 3-point discrete Laplacian on a grid of N points.

Example 9. *Beam–Warming* [20].

$$A \;=\; \begin{pmatrix} -1.5 & 2.0 & -1.5 & & \\ -\frac{1}{3} & -\frac{1}{2} & 1 & -\frac{1}{6} & \\ & -\frac{1}{3} & -\frac{1}{2} & 1 & -\frac{1}{6} \\ & & -\frac{1}{3} & -\frac{1}{2} & 1 \\ & & 0.7 & -2.6 & 2.1 \end{pmatrix} \qquad (48 \times 48).$$

Example 10. *Wilkinson* [21].

$$A \;=\; \begin{pmatrix} \frac{1}{N} & 1 & & & \\ & \frac{2}{N} & 1 & & \\ & & \ddots & \ddots & \\ & & & \frac{N-1}{N} & 1 \\ & & & & 1 \end{pmatrix} \qquad (48 \times 48).$$

Example 11. *Chebyshev points.*

$$A \;=\; \begin{pmatrix} x_1 & \gamma_1 & & & \\ & x_2 & \gamma_2 & & \\ & & \ddots & \ddots & \\ & & & x_{N-1} & \gamma_{N-1} \\ & & & & x_N \end{pmatrix} \qquad (48 \times 48),$$

where $\gamma_k = 0.5 - x_k$, $x_k = \cos\left(\frac{k-1}{N-1}\pi\right)$, $k = 1, \ldots, N$.

Example 12. *Random* [21].

$$A \;=\; \text{random} \qquad (48 \times 48),$$

where by random we mean that the entries of A are independently drawn from the real normal distribution with mean 0 and variance $1/N$.

Example 13. *Random triangular* [21].

$$A \;=\; \text{random triangular} \quad (48 \times 48),$$

by which we mean that A is the strictly upper triangular part of the random matrix of Example 12.

Table 1

Computed coefficients of p_8^* for the Grcar and bull's head matrices (Examples 3 and 5). All but perhaps the last two digits printed are believed to be correct.

Grcar computed p_8^*	Bull's head computed p_8^*
1	1
-7.90306320	$-0.00279600 - 0.00031731\,i$
41.3354079	$0.03375518 + 0.00745584\,i$
-150.565236	$-0.22640678 - 0.07463880\,i$
419.059092	$0.90451365 + 0.40702758\,i$
-897.405790	$-2.12812512 - 1.29015885\,i$
1464.45030	$2.67821078 + 2.33940114\,i$
-1722.68403	$-1.34847513 - 2.19132640\,i$
1271.98751	$0.05968294 + 0.74912144\,i$

Table 2

Norms $\|p_8^*(A)\|$ for Examples 1–14. All digits printed are believed to be correct, as the estimated relative accuracies are all less than 10^{-11}.

Example		Computed $\|p_8^*(A)\|$
1.	Diagonal	0.0063675408
2.	Bidiagonal	0.0551494047
3.	Grcar	1766.3135313
4.	Ellipse	7710.2711611
5.	Bull's head	1239.4186173
6.	Lemniscate1	1.0000000000
7.	Lemniscate2	834.73857463
8.	Gauss–Seidel	0.0049251285
9.	Beam–Warming	7.4348443860
10.	Wilkinson	6.2747795054
11.	Chebyshev points	46.395131600
12.	Random	2.9537221027
13.	Random triangular	0.0039633789
14.	Convection-diffusion	2623904.6097

Example 14. *Convection-diffusion matrix* [13], [18]. The matrix A is the projection of the $2N \times 2N$ Chebyshev spectral discretization matrix of a convection-diffusion operator onto the invariant subspace associated with the N eigenvalues of maximal real part ($N = 48$).

In Table 1, for later authors who may wish to compare the coefficients of some Chebyshev polynomials of matrices, we list the coefficients of p_8^* for the matrices of Examples 3 and 5. In Table 2, we list $\|p_8^*(A)\|$ for all 14 examples.

The plots for our 14 examples are shown in Figures 1–14.

Let us first consider Example 1, the special case where A is diagonal. For any Chebyshev polynomial of a matrix, we know that the Chebyshev lemniscate must contain the spectrum (16). In the present case, by the characterization theorems for the classical complex Chebyshev approximation problem [3, p. 143], we know that the nth Chebyshev lemniscate must in fact touch the spectrum $\Lambda(A)$ at no fewer than $n + 1$ points. This property is evident in Figure 1, where we see that $\mathcal{L}_n(A)$ hugs $\Lambda(A)$ rather closely, and increasingly so as n increases (see the cover illustration of [22]). It is interesting also to note how quickly one of the roots of the polynomials

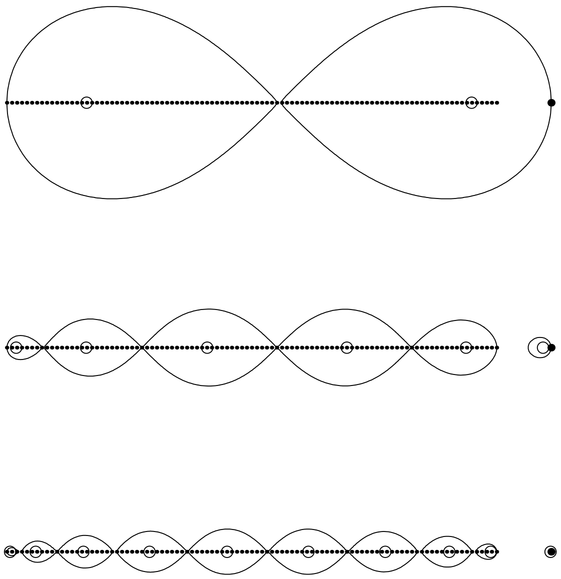


FIG. 1. *Diagonal. Since A is normal, the Chebyshev lemniscate touches the spectrum at at least $n+1$ points, and the roots of p_n^* lie in the convex hull of the spectrum.*

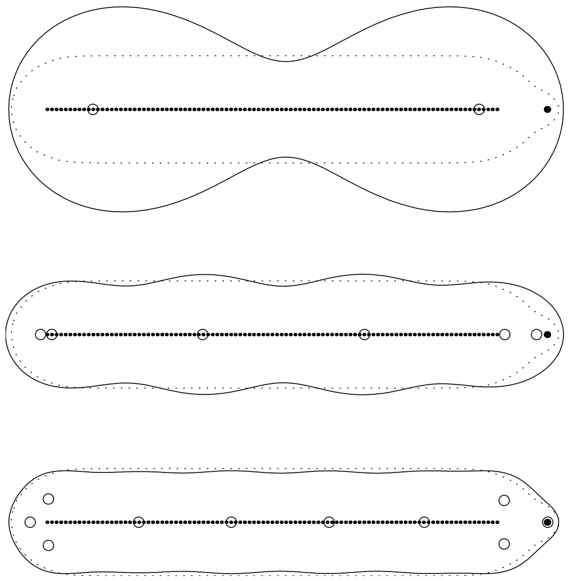


FIG. 2. *Bidiagonal—a nonnormal analogue of Example* 1. *The dotted curves are the $\epsilon = 10^{-1.5}$ pseudospectrum of A.*

p_n^*, which are analogous to the "Ritz values" often taken as eigenvalue estimates in Arnoldi or Lanczos iterations, converges to the outlier eigenvalue at $z = 1$. By $n = 6$, one of the roots of p_6^* is already very close to the outlier, and the distance between them decreases geometrically as n increases. In the remainder of the spectrum, on the other hand, no individual Ritz value is converging rapidly to any one eigenvalue of A. Rather, it is the Chebyshev lemniscate generated by these Ritz values jointly that is capturing the spectrum.

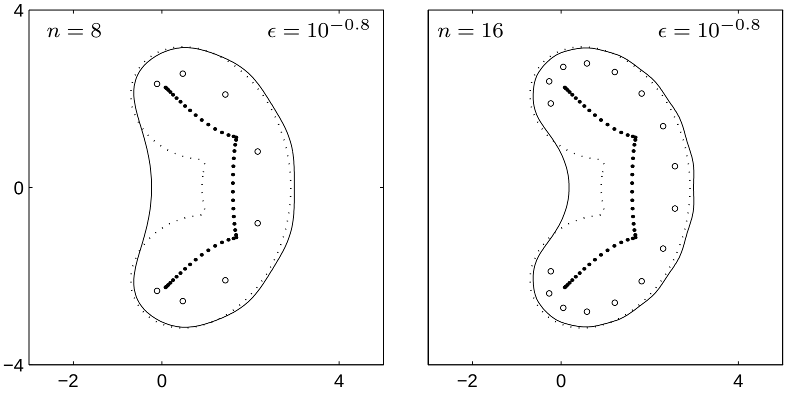


FIG. 3. *Grcar—a nonnormal Toeplitz matrix.*

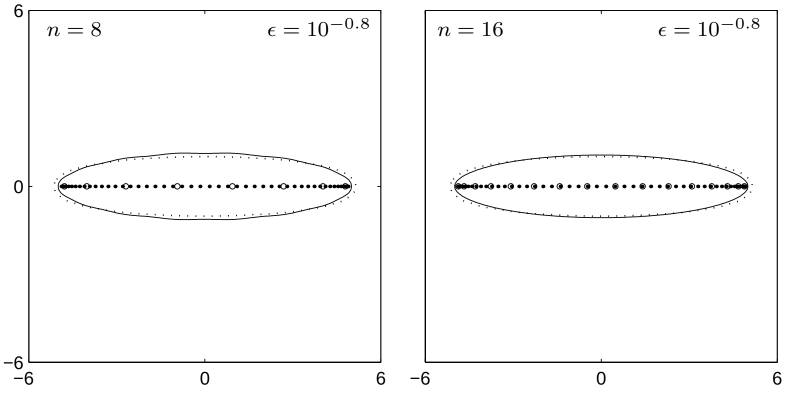


FIG. 4. *Ellipse. The eigenvalues are all real, but the pseudospectra bulge into the complex plane.*

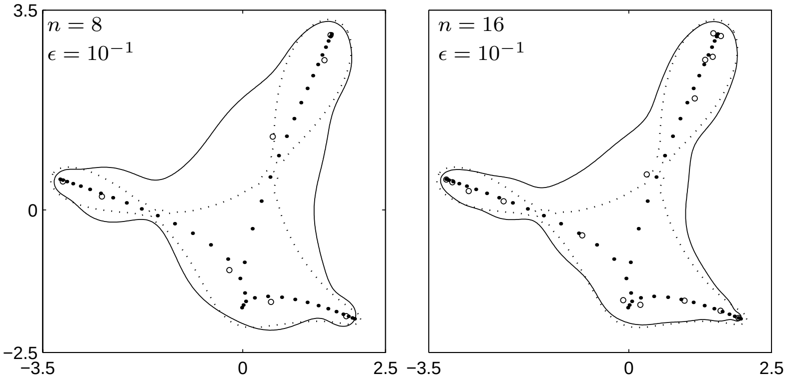


FIG. 5. *Bull's head—our only example with complex entries, hence a figure with no symmetries.*

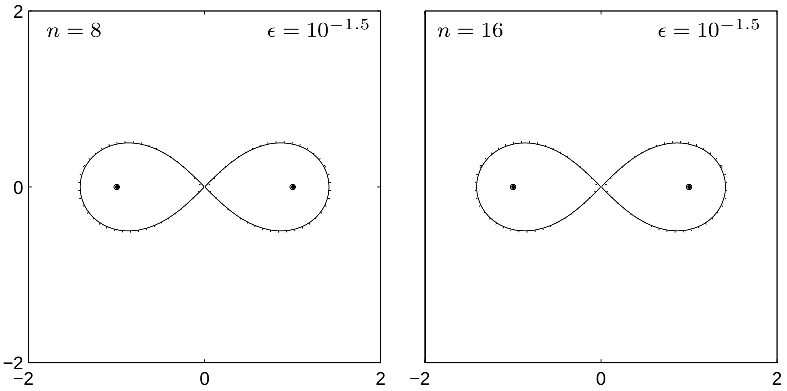


FIG. 6. *Lemniscate1. The zeros of p_n^* are $\{-1, 1\}$. The dotted and solid curves coincide almost exactly.*

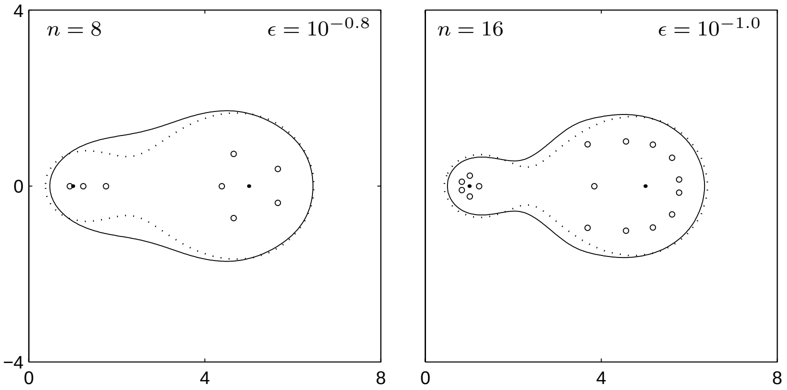


FIG. 7. *Lemniscate2. Here one eigenvalue of the matrix has twice the multiplicity of the other.*

One might expect $\mathcal{L}_n(A)$ to approximate $\Lambda(A)$ even if A is nonnormal. But from Figures 2–14, the reader will see that this does not happen. Nonetheless, though $\mathcal{L}_n(A)$ does not always approximate $\Lambda(A)$ very closely, it still gains some information about A. The plots show that for these examples, to a rather startling degree,

$$\mathcal{L}_n(A) \approx \Lambda_\epsilon(A) \tag{17}$$

for some $\epsilon \geq 0$, where $\Lambda_\epsilon(A)$ is again the ϵ-pseudospectrum of A. In particular, the agreement of the Chebyshev lemniscate of p_n^* with a pseudospectrum of A is far closer in most of these examples than the agreement of the roots of p_n^* with the eigenvalues of A. For example, consider Figure 2, the bidiagonal matrix that is the nonnormal analogue of Example 1 with the same spectrum.

Except for the outlier eigenvalue, the roots of p_n^* bear no resemblance to individual eigenvalues of A. On the other hand, the Chebyshev lemniscates of these polynomials show a striking resemblance to the $\epsilon = 10^{-1.5}$ pseudospectrum of A. Clearly the Chebyshev polynomial is approximating A in a fashion that goes beyond approximation of individual eigenvalues.

The other examples illustrate the same effect. In every case, the lemniscate of

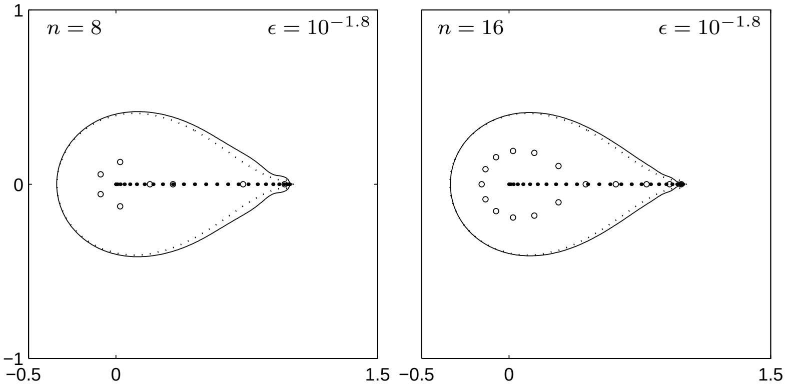


FIG. 8. *Gauss–Seidel. Half of the eigenvalues are at the origin.*

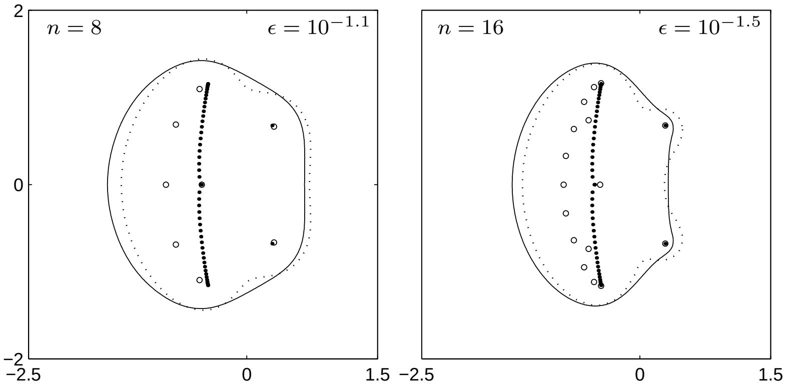


FIG. 9. *Beam–Warming—a "quasi-Toeplitz" matrix.*

the Chebyshev polynomial shows a compelling approximation to the pseudospectrum. We do not claim that this effect is universal (these examples have been picked for their pronounced and cleanly structured nonnormality), but it is certainly common.

A partial explanation of this phenomenon is as follows. It is well known that a matrix polynomial $p(A)$ can be expressed as a Cauchy integral

$$p(A) \ = \ \frac{1}{2\pi i} \int p(z) \, (zI - A)^{-1} dz,$$

where the integration is over any closed contour or union of contours enclosing the spectrum of A once in the counterclockwise direction [8]. Taking absolute values gives the inequality

$$\|p(A)\| \ \le \ \frac{1}{2\pi} \int |p(z)| \, \|(zI - A)^{-1}\| \, |dz|. \tag{18}$$

Now suppose we seek p such that $\|p(A)\|$ is small. When the degree of p is smaller than the dimension of A, it is impossible to achieve this in general by putting zeros of p wherever A has eigenvalues, which would make the integral zero. Instead, we must settle for making $|p(z)|$ small where $\|(zI - A)^{-1}\|$ is large. This immediately suggests a link between lemniscates of p and pseudospectra of A.

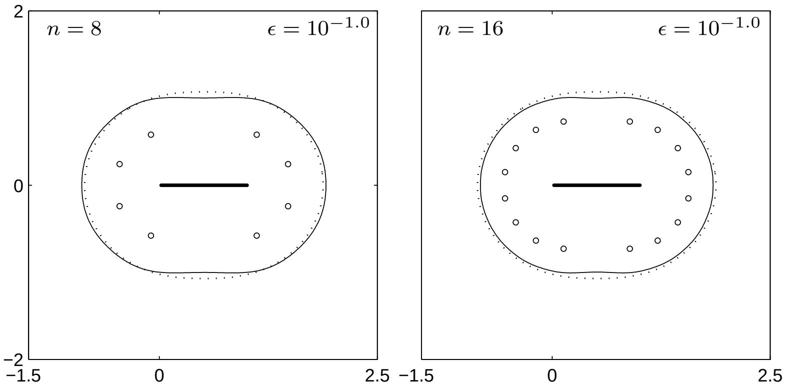


FIG. 10. *Wilkinson—evenly spaced eigenvalues on the real axis.*

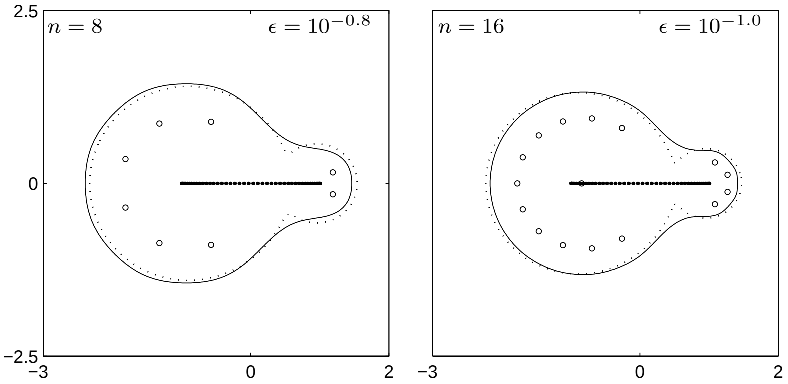


FIG. 11. *Chebyshev points—unevenly spaced eigenvalues.*

From this kind of reasoning we can derive bounds on $\|p_n^*(A)\|$. For example, to minimize $\|p(A)\|$ one might seek to minimize $\|p\|_{\Lambda_\epsilon(A)}$ for some ϵ that is not too small (hence $|p(z)|$ is small over the region where $\|(zI - A)^{-1}\|$ is larger than ϵ^{-1}). From (18) and the minimality of $\|p_n^*(A)\|$ we conclude that

$$\|p_n^*(A)\| \;\le\; \frac{L_\epsilon}{2\pi\epsilon}\,\min_p \|p\|_{\Lambda_\epsilon(A)}, \tag{19}$$

where L_ϵ is the arclength of the boundary of $\Lambda_\epsilon(A)$. At this point one runs into the fact that $\min_p \|p\|_{\Lambda_\epsilon(A)}$ can be huge if ϵ is not small, since the minimum typically increases geometrically with ϵ. Therefore, a compromise must be made on ϵ so that the quantity $\min_p \|p\|_{\Lambda_\epsilon(A)}/\epsilon$ on the right-hand side of (19) is as small as possible.

For some matrices A and choices of n and ϵ, the estimate just described can be quite good. It is not always very good, however, and so far our attempts to make a more precise link between lemniscates of Chebyshev polynomials and pseudospectra of the underlying matrix have been unsuccessful except in certain limiting cases $n \to \infty$ described in [17]. Rather than present partial results that do not bring this matter to a very satisfactory state, we prefer to leave the explanation of the behavior of Figures 2–14 as an open problem.

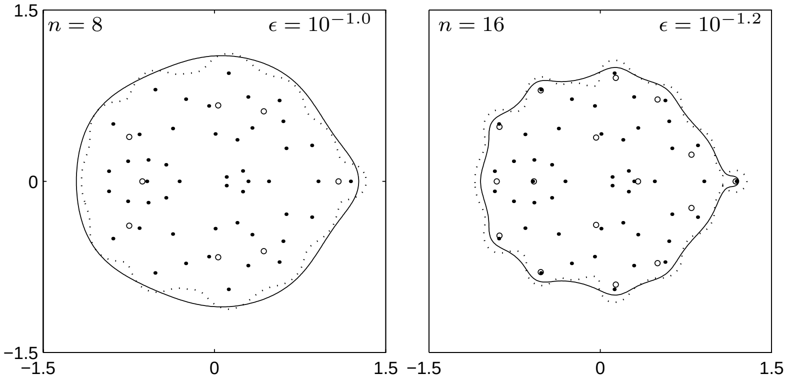


FIG. 12. *Random. This matrix is only mildly nonnormal.*

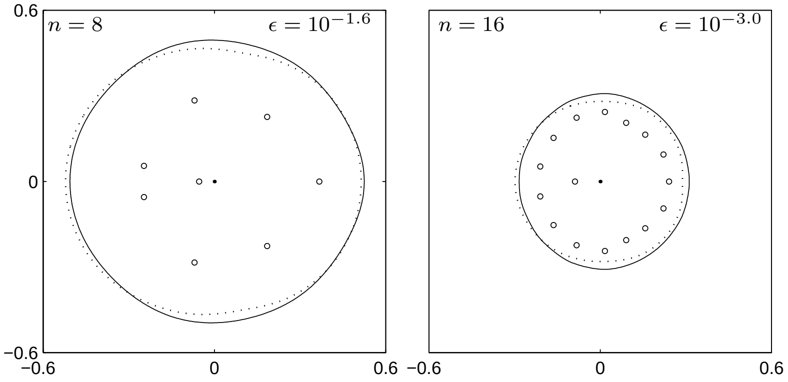


FIG. 13. *Random triangular. The nonnormality is now pronounced.*

7. Conclusions. This paper has made two contributions. The first is a reasonably fast and apparently robust algorithm for computing the Chebyshev polynomials of a matrix, based on primal-dual interior point methods in semidefinite programming. The second is an experimental study of these polynomials that indicates that the associated lemniscates sometimes closely approximate certain pseudospectra of A.

We have said little about applications in iterative numerical linear algebra, though that was our original motivation. There are many possibilities here that might be explored now that an algorithm is available. For example, our algorithm may prove useful in analyzing the convergence of Krylov subspace iterations, or the construction of preconditioners for such iterations, by means of model problems of moderate dimension.

It was mentioned in the introduction that for applications to iterative solution of equations rather than eigenvalue calculations it is appropriate to minimize $\|p(A)\|$ with the normalization $p(0) = 1$ instead of $\lim_{z\to\infty} p(z)/z^n = 1$. Plots of lemniscates for these "ideal GMRES polynomials" can be found in the first author's dissertation [17]. Because this normalization gives a special status to the origin, these problems are no longer translation-invariant in the complex plane, and the lemniscates take special pains to avoid the origin. They also tend to display scallop patterns near

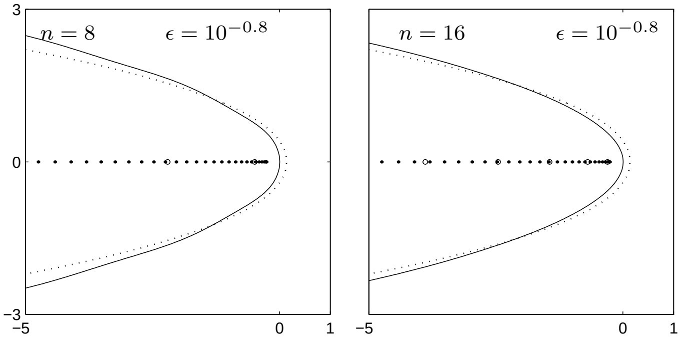


FIG. 14. *Convection-diffusion. A matrix approximation to the canonical nonnormal differential operator.*

the spectrum or pseudospectra.

Interesting connections can also be made to the notion of a *generalized Kreiss matrix theorem.* The usual Kreiss matrix theorem relates the norms $\|A^n\|$ to the behavior of the pseudospectra of A near the unit disk. Generalizations are obtained by looking at norms $\|p(A)\|$ for other polynomials p and the behavior of the pseudospectra near other regions. These matters are investigated in [19].

We consider the idea of the Chebyshev polynomials of a matrix a natural one, suggesting many questions to be explored. We hope that more will be learned about the behavior of these polynomials in the years ahead and that applications in other areas will be found.

REFERENCES

[1] F. Alizadeh, J.-P. A. Haeberly, and M. L. Overton, *Primal-dual interior-point methods for semidefinite programming: Convergence rates, stability, and numerical results*, SIAM J. Optim., 8 (1998), pp. 746–768.

[2] J. H. Curtiss, *Faber polynomials and the Faber series*, Amer. Math. Monthly, 78 (1971), pp. 577–596.

[3] P. J. Davis, *Interpolation and Approximation*, Dover, New York, 1975.

[4] T. A. Driscoll, *A MATLAB toolbox for Schwarz-Christoffel mapping*, ACM Trans. Math. Software, 22 (1996), pp. 168–186.

[5] G. Faber, *Über Tschebyscheffsche Polynome*, J. Reine Angew. Math., 150 (1920), pp. 79–106.

[6] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 2nd ed., Johns Hopkins University Press, Baltimore, MD, 1989.

[7] A. Greenbaum and L. N. Trefethen, *GMRES/CR and Arnoldi/Lanczos as matrix approximation problems*, SIAM J. Sci. Comput., 15 (1994), pp. 359–368.

[8] T. Kato, *Perturbation Theory for Linear Operators*, Springer-Verlag, Berlin, 1966.

[9] C.-J. Lin and R. Saigal, *An Infeasible Start Predictor-Corrector Method for Semi-Definite Linear Programming*, manuscript, Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI, 1995.

[10] R. D. C. Monteiro, *Primal–dual path-following algorithms for semidefinite programming*, SIAM J. Optim., 7 (1997), pp. 663–678.

[11] Y. Nesterov and A. Nemirovskii, *Interior Point Polynomial Methods in Convex Programming*, Stud. Appl. Math. 13, SIAM, Philadelphia, PA, 1994.

[12] Y. E. Nesterov and M. J. Todd, *Primal-dual interior-point methods for self-scaled cones*, SIAM J. Optim., 8 (1998), pp. 324–364.

[13] S. C. Reddy and L. N. Trefethen, *Pseudospectra of the convection-diffusion operator*, SIAM J. Appl. Math., 54 (1994), pp. 1634–1649.

[14] L. Reichel and L. N. Trefethen, *Eigenvalues and pseudo-eigenvalues of Toeplitz matrices*, Linear Algebra Appl., 169 (1992), pp. 153–185.

[15] P. T. P. Tang, *A fast algorithm for linear complex Chebyshev approximations*, Math. Comp., 51 (1988), pp. 721–739.

[16] M. J. Todd, K.-C. Toh, and R. H. Tütüncü, *On the Nesterov–Todd direction in semidefinite programming*, SIAM J. Optim., 8 (1998), pp. 769-796.

[17] K.-C. Toh, *Matrix Approximation Problems and Nonsymmetric Iterative Methods*, Ph.D. thesis, Cornell University, Ithaca, NY, 1996.

[18] K.-C. Toh and L. N. Trefethen, *Calculation of pseudospectra by the Arnoldi iteration*, SIAM J. Sci. Comput., 17 (1996), pp. 1–15.

[19] K.-C. Toh and L. N. Trefethen, *The Kreiss matrix theorem on a general complex domain*, SIAM J. Matrix Anal. Appl., to appear.

[20] L. N. Trefethen, *Spectra and Pseudospectra: The Behavior of Non-Normal Matrices and Operators*, in preparation.

[21] L. N. Trefethen, *Pseudospectra of matrices*, in Numerical Analysis 1991, D. F. Griffiths and G. A. Watson, eds., Longman Scientific and Technical, Harlow, UK, 1992.

[22] L. N. Trefethen and D. Bau, III, *Numerical Linear Algebra*, SIAM, Philadelphia, PA, 1997.

[23] L. Vandenberghe and S. Boyd, *Semidefinite programming*, SIAM Rev., 38 (1996), pp. 49–95.

[24] H. Widom, *Extremal polynomials associated with a system of curves in the complex plane*, Adv. Math., 3 (1969), pp. 127–232.

[25] Y. Zhang, *On extending some primal-dual interior-point algorithms from linear programming to semidefinite programming*, SIAM J. Optim., 8 (1998), pp. 365–386.

SIAM J. MATRIX ANAL. APPL.
Vol. 20, No. 2, pp. 420–427

THE ASYMPTOTIC BEHAVIOR OF THE EIGENVALUES OF A SINGULARLY PERTURBED LINEAR PENCIL[*]

BRANKO NAJMAN[†]

Abstract. The pencil $\widetilde{T}(\lambda,\varepsilon)=\varepsilon\lambda^2 I+\lambda C+K$, a singular perturbation of the linear pencil $\widetilde{T}(\lambda,0)=\lambda C+K$, is considered, and the asymptotic behavior of its large eigenvalues (i.e., those converging to infinity as ε goes to zero) is analyzed. The Newton diagram method is employed to determine the leading power of the Puiseux ε-expansion of the large eigenvalues of $\widetilde{T}(\lambda,\varepsilon)$. These leading powers are completely identified either in the case when C is diagonalizable and nonsingular, or when both C and K are hermitian matrices. In the remaining cases, some partial information is still obtained.

Key words. matrix pencil, eigenvalue, singular perturbation, Puiseux expansion

AMS subject classifications. 15A18, 15A22, 47A56, 47A55

PII. S0895479896299949

1. Introduction.

The investigation of the eigenvalues of analytically perturbed matrix polynomials is a part of the analytic perturbation theory [1], [4]. In recent years it has attracted considerable attention and new approaches have been applied, proving new results and deepening our understanding of some special cases (see [2], [5], [6], [7], [8]). In this paper, the pencil

$$\widetilde{T}(\lambda,\varepsilon)=\varepsilon\lambda^2 I+\lambda C+K$$

is studied. It can be considered as a second order (hence singular) perturbation of the linear pencil $\widetilde{T}(\lambda,0)=\lambda C+K$. Throughout the paper, it is assumed that K and C are $n\times n$ matrices and that the pencil $\mu K+C$ is *regular*, i.e., the determinant $q(\mu)=\det(\mu K+C)$ is not identically equal to zero. It is evident that the perturbed pencil has more finite eigenvalues than the unperturbed; thus some of the eigenvalues $\lambda(\varepsilon)$ of $\widetilde{T}(\lambda,\varepsilon)$ "converge to ∞" as $\varepsilon\to 0$. We call these eigenvalues "large," and we are interested in their asymptotic behavior. This behavior is determined by the first nonvanishing term of the *Puiseux expansion*

$$\lambda(\varepsilon)=\sum_{j=1}^{\infty}\gamma_j\left(\frac{1}{\varepsilon^p}\right)^j, \tag{1.1}$$

which holds in a deleted neighborhood of $\varepsilon=0$ with some rational number p. In the following, the first nonvanishing term of this Puiseux expansion is called the *first term*, and the exponent $-jp$ of the first term is called the *leading exponent* of the expansion (1.1).

[*] Received by the editors March 1, 1996; accepted for publication (in revised form) by M. Overton May 7, 1998; published electronically October 29, 1998.
http://www.siam.org/journals/simax/20-2/29994.html

[†] The author is deceased. Former address: Department of Mathematics, University of Zagreb, Bijenicka 30, 41001 Zagreb, Croatia. Final work on this paper was done by Heinz Langer (hlanger@mail.zserv.tuwien.ac.at) and Julio Moro (jmoro@math.uc3m.es).

We employ the Newton diagram method, which has previously been applied in [5], [6], [7], [8]. We show, in particular, that

1) the total multiplicity of the “large” eigenvalues is $n + a$, where n is the size of the matrices C and K and a is the algebraic multiplicity of zero as the root of the polynomial q;
2) if C is diagonalizable and nonsingular, then all the n “large” eigenvalues $\lambda(\varepsilon)$ have Puiseux expansions with the first term $\frac{\gamma}{\varepsilon}, \gamma \neq 0$, i.e.,
$$\frac{1}{\lambda(\varepsilon)} = \frac{1}{\gamma}\varepsilon + o(|\varepsilon|) \quad (\varepsilon \to 0);$$
3) if C is diagonalizable and singular with a zero eigenvalue of multiplicity b, then exactly $n-b$ “large” eigenvalues $\lambda(\varepsilon)$ have Puiseux expansions with the first term $\frac{\gamma}{\varepsilon}, \gamma \neq 0$, and exactly $a + b$ “large” eigenvalues $\lambda(\varepsilon)$ have Puiseux expansions with the first term $(\gamma/\varepsilon^p), \gamma \neq 0$, for some $0 < p \leq \frac{1}{2}$;
4) if C is diagonalizable and $a = b$, then, in addition to the $n - a$ “large” eigenvalues, which have Puiseux expansions with the first term $\frac{\gamma}{\varepsilon}, \gamma \neq 0$, there are $2a$ “large” eigenvalues which have Puiseux expansions starting with the first term $\frac{\gamma}{\sqrt{\varepsilon}}, \gamma \neq 0$;
5) if C and K are hermitian and one of them is uniformly definite, then $a = b$ and case 4 applies;
6) if C and K are hermitian and indefinite, then, in addition to the $n-b$ “large” eigenvalues $\lambda(\varepsilon)$, which have Puiseux expansions with the first term γ/ε, $\gamma \neq 0$, the remaining $a + b$ “large” eigenvalues have Puiseux expansions completely determined by the canonical form of the pair (C, K); see Theorem 3.2 for the precise statement.

2. The results.

Note that

$$\widetilde{T}(\lambda, \varepsilon) = \lambda^2 T\left(\frac{1}{\lambda}, \varepsilon\right), \tag{2.1}$$

where

$$T(\mu, \varepsilon) = \mu^2 K + \mu C + \varepsilon I$$

can be considered as a perturbation of $T(\mu, 0) = \mu^2 K + \mu C$.

From (2.1) it follows that for every eigenvalue $\lambda \neq 0$ of $\widetilde{T}(\,\cdot\,, \varepsilon)$, the number $\mu_0 = (1/\lambda_0)$ is an eigenvalue of $T(\,\cdot\,, \varepsilon)$ and conversely. The analysis of the eigenvalues near ∞ of $\widetilde{T}(\,\cdot\,, \varepsilon)$ reduces to the analysis of the eigenvalues near zero of $T(\,\cdot\,, \varepsilon)$ and vice versa.

Let

$$\mathbf{t}(\mu, \varepsilon) = \det T(\mu, \varepsilon) = \sum_{i=0}^{i=2n} \sum_{k=0}^{k=n} \frac{t_{ik}}{i!k!} \mu^i \varepsilon^k,$$

where

$$t_{ik} = \frac{\partial^{i+k}\mathbf{t}}{\partial\mu^i \partial\varepsilon^k}(0, 0).$$

Then the eigenvalues μ of $T(\,\cdot\,, \varepsilon)$ admit expansions

$$\mu(\varepsilon) = \sum_{j=1}^{\infty} \delta_j \varepsilon^{pj}, \tag{2.2}$$

and we are interested in the first nonvanishing term of this expansion; the corresponding exponent is again called the *leading exponent* of the expansion (2.2). In order to determine it, we use the Newton diagram method, which was used also in [5], [6], [7], [8]. Recall that this method consists of the following (see [1, Appendix A.7] and [10, section I.2]). For $i, k \geq 0$, plot all the points (i, k) with $t_{ik} \neq 0$ on a Cartesian grid and take the lower boundary of the convex hull of all the plotted points. This lower boundary is called the *Newton diagram* $\mathrm{ND}(\mathbf{t})$ of $\mathbf{t}$. Then consider the falling part of the Newton diagram. It is a polygonal line, and the slopes of the different segments of this polygonal line, multiplied by -1, are the leading exponents of the expansions (2.2). Moreover, the length of the horizontal projection of each segment is the number of eigenvalues with that particular leading exponent. In the following, by $\mathrm{E}(\mathbf{t})$ we denote the extremal points of $\mathrm{ND}(\mathbf{t})$; these are simply the points where the slopes of the segments of $\mathrm{ND}(\mathbf{t})$ change.

Set $a = 0$ if $\det C \neq 0$, and let a be the multiplicity of zero as the root of the polynomial $q(\mu) = \det(\mu K + C)$ if $\det C = 0$.

PROPOSITION 2.1. *The points $P_i = (0, n)$ and $P_e = (n + a, 0)$ are extremal points of the Newton diagram $\mathrm{ND}(\mathbf{t})$ of $\mathbf{t}$.*

Proof. Evidently,

$$t_{0k} = \frac{d^k}{d\varepsilon^k} \det(\varepsilon I)|_{\varepsilon=0} = \frac{d^k}{d\varepsilon^k} \varepsilon^n = n!\delta_{nk},$$

$$t_{i0} = \frac{d^i}{d\mu^i} \det T(\mu, 0)|_{\mu=0} = \frac{d^i}{d\mu^i}(\mu^n \det(\mu K + C))|_{\mu=0} = \frac{d^i}{d\mu^i}(\mu^n q(\mu))|_{\mu=0};$$

hence

$$t_{i0} = \begin{cases} 0 & \text{if} \quad i < n, \\ n! \det C & \text{if} \quad i = n, \\ \dfrac{i!}{(i-n)!} \dfrac{d^{i-n} q}{d\mu^i}(0) & \text{if} \quad i > n. \end{cases}$$

It follows that $t_{i0} = 0$ if $i < n + a, t_{n+a,0} \neq 0$, and, consequently, $P_i, P_e \in \mathrm{E}(\mathbf{t})$. □

The coefficients t_{ik} can be considered as sums

$$\sum \det[f_1, \ldots, f_n] \tag{2.3}$$

of determinants, where each column f_j is either the jth column of $T(\mu, \varepsilon)$ or it is obtained by differentiating the jth column of $T(\mu, \varepsilon)$ with respect to μ or ε. Therefore, if $i + k < n$, then at least one column in each determinant is a column of $T(\mu, \varepsilon)$, and therefore is zero at $\mu = \varepsilon = 0$. Hence no point of (i, k) with $i + k < n$ may appear on the Newton diagram $\mathrm{ND}(\mathbf{t})$.

Recall that a "large" eigenvalue $\lambda(\varepsilon)$ of $\widetilde{T}(\lambda, \varepsilon)$ is an eigenvalue with the property $\lim_{\varepsilon \to 0} |\lambda(\varepsilon)| = \infty$. The next theorem is an easy consequence of Proposition 2.1 and the above consideration.

THEOREM 2.2. *The pencil $\widetilde{T}$ has $n + a$ "large" eigenvalues. At least a of them have Puiseux expansions* (1.1) *with the first term $\gamma\varepsilon^{-p}$ for some $\gamma \neq 0$, $p \in (0, 1)$. If $a = 0$, then all "large" eigenvalues have Puiseux expansions with the first term $\frac{\gamma}{\varepsilon}$ for some $\gamma \neq 0$; if $a \geq 1$, then there are at most $n - 1$ eigenvalues with the first term $\frac{\gamma}{\varepsilon}$, and at least $a + 1$ "large" eigenvalues have Puiseux expansions with the first term (γ/ε^p) for some $\gamma \neq 0$, $p \in (0, 1)$.*

Proof. According to Proposition 2.1 the points $P_i = (0, n)$ and $P_e = (n+a, 0)$ are always on $\mathrm{ND}(\mathbf{t})$, while no point $(0, k)$ with $k < n$ and no point $(i, 0)$ with $i < n + a$ are on $\mathrm{ND}(\mathbf{t})$. Since no point (i, k) with $i + k < n$ belongs to $\mathrm{ND}(\mathbf{t})$, the negative slope of the steepest segment (that is, of the first segment) is at most one. If $a = 0$, then $\mathrm{ND}(\mathbf{t})$ reduces to the line of slope -1 connecting P_i with $(n, 0)$. If $a \geq 1$, the first segment with $p = 1$ can have a horizontal projection of length at most $n-1$, since otherwise $\mathrm{ND}(\mathbf{t})$ would hit the horizontal axis at $(n, 0)$, which is impossible. $\square$

Denote by b the defect of C, that is, the dimension of the kernel of C. Evidently, $a = 0$ if and only if $b = 0$. In the following we assume that C is diagonalizable. Then b coincides with the multiplicity of zero as a root of the polynomial $p(\mu) = \det(C - \mu I)$.

PROPOSITION 2.3. *Assume that C is diagonalizable. Then the following statements hold:*

(a) *If $b \geq 1$, then $a \geq b$.*

(b) *The point $P_m = (n - b, b)$ is an extremal point of* $\mathrm{ND}(\mathbf{t})$.

(c) *If the set* $\mathrm{E}(\mathbf{t})$ *contains other points besides P_i, P_m, and P_e, then the part of the* $\mathrm{ND}(\mathbf{t})$ *between P_m and P_e lies to the right of the line $x + 2y = n + b$; there are no points of* $\mathrm{E}(\mathbf{t})$ *between P_i and P_m.*

Proof. (a) Let S be a nonsingular and D a diagonal matrix such that $C = S^{-1}DS$, and set $H = SKS^{-1}$. Then $q(\mu) = \det(\mu H + D)$, and b is the number of zero diagonal entries of D. Without loss of generality, we can assume $D = \mathrm{diag}(0, \ldots, 0, d_{b+1}, \ldots, d_n)$ with $d_j \neq 0$ for $j = b+1, \ldots, n$. Then

$$q(\mu) = \det \begin{bmatrix} \mu H_1 & \mu H_2 \\ \mu H_3 & D_4 + \mu H_4 \end{bmatrix} = \mu^b \det \begin{bmatrix} H_1 & \mu H_2 \\ H_3 & D_4 + \mu H_4 \end{bmatrix},$$

where H_1 is a $b \times b$ matrix, and H_4 and $D_4 = \mathrm{diag}\,(d_{b+1}, \ldots, d_n)$ are $(n-b) \times (n-b)$ matrices. It follows that

$$\frac{d^j q}{d\mu^j}(0) = 0 \quad \text{if} \quad j < b,$$

$$\frac{d^b q}{d\mu^b}(0) = b! \det \begin{bmatrix} H_1 & 0 \\ H_3 & D_4 \end{bmatrix} = b!\, d_{b+1} \cdots d_n \det H_1;$$

hence $a \geq b$.

(b), (c). Without loss of generality, we can assume $b > 0$ and

$$T(\mu, \varepsilon) = \begin{bmatrix} \mu^2 H_1 + \varepsilon I & \mu^2 H_2 \\ \mu^2 H_3 & \mu^2 H_4 + \mu D_4 + \varepsilon I \end{bmatrix}.$$

First we show that $P_m = (n-b, b)$ belongs to $\mathrm{ND}(\mathbf{t})$. To this end we write $t_{n-b,b}$ as a sum of determinants (2.3). The columns $f_1, \ldots, f_n$ are obtained from the corresponding columns of $T(\mu, \varepsilon)$ by differentiating $(n-b)$-times with respect to μ, b-times with respect to ε, and setting $\mu = \varepsilon = 0$. It is easy to see that the only nonzero term in this sum arises if we differentiate the last $n - b$ columns with respect to μ and the first b columns with respect to ε; hence $t_{n-b,b} = d_{b+1} \cdots d_n \neq 0$.

Now let $0 < k \leq b$ and consider the coefficients (2.3) in the expansion of $\mathbf{t}$. The column f_j is obtained by differentiating the jth column of $T(\lambda, \varepsilon)$. If $j \leq b$ and

$f_j \neq 0$, then f_j is obtained by differentiating with respect to ε once or differentiating with respect to μ twice. If $j > b$ and $f_j \neq 0$, then f_j is obtained by differentiating with respect to ε once or differentiating with respect to μ once or twice. Assume $f_1, \dots, f_m$ are all nonzero and $l \leq k$ of the $f_1, \dots, f_b$ are obtained by differentiating once with respect to ε. Then $b-l$ of these columns are obtained by differentiating twice with respect to μ. The remaining columns $f_{b+1}, \dots, f_n$ are obtained by differentiating $(k-l)$-times with respect to ε and $(i-2(b-l))$-times with respect to μ. This implies $i - 2(b-l) \geq n - b - (k-l)$, i.e., $i \geq n + b - k - l \geq n + b - 2k$. Hence $t_{ik} = 0$ if $i < n + b - 2k$, implying (b) and (c). □

By the same reasoning as in the second part of the proof, more information can be obtained about the coefficients t_{ik}. For example, the representation (2.3) of $t_{1,n-1}$ contains determinants which are zero because the first b columns have not been differentiated with respect to μ; hence they become zero when we set $\mu = 0$; any nonzero contribution must originate from each of the first b columns being differentiated once with respect to ε and not differentiated with respect to μ. It follows that

$$\begin{aligned} t_{1,n-1} &= b!\,\frac{\partial}{\partial\mu}\frac{\partial^{n-b-1}}{\partial\varepsilon^{n-b-1}}\,\det(\mu^2 H_4 + \mu D_4 + \varepsilon I)|_{\varepsilon=0,\mu=0} \\ &= b!\,(n-b-1)!\,(d_{b+1} + \cdots + d_n); \end{aligned} \tag{2.4}$$

hence $t_{1,n-1} = 0$ if $\operatorname{tr} D_4 = 0$. Similarly, $t_{1,n-2} = 0$ since all the determinants in the expansion contain a zero column and $t_{2,n-2} = b!(n-b-2)!\sigma_2$, where σ_2 is the second symmetric function of the $d_{b+1}, \dots, d_n$. Analogously, $t_{ij} = 0$ if $i + j < n$ and $t_{i,n-i} = \gamma_i \sigma_i$, where $\gamma_i \neq 0$ depends only on n and b, and σ_i is the ith elementary symmetric function of $d_{b+1}, \dots, d_n$.

Corollary 2.4. *If $a = b$, then* $\mathrm{E}(\mathbf{t}) = \{P_i, P_m, P_e\}$.

From Proposition 2.3 and Corollary 2.4 we obtain the following theorem.

Theorem 2.5. *Assume that C is diagonalizable.*

a) *If C is nonsingular, then all the n "large" eigenvalues have Puiseux expansions with the first term $\frac{\gamma}{\varepsilon}, \gamma \neq 0$.*

b) *If C is singular, then $n-b$ "large" eigenvalues have Puiseux expansions with the first term $\frac{\gamma}{\varepsilon}, \gamma \neq 0$, and $a+b$ "large" eigenvalues have Puiseux expansions with the first term $(\gamma/\varepsilon^p), \gamma \neq 0$, $p \leq \frac{1}{2}$.*

c) *If $a = b$, then $n - a$ "large" eigenvalues have Puiseux expansions with the first term $\frac{\gamma}{\varepsilon}, \gamma \neq 0$, and $2a$ "large" eigenvalues have Puiseux expansions with the first term $\frac{\gamma}{\sqrt{\varepsilon}}$, $\gamma \neq 0$.*

3. The hermitian case. In this section we consider the case of hermitian matrices C and K. Then Theorem 2.5 applies; however, more specific results can be obtained using the canonical form of the pencil $\mu K + C$ (see [9], [3]). Denote

$$J(1,\sigma) = \sigma, \qquad H(h,1,\sigma) = \sigma h,$$

and, if $m > 1$,

$$J(m,\sigma) = \sigma \begin{bmatrix} & & & 1 \\ & & \cdot & \\ & \cdot & & \\ \cdot & & & \\ 1 & & & \end{bmatrix}, \qquad H(h,m,\sigma) = \sigma \begin{bmatrix} & & & & h \\ & & & \cdot & 1 \\ & & \cdot & \cdot & \\ & \cdot & \cdot & & \\ h & 1 & & & \end{bmatrix},$$

where $h \in \mathbf{C}, \sigma \in \{-1, 1\}$. Then there exists a nonsingular matrix X such that

$$K = X^* K_1 X, \qquad C = X^* C_1 X,$$

$$K_1 = \left(\bigoplus_{j=1}^{b} J(m_j, \sigma_j) \right) \oplus \left(\bigoplus_{k=1}^{q} J(\widehat{m}_k, \sigma_k) \right) \oplus \left(\bigoplus_{l=-1}^{-r} H(0, m_l, \sigma_l) \right),$$
$$C_1 = \left(\bigoplus_{j=1}^{b} H(0, m_j, \sigma_j) \right) \oplus \left(\bigoplus_{k=1}^{q} H(h_k, \widehat{m}_k, \sigma_k) \right) \oplus \left(\bigoplus_{l=-1}^{-r} J(m_l, \sigma_l) \right)$$

with $\sigma_j, \sigma_k, \sigma_l \in \{-1, 1\}$, $h_k \in \mathbf{C}$, $j = 1, \ldots, b$, $k = 1, \ldots, q$, $l = -1, \ldots, -r$, and the nonreal h appear only symmetric—that means together with $\overline{h}$.

Without loss of generality, we assume that the numbers $m_1, \ldots, m_b$ are ordered in ascending order:

$$m_1 = \cdots = m_{n_1} < m_{n_1+1} = \cdots = m_{n_1+n_2} < \cdots$$
$$< m_{n_1+\cdots+n_{s-1}+1} = \cdots = m_{n_1+\cdots+n_s}$$

and set $\widetilde{n}_j := n_1 + \cdots + n_j$, $\widetilde{m}_j := m_{\widetilde{n}_j}$, $j = 1, \ldots, s$; hence $0 < \widetilde{m}_1 < \cdots < \widetilde{m}_s$. Then $b = \widetilde{n}_s$ is the geometric multiplicity and

$$a = m_1 + \cdots + m_b = n_1 \widetilde{m}_1 + \cdots + n_s \widetilde{m}_s$$

is the algebraic multiplicity of the eigenvalue 0 of the pencil $\mu K_1 + C_1$. Moreover,

$$T(\mu, \varepsilon) = X^* T_1(\mu, \varepsilon) X, \quad \text{where} \quad T_1(\mu, \varepsilon) = \mu^2 K_1 + \mu C_1 + \varepsilon P_1$$

with the positive matrix $P_1 = (XX^*)^{-1}$. Evidently, the eigenvalues of T and T_1 coincide; therefore, in the following we consider T_1 instead of T and write T instead of T_1.

Introduce for $j = 1, \ldots, s-1$ the $s-1$ points

$$P^{(j)} = (n - b + n_1(\widetilde{m}_1 + 1) + \cdots + n_j(\widetilde{m}_j + 1), b - n_1 - \cdots - n_j)$$
$$= (n - b + \widetilde{n}_j + n_1 \widetilde{m}_1 + \cdots + n_j \widetilde{m}_j, b - \widetilde{n}_j).$$

The point P_m is formally given as $P^{(0)}$ and P_e as $P^{(s)}$.

PROPOSITION 3.1. *The set* $\mathrm{E}(\mathbf{t})$ *of the extremal points of the Newton diagram of* $\mathbf{t}$ *consists of the* $s+2$ *points* $P_i, P_m, P^{(1)}, \ldots, P^{(s-1)}, P_e$.

Proof. The matrix $T(\mu, \varepsilon)$ can be written in the form

$$T(\mu, \varepsilon) = \begin{bmatrix} T_{11}(\mu, \varepsilon) & \varepsilon P_{12} & \varepsilon P_{13} \\ \varepsilon P_{12}^* & T_{22}(\mu, \varepsilon) & \varepsilon P_{23} \\ \varepsilon P_{13}^* & \varepsilon P_{23}^* & T_{33}(\mu, \varepsilon) \end{bmatrix}$$

with $T_{11}(\mu, \varepsilon)$ corresponding to the first b blocks, $T_{22}(\mu, \varepsilon)$ to the middle q, and $T_{33}(\mu, \varepsilon)$ to the last r blocks. Again, all the off-diagonal blocks in $T_{ii}(\mu, \varepsilon)$ are equal to the corresponding blocks of $P_1 = (XX^*)^{-1}$ multiplied by ε. The off-diagonal blocks do not contribute to the Newton diagram of $\mathbf{t}$ except for the term $\varepsilon^n \det P_1$, in the sense that

$$\mathbf{t}(\mu, \varepsilon) = \varepsilon^n \det P_1 + \det T_{11}(\mu, \varepsilon) \det T_{22}(\mu, \varepsilon) \det T_{33}(\mu, \varepsilon) + \cdots,$$

where $\cdots$ stands for higher order terms giving rise to points which lie strictly above the Newton diagram. The diagonal blocks of $T_{11}(\mu, \varepsilon)$ are of the form

$$\mu^2 J(m_j, \sigma_1) + \mu H(0, m_j, \sigma_j) + \varepsilon P_j^{(1)},$$

where $P_j^{(1)}$ is the corresponding block of P_1; it is a positive definite matrix. Its determinant is of the form

$$\alpha_j \mu^{2m_j} + \beta_j \mu^{m_j - 1} \varepsilon + \gamma_j \varepsilon^{m_j} + \sum \delta_{ij} \mu^i \varepsilon^j,$$

where α_j, β_j, and γ_j are not zero and the terms $\delta_{ij}\mu^i\varepsilon^j$ do not contribute to the Newton diagram of $\det T_{11}(\mu, \varepsilon)$. Similarly, the diagonal blocks of $T_{22}(\mu, \varepsilon)$ are of the form $\mu^2 J(\widehat{m}_k, \sigma_k) + \mu H(h_k, \widehat{m}_k, \sigma_k) + \varepsilon P_k^{(2)}$, where again $P_k^{(2)}$ is the corresponding (positive definite) principal submatrix of P_1. Its determinant is of the form

$$\alpha'_k \mu^{2\widehat{m}_k} + \beta'_k \mu^{\widehat{m}_k} + \gamma'_k \varepsilon^{m_k} + \sum \delta'_{ij} \mu^i \varepsilon^j,$$

where again $\alpha'_k, \beta'_k, \gamma'_k \neq 0$ and the terms $\delta'_{ij}\mu^i\varepsilon^j$ are nonessential. The diagonal blocks of $T_{33}(\mu, \varepsilon)$ are of the form $\mu^2 H(0, m_l, \sigma_l) + \mu J(m_l, \sigma_l) + \varepsilon P_l^{(1)}$ with $P_l^{(l)}$ being a principal submatrix of P_1. Its determinant is of the form

$$\alpha''_l \mu^{m_l} + \gamma''_l \varepsilon^{m_l} + \sum \delta''_{ij} \mu^i \varepsilon^j$$

with $\alpha''_l, \gamma''_l \neq 0$, and $\delta''_{ij}\mu^i\varepsilon^j$ representing nonessential terms. We know from Proposition 2.3 that $P_m = (n-b, b) \in \mathrm{E}(\mathbf{t})$. The corresponding term $\mu^{n-b}\varepsilon^b$ in $\mathbf{t}(\mu, \epsilon)$ comes from the product of *all* terms $\beta_j\mu^{m_j-1}\varepsilon$, $\beta'_k\mu^{m_k}$, and $\alpha''_l\mu^{m_l}$ in

$$\det T_{11}(\mu, \varepsilon) \det T_{22}(\mu, \varepsilon)\ \det T_{33}(\mu, \varepsilon).$$

The points (i, k) lying on $\mathrm{ND}(\mathbf{t})$ and with $i > n-b$ are obtained from similar products, where the exponent of ε decreases as little as possible with the least possible exponent for μ. The way to do this is to gradually replace terms $\beta_j\mu^{m_j-1}\varepsilon$ by terms $\alpha_j\mu^{2m_j}$ in the previous product, starting from terms with the index $\widetilde{m}_1$, the smallest dimension, then doing the same with terms with the index $\widetilde{m}_2$, etc. Each of these substitutions lowers the exponent of ε in 1 and increases that of μ in $\widetilde{m}_j + 1$. This gives rise to the points $P^{(j)}$ and shows that there are no nonzero terms $a_{kj}\mu^k\varepsilon^j$ with (k, j) to the left of the polygonal line determined by P_i and $P^{(l)}$, $l = 0, \ldots, s$. □

From Proposition 3.1 one immediately obtains the following theorem.

THEOREM 3.2. *There are $n-b$ "large" eigenvalues $\lambda_j(\varepsilon)$ which behave asymptotically as (γ_j/ε) if $\varepsilon \to 0$ with $\gamma_j \neq 0$, $\;j = 1, \ldots, n-b$, and for each $j = 1, \ldots, s$, there are $n_j(\widetilde{m}_j + 1)$ "large" eigenvalues $\lambda_{j\nu\tau}(\varepsilon)$ which behave asymptotically as $\gamma_{j\nu}\Theta_{1+\tilde{m}_j,\tau}\varepsilon^{-1/(1+\tilde{m}_j)}$ if $\varepsilon \to 0$ with $\gamma_{j\tau} \neq 0, \tau = 1, \ldots, 1 + \widetilde{m}_j, \nu = 1, \ldots, n_j$, where $\Theta_{m\tau} = \exp\left(\frac{2\pi i}{m}(\tau - 1)\right)$.*

REFERENCES

[1] H. BAUMGÄRTEL, *Analytic Perturbation Theory for Matrices and Operators*, Akademie-Verlag, Berlin, 1984.

[2] I. GOHBERG, P. LANCASTER, AND L. RODMAN, *Perturbations of analytic hermitian matrix functions*, Appl. Anal., 20 (1985), pp. 23–48.

[3] I. Gohberg, P. Lancaster, and L. Rodman, *Matrices and Indefinite Scalar Products*, Birkhäuser, Basel, 1983.

[4] T. Kato, *Perturbation Theory for Linear Operators*, Springer-Verlag, Berlin, Heidelberg, New York, 1996.

[5] H. Langer and B. Najman, *Remarks on the perturbation of analytic matrix functions* II, Integral Equations Operator Theory, 12 (1989), pp. 392–407.

[6] H. Langer and B. Najman, *Remarks on the perturbation of analytic matrix functions* III, Integral Equations Operator Theory, 15 (1992), pp. 798–806.

[7] H. Langer, B. Najman, and K. Veselić, *Perturbation of the eigenvalues of quadratic matrix polynomials*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 474–489.

[8] M. Radjabalipour and A. Salemi, *On eigenvalues of perturbed quadratic matrix polynomials*, Integral Equations Operator Theory, 22 (1995), pp. 242–247.

[9] R. C. Thompson, *Pencils of real symmetric and skew matrices*, Linear Algebra Appl., 147 (1991), pp. 323–371.

[10] M. M. Vainberg and V. A. Trenogin, *Theory of Branching of Solutions of Non-Linear Equations*, Noordhoff, Leyden, 1974.

SIAM J. MATRIX ANAL. APPL.
Vol. 20, No. 2, pp. 428–436

RANK MODIFICATIONS OF SEMIDEFINITE MATRICES ASSOCIATED WITH A SECANT UPDATE FORMULA[*]

MOODY T. CHU[†], R. E. FUNDERLIC[‡], AND GENE H. GOLUB[§]

Abstract. This paper analyzes rank modification of symmetric positive definite matrices H of the form $H - M + P$, where $H - M$ denotes a step of reducing H to a lower-rank, symmetric and positive semidefinite matrix and $(H - M) + P$ denotes a step of restoring $H - M$ to a symmetric positive definite matrix. These steps and their generalizations for rectangular matrices are fully characterized. The well-known BFGS and DFP updates used in Hessian and inverse Hessian approximations provided the motivation for the generalizations and are special cases with H and P having rank one.

Key words. rank-one reduction, Wedderburn theorem, BFGS update, DFP update, quasi-Newton methods, rank subtractivity, rank additivity

AMS subject classifications. 65F30, 65K05, 49D15

PII. S1064827597326651

1. Introduction.

In many applications, it is necessary to alter modestly a given matrix in order to delete some old information or to take in some new information. This paper concerns rank modifications during such an alteration process. We use the terminology that a matrix M *reduces the rank of* H *by* $\operatorname{rank}(M)$ to mean that the rank of the difference $H - M$ is $\operatorname{rank}(M)$ less than that of H. Likewise, we say a matrix P *restores the rank of* $H - M$ if $(H - M) + P$ has rank the same as that of H. Given a symmetric and positive definite matrix H, the purpose of this paper is to address the following two issues:

1. Characterize conditions on M so that $H - M$ is symmetric, positive semidefinite and that $\operatorname{rank}(H - M) = \operatorname{rank}(H) - \operatorname{rank}(M)$.
2. Characterize conditions on P so that $H - M + P$ is symmetric, positive definite and that $\operatorname{rank}(H - M + P) = \operatorname{rank}(H - M) + \operatorname{rank}(P)$, knowing that M has reduced the rank of H by $\operatorname{rank}(M)$.

These points will be seen to completely generalize the BFGS update formula in terms of rank reduction and rank restoration. Furthermore, the rank reduction and restoration steps generalize naturally to rectangular matrices.

There are many other ways to describe rank modifications. For example,

$$H_+ = MH_cM^T + PP^T$$

is another scheme to modify the rank of a given H_c. That is, if M is some lower-rank matrix, then $\operatorname{rank}(MH_cM^T) \leq \operatorname{rank}(H)$. With a suitable choice of P, the rank of H_+ can be restored to that of the original H_c. An application of this can be found

[*] Received by the editors July 3, 1996; accepted for publication (in revised form) by P. Gill April 9, 1998; published electronically October 29, 1998.

http://www.siam.org/journals/simax/20-2/30602.html

[†] Department of Mathematics, North Carolina State University, Raleigh, NC 27695-8205 (chu@math.ncsu.edu). The research of this author was supported in part by National Science Foundation grant DMS-9422280.

[‡] Department of Computer Science, North Carolina State University, Raleigh, NC 27695-8206 (ref@adm.csc.ncsu.edu).

[§] Department of Computer Science, Stanford University, Stanford, CA 94305 (golub@sccm.stanford.edu). The research of this author was supported in part by National Science Foundation grant CCR-9505393.

in [8, Theorem 3.4.3]. As another example, any symmetric update with rank at most two of a symmetric positive definite matrix H_c is of the form

$$H_+ = H_c + \begin{bmatrix} y & z \end{bmatrix} \begin{bmatrix} p & q \\ q & r \end{bmatrix} \begin{bmatrix} y^T \\ z^T \end{bmatrix}, \tag{1.1}$$

where y, z are vectors and p, q, r are scalars. Brodlie, Gourlay, and Greenstadt [2] studied a special case of (1.1); i.e.,

$$H_+ = H_c + \begin{bmatrix} u & H_c v \end{bmatrix} \begin{bmatrix} v^T H_c v & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} u^T \\ (H_c v)^T \end{bmatrix}. \tag{1.2}$$

Their strategy expresses the update in product form

$$H_+ = (I + uv^T)H_c(I + uv^T)^T,$$

thereby maintaining the positive semidefiniteness.

Our motivation of considering rank modifications in the form $H - M + P$ comes from a well-known secant update formula, the BFGS (or its dual DFP) update that approximates the Hessian (or its inverse) of a given real-valued nonlinear function. More details about these secant updates can be found in, for example, [5, 8]. For the purpose of illustrating our general results we quickly recapitulate the formula involved in the BFGS (or DFP) update as follows. Suppose a twice continuously differentiable function $f : R^n \longrightarrow R$ is to be optimized. Let $G(x)$ stand for its Hessian matrix. Let x_c and x_+ denote, respectively, the current and the next approximate of the optimal solution. The secant equation to be satisfied by the approximation H_+ of $G(x_+)$ (or K_+ of $G(x_+)^{-1}$) is

$$H_+ s_c = y_c \quad (\text{or } K_+ y_c = s_c), \tag{1.3}$$

where

$$\begin{aligned} s_c &:= x_+ - x_c, \\ y_c &:= g(x_+) - g(x_c). \end{aligned}$$

The BFGS update H_+ from H_c (or DFP update K_+ from K_c) is given by

$$H_+ := H_c - \frac{H_c s_c s_c^T H_c}{s_c^T H_c s_c} + \frac{y_c y_c^T}{y_c^T s_c} \quad \left(\text{or } K_+ := K_c - \frac{K_c y_c y_c^T K_c}{y_c^T K_c y_c} + \frac{s_c s_c^T}{s_c^T y_c} \right). \tag{1.4}$$

The dual relationship between BFGS and DFP with the interchanges $H \leftrightarrow K$ and $y_c \leftrightarrow s_c$ is well known. Since only the format of the update matters in our discussion, we will refer only to BFGS in the sequel.

The BFGS formula has been studied, extensively applied, and is well understood within the optimization community. For example, it is easy to see that the second term on the right side of (1.4) precisely reduces the rank of H_c by one, while the third term on the right side precisely restores the rank by one. Furthermore, it is easy to see that the matrix that is rank reduced by the second term is positive semidefinite [8, Exercise 3.10]. It is an important property [8, Theorem 3.2.2] that $y_c^T s_c > 0$ is sufficient for the rank restored matrix to be positive definite.

The format of BFGS fits our general theme $H - M + P$ as a special case where M and P are of rank one. From this viewpoint, we develop general conditions on how the rank modification should be carried out to maintain the positive semidefiniteness and hence generalize the class of BFGS update both in terms of choices for update vectors and rank of the modifications.

We will see that any matrix of the form

$$M = HS(S^T HS)^{-1} S^T H \tag{1.5}$$

when subtracted from H reduces the rank of H by $\operatorname{rank}(M)$. Conversely, if the rank of H is symmetrically reduced, then the reducing matrix must be of the form (1.5). Moreover, the conditions under which $H - M$ is positive semidefinite if H is positive semidefinite will be characterized. Similarly, we characterize and analyze the conditions on Λ and Y in order for the rank-restoration step of adding a matrix P of the form $Y\Lambda^{-1}Y^T$ to $H-M$ to be effective. For this rank restoration we show exactly how matrices Λ and Y must relate to M to maintain positive definiteness. In short, we will see that the underlying rationale for the rank-one steps of rank reduction and restoration in the BFGS formula generalizes naturally to matrices M and P of rank higher than one.

We begin in section 2 with a brief discussion of several known results used as tools to help understand the phenomena of rank reduction and restoration. In particular, we review the Wedderburn rank-reduction formula and its generalization. These rank-subtractivity characterizations, i.e., $\operatorname{rank}(H - M) = \operatorname{rank}(H) - \operatorname{rank}(M)$, are important for both the rank-reduction and rank-restoration steps. The rank-subtractivity characterizations are followed by a rank-addition characterization, i.e., $\operatorname{rank}(A + P) = \operatorname{rank}(A) + \operatorname{rank}(P)$, that is key to understanding the restoration step. Finally, a spectral result due to Weyl [18] and a simple consequence show that the reduction step results in a positive semidefinite matrix. In section 3 we present our main results. We begin with a general rank-reduction result for symmetric positive definite matrices followed by a general rank-restoration theorem for rectangular matrices and a symmetric rank-restoration theorem. We conclude with a general characterization of maintaining positive definiteness followed by showing its consistency with the BFGS update. Finally, in section 4 we summarize our results and suggest future study of the simultaneous use of multiple vectors for quasi-Newton methods.

2. Preliminaries.

We begin with a simple, but far reaching result proved by Wedderburn [17, p. 69]. (See Acknowledgment 2.)

LEMMA 2.1. *If $x \in R^n$ and $y \in R^m$ are vectors such that $\omega = y^T Ax \neq 0$, then the matrix*

$$B := A - \omega^{-1} Axy^T A \tag{2.1}$$

has rank exactly one less than the rank of A.

Egerváry [7], though apparently unaware of Wedderburn's result, proved the complete characterization of rank-one subtractivity.

LEMMA 2.2. *Let $u \in R^m$ and $v \in R^n$. Then the rank of the matrix $B = A - \sigma^{-1}uv^T$ is less than that of A if and only if there are vectors $x \in R^n$ and $y \in R^m$ such that $u = Ax$, $v = A^T y$, and $\sigma = y^T Ax$, in which case* $\operatorname{rank}(B) = \operatorname{rank}(A) - 1$.

The Wedderburn rank-one reduction formula (2.1) has led to a general matrix factorization process [3] including the LDU and QR decompositions, the Lanczos algorithm, and the singular value decomposition as its special cases. It was also Lemma 2.2 and its known generalizations that suggested the generalized view of the

BFGS update. The following result by Cline and Funderlic [4] generalizes the Wedderburn rank-one formula to higher rank modifications and is key in our rank-reduction step.

We introduce the terminology *full rank factorization* [1] of a matrix A to mean any factorization $X\Lambda Y^T$ of A such that X and Y each have $\text{rank}(A)$ linearly independent columns and Λ is nonsingular. One way to show such factorizations always exist is to consider an ordered singular value factorization. The diagonal matrix can be reduced to having only nonzero singular values, and the appropriate last rows or columns can be deleted from the other factors.

Lemma 2.3 (general rank-subtractivity lemma). *Let $UR^{-1}V^T$ be a given full rank factorization of a given matrix M. Then* $\text{rank}(A - M) = \text{rank}(A) - \text{rank}(M)$ *if and only if $U = AX$, $V^T = Y^TA$, and $Y^TAX = R$ for some matrices X and Y.*

We remark that the results in both directions of Lemma 2.3 include somewhat subtle implicit considerations of size and rank. For example, X, Y, U, and V each have full column rank equal to $\text{rank}(M) = \text{rank}(R)$ since $Y^TAX = R$ and $UR^{-1}V^T$ is a full rank decomposition of M.

The next result which is the symmetric analogue of Lemma 2.3 is fundamental in understanding symmetric rank modifications. It seems to have been overlooked in the various papers on rank subtractivity.

Lemma 2.4 (symmetric rank-subtractivity lemma). *Let H and M be symmetric matrices. Then* $\text{rank}(H - M) = \text{rank}(H) - \text{rank}(M)$ *if and only if there is a matrix S such that $M = HS(S^THS)^{-1}S^TH$.*

Proof. We first prove the *if* part. By the assumption that S^THS is invertible, the form of M is a full column rank factorization. Lemma 2.3 immediately gives rank subtractivity.

Now we prove the *only if* part. Due to its symmetry, a full rank decomposition of M of the form

$$M = UR^{-1}U^T \tag{2.2}$$

always exists. By Lemma 2.3, there are matrices S and Y such that $U = HS$, $U^T = Y^TH = S^TH^T = S^TH$, and $Y^THS = R$. But $Y^TH = S^TH$ so that $R = S^THS$. Substituting in (2.2) gives

$$M = HS(S^THS)^{-1}S^TH.$$

The assertion is proved. □

This symmetric version is directly related to the BFGS formula and is used for our general symmetric $H - M + P$ scheme. The rank-one case of Lemma 2.4, as is known, gives that the first step of BFGS always reduces the rank of H_c by one, and in addition the converse shows that if the first step reduces the rank of H_c by one, then the matrix that produced the reduction must be of the form $M = Hs(s^THs)^{-1}s^TH$.

The following result from [4, Theorem 3.1] is the key in the rank-restoration step.

Lemma 2.5 (general rank-additivity lemma). *Let A and P be any matrices that are additively compatible. Then rank is additive,*

$$\text{rank}(A + P) = \text{rank}(A) + \text{rank}(P),$$

if and only if there is a matrix B such that $AB = 0$, $BA = 0$, and $PBP = P$.

This rank-additivity result can be understood quite intuitively and reasonably; e.g., let $A := \text{diag}(1, 0, 0)$ and $P := \text{diag}(0, 0, 2)$. Then a sufficient matrix B is

$\mathrm{diag}(0,0,\frac{1}{2})$ as B annihilates A on both sides and is an appropriate generalized inverse for P. If P were rather $\mathrm{diag}(1,0,2)$, then there is no matrix B that both annihilates A and (generalized) inverts P.

The next result appears in [13, p. 70]. Stewart and Sun [16, p. 203] traced the result back to a 1912 paper by Weyl [18] and showed its relation to the Wielandt–Hoffman theorem.

LEMMA 2.6 (Weyl). *Let H and B be symmetric and the eigenvalues of H, B, and $H+B$ be denoted by η_i, β_i, and λ_i, respectively, with orderings*

$$\begin{aligned}\eta_n &\le \eta_{n-1} \le \cdots \le \eta_1,\\ \beta_n &\le \beta_{n-1} \le \cdots \le \beta_1,\\ \lambda_n &\le \lambda_{n-1} \le \cdots \le \lambda_1.\end{aligned}$$

Then

$$\eta_n \le \lambda_i - \beta_i \le \eta_1 \tag{2.3}$$

for $i = 1,\ldots,n$.

The following result is a simple consequence of Weyl's lemma, but plays an important role in the rank-reduction step.

LEMMA 2.7. *If $H \in R^{n\times n}$ is symmetric positive definite and B is symmetric, then $H+B$ has at least $n-\mathrm{rank}(B)$ positive eigenvalues.*

Proof. Arrange all the eigenvalues as in Weyl's lemma, Lemma 2.6. Observe that the smallest eigenvalue η_n of H is positive. Also exactly $n-\mathrm{rank}(B)$ eigenvalues β_i, say, $i = r, r-1, \ldots, r-(n-\mathrm{rank}(B)-1)$ for a certain r, of B are zero. It follows from (2.3) that at least these eigenvalues λ_i, $i = r, r-1, \ldots, r-(n-\mathrm{rank}(B)-1)$, of $H+B$ must be positive. □

3. Main results. We now give the four major results associated with rank modification. The first ensures positive semidefiniteness in the reduction step. The second generalizes restoration for rectangular matrices. Next restoration is restricted to rank-reduced symmetric matrices, and finally positive definite rank restoration is characterized. The well-known rank-reduction and rank-restoration steps of the BFGS formula are special cases of the generalized treatment.

THEOREM 3.1 (rank-reduction theorem). *Suppose that H is symmetric positive semidefinite, M is symmetric, and* $\mathrm{rank}(H-M) = \mathrm{rank}(H) - \mathrm{rank}(M)$. *Then $H-M$ is positive semidefinite.*

Proof. We first prove the theorem when H is positive definite. By letting the matrix $-M$ take the role of B in Lemma 2.7, $H-M$ must have at least $n-\mathrm{rank}(M)$ positive eigenvalues. From the hypothesis that $H-M$ is rank subtractive, $H-M$ has $\mathrm{rank}(M)$ zero eigenvalues. Thus we have accounted for all the eigenvalues of $H-M$, and they are all either positive or zero. Thus $H-M$ is positive semidefinite.

Suppose now H is only semidefinite. We can diagonalize H so that any rank deficiency is depicted by trailing zeros in the lower-right block of the diagonal matrix $D = U^THU$. The rank-subtractivity hypothesis and Lemma 2.4 ensure that

$$M = HS(S^THS)^{-1}S^TH$$

for some matrix S. The inertia of

$$H - M = H - HS(S^THS)^{-1}S^TH$$

is equivalent to that of

$$D - DZ(Z^T DZ)^{-1} Z^T D,$$

where $U^T S = Z$. Note that the zero structure in D eliminates any effect from Z in the lower-right corner of D. Thus it suffices to consider the leading nonzero portion of D which is positive definite. □

The rank-subtractivity, Lemma 2.3, and rank-additivity, Lemma 2.5, lemmas can be thought of as characterizations of rank reduction and restoration for *arbitrary* matrices. We now characterize rank restoration of any matrix $A := H - M$ such that A's rank is subtractive; i.e., $\operatorname{rank}(H - M) = \operatorname{rank}(H) - \operatorname{rank}(M)$. Though this result is more general than needed for our main symmetric theme, it may provide future insights for nonsymmetric or rectangular matrix applications (e.g., Gerber and Luk [9] make use of rectangular matrices for Broyden methods and Deuflhard and Freund [6] develop secant methods with nonsymmetric matrices).

THEOREM 3.2 (general rank-restoration theorem). *Suppose* $\operatorname{rank}(H - M) = \operatorname{rank}(H) - \operatorname{rank}(M)$. *(Therefore, by Lemma* 2.3,

$$M = HS_2(S_1^T HS_2)^{-1} S_1^T H$$

for some matrices S_1 *and* S_2*.) Let* P *be a matrix of* $\operatorname{rank}(M)$*, and let* $P = Y_1 \Lambda^{-1} Y_2^T$ *be a full rank decomposition. Then the following results hold:*

1. *If* $Y_2^T S_2$ *and* $Y_1^T S_1$ *are nonsingular, then the rank is restored; i.e.,*

$$\operatorname{rank}[(H - M) + P] = \operatorname{rank}(H). \tag{3.1}$$

2. *Conversely, if* H *has full column or row rank and* (3.1) *holds, then* $Y_2^T S_2$ *or* $Y_1^T S_1$ *are nonsingular, respectively.*

Proof. To prove the first part of the theorem, define

$$A := H - M = H - HS_2(S_1^T HS_2)^{-1} S_1^T H$$

and

$$B := S_2(Y_2^T S_2)^{-1} \Lambda (S_1^T Y_1)^{-1} S_1^T.$$

Clearly both $AS_2 = 0$ and $S_1^T A = 0$; therefore, $AB = 0$ and $BA = 0$. Furthermore, by direct verification, $PBP = P$. Thus from the general rank-additivity lemma, Lemma 2.5, we have

$$\operatorname{rank}[(H - M) + P] = \operatorname{rank}(H - M) + \operatorname{rank}(P) = \operatorname{rank}(H) - \operatorname{rank}(M) + \operatorname{rank}(P).$$

Note that

$$\operatorname{rank}(P) = \operatorname{rank}(Y_i) = \operatorname{rank}(Y_i^T S_i) = \operatorname{rank}(S_1^T HS_2) = \operatorname{rank}(M),\ i = 1, 2,$$

where the first and the last equalities are due to the full rank decomposition, while the second and the third equalities follow from the nonsingularity of $Y_i^T S_i$ and $S_1^T HS_2$. The assertion (3.1) therefore is proved.

We prove the second part of the theorem by contradiction. Suppose H has full column rank and $Y_2^T S_2$ is singular. Then there would be a nonzero vector x such that $S_2 x \neq 0$ and $Y_2^T S_2 x = 0$. But then $(H - M + P)S_2 x = 0$, a contradiction to

$H - M + P$ having full column rank since $\operatorname{rank}(H - M + P) = \operatorname{rank}(H)$. The proof of the full row rank case is similar. □

We return to our general symmetric theme with the following version of the above general result.

THEOREM 3.3 (symmetric rank-restoration theorem). *Suppose H and M are symmetric matrices satisfying* $\operatorname{rank}(H - M) = \operatorname{rank}(H) - \operatorname{rank}(M)$. *(Therefore, by Lemma* 2.4,

$$M = HS(S^THS)^{-1}S^TH$$

for some matrix S.) Let P be a symmetric matrix of $\operatorname{rank}(S)$, *and let $P = Y\Lambda^{-1}Y^T$ be a full rank decomposition. Then the following results hold:*

1. *If Y^TS is nonsingular, then the rank is restored; i.e.,*

$$\operatorname{rank}[(H - M) + P] = \operatorname{rank}(H). \tag{3.2}$$

2. *Conversely, if H is nonsingular and* (3.2) *holds, then Y^TS is nonsingular.*

Proof. The proof follows immediately from Theorem 3.2 with the choice $S := S_1 = S_2$. □

Notice that in Theorem 3.3 the restoration step (adding a matrix P) specifically presupposes a reduction step (subtracting a matrix M) that has a necessary structure. The actual relationship of the restoration step to the reduction step is particularly important. Just as M has a factor S, P has a factor Y. For rank to be restored, Y and S must be related in the fundamental way of the symmetric restoration theorem.

It is also important to note that the assumption of H being nonsingular (or having full column or row rank in the more general restoration theorem) cannot be weakened in the second part of the above theorem. To see this, use the notation of e_i to denote principal axis vectors, and let $H = \operatorname{diag}(1,1,0) = e_1e_1^T + e_2e_2^T$, $M = \operatorname{diag}(1,0,0) = e_1e_1^T$, and $P = \operatorname{diag}(0,0,1) = e_3e_3^T$. These choices result in a successful rank restoration to $H - M + P = \operatorname{diag}(1,0,1)$, i.e., $\operatorname{rank}(H - M + P) = \operatorname{rank}(H) = 2$; however, $y^Ts = e_3^Te_1 = 0$, a singular matrix of order one.

Finally, we recall that our main concerns, motivated by the BFGS formula, are not only the rank restoration, but also the maintenance of positive definiteness. Toward this end, we provide the next result which completes the *generalization* proposed at the beginning of this paper.

COROLLARY 3.4. *Let H, M, P, S, and Y be the same as in Theorem* 3.3. *Suppose further that H is positive definite. Then $H - M + P$ is positive definite if and only if P is positive semidefinite and Y^TS is nonsingular.*

Proof. We first prove the *if* part. Note that the nonsingularity of Y^TS already implies the nonsingularity of $H-M+P$ from the first part of the symmetric restoration theorem, Theorem 3.3. To infer that $H - M + P$ is positive definite we need only determine that it is positive semidefinite. We know from the reduction theorem, Theorem 3.1, that $H - M$ is positive semidefinite. Together with the assumption that P is positive semidefinite, $H - M + P$ is positive definite.

We now prove the *only if* part. First, Y^TS is nonsingular by the second part of the symmetric restoration theorem, Theorem 3.3, since $H - M + P$ has full rank; therefore

$$\operatorname{rank}(P) = \operatorname{rank}(H - M + P) - \operatorname{rank}(H - M).$$

By Theorem 3.1, where the role of H is played by $H - M + P$ and M by $H - M$, we conclude that P is positive semidefinite. □

We now show the consistency of the general higher rank results with the BFGS rank-one case. Recall that implicitly assumed in the BFGS formula (1.4) are that $s_c^T H_c s_c \neq 0$ and $y_c^T s_c \neq 0$. The latter inequality means that $Y^T S$ is nonsingular in the symmetric restoration theorem. Thus the BFGS formula, composed of its reduction and restoration steps, yields a symmetric nonsingular matrix. Moreover, H_+ is positive definite if and only if $(y_c^T s_c)^{-1} y_c y_c^T$ is positive semidefinite, which is equivalent to $y_c^T s_c > 0$. In summary, if H_c is positive definite and s_c and y_c are any vectors such that $y_c^T s_c > 0$, then $s_c^T H_c s_c > 0$ and P is positive semidefinite so that H_+ is positive definite. Conversely, if there are vectors s_c and y_c for which H_+ is positive definite, then necessarily $y_c^T s_c > 0$.

4. Summary and conclusions. The two rank-one updates involved in the BFGS formula have the function of, in addition to satisfying the secant equation, first reducing and then restoring the rank of the original matrix by one while maintaining positive definiteness. These notions of rank reduction and rank restoration, along with maintaining positive definiteness, have been generalized in this paper to include higher rank modifications in the form $H - M + P$. Specifically, the Wedderburn result and its generalization, Lemma 2.3, provide an exact prescription for the form of the rank update, whereas the rank-restoration results, Theorem 3.3 and Corollary 3.4, specify how such an update will maintain positive definiteness. Theorem 3.2, our most general result, characterizes rank restoration for rectangular matrices.

In addition to reaffirming some fundamental results known in the BFGS formula with our emphasis on rank modification, the theory developed here should help to understand higher rank modifications of the Hessian matrix. We hope this will lead to new considerations or development of a multivector update where, e.g., the heuristic secant equation is replaced by more sophisticated multistep formulas (see, for example, [15, Chapter 7]).

Acknowledgment 1. Professor R. E. Hartwig has kindly pointed out a quicker proof of Theorem 3.1, not directly related to eigenvalues. Let $H = LL^T$ denote the Cholesky decomposition of H. Then

$$H - M = L\left(I - L^T S (S^T L L^T S)^{-1} S^T L\right) L^T. \tag{4.1}$$

Note that the middle factor $I - L^T S (S^T L L^T S)^{-1} S^T L$ in (4.1) is a projection matrix. It follows that $H - M$ is positive semidefinite. The proof based on Weyl's lemma, on the other hand, gives additional insight into the spectrum of $H - M$.

Acknowledgment 2. While this paper was going through copy editing by SIAM, Professor L. Hubert sent us the preprint [14], in which it was pointed out that the 1979 result by Cline and Funderlic [4], Lemma 2.3, was available in its complete form in the applied statistics and psychometrics literature as early as 1957 and should be attributed to Guttman [11, 12]. The sufficient part was available as early as 1944 [10]. From the discussion in the latter, the result was known and had been applied for some years prior to 1944. Guttman attributes what we have called Wedderburn's rank-one reduction formula to Lagrange. We are grateful to Professor Hubert for setting the record straight and for connecting numerical linear algebra with applied statistics and psychometrics in another strong way.

REFERENCES

[1] A. Ben-Israel and T. N. E. Greville, *Generalized Inverses: Theory and Applications*, John Wiley, New York, 1974.

[2] 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. Appl., 11 (1973), pp. 73–82.

[3] M. T. Chu, R. E. Funderlic, and G. H. Golub, *A rank-one reduction formula and its applications to matrix factorizations*, SIAM Rev., 37 (1995), pp. 512–530.

[4] R. E. Cline and R. E. Funderlic, *The rank of a difference of matrices and associated generalized inverses*, Linear Algebra Appl., 24 (1979), pp. 185–215.

[5] J. E. Dennis, Jr. and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1983.

[6] A. Deuflhard and R. W. Freund, *Fast secant methods for the iterative solution of nonsymmetric linear equations*, Sci. Engrg., 2 (1990), pp. 244–276.

[7] E. Egerváry, *On rank-diminishing operators and their applications to the solution of linear equations*, Z. Angew. Math. Phys., 11 (1960), pp. 376–386.

[8] R. Fletcher, *Practical Methods of Optimization*, 2nd ed., John Wiley, Chichester, UK, 1987.

[9] R. R. Gerber and F. T. Luk, *A generalized Broyden's method for solving simultaneous linear equations*, SIAM J. Numer. Anal., 18 (1981), pp. 882–890.

[10] L. Guttman, *General theory and methods for matrix factoring*, Psychometrika, 9 (1944), pp. 1–16.

[11] L. Guttman, *Multiple group methods for common-factor analysis: Their basis, computation, and interpretation*, Psychometrika, 17 (1952), pp. 209–222.

[12] L. Guttman, *A necessary and sufficient formula for matrix factoring*, Psychometrika, 22 (1957), pp. 79–81.

[13] A. S. Householder, *The Theory of Matrices in Numerical Analysis*, Blaisdell, New York, 1964, and Dover, New York, 1975.

[14] L. Hubert, J. Meulman, and W. Heiser, *A Brief Tale of Two Purposes for Matrix Factorization*, Preprint, Department of Psychology, University of Illinois, 1998.

[15] J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, San Diego, CA, 1970.

[16] G. W. Stewart and J. G. Sun, *Matrix Perturbation Theory*, Academic Press, Boston, 1990.

[17] J. H. M. Wedderburn, *Lectures on Matrices, Colloquium Publications*, Vol. XVII, American Mathematical Society, New York, 1934, and Dover, New York, 1964.

[18] H. Weyl, *Das asymptotische Verteilungsgesetz der Eigenwerte linearer partieller Differentialgleichungen*, Math. Ann., 71 (1912), pp. 441–479.

SIAM J. Matrix Anal. Appl.
Vol. 20, No. 2, pp. 437–445

SUFFICIENT CONDITIONS FOR REGULARITY AND SINGULARITY OF INTERVAL MATRICES*

GEORG REX† AND JIRI ROHN‡

Abstract. Several verifiable sufficient conditions for regularity and singularity of interval matrices are given. As an application, a verifiable sufficient condition is derived for an interval matrix to have all eigenvalues real.

Key words. interval matrix, regularity, singularity, inverse matrix, eigenvalue, positive definiteness

AMS subject classifications. 15A18, 65G10

PII. S0895479896310743

1. Introduction.

As is well known, an interval matrix

$$A^I = [\underline{A}, \overline{A}] = \{A;\ \underline{A} \le A \le \overline{A}\}$$

(where $\underline{A}$ and $\overline{A}$ are $n \times n$ matrices and the inequalities are understood componentwise) is called regular if each $A \in A^I$ is nonsingular, and it is said to be singular otherwise (i.e., if it contains a singular matrix). Regularity of interval matrices plays an important role in the theory of linear interval equations (cf. Neumaier [13]), but it is also important in another respect since several frequently used properties of interval matrices (as positive definiteness, P-property, stability, and Schur stability) may be verified via checking regularity (see Rohn and Rex [24], Rohn [22]).

The problem of checking regularity of interval matrices has been proved to be NP-hard (Poljak and Rohn [15], [16]; see also Nemirovskii [10]). In its most recent version [23], the result says that for each rational $\varepsilon > 0$ checking regularity is NP-hard in the class of interval matrices of the form

$$[A - \varepsilon E, A + \varepsilon E],$$

where A is a nonnegative symmetric positive definite rational matrix and E is the matrix of all ones.

In view of this NP-hardness result and of the current status of the complexity theory (the conjecture P≠NP; cf. Garey and Johnson [3]), no easily performable (i.e., polynomial-time) algorithms for checking regularity of interval matrices may be expected to exist. In practical computations we must therefore resort to verifiable sufficient conditions for both regularity and singularity of interval matrices. In order to cover a possibly wide class of interval matrices, it is recommendable to have more such conditions at one's disposal since some sufficient conditions may be better suited

*Received by the editors October 15, 1996; accepted for publication (in revised form) by A. Bunse-Gerstner March 24, 1998; published electronically October 29, 1998. This research was supported by Charles University Grant Agency grant GAUK 195/96.

http://www.siam.org/journals/simax/20-2/31074.html

†Institute of Mathematics, University of Leipzig, Augustusplatz 10-11, D-04109 Leipzig, Germany (rex@mathematik.uni-leipzig.de).

‡Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic, and Institute of Computer Science, Academy of Sciences, Prague, Czech Republic (rohn@uivt.cas.cz). The research of this author was supported by the Sächsisches Staatsministerium für Wissenschaft und Kunst in Dresden, Germany.

for specific classes of interval matrices than the other ones. Such a situation is well known for the problem of stability of interval matrices which is also NP-hard (Nemirovskii [10], Rohn [23]), where a number of sufficient conditions of different types are known; see the survey paper by Mansour [9].

The purpose of this paper is threefold. First, we give three sufficient regularity conditions and three sufficient singularity conditions, grouped into pairs according to their form (conditions using midpoint inverse, conditions using eigenvalues, and conditions based on checking positive definiteness). Two of them (Theorems 3.1 and 4.1) are already known; the others (Theorems 3.3, 4.2, 5.1, and 5.2) are new. Second, we show that all these verifiable sufficient conditions can be derived in a rather uniform way from two necessary and sufficient conditions that themselves are not of practical use since they require a number of arithmetic operations which is exponential in the matrix size n. Third, as an application of the previous results, we give in Theorem 6.1 a verifiable sufficient condition for an interval matrix to have all eigenvalues real.

We shall use the following notations. The absolute value of a matrix $A = (a_{ij})$ is denoted by $|A| = (|a_{ij}|)$; the same notation applies to vectors as well. $\varrho(A)$ is the spectral radius of A, and $\lambda_{\min}(A)$ and $\lambda_{\max}(A)$ stand for the minimal and maximal eigenvalue of a symmetric matrix A, respectively. As is well known, $\lambda_{\min}(A) = \min_{\|x\|_2=1} x^TAx$ and $\lambda_{\max}(A) = \max_{\|x\|_2=1} x^TAx$ hold for a symmetric matrix A; see Golub and van Loan [4]. I denotes the unit matrix.

2. Necessary and sufficient conditions.

For an interval matrix

$$A^I = [\underline{A}, \overline{A}], \tag{2.1}$$

let us introduce

$$A_c = \tfrac{1}{2}(\underline{A} + \overline{A})$$

(the midpoint matrix) and

$$\Delta = \tfrac{1}{2}(\overline{A} - \underline{A})$$

(the radius matrix). Then we can write (2.1) as

$$A^I = [A_c - \Delta, A_c + \Delta], \tag{2.2}$$

a form better suited for formulations of the subsequent conditions.

The first known necessary and sufficient condition for singularity of interval matrices is due to Oettli and Prager. In fact, the formulation given below cannot be found explicitly in their original paper [14], but it follows easily from the basic theorem on linear interval equations given there when applied to systems with zero right-hand sides (see Neumaier [13] or Rohn [20]).

THEOREM 2.1 (due to Oettli and Prager [14]). *An interval matrix* (2.2) *is singular if and only if the inequality*

$$|A_cx| \le \Delta|x| \tag{2.3}$$

has a nontrivial solution.

The proof, given, e.g., in [20], is constructive: if (2.3) holds for some $x \neq 0$, then for the vectors $y, z \in \mathbb{R}^n$ defined by

$$y_i = \begin{cases} (A_cx)_i/(\Delta|x|)_i & \text{if } (\Delta|x|)_i \neq 0, \\ 1 & \text{if } (\Delta|x|)_i = 0 \end{cases}$$

and

$$z_j = \begin{cases} 1 & \text{if } x_j \geq 0, \\ -1 & \text{if } x_j < 0 \end{cases}$$

$(i, j = 1, \ldots, n)$, the matrix A given by

$$A_{ij} = (A_c)_{ij} - y_i z_j \Delta_{ij}$$

$(i, j = 1, \ldots, n)$ belongs to A^I and is singular (since $Ax = 0$). Hence, we can construct a singular matrix in A^I if we know a nontrivial solution to (2.3). However, in view of the NP-hardness result, such a solution is not to be found easily.

The following necessary and sufficient regularity condition employs again the inequality of the form (2.3), with the $\leq$ sign being converted to $>$. We emphasize that the strict inequality is again meant componentwise.

THEOREM 2.2 (due to Rohn [21]). *An interval matrix* (2.2) *is regular if and only if for each orthant $\mathcal{O}$ of $\mathbb{R}^n$ there exists a solution of the inequality*

$$|A_c x| > \Delta |x| \tag{2.4}$$

satisfying $A_c x \in \mathcal{O}$.

Unlike the previous theorem, the proof of this result is more involved and employs some nontrivial facts concerning the linear complementarity problem, P-matrices, and solvability of linear equations. Again, Theorem 2.2 is of little practical use since it requires a proof of existence of 2^n solutions of the inequality (2.4). However, Theorems 2.1 and 2.2 form a basis for deriving some verifiable sufficient conditions for regularity and singularity of interval matrices that will be given in the three subsequent sections.

3. Sufficient conditions using the midpoint inverse. Two known sufficient conditions (given below as Corollaries 3.2 and 3.4) use the midpoint inverse A_c^{-1} in their formulations. Since using the inverse matrix computed in a finite precision arithmetic may affect validity of these conditions, it is advantageous to formulate them in terms of an approximate inverse R instead of the exact inverse A_c^{-1}. The first such formulation appeared, although implicitly, in Rump's paper [25]. We reprove the condition here, using another idea based on Theorem 2.1.

THEOREM 3.1. *Let R be an arbitrary matrix such that*

$$\varrho(|I - RA_c| + |R|\Delta) < 1 \tag{3.1}$$

holds. Then $[A_c - \Delta, A_c + \Delta]$ is regular.

Proof. Assume to the contrary that $[A_c - \Delta, A_c + \Delta]$ is singular, so that by Theorem 2.1 there exists an $x \neq 0$ satisfying $|A_c x| \leq \Delta |x|$. Then we have

$$\begin{aligned} |x| = |(I - RA_c)x + RA_c x| &\leq |I - RA_c| \cdot |x| + |R| \cdot |A_c x| \\ &\leq (|I - RA_c| + |R|\Delta)|x|, \end{aligned}$$

hence

$$1 \leq \varrho(|I - RA_c| + |R|\Delta)$$

by the Perron–Frobenius theorem (see Neumaier [13, Cor. 3.2.3]), which is a contradiction. □

If we take $R = A_c^{-1}$, we immediately obtain the following result.

Corollary 3.2. *Let A_c be nonsingular and*

$$\varrho(|A_c^{-1}|\Delta) < 1 \tag{3.2}$$

hold. Then $[A_c - \Delta, A_c + \Delta]$ is regular.

The condition (3.2) was first published by Beeck [2], although allegedly (Neumaier [12]) its priority is due to Ris, who had proved it earlier in his unpublished Ph.D. thesis [19].

In his recent papers [28], [26], Rump proved that each regular $n \times n$ interval matrix $[A_c - \Delta, A_c + \Delta]$ satisfies

$$\varrho(|A_c^{-1}|\Delta) < (3 + 2\sqrt{2})n$$

and that for each $n \geq 1$ there exists a regular $n \times n$ interval matrix such that

$$\varrho(|A_c^{-1}|\Delta) > n - 1.$$

These facts help to clarify the strength of the sufficient condition (3.2).

Since (3.2) is a special case of (3.1) for $R = A_c^{-1}$, it was believed for some time that (3.1) is more general than (3.2). Rather surprisingly, it turned out that it is not so: Rex and Rohn [18] proved that if (3.1) is valid, then A_c is nonsingular and

$$\varrho(|A_c^{-1}|\Delta) \leq \varrho(|I - RA_c| + |R|\Delta)$$

holds; hence (3.1) implies (3.2), so that both conditions cover the same class of interval matrices. This result also shows that the midpoint inverse A_c^{-1} is the best option for the choice of R. For related results, see Neumaier [11] and Rex [17].

Let us note that condition (3.2) is verifiable in polynomial time since it is equivalent to

$$(I - |A_c^{-1}|\Delta)^{-1} \geq 0$$

and the inverse matrix can be evaluated in polynomial time by a modified Gaussian elimination (Bareiss [1]); this statement is of theoretical interest only since efficient numerical methods for checking (3.2) are available. The same reasoning applies also to (3.1) provided R is computed in polynomial time.

Next we prove a sufficient singularity condition of a similar type. Let A_j denote the jth column of a matrix A.

Theorem 3.3. *Let there exist a matrix R such that*

$$(I + |I - A_cR|)_j \leq (\Delta|R|)_j \tag{3.3}$$

holds for some $j \in \{1, \ldots, n\}$. Then $[A_c - \Delta, A_c + \Delta]$ is singular.

Proof. Assumption (3.3) implies

$$\begin{aligned}|A_cR_j| = |A_cR|_j = |I - (I - A_cR)|_j &\leq I_j + |I - A_cR|_j \\ &\leq (\Delta|R|)_j = \Delta|R_j|,\end{aligned}$$

so that for $x := R_j$ we have

$$|A_cx| \leq \Delta|x|,$$

where $x \neq 0$ due to (3.3); hence $[A_c - \Delta, A_c + \Delta]$ is singular by Theorem 2.1. ◻

Since the vector $x = R_j$ satisfies (2.3), we may employ the procedure described after Theorem 2.1 to construct a singular matrix contained in $[A_c - \Delta, A_c + \Delta]$. Setting $R = A_c^{-1}$, we immediately obtain as a special case a result from [20].

COROLLARY 3.4. *Let A_c be nonsingular and let*

$$\max_j (\Delta |A_c^{-1}|)_{jj} \geq 1 \tag{3.4}$$

hold. Then $[A_c - \Delta, A_c + \Delta]$ is singular.

Proof. Let j be the index for which $(\Delta |A_c^{-1}|)_{jj} \geq 1$. Then (3.3) holds with $R = A_c^{-1}$, and Theorem 3.3 applies. □

While this paper was in the review process, Rump published a generalization of condition (3.4): if

$$\max_{ij} (\Delta |A_c^{-1}|)_{ij} (\Delta |A_c^{-1}|)_{ji} \geq 1 \tag{3.5}$$

holds, then $[A_c - \Delta, A_c + \Delta]$ is singular [27, Thm. 6.5]. Obviously, (3.4) is a special case of (3.5) for $i = j$.

4. Sufficient conditions using eigenvalues.

If A_c is nearly singular, then the conditions using approximate midpoint inverse may turn ineffective. Rump [26] was the first to derive a condition where no inverse matrix computation is required, at the expense of necessity to evaluate eigenvalues. Here we reprove his result by another means.

THEOREM 4.1. *Let*

$$\lambda_{\max}(\Delta^T \Delta) < \lambda_{\min}(A_c^T A_c) \tag{4.1}$$

hold. Then $[A_c - \Delta, A_c + \Delta]$ is regular.

Proof. Assume to the contrary that $[A_c - \Delta, A_c + \Delta]$ is singular, so that

$$|A_c x| \leq \Delta |x|$$

holds for some $x \neq 0$, which may be normalized to achieve $\|x\|_2 = 1$. Then we have

$$\begin{aligned} \lambda_{\min}(A_c^T A_c) &\leq x^T A_c^T A_c x \leq |A_c x|^T |A_c x| \leq (\Delta |x|)^T (\Delta |x|) \\ &= |x|^T \Delta^T \Delta |x| \leq \lambda_{\max}(\Delta^T \Delta), \end{aligned}$$

which contradicts (4.1). □

Let us note that the matrices $A_c^T A_c$ and $\Delta^T \Delta$ are symmetric; hence their eigenvalues appearing in (4.1) are real. Rump [29] and independently Vacek [30] found counterexamples demonstrating that neither of the conditions (3.2), (4.1) is a consequence of the other one.

The above result employed Theorem 2.1; using Theorem 2.2, we arrive at a sufficient singularity condition formulated in similar terms.

THEOREM 4.2. *Let*

$$\lambda_{\max}(A_c^T A_c) \leq \lambda_{\min}(\Delta^T \Delta) \tag{4.2}$$

hold. Then $[A_c - \Delta, A_c + \Delta]$ is singular.

Proof. Assume to the contrary that $[A_c - \Delta, A_c + \Delta]$ is regular. Then according to Theorem 2.2, applied to the nonnegative orthant $\mathcal{O}$, there exists an x satisfying

$$A_c x > \Delta |x|,$$

which can be normalized so that $\|x\|_2 = 1$. Then we have

$$\lambda_{\max}(A_c^T A_c) \geq x^T A_c^T A_c x > |x|^T \Delta^T \Delta |x| \geq \lambda_{\min}(\Delta^T \Delta),$$

contrary to (4.2). □

5. Sufficient conditions using positive definiteness.

The necessity of evaluating eigenvalues in Theorems 4.1 and 4.2 may be avoided if we use instead a positive definiteness check. Let us recall that a symmetric matrix A (it will be seen that symmetry poses no restriction here) is positive definite if and only if all its leading principal minors are positive (Sylvester determinant criterion; see Wilkinson [31]). Since positivity of all leading principal minors may be checked by employing a modified Gaussian elimination which is performable in polynomial time (Bareiss [1]), we can see that checking positive definiteness of symmetric matrices may be performed by a polynomial-time algorithm. This is the advantage of criteria presented in this section; their disadvantage consists of the fact that they require evaluation of $A_c^T A_c$ (or $\Delta^T \Delta$), which squares the condition number.

Theorem 5.1. *Let the matrix*

$$A_c^T A_c - \|\Delta^T \Delta\| I \tag{5.1}$$

be positive definite for some consistent matrix norm $\|\cdot\|$. Then $[A_c - \Delta, A_c + \Delta]$ is regular.

Proof. As in the proof of Theorem 4.1, assuming to the contrary that $[A_c - \Delta, A_c + \Delta]$ is singular, we may ensure existence of an x with $\|x\|_2 = 1$ satisfying

$$x^T A_c^T A_c x \leq |x|^T \Delta^T \Delta |x| \leq \lambda_{\max}(\Delta^T \Delta) = \varrho(\Delta^T \Delta) \leq \|\Delta^T \Delta\| = \|\Delta^T \Delta\|(x^T x);$$

hence

$$x^T (A_c^T A_c - \|\Delta^T \Delta\| I) x \leq 0,$$

which means that the matrix (5.1) is not positive definite, which is a contradiction. □

Notice that the matrix (5.1) is symmetric, which justifies the discussion made at the beginning of this section. Since $\|\Delta^T \Delta\| \leq \|\Delta^T\| \cdot \|\Delta\|$, Theorem 5.1 will remain valid if we replace (5.1) with the matrix

$$A_c^T A_c - \|\Delta^T\| \cdot \|\Delta\| I,$$

which yields a weaker result where, however, $\Delta^T \Delta$ need not be computed. We note that any of the usual matrix norms $\|\cdot\|_1$, $\|\cdot\|_2$, $\|\cdot\|_\infty$, and $\|\cdot\|_F$ is consistent [5] and may be employed in Theorem 5.1. The theorem will not stay in force if the matrix (5.1) is replaced by

$$A_c^T A_c - \Delta^T \Delta.$$

Indeed, for

$$A_c = \begin{pmatrix} 1 & 4 \\ 4 & 1 \end{pmatrix}, \qquad \Delta = \begin{pmatrix} 2 & 2 \\ 2 & 2 \end{pmatrix},$$

the matrix $A_c^T A_c - \Delta^T \Delta = 9I$ is positive definite, but $[A_c - \Delta, A_c + \Delta]$ contains the singular matrix

$$\begin{pmatrix} 3 & 3 \\ 3 & 3 \end{pmatrix}$$

(Rump [29]). Finally, we formulate in similar terms a sufficient singularity condition.

THEOREM 5.2. *Let the matrix*

$$\Delta^T\Delta - A_c^T A_c \tag{5.2}$$

be positive semidefinite. Then $[A_c - \Delta, A_c + \Delta]$ *is singular.*

Proof. Assume to the contrary that $[A_c - \Delta, A_c + \Delta]$ is regular. Then Theorem 2.2 (applied to the nonnegative orthant) implies existence of an $x \neq 0$ satisfying

$$A_c x > \Delta|x|$$

and henceforth also

$$x^T A_c^T A_c x > |x|^T \Delta^T \Delta |x| \geq x^T \Delta^T \Delta x,$$

which means that

$$x^T(\Delta^T\Delta - A_c^T A_c)x < 0$$

and the matrix (5.2) is not positive semidefinite, which contradicts the assumption. □

6. Application: Condition for an interval matrix to have real eigenvalues only.

As an application of the above results, different from those mentioned in the introduction, we shall consider the problem of checking that each $A \in A^I$ has real eigenvalues only. The single reference on this problem known to us is the paper by Hollot and Bartlett [6]; the necessary and sufficient condition given there, however, is not of practical use since it is exponential in the matrix size. We have this verifiable sufficient condition.

THEOREM 6.1. *Let* A_c *have* n *simple real eigenvalues*

$$\lambda_1(A_c) < \lambda_2(A_c) < \cdots < \lambda_n(A_c),$$

and let there exist real numbers $\mu_0, \ldots, \mu_n$ *satisfying*

$$\mu_0 < \lambda_1(A_c) < \mu_1 < \lambda_2(A_c) < \mu_2 < \cdots < \lambda_n(A_c) < \mu_n \tag{6.1}$$

such that the interval matrix

$$[A_c - \mu_j I - \Delta, A_c - \mu_j I + \Delta] \tag{6.2}$$

is regular for $j = 0, \ldots, n$. *Then each* $A \in A^I$ *has* n *simple real eigenvalues satisfying*

$$\mu_0 < \lambda_1(A) < \mu_1 < \lambda_2(A) < \mu_2 < \cdots < \lambda_n(A) < \mu_n. \tag{6.3}$$

Proof. For an $A \in A^I$, let

$$p(\lambda) = \det(A - \lambda I)$$

denote its characteristic polynomial and let

$$p_c(\lambda) = \det(A_c - \lambda I)$$

be the characteristic polynomial of A_c. Then for each $j \in \{0, \ldots, n\}$ we have $|(A - \mu_j I) - (A_c - \mu_j I)| = |A - A_c| \leq \Delta$; hence

$$A - \mu_j I \in [A_c - \mu_j I - \Delta, A_c - \mu_j I + \Delta],$$

and regularity of (6.2) implies

$$p(\mu_j)p_c(\mu_j) > 0 \tag{6.4}$$

since $p(\mu_j)p_c(\mu_j) \leq 0$ would imply, by continuity of the determinant, existence of a singular matrix in (6.2), which is a contradiction. Now, since all eigenvalues of A_c are real and simple, (6.1) gives

$$p_c(\mu_j)p_c(\mu_{j+1}) < 0 \tag{6.5}$$

for $j = 0, \dots, n-1$. For each such j we have from (6.4)

$$p(\mu_j)p_c(\mu_j)p(\mu_{j+1})p_c(\mu_{j+1}) > 0,$$

which in view of (6.5) implies

$$p(\mu_j)p(\mu_{j+1}) < 0;$$

hence the characteristic polynomial of A has a root in each of the open intervals (μ_j, μ_{j+1}), $j = 0, \dots, n-1$. This proves that A has exactly n simple real eigenvalues satisfying (6.3). □

Regularity of the interval matrix (6.2) may be checked by any of the sufficient regularity conditions presented above. Theorem 5.1 seems to be particularly suited here since it requires checking positive definiteness of the matrices

$$(A_c - \mu_j I)^T(A_c - \mu_j I) - \|\Delta^T\Delta\| I = A_c^T A_c - \mu_j(A_c^T + A_c) + (\mu_j^2 - \|\Delta^T\Delta\|)I,$$

which may be easily updated for different values of μ_j.

7. Concluding remarks. In a very recent development, Jansson [7] proposed a necessary and sufficient regularity condition, based on a quite different idea, which is not a priori exponential (an exponential growth occurs only at worst–case-type examples). Computational results reported in [8] look promising: interval matrices up to the size $n = 50$ were checked in acceptable time. Nevertheless, in view of the NP-hardness result and of the famous conjecture P≠NP (see Garey and Johnson [3] for details) there remains only very little hope that necessary and sufficient regularity conditions verifiable in polynomial time might be found.

Acknowledgments. The second author wishes to acknowledge the support granted to him by the Sächsisches Staatsministerium für Wissenschaft und Kunst in Dresden for funding his stay at the Center of Theoretical Sciences of the University of Leipzig. Helpful comments by the referees and by Prof. S. M. Rump are acknowledged with thanks.

REFERENCES

[1] E. F. Bareiss, *Sylvester's identity and multistep integer-preserving Gaussian elimination*, Math. Comp., 103 (1968), pp. 565–578.

[2] H. Beeck, *Zur Problematik der Hüllenbestimmung von Intervallgleichungssystemen*, in Interval Mathematics, Lecture Notes in Comput. Sci. 29, K. Nickel, ed., Springer-Verlag, Berlin, 1975, pp. 150–159.

[3] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP–Completeness*, Freeman, San Francisco, CA, 1979.

[4] G. H. Golub and C. F. van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, MD, 1996.

[5] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia, 1996.

[6] C. V. Hollot and A. C. Bartlett, *On the eigenvalues of interval matrices*, in Proc. 26th IEEE Conf. on Decision and Control, Los Angeles, CA, IEEE Computer Society Press, Los Alamitos, CA, 1987, pp. 794–799.

[7] C. Jansson, *Calculation of exact bounds for the solution set of linear interval systems*, Linear Algebra Appl., 251 (1997), pp. 321–340.

[8] C. Jansson and J. Rohn, *An algorithm for checking regularity of interval matrices*, SIAM J. Matrix Anal. Appl., to appear.

[9] M. Mansour, *Robust stability of interval matrices*, in Proc. 28th IEEE Conf. on Decision and Control, Tampa, FL, IEEE Computer Society Press, Los Alamitos, CA, 1989, pp. 46–51.

[10] A. Nemirovskii, *Several NP-hard problems arising in robust stability analysis*, Math. Control Signals Systems, 6 (1993), pp. 99–105.

[11] A. Neumaier, *New techniques for the analysis of linear interval equations*, Linear Algebra Appl., 58 (1984), pp. 273–325.

[12] A. Neumaier, *Linear interval equations*, in Interval Mathematics 1985, Lecture Notes in Comput. Sci. 212, K. Nickel, ed., Springer-Verlag, Berlin, 1985, pp. 109–120.

[13] A. Neumaier, *Interval Methods for Systems of Equations*, Cambridge University Press, Cambridge, 1990.

[14] W. Oettli and W. Prager, *Compatibility of approximate solution of linear equations with given error bounds for coefficients and right-hand sides*, Numer. Math., 6 (1964), pp. 405–409.

[15] S. Poljak and J. Rohn, *Radius of Nonsingularity*, Research Report, KAM Series 88–117, Faculty of Mathematics and Physics, Charles University, Prague, 1988.

[16] S. Poljak and J. Rohn, *Checking robust nonsingularity is NP-hard*, Math. Control Signals Systems, 6 (1993), pp. 1–9.

[17] G. Rex, *Zum Regularitätsnachweis von Matrizen*, Z. Angew. Math. Mech., 75 (1995), pp. S549–S550.

[18] G. Rex and J. Rohn, *A note on checking regularity of interval matrices*, Linear and Multilinear Algebra, 39 (1995), pp. 259–262.

[19] F. N. Ris, *Interval Analysis and Applications to Linear Algebra*, Ph.D. thesis, Oxford University, Oxford, 1972.

[20] J. Rohn, *Systems of linear interval equations*, Linear Algebra Appl., 126 (1989), pp. 39–78.

[21] J. Rohn, *Linear Interval Equations: Enclosing and Nonsingularity*, Research Report, KAM Series 89–141, Faculty of Mathematics and Physics, Charles University, Prague, 1989.

[22] J. Rohn, *Positive definiteness and stability of interval matrices*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 175–184.

[23] J. Rohn, *Checking positive definiteness or stability of symmetric interval matrices is NP-hard*, Comment. Math. Univ. Carolin., 35 (1994), pp. 795–797.

[24] J. Rohn and G. Rex, *Interval P-matrices*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 1020–1024.

[25] S. M. Rump, *Solving algebraic problems with high accuracy*, in A New Approach to Scientific Computation, U. Kulisch and W. Miranker, eds., Academic Press, New York, 1983, pp. 51–120.

[26] S. M. Rump, *Verification methods for dense and sparse systems of equations*, in Topics in Validated Computations, J. Herzberger, ed., North–Holland, Amsterdam, 1994, pp. 63–135.

[27] S. M. Rump, *Bounds for the componentwise distance to the nearest singular matrix*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 83–103.

[28] S. M. Rump, *Almost sharp bounds for the componentwise distance to the nearest singular matrix*, Linear and Multilinear Algebra, 42 (1998), pp. 93–108.

[29] S. M. Rump, *Personal communication.*

[30] J. Vacek, *Personal communication.*

[31] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford University Press, London, 1965.

SIAM J. MATRIX ANAL. APPL.
Vol. 20, No. 2, pp. 446–465

TOEPLITZ PRECONDITIONERS CONSTRUCTED FROM LINEAR APPROXIMATION PROCESSES*

STEFANO SERRA CAPIZZANO†

Abstract. Preconditioned conjugate gradients (PCG) are widely and successfully used methods to solve Toeplitz linear systems $A_n(f)\mathbf{x} = \mathbf{b}$. Here we consider preconditioners belonging to trigonometric matrix algebras and to the band Toeplitz class and we analyze them from the viewpoint of the function theory in the case where f is supposed continuous and strictly positive. First we prove that the necessary (and sufficient) condition, in order to devise a superlinear PCG method, is that the spectrum of the preconditioners is described by a sequence of approximation operators "converging" to f. The other important information we deduce is that while the matrix algebra approach is substantially not sensitive to the approximation features of the underlying approximation operators, the band Toeplitz approach is. Therefore, the only class of methods for which we may obtain impressive evidence of *superlinear* convergence behavior is the one [S. Serra, *Math. Comp.*, 66 (1997), pp. 651–665] based on band Toeplitz matrices with weakly increasing bandwidth.

Key words. approximation operators, Toeplitz matrix, matrix algebra, clustering and preconditioning

AMS subject classification. 65F10

PII. S0895479897316904

1. Introduction.

Toeplitz matrices and operators arise in and have application to a wide variety of fields of pure and applied mathematics, i.e., numerical analysis, probability theory, complex analysis, K-theory, statistical mechanics, and so on, [61]. In this paper, in order to make efficient computations, we consider the approximation of finite self-adjoint Toeplitz operators $A_n(f)$ by means of matrix algebra operators or band Toeplitz operators. Here the Toeplitz matrix $A_n(f)$ of size $n \times n$ is generated by a Lebesgue-integrable function f in the sense that the entries of $A_n(f)$ along the kth diagonal are given by the kth Fourier coefficient a_k of f:

$$[A_n(f)]_{i,j} = a_{i-j}, \quad a_k = \frac{1}{2\pi}\int_{-\pi}^{\pi} f(x)e^{-\mathbf{i}kx}dx, \quad \mathbf{i}^2 = -1, \quad k \in \mathbf{Z}. \tag{1.1}$$

In some applications [34, 33, 42, 31] the problem is often the solution of linear systems $A_n(f)\mathbf{x} = \mathbf{b}$, where the dimension n is related to a "grid parameter" of the discretization. Therefore, in most of these cases, the systems show very large dimensions and so we have to find efficient resolution methods.

In previous papers, we have introduced iterative solvers based on band Toeplitz matrices [29, 44, 30, 47, 46, 48] for the ill-conditioned case, by showing that the "functional approach" is the most successful to handle the situations in which the sequence of the Euclidean condition numbers of $\{A_n(f)\}_n$ is unbounded. By following these ideas we obtained optimal but also *superlinear* methods for these ill-conditioned linear systems.

Here, we come back to the well-conditioned case in which the generating function f is strictly positive. Until now, the best methods [56, 12, 6, 18, 17, 36, 58, 27, 4,

*Received by the editors February 14, 1997; accepted for publication (in revised form) by D. Calvetti January 27, 1998; published electronically November 16, 1998.

http://www.siam.org/journals/simax/20-2/31690.html

†Dipartimento di Energetica, Università di Firenze, Via Lombroso 6/17, 50134 Firenze, Italy, and Dipartimento di Informatica, Universitá di Pisa, Corso Italia 40, 56100 Pisa, Italy (serra@mail.dm.unipi.it).

38, 23, 24, 53, 52, 25, 26] (see also the very rich survey of R. Chan and M. Ng [11] and references therein) have been considered to be those based on the preconditioned conjugate gradient (PCG) methods in which the preconditioners are chosen in some trigonometric matrix algebras [52]. By following this approach, originated in the pioneering work of Gilbert Strang [56], *superlinear* methods have been proposed. In spite of the variety in the definition of the preconditioners, the performances of all these PCG methods are substantially equivalent. Actually, with the exception of the *superoptimal* preconditioner [58], which is not as good as expected [26], all the other preconditioners (natural approach [56, 3], optimal [18, 3, 36], and so on) show the same kind of performances. This fact is especially evident in the paper of R. Chan and M. Yeung [17], where several circulant preconditioners constructed from approximation kernels $\mathcal{S}_n$ are considered and analyzed.

The related invariance of the convergence features, with respect to the approximation kernels was, in my opinion, a bit surprising and unnatural. Actually, given a function f, the Cesàro sum $\mathcal{S}_n = C_n(f)$ and the De La Vallée approximation scheme [63, p. 115] $\mathcal{S}_n = DL_n(f)$ are uniformly convergent to f for any continuous function f. However, the related asymptotic rates of convergence are very different since, for instance, given a regular function $f \in C^k_{2\pi}$, $k \geq 1$, we have $\|C_n(f) - f\|_\infty \leq c_1 n^{-1}$ and $\|DL_n(f) - f\|_\infty \leq c_2 n^{-k} \omega_{f^{(k)}}(n^{-1})$, with $\omega_h(\cdot)$ denoting the modulus of continuity of h [63]. So there exists a huge gap in the approximation properties between $C_n(f)$ and $DL_n(f)$ as is well known in the field of the approximation theory [19, 37, 40, 41, 63].

Nevertheless, the circulant preconditioners constructed from these two operators lead to PCG methods which are really equivalent as stressed by the numerical experiments performed in [17].

In this paper, under weak and feasible assumptions on the properties of a given sequence of trigonometric algebras $\mathcal{A} = \{\mathcal{A}_n\}_n$, first we prove that a sequence of matrices $\alpha_n \in \mathcal{A}_n$ is a good approximation of $A_n(f)$, in the matrix sense defined in [52] (see section 3), if and only if α_n has eigenvalues defined by a polynomial operator $\mathcal{S}_n$ which approximates f in some sense. In other words, the main fact is that the *superlinearity* of the PCG method with α_n as $n \times n$ preconditioner is equivalent to saying that $\mathcal{S}_n - f$ is "small" on some suitable subset of the fundamental interval $[-\pi, \pi]$. Therefore, the assumption "$\mathcal{S}_n$ converges to f" is a necessary ingredient to devise a superlinear PCG method, but, unfortunately and surprisingly, if f is positive then we show that there exists a property of insensitivity to the goodness of the approximation process and we show that this is something in common and inherent to all the preconditioners based on trigonometric matrix algebras.

To overcome this problem, that is, to exploit the convergence features of $\mathcal{S}_n$, we propose the use of band Toeplitz preconditioners with weakly increasing bandwidth and associated with $\mathcal{S}_n$. The theoretical analysis and the numerical experiments [47] prove that the band Toeplitz approach is very sensitive to the convergence properties of the considered approximation process. Therefore, as n goes to infinity, the number of PCG iterations collapses to 1, while in the case of the matrix algebra approach this number goes to a constant value which depends on the required accuracy, on the symbol f, and on the specific matrix algebras, but not on the approximation process $\mathcal{S}_n$. In conclusion, the unexpected practical result of this analysis is that the band Toeplitz PCG methods perform better than the matrix algebra PCG methods even when the generating function of the considered system is continuous and positive.

In the ill-conditioned case the superiority of the band Toeplitz (BT) approach is more evident. First we remark that, when f is nonnegative and has isolated zeros, the results concerning the matrix algebra approach are in the opposite direction with respect to the positive case. The better the approximation process the better the quality of the cluster around the unity of the preconditioned matrices [57, 25]. However, the cluster is always weak (except for rare exceptions [57, 25, 53]). On the other side, the BT approach is better because in the case of zeros of finite order it is possible to construct linear and superlinear preconditioners [7, 27, 13, 14, 44, 54, 47, 48].

This paper is organized in the following way: in the second section we introduce the most interesting trigonometric matrix algebras and we introduce some classical approximation schemes which lead to general matrix algebra preconditioners generalizing in various senses those of R. Chan and Yeung. Then, by using the same approximation processes [28], we introduce some new BT preconditioners. In section 3 we analyze the convergence features of these PCG methods by imposing as a significant constraint that the cost per iteration of the proposed procedures must be upperbounded by $O(n \log n)$ operations. This goal is natural in the case of trigonometric matrix algebra preconditioners [12, 3, 4] via the fast Fourier/trigonometric transform (FFT/FTT) [60, 39, 35], while it implies some conditions on the *growth of the bandwidth* $l(n)$ as a function of the dimension n of the preconditioners when a BT technique is used: the asymptotic behavior of the quantity $l(n)$ is the key ingredient to prove the superlinearity of the PCG methods based on BT preconditioners. A final section 4 of numerical experiments and of conclusive remarks ends the paper.

2. The preconditioners from linear approximation schemes.

In this section, first, we give a formal definition of trigonometric matrix algebras and we introduce some of them. Second, we define the concept of approximation of Toeplitz matrices related to the function f in a given algebra depending on some fixed approximation process of the symbol f itself.

2.1. Trigonometric matrix algebras.

Let U be a unitary complex $n \times n$ matrix. Then by $\mathcal{M}(U)$ we denote the commutative algebra of all the matrices simultaneously diagonalized by the U transform, that is,

$$\mathcal{M}(U) = \{A = U\Delta U^* : \ \Delta \ \text{ is a diagonal complex matrix}\}.$$

Here the symbol $*$ means transpose and conjugate. Now we define a special subset of matrix algebras that we call trigonometric matrix algebras and we denote them by $\mathcal{A}_T$. Let $\{v_j\}_{j\in N}$ be a sequence of trigonometric functions on an interval I and $\mathcal{W} = \{\mathcal{W}_n\}_n$ be a sequence of grids of n points on I, namely, $\mathcal{W}_n = \{x_i^{(n)}, i = 0, \ldots, n-1\}$. Let us suppose that the generalized Vandermonde matrix $V = k_n(v_j(x_i^{(n)}))_{i,j=0}^{n-1}$, where k_n is a suitable scaling factor, is a unitary matrix. Then, an algebra of the form $\mathcal{M}(U)$ is a trigonometric algebra if $U = V^*$ with V being a generalized trigonometric Vandermonde matrix.

Examples of such algebras are the circulant $\mathcal{C}$, the τ, the Hartley $\mathcal{H}$, for which the matrices U [4] are

$$\begin{aligned} U = F \quad &= \quad \left(\frac{1}{\sqrt{n}} e^{\mathbf{i} j x_i^{(n)}}\right), \quad i,j = 0,\ldots,n-1, \\ &\mathcal{W}_n = \left\{x_i^{(n)} = \frac{2i\pi}{n} : i = 0,\ldots,n-1\right\} \subset I = [-\pi,\pi], \end{aligned}$$

$$\begin{aligned}
U = S \quad &= \quad \left(\sqrt{\frac{2}{n+1}}\sin((j+1)x_i^{(n)})\right), \quad i,j=0,\dots,n-1,\\
&\mathcal{W}_n = \left\{x_i^{(n)} = \frac{(i+1)\pi}{n+1} : i=0,\dots,n-1\right\} \subset I = [0,\pi],\\
U = H \quad &= \quad \left(\frac{1}{\sqrt{n}}\left[\sin(jx_i^{(n)}) + \cos(jx_i^{(n)})\right]\right), \quad i,j=0,\dots,n-1,\\
&\mathcal{W}_n = \left\{x_i^{(n)} = \frac{2i\pi}{n} : i=0,\dots,n-1\right\} \subset I = [-\pi,\pi],
\end{aligned}$$

respectively. For the class of the ϵ-circulants [22] we may consider the case of ϵ on the unit complex circle because it assures that the matrix U continues to be unitary. More precisely, if $\epsilon = e^{\mathbf{i}2\pi\psi}$ then the matrix U has the following representation:

$$\begin{aligned}
U = F_\epsilon \quad &= \quad \left(\tfrac{1}{\sqrt{n}}e^{\mathbf{i}(j+\psi)x_i^{(n)}}\right), \quad i,j=0,\dots,n-1,\\
&\mathcal{W}_n = \left\{x_i^{(n)} = \tfrac{2i\pi}{n} : i=0,\dots,n-1\right\} \subset I = [-\pi,\pi].
\end{aligned}$$

Notice that all these algebras are trigonometric with equispaced meshes.

2.2. Approximation schemes and matrix algebra preconditioners. We introduce a sequence of finite dimensional spaces V_n such that $V_n \subset V_{n+1}$ and $\bigcup V_n$ is dense in the space $C_{2\pi}$ of the continuous 2π periodic functions equipped with the supremum norm. The correspondence between functions and sequences of nested Toeplitz matrices defined by Fourier coefficients suggests to us that a natural, but not compulsory, choice of the spaces V_n is given by the trigonometric polynomials of degree at most n. When we consider symmetric Toeplitz matrices, the associated generating function is also even and therefore, in this case, it is natural to define V_n as the space of even trigonometric polynomials. Let us define $\mathcal{S}_n : C_{2\pi} \to V_n$ as a sequence of linear approximation processes [41, 28, 20, 21] and let $\mathcal{A}_n \in \mathcal{A}_T$ be a fixed trigonometric matrix algebra of size n. With these notations we indicate by $\mathcal{A}_n(\mathcal{S}_n(f))$ the matrix belonging to the algebra $\mathcal{A}_n$ such that

$$\mathcal{A}_n(\mathcal{S}_n(f)) = UDU^*, \qquad D_{i,i} = \mathcal{S}_n(f)(x_i^{(n)}),$$

where $x_i^{(n)}$ are the grid points of the algebra and U is the unitary transformation related to the matrix algebra $\mathcal{A}_n$. In the literature, for all the most commonly known and used algebras, the following approximation processes $\mathcal{S}_n$ have been considered.

- The Fourier polynomial $F_n(f) = \sum_{j=-n}^{n} a_j e^{\mathbf{i}jx}$ which converges uniformly to f when f belongs to the Dini–Lipschitz class [53, 63]. (For the definition of the coefficients a_k see (1.1).) On the other hand, this is a quasi-optimal approximation procedure since the related Lebesgue constant is asymptotically equal to a constant times $\log n$ [63].
- The Cesàro sum $C_n(f) = (n+1)^{-1}\sum_{j=0}^{n} F_j(f)$ uniformly converging to f for any continuous 2π periodic function. The corresponding low rate of convergence is such that the related approximation error is asymptotical to n^{-1} (the Voronowskaja result [41]), even for very regular functions.
- The De La Vallée Poussin operator $DL_n(f) = 2C_n(f) - C_{n/2}(f)$ which leads to the error (up to a small positive constant) of the polynomial of best approximation [63].

2.3. The case of the optimal Frobenius approximation. Let us define the operator $\mathcal{P}_U$ defined on $\mathbf{C}^{n\times n}$ and taking values in $\mathcal{M}(U)$ where both the Banach spaces are equipped with the Frobenius norm $\|X\|_F^2 = \sum_{i,j} |x_{i,j}|^2$. Then

$$\mathcal{P}_U(A) = \arg \min_{X\in\mathcal{M}(U)} \|A - X\|_F$$

where the minimum exists and is unique since $\mathcal{M}(U)$ is a linear finite dimensional space. By means of simple algebraic arguments, it is possible to prove the following lemma (see also [10, 9] for the circulant case).

LEMMA 2.1 ([26, 8]). *With $A, B \in \mathbf{C}^{n\times n}$ and the previous definition of $\mathcal{P}_U$, we have*

1. $\mathcal{P}_U(A) = U^*\sigma(UAU^*)U$, *with $\sigma(X)$ the diagonal matrix having $(X)_{i,i}$ as diagonal elements,*
2. $\mathcal{P}_U(\alpha A + \beta B) = \alpha\mathcal{P}_U(A) + \beta\mathcal{P}_U(B)$ *and $\alpha, \beta \in \mathbf{C}$,*
3. $\mathcal{P}_U(A^*) = (\mathcal{P}_U(A))^*$,
4. trace$(\mathcal{P}_U(A))$ =trace(A),
5. $\|\mathcal{P}_U\| = 1$ *with $\|\cdot\|$ the dual* 2 *norm,*
6. $\|\mathcal{P}_U\| = 1$ *with $\|\cdot\|$ the dual F norm,*
7. $\|A - \mathcal{P}_U(A)\|_F^2 = \|A\|_F^2 - \|\mathcal{P}_U(A)\|_F^2$.

It is also interesting to note the following result proved and used for specific matrix algebras [10, 9, 15, 38, 53] but extended in an abstract way in [26].

LEMMA 2.2 ([26]). *If A is Hermitian ($A = A^*$), then the eigenvalues of $\mathcal{P}_U(A)$ are contained in the closed real interval $[\lambda_1(A), \lambda_n(A)]$ where $\lambda_j(A)$ are the eigenvalues of A ordered in a nondecreasing way. Moreover, when A is positive definite $\mathcal{P}_U(A)$ is also positive definite.*

We notice that the preceding operator among matrices has a correspondence in terms of approximation processes. Actually we find that [4, 52] $\mathcal{P}_U(A_n(f)) = \mathcal{A}_n(\mathcal{S}_n(f))$, with $\mathcal{A}_n = \mathcal{M}(U)$. In particular the following relationships hold.

- If $\mathcal{A} = \{\mathcal{A}_n\}_n$ is the circulant class, then the operator $\mathcal{S}_n$ is the n-degree Cesàro sum $C_n(f)$ [17].
- If $\mathcal{A} = \{\mathcal{A}_n\}_n$ is the τ class, then the symbol $\mathcal{S}_n$ denotes the n-degree Cesàro sum plus another infinitesimal term involving the Chebyshev polynomials of the second kind. More precisely, $\mathcal{S}_n(f) = (C_n(f))(x) + \frac{\cos(x)}{n+1}\sum_{k+1} a_k U_k(\cos(x))$ where U_k is the kth Chebyshev polynomial of second kind [52].
- If $\mathcal{A} = \{\mathcal{A}_n\}_n$ is the Hartley class, then $\mathcal{S}_n$ is the n-degree Cesàro sum plus an infinitesimal correction term involving trigonometric sums. A detailed expression of $\mathcal{S}_n(f)$ is $(C_n(f))(x) - \frac{2}{n+1}\sum_k a_k \sin(kx)$ [4].

2.4. Approximation schemes and BT preconditioners. In this subsection we are concerned with the construction of BT preconditioners associated with approximation schemes. Let us consider the nondecreasing sequence $l(n) < n$ and the approximation processes $\mathcal{S}_n$ defined in the preceding subsections; then the BT preconditioner takes the following form: $A_n(\mathcal{S}_{l(n)}(f))$.

In section 3 we study the convergence speed of the PCG applied to the considered kind of Toeplitz linear systems in terms of the generating functions f and $g = \mathcal{S}_{l(n)}(f)$.

Here we recall some known definitions and results to be used in the next section.

DEFINITION 2.3. *Let h be a measurable function. The essential range of the function h defined on I is the set of all y real numbers for which, $\forall\epsilon > 0$, the Lebesgue measure of $\{x \in I : h(x) \in (y - \epsilon, y + \epsilon)\}$ is positive* [45]. *The function h is called "sparsely vanishing" if the Lebesgue measure of $\{x \in I : h(x) = 0\}$ is zero* [57, 25].

THEOREM 2.4 ([34]). *Let m_f and M_f be the* essinf *and the* esssup *of f in $[-\pi,\pi]$. If $m_f < M_f$, then, $\forall n > 0$, we have*

$$m_f < \lambda_i(A_n(f)) < M_f,$$

where $\lambda_i(X)$ is the ith eigenvalue of X arranged in nondecreasing order. If $m_f \geq 0$, then $A_n(f)$ is positive definite.

THEOREM 2.5. *Let $f, g \in L^1[-\pi,\pi]$ be two functions essentially nonnegative, i.e., $m_f, m_g \geq 0$. The matrices $A_n(f), A_n(g)$ are positive definite (see Theorem* (2.4)*) and the eigenvalues $\lambda_i^{(n)}$ of $A_n^{-1}(g)A_n(f)$ arranged in nondecreasing order are such that*

1. *$\lambda_i^{(n)} \in (r, R)$, r, R being the* essinf *and the* esssup *of f/g, respectively* [27, 7]*;*
2. *if g is sparsely vanishing, then $\bigcup_{n\in\mathbf{N}} \bigcup_{i\leq n} \lambda_i^{(n)}$ is dense in the "essential range" $\mathcal{ER}(f/g)$ of f/g* [48, 51] *and moreover the eigenvalues $\{\{\lambda_i^{(n)}\}_{i\leq n}\}_n$ distribute as f/g* [55]*;*
3. $\lim_{n\to\infty} \lambda_1^{(n)} = r$, $\lim_{n\to\infty} \lambda_n^{(n)} = R$ [14, 51].

Notice that Theorem 2.5 suggests a "global property" of distribution of the eigenvalues. For instance, a consequence of the second part of this result is that for any nonnegative integer k fixed with respect to the dimension n we find the following limit relations:

$$\lim_{n\to\infty} \lambda_k^{(n)} = r, \quad \lim_{n\to\infty} \lambda_{n-k}^{(n)} = R.$$

Finally, owing to the sophisticated results [1] concerning the convergence speed of PCG algorithms, we stress that the knowledge of the general distribution of the preconditioned spectra [50, 55] is useful in order to understand very precisely the convergence rates of PCG methods based on Toeplitz preconditioners.

3. The analysis of convergence.

In the following we focus our attention on the convergence analysis of several PCG methods applied to systems of the form

$$A_n(f)\mathbf{x} = \mathbf{b},$$

where f is positive and continuous. Let us start with the following definitions.

DEFINITION 3.1. *Given a sequence $\mathcal{A} = \{\mathcal{A}_n\}_n$ of $n \times n$ algebras $\mathcal{A}_n \in \mathcal{A}_T$, we say that "$\{\mathcal{A}_n(\mathcal{S}_n(f))\}$ (strongly) converges to $\{A_n(f)\}$" if, for any $\epsilon > 0$, there exists $\bar{n}$ such that, for $n \geq \bar{n}$, $A_n(f) - \mathcal{A}_n(\mathcal{S}_n(f))$ has eigenvalues in $(-\epsilon, \epsilon)$ except for a constant number Q_ϵ of outliers.*

DEFINITION 3.2. *Given a sequence $\mathcal{A} = \{\mathcal{A}_n\}_n$ of $n\times n$ algebras $\mathcal{A}_n \in \mathcal{A}_T$, we say that "$\{\mathcal{A}_n(\mathcal{S}_n(f))\}$ weakly converges to $\{A_n(f)\}$" if, for any $\epsilon > 0$, there exists $\bar{n}$ such that, for $n \geq \bar{n}$, $A_n(f) - \mathcal{A}_n(\mathcal{S}_n(f))$ has eigenvalues in $(-\epsilon, \epsilon)$ except for $Q_\epsilon = o(n)$ outliers.*

When the number $\sup_\epsilon Q_\epsilon$ is a finite number we say that the convergence is also uniform. The following result due to Tyrtyshnikov gives a criterion for establishing whether convergence occurs.

LEMMA 3.3 ([59]). *Let $\{A_n\}$ and $\{B_n\}$ be two sequences of $n \times n$ Hermitian matrices. If $\|A_n - B_n\|_F^2 = o(n)$, then the convergence is weak. In particular, when $\|A_n - B_n\|_F^2 = O(1)$, then $\{A_n\}$ and $\{B_n\}$ converge in the strong sense.*

Here we want to establish whether a given sequence of algebras is good or not for approximating Toeplitz matrices. We consider a criterion regarding BT matrices.

DEFINITION 3.4. *$\mathcal{A} = \{\mathcal{A}_n\}_n$ is a good sequence of algebras if and only if $\{\mathcal{A}_n(p)\}$ converges to $\{A_n(p)\}$ for any trigonometric polynomial p of fixed degree.*

Observe that circulant, τ, and Hartley matrix algebras are good algebras. (See also Proposition 3.10 in subsection 3.2.)

THEOREM 3.5. *Let f be a continuous periodic function and $\{\mathcal{S}_n(f)\}$ be a sequence of approximation processes. We suppose that $\forall\epsilon > 0$, $\exists\bar{n}_{\epsilon,\mathcal{S}_n}$ such that, for any $n \geq \bar{n}_{\epsilon,\mathcal{S}_n}$,*

$$\left|\mathcal{S}_n(f)(x_i^{(n)}) - f(x_i^{(n)})\right| < \epsilon, \quad x_i^{(n)} \in \mathcal{W}_n(\mathcal{A}),$$

$\forall i \in \{1,\ldots,n\}/J_\epsilon$, with J_ϵ set of indices of cardinality constant with regard to n. Then, under the assumption that $\mathcal{A} = \{\mathcal{A}_n\}_n$ is a good sequence of algebras, $\{\mathcal{A}_n(\mathcal{S}_n(f))\}$ converges to $\{A_n(f)\}$.

Proof. Let p_k be the polynomial having degree k of best approximation of f in supremum norm [37]. Fix the integer M such that $\|f - p_M\|_\infty < \epsilon/3$. Then, by using the Szegö theorem (see [34, p. 64]) and the assumptions, we have $\|A_n(f) - A_n(p_M)\|_2 < \epsilon/3$, $\|\mathcal{A}_n(\mathcal{Z}_n(f) - p_M)\|_2 < \epsilon/3$, where $\mathcal{Z}_n(f)(x_i^{(n)}) = \mathcal{S}_n(f)(x_i^{(n)})$ if i is not in J_ϵ and equals $f(x_i^{(n)})$ otherwise. Therefore, from the identity

$$\begin{aligned} A_n(f) - \mathcal{A}_n(\mathcal{S}_n(f)) \quad = \quad & A_n(f) - A_n(p_M) - (\mathcal{A}_n(\mathcal{Z}_n(f)) - \mathcal{A}_n(p_M)) \\ & + A_n(p_M) - \mathcal{A}_n(p_M) + (\mathcal{A}_n(\mathcal{Z}_n(f)) - \mathcal{A}_n(\mathcal{S}_n(f))) \end{aligned}$$

we have that, except for a term of norm bounded by $2\epsilon/3$, the difference $A_n(f) - \mathcal{A}_n(\mathcal{S}_n(f))$ coincides with $A_n(p_M) - \mathcal{A}_n(p_M)$ plus $\mathcal{A}_n(\mathcal{Z}_n(f)) - \mathcal{A}_n(\mathcal{S}_n(f))$. Since $\mathcal{A}$ is a good sequence of algebras, we may split each Hermitian matrix $A_n(p_M) - \mathcal{A}_n(p_M)$ into two parts. The first one has a norm bounded by $\epsilon/3$, the second one has constant rank. Moreover, from the definition of $\mathcal{Z}_n(f)$, it follows that $\mathcal{A}_n(\mathcal{Z}_n(f)) - \mathcal{A}_n(\mathcal{S}_n(f))$ has rank at most equal to $\#(J_\epsilon)$. Therefore, by invoking the Cauchy interlace theorem [62], the claimed result is obtained. ◻

The following corollary is useful to derive and analyze good preconditioners for the conjugate gradient method.

COROLLARY 3.6. *Under the assumption of Theorem 3.5, if f and $\mathcal{S}_n(f)$ are positive, then we have that for any $\epsilon > 0$, for n large enough, the matrices of the sequence*

$$\left\{(\mathcal{A}_n(\mathcal{S}_n(f)))^{-1} A_n(f)\right\}$$

have eigenvalues in $(1-\epsilon, 1+\epsilon)$ except, at most, $Q_\epsilon = O(1)$ outliers.

Now we want to analyze the asymptotical behavior of the number of iterations N_η to reach the solution within an accuracy η. We prove that, for n large enough, this number of iterations is not sensitive to the precision of the approximation scheme $\mathcal{S}_n(f)$. First we summarize some results in the following theorem.

THEOREM 3.7. *We suppose that $\forall\epsilon > 0$ $\exists\bar{n}_{\epsilon,\mathcal{S}_n}$ such that, for any $n \geq \bar{n}_{\epsilon,\mathcal{S}_n}$,*

$$\left|\mathcal{S}_n(f)(x_i^{(n)}) - f(x_i^{(n)})\right| < \epsilon, \quad x_i^{(n)} \in \mathcal{W}_n(\mathcal{A}),$$

for all $i \in \{1,\ldots,n\}/J_\epsilon$, with J_ϵ set of indices of cardinality constant with respect to n. In addition, if $\mathcal{A} = \{\mathcal{A}_n\}_n$, with $\mathcal{A}_n \in \mathcal{A}_T$, is a good sequence of algebras, then $\forall\epsilon > 0$, there exists $\bar{n}_{\epsilon,\mathcal{S}_n}$ such that for any $n \geq \bar{n}_{\epsilon,\mathcal{S}_n}$ we have

$$A_n(f) - \mathcal{A}_n(\mathcal{S}_n(f)) = LR_{M_\epsilon} + LN_\epsilon,$$

where $\|LN_\epsilon\|_2 \leq \epsilon$ and LR_{M_ϵ} is a term having rank $r(M_\epsilon)$ depending only on ϵ and on the algebra.

Proof. It is sufficient to recall Theorem 3.5. □

Let us fix $\eta > 0$ and let us call N_η the number of iterations of the PCG method such that the relative error computed with regard to the residuals is less than η:

$$\frac{\|\mathbf{r}_{N_\eta}\|}{\|\mathbf{r}_0\|} < \eta.$$

We notice that we can find $\epsilon = \epsilon(\eta)$ small enough so that after $r(M_\epsilon)$ iterations we observe superlinear convergence [12, 3, 16]. So the greatest part of iterations is given by $r(M_\epsilon)$, which depends on how well the given sequence $\mathcal{A}$ of algebras approximates the BT matrices. Actually, we may always find ϵ such that the number of iterations is bounded by $r(M_\epsilon) + 1$, $\epsilon = \epsilon(\eta)$ (see also Theorem 3.8).

In conclusion the role of the approximation scheme $\mathcal{S}_n$ is marginal and is related to the dimension $\bar{n}$ after which we may appreciate a stabilized asymptotical behavior. Actually, the number of iterations $r(M_\epsilon) + 1$, (with $\epsilon = \epsilon(\eta)$), is dependent on the precision η, on the regularity of the function f (the number M_ϵ), and on the goodness of the algebra (the function $r(\cdot)$). Finally notice that $N_\eta = Q_{\epsilon(\eta)} + 1$, where Q_ϵ is the number of outliers mentioned in Definition 3.1.

At this point we have proved that if $\mathcal{S}_n(f)$ converges to f on $\mathcal{W}(\mathcal{A}) = \{\mathcal{W}_n(\mathcal{A})\}$, then, under feasible assumptions on $\{\mathcal{A}_n(p_M)\}$, M being a fixed integer number, we have the matrix convergence of $\{\mathcal{A}_n(\mathcal{S}_n(f))\}$ to $\{A_n(f)\}$. Now we consider the converse problem. Let $\mathcal{P}_n \in \mathcal{A}_n$ be such that $\{\mathcal{P}_n\}$ converges to $\{A_n(f)\}$ in the matrix sense. The question is: is it true that $\mathcal{P}_n = \mathcal{A}_n(\mathcal{S}_n(f))$ for some approximation scheme $\mathcal{S}_n(f)$ for f? In other words, we demonstrated that operator convergence implies matrix convergence. Is the converse true? Surprisingly enough, the answer is positive under the usual assumption that $\{\mathcal{A}_n(p_M)\}$ converges to $\{A_n(p_M)\}$ for any polynomial p_M of fixed degree independent of n.

THEOREM 3.8. *Let $\{\mathcal{P}_n\}$ be a sequence of positive definite matrices of $\mathcal{A}_n$ converging to $\{A_n(f)\}$ in the strong sense. Let us suppose that $\mathcal{A}$ is a good sequence of algebras. Then $\mathcal{P}_n = \mathcal{A}_n(\mathcal{S}_n(f))$ where $\mathcal{S}_n(f)$ is an approximation scheme for f "almost everywhere" on $\mathcal{W}(\mathcal{A}) = \{\mathcal{W}_n(\mathcal{A})\}$, i.e., $\forall\epsilon > 0$, $\exists\bar{n}_{\epsilon,\mathcal{S}_n}$ such that, for any $n \geq \bar{n}_{\epsilon,\mathcal{S}_n}$,*

$$\left|\mathcal{S}_n(f)(x_i^{(n)}) - f(x_i^{(n)})\right| < \epsilon, \quad x_i^{(n)} \in \mathcal{W}_n(\mathcal{A}),$$

$\forall i \in \{1, \dots, n\}/J_\epsilon$, with J_ϵ set of indices of cardinality constant with respect to n.

Proof. By definition of strong convergence it follows that, $\forall\epsilon > 0$, there exists an integer $\bar{n}$ such that, for any $n \geq \bar{n}$, we find

$$\mathcal{P}_n - A_n(f) = LR_\epsilon + \epsilon_n, \tag{3.1}$$

where $\|\epsilon_n\|_2 < \epsilon$ and LR_ϵ is a term having constant rank depending on ϵ and $\mathcal{A}$ but not on n. By the Weierstrass theorem, we may choose a constant M such that $f - p_M$ has supremum norm smaller than ϵ, p_M being a trigonometric polynomial. From this we may write

$$\mathcal{P}_n - A_n(f) = \mathcal{P}_n - \mathcal{A}_n(p_M) + T_1 + T_2,$$

where $T_1 = \mathcal{A}_n(p_M) - A_n(p_M)$ and $T_2 = A_n(p_M) - A_n(f)$. Moreover, by the assumption of the *goodness* of the algebras $\mathcal{A}$, we have that T_1 is the sum of a term of norm smaller than ϵ and a term of constant rank, while T_2 has norm strictly less than ϵ (see Theorem 3.5). In conclusion, by recalling (3.1), we find

$$\mathcal{P}_n - \mathcal{A}_n(p_M) = UDU^* = LR_\epsilon + LN_\epsilon.$$

Finally the elements $D_{i,i} = \lambda_i(\mathcal{P}_n) - p_M(x_i^{(n)})$ have to be almost all infinitesimal, that is $\lambda_i(\mathcal{P}_n) = \mathcal{S}_n(f)(x_i^{(n)})$ with $\mathcal{S}_n(f)$ converging to f almost everywhere on the grid points of the algebras. $\square$

By referring to the optimal approximation in Frobenius's norm, we noticed that [52] $\mathcal{P}_U(A_n(f))$ can be expressed as $\mathcal{A}_n(\mathcal{S}_n(f))$ with $\mathcal{S}_n(f)$ converging to f in the case where $\mathcal{A}$ is $\mathcal{H}$, τ or $\mathcal{C}$. Now, with the help of Theorem 3.8, we can prove that this is a general statement holding for any algebra which is good in the sense of Definition 3.4: in [52] we proved that $\{\mathcal{P}_U(A_n(f))\}$ converges to $\{A_n(f)\}$ if $\mathcal{A} = \{\mathcal{A}_n\}_n$ is a good sequence of algebra. So, if $\mathcal{A}$ is good, then $\{\mathcal{P}_U(A_n(f))\}$ converges to $\{A_n(f)\}$ and therefore, in light of Theorem 3.8, $\mathcal{P}_U(A_n(f))$ is representable as $\mathcal{A}_n(\mathcal{S}_n(f))$ where $\mathcal{S}_n(f)$ converges to f at least on $\mathcal{W}_n(\mathcal{A})$ with, at most, the exception of a constant number of grid points.

Finally, it should be pointed out that we have similar results when the concept of weak convergence is considered (see [52] for the optimal Frobenius approximation).

3.1. Some specific algebras. We start by analyzing circulant, τ, and Hartley algebras from the point of view of the generating functions and of the "goodness" of these algebras with regard to Definition 3.4.

3.1.1. The circulant algebra. It is well known [22] that the set of the circulant matrices is generated by the cyclic matrix

$$Z = \begin{pmatrix} 0 & \dots & 0 & 1 \\ 1 & & & 0 \\ & \ddots & & \vdots \\ 0 & & 1 & 0 \end{pmatrix}$$

in the sense that each matrix of the algebra can be written as a polynomial of Z. Moreover, the eigenvalues of Z are the nth complex roots of the unity and, therefore, by exploiting the structure of the powers of Z it is trivial to deduce that the ith eigenvalue of a circulant matrix C, having the vector $(c_0, c_1, \dots, c_{n-1})$ as first column, is given by $p(\omega_i)$ where

$$p(\omega) = \sum_{k=0}^{n-1} c_k \omega^k, \quad \omega \in \mathbf{C}$$

and $\omega_i^{(n)} = e^{\mathbf{i}\frac{2\pi i}{n}}$. Since we are considering symmetric Toeplitz matrices, we restrict our attention to the subalgebra of the symmetric circulant matrices. In this case the symmetry imposes that $c_j = c_{n-j} \in \mathbf{R}$, $j = 1, \dots, n-1$. As a consequence, we may rewrite the eigenvalue function as cosine sum

$$c(x) = c_0 + 2\sum_{k=0}^{m} c_k \cos(kx), \quad x \in [0, 2\pi],$$

with $n = 2m$ and the ith eigenvalue as $c(x_i^{(n)})$, $x_i^{(n)} = \arg\, \omega_i^{(n)} \in \mathcal{W}(\mathcal{C})$. Given a symmetric Toeplitz matrix $A_n(f)$, the natural circulant approximation [12] related to the choice of $c_j = a_j$, $j = 0, \dots, m$, while the optimal approximation with respect to the Frobenius norm is given by

$$c_j = \frac{ja_{n-j} + (n-j)a_j}{n}.$$

By manipulating these relationships and by means of the definition of $c(x)$ or $p(\omega)$, it is easy to prove that the eigenvalue function of the natural approximation is the Fourier sum of order m, while the eigenvalue function of the optimal approximation is the Cesàro sum of order n [17].

3.1.2. The τ algebra. This inherently symmetric matrix algebra admits a Toeplitz generator given by

$$W = \begin{pmatrix} 0 & 1 & & \\ 1 & 0 & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & 0 \end{pmatrix}.$$

Also in this case we have an analytic expression of the ith eigenvalue [2] of the generator given by $2\cos(x_i^{(n)})$, where $x_i^{(n)} = \frac{i\pi}{n+1} \in \mathcal{W}(\tau)$, $i = 1, \dots, n$.

An interesting representation of any matrix belonging to the τ class can be given in terms of an associated Toeplitz matrix: more precisely given a τ matrix T, there exists a symmetric Toeplitz matrix A_n

$$A_n = \begin{pmatrix} t_0 & t_1 & \dots & t_{n-1} \\ t_1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & t_1 \\ t_{n-1} & \dots & t_1 & t_0 \end{pmatrix} \tag{3.2}$$

and a Hankel matrix

$$\mathrm{HC}(A_n) = \begin{pmatrix} t_2 & \dots & t_{n-1} & 0 & 0 \\ \vdots & \iddots & & \iddots & 0 \\ t_{n-1} & & \iddots & & t_{n-1} \\ 0 & \iddots & & \iddots & \vdots \\ 0 & 0 & t_{n-1} & \dots & t_2 \end{pmatrix}$$

such that

$$T = A_n - \mathrm{HC}(A_n). \tag{3.3}$$

When dealing with Toeplitz problems, the Hankel correction yields a preconditioner that works fine, especially when A_n is banded since A_n is obtained through a low rank correction. It follows that the matrix $T^{-1}A_n$ has only a constant number (with respect to the dimension n) of eigenvalues different from 1.

This Toeplitz + Hankel representation is also important from a spectral point of view: actually, it can be proved [2] that the ith eigenvalue of T can be written as

$$t(x) = t_0 + 2\sum_{k=0}^{n} t_k \cos(kx), \quad x \in [0, \pi],$$

with $x = x_i^{(n)} \in \mathcal{W}(\tau)$.

Now, we consider a complementary point of view. Given a symmetric Toeplitz matrix $A_n(f)$, the natural τ approximation [3] is related to the choice of $t_j = a_j$, $j = 0, \ldots, n$, while the optimal approximation in Frobenius norm is given by

$$t_j = \frac{ja_{n-j} + (n-j)a_j}{n} + O\left(n^{-1}\right).$$

More precisely we have that the first column of this τ matrix is an n-dimensional vector ϕ with $\phi_1 = a_0 - \frac{n-2}{n+1}a_2$, $\phi_2 = a_1 - \frac{n-3}{n+1}a_3$, $\phi_i = \frac{n-i+3}{n+1}a_{i-1} - \frac{n-i-1}{n+1}a_{i+1}$, $i = 3, \ldots, n-2$, $\phi_{n-1} = \frac{4}{n+1}a_{n-2}$ and $\phi_n = \frac{3}{n+1}a_{n-1}$.

By manipulating these relationships and by means of the definition of $t(x)$, it is easy to prove that the eigenvalue function of the natural approximation is the Fourier sum of order n, while the eigenvalue function of the optimal approximation is the Cesàro sum of order n plus a linear combination of Chebyshev polynomials of the second kind (see [25] and the end of section 2.3).

3.2. The Hartley class. The Hartley class does not have a generator, but it can be described by using circulant matrices. Actually any matrix X belonging to the algebra $\mathcal{H}$ can be expressed as the sum of two independent matrices, the first being symmetric and circulant C, the second being the product of a special permutation matrix J by a skewcirculant matrix B. More precisely, $J_{1,1} = J_{s,n+2-s} = 1$, $s = 2, \ldots, n$, and

$$X = C + JB,$$

where the first column of C has coefficients c_j such that $c_j = c_{n-j} \in \mathbf{R}$, $j = 1, \ldots, n-1$, and the first column of B has coefficients $b_j = -b_{n-j} \in \mathbf{R}$, $j = 1, \ldots, n-1$, $b_0 = 0$.

By performing simple computations [4], we find that the ith eigenvalue of X is given by

$$h(x) = \sum_{k=0}^{n} h_k(\cos(kx) + \sin(kx)), \tag{3.4}$$

where $h_k = c_k + b_k$ and $x = x_i^{(n)} \in \mathcal{W}(\mathcal{H})$. By exploiting the symmetry of the coefficients c_k we observe that $\sum_{k=0}^{n} c_k \sin(kx_i^{(n)}) = 0$, while the antisymmetry of the coefficients b_k implies $\sum_{k=0}^{n} b_k \cos(kx_i^{(n)}) = 0$. In conclusion we obtain this more appealing expression of the eigenvalue function:

$$h(x) = \sum_{k=0}^{n} c_k \cos(kx) + b_k \sin(kx).$$

This representation of $h(x)$ is much more natural since $h(x)$ is split into two parts. The first is given by a cosines polynomial $c(x) = \sum_{k=0}^{n} c_k \cos(kx)$ which is related only to the circulant part of the Hartley matrix. The second one is given by a sines polynomial $s(x) = \sum_{k=0}^{n} b_k \sin(kx)$ associated with the Hankel contribution. Because we are interested in the approximation of a symmetric Toeplitz matrix generated by an even function, it is evident that the Hankel contribution and, consequently, the even part of the generating function $h(x)$, have to be kept small. This fact is proved in the following theorem.

THEOREM 3.9. *The weak matrix convergence of a Hartley sequence of matrices H to the Toeplitz sequence $\{A_n(f)\}$ implies that $\|JB\|_F^2 = o(n)$ or $\|s(x)\|_{L^2}^2 = o(1)$.*

The strong matrix convergence of a Hartley matrix sequence to the Toeplitz sequence $\{A_n(f)\}$ *implies that* $\|JB\|_F^2 = O(1)$ *or* $\|s(x)\|_{L^2}^2 = O(n^{-1})$.

Proof. We start by noticing that $d = \|A_n(f) - H\|_F \geq \inf_Y \|Y - JB\|_F$, where the infimum is calculated over the Y's belonging to the space of the symmetric Toeplitz matrices and where B is the skew circulant component of the Hartley matrix H. Therefore we have

$$d^2 \geq \inf_Y \sum_k \sum_{i+j-2=k} |Y_{i,j} - b_k|^2$$

with $b_k = -b_{n-k}$, $k = 1, \dots, n-1$, $b_0 = 0$. Let us define the following sets of indices:

$$\begin{aligned}
J_k^+ &= \{j \in \{1-n, \dots, n-1\} : \ j \text{ is even and } \exists b_k \text{ on the diagonal } j\};\\
J_k^- &= \{j \in \{1-n, \dots, n-1\} : \ j \text{ is even and } \exists -b_k \text{ on the diagonal } j\};\\
I_k^+ &= \{j \in \{1-n, \dots, n-1\} : \ j \text{ is odd and } \exists b_k \text{ on the diagonal } j\};\\
I_k^- &= \{j \in \{1-n, \dots, n-1\} : \ j \text{ is odd and } \exists -b_k \text{ on the diagonal } j\};\\
J_k &= J_k^+ \cap J_k^-;\\
I_k &= I_k^+ \cap I_k^-.
\end{aligned}$$

From the Hankel structure of JB for $n = 2m$, it follows that

- the element $-b_1$ occurs only on the main antidiagonal so that all the $-b_1$s are on even diagonals; $\#J_1^- = n, \#I_1^- = 0$;
- the element b_1 occurs only on the second and $(n+2)$th antidiagonals so that we have two b_1s on the central odd diagonals and $n-2$ b_1s on the central even diagonals; $\#J_1^+ = n-2, \#I_1^+ = 2$;
- the element $-b_2$ occurs only on the $(n-1)$th and $(2n)$th antidiagonals so that one $-b_2$ is on the main central diagonal and $n-1$ elements equal to $-b_2$ are on the central odd diagonals; $\#J_2^- = 1, \#I_2^- = n-1$;
- the element b_2 occurs only on the third and $(n+3)$th antidiagonal so that we have three b_2s on the central even diagonals and $n-3$ elements equal to b_2 on the central odd diagonals; $\#J_1^+ = 3, \#I_1^+ = n-3$;
- In general, the element $-b_k$, $k = 1, \dots, m$, occurs only on the $n-(k+1)$th and $2n-(k+2)$th antidiagonals so that $\#J_k^- = n-(k-1), \#I_1^- = k-1$ if k is odd; otherwise we obtain $\#J_k^- = k-1, \#I_1^- = n-(k-1)$;
- Analogously, the element b_k, $k = 1, \dots, m$ occurs only on the $n+(k+1)$th and $(k+2)$th antidiagonals so that $\#J_k^+ = n-(k+1), \#I_1^- = k+1$ if k is odd; otherwise we obtain $\#J_k^- = k+1, \#I_1^- = n-(k+1)$.

In conclusion, we find that, for any $k \in \{1, \dots, m\}$, $2m = n$, we have $\#I_k = n-(k+1)$ and $\#J_k = k-1$ if k is even and $\#J_k = n-(k+1)$ and $\#I_k = k-1$ if k is odd. As a matter of fact, we have deduced that $\#I_k + \#J_k$ is constant and equals $n-2$. Therefore

$$\begin{aligned}
d^2 &\geq \inf_Y \sum_k \sum_{j \in I_k} \left[\sum_{\{(s,t):\ B_{s,t} \in \{\pm b_k\}, s-t=j\}} |Y_{s,t} - b_k|^2 + |Y_{s,t} + b_k|^2 \right]\\
&\quad + \sum_k \sum_{j \in J_k} \left[\sum_{\{(s,t):\ B_{s,t} \in \{\pm b_k\}, s-t=j\}} |Y_{s,t} - b_k|^2 + |Y_{s,t} + b_k|^2 \right]\\
&\geq 2 \textstyle\sum_k (\#J_k + \#I_k) b_k^2 = 2(n-2) \sum_k b_k^2.
\end{aligned}$$

So, if $d^2 = o(n)$, then $\sum_k b_k^2 = o(1)$ which is equivalent to $\|JB\|_F^2 = o(n)$ and $\|s(x)\|_{L^2}^2 = o(1)$. If $d^2 = O(1)$, then $\sum_k b_k^2 = O(n^{-1})$ and analogously we find $\|JB\|_F^2 = O(1)$ and $\|s(x)\|_{L^2}^2 = O(n^{-1})$. □

In conclusion, a good Hartley approximation of $A_n(f)$ is substantially given by a good symmetric circulant approximation: the contribution of the Hankel part must be not only small but negligible. By the way, this is the case of the optimal Frobenius approximation, for which it has been proved [4] that $b_i = n^{-1}(a_i - a_{n-i})$ and therefore $|b_i| = O(n^{-1}\max\{|a_i|, |a_{n-i}|\})$ so that

$$\sum_i b_i^2 \leq \frac{c}{n}\|f\|_{L^2}^2,$$

with c a suitable positive constant. Since $C_{2\pi} \subset L^2$, we deduce

$$\|s(x)\|_{L^2}^2,\ \frac{\|JB\|_F^2}{n} = O(n^{-1}). \tag{3.5}$$

In addition, we want to mention that (3.5), as well as Theorem 3.9, holds for any f belonging to the periodic L^2 functions.

Finally, we want to show that all these algebras are "good," and, in particular, the τ class is "better" with respect to the others.

PROPOSITION 3.10. *For any trigonometric even polynomial p_M of fixed degree M independent of n, we have that*

- *$\mathcal{A} = \tau$ implies $A_n(p_M) - \mathcal{A}_n(p_M) = LR_1$, with $\operatorname{rank}(LR_1) = 2(M-1)$;*
- *$\mathcal{A} = \mathcal{C}$ implies $A_n(p_M) - \mathcal{A}_n(p_M) = LR_2$, with $\operatorname{rank}(LR_2) = 2M$;*
- *$\mathcal{A} = \mathcal{H}$ implies $A_n(p_M) - \mathcal{A}_n(p_M) = LR_2$ and $\mathcal{H}_n(p_M) = \mathcal{C}_n(p_M)$.*

Proof. It is enough to refer to the structure of these algebras and to the expression of the related generating functions. □

THEOREM 3.11. *If $\mathcal{S}_n(f)$ converges to f on the grid points $\mathcal{W}(\mathcal{A})$, then we may choose η small enough so that there exists $\epsilon = \epsilon(\eta)$ and $\bar{n}_{\epsilon,\mathcal{S}_n}$: $\forall n \geq \bar{n}_{\epsilon,\mathcal{S}_n}$,*

$$N_\eta = \operatorname{rank}(LR(p_{M_\epsilon})) + 1,$$

where LR is the low rank matrix related to the goodness of the algebras $\mathcal{A}$ and where the degree $M = M_\epsilon$ of the polynomial P_M implies $\|\mathcal{S}_n(f) - p_M\|_\infty + \|f - p_M\|_\infty \leq \epsilon(\eta)$ so that, after $\operatorname{rank}(L(p_{M_\epsilon}))$ iterations, only one iteration is required for convergence within an accuracy η.

Observe that the preceding results tell us a lot about the convergence of the PCG methods associated with matrix algebra and linear approximation processes. When there is strong convergence (strong or proper clustering in the terminology used in [59]) we have (after N_η iterations) superlinear convergence, but we may have a sublinear behavior when the weak convergence (weak or general clustering) case occurs [57, 23]. However, also with regard to the strong matrix algebra convergence, the BT technique [47] has more potential as described in the following where we prove that, for a suitable choice of the sequence $l(n)$ and of the operator $\mathcal{S}_n$, we define methods having a total cost of $O(n \log n)$ ops and where $N_\eta = 1$ for n large enough.

3.3. BT preconditioning and approximation processes. In this brief subsection we analyze the convergence of BT preconditioners generated by an approximation process $\mathcal{S}_n(f)$ converging to the function f. For this purpose we denote by $E_n(f)$ the error $f - \mathcal{S}_n(f)$; clearly, as n tends to infinity, we have $E_n(f) \to 0$. At this point let us consider $A_n(\mathcal{S}_{l(n)}(f))$ as preconditioner where $l(n)$ is a nondecreasing sequence of nonnegative integers. In the light of Theorem 2.5, the preconditioned matrix

$$A_n^{-1}(\mathcal{S}_{l(n)}(f))A_n(f) \tag{3.6}$$

has eigenvalues belonging to the open set (m_n, M_n) with

$$m_n = \inf_x \frac{f(x)}{\mathcal{S}_{l(n)}(f)(x)} \geq 1 - \frac{\|E_{l(n)}(f)\|_\infty}{m_f - \|E_{l(n)}(f)\|_\infty},$$

$$M_n = \sup_x \frac{f(x)}{\mathcal{S}_{l(n)}(f)(x)} \leq 1 + \frac{\|E_{l(n)}(f)\|_\infty}{m_f - \|E_{l(n)}(f)\|_\infty}$$

and $m_f = \min f > 0$. The preceding relationships imply that $\lim_{n\to\infty} = \frac{M_n}{m_n} = 1$ and, by virtue of the convergence results on PCG methods [1], we find that there exists an integer $\bar{n}$ such that, if $n \geq \bar{n}$ then the number N_η of iterations to reach the solution within an accuracy η equals 1. In other words, there exists a bandwidth $\bar{L}$ depending on the approximation process, on f and on η, so that the PCG method with coefficient matrix $A_n(f)$ and $A_n(\mathcal{S}_{\bar{L}}(f))$ as preconditioner requires only one iteration. Of course the integer $\bar{n}$ depends on how fast the sequence $\mathcal{S}_{l(n)}(f)$ converges to f.

For instance, let us set $\mathcal{S}_n(f) = C_n(f)$, the Cesàro sum of f, and $\mathcal{S}_n(f) = DL_n(f)$ the De La Vallée Poussin operator [63], where $f \in C_{2\pi}^\infty$ is chosen such that its best approximation by polynomials of degree m produces an error which is of order $O(2^{-m})$. By the well-known approximation properties of these two operators, it follows that

$$\begin{array}{lcl} \|f - C_n(f)\|_\infty & \leq & \frac{c_1}{n}, \quad c_1 > 0, \\ \|f - DL_n(f)\|_\infty & \leq & \frac{c_2}{2^n}, \quad c_2 > 0. \end{array}$$

Now, let us fix $l(n) = \log n$ as done in [47]. Therefore, in the first case, i.e., for $C_{l(n)}(f)$, we find

$$m_n \geq 1 - \frac{c_1^*}{\log n}, \qquad M_n \leq 1 + \frac{c_1^{**}}{\log n},$$

while, for the operator $DL_{l(n)}(f)$, we obtain

$$m_n \geq 1 - \frac{c_1^*}{n}, \qquad M_n \leq 1 + \frac{c_1^{**}}{n}.$$

By the standard error analysis carried out by Axelsson and Lindskog [1], we deduce that the minimum integer $\bar{n}$ for which only one iteration is required is, in the first case, $\bar{n} \geq 2^{\frac{k_1}{\eta}}$, k_1 being a suitable positive constant. In the second case we find $\bar{n} \geq \frac{k_2}{\eta}$, k_2 being a positive constant.

Observe that, due to the slowness of convergence of the Cesàro sum, the first estimate has only a theoretical appeal, since the value $2^{k_1/\eta}$ grows too fast with η and therefore huge dimensions are required to converge within only one PCG iteration. On the other hand, we point out that in the case of the latter example the degree of decay of the coefficients a_k is also exponential: therefore, if f is strictly positive (the operators $\{A_n(f)\}_n$ have uniformly bounded inverse), then the related linear systems can be globally treated as banded systems.

3.3.1. How to choose the bandwidth $l(n)$. The cost per iteration of the PCG procedure is given by (a) few matrix-vector products and inner-vector products plus the cost (b) of a solution of a system whose coefficient matrix is just the preconditioned matrix. Owing to the use of FFTs/FTTs, the cost (a) can be bounded by $O(n \log n)$ ops [60, 35, 39] while the cost (b) is related to the choice of a good band solver and, of course, to the choice of the bandwidth $l(n)$. Observe that the more $l(n)$ grows, the fewer PCG iterations are required to reach the convergence and, actually, due to superlinearity of the procedure we may arrive at only one iteration. Therefore the "optimal choice" is to maximize $l(n)$ under the constraint that the cost (b) of the preconditioning system cannot exceed $O(n \log n)$ ops. We have the following possibilities.

- Standard band solvers (related to the Gaussian elimination) [32]: if the bandwidth is l, then $O(l^2 n)$ ops are required. Consequently, with this choice, we have $l(n) \leq \min\{c\sqrt{\log n}, \bar{L}\}$.
- Multigrid methods for symmetric Toeplitz systems [29, 30]: we find a cost of $O(ln)$ ops and then we obtain $l(n) \leq \min\{c \log n, \bar{L}\}$ (the choice made in [47]).
- A recent displacement-rank recursive technique introduced by Bini and Meini [5]: the cost is given by $O(n \log l + l \log^2 l)$ and therefore a bandwidth $l(n) = O(\frac{n}{\log n})$ is allowed.

Observe that the impact of the third technique on the BT preconditioning technique is dramatic: since a bandwidth of $l(n) = O(\frac{n}{\log n})$ is allowed, we may really appreciate the superlinearity of the proposed method. For instance, the value $\bar{n}$ in the preceding example with $\mathcal{S}_n(f) = DL_n(f)$ can be only logarithmic with η^{-1}! On the other hand the implementation of this technique is not trivial. Nevertheless a FORTRAN code exists and can be found by anonymous FTP at MORSE.DM.UNIPI.IT in the directory pub/TOEPLITZ/SOFTWARE/BTS, files symm.tar.Z and scalar.tar.Z.

4. Numerical experiments and conclusive remarks.

4.1. Numerical experiments. In all the numerical experiments we consider two kinds of test functions: $f_\alpha = (x^2/2) + \alpha$ and $f_\beta = ((x/\pi)^2 - 1)^2 + \beta$. Moreover we consider two different types of data vectors b. The first is made up by all ones. The second is randomly generated. In all the tables the numbers between parentheses are related to the number of PCG iterations when a random data vector b is considered. The stopping criterion is given by the two-norm of the residual less than 10^{-7}. All the experiments are done by using MATLAB.

4.1.1. Matrix algebra preconditioners. For the choice of the preconditioners we considered two algebras, namely the τ class and the circulants, and BT matrices. Concerning the algebras we have constructed the *natural preconditioners* (Strang type) C_{nat} and τ_{nat} whose related approximation process is given by the Fourier polynomial of degree $n/2$ and n, respectively, and the *optimal preconditioners* (T. Chan type) C_{opt} and τ_{opt} whose approximation processes have the same asymptotical behavior as the Cesàro sum.

Concerning the results displayed in Tables 4.1, 4.2, and 4.3, two remarks are needed. First we notice that the number of iterations for the τ preconditioners is generally slightly less than for the circulant case. This agrees with Proposition 3.10 where it is proved that the border conditions are slightly heavier in the circulant case. Second, if we observe the behavior of the preconditioners τ_{nat} and τ_{opt} we conclude that their behavior is substantially identical, in particular for high or large dimensions.

Table 4.1
Number of PCG steps in the case of f_α with $\alpha = 5 + \pi^2/6$.

n	τ_{nat}	C_{nat}	τ_{opt}	C_{opt}
32	3 (3)	4 (5)	3 (3)	4 (5)
64	3 (3)	4 (5)	3 (3)	4 (5)
128	3 (3)	4 (5)	3 (3)	4 (5)
256	3 (3)	4 (4)	3 (3)	4 (4)
512	3 (3)	4 (4)	3 (3)	4 (4)

Table 4.2
Number of PCG steps in the case of f_β with $\beta = 1$.

n	τ_{nat}	C_{nat}	τ_{opt}	C_{opt}
32	3 (3)	3 (4)	3 (3)	4 (5)
64	3 (3)	3 (4)	3 (3)	4 (5)
128	3 (3)	3 (4)	3 (3)	3 (4)
256	3 (3)	3 (4)	3 (3)	3 (4)
512	2 (3)	3 (4)	3 (3)	3 (4)

Table 4.3
Number of PCG steps in the case of f_β with $\beta = 0.01$.

n	τ_{nat}	C_{nat}	τ_{opt}	C_{opt}
32	3 (3)	4 (5)	3 (4)	7 (10)
64	3 (3)	4 (5)	3 (4)	7 (10)
128	3 (3)	4 (5)	3 (4)	6 (9)
256	3 (3)	4 (5)	3 (4)	5 (8)
512	2 (3)	3 (4)	3 (3)	5 (6)

Table 4.4
Number of PCG steps in the case of f_α with $\alpha = 5 + \pi^2/6$.

n	$A_n(F_{l_1(n)})$	$A_n(F_{l_2(n)})$
32	5 (5)	5 (5)
64	4 (5)	3 (4)
128	4 (4)	3 (3)
256	3 (4)	2 (3)
512	3 (4)	2 (3)

Nevertheless they are characterized by two very different approximation processes, one substantially faster than the other. This insensitivity to the convergence rate of the approximation process fully agrees with Theorem 3.7. The same remark holds true if we analyze the behavior of the preconditioners C_{nat} and C_{opt}.

4.1.2. BT preconditioners. Here we consider a unique approximation process, the Fourier process F_l. The bandwidth functions $l(n)$ are of two types: the first one $l_1(n) = 2(\log(n) - 4) + 1$ of logarithmic order and the second $l_2(n) = 2\lfloor\sqrt{n} - 4\rfloor + 1$ of $\sqrt{n}$-order. The results displayed in Tables 4.4 to 4.7 indicate an evident superlinearity of the related PCG methods.

Finally, we report two further numerical experiments concerning two cases in which f has zeros and is nonnegative or has a nondefinite sign. In the nonnegative case, the approximation process K_l is related to the Chebyshev cosine expansion and is described in [47]. The superiority with regard to the matrix algebra approach is impressive as shown in Table 4.8.

The example reported in Table 4.9 is more delicate. In this case f is continuous and nondefinite since $\min f = -0.9$ and $\max f = 0.1$. Moreover the preconditioner is

Table 4.5
Number of PCG steps in the case of f_α with $\alpha = 0.1$.

n	$A_n(F_{l_1(n)})$	$A_n(F_{l_2(n)})$
32	12 (18)	12 (18)
64	11 (13)	7 (8)
128	11 (15)	6 (6)
256	8 (8)	5 (5)
512	8 (8)	4 (4)

Table 4.6
Number of PCG steps in the case of f_β with $\beta = 1$.

n	$A_n(F_{l_1(n)})$	$A_n(F_{l_2(n)})$
32	4 (4)	4 (4)
64	3 (3)	2 (3)
128	2 (3)	2 (2)
256	2 (3)	2 (2)
512	2 (2)	1 (2)

Table 4.7
Number of PCG steps in the case of f_β with $\beta = 0.01$.

n	$A_n(F_{l_1(n)})$	$A_n(F_{l_2(n)})$
32	7 (10)	7 (10)
64	6 (9)	4 (5)
128	5 (6)	3 (4)
256	4 (5)	2 (3)
512	3 (5)	2 (3)

Table 4.8
Number of PCG steps in the case of $f = x^4$.

n	C_{opt}	$A_n(K_{l_1(n)})$
64	26	21
128	77	15
256	179	13
512	406	11

Table 4.9
Number of PCG steps in the case of f_β with $\beta = -0.9$.

n	$A_n(F_{l_2(n)})$
32	8 (9)
64	4 (5)
128	3 (4)
256	3 (3)
512	2 (3)

also nondefinite since its generating function is the Fourier approximation F_l of f of degree $l = l_2(n)$. Moreover,

$$\sup_x |F_{l_2(n)}(f)/f - 1| = \infty \tag{4.1}$$

for n large enough since F_l and f have zeros in different positions. Nevertheless $F_{l_2(n)}(f)/f$ converges to 1 in another sense and, more precisely, in measure [43]:

actually, for any $\epsilon > 0$ we have that

$$\lim_{n\to\infty} m\{x:\ |F_{l_2(n)}(f)/f - 1| > \epsilon\} = 0.$$

This kind of convergence between functions has a counterpart in the spectral distribution of the eigenvalues of the preconditioned matrices. In fact for $n = 256$ we observe that the real part of the eigenvalues of the preconditioned matrix lies between 1 and 1.14 while the imaginary part has absolute value bounded by $5.89 * 10^{-6}$. In spite of (4.1), we have no outliers: we remark that the absence of outliers can be explained by means of the "boundary layer" argument introduced in [54].

4.2. Final remarks. It should be stressed that, under the constraint that the approximation operator $\mathcal{S}_n$ converges to f, the qualitative convergence of the related PCG is almost the same in all the cases where the preconditioner associated with $\mathcal{S}_n$ belongs to a trigonometric algebra. This means that the matrix-algebra preconditioners are not really sensitive to the order of the approximation. This fact is basically due to the existence of "border conditions" in the approximation of $\{A_n(f)\}$ by $\{\mathcal{A}_n(\mathcal{S}_n)\}$: in other words, the difference between $A_n(f)$ and $\mathcal{A}_n(\mathcal{S}_n)$ can be always viewed as the sum of a part of a small norm which is sensitive to the goodness of the approximation scheme and a part of constant rank LR which is in a certain sense inherent to the considered algebra.

To conclude, the presence of LR hides the quality of the convergence of $\{\mathcal{A}_n(\mathcal{S}_n)\}$ to $\{A_n(f)\}$, for n large enough, in the sense that we expect that a number of steps of the PCG method close to $\text{rank}(LR) + 1$ are required for the convergence [1].

On the other hand, when we use the BT preconditioning, the border conditions no longer exist, but, instead of the convergence on the mesh $\mathcal{W}$, we have to require the uniform convergence of $\mathcal{S}_n$ to f. In this case, we may fully appreciate the degree of convergence of $\{A_n(\mathcal{S}_{l(n)})\}$ to $\{A_n(f)\}$ when $l(n)$ is a slowly increasing bounded function. As shown in section 3 and in subsection 4.1, this explanation is plainly evident from a theoretical and practical point of view. For other numerical results regarding the "insensitivity" of the matrix algebra approach to the goodness of the approximation scheme, see the numerical experiments in [17], for the circulant case, and Table 2 in [53] for the τ case. Finally, for additional numerical evidence of the superlinearity of the BT preconditioning approach (in the more difficult case where f has zeros), refer to the last table in [47].

REFERENCES

[1] O. Axelsson and G. Lindskög, *The rate of convergence of the preconditioned conjugate gradient method*, Numer. Math., 52 (1986), pp. 499–523.

[2] D. Bini and M. Capovani, *Spectral and computational properties of band symmetric Toeplitz matrices*, Linear Algebra Appl., 52/53 (1983), pp. 99–126.

[3] D. Bini and F. Di Benedetto, *A new preconditioner for the parallel solution of positive definite Toeplitz linear systems*, in Proc. 2nd SPAA Conf., Crete, Greece, 1990, pp. 220–223.

[4] D. Bini and P. Favati, *On a matrix algebra related to the discrete Hartley transform*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 500–507.

[5] D. Bini and B. Meini, *Effective methods for solving banded Toeplitz systems*, SIAM J. Matrix Anal. Appl., to appear.

[6] R. H. Chan *Circulant preconditioners for Hermitian Toeplitz systems*, SIAM J. Matrix Anal. Appl., 10 (1989), pp. 542–550.

[7] R. H. Chan, *Toeplitz preconditioners for Toeplitz systems with nonnegative generating functions*, IMA J. Numer. Anal., 11 (1991), pp. 333–345.

[8] R. H. Chan, T. F. Chan, and C. Wong, *Cosine Transform Based Preconditioners for Total Variation Minimization Problems in Image Restoration*, in Proc. 2nd IMACS Internat. Symp. on Iterative Methods in Linear Algebra, Blagoevgrad, Bulgaria, 1995, pp. 311–329.

[9] R. H. Chan and X. Jin, *A family of block preconditioners for block systems*, SIAM J. Sci. Stat. Comp., 13 (1992), pp. 1218–1235.

[10] R. H. Chan, X. Jin, and M. C. Yeung, *The circulant operator in the Banach algebra of matrices*, Linear Algebra Appl., 149 (1991), pp. 41–53.

[11] R. H. Chan and M. Ng, *Conjugate gradient methods for Toeplitz systems*, SIAM Rev., 38 (1996), pp. 427–482.

[12] R. H. Chan and G. Strang, *Toeplitz equations by conjugate gradients with circulant preconditioner*, SIAM J. Sci. Stat. Comput., 10 (1989), pp. 104–119.

[13] R. H. Chan and P. Tang, *Fast band-Toeplitz preconditioners for Hermitian Toeplitz systems*, SIAM J. Sci. Comput., 15 (1994), pp. 164–171.

[14] R. H. Chan and P. Tang, *Constrained minimax approximation and optimal preconditioners for Toeplitz matrices*, Numer. Algorithms, 5 (1993), pp. 353–364.

[15] R. H. Chan and M. C. Yeung, *Circulant preconditioners for Toeplitz matrices with positive continuous generating functions*, Math. Comp., 58 (1992), pp. 233–240.

[16] R. H. Chan and M. C. Yeung, *Jackson's theorem and circulant preconditioned Toeplitz systems*, J. Approx. Theory, 70 (1992), pp. 191–205.

[17] R. H. Chan and M. C. Yeung, *Circulant preconditioners constructed from kernels*, SIAM J. Numer. Anal., 29 (1992), pp. 1093–1103.

[18] T. F. Chan, *An optimal circulant preconditioner for Toeplitz systems*, SIAM J. Sci. Stat. Comput., 9 (1988), pp. 766–771.

[19] E. Cheney, *Introduction to Approximation Theory*, McGraw–Hill, New York, 1966.

[20] F. Costabile, M. I. Gualtieri, and S. Serra,[1] *Asymptotic expansions and extrapolation for Bernstein polynomials with applications*, BIT, 36 (1996), pp. 676–687.

[21] F. Costabile, M. I. Gualtieri, and S. Serra, *Asymptotic expansions for some classical operators and their use in approximation theory*, in Proc. in 7th International Coll. on Differential Equations, D. Bainov, ed., Plovdiv, Bulgaria, 1996, pp. 67–74.

[22] P. Davis, *Circulant Matrices*, John Wiley and Sons, New York, 1979.

[23] F. Di Benedetto, *Analysis of preconditioning techniques for ill-conditioned Toeplitz matrices*, SIAM J. Sci. Comput., 16 (1995), pp. 682–697.

[24] F. Di Benedetto, *Preconditioning of block Toeplitz matrices by sine transforms*, SIAM J. Sci. Comput., 18 (1997), pp. 499–515.

[25] F. Di Benedetto and S. Serra Capizzano, *A unifying approach to abstract matrix algebra preconditioning*, Numer. Math., to appear.

[26] F. Di Benedetto and S. Serra Capizzano, *Optimal and Superoptimal Matrix Algebra Operators*, TR. 360, Department of Mathematics, University of Genova, Italy, 1997.

[27] F. Di Benedetto, G. Fiorentino, and S. Serra, *C.G. preconditioning for Toeplitz matrices*, Comput. Math. Appl., 25 (1993), pp. 35–45.

[28] Z. Dizian and G. Freud, *Linear approximation processes with limited oscillation*, J. Approx. Theroy, 12 (1974), pp. 23–31.

[29] G. Fiorentino and S. Serra, *Multigrid methods for Toeplitz matrices*, Calcolo, 28 (1991), pp. 283–305.

[30] G. Fiorentino and S. Serra, *Multigrid methods for symmetric positive definite block Toeplitz matrices with nonnegative generating functions*, SIAM J. Sci. Comput., 17 (1996), pp. 1068–1081.

[31] I. Gohberg and I. Fel'dman, *Convolution Equations and Projection Methods for Their Solution*, Transl. Math. Monogr. 41, AMS, Providence, RI, 1974.

[32] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, MD, 1983.

[33] U. Grenander and M. Rosemblatt, *Statistical Analysis of Stationary Time Series*, 2nd ed., Chelsea, New York, 1984.

[34] U. Grenander and G. Szegö, *Toeplitz Forms and Their Applications*, 2nd ed., Chelsea, New York, 1984.

[35] G. Heinig and K. Rost, *Representation of Toeplitz-plus-Hankel matrices using trigonometric transformations with applications to fast matrix-vector multiplication*, private communication, 1997.

[1] The author Stefano Serra Capizzano and Stefano Serra cited in the references are the same person.

[36] T. Huckle, *Circulant and skewcirculant matrices for solving Toeplitz matrix problems*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 767–777.

[37] D. Jackson, *The Theory of Approximation*, AMS, New York, 1930.

[38] X. A. Jin, *Hartley preconditioners for Toeplitz systems generated by positive continuous functions*, BIT, 34 (1994), pp. 367–371.

[39] T. Kailath and V. Olshevsky, *Displacement structure approach to discrete-trigonometric-transform based preconditioners of G. Strang type and T. Chan type*, Calcolo, 33 (1996), pp. 191–208.

[40] P. P. Korovkin, *Linear Operators and Approximation Theory*, Hindustan Publishing Co., Delhi, 1960.

[41] I. P. Natanson, *Constructive Function Theory,* I, Frederick Ungar Publishing Co., New York, 1964.

[42] A. Oppenheim, *Applications of Digital Signal Processing*, Prentice–Hall, Englewood Cliffs, NJ, 1978.

[43] W. Rudin, *Real and Complex Analysis*, 3rd ed., McGraw–Hill, Singapore, 1986.

[44] S. Serra, *Preconditioning strategies for asymptotically ill-conditioned block Toeplitz systems*, BIT, 34 (1994), pp. 579–594.

[45] S. Serra, *Conditioning and solution, by means of preconditioned conjugate gradient methods, of Hermitian (block) Toeplitz systems*, in Proc. in Advanced Signal Processing Algorithms, Architectures, and Implementations—SPIE Conf., F. Luk, ed., San Diego, CA, 1995, pp. 326–337.

[46] S. Serra, *Preconditioning strategies for Hermitian Toeplitz systems with nondefinite generating functions*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 1007–1019.

[47] S. Serra, *Optimal, quasi-optimal and superlinear band-Toeplitz preconditioners for asymptotically ill-conditioned positive definite Toeplitz systems*, Math. Comp., 66 (1997), pp. 651–665.

[48] S. Serra, *New PCG based algorithms for the solution of Hermitian Toeplitz systems*, Calcolo, 32 (1995), pp. 153–176.

[49] S. Serra, *Sulle proprietà spettrali di matrici precondizionate di Toeplitz*, Boll. Un. Mat. Ital. A, 11 (1997), pp. 463–483.

[50] S. Serra, *The extension of the concept of* generating *function to a class of preconditioned Toeplitz matrices*, Linear Algebra Appl., 267 (1997), pp. 139–161.

[51] S. Serra, *On the extreme eigenvalues of Hermitian (block) Toeplitz matrices*, Linear Algebra Appl., 270 (1998), pp. 109–129.

[52] S. Serra, *A Korovkin-type theory for finite Toeplitz operators via matrix algebras*, Numer. Math., to appear. Also invited lecture SIAM Meeting—Minisymposium on "Fast Toeplitz Solvers," Stanford, CA, July 14–18, 1997.

[53] S. Serra, *Superlinear PCG methods for symmetric Toeplitz systems*, Math. Comp., to appear.

[54] S. Serra, *How to choose the best iterative strategy for symmetric Toeplitz systems*, SIAM J. Numer. Anal., to appear.

[55] S. Serra Capizzano, *An ergodic theorem for classes of preconditioned matrices*, Linear Algebra Appl., 282 (1998), pp. 161–183.

[56] G. Strang, *A proposal for Toeplitz matrix calculation*, Stud. Appl. Math., 74 (1986), pp. 171–176.

[57] V. Strela and E. Tyrtyshnikov, *Which circulant preconditioner is better?*, Math. Comp., 65 (1996), pp. 137–150.

[58] E. Tyrtyshnikov, *Optimal and superoptimal circulant preconditioners*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 459–473.

[59] E. Tyrtyshnikov, *A unifying approach to some old and new theorems on distribution and clustering*, Linear Algebra Appl., 232 (1996), pp. 1–43.

[60] C. Van Loan, *Computational Frameworks for the Fast Fourier Transform*, SIAM, Philadelphia, PA, 1992.

[61] H. Widom, *Toeplitz matrices*, in Studies in Real and Complex Analysis, I. Hirshman Jr., ed., MAA, 1965.

[62] J. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, 1965.

[63] A. Zygmund, *Trigonometric Series*, Cambridge University Press, Cambridge, 1959.

SIAM J. Matrix Anal. Appl.
Vol. 20, No. 2, pp. 466–470

INEQUALITIES FOR UNITARILY INVARIANT NORMS[*]

XINGZHI ZHAN[†]

Abstract. Let A, B, X be complex matrices with A, B positive semidefinite. It is proved that

$$(2+t)||A^rXB^{2-r}+A^{2-r}XB^r|| \le 2||A^2X+tAXB+XB^2||$$

for any unitarily invariant norm $||\cdot||$ and real numbers r, t satisfying $1 \le 2r \le 3$, $-2 < t \le 2$. The case $r = 1$, $t = 0$ of this result is the well-known arithmetic-geometric mean inequality due to R. Bhatia and C. Davis [*SIAM J. Matrix Anal. Appl.*, 14 (1993), pp. 132–136]. Several other unitarily invariant norm inequalities are derived.

Key words. unitarily invariant norm, Hadamard product, arithmetic-geometric mean inequality

AMS subject classifications. 15A60, 15A18, 15A45, 47A30

PII. S0895479898323823

1. Introduction.

Let $M_{m,n}$ be the space of $m \times n$ complex matrices and $M_n \equiv M_{n,n}$. Bhatia and Davis [3] proved that for arbitrary $A, B, X \in M_n$ and any unitarily invariant norm $||\cdot||$,

$$2||AXB^*|| \le ||A^*AX + XB^*B||, \tag{1}$$

which is known as the arithmetic-geometric mean inequality. This result has aroused much interest. For different proofs see [3], [6], [7], and [10]. For equivalent inequalities see [5]. For its geometrical meaning see [2].

We remark that the converse is also true; that is, if (1) holds for all $A, B, X \in M_n$, then $||\cdot||$ must be unitarily invariant. For unitary matrices U, V, setting $A = U$ and $B = V^*$ in (1) gives $||UXV|| \le ||X||$. Replacing X by U^*XV^* in this inequality yields $||X|| \le ||U^*XV^*||$. Thus $||UXV|| = ||X||$ for all unitary U, V.

We shall generalize (1) by introducing two parameters r and t. The tool used here is the induced norm of the Hadamard multiplier operator with respect to unitarily invariant norms, and the starting point is a result for matrix equations.

2. Preliminaries.

The Hadamard product of $A = (a_{ij})$ and $B = (b_{ij}) \in M_n$ is $A \circ B \equiv (a_{ij}b_{ij}) \in M_n$. Given $A \in M_n$, consider the Hadamard multiplier operator T_A on M_n defined by

$$T_A(X) \equiv A \circ X \quad \text{for } X \in M_n.$$

We use the same notation to denote a norm on M_n and the induced norm of the linear operator T_A with respect to this norm. Let $||\cdot||_\infty$ be the spectral (operator) norm. Then

$$||T_A||_\infty \equiv \sup\{||A \circ X||_\infty : ||X||_\infty \le 1,\, X \in M_n\}.$$

The following useful fact can be deduced from [1].

[*] Received by the editors April 15, 1998; accepted for publication (in revised form) by R. Bhatia, May 26, 1998; published electronically November 16, 1998.
http://www.siam.org/journals/simax/20-2/32382.html

[†] Institute of Mathematics, Peking University, Beijing 100871 China (zhan@sxx0.math.pku.edu.cn).

LEMMA 1. *For any unitarily invariant norm $||\cdot||$ on M_n and all $A \in M_n$,*

$$||T_A|| \leq ||T_A||_\infty.$$

For Hermitian $A, B \in M_n$ we write $A \geq B$ to mean that $A - B$ is positive semidefinite. A real valued function f is said to be *matrix monotone increasing* on an interval J if for all Hermitian matrices A and B of all orders whose eigenvalues lie in J,

$$A \geq B \qquad \text{implies} \qquad f(A) \geq f(B).$$

It is *matrix monotone decreasing* if the inequality is reversed after the application of f. Simple examples of matrix monotone increasing functions on $(0, \infty)$ are $x^p\,(0 < p < 1)$, $\log x$, and $\log(1+x)$. The following basic fact can be found in [9, Theorem 2].

LEMMA 2. *A function $f : (0, \infty) \to (0, \infty)$ is matrix monotone increasing if and only if $f(x)/x$ is matrix monotone decreasing.*

Applying this lemma to $f(x) = x^p (0 < p \leq 1)$ we know that $x^q(-1 \leq q \leq 0)$ is matrix monotone decreasing. This also follows from the fact that $A \geq B > 0 \Longleftrightarrow B^{-1} \geq A^{-1} > 0$.

We will need the following two results due to Kwong [9, Theorems 9, 10]. For related work see [8].

LEMMA 3. *Let the function $g(x) = f(x) + h(x)$ with f positive matrix monotone increasing and h positive matrix monotone decreasing. Then for any positive definite $A \in M_n$ and positive semidefinite $P \in M_n$, the solution X of the matrix equation*

$$AX + XA = g(A)P + Pg(A)$$

is positive semidefinite.

LEMMA 4. *Let $t \in (-2, 2]$. Under the same hypotheses as in Lemma* 3 *the solution Y of the matrix equation*

$$A^2Y + YA^2 + tAYA = g(A)P + Pg(A)$$

is positive semidefinite.

By continuity we may choose f or h not strictly positive; say, let $f = 0$ or $h = 0$.

3. Main results.

LEMMA 5. *Let $\sigma_1, \sigma_2, \ldots, \sigma_n$ be positive real numbers and $t \in (-2, 2]$, $r \in [-1, 1]$. Then the $n \times n$ matrix*

$$\left(\frac{\sigma_i^r + \sigma_j^r}{\sigma_i^2 + t\sigma_i\sigma_j + \sigma_j^2}\right)_{i,j=1,2,\ldots,n}$$

is positive semidefinite.

Proof. Apply Lemma 4 with $A = \operatorname{diag}(\sigma_1, \ldots, \sigma_n)$ and P being the matrix all of whose entries are 1, $f(x) = x^r$ $(0 < r \leq 1)$, $h(x) = 0$ and $f(x) = 0$, $h(x) = x^r$ $(-1 \leq r \leq 0)$, respectively. □

Using a different technique, Bhatia and Parthasarathy [4, Theorem 5.2] recently gave another proof of Lemma 5. Further they showed that the interval $(-2, 2]$ of t is largest possible for the conclusion to hold for all the orders n [4, Theorem 5.1]. We are now ready to prove the main results.

THEOREM 6. *Let $A \in M_m$, $B \in M_n$ be positive semidefinite and $X \in M_{m,n}$ be arbitrary. Then*

$$(2) \qquad (2+t)||A^r X B^{2-r} + A^{2-r} X B^r|| \le 2||A^2 X + tAXB + XB^2||$$

for any unitarily invariant norm $||\cdot||$ and real numbers r, t satisfying $1 \le 2r \le 3$, $-2 < t \le 2$.

Proof. We need only consider the square case $m = n$, since nonsquare matrices can be augmented to square ones with zero blocks, which does not change their unitarily invariant norms. We first prove the special case $A = B$, i.e.,

$$(3) \qquad (2+t)||A^r X A^{2-r} + A^{2-r} X A^r|| \le 2||A^2 X + tAXA + XA^2||.$$

By continuity, without loss of generality, assume that A is positive definite. Let $A = U\Sigma U^*$ be the spectral decomposition with U unitary and $\Sigma = \operatorname{diag}(\sigma_1, \sigma_2, \dots, \sigma_n)$, $\sigma_1 \ge \cdots \ge \sigma_n > 0$. Since $||\cdot||$ is unitarily invariant, (3) is equivalent to

$$(2+t)||\Sigma^r U^* X U \Sigma^{2-r} + \Sigma^{2-r} U^* X U \Sigma^r|| \le 2||\Sigma^2 U^* X U + t\Sigma U^* X U \Sigma + U^* X U \Sigma^2||,$$

which may be rewritten as

$$(2+t)||[(\sigma_i^r \sigma_j^{2-r} + \sigma_i^{2-r}\sigma_j^r) y_{ij}]|| \le 2||[(\sigma_i^2 + t\sigma_i\sigma_j + \sigma_j^2) y_{ij}]|| \quad \text{for all } Y = [y_{ij}] \in M_n.$$

This is the same as

$$||G \circ Z|| \le ||Z|| \quad \text{for all } Z \in M_n,$$

where

$$G = \frac{2+t}{2}\left(\frac{\sigma_i^r \sigma_j^{2-r} + \sigma_i^{2-r}\sigma_j^r}{\sigma_i^2 + t\sigma_i\sigma_j + \sigma_j^2}\right) \in M_n$$

or, equivalently,

$$(4) \qquad ||T_G|| \le 1.$$

See section 2 for the operator T_G. $1 \le 2r \le 3 \Longrightarrow -1 \le 2(1-r) \le 1$. By Lemma 5

$$G = \frac{2+t}{2}\Sigma^r \left(\frac{\sigma_i^{2(1-r)} + \sigma_j^{2(1-r)}}{\sigma_i^2 + t\sigma_i\sigma_j + \sigma_j^2}\right)\Sigma^r$$

is positive semidefinite. A well-known result of Schur [11] (see also [1, p. 363]) says that the Hadamard multiplier norm of a positive semidefinite matrix with respect to the spectral norm is equal to its largest diagonal entry. Now G is a correlation matrix, i.e., a positive semidefinite matrix with each diagonal entry equal to 1. Thus $||T_G||_\infty = 1$. By Lemma 1 the inequality (4) holds. This proves (3).

Note that for any unitarily invariant norm $||\cdot||$ and any matrix F,

$$\left\|\begin{pmatrix} 0 & F \\ 0 & 0 \end{pmatrix}\right\| = ||F||.$$

For the general case, applying (3) with A and X replaced by

$$\begin{pmatrix} A & 0 \\ 0 & B \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 0 & X \\ 0 & 0 \end{pmatrix},$$

respectively, completes the proof. □

Denote the modulus of $A \in M_n$ by $|A| = (A^*A)^{1/2}$. The special case $r = 1$ of Theorem 6, together with polar decompositions, yields the following corollary.

Corollary 7. *Let $A \in M_m$, $B \in M_n$, and $X \in M_{m,n}$ be arbitrary. Then for any unitarily invariant norm $||\cdot||$ and any $t \in (-2, 2]$,*

$$(2+t)||AXB^*|| \le ||A^*AX + t|A|\,X\,|B| + XB^*B||. \tag{5}$$

The inequality (1) corresponds to the case $t = 0$ of (5).

One of the referees of this paper observed that the case $-2 < t \le 0$ of (5) follows from (1):

$$\begin{aligned} ||A^*AX + t|A|\,X\,|B| + XB^*B|| &\ge ||A^*AX + XB^*B|| - |t|\,||\,|A|\,X\,|B|\,|| \\ &\ge (2+t)||\,|A|\,X\,|B|\,|| \\ &= (2+t)||AXB^*||. \end{aligned}$$

An immediate consequence of Corollary 7 is the following corollary.

Corollary 8. *Let $P \in M_m$, $Q \in M_n$, and $T \in M_{m,n}$ with P, Q positive definite. Then for any unitarily invariant norm $||\cdot||$ and any $t \in (-2, 2]$,*

$$(2+t)||T|| \le ||PTQ^{-1} + tT + P^{-1}TQ||.$$

Note that the matrix monotone increasing function $\log(1+x) < x$ for all $x > 0$. Taking $f = \log(1+x)$ and $h = 0$ in Lemma 4, using the same ideas as in the proof of Theorem 6 we obtain the following theorem.

Theorem 9. *Let $A \in M_m$, $B \in M_n$ be positive semidefinite and $X \in M_{m,n}$ be arbitrary. Then for any unitarily invariant norm $||\cdot||$ and $-2 < t \le 2$,*

$$(2+t)||\sqrt{A}[\log(I+A)X + X\log(I+B)]\sqrt{B}|| \le 2||A^2X + tAXB + XB^2||,$$

where I is the identity matrix and $\sqrt{A}$ is the unique positive semidefinite square root of A.

Taking $f = \log(1+x)$ and $h = 0$ in Lemma 3, using the method in the proof of Theorem 6 again we get the following theorem.

Theorem 10. *Let $A \in M_m$, $B \in M_n$ be positive semidefinite and $X \in M_{m,n}$ be arbitrary. Then*

$$||\log(I+A)X + X\log(I+B)|| \le ||AX + XB||$$

for any unitarily invariant norm $||\cdot||$.

It is interesting to see that along the above line of argument by taking $f = x^p$ $(0 < p \le 1)$ and $h = 0$ in Lemma 3, we get another proof of the following well-known Heinz inequality (e.g., [3]): let $A, B \ge 0$ and X be arbitrary. Then

$$||A^rXB^{1-r} + A^{1-r}XB^r|| \le ||AX + XB||$$

for any unitarily invariant norm and $0 \le r \le 1$.

REFERENCES

[1] T. Ando, R. A. Horn, and C. R. Johnson, *The singular values of a Hadamard product: A basic inequality*, Linear and Multilinear Algebra, 21 (1987), pp. 345–365.

[2] E. Andruchow, G. Corach, and D. Stojanoff, *Geometric operator inequalities*, Linear Algebra Appl., 258 (1997), pp. 295–310.

[3] R. Bhatia and C. Davis, *More matrix forms of the arithmetic-geometric mean inequality*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 132–136.

[4] R. Bhatia and K. R. Parthasarathy, *Positive Definite Functions and Operator Inequalities*, preprint, 1997.

[5] T. Furuta, *A note on the arithmetic-geometric mean inequality for every unitarily invariant matrix norm*, Linear Algebra Appl., 208/209 (1994), pp. 223–228.

[6] R. A. Horn, *Norm bounds for Hadamard products and an arithmetic-geometric mean inequality for unitarily invariant norms*, Linear Algebra Appl., 223/224 (1995), pp. 355–361.

[7] F. Kittaneh, *A note on the arithmetic-geometric mean inequality for matrices*, Linear Algebra Appl., 171 (1992), pp. 1–8.

[8] M. K. Kwong, *On the definiteness of the solutions of certain matrix equations*, Linear Algebra Appl., 108 (1988), pp. 177–197.

[9] M. K. Kwong, *Some results on matrix monotone functions*, Linear Algebra Appl., 118 (1989), pp. 129–153.

[10] R. Mathias, *An arithmetic-geometric-harmonic mean inequality involving Hadamard products*, Linear Algebra Appl., 184 (1993), pp. 71–78.

[11] I. Schur, *Bemerkungen zur Theorie der beschränkten Bilinearformen mit unendlich vielen Veränderlichen*, J. Reine Angew. Math., 140 (1911), pp. 1–28.

SIAM J. MATRIX ANAL. APPL.
Vol. 20, No. 2, pp. 471–492

RELATIVE PERTURBATION THEORY: II. EIGENSPACE AND SINGULAR SUBSPACE VARIATIONS[*]

REN-CANG LI[†]

Abstract. The classical perturbation theory for Hermitian matrix eigenvalue and singular value problems provides bounds on invariant subspace variations that are proportional to the reciprocals of *absolute gaps* between subsets of spectra or subsets of singular values. These bounds may be bad news for invariant subspaces corresponding to clustered eigenvalues or clustered singular values of much smaller magnitudes than the norms of matrices under considerations. In this paper, we consider how eigenspaces of a Hermitian matrix A change when it is perturbed to $\widetilde{A} = D^*AD$ and how singular spaces of a (nonsquare) matrix B change when it is perturbed to $\widetilde{B} = D_1^*BD_2$, where D, D_1, and D_2 are nonsingular. It is proved that under these kinds of perturbations, the changes of invariant subspaces are proportional to the reciprocals of *relative gaps* between subsets of spectra or subsets of singular values. The classical Davis–Kahan $\sin\theta$ theorems and Wedin $\sin\theta$ theorems are extended.

Key words. multiplicative perturbation, relative perturbation theory, relative gap, eigenvector, singular vector, structured Sylvester equation, graded matrix

AMS subject classifications. 15A18, 15A42, 65F15, 65F35, 65G99

PII. S0895479896298506

1. Introduction.

Let A and $\widetilde{A}$ be two $n \times n$ Hermitian matrices with eigendecompositions

$$A = (U_1, U_2) \begin{pmatrix} \Lambda_1 & \\ & \Lambda_2 \end{pmatrix} \begin{pmatrix} U_1^* \\ U_2^* \end{pmatrix} \quad \text{and} \quad \widetilde{A} = (\widetilde{U}_1, \widetilde{U}_2) \begin{pmatrix} \widetilde{\Lambda}_1 & \\ & \widetilde{\Lambda}_2 \end{pmatrix} \begin{pmatrix} \widetilde{U}_1^* \\ \widetilde{U}_2^* \end{pmatrix}, \tag{1.1}$$

where $U = \begin{array}{c} \begin{array}{cc} {\scriptstyle k} & {\scriptstyle n-k} \end{array} \\ \begin{pmatrix} U_1 & U_2 \end{pmatrix} \end{array}$, $\widetilde{U} = \begin{array}{c} \begin{array}{cc} {\scriptstyle k} & {\scriptstyle n-k} \end{array} \\ \begin{pmatrix} \widetilde{U}_1 & \widetilde{U}_2 \end{pmatrix} \end{array}$ are unitary, and

$$\Lambda_1 = \operatorname{diag}(\lambda_1, \ldots, \lambda_k), \quad \Lambda_2 = \operatorname{diag}(\lambda_{k+1}, \ldots, \lambda_n), \tag{1.2}$$

$$\widetilde{\Lambda}_1 = \operatorname{diag}(\widetilde{\lambda}_1, \ldots, \widetilde{\lambda}_k), \quad \widetilde{\Lambda}_2 = \operatorname{diag}(\widetilde{\lambda}_{k+1}, \ldots, \widetilde{\lambda}_n). \tag{1.3}$$

Suppose now that A and $\widetilde{A}$ are *close* and that the spectrum of Λ_1 and that of $\widetilde{\Lambda}_2$ (or the spectrum of $\widetilde{\Lambda}_1$ and that of Λ_2) are *well separated*. The question is, *How*

[*] Received by the editors February 2, 1996; accepted for publication (in revised form) by B. Kagstrom April 29, 1998; published electronically November 23, 1998. Part of this work was done during the summer of 1994 while the author was at the Department of Mathematics, University of California at Berkeley. A preliminary version of this paper appeared as Technical Report UBC//CSD-94-856, Computer Science Division, Department of EECS, University of California at Berkeley. This material is based in part upon work supported by Argonne National Laboratory under grant 20552402, the University of Tennessee through the Advanced Research Projects Agency under contract DAAL03-91-C-0047, the National Science Foundation under grant ASC-9005933, the National Science Infrastructure grants CDA-8722788 and CDA-9401156, a Householder Fellowship in Scientific Computing at Oak Ridge National Laboratory, and the Applied Mathematical Sciences Research Program, Office of Energy Research, United States Department of Energy contract DE-AC05-96OR22464 with Lockheed Martin Energy Research Corp.

http://www.siam.org/journals/simax/20-2/29850.html

[†] Mathematical Science Section, Oak Ridge National Laboratory, P.O. Box 2008, Bldg. 6012, Oak Ridge, TN 37831-6367. Current address: Department of Mathematics, University of Kentucky, Lexington, KY 40506 (rcli@ms.uky.edu).

close are the eigenspaces spanned by the columns of U_i and $\widetilde{U}_i$? This question has been answered well by four celebrated theorems: the so-called $\sin\theta$, $\tan\theta$, $\sin 2\theta$, and $\tan 2\theta$ theorems due to Davis and Kahan [3] for arbitrary additive perturbations in the sense that the perturbations to A can be made arbitrary. It is proved that the changes of invariant subspaces are proportional to the reciprocals of *absolute gaps* between subsets of spectra.

In the case of multiplicative perturbations when A is perturbed to $\widetilde{A} = D^*AD$, Eisenstat and Ipsen [7] first attacked the question by bounding the angles between a one-dimensional eigenspace of A and $\widetilde{A}$'s eigenspace spanned by the columns of $\widetilde{U}_1$, and they ultimately obtained bounds for the angle between A's eigenspace spanned by the columns of U_1 and $\widetilde{A}$'s eigenspace spanned by the columns of $\widetilde{U}_1$. The study suggests that the changes of invariant subspaces be proportional to the reciprocals of *relative gaps* between subsets of spectra.

This paper will study the same question, but using a different approach. It is explained that bounding the angle between the eigenspaces is related to bounding the solutions to *Sylvester equations* $\Omega X - X\Gamma = S$, where S has very special structures. Our approach is more or less along the lines of Davis and Kahan [3] and Bhatia, Davis, and McIntosh [2], where no special structures for S are known. There are a number of advantages of the new approach over Eisenstat and Ipsen's. The new approach can deal directly with an eigenspace and its perturbed one, unlike Eisenstat and Ipsen's approach, and consequently gets rid of the unpleasant fact $\sqrt{k}$ in Eisenstat and Ipsen's bounds; the new approach makes no distinctions in treating an eigenvector and an eigenspace and two eigenspaces of the same or different dimensions; the new approach is mathematically more elegant and makes it possible to extend Davis–Kahan theorems in all unitarily invariant norms.

A similar question for *singular value decompositions* will be answered also.

Although special, multiplicative perturbations cover component-wise relative perturbations to entries of symmetric tridiagonal matrices with zero diagonal [5, 9], entries of bidiagonal and biacyclic matrices [1, 4, 5], and more *realistically* perturbations in graded nonnegative definite Hermitian matrices [6, 14] and in graded matrices of singular value problems [6, 14] and more [8].

The rest of this paper is organized as follows. Section 2 serves two purposes: to present essential preliminary definitions and lemmas, and to briefly discuss technical similarities and differences between our extensions and the classical Davis–Kahan theorems. Section 3 details relative perturbation theorems for A and $\widetilde{A} = D^*AD$ and for nonnegative definite Hermitian matrices A that themselves may be very ill conditioned but can be scaled to a well-conditioned one. Section 3 also remarks on how to extend our approach to the perturbations of diagonalizable matrices. Section 4 develops analogous relative perturbation theorems but for the singular value problems. The proofs for theorems in section 4 turn out to be quite long and therefore are postponed to section 5.

2. Preliminaries.

Throughout this paper, we follow notation used in the first part of this series [12].

2.1. Relative distances.

We shall use the following two kinds of relative distances to measure relative accuracies in numerical approximations: ϱ_p and χ are defined for α, $\widetilde{\alpha} \in \mathbb{C}$ by

$$(2.1)\qquad \varrho_p(\alpha, \widetilde{\alpha}) = \frac{|\alpha - \widetilde{\alpha}|}{\sqrt[p]{|\alpha|^p + |\widetilde{\alpha}|^p}} \quad \text{for } 1 \le p \le \infty, \text{ and } \quad \chi(\alpha, \widetilde{\alpha}) = \frac{|\alpha - \widetilde{\alpha}|}{\sqrt{|\alpha\widetilde{\alpha}|}},$$

with convention $0/0 = 0$ for convenience. Both have better mathematical properties than the classical measurement $|\delta|$: the relative error in $\widetilde{\alpha} = \alpha(1+\delta)$ as an approximation to α is

$$\delta = \text{relative error in } \widetilde{\alpha} = \frac{\widetilde{\alpha} - \alpha}{\alpha}, \tag{2.2}$$

which is, however, good enough and actually more convenient to use in numerical computations. So we shall also present bounds using this classical measurement. The use of any particular relative distance in our perturbation bounds comes naturally with their derivations. From the numerical point of view, any one of the relative distances is just as good as the others, but theoretically they provide bounds of very different features. It can be proved that these relative distances are topologically equivalent; see [12] for details.

2.2. Angles between two subspaces. Since this paper concerns the variations of subspaces, we need some metrics to measure the differences between two subspaces. In this, we follow Davis and Kahan [3] and Stewart and Sun [15, Chapters I and II]. Wedin [17] presented an illuminating discussion on angles between two subspaces, too. Let X, $\widetilde{X} \in \mathbb{C}^{n\times k}$ $(n > k)$ have full column rank k, and define the angle matrix $\Theta(X, \widetilde{X})$ between X and $\widetilde{X}$ as

$$\Theta(X, \widetilde{X}) \overset{\text{def}}{=} \arccos((X^*X)^{-1/2}X^*\widetilde{X}(\widetilde{X}^*\widetilde{X})^{-1}\widetilde{X}^*X(X^*X)^{-1/2})^{1/2}. \tag{2.3}$$

The *canonical angles* between the subspaces $\mathcal{X} = \mathcal{R}(X)$ and $\widetilde{\mathcal{X}} = \mathcal{R}(\widetilde{X})$ are defined to be the singular values of the Hermitian matrix $\Theta(X, \widetilde{X})$, where $\mathcal{R}(X)$ denotes the subspace spanned by X's columns. The following lemma is well known. For a proof of it, the reader is referred to, e.g., Li [11, Lemma 2.1].

LEMMA 2.1. *Suppose that* $\begin{array}{c} \begin{array}{cc} \scriptstyle k & \scriptstyle n-k \end{array} \\ \begin{pmatrix} \widetilde{X} & \widetilde{X}_1 \end{pmatrix} \end{array} \in \mathbb{C}^{n\times n}$ *is a nonsingular matrix, and partition*

$$(\widetilde{X}, \widetilde{X}_1)^{-1} = \begin{array}{c} \scriptstyle k \\ \scriptstyle n-k \end{array} \begin{pmatrix} \widetilde{Y}^* \\ \widetilde{Y}_1^* \end{pmatrix}.$$

Then for any unitarily invariant norm $|\!|\!| \cdot |\!|\!|$,

$$\left|\!\left|\!\left| \sin\Theta(X, \widetilde{X}) \right|\!\right|\!\right| = \left|\!\left|\!\left| (\widetilde{Y}_1^*\widetilde{Y}_1)^{-1/2}\widetilde{Y}_1^*X(X^*X)^{-1/2} \right|\!\right|\!\right|.$$

In this lemma, as well as many other places in the rest of this paper, we talk about the "same" unitarily invariant norm $|\!|\!| \cdot |\!|\!|$ that applies to matrices of different dimensions at the same time. Such applications of a unitarily invariant norm are understood in the following sense: first there is a unitarily invariant norm $|\!|\!| \cdot |\!|\!|$ on $\mathbb{C}^{M\times N}$ for sufficiently large integers M and N; then, for a matrix $X \in \mathbb{C}^{m\times n}$ $(m \le M$ and $n \le N)$, $|\!|\!| X |\!|\!|$ is defined by appending X with zero blocks to make it $M \times N$ and then taking the unitarily invariant norm of the enlarged matrix.

Taking $X = U_1$ and $\widetilde{X} = \widetilde{U}_1$ as in (1.1), by Lemma 2.1, one has

$$\Theta(U_1, \widetilde{U}_1) = \arccos(U_1^*\widetilde{U}_1\widetilde{U}_1^*U_1)^{1/2} \quad \text{and} \quad \left|\!\left|\!\left| \sin\Theta(U_1, \widetilde{U}_1) \right|\!\right|\!\right| = \left|\!\left|\!\left| \widetilde{U}_2^*U_1 \right|\!\right|\!\right|. \tag{2.4}$$

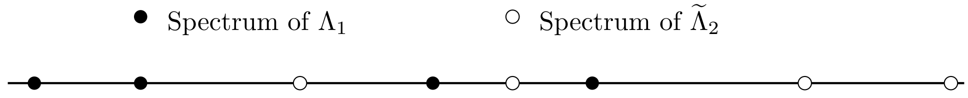

● Spectrum of Λ_1
○ Spectrum of $\widetilde{\Lambda}_2$

FIG. 2.1. *The spectrum of* Λ_1 *and that of* $\widetilde{\Lambda}_2$ *are disjoint.*

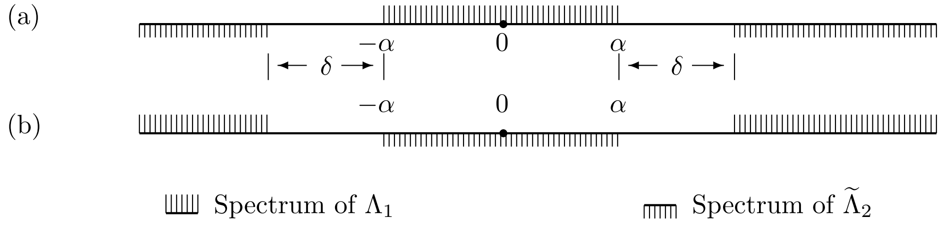


FIG. 2.2. *The spectrum of* Λ_1 *and that of* $\widetilde{\Lambda}_2$ *are separated by two intervals, and one of the spectra scatters around the origin.*

Occasionally, it is also of interest to measure how far a lower-dimensional subspace is away from a higher-dimensional one. In such a situation, an angle matrix can still be defined as in (2.3), but with X having *fewer* columns than $\widetilde{X}$. If we do so, Lemma 2.1 remains valid. However, $\Theta(\,\cdot\,,\,\cdot\,)$ is no longer symmetric with respect to its arguments. In the case of eigenspace variation in which we are interested we may take $U_{1,\mathrm{sub}}$, a submatrix consisting of a few (or just one) of U_1's columns and ask how close A's eigenspace $\mathcal{R}(U_{1,\mathrm{sub}})$ is to $\widetilde{A}$'s eigenspace $\mathcal{R}(\widetilde{U}_1)$. Still, we have

$$\Theta(U_{1,\mathrm{sub}}, \widetilde{U}_1) = \arccos(U_{1,\mathrm{sub}}^* \widetilde{U}_1 \widetilde{U}_1^* U_{1,\mathrm{sub}})^{1/2} \text{ and} \tag{2.4a}$$

$$\left|\!\left|\!\left| \sin\Theta(U_{1,\mathrm{sub}}, \widetilde{U}_1) \right|\!\right|\!\right| = \left|\!\left|\!\left| \widetilde{U}_2^* U_{1,\mathrm{sub}} \right|\!\right|\!\right| .$$

2.3. Separation of spectra. In deriving bounds on the sines of the angles between the eigenspaces $\mathcal{R}(U_1)$ and $\mathcal{R}(\widetilde{U}_1)$, certain disjointness between Λ_1's spectrum and $\widetilde{\Lambda}_2$'s (or between $\widetilde{\Lambda}_1$'s and Λ_2's) is assumed. Depending on what matrix norms are used, two different kinds of separations, one stronger than the other, are considered. For bounds in Frobenius norm, only the disjointness is required as in Davis and Kahan [3]; see Figure 2.1. For bounds in all unitarily invariant norms, not only is the disjointness between the two spectra required, but they also have to be separated by two intervals; see Figure 2.2. Such a separation requirement is similar to Davis and Kahan's $\sin\theta$ theorems in all unitarily invariant norms, but it differs from theirs in that here either the spectrum of Λ_1 or that of $\widetilde{\Lambda}_2$ has to scatter around the origin. This substantial difference exists for a reason. In the absolute perturbation theory, shifting A and $\widetilde{A}$ by a scalar μ to $A - \mu I$ and $\widetilde{A} - \mu I$ retains every relevant object—eigenspaces, the residuals such as $R = \widetilde{A}U_1 - U_1\Lambda_1$ and most of all the absolute gap δ, and thus the positions of the intervals in Davis and Kahan's assumptions are not intrinsically important. For our relative case, the emphasis is on issues associated with eigenvalues of relatively (much) smaller magnitudes than the norm of the matrix, and shifting affects fundamentally the underlying properties of the problem.

Classical Davis–Kahan theorems use the absolute gap—the minimum distance—between $\lambda(\Lambda_1)$ and $\lambda(\widetilde{\Lambda}_2)$; our relative perturbation theorems, however, will use relative gaps measured by any one of the relative "*distances*" (2.1) and (2.2). We shall use notation η_p, η_χ, and η_{c} to denote three different kinds of relative gaps defined in terms of ϱ_p, χ, and the *c*lassical measurement in the case of Figure 2.1; we underline the η's to indicate the stronger separation by intervals as in Figure 2.2. The use of different relative gaps comes quite naturally with different perturbation equations, which yields various bounds on the sines of the angles. It appears that η_χ and $\underline{\eta}_\chi$ are natural choices for nonnegative definite Hermitian matrices, and bounds that use them are normally sharper than bounds that use other kinds of relative gaps. η_p, $\eta_{\rm c}$, and their underlined ones are natural choices for any Hermitian matrices, and bounds that use them are comparable as we shall see in later sections; but mathematically, bounds with η_p are more beautiful because it is defined in terms of ϱ_p, which is a metric on $\mathbb{R}$ [12] and thus treats Λ_1 and $\widetilde{\Lambda}_2$ equally, while $\eta_{\rm c}$ is perhaps more convenient to use in actually numerical approximations.

All our perturbation bounds in this paper use relative gaps mentioned above. However, sometimes it is more convenient to have bounds that use relative gaps between $\lambda(\Lambda_1)$ and $\lambda(\Lambda_2)$ rather than $\lambda(\widetilde{\Lambda}_2)$. For this purpose, Li [13] presented inequality relations between relative gaps for $\lambda(\Lambda_1)$ and $\lambda(\Lambda_2)$ and those for $\lambda(\Lambda_1)$ and $\lambda(\widetilde{\Lambda}_2)$.

Finally, we remark that if we are interested in how close A's eigenspace (spanned by a few (not all) columns of U_1) is to $\widetilde{A}$'s eigenspace $\mathcal{R}(\widetilde{U}_1)$, spectrum separation assumptions will then be required only between the subset of $\lambda(\Lambda_1)$ corresponding to those selected columns of U_1 and the spectrum of $\widetilde{\Lambda}_2$.

2.4. On Sylvester equations $\Omega X - X\Gamma = S$ with structured S. This subsection illustrates another technical *similarity* and *difference* of our development of relative perturbation theory to Davis and Kahan's classical development [3], where Ω and Γ are two self-adjoint operators, and S is an *arbitrary* operator in certain norm ideals. In our case, however, S takes one of the forms

$$\Omega E + F\Gamma \quad \text{and} \quad \Omega^{1/2}E\Gamma^{1/2}.$$

In what follows, we shall try to exploit the situations, and by doing so, we are able to derive better bounds on the solution X.

LEMMA 2.2. *Let $\Omega \in \mathbb{C}^{s\times s}$ and $\Gamma \in \mathbb{C}^{t\times t}$ be two Hermitian matrices,*[1] *and let E, $F \in \mathbb{C}^{s\times t}$. If $\lambda(\Omega) \bigcap \lambda(\Gamma) = \emptyset$, then $\Omega X - X\Gamma = \Omega E + F\Gamma$ has a unique solution $X \in \mathbb{C}^{s\times t}$, and moreover,*

$$\|X\|_{\rm F} \le \sqrt{\|E\|_{\rm F}^2 + \|F\|_{\rm F}^2} \Big/ \eta_2,$$

where $\eta_2 \overset{\rm def}{=} \min_{\omega\in\lambda(\Omega),\, \gamma\in\lambda(\Gamma)} \varrho_2(\omega,\gamma)$. *If, in addition, $F = 0$, we have a better bound*

$$\|X\|_{\rm F} \le \|E\|_{\rm F}/\eta_{\rm c},$$

where[2] $\eta_{\rm c} \overset{\rm def}{=} \min_{\omega\in\lambda(\Omega),\, \gamma\in\lambda(\Gamma)} |\omega-\gamma|/|\omega|$.

Proof. For any $s\times s$ unitary P and $t\times t$ unitary Q, the substitutions

$$\Omega \leftarrow P^*\Omega P, \quad \Gamma \leftarrow Q^*\Gamma Q, \quad X \leftarrow P^*XQ, \quad E \leftarrow P^*EQ, \quad \text{and} \quad F \leftarrow P^*FQ$$

[1] Lemmas 2.2–2.5 are actually true for *normal* matrices Ω and Γ.

[2] Notice that $\eta_{\rm c} \ge \eta_2$. This can be seen as follows. Assume $\eta_{\rm c} = |\omega-\gamma|/|\omega|$ for some $\omega \in \lambda(\Omega)$, $\gamma \in \lambda(\Gamma)$. Then $\eta_{\rm c} \ge \varrho_2(\omega,\gamma) \ge \eta_2$.

leave the lemma unchanged, so we may assume without loss of generality that $\Omega = \mathrm{diag}(\omega_1, \omega_2, \ldots, \omega_s)$ and $\Gamma = \mathrm{diag}(\gamma_1, \gamma_2, \ldots, \gamma_t)$.

Write $X = (x_{ij})$, $E = (e_{ij})$, and $F = (f_{ij})$. Entrywise, $\Omega X - X\Gamma = \Omega E + F\Gamma$ reads $\omega_i x_{ij} - x_{ij}\gamma_j = \omega_i e_{ij} + f_{ij}\gamma_j$. Thus x_{ij} exists uniquely provided $\omega_i \neq \gamma_j$, which holds since $\lambda(\Omega) \bigcap \lambda(\Gamma) = \emptyset$, an empty set; and moreover;

$$|(\omega_i - \gamma_j)x_{ij}|^2 = |\omega_i x_{ij} - x_{ij}\gamma_j|^2 = |\omega_i e_{ij} + f_{ij}\gamma_j|^2 \leq (|\omega_i|^2 + |\gamma_j|^2)(|e_{ij}|^2 + |f_{ij}|^2)$$

by the Cauchy–Schwarz inequality. This implies

$$\begin{aligned} |x_{ij}|^2 &\leq \frac{|e_{ij}|^2 + |f_{ij}|^2}{[\varrho_2(\omega_i, \gamma_j)]^2} \leq \frac{|e_{ij}|^2 + |f_{ij}|^2}{\eta_2^2} \\ \Rightarrow \|X\|_{\mathrm{F}}^2 = \sum_{i,\,j} |x_{ij}|^2 &\leq \frac{\sum_{i,\,j} |e_{ij}|^2 + \sum_{i,\,j} |f_{ij}|^2}{\eta_2^2} = \frac{\|E\|_{\mathrm{F}}^2 + \|F\|_{\mathrm{F}}^2}{\eta_2^2}, \end{aligned}$$

as was to be shown. The case $F = 0$ can be handled similarly. □

LEMMA 2.3. *Let $\Omega \in \mathbb{C}^{s\times s}$ and $\Gamma \in \mathbb{C}^{t\times t}$ be two Hermitian matrices, and let E, $F \in \mathbb{C}^{s\times t}$. If there exist $\alpha \geq 0$ and $\delta > 0$ such that*

$$\|\Omega\|_2 \leq \alpha \quad \textit{and} \quad \|\Gamma^{-1}\|_2^{-1} \geq \alpha + \delta \tag{2.5}$$

or

$$\|\Omega^{-1}\|_2^{-1} \geq \alpha + \delta \quad \textit{and} \quad \|\Gamma\|_2 \leq \alpha, \tag{2.6}$$

then $\Omega X - X\Gamma = \Omega E + F\Gamma$ has a unique solution $X \in \mathbb{C}^{s\times t}$, and moreover for any unitarily invariant norm $|||\cdot|||$,

$$|||X||| \leq \sqrt[q]{|||E|||^q + |||F|||^q} \Big/ \underline{\eta}_p,$$

where $\underline{\eta}_p \overset{\mathrm{def}}{=} \varrho_p(\alpha, \alpha + \delta)$. If, in addition, $F = 0$, we have a better bound

$$|||X||| \leq |||E||| / \underline{\eta}_{\mathrm{c}},$$

where $\underline{\eta}_{\mathrm{c}} = \delta/\alpha$ when (2.5) holds, and $\underline{\eta}_{\mathrm{c}} = \delta/(\alpha + \delta)$ when (2.6) holds.

Proof. First of all, the conditions of this lemma imply $\lambda(\Omega) \bigcap \lambda(\Gamma) = \emptyset$, thus X exists uniquely by Lemma 2.2. In what follows, we consider the case (2.5); the other case (2.6) is analogous. Post-multiply $\Omega X - X\Gamma = \Omega E + F\Gamma$ by Γ^{-1} to get

$$\Omega X \Gamma^{-1} - X = \Omega E \Gamma^{-1} + F. \tag{2.7}$$

Under the assumptions $\|\Omega\|_2 \leq \alpha$ and $\|\Gamma^{-1}\|_2^{-1} \geq \alpha + \delta \Rightarrow \|\Gamma^{-1}\|_2 \leq \frac{1}{\alpha+\delta}$, we have

$$\begin{aligned} \left|\left|\left|\Omega X \Gamma^{-1} - X\right|\right|\right| &\geq |||X||| - \left|\left|\left|\Omega X \Gamma^{-1}\right|\right|\right| \geq |||X||| - \|\Omega\|_2 \, |||X||| \, \|\Gamma^{-1}\|_2 \\ &\geq |||X||| - \alpha \, |||X||| \, \frac{1}{\alpha + \delta} = \left(1 - \frac{\alpha}{\alpha + \delta}\right) |||X||| \,, \end{aligned}$$

and

$$\begin{aligned} \left|\left|\left|\Omega E \Gamma^{-1} + F\right|\right|\right| &\leq \left|\left|\left|\Omega E \Gamma^{-1}\right|\right|\right| + |||F||| \leq \|\Omega\|_2 \, |||E||| \, \|\Gamma^{-1}\|_2 + |||F||| \\ &\leq \alpha \, |||E||| \, \frac{1}{\alpha + \delta} + |||F||| \leq \sqrt[p]{1 + \frac{\alpha^p}{(\alpha + \delta)^p}} \sqrt[q]{|||E|||^q + |||F|||^q}. \end{aligned}$$

By (2.7), we deduce that

$$\left(1 - \frac{\alpha}{\alpha+\delta}\right) \|\!|X|\!\| \le \sqrt[p]{1 + \frac{\alpha^p}{(\alpha+\delta)^p}}\,\sqrt[q]{\|\!|E|\!\|^q + \|\!|F|\!\|^q}$$

from which the desired inequality follows. Also, the case $F = 0$ can be handled similarly. □

Lemma 2.4. *Let $\Omega \in \mathbb{C}^{s\times s}$ and $\Gamma \in \mathbb{C}^{t\times t}$ be two nonnegative definite Hermitian matrices, and let $E \in \mathbb{C}^{s\times t}$. If $\lambda(\Omega)\bigcap\lambda(\Gamma) = \emptyset$, then $\Omega X - X\Gamma = \Omega^{1/2}E\Gamma^{1/2}$ has a unique solution $X \in \mathbb{C}^{s\times t}$, and moreover,*

$$\|X\|_{\mathrm{F}} \le \|E\|_{\mathrm{F}}/\eta_\chi,$$

where $\eta_\chi \overset{\mathrm{def}}{=} \min_{\omega\in\lambda(\Omega),\,\gamma\in\lambda(\Gamma)} \chi(\omega,\gamma)$.

Proof. For any $s \times s$ unitary P and $t \times t$ unitary Q, the substitutions

$$\Omega \leftarrow P^*\Omega P, \quad \Gamma \leftarrow Q^*\Gamma Q, \quad X \leftarrow P^*XQ, \quad \text{and} \quad E \leftarrow P^*EQ$$

leave the lemma unchanged, so we may assume without loss of generality that $\Omega = \mathrm{diag}(\omega_1, \omega_2, \ldots, \omega_s)$ and $\Gamma = \mathrm{diag}(\gamma_1, \gamma_2, \ldots, \gamma_t)$.

Write $X = (x_{ij})$, $E = (e_{ij})$. Entrywise, $\Omega X - X\Gamma = \Omega^{1/2}E\Gamma^{1/2}$ reads $\omega_i x_{ij} - x_{ij}\gamma_j = \sqrt{\omega_i}e_{ij}\sqrt{\gamma_j}$. As long as $\omega_i \ne \gamma_j$, x_{ij} exists uniquely, and

$$|x_{ij}|^2 = |e_{ij}|^2/\chi(\omega_i,\gamma_j) \le |e_{ij}|^2/\eta_\chi,$$

summing which over $1 \le i \le s$ and $1 \le j \le t$ leads to the desired inequality. □

Lemma 2.5. *Let $\Omega \in \mathbb{C}^{s\times s}$ and $\Gamma \in \mathbb{C}^{t\times t}$ be two nonnegative definite Hermitian matrices, and let $E \in \mathbb{C}^{s\times t}$. If there exist $\alpha \ge 0$ and $\delta > 0$ such that (2.5) or (2.6) holds, then $\Omega X - X\Gamma = \Omega^{1/2}E\Gamma^{1/2}$ has a unique solution $X \in \mathbb{C}^{s\times t}$, and moreover,*

$$\|\!|X|\!\| \le \|\!|E|\!\|/\underline{\eta}_\chi,$$

where $\underline{\eta}_\chi \overset{\mathrm{def}}{=} \chi(\alpha, \alpha+\delta)$.

Proof. The existence and uniqueness of X are easy to see because the conditions of this lemma imply $\lambda(\Omega)\bigcap\lambda(\Gamma) = \emptyset$. In what follows, we consider the case (2.5) only; the other case (2.6) is analogous. Post-multiply $\Omega X - X\Gamma = \Omega^{1/2}E\Gamma^{1/2}$ by Γ^{-1} to get

$$\Omega X\Gamma^{-1} - X = \Omega^{1/2}E\Gamma^{-1/2}. \tag{2.8}$$

Under the assumptions $\|\Omega\|_2 \le \alpha$ and $\|\Gamma^{-1}\|_2^{-1} \ge \alpha + \delta \Rightarrow \|\Gamma^{-1}\|_2 \le \frac{1}{\alpha+\delta}$, we have

$$\left\|\!\left|\Omega X\Gamma^{-1} - X\right|\!\right\| \ge \left(1 - \frac{\alpha}{\alpha+\delta}\right)\|\!|X|\!\|,$$

as in the proof of Lemma 2.3, and

$$\left\|\!\left|\Omega^{1/2}E\Gamma^{-1/2}\right|\!\right\| \le \|\Omega^{1/2}\|_2\,\|\!|E|\!\|\,\|\Gamma^{-1/2}\|_2 \le \sqrt{\alpha}\,\|\!|E|\!\|\,\frac{1}{\sqrt{\alpha+\delta}}.$$

By (2.8), we deduce that

$$\left(1 - \frac{\alpha}{\alpha+\delta}\right)\|\!|X|\!\| \le \sqrt{\frac{\alpha}{\alpha+\delta}}\,\|\!|E|\!\|,$$

from which the desired inequality follows. □

Remark 2.1. For Sylvester equation $\Omega X - X\Gamma = S$, with S having no special structures, Bhatia, Davis, and McIntosh [2] also proved bounds, independent of X's dimensions, on $\|\!|X|\!\|$ under the conditions that Ω and Γ are normal and $\lambda(\Omega)\bigcap\lambda(\Gamma) = \emptyset$ only. It is easy to see that (2.5) or (2.6) describes similar spectral distributions to Figure 2.2. Thus an open question naturally arises: *could Lemmas* 2.3 *and* 2.5 *be extended to normal matrices* Ω *and* Γ *and* $\lambda(\Omega)\bigcap\lambda(\Gamma) = \emptyset$ *only?*

3. Relative perturbation theorems for eigenspace variations. Let A and $\widetilde{A}$ be two Hermitian matrices whose eigendecompositions are

$$A = (U_1, U_2)\begin{pmatrix} \Lambda_1 & \\ & \Lambda_2 \end{pmatrix}\begin{pmatrix} U_1^* \\ U_2^* \end{pmatrix} \text{ and } \widetilde{A} = (\widetilde{U}_1, \widetilde{U}_2)\begin{pmatrix} \widetilde{\Lambda}_1 & \\ & \widetilde{\Lambda}_2 \end{pmatrix}\begin{pmatrix} \widetilde{U}_1^* \\ \widetilde{U}_2^* \end{pmatrix}, \tag{1.1}$$

where $U = \begin{pmatrix} \overset{k}{U_1} & \overset{n-k}{U_2} \end{pmatrix}$, $\widetilde{U} = \begin{pmatrix} \overset{k}{\widetilde{U}_1} & \overset{n-k}{\widetilde{U}_2} \end{pmatrix}$ are unitary, and Λ_i's and $\widetilde{\Lambda}_j$'s are defined as in (1.2) and (1.3). Define

$$R = \widetilde{A}U_1 - U_1\Lambda_1 = (\widetilde{A} - A)U_1.$$

Notice that

$$\begin{aligned} \widetilde{U}_2^* R &= \widetilde{U}_2^* \widetilde{A} U_1 - \widetilde{U}_2^* U_1 \Lambda_1 = \widetilde{\Lambda}_2 \widetilde{U}_2^* U_1 - \widetilde{U}_2^* U_1 \Lambda_1, \qquad \text{and} \\ \widetilde{U}_2^* R &= \widetilde{U}_2^* \left[D^* A D (I - D^{-1}) + (D^* - I)A \right] U_1 \\ &= \widetilde{\Lambda}_2 \widetilde{U}_2^* (I - D^{-1}) U_1 + \widetilde{U}_2^* (D^* - I) U_1 \Lambda_1. \end{aligned}$$

Thus, we have

$$\widetilde{\Lambda}_2 \widetilde{U}_2^* U_1 - \widetilde{U}_2^* U_1 \Lambda_1 = \widetilde{\Lambda}_2 \widetilde{U}_2^* (I - D^{-1}) U_1 + \widetilde{U}_2^* (D^* - I) U_1 \Lambda_1. \tag{3.1}$$

Let $X \overset{\text{def}}{=} \widetilde{U}_2^* D^* U_1 = \widetilde{U}_2^* U_1 - \widetilde{U}_2^* (I - D^*) U_1$. Another formulation[3] of (3.1) is

$$\widetilde{\Lambda}_2 X - X \Lambda_1 = \widetilde{\Lambda}_2 \widetilde{U}_2^* (D^* - D^{-1}) U_1. \tag{3.2}$$

Both (3.1) and (3.2) are in the form of Sylvester equations with special structures which are vital to the development of our following perturbation theorems. Notice by Lemma 2.1 that

$$\left\|\!\left| \sin\Theta(U_1, \widetilde{U}_1) \right|\!\right\| = \left\|\!\left| \widetilde{U}_2^* U_1 \right|\!\right\| \le \|\!|X|\!\| + \left\|\!\left| \widetilde{U}_2^* (I - D^*) U_1 \right|\!\right\|. \tag{3.3}$$

(3.1) makes ϱ_2 a natural choice for measuring the relative gap between $\lambda(\Lambda_1)$ and $\lambda(\widetilde{\Lambda}_2)$, while (3.2) makes the classical measurement a natural choice.

THEOREM 3.1. *Let A and $\widetilde{A} = D^*AD$ be two $n \times n$ Hermitian matrices with eigendecompositions* (1.1)–(1.3)*, where D is nonsingular. If $\lambda(\Lambda_1)\bigcap\lambda(\widetilde{\Lambda}_2) = \emptyset$, then*

$$\|\sin\Theta(U_1, \widetilde{U}_1)\|_{\mathrm{F}} \le \frac{\sqrt{\|(I - D^{-1})U_1\|_{\mathrm{F}}^2 + \|(I - D^*)U_1\|_{\mathrm{F}}^2}}{\eta_2}, \tag{3.4}$$

$$\|\sin\Theta(U_1, \widetilde{U}_1)\|_{\mathrm{F}} \le \|(I - D^*)U_1\|_{\mathrm{F}} + \frac{\|(D^* - D^{-1})U_1\|_{\mathrm{F}}}{\eta_{\mathrm{c}}}, \tag{3.5}$$

[3] Or, $\widetilde{\Lambda}_2 Y - Y\Lambda_1 = \widetilde{U}_2^*(D^* - D^{-1})U_1\Lambda_1$, where $Y = \widetilde{U}_2^* U_1 - \widetilde{U}_2^*(I - D^{-1})U_1$. Such a formulation can be used to produce perturbation bounds different from but of the same spirit as (3.5) and (3.8) whose derivations rely on (3.2).

where

$$\eta_2 \stackrel{\text{def}}{=} \min_{\mu\in\lambda(\Lambda_1),\,\widetilde{\mu}\in\lambda(\widetilde{\Lambda}_2)} \varrho_2(\mu,\widetilde{\mu}) \quad \textit{and} \quad \eta_{\text{c}} \stackrel{\text{def}}{=} \min_{\mu\in\lambda(\Lambda_1),\,\widetilde{\mu}\in\lambda(\widetilde{\Lambda}_2)} \frac{|\mu-\widetilde{\mu}|}{|\widetilde{\mu}|}.$$

Proof. Lemma 2.2 and (3.1) yield (3.4), whereas Lemma 2.2, (3.2), and (3.3) yield (3.5). □

Remark 3.1. Without assuming the multiplicative structure in perturbing A to $\widetilde{A}$, we shall end up with Sylvester equation $\widetilde{\Lambda}_2\widetilde{U}_2^*U_1 - \widetilde{U}_2^*U_1\Lambda_1 = \widetilde{U}_2^*R$, which leads to

$$\|\sin\Theta(U_1,\widetilde{U}_1)\|_{\text{F}} \le \|R\|_{\text{F}} \Bigg/ \min_{\mu\in\lambda(\Lambda_1),\,\widetilde{\mu}\in\lambda(\widetilde{\Lambda}_2)} |\mu-\widetilde{\mu}|\ .$$

This is a Davis–Kahan $\sin\theta$ theorem. Our other theorems in this section relate to Davis–Kahan $\sin\theta$ theorems analogously.

Remark 3.2. Let $U_{1,\text{sub}}$ be a submatrix consisting of a few (or just one) of U_1's columns, and let $\Lambda_{1,\text{sub}}$ be the corresponding eigenvalue matrix. (3.1) and (3.2) imply that

$$\begin{aligned}\widetilde{\Lambda}_2\widetilde{U}_2^*U_{1,\text{sub}} - \widetilde{U}_2^*U_{1,\text{sub}}\Lambda_{1,\text{sub}} &= \widetilde{\Lambda}_2\widetilde{U}_2^*(I-D^{-1})U_{1,\text{sub}} + \widetilde{U}_2^*(D^*-I)U_{1,\text{sub}}\Lambda_{1,\text{sub}},\\ \widetilde{\Lambda}_2X_{\text{sub}} - X_{\text{sub}}\Lambda_{1,\text{sub}} &= \widetilde{\Lambda}_2\widetilde{U}_2^*(D^*-D^{-1})U_{1,\text{sub}},\end{aligned}$$

where $X_{\text{sub}} \stackrel{\text{def}}{=} \widetilde{U}_2^*D^*U_{1,\text{sub}} = \widetilde{U}_2^*U_{1,\text{sub}} - \widetilde{U}_2^*(I-D^*)U_{1,\text{sub}}$. So our approach needs no modifications when it comes to bound the closeness of either $\mathcal{R}(U_{1,\text{sub}})$ to $\mathcal{R}(\widetilde{U}_1)$ or $\mathcal{R}(U_1)$ to $\mathcal{R}(\widetilde{U}_1)$. It can be seen that *all the theorems in this section remain valid if* U_1 *is replaced by* $U_{1,\text{sub}}$ *and the relative gaps are redefined as those between* $\lambda(\Lambda_{1,\text{sub}})$ *and* $\lambda(\widetilde{\Lambda}_2)$.

Eisenstat and Ipsen [7] obtained the following: *Under the assumptions of Theorem 3.1,*

$$\|\sin\Theta(U_1,\widetilde{U}_1)\|_2 \le \sqrt{k}\left(\|I-D^*\|_2 + \frac{\|I-D^{-*}D^{-1}\|_2}{\eta_{\text{c}}}\right). \tag{3.6}$$

It is a good bound for $k=1$. But for $k\ge 2$, it is less competitive. To compare this inequality with (3.5), we notice that

$$\begin{aligned}\|\sin\Theta(U_1,\widetilde{U}_1)\|_2 &\le \|\sin\Theta(U_1,\widetilde{U}_1)\|_{\text{F}},\\ \|(I-D^*)U_1\|_{\text{F}} &\le \sqrt{k}\|(I-D^*)U_1\|_2 \le \sqrt{k}\|I-D^*\|_2,\\ \|(D^*-D^{-1})U_1\|_{\text{F}} &\le \sqrt{k}\|D^*-D^{-1}\|_2 \le \sqrt{k}\|D^*\|_2\|I-D^{-*}D^{-1}\|_2,\\ \|I-D^{-*}D^{-1}\|_2 &\le \|D^{-*}\|_2\|D^*-D^{-1}\|_2.\end{aligned}$$

Thus (3.5) and (3.6) imply that

$$\|\sin\Theta(U_1,\widetilde{U}_1)\|_2 \le \sqrt{k}\left(\|I-D^*\|_2 + \frac{\|D^*\|_2\|I-D^{-*}D^{-1}\|_2}{\eta_{\text{c}}}\right), \tag{3.5a}$$

$$\|\sin\Theta(U_1,\widetilde{U}_1)\|_{\text{F}} \le k\left(\|I-D^*\|_{\text{F}} + \frac{\|D^{-*}\|_2\|D^*-D^{-1}\|_{\text{F}}}{\eta_{\text{c}}}\right). \tag{3.6a}$$

Now for inequalities (3.5) and (3.6) to be of any significance at all, D must be fairly close to the identity matrix, under which $\|D^*\|_2 \approx 1$ and thus (3.5a)—a weakened

(3.5)—is about as good as (3.6). Here is an example for which (3.5) improves (3.6) by at least a factor of $\sqrt{k}$. Take $D = I - \epsilon w w^*$, where $\epsilon > 0$ is a small number and w a vector with $\|w\|_2 = 1$. Then $D^{-1} = I + \epsilon w w^*/(1-\epsilon)$. Thus,

$$D^* - D^{-1} = -(2-\epsilon)\frac{\epsilon}{1-\epsilon} w w^* \quad \text{and} \quad I - D^{-*}D^{-1} = -\left(2 + \frac{\epsilon}{1-\epsilon}\right)\frac{\epsilon}{1-\epsilon} w w^*.$$

Hence (3.5) yields

$$\|\sin\Theta(U_1, \widetilde{U}_1)\|_2 \le \frac{\epsilon}{1-\epsilon} + \frac{1}{\eta_{\rm c}}(2-\epsilon)\frac{\epsilon}{1-\epsilon}, \tag{3.5b}$$

and (3.6) becomes

$$\|\sin\Theta(U_1, \widetilde{U}_1)\|_2 \le \sqrt{k}\left[\frac{\epsilon}{1-\epsilon} + \frac{1}{\eta_{\rm c}}\left(2 + \frac{\epsilon}{1-\epsilon}\right)\frac{\epsilon}{1-\epsilon}\right]. \tag{3.6b}$$

It is can be seen that when $\Theta(U_1, \widetilde{U}_1)$ is almost a multiple of identity and both $I - D^*$ and $I - D^{-*}D^{-1}$ are almost of rank 1, (3.6a) is not much weaker than (3.6), and in this case (3.5) improves (3.6a) by nearly a factor of k. But in any event the improvement can be by a factor at most k.

Now we compare (3.4) and (3.5). Although $\eta_{\rm c} \ge \eta_2$ always, and that $\eta_{\rm c}$ may be much larger than η_2, it appears (3.4) and (3.5) are comparable by a constant factor unless D is much closer to a unitary matrix than the identity matrix[4] in which case the second term on the right-hand side of (3.5) becomes negligible and consequently we expect (3.5) to be sharper than (3.4). On one hand, (3.5) can always produce a bound that is only weaker than (3.4) by a factor of $\sqrt{4 + 2\sqrt{2}}$:

$$\begin{aligned}\|\sin\Theta(U_1, \widetilde{U}_1)\|_{\rm F} &\le \frac{\sqrt{2}\|(I - D^*)U_1\|_{\rm F}}{\eta_2} + \frac{\|(D^* - I)U_1\|_{\rm F} + \|(I - D^{-1})U_1\|_{\rm F}}{\eta_2} \\ &\le \sqrt{4 + 2\sqrt{2}}\,\frac{\sqrt{\|(I - D^{-1})U_1\|_{\rm F}^2 + \|(I - D^*)U_1\|_{\rm F}^2}}{\eta_2},\end{aligned}$$

where we have used $\eta_2 \le \sqrt{2}$ and $\eta_2 \le \eta_{\rm c}$. On the other hand, (3.4) cannot be much worse than (3.5) in general, also by a constant factor, at least for the interesting cases when $E \overset{\rm def}{=} I - D$ is tiny. In fact, the following arguments show that when η_2 and $\eta_{\rm c}$ are not too much apart, (3.4) may be sharper. Assume that D differs from I not much worse than from its closest unitary matrix. We have

$$\begin{aligned}\sqrt{\|I - D^{-1}\|_{\rm F}^2 + \|I - D^*\|_{\rm F}^2} &\approx \sqrt{2}\|E\|_{\rm F} + O(\|E\|_{\rm F}^2), \\ \|D^* - D^{-1}\|_{\rm F} &\approx 2\|E\|_{\rm F} + O(\|E\|_{\rm F}^2).\end{aligned}$$

(3.4) and (3.5) become

$$\|\sin\Theta(U_1, \widetilde{U}_1)\|_{\rm F} \le \frac{\sqrt{2}\|E\|_{\rm F}}{\eta_2} + O(\|E\|_{\rm F}^2), \tag{3.4c}$$

$$\|\sin\Theta(U_1, \widetilde{U}_1)\|_{\rm F} \le \|E\|_{\rm F} + \frac{2\|E\|_{\rm F}}{\eta_{\rm c}} + O(\|E\|_{\rm F}^2). \tag{3.5c}$$

[4] By this we mean that $\|I - D\|_2 \gg \|D^* - D^{-1}\|_2$.

Write $\eta_{\rm c} = \gamma\eta_2$, where $1 \le \gamma$, and let $\eta_{\rm c} = |\lambda_s - \widetilde{\lambda}_t|/|\widetilde{\lambda}_t|$. $\eta_{\rm c} = \gamma\eta_2 \le \gamma\varrho_2(\lambda_s, \widetilde{\lambda}_t)$ implies $|\lambda_s|/|\widetilde{\lambda}_t| \le \sqrt{\gamma^2-1}$, and so $\eta_{\rm c} \ge \sqrt{\gamma^2-1} - 1$. The ratio of the right-hand side of (3.4c) over that of (3.5c) is (terms of $O(\|E\|^2)$ are ignored)

$$\frac{\sqrt{2}/(\eta_{\rm c}/\gamma)}{1 + 2/\eta_{\rm c}} = \frac{\sqrt{2}\,\gamma}{\eta_{\rm c} + 2}.$$

Now for $1 \le \gamma \le \sqrt{2}$, this ratio is less than 1, which means that (3.4c) is sharper; for $\gamma > \sqrt{2}$, this ratio is bounded by

$$\frac{\sqrt{2}\,\gamma}{\eta_{\rm c} + 2} \le \frac{\sqrt{2}\,\gamma}{\sqrt{\gamma^2-1}+1} \le \sqrt{2}$$

since $\sqrt{2}\,\gamma/(\sqrt{\gamma^2-1}+1)$ is monotonically increasing for $\sqrt{2} \le \gamma$. This means that when $I - D$ is tiny and that D is about as equally close to a unitary matrix as to the identity matrix, (3.4) cannot be worse than (3.5) by a factor more than $\sqrt{2}$.

THEOREM 3.2. *Let A and $\widetilde{A} = D^*AD$ be two $n \times n$ Hermitian matrices with eigendecompositions* (1.1)–(1.3)*, where D is nonsingular. Assume that the spectra of Λ_1 and $\widetilde{\Lambda}_2$ distribute as in Figure* 2.2*. Then for any unitarily invariant norm $|\!|\!|\cdot|\!|\!|$,*

$$\left|\!\left|\!\left|\sin\Theta(U_1, \widetilde{U}_1)\right|\!\right|\!\right| \le \frac{\sqrt[q]{|\!|\!|(I - D^{-1})U_1|\!|\!|^q + |\!|\!|(I - D^*)U_1|\!|\!|^q}}{\underline{\eta}_p}, \tag{3.7}$$

$$\left|\!\left|\!\left|\sin\Theta(U_1, \widetilde{U}_1)\right|\!\right|\!\right| \le |\!|\!|(I - D^*)U_1|\!|\!| + \frac{|\!|\!|(D^* - D^{-1})U_1|\!|\!|}{\underline{\eta}_{\rm c}}, \tag{3.8}$$

where

$$\underline{\eta}_p \stackrel{\rm def}{=} \varrho_p(\alpha, \alpha+\delta) \quad \textit{and} \quad \underline{\eta}_{\rm c} \stackrel{\rm def}{=} \begin{cases} \delta/(\alpha+\delta) & \textit{if Figure } 2.2\text{(a)}, \\ \delta/\alpha & \textit{if Figure } 2.2\text{(b)}. \end{cases}$$

Proof. Lemma 2.3 and (3.1) yield (3.7), whereas Lemma 2.3, (3.2), and (3.3) yield (3.8). □

Theorems 3.1 and 3.2 deal with rather restrictive perturbations to A. In what follows we show how similar ideas can be applied to a more realistic situation when scaled A is much better conditioned. Consider *nonnegative definite* Hermitian matrix $A = S^*HS \in \mathbb{C}^{n\times n}$ which is perturbed to $\widetilde{A} = S^*\widetilde{H}S$, where S is a scaling matrix and usually diagonal. But this is not necessary to the theorems below. The elements of S can vary wildly and can even be singular. H is nonsingular and usually better conditioned than A itself. Set $\Delta H \stackrel{\rm def}{=} \widetilde{H} - H$. Notice that

$$\begin{aligned} A &= S^*HS = (H^{1/2}S)^*\, H^{1/2}S, \\ \widetilde{A} &= S^*H^{1/2}\big(I + H^{-1/2}(\Delta H)H^{-1/2}\big)H^{1/2}S \\ &= \Big(\big(I + H^{-1/2}(\Delta H)H^{-1/2}\big)^{1/2}H^{1/2}S\Big)^*\ \big(I + H^{-1/2}(\Delta H)H^{-1/2}\big)^{1/2}H^{1/2}S. \end{aligned}$$

Set

$$\begin{aligned} B &= S^*H^{1/2}, \\ \widetilde{B} &= S^*H^{1/2}\big(I + H^{-1/2}(\Delta H)H^{-1/2}\big)^{1/2} \\ &\stackrel{\rm def}{=} BD, \end{aligned}$$

where $D = \big(I + H^{-1/2}(\Delta H)H^{-1/2}\big)^{1/2}$. Given the eigendecompositions of A and $\widetilde{A}$ as in (1.1)–(1.3), it can be seen that B and $\widetilde{B}$ admit the following SVDs.

$$\begin{aligned} B &= (U_1, U_2)\begin{pmatrix} \Lambda_1^{1/2} & \\ & \Lambda_2^{1/2} \end{pmatrix}\begin{pmatrix} V_1^* \\ V_2^* \end{pmatrix}, \\ \widetilde{B} &= (\widetilde{U}_1, \widetilde{U}_2)\begin{pmatrix} \widetilde{\Lambda}_1^{1/2} & \\ & \widetilde{\Lambda}_2^{1/2} \end{pmatrix}\begin{pmatrix} \widetilde{V}_1^* \\ \widetilde{V}_2^* \end{pmatrix}, \end{aligned}$$

where U_i, $\widetilde{U}_i$ are the same as in (1.1), and $\begin{pmatrix} \overset{k}{V_1} & \overset{n-k}{V_2} \end{pmatrix}$ and $\begin{pmatrix} \overset{k}{\widetilde{V}_1} & \overset{n-k}{\widetilde{V}_2} \end{pmatrix}$ are unitary. We have

$$\widetilde{A} - A = \widetilde{B}\widetilde{B}^* - BB^* = \widetilde{B}D^*B^* - \widetilde{B}D^{-1}B^* = \widetilde{B}(D^* - D^{-1})B^*.$$

Pre- and post-multiply the equations by $\widetilde{U}_2^*$ and U_1, respectively, to get

$$\widetilde{\Lambda}_2\widetilde{U}_2^*U_1 - \widetilde{U}_2^*U_1\Lambda_1 = \widetilde{\Lambda}_2^{1/2}\widetilde{V}_2^*(D^* - D^{-1})V_1\Lambda_1^{1/2}, \tag{3.9}$$

a Sylvester equation. Notice that for any unitarily invariant norm,

$$\begin{aligned} &\left|\!\left|\!\left| \widetilde{V}_2^*(D^* - D^{-1})V_1 \right|\!\right|\!\right| \le |\!|\!| D^* - D^{-1} |\!|\!| \\ &= \left|\!\left|\!\left| \Big(I + H^{-1/2}(\Delta H)H^{-1/2}\Big)^{1/2} - \Big(I + H^{-1/2}(\Delta H)H^{-1/2}\Big)^{-1/2} \right|\!\right|\!\right| \\ &\le \| \Big(I + H^{-1/2}(\Delta H)H^{-1/2}\Big)^{-1/2} \|_2 \left|\!\left|\!\left| H^{-1/2}(\Delta H)H^{-1/2} \right|\!\right|\!\right| \\ &\le \frac{\|H^{-1}\|_2 \, |\!|\!|\Delta H|\!|\!|}{\sqrt{1 - \|H^{-1}\|_2\|\Delta H\|_2}}. \end{aligned}$$

(3.9) makes relative distance χ a natural choice. Lemmas 2.4, 2.5, and (3.9) produce the following two theorems.

THEOREM 3.3. *Let $A = S^*HS$ and $\widetilde{A} = S^*\widetilde{H}S$ be two $n \times n$ Hermitian matrices with eigendecompositions* (1.1)–(1.3). *Assume H is positive definite and $\|H^{-1}\|_2\|\Delta H\|_2 < 1$. If $\eta_\chi \overset{\text{def}}{=} \min_{\mu\in\lambda(\Lambda_1),\,\widetilde{\mu}\in\lambda(\widetilde{\Lambda}_2)} \chi(\mu, \widetilde{\mu}) > 0$, then*

$$\|\sin\Theta(U_1, \widetilde{U}_1)\|_{\mathrm{F}} \le \frac{\|D - D^{-1}\|_{\mathrm{F}}}{\eta_\chi} \le \frac{\|H^{-1}\|_2}{\sqrt{1 - \|H^{-1}\|_2\|\Delta H\|_2}} \frac{\|\Delta H\|_{\mathrm{F}}}{\eta_\chi}. \tag{3.10}$$

where $D = (I + H^{-1/2}(\Delta H)H^{-1/2})^{1/2} = D^*$.

THEOREM 3.4. *Let $A = S^*HS$ and $\widetilde{A} = S^*\widetilde{H}S$ be two $n \times n$ Hermitian matrices with eigendecompositions* (1.1)—(1.3). *H is positive definite and $\|H^{-1}\|_2\|\Delta H\|_2 < 1$. Assume that the spectra of Λ_1 and $\widetilde{\Lambda}_2$ distribute as in Figure* 2.2. *Then for any unitarily invariant norm $|\!|\!| \cdot |\!|\!|$,*

$$\left|\!\left|\!\left| \sin\Theta(U_1, \widetilde{U}_1) \right|\!\right|\!\right| \le \frac{|\!|\!| D - D^{-1} |\!|\!|}{\underline{\eta}_\chi} \le \frac{\|H^{-1}\|_2}{\sqrt{1 - \|H^{-1}\|_2\|\Delta H\|_2}} \frac{|\!|\!|\Delta H|\!|\!|}{\underline{\eta}_\chi}, \tag{3.11}$$

where $\underline{\eta}_\chi \overset{\text{def}}{=} \chi(\alpha, \alpha + \delta)$ and $D = (I + H^{-1/2}(\Delta H)H^{-1/2})^{1/2} = D^*$.

Remark 3.3. Our approach can be extended straightforwardly to diagonalizable matrices. Consider A and $\widetilde{A} = D_1^* A D_2$, where D_1 and D_2 are nonsingular. Suppose that both A and $\widetilde{A}$ are diagonalizable and let

$$A(X_1, X_2) = (X_1, X_2)\begin{pmatrix} \Lambda_1 & \\ & \Lambda_2 \end{pmatrix} \quad \text{and} \quad \widetilde{A}(\widetilde{X}_1, \widetilde{X}_2) = (\widetilde{X}_1, \widetilde{X}_2)\begin{pmatrix} \widetilde{\Lambda}_1 & \\ & \widetilde{\Lambda}_2 \end{pmatrix},$$

where $\begin{pmatrix} \overset{k}{X_1} & \overset{n-k}{X_2} \end{pmatrix}$ and $\begin{pmatrix} \overset{k}{\widetilde{X}_1} & \overset{n-k}{\widetilde{X}_2} \end{pmatrix}$ are nonsingular, and Λ_i and $\widetilde{\Lambda}_j$ are defined as in (1.2) and (1.3) with λ_i's and $\widetilde{\lambda}_j$'s possibly complex. Partition

$$(X_1, X_2)^{-1} = \begin{array}{c} k \\ n-k \end{array}\begin{pmatrix} Y_1^* \\ Y_2^* \end{pmatrix} \quad \text{and} \quad (\widetilde{X}_1, \widetilde{X}_2)^{-1} = \begin{array}{c} k \\ n-k \end{array}\begin{pmatrix} \widetilde{Y}_1^* \\ \widetilde{Y}_2^* \end{pmatrix}.$$

Define $R \overset{\text{def}}{=} \widetilde{A} X_1 - X_1 \Lambda_1 = (\widetilde{A} - A) X_1$. We have

$$\begin{aligned} \widetilde{Y}_2^* R &= \widetilde{Y}_2^* \widetilde{A} X_1 - \widetilde{Y}_2^* X_1 \Lambda_1 = \widetilde{\Lambda}_2 \widetilde{Y}_2^* X_1 - \widetilde{Y}_2^* X_1 \Lambda_1, \\ \widetilde{Y}_2^* R &= \widetilde{Y}_2^* \left[\widetilde{A}(I - D_2^{-1}) + (D_1^* - I) A \right] X_1 \\ &= \widetilde{\Lambda}_2 \widetilde{Y}_2^* (I - D_2^{-1}) X_1 + \widetilde{Y}_2^* (D_1^* - I) X_1 \Lambda_1. \end{aligned}$$

Thus we have the following perturbation equations:

$$\widetilde{\Lambda}_2 \widetilde{Y}_2^* X_1 - \widetilde{Y}_2^* X_1 \Lambda_1 = \widetilde{\Lambda}_2 \widetilde{Y}_2^* (I - D_2^{-1}) X_1 + \widetilde{Y}_2^* (D_1^* - I) X_1 \Lambda_1, \tag{3.12}$$

$$\widetilde{\Lambda}_2 Z - Z \Lambda_1 = \widetilde{\Lambda}_2 \widetilde{Y}_2^* (D_1^* - D_2^{-1}) X_1, \tag{3.13}$$

where $Z \overset{\text{def}}{=} \widetilde{Y}_2^* D_1^* X_1 = \widetilde{Y}_2^* X_1 - \widetilde{Y}_2^* (I - D_1^*) X_1$, from which various bounds on $\sin \Theta(X_1, \widetilde{X}_1)$ can be derived under certain conditions. For example, let

$$\eta_2 \overset{\text{def}}{=} \min_{\mu \in \lambda(\Lambda_1),\, \widetilde{\mu} \in \lambda(\widetilde{\Lambda}_2)} \varrho_2(\mu, \widetilde{\mu}).$$

If $\eta_2 > 0$, then by Lemma 2.2 we have

$$\begin{aligned} \|\widetilde{Y}_2^* X_1\|_{\mathrm{F}} &\le \frac{1}{\eta_2} \sqrt{\|\widetilde{Y}_2^* (I - D_2^{-1}) X_1\|_{\mathrm{F}}^2 + \|\widetilde{Y}_2^* (D_1^* - I) X_1\|_{\mathrm{F}}^2} \\ &\le \frac{1}{\eta_2} \|\widetilde{Y}_2^*\|_2 \|X_1\|_2 \sqrt{\|I - D_2^{-1}\|_{\mathrm{F}}^2 + \|D_1^* - I\|_{\mathrm{F}}^2}. \end{aligned}$$

Notice that by Lemma 2.1

$$\begin{aligned} \|\sin \Theta(X_1, \widetilde{X}_1)\|_{\mathrm{F}} &= \|(\widetilde{Y}_2^* \widetilde{Y}_2)^{-1/2} \widetilde{Y}_2^* X_1 (X_1^* X_1)^{-1/2}\|_{\mathrm{F}} \\ &\le \|(\widetilde{Y}_2^* \widetilde{Y}_2)^{-1/2}\|_2 \|\widetilde{Y}_2^* X_1\|_{\mathrm{F}} \|(X_1^* X_1)^{-1/2}\|_2. \end{aligned}$$

Then a bound on $\|\sin \Theta(X_1, \widetilde{X}_1)\|_{\mathrm{F}}$ is immediately available.

4. Relative perturbation theorems for singular subspace variation. Let B and $\widetilde{B}$ be two $m \times n$ $(m \geq n)$ (complex) matrices with SVDs

$$B = (U_1, U_2) \begin{pmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} V_1^* \\ V_2^* \end{pmatrix} \quad \text{and} \quad \widetilde{B} = (\widetilde{U}_1, \widetilde{U}_2) \begin{pmatrix} \widetilde{\Sigma}_1 & 0 \\ 0 & \widetilde{\Sigma}_2 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \widetilde{V}_1^* \\ \widetilde{V}_2^* \end{pmatrix}, \tag{4.1}$$

where $U = \begin{pmatrix} \overset{k}{U_1} & \overset{m-k}{U_2} \end{pmatrix}$ and $\widetilde{U} = \begin{pmatrix} \overset{k}{\widetilde{U}_1} & \overset{m-k}{\widetilde{U}_2} \end{pmatrix}$ are $m \times m$ unitary, and $V = \begin{pmatrix} \overset{k}{V_1} & \overset{n-k}{V_2} \end{pmatrix}$ and $\widetilde{V} = \begin{pmatrix} \overset{k}{\widetilde{V}_1} & \overset{n-k}{\widetilde{U}_2} \end{pmatrix}$ are $n \times n$ unitary, $1 \leq k < n$, and

$$\Sigma_1 = \operatorname{diag}(\sigma_1, \ldots, \sigma_k), \quad \Sigma_2 = \operatorname{diag}(\sigma_{k+1}, \ldots, \sigma_n), \tag{4.2}$$

$$\widetilde{\Sigma}_1 = \operatorname{diag}(\widetilde{\sigma}_1, \ldots, \widetilde{\sigma}_k), \quad \widetilde{\Sigma}_2 = \operatorname{diag}(\widetilde{\sigma}_{k+1}, \ldots, \widetilde{\sigma}_n). \tag{4.3}$$

Define residuals

$$R_{\mathrm{R}} \overset{\text{def}}{=} \widetilde{B}V_1 - U_1\Sigma_1 = (\widetilde{B} - B)V_1 \quad \text{and} \quad R_{\mathrm{L}} \overset{\text{def}}{=} \widetilde{B}^*U_1 - V_1\Sigma_1 = (\widetilde{B}^* - B^*)U_1. \tag{4.4}$$

Our development for the singular value problems more or less resembles what we did for the eigenvalue problems. Most comments and comparisons we made in section 3 apply here, too, with perhaps a little change. Nonetheless there is a little bit of complication here, namely we shall work with two residuals, R_{R} and R_{L}, and end up with bounding solutions to two coupled matrix equations. The proofs of theorems in this section are relatively long and are postponed to the next section.

THEOREM 4.1. *Let B and $\widetilde{B} = D_1^* B D_2$ be two $m \times n$ $(m \geq n)$ (complex) matrices with SVDs* (4.1)–(4.3)*, where D_1 and D_2 are nonsingular. Let*

$$\eta_2 \overset{\text{def}}{=} \min_{\mu \in \sigma(\Sigma_1),\, \widetilde{\mu} \in \sigma_{\text{ext}}(\widetilde{\Sigma}_2)} \varrho_2(\mu, \widetilde{\mu}) \quad \textit{and} \quad \eta_{\mathrm{c}} \overset{\text{def}}{=} \min_{\mu \in \sigma(\Sigma_1),\, \widetilde{\mu} \in \sigma_{\text{ext}}(\widetilde{\Sigma}_2)} \frac{|\mu - \widetilde{\mu}|}{|\widetilde{\mu}|}, \tag{4.5}$$

where $\sigma_{\text{ext}}(\widetilde{\Sigma}_2) \equiv \sigma(\widetilde{\Sigma}_2) \cup \{0\}$ if $m > n$, and $\sigma_{\text{ext}}(\widetilde{\Sigma}_2) \equiv \sigma(\widetilde{\Sigma}_2)$ otherwise. If η_{c}, $\eta_2 > 0$, then

$$\begin{aligned} &\sqrt{\|\sin\Theta(U_1, \widetilde{U}_1)\|_{\mathrm{F}}^2 + \|\sin\Theta(V_1, \widetilde{V}_1)\|_{\mathrm{F}}^2} \\ &\qquad \leq \frac{\sqrt{\|(I - D_1^*)U_1\|_{\mathrm{F}}^2 + \|(I - D_1^{-1})U_1\|_{\mathrm{F}}^2 + \|(I - D_2^*)V_1\|_{\mathrm{F}}^2 + \|(I - D_2^{-1})V_1\|_{\mathrm{F}}^2}}{\eta_2}, \end{aligned} \tag{4.6}$$

$$\begin{aligned} \sqrt{\|\sin\Theta(U_1, \widetilde{U}_1)\|_{\mathrm{F}}^2 + \|\sin\Theta(V_1, \widetilde{V}_1)\|_{\mathrm{F}}^2} &\leq \sqrt{\|(I - D_1^*)U_1\|_{\mathrm{F}}^2 + \|(I - D_2^*)V_1\|_{\mathrm{F}}^2} \\ &\quad + \frac{1}{\eta_{\mathrm{c}}} \sqrt{\|(D_1^* - D_1^{-1})U_1\|_{\mathrm{F}}^2 + \|(D_2^* - D_2^{-1})V_1\|_{\mathrm{F}}^2}. \end{aligned} \tag{4.7}$$

This theorem is an extension of a Wedin $\sin\theta$ theorem [16], where no multiplicative structure in the perturbation of B to $\widetilde{B}$ is assumed. He proved

$$\sqrt{\|\sin\Theta(U_1,\widetilde{U}_1)\|_{\mathrm{F}}^2+\|\sin\Theta(V_1,\widetilde{V}_1)\|_{\mathrm{F}}^2}\le\frac{\sqrt{\|R_{\mathrm{R}}\|_{\mathrm{F}}^2+\|R_{\mathrm{L}}\|_{\mathrm{F}}^2}}{\delta}, \tag{4.8}$$

where $\delta\overset{\mathrm{def}}{=}\min_{\mu\in\sigma(\Sigma_1),\,\widetilde{\mu}\in\sigma_{\mathrm{ext}}(\widetilde{\Sigma}_2)}|\mu-\widetilde{\mu}|$. Our other theorems in this section relate to Wedin $\sin\theta$ theorems analogously.

Remark 4.1. *Ghost singular values* 0 are appended to $\sigma(\widetilde{\Sigma}_2)$ when $m>n$. This is not necessary for the sensitivity of the V-factor alone, but rather the U-factor depends on the ghost singular values. More fine analysis can be given to illustrate this point. To keep the theorem relatively concise, we shall not go into this matter further.

THEOREM 4.2. *Let B and $\widetilde{B}=D_1^*BD_2$ be two $m\times n$ $(m\ge n)$ (complex) matrices with SVDs* (4.1)–(4.3)*, where D_1 and D_2 are nonsingular. If there exist $\alpha\ge 0$ and $\delta>0$ such that*

$$\min_{1\le i\le k}\sigma_i\ge\alpha+\delta \quad \textit{and} \quad \max_{1\le j\le n-k}\widetilde{\sigma}_{k+j}\le\alpha,$$

then for any unitarily invariant norm $|\!|\!|\cdot|\!|\!|$

$$\begin{aligned}&\max\left\{|\!|\!|\sin\Theta(U_1,\widetilde{U}_1)|\!|\!|,\,|\!|\!|\sin\Theta(V_1,\widetilde{V}_1)|\!|\!|\right\}\\ &\qquad\le\frac{1}{\underline{\eta}_p}\max\left\{\sqrt[q]{|\!|\!|(I-D_2^{-1})V_1|\!|\!|^q+|\!|\!|(D_1^*-I)U_1|\!|\!|^q},\right.\\ &\qquad\qquad\qquad\left.\sqrt[q]{|\!|\!|(I-D_1^{-1})U_1|\!|\!|^q+|\!|\!|(D_2^*-I)V_1|\!|\!|^q}\right\},\end{aligned} \tag{4.9}$$

$$\begin{aligned}&\left|\!\left|\!\left|\begin{pmatrix}\sin\Theta(U_1,\widetilde{U}_1) & \\ & \sin\Theta(V_1,\widetilde{V}_1)\end{pmatrix}\right|\!\right|\!\right|\\ &\le\frac{\sqrt[q]{\left|\!\left|\!\left|\begin{pmatrix}(I-D_1^{-1})U_1 & \\ & (I-D_2^{-1})V_1\end{pmatrix}\right|\!\right|\!\right|^q+\left|\!\left|\!\left|\begin{pmatrix}(I-D_1^*)U_1 & \\ & (I-D_2^*)V_1\end{pmatrix}\right|\!\right|\!\right|^q}}{\underline{\eta}_p},\end{aligned} \tag{4.10}$$

where $\underline{\eta}_p\overset{\mathrm{def}}{=}\varrho_p(\alpha,\alpha+\delta)$, and

$$\begin{aligned}\max\left\{|\!|\!|\sin\Theta(U_1,\widetilde{U}_1)|\!|\!|,\,|\!|\!|\sin\Theta(V_1,\widetilde{V}_1)|\!|\!|\right\}&\le\max\left\{|\!|\!|(I-D_1^*)U_1|\!|\!|,\,|\!|\!|(I-D_2^*)V_1|\!|\!|\right\}\\ &\quad+\frac{1}{\underline{\eta}_{\mathrm{c}}}\max\left\{|\!|\!|(D_1^*-D_1^{-1})U_1|\!|\!|,\,|\!|\!|(D_2^*-D_2^{-1})V_1|\!|\!|\right\},\end{aligned} \tag{4.11}$$

$$\begin{aligned}\left|\!\left|\!\left|\begin{pmatrix}\sin\Theta(U_1,\widetilde{U}_1) & \\ & \sin\Theta(V_1,\widetilde{V}_1)\end{pmatrix}\right|\!\right|\!\right|&\le\left|\!\left|\!\left|\begin{pmatrix}(I-D_1^*)U_1 & \\ & (I-D_2^*)V_1\end{pmatrix}\right|\!\right|\!\right|\\ &\quad+\frac{1}{\underline{\eta}_{\mathrm{c}}}\left|\!\left|\!\left|\begin{pmatrix}(D_1^*-D_1^{-1})U_1 & \\ & (D_2^*-D_2^{-1})V_1\end{pmatrix}\right|\!\right|\!\right|,\end{aligned} \tag{4.12}$$

where $\underline{\eta}_{\mathrm{c}}\overset{\mathrm{def}}{=}\delta/\alpha$.

We have both (4.9) and (4.10), and both (4.11) and (4.12) for theoretical considerations. In fact (4.9) differs from (4.10) in many ways from the theoretical point of view, and the two are independent.

Remark 4.2. Intuitively, D_2 should not affect $\mathcal{R}(U_1)$ much as long as it is close to a unitary matrix. In fact, if D_2 is unitary it does not affect $\mathcal{R}(U_1)$ at all. This suggests that when one or both D_i's are closer to unitary matrices[5] than to the identity matrices, better bounds may be possible. Li [13] indeed presented theorems, showing that D_1 contributes to $\sin\Theta(V_1, \widetilde{V}_1)$ by its departure from *some* unitary matrix rather than the identity matrix and similar conclusions hold for D_2 and $\sin\Theta(U_1, \widetilde{U}_1)$. The reader is referred to Li [13] for details.

Remark 4.3. Theorems 4.1 and 4.2, applied to a special case [13] when one of the D_i's is the identity matrix and the other one takes the form $\begin{pmatrix} I & X \\ & I \end{pmatrix}$, yield a *deflation* criterion that is *sooner* and *cheaper* in Demmel–Kahan QR algorithm [5] for computing the singular value system of a bidiagonal matrix; see [10].

Theorems 4.1 and 4.2 deal with rather restrictive perturbations to B. In what follows we show how to apply them to a more realistic situation when scaled B is much better conditioned. Consider $B = GS \in \mathbb{C}^{n\times n}$ which is perturbed to $\widetilde{B} = \widetilde{G}S \in \mathbb{C}^{n\times n}$, where S is a scaling matrix and G is nonsingular. Interesting cases are when G is much better conditioned than B. Set $\Delta G \overset{\text{def}}{=} \widetilde{G} - G$. If $\|(\Delta G)G^{-1}\|_2 < 1$, then $\widetilde{G} = G + \Delta G = [I + (\Delta G)G^{-1}]G$ is nonsingular also.

THEOREM 4.3. *Let $B = GS \in \mathbb{C}^{n\times n}$ and $\widetilde{B} = \widetilde{G}S \in \mathbb{C}^{n\times n}$ with SVDs* (4.1)–(4.3)*, where G is nonsingular. Assume $\|(\Delta G)G^{-1}\|_2 < 1$. If*

$$\eta_2 \overset{\text{def}}{=} \min_{\mu\in\sigma(\Sigma_1),\, \widetilde{\mu}\in\sigma(\widetilde{\Sigma}_2)} \varrho_2(\mu, \widetilde{\mu}) > 0,$$

then

$$\begin{aligned} (4.13) \qquad & \sqrt{\left\|\sin\Theta(U_1, \widetilde{U}_1)\right\|_{\mathrm{F}}^2 + \left\|\sin\Theta(V_1, \widetilde{V}_1)\right\|_{\mathrm{F}}^2} \\ & \leq \frac{\sqrt{\|(\Delta G)G^{-1}U_1\|_{\mathrm{F}}^2 + \|[I + G^{-*}(\Delta G)^*]^{-1}G^{-*}(\Delta G)^*U_1\|_{\mathrm{F}}^2}}{\eta_2} \\ & \leq \|G^{-1}\|_2 \sqrt{1 + \frac{1}{(1 - \|G^{-1}\|_2\|\Delta G\|_2)^2}}\; \frac{\|\Delta G\|_{\mathrm{F}}}{\eta_2}. \end{aligned}$$

Proof. Write $\widetilde{B} = \widetilde{G}S = [I + (\Delta G)G^{-1}]GS = D_1^* B D_2$, where $D_1^* = I + (\Delta G)G^{-1}$ and $D_2 = I$. Apply Theorem 4.1 to get (4.13). □

THEOREM 4.4. *Let $B = GS \in \mathbb{C}^{n\times n}$ and $\widetilde{B} = \widetilde{G}S \in \mathbb{C}^{n\times n}$ with SVDs* (4.1)–(4.3)*, where G is nonsingular. Assume $\|(\Delta G)G^{-1}\|_2 < 1$. If there exist $\alpha \geq 0$ and $\delta > 0$ such that*

$$\min_{\mu\in\sigma(\Sigma_1)} \mu \geq \alpha + \delta \quad \textit{and} \quad \max_{\widetilde{\mu}\in\sigma(\widetilde{\Sigma}_2)} \widetilde{\mu} \leq \alpha$$

or

$$\min_{\mu\in\sigma(\Sigma_1)} \mu \leq \alpha \quad \textit{and} \quad \max_{\widetilde{\mu}\in\sigma(\widetilde{\Sigma}_2)} \widetilde{\mu} \geq \alpha + \delta,$$

[5] When both D_1 and D_2 are unitary, $\widetilde{B} = D_1^* B D_2 = D_1^* U\Sigma V^* D_2$ is an SVD of $\widetilde{B}$, which implies $\widetilde{U} = D_1^* U$ and $\widetilde{V} = D_2^* V$. Thus perturbations to singular subspaces, in this case, are independent of the gap between $\sigma(\Sigma_1)$ and $\sigma(\widetilde{\Sigma}_2)$, as long as they are disjoint.

then for any unitarily invariant norm $|||\cdot|||$,

$$(4.14)\quad \max\left\{\left|\left|\left|\sin\Theta(U_1,\widetilde{U}_1)\right|\right|\right|,\left|\left|\left|\sin\Theta(V_1,\widetilde{V}_1)\right|\right|\right|\right\}$$
$$\leq \frac{\max\left\{|||(\Delta G)G^{-1}U_1|||,|||[I+G^{-*}(\Delta G)^*]^{-1}G^{-*}(\Delta G)^*U_1|||\right\}}{\eta_\infty}$$
$$\leq \frac{\|G^{-1}\|_2}{1-\|G^{-1}\|_2\|\Delta G\|_2}\ \frac{|||\Delta G|||}{\eta_\infty},$$

$$(4.15)\quad \left|\left|\left|\begin{pmatrix}\sin\Theta(U_1,\widetilde{U}_1) & \\ & \sin\Theta(V_1,\widetilde{V}_1)\end{pmatrix}\right|\right|\right|$$
$$\leq \frac{\sqrt[q]{|||(\Delta G)G^{-1}U_1|||^q+|||[I+G^{-*}(\Delta G)^*]^{-1}G^{-*}(\Delta G)^*U_1|||^q}}{\underline{\eta}_p}$$
$$\leq \|G^{-1}\|_2\sqrt[q]{1+\frac{1}{(1-\|G^{-1}\|_2\|\Delta G\|_2)^q}}\ \frac{|||\Delta G|||}{\underline{\eta}_p}.$$

Proof. Again write $\widetilde{B}=\widetilde{G}S=[I+(\Delta G)G^{-1}]GS=D_1^*BD_2$, where $D_1^*=I+(\Delta G)G^{-1}$ and $D_2=I$. Apply Theorem 4.2 to get (4.14) and (4.15). □

Remark 4.4. Better bounds, especially when $(\Delta G)G^{-1}$ is nearly a skew Hermitian matrix, can be proved for the angle $\Theta(V_1,\widetilde{V}_1)$. To prevent the paper from getting too long, we refer the reader to Li [13].

Remark 4.5. Theorems 4.3 and 4.4 can be extended to cover nonsquare matrices. Assume $B=GS$ and $\widetilde{B}=\widetilde{G}S$ are $m\times n$ $(m\geq n)$; S is a scaling matrix and both G and $\widetilde{G}$ are $m\times n$; G has full column rank. Let $G^\dagger=(G^*G)^{-1}G^*$ the pseudoinverse of G. Notice that $G^\dagger G=I$. Then

$$\widetilde{B}=\widetilde{G}S=(G+\Delta G)S=(I+(\Delta G)G^\dagger)GS=(I+(\Delta G)G^\dagger)B.$$

If $\|(\Delta G)G^\dagger\|_2\leq\|G^\dagger\|_2\|\Delta G\|_2<1$, $\widetilde{G}$ has full column rank, too. Now applying Theorems 4.1 and 4.2, we find that Theorems 4.3 and 4.4 remain valid with G^{-1} replaced by $G^\dagger$, and $\sigma(\widetilde{\Sigma}_2)$ by $\sigma_{\text{ext}}(\widetilde{\Sigma}_2)$.

5. Proofs of Theorems 4.1 and 4.2.

We can always augment B and $\widetilde{B}$ by $m\times(m-n)$ zero blocks to their rights to make them square. The augmented B and $\widetilde{B}$ will have straightforward SVDs based on those of B and $\widetilde{B}$. It turns out doing so does not affect the U-factors, and the V-factors are affected in a trivial way such that $|||\sin(V_1,\widetilde{V}_1)|||$ stays the same; see [13]. In what follows we shall deal with the square case only.

Let $R_{\text{R}}=\widetilde{B}V_1-U_1\Sigma_1=(\widetilde{B}-B)V_1$ and $R_{\text{L}}=\widetilde{B}^*U_1-V_1\Sigma_1=(\widetilde{B}^*-B^*)U_1$. When $m=n$, the SVDs (4.1)–(4.3) read

(5.1)
$$B=(U_1,U_2)\begin{pmatrix}\Sigma_1 & \\ & \Sigma_2\end{pmatrix}\begin{pmatrix}V_1^*\\ V_2^*\end{pmatrix}\quad\text{and}\quad \widetilde{B}=(\widetilde{U}_1,\widetilde{U}_2)\begin{pmatrix}\widetilde{\Sigma}_1 & \\ & \widetilde{\Sigma}_2\end{pmatrix}\begin{pmatrix}\widetilde{V}_1^*\\ \widetilde{V}_2^*\end{pmatrix}.$$

Notice that

$$\widetilde{U}_2^*R_{\text{R}}=\widetilde{U}_2^*\widetilde{B}V_1-\widetilde{U}_2^*U_1\Sigma_1=\widetilde{\Sigma}_2\widetilde{V}_2^*V_1-\widetilde{U}_2^*U_1\Sigma_1,$$
$$\widetilde{U}_2^*R_{\text{R}}=\widetilde{U}_2^*\left[\widetilde{B}(I-D_2^{-1})+(D_1^*-I)B\right]V_1$$
$$=\widetilde{\Sigma}_2\widetilde{V}_2^*(I-D_2^{-1})V_1+\widetilde{U}_2^*(D_1^*-I)U_1\Sigma_1$$

to get

$$\widetilde{\Sigma}_2 \widetilde{V}_2^* V_1 - \widetilde{U}_2^* U_1 \Sigma_1 = \widetilde{\Sigma}_2 \widetilde{V}_2^* (I - D_2^{-1}) V_1 + \widetilde{U}_2^* (D_1^* - I) U_1 \Sigma_1. \tag{5.2}$$

On the other hand,

$$\begin{aligned} \widetilde{V}_2^* R_{\mathrm{L}} &= \widetilde{V}_2^* \widetilde{B}^* U_1 - \widetilde{V}_2^* V_1 \Sigma_1 = \widetilde{\Sigma}_2 \widetilde{U}_2^* U_1 - \widetilde{V}_2^* V_1 \Sigma_1. \\ \widetilde{V}_2^* R_{\mathrm{L}} &= \widetilde{V}_2^* \left[\widetilde{B}^* (I - D_1^{-1}) + (D_2^* - I) B^* \right] U_1 \\ &= \widetilde{\Sigma}_2 \widetilde{U}_2^* (I - D_1^{-1}) U_1 + \widetilde{V}_2^* (D_2^* - I) V_1 \Sigma_1, \end{aligned}$$

which produce

$$\widetilde{\Sigma}_2 \widetilde{U}_2^* U_1 - \widetilde{V}_2^* V_1 \Sigma_1 = \widetilde{\Sigma}_2 \widetilde{U}_2^* (I - D_1^{-1}) U_1 + \widetilde{V}_2^* (D_2^* - I) V_1 \Sigma_1. \tag{5.3}$$

(5.2) and (5.3) take an equivalent form as a single matrix equation with dimensions doubled.

$$\begin{aligned} &\begin{pmatrix} & \widetilde{\Sigma}_2 \\ \widetilde{\Sigma}_2 & \end{pmatrix} \begin{pmatrix} \widetilde{U}_2^* U_1 & \\ & \widetilde{V}_2^* V_1 \end{pmatrix} - \begin{pmatrix} \widetilde{U}_2^* U_1 & \\ & \widetilde{V}_2^* V_1 \end{pmatrix} \begin{pmatrix} & \Sigma_1 \\ \Sigma_1 & \end{pmatrix} \\ &\qquad = \begin{pmatrix} & \widetilde{\Sigma}_2 \\ \widetilde{\Sigma}_2 & \end{pmatrix} \begin{pmatrix} \widetilde{U}_2^* (I - D_1^{-1}) U_1 & \\ & \widetilde{V}_2^* (I - D_2^{-1}) V_1 \end{pmatrix} \\ &\qquad\quad + \begin{pmatrix} \widetilde{U}_2^* (D_1^* - I) U_1 & \\ & \widetilde{V}_2^* (D_2^* - I) V_1 \end{pmatrix} \begin{pmatrix} & \Sigma_1 \\ \Sigma_1 & \end{pmatrix}. \end{aligned} \tag{5.4}$$

(5.2)–(5.4) can also be rearranged in such a way that sharper bounds can be proved when D_i's are closer to unitary matrices than the identity matrix. Write

$$X \overset{\text{def}}{=} \widetilde{V}_2^* D_2^* V_1 = \widetilde{V}_2^* V_1 - \widetilde{V}_2^* (I - D_2^*) V_1 \quad \text{and} \quad Y \overset{\text{def}}{=} \widetilde{U}_2^* D_1^* U_1 = \widetilde{U}_2^* U_1 - \widetilde{U}_2^* (I - D_1^*) U_1.$$

We have, from (5.2) and (5.3),

$$\widetilde{\Sigma}_2 X - Y \Sigma_1 = \widetilde{\Sigma}_2 \widetilde{V}_2^* (D_2^* - D_2^{-1}) V_1, \tag{5.5}$$

$$\widetilde{\Sigma}_2 Y - X \Sigma_1 = \widetilde{\Sigma}_2 \widetilde{U}_2^* (D_1^* - D_1^{-1}) U_1, \tag{5.6}$$

and, from (5.4),

$$\begin{aligned} &\begin{pmatrix} & \widetilde{\Sigma}_2 \\ \widetilde{\Sigma}_2 & \end{pmatrix} \begin{pmatrix} Y & \\ & X \end{pmatrix} - \begin{pmatrix} Y & \\ & X \end{pmatrix} \begin{pmatrix} & \Sigma_1 \\ \Sigma_1 & \end{pmatrix} \\ &\qquad = \begin{pmatrix} & \widetilde{\Sigma}_2 \\ \widetilde{\Sigma}_2 & \end{pmatrix} \begin{pmatrix} \widetilde{U}_2^* (D_1^* - D_1^{-1}) U_1 & \\ & \widetilde{V}_2^* (D_2^* - D_2^{-1}) V_1 \end{pmatrix}. \end{aligned} \tag{5.7}$$

(5.2)–(5.4) make ϱ_p a natural choice for measuring the relative gaps between $\sigma(\Sigma_1)$ and $\sigma(\widetilde{\Sigma}_2)$, while (5.5)–(5.7) make the classical measurement (2.2) a natural choice.

Remark 5.1. Just as in Remark 3.1, perturbation equations (5.2)–(5.7) can be modified to bound the closeness of the singular subspaces spanned by a few selected columns of U_1 and V_1 to $\mathcal{R}(\widetilde{U}_1)$ and $\mathcal{R}(\widetilde{V}_1)$.

Proof of Theorem 4.1. Notice that the eigenvalues of $\begin{pmatrix} & \widetilde{\Sigma}_2 \\ \widetilde{\Sigma}_2 & \end{pmatrix}$ are $\pm\widetilde{\sigma}_{k+j}$, that those of $\begin{pmatrix} & \Sigma_1 \\ \Sigma_1 & \end{pmatrix}$ are $\pm\sigma_i$, and that

$$\varrho_2(\sigma_i, -\widetilde{\sigma}_{k+j}) \geq \varrho_2(\sigma_i, \widetilde{\sigma}_{k+j}) \quad \text{and} \quad \varrho_2(-\sigma_i, \widetilde{\sigma}_{k+j}) \geq \varrho_2(\sigma_i, \widetilde{\sigma}_{k+j}).$$

By Lemma 2.2 and (5.4), we have

$$\begin{aligned} &\|\widetilde{U}_2^* U_1\|_{\mathrm{F}}^2 + \|\widetilde{V}_2^* V_1\|_{\mathrm{F}}^2 \\ &\leq \frac{1}{\eta_2^2}\Big[\|\widetilde{U}_2^*(I - D_1^{-1})U_1\|_{\mathrm{F}}^2 + \|\widetilde{U}_2^*(D_1^* - I)U_1\|_{\mathrm{F}}^2 + \|\widetilde{V}_2^*(I - D_2^{-1})V_1\|_{\mathrm{F}}^2 \\ &\qquad + \|\widetilde{V}_2^*(D_2^* - I)V_1\|_{\mathrm{F}}^2\Big] \\ &\leq \frac{1}{\eta_2^2}\left[\|(I - D_1^{-1})U_1\|_{\mathrm{F}}^2 + \|(D_1^* - I)U_1\|_{\mathrm{F}}^2 + \|(I - D_2^{-1})V_1\|_{\mathrm{F}}^2 + \|(D_2^* - I)V_1\|_{\mathrm{F}}^2\right], \end{aligned}$$

which gives (4.6). By Lemma 2.2 and (5.7), we have

$$\begin{aligned} \sqrt{\|X\|_{\mathrm{F}}^2 + \|Y\|_{\mathrm{F}}^2} &\leq \frac{1}{\eta_{\mathrm{c}}}\sqrt{\|\widetilde{U}_2^*(D_1^* - D_1^{-1})U_1\|_{\mathrm{F}}^2 + \|\widetilde{V}_2^*(D_2^* - D_2^{-1})V_1\|_{\mathrm{F}}^2} \\ &\leq \frac{1}{\eta_{\mathrm{c}}}\sqrt{\|(D_1^* - D_1^{-1})U_1\|_{\mathrm{F}}^2 + \|(D_2^* - D_2^{-1})V_1\|_{\mathrm{F}}^2}, \end{aligned}$$

which, together with

$$\begin{aligned} \sqrt{\|\widetilde{V}_2^* V_1\|_{\mathrm{F}}^2 + \|\widetilde{U}_2^* U_1\|_{\mathrm{F}}^2} &= \sqrt{\|X + \widetilde{V}_2^*(I - D_2^*)V_1\|_{\mathrm{F}}^2 + \|Y + \widetilde{U}_2^*(I - D_1^*)U_1\|_{\mathrm{F}}^2} \\ &\leq \sqrt{\|X\|_{\mathrm{F}}^2 + \|Y\|_{\mathrm{F}}^2} + \sqrt{\|\widetilde{V}_2^*(I - D_2^*)V_1\|_{\mathrm{F}}^2 + \|\widetilde{U}_2^*(I - D_1^*)U_1\|_{\mathrm{F}}^2}, \end{aligned}$$

imply (4.7). □

Remark 5.2. Without assuming the multiplicative structure in the perturbation of B to $\widetilde{B}$, we shall end up with

$$\widetilde{\Sigma}_2\widetilde{V}_2^* V_1 - \widetilde{U}_2^* U_1 \Sigma_1 = \widetilde{U}_2^* R_{\mathrm{R}} \quad \text{and} \quad \widetilde{\Sigma}_2 \widetilde{U}_2^* U_1 - \widetilde{V}_2^* V_1 \Sigma_1 = \widetilde{V}_2^* R_{\mathrm{L}},$$

which lead to Wedin $\sin\theta$ theorems, e.g., (4.8).

LEMMA 5.1. *Let $\Omega \in \mathbb{C}^{s\times s}$ and $\Gamma \in \mathbb{C}^{t\times t}$ be two Hermitian matrices, and let E, $\widetilde{E}$, F, $\widetilde{F}, \in \mathbb{C}^{s\times t}$. If there exist $\alpha \geq 0$ and $\delta > 0$ such that*

$$\|\Omega\|_2 \leq \alpha \quad \textit{and} \quad \|\Gamma^{-1}\|_2^{-1} \geq \alpha + \delta \tag{5.8}$$

or

$$\|\Omega^{-1}\|_2^{-1} \geq \alpha + \delta \quad \textit{and} \quad \|\Gamma\|_2 \leq \alpha, \tag{5.9}$$

then $\Omega X - Y\Gamma = \Omega E + F\Gamma$ and $\Omega Y - X\Gamma = \Omega\widetilde{E} + \widetilde{F}\Gamma$ has a unique solution X, $Y \in \mathbb{C}^{s\times t}$, and moreover for any unitarily invariant norm $|||\cdot|||$,

$$\max\{|||X|||\,, |||Y|||\} \leq \frac{1}{\underline{\eta}_p}\max\left\{\sqrt[q]{|||E|||^q + |||F|||^q},\ \sqrt[q]{\left|\left|\left|\widetilde{E}\right|\right|\right|^q + \left|\left|\left|\widetilde{F}\right|\right|\right|^q}\right\}, \tag{5.10}$$

where $\underline{\eta}_p \overset{\text{def}}{=} \varrho_p(\alpha, \alpha+\delta)$. *If, in addition,* $F = \widetilde{F} = 0$, *we have a better bound*

$$\max\left\{|||X|||\,,\ |||Y|||\right\} \le \frac{1}{\underline{\eta}_{\mathrm{c}}} \max\left\{|||E|||\,,\ \left|\left|\left|\widetilde{E}\right|\right|\right|\right\},$$

where $\underline{\eta}_{\mathrm{c}} = \delta/\alpha$ *when* (5.8) *holds and* $\underline{\eta}_{\mathrm{c}} = \delta/(\alpha+\delta)$ *when* (5.9) *holds.*

Proof. The proof of the existence and uniqueness of X, $Y \in \mathbb{C}^{s\times t}$ is left to the reader. We present a proof of (5.10) for the case (5.8). A proof for the other case is analogous. Consider first the subcase $|||X||| \ge |||Y|||$. Postmultiply $\Omega Y - X\Gamma = \Omega\widetilde{E} + \widetilde{F}\Gamma$ by Γ^{-1} to get

$$\Omega Y\Gamma^{-1} - X = \Omega\widetilde{E}\Gamma^{-1} + \widetilde{F}. \tag{5.11}$$

Then we have, by $\|\Omega\|_2 \le \alpha$ and $\|\Gamma^{-1}\|_2^{-1} \ge \alpha+\delta \Rightarrow \|\Gamma^{-1}\|_2 \le \frac{1}{\alpha+\delta}$, that

$$\begin{aligned}\left|\left|\left|\Omega Y\Gamma^{-1} - X\right|\right|\right| &\ge |||X||| - \left|\left|\left|\Omega Y\Gamma^{-1}\right|\right|\right| \ge |||X||| - \|\Omega\|_2\,|||Y|||\,\|\Gamma^{-1}\|_2\\ &\ge |||X||| - \alpha\,|||Y|||\,\frac{1}{\alpha+\delta} \ge |||X||| - \alpha\,|||X|||\,\frac{1}{\alpha+\delta}\\ &= \left(1 - \frac{\alpha}{\alpha+\delta}\right)|||X|||\end{aligned}$$

and

$$\begin{aligned}\left|\left|\left|\Omega\widetilde{E}\Gamma^{-1} + \widetilde{F}\right|\right|\right| &\le \left|\left|\left|\Omega\widetilde{E}\Gamma^{-1}\right|\right|\right| + \left|\left|\left|\widetilde{F}\right|\right|\right| \le \|\Omega\|_2\left|\left|\left|\widetilde{E}\right|\right|\right|\,\|\Gamma^{-1}\|_2 + \left|\left|\left|\widetilde{F}\right|\right|\right|\\ &\le \alpha\left|\left|\left|\widetilde{E}\right|\right|\right|\frac{1}{\alpha+\delta} + \left|\left|\left|\widetilde{F}\right|\right|\right| \le \sqrt[p]{1 + \frac{\alpha^p}{(\alpha+\delta)^p}}\ \sqrt[q]{\left|\left|\left|\widetilde{E}\right|\right|\right|^q + \left|\left|\left|\widetilde{F}\right|\right|\right|^q}.\end{aligned}$$

By (5.11), we deduce that

$$\left(1 - \frac{\alpha}{\alpha+\delta}\right)|||X||| \le \sqrt[p]{1 + \frac{\alpha^p}{(\alpha+\delta)^p}}\ \sqrt[q]{\left|\left|\left|\widetilde{E}\right|\right|\right|^q + \left|\left|\left|\widetilde{F}\right|\right|\right|^q}$$

which produces that if $|||X||| \ge |||Y|||$, $|||X||| \le \frac{1}{\underline{\eta}_p}\sqrt[q]{|||\widetilde{E}|||^q + |||\widetilde{F}|||^q}$. Similarly, if $|||X||| < |||Y|||$, from $\Omega X - Y\Gamma = \Omega E + F\Gamma$, we can obtain $|||Y||| \le \frac{1}{\underline{\eta}_p}\sqrt[q]{|||E|||^q + |||F|||^q}$. Inequality (5.10) now follows. The case $F = \widetilde{F} = 0$ can be handled analogously. □

Proof of Theorem 4.2. By (5.2) and (5.3) and Lemma 5.1, we have

$$\begin{aligned}&\max\left\{\left|\left|\left|\widetilde{U}_2^*U_1\right|\right|\right|,\ \left|\left|\left|\widetilde{V}_2^*V_1\right|\right|\right|\right\}\\ &\le \frac{1}{\underline{\eta}_p}\max\left\{\sqrt[q]{\left|\left|\left|\widetilde{V}_2^*(I - D_2^{-1})V_1\right|\right|\right|^q + \left|\left|\left|\widetilde{U}_2^*(D_1^* - I)U_1\right|\right|\right|^q},\right.\\ &\qquad\qquad \left.\sqrt[q]{\left|\left|\left|\widetilde{U}_2^*(I - D_1^{-1})U_1\right|\right|\right|^q + \left|\left|\left|\widetilde{V}_2^*(D_2^* - I)V_1\right|\right|\right|^q}\right\}\\ &\le \frac{1}{\underline{\eta}_p}\max\left\{\sqrt[q]{\left|\left|\left|(I - D_2^{-1})V_1\right|\right|\right|^q + |||(D_1^* - I)U_1|||^q},\right.\\ &\qquad\qquad \left.\sqrt[q]{\left|\left|\left|(I - D_1^{-1})U_1\right|\right|\right|^q + |||(D_2^* - I)V_1|||^q}\right\},\end{aligned}$$

as required. Lemma 2.3 and (5.4) yield that

$$\left|\!\left|\!\left| \begin{pmatrix} \widetilde{U}_2^* U_1 & \\ & \widetilde{V}_2^* V_1 \end{pmatrix} \right|\!\right|\!\right| \le \frac{1}{\underline{\eta}_p} \left(\left|\!\left|\!\left| \begin{pmatrix} \widetilde{U}_2^* (I - D_1^{-1}) U_1 & \\ & \widetilde{V}_2^* (I - D_2^{-1}) V_1 \end{pmatrix} \right|\!\right|\!\right|^q \right.$$
$$\left. + \left|\!\left|\!\left| \begin{pmatrix} \widetilde{U}_2^* (D_1^* - I) U_1 & \\ & \widetilde{V}_2^* (D_2^* - I) V_1 \end{pmatrix} \right|\!\right|\!\right|^q \right)^{1/q}, \tag{5.12}$$

since the conditions of Theorem 4.2 imply

$$\left\| \begin{pmatrix} & \widetilde{\Sigma}_2 \\ \widetilde{\Sigma}_2 & \end{pmatrix} \right\|_2 \le \alpha, \quad \left\| \begin{pmatrix} & \Sigma_1 \\ \Sigma_1 & \end{pmatrix}^{-1} \right\|_2 \le \frac{1}{\alpha + \delta}.$$

Since $\widetilde{U}_2^* U_1$ and $\sin\Theta(U_1, \widetilde{U}_1)$ have the same nonzero singular values and so do $\widetilde{V}_2^* V_1$ and $\sin\Theta(V_1, \widetilde{V}_1)$,

$$\left|\!\left|\!\left| \begin{pmatrix} \widetilde{U}_2^* U_1 & \\ & \widetilde{V}_2^* V_1 \end{pmatrix} \right|\!\right|\!\right| = \left|\!\left|\!\left| \begin{pmatrix} \sin\Theta(U_1, \widetilde{U}_1) & \\ & \sin\Theta(V_1, \widetilde{V}_1) \end{pmatrix} \right|\!\right|\!\right|. \tag{5.13}$$

Note also

$$\begin{pmatrix} \widetilde{U}_2^* (I - D_1^{-1}) U_1 & \\ & \widetilde{V}_2^* (I - D_2^{-1}) V_1 \end{pmatrix} = \begin{pmatrix} \widetilde{U}_2^* & \\ & \widetilde{V}_2^* \end{pmatrix} \begin{pmatrix} (I - D_1^{-1}) U_1 & \\ & (I - D_2^{-1}) V_1 \end{pmatrix},$$
$$\begin{pmatrix} \widetilde{U}_2^* (D_1^* - I) U_1 & \\ & \widetilde{V}_2^* (D_2^* - I) V_1 \end{pmatrix} = \begin{pmatrix} \widetilde{U}_2^* & \\ & \widetilde{V}_2^* \end{pmatrix} \begin{pmatrix} (D_1^* - I) U_1 & \\ & (D_2^* - I) V_1 \end{pmatrix}.$$

Thus we arrive at

$$\left|\!\left|\!\left| \begin{pmatrix} \widetilde{U}_2^* (I - D_1^{-1}) U_1 & \\ & \widetilde{V}_2^* (I - D_2^{-1}) V_1 \end{pmatrix} \right|\!\right|\!\right| \le \left|\!\left|\!\left| \begin{pmatrix} (I - D_1^{-1}) U_1 & \\ & (I - D_2^{-1}) V_1 \end{pmatrix} \right|\!\right|\!\right|, \tag{5.14}$$

$$\left|\!\left|\!\left| \begin{pmatrix} \widetilde{U}_2^* (D_1^* - I) U_1 & \\ & \widetilde{V}_2^* (D_2^* - I) V_1 \end{pmatrix} \right|\!\right|\!\right| \le \left|\!\left|\!\left| \begin{pmatrix} (D_1^* - I) U_1 & \\ & (D_2^* - I) V_1 \end{pmatrix} \right|\!\right|\!\right|. \tag{5.15}$$

Inequality (4.10) is a consequence of (5.12)–(5.15). By (5.5) and (5.6) and Lemma 5.1, we have

$$\begin{aligned} \max\{|\!|\!| X |\!|\!|, |\!|\!| Y |\!|\!|\} &\le \frac{1}{\underline{\eta}_{\rm c}} \max\left\{ \left|\!\left|\!\left| \widetilde{V}_2^* (D_2^* - D_2^{-1}) V_1 \right|\!\right|\!\right|, \left|\!\left|\!\left| \widetilde{U}_2^* (D_1^* - D_1^{-1}) U_1 \right|\!\right|\!\right| \right\} \\ &\le \frac{1}{\underline{\eta}_{\rm c}} \max\left\{ |\!|\!| (D_2^* - D_2^{-1}) V_1 |\!|\!|, |\!|\!| (D_1^* - D_1^{-1}) U_1 |\!|\!| \right\}. \end{aligned}$$

Notice that

$$\begin{aligned} \left|\!\left|\!\left| \widetilde{V}_2^* V_1 \right|\!\right|\!\right| &\le |\!|\!| X |\!|\!| + \left|\!\left|\!\left| \widetilde{V}_2^* (I - D_2^*) V_1 \right|\!\right|\!\right|, \\ \left|\!\left|\!\left| \widetilde{U}_2^* U_1 \right|\!\right|\!\right| &\le |\!|\!| X |\!|\!| + \left|\!\left|\!\left| \widetilde{U}_2^* (I - D_1^*) U_1 \right|\!\right|\!\right|. \end{aligned}$$

Inequality (4.11) now follows. Similarly apply Lemma 2.3 to (5.7) to get (4.12). □

6. Conclusions. We have developed a relative perturbation theory for eigenspace and singular subspace variations under multiplicative perturbations. In the theory, extensions of Davis–Kahan $\sin\theta$ theorems and Wedin $\sin\theta$ theorems from the classical perturbation theory are made. Our unifying treatment covers almost all previously studied cases over the last six years or so and yet produces sharper bounds. Straightforward extensions of the underlying theory of this paper to diagonalizable matrices are outlined.

The theory is built upon bounds on solutions to various Sylvester equations with structured right-hand sides. This provides technical links to the classical Davis and Kahan development for eigenvalue problems and to Wedin's development for singular value problems. Although these equations are used as tools in this paper to study eigenspace and singular subspace variations, we believe they should deserve at least as much attention as the bounds they lead to.

Acknowledgment. I thank Professor W. Kahan for his consistent encouragement and support and Professor J. Demmel and Professor B. N. Parlett for helpful discussions. The referees' constructive comments for improving the presentation are greatly appreciated.

REFERENCES

[1] J. Barlow and J. Demmel, *Computing accurate eigensystems of scaled diagonally dominant matrices*, SIAM J. Numer. Anal., 27 (1990), pp. 762–791.

[2] R. Bhatia, C. Davis, and A. McIntosh, *Perturbation of spectral subspaces and solution of linear operator equations*, Linear Algebra Appl., 52–53 (1983), pp. 45–67.

[3] C. Davis and W. Kahan, *The rotation of eigenvectors by a perturbation.* III, SIAM J. Numer. Anal., 7 (1970), pp. 1–46.

[4] J. Demmel and W. Gragg, *On computing accurate singular values and eigenvalues of matrices with acyclic graphs*, Linear Algebra Appl., 185 (1993), pp. 203–217.

[5] J. Demmel and W. Kahan, *Accurate singular values of bidiagonal matrices*, SIAM J. Sci. Statist. Comput., 11 (1990), pp. 873–912.

[6] J. Demmel and K. Veselić, *Jacobi's method is more accurate than QR*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 1204–1245.

[7] S. C. Eisenstat and I. C. F. Ipsen, *Relative perturbation bounds for eigenspaces and singular vector subspaces*, in Proceedings of the Fifth SIAM Conference on Applied Linear Algebra, J. G. Lewis, ed., SIAM, Philadelphia, 1994, pp. 62–66.

[8] S. C. Eisenstat and I. C. F. Ipsen, *Relative perturbation techniques for singular value problems*, SIAM J. Numer. Anal., 32 (1995), pp. 1972–1988.

[9] W. Kahan, *Accurate Eigenvalues of a Symmetric Tridiagonal Matrix*, Technical Report CS41, Computer Science Department, Stanford University, Stanford, CA, 1966 (revised June 1968).

[10] R.-C. Li, *On Deflating Bidiagonal Matrices*, manuscript, Department of Mathematics, University of California, Berkeley, CA, 1994.

[11] R.-C. Li, *On perturbations of matrix pencils with real spectra*, Math. Comp., 62 (1994), pp. 231–265.

[12] R.-C. Li, *Relative perturbation theory:* (I)*. Eigenvalue and singular value variations*, SIAM J. Matrix Anal. Appl., 19 (1998), pp. 956–982.

[13] R.-C. Li, *Relative perturbation theory:* (II) *Eigenspace and singular subspace variations*, Technical Report UCB//CSD-94-856, Computer Science Division, Department of EECS, University of California at Berkeley, 1994, also LAPACK Working notes # 85 (revised January 1996 and April 1996, available at http://www.netlib.org/lapack/lawns/lawn85.ps).

[14] R. Mathias, *Spectral Perturbation Bounds for Positive Definite Matrices*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 959–980.

[15] G. W. Stewart and J.-G. Sun, *Matrix Perturbation Theory*, Academic Press, Boston, 1990.

[16] P.-Å. Wedin, *Perturbation bounds in connection with singular value decomposition*, BIT, 12 (1972), pp. 99–111.

[17] P.-Å. Wedin, *On angles between subspaces*, in Matrix Pencils, B. Kågström and A. Ruhe, eds., Springer–Verlag, New York, 1983, pp. 263–285.

SIAM J. MATRIX ANAL. APPL.
Vol. 20, No. 2, pp. 493–512

STRUCTURED BACKWARD ERROR AND CONDITION OF GENERALIZED EIGENVALUE PROBLEMS*

DESMOND J. HIGHAM† AND NICHOLAS J. HIGHAM‡

Abstract. Backward errors and condition numbers are defined and evaluated for eigenvalues and eigenvectors of generalized eigenvalue problems. Both normwise and componentwise measures are used. Unstructured problems are considered first, and then the basic definitions are extended so that linear structure in the coefficient matrices (for example, Hermitian, Toeplitz, Hamiltonian, or band structure) is preserved by the perturbations.

Key words. generalized eigenvalue problem, quadratic eigenvalue problem, backward error, condition number, structured matrices

AMS subject classifications. 65F15, 65F35

PII. S0895479896313188

1. Introduction.

1.1. Backward error and condition. Backward errors and condition numbers play an important role in modern numerical linear algebra. Backward errors reveal the stability of a numerical method. Condition numbers explain the sensitivity of the solution of a problem to perturbations in the data, and, in the case where this perturbation is a backward error, the product of condition number times backward error provides a first order error bound for the computed solution. The theory of backward error and conditioning is now well developed for linear systems and least squares problems. For eigenproblems, although there is a large literature on perturbation theory (see [30] and the references therein), a number of aspects of backward errors and condition numbers have not been considered, particularly for the generalized eigenvalue problem. Our aim here is to give a thorough development of backward error and condition for the generalized eigenvalue problem, for both normwise and componentwise measures, with particular emphasis on respecting structure in the coefficient matrices.

We consider the generalized eigenvalue problem $Ax = \lambda Bx$, where $A, B \in \mathbb{C}^{n\times n}$. If $x \neq 0$ then we say that λ is an eigenvalue and x the corresponding eigenvector of the pair (A, B). We develop backward errors for approximate eigenpairs and condition numbers for eigenvalues and eigenvectors. We do not treat deflating subspaces (the appropriate generalization of invariant subspaces), leaving this important topic to future work; for existing results on deflating subspaces see Stewart and Sun [30, Section VI.2.4] and Kågström and Poromaa [21].

In section 2 we develop normwise backward errors and condition numbers for a general class of normwise measures of the perturbations to A and B. In particular, we show that, for the 2-norm, requiring the perturbations to respect Hermitian structure in A and B has no effect on the backward error of an approximate eigenpair corresponding to a real approximate eigenvalue and has no effect on the condition number

[*] Received by the editors December 3, 1996; accepted for publication (in revised form) by J. Varah February 19, 1998; published electronically November 23, 1998.

http://www.siam.org/journals/simax/20-2/31318.html

[†] Department of Mathematics, University of Strathclyde, Glasgow G1 1XH, Scotland (na.dhigham@na-net.ornl.gov, http://www.strath.ac.uk/~aas96106/).

[‡] Department of Mathematics, University of Manchester, Manchester M13 9PL, England (higham@ma.man.ac.uk, http://www.ma.man.ac.uk/~higham/).

of an eigenvalue of a definite pair. It is widely appreciated that for problems with badly scaled or sparse data componentwise analysis can yield much stronger results than normwise analysis [18]. In section 3 we give a treatment analogous to that of section 2 but for componentwise measures.

Many applications lead to eigenproblems containing matrices with linear structure, and only perturbations that preserve the structure may be physically meaningful [31]. For example, the quadratic eigenvalue problem has the form

$$(\lambda^2 C + \lambda D + E)v = 0, \qquad C, D, E \in \mathbb{C}^{n\times n}. \tag{1.1}$$

By writing $u = \lambda v$ we can express the problem as a generalized eigenvalue problem:

$$\begin{bmatrix} D & E \\ E & 0 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \lambda \begin{bmatrix} -C & 0 \\ 0 & E \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}. \tag{1.2}$$

The coefficient matrices are clearly highly structured, and arbitrary perturbations to these matrices do not correspond to perturbations of the original quadratic eigenvalue problem—we must respect the structure in order to obtain meaningful results. Another highly structured eigenproblem is that of a Hamiltonian matrix

$$\mathcal{H} = \begin{bmatrix} F & G \\ H & -F^T \end{bmatrix}, \qquad F \in \mathbb{R}^{n\times n}, \quad G = G^T \in \mathbb{R}^{n\times n}, \quad H = H^T \in \mathbb{R}^{n\times n}.$$

It has been a long-standing open problem to develop numerical methods for the Hamiltonian eigenproblem that require only $O(n^3)$ operations and for which the computed eigenvalues or invariant subspaces are the exact ones of a nearby Hamiltonian matrix; that is, the backward error preserves the Hamiltonian structure. Benner, Mehrmann, and Xu [4] have developed a numerical method that comes close to satisfying these requirements, having a Hamiltonian backward error with respect to a Hamiltonian matrix related to $\mathcal{H}$. For testing this method and deriving error bounds it is therefore useful to have backward errors and condition numbers that respect Hamiltonian structure.

In section 4 we define backward errors and condition numbers that respect arbitrary linear structure in the matrices A and B. The backward error is characterized as the minimum norm solution to a rectangular system, while explicit formulas are obtained for the condition numbers. We have previously carried out similar analysis for linear systems [17]; there we used a Kronecker product-based approach, but here we use a different technique based on “pattern matrices.”

Brief numerical experiments are reported in section 5 in order to illustrate the analysis.

1.2. Preliminaries. For the backward error analysis we make no assumptions on A and B, but for the definition and derivation of condition numbers we assume that the pair (A, B) is *regular*, that is, that $\det(A - \lambda B)$ is not identically zero in λ.

Significant advantages accrue from treating the generalized eigenvalue problem in the form $\beta Ax = \alpha Bx$, where an eigenvalue is now defined by a pair of scalars (α, β) [30, p. 272]. For example, when B is singular a nonzero null vector of B is an eigenvector with $(\alpha, \beta) = (\alpha, 0)$, whereas in the original formulation we have an eigenvalue $\lambda = \infty$. The (α, β) framework thus elegantly handles infinite eigenvalues and treats A and B symmetrically. Moreover, perturbation expansions of α and β individually provide complete information about eigenvalue sensitivity [29], [30, p. 293], although some of this information is inevitably lost in the definition of a single condition number. We have chosen to derive and state all our results in terms of λ for three main reasons. First, λ is the desired quantity in most practical applications of the generalized eigenvalue problem. Second, analysis with the (α, β) form is naturally carried out using the chordal metric, but this metric is scale dependent (see section 2.2). Finally, the standard eigenproblem ($B = I$) is an important special case and we would like our results to be directly applicable to it.

For parts of the analysis we will consider a Hermitian pair (A, B) (that is, a pair in which A and B are Hermitian) and assume that it is *definite*, that is, that

$$\min\bigl\{ \bigl((x^*Ax)^2 + (x^*Bx)^2\bigr)^{1/2} : x \in \mathbb{C}^n,\ \|x\|_2 = 1 \bigr\}$$

is positive, which is certainly true if A or B is positive definite. A definite pair (A, B) has the property that there is a nonsingular matrix X such that X^*AX and X^*BX are diagonal. For details of the theory of the generalized eigenvalue problem see [16, Sections 7.7, 8.7], [25, Chap. 15], or [30, Chap. 6].

We require some definitions involving norms. The norm $\|\cdot\|^D$ dual to a given vector norm $\|\cdot\|$ on $\mathbb{C}^n$ is defined by

$$\|x\|^D = \max_{w \neq 0} \frac{|w^*x|}{\|w\|},$$

and we say that z is a vector dual to y if

$$z^*y = \|z\|^D \|y\| = 1.$$

The mixed subordinate matrix norm $\|\cdot\|_{\alpha,\beta}$ on $\mathbb{C}^{n\times n}$ is defined by

$$\|A\|_{\alpha,\beta} = \max_{x \neq 0} \frac{\|Ax\|_\beta}{\|x\|_\alpha}.$$

The generality obtained by allowing $\alpha \neq \beta$ causes no complications in the statements and proofs of our results and permits coverage of the potentially useful norms

$$\|A\|_{1,\beta} = \max_j \|A(:,j)\|_\beta, \qquad \|A\|_{\alpha,\infty} = \max_i \|A(i,:)^*\|_\alpha^D,$$

which include the special case $\|A\|_{1,\infty} = \max_{i,j} |a_{ij}|$. We note for later reference that

$$\|xy^*\|_{\alpha,\beta} = \|x\|_\beta \|y\|_\alpha^D.$$

For complex α we define

$$\operatorname{sign}(\alpha) = \begin{cases} \dfrac{\overline{\alpha}}{|\alpha|}, & \alpha \neq 0, \\ 0, & \alpha = 0. \end{cases}$$

The sign of a vector z is defined componentwise as $\operatorname{sign}(z) = (\operatorname{sign}(z_i))$.

2. Normwise analysis.

2.1. Backward errors. We begin by considering backward errors for the generalized eigenvalue problem. The formulas obtained are useful for testing the stability of practical eigensolvers [1], [8], [21].

The normwise backward error of an approximate eigenpair $(\widetilde{x}, \widetilde{\lambda})$ is defined by

$$\begin{aligned}\eta(\widetilde{x},\widetilde{\lambda}) := \min\{\, \epsilon : (A+\Delta A)\widetilde{x} = \widetilde{\lambda}(B+\Delta B)\widetilde{x},\ \ &\|\Delta A\|_{\alpha,\beta} \le \epsilon \|E\|_{\alpha,\beta}, \\ &\|\Delta B\|_{\alpha,\beta} \le \epsilon \|F\|_{\alpha,\beta}\,\},\end{aligned} \tag{2.1}$$

where E and F are given matrices of tolerances. Note that if A and B are real and $\widetilde{\lambda}$ is nonreal, the optimal perturbations in (2.1) may be nonreal. How to restrict to real perturbations is considered at the end of section 4.

The following theorem, which is a straightforward modification of a result of Rigal and Gaches on the normwise backward error for a linear system [26], gives an explicit expression for $\eta(\widetilde{x},\widetilde{\lambda})$. For the case $\alpha=\beta$, this theorem and the following lemma are given by Frayssé and Toumazou [14].

THEOREM 2.1. *The normwise backward error* $\eta(\widetilde{x},\widetilde{\lambda})$ *is given by*

$$\eta(\widetilde{x},\widetilde{\lambda}) = \frac{\|r\|_\beta}{(\|E\|_{\alpha,\beta} + |\widetilde{\lambda}|\|F\|_{\alpha,\beta})\|\widetilde{x}\|_\alpha}, \tag{2.2}$$

where $r = \widetilde{\lambda}B\widetilde{x} - A\widetilde{x}$.

Proof. It is straightforward to show that the right-hand side of (2.2) is a lower bound for $\eta(\widetilde{x},\widetilde{\lambda})$. This lower bound is easily seen to be attained for the feasible perturbations

$$\Delta A = \frac{\|E\|_{\alpha,\beta}}{\|E\|_{\alpha,\beta} + |\widetilde{\lambda}|\|F\|_{\alpha,\beta}} rz^*, \qquad \Delta B = -\operatorname{sign}(\widetilde{\lambda}) \frac{\|F\|_{\alpha,\beta}}{\|E\|_{\alpha,\beta} + |\widetilde{\lambda}|\|F\|_{\alpha,\beta}} rz^*, \tag{2.3}$$

where z is a vector dual to $\widetilde{x}$ with respect to the α-norm. □

If we are interested only in the approximate eigenvalue $\widetilde{\lambda}$ and are not concerned about $\widetilde{x}$, then a more appropriate measure of backward error may be

$$\eta(\widetilde{\lambda}) := \min_{\widetilde{x}\ne 0} \eta(\widetilde{x},\widetilde{\lambda}).$$

This quantity has a closed-form expression, as shown by the next result.

LEMMA 2.2. *If* $\widetilde{\lambda}$ *is not an eigenvalue of the pair* (A,B) *then*

$$\eta(\widetilde{\lambda}) = \frac{1}{\|(\widetilde{\lambda}B - A)^{-1}\|_{\beta,\alpha}\,(\|E\|_{\alpha,\beta} + |\widetilde{\lambda}|\|F\|_{\alpha,\beta})}.$$

Proof. The result follows from Theorem 2.1 on using the equality, for nonsingular $C \in \mathbb{C}^{n\times n}$, $\min_{x\ne 0} \|Cx\|_\beta/\|x\|_\alpha = \|C^{-1}\|_{\beta,\alpha}^{-1}$. □

Similarly, for a given $\widetilde{x}$ we can consider minimizing the backward error over all $\widetilde{\lambda}$:

$$\eta(\widetilde{x}) = \min_{\widetilde{\lambda}} \eta(\widetilde{x},\widetilde{\lambda}) = \min_{\widetilde{\lambda}} \frac{\|\widetilde{\lambda}B\widetilde{x} - A\widetilde{x}\|_\beta}{(\|E\|_{\alpha,\beta} + |\widetilde{\lambda}|\|F\|_{\alpha,\beta})\|\widetilde{x}\|_\alpha}. \tag{2.4}$$

This optimization problem appears to be analytically intractable, even for the 2-norm ($\alpha = \beta = 2$). However, we can obtain an upper bound for $\eta(\widetilde{x})$ by choosing $\widetilde{\lambda}$ to minimize the numerator in (2.4) only; for the 2-norm this value of $\widetilde{\lambda}$ is $\widetilde{x}^* B^* A\widetilde{x}/(\widetilde{x}^* B^* B\widetilde{x})$ if $B\widetilde{x} \neq 0$.

If A and B are Hermitian then it is desirable that the perturbations ΔA and ΔB in the definition of backward error preserve this property. Therefore the following backward error is of interest:

$$\begin{aligned}\eta^H(\widetilde{x}, \widetilde{\lambda}) := \min\{\, \epsilon : (A + \Delta A)\widetilde{x} = \widetilde{\lambda}(B + \Delta B)\widetilde{x},\ \ \Delta A = \Delta A^*,\ \ \Delta B = \Delta B^*,\\ \|\Delta A\|_{\alpha,\beta} \leq \epsilon\|E\|_{\alpha,\beta},\ \ \|\Delta B\|_{\alpha,\beta} \leq \epsilon\|F\|_{\alpha,\beta}\,\}.\end{aligned} \tag{2.5}$$

Clearly, $\eta^H(\widetilde{x}, \widetilde{\lambda}) \geq \eta(\widetilde{x}, \widetilde{\lambda})$, and the optimal perturbations for $\eta(\widetilde{x}, \widetilde{\lambda})$ in (2.3) are not Hermitian in general. We wish to determine by how much $\eta^H(\widetilde{x}, \widetilde{\lambda})$ can exceed $\eta(\widetilde{x}, \widetilde{\lambda})$. The following result, which is an adaptation of a result of Bunch, Demmel, and Van Loan [5] for linear systems, shows that requiring the backward error perturbations to be Hermitian has no effect on the backward error in the 2-norm, provided that $\widetilde{\lambda}$ is real. The same result has also been obtained by Smoktunowicz [28], where it is stated only for definite pairs.

THEOREM 2.3. *If A and B are Hermitian and $\widetilde{\lambda}$ is real, then, for the 2-norm ($\alpha = \beta = 2$), we have $\eta_2^H(\widetilde{x}, \widetilde{\lambda}) = \eta_2(\widetilde{x}, \widetilde{\lambda})$.*

Proof. Let $r = \widetilde{\lambda}B\widetilde{x} - A\widetilde{x}$. We first find a Hermitian H that satisfies the constraint $H\widetilde{x} := (\Delta A - \widetilde{\lambda}\Delta B)\widetilde{x} = r$ in (2.5). We take $H := (\|r\|_2/\|\widetilde{x}\|_2)P$, where P is a suitably chosen Householder matrix—such a P exists since $\widetilde{\lambda}$ real implies $\widetilde{x}^* r$ real (if $r = \widetilde{x}$, we have to take $H = I$ instead). To satisfy $H = \Delta A - \widetilde{\lambda}\Delta B$ with Hermitian ΔA and ΔB, we define

$$\Delta A = \frac{\|E\|_2}{\|E\|_2 + |\widetilde{\lambda}|\|F\|_2} H, \qquad \Delta B = -\operatorname{sign}(\widetilde{\lambda}) \frac{\|F\|_2}{\|E\|_2 + |\widetilde{\lambda}|\|F\|_2} H. \tag{2.6}$$

Now

$$\|H\|_2 = \|r\|_2/\|\widetilde{x}\|_2 = \eta_2(\widetilde{x}, \widetilde{\lambda})(\|E\|_2 + |\widetilde{\lambda}|\|F\|_2),$$

using (2.2). From (2.6) it follows that $\eta_2^H(\widetilde{x}, \widetilde{\lambda}) \leq \eta_2(\widetilde{x}, \widetilde{\lambda})$. But $\eta_2^H(\widetilde{x}, \widetilde{\lambda}) \geq \eta_2(\widetilde{x}, \widetilde{\lambda})$ by definition, so equality must hold. □

Note that if B is Hermitian positive definite, the perturbation ΔB that achieves $\eta_2^H(\widetilde{x}, \widetilde{\lambda})$ in (2.5) does not necessarily keep $B + \Delta B$ positive definite. However, $B + \Delta B$ certainly will be positive definite if

$$\eta_2^H(\widetilde{x}, \widetilde{\lambda})\|F\|_2 = \|\Delta B\|_2 < \lambda_{\min}(B), \tag{2.7}$$

where $\lambda_{\min}$ denotes the smallest eigenvalue.

It is worth pausing to discuss the significance of Theorem 2.3. One way to solve the generalized eigenvalue problem is by the QZ algorithm, which computes the generalized Schur decomposition and is normwise backward stable. If the QZ algorithm is applied to a Hermitian pair (A, B) then it does not preserve Hermitian structure. Theorem 2.3 implies that, nevertheless, each computed eigenpair containing a real eigenvalue is exact for a Hermitian pair that is a slight perturbation of (A, B), with the perturbation being different, in general, for each eigenpair.

Definitions and results analogous to those above hold for the backward error corresponding to an approximate eigenvalue $\widetilde{\lambda}$ and corresponding left eigenvector $\widetilde{y}$ ($\widetilde{y}^*A \approx \widetilde{\lambda}\widetilde{y}^*B$).

Also of interest is the normwise backward error of a triple $(\widetilde{x}, \widetilde{y}, \widetilde{\lambda})$, where $\widetilde{y}$ is an approximate left eigenvector. For the 2-norm, this backward error is defined by

$$\eta(\widetilde{x},\widetilde{y},\widetilde{\lambda}) := \min\{\,\epsilon : (A+\Delta A)\widetilde{x} = \widetilde{\lambda}(B+\Delta B)\widetilde{x},\ \ \widetilde{y}^*(A+\Delta A) = \widetilde{\lambda}\widetilde{y}^*(B+\Delta B),$$
$$\|\Delta A\|_2 \le \epsilon\|E\|_2,\ \ \|\Delta B\|_2 \le \epsilon\|F\|_2\,\}.$$

This backward error is evaluated explicitly in the following result.

THEOREM 2.4. *For the* 2*-norm* ($\alpha = \beta = 2$)*, we have*

$$\eta_2(\widetilde{x},\widetilde{y},\widetilde{\lambda}) = \frac{1}{\|E\|_2 + |\widetilde{\lambda}|\|F\|_2} \max\left\{ \frac{\|r\|_2}{\|\widetilde{x}\|_2}, \frac{\|s\|_2}{\|\widetilde{y}\|_2} \right\},$$

where $r = \widetilde{\lambda}B\widetilde{x} - A\widetilde{x}$ *and* $s^* = \widetilde{\lambda}\widetilde{y}^*B - \widetilde{y}^*A$.

Proof. By taking 2-norms in the equations $r = \Delta A\widetilde{x} - \widetilde{\lambda}\Delta B\widetilde{x}$ and $s^* = \widetilde{y}^*\Delta A - \widetilde{\lambda}\widetilde{y}^*\Delta B$, we find that the claimed expression for $\eta_2(\widetilde{x},\widetilde{y},\widetilde{\lambda})$ is certainly a lower bound for it. We must show that this lower bound is attained.

Let $G = \Delta A - \widetilde{\lambda}\Delta B$. Then G satisfies the constraints

$$G\widetilde{x} = r, \quad \widetilde{y}^*G = s^*, \quad \text{and} \quad \widetilde{y}^*r = s^*\tilde{x}. \tag{2.8}$$

A result of Kahan, Parlett and Jiang [22, Thm. 2′] (see also Saad [27, Thm. 3.10]) shows that the minimum value of $\|G\|_2$ subject to G satisfying the constraints (2.8) is

$$\max\left\{ \frac{\|r\|_2}{\|\widetilde{x}\|_2}, \frac{\|s\|_2}{\|\widetilde{y}\|_2} \right\}.$$

Let $G_{\rm opt}$ be a matrix that achieves this minimum, and define

$$\Delta A = \frac{\|E\|_2}{\|E\|_2 + |\widetilde{\lambda}|\|F\|_2} G_{\rm opt}, \qquad \Delta B = -\operatorname{sign}(\widetilde{\lambda})\frac{\|F\|_2}{\|E\|_2 + |\widetilde{\lambda}|\|F\|_2} G_{\rm opt}.$$

Then $\Delta A - \widetilde{\lambda}\Delta B = G$,

$$\|\Delta A\|_2 = \frac{\|E\|_2}{\|E\|_2 + |\widetilde{\lambda}|\|F\|_2} \max\left\{ \frac{\|r\|_2}{\|\widetilde{x}\|_2}, \frac{\|s\|_2}{\|\widetilde{y}\|_2} \right\},$$

and $\|\Delta B\|_2$ satisfies the analogous equality. We have therefore shown that the lower bound for $\eta_2(\widetilde{x},\widetilde{y},\widetilde{\lambda})$ is attained. □

We remark that the formula for $\eta_2(\widetilde{x},\widetilde{y},\widetilde{\lambda})$ in Theorem 2.4 is the maximum of $\eta_2(\widetilde{x},\widetilde{\lambda})$ and the analogous backward error for $(\widetilde{y},\widetilde{\lambda})$.

2.2. Condition numbers. Let λ be a simple, finite, nonzero eigenvalue of the pair (A, B), with corresponding right eigenvector x and left eigenvector y, so that $Ax = \lambda Bx$ and $y^*A = \lambda y^*B$. A normwise condition number of λ can be defined as follows:

$$\kappa(\lambda) := \lim_{\epsilon\to 0}\sup\left\{ \frac{|\Delta\lambda|}{\epsilon|\lambda|} : (A+\Delta A)(x+\Delta x) = (\lambda+\Delta\lambda)(B+\Delta B)(x+\Delta x),\right.$$
$$\left. \|\Delta A\|_{\alpha,\beta} \le \epsilon\|E\|_{\alpha,\beta},\ \ \|\Delta B\|_{\alpha,\beta} \le \epsilon\|F\|_{\alpha,\beta} \right\}. \tag{2.9}$$

This definition is a little loose, because if λ' is an eigenvalue distinct from λ with corresponding eigenvector x', then we can take $\Delta A = \Delta B = 0$, $x + \Delta x \equiv x'$, and $\lambda + \Delta\lambda \equiv \lambda'$ to obtain $\kappa(\lambda) = \infty$. The definition therefore needs to be augmented with the requirement that $\Delta x \to 0$ as $\epsilon \to 0$. For simplicity of presentation we omit this requirement from the definitions of condition numbers.

From the definition of $\kappa(\lambda)$ we have, for the perturbed system in (2.9),

$$\frac{|\Delta\lambda|}{|\lambda|} \le \kappa(\lambda)\epsilon + O(\epsilon^2). \tag{2.10}$$

Expanding the constraint in (2.9) and premultiplying by y^* lead to

$$\begin{aligned}\Delta\lambda &= \frac{y^*\Delta A x - \lambda y^*\Delta B x + y^*\Delta A\Delta x - \lambda y^*\Delta B\Delta x}{y^*Bx + y^*B\Delta x + y^*\Delta B x + y^*\Delta B\Delta x}\\ &= \frac{y^*\Delta A x - \lambda y^*\Delta B x}{y^*Bx} + O(\epsilon^2). \qquad (2.11)\end{aligned}$$

To evaluate $\kappa(\lambda)$ we need to obtain a sharp bound for the first order term in this expansion. The following result is given in [14] for the case $\alpha = \beta$.

THEOREM 2.5. *The normwise condition number $\kappa(\lambda)$ is given by*

$$\kappa(\lambda) = \frac{\|y\|_\beta^D\|x\|_\alpha(\|E\|_{\alpha,\beta} + |\lambda|\|F\|_{\alpha,\beta})}{|\lambda||y^*Bx|}.$$

Proof. The given expression is clearly an upper bound for $\kappa(\lambda)$. We now show that the bound is attained. Let $G = \|x\|_\alpha uv^*$, where u is of unit β-norm and satisfies $u^*y = \|y\|_\beta^D$ and v is dual to x with respect to the α-norm. Then $y^*Gx = \|x\|_\alpha\|y\|_\beta^D$ and $\|G\|_{\alpha,\beta} = 1$. Let $\Delta A = \epsilon\|E\|_{\alpha,\beta}G$ and $\Delta B = -\operatorname{sign}(\lambda)\epsilon\|F\|_{\alpha,\beta}G$. Then $\|\Delta A\|_{\alpha,\beta} \le \epsilon\|E\|_{\alpha,\beta}$ and $\|\Delta B\|_{\alpha,\beta} \le \epsilon\|F\|_{\alpha,\beta}$ and the modulus of the first order term in (2.11) is $\epsilon\|y\|_\beta^D\|x\|_\alpha(\|E\|_{\alpha,\beta} + |\lambda|\|F\|_{\alpha,\beta})/|y^*Bx|$; dividing (2.11) by $\epsilon|\lambda|$ and taking the limit as $\epsilon \to 0$ then gives the desired equality. □

If λ is infinite then $\kappa(\lambda)$ is not defined, but one can consider the problem $\lambda^{-1}Ax = Bx$, which has a corresponding zero eigenvalue. For zero eigenvalues $\kappa(\lambda)$ is also undefined, and the absolute condition number (defined as in (2.9) but with $|\Delta\lambda|/\epsilon$ as the quantity to be maximized) is then the appropriate one to consider.

As for the backward error, if A and B are Hermitian it is natural to restrict the perturbations ΔA and ΔB in (2.9) to be Hermitian. The next lemma shows that for a definite pair this has no effect on the condition number in the 2-norm.

LEMMA 2.6. *Let the Hermitian pair (A, B) be definite. Let $\kappa^H(\lambda)$ denote the condition number defined as in* (2.9) *but with the additional requirement that ΔA and ΔB are Hermitian. Then, for the 2-norm ($\alpha = \beta = 2$), $\kappa_2^H(\lambda) = \kappa_2(\lambda)$.*

Proof. Since (A, B) is a definite pair, we can take $y = x$. For the 2-norm, the matrix G constructed in the proof of Theorem 2.5 is therefore $G = \|x\|_2^{-2}xx^*$, which is Hermitian. The result is immediate. □

It is instructive to compare the condition number $\kappa(\lambda)$ with one of Stewart and Sun [30, pp. 293–294] (cf. [14]). In our notation, they derive the approximate bound (correct to first order)

$$\chi(\lambda, \widetilde{\lambda}) := \frac{|\lambda - \widetilde{\lambda}|}{\sqrt{|\lambda|^2 + 1}\sqrt{|\widetilde{\lambda}|^2 + 1}} \lesssim \frac{\|x\|_2\|y\|_2}{\sqrt{|y^*Ax|^2 + |y^*Bx|^2}}\|\,[\,\Delta A \quad \Delta B\,]\,\|_2, \tag{2.12}$$

where $\widetilde{\lambda} = \lambda + \Delta\lambda$, and they regard the first factor in the upper bound as a condition number. The function χ is the chordal metric and has the property that $\chi(\lambda, \mu) = \chi(\lambda^{-1}, \mu^{-1})$, which is appropriate for the generalized eigenvalue problem since $Ax = \lambda Bx$ and $\lambda^{-1}Ax = Bx$ are equally valid representations. Note also that the bound (2.12) is symmetric in A and B. Unfortunately, the chordal metric is scale dependent: $\chi(\alpha\lambda, \alpha\mu) \neq \chi(\lambda, \mu)$ in general. Hence $\chi(\lambda, \widetilde{\lambda})$ tends to differ greatly from the relative error when $|\lambda|$ and $|\widetilde{\lambda}|$ are both large or both small.

A normwise condition number for the eigenvector x corresponding to the simple eigenvalue λ can be defined by

$$\begin{aligned}\kappa(x) := \lim_{\epsilon\to 0} \sup\Bigg\{ \frac{\|\Delta x\|_\alpha}{\epsilon\|x\|_\alpha} &: (A + \Delta A)(x + \Delta x) = (\lambda + \Delta\lambda)(B + \Delta B)(x + \Delta x),\\ &\quad g^*Bx = g^*B(x + \Delta x) \equiv 1,\\ &\quad \|\Delta A\|_{\alpha,\beta} \le \epsilon\|E\|_{\alpha,\beta}, \ \ \|\Delta B\|_{\alpha,\beta} \le \epsilon\|F\|_{\alpha,\beta} \Bigg\}.\end{aligned} \tag{2.13}$$

Because an eigenvector corresponding to a simple eigenvalue is unique only up to scalar multiples, it is important to normalize the eigenvectors for the perturbation theory. We use a linear normalization in (2.13) based on a constant vector g, which could, for example, be x or the left eigenvector y. The matrix B is included in the normalization equation because it simplifies the subsequent analysis; if B is nonsingular then we can set $g^* := g^*B^{-1}$ in order to remove B. See Chaitin-Chatelin and Frayssé [6, Section 4.4.2] and Stewart and Sun [30, pp. 240–241] for discussions of the normalization issue.

The next result gives an expression for the condition number; it generalizes a result in [6, Section 4.4.2], [7, Section 4.2.1] that applies to the standard eigenproblem.

THEOREM 2.7. *The normwise condition number $\kappa(x)$ is given by*

$$\kappa(x) = \|V(W^*(A - \lambda B)V)^{-1}W^*\|_{\beta,\alpha}\big(\|E\|_{\alpha,\beta} + |\lambda|\|F\|_{\alpha,\beta}\big), \tag{2.14}$$

where the full rank matrices $V, W \in \mathbb{C}^{n\times(n-1)}$ are chosen so that $g^*BV = 0$ and $W^*Bx = 0$.

Proof. From Theorem A.1 in the appendix we see that we have to find a sharp bound for $\|V(W^*(A-\lambda B)V)^{-1}W^*(\Delta A-\lambda\Delta B)x\|_\alpha/(\epsilon\|x\|_\alpha)$. This quantity is clearly bounded by the claimed expression for $\kappa(x)$. Writing $Z = V(W^*AV - \lambda I)^{-1}W^*$, equality is attained for

$$\Delta A = \epsilon\|E\|_{\alpha,\beta}\|x\|_\alpha ph^*, \qquad \Delta B = -\operatorname{sign}(\lambda)\epsilon\|F\|_{\alpha,\beta}\|x\|_\alpha ph^*,$$

where p satisfies $\|p\|_\beta = 1$ and $\|Zp\|_\alpha = \|Z\|_{\beta,\alpha}$, and h is dual to x with respect to the α-norm. ∎

Note that the expression for $\kappa(x)$ in (2.14) is finite and does not depend on the particular choice of V and W, as shown in the appendix.

As a comparison, we recall that Golub and Van Loan [16, p. 346] and Wilkinson [32, pp. 68–70] give perturbation expansions for the eigenvector corresponding to a simple eigenvalue, in the case $B = I$. These expansions do not readily lead to the identification of a condition number. Indeed, as Wilkinson notes [32, pp. 70, 85], examination of individual terms can give a misleading impression of the sensitivity because of the possibility of cancellation in the overall sum.

We show how Theorem 2.7 leads to an informative bound for $\kappa(x)$ when A is Hermitian, B is Hermitian positive definite, and we take $g = x$. In this case there is a nonsingular X so that

$$X^*BX = I, \qquad X^*AX = D = \operatorname{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n).$$

It is easy to verify that we can set $X =: [\,x \quad V\,]$ and $W = V$. Putting $E = A$ and $F = B$ and using the 2-norm, we have

$$\begin{aligned}\kappa_2(x) &= \|V(V^*X^{-*}(D-\lambda I)X^{-1}V)^{-1}V^*\|_2(\|A\|_2 + |\lambda|\|B\|_2)\\ &= \|V\operatorname{diag}(\lambda_2-\lambda,\ldots,\lambda_n-\lambda)^{-1}V^*\|_2(\|A\|_2+|\lambda|\|B\|_2)\\ &\le \frac{\|V\|_2^2}{\min_{\lambda_i\ne\lambda}|\lambda-\lambda_i|}(\|A\|_2+|\lambda|\|B\|_2).\end{aligned}$$

In the case of the standard eigenvalue problem, where $B = I$, we can take $\|V\|_2 = 1$ (and we would take $F = 0$, so that there would be no $|\lambda|\|B\|_2$ term).

3. Componentwise analysis.

3.1. Backward errors.

3.1. Backward errors. The componentwise backward error of an approximate eigenpair $(\widetilde{x}, \widetilde{\lambda})$ is defined by

$$\omega(\widetilde{x},\widetilde{\lambda}) := \min\{\,\epsilon : (A+\Delta A)\widetilde{x} = \widetilde{\lambda}(B+\Delta B)\widetilde{x},\ \ |\Delta A| \le \epsilon E,\ \ |\Delta B| \le \epsilon F\,\}, \tag{3.1}$$

where E and F are now assumed to have nonnegative entries and inequalities hold componentwise.

The Oettli–Prager theorem [24] gives an explicit expression for the componentwise backward error of an approximate solution to a linear system. The next result is an analogue of that result for the eigenvalue problem and is given for the special case $B = I$, $F = 0$ in [10].

THEOREM 3.1. *The componentwise backward error is given by*

$$\omega(\widetilde{x},\widetilde{\lambda}) = \max_i \frac{|r_i|}{\big((E+|\widetilde{\lambda}|F)|\widetilde{x}|\big)_i}, \tag{3.2}$$

where $r = \widetilde{\lambda}B\widetilde{x} - A\widetilde{x}$, and $\xi/0$ is interpreted as zero if $\xi = 0$ and infinity otherwise.

Proof. It is easy to show that the right-hand side of (3.2) is a lower bound for $\omega(\widetilde{x},\widetilde{\lambda})$ and that this bound is attained for the feasible perturbations

$$\Delta A = D_1 E D_2, \qquad \Delta B = -\operatorname{sign}(\widetilde{\lambda}) D_1 F D_2, \tag{3.3}$$

where $D_1 = \operatorname{diag}\big(r_i/((E+|\widetilde{\lambda}|F)|\widetilde{x}|)_i\big)$ and $D_2 = \operatorname{diag}\big(\operatorname{sign}(\widetilde{x})\big)$. □

As for the normwise backward error, it is also of interest to consider the minima of $\omega(\widetilde{x},\widetilde{\lambda})$ over all $\widetilde{x}$ and over all $\widetilde{\lambda}$:

$$\omega(\widetilde{\lambda}) = \min_{\widetilde{x}\ne 0}\omega(\widetilde{x},\widetilde{\lambda}), \qquad \omega(\widetilde{x}) = \min_{\widetilde{\lambda}\ne 0}\omega(\widetilde{x},\widetilde{\lambda}).$$

We have been unable to derive any useful bounds for $\omega(\widetilde{\lambda})$ and $\omega(\widetilde{x})$.

Drmač [11] derives an algorithm for solving the generalized eigenvalue problem when $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times n}$ are both symmetric positive definite. He shows that the computed eigenvalues are the exact ones of a pair $(A+\Delta A, B+\Delta B)$ satisfying

$$|\Delta a_{ij}| \le f(n)u\sqrt{a_{ii}a_{jj}}, \qquad |\Delta b_{ij}| \le f(n)u\sqrt{b_{ii}b_{jj}},$$

where u is the unit roundoff. To test an implementation of this method it is therefore appropriate, for each computed eigenpair $(\widehat{x}, \widehat{\lambda})$, to compare $\omega(\widehat{x}, \widehat{\lambda})$ with a suitable multiple of u, taking $e_{ij} = \sqrt{a_{ii}a_{jj}}$, $f_{ij} = \sqrt{b_{ii}b_{jj}}$.

3.2. Condition numbers. A componentwise condition number for a simple eigenvalue λ analogous to the normwise condition number (2.9) is defined by

$$\operatorname{cond}(\lambda) := \lim_{\epsilon \to 0} \sup \left\{ \frac{|\Delta\lambda|}{\epsilon|\lambda|} : (A + \Delta A)(x + \Delta x) = (\lambda + \Delta\lambda)(B + \Delta B)(x + \Delta x), \right.$$

$$\left. |\Delta A| \le \epsilon E, \quad |\Delta B| \le \epsilon F \right\}. \tag{3.4}$$

From this definition it follows that, for the perturbed system in (3.4),

$$\frac{|\Delta\lambda|}{|\lambda|} \le \operatorname{cond}(\lambda)\epsilon + O(\epsilon^2). \tag{3.5}$$

A special case of the following result with $B = I$, $E = |A|$, and $F = 0$ is given by Geurts [15].

THEOREM 3.2. *The componentwise condition number* $\operatorname{cond}(\lambda)$ *is given by*

$$\operatorname{cond}(\lambda) = \frac{|y^*|E|x| + |\lambda||y^*|F|x|}{|\lambda||y^*Bx|}.$$

Proof. The proof is analogous to the proof of Theorem 2.5. The perturbations that are used to show that the expression for $\operatorname{cond}(\lambda)$ is attained are $\Delta A = \epsilon D_1 E D_2$ and $\Delta B = -\operatorname{sign}(\lambda)\epsilon D_1 F D_2$, where $D_1 = \operatorname{diag}(\operatorname{sign}(y^*))$ and $D_2 = \operatorname{diag}(\operatorname{sign}(x))$. □

We consider briefly the special case where $A \ge 0$ is irreducible and B is diagonal with positive diagonal entries. Here, the generalized eigenvalue problem is equivalent to the standard eigenvalue problem for $B^{-1}A$. Since $B^{-1}A$ is nonnegative and irreducible, the Perron–Frobenius theory can be applied [20, Thm. 8.4.4] to show that the spectral radius $\rho(B^{-1}A)$ is a positive eigenvalue with corresponding positive left and right eigenvectors. The following special result holds for $\lambda = \rho(B^{-1}A)$; it generalizes a result of Elsner et al. [13, Thm. 1] for the standard eigenproblem.

THEOREM 3.3. *Suppose* $A \ge 0$ *is irreducible and* $B = \operatorname{diag}(b_{ii})$, *with* $b_{ii} > 0$ *for all* i. *Let* $\lambda > 0$ *be the Perron root of* $B^{-1}A$, *assumed to be simple, and let* $E = A$ *and* $F = B$. *Then* $\operatorname{cond}(\lambda) = 2$. *Moreover, if* $\lambda + \Delta\lambda$ *is the Perron root of the pair* $(A + \Delta A, B + \Delta B)$ *defined in* (3.4) *then, for* $0 \le \epsilon < 1$,

$$\frac{|\Delta\lambda|}{|\lambda|} \le \frac{2\epsilon}{1 - \epsilon} \tag{3.6}$$

(*which improves on* (3.5) *by quantifying the second order term*).

Proof. That $\operatorname{cond}(\lambda) = 2$ is trivial to verify. For the second part, note that since $|\Delta B| \le \epsilon B$, with B diagonal, and $|\Delta A| \le \epsilon A$,

$$\left(\frac{1 - \epsilon}{1 + \epsilon}\right) B^{-1}A \le (B + \Delta B)^{-1}(A + \Delta A) \le \left(\frac{1 + \epsilon}{1 - \epsilon}\right) B^{-1}A.$$

Since $\rho(\cdot)$ is monotone on the nonnegative matrices [20, Cor. 8.1.19],

$$\left(\frac{1-\epsilon}{1+\epsilon}\right)\rho(B^{-1}A) \le \rho\bigl((B+\Delta B)^{-1}(A+\Delta A)\bigr) \le \left(\frac{1+\epsilon}{1-\epsilon}\right)\rho(B^{-1}A),$$

which rearranges to give (3.6). ∎

As in the normwise case, forcing the backward error perturbations to respect Hermitian structure has no effect on the condition number for a definite pair.

LEMMA 3.4. *Let the Hermitian pair (A, B) be definite. Let* $\operatorname{cond}^H(\lambda)$ *denote the condition number defined as in* (3.4) *but with the additional requirement that ΔA and ΔB are Hermitian, and assume that E and F are Hermitian. Then* $\operatorname{cond}^H(\lambda) = \operatorname{cond}(\lambda)$.

Proof. Since (A, B) is a definite pair, we can take $y = x$, so the perturbations ΔA and ΔB constructed in the proof of Theorem 3.2 are Hermitian. The result is immediate. ∎

We define a componentwise condition number for the eigenvector x corresponding to the simple eigenvalue λ by

$$\operatorname{cond}(x) := \lim_{\epsilon\to 0}\sup\left\{\frac{\|\Delta x\|_\infty}{\epsilon\|x\|_\infty} : (A+\Delta A)(x+\Delta x) = (\lambda+\Delta\lambda)(B+\Delta B)(x+\Delta x),\right.$$

$$\left. g^*Bx = g^*B(x+\Delta x) \equiv 1,\ \ |\Delta A| \le \epsilon E,\ \ |\Delta B| \le \epsilon F \right\}. \tag{3.7}$$

THEOREM 3.5. *The componentwise condition number* $\operatorname{cond}(x)$ *is given by*

$$\operatorname{cond}(x) = \frac{\|\,|V(W^*(A-\lambda B)V)^{-1}W^*|(E+|\lambda|F)|x|\,\|_\infty}{\|x\|_\infty}, \tag{3.8}$$

where the full rank matrices $V, W \in \mathbb{C}^{n\times(n-1)}$ are chosen so that $g^*BV = 0$ and $W^*Bx = 0$.

Proof. The proof is similar to the proof of Theorem 2.7. The perturbations that give equality are

$$\Delta A = \epsilon D_1 E D_2, \qquad \Delta B = -\operatorname{sign}(\lambda)\epsilon D_1 F D_2,$$

where $D_1 = \operatorname{diag}(\xi_j)$, $\xi_j = \operatorname{sign}((V(W^*(A-\lambda B)V)^{-1}W^*)_{kj})$, $D_2 = \operatorname{diag}(\operatorname{sign}(x))$, where the kth component of the vector in the numerator of (3.8) has the largest absolute value. ∎

In the special case $B = I$, $F = 0$, Theorem 3.5 reduces to a result in [6, Section 4.4.2].

4. Structured backward error and condition number. In sections 2 and 3 we considered a normwise backward error and normwise and componentwise condition numbers that respect Hermitian structure in the generalized eigenvalue problem. There are other structures of interest, such as Toeplitz [3], circulant and Hankel structure (the first of which can be general, symmetric, or Hermitian and all of which can be defined in the point or the block sense), augmented system structure [12], Hamiltonian structure [4], and general banded and sparse patterns. We begin with a simple illustration of the effect of taking account of structure. We consider a standard eigenproblem $Ax = \lambda x$ with a circulant A:

$$A = \begin{bmatrix} a & b \\ b & a \end{bmatrix}.$$

We take $a = b = 1$ and

$$\widetilde{x} = \begin{bmatrix} 1+\epsilon \\ -1 \end{bmatrix}, \quad \widetilde{\lambda} = \epsilon \quad (0 \le \epsilon \le 1),$$

which form an exact eigenpair for $\epsilon = 0$. We have

$$r = \widetilde{\lambda}\widetilde{x} - A\widetilde{x} = \begin{bmatrix} \epsilon^2 \\ -2\epsilon \end{bmatrix}.$$

The normwise backward error for the ∞-norm, with $E = A$, $B = I$, $F = 0$, is $\eta(\widetilde{x}, \widetilde{\lambda}) = \epsilon/(1+\epsilon)$. However, since A is a circulant, we may wish to preserve the circulant structure when we perturb A in the definition of the backward error; indeed, only circulant perturbations may be physically meaningful. Then we seek Δa, Δb such that

$$\begin{bmatrix} 1+\Delta a & 1+\Delta b \\ 1+\Delta b & 1+\Delta a \end{bmatrix} \begin{bmatrix} 1+\epsilon \\ -1 \end{bmatrix} = \epsilon \begin{bmatrix} 1+\epsilon \\ -1 \end{bmatrix}.$$

These two equations in two unknowns have a unique solution, for which $\Delta a = -1 + O(\epsilon)$, $\Delta b = -1$. Hence, if ϵ is small, then while $(\widetilde{x}, \widetilde{\lambda})$ is an exact eigenpair for a (noncirculant) matrix close to A, the nearest circulant of which it is an exact eigenpair is relatively far from A. Thus the permitted structure of backward perturbations can greatly affect the backward error.

In this section we assume that $A \in \mathbb{C}^{n\times n}$ and $B \in \mathbb{C}^{n\times n}$ can be parametrized[1]

$$A = \sum_{i=1}^{t} a_i U_i, \qquad B = \sum_{i=1}^{t} a_i V_i, \tag{4.1}$$

where $U_i \in \mathbb{C}^{n\times n}$ and $V_i \in \mathbb{C}^{n\times n}$ are matrices of constants, typically 0s and 1s, and the complex numbers a_i are independent parameters. Thus we are assuming that A and B are linear functions of a vector of parameters $a = (a_i)$. This formulation allows A and B to share parameters, as is necessary for the quadratic eigenvalue problem application (1.2). If B, for example, does not depend on a_j, then we set $V_j = 0$. The case of no structure corresponds to

$$t = 2n^2, \quad \{U_i\}_{i=1}^{n^2} = \{V_i\}_{i=n^2+1}^{2n^2} = \{e_i e_j^T\}_{i,j=1}^{n}, \quad \begin{cases} U_i = 0, & i > n^2, \\ V_i = 0, & i \le n^2, \end{cases} \tag{4.2}$$

where e_i is the ith unit vector.

Nonlinear structure is also of interest, such as Cauchy and Vandermonde structure. Condition numbers can be derived by linearizing and applying the techniques given here, but evaluating backward errors is a nonlinear optimization problem in general. See Bartels and Higham [2] for results for Vandermonde-like linear systems.

We consider perturbations $\Delta A = \sum_{i=1}^{t} \Delta a_i U_i$ and $\Delta B = \sum_{i=1}^{t} \Delta a_i V_i$ and measure them by

$$\psi_p(\Delta a) = \|D^{-1}\Delta a\|_p, \quad D = \mathrm{diag}(g),$$

[1] In fact, it is only the perturbations ΔA and ΔB that need to have such structure in our development, but it is natural to assume that A and B are structured too.

where g is a vector of nonnegative tolerances and $\|\cdot\|_p$ is the Hölder p-norm on $\mathbb{C}^n$:

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}, \qquad p \geq 1.$$

Note that $p = \infty$ corresponds to the componentwise measure used in (3.1), albeit now applied to the vector of parameters rather than the matrices as a whole.

We define the structured componentwise backward error of an approximate eigenpair $(\widetilde{x}, \widetilde{\lambda})$ by

$$\overline{\omega}_p(\widetilde{x}, \widetilde{\lambda}) := \min \left\{ \psi_p(\Delta a) : (A + \Delta A)\widetilde{x} = \widetilde{\lambda}(B + \Delta B)\widetilde{x}, \right.$$
$$\left. \Delta A = \sum_{i=1}^{t} \Delta a_i U_i, \quad \Delta B = \sum_{i=1}^{t} \Delta a_i V_i \right\}. \tag{4.3}$$

Using the residual $r = \widetilde{\lambda}B\widetilde{x} - A\widetilde{x}$, the constraint in (4.3) can be rewritten

$$\begin{aligned} r &= \Delta A\widetilde{x} - \widetilde{\lambda}\Delta B\widetilde{x} \\ &= \left(\sum_{i=1}^{t} \Delta a_i U_i\right)\widetilde{x} - \widetilde{\lambda}\left(\sum_{i=1}^{t} \Delta a_i V_i\right)\widetilde{x} \\ &= \sum_{i=1}^{t} \Delta a_i (U_i - \widetilde{\lambda}V_i)\widetilde{x} \\ &= \widetilde{C}\Delta a, \end{aligned}$$

where

$$\widetilde{C} = \left[(U_1 - \widetilde{\lambda}V_1)\widetilde{x}, \ldots, (U_t - \widetilde{\lambda}V_t)\widetilde{x}\right] \in \mathbb{C}^{n \times t}.$$

Hence, defining

$$\widetilde{\Delta a} = D^{-1}\Delta a,$$

we have

$$r = \widetilde{C}\Delta a = \widetilde{C}D\widetilde{\Delta a} =: M\widetilde{\Delta a}, \qquad M \in \mathbb{C}^{n \times t}.$$

The linear system

$$M\widetilde{\Delta a} = r \tag{4.4}$$

can be under- or overdetermined, depending on the value of t. The backward error $\overline{\omega}_p(\widetilde{x}, \widetilde{\lambda})$ is the norm of a solution of minimal p-norm to (4.4). If $n > t$ or M is rank-deficient there may be no solution to (4.4), in which case we regard the structured componentwise backward error $\overline{\omega}_p(\widetilde{x}, \widetilde{\lambda})$ as infinite. Assuming that the system is consistent, for the 2-norm we have $\overline{\omega}_2(\widetilde{x}, \widetilde{\lambda}) = \|M^{+}r\|_2$, where M^{+} is the pseudo-inverse of M.

The structure of the matrix M depends on that of A and B in (4.1), and it may be possible to exploit this structure when solving (4.4). For the unstructured case with

$p = \infty$, we recover the expression (3.2) on using (4.2) and identifying D with E and F in (3.1), because (4.4) then reduces to n independent minimal ∞-norm problems that can be solved explicitly.

A structured componentwise condition number for the simple eigenvalue λ can be defined by

$$\overline{\mathrm{cond}}_p(\lambda) := \lim_{\epsilon\to 0} \sup\left\{ \frac{|\Delta\lambda|}{\epsilon|\lambda|} : (A+\Delta A)(x+\Delta x) = (\lambda+\Delta\lambda)(B+\Delta B)(x+\Delta x),\right.$$

$$\left. \Delta A = \sum_{i=1}^{t} \Delta a_i U_i,\ \ \Delta B = \sum_{i=1}^{t} \Delta a_i V_i,\ \ \psi_p(\Delta a) \le \epsilon \right\}. \tag{4.5}$$

THEOREM 4.1. *The structured componentwise condition number of λ is given by*

$$\overline{\mathrm{cond}}_p(\lambda) = \frac{\|y^*CD\|_q}{|\lambda||y^*Bx|},$$

where $p^{-1}+q^{-1}=1$ and

$$C = [(U_1 - \lambda V_1)x, \ldots, (U_t - \lambda V_t)x] \in \mathbb{C}^{n\times t}. \tag{4.6}$$

Proof. The expansion (2.11) shows that we have to find a sharp bound for $y^*\Delta Ax - \lambda y^*\Delta Bx$. For the perturbations ΔA and ΔB in (4.5) we have

$$\begin{aligned} |y^*\Delta Ax - \lambda y^*\Delta Bx| &= \left| y^*\left(\sum_{i=1}^{t} \Delta a_i U_i\right)x - \lambda y^*\left(\sum_{i=1}^{t} \Delta a_i V_i\right)x \right| \\ &= |y^*C\Delta a| \\ &= |y^*CD\cdot D^{-1}\Delta a| \\ &\le \|y^*CD\|_q \psi_p(\Delta a). \end{aligned}$$

Equality is obtained for suitable Δa because equality is always possible in the Hölder inequality. □

It is not hard to see that for $p=\infty$ we recover the formula for the unstructured condition number $\mathrm{cond}(\lambda)$ in Theorem 3.2 when we use (4.2) and identify D with E and F.

A structured componentwise condition number for the eigenvector x corresponding to the simple eigenvalue λ can be defined by

$$\overline{\mathrm{cond}}_p(x) := \lim_{\epsilon\to 0} \sup\left\{ \frac{\|\Sigma^{-1}\Delta x\|_p}{\epsilon} : (A+\Delta A)(x+\Delta x) = (\lambda+\Delta\lambda)(B+\Delta B)(x+\Delta x),\right.$$

$$g^*Bx = g^*B(x+\Delta x) \equiv 1,$$

$$\left. \Delta A = \sum_{i=1}^{t} \Delta a_i U_i,\ \ \Delta B = \sum_{i=1}^{t} \Delta a_i V_i,\ \ \psi_p(\Delta a) \le \epsilon \right\}, \tag{4.7}$$

where $\Sigma = \mathrm{diag}(\sigma_i)$, with the σ_i positive tolerances. In the particular case $\sigma_i \equiv \|x\|_p$, we are measuring Δx in the usual normwise relative fashion.

THEOREM 4.2. *Assume that B is nonsingular. The structured componentwise condition number of x is given by*

$$\overline{\mathrm{cond}}_p(x) = \left\|\Sigma^{-1}V(W^*(A-\lambda B)V)^{-1}W^*CD\right\|_p,$$

where C is defined in (4.6) *and the full rank matrices $V, W \in \mathbb{C}^{n\times(n-1)}$ are chosen so that $g^*V = 0$ and $W^*x = 0$.*

Proof. From Theorem A.1 we see that we must find a sharp bound for $\|Z(\mathit{\Delta} A - \lambda\mathit{\Delta} B)x\|_p/\epsilon$, where $Z = \mathit{\Sigma}^{-1}V(W^*(A-\lambda B)V)^{-1}W^*$. As in the proof of Theorem 4.1, we have

$$\|Z(\mathit{\Delta} A - \lambda\mathit{\Delta} B)x\|_p = \|ZCD\cdot D^{-1}\mathit{\Delta} a\|_p \le \|ZCD\|_p\psi_p\mathit{\Delta} a,$$

and the latter inequality is attained for suitable $\mathit{\Delta} a$. □

In the above analysis, the parameter vector $\mathit{\Delta} a$ is complex in general, although it can be taken to be real when all the data are real. In certain circumstances it is appropriate to restrict $\mathit{\Delta} a$ to be real, even though the data are complex. For example, a 2×2 Hermitian perturbation $\mathit{\Delta} A$ can be parametrized

$$\mathit{\Delta} A = \begin{bmatrix} \mathit{\Delta} a_1 & \mathit{\Delta} a_2 - \mathit{\Delta} a_3 i \\ \mathit{\Delta} a_2 + \mathit{\Delta} a_3 i & \mathit{\Delta} a_4 \end{bmatrix}, \qquad \mathit{\Delta} a_k \in \mathbb{R}, \qquad k = 1\colon 4.$$

The backward error derivation must then be modified by taking real and imaginary parts in (4.4) to obtain a real system, of which a minimal norm solution is required. The quantities derived in Theorems 4.1 and 4.2 are now upper bounds for the condition numbers; the bounds are attained for $p = 1$ and therefore are within a factor t of being attained for other values of p.

5. Numerical experiments. To illustrate our results we present some numerical examples. All computations were carried out in MATLAB, which has unit roundoff $u = 2^{-53} \approx 1.1\times 10^{-16}$. In each example we used the QZ algorithm [23] to compute the eigensystem. Since MATLAB's implementation of the QZ algorithm does not provide left eigenvectors, we computed left eigenvectors using inverse iteration. Condition numbers were evaluated using the computed eigenvalues and eigenvectors in place of the exact ones.

The first example is the problem with

$$A = \begin{bmatrix} 0.1 & 0.2 \\ 0.3 & 0.4 \end{bmatrix}, \qquad B = \begin{bmatrix} 0.1 & 0.1 \\ 0 & \sqrt{u} \end{bmatrix},$$

which is attributed to Wilkinson by Moler and Stewart [23]. One eigenvalue is close to -2 and the other is of order 10^7. The eigenpair corresponding to the small eigenvalue is found to be well conditioned. For the large eigenvalue and its eigenvector we find that, with $E = A$ and $F = B$ and using the 2-norm,

$$\begin{aligned} & \kappa(\lambda) = 1.9\times 10^7, \quad \mathrm{cond}(\lambda) = 1.0\times 10^1, \\ \text{for } g = x\text{:} \quad & \kappa(x) = 8.5\times 10^{13}, \quad \mathrm{cond}(x) = 6.0\times 10^{13}, \\ \text{for } g = y\text{:} \quad & \kappa(x) = 1.0\times 10^1, \quad \mathrm{cond}(x) = 8.0\times 10^0. \end{aligned}$$

This example shows that an eigenvalue may have different sensitivity with respect to normwise and componentwise perturbations and that the sensitivity of an eigenvector can depend strongly on how it is normalized.

The second example concerns symmetric structure in A and B. The matrix A is the `ipjfact` matrix ($a_{ij} = (i+j)!$) from The Test Matrix Toolbox [19] but with its rows and columns in reverse order, and B is the Pascal matrix from the same source; both matrices are positive definite.

Table 5.1
Results for symmetric 8×8 *A and B.*

	Backward	Condition	
	error	λ	x
Unstructured normwise (η, κ)	2.1e-17	4.2e15	3.5e15
Structured normwise ($\overline{\omega}, \overline{\mathrm{cond}}$)	2.1e-17	5.2e15	4.2e15
Unstructured componentwise (ω, cond)	1.5e-8	1.6e6	2.6e6
Structured componentwise ($\overline{\omega}, \overline{\mathrm{cond}}$)	3.8e-8	4.4e5	7.7e5

Table 5.2
Results for Toeplitz 16×16 *A and B.*

	Backward	Condition	
	error	λ	x
Unstructured normwise (η, κ)	2.5e-17	5.2e5	4.6e6
Structured normwise ($\overline{\omega}, \overline{\mathrm{cond}}$)	8.3e-17	3.7e5	5.7e6
Unstructured componentwise (ω, cond)	5.4e-14	6.6e1	4.3e2
Structured componentwise ($\overline{\omega}, \overline{\mathrm{cond}}$)	4.5e-12	5.6e0	7.0e1

In the third example we have Toeplitz A and B. A Toeplitz matrix $T \in \mathbb{C}^{n\times n}$ can be defined by the sequence $t_{n1}, t_{n-1,1}, \dots, t_{11}, t_{12}, \dots, t_{1n}$. With this notation, A is defined by $1, 2^2, \dots, (2n-1)^2$, and B is defined by $10^{-8}(1, \alpha, \alpha^2, \dots, \alpha^{2n-1})$, where $\alpha = 10^{8/(2n-1)}$, except that the diagonal of B is multiplied by -1.

In both examples we computed the eigenvalue with smallest real part (which happens to be real in both cases) and the corresponding left and right eigenvectors. The choice of the various parameters is summarized as follows:

- Normalization for eigenvector condition numbers: $g = y$ (left eigenvector).
- Normwise backward error η and condition numbers κ: 2-norm with $E = A$ and $F = B$.
- Componentwise backward error ω and condition numbers cond: $E = |A|$ and $F = |B|$.
- Structured backward error $\overline{\omega}$ and condition number $\overline{\mathrm{cond}}$: $g = \|a\|_2$, giving "normwise" measure, and $g = |a|$, giving "componentwise" measure, using $\Sigma = \mathrm{diag}(\|x\|_p)$ and $p = 2$.

The results are shown in Tables 5.1 and 5.2. Several features are worth noting. For the symmetric problem we see that the componentwise backward error is much larger than the unit roundoff and that requiring symmetry of the backward perturbations has little effect on this backward error (we know the same is true of the normwise backward error by Theorem 2.3). On the other hand, the eigenvalue and eigenvector are much less sensitive to componentwise perturbations than to normwise ones, and again a restriction to symmetric perturbations makes little difference. For the Toeplitz problem, requiring Toeplitz perturbations increases the componentwise backward error by two orders of magnitude and reduces the componentwise condition numbers by an order of magnitude.

In the examples above, the matrix M in (4.4) was always of full rank. In similar experiments with A and B both having symmetric Toeplitz structure we found that M was usually numerically rank-deficient and that the system (4.4) did not always have a solution (more precisely, the minimal-norm least-squares solution did not always have a small residual, so that the structured backward error was sometimes infinite). This behavior appears to be due to symmetries in the eigenvectors. It is known that any eigenvector x of a symmetric Toeplitz matrix with distinct eigenvalues satisfies

Table 5.3
Condition numbers for quadratic eigenvalue problem with data (5.1).

	Unstructured	Structured
Normwise	2.0e12	1.0e8
Componentwise	4.0e0	1.0e0

either $Jx = x$ or $Jx = -x$, where J is the identity matrix with its columns in reverse order [9]. It is easy to verify that, for such an x, the matrix $[U_1x, \dots, U_nx]$ has some repeated rows (up to sign) and so is rank-deficient, where $U_1, \dots, U_n$ are the pattern matrices corresponding to symmetric Toeplitz structure in an $n \times n$ real matrix. Similar symmetries appear to hold for the generalized eigenvalue problem, causing numerical rank-deficiency of M for "good" approximate eigenvectors $\widetilde{x}$.

Finally, we consider the generalized eigenvalue problem (1.2) corresponding to the quadratic eigenvalue problem (1.1), with

$$C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad D = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \quad E = \begin{bmatrix} 10^{-4} & 1 \\ 0 & 10^{-8} \end{bmatrix}. \tag{5.1}$$

We report in Table 5.3 condition numbers for the eigenvalue $\lambda = -1.0001 \times 10^{-4}$ (to five significant figures). The parameter vector is $a = [\mathrm{vec}(C)^T \, \mathrm{vec}(D)^T \, \mathrm{vec}(E)^T]^T$, where vec stacks the columns of a matrix into one long vector. For the normwise condition number we take $g = [\|C\|_2 e^T, \|D\|_2 e^T, \|E\|_2 e^T]^T$, where $e \in \mathbb{R}^2$ is the vector of 1s, and, for the componentwise condition number, $g = |a|$ and $p = 1$. We see that the normwise condition number for the quadratic eigenvalue problem is four orders of magnitude smaller than the condition number we obtain by ignoring the structure in the corresponding generalized eigenvalue problem. The componentwise condition numbers are much smaller than the normwise ones, and the structured condition number reveals perfect conditioning.

There are many interesting open problems in the areas of establishing the existence of finite structured backward errors and bounding the difference between structured and unstructured backward errors and condition numbers.

Appendix. The following result generalizes analysis of Geurts [15], Chaitin-Chatelin and Frayssé [6, Section 4.4.2], and Chatelin [7, Section 4.2.1] for the standard eigenproblem.

Theorem A.1. *Let $A, B \in \mathbb{C}^{n\times n}$. Let λ be a simple eigenvalue and x the corresponding eigenvector of the pair (A, B) and let*

$$(A + \Delta A)(x + \Delta x) = (\lambda + \Delta\lambda)(B + \Delta B)(x + \Delta x).$$

Normalize x and $x + \Delta x$ by

$$g^*Bx = g^*B(x + \Delta x) = 1, \tag{A.1}$$

where g is a given vector (it is assumed that $g^*Bx \neq 0$). For sufficiently small ΔA and ΔB there is a unique Δx which can be expressed, to first order, as

$$\Delta x = -V(W^*(A - \lambda B)V)^{-1}W^*(\Delta A - \lambda\Delta B)x, \tag{A.2}$$

where the full rank matrices $V, W \in \mathbb{C}^{n\times(n-1)}$ are chosen so that $g^*BV = 0$ and $W^*Bx = 0$.

Proof. For sufficiently small ΔA and ΔB, $\lambda + \Delta\lambda$ is a simple eigenvalue of the pair $(A + \Delta A, B + \Delta B)$ [30, Thm. VI.2.1], making Δx unique when normalized as in (A.1).

We have

$$\Delta A x + A\Delta x = \lambda\Delta B x + \lambda B\Delta x + \Delta\lambda B x, \tag{A.3}$$

where this and all subsequent equations include first order terms only. Premultiplying by W^* gives

$$W^*(A - \lambda B)\Delta x = -W^*(\Delta A - \lambda\Delta B)x. \tag{A.4}$$

Define the matrices $M = [\,x \quad V\,]$, $N = [\,g \quad W\,]$. We show that M is nonsingular. Note first that

$$g^*BM = [\,g^*Bx \quad g^*BV\,] = [\,1 \quad 0\,] = e_1^T. \tag{A.5}$$

Hence if $Mt = 0$ then $0 = g^*BMt = t_1$, and then $0 = Mt = Vt(2\colon n)$, which implies $t(2\colon n) = 0$. A similar proof shows that N is nonsingular.

Write $\Delta x = Mz$. Then

$$\begin{aligned} W^*(A - \lambda B)\Delta x &= W^*(A - \lambda B)\,[\,x \quad V\,]\,z \\ &= W^*\,[\,0 \quad (A - \lambda B)V\,]\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \\ &= W^*(A - \lambda B)Vz_2. \qquad \text{(A.6)} \end{aligned}$$

Now we show that the matrix $W^*(A - \lambda B)V \in \mathbb{C}^{(n-1)\times(n-1)}$ is nonsingular.

We have

$$\begin{aligned} N^*(A - \mu B)M &= \begin{bmatrix} g^* \\ W^* \end{bmatrix}[\,(\lambda - \mu)Bx \quad (A - \mu B)V\,] \\ &= \begin{bmatrix} \lambda - \mu & g^*(A - \mu B)V \\ 0 & W^*(A - \mu B)V \end{bmatrix}. \end{aligned}$$

Hence $\det(N^*)\det(A-\mu B)\det(M) = (\lambda-\mu)\det(W^*(A-\mu B)V)$. Since N and M are nonsingular and λ is a simple eigenvalue of (A, B) it follows that $\det(W^*(A - \mu B)V)$ is nonzero for $\mu = \lambda$, as required.

From (A.6) and (A.4) we obtain $z_2 = -(W^*(A - \lambda B)V)^{-1}W^*(\Delta A - \lambda\Delta B)x$. To determine the scalar z_1 we note that, from the normalization condition (A.1) and (A.5),

$$0 = g^*B\Delta x = g^*BMz = e_1^T z = z_1.$$

Hence $\Delta x = Vz_2$, as required. □

The matrices V and W appearing in (A.2) can be explicitly constructed as follows. Let $Q_v^T B^*g = R_v = \pm\|B^*g\|_2 e_1$ and $Q_w^T Bx = R_w = \pm\|Bx\|_2 e_1$ be QR factorizations. Then we can take $V = Q_v(:, 2\colon n)$ and $W = Q_w(:, 2\colon n)$.

Note that the matrix $V(W^*(A - \lambda B)V)^{-1}W^*$ is independent of the particular choice of V and W, since Δx in (A.2) is unique for sufficiently small ΔA and ΔB.

REFERENCES

[1] M. F. Anjos, S. J. Hammarling, and C. C. Paige, *Solving the Generalized Symmetric Eigenvalue Problem*, manuscript, 1992.

[2] S. G. Bartels and D. J. Higham, *The structured sensitivity of Vandermonde-like systems*, Numer. Math., 62 (1992), pp. 17–33.

[3] R. M. Beam and R. F. Warming, *The asymptotic spectra of banded Toeplitz and quasi-Toeplitz matrices*, SIAM J. Sci. Comput., 14 (1993), pp. 971–1006.

[4] P. Benner, V. Mehrmann, and H. Xu, *A New Method for Computing the Stable Invariant Subspace of a Real Hamiltonian Matrix*, Preprint SFB393/97-01, Fak. f. Mathematik, Technische Universität Chemnitz-Zwickau, Chemnitz, FRG, 1997; J. Comp. Math. Appl., to appear.

[5] J. R. Bunch, J. W. Demmel, and C. F. Van Loan, *The strong stability of algorithms for solving symmetric linear systems*, SIAM J. Matrix Anal. Appl., 10 (1989), pp. 494–499.

[6] F. Chaitin-Chatelin and V. Frayssé, *Lectures on Finite Precision Computations*, SIAM, Philadelphia, PA, 1996.

[7] F. Chatelin, *Eigenvalues of Matrices*, Wiley, Chichester, UK, 1993.

[8] J. K. Cullum and R. A. Willoughby, *A QL procedure for computing the eigenvalues of complex symmetric tridiagonal matrices*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 83–109.

[9] G. Cybenko, *On the eigenstructure of Toeplitz matrices*, IEEE Trans. Acoust., Speech, Signal Processing, ASSP-32 (1984), pp. 918–921.

[10] A. S. Deif, *Realistic a priori and a posteriori error bounds for computed eigenvalues*, IMA J. Numer. Anal., 9 (1990), pp. 323–329.

[11] Z. Drmač, *A tangent algorithm for computing the generalized singular value decomposition*, SIAM J. Numer. Anal., 35 (1998), pp. 1804–1832.

[12] H. Elman and D. Silvester, *Fast nonsymmetric iterations and preconditioning for Navier-Stokes equations*, SIAM J. Sci. Comput., 17 (1996), pp. 33–46.

[13] L. Elsner, I. Koltracht, M. Neumann, and D. Xiao, *On accurate computations of the Perron root*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 456–467.

[14] V. Frayssé and V. Toumazou, *A note on the normwise perturbation theory for the regular generalized eigenproblem $Ax = \lambda Bx$*, Numer. Linear Algebra Appl., 5 (1998), pp. 1–10.

[15] A. J. Geurts, *A contribution to the theory of condition*, Numer. Math., 39 (1982), pp. 85–96.

[16] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 2nd ed., Johns Hopkins University Press, Baltimore, MD, 1989.

[17] D. J. Higham and N. J. Higham, *Backward error and condition of structured linear systems*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 162–175.

[18] N. J. Higham, *A survey of componentwise perturbation theory in numerical linear algebra*, in Mathematics of Computation 1943–1993: A Half Century of Computational Mathematics, W. Gautschi, ed., Proc. Sympos. Appl. Math. 48, AMS, Providence, RI, 1994, pp. 49–77.

[19] N. J. Higham, *The Test Matrix Toolbox for* Matlab *(version* 3.0*)*, Numerical Analysis Report No. 276, Manchester Centre for Computational Mathematics, Manchester, England, 1995.

[20] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, London, 1985.

[21] B. Kågström and P. Poromaa, *Computing eigenspaces with specified eigenvalues of a regular matrix pair (A, B) and condition estimation: Theory, algorithms and software*, Numer. Algorithms, 12 (1996), pp. 369–407.

[22] W. Kahan, B. N. Parlett, and E. Jiang, *Residual bounds on approximate eigensystems of nonnormal matrices*, SIAM J. Numer. Anal., 19 (1982), pp. 470–484.

[23] C. B. Moler and G. W. Stewart, *An algorithm for generalized matrix eigenvalue problems*, SIAM J. Numer. Anal., 10 (1973), pp. 241–256.

[24] W. Oettli and W. Prager, *Compatibility of approximate solution of linear equations with given error bounds for coefficients and right-hand sides*, Numer. Math., 6 (1964), pp. 405–409.

[25] B. N. Parlett, *The Symmetric Eigenvalue Problem*, Prentice–Hall, Englewood Cliffs, NJ, 1980. Reprinted by SIAM, Philadelphia, PA, 1998.

[26] J. L. Rigal and J. Gaches, *On the compatibility of a given solution with the data of a linear system*, J. Assoc. Comput. Mach., 14 (1967), pp. 543–548.

[27] Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, Manchester University Press, Manchester, and Halsted Press, New York, 1992.

[28] A. Smoktunowicz, *The Strong Stability of Algorithms for Solving the Symmetric Eigenproblem*, manuscript, 1995.

[29] G. W. Stewart, *Perturbation theory for the generalized eigenvalue problem*, in Recent Advances in Numerical Analysis, C. de Boor and G. H. Golub, eds., Academic Press, New York, 1978, pp. 193–206.

[30] G. W. Stewart and Ji-guang Sun, *Matrix Perturbation Theory*, Academic Press, London, 1990.

[31] P. M. Van Dooren, *Structured linear algebra problems in digital signal processing*, in Numerical Linear Algebra, Digital Signal Processing and Parallel Algorithms, G. H. Golub and P. M. Van Dooren, eds., NATO Adv. Sci. Inst. Ser. F Comput. Systems Sci. 70, Springer-Verlag, Berlin, 1991, pp. 361–384.

[32] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford University Press, London, 1965.

SIAM J. Matrix Anal. Appl.
Vol. 20, No. 2, pp. 513–561

ACCURATE SYMMETRIC INDEFINITE LINEAR EQUATION SOLVERS[*]

CLEVE ASHCRAFT[†], ROGER G. GRIMES[†], AND JOHN G. LEWIS[†]

Abstract. The Bunch–Kaufman factorization is widely accepted as the algorithm of choice for the direct solution of symmetric indefinite linear equations; it is the algorithm employed in both LINPACK and LAPACK. It has also been adapted to sparse symmetric indefinite linear systems.

While the Bunch–Kaufman factorization is normwise backward stable, its factors can have unusual scaling, with entries bounded by terms depending both on $\|A\|$ and on $\kappa(A)$. This scaling, combined with the block nature of the algorithm, may degrade the accuracy of computed solutions unnecessarily. Overlooking the lack of a triangular factor bound leads to a further complication in LAPACK such that the LAPACK Bunch–Kaufman factorization can be unstable.

We present two alternative algorithms, close cousins of the Bunch–Kaufman factorization, for solving dense symmetric indefinite systems. Both share the positive attributes of the Bunch–Kaufman algorithm but provide better accuracy by bounding the triangular factors. The price of higher accuracy can be kept low by choosing between our two algorithms. One is appropriate as the replacement for the blocked LAPACK Bunch–Kaufman factorization; the other would replace the LINPACK-like unblocked factorization in LAPACK.

Solving sparse symmetric indefinite systems is more problematic. We conclude that the Bunch–Kaufman algorithm cannot be rescued effectively in the sparse case. Imposing the constraint of bounding the triangular factors leads naturally to one particular version of the Duff–Reid algorithm, which we show gives better accuracy than Liu's sparse variant of the Bunch–Kaufman algorithm. We extend the work of Duff and Reid in two respects that often provide higher efficiency: a more effective procedure for finding pivot blocks and a stable extension to pivot blocks of size larger than two.

Key words. dense and sparse matrices, symmetric indefinite matrices, diagonal pivoting, block pivoting, matrix inertia

AMS subject classifications. 65F05, 65F50, 15A06, 15A12

PII. S0895479896296921

1. Introduction.

Our goal is to solve linear systems $Ax = b$ efficiently and accurately when A is a symmetric indefinite matrix. There are three well-known algorithms for solving this problem when A is a dense matrix: the Bunch–Kaufman algorithm [8], the Bunch–Parlett algorithm [9], and Aasen's algorithm [1]. The most recent of these, the Bunch–Kaufman algorithm, is the most commonly used; it has been chosen for both LINPACK and LAPACK. For this reason we primarily address the Bunch–Kaufman algorithm, although we will have observations on the other algorithms.

The Bunch–Kaufman algorithm factors A as $PAP^T = LDL^T$, where L is a unit lower triangular matrix[1] and D is block diagonal, composed solely of 1×1 and 2×2 blocks. The work required is only slightly more than for the Cholesky decomposition of a positive definite symmetric matrix.

[*] Received by the editors January 3, 1996; accepted for publication (in revised form) by J. W. H. Liu May 17, 1997; published electronically December 11, 1998.
http://www.siam.org/journals/simax/20-2/29692.html

[†] Applied Research and Technology, The Boeing Company, Seattle, WA 98124-2207 (cleve@redwood.rt.cs.boeing.com, roger.g.grimes@boeing.com, john.g.lewis@boeing.com).

[1] Bunch and Kaufman describe an MDM^T factorization where M can be chosen to be either lower or upper triangular. For convenience we use the lower triangular variant exclusively in this paper.

Accuracy of triangular factorization solvers is commonly addressed by Wilkinson's backward error analysis [32]. First, one shows that the computed LDL^T factorization is the exact factorization of a perturbed matrix; that is,

$$LDL^T = A + \Delta A,$$

where the matrix ΔA accounts for the rounding errors that occurred during the factorization. A second step is needed to show that the computed solution x exactly solves

$$b = (L + \Delta L)(D + \Delta D)(L^T + \Delta L^T)x.$$

The matrices $\Delta L, \Delta D$, and ΔL^T reflect the usually smaller accumulation of errors that occur during each of the three solution steps. Normally we expect each to be a small relative perturbation. For example, Wilkinson's analysis of triangular systems shows that $|\Delta L| \leq n\epsilon|L|$. Assuming that $\Delta L, \Delta D$, and ΔL^T are $\mathcal{O}(\epsilon)$ terms and that $\mathcal{O}(\epsilon^2)$ terms can be ignored, the separate analyses can be combined into a single backward error equation

$$b = \left(A + \Delta A + \Delta L \cdot D \cdot L^T + L \cdot \Delta D \cdot L^T + L \cdot D \cdot \Delta L^T\right) x.$$

Usually the most difficult aspect of analyzing factorization algorithms is bounding ΔA. Bunch and Kaufman [8] followed Wilkinson's elegant approach. The kth step of the Bunch–Kaufman factorization, applied to the reduced matrix $A^{(k-1)}$, results in a new reduced matrix of lower order. The selection of pivots in the Bunch–Kaufman factorization was exquisitely crafted to bound the entries in the new reduced matrix beneath a quantity that depends primarily on the magnitude of the largest entries in A itself. The care with which the pivot selection algorithm was developed is evident from the details needed for the proof.

Bunch and Kaufman, like all of their contemporaries, addressed only the reduced matrix. Unfortunately a bound on the reduced matrix is not sufficient to show that either the backward perturbation matrix ΔA is small or the solution perturbation terms are small. The issue is that the Bunch–Kaufman algorithm is a block algorithm. That bounding the reduced matrix of a blocked algorithm does not bound ΔA is a recent observation by Demmel, Higham, and Schreiber [10, 11, 24]. The proof of stability in [8] is therefore incomplete.

The difficulty caused by blocking can be illustrated by considering a sequence of operations that arise in proving that ΔA and solution error terms are small. Analysis of a 1×1 elimination step uses the fact that $w = a_{ij}/d_k, \ldots, z = w \times d_k$ results in z being a small relative perturbation of a_{ij}. In contrast an elimination step with a 2×2 block $D_{k:k+1}$ results in a corresponding sequence of operations $W = \left[\begin{array}{cc} a_{i1} & a_{i2} \end{array}\right] D_{k:k+1}^{-1}, \ldots, Z = WD_{k:k+1}$. We cannot assert without further analysis that we have better than $\kappa(D_{k:k+1})\epsilon$ relative error in Z.

There is a simple way to prove stability for most symmetric indefinite factorization algorithms, in particular for the Bunch–Parlett algorithm. An extension of the arguments in [11] requires only that $\|L\|\|D\|\|L^T\| \leq c_1\|A\|$ for some constant c_1 and that 2×2 systems of equations be solved in a normwise backward stable manner. It follows from these two conditions, for example, that $\|Z - \left[\begin{array}{cc} a_{i1} & a_{i2} \end{array}\right]\| \leq c_1 c_2 \epsilon \; \|\left[\begin{array}{cc} a_{i1} & a_{i2} \end{array}\right]\|$, where $c_2\epsilon$ is the small constant from the stability analysis of 2×2 linear equations. An alternative norm condition is $\|L\| \leq c_3$; a proof of backward stability with this condition is given in Appendix B. The Bunch–Parlett algorithm satisfies this norm condition.

Solving 2×2 systems of equations in a normwise backward stable manner can be addressed simply by using Gaussian elimination with partial or complete pivoting. More traditionally, explicit inversion of $D_{k:k+1}$ or Cramer's rule has been used, which satisfies the backward stability property only if additional conditions, such as constant bounded condition number, are available. The more important issue is obtaining a bound $\|L\|\|D\|\|L^T\| \leq c_1\|A\|$; finding algorithms that produce such bounds is the major theme of this paper.

When the product of norms condition holds, as in LU, QR, or Cholesky factorizations, proofs that the perturbation terms in the backward error analysis are $\mathcal{O}(\|A\|)\epsilon$ are simple. This condition does not hold for the Bunch–Kaufman factorization. The entries in the diagonal blocks of the Bunch–Kaufman factorization are entries in reduced matrices, which by the bound on reduced matrix growth are less than a small multiple of $\|A\|$. An $\mathcal{O}(\|A\|)$ bound for the product of norms of the factors requires that the Bunch–Kaufman algorithm impose a bound near unity on $\|L\|$; unfortunately it does not. We will demonstrate in section 2.2 that at best we get a weak bound of $\kappa(A)$ on $\|L\|$.

Bounding the factors is not essential, but stability without a bound near unity on $\|L\|$ requires that large entries in L always be scaled by small entries in D. The unusual aspects of the Bunch–Kaufman algorithm ensure precisely this. However, the proof in [8] is incomplete. Higham [24] has recently given a complete analysis of the Bunch–Kaufman algorithm that shows that all of the perturbation terms $\Delta A, \Delta L \cdot D \cdot L^T, L \cdot \Delta D \cdot L^T$, and $L \cdot D \cdot \Delta L^T$ are appropriately small, even in floating point arithmetic. A crucial part of his analysis requires that the solution of 2×2 systems using 2×2 blocks of D be *componentwise* backward stable. This is true for the 2×2 blocks from the Bunch–Kaufman factorization when the solution algorithm is the scaled Cramer's rule scheme used in LINPACK and LAPACK or Gaussian elimination with partial pivoting. The Bunch–Kaufman algorithm is backward stable when 2×2 systems are solved with one of these algorithms.

Nonetheless, the solutions obtained without a bound on $\|L\|$ can be less accurate than they should be. This has not been observed in the past, perhaps because of two historical biases. We may neglect error in triangular solves because Wilkinson [32] observed that triangular solves often behave much better than the error bounds. We may also neglect the triangular solves because the backward error equation attributes an $n\|A\|\epsilon$ relative backward error to the factorization error term ΔA. In most cases the factorization error is the dominant term.

But more is known about Wilkinson's observation. A special case of an analysis by Higham [22, 25] is that there is a satisfactory forward error analysis, independent of the condition number of L, for unit triangular matrices where all off-diagonal entries are bounded by one. Solves with such triangular matrices should behave better than the standard error bound. Similar conditions apply to triangular matrices produced by several standard decompositions. The triangular matrices produced by the Bunch–Kaufman algorithm can be arbitrarily far from satisfying Higham's condition. They do not have to behave well.

The problem with the Bunch–Kaufman algorithm is that large entries in L can create situations in which the desired results are computed as the difference of much larger quantities. Consider the 5×5 matrix $A = LDL^T$, where

$$L = \begin{bmatrix} 1 & & & & \\ 1 & 1 & & & \\ 1 & 0 & 1 & & \\ 1 & -\frac{20}{13} & -\frac{8}{17} & 1 & \\ 1 & \frac{6}{13} \times 10^{6} & -\frac{1}{17} & 0 & 1 \end{bmatrix},$$

$$D = \begin{bmatrix} 1 & & & & \\ & \frac{1}{3} \times 10^{-19} & \frac{6}{7} \times 10^{-7} & & \\ & \frac{6}{7} \times 10^{-7} & -\frac{3}{13} \times 10^{-6} & & \\ & & & -\frac{4}{17} \times 10^{-5} & \frac{2}{7} \\ & & & \frac{2}{7} & \frac{1}{300} \end{bmatrix}.$$

All entries of the product $LDL^T = A$ satisfy $.998 < a_{ij} < 1.27$, but by construction L has large entries. We take b as $A\hat{x}$, where $\hat{x} = \begin{bmatrix} 1 & 1/3 & 1/7 & 2/3 & 4/13 \end{bmatrix}$. The second entry of the exact solution is $1/3$, but part of the computation of this entry of the floating point solution is

$$\left(1.42011\ldots \times 10^{5}\right) - \left(\frac{6}{13} \times 10^{6}\right) \times 0.307692\ldots.$$

Cancelation yields only eight correct digits in IEEE double precision.

Is this important? This example is clearly contrived. The computed result is within the uncertainty suggested by $\kappa(A) \approx 4.5 \times 10^{8}$. Would the large entries in L make a difference in real applications, and why should we expect to be able to do better?

We first encountered difficulties in solving symmetric indefinite systems accurately in a sparse nonlinear optimization algorithm, in which the indefinite matrices are the so-called KKT matrices. The solutions to poorly conditioned systems led to convergence difficulties in the optimization code. Simple, incomplete fixes to the factorization scheme that forced a tight bound on $\|L\|$ solved the convergence problem. These matrices were large and sparse, generated by a black box code, resulting in stranger scalings than one might expect. Section 2.7.1 contains simple prescriptions for creating small dense problems for which a Bunch–Kaufman linear equation solver gives significantly lower accuracy than our alternatives.

The Bunch–Kaufman mixture of large and small quantities has subtle dangers beyond somewhat inaccurate solutions. A seemingly minor change to the factorization procedure for the LAPACK implementation[2] makes the entire factorization potentially unstable in cases where $\|L\|$ is large (see Appendix A). The problem here again is that small results are computed as the difference of much larger terms, yet the change is really a substitution of a spectral or singular value decomposition for solving 2×2 systems instead of Cramer's rule. One would have expected this to be more stable, but it does not provide the needed componentwise backward stability condition. Neither does Gaussian elimination with complete pivoting, which in this context costs only one more comparison than partial pivoting. Implementations using complete pivoting are unstable, while partial pivoting is provably stable. (Apparently using QR or LQ factorizations for the 2×2 systems is satisfactory.) Similar problems

[2] The comments on LAPACK apply only to versions 2.0 and earlier.

afflict the LAPACK matrix inversion routine. The Bunch–Kaufman algorithm is on the edge of instability, but we can easily avoid its dangers by bounding the norms of the factors.

The main theme of this paper is simple: bound $\|L\|$. We present general solutions, which we obtain without a large penalty in efficiency. Our solutions address the dense problem and the sparse problem separately. The solutions are quite different because the constraints of sparsity prevent us from applying the dense solution to the sparse problem. The reader who is interested in only one of these problems will find that the two solutions are largely independent of one another. However, the analysis of the dense problem is important in motivating why we reject the Bunch–Kaufman algorithm for sparse problems.

The dense problem is presented in section 2. In section 2.1 we review the Bunch–Kaufman algorithm. Section 2.2 provides a restatement of the reduced matrix bound from [8], which we augment with analyses showing that L is not bounded in the usual ways. We introduce the bounded Bunch–Kaufman algorithm, a refinement of the Bunch–Kaufman algorithm that bounds the size of entries in L, in section 2.4. The refinement has a cost in efficiency, but the additional time usually is quite small compared with the total CPU time of the LINPACK or the unblocked LAPACK algorithm. An examination of the faster blocked algorithm in LAPACK leads to a second new algorithm, a refinement of the Bunch–Parlett algorithm. The fast Bunch–Parlett algorithm is presented in section 2.5. This algorithm often requires less work than the current LAPACK implementation and has comparable speed yet avoids the inaccuracy problems of LAPACK. The performance of both of the new algorithms depends in part on the numerical entries of the matrices being factored. We provide a set of examples that produce the worst cases for these algorithms in section 2.6. These bad cases are followed by a demonstration of the effectiveness, both accuracy and speed, of the new algorithms on other matrices; this is the subject of section 2.7. Finally, we discuss the alternative LTL^T factorization of Aasen in section 2.8 and give summary conclusions for the dense problem in section 2.9.

We consider sparse algorithms in section 3. Our discussion, like our experience, begins with Liu's sparse variant of the Bunch–Kaufman algorithm [29]. Liu's algorithm is presented in section 3.1, where we also give an elementary discussion of the constraints imposed by the goal of preserving sparsity. In section 3.2 we show that Liu's algorithm shares the defect of not bounding $\|L\|$. In fact, compromises made for sparsity exacerbate the problem considerably. We show that sparsity also prevents us from applying the pivoting strategy that solved the dense problem. In section 3.3 we turn to a different approach, an explicit block factorization. Where the Bunch–Kaufman algorithms achieve stability bounds in an implicit manner, by placing simple constraints on the relative sizes of the entries in allowable pivot blocks, we turn instead to inexpensive, but explicit, tests. We compare our explicit tests to the similar tests used in the sparse algorithm by Duff and Reid [14, 16, 17, 18] in section 3.4. Our tests match the later versions of the Duff–Reid algorithm, although our motivation differs from Duff and Reid. Our new framework provides new arguments for use of the revised Duff–Reid algorithm and completes a proof of stability. In section 3.5 we present a new strategy for determining which potential pivots to test, a strategy that emphasizes 2 × 2 pivots and that reduces the extra fill caused by pivoting for stability. We follow this with a discussion of some strategies to identify larger pivot blocks, from which we hope to gain additional speed from use of block operations. The algorithm in section 3.6 is a much simpler block algorithm than those developed from Bunch–Kaufman-like algorithms. We take a short detour to discuss the difficulty of developing a sparse LTL^T factorization in section 3.7. In section 3.8 we present numerical results on sparse examples. Finally, we close with a brief discussion on the difficulties of developing a sparse variant of Aasen's algorithm, which we have not pursued.

2. Implicitly stable algorithms for dense matrices.

Aasen's algorithm, the Bunch–Kaufman algorithm, and the Bunch–Parlett algorithm all provide stable factorizations of symmetric indefinite matrices. The Bunch–Kaufman algorithm and the Bunch–Parlett algorithm both factor A as $PAP^T = LDL^T$, where L is triangular and D is block diagonal, composed solely of 1×1 and 2×2 blocks. Aasen's algorithm computes an LTL^T factorization, where T is symmetric tridiagonal. The Bunch–Kaufman algorithm has advantages over the other two algorithms in that

- it requires $\mathcal{O}(n^2)$ comparisons, whereas the Bunch–Parlett algorithm requires $\mathcal{O}(n^3)$;
- it usually requires fewer interchanges of columns than the Bunch–Parlett algorithm;
- it is noticeably easier to compute the matrix inertia and solve linear equations with its block diagonal factor than the tridiagonal factor of Aasen's algorithm;
- its outer-product form fits sparse data structures better than the inner-product form of Aasen's algorithm does.

Unfortunately the Bunch–Kaufman algorithm has worse accuracy than its two competitors. Our goal, which we achieve in two different ways, is an algorithm that retains most of the advantages of the Bunch–Kaufman algorithm and has superior accuracy. We begin with a review of the Bunch–Kaufman algorithm.

2.1. The Bunch–Kaufman pivoting strategy.

The Bunch–Kaufman algorithm factors A as $PAP^T = LDL^T$, where L is triangular and D is a matrix of 1×1 and 2×2 diagonal blocks. The schematic matrix in Figure 2.1 is used to illustrate the reduction step. Here a_{r1} is the off-diagonal entry of largest magnitude in the first column of the current reduced matrix $A^{(k-1)}$, with $\gamma_1 = |a_{r1}|$; $\gamma_r \geq \gamma_1$ is the magnitude of the largest off-diagonal entry in the rth column (or row). The Bunch–Kaufman pivoting strategy (technically, Algorithm A of [8]) is given in Figure 2.2.

$$\begin{array}{cc}
\left[\begin{array}{cccccc}
a_{11} & \cdot & \cdot & a_{r1} & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
a_{r1} & \cdot & \cdot & a_{rr} & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot
\end{array}\right] &
\begin{array}{c} \gamma_1 \\ \\ \\ \gamma_r \\ \\ \\ \end{array} \\
\begin{array}{cccccc} \gamma_1 & & & \gamma_r & & \end{array} &
\end{array}$$

FIG. 2.1. *Bunch–Kaufman pivot entries.*

The if then else structure should be interpreted in the normal programming language sense; the tests should proceed in the order given in order to obtain the required stability bound. (Case 3 is independent of cases 1 and 2, but the specified order has the advantage of reducing the number of interchanges.) The parameter α is usually

```
if $\gamma_1 = 0$ then
        {0}: nothing necessary for column $k$
else if $|a_{11}| \geq \alpha\gamma_1$ then
        {1}: use $a_{11}$ as a $1 \times 1$ pivot
else if $|a_{11}|\gamma_r \geq \alpha\gamma_1^2$ then
        {2}: use $a_{11}$ as a $1 \times 1$ pivot
else if $|a_{rr}| \geq \alpha\gamma_r$ then
        {3}: use $a_{rr}$ as a $1 \times 1$ pivot (interchange columns)
else
        {4}: use $\begin{bmatrix} a_{11} & a_{r1} \\ a_{r1} & a_{rr} \end{bmatrix}$ as a $2 \times 2$ pivot (interchanging columns)
end if
```

FIG. 2.2. *Bunch–Kaufman pivot selection.*

chosen as $\alpha = (1 + \sqrt{17})/8$. This choice makes the bound on reduced matrix growth for a 2×2 pivot equal the bound on maximum reduced matrix growth for two consecutive 1×1 pivots and thereby minimizes the worst case growth for an arbitrary selection of 1×1 and 2×2 pivots.

2.2. Bounds for the Bunch–Kaufman pivoting strategy. The seminal Bunch–Kaufman paper [8] presents a classical backward analysis for the Bunch–Kaufman algorithm, deriving an upper bound on the growth of elements in the reduced matrices $A^{(k)}$. We replicate this analysis to show how carefully the Bunch–Kaufman algorithm controls the interaction of the magnitudes of different components of the factorization. We extend the analysis to show that the Bunch–Kaufman algorithm does not bound the size of entries in the factor L below any constant multiple of $\|A\|$.

In a typical step of the factorization, after eliminating $k-1$ columns, we proceed from the reduced matrix $A^{(k-1)}$ to $A^{(k)}$ with a 1×1 pivot or to $A^{(k+1)}$ with a 2×2 pivot. Consider first a 1×1 pivot with the first column of $A^{(k-1)}$. (For notational simplicity we include the superscript on the reduced matrix only for the entries of the new reduced matrix.) Let the entries of the triangular factor L of $A^{(k-1)}$ be denoted by l_{ij}. Then

$$\begin{aligned} a_{ij}^{(k)} &= a_{ij} - \frac{a_{i1}a_{1j}}{a_{11}}, \quad i > 1, \quad j > 1, \\ l_{i1} &= \frac{a_{i1}}{a_{11}}, \quad i > 1. \end{aligned}$$

A 1×1 pivot with column r of $A^{(k-1)}$ is similar. A 2×2 pivot with columns 1 and r of $A^{(k-1)}$ results in

$$\begin{aligned} a_{ij}^{(k+1)} &= a_{ij} - \begin{bmatrix} a_{i1} & a_{ir} \end{bmatrix} \begin{bmatrix} a_{11} & a_{r1} \\ a_{r1} & a_{rr} \end{bmatrix}^{-1} \begin{bmatrix} a_{1j} \\ a_{rj} \end{bmatrix} \\ &= a_{ij} - \frac{1}{(a_{11}a_{rr} - a_{r1}^2)} \begin{bmatrix} a_{i1} & a_{ir} \end{bmatrix} \begin{bmatrix} a_{rr} & -a_{r1} \\ -a_{r1} & a_{11} \end{bmatrix} \begin{bmatrix} a_{1j} \\ a_{rj} \end{bmatrix} \\ &= a_{ij} - \frac{a_{11}a_{ir}a_{rj} - a_{r1}(a_{i1}a_{rj} + a_{ir}a_{1j}) + a_{rr}a_{i1}a_{1j}}{a_{11}a_{rr} - a_{r1}^2}, \quad i \neq 1, r, \quad j \neq 1, r, \end{aligned}$$

and

$$\begin{bmatrix} l_{i1} & l_{ir} \end{bmatrix} = \begin{bmatrix} a_{i1} & a_{ir} \end{bmatrix} \begin{bmatrix} a_{11} & a_{1r} \\ a_{1r} & a_{rr} \end{bmatrix}^{-1}.$$

Componentwise, the new entries of L are

$$\begin{aligned} l_{i1} &= \frac{a_{i1}a_{rr} - a_{r1}a_{ir}}{a_{11}a_{rr} - a_{r1}^2}, \quad i \neq 1, r, \\ l_{ir} &= \frac{a_{11}a_{ir} - a_{r1}a_{i1}}{a_{11}a_{rr} - a_{r1}^2}, \quad i \neq 1, r. \end{aligned}$$

(We assume that exchanges involving rows and columns 1 and r of A take place after the reduction step.)

Let μ be the magnitude of the largest entry in $A^{(k-1)}$ and μ' the maximum magnitude of any entry in the new reduced matrix. The Bunch–Kaufman algorithm guarantees that μ' increases only slightly above μ. We also want to monitor the size of entries allowed in L and the conditioning of individual blocks. We examine the computation for each of the four pivot choices.

- Case 1. a_{11} is a 1×1 pivot, with $|a_{11}| \geq \alpha\gamma_1$. It follows that

$$\begin{aligned} a_{ij}^{(k)} &= a_{ij} - \frac{a_{i1}a_{1j}}{a_{11}}, \\ |a_{ij}^{(k)}| &\leq |a_{ij}| + \frac{|a_{i1}a_{1j}|}{|a_{11}|} \leq |a_{ij}| + \gamma_1 \frac{|a_{i1}|}{|a_{11}|}, \\ \mu' &\leq \mu + \gamma_1 \frac{\gamma_1}{|a_{11}|} \leq \mu \left(1 + \frac{1}{\alpha}\right), \end{aligned}$$

and

$$l_{i1} = \frac{a_{i1}}{a_{11}} \;\Rightarrow\; |l_{i1}| \leq \frac{1}{\alpha}.$$

We obtain a relative growth bound for $A^{(k)}$ and an absolute bound for L.

- Case 2. a_{11} is a 1×1 pivot, with $|a_{11}|\gamma_r \geq \alpha\gamma_1^2$, from which follows $\frac{\gamma_1}{|a_{11}|} \leq \left(\frac{\gamma_r}{\gamma_1}\right)\left(\frac{1}{\alpha}\right)$. Then

$$\begin{aligned} a_{ij}^{(k)} &= a_{ij} - \frac{a_{i1}a_{1j}}{a_{11}}, \\ |a_{ij}^{(k)}| &\leq \mu + \frac{\gamma_1^2}{|a_{11}|}, \\ \mu' &\leq \mu + \frac{\gamma_r}{\alpha} \leq \mu \left(1 + \frac{1}{\alpha}\right), \end{aligned}$$

and

$$l_{i1} = \frac{a_{i1}}{a_{11}} \;\Rightarrow\; |l_{i1}| \leq \frac{\gamma_1}{|a_{11}|} \leq \left(\frac{\gamma_r}{\gamma_1}\right)\left(\frac{1}{\alpha}\right).$$

Here is the problem. Although growth in the reduced matrix is bounded exactly as in case 1, *there is no upper bound on* γ_r/γ_1, *and so there is no bound on the size of entries in* L. We will show in section 2.7.1 how to create examples where elements of L become arbitrarily large.

- Case 3. a_{rr} is a 1×1 pivot, with $|a_{rr}| \geq \alpha\gamma_r$. As in case 1,
$$\begin{aligned} a_{ij}^{(k)} &= a_{ij} - \frac{a_{ir}a_{rj}}{a_{rr}}, \; i,j \neq r, \\ |a_{ij}^{(k)}| &\leq \mu + \gamma_r \frac{\gamma_r}{|a_{rr}|}, \\ \mu' &\leq \mu\left(1 + \frac{1}{\alpha}\right), \end{aligned}$$
and, for $i \neq r$,
$$l_{ir} = \frac{a_{ir}}{a_{rr}} \;\Rightarrow\; |l_{ir}| \leq \frac{1}{\alpha}.$$
Again, we obtain satisfactory bounds for $A^{(k)}$ and L.
- Case 4. $\left[\begin{smallmatrix} a_{11} & a_{r1} \\ a_{r1} & a_{rr} \end{smallmatrix}\right]$ is a 2×2 pivot. Then
$$a_{ij}^{(k+1)} = a_{ij} - \frac{a_{11}a_{ir}a_{rj} - a_{r1}(a_{i1}a_{rj} + a_{ir}a_{1j}) + a_{rr}a_{i1}a_{1j}}{a_{11}a_{rr} - a_{r1}^2}, \; i \neq 1, r, \; j \neq 1, r.$$
By case 2, $|a_{11}\gamma_r| < \alpha\gamma_1^2$. By case 3, $|a_{rr}| < \alpha\gamma_r$. Together these imply that $|a_{11}a_{rr}| < \alpha^2\gamma_1^2$. In turn, $|a_{11}a_{rr} - \gamma_1^2| \geq \gamma_1^2 - |a_{11}a_{rr}| > \gamma_1^2(1-\alpha^2)$. Therefore
$$\begin{aligned} |a_{ij}^{(k+1)}| &\leq |a_{ij}| + \left|\frac{a_{11}a_{ir}a_{rj} - a_{r1}(a_{i1}a_{rj} + a_{ir}a_{1j}) + a_{rr}a_{i1}a_{1j}}{a_{11}a_{rr} - \gamma_1^2}\right| \\ &< \mu + \frac{|a_{11}|\gamma_r^2 + \gamma_1(\gamma_1\gamma_r + \gamma_r\gamma_1) + |a_{rr}|\gamma_1^2}{\gamma_1^2(1-\alpha^2)}. \end{aligned}$$
The conditions $|a_{rr}| < \alpha\gamma_r$ and $|a_{11}|\gamma_r < \alpha\gamma_1^2$ from the failure of cases 2 and 3 come into play again to simplify the bound on $|a_{ij}|$ as
$$\begin{aligned} |a_{ij}^{(k+1)}| &< \mu + \frac{\gamma_1^2(\alpha\gamma_r + 2\gamma_r + \alpha\gamma_r)}{\gamma_1^2(1-\alpha^2)}, \\ \mu' &\leq \mu\left(1 + \frac{2(1+\alpha)}{1-\alpha^2}\right) = \mu\left(1 + \frac{2}{1-\alpha}\right) \;. \end{aligned}$$
Now consider the entries l_{i1} and l_{ir} of the two new columns of L, with $i \neq 1, r$:
$$\begin{aligned} l_{i1} &= \frac{a_{i1}a_{rr} - a_{r1}a_{ir}}{a_{11}a_{rr} - a_{r1}^2}, \\ |l_{i1}| &< \frac{\gamma_1(\alpha\gamma_r) + \gamma_1\gamma_r}{\gamma_1^2(1-\alpha^2)} = \frac{\gamma_1\gamma_r(1+\alpha)}{\gamma_1^2(1-\alpha^2)} = \left(\frac{\gamma_r}{\gamma_1}\right)\left(\frac{1}{1-\alpha}\right), \\ l_{ir} &= \frac{a_{11}a_{ir} - a_{r1}a_{i1}}{a_{11}a_{rr} - a_{r1}^2}, \\ |l_{ir}| &< \frac{|a_{11}|\gamma_r + \gamma_1^2}{\gamma_1^2(1-\alpha^2)} < \frac{\gamma_1^2(1+\alpha)}{\gamma_1^2(1-\alpha^2)} = \frac{1}{1-\alpha} \;. \end{aligned}$$
Here is the same problem—the ratio γ_r/γ_1 appears in the bound for the first of the two new columns of L, with the 2×2 pivot giving a slightly larger bound on the size of elements of L than case 2.
The 2×2 block of D resulting from case 4 can also be arbitrarily badly conditioned; see section 2.7.1 for examples. As with the other aberrant bounds, this cannot occur if γ_r/γ_1 is near unity, as we will see in the following section.

In summary, the Bunch–Kaufman algorithm ensures that the entries in the reduced matrix are bounded. However, the bounds on the entries in L are a function of γ_r/γ_1, which is not controlled. The conditions that cause large entries in L in cases 2 and 4 do not cause growth in the reduced matrices because of the intricate relationships of the values of a_{11}, a_{r1}, and a_{rr}. Indeed, the somewhat odd case 2 is simply the last condition that is needed to avoid growth in the reduced matrix in case 4. Otherwise, case 2 does not appear to be a good thing to do.

Bounding the reduced matrix controls only one aspect of the process. We will show that a simple change to the algorithm allows us to bound the entries in L by a small constant independent of A and to give a similar bound on the conditioning of any 2×2 pivot block.

2.3. The Bunch–Parlett algorithm. The Bunch–Kaufman algorithm is the successor to the Bunch–Parlett algorithm, which uses the pivot selection shown in Figure 2.3. The Bunch–Parlett algorithm requires $\mathcal{O}(n^2)$ comparisons at each step. A complete factorization requires $\mathcal{O}(n^3)$ comparisons, the same complexity as the arithmetic to compute the factorization. In contrast, the Bunch–Kaufman algorithm requires only n or $2n$ comparisons at each step, $\mathcal{O}(n^2)$ overall. This much more reasonable overhead is the primary reason that the Bunch–Kaufman algorithm is preferred over the Bunch–Parlett algorithm.

```
Find the diagonal entry $a_{ss}$ of largest magnitude
Find an off-diagonal entry $a_{ri}$ of maximum magnitude in $A^{(k-1)}$
if $\max\{|a_{ss}|, |a_{ri}|\} > 0$ then
    if $|a_{ss}| \geq \alpha|a_{ri}|$ then
        use $a_{ss}$ as a $1 \times 1$ pivot
    else
        use $\begin{bmatrix} a_{ii} & a_{ri} \\ a_{ri} & a_{rr} \end{bmatrix}$ as a $2 \times 2$ pivot (interchanging columns)
    endif
endif
```

FIG. 2.3. *Bunch–Parlett pivot selection.*

However, there are no questions regarding the stability of the Bunch–Parlett algorithm. A complete analysis of the factorization and solution phases is given in [7]. Alternatively, it is easy to modify the analysis of the previous section to demonstrate that the Bunch–Parlett algorithm bounds the entries of L near unity. The key observations are that in a 2×2 step $|a_{ri}| = \gamma_i = \gamma_r$ and $|a_{ri}| \geq \max\{|a_{ii}|, |a_{rr}|\}/\alpha$. It follows that $|l_{ij}| \leq 1/\alpha$ in a 1×1 pivot column and $|l_{ij}| \leq 1/(1-\alpha)$ in a 2×2 pivot column. We will also show in the next section that the diagonal blocks corresponding to 2×2 pivots have a condition number no larger than $(1+\alpha)/(1-\alpha)$. Solving 2×2 systems with almost any algorithm, in particular either Cramer's rule or Gaussian elimination with partial pivoting, is normwise backward stable, and so is the entire factorization.

2.4. A bounded Bunch–Kaufman algorithm. The ratio of the two magnitudes γ_1 and γ_r controls the size of entries of L computed by the Bunch–Kaufman algorithm. We cannot control this ratio as long as we insist that the first column of the reduced matrix be involved in the pivot selection. By a more suitable choice of columns we can develop an algorithm that combines the stability properties of Bunch–Parlett with search costs little more than Bunch–Kaufman. The idea is simple: allow the aberrant cases only when the ratio γ_r/γ_i is unity.

Our simplest variant of the Bunch–Kaufman algorithm we denote as the *bounded Bunch–Kaufman* algorithm, which uses the pivot selection strategy presented in Figure 2.4. The condition $\gamma_r/\gamma_i = 1$ removes the possibility of case 2. Because 2×2 pivots are used only when $\gamma_r/\gamma_i = 1$, the analysis of section 2.2 shows that the entries of L are bounded above by $\max\{1/(1-\alpha), 1/\alpha\}$.

```
$\gamma_1 \leftarrow$ maximum magnitude of any off-diagonal entry in column 1
if $\gamma_1 = 0$ then
    nothing necessary for this column
else if $|a_{11}| \geq \alpha\gamma_1$ then
    use $a_{11}$ as a $1 \times 1$ pivot
else
    $i \leftarrow 1; \gamma_i = \gamma_1$
    repeat
        $r \leftarrow$ row index of first entry of maximum magnitude in column $i$
        $\gamma_r \leftarrow$ maximum magnitude of any off-diagonal entry in column $r$
        if $|a_{rr}| \geq \alpha\gamma_r$ then
            use $a_{rr}$ as a $1 \times 1$ pivot
        else if $\gamma_i = \gamma_r$ then
            use $\begin{bmatrix} a_{ii} & a_{ri} \\ a_{ri} & a_{rr} \end{bmatrix}$ as a $2 \times 2$ pivot (interchanging columns)
        else
            $i \leftarrow r; \gamma_i \leftarrow \gamma_r$
        endif
    until
        a pivot is chosen
endif
```

FIG. 2.4. *Bounded Bunch–Kaufman pivot selection.*

By bounding $\|L\|$, this algorithm bounds growth in the reduced matrix. Suppose that columns k and $k+1$ of the factorization represent a 2×2 step. Let $D_{k:k+1}$ be the 2×2 diagonal block, with off-diagonal entry of magnitude γ_r and diagonal entries bounded by $\alpha\gamma_r$. Let $L_{*,k:k+1}$ denote the $(n-k-1) \times 2$ matrix holding the interesting entries from the corresponding pair of columns of L. The triangle inequality shows that any entry in $L_{*,k:k+1} D_{k:k+1} L^T_{*,k:k+1}$ is bounded by $2\gamma_r(1+\alpha)/(1-\alpha)^2$. The resulting bound on growth in the reduced matrix is somewhat larger than the explicit bound from section 2.2, but it is trivial to derive. (Obviously this particular scheme satisfies the same bound on growth as the Bunch–Kaufman algorithm.)

Further, the diagonal blocks are all well conditioned. Note that a 2×2 block $D_{k:k+1}$ occurs only when $|a_{ri}| = \gamma_r = \gamma_i$ and $\max\{|a_{ii}|, |a_{rr}|\} \leq \alpha\gamma_r$. A Gerschgorin circle bound demonstrates that $|\lambda_{\max}(D_{k:k+1})| \leq |a_{ri}|(1+\alpha)$. It follows from $|\det(D_{k:k+1})| > |a_{ri}|^2(1-\alpha^2)$ that $|\lambda_{\min}| > |a_{ri}|(1-\alpha)$ and $\kappa(D_{k:k+1}) < (1+\alpha)/(1-\alpha)$. So 2×2 systems can be solved using explicit inversion of $D_{k:k+1}$ or by Gaussian elimination with row interchanges that use the off-diagonal entry as the first pivot entry. (Note that this is complete pivoting in this special case.)

The condition that L is bounded near unity is key to showing backward stability. The choice of columns is crucial. The heart of this algorithm is the search for an off-diagonal entry a_{ri} that is simultaneously largest in the rth row and in the ith column. This search must terminate after no more than n steps because the size of the off-diagonal pivot entry increases at each step.

The cost of this search is an issue. Our algorithm lies between the Bunch–Kaufman and the Bunch–Parlett algorithms in numbers of comparisons. It is easy to create cases in which the cost of a single factorization step is the same as either of these algorithms. In the worst case it is no better than the Bunch–Parlett algorithm; in the best case it costs no more than the Bunch–Kaufman algorithm. In general it should not cost as much as the Bunch–Parlett algorithm because the condition we need is less stringent than the maximum off-diagonal element. We need only an element that is the largest entry in both the row and column of A in which it appears, a "local maximum" off-diagonal entry.

In practice we have found that the process of finding a local maximum off-diagonal entry usually requires far less than $\mathcal{O}(n^2)$ comparisons. We do not consider random matrices to be adequate reflections of real applications, but in this case they are useful indications of why we see what we find. (Trefethen and Schreiber [31] provide arguments that random matrices adequately represent the limiting behavior of the factorization scheme.) Our empirical results, given in sections 2.7.2 and 2.7.4, show that, on average, fewer than $2.5n$ comparisons suffice to find a suitable pair of columns. In Appendix C we present a probabilistic analysis that shows that the expected number of comparisons needed to find a local maximum off-diagonal entry is bounded above by $en \approx 2.718n$ for any matrix with all entries drawn independently from the same random distribution. We expect a smaller cost for the bounded Bunch–Kaufman algorithm because acceptable 1×1 pivots shortcut the process. A related analysis for the probability of taking 1×1 pivots is discussed in section 2.5.

2.5. A fast Bunch–Parlett algorithm. Both the Bunch–Kaufman and the bounded Bunch–Kaufman algorithms can be described as variants of the Bunch–Parlett algorithm, in which the conditions for both a 1×1 and a 2×2 pivot are weakened in order to reduce the cost of the pivot search. The 2×2 condition in the Bunch–Kaufman algorithm is too weak to bound L. The bounded Bunch–Kaufman algorithm requires a local maximum off-diagonal entry, which does bound L.

The same approach can be used to modify the Bunch–Parlett algorithm directly. Allowing a 1×1 pivot when the largest diagonal entry is a large enough fraction of the largest entry in its column and a 2×2 pivot only when the off-diagonal entry in the pivot block is locally maximum leads to the *fast Bunch–Parlett* pivot selection scheme shown in Figure 2.5.

At first glance this algorithm is simply the bounded Bunch–Kaufman algorithm with the extra condition that the initial pivot candidate is the column holding the largest diagonal value; that condition removes the need for 1×1 pivot tests in the inner loop.[3] The fast Bunch–Parlett algorithm has the same stability properties as the bounded Bunch–Kaufman and Bunch–Parlett algorithms. It always requires more comparisons than the Bunch–Kaufman algorithm because of the search to find the largest diagonal entry. We would expect the fast Bunch–Parlett algorithm to be

[3] Modifying the Bunch–Kaufman algorithm by choosing the largest diagonal entry was suggested in [8], where it was argued that finding the largest diagonal improved the numerical results. An explanation of that improvement in accuracy is that such an algorithm usually bounds L because the initial test at each step almost always finds a satisfactory 1×1 pivot.

```
$a_{ss} \leftarrow$ the diagonal entry of largest magnitude in $A^{(k-1)}$
$\gamma_s \leftarrow$ magnitude of largest off-diagonal entry in column $s$
if $\gamma_s = 0$ then
    nothing necessary for the $s$th column
else if $|a_{ss}| \geq \alpha\gamma_s$ then
    use $a_{ss}$ as a $1 \times 1$ pivot
else
    $i \leftarrow s; \gamma_i = \gamma_s$
    repeat
        $r \leftarrow$ row index of first entry of maximum magnitude in column $i$
        $\gamma_r \leftarrow$ maximum magnitude of any off-diagonal entry in column $r$
        if $\gamma_i = \gamma_r$ then
            use $\begin{bmatrix} a_{ii} & a_{ri} \\ a_{ri} & a_{rr} \end{bmatrix}$ as a $2 \times 2$ pivot (interchanging columns)
        else
            $i \leftarrow r; \gamma_i \leftarrow \gamma_r$
        endif
    until
        a pivot is chosen
endif
```

FIG. 2.5. *Fast Bunch–Parlett pivot selection.*

slightly slower than the bounded Bunch–Kaufman algorithm because of this extra work and because it requires more column interchanges. This is indeed what we find in a conventional, unblocked factorization, as shown in section 2.7.2.

However, the fast Bunch–Parlett algorithm has dramatic advantages when we consider a partitioned or blocked factorization, as in LAPACK. The blocked version of the fast Bunch–Parlett algorithm, discussed in section 2.7.3, can require less overall work than blocked versions of either the Bunch–Kaufman or the bounded Bunch–Kaufman algorithm. Our experience, described in section 2.7.4, is that the blocked fast Bunch–Parlett algorithm is faster than the blocked bounded Bunch–Kaufman algorithm and is very competitive with the block Bunch–Kaufman algorithm.

The difference is that the largest diagonal entry is a much better candidate for a 1×1 pivot than the first (or any arbitrary) diagonal entry, at least in a probabilistic sense. Work is discarded in a blocked implementation when a column is tested as a pivot but not used (see section 2.7.3). The blocked algorithm is most efficient when the *initial* test for a 1×1 pivot is satisfied. The choice of the maximum diagonal value makes this occur frequently, at least on random matrices. A formal argument is given in Appendix D. Table 2.1 gives these probabilities numerically for matrices whose entries are all $\mathcal{N}(0, 1)$, as a function of n.

Informally, the difference between these algorithms is the difference between an arbitrary entry and an entry that is the maximum of n entries. The initial pivot test compares one of these with α times the maximum of $n-1$ other entries. As n increases, the likelihood of an arbitrary entry satisfying the test decreases, but the likelihood of the maximum diagonal satisfying the test increases. Our concern is with the entire

Table 2.1
Likelihood of satisfying initial 1×1 *pivot test.*

	Matrix order						
Starting entry	10	20	30	50	100	150	200
Arbitrary diagonal	.26	.19	.15	.12	.09	.07	.06
Maximum diagonal	.88	.93	.95	.97	.99	.99	.99
(Ratio)	3.3	5.0	6.2	8.0	11.2	13.5	15.5

Table 2.2
Summary of pivoting for $20,000$ $\mathcal{N}(0,1)$ *random matrices.*

	Bunch–Kaufman	Bounded Bunch–Kaufman	Fast Bunch–Parlett
1×1 with first column	14%	14%	97%
Columns in 1×1 pivots	38%	44%	97%
Columns in 2×2 pivots	62%	56%	3%

factorization, not just a single column. We present in Table 2.2 the percentage of cases where the initial 1×1 test is satisfied throughout the course of the factorizations of 20,000 400×400 matrices generated by LAPACK's pseudo-$\mathcal{N}(0,1)$ generator.

We are not the first to observe the rapid convergence of the search for a local maximum off-diagonal entry. Fletcher, in a nearly forgotten paper [19], introduced what we have called the bounded Bunch–Kaufman algorithm as part of a scheme for reducing the search cost of the Bunch–Parlett algorithm. However, he considered this algorithm to be only a stepping stone to another algorithm that converged yet more rapidly, at the cost of looser bounds on growth in the reduced matrix. This algorithm is very close to the bounded Bunch–Kaufman algorithm with smaller values of α. He considered the latter aspect to be the major contribution of his method. The crucial property of the local maximum off-diagonal entry for Fletcher was only that it allowed a 2×2 pivot that bounded growth in the reduced matrix. We rediscovered this algorithm from our desire to use the stronger property of the local maximum off-diagonal entry, that it provides a 2×2 pivot that bounds L.

Fletcher's algorithm was supplanted by the Bunch–Kaufman algorithm, which appeared to have all the required properties and lower cost. His idea of choosing looser bounds was derived from early ideas for sparse linear systems and is seen in the dense matrix literature as being more appropriate to sparse systems [20]. However, the underpinnings of the search are applicable only to dense problems, as shown in section 3.2. From the sparse perspective [17], this is a dense matrix algorithm. In fact the paper contains ideas that are still relevant to both types of problems.

2.6. Worst cases for the bounded Bunch–Kaufman and fast Bunch–Parlett algorithms. It is easy to generate examples in which the current step of either the bounded Bunch–Kaufman or the fast Bunch–Parlett algorithm requires $\mathcal{O}(n^2)$ comparisons. The following examples, suggested to us by Stan Eisenstat, provoke this behavior at every step. For the bounded Bunch–Kaufman algorithm, choose a matrix from the family

$$\begin{bmatrix} 0 & & & 2 \\ & 4 & 4 & \\ & 4 & 0 & 3 \\ 2 & & 3 & 0 \end{bmatrix}, \quad \begin{bmatrix} 0 & & & & & 2 \\ & 6 & 6 & & & \\ & 6 & 0 & 5 & & \\ & & 5 & 0 & 4 & \\ & & & 4 & 0 & 3 \\ 2 & & & & 3 & 0 \end{bmatrix}, \quad \begin{bmatrix} 0 & & & & & & & 2 \\ & 8 & 8 & & & & & \\ & 8 & 0 & 7 & & & & \\ & & 7 & 0 & 6 & & & \\ & & & 6 & 0 & 5 & & \\ & & & & 5 & 0 & 4 & \\ & & & & & 4 & 0 & 3 \\ 2 & & & & & & 3 & 0 \end{bmatrix}, \quad \ldots .$$

These require a search of the entire matrix at each step. Further, each step terminates with a 1 × 1 pivot, achieving the obvious maximum in terms of comparisons.

Similarly, for the fast Bunch–Parlett algorithm, choose a matrix from the family

$$\begin{bmatrix} \frac{1}{4} & & & 2 \\ & 0 & 4 & \\ & 4 & 0 & 3 \\ 2 & & 3 & 0 \end{bmatrix}, \quad \begin{bmatrix} \frac{1}{6} & & & & & 2 \\ & 0 & 6 & & & \\ & 6 & 0 & 5 & & \\ & & 5 & 0 & 4 & \\ & & & 4 & 0 & 3 \\ 2 & & & & 3 & 0 \end{bmatrix}, \quad \begin{bmatrix} \frac{1}{8} & & & & & & & 2 \\ & 0 & 8 & & & & & \\ & 8 & 0 & 7 & & & & \\ & & 7 & 0 & 6 & & & \\ & & & 6 & 0 & 5 & & \\ & & & & 5 & 0 & 4 & \\ & & & & & 4 & 0 & 3 \\ 2 & & & & & & 3 & 0 \end{bmatrix}, \quad \ldots .$$

The fast Bunch–Parlett algorithm will make a complete search at each step. Because the fast Bunch–Parlett algorithm terminates in a 2 × 2 pivot whenever the search is nontrivial, its worst case has half as many comparisons as the bounded Bunch–Kaufman or Bunch–Parlett algorithms.

2.7. Empirical studies of a LAPACK-like implementation. Do the bounded Bunch–Kaufman and fast Bunch–Parlett algorithms really attain accuracy superior to the Bunch–Kaufman algorithm, and at what cost?

We consider LAPACK [2] to represent the current state of the art, so we used the double precision DSYTxx suite of Bunch–Kaufman subprograms from LAPACK version 2.0 as the basis for a Fortran 77 implementation of both of the new algorithms. We made one small but significant modification to the unblocked LAPACK Bunch–Kaufman factorization to avoid a true instability induced by our test cases, as discussed in Appendix A; otherwise our Bunch–Kaufman codes are standard LAPACK. We also used the LAPACK certification programs to evaluate the results of our modifications. Our goals were to demonstrate the effect of large entries in L on the solution of the linear system and to verify empirically our claim of $\mathcal{O}(n)$ comparisons to find a local maximum off-diagonal entry, showing thereby the efficiency of this scheme for choosing a pivot that bounds L.

2.7.1. Accuracy. The conditions that result in large backward error bounds for the Bunch–Kaufman factorization require that the entire leading column of the reduced matrix be much smaller than the entries in the rth column. The LAPACK tests are unlikely to produce such matrices. To test the effects of large entries in L, we generate matrices that allow large entries in L in the very first pivot step. Each matrix has entries that are random $\mathcal{N}(0,\cdot)$, where the variance for the distribution is chosen as a function of the location of the entries. Figure 2.6 illustrates how the variances are assigned. We use the LAPACK pseudorandom number generator to generate all at once a matrix of $\mathcal{N}(0,1)$ entries; the variances ψ_1, ψ_2, and ψ_3 are then applied as scale factors as indicated in Figure 2.6.

$$\begin{array}{ccccc|ccccc}
\psi_1 & & & & & & & & & \\
\psi_2 & \psi_1 & & & & & & & & \\
\psi_2 & \psi_2 & \psi_1 & & & & & & & \\
\vdots & \vdots & \vdots & \ddots & & & & & & \\
\psi_2 & \psi_2 & \psi_2 & \dots & \psi_1 & & & & & \\
\hline
\psi_2 & \psi_2 & \psi_2 & \dots & \psi_2 & \psi_3 & & & & \\
\psi_2 & \psi_2 & \psi_2 & \dots & \psi_2 & 1 & \psi_3 & & & \\
\psi_2 & \psi_2 & \psi_2 & \dots & \psi_2 & 1 & 1 & \psi_3 & & \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \\
\psi_2 & \psi_2 & \psi_2 & \dots & \psi_2 & 1 & 1 & 1 & \dots & \psi_3 \\
 & & m & & & & & & &
\end{array}$$

FIG. 2.6. *Variance for each entry of test matrix.*

We likely induce a first column much smaller than any of the last $n - m$ columns by taking ψ_1 and ψ_2 both to be small. Further, we choose the variances, or scale factors, so that the initial pivot selection by the ordinary Bunch–Kaufman algorithm is either case 2 or case 4. Typically we take $\psi_1 \leq \psi_2/10$ and $\psi_3 \leq 1/10$ to make case 1 and case 3 unlikely. We expect instead to take a pivot that creates large entries in L, either a 1 × 1 pivot using the first column or a 2 × 2 pivot, where the magnitudes of the entries in the diagonal block satisfy

$$\left[\begin{array}{cc} |a_{11}| & |a_{r1}| \\ |a_{r1}| & |a_{rr}| \end{array}\right] \sim \left[\begin{array}{cc} \psi_1 & \psi_2 \\ \psi_2 & \psi_3 \end{array}\right].$$

We expect to induce a 1 × 1 case 2 pivot when ψ_1 and ψ_2 are small and close in size, so $\alpha\psi_2^2 < \psi_1$ $(< \alpha\psi_2)$; taking ψ_1 very small, so that $\psi_1 < \alpha\psi_2^2$, usually results in a 2 × 2 pivot. In either case the entries in L are bounded by $\mathcal{O}(1/\psi_2)$. The relative sizes of ψ_2 and ψ_3 control the conditioning of the 2 × 2 block.

Ill conditioning is a necessary precursor to large $\|L\|$. Generating large entries in L requires that the first column of the current reduced matrix be much smaller than at least one other column. The small Rayleigh quotient residual for the vector e_1 implies that the reduced matrix has an eigenvalue near a_{11}. Yet $\|A^{(k-1)}_{*j}\|_2 \gg |a_{11}|$, $j = m+1, \dots, n$, implies that the largest singular value of $A^{(k-1)}$ is much greater than $|a_{11}|$. Thus, large entries in L imply that at least one reduced matrix is ill conditioned. We can also show that the smallest singular value of a reduced matrix is no smaller than the smallest singular value of the larger matrix from which it was derived; that is, the sequence $\sigma_1^{(0)}, \sigma_1^{(1)}, \sigma_1^{(2)}, \dots$ is monotone increasing. The largest singular value can also grow from step to step but at a rate bounded by growth in the reduced matrix. Thus, if the $(k-1)$th reduced matrix has condition $\overline{\kappa}$, the original matrix had condition at least as large as $\overline{\kappa}/\left(n(1+1/\alpha)^{k-1}\right)$.

When m, the number of leading columns with tiny entries, is greater than one, a poor initial pivot will cause obvious damage to the first reduced matrix.[4] Our test matrices were a series of 10 randomly generated matrices, each subjected to each of

[4] For some distributions of values, a 2 × 2 pivot will be taken involving two of the first m columns. This allows the Bunch–Kaufman algorithm to escape the trap we have set for it.

the scalings in Table 2.3 for various values of m. The results reported here are for matrices of order 50, with $m = 3$. The results presented here are for unblocked codes. As noted above, the Bunch–Kaufman code incorporates the modification discussed in Appendix A; otherwise its factorization is irretrievably damaged on our extreme cases.

TABLE 2.3
Parameterization of scaled test matrices.

Initial pivot	ψ_1	ψ_2	ψ_3	t_1	t_2
1×1 pivot	10^{-t_1}	10^{-t_2}	1/10	$t_2 + 1, \ldots, 2t_2$	$1, \ldots, 6$
2×2 block					
Well-conditioned	10^{-t_1}	10^{-t_2}	10^{-t_2}	$2t_2 + 1, \ldots, 3t_2$	$2, \ldots, 6$
General	10^{-t_1}	10^{-t_2}	1/10	$2t_2 + 1, \ldots, 3t_2$	$1, \ldots, 6$

We used four measures[5] from the LAPACK certification program (Table 2.4) to evaluate our implementation of the bounded Bunch–Kaufman and fast Bunch–Parlett algorithms. Here x^* is the known solution to the linear system, from which the right-hand side is generated; x is the solution computed using the LDL^T factorization; $x^{(r)}$ is the result of fixed precision iterative refinement on x; ϵ is the usual machine precision; $\kappa_1(A) = \|A\|_1 \|A^{-1}\|_1$. All of these measures are normalized by worst case perturbation bounds, so that only results much larger than one are suspect. Many of our test matrices are poorly conditioned; the bounds allow substantial errors. We used n random right-hand sides with each matrix, where n is the order of the matrix.

TABLE 2.4
Accuracy measures.

Decomposition error	$\|LDL^T - A\|_1 \; / \; (n\|A\|_1 \, \epsilon)$
Error in inverse	$\|I - AA^{-1}\|_1 \; / \; (n\kappa_1(A)\epsilon)$
Solution error	$\|x - x^*\|_1 \; / \; (\|x^*\|_1 \, \kappa_1(A)\epsilon)$
Refined solution error	$\|x^{(r)} - x^*\|_1 \; / \; (\|x^*\|_1 \, \kappa_1(A)\epsilon)$.

The major conclusion from these experiments is that the solutions from the Bunch–Kaufman algorithm were noticeably less accurate than the solutions from either the bounded Bunch–Kaufman or fast Bunch–Parlett algorithms; however, the Bunch–Kaufman solutions still passed the certification. As expected, all algorithms create backward stable factorizations. The results for bounded Bunch–Kaufman and fast Bunch–Parlett are almost identical.

The primary measure of interest is refined solution error. From [2] we expect that the Bunch–Kaufman algorithm together with fixed precision iterative refinement (FPIR) to be at least componentwise relatively forward stable. In fact, Higham's stability analysis of the Bunch–Kaufman algorithm suffices also to show componentwise relative backward stability of the Bunch–Kaufman algorithm followed by FPIR [23, 24]. We expect, therefore, to achieve higher accuracy by applying iterative refinement. Our experiments, generating random matrices at each of several sizes, show that

- FPIR is often very effective at reducing the Bunch–Kaufman solution error, giving considerably smaller refined solution errors in some cases. For our

[5] The particular measure used for accuracy of the inverse is what is used in the LAPACK certification.

certification suite the solution error without iterative refinement was as much as 4×10^5 larger than the refined solution error for matrices of order 50. In general, the degradation in accuracy reflected the size of $\|L\|$—a quick estimate of the reduced accuracy is $\|L\|_1/n$.

- The bounded Bunch–Kaufman and fast Bunch–Parlett algorithms appear empirically to achieve the same accuracy *without iterative refinement* as the Bunch–Kaufman algorithm achieves with iterative refinement. In the cases we observed, the refined solution error was never smaller than 1/5 of the solution error. For ill-conditioned problems, more than half the digits presumed lost in the worst case error bound are in fact good.

These results are consistent with the experience in applications that brought this issue to our attention in the first place. Used without iterative refinement, the Bunch–Kaufman algorithm may not give sufficient accuracy to ill-conditioned problems. Iterative refinement requires more computation and also mandates saving the original matrix A, thereby doubling storage.

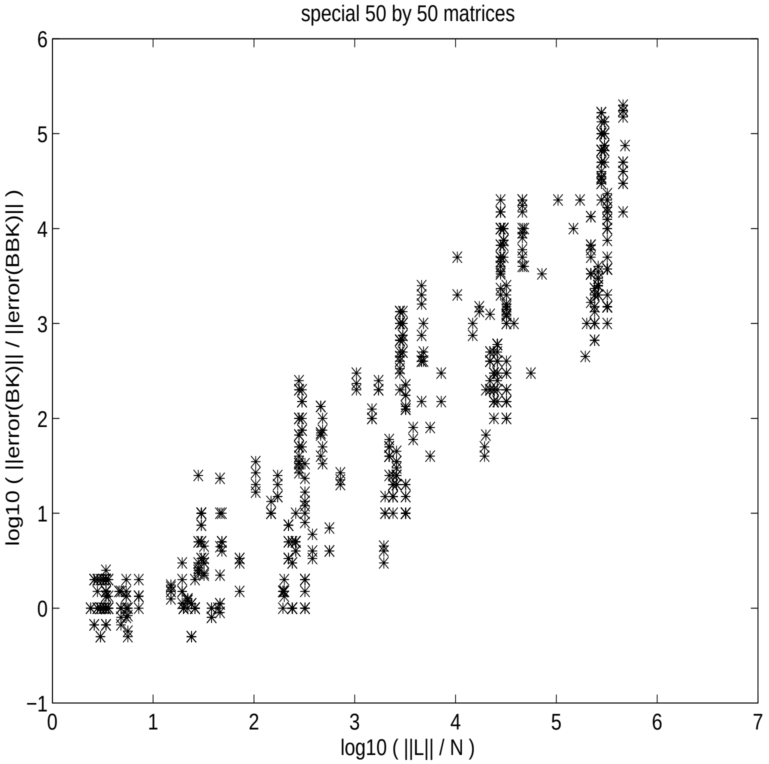


FIG. 2.7. *Comparison of relative solution error on scaled* $\mathcal{N}(0,1)$ *random matrices.*

Figure 2.7 presents a comparison of the solution error from a Bunch–Kaufman implementation and a bounded Bunch–Kaufman implementation using matrices of order 50, generated according to the prescription in Table 2.3 with $m = 3$. The figure is a log-log scatter plot of the Bunch–Kaufman solution error scaled by the bounded Bunch–Kaufman solution error. The upward trend demonstrates the dependence of the Bunch–Kaufman solution error on $\|L\|$. These results are typical for $m \geq 2$; the tests with $m = 1$ do not produce large relative errors despite having large entries in L.

In contrast, comparisons of the Bunch–Kaufman solution error after iterative refinement, the bounded Bunch–Kaufman solution error before and after iterative refinement, or the fast Bunch–Parlett solution error before and after iterative refinement are quite uninteresting. The range of variation for these algorithms on the problems shown in Figure 2.7 is given in Table 2.5.

Table 2.5
Comparison of relative solution error on scaled $\mathcal{N}(0,1)$ matrices.

Error Relative to Bounded Bunch–Kaufman Error				
	min.	median	average	max.
Bunch–Kaufman with refinement	0.25	0.60	0.50	1.5
Bounded Bunch–Kaufman with refinement	0.25	0.59	0.50	1.5
Fast Bunch–Parlett	0.33	1.00	0.99	3.0
Fast Bunch–Parlett with refinement	0.25	0.61	0.50	1.5

2.7.2. Efficiency of unblocked factorizations. What is the cost of better accuracy? We used random symmetric matrices, with each entry drawn from $\mathcal{N}(0,1)$. Table 2.6 summarizes the total search requirements for each procedure to factor 20,000 random matrices of order 400. The results confirm our probabilistic analysis and show that the search for a local maximum converges very quickly. The results for all three algorithms reflect the fact that the pivot searches terminate immediately when a satisfactory 1×1 pivot is found.

Table 2.6
Pivoting cost for $20,000$ *random* $\mathcal{N}(0,1)$ *matrices.*

		Bunch– Kaufman	Bounded Bunch– Kaufman	Fast Bunch– Parlett
Average:	column searches per pivot step	1.80	2.44	1.02
	total comparison operations†	0.46	0.67	0.75‡
	column interchanges $/n$	0.45	0.75	0.98
Worst case:	column searches for a single pivot	2	10	7
	total comparison operations†	0.51	0.76	0.75‡

†As percent of estimated total operations $\frac{n^3}{3} + \frac{n^2}{2} + \frac{19n}{6}$.
‡Includes comparisons to find largest diagonal entry.

Each column search in $A^{(k-1)}$ requires $n-k$ comparisons. The total number of comparisons is presented as a percentage of the estimated number of floating point operations needed to compute the factorization. In the worst case, for bounded Bunch–Kaufman pivoting added only 0.76% work and for fast Bunch–Parlett pivoting added only 0.75%. On average, the price of better accuracy for bounded Bunch–Kaufman is 0.21% more floating point work and for fast Bunch–Parlett 0.29%.

The results on these matrices confirms our expected decreasing order of complexity for pivot searches and interchanges, Bunch–Parlett, fast Bunch–Parlett, bounded Bunch–Kaufman, and Bunch–Kaufman. (Fast Bunch–Parlett requires the equivalent of an additional column search to find the largest diagonal entry and interchanges cost more than searches.) The real issue is performance. Our implementations are a simple modification of the LAPACK algorithm. How do they compare to the LAPACK Bunch–Kaufman algorithm in speed?

We compared the three unblocked, full storage mode implementations on a Sun SPARCstation 20, using the standard Fortran 77 distribution of LAPACK version 2.0. The LAPACK codes make heavy use of BLAS2 and BLAS3 kernels. We used a highly efficient Fortran 77 version of dgemm and dgemv for RISC computers obtained from Michel Dayde at CERFACS, to which we added the needed BLAS2-like kernels following the same programming style. (See [6] for an alternative source of high performance kernels.) The figures in Table 2.7 are "effective megaflops per second," that is, time required in seconds divided by millions of operations required to computed an unpivoted LDL^T factorization, averaged over 600 symmetric random $\mathcal{N}(0,1)$ matrices of each order.

Table 2.7
Performance (megaflops/sec) on Sun SPARCstation 20*—symmetric random* $\mathcal{N}(0,1)$ *matrices.*

Blocking		Matrix order							
		10	20	40	80	160	320	400	500
None	BK	3.7	9.0	15.3	19.4	20.4	19.3	17.4	15.9
	BBK	3.6	8.3	13.7	18.0	19.4	19.4	16.8	15.1
	FBP	3.1	6.7	11.7	14.8	13.8	13.8	11.8	10.7
8	BK	2.0	5.0	9.8	14.7	17.7	19.2	19.1	18.9
	BBK	1.9	4.7	8.9	13.4	17.0	18.7	18.5	18.3
	FBP	1.5	4.0	7.8	12.6	16.0	18.1	18.0	18.2
24	BK			12.9	18.4	23.8	27.9	28.7	29.2
	BBK			11.0	15.7	21.4	25.7	26.4	26.9
	FBP			10.6	15.9	21.7	25.6	26.5	27.2
32	BK			12.1	18.2	22.9	26.4	27.8	23.9
	BBK			10.2	15.3	20.2	24.0	25.2	20.5
	FBP			10.0	15.8	19.0	23.6	25.2	23.0
64	BK				14.6	16.7	14.4	14.5	14.4
	BBK				11.2	13.7	11.9	12.1	12.2
	FBP				12.7	16.2	14.5	14.7	14.7

The first three rows of figures in Table 2.7 show that an unblocked implementation of the bounded Bunch–Kaufman algorithm compares quite well to an unblocked implementation of the Bunch–Kaufman algorithm. The cost of higher accuracy is small. The fast Bunch–Parlett algorithm begins with $2n-1$ comparisons, already more comparisons than the Bunch–Kaufman algorithm. This scheme will be effective only if the n comparisons to find the largest diagonal reduce work elsewhere. In unblocked algorithms there is no wasted work, so we expect and observe the fast Bunch–Parlett algorithm to be slower than the Bunch–Kaufman and bounded Bunch–Kaufman algorithms. The fast Bunch–Parlett algorithm uses far fewer 2×2 pivots, which also reduces speed in the unblocked case.

2.7.3. Blocking in LAPACK. The LAPACK codes attain high performance on sophisticated architectures through use of *blocked* factorization schemes.[6] These allow the use of BLAS2 and BLAS3 kernels, which amortize memory accesses over n^2 or n^3 operations.

The LAPACK implementation of the Bunch–Kaufman algorithm computes the reduced matrix $A^{(k-1)}$ as an explicit Schur complement. The partial factorization

[6] Technically the LAPACK algorithms are *partitioned* algorithms, but the term "blocked" has become accepted in this context. Block thereby has two different meanings in this paper, which may be clarified by noting that blocking in the LAPACK Bunch–Kaufman algorithm represents large partitions containing 1×1 and 2×2 blocks.

after $k-1$ steps can be written as

$$\begin{bmatrix} A_{11} & A_{21}^T \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} L_{11} & 0 \\ L_{21} & I \end{bmatrix} \begin{bmatrix} D_{11} & 0 \\ 0 & A^{(k-1)} \end{bmatrix} \begin{bmatrix} L_{11}^T & L_{21}^T \\ 0 & I \end{bmatrix},$$

where A_{11} is square, of order $k-1$, $A_{11} = L_{11}D_{11}L_{11}^T$, and $A_{21} = L_{21}D_{11}L_{11}^T$. It follows that

$$A^{(k-1)} = A_{22} - (L_{21}D_{11})\, L_{21}^T,$$

which is the basis for the blocked LAPACK Bunch–Kaufman algorithm. The matrix $W_{21} = L_{21}D_{11}$ is an intermediate result in the computation of L_{21}; it is stored in temporary storage, which allows $A^{(k-1)}$ to be computed by a matrix-matrix multiplication and addition.

Pivoting complicates the use of this block structure. At the kth step pivoting can require access to *any* column of $A^{(k-1)}$. The blocked version of this algorithm is therefore not as effective as blocked versions of other algorithms. A somewhat incomplete description of the difficulties is given in [12]. Kaufman [27] discusses inefficiencies that arise because the BLAS3 kernels do not include a matrix-matrix product where the product is known to be symmetric. As a result, all of the blocked codes discussed here avoid duplicating work or storage by making only restricted use of general matrix-matrix products.

The LAPACK implementation uses the blocked equation for the reduced matrix in two ways. The matrix-matrix operation $A^{(k-1)} = A_{22} - W_{21}L_{21}^T$ is used every b steps to generate an explicit reduced matrix. Here b is a blocking parameter set as a function of machine characteristics and available temporary storage; typically b ranges between 8 and 64. Matrix-matrix operations are not used otherwise (as, say, to generate $L_{21} = A_{21}L_{11}^{-T}D_{11}^{-1}$) because of the pivoting constraints.

At intermediate steps matrix-vector multiplications are used to generate the columns of $A^{(k-1)}$ that are needed for pivoting. The first column of the reduced matrix is always required in the Bunch–Kaufman algorithm; the rth column is computed only if the first column fails as a 1 × 1 pivot in case 1. We expect this to happen frequently. When it does, two of the three remaining cases result in 1 × 1 pivots. In both of these cases, the work required to compute the column not used in the pivot is simply discarded. Only in cases 1 and 4 of the Bunch–Kaufman algorithm are up to $2b(n-k)$ operations not discarded. There is a conflict between discarded work, which increases with the blocking parameter and thereby suggests use of small blocks, and performance of the BLAS3 kernels, which improves with large blocks.

This conflict can be exacerbated in the bounded Bunch–Kaufman and fast Bunch–Parlett algorithms, where additional searches to find an appropriate pivot will create more work that must be discarded. That is a serious issue for the bounded Bunch–Kaufman algorithm. But it is here that the fast Bunch–Parlett algorithm performs well because its search usually terminates in a single step, wasting no work.[7]

[7] The implicit representation of the reduced matrix has other effects on the bounded Bunch–Kaufman and fast Bunch–Parlett algorithms. The entry a_{ir} is computed by a different computation than a_{ri}, causing the search termination test to fail because of differences in rounding errors. We replace the test $\gamma_i = \gamma_r$ by $\gamma_i \geq \gamma_r/(1+\sqrt{\epsilon})$. Even this tolerance can result in one unnecessary matrix-vector product in a case where the remaining columns represent the (near) nullspace of a rank-deficient matrix. In addition the fast Bunch–Parlett algorithm needs access to the diagonal entries from the reduced matrix. Since these values are not otherwise cheaply available, they are maintained separately and explicitly updated at each step, requiring $\mathcal{O}(n^2)$ extra work.

TABLE 2.8
Factorization performance on CRI computers.

	Megaflops/sec for $n = 400$					
		YMP			C-90	
b	BK	BBK	FBP	BK	BBK	FBP
1	168	168	165	312	300	334
32	167	152	156	269	261	257
48	194	168	182	352	320	331
64	202	173	193	382	333	355
80	207	175	202	413	341	372

2.7.4. Efficiency of blocked factorizations. Table 2.7 shows the performance of all three algorithms for various blocking sizes on a Sun SPARCstation 20. As expected, discarded work increases the overhead of the blocked bounded Bunch–Kaufman algorithm. The relative performance of the fast Bunch–Parlett algorithm improves because less work is discarded. The computer architecture of this pipelined scalar RISC machine with two levels of cache storage results in slowdowns when the storage requirements exceed the cache sizes. This occurs in the unblocked case when the matrix itself is too large for the secondary cache, which particularly penalizes 1×1 pivots. The blocked algorithms slow when the blocksize becomes so large that $b \times b$ blocks no longer fit in the primary cache.

Table 2.8 gives effective megaflops/sec for the three algorithms on Cray Y-MP and C-90 computers for symmetric random $\mathcal{N}(0, 1)$ matrices of size 400. The results from CRI supercomputers are similar to the results on the Sun SPARCstation 20 but without cache effects. The anomalous performance of the unblocked fast Bunch–Parlett algorithm on the C-90 is due to a fast assembly language kernel for a rank 1 modification, which the fast Bunch–Parlett algorithm favors. Normally we expect rank 2 modifications, and hence the unblocked bounded Bunch–Kaufman algorithm, to be faster, but here the rank 2 kernel is in Fortran 77 and is slower than the rank 1 kernel in assembly language.

By choosing appropriately between the bounded Bunch–Kaufman or fast Bunch–Parlett algorithms, the incremental cost of attaining better accuracy in solving dense symmetric indefinite linear systems is clearly small. (As an ironic example, the unblocked routine used by the blocked fast Bunch–Parlett algorithm to process the final block reduced matrix should be the unblocked bounded Bunch–Kaufman algorithm, not the unblocked fast Bunch–Parlett algorithm. The timings here reflect the latter.) Our experiments showed no appreciable difference in accuracy between the fast Bunch–Parlett and bounded Bunch–Kaufman algorithms. The choice between them should be made on the basis of speed in a particular setting.

2.8. $\boldsymbol{LTL^T}$ factorizations. The Parlett–Reid algorithm [30] is an alternative to LDL^T factorizations. It factors a symmetric matrix A as

$$A = LTL^T,$$

where L is unit lower triangular, with no entries of magnitude larger than one, and T is symmetric tridiagonal. Stability of this algorithm follows immediately if the requisite solves with T are based on a stable factorization, such as a pivoted LU factorization or an orthogonal QR factorization.

The algorithm is straightforward. Assume that the first q steps reduce A to the form

$$\begin{bmatrix} T_q & \beta_q e_q e_1^T \\ \beta_q e_1 e_q^T & A^{(q)} \end{bmatrix},$$

where T_q is tridiagonal of order q. Then factorization step $q+1$ consists of

- symmetrically interchanging rows and columns $q+2$ and m, where $A^{(q)}_{m,q+1}$ has the maximum magnitude among $A^{(q)}_{q+2,q+1}, A^{(q)}_{q+3,q+1}, \ldots, A^{(q)}_{n,q+1}$, the off-diagonal entries in column $q+1$.
- Choose the elementary elimination matrix $E_{q+1} = I_n - we^T_{q+2}$ to eliminate rows $q+3$ to n of column $q+1$. Then $A^{(q+1)} = E_{q+1}P_{q+1}A^{(q)}P^T_{q+1}E^T_{q+1}$ is a matrix with the tridiagonal form extended by one row and column.

This algorithm requires only a search of a single column of the reduced matrix at each elimination step, less searching than any of the LDL^T factorizations. Nonetheless, it is not currently the algorithm of choice in either the dense or sparse cases.

The first obstacle to its use is that computing $E_{q+1}P_{q+1}A^{(q)}P^T_{q+1}E^T_{q+1}$ directly costs twice as much as the equivalent step of a Cholesky factorization or the LDL^T factorizations. This operation can be written as a rank 2 modification $M - uw^T - wu^T$, but not as a symmetric rank 1 modification. An outer product (right-looking) implementation requires more operations than the worst pivoting cases of our stable LDL^T factorizations.

Aasen [1] overcame this obstacle with a left-looking (inner product) algorithm, for which the high-order term in the operations count is identical to the Cholesky algorithm. Aasen's observation is that at each elimination step only the leading column of the reduced matrix is needed to determine the pivoting sequence and the elementary transformation. Aasen computes this leading column through the unsymmetric factorization

$$A = LTL^T = (L)(TL^T) = LH,$$

where $H = TL^T$ is upper Hessenberg. The matrix H is not stored. At the jth elimination step, the jth column of H is computed, using the first j columns of L and T, which are stored and have already been computed. The leading column of the reduced matrix, the jth column of $A^{(j-1)}$ is computed as the product $LH_{*,j}$, after which $H_{*,j}$ is discarded. Only the first j columns of L are required because of the Hessenberg structure of H.

Aasen's algorithm stands alone as the only stable algorithm which has $n^3/6$ complexity in operations, a guarantee of no more than $n^2/2$ work in searches, and well-behaved bounds on the entries of its factors. Should it not be the algorithm of choice?

The issue is speed. Barwell and George [5] compared unblocked versions of Aasen's algorithm and the Bunch–Kaufman algorithm, among others, and concluded that there was negligible difference in performance between these fastest two candidates in their study. A more recent LAPACK project report [3] compared unblocked and blocked versions of these two algorithms. They concluded that Aasen's algorithm was faster asymptotically in the unblocked case and slower in the blocked case. Unfortunately this study is somewhat incomplete in that no details of the blocked algorithms were given and only factorization times were considered. The test codes used are apparently lost. Further, the range of machines is limited and obsolete.

Nonetheless, it is easy to predict the trends of these conclusions for modern implementations. The difference in the unblocked case is attributable to the difference in speed between a matrix-vector product (Aasen) and a low rank modification (Bunch–Kaufman). This advantage is probably diminished when symmetric (packed) storage is used. It is lost more in the blocked case when both algorithms are based on matrix-matrix multiplications. With the playing field more level, three disadvantages of Aasen's algorithm become more apparent. The cost of a second search in Bunch–Kaufman is more than offset by the cost of computing the intermediate result $H_{*,j}$ in Aasen's algorithm. The factorization of T and solves with those factors also represent entirely scalar bottlenecks within Aasen's scheme, which become more prominent as the scalar bottlenecks of the other algorithms are addressed.

2.9. Summarizing the current state of the dense problem. Is one of these algorithms the universal algorithm of choice? It appears not. Rather it appears that the fastest algorithm for a given application will depend on the architecture of the target platform and perhaps on numerical properties of the application. Here are some of the issues:

- The Bunch–Kaufman algorithm should not be used as a general black-box solver, except in conjunction with FPIR. Iterative refinement requires storing a copy of the original matrix.
- Where code simplicity is important, or where unblocked algorithms are otherwise desirable, the leading candidates are Aasen's algorithm, the bounded Bunch–Kaufman algorithm, and perhaps the Bunch–Kaufman algorithm with FPIR. More study is needed to determine the relative performance of implementations with symmetric packed storage.
- Where high performance is the goal, the leading candidate is the blocked fast Bunch–Parlett algorithm. Aasen's algorithm and perhaps Bunch–Kaufman with FPIR are also possible. More study is needed of blocked implementations of Aasen's algorithm and of the effect of symmetric packed storage.
- The relative performance of the solution phases, which differ for LDL^T and LTL^T factorizations, should be considered.
- Application-specific characteristics may render the fast Bunch–Parlett or bounded Bunch–Kaufman algorithms ineffective or may enable the Bunch–Kaufman algorithm to be used safely.

Unfortunately we have added to the complexity of the decision of finding the best algorithm for the symmetric indefinite linear system problem. The user seeking best performance must participate in the decision.

3. Explicitly stable pivoting algorithms for sparse matrices. We return to our original goal, accurate solutions of *sparse* symmetric indefinite systems. There are two keys to solving the sparse indefinite problem accurately. The pivoting scheme must ensure stability and accuracy while minimizing the impact of pivoting on sparsity. The data structures of the factorization must gracefully permit pivoting with minimum overhead. Duff and Reid's multifrontal algorithm [17] provides both. In particular, the multifrontal data structures seem especially appropriate for the indefinite case. We used the multifrontal algorithm as the basis for our sparse indefinite factorization code [4], which appears in a number of industrial application codes.

However, the popularity of the dense Bunch–Kaufman algorithm and the superior sparsity results reported by Liu [29] led us to implement Liu's sparse variant of the Bunch–Kaufman algorithm in place of the Duff–Reid pivot selection scheme. It was the failure of Liu's algorithm that led us to reconsider the Bunch–Kaufman algorithm in general. In the remainder of this section we provide an analysis of accuracy and stability for a series of pivot selection algorithms for sparse indefinite systems.

We begin (section 3.1) by reviewing Liu's threshold algorithm. In section 3.2 we show that, like the dense Bunch–Kaufman algorithm, it does not provide a satisfactory bound on $\|L\|$. However, the tools that fixed the dense Bunch–Kaufman algorithm are unavailable in the sparse case and we are unable to prescribe a simple set of simple rules, comparing only the magnitudes of entries of A, that has the byproduct of providing implicitly a bound on $\|L\|$. Instead we developed a *block* pivot selection algorithm based explicitly on bounding $\|L\|$, from which stability and accuracy follow. The analysis of our explicit bounds on $\|L\|$ for 2×2 pivots is given in section 3.3. We compare our strategy to the algorithm of Duff et al. [14, 16, 17, 18] in section 3.4. Our pivot acceptance test is the same as the later version of their algorithm, but our motivation, bounding $\|L\|$, is different. The perspective espoused in section 3.3 thereby provides a proof of stability for the Duff–Reid algorithm.

Our view of symmetric indefinite pivoting as a block pivoting problem leads beyond theoretical results. Within the standard tradition of 2×2 and 1×1 pivots it leads to a more effective strategy for choosing sparse pivots, presented in section 3.5. It also opens the possibility of block pivots of order larger than two, which may increase the speed of the algorithm. We present this generalization in section 3.6. Following a digression in section 3.7 to explain why we do not pursue a sparse version of Aasen's algorithm, we conclude with a series of experiments (section 3.8) that demonstrate the performance of our approaches.

3.1. Sparse threshold Bunch–Kaufman pivoting. In the sparse symmetric indefinite multifrontal algorithm [17], the task at a given step is to compute the *partial* factorization of a dense symmetric indefinite (frontal) matrix. That is, given the partitioned matrix

$$A = \begin{bmatrix} A_{11} & A_{21}^T \\ A_{21} & A_{22} \end{bmatrix},$$

the goal is to compute an LDL^T factorization of the $n_1 \times n_1$ matrix A_{11} so that this factorization is stable, the overall LDL^T factorization of A is stable, and knowledge of the entries of A_{22} is not required. (In fact, A_{22} is not available in the multifrontal algorithm.)

Liu's sparse threshold Bunch–Kaufman strategy [29] is given in Figure 3.1. It appears to represent only a slight change from the standard Bunch–Kaufman algorithm. Figure 3.2 shows the location of key entries in the partitioned strategy. Here a_{r1} is the entry of largest magnitude on the off-diagonal of the first column, with $\gamma_1 = |a_{r1}|$. To respect the partitioning of A, a_{q1} is chosen to be the largest off-diagonal entry in the first column of A_{11}. The largest off-diagonal entry in column q has magnitude γ_q. Note that a_{r1}, a_{rr}, and γ_r are shown only for contrast with the standard algorithm; they do not enter into the threshold scheme. The role of the variables $\widehat{\alpha}$ and τ is described below.

There are two essential characteristics of the sparse scheme. The potential second pivot column must be taken from the first block column (the "fully assembled" part of the frontal matrix) because the entries in A_{22} are unavailable. The parameters $\widehat{\alpha}$ and τ permit weaker stability conditions that allow an elimination step to take place when the usual Bunch–Kaufman conditions cannot be satisfied in the fully assembled part of the frontal matrix. Liu intends the user to set the parameter $\widehat{\alpha}$ to strike a balance between degradation of accuracy and loss of sparsity. Often $\widehat{\alpha}$ will be significantly smaller than the Bunch–Kaufman α. Following [9], the parameter τ is introduced so that the bound on growth of entries in two consecutive 1×1 pivots is the same as the

```
if $\gamma_1 = 0$ then
    $\{\mathbf{0}\}$: nothing necessary for the first column
else if $|a_{11}| \geq \widehat{\alpha}\gamma_1$ then
    $\{\mathbf{1}\}$: use $a_{11}$ as a $1 \times 1$ pivot
else if $|a_{11}|\gamma_q \geq \widehat{\alpha}\gamma_1^2$ then
    $\{\mathbf{2}\}$: use $a_{11}$ as a $1 \times 1$ pivot
else if $|a_{qq}| \geq \widehat{\alpha}\gamma_q$ then
    $\{\mathbf{3}\}$: use $a_{qq}$ as a $1 \times 1$ pivot
else if $|a_{q1}| \geq \tau\gamma_1$ then
    $\{\mathbf{4}\}$: use $\begin{bmatrix} a_{11} & a_{q1} \\ a_{q1} & a_{qq} \end{bmatrix}$ as a $2 \times 2$ pivot
else
    no pivot found; repeat search using next column
end if
```

FIG. 3.1. *Liu's threshold Bunch–Kaufman pivot strategy.*

$$\begin{array}{cccc|cccc|c}
a_{11} & \cdot & a_{q1} & \cdot & \cdot & a_{r1} & \cdot & \cdot & \gamma_1 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \\
a_{q1} & \cdot & a_{qq} & \cdot & \cdot & \cdot & \cdot & \cdot & \gamma_q \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \\
\hline
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \\
a_{r1} & \cdot & \cdot & \cdot & \cdot & a_{rr} & \cdot & \cdot & \gamma_r \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \\
 & & & & & & & & \\
\gamma_1 & & \gamma_q & & & \gamma_r & & &
\end{array}$$

FIG. 3.2. *Threshold Bunch–Kaufman pivot selection in* $A^{(k-1)}$.

bound for a single 2×2 pivot. (In [29] this condition is shown to be satisfied when τ is a root of $4\widehat{\alpha}^3 + 3\widehat{\alpha}^2 - 2\tau^2\widehat{\alpha} - \tau^2$, restricted to the interval $(\widehat{\alpha}, 1]$.) Consequently, it is easier to take either 1×1 or 2×2 pivots from A_{11}. The threshold τ is smaller than one, which permits 2×2 pivots when the off-diagonal entry is other than the largest in the first column. This is clearly important when the largest entry is found in A_{21} and is unavailable as a pivot entry. However, the cost of $\widehat{\alpha} < \alpha$ and $\tau < 1$ is a larger bound on the growth of entries in the reduced matrix.

Even with weaker conditions we may be unable to find any pivots for A_{11}, in which case we adjust the partitioning of A to account for this failure. That is, we redefine the partitioning so that A_{11} represents the submatrix in the "fully assembled" part of the frontal matrix that we were able to factor stably. Columns removed from the first block column are not eliminated until later frontal steps. The order of A_{22} increases, which generally increases both the storage and the number of operations required for the factorization of the larger sparse matrix. The data structures of the multifrontal algorithm permit these rearrangements more easily than other sparse factorization schemes. Nonetheless, preservation of sparsity requires that as many pivots as possible be chosen from A_{11}.

3.2. Analysis of sparse threshold Bunch–Kaufman pivoting. In section 2.2 we showed that the Bunch–Kaufman algorithm does not bound the size of entries in the factor L by any constant multiple of $\|A\|$. Here we provide the analogous analysis for Liu's threshold algorithm. The bound for the reduced matrix is from Liu [29].

Cases 1, 2, and 3 of Liu's algorithm are essentially the same as standard Bunch–Kaufman. If μ' is the magnitude of the largest entry in the new reduced matrix, the analyses in section 2.2 carry over directly to show that

$$\begin{aligned}
&\text{case 1:} \quad \mu' \le \mu\left(1+\frac{1}{\widehat{\alpha}}\right), \ |l_{i1}| \le \frac{1}{\widehat{\alpha}},\\
&\text{case 2:} \quad \mu' \le \mu\left(1+\frac{1}{\widehat{\alpha}}\right), \ |l_{i1}| \le \left(\frac{\gamma_q}{\gamma_1}\right)\left(\frac{1}{\widehat{\alpha}}\right),\\
&\text{case 3:} \quad \mu' \le \mu\left(1+\frac{1}{\widehat{\alpha}}\right), \ |l_{i1}| \le \frac{1}{\widehat{\alpha}}.
\end{aligned}$$

The bound of concern is the bound for $|l_{i1}|$ in the second case. As before, γ_q/γ_1 can be arbitrarily large. The trade-off between sparsity and stability will usually result in $\widehat{\alpha}$ strictly less than α, so these bounds are worse than the bounds for the dense case.

The analysis of case 4 is nearly the same as in section 2.2. Here we use

$$\begin{bmatrix} a_{11} & a_{q1} \\ a_{q1} & a_{qq} \end{bmatrix}$$

as a 2×2 pivot. Then

$$a_{ij}^{(k+1)} = a_{ij} - \frac{a_{11}a_{iq}a_{qj} - a_{q1}(a_{i1}a_{qj} + a_{iq}a_{1j}) + a_{qq}a_{i1}a_{1j}}{a_{11}a_{qq} - a_{q1}^2}, \quad i \ne 1, q, \ j \ne 1, q.$$

The failure to take either case 2 or case 3 means that $|a_{11}\gamma_q| < \widehat{\alpha}\gamma_1^2$ and $|a_{qq}| < \widehat{\alpha}\gamma_q$. These together imply that $|a_{11}||a_{qq}| < \widehat{\alpha}^2\gamma_1^2$. The third ingredient is $a_{q1}^2 \ge \tau^2\gamma_1^2$ by the sparsity threshold condition. The definition of τ requires $\widehat{\alpha} < \tau$, so $|a_{11}a_{qq} - a_{q1}^2| > \gamma_1^2(\tau^2 - \widehat{\alpha}^2)$. It follows that

$$\begin{aligned}
|a_{ij}^{(k+1)}| &\le |a_{ij}| + \left|\frac{a_{11}a_{iq}a_{qj} - a_{q1}(a_{i1}a_{qj} + a_{iq}a_{1j}) + a_{qq}a_{i1}a_{1j}}{a_{11}a_{qq} - a_{q1}^2}\right|,\\
\mu' &< \mu + \frac{|a_{11}|\gamma_q^2 + 2\gamma_1\gamma_q|a_{q1}| + |a_{qq}|\gamma_1^2}{\gamma_1^2(\tau^2 - \widehat{\alpha}^2)}.
\end{aligned}$$

Again $|a_{11}\gamma_q| < \widehat{\alpha}\gamma_1^2$ and $|a_{qq}| < \widehat{\alpha}\gamma_q$ suffice to show that

$$\mu' < \mu + \frac{\gamma_1^2(\widehat{\alpha}\gamma_q + 2\gamma_q + \widehat{\alpha}\gamma_q)}{\gamma_1^2(\tau^2 - \widehat{\alpha}^2)} \le \mu\left(1 + \frac{2(1+\widehat{\alpha})}{(\tau^2 - \widehat{\alpha}^2)}\right).$$

Our real interest is with $L^{(k)}$, the kth elementary transformation, whose first two columns are given by

$$\begin{bmatrix} l_{i1} & l_{iq} \end{bmatrix} = \begin{bmatrix} a_{i1} & a_{iq} \end{bmatrix} \begin{bmatrix} a_{11} & a_{1q} \\ a_{1q} & a_{qq} \end{bmatrix}^{-1}.$$

The entries l_{i1} and l_{iq} of the two new columns of L, with $i \neq 1, q$, satisfy

$$\begin{aligned} l_{i1} &= \frac{a_{i1}a_{qq} - a_{q1}a_{iq}}{a_{11}a_{qq} - a_{q1}^2}, \\ |l_{i1}| &\leq \frac{\gamma_1(\widehat{\alpha}\gamma_q) + \min\{\gamma_1, \gamma_q\}\gamma_q}{\gamma_1^2(\tau^2 - \widehat{\alpha}^2)} \leq \left(\frac{\gamma_q}{\gamma_1}\right)\left(\frac{\widehat{\alpha}+1}{\tau^2 - \widehat{\alpha}^2}\right), \\ l_{iq} &= \frac{a_{11}a_{iq} - a_{q1}a_{i1}}{a_{11}a_{qq} - a_{q1}^2}, \\ |l_{iq}| &\leq \frac{\widehat{\alpha}\gamma_1^2 + \min\{\gamma_1, \gamma_q\}\gamma_1}{\gamma_1^2(\tau^2 - \widehat{\alpha}^2)} \leq \left(\frac{\widehat{\alpha}+1}{\tau^2 - \widehat{\alpha}^2}\right). \end{aligned}$$

Again γ_q/γ_1 reflects the fact that one of the two columns of $\|L\|$ can be arbitrarily large. Liu's algorithm suffers the same weakness in the bounds on L as the Bunch–Kaufman algorithm.

The condition of equal bounds on reduced matrix growth does not produce equal bounds on $\|L\|$ for 1×1 and 2×2 pivots. From the definition of τ one can derive

$$\tau^2 - \widehat{\alpha}^2 = \widehat{\alpha}^2\left(\frac{2\widehat{\alpha}+2}{2\widehat{\alpha}+1}\right).$$

Thus, the 2×2 pivot bound includes a $1/\widehat{\alpha}^2$ term in addition to the γ_q/γ_1 term. It is typical in sparse applications to take $\widehat{\alpha}$ small to preserve sparsity; $\widehat{\alpha} = 0.25, 0.1$, or even 0.001 have all been advocated in the literature. Unfortunately this means that entries in L from 2×2 pivots can be noticeably larger than those from 1×1 pivots.

3.3. Pivoting to bound L. The bounded Bunch–Kaufman and fast Bunch–Parlett algorithms bound L by taking a local maximum off-diagonal entry of A as the off-diagonal entry in a 2×2 pivot. These solutions are not available to us in the partitioned matrix because this special entry need not be in A_{11}. Using an entry that is a local maximum off-diagonal entry of A_{11}, but not a local maximum off-diagonal entry of A, puts no bounds on the entries in L. We need another mechanism to bound L.

One could envision computing, and possibly discarding, the entries of L, but the extra work is considerable for 2×2 pivots. Instead, we adopt a pivot selection strategy based on a computable bound for the entries of L that will result from a given 2×2 pivot, a bound that does not depend on the failure of 1×1 cases. From the equation

$$\left[\begin{array}{cc} l_{i1} & l_{iq} \end{array}\right] = \left[\begin{array}{cc} a_{i1} & a_{iq} \end{array}\right] \left[\begin{array}{cc} a_{11} & a_{1q} \\ a_{1q} & a_{qq} \end{array}\right]^{-1},$$

it follows that

$$\begin{aligned} |l_{i1}| &= \frac{|a_{i1}a_{qq} - a_{q1}a_{iq}|}{|a_{11}a_{qq} - a_{q1}^2|} \leq \frac{|a_{qq}|\gamma_1 + |a_{q1}|\gamma_q}{|a_{11}a_{qq} - a_{q1}^2|}, \\ |l_{iq}| &= \frac{|a_{11}a_{iq} - a_{q1}a_{i1}|}{|a_{11}a_{qq} - a_{q1}^2|} \leq \frac{|a_{11}|\gamma_q + |a_{q1}|\gamma_1}{|a_{11}a_{qq} - a_{q1}^2|}. \end{aligned}$$

These inequalities hold with no assumptions on the relative sizes of the entries in A.

We use these inequalities directly to obtain tests that guarantee the same bounds on the reduced matrix and $\|L\|$ that we would obtain in the best cases of our previous algorithms. Whether the bounding quantities are sufficiently small can be computed directly with relatively small expense as long as we know $a_{11}, a_{q1}, a_{qq}, \gamma_1,$ and γ_q.

Satisfying these bounds on L will also bound growth in the reduced matrix. Assume that $|l_{ij}|$ is bounded by $1/\widehat{\alpha}$ for all i and j. Then

$$\begin{aligned} |a_{ij}^{(k+1)}| &\leq |a_{ij}| + \left[\begin{array}{cc} |l_{i1}| & |l_{iq}| \end{array}\right] \left[\begin{array}{c} |a_{1j}| \\ |a_{qj}| \end{array}\right] \\ &\leq |a_{ij}| + \frac{2}{\widehat{\alpha}} \max_{u,v} |a_{uv}| \leq \left(1 + \frac{2}{\widehat{\alpha}}\right) \max_{u,v} |a_{uv}|. \end{aligned}$$

Thus, growth in the reduced matrix and the bound on $\|L\|$ will be acceptable if we take

$$D_{k,k+1} = \left[\begin{array}{cc} a_{11} & a_{1q} \\ a_{1q} & a_{qq} \end{array}\right]$$

as a pivot when

$$\max\left\{ \frac{|a_{qq}|\gamma_1 + |a_{q1}|\gamma_q}{|a_{11}a_{qq} - a_{q1}^2|}, \frac{|a_{11}|\gamma_q + |a_{q1}|\gamma_1}{|a_{11}a_{qq} - a_{q1}^2|} \right\} \leq \frac{1}{\widehat{\alpha}}.$$

By placing the same bound on L from a 2 × 2 step as a 1 × 1 step, we achieve both a satisfactory bound on L and a sharper bound on the growth in the reduced matrix than given by a Bunch–Kaufman-like algorithm. A simple explicit bounding strategy is given in Figure 3.3.

```
if $\gamma_1 = 0$ then
    nothing necessary for the first column
else if $|a_{11}| \geq \widehat{\alpha}\gamma_1$ then
    use $a_{11}$ as a $1 \times 1$ pivot
else if $|a_{qq}| \geq \widehat{\alpha}\gamma_q$ then
    use $a_{qq}$ as a $1 \times 1$ pivot
else if $\max\{|a_{qq}|\gamma_1 + |a_{q1}|\gamma_q, |a_{11}|\gamma_q + |a_{q1}|\gamma_1\} \leq \frac{|a_{11}a_{qq} - a_{q1}^2|}{\widehat{\alpha}}$ then
    use $\left[\begin{array}{cc} a_{11} & a_{q1} \\ a_{q1} & a_{qq} \end{array}\right]$ as a $2 \times 2$ pivot
else
    no pivot found; repeat search using next column
end if
```

FIG. 3.3. *An explicit bounding sparse pivot strategy.*

This strategy does not bound the condition number of a 2 × 2 pivot block. Consequently, backward stability requires that we use some normwise backward stable scheme for the solution of all 2 × 2 linear systems, not Cramer's rule. Our codes use Gaussian elimination with complete pivoting. Our choice of 2 × 2 pivot block does not assure that the diagonal blocks of D associated with 2 × 2 pivots have eigenvalues of opposite signs, so there is a minor increase in the complexity of computing the inertia of A. Note also that the 2 × 2 pivot bounds can be sharpened somewhat when $|a_{q1}|$ is equal to γ_1 or γ_q if the second largest off-diagonal entry in each of the two columns is known. A related idea appears in [19].

3.4. Comparison with the Duff–Reid algorithms. The original multifrontal factorization of Duff and Reid [17] included capabilities for symmetric indefinite systems. Here and in later papers Duff and Reid [14, 18] present an approach of essentially the same structure as the code in Figure 3.3 but based on bounding growth in the reduced matrix.

The Duff–Reid condition for 2×2 pivots bounds the modification to the i, jth entry of the reduced matrix,

$$\begin{aligned} a_{ij}^{(k+1)} &= a_{ij} - \delta_{ij} = a_{ij} - \left[\begin{array}{cc} a_{i1} & a_{iq} \end{array}\right] D_{k,k+1}^{-1} \left[\begin{array}{c} a_{1j} \\ a_{qj} \end{array}\right] \\ &= a_{ij} - \left[\begin{array}{cc} l_{i1} & l_{iq} \end{array}\right] \left[\begin{array}{c} a_{1j} \\ a_{qj} \end{array}\right]. \end{aligned}$$

The key requirement in [14, 16, 17, 18] is that $|\delta_{ij}| \leq \max_{u,v} |a_{uv}|/\widehat{\alpha}$, bounding growth in the reduced matrix. There are several tests that guarantee this condition without testing or even computing all components of columns of L. For example, the inequality

$$|\delta_{ij}| \leq \left\| \begin{array}{cc} a_{i1} & a_{iq} \end{array} \right\|_1 \left(\left\| \left[\begin{array}{cc} a_{11} & a_{q1} \\ a_{q1} & a_{qq} \end{array}\right]^{-1} \right\|_\infty \left\| \begin{array}{c} a_{1j} \\ a_{qj} \end{array} \right\|_\infty \right)$$

means that the reduced matrix growth condition is satisfied whenever

$$\left\| \left[\begin{array}{cc} a_{11} & a_{q1} \\ a_{q1} & a_{qq} \end{array}\right]^{-1} \right\|_\infty \max\{\gamma_1, \gamma_q\} \leq \frac{1}{\widehat{\alpha}} .$$

Duff and Reid used this normwise bound as a test in [17]. Note that it is equivalent to the bound $\|L_{1,q}\|_\infty \leq \|D_{k,k+1}^{-1}\|_\infty \|A_{1,q}\|_\infty \leq 1/\widehat{\alpha}$, so this approach bounds both growth in the reduced matrix and the size of L.

In practice this test proved unnecessarily severe. In their later papers [14, 18] Duff and Reid changed to a matrix magnitude test using $|D_{k,k+1}^{-1}|$. That is, from the inequality

$$|\delta_{ij}| \leq \left[\begin{array}{cc} |a_{i1}| & |a_{iq}| \end{array}\right] \left(\left|D_{k,k+1}^{-1}\right| \left[\begin{array}{c} |a_{1j}| \\ |a_{qj}| \end{array}\right] \right)$$

one can obtain the sufficient condition for bounding reduced matrix growth

$$|D_{k,k+1}^{-1}| \left[\begin{array}{c} \gamma_1 \\ \gamma_q \end{array}\right] \leq \left[\begin{array}{c} 1/\widehat{\alpha} \\ 1/\widehat{\alpha} \end{array}\right].$$

This is a sharper estimate of growth in the reduced matrix, which allows some blocks to be used as pivots that would be rejected by the earlier test. Again, this condition also bounds L through the inequality $\left[\begin{array}{cc} |l_{i1}| & |l_{iq}| \end{array}\right] \leq |D_{k,k+1}^{-1}| \left[\begin{array}{cc} |a_{i1}| & |a_{iq}| \end{array}\right] \leq \left[\begin{array}{cc} 1/\widehat{\alpha} & 1/\widehat{\alpha} \end{array}\right]$. Indeed, this second test is identical to the test we obtain from the goal of explicitly bounding L.

For Duff and Reid, bounding the entries in L was a means to the end of bounding growth in the reduced matrix. For us, bounding L is the goal, because that bound, coupled with a backward stable scheme for solving 2×2 linear systems, suffices to show backward stability for the entire process.

It is interesting to note three pitfalls into which focusing only on bounding $|\delta_{ij}|$ can lead. Using explicit inversion or Cramer's rule for solving the 2×2 systems is one. A more dangerous trap is to take one step further in refining the test than do Duff and Reid, to a componentwise magnitude bound. The inequality

$$|\delta_{ij}| \leq \left[\ |a_{i1}| \quad |a_{iq}|\ \right] \left|D_{k,k+1}^{-1}\right| \left[\begin{array}{c} |a_{1j}| \\ |a_{qj}| \end{array} \right]$$

leads to an explicit test to accept $D_{k,k+1}$ whenever

$$\frac{|a_{11}|\gamma_q^2 + 2|a_{q1}|\gamma_1\gamma_q + |a_{qq}|\gamma_1^2}{|a_{11}a_{qq} - a_{q1}^2|} \leq \frac{\max\{\gamma_1, \gamma_q\}}{\widehat{\alpha}}.$$

However, this is nothing more than an explicit version of the Bunch–Kaufman test. It provides no useful bound on L, as can be shown by the same counterexamples that afflict the Bunch–Kaufman algorithm. It bounds growth in the reduced matrix, but it does not suffice to show backward stability of the factorization, and it is likely to lead to other difficulties.

A third trap has been advocated in the literature. The tradition begun by Bunch and Parlett is to take the bound on reduced matrix growth from a 2×2 pivot to be the same as the square of the bound on a 1×1 pivot. In other words, the worst growth allowed in a double step is the same as the bound from two consecutive worst-case single steps. This has the aesthetic appeal of giving the same worst-case bounds for growth in the reduced matrix for all pivot sequences. In the form of the tests used by Duff and Reid this could be achieved by imposing the weaker condition

$$|\delta_{ij}| \leq \left(\frac{1}{\widehat{\alpha}^2} + \frac{2}{\widehat{\alpha}}\right) \max_{u,v} |a_{uv}|.$$

It can be satisfied by factors where the entries of L from double steps are bounded above by $1/(2\widehat{\alpha}^2)$. The difference between this bound and the bound we advocate, $1/\widehat{\alpha}$, is not important when $0.5 \leq \widehat{\alpha} \leq \alpha \approx .6404\ldots$, the usual range for dense algorithms. Sparsity changes the picture because noticeably smaller values of $\widehat{\alpha}$ are often used. For example, Duff and Reid [18] use 0.001 as a default value for $\widehat{\alpha}$. We contend that entries in L as large as $1/(2\widehat{\alpha}^2) \approx 10^6$ are dangerous. While Duff and Reid bound L in a double step by $1/\widehat{\alpha}$, the alternative two parameter version of the Duff–Reid algorithm by Liu [28] has only the weaker $1/(2\widehat{\alpha}^2)$ bound on L. This latter scheme we consider in hindsight to be a mistake.

3.5. An exhaustive pivoting strategy for sparse symmetric indefinite matrices. Our analysis of the sparse problem has thus far revolved solely around the issue of testing when a given (block) pivot is acceptable. The performance differences between the dense bounded Bunch–Kaufman and fast Bunch–Parlett algorithms demonstrate the effect of the order in which we test potential pivots. This ordering issue is of far greater importance in the sparse case. We propose here a new ordering of potential pivots that addresses two paramount issues: sparsity preservation and computational speed.

Our standard ordering is a slightly refined version of the search procedure in Duff and Reid's MA27 code [16]. Each multifrontal elimination step is a block elimination step, in which columns $i_1 < i_2 < \cdots < i_u$ are candidates for elimination due solely to their sparsity structure. A second set of columns, with indices $i_{u+1} < i_{u+2} < \cdots < i_v$, are also candidates for elimination because their numerical entries prevented their being eliminated at earlier factorization steps. Arrange these v columns in a queue,

```
initialize queue to $i_1 < i_2 < \cdots < i_u; i_{u+1} < i_{u+2} < \cdots < i_v$
repeat
    $pivot\_found$ = false
    repeat
        $j$ = index at front of queue
        $q_j$ = index of an off-diagonal entry $a_{q_j j}$ of largest
            magnitude $\gamma_j$ in column $j$ of $A_{11}$
        remove $j$ from queue
        if $\gamma_j = 0$ then
            nothing necessary for the $j$th column
        else if $|a_{jj}| \geq \widehat{\alpha}\gamma_j$ then
            $pivot\_found$ = true
            use $a_{jj}$ as a $1 \times 1$ pivot
        else
        if $D_{j,q_j} = \begin{bmatrix} a_{jj} & a_{q_j j} \\ a_{q_j j} & a_{q_j q_j} \end{bmatrix}$ is acceptable as $2 \times 2$ pivot then
            $pivot\_found$ = true
            remove $q_j$ from middle of queue
            use $D_{j,q_j}$ as a $2 \times 2$ pivot
        else
            add $j$ to rear of queue
        end if
    until
        $pivot\_found$ or
        all columns in queue tested since last successful pivot step
until queue empty or not $pivot\_found$
```

FIG. 3.4. *Standard pivot ordering.*

in the order $i_1, i_2, \ldots, i_v$. An outline of the standard search procedure is given in Figure 3.4.

A major complication in the sparse case is that an acceptable pivot need not exist. This standard procedure searches two possible pivots in any given column, the 1×1 diagonal pivot and the 2×2 pivot that uses the largest off-diagonal entry in the first block row of the column, to which either of the Duff–Reid tests or Liu's condition can be applied. All m columns are postponed if no pivot is found from these $2m - 1$ candidates, where A_{11} has m columns.

Failing to eliminate a column in the partial factorization creates more fill in L. The standard strategy evaluates only $m - 1$ of the $m(m-1)/2$ possible 2×2 pivots. Each test requires knowledge of the largest entry in at least one column of A. Often the length of each column is significantly greater than m. In any case in which no pivot is found, all m column maximums will have been found at least once and used on average twice.[8] We propose a different approach, to reuse the maximums. As we will see, we can check *all* possible 2×2 pivots with only $m(m-1)/2$ additional scalar operations. While we cannot prove that the work saved in later steps is of

[8] The Duff–Reid MA27 code avoids some double searches by first evaluating the 2×2 test with $\gamma_{q_j} = 0$ and only truly evaluating γ_{q_j} if this partial test is successful.

larger complexity, we have found it generally worthwhile to search exhaustively for 2×2 pivots.

Reusing the work in the column searches saves work and allows us to emphasize faster inner loops in the factorization. Block operations with large block size, as used by LAPACK, are generally not feasible in the sparse case. Our sights are much lower—2×2 pivots execute more operations faster than 1×1 pivots. Because our acceptance test for 2×2 pivots is independent of all 1×1 conditions, we can choose a pivot test order that favors 2×2 pivots. The heuristic procedure that works best in our experiments is not entirely straightforward. Figure 3.5 is a schematic representation in which the entries denote the order in which the tests are made. Diagonal entries correspond to a 1×1 pivot test with the corresponding diagonal entry in A_{11}. Off-diagonal entries represent a 2×2 pivot test with the corresponding off-diagonal entry in A_{11} and the appropriate diagonal entries.

$$\left[\begin{array}{rrrrr|rrr}
11 & & & & & & & \\
1 & 12 & & & & & & \\
2 & 3 & 13 & & & & & \\
4 & 5 & 6 & 14 & & & & \\
7 & 8 & 9 & 10 & 15 & & & \\
\cline{1-5}
16 & 17 & 18 & 19 & 20 & 21 & & \\
22 & 23 & 24 & 25 & 26 & 27 & 28 & \\
\dots & \dots & \dots & \dots & \dots & \dots & \dots & \ddots
\end{array}\right]$$

FIG. 3.5. *Schematic of exhaustive pivot search order* ($m_{2\times 2} = 5$).

The pattern in Figure 3.5 reuses each γ_i as often and as soon as possible. It is biased toward 2×2 pivots in two different ways. As we compute each new column maximum γ_j we immediately check all possible 2×2 pivots with entries a_{ij} for which we have already available the other column maximum γ_i needed for the test. We also refuse to check or accept any 1×1 pivot until we have examined (and failed) on all 2×2 pivots drawn pairwise from the leading $m_{2\times 2}$ columns, where $m_{2\times 2}$ is a heuristic parameter. We recommend $m_{2\times 2} = 5$.

The algorithmic details of our revised pivoting strategy are presented in Figure 3.6. Our preference for 2×2 pivots is primarily to emphasize speed of the numeric kernels. Although our strategy may also find acceptable pivots where the standard ordering finds none, we shall see in section 3.8 that the differences in fill between the two approaches is small. That the exhaustive pivot strategy usually leads to faster factorizations and solves is due to its greater use of 2×2 pivots. In contrast, Duff and Reid [14, 18] stress choosing special 2×2 pivots to take good advantage of the unusual sparsity structure in symmetric matrices with large zero diagonal blocks. Their scheme can be quite effective in reducing fill for such problems.

The pivot selection scheme in Figure 3.6 resulted from extensive experimentation. A number of other possible orderings for pivot selection, in particular including ordering the columns by size of diagonal entry, proved less effective because of large overhead. In addition to the bias toward 2×2 pivots, the one other important feature in Figure 3.6 is the initial skip over columns that were previously postponed. This is included in our standard ordering, where it is also effective.

```
initialize queue to $i_1 < i_2 < \cdots < i_u; i_{u+1} < i_{u+2} < \cdots < i_v$
repeat
    let $j_1, j_2, \ldots, j_k$ denote the ordering of the
    columns now present in the queue
    $t = 1$; $pivot\_found$ = false
    repeat
        $i = j_t$; compute $\gamma_i$; remove $j_t$ from queue
        if $\gamma_i = 0$ then
            nothing necessary for column $j_t$; $pivot\_found$ = true
        else
            $p = 1$
            while $p < i$ and not $pivot\_found$ do
                if $\max\{ |a_{ii}|\gamma_{j_p} + |a_{ij_p}|\gamma_i,$
                        $|a_{j_pj_p}|\gamma_i + |a_{ij_p}|\gamma_{j_p}\} \le \frac{|a_{j_pj_p}a_{ii} - a_{ij_p}^2|}{\widehat{\alpha}}$ then
                    use $\begin{bmatrix} a_{j_pj_p} & a_{ij_p} \\ a_{ij_p} & a_{ii} \end{bmatrix}$ as a $2 \times 2$ pivot
                    remove $j_p$ from queue; $pivot\_found$ = true
                else
                    $p = p + 1$
                end if
            end while
            if not $pivot\_found$ and $t = m_{2\times 2}$ then
                $p = 1$
                while $p < t$ and not $pivot\_found$ do
                    if $|a_{j_pj_p}| \ge \widehat{\alpha}\gamma_{j_p}$ then
                      use $a_{j_pj_p}$ as a $1 \times 1$ pivot
                      remove $j_p$ from queue; $pivot\_found$ = true
                    end if
                end while
            end if
            if not $pivot\_found$ and $t \ge m_{2\times 2}$ and $|a_{ii}| \ge \widehat{\alpha}\gamma_i$ then
                use $a_{ii}$ as a $1 \times 1$ pivot
            else
                add $i$ to rear of queue; $t = t + 1$
            end if
        end if
    until
        $pivot\_found$ or
        all columns in queue tested since last successful pivot step
until  queue is empty or not $pivot\_found$
```

FIG. 3.6. *Exhaustive search explicit bounding sparse pivot algorithm.*

3.6. Larger pivot blocks. The success of the algorithm in the previous section in increasing speed by emphasizing 2 × 2 pivots leads naturally to the idea of using other, larger block operations to gain speed. However, most of the dense partial factorizations in the sparse factorization are too small to make use of the left-looking blocking strategy used in LAPACK, which requires blocks of large order. In this section we generalize this scheme to use block pivots of rather small order, say, $s \le 5$, in an explicit right-looking factorization.

We partition the permuted reduced matrix $A^{(k-1)}$ as

$$A^{(k-1)} = \left[\begin{array}{c|c} D_k & B_k^T \\ \hline B_k & C_k \end{array}\right]$$

and compute the Schur complement $C_k - B_k D_k^{-1} B_k^T$ as a block operation. We implement this symmetric rank s modification directly, using "unroll and jam" techniques as in [13]. This requires separate code for each value of s allowed, but it allows much of the data reuse of BLAS3 operations, with less overhead. On dense matrices our block codes run only slightly slower on RISC computers than LAPACK when we take s to be three or four.

The tests we developed in section 3.3 ensure that the entries of L resulting from a 2×2 pivot are bounded by $1/\widehat{\alpha}$. It is straightforward to generalize these tests to block pivots of larger order. From the permuted reduced matrix $A^{(k-1)}$ take the leading principal minor of order s for use as an $s \times s$ block pivot D_k. If D_k^{-1} is computed by some stable algorithm, the computable bound

$$|D_k^{-1}| \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_s \end{bmatrix} \le \begin{bmatrix} 1/\widehat{\alpha} \\ 1/\widehat{\alpha} \\ \vdots \\ 1/\widehat{\alpha} \end{bmatrix}$$

ensures that $|l_{ij}| \le 1/\widehat{\alpha}$. We require that D_k satisfy this condition. Then by the inequality

$$\begin{aligned} |a_{ij}^{(k+s-1)}| &\le |a_{ij}| + \begin{bmatrix} |l_{i1}| & |l_{i2}| & \cdots & |l_{is}| \end{bmatrix} \begin{bmatrix} |a_{1j}| \\ |a_{2j}| \\ \vdots \\ |a_{sj}| \end{bmatrix} \\ &\le |a_{ij}| + \frac{s}{\widehat{\alpha}} \max_{u,v} |a_{uv}| \le \left(1 + \frac{s}{\widehat{\alpha}}\right) \max_{u,v} |a_{uv}|, \end{aligned}$$

we obtain a bound of $(1 + (s/\widehat{\alpha}))$ on growth in the reduced matrix. As in the 2×2 case, $s \times s$ linear systems must be solved using some normwise backward stable algorithm. Our code uses a QR factorization of D_k. Together with the bound on L, this suffices to show backward stability of the factorization for symmetric block pivots of any size.

Our blocks are much smaller than those in LAPACK for several reasons. Code for the explicitly unrolled operations is complex. Most importantly, the cost of finding numerically acceptable block pivots is considerable. It is unlikely that the leading $s \times s$ principal minor will be acceptable as a block pivot; we must expect to test several candidates. The fully assembled block A_{11} has order n_1, so it contains $\binom{n_1}{s}$ possible symmetric block pivots of order s. The exhaustive search strategy we used to find 2×2 pivots in section 3.5 is impractical for s much greater than two.

To keep the cost manageable, we restrict ourselves to small blocksizes and use heuristics to order the tests of possible pivots. We investigated several different strategies, which we will document more fully in a separate paper. Here we present only one strategy, one in the same spirit as the algorithm in Figure 3.6. We select M, the $s \times s$ principal minor of A_{11} that contains the s largest diagonal entries. We look for block pivots drawn from this submatrix, beginning with M itself as a $s \times s$ block pivot. If it is unacceptable, we systematically test each of the s possible $(s-1) \times (s-1)$ principal minors of M, terminating if we find an acceptable block pivot. The search continues when necessary by testing smaller and smaller principal minors of M. If we fail to find even an acceptable 1×1 pivot from M, we resort to the exhaustive search of section 3.5, noting that all possible symmetric pivots from M have already been rejected.

In section 3.8 we describe our preliminary numerical experience with this block algorithm. In closing, we note that Jones and Patrick [26] also developed a less general block pivot algorithm using a extension of Liu's Bunch–Kaufman algorithm to bound growth in the reduced matrix. They reported unsatisfactory results when they allowed s to grow. The explanation is simple—they allowed entries in L as large as $(1/\widehat{\alpha})^s$ and growth in $|\delta_{ij}|$ as large as $(1+(1/\widehat{\alpha}))^s$. Our algorithm bounds L more tightly and solves the problem in a general way.

3.7. Another option—a sparse LTL^T factorization. The Aasen/Parlett–Reid LTL^T factorization is a potential alternative to the sparse LDL^T factorization, yet the difficulties of creating a sparse Aasen-like algorithm are daunting. Here we briefly explore some of the issues one would face in developing a sparse variant of Aasen's algorithm.

Sparse LDL^T factorizations try to preserve the desirable properties of sparse Cholesky factorizations, in particular the efficient static data structures. The multifrontal approach is the basis for all currently successful codes. Here the key ingredient is that pivoting is allowed only within the diagonal blocks defined by (supernodes of) the frontal assembly tree, the A_{11} blocks in the preceding sections. Pivoting requires dynamic data structures, but this restriction on the choice of pivots makes the changes in data structures much more manageable than would be required for completely general pivot schemes.

The requirements for data structures for a sparse LTL^T factorization are similar to those of a sparse LDL^T factorization. We assume therefore that an efficient sparse LTL^T factorization would be done using a block structure very similar to that of a multifrontal factorization. We see three major obstacles to a successful sparse multifrontal-like LTL^T factorization: computing the product (TL^T) efficiently when L is sparse, understanding and storing the different fill-in that results from the different form of elimination used, and modifying the basic algorithm to meet the restriction on pivoting.

The first issue can be dealt with by using the block structure in an outer product manner. Consider applying q elimination steps simultaneously with a block version of the Parlett–Reid algorithm. The scalar Parlett–Reid algorithm requires a rank 2 modification at each elimination step, only after some analysis of what appears to be a rank 3 modification. Aasen succeeded in reducing the cost to that of a rank 1 modification. We can show that, after computing q steps of the factorization, the reduced matrix can be updated explicitly by a rank $q+2$ modification. We have not tried, by analogy with the point case, to reduce this to a rank $q+1$ or rank q modification. We would expect this block version to be more efficient than applying

q steps of Aasen's algorithm in sequence, and might also prove useful in a dense partitioned LTL^T factorization. Compared to an LDL^T factorization, the additional rank 1 or rank 2 work is a penalty but not an insurmountable obstacle.

Fill can be analyzed through a simple variant of the standard elimination process for sparse Cholesky. A straightforward implementation of such a sparse Aasen–Parlett–Reid scheme has a best case for fill that is generally worse than the best case for an LDL^T factorization. Let $\eta(L^c_{*j})$ denote the number of columns in the jth column of the Cholesky factor L^c. Then the best case for the LDL^T factorization is the same structure, which has $\sum_j \eta(L^c_{*j})$ nonzeros. A lower bound for the best case for a simple sparse LTL^T factorization is $\sum_j \eta(L^c_{*p(j)})$, where $p(j)$ is the index of the parent of node j in the elimination tree of the Cholesky factor. There appear to be ways to reduce the difference marginally but at a considerable cost in code complexity. Table 3.1 gives the results of a more careful computation of the increase in best case storage requirements for the LTL^T factorization, compared with the best case LDL^T factors, for the test problems used in this paper. Additional fill usually creates noticeably larger increases in work than in storage. Table 3.1 thus suggests that an LTL^T factorization will be slower than an LDL^T factorization.

Table 3.1
Storage increase, best case LTL^T relative to best case LDL^T.

Matrix	Order	# of frontal matrices	Storage increase
BCSSTK08	1074	756	50.6%
TRAJ06B	1665	618	33.1%
BCSSTK24	3562	407	6.2%
BCSSTK38	8032	1557	7.6%
BCSSTK35	30237	3903	6.0%

We would expect to use Aasen's algorithm within a frontal elimination step. Our compromise between sparsity and data structures is to allow a pivot only from within the A_{11} block. This may prevent a column from being eliminated; Aasen's algorithm must be modified for use as a partial factorization. The work done in Aasen's algorithm to compute the leading column of the reduced matrix may have to be discarded. This is the same issue we faced in the fast Bunch–Parlett and bounded Bunch–Kaufman algorithms. Here, again, sparsity prevents us from using a simple tool to eliminate this cost. A practical sparse code appears more likely to require that we avoid discarding this work as much as possible, which can be done by tracking for each column in the active matrix where it was last explicitly updated. But this represents additional complexity in coding and a likely degradation in performance.

These issues all make it likely that a multifrontal sparse LTL^T algorithm will perform less efficiently than good multifrontal LDL^T factorizations. However, the bottom line is likely to come from the differences in the actual pivots dictated by the numerical entries. The differences in pivoting choice are hard to predict, but the simple differences are not advantageous to the LTL^T scheme. The most obvious cases in which the LTL^T factorization can accept a column rejected as a 1×1 pivot in the LDL^T factorization would still be taken as 2×2 LDL^T pivots, whereas there are obvious cases in which the LTL^T scheme rejects perfectly good 1×1 pivots in the LDL^T case. For these reasons we did not implement a sparse LTL^T factorization and cannot directly answer how it would fare in comparison to our LDL^T factorizations.

3.8. Numerical experiments. We have argued for a sparse factorization that comes with strong guarantees for accuracy. We have described schemes which we believe reduce the costs of getting such factorizations. In this section we give what we believe is a general view of the numerical experience one can expect with such factorizations.

3.8.1. Codes. Our numerical experiments used two different computer codes in our experiments. We began with our workhorse Fortran 77 code from the BCSLIB-EXT software library [4]. This widely used commercial code is based on Liu's threshold Bunch–Kaufman algorithm. We developed a variant of this code that implements the algorithm of section 3.5, which we refer to as the exhaustive explicit pivoting code (EP). Larger block pivots were tested in a new code being developed in C. This code is efficient, but the results from these experiments should be considered as preliminary. We describe this as the explicit block pivoting code (EBP) in our tables.

Both codes allow two modes of factorization. In the indefinite mode the Fortran 77 code implements the EP algorithm, which favors the use of 2×2 pivots. For either 1×1 or 2×2 cases the low rank modification to the reduced matrix is made immediately. The inner loop applies the modification to each of two columns simultaneously, reducing the number of memory loads. The C code implements the EBP algorithm of section 3.6, which favors larger block pivots. This code uses an inner loop that performs a rank s modification to each of four columns of the reduced matrix simultaneously. The larger values of s and the simultaneous modifications of four columns reduce memory traffic even further than does our EP implementation.

Each of the codes also allows a mode for definite matrices, in which case pivoting is not required. Each column is treated as a 1×1 pivot, which are grouped together to form larger block pivots. In the Fortran 77 code each successive triple of columns is grouped as a block, and the resulting rank three modification is applied in an inner loop to two columns simultaneously. The C code takes the leading $s \times s$ principal minor as a block pivot and thereafter uses the same code as the indefinite case.

The user must determine the choice of mode in advance. Both codes can factor an indefinite matrix with the definite mode, producing an unpivoted LDL^T factorization with D diagonal. There are no stability guarantees for such a factorization. We include the definite mode primarily to avoid the cost of pivoting when the matrix is known to be definite or has special properties that guarantee stability of the factorization. We never advocate its use on general indefinite matrices, and we include it in this discussion only to be able to address the penalty associated with using the general codes instead of the definite codes on a matrix that happens to be definite.

We provide data to compare our EP and EBP algorithms, Liu's sparse threshold Bunch–Kaufman (LBK) algorithm, and the Duff and Reid strategy using the matrix magnitude test (DR). The latter two schemes are both implemented using the standard pivot search order (Figure 3.4). Of these, all but Liu's algorithm bound L. The major issues we address include the additional cost of and accuracy from obtaining a bounded L, described by comparisons of Liu's algorithm with the Duff–Reid algorithm and with our EP algorithm; the effect of choosing smaller values of $\widehat{\alpha}$ on efficiency and accuracy, for all algorithms; and the effect of choosing larger block sizes in our EBP algorithm.

The codes were run on a SPARC 20 under the Solaris operating system. The implementations of the EP, Duff–Reid, and LBK algorithms use the same Fortran 77 framework and were compiled by the `f77` compiler with `-O3` optimization. The block pivot code was compiled under the `gcc` complier with the `-O4` optimization.

Table 3.2
Test matrices.

TRAJ06B		A symmetric indefinite KKT matrix from a numerical optimization code. Liu's algorithm for the particular value $\widehat{\alpha} = .056$ (corresponding to $\tau = 0.1$) suffers substantial loss in accuracy on this matrix, which led to our investigation of pivoting strategies. This matrix is typical of KKT matrices in having large numbers of eigenvalues of both signs, which is due to its form $A = \left[\begin{smallmatrix} H & G^T \\ G & 0 \end{smallmatrix} \right]$. This matrix represents the type of matrix for which the approach in Duff–Reid [14, 18] may be particularly appropriate.
BCSSTK35		A symmetric indefinite structural engineering matrix which is presumed in its origin to be positive (semi)-definite; numerically semi-definiteness translates into very small, but negative, eigenvalues.
BCSSTK24		A symmetric positive definite structural engineering matrix
{BCSSTK08, {BCSSTK24, {BCSSTK38,	BCSSTM08}, BCSSTM24}, BCSSTM38}	Matrix pairs (mass and stiffness matrices) representing generalized eigenproblems from structural engineering vibration analysis

3.8.2. Test problems and statistics. We tested our algorithms on a number of matrices drawn from the Harwell–Boeing sparse matrix collection [15] or soon to be added to it. The matrices described here are intended to demonstrate a wide spectrum of problems. We present detailed data on the first three test matrices in Table 3.2.

We present statistics for each of these matrices in Tables 3.3–3.5. Each table includes four separate subtables: symbolic factorization statistics for a Cholesky factorization; numerical results for a nonpivoted LDL^T factorization; numerical results for the 2×2 pivoting codes (Liu, Duff–Reid, and EP); and numerical results for the block pivoting code (EBP). In the numerical subtables we give a common set of statistics: the CPU time required, the maximum relative error in solving several linear systems whose solution vectors are drawn from a uniform distribution on $[0, 1]$, the magnitude of the largest entry in the computed lower triangular factor, and the total number of pivot steps. The last of these demonstrates whether the bias toward block pivots is realized. (The second subtable is missing from TRAJ06B, which has zero diagonal entries and cannot be factored without pivoting.)

We do not recommend using the last two subtables to compare the relative speeds of the Fortran 77 and C codes. The implementation differences in data reuse and the use of several compilers make it impossible to do more than weakly account for the algorithmic differences.

We also present summary CPU times for three families of matrices drawn from symmetric generalized eigenproblems $K\Phi = M\Phi\Lambda$. In each family each matrix is of the form $K - \sigma M$, where the values used for σ represent different shifts as could be obtained from an eigenvalue code using the spectral transformation [21]. The three families of matrices all represent structural engineering eigenproblems, using the pairs of matrices {BCSSTK08, BCSSTM08}, {BCSSTK24, BCSSTM24}, and {BCSSTK38, BCSSTM38}.

3.8.3. Results. The fundamental question is, What is the cost of obtaining a factorization with L bounded tightly?

The first problem, TRAJ06B, illustrates why we have to pay this cost. Exactly zero entries on the diagonal of this matrix defeat a factorization without numerical pivoting. Pivoting without a good bound on L gives inferior accuracy. The second example, BCSSTK35, presents the other side of the coin—pivoting cannot overcome ill conditioning. This matrix is numerically singular, and all of the algorithms provide similar and poor accuracy for this problem.

The remaining problems demonstrate a common theme in this area: very often less stable algorithms appear to perform numerically just as well as more reliable algorithms. Counterexamples are uncommon. The structural engineering eigenproblem matrices in Tables 3.6–3.8, and corresponding shifted versions of the pencil BCSST{K,M}35, produced no large entries in L, even when we used unpivoted factorizations.[9]

The simplest cost comparison of any of the pivoted factorizations with the unpivoted factorizations shows a considerable increase in time and storage for computing a factorization with stability guarantees. The third example, BCSSTK24, illustrates why. The Cholesky factorization is stable but gives only a $\sqrt{\|A\|}$ bound on $\|l_{ij}\|$. The Duff–Reid, EP, and EBP algorithms must pivot to maintain the required bounds on L even in this positive definite case.

Comparing the pivoted factorizations, we see in general a noticeable drop in cost between $\widehat{\alpha} = 0.1$ and $\widehat{\alpha} = 0.01$. For such values of $\widehat{\alpha}$, the Duff–Reid and EP algorithms can be faster than Liu's algorithm. Still smaller values of $\widehat{\alpha}$ yielded only marginal improvements in speed. As a general rule, the EP algorithm has the advantage over the Duff–Reid algorithm. This seems to be due primarily to greater use of 2×2 pivots. The exceptional case is TRAJ06B, on which EP is unable to choose 2×2 pivots as regularly. Exhaustive search does not perform well on this problem because of unusual row scaling.

We argue against speed at the cost of accuracy. Our problems fell into two categories. Accuracy for the structural engineering eigenproblems seemed quite insensitive to the choice of $\widehat{\alpha}$. In contrast TRAJ06B shows a general degradation in accuracy for smaller values of $\widehat{\alpha}$. We recommend taking $\widehat{\alpha} = 0.01$ as an effective compromise between speed and stability.

Our preliminary results on block pivots agree with the results on the standard codes: that choosing 2×2 pivots in preference to 1×1 pivots provides a speedup. We see a further, but smaller, gain in going to 3×3 pivots. In general, 4×4 pivots did not give more speed. There are several reasons that limit the effectiveness of larger pivot blocks. The sparsity of the matrices is a limiting factor in that the small sizes of many of the frontal matrices prevent choosing many large pivots and usually leave a small pivot block as the only possible choice at the last elimination step. Small frontal matrices also have greater overhead, relative to numeric work, than larger frontal matrices. The computer architecture and compilers are also limiting. The effectiveness of these "unroll and jam" techniques depends heavily on the availability of substantial numbers of registers and of effective optimizing compilers. We have observed substantially better results on machines like the CRI C-90 computer than we obtained on the Sun SPARCstation 20.

Tables 3.6–3.8 address how the cost of the indefinite factorization varies with the distribution of eigenvalues of the factored matrix. Two of the three families of shifted matrices show an increase in cost as the number of negative eigenvalues increases, but then show a marked decrease in difficulty when the distribution of negative and positive eigenvalues is more balanced. However, the third family of shifted matrices shows very little variation throughout.

[9] Details for the three eigensystem families can be obtained from the authors.

Table 3.3
traj06b: *Trajectory optimization, inertia* = $\langle 794, 0, 871\rangle$.

Symbolic factorization	
Matrix order	1665
# of frontal matrices	618
Nonzeros in L	25,950
Factor operations	0.57M

	2 × 2 pivoting sparse codes					
	$\widehat{\alpha} = 0.1$			$\widehat{\alpha} = 0.01$		
	LBK	DR	EP	LBK	DR	EP
CPU	0.32	0.47	0.81	0.29	0.18	0.20
$\|e\|_\infty/\|x\|_\infty$	2e-6	2e-10	2e-10	1e-10	3e-11	2e-11
$\max_{i,j} \|L_{ij}\|$	4.6e7	10	10	6.5e3	100	99
Rel. nz(L)	1.83	2.24	2.24	1.51	1.52	1.52
Rel. ops	3.37	6.42	6.47	2.12	2.17	2.25

	Block pivot code (EBP)					
	$\widehat{\alpha} = 0.1$			$\widehat{\alpha} = 0.01$		
	$s = 2$	$s = 3$	$s = 4$	$s = 2$	$s = 3$	$s = 4$
CPU	0.80	0.81	0.89	0.39	0.40	0.47
$\|e\|_\infty/\|x\|_\infty$	8e-13	2e-12	1e-12	2e-11	1e-11	6e-12
$\max_{i,j} \|L_{ij}\|$	10	10	10	98	100	98
Rel. nz(L)	2.17	2.17	2.17	1.50	1.50	1.50
Rel. ops	6.89	6.89	6.89	2.15	2.14	2.14

Table 3.4
bcsstk35: *Automobile seat frame and body attachment, inertia* = $\langle 3, 0, 30234\rangle$.

Symbolic factorization	
Matrix order	30237
# of frontal matrices	3903
Nonzeros in L	2.66M
Factor operations	381.1M

	LDL^T factorization, no pivoting				
	BCSLIB-EXT	Block pivot code			
	D diagonal	$s = 1$	$s = 2$	$s = 3$	$s = 4$
CPU	23.4	31.8	24.0	22.7	23.5
$\|e\|_\infty/\|x\|_\infty$	2e-5	3e-6	7e-5	7e-5	2e-4
$\max_{i,j} \|L_{ij}\|$	338	338	338	474	338

	2 × 2 pivoting sparse codes					
	$\widehat{\alpha} = 0.1$			$\widehat{\alpha} = 0.01$		
	LBK	DR	EP	LBK	DR	EP
CPU	40.8	43.8	31.6	40.1	41.3	28.9
$\|e\|_\infty/\|x\|_\infty$	2e-5	2e-5	3e-5	2e-5	2e-5	2e-5
$\max_{i,j} \|L_{ij}\|$	338	10	10	338	99	99
Rel. nz(L)	1.00	1.02	1.02	1.00	1.00	1.00
Rel. ops	1.01	1.04	1.07	1.00	1.01	1.03

	Block pivot code (EBP)					
	$\widehat{\alpha} = 0.1$			$\widehat{\alpha} = 0.01$		
	$s = 2$	$s = 3$	$s = 4$	$s = 2$	$s = 3$	$s = 4$
CPU	26.5	25.7	26.3	25.0	24.1	24.6
$\|e\|_\infty/\|x\|_\infty$	1e-5	2e-6	5e-6	2e-6	7e-6	5e-6
$\max_{i,j} \|L_{ij}\|$	10	10	10	99	99	99
Rel. nz(L)	1.03	1.03	1.03	1.00	1.00	1.00
Rel. ops	1.07	1.07	1.07	1.00	1.00	1.00

Table 3.5
BCSSTK24: *Winter sports arena, inertia* = $\langle 0, 0, 3562 \rangle$.

Symbolic factorization	
Matrix order	3562
# of frontal matrices	407
Nonzeros in L	284,334
Factor operations	36.7M

	LDL^T factorization, no pivoting				
	BCSLIB-EXT	Block pivot code			
	D diagonal	$s = 1$	$s = 2$	$s = 3$	$s = 4$
CPU	2.3	2.9	2.3	2.1	2.2
$\\|e\\|_\infty / \\|x\\|_\infty$	2e-8	8e-9	1e-8	3e-8	3e-8
$\max_{i,j} \|L_{ij}\|$	367	367	376	489	383

	2×2 pivoting sparse codes					
	$\widehat{\alpha} = 0.1$			$\widehat{\alpha} = 0.01$		
	LBK	DR	EP	LBK	DR	EP
CPU	4.4	8.9	7.0	3.8	4.5	3.3
$\\|e\\|_\infty / \\|x\\|_\infty$	2e-8	2e-8	1e-8	2e-8	2e-8	1e-8
$\max_{i,j} \|L_{ij}\|$	453	10	10	367	99	98
Rel. nz(L)	1.05	1.48	1.50	1.00	1.04	1.05
Rel. ops	1.11	2.28	2.32	1.00	1.12	1.16

	Block pivot code (EBP)					
	$\widehat{\alpha} = 0.1$			$\widehat{\alpha} = 0.01$		
	$s = 2$	$s = 3$	$s = 4$	$s = 2$	$s = 3$	$s = 4$
CPU	7.2	6.9	6.9	2.7	2.6	2.6
$\\|e\\|_\infty / \\|x\\|_\infty$	6e-9	1e-8	8e-9	2e-8	4e-9	2e-8
$\max_{i,j} \|L_{ij}\|$	10	10	10	100	99	99
Rel. nz(L)	1.44	1.44	1.44	1.05	1.05	1.05
Rel. ops	2.09	2.10	2.09	1.14	1.13	1.13

Table 3.6
BCSST{K,M}08: *TV studio,* $K - \sigma M$.

Symbolic factorization	
Matrix order	1074
# of frontal matrices	756
Nonzeros in L	29,973
Factor operations	1.94M

		2×2 pivoting sparse codes					
	# of neg.	$\widehat{\alpha} = 0.1$			$\widehat{\alpha} = 0.01$		
σ	eigenvals.	LBK	DR	EP	LBK	DR	EP
0	0	0.31	0.41	0.34	0.25	0.26	0.22
140	18	0.47	0.82	0.65	0.25	0.26	0.23
196	74	0.67	1.28	0.97	0.25	0.28	0.24
220	100	0.69	1.32	1.00	0.25	0.29	0.25
10000	618	0.32	0.35	0.29	0.25	0.26	0.23

		Block pivot code (EBP)					
	# of neg.	$\widehat{\alpha} = 0.1$			$\widehat{\alpha} = 0.01$		
σ	eigenvals.	$s = 2$	$s = 3$	$s = 4$	$s = 2$	$s = 3$	$s = 4$
0	0	0.81	0.79	0.83	0.44	0.43	0.44
140	18	1.76	1.85	1.84	0.45	0.45	0.45
196	74	2.54	2.59	2.64	0.47	0.48	0.47
220	100	2.54	2.60	2.62	0.51	0.49	0.50
10000	618	0.62	0.64	0.65	0.46	0.50	0.46

Table 3.7
bcsst{k,m}24: *Winter sports arena: K - σ M.*

Symbolic factorization	
Matrix order	3,562
# of frontal matrices	407
Nonzeros in L	284,334
Factor operations	36.7M

		2 × 2 pivoting sparse codes					
	# of neg.	$\widehat{\alpha} = 0.1$			$\widehat{\alpha} = 0.01$		
σ	eigenvals.	LBK	DR	EP	LBK	DR	EP
4.0	0	4.5	9.0	7.0	3.8	4.4	3.3
60.0	20	4.6	9.5	7.4	4.0	4.5	3.4
200.0	63	4.5	10.8	8.3	3.8	5.1	3.9
350.0	98	4.3	11.1	8.6	3.8	5.5	4.1
100000.0	1782	5.7	5.8	4.4	4.1	4.5	3.4

		Block pivot code (EBP)					
	# of neg.	$\widehat{\alpha} = 0.1$			$\widehat{\alpha} = 0.01$		
σ	eigenvals.	$s = 2$	$s = 3$	$s = 4$	$s = 2$	$s = 3$	$s = 4$
4.0	0	7.3	9.1	7.0	2.8	2.6	2.7
60.0	20	7.6	7.2	10.0	2.9	2.7	2.8
200.0	63	8.4	7.9	8.0	3.3	3.2	3.3
350.0	98	8.8	8.6	8.6	3.3	3.4	3.3
100000.0	1782	4.4	4.3	4.3	3.0	3.0	2.9

Table 3.8
bcsst{k,m}38: *Airplane engine component, $K - \sigma M$.*

Symbolic factorization	
Matrix order	8,032
# of frontal matrices	1,557
Nonzeros in L	708,593
Factor operations	111.4M

		2 × 2 pivoting sparse codes					
	# of neg.	$\widehat{\alpha} = 0.1$			$\widehat{\alpha} = 0.01$		
σ	eigenvals.	LBK	DR	EP	LBK	DR	EP
0	0	12.0	14.5	10.6	11.8	12.6	9.3
1.6e6	46	11.9	15.6	11.3	11.8	13.3	9.6
3.0e6	78	12.0	15.4	11.0	11.8	13.3	9.7
4.0e6	101	12.0	15.4	11.0	11.8	13.4	9.7
1.0e10	2262	11.6	16.8	12.3	11.8	13.3	9.7

		Block pivot code (EBP)					
	# of neg.	$\widehat{\alpha} = 0.1$			$\widehat{\alpha} = 0.01$		
σ	eigenvals.	$s = 2$	$s = 3$	$s = 4$	$s = 2$	$s = 3$	$s = 4$
0	0	9.5	9.3	9.3	7.7	7.5	7.7
1.6e6	46	10.2	9.9	9.7	8.0	7.6	7.8
3.0e6	78	10.2	10.4	9.8	7.7	7.7	7.9
4.0e6	101	10.2	10.2	9.8	7.8	7.6	7.8
1.0e10	2262	10.5	10.3	10.1	8.0	7.7	7.9

4. Summary remarks. We have presented algorithms that provide more accurate solutions to symmetric indefinite linear systems by bounding the entries in the factor L. For dense matrices, the cost of bounding L and, thereby, of higher accuracy is minimal. For sparse matrices, our strategies to find 2 × 2 and larger block pivots results in codes that are faster and more accurate than their predecessors.

Appendix A. The hidden dangers of scaling—Instability in LAPACK. LAPACK uses BLAS2 and BLAS3 kernels wherever possible. Kaufman [27] observes that this may not always be appropriate for reasons of efficiency. We uncovered a special case where the use of BLAS2 kernels produced instability.

The LAPACK codes include a conventional unblocked Bunch–Kaufman code, which is used to process the final block in a blocked factorization, for problems of order smaller than b and for any problem where the user has not provided the temporary storage needed for blocking. The LAPACK unblocked implementation in release 2.0 and earlier used a clever trick with rank 2 modifications to circumvent the lack of an appropriate rank 2 kernel in BLAS2. This rearrangement is an issue because the price is instability when entries in L are large.

The reduction step for a 2×2 pivot is

$$A^{(k+1)} = A^{(k-1)} - A_{k,k+1} D_{k,k+1}^{-1} A_{k,k+1}^T = A^{(k-1)} - A_{k,k+1} L_{k,k+1}^T,$$

where $A_{k,k+1}$ is an $(n-k-1) \times 2$ matrix from columns k and $k+1$ of A and $D_{k,k+1}$ is the 2×2 pivot block. This is a symmetric rank 2 modification, but only the latter form is present in BLAS2, which does not recognize the implicit symmetry. The LAPACK implementation retains use of BLAS2 by replacing the symmetric rank 2 modification with a sum of two symmetric rank 1 matrices, using the eigendecomposition $Q \Lambda Q^T$ of $D_{k,k+1}$. The matrix $Z = A_{k,k+1} Q$ is formed explicitly by Givens rotations, after which

$$A^{(k+1)} = A^{(k-1)} - (A_{k,k+1} Q) \Lambda^{-1} (Q^T A_{k,k+1}^T) = A^{(k-1)} - Z \Lambda^{-1} Z^T$$

is computed as two independent rank 1 modifications. Finally, $L_{k,k+1}$ is computed as $L_{k,k+1} = \left(Z \Lambda^{-1} \right) Q^T$. The hope is that sufficiently higher performance on the $\mathcal{O}(n^2)$ work of the symmetric rank 1 modifications will more than offset the extra $\mathcal{O}(n)$ work of the rotations.

This leads to disaster in the factorization. The worst results for our first test suite, the scaled matrices of order 50 from section 2.7.1, are given in Table A.1, using the measures from section 2.7.1.

Table A.1
Instability of the unblocked LAPACK Bunch–Kaufman algorithm.

Decomposition error	7.3×10^4
Error in inverse	3.0×10^5
Residual	4.6×10^5
Solution error	5.7×10^1

The failure comes from allowing very large entries in L. Consider one of our scaled random matrices, with $m = 1$, $\sigma_1 = 10^{-2t_2-1}, \sigma_2 = \sigma_3 = 10^{-t_2}$, with the extra conditions that $a_{11} = 10^{-2t_2-1}, a_{21} = a_{22} = 10^{-t_2}$. The first reduction step is a 2×2 pivot with the leading 2×2 block. The eigenvalues of the diagonal block D are approximately $10^{-t_2} \left(1 \pm \sqrt{5}\right)/2$. As t_2 increases, the eigenvector matrix Q approaches

$$\begin{bmatrix} 0.8507 & 0.5257 \\ -0.5257 & 0.8507 \end{bmatrix}.$$

The new reduced matrix is calculated as

$$a_{ij} = a_{ij} - \frac{z_{i1} z_{j1}}{\lambda_1} - \frac{z_{i2} z_{j2}}{\lambda_2}.$$

The largest entries in the original matrix, in Z and in the reduced matrix, are all $\mathcal{O}(1)$. But the reduced matrix is computed as $\mathcal{O}(1) - \mathcal{O}(10^{t_2}) - \mathcal{O}(10^{t_2})$. The two $\mathcal{O}(10^{t_2})$ terms must cancel massively to achieve an $\mathcal{O}(1)$ result. The result is inaccurate entries in the reduced matrix. The original Bunch–Kaufman algorithm stands in sharp contrast—the analysis of section 2.2 shows that each term in the rank 2 modification is bounded by $\mu(= \mathcal{O}(1))$ through the careful choice of conditions imposed by the failures of cases 1, 3, and especially 2. Were the new reduced matrix to be calculated as

$$A^{(k+1)} = A^{(k-1)} - (L_{k,k+1}Q)\Lambda(Q^T L_{k,k+1}^T),$$

a similar cancelation would occur because entries of $L_{k,k+1}Q$ will be $\mathcal{O}(10^{t_2})$. The instability in the LAPACK code is fixed easily, by performing the rank 2 modification as in LINPACK, as $A^{(k-1)} - A_{k,k+1}L_{k,k+1}^T$.

The LAPACK scheme is equivalent to solving the 2×2 systems with the singular value decomposition. We would generally expect this to be a very stable approach, but here it is a mistake, as is substituting Gaussian elimination with complete pivoting for Cramer's rule. Again a very stable method causes similar instability of the Bunch–Kaufman factorization.

Except in this section, our comparisons of numerical accuracy and timing consistently use rank 2 modifications $A - A_{k,k+1}L_{k,k+1}^T$ for both the slightly rewritten LAPACK code and our new algorithms. Our rank 2 modifications are done in a separate BLAS-like subroutine to facilitate tuning the code for higher performance while requiring no extra vector storage. As Kaufman [27] argues, this "generalized" symmetric rank 2 modification should be part of the BLAS2 specifications; we agree. Its vendor-supported performance should exceed that of the rank 1 kernel, which would remove the argument for misusing the rank 1 kernel. The bounded Bunch–Kaufman and the fast Bunch–Parlett algorithms are not sensitive to the LAPACK rearrangement of the rank 2 modification, inasmuch as both L and $\kappa(D_{k,k+1})$ are bounded near unity. Nonetheless, the rank 2 kernel missing from BLAS2 really is the appropriate form for these algorithms as well.

We also encountered a similar problem with the LAPACK matrix inversion routine for symmetric indefinite problems. The LAPACK Bunch–Kaufman factorization computes very poor explicit inverses for many of our tests when entries in L are large. The LAPACK explicit inversion routine uses a recursive algorithm, not an explicit block solution of $AX = I$. The factorization from the bounded Bunch–Kaufman algorithm provides accuracy similar to the results of an explicit solution. We would expect the LAPACK Bunch–Kaufman factorization to give poorer results on ill-conditioned problems because iterative refinement is not used in this context. The recursive results are much worse than that. The worst results from the LAPACK Bunch–Kaufman factorization are 10^5 times larger than the theoretical bound; on such problems the results from the bounded Bunch–Kaufman algorithm are as much as 11 orders of magnitude better than those from the LAPACK Bunch–Kaufman factorization. We did not analyze this problem further because it was not of direct interest to our applications. We would expect to find that these poor results in LAPACK's inversion routine are due to cancelation caused by large entries in L.

Appendix B. Backward stability of LDL^T factorizations with bounded L. Let the symmetric indefinite matrix A be factored as LDL^T, where D is comprised of 1×1 and 2×2 blocks and L is unit lower triangular. Suppose that $\max_{i,j} |l_{ij}| \leq c_l$ for a constant c_l. Assume that 2×2 systems in the factorization and solution phases are solved by a normwise backward method. Then the factorization and linear equation solution are normwise backward stable. Specifically,

$$\begin{array}{rcll} A &=& LDL^T + E, & \|E\| \le d(n)\epsilon\|A\| + \mathcal{O}(\epsilon^2), \\ (A+F)\hat{x} &=& b, & \|F\| \le d'(n)\epsilon\|A\| + \mathcal{O}(\epsilon^2). \end{array}$$

The proof is in the style and form of Demmel and Higham [11]. Details of constants and slowly growing functions of n are not given.

The first elimination step, whether a 1×1 or a 2×2 step, produces the partial factorization

$$A = \begin{bmatrix} I & 0 \\ \widehat{L}_{11} & I \end{bmatrix} \begin{bmatrix} \widehat{D}_{11} & 0 \\ 0 & \widehat{S} \end{bmatrix} \begin{bmatrix} I & \widehat{L}_{11}^T \\ 0 & I \end{bmatrix}.$$

The computed Schur complement $\widehat{S}$ satisfies $\widehat{S} = A_{22} - \widehat{L}_{11}A_{12} - \Delta S$. It follows that $\|\Delta S\| \le c_r\epsilon(\|A_{22}\| + \|\widehat{L}_{11}\|\,\|A_{12}\| + \mathcal{O}(\epsilon^2)$, where c_r is a small constant that depends on the number of operations in a rank 1 or a rank 2 modification. A componentwise view gives a bound $\|\widehat{S}\| \le (1 + f_{12}c_l + \mathcal{O}(\epsilon))\|A\|$ on growth in the reduced matrix due to a single step; the factor f_{12} is one for a 1×1 case and two for a 2×2 case. By induction the remaining elimination steps compute the factorization $\widehat{S} = \widehat{L}_S\widehat{D}_S\widehat{L}_S^T$, which satisfies

$$\widehat{L}_S\widehat{D}_S\widehat{L}_S^T = \widehat{S} + E_S,\ \ \|E_S\| \le d(\tilde{n})\epsilon(\|\widehat{S}\|) + \mathcal{O}(\epsilon^2) \le d(\tilde{n})(1 + f_{12}c_l)\epsilon(\|A\|) + \mathcal{O}(\epsilon^2),$$

where $\tilde{n}$ is $n-1$ or $n-2$.

Substituting the recursive result into the output of the first elimination step yields

$$A - \widehat{L}\widehat{D}\widehat{L}^T = A - \begin{bmatrix} I & 0 \\ \widehat{L}_{11} & \widehat{L}_S \end{bmatrix} \begin{bmatrix} \widehat{D}_{11} & 0 \\ 0 & \widehat{D}_S \end{bmatrix} \begin{bmatrix} I & \widehat{L}_{11}^T \\ 0 & \widehat{L}_S^T \end{bmatrix} = \begin{bmatrix} 0 & E_{21}^T \\ E_{21} & \Delta S + E_S \end{bmatrix}.$$

The term E_{21} represents the backward error in solving the linear system $D_{11}L_{21} = A_{21}$. Our hypothesis is that we can solve this system with small normwise backward error. That is,

$$(D_{11} + \Delta D_{11})L_{21} = A_{21}, \quad \text{with } \|\Delta D_{11}\| \le c_d\epsilon\|D_{11}\|,$$

and so

$$\|E_{21}\| \le \|\Delta D_{11}\|\,\|L\| \le c_L(n)c_d\epsilon\|D_{11}| = c_L(n)c_d\epsilon\|A_{11}|,$$

where $c_L(n)$ is a slowly growing function that bounds $\|L\|$. Combining all terms, it follows that

$$\begin{aligned} \|A - \widehat{L}\widehat{D}\widehat{L}^T\| &\le \epsilon\ \{c_L(n)c_d\|A_{11}\| + \|\Delta S\| + \|E_S\|\} + \mathcal{O}(\epsilon^2) \\ &\le c_L(n)\ \{c_d + (c_r + d(\tilde{n}))(1 + f_{12}c_l)\}\,\epsilon\|A\| + \mathcal{O}(\epsilon^2). \end{aligned}$$

The bound on F is obtained from the equation

$$b = \left(A + \Delta A + \Delta L \cdot D \cdot L^T + L \cdot \Delta D \cdot L^T + L \cdot D \cdot \Delta L^T\right)x.$$

Again substituting bounds for $\|L\|$ and $\|D\|$ yields an expression only involving $\|A\|$ and a small function of n.

Appendix C. Expected length of search for a local maximum off-diagonal entry. Let A be an $n \times n$ symmetric matrix, with entries drawn independently from a probability distribution. We want to estimate the work to find a local maximum off-diagonal entry, that is, an entry a_{ij} such that $|a_{ij}| \geq |a_{ik}|$ for $k \neq i$ and $|a_{ij}| \geq |a_{kj}|$ for $k \neq j$, using the search procedure described in section 2.4. How much work do we expect to perform in the search? Let the number of columns we search, s, be the measure of work.

The expected number of columns to be searched is

$$\mathcal{E}(s) = \sum_{k=1}^{n-1} k\mathcal{P}(s = k) = \sum_{k=0}^{n-2} \mathcal{P}(s > k).$$

The search stops as soon as a local maximum is found, so

$$\mathcal{P}(s > k) = \mathcal{P}(s > 0) \prod_{i=1}^{k} \mathcal{P}(s > i | s > i-1).$$

The procedure begins by searching two columns, so $\mathcal{P}(s > 0) = 1$ and $\mathcal{P}(s > 1) = 1$. For $i > 1$ the conditional probability $\mathcal{P}(s > i | s > i-1)$ is the probability that we find a new maximum in the ith row/column searched. That is, the maximum of the $n-i$ previously unchecked off-diagonal entries in this ith row/column is also the maximum of the $\sum_{k=1}^{i}(n-k) = in - i(i+1)/2$ distinct entries in all of the first i rows/columns searched. That is, $\mathcal{P}(s > i | s > i-1)$ is the probability that one of $n-i$ entries is the maximum of $in - i(i-1)/2$ entries, so

$$\mathcal{P}(s > i | s > i-1) = \frac{n-i}{in - i(i+1)/2} = \left(\frac{1}{i}\right)\left(\frac{n-i}{n-(i+1)/2}\right) < \frac{1}{i}.$$

It follows that

$$\mathcal{E}(s) < \sum_{k=0}^{n-2} \prod_{i=1}^{k} \frac{1}{i} \leq \sum_{k=0}^{\infty} \frac{1}{k!} = e.$$

Appendix D. Probability of selecting initial pivot column immediately. The choice of which column is used as an initial pivot candidate is the obvious difference between the bounded Bunch–Kaufman and fast Bunch–Parlett algorithms. The fast Bunch–Parlett algorithm chooses $\max_i |a_{ii}|$; the bounded Bunch–Kaufman algorithm uses whatever value is held by a_{11}. Intuitively the former, as the maximum of n values, is more likely to satisfy the initial 1×1 pivot test. In this appendix we quantify this intuition in a formal manner for matrices whose entries are all drawn from the same random distribution. We are able to quantify the results numerically for matrices whose entries are $\mathcal{N}(0,1)$.

Suppose that A is symmetric indefinite, with all entries a_{ij} drawn randomly from a variable x with density function f symmetric around the origin and with cumulative distribution F. The cumulative distribution of $|x|$ is $2F(|x|) - 1$. Then

$$\mathcal{P}\left(\frac{|a_{11}|}{\alpha} \geq \max\{|a_{21}|, |a_{31}|, \ldots, |a_{n1}|\}\right) = \int_0^\infty \left(2F\left(\frac{y}{\alpha}\right) - 1\right)^{n-1} f(y)dy.$$

In contrast, let a_{ss} be the diagonal entry of A of largest magnitude. Then $|a_{ss}| \sim G$, where

$$G(|y|) = \mathcal{P}(\max\{|a_{11}|, |a_{22}|, \ldots, |a_{nn}|\} \leq |y|) = (2F(|y|) - 1)^n,$$

from which it follows that

$$\begin{aligned}\mathcal{P}\left(\frac{|a_{ss}|}{\alpha} \geq \max_{i\neq s}\{|a_{is}|\}\right) &= \int_0^\infty \left(2F\left(\frac{y}{\alpha}\right)-1\right)^{n-1} dG(y)\\ &= 2n\int_0^\infty \left(2F\left(\frac{y}{\alpha}\right)-1\right)^{n-1} \left(2F\left(y\right)-1\right)^{n-1} f(y)dy.\end{aligned}$$

We can evaluate these integrals for specific distributions for which f and F are given. When the entries of A are drawn randomly from $\mathcal{N}(0,1)$,

$$f(y) = \frac{1}{\sqrt{2\pi}}e^{-\frac{y^2}{2}} \text{ and } F(y) = \int_{-\infty}^{y} f(x)dx\ .$$

Table 2.1 in section 2.5 gives the probabilities for this case for matrices of various moderate orders.

Acknowledgments. We have built on the work of many others. We have been directly assisted by Michel Daydé, with his portable BLAS3 package, and by Linda Kaufman, who provided encouragement and copies of her technical report and codes. Our colleague Fritz Scholz provided insight into the complexity of the local maximum off-diagonal search. Stan Eisenstat, Nick Higham, Alex Pothen, John Reid, and the anonymous referees were all careful and supportive readers, and each suggested changes that improved the presentation and the algorithms. We particularly thank Beresford Parlett for his enthusiastic support and his infectious desire to get the solutions right, on this and on many other projects.

REFERENCES

[1] J.O. Aasen, *On the reduction of a symmetric matrix to tridiagonal form*, BIT, 11 (1971), pp. 233–242.

[2] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen, *LAPACK Users' Guide, Release* 2.0, 2nd ed., SIAM, Philadelphia, 1995.

[3] E. Anderson and J. Dongarra, *Evaluating block algorithm variants in LAPACK*, in Proc. 5th SIAM Conference on Parallel Processing for Scientific Computing, J. Dongarra, P. Messina, D. Sorensen, and R. Voigt, eds., SIAM, Philadelphia, 1990, pp. 1–8.

[4] *The Boeing Extended Mathematical Subprogram Library*, Boeing Computer Services, P.O. Box 24346, Seattle, WA 98124-0346, 1989.

[5] V. Barwell and A. George, *A comparison of algorithms for solving symmetric indefinite systems of linear equations*, ACM Trans. Math. Software, 2 (1976), pp. 242–251.

[6] J. Bilmes, K. Asanović, J. Demmel, D. Lam, and C.-W. Chin, *Optimizing Matrix Multiply Using PHiPAC: A Portable, High-Performance, ANSI C Coding Methodology*, Tech. report LAPACK Working Note 111, University of Tennessee at Knoxville, Knoxville, TN, 1996.

[7] J.R. Bunch, *Analysis of the diagonal pivot method*, SIAM J. Numer. Anal., 8 (1971), pp. 656–680.

[8] J.R. Bunch and L. Kaufman, *Some stable methods for calculating inertia and solving symmetric linear systems*, Math. Comp., 31 (1977), pp. 163–179.

[9] J.R. Bunch and B.N. Parlett, *Direct methods for solving symmetric indefinite systems of linear equations*, SIAM J. Numer. Anal., 8 (1971), pp. 639–655.

[10] J.W. Demmel and N.J. Higham, *Stability of block algorithms with fast level-*3 *BLAS*, ACM Trans. Math. Software, 18 (1992), pp. 274–292.

[11] J.W. Demmel, N.J. Higham, and R.S. Schreiber, *Stability of block LU factorization*, Numer. Linear Algebra Appl., 2 (1995), pp. 173–190.

[12] J.J. Dongarra, I.S. Duff, D.C. Sorensen, and H. van der Vorst, *Solving Linear Systems on Vector and Shared Memory Computers*, SIAM, Philadelphia, 1991.

[13] J.J. Dongarra and S.C. Eisenstat, *Squeezing the most out of an algorithm in Cray Fortran*, ACM Trans. Math. Software, 10 (1984), pp. 219–230.

[14] I.S. Duff, N.I.M. Gould, J.K. Reid, J.A. Scott, and K. Turner, *The factorization of sparse symmetric indefinite matrices*, J. Inst. Maths. Appl., 11 (1991), pp. 181–204.

[15] I.S. Duff, R.G. Grimes, and J.G. Lewis, *Sparse matrix test problems*, ACM Trans. Math. Software, 15 (1989), pp. 1–14.

[16] I.S. Duff and J.K. Reid, MA27: *A Set of Fortran Subroutines for Solving Sparse Symmetric Sets of Linear Equations*, Tech. report AERE R 10533, AERE Harwell, Didcot, Oxon, UK, 1982.

[17] I.S. Duff and J.K. Reid, *The multifrontal solution of indefinite sparse symmetric linear equations*, ACM Trans. Math. Software, 9 (1983), pp. 302–325.

[18] I.S. Duff and J.K. Reid, MA47, *a Fortran Code for Direct Solution of Indefinite Sparse Symmetric Linear Systems*, Tech. report RAL-95-001, Central Computing Department, Rutherford Appleton Laboratory, Didcot, Oxon, UK, 1995.

[19] R. Fletcher, *Factorizing symmetric indefinite matrices*, Linear Algebra Appl., 14 (1976), pp. 257–272.

[20] G. Golub and C. Van Loan, *Matrix Computations*, 2nd ed., Johns Hopkins Press, Baltimore, MD, 1989.

[21] R.G. Grimes, J.G. Lewis, and H.D. Simon, *A shifted block Lanczos algorithm for solving sparse symmetric generalized eigenproblems*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 228–272.

[22] N.J. Higham, *The accuracy of solutions to triangular systems*, SIAM J. Numer. Anal., 26 (1989), pp. 1252–1265.

[23] N.J. Higham, *Iterative refinement for linear systems and LAPACK*, IMA J. Numer. Anal., 17 (1997), pp. 495–509.

[24] N.J. Higham, *Stability of the Diagonal Pivoting Method with Partial Pivoting*, Numerical Analysis Report No. 265, Manchester Centre for Computational Mathematics, Manchester, England, 1995; SIAM J. Matrix Anal. Appl., 18 (1997), pp. 52–65.

[25] N.J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia, 1996.

[26] M.T. Jones and M.L. Patrick, *Factoring symmetric indefinite matrices on high-performance architectures*, SIAM J. Sci. Comput., 15 (1994), pp. 273–283.

[27] L. Kaufman, *Computing the MDM^T factorization*, ACM Trans. Math. Software, 21 (1995), pp. 476–489.

[28] J.W.-H. Liu, *On threshold pivoting in the multifrontal method for sparse indefinite systems*, ACM Trans. Math. Software, 13 (1987), pp. 250–261.

[29] J.W.-H. Liu, *A partial pivoting strategy for sparse symmetric matrix decomposition*, ACM Trans. Math. Software, 13 (1987), pp. 173–182.

[30] B.N. Parlett and J.K. Reid, *On the solution of a system of linear equations whose matrix is symmetric but not definite*, BIT, 10 (1970), pp. 386–397.

[31] L.N. Trefethen and R. Schreiber, *Average case stability of Gaussian elimination*, SIAM J. Matrix Anal. Appl., 11 (1990), pp. 335–360.

[32] J.H. Wilkinson, *Rounding Errors in Algebraic Processes*, Prentice–Hall, Englewood Cliffs, NJ, 1963.

SIAM J. MATRIX ANAL. APPL.
Vol. 20, No. 3, pp. 563–574

ON SOME EIGENVECTOR-EIGENVALUE RELATIONS*

S. ELHAY†, G. M. L. GLADWELL‡, G. H. GOLUB§, AND Y. M. RAM¶

Abstract. This paper generalizes the well-known identity which relates the last components of the eigenvectors of a symmetric matrix $\mathbf{A}$ to the eigenvalues of $\mathbf{A}$ and of the matrix $\mathbf{A}_{n-1}$, obtained by deleting the last row and column of $\mathbf{A}$. The generalizations relate to matrices and to Sturm–Liouville equations.

Key words. eigenvalue, eigenvector, matrix, Sturm–Liouville equation, Green's function

AMS subject classifications. 15A18, 15A24, 34B25, 34B27

PII. S089547989631072X

1. Introduction.

We use the term *Jacobi matrix* to denote a real symmetric tridiagonal matrix with positive off-diagonal terms. It is well known that a Jacobi matrix $\mathbf{A}$ may be uniquely constructed from the eigenvalues $\{\lambda_i\}_1^n$ and $\{\mu_i\}_1^{n-1}$ of $\mathbf{A}$ and of its leading principal minor $\mathbf{A}_{n-1}$, respectively. The first step in the reconstruction procedure is to use the given eigenvalues, which must interlace so that

$$\lambda_1 < \mu_1 < \lambda_2 < \cdots < \mu_{n-1} < \lambda_n \tag{1.1}$$

to yield the last elements, $x_n^{(i)}$, of the normalized eigenvectors $\mathbf{x}^{(\mathbf{i})}$ of $\mathbf{A}$. We use the fact that $\{\mu_j\}_1^{n-1}$ are the zeros of

$$\sum_{i=1}^{n} \frac{[x_n^{(k)}]^2}{\lambda_i - \lambda} = 0, \tag{1.2}$$

so that

$$\sum_{k=1}^{n} \frac{[x_n^{(k)}]^2}{\lambda_k - \lambda} = c\frac{\prod_{j=1}^{n-1}(\mu_j - \lambda)}{\prod_{j=1}^{n}(\lambda_j - \lambda)}. \tag{1.3}$$

The identity $\sum_{k=1}^{n}[x_n^{(k)}]^2 = 1$ implies $c = 1$, and thus

$$[x_n^{(k)}]^2 = \frac{\prod_{j=1}^{n-1}(\mu_j - \lambda_k)}{\prod_{j=1,j\neq k}^{n}(\lambda_j - \lambda_k)}. \tag{1.4}$$

This relation, which has been known for many years (see, say, [7]), is an example of what we term an *eigenvector-eigenvalue relation.*

[*] Received by the editors October 16, 1996; accepted for publication (in revised form) by J. Varah February 19, 1998; published electronically January 13, 1999.
http://www.siam.org/journals/simax/20-3/31072.html

[†] Department of Computer Science, University of Adelaide, Adelaide SA 5005, Australia (elhay@cs.adelaide.edu.au).

[‡] Solid Mechanics Division, Faculty of Engineering,University of Waterloo, Waterloo, ON N2L 3G1 (ggladwel@uwaterloo.ca).

[§] Computer Science Department, Stanford University, Stanford, CA, 94305 (golub@sccm.stanford.edu). The work of this author was supported in part by National Science Foundation grant DMS-9403899.

[¶] Department of Mechanical Engineering, University of Adelaide, Adelaide SA 5005, Australia (yram@aelmg.adelaide.edu.au).

Many other such relations are known; all involve the eigenvalues of $\mathbf{A}$ and of some modification of $\mathbf{A}$. As another example, if $\{\lambda_i^*\}_1^n$ are the eigenvalues of the Jacobi matrix $\mathbf{A}^*$, obtained by replacing the last diagonal element a_n by a_n^*, then [6]

$$1 + (a_n^* - a_n)\sum_{k=1}^n \frac{[x_n^{(k)}]^2}{\lambda_k - \lambda} = \prod_{j=1}^n \left(\frac{\lambda_j^* - \lambda}{\lambda_j - \lambda}\right), \tag{1.5}$$

from which we deduce

$$(a_n^* - a_n)[x_n^{(k)}]^2 = \frac{\prod_{j=1}^n (\lambda_j^* - \lambda_k)}{\prod_{j=1, j\neq k}^n (\lambda_j - \lambda_k)}. \tag{1.6}$$

By comparing the traces of $\mathbf{A}$ and $\mathbf{A}^*$ we obtain

$$(a_n^* - a_n) = \sum_{k=1}^n (\lambda_k^* - \lambda_k). \tag{1.7}$$

Equations (1.6), (1.7) provide an eigenvector-eigenvalue relation; it is a particular case of a relation referring to the eigenvalues to $\mathbf{A}$ and some rank-one modification of A; see [10].

Equations (1.4), (1.5), and (1.6) relate to a Jacobi matrix; there are analogous results for continuous systems governed by Sturm–Liouville equations. We give an example, the analogue of (1.5), not the most general result.

Let $\{\lambda_i\}_0^\infty$, $\{\lambda_i^*\}_0^\infty$ be the eigenvalues of the nonuniform string equation

$$y''(x) + \lambda\rho(x)y(x) = 0, \tag{1.8}$$

subject to two sets of end conditions

$$y'(0) - hy(0) = 0 = y'(l) + Hy(l), \tag{1.9}$$

$$y'(0) - hy(0) = 0 = y'(l) + H^*y(l), \tag{1.10}$$

differing only at the right end, and let $\rho(x)$ be continuous; then the end values $y_m(l)$ of the normalized eigenfunctions of (1.8), subject to the condition (1.9), satisfy the equation

$$1 + (H^* - H)\sum_{m=0}^\infty \frac{[y_m(l)]^2}{\lambda_m - \lambda} = c\frac{\prod_{m=0}^\infty (1 - \frac{\lambda}{\lambda_m^*})}{\prod_{m=0}^\infty (1 - \frac{\lambda}{\lambda_m})}. \tag{1.11}$$

Since λ_m, $\lambda_m^* = O(m^2)$ for large m, both infinite products converge. Again we find

$$(H^* - H)[y_n(l)]^2 = c\lambda_n \prod_{m=0}^\infty \left(1 - \frac{\lambda_n}{\lambda_m^*}\right) \Bigg/ \prod_{m=0}^\infty{}' \left(1 - \frac{\lambda_n}{\lambda_m}\right), \tag{1.12}$$

where $c\prod_{m=0}^\infty (\frac{\lambda_m}{\lambda_m^*}) = 1$, and the prime means $m \neq n$. This example appears in [8] and is rederived in [6, p. 180].

The purpose of this paper is to explore some generalizations of the relations we have described. They will refer to discrete and continuous systems. They will involve squares of eigenvector/eigenfunction values at interior points and products of such values at two different points.

2. The generalized eigenvalue problem. Let **A**, **B** be symmetric matrices of order n, with **B** positive definite, and let $\{\lambda_i\}_1^n$, $\{\mathbf{x}^{(i)}\}_1^n$ be the eigenpairs of

$$(\mathbf{A} - \lambda\mathbf{B})\mathbf{x} = \mathbf{0}; \tag{2.1}$$

then **A**, **B** may be simultaneously diagonalized so that

$$\mathbf{X}^T\mathbf{A}\mathbf{X} = \wedge, \qquad \mathbf{X^T B X} = \mathbf{I}, \tag{2.2}$$

where $\wedge = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$, $\mathbf{X} = [\mathbf{x^{(1)}}, \mathbf{x^{(2)}}, \ldots, \mathbf{x^{(n)}}]$, and

$$\mathbf{A} - \lambda\mathbf{B} = \mathbf{X}^{-T}(\wedge - \lambda\mathbf{I})\mathbf{X}^{-1}. \tag{2.3}$$

Provided that $\lambda \neq \lambda_i$, $i = 1, 2, \ldots, n$, we may invert this to give

$$(\mathbf{A} - \lambda\mathbf{B})^{-1} = \mathbf{X}(\wedge - \lambda\mathbf{I})^{-1}\mathbf{X}^T. \tag{2.4}$$

This gives the following lemma.

LEMMA 2.1. *Provided the eigenvalue λ_k is simple, then*

$$x_i^{(k)}x_j^{(k)} = \lim_{\lambda\to\lambda_k}(\lambda_k - \lambda)\mathbf{e_i^T}(\mathbf{A} - \lambda\mathbf{B})^{-1}\mathbf{e_j} \tag{2.5}$$

where $\mathbf{e}_i$ denotes the ith column of **I**.

Proof. Let $\alpha_{ij}(\lambda)$ be the i, j element of $(\mathbf{A} - \lambda\mathbf{B})^{-1}$, so that

$$\alpha_{ij}(\lambda) = \mathbf{e}_i^T(\mathbf{A} - \lambda\mathbf{B})^{-1}\mathbf{e}_j. \tag{2.6}$$

Then (2.4) gives

$$\alpha_{ij}(\lambda) = \sum_{k=1}^n \frac{x_i^{(k)}x_j^{(k)}}{\lambda_k - \lambda} \tag{2.7}$$

so that, provided the eigenvalue λ_k is simple,

$$x_i^{(k)}x_j^{(k)} \quad = \lim\nolimits_{\lambda\to\lambda_k}(\lambda_k - \lambda)\alpha_{ij}(\lambda). \qquad \square \tag{2.8}$$

We now apply this result with some special choices of **A** and **B**. We introduce some notation. Let $\mathbf{A}_k(\mathbf{A}_k^R)$, $1 \leq k \leq n$, denote the leading (trailing) principal submatrix of order k of the square matrix **A**. We let $P_k(\lambda)$,$Q_k(\lambda)$ denote, respectively, the kth order leading and trailing principal minors of $\mathbf{A} - \lambda\mathbf{B}$, and $(\mu_i)_1^{n-1}$ denote the zeros of $P_{n-1}(\lambda)$.

THEOREM 2.1.

$$[x_n^{(k)}]^2 = \frac{|\mathbf{B}_{n-1}|\prod_{j=1}^{n-1}(\mu_j - \lambda_k)}{|\mathbf{B}_n|\prod_{j=1}^{n\prime}(\lambda_j - \lambda_k)}, \tag{2.9}$$

where $\prime$ denotes $j \neq k$.

Proof. Consider the equation

$$(\mathbf{A} - \lambda\mathbf{B})\mathbf{y} = \mathbf{e}_n. \tag{2.10}$$

If $\lambda \neq \lambda_i$ $(i = 1, 2, \ldots, n)$, then

$$y_n = \mathbf{e}_n^T(\mathbf{A} - \lambda\mathbf{B})^{-1}\mathbf{e}_n = \alpha_{nn}(\lambda). \tag{2.11}$$

But Cramer's rule applied to (2.10) gives

$$y_n = \frac{|\mathbf{A}_{n-1} - \lambda \mathbf{B}_{n-1}|}{|\mathbf{A}_n - \lambda \mathbf{B}_n|} = \frac{P_{n-1}(\lambda)}{P_n(\lambda)}. \tag{2.12}$$

But $\{\mu_i\}_1^{n-1}$ are the zeros of $P_{n-1}(\lambda)$ and $(\lambda_i)_1^n$ are the zeros of $P_n(\lambda)$, so that

$$\alpha_{nn}(\lambda) = y_n = \frac{|\mathbf{B}_{n-1}| \prod_{j=1}^{n-1}(\mu_j - \lambda)}{|\mathbf{B}_n| \prod_{j=1}^{n}(\lambda_j - \lambda)}, \tag{2.13}$$

and Lemma 2.1, with $i = j = n$, gives

$$[x_n^{(k)}]^2 = \frac{|\mathbf{B}_{n-1}| \prod_{j=1}^{n-1}(\mu_j - \lambda_k)}{|\mathbf{B}_n| \prod_{j=1}^{n'}(\lambda_j - \lambda_k)}. \qquad \square \tag{2.14}$$

Theorem 2.1 generalizes (1.4). It holds provided that λ_k is simple. We recall that the eigenvalues of a Jacobi matrix (to which (1.4) applies) are always simple, and the eigenvalues λ_i, μ_i appearing in (1.4) always *strictly* interlace according to (1.1), so that (1.4) never breaks down and $[x_n^{(k)}]^2$ is always positive; $x_n^{(k)}$ is never zero. For the generalized eigenvalue problem (2.1), the eigenvalues need not be simple, and the eigenvalues λ_i, μ_i satisfy only

$$\lambda_1 \le \mu_1 \le \lambda_2 \le \cdots \le \mu_{n-1} \le \lambda_n. \tag{2.15}$$

If λ_k is simple, so that $\lambda_{k-1} < \lambda_k < \lambda_{k+1}$, then (2.9) holds for that value of k; $x_n^{(k)}$ will be zero if $\mu_{k-1} = \lambda_k$ or $\mu_{k+1} = \lambda_k$. If λ_k is not simple, then we have the following corollary.

COROLLARY 2.1. *Suppose* λ_k *has multiplicity* s, *so that*

$$\lambda_{k-1} < \lambda_k = \mu_k = \lambda_{k+1} = \cdots = \mu_{k+s-2} = \lambda_{k+s-1} \le \mu_{k+s-1} < \lambda_{k+s}.$$

Then

$$[x_n^{(k)}]^2 = \frac{|B_{n-1}| \prod_{j=1}^{n-1}{}'(\mu_j - \lambda_k)}{|B_n| \prod_{j=1}^{n}{}''(\lambda_j - \lambda_k)}, \tag{2.16}$$

where $'$ *means* $j = 1, 2, \ldots, k-1$, $k+s-1, \ldots, n-1$ *and* $''$ *means* $j = 1, 2, \ldots, k-1$, $k+s, \ldots, n$.

Proof. The numerator and denominator of (2.14) will have a common factor $(\lambda_k - \lambda)^{s-1}$ which can be cancelled. $\square$

The results in Theorem 2.1 and its corollary are obtained from Lemma 2.1 for $i = j = n$. There are analogous expressions for $[x_1^{(k)}]^2$, but there are, in general, no simple extensions to $[x_m^{(k)}]^2$ where $1 < m < n$. To obtain simple extensions we must restrict the forms of $\mathbf{A}$ and $\mathbf{B}$ and suppose that they are both tridiagonal.

We obtain the following lemma.

LEMMA 2.2. *Suppose* $\mathbf{A}$ *and* $\mathbf{B}$ *are tridiagonal matrices with codiagonals* $(b_1, b_2, \ldots, b_{n-1})$ *and* $(d_1, d_2, \ldots, d_{n-1})$, *respectively, and suppose* $1 \le i \le j \le n$ *and* $\lambda \ne \lambda_k$, $k = 1, 2, \ldots, n$. *Then*

$$\alpha_{ij}(\lambda) = (-1)^{i+j} \frac{P_{i-1}(\lambda) \prod_{k=i}^{j-1}(b_k - \lambda d_k) Q_{n-j}(\lambda)}{P_n(\lambda)}. \tag{2.17}$$

Proof. Consider the equation

$$(\mathbf{A}-\lambda\mathbf{B})y=\mathbf{e}_j. \tag{2.18}$$

If $\lambda \neq \lambda_k$, $k=1,2,\ldots,n$, then

$$y_i=\mathbf{e}_i^T(\mathbf{A}-\lambda\mathbf{B})^{-1}\mathbf{e}_j=\alpha_{ij}(\lambda). \tag{2.19}$$

But Cramer's rule applied to (2.18) shows that

$$\alpha_{ij}(\lambda)=y_i=\frac{\hat{p}_{ij}(\lambda)}{p_n(\lambda)}, \tag{2.20}$$

where $\hat{p}_{ij}(\lambda)$ is the determinant of the matrix obtained by replacing the ith column of $\mathbf{A}-\lambda\mathbf{B}$ by $\mathbf{e}_j$. Thus

$$\hat{p}_{ij}(\lambda)=\begin{array}{|ccccccc|}
 & & & i & & & \\
a_1-\lambda c_1 & b_1-\lambda d_1 & & 0 & & & \\
b_1-\lambda d_1 & a_2-\lambda c_2 & \cdots & 0 & & & \\
\cdots & \cdots & \cdots & 1 & & & \\
 & \cdots & & 0 & \cdots & a_{n-1}-\lambda c_{n-1} & b_{n-1}-\lambda d_{n-1} \\
 & & & 0 & \cdots & b_{n-1}-\lambda d_{n-1} & a_n-\lambda c_n
\end{array}\; j. \tag{2.21}$$

Assuming that $i \le j$, and expanding $\hat{p}_{ij}(\lambda)$ along its ith column, we obtain the stated result. □

Note that the numerator involves a leading minor, a product, and a trailing minor.

Let us consider the interpretation of (2.18). The original equation may be interpreted as the equation governing the free vibration of a mechanical system with stiffness and inertia matrices $\mathbf{A}$, $\mathbf{B}$, respectively. The eigenvalues λ_i are the squares, ω_i^2, of the natural frequencies; and the $\mathbf{x}_i$, the eigenvectors, are the mode shapes. The term $\alpha_{ij}(\lambda)$ is the influence function linking coordinates i and j: it gives the solution y_i for a unit in the jth place on the right-hand side of (2.14). We can think of this as the *displacement* at i due to a unit *load* at j; thus $\alpha_{ij}(\lambda)$ is the *receptance* [3] between points i and j. Equation (2.18) thus states that $\alpha_{ij}(\lambda)$ is zero when $P_{i-1}(\lambda)=0$, $Q_{n-j}(\lambda)=0$, or the product is zero. But the zeros of $P_{i-1}(\lambda)$ are simply the eigenvalues $(\lambda_k^L)_1^{i-1}$ of the equation

$$(\mathbf{A}_{i-1}-\lambda\mathbf{B}_{i-1})\mathbf{x}=\mathbf{0}, \tag{2.22}$$

while the zeros of $Q_{n-j}(\lambda)$ are the eigenvalues $(\lambda_k^R)_1^{n-j}$ of the equation

$$(\mathbf{A}_{n-j}^R-\lambda\mathbf{B}_{n-j}^R)\mathbf{x}=\mathbf{0}. \tag{2.23}$$

(Remember that $A_k(A_k^R)$ is the leading (trailing) principal submatrix of order k.) In the important physical case in which $b_k<0$ and $d_k \ge 0$, the product in (2.18) has no positive zero; we examine this application in section 5.

We now have the following theorem.

THEOREM 2.2. *Suppose* $\mathbf{B}$ *is diagonal,* $1 \le i \le j \le n$, *and* $1 \le k \le n$. *Then*

$$\frac{x_i^{(k)}x_j^{(k)}}{\alpha_{ij}(0)}=\frac{\lambda_k\prod_{m=1}^{i-1}(1-\frac{\lambda_k}{\lambda_m^L})\prod_{m=1}^{n-j}(1-\frac{\lambda_k}{\lambda_m^R})}{\prod_{m=1}^{n}{}'(1-\frac{\lambda_k}{\lambda_m})}. \tag{2.24}$$

Proof. When $\mathbf{B}$ is diagonal, $d_r = 0$, so that (2.17) reduces to

$$\alpha_{ij} = c\frac{P_{i-1}(\lambda)Q_{n-j}(\lambda)}{P_n(\lambda)}, \qquad c = (-1)^{i+j}\prod_{r=i}^{j-1} b_r. \tag{2.25}$$

On normalizing α_{ij} by its value for $\lambda = 0$, and by factorizing the various terms, we find

$$\Phi_{ij}(\lambda) \equiv \frac{\alpha_{ij}(\lambda)}{\alpha_{ij}(0)} = \frac{\prod_{m=1}^{i-1}(1-\frac{\lambda}{\lambda_m^L})\prod_{m=1}^{n-j}(1-\frac{\lambda}{\lambda_m^R})}{\prod_{m=1}^{n}(1-\frac{\lambda}{\lambda_m})}. \tag{2.26}$$

Using this in (2.6), we find the required result. □

We note that, when $\mathbf{B}$ is diagonal, the $(\lambda_m)_1^n$ are distinct, so that the denominator in (2.24) cannot be zero. The numerator can be zero, since one of each of the sets $(\lambda_m^L)_1^{i-1}$ and $(\lambda_m^R)_1^{n-j}$ can be equal to λ_k. Theorem 2.1, Corollary 2.1, and Theorem 2.2 provide generalization of the known result (1.4).

We now explore some continuous analogues of these results.

3. Sturm–Liouville systems.

Let L denote the Sturm–Liouville operator given by

$$Ly(x) = -(p(x)y'(x))' + q(x)y(x), \tag{3.1}$$

and consider the system

$$Ly(x) = \lambda\rho(x)y(x), \tag{3.2}$$

subject to the end conditions

$$p(0)y'(0) - hy(0) = 0 = p(l)y'(l) + Hy(l). \tag{3.3}$$

Such a system can model the free vibration of a rod or string fastened at its ends by springs of stiffness h, H, respectively; h or H is zero at a free end, infinite at a fixed end.

We are not concerned here with the most general regularity conditions satisfied by p, q, and ρ. We assume that $p(x)$, $p'(x)$, $q(x)$, and $\rho(x)$ are continuous in $(0, l)$. References to the general theory may be found in [9], [1], and [2].

The *Green*'s *function* $G(x, s, \lambda)$ for the system satisfies, as a function of x

(a) the equation (3.2), except at s,

(b) the end conditions (3.3),

(c) the jump condition

$$[p(x)y'(x)]_{x=s-}^{x=s+} = -1. \tag{3.4}$$

It is well known [4, Chapter 5] that if λ is not an eigenvalue of (3.2), (3.3), then $G(x, s, \lambda)$ may be constructed as follows. Let $\phi(x)$, $\psi(x)$ be solutions of (3.2) satisfying

$$p(0)\phi'(0) - h\phi(0) = 0 = p(l)\psi'(l) + H\psi(l), \tag{3.5}$$

respectively; then

$$p(x)\{\phi(x)\psi'(x) - \phi'(x)\psi(x)\} = \text{constant}. \tag{3.6}$$

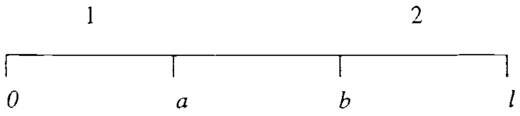


FIG. 1.

This constant is zero if λ is an eigenvalue of (3.2), (3.3) and nonzero otherwise; in the latter case we can choose the constant to be -1, and then

$$G(x,s,\lambda)=\begin{cases}\phi(x)\psi(s), & 0\le x\le s,\\ \phi(s)\psi(x), & s\le x\le l.\end{cases}\tag{3.7}$$

The functions ϕ, ψ are functions of x and λ; if $\lambda=0$ is not an eigenvalue of (3.2), (3.3), i.e., if h, H are not both zero, then

$$G(x,s,0)=\begin{cases}\phi_0(x)\psi_0(s), & 0\le x\le s,\\ \phi_0(s)\psi_0(x), & s\le x\le l,\end{cases}\tag{3.8}$$

where $\phi_0(x)$, $\psi_0(x)$ denote $\phi(x)$, $\psi(x)$, respectively, for $\lambda=0$.

The Green's function $G(x,s,\lambda)$ is the analogue of the receptance $\alpha_{ij}(\lambda)$ for matrix systems; like $\alpha_{ij}(\lambda)$, it has poles and zeros. In section 2 we explicitly showed $\alpha_{ij}(\lambda)$ as a product of two polynomials $P_{i-1}(\lambda)$, $Q_{n-j}(\lambda)$ divided by $P_n(\lambda)$. The polynomials $P_{i-1}(\lambda)$, $Q_{n-j}(\lambda)$ related to the parts of the system, respectively, to the left of i and to the right of j, and their zeros were the eigenvalues of these parts, as shown in (2.22), (2.23). We will now show how the Green's function $G(a,b,\lambda)$, with $a\le b$, may be expressed in an analogous way as a product of two quantities referring, respectively, to the parts of the system to the left of a, and to the right of b, divided by a third quantity relating to the whole interval $(0,l)$.

Suppose $a\le b$, and define the normalized Green's function

$$\Phi(a,b,\lambda)=\frac{G(a,b,\lambda)}{G(a,b,0)}=\frac{\phi(a)\psi(b)}{\phi_0(a)\psi_0(b)}.\tag{3.9}$$

Let $\{\lambda_n^L\}_1^\infty$, $\{\lambda_n^R\}_1^\infty$ be the eigenvalues of the subsystems S_1, S_2 governed by (3.2) with the end conditions

$$1.\quad p(0)y'(0)-hy(0)=0,\qquad y(a)=0,\tag{3.10}$$

$$2.\quad y(b)=0,\qquad p(l)y'(l)+Hy(l)=0,\tag{3.11}$$

and let $\{\lambda_n\}_1^\infty$ be the eigenvalues of (3.1)–(3.3). We prove the following theorem.

THEOREM 3.1.

$$\Phi(a,b,\lambda)=\frac{\prod_{n=1}^\infty(1-\frac{\lambda}{\lambda_n^L})\prod_{n=1}^\infty(1-\frac{\lambda}{\lambda_n^R})}{\prod_{n=1}^\infty(1-\frac{\lambda}{\lambda_n})}.\tag{3.12}$$

Proof. Fig. 1 shows the system divided into three parts.

To find the Green's function of system 1, we need function $\phi_1(x)$, $\psi_1(x)$ satisfying, respectively, the left and right end conditions of system 1 and the jump condition; we may take

$$\phi_1(x)=\phi(x),\qquad \psi_1(x)=\psi(x)-\frac{\psi(a)}{\phi(a)}\phi(x).\tag{3.13}$$

We now introduce the function

$$H_1(\lambda) = \lim_{\varepsilon\to 0} \frac{G_1(\varepsilon, a-\varepsilon, 0)}{G_1(\varepsilon, a-\varepsilon, \lambda)} \tag{3.14}$$

$$= \lim_{\varepsilon\to 0} \frac{\phi_{1,0}(\varepsilon)}{\phi_1(\varepsilon)} \cdot \frac{\psi_{1,0}(a-\varepsilon)}{\psi_1(a-\varepsilon)}, \tag{3.15}$$

where $\phi_{1,0}(x)$, $\psi_{1,0}(x)$ denote, respectively, the values of $\phi_1(x)$, $\psi_1(x)$ for $\lambda = 0$. Since $\psi_1(a) = 0 = \psi_{1,0}(a)$, we have

$$\lim_{\varepsilon\to 0} \frac{\psi_{1,0}(a-\varepsilon)}{\psi_1(a-\varepsilon)} = \frac{\psi'_{1,0}(a)}{\psi'_1(a)} = \frac{\phi(a)}{\phi_0(a)} \tag{3.16}$$

because of (3.6). The limit of the first quotient in (3.14) is

$$\lim_{\varepsilon\to 0} \frac{\phi_{1,0}(\varepsilon)}{\phi_1(\varepsilon)} = \lim_{\varepsilon\to 0} \frac{\phi_0(\varepsilon)}{\phi_1(\varepsilon)} = \frac{\phi_0(0)}{\phi(0)} \tag{3.17}$$

if h is finite, and $\phi'_0(0)/\phi'(0)$ if h is infinite. Thus

$$H_1(\lambda) = \begin{cases} \frac{\phi_0(0)}{\phi(0)} \cdot \frac{\phi(a)}{\phi_0(a)} & \text{if } h \text{ is finite,} \\ \frac{\phi'_0(0)}{\phi'(0)} \cdot \frac{\phi(a)}{\phi_0(a)} & \text{if } h \text{ is infinite.} \end{cases} \tag{3.18}$$

We may now consider system 2 similarly. We take

$$\phi_2(x) = \phi(x) - \frac{\phi(b)}{\psi(b)}\psi(x), \qquad \psi_2(x) = \psi(x), \tag{3.19}$$

define

$$H_2(\lambda) = \lim_{\varepsilon\to 0} \frac{G_2(b+\varepsilon, l-\varepsilon, 0)}{G_2(b+\varepsilon, l-\varepsilon, \lambda)}, \tag{3.20}$$

and find

$$H_2(\lambda) = \begin{cases} \frac{\psi(b)}{\psi_0(b)} \cdot \frac{\psi_0(l)}{\psi(l)} & \text{if } H \text{ is finite,} \\ \frac{\psi(b)}{\psi_0(b)} \cdot \frac{\psi'_0(l)}{\psi'(l)} & \text{if } H \text{ is infinite.} \end{cases} \tag{3.21}$$

Finally we take the whole system, with end conditions (3.3), and define

$$H(\lambda) = \lim_{\varepsilon\to 0} \frac{G(\varepsilon, l-\varepsilon, 0)}{G(\varepsilon, l-\varepsilon, \lambda)} \tag{3.22}$$

and find that this has one of the values

$$\frac{\phi_0(0)\psi_0(l)}{\phi(0)\psi(l)}, \quad \frac{\phi'_0(0)\psi_0(l)}{\phi'(0)\psi(l)}, \quad \frac{\phi_0(0)\psi'_0(l)}{\phi(0)\psi'(l)}, \quad \frac{\phi'_0(0)\psi'_0(l)}{\phi'(0)\psi'(l)} \tag{3.23}$$

according to whether both, one, the other, or neither of h, H are finite.

Combining (3.18), (3.21), (3.22) we find the fundamental relation for $\Phi(a, b, \lambda)$ defined in (3.9):

$$\Phi(a, b, \lambda) = \frac{H_1(\lambda)H_2(\lambda)}{H(\lambda)}. \tag{3.24}$$

We now show that $H_1(\lambda)$, $H_2(\lambda)$, and $H(\lambda)$ may, like $P_{i-1}(\lambda)$, $Q_{n-j}(\lambda)$ and $P_n(\lambda)$ in (2.21), be expressed as product of factors relating to the subsystems 1, 2, and the whole.

First consider the case when h is finite. It is known (see, for example, [11]) that, treated as a function of λ, the quantity $\phi(a)/\phi(0)$ is an entire function of λ, of order 1/2. By Hadamard's factorization theorem it may therefore be expressed as a product of its factors; i.e.,

$$\frac{\phi(a)}{\phi(0)} = c\prod_{n=1}^{\infty}\left(1-\frac{\lambda}{a_n}\right). \tag{3.25}$$

But the zeros of $\phi(a)$ are precisely the eigenvalues of system 1 (i.e., λ_n^L) so that $a_n = \lambda_n^L$. It is known [11] that $\lambda_n^L = O(n^2)$ for large n, so that the product (3.24) converges. The constant c is $\phi_0(a)/\phi_0(0)$. Thus

$$H_1(\lambda) = \frac{\phi(a)}{\phi(0)}\cdot\frac{\phi_0(0)}{\phi_0(a)} = \prod_{n=1}^{\infty}\left(1-\frac{\lambda}{\lambda_n^L}\right). \tag{3.26}$$

If h is infinite, $H_1(\lambda)$ is defined by the second line of (3.18), but the final result for $H_1(\lambda)$ still holds.

We may deduce, in a similar way, that

$$H_2(\lambda) = \prod_{n=1}^{\infty}\left(1-\frac{\lambda}{\lambda_n^R}\right). \tag{3.27}$$

To investigate $H(\lambda)$, we note that the jump condition (3.6) and second of the end conditions (3.5) give

$$\psi(l)\{p(l)\phi'(l)+H\phi(l)\} = 1 = \psi_0(l)\{p(l)\phi_0'(l)+H\phi_0(l)\} \tag{3.28}$$

so that when h, H are finite, and not both zero,

$$H(\lambda) = \frac{p(l)\phi'(l)+H\phi(l)}{\phi(0)}\cdot\frac{\phi_0(0)}{p(l)\phi_0'(l)+H\phi_0(l)}. \tag{3.29}$$

Again, $H(\lambda)$ is an entire function of order 1/2, whose zeros are those λ for which there is a solution $\phi(x)$ satisfying *both* end conditions (3.3), i.e., $\lambda = \lambda_n$. Thus

$$H(\lambda) = \prod_{n=1}^{\infty}\left(1-\frac{\lambda}{\lambda_n}\right). \tag{3.30}$$

This result still holds when one or both of h, H are infinite.

Equations (3.24), (3.26), (3.27), (3.29) now yield the stated result (3.12). □

COROLLARY 3.1. *Let* $\{u_r(x)\}$ *be the eigenfunctions of* (3.1)–(3.3)*; then*

$$\frac{u_r(a)u_r(b)}{\sum_{r=1}^{\infty}\frac{u_r(a)u_r(b)}{\lambda_r}} = \frac{\lambda_r\prod_{n=1}^{\infty}(1-\frac{\lambda_r}{\lambda_n^L})\prod_{n=1}^{\infty}(1-\frac{\lambda_r}{\lambda_n^R})}{\prod_{n=1}^{\infty}{}'(1-\frac{\lambda_r}{\lambda_n})}. \tag{3.31}$$

Proof. The usual modal expansion of $G(a,b,\lambda)$, namely,

$$G(a,b,\lambda) = \sum_{r=1}^{\infty}\frac{u_r(a)u_r(b)}{\lambda_r-\lambda}, \tag{3.32}$$

when substituted in (3.9), gives

$$\Phi(a,b,\lambda) = \frac{\sum_{r=1}^{\infty} \frac{u_r(a)u_r(b)}{\lambda_r - \lambda}}{\sum_{r=1}^{\infty} \frac{u_r(a)u_r(b)}{\lambda_r}}, \tag{3.33}$$

which, when combined with Theorem 3.1, yields the required result. □

We note that

$$\sum_{r=1}^{\infty} \frac{u_r(a)u_r(b)}{\lambda_r} = \phi_0(a)\psi_0(b), \tag{3.34}$$

and $\phi_0(a), \psi_0(b)$ may be expressed explicitly in terms of $p(x)$, $q(x)$, h, H.

4. Conclusion. Corollary 3.1 provides the desired continuous analogue of Theorem 2.2. The analysis shows that the continuous analogues of the leading and trailing minors are the functions $H_1(\lambda)$, $H_2(\lambda)$, defined in (3.14), (3.20). We note that $H_1(\lambda)$, $H_2(\lambda)$ are not just reciprocals of the Green's functions, as one might expect at first thought, but limiting values of the normalized reciprocals of the Green's functions between the points near the ends of the intervals (0, a) and (b, l).

Various other eigenvector-eigenvalue relations may be obtained from Theorem 2.2 by taking $i = j$, $i = 1$, or $j = n$ or in Corollary 3.1 by taking $a = b$, $a = 0$, or $b = l$.

5. An application. Consider the mass spring system shown in Fig. 2. This is the customary model of an in-line axially vibrating system; by replacing the masses and axial springs by polar inertias and torsional springs, respectively, we obtain the analogous torsionally vibrating system. We show that the analysis may be used to find those frequencies of oscillation at which a particular displacement vanishes. These frequencies are the vibration absorber frequencies which play an important role in vibration isolation. We may tune, for example, the working frequencies of an unbalanced motor, mounted on a shaft, to eliminate the steady state vibrations at a certain position where sensitive mechanical or electrical equipment is located, such as isolation from vibration and noise-induced vibration usually required in aircraft structures. (For other applications, see [5, Sections 3.2, 3.3, 4.2].)

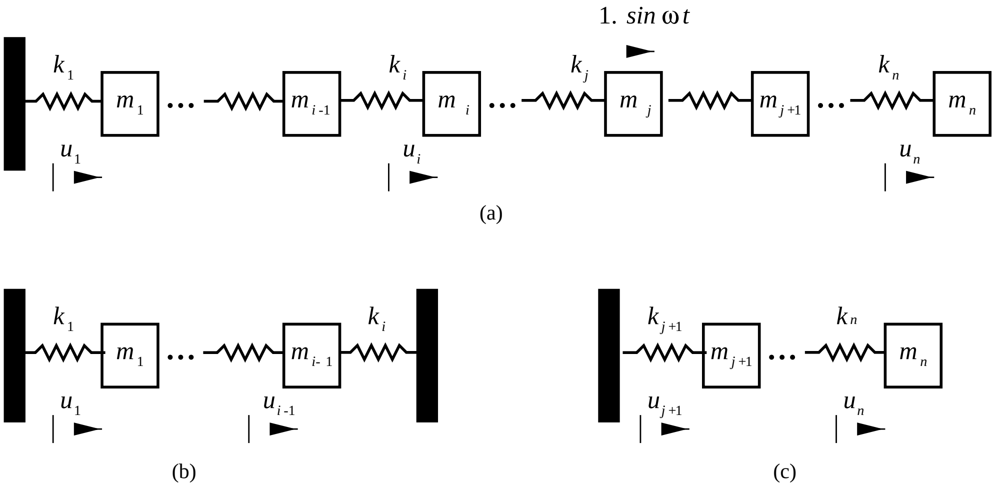


FIG. 2. *A spring-mass system* (a) *has left and right end parts shown in* (b) *and* (c), *respectively.*

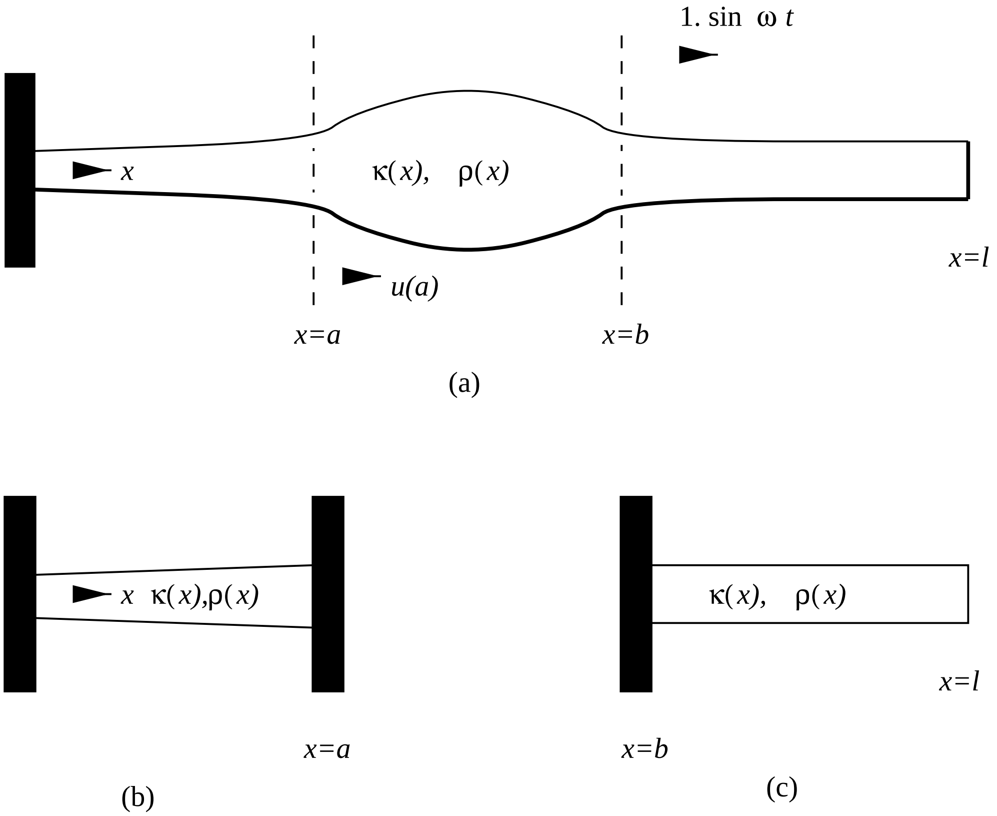


FIG. 3. *The rod* (a) *has left and right end parts* (b) *and* (c).

The receptance α_{ij} in (2.6) gives the response at mass i due to a unit load with frequency ω ($\omega^2 = \lambda$) at mass j. Without loss of generality we can consider just the case $i \leq j$. The expression (2.22) for α_{ij} shows that u_i will be zero if $\lambda = \lambda_m^L$ for some $m = 1, 2, \ldots, i-1$ or $\lambda = \lambda_m^L$ for some $m = 1, 2, \ldots, n-j$. Moreover, by applying superposition, we see that if loads are applied at masses $i+1, \ldots, n$ with the same frequency ω ($\omega^2 = \lambda$), then u_i will be zero if $\lambda = \lambda_m^L$ for some $m = 1, 2, \ldots, i-1$.

The continuous analogue of the discrete system of Fig. 2 is the axially vibrating rod shown in Fig. 3a. Now we apply Theorem 3.1. If a load with frequency ω ($\omega^2 = \lambda$) is applied at $x = b$, then the displacement at $x = a$ will be zero if $\lambda = \lambda_n^L$ or $\lambda = \lambda_n^R$ for some n. By superposition, we see that if loads are applied to the part $a < x \leq l$ with the same frequency ω, then $u(a)$ will be zero if $\lambda = \lambda_n^L$ for some n.

REFERENCES

[1] L. E. ANDERSSON, *Inverse eigenvalue problems with discontinuous coefficients*, Inverse Problems, 4 (1988), pp. 353–397.

[2] L. E. ANDERSSON, *Inverse eigenvalue problems for a Sturm-Liouville equation in impedance form*, Inverse Problems, 4 (1988) pp. 929–971.

[3] R. E. D. BISHOP AND D. C. JOHNSON, *The Mechanics of Vibration*, Cambridge University Press, London, 1960.

[4] R. COURANT AND D. HILBERT, *Methods of Mathematical Physics, Vol. I*, Interscience, New York, 1953.

[5] J. P. Den Hartog, *Mechanical Vibrations*, McGraw-Hill, New York, 1956.

[6] G. M. L. Gladwell, *Inverse Problems in Vibration*, Martinus Nijhoff Publishers, Dordrecht, the Netherlands, 1986.

[7] G. H. Golub, *Some uses of the Lanczos algorithm in numerical linear algebra*, in Topics in Numerical Analysis, J. H. H. Miller, ed., Springer-Verlag, Heidelberg, New York, 1973, pp. 23–31.

[8] B. M. Levitan, *On the determination of a Sturm-Liouville equation by spectra*, Izv. Akad. Nauk SSSR Ser. Mat., 28 (1964), pp. 63–68 (in Russian); Amer. Math. Soc. Transl., 68 (1964), pp. 1–20.

[9] J. R. McLaughlin, *Analyticaly methods for recovering coefficients in differential equations from spectral data*, SIAM Rev., 28 (1986), pp. 53–72.

[10] Y. M. Ram, *Inverse eigenvalue problems for a modified vibrating system*, SIAM J. Appl. Math, 53 (1993), pp. 1762–1775.

[11] E. C. Titchmarsh, *Eigenfunction Expansions, Part* I. Oxford University Press, Oxford, 1962.

SIAM J. MATRIX ANAL. APPL.
Vol. 20, No. 3, pp. 575–595

NONLINEAR EIGENPROBLEMS*

PHILIPPE GUILLAUME†

Abstract. Let $A(\lambda)$ be a holomorphic matrix-valued function defined on a domain $\Omega \subset \mathbb{C}$. The nonlinear eigenproblem of finding generalized eigenpairs $\lambda,\ v$ such that $A(\lambda)v = 0$ is considered. The method which is exposed for solving this problem is based on the derivatives of the function $x(\lambda) = A(\lambda)^{-1}b$, where b is a given vector. Theoretical convergence results are established, and an algorithm is proposed for computing a few eigenvalues close to a given complex number together with some corresponding eigenvectors.

Key words. nonlinear eigenproblem, polynomial eigenproblem, quadratic eigenproblem, generalized eigenproblem, Galerkin approximation, higher order derivatives

AMS subject classifications. 65F15, 65H17, 65N25, 15A18, 15A90

PII. S0895479897324172

1. Introduction.

Since automatic differentiation tools have appeared [12], [8], [7], new methods involving the derivatives of a given problem with respect to a parameter-like shape or frequency have been developed [3], [5], [2], [6]. They are based on the computation of the power series expansion of the solution to a linear system

$$A(\lambda)x(\lambda) = b$$

with respect to the parameter λ. It follows from the Leibniz formula that each derivative $x'(\lambda),\ x''(\lambda), \dots,\ x^{(k)}(\lambda)$ is the solution to a linear system involving the same matrix $A(\lambda)$:

$$A(\lambda)x^{(k)}(\lambda) = -\sum_{j=1}^{k} \binom{k}{j} A^{(j)}(\lambda)x^{(k-j)}(\lambda),$$

where $\binom{k}{j} = k!/(j!(k-j)!)$. In this paper we study some properties of the derivatives $x^{(k)}(\lambda)$ and we use them for solving the nonlinear eigenproblem

$$A(\lambda)v = 0.$$

If the matrix $A(\lambda)$ is a polynomial, it is possible to transform this problem into a linear eigenproblem by increasing the order of the system. For example, quadratic eigenproblems are frequently solved by adding the unknown $w = \lambda v$ [4], [14], [16], and the size of the resulting eigenvalue problem becomes twice the size of the original eigenproblem. Similarly, if $A(\lambda)$ is a polynomial of degree m, the size of the resulting eigenvalue problem is multiplied by m. Combined with a projection method, this technique allows the simultaneous computation of several eigenpairs. Another method which avoids increasing the order of the system has been proposed in [10] and is closely related to Newton's method for solving the problem $A(\lambda)v = 0,\ ||v||_2 = 1$. It allows

[*] Received by the editors July 9, 1997; accepted for publication (in revised form) by G. Golub April 22, 1998; published electronically January 27, 1999.
http://www.siam.org/journals/simax/20-3/32417.html

[†] UMR 5640, CNRS-INSA-Université Paul Sabatier, Avenue de Rangueil, 31077 Toulouse cedex, France (guillaum@gmm.insa-tlse.fr).

the computation of one eigenvector associated with an eigenvalue close to a given complex number. More recently, another approach has been described in [19], [20], which use a Jacobi–Davidson technique. At each iteration a correction equation of the same size as the original eigenproblem is solved, and if it is solved accurately, then quadratic convergence is obtained for the linear eigenvalue problem $(A - \lambda B)v = 0$. In fact this equation is solved only approximately, and the convergence is somewhere between linear and quadratic. A particularity of this technique is that the matrix of the correction equation changes at each iteration.

The method introduced in this paper is a generalization of the inverse iteration described in [10]. The convergence is only linear, although it should be possible to improve it by incorporating some ideas developed in [19]. However, the convergence is established in the general case where $A(\lambda)$ is holomorphic. Of particular interest is the fact that the same matrix is used all along the iteration process (10). This matrix has the same size as the original eigenproblem; thus, if the latter does not have a huge size, a single factorization can be computed once for all and used at each step.

We follow standard notational conventions in the field of matrix computations. Specifically, matrices are denoted by upper case italic letters, vectors by lower case italic letters, and scalars by lower case Greek letters or lower case italic if there is no confusion. Scalar- or vector-valued polynomials are denoted by lower case italic letters, and $\mathbb{C}^n[\lambda]$ denotes the space of polynomials with coefficients in $\mathbb{C}^n$. The 2-norm of a vector or a matrix is denoted $||\,.\,||_2$.

This paper is organized as follows. Section 2 discusses the convergence of the derivatives to a generalized eigenvector. In section 3, the convergence is extended to a block variant where several approximate eigenpairs are computed simultaneously. Section 4 describes a partial orthogonalization algorithm for generating a low dimensional nonlinear eigenproblem which can be solved by the standard methods mentioned above. This algorithm is probably not optimal, and hopefully improvements will be brought in the next future. Finally, in section 5 we report a few numerical experiments illustrating the convergence properties and the performances of the different methods.

2. Convergence of a single approximate eigenpair.

In this section, first we expose the relationship between the generalized eigenpairs of $A(\lambda)$ and the partial fraction decompositions of the resolvent $R(\lambda) = A(\lambda)^{-1}$ and the *generating function* $x(\lambda) = R(\lambda)b$, where b is a given vector. In the case $A(\lambda) = A - \lambda I$, it reduces to classical relations from the spectral theory of linear operators [1], [9]. Next we prove that the derivatives of $x(\lambda)$ converge to a generalized eigenvector of $A(\lambda)$, and we extend this result to some other sequences constructed from the derivatives of $x(\lambda)$.

2.1. General framework.

Let $A(\lambda)$ be a holomorphic $n \times n$ matrix-valued function defined on a domain $\Omega \subset \mathbb{C}$, such that $\det A(\lambda)$ is not identically zero. For a given $\alpha \in \Omega$ such that $A(\alpha)$ is invertible, we seek the nearest values λ for which the matrix $A(\lambda)$ is not invertible, and corresponding nontrivial vectors v such that

$$A(\lambda)v = 0. \tag{1}$$

Such a complex number λ is called a generalized eigenvalue, and the vector v is called a generalized eigenvector [16], [20].

By possibly using a similarity transformation of the complex plane, without any loss of generality, we can assume that $\alpha = 0$ and that $A(\lambda)$ is holomorphic on a neighborhood of the closed unit disk $\overline{D}(0, 1)$. Hence there are finitely many generalized eigenvalues within the open disk $D(0,1)$, which can be increasingly ordered

$$0 < |\lambda_0| \leq |\lambda_1| \leq \cdots \leq |\lambda_s| < 1.$$

For $\lambda \in D(0,1)\backslash \{\lambda_0, \lambda_1, \ldots, \lambda_s\}$, we define the resolvent

$$R(\lambda) = A(\lambda)^{-1},$$

which is meromorphic on the disk $D(0,1)$ and can be decomposed into partial fractions

$$R(\lambda) = \sum_{i=0}^{s} \sum_{j=1}^{j(i)} \frac{S_{i,j}}{(\lambda - \lambda_i)^j} + H(\lambda), \qquad S_{i,j} \in \mathbb{C}^{n \times n}, \tag{2}$$

where the matrices $S_{i,j(i)}$ are nonzero and the $n \times n$ matrix-valued function $H(\lambda)$ is holomorphic on $D(0,1)$. As usual, the eigenvalue λ_i is said to be *simple* if the corresponding null space of $A(\lambda_i)$ has dimension 1 and $j(i) = 1$, *semisimple* if this dimension is greater than 1 and $j(i) = 1$, and *defective* if $j(i) > 1$. The resolvent also reads $R(\lambda) = \operatorname{adj} A(\lambda)/\det A(\lambda)$, where $\operatorname{adj} A(\lambda)$ is the adjugate matrix of $A(\lambda)$ (i.e., the transposed cofactors matrix), and $\operatorname{adj} A(\lambda) = 0$ iff $\operatorname{rank} A(\lambda) \leq n-2$. Hence, $j(i) \leq \operatorname{val}(\lambda_i)$, which denotes the valuation of $\det A(\lambda)$ at λ_i, and $j(i) = \operatorname{val}(\lambda_i)$ iff $\operatorname{rank} A(\lambda_i) = n-1$. Particularly, λ_i is a simple eigenvalue of $A(\lambda)$ iff λ_i is a simple zero of $\det A(\lambda)$. In the case of an ordinary eigenvalue problem $A(\lambda) = A - \lambda I$, the number $j(i)$ is the size of the largest Jordan block associated with the eigenvalue λ_i [9].

The decomposition (2) is related to the generalized eigenvectors of (1) in the following way.

PROPOSITION 2.1. *Let $b \in \mathbb{C}^n$ and $0 \leq i \leq s$. Assume that $\{j;\ S_{i,j}b \neq 0\}$ is nonempty and let $j'(i) = \max\{j;\ S_{i,j}b \neq 0\}$. Then $S_{i,j'(i)}b$ is a generalized eigenvector associated with λ_i :*

$$A(\lambda_i) S_{i,j'(i)}\, b = 0. \tag{3}$$

Proof. The coefficient of $1/(\lambda - \lambda_i)^{j'(i)}$ in the partial fraction decomposition of $b = A(\lambda)R(\lambda)b$ is 0. □

Remark. The projections properties associated with the resolvent $R(\lambda)$ through the Dunford–Taylor integral [9], [16] are lost when $A(\lambda)$ does not have the form $A - \lambda I$. Unlike the case of an ordinary eigenvalue problem, when the eigenvalue λ_i is not simple, there may exist some $v \in \mathbb{C}^n$ with $A(\lambda_i)v = 0$, although there is no $b \in \mathbb{C}^n$ such that $v = S_{i,j'(i)}b$. This is illustrated by the following example:

$$A(\lambda) = \begin{pmatrix} \lambda - \lambda_0 & 0 \\ 0 & (\lambda - \lambda_0)^2 \end{pmatrix}, \quad v = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

Hence such vectors v cannot be retrieved from the partial fraction decomposition of the functions $R(\lambda)b$, $b \in \mathbb{C}^n$, and we will fail to find them in this way. The other vectors $S_{i,j}b$ for $j < j'(i)$ are not necessarily eigenvectors; however, it can be shown that they satisfy the relations

$$\frac{A^{(m)}(\lambda_i)}{m!} S_{i,j'(i)} b + \frac{A^{(m-1)}(\lambda_i)}{(m-1)!} S_{i,j'(i)-1} b + \cdots + A(\lambda_i) S_{i,j'(i)-m} b = 0, \quad 0 \leq m \leq j'(i) - 1.$$

In the case of an ordinary eigenvalue problem $A(\lambda) = A - \lambda I$, these relations read

$$(A - \lambda_i I)^m S_{i,j'(i)-m+1} b = 0, \quad 1 \leq m \leq j'(i),$$

where the Jordan decomposition can be recognized.

2.2. The generating function. Let $b \in \mathbb{C}^n$ be a nontrivial vector. The relationship (3) between the generalized eigenpairs of $A(\lambda)$ and the partial fraction decomposition of the resolvent (2) leads to introducing the *generating function* $x(\lambda)$ as the meromorphic vector-valued function defined by

$$A(\lambda)x(\lambda) = b, \qquad \lambda \in D(0,1)\setminus\{\lambda_0, \lambda_1, \ldots, \lambda_s\}\,. \tag{4}$$

If we multiply (2) on the right side by b, we obtain the partial fraction decomposition of $x(\lambda)$:

$$x(\lambda) = \sum_{i=0}^{s}\sum_{j=1}^{j(i)} \frac{v_{i,j}}{(\lambda-\lambda_i)^j} + h(\lambda), \qquad v_{i,j} \in \mathbb{C}^n, \tag{5}$$

where $v_{i,j} = S_{i,j}b$, and $h(\lambda) = H(\lambda)b$ is holomorphic on $D(0,1)$. Obviously, λ_i is a pole of $x(\lambda)$ iff at least one of the $S_{i,j}b$ is nonzero. The following proposition gives another characterization for λ_i being a pole of $x(\lambda)$.

PROPOSITION 2.2. *Let $b \in \mathbb{C}^n$ and $0 \le i \le s$. The generalized eigenvalue λ_i is a pole of $x(\lambda) = R(\lambda)b$ iff the linear system*

$$A(\lambda)p(\lambda) - b = O(\lambda-\lambda_i)^{j(i)} \tag{6}$$

has no solution $p \in \mathbb{C}^n[\lambda]$. In this case, for $j'(i) = \max\{j;\ S_{i,j}b \neq 0\}$, the vector $v_{i,j'(i)}$ is a generalized eigenvector of $A(\lambda)$ associated with λ_i :

$$A(\lambda_i)v_{i,j'(i)} = 0. \tag{7}$$

Proof. If (6) has a solution $p(\lambda)$, then multiplying both sides by $R(\lambda)$ yields near λ_i

$$p(\lambda) - x(\lambda) = O(1);$$

thus λ_i is not a pole of $x(\lambda)$. Conversely, if λ_i is not a pole of $x(\lambda)$, then the Taylor polynomial $p(\lambda) = \sum_{k=0}^{j(i)-1} x^{(k)}(\lambda_i)(\lambda-\lambda_i)^k/k!$ is a solution to (6). Equation (7) was derived in Proposition 2.1. □

The practical interest of this proposition is the following.

COROLLARY 2.3. *If b does not belong to the range of $A(\lambda_i)$, then λ_i is a pole of $x(\lambda)$.*

Proof. If b does not belong to the range of $A(\lambda_i)$, then (6) has no solution; thus λ_i is a pole of $x(\lambda)$. □

2.3. Computation of the derivatives. The power series expansions of the functions $A(\lambda)$ and $x(\lambda)$ are denoted by

$$A(\lambda) = \sum_{k\ge 0} A_k\lambda^k, \qquad A_k \in \mathbb{C}^{n\times n}, \tag{8}$$

$$x(\lambda) = \sum_{k\ge 0} x_k\lambda^k, \qquad x_k \in \mathbb{C}^n. \tag{9}$$

The generalized eigenvectors of $A(\lambda)$ will be approximated by linear combinations of the vectors x_k, which is the reason why we call $x(\lambda)$ the generating function. As the kth order derivative at zero of $x(\lambda)$ reads $x^{(k)}(0) = k!\,x_k$, and as we are more interested in its direction than its length, we also use the term "derivative" for x_k.

We assume that the matrices A_k are known. This happens in many finite-element implementations where they are assembled separately. They can also be obtained from the automatic differentiation of the function which computes the matrix $A(\lambda)$. The coefficient of λ^k in the power series expansion of $A(\lambda)x(\lambda)$ is $\sum_{j=0}^{k} A_j x_{k-j}$ and $A(\lambda)x(\lambda) = b$; thus $A_0 x_0 = b$ and $\sum_{j=0}^{k} A_j x_{k-j} = 0$ for $k \geq 1$. Hence the computation of the vectors x_k can be achieved in the following way.

Algorithm 1. **Generalized inverse iteration.**

```
1. Start:
    • choose an initial nontrivial vector $b$.
    • compute $x_0$ solution to $A_0 x_0 = b$.
2. Iterate: for $k = 1, 2, \ldots$, compute $x_k$ solution to
    (10)    $A_0 x_k = -\sum_{j=1}^{k} A_j x_{k-j}.$
```

All these systems have the same matrix A_0; thus a single factorization is needed for computing the wanted derivatives. In the case of a polynomial eigenvalue problem, (10) reduces to the inverse iteration method described by Peters and Wilkinson in [10], except for the normalization of the vectors x_k. Observe that unlike the linear case, some x_k may be zero; thus an appropriate change of variable $\lambda = \rho\lambda'$ is preferable to normalization. In this case, (10) becomes $A_0 x'_k = -\sum_{j=1}^{k} A'_j x'_{k-j}$ with $A'_j = \rho^j A_j$ and $x'_k = \rho^k x_k$.

2.4. Convergence of the derivatives. The next theorem states the fundamental property of the vectors x_k, that is, the derivatives of $x(\lambda)$ converge to a generalized eigenvector of the matrix $A(\lambda)$. For convenience here we suppose that b is chosen in such a way that $v_{i,j(i)} = S_{i,j(i)} b \neq 0$ for $0 \leq i \leq \ s$. This assumption is not very restrictive in the sense that the matrices $S_{i,j(i)}$ are nonzero and the complementary set of $\cup_{i=0}^{s} \ker S_{i,j(i)}$ is open and dense in $\mathbb{C}^n$. We suppose also $s \geq 1$.

Theorem 2.4. *Assume that* $|\lambda_0| < |\lambda_1|$. *There exists an integer* k_0 *such that* $x_k \neq 0$ *for* $k \geq k_0$. *Let*

$$z_k = \frac{x_k}{||x_k||_2}, \quad k \geq k_0.$$

The sequence $(|\lambda_0|/\overline{\lambda_0})^k z_k$ *converges to the generalized eigenvector* $\gamma_0 v_{0,j(0)}$ *of problem* (1), *where* γ_0 *is a nonzero complex number. If* $v_{0,j} = 0$ *for all* $j < j(0)$, *then*

$$\left(\frac{|\lambda_0|}{\overline{\lambda_0}}\right)^k z_k - \gamma_0 v_{0,j(0)} = O\left(k^{\tau - j(0)} \left|\frac{\lambda_0}{\lambda_1}\right|^k\right),$$

where $\tau = \max\{j(i);\ |\lambda_i| = |\lambda_1|\}$ *is the largest order of the next poles with equal moduli. Otherwise*

$$\left(\frac{|\lambda_0|}{\overline{\lambda_0}}\right)^k z_k - \gamma_0 v_{0,j(0)} = O\left(\frac{1}{k^{j(0)-\tau'}}\right),$$

where $\tau' = \max\{j;\ v_{0,j} \neq 0,\ j < j(0)\}$. *If* $j(0) = 1$, *then for all indices* l *such that the* l*th coordinate* $(v_{0,j(0)})_l$ *of the vector* $v_{0,j(0)}$ *is nonzero, one has*

$$\frac{(x_k)_l}{(x_{k+1})_l} - \lambda_0 = O\left(k^{\tau-1} \left|\frac{\lambda_0}{\lambda_1}\right|^k\right),$$

otherwise

$$\frac{(x_k)_l}{(x_{k+1})_l} - \lambda_0 = O\left(\frac{1}{k}\right).$$

Proof. The function $h(\lambda)$ in (5) is holomorphic on $D(0,1)$ and $|\lambda_1| < 1$, thus there exists a positive number c such that the coefficients h_k in $h(\lambda) = \sum_{k\geq 0} h_k \lambda^k$ satisfy

$$|h_k| \leq \frac{c}{|\lambda_1|^k} \quad \text{for all } k \geq 0.$$

Taking the derivative of order k in (5), we obtain

$$x_k = \sum_{i=0}^{s} \sum_{j=1}^{j(i)} \frac{(j+k-1)(j+k-2)\cdots(k+1)}{(j-1)!(-1)^j {\lambda_i}^{j+k}} v_{i,j} + h_k, \tag{11}$$

with the convention $(j+k-1)(j+k-2)\cdots(k+1) = 1$ if $j = 1$. If $v_{0,j} = 0$ for all $j < j(0)$, using $|h_k| \leq c/|\lambda_1|^k$, we get

$$x_k = \frac{(j(0)+k-1)\cdots(k+1)}{(j(0)-1)!(-1)^{j(0)} {\lambda_0}^{j(0)+k}} v_{0,j(0)} + O\left(\frac{k^{\tau-1}}{|\lambda_1|^k}\right). \tag{12}$$

For k sufficiently large, x_k is nonzero and

$$z_k = \frac{(-1)^{j(0)} |\lambda_0|^{j(0)+k}}{{\lambda_0}^{j(0)+k} \; ||v_{0,j(0)}||_2} v_{0,j(0)} + O\left(k^{\tau - j(0)} \left|\frac{\lambda_0}{\lambda_1}\right|^k\right),$$

which gives the result with $\gamma_0 = (-1)^{j(0)} |\lambda_0|^{j(0)} / ({\lambda_0}^{j(0)} \; ||v_{0,j(0)}||_2)$. If there is a $v_{0,\tau'} \neq 0$ with $\tau' < j(0)$ and $v_{0,j} = 0$ for $\tau' < j < j(0)$, then we deduce from (11) that

$$x_k = \frac{(j(0)+k-1)\cdots(k+1)}{(j(0)-1)!(-1)^{j(0)} {\lambda_0}^{j(0)+k}} v_{0,j(0)} + O\left(\frac{(\tau'+k-1)\cdots(k+1)}{(\tau'-1)! \, |\lambda_0|^{\tau'+k}}\right),$$

and

$$z_k = \frac{(-1)^{j(0)} |\lambda_0|^{j(0)+k}}{{\lambda_0}^{j(0)+k} \; ||v_{0,j(0)}||_2} v_{0,j(0)} + O\left(\frac{1}{k^{j(0)-\tau'}}\right).$$

The convergence of the eigenvalues is obtained from (12) in a similar way. □

2.5. Generalization. The previous result can be generalized to other eigenpairs by using an elimination technique. Although the sequences of vectors which are considered here are not computable (unless the eigenvalues are known), their properties will be useful for proving the simultaneous convergence of several eigenpairs in the next section.

For given m and d, $0 \leq m < d \leq s$, let $p(\lambda) = \sum_{k=0}^{\sigma} p_k \lambda^k$ be the Hermite interpolation of degree $\sigma = j(0) + \cdots + j(m-1) + j(m+1) + \cdots + j(d-1)$ satisfying

$$p(\lambda_m) = 1, \tag{13}$$

$$p^{(k)}(\lambda_i) = 0, \quad 0 \leq k \leq j(i)-1, \quad 0 \leq i \leq m-1, \quad m+1 \leq i \leq d-1. \tag{14}$$

Instead of the sequence x_k, we now consider the vectors y_k associated with the power series expansion of the function

$$y(\lambda) = p(\lambda)x(\lambda) = \sum_{k\geq 0} y_k \lambda^k.$$

Notice that each vector y_k is a linear combination of the vectors $x_{k-\sigma},\ x_{k-\sigma+1}, \dots,\ x_k$:

$$y_k = \sum_{i=0}^{\min(k,\sigma)} p_i\ x_{k-i}. \tag{15}$$

More generally, we have the following elementary lemma, which will be used several times.

LEMMA 2.5. *Let $q \in \mathbb{C}[\lambda]$ be a polynomial such that $q(0) \neq 0$, and consider the function $u(\lambda)$ defined by*

$$u(\lambda) = q(\lambda)x(\lambda) = \sum_{k\geq 0} u_k \lambda^k.$$

Then for all $k \geq 0$, one has

$$\operatorname{span}\{x_0,\ x_1, \dots,\ x_k\} = \operatorname{span}\{u_0,\ u_1, \dots,\ u_k\},$$

and the vectors $\{u_0,\ u_1, \dots,\ u_k\}$ are linearly independent iff the vectors $\{x_0,\ x_1, \dots,\ x_k\}$ are linearly independent.

The extension of Theorem 2.4 is the following.

PROPOSITION 2.6. *Assume that $|\lambda_m| < |\lambda_d|$. There exists an integer k_0 such that $y_k \neq 0$ for $k \geq k_0$. Let*

$$z_k = \frac{y_k}{||y_k||_2}, \quad k \geq k_0.$$

The sequence $(|\lambda_m|/\overline{\lambda_m})^k z_k$ converges to the generalized eigenvector $\gamma_m v_{m,j(m)}$ of problem (1), with $\gamma_m = \frac{|\lambda_m|^{j(m)}(-1)^{j(m)}}{{\lambda_m}^{j(m)}\ ||v_{m,j(m)}||_2}$. If $v_{m,j} = 0$ for all $j < j(m)$, then

$$\left(\frac{|\lambda_m|}{\overline{\lambda_m}}\right)^k z_k - \gamma_m v_{m,j(m)} = O\left(k^{\tau - j(m)} \left|\frac{\lambda_m}{\lambda_d}\right|^k\right),$$

where $\tau = \max\{j(i);\ |\lambda_i| = |\lambda_d|\}$. Otherwise

$$\left(\frac{|\lambda_m|}{\overline{\lambda_m}}\right)^k z_k - \gamma_m v_{m,j(m)} = O\left(\frac{1}{k^{j(m)-\tau'}}\right),$$

where $\tau' = \max\{j;\ v_{m,j} \neq 0,\ j < j(m)\}$. If $j(m) = 1$, then for all indices l such that the lth coordinate $(v_{m,j(m)})_l$ of the vector $v_{m,j(m)}$ is nonzero, one has

$$\frac{(y_k)_l}{(y_{k+1})_l} - \lambda_m = O\left(k^{\tau-1}\left|\frac{\lambda_m}{\lambda_d}\right|^k\right);$$

otherwise

$$\frac{(y_k)_l}{(y_{k+1})_l} - \lambda_m = O\left(\frac{1}{k}\right).$$

Proof. At each point λ_i, the polynomial $p(\lambda)$ can be written

$$p(\lambda) = \sum_{k=0}^{\sigma} p_k(\lambda_i)(\lambda - \lambda_i)^k,$$

where $p_k(\lambda_i) = p^{(k)}(\lambda_i)/k!$; thus the partial fraction decomposition of $y(\lambda)$ is related to the decomposition (5) of $x(\lambda)$ by

$$\begin{aligned} y(\lambda) &= \sum_{i=0}^{s} \sum_{j=1}^{j(i)} \sum_{k=0}^{\sigma} \frac{p_k(\lambda_i)}{(\lambda - \lambda_i)^{j-k}} v_{i,j} + h(\lambda) \\ &= \sum_{i=0}^{s} \sum_{l=1}^{j(i)} \left(\sum_{j=l}^{j(i)} p_{j-l}(\lambda_i) v_{i,j} \right) \frac{1}{(\lambda - \lambda_i)^l} + g(\lambda), \end{aligned}$$

where the function $g(\lambda)$ is holomorphic on $D(0,1)$ and involves $h(\lambda)$ and some other contributions of the form $(\lambda - \lambda_i)^l u,\ l \geq 0,\ u \in \mathbb{C}^n$. Due to the definition (13), (14) of $p(\lambda)$, we obtain

$$\begin{aligned} y(\lambda) &= \sum_{l=1}^{j(m)} \frac{w_{m,l}}{(\lambda - \lambda_m)^l} + \sum_{i=d}^{s} \sum_{l=1}^{j(i)} \frac{w_{i,l}}{(\lambda - \lambda_i)^l} + g(\lambda), \\ w_{i,l} &= \sum_{j=l}^{j(i)} p_{j-l}(\lambda_i) v_{i,j}, \quad w_{i,m,j(m)} = v_{m,j(m)}, \end{aligned}$$

and the rest of the proof runs as in Theorem 2.4. □

3. Convergence of several approximate eigenpairs. In this section we generalize the previous results into a block variant, which allows several eigenpairs to be computed simultaneously. Although the resulting algorithm is in practice less effective than the one presented in section 4, its convergence properties provide insight into the projection process for a nonlinear eigenproblem. First we describe a sequence of approximate eigenproblems and the associated algorithm. Next, in order to prepare the convergence study, we introduce a reduced eigenproblem which is not computable but will appear as the limit of the approximate eigenproblems. Finally we prove the simultaneous convergence of the approximate eigenpairs by comparing the approximate eigenproblems with the reduced eigenproblem.

For the sake of clarity, we simplify the problem by supposing from now on that all the generalized eigenvalues of problem (1) within the disk $D(0,1)$ are simple. The case of semisimple or defective eigenvalues can be treated in a similar way, but the notation becomes more complicated. Such an assumption is usual in the case of an ordinary eigenproblem because the set of matrices having only simple eigenvalues is dense and open in $\mathbb{C}^{n \times n}$ [1]. This holds also for polynomial eigenproblems of degree $m \geq 1$, because the discriminant of $\det A(\lambda)$ is not identically zero on $\mathbb{C}^{(m+1) \times n \times n}$, and the set where $\det A(\lambda) \in \mathbb{C}[\lambda]$ has only simple roots is dense and open in $\mathbb{C}^{(m+1) \times n \times n}$. In the general case it remains true in the following way. On the one hand, if the matrix $A(\lambda)$ has semisimple or defective eigenvalues in $D(0,1)$, it can be approximated by truncating the series (8), which brings us back to the polynomial case. Hence for all $\varepsilon > 0$, there exists a polynomial matrix $P(\lambda)$ having only simple eigenvalues in $D(0,1)$ such that $\sup_{\lambda \in \overline{D}(0,1)} ||A(\lambda) - P(\lambda)||_2 < \varepsilon$. On the other hand, if the zeros of $\det A(\lambda)$

in $D(0,1)$ are simple, then for $E(\lambda)$ holomorphic on Ω and $\sup_{\lambda\in\overline{D}(0,1)} ||E(\lambda)||_2$ sufficiently small, the zeros of $\det(A+E)(\lambda)$ in $D(0,1)$ remain simple.

We suppose also that $b \notin \operatorname{ran} A(\lambda_i)$ for $0 \leq i \leq s$. It follows from Corollary 2.3 that the function $x(\lambda)$ reads

$$x(\lambda) = \sum_{i=0}^{s} \frac{v_i}{\lambda - \lambda_i} + h(\lambda), \tag{16}$$

with $v_i = S_{i,1} b \neq 0$, and thanks to Proposition 2.2, we have

$$A(\lambda_i) v_i = 0,$$

where the null space of $A(\lambda_i)$ has dimension 1.

3.1. The approximate eigenproblems.

For a given d, $0 < d \leq s$, let

$$\mathcal{T}_k(x) = \operatorname{span}\{x_{k-d+1}, \ldots,\ x_k\}, \quad X_k = [x_{k-d+1}| \ldots |\ x_k], \quad k \geq d-1.$$

The approximate eigenproblems are defined as follows: find the generalized eigenpairs $\lambda, \widetilde{v}$ solution to

$$X_k^* A(\lambda) X_k \widetilde{v} = 0. \tag{17}$$

We will see that some of the eigenpairs of the original eigenproblem (1) can be retrieved from these eigenpairs by letting k tend to infinity. As problem (17) is supposed to be low dimensional, we assume that we can effectively solve it by transforming a polynomial problem into a linear problem [4], [14], [16] if $A(\lambda)$ is polynomial or by using Newton-based methods [10], [20], [19] in the general case. The algorithm is the following.

ALGORITHM 2. **Block variant.**

1. **Start:**
   - choose an initial nontrivial vector b, the search space dimension d, and the number of iterations k.
   - compute x_0 solution to $A_0 x_0 = b$.
2. **Loop:** for $i = 1, 2, \ldots, k$, compute x_i solution to

   $$A_0\, x_i = -\sum_{j=1}^{i} A_j\, x_{i-j}, \tag{18}$$

3. **Projection:** solve the problem

   $$X_k^* A(\lambda) X_k \widetilde{v} = 0, \tag{19}$$

   where $X_k = [x_{k-d+1}| \ldots |x_k]$.
4. **Check:** let $\lambda, \widetilde{v}$ be a solution to (19). Define

   $$v = X_k \widetilde{v}.$$

   If $||A(\lambda)v||_2/||v||_2$ is small, then λ, v is an approximate eigenpair of problem (1).

The reason why it must be checked whether $||A(\lambda)v||_2/||v||_2$ is small comes from the fact that the reduced eigenproblem (22) may have other eigenvalues than the original eigenproblem (cf. next section) and will be discussed after Theorem 3.2.

3.2. The reduced eigenproblem. Let

$$B(\lambda) = V^* A(\lambda) V, \quad V = [v_0 | \ldots | v_{d-1}]. \tag{20}$$

We suppose that $\det B(\lambda)$ is not identically 0, and we define the reduced generating function $\widetilde{y}(\lambda)$ by

$$B(\lambda)\widetilde{y}(\lambda) = V^* b. \tag{21}$$

The following elementary proposition states that under some reasonable assumptions, the eigenpairs we are seeking can be associated with some of the eigenpairs of the reduced eigenproblem

$$B(\lambda)\widetilde{w} = 0. \tag{22}$$

Recall that $b \notin \operatorname{ran} A(\lambda_i)$ and λ_i is a simple eigenvalue of $A(\lambda)$ for $0 \le i \le s$.

PROPOSITION 3.1. *Let $0 \le i \le d-1$. Assume that $\det B(\lambda)$ is not identically 0. If V^*b does not belong to the range of $B(\lambda_i)$, then λ_i is a pole of the function $\widetilde{y}(\lambda)$ and a generalized eigenvalue of $B(\lambda)$. If this eigenvalue is simple for $B(\lambda)$, then the coefficient $\widetilde{w}_i$ of $1/(\lambda - \lambda_i)$ in the partial fraction decomposition of $\widetilde{y}(\lambda)$ is related to the eigenvector v_i of $A(\lambda)$ by*

$$v_i = \beta_i V \widetilde{w}_i, \quad \beta_i \in \mathbb{C}. \tag{23}$$

Proof. The first part follows from Proposition 2.2. Define $e_i \in \mathbb{C}^d$ by $Ve_i = v_i$. If the eigenvalue λ_i is simple for $B(\lambda)$, then $\widetilde{w}_i$ is an eigenvector of $B(\lambda_i)$ (Proposition 2.2). We have both

$$B(\lambda_i)\widetilde{w}_i = 0,$$
$$B(\lambda_i)e_i = V^* A(\lambda_i) v_i = 0;$$

thus there exists $\beta_i \in \mathbb{C}$ such that $e_i = \beta_i \widetilde{w}_i$, which implies $v_i = Ve_i = \beta_i V \widetilde{w}_i$. □

Notice that when the matrix $A(\lambda_i)$ is self-adjoint, the condition $V^*b \notin \operatorname{ran} B(\lambda_i)$ is automatically fulfilled if the previous assumptions $b \notin \operatorname{ran} A(\lambda_i)$ and $\ker A(\lambda_i) \subset \operatorname{ran} V$ are satisfied, and it follows directly from the property $\operatorname{ran} A(\lambda_i) = (\ker A(\lambda_i))^{\perp}$.

Remark. The matrix $B(\lambda)$ may have a semisimple eigenvalue, although $A(\lambda)$ has only simple eigenvalues. Consider, for instance, the matrices

$$A(\lambda) = \begin{pmatrix} \lambda - 1 & 0 & 0 \\ 0 & (\lambda-1)(\lambda-2) & 1 \\ 0 & \lambda - 2 & 0 \end{pmatrix}, \quad V = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}.$$

Similarly, $B(\lambda)$ may have other eigenvalues than $A(\lambda)$ (and even smaller ones). Consider for instance the matrices

$$A(\lambda) = \begin{pmatrix} (\lambda-1)(\lambda-2) & \lambda - 3 & 0 \\ 0 & \lambda - 3 & 1 \\ \lambda - 2 & 0 & 0 \end{pmatrix}, \quad V = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}.$$

3.3. Convergence of the approximate eigenpairs. The following theorem states the convergence of d eigenpairs of the approximate eigenproblems (17) to the eigenpairs λ_i, v_i, $0 \le i \le d-1$ of the original eigenproblem (1).

The approximate generating function $\widetilde{x}_k(\lambda)$ is defined (if it exists) by

$$X_k^* A(\lambda) X_k \widetilde{x}_k(\lambda) = X_k^* b. \tag{24}$$

Theorem 3.2. *Assume that* $\det B(\lambda)$ *is not identically* 0, *that* V^*b *does not belong to the range of* $B(\lambda_i)$ *for* $0 \le i \le d-1$ *and that* λ_i *is a simple generalized eigenvalue both for the original eigenproblem* (1) *and the reduced eigenproblem* (22) *for* $0 \le i \le d-1$. *Then there exist an integer* k_0 *and positive numbers* ρ_i, $0 \le i \le d-1$, *such that for* $0 \le i \le d-1$ *and* $k \ge k_0$, *the approximate generating function* $\widetilde{x}_k(\lambda)$ *is well defined, it has exactly one simple pole in each disk* $D(\lambda_i,\, \rho_i)$, *and it reads*

$$\widetilde{x}_k(\lambda) = \sum_{i=0}^{d-1} \frac{\widetilde{v}_{ki}}{\lambda - \lambda_{ki}} + \widetilde{h}_k(\lambda),$$

where $\widetilde{h}_k(\lambda)$ *is meromorphic on* $D(0,1)$. *For* $0 \le i \le d-1$ *and* $k \ge k_0$, *the pole* λ_{ki} *is the unique generalized eigenvalue of the approximate eigenproblem* (17) *in the disk* $D(\lambda_i,\, \rho_i)$. *This eigenvalue is simple, and* $\widetilde{v}_{ki}$ *is a generalized eigenvector associated with this eigenvalue. For* $0 \le i \le d-1$, *the approximate eigenpairs* λ_{ki}, $\beta_i X_k \widetilde{v}_{ki}$ *converge linearly to the generalized eigenpair* λ_i, v_i *of problem* (1)*:*

$$\beta_i X_k \widetilde{v}_{ki} = v_i + O\left(\left|\frac{\lambda_{d-1}}{\lambda_d}\right|^k\right),$$

$$\beta_i \lambda_{ki} X_k \widetilde{v}_{ki} = \lambda_i v_i + O\left(\left|\frac{\lambda_{d-1}}{\lambda_d}\right|^k\right),$$

where β_i *was defined in Proposition* 3.1.

Proof. First we need to define another family of vectors spanning $\mathcal{T}_k(x)$, which combined with Proposition 2.6 leads to the uniform convergence on compact subsets of the approximate generating functions (expressed in the new basis) to the reduced generating function. Next we prove the convergence of the approximate eigenpairs via the Cauchy formula and Rouché's theorem [15].

For $0 \le i \le d-1$, let $p_i(\lambda) = \sum_{k=0}^{d-1} p_{i,k} \lambda^k$ be the Lagrange polynomial of degree $d-1$ such that

$$p_i(\lambda_j) = \delta_{ij}, \quad j = 0, \dots, d-1,$$

where δ_{ij} is the Kronecker symbol. Let

$$y_i(\lambda) = p_i(\lambda) x(\lambda).$$

The power series expansion of $y_i(\lambda)$ is denoted by

$$y_i(\lambda) = \sum_{k \ge 0} y_{i,k}\, \lambda^k, \quad y_{i,k} = \sum_{j=0}^{\min(d-1,k)} p_{i,j}\; x_{k-j}. \tag{25}$$

For $k \ge 0$, define $(z_{i,k})_{0 \le i \le d-1}$ by

$$z_{i,k} = \left(\frac{|\lambda_i|}{\overline{\lambda_i}}\right)^k \frac{y_{i,k}}{\gamma_i ||y_{i,k}||_2},$$

where γ_i was defined in Proposition 2.6. Let $V_k = [\, z_{0,k}|\dots|z_{d-1,k}]$. We have

$$V_k = X_k Q_k, \tag{26}$$

where $Q_k \in \mathbb{C}^{d\times d}$. We know from Proposition 2.6 that $z_{i,k} - v_i = O(|\lambda_i/\lambda_d|^k)$ for $0 \le i \le d-1$, thus

$$V_k - V = O\left(\left|\frac{\lambda_{d-1}}{\lambda_d}\right|^k\right). \tag{27}$$

Define the functions $\widetilde{y}_k(\lambda)$ by

$$V_k^* A(\lambda) V_k \widetilde{y}_k(\lambda) = V_k^* b. \tag{28}$$

As $\det V^* A(\lambda) V = \det B(\lambda)$ is not identically 0 and V_k converges to V, there exists an integer k_1 such that the meromorphic function $\widetilde{y}_k(\lambda)$ is well defined and the rank of V_k is exactly d for $k \ge k_1$. This implies that Q_k is invertible for $k \ge k_1$, and using (24) and (26), we obtain the following relations between $\widetilde{x}_k(\lambda)$ and $\widetilde{y}_k(\lambda)$:

$$\widetilde{x}_k(\lambda) = Q_k \widetilde{y}_k(\lambda), \quad X_k \widetilde{x}_k(\lambda) = V_k \widetilde{y}_k(\lambda), \quad k \ge k_1. \tag{29}$$

Due to the assumptions of the theorem, the generating function $\widetilde{y}(\lambda)$ defined in (21) reads

$$\widetilde{y}(\lambda) = \sum_{i=0}^{d-1} \frac{\widetilde{w}_i}{\lambda - \lambda_i} + \widetilde{g}(\lambda), \tag{30}$$

where the function $\widetilde{g}(\lambda)$ is meromorphic on $D(0,1)$ and has a finite number of poles which are in $D(0,1)\backslash\{\lambda_0,\dots,\lambda_{d-1}\}$. Let $\mathcal{O}$ be an open neighborhood of $\{\lambda_0,\dots,\lambda_{d-1}\}$ where $\widetilde{g}(\lambda)$ is holomorphic, and $\dot{\mathcal{O}} = \mathcal{O}\setminus\{\lambda_0,\dots,\lambda_{d-1}\}$. It follows from (20), (21) (27), and (28) that $\widetilde{y}_k(\lambda)$ converges uniformly to $\widetilde{y}(\lambda)$ on all compact subsets of $\dot{\mathcal{O}}$. More precisely, given a compact subset $\mathcal{K} \subset \dot{\mathcal{O}}$, we have

$$\sup_{\lambda\in\mathcal{K}} ||\widetilde{y}_k(\lambda) - \widetilde{y}(\lambda)||_2 = O\left(\left|\frac{\lambda_{d-1}}{\lambda_d}\right|^k\right). \tag{31}$$

Select a radius ρ_i such that $\overline{D}(\lambda_i,\rho_i) \subset \mathcal{O}$ and $\overline{D}(\lambda_i,\rho_i)$ contains no other 0 of $\det B(\lambda)$ than λ_i. When applying the Cauchy formula to (30), we have

$$\widetilde{w}_i = \frac{1}{2i\pi}\int_{\Gamma_i^+} \widetilde{y}(\lambda)d\lambda,$$
$$\lambda_i\widetilde{w}_i = \frac{1}{2i\pi}\int_{\Gamma_i^+} \lambda\widetilde{y}(\lambda)d\lambda,$$

where Γ_i^+ is the oriented boundary of $D(\lambda_i,\rho_i)$. Using (31), we obtain

$$\widetilde{w}_i - \frac{1}{2i\pi}\int_{\Gamma_i^+} \widetilde{y}_k(\lambda)d\lambda = O\left(\left|\frac{\lambda_{d-1}}{\lambda_d}\right|^k\right), \tag{32}$$

$$\lambda_i\widetilde{w}_i - \frac{1}{2i\pi}\int_{\Gamma_i^+} \lambda\widetilde{y}_k(\lambda)d\lambda = O\left(\left|\frac{\lambda_{d-1}}{\lambda_d}\right|^k\right). \tag{33}$$

Multiplying (32) and (33) on the left by V_k and using (29), we get

$$V_k\widetilde{w}_i - \frac{1}{2i\pi}\int_{\Gamma_i^+} X_k\widetilde{x}_k(\lambda)d\lambda = O\left(\left|\frac{\lambda_{d-1}}{\lambda_d}\right|^k\right),$$

$$\lambda_i V_k\widetilde{w}_i - \frac{1}{2i\pi}\int_{\Gamma_i^+} \lambda X_k\widetilde{x}_k(\lambda)d\lambda = O\left(\left|\frac{\lambda_{d-1}}{\lambda_d}\right|^k\right),$$

and finally with (23) and (27), we obtain

$$v_i - \frac{\beta_i}{2i\pi}\int_{\Gamma_i^+} X_k\widetilde{x}_k(\lambda)d\lambda = O\left(\left|\frac{\lambda_{d-1}}{\lambda_d}\right|^k\right), \tag{34}$$

$$\lambda_i v_i - \frac{\beta_i}{2i\pi}\int_{\Gamma_i^+} \lambda X_k\widetilde{x}_k(\lambda)d\lambda = O\left(\left|\frac{\lambda_{d-1}}{\lambda_d}\right|^k\right). \tag{35}$$

On the one hand, as $\widetilde{w}_i \neq 0$ in (32), there exists $k_2 \geq k_1$ such that the function $\widetilde{y}_k(\lambda)$ has at least one pole in each disk $D(\lambda_i, \rho_i)$ for $k \geq k_2$. On the other hand, $\det V_k^* A(\lambda) V_k$ converges uniformly to $\det B(\lambda)$ on the disk $D(0,1)$, the function $\det B(\lambda)$ has a unique and simple 0 in the disk $D(\lambda_i, \rho_i)$, and $\det B(\lambda) \neq 0$ for all $\lambda \in \Gamma_i$. It follows from Rouché's theorem that there exists $k_0 \geq k_2$ such that for all $k \geq k_0$, the function $\det V_k^* A(\lambda) V_k$ has a unique and simple zero in each disk $D(\lambda_i, \rho_i)$; i.e., problem (17) has a unique and simple eigenvalue λ_{ki} in each disk $D(\lambda_i, \rho_i)$. Hence the function $\widetilde{y}_k(\lambda)$ has one simple pole at most in each disk $D(\lambda_i, \rho_i)$. We conclude that the function $\widetilde{y}_k(\lambda)$ has a unique and simple pole λ_{ki} in each disk $D(\lambda_i, \rho_i)$ for all $k \geq k_0$, and using (29) we obtain the same conclusion for the function $\widetilde{x}_k(\lambda)$, which reads

$$\widetilde{x}_k(\lambda) = \sum_{i=0}^{d-1} \frac{\widetilde{v}_{ki}}{\lambda - \lambda_{ki}} + \widetilde{h}_k(\lambda),$$

where $\widetilde{h}_k(\lambda)$ is meromorphic on $D(0,1)$ and holomorphic on each $D(\lambda_i, \rho_i)$. It follows that (34) and (35) become the following for $k \geq k_0$:

$$\beta_i X_k \widetilde{v}_{ki} = v_i + O\left(\left|\frac{\lambda_{d-1}}{\lambda_d}\right|^k\right),$$

$$\beta_i \lambda_{ki} X_k \widetilde{v}_{ki} = \lambda_i v_i + O\left(\left|\frac{\lambda_{d-1}}{\lambda_d}\right|^k\right). \qquad \square$$

Remark. This result is weaker than in the case of a linear eigenvalue problem, where the error estimate on the eigenvalue λ_i is $O(|\lambda_i/\lambda_d|^k)$ instead of $O(|\lambda_{d-1}/\lambda_d|^k)$, and it is obtained from the classical error estimates of the Rayleigh–Ritz analysis [11] which cannot be used in the nonlinear case. However, numerical experiments seem to indicate that the latter stronger result remains true in the nonlinear case (see Figure 2).

As mentioned in section 3.2, the reduced problem (22) may have other eigenvalues in the disk $D(0,1)$ than λ_i, $0 \leq i \leq d-1$, say, λ'_i, $1 \leq i \leq s'$. These extra eigenvalues can belong to the set $\{\lambda_d, \lambda_{d+1}, \ldots, \lambda_s\}$ or not. Suppose that these eigenvalues are also simple, and associated with eigenvectors $\widetilde{w}'_i$ satisfying

$$\widetilde{y}(\lambda) = \sum_{i=0}^{d-1} \frac{\widetilde{w}_i}{\lambda - \lambda_i} + \sum_{i=1}^{s'} \frac{\widetilde{w}'_i}{\lambda - \lambda'_i} + \widetilde{f}(\lambda),$$

where $\widetilde{f}(\lambda)$ is holomorphic on $D(0,1)$. Select a radius $\rho < 1$ such that the open disk $D(0,\rho)$ contains the eigenvalues $\{\lambda_0, \lambda_1, \ldots, \lambda_{d-1}\}$ and $\{\lambda'_1, \lambda'_2, \ldots, \lambda'_{s'}\}$. With the same arguments as in the proof of Theorem 3.2, it can be shown that for k sufficiently large the approximate eigenproblem (17) will have exactly s' other eigenvalues λ'_{ki}, $1 \leq i \leq s'$ in $D(0,\rho)$ converging to λ'_i and associated with eigenvectors $\widetilde{v}'_{ki}$, satisfying

$$\widetilde{x}_k(\lambda) = \sum_{i=0}^{d-1} \frac{\widetilde{v}_{ki}}{\lambda - \lambda_{ki}} + \sum_{i=1}^{s'} \frac{\widetilde{v}'_{ki}}{\lambda - \lambda'_{ki}} + \widetilde{f}_k(\lambda),$$

where $\widetilde{f}_k(\lambda)$ is holomorphic on $D(0,1)$. It follows from $X_k\widetilde{x}_k(\lambda) = V_k\widetilde{y}_k(\lambda)$ (cf. (29)) that the vectors $X_k\widetilde{v}'_{ki}$, $1 \leq i \leq s'$ converge to the vectors $v'_i := V\widetilde{w}'_i$ with $V^*A(\lambda'_i)v'_i = 0$. Hence one can decide whether λ'_{ki}, $1 \leq i \leq s'$ is an approximate eigenvalue of the original problem by watching if $A(\lambda'_{ki})X_k\widetilde{v}'_{ki}$ vanishes. An example of such an extra eigenvalue can be seen in Figure 2, where six eigenvalues are found within a search space of dimension 5. Curiously, the chosen vector b has "seen" two eigenvalues quite far away from the origin.

4. A partial orthogonalization algorithm. Increasing space methods like Arnoldi's or Lanczos methods are often preferred to subspace iterations with a fixed dimension. A simple reason is that a Galerkin approximation is expected to be more accurate on spaces which contain the iterated subspaces. Besides, a subspace iteration must somehow be initialized by an increasing space technique. Here we proceed in a similar way. Instead of the spaces $\mathcal{T}_k(x) = \text{span}\,\{x_{k-d+1}, \ldots,\ x_k\}$, we work with the spaces

$$\mathcal{A}_k(x) = \text{span}\,\{x_0,\ x_1, \ldots,\ x_k\}\,.$$

Now it is well known that one has to care about the way the spaces are expanded, or else the dimension fails to increase after a few steps. This comes from the convergence of the vectors x_k and the round-off errors of the computer. The usual way to overcome this difficulty is to orthogonalize the vectors, as in Arnoldi's method. The algorithm exposed in this section does not perform a complete orthogonalization. At each space expansion, the incoming vector is made orthogonal to the previous ones, but these have to be modified in order to compute the next vector. This comes from the fact that the computation of x_k (10) involves all the other vectors x_i, $0 \leq i \leq k-1$, unlike an ordinary eigenvalue problem, where x_k depends only on x_{k-1}. Nevertheless the aim of allowing the dimension to increase seems to be achieved through this procedure. First we describe the basic scheme of the search space expansion, from which the algorithm will be next derived.

Let $q(\lambda) = \sum_{i=0}^{m} q_i\lambda^i$ be a polynomial with $q(0) \neq 0$ and let $y(\lambda) = q(\lambda)x(\lambda)$. As mentioned earlier in Lemma 2.5, we have $\mathcal{A}_k(x) = \mathcal{A}_k(y)$. The current basis of $\mathcal{A}_k(x)$ is supposed to be $\{y_0,\ y_1, \ldots,\ y_k\}$, and we aim to compute a basis of $\mathcal{A}_{k+1}(x)$. This could be done by using the relation $A(\lambda)y(\lambda) = q(\lambda)b$, that is, computing y_{k+1} solution to

$$A_0 y_{k+1} = -\sum_{j=1}^{k+1} A_j\, y_{k+1-j} + q_{k+1}b,$$

but we are faced with the difficulty that even if the vectors $\{y_0,\ y_1, \ldots,\ y_{k+1}\}$ are linearly independent if computed with exact arithmetic, they may be numerically dependent and the expansion fails. For this reason we define the vectors $\{z_0,\ z_1, \ldots,\ z_k\}$ and expand them in the following way.

Let $Y = [y_0|y_1| \ldots |\ y_{k-1}]$. A vector z_k is chosen in $\mathcal{A}_k(x)$, orthogonal to the vectors $\{y_0,\ y_1, \ldots,\ y_{k-1}\}$:

$$\begin{aligned} a &= -\left(Y^*Y\right)^{-1} Y^*\, y_k, \\ z_k &= \ y_k + Y\, a. \end{aligned} \tag{36}$$

Thus we have, with the numbering $a = (\alpha_0, \alpha_1, \ldots, \alpha_{k-1})^T$,

$$z_k = y_k + \alpha_{k-1} y_{k-1} + \cdots + \alpha_1 y_1 + \alpha_0 y_0.$$

Define next for $j = k-1, k-2, \ldots, 0$

$$z_j = y_j + \alpha_{k-1} y_{j-1} + \cdots + \alpha_{k-j} y_0, \tag{37}$$

and expand $\mathcal{A}_k(x)$ by adding the vector z_{k+1} obtained by solving the equation

$$A_0\, z_{k+1} = -\sum_{j=1}^{k+1} A_j\, z_{k+1-j}, \tag{38}$$

which is the same as the one defining x_{k+1} in (10) from the other x_j, $j < k+1$.

LEMMA 4.1. *Assume that the vectors* $\{x_0,\ x_1, \ldots,\ x_{k+1}\}$ *are linearly independent. Then the vectors* $\{z_0,\ z_1, \ldots,\ z_{k+1}\}$ *constructed above are also linearly independent and they satisfy*

$$\operatorname{span}\{z_0,\ z_1, \ldots,\ z_{k+1}\} = \mathcal{A}_{k+1}(x).$$

Proof. Consider the polynomial $p(\lambda) = \sum_{i=0}^{k+1} p_i \lambda^i$ with

$$\begin{aligned} p_0 &= 1, \\ p_i &= \alpha_{k-i}, \quad 1 \le i \le k, \\ p_{k+1} &= \frac{-1}{q_0} \sum_{i=0}^{k} p_i q_{k+1-i}. \end{aligned}$$

Define the function $z(\lambda)$ by

$$z(\lambda) = p(\lambda) y(\lambda) = p(\lambda) q(\lambda) x(\lambda).$$

By construction, the power series coefficients of the function $z(\lambda)$ up to the order $k+1$ are exactly the z_j defined by (36), (37), and (38). Moreover, we have $p(0)q(0) \neq 0$; thus the result follows from Lemma 2.5. □

We are now in a position to describe the partial orthogonalization algorithm for constructing the basis of $\mathcal{A}_k(x)$ which is used for the projecting process.

ALGORITHM 3. **Partial orthogonalization.**

1. **Start:**
   - choose an initial nontrivial vector b, and the search space dimension $k+1$.
   - compute x_0 solution to $A_0 x_0 = b$, and define $y_0^0 = x_0$.
2. **Loop:** for $i = 0, 1, \dots, k-1$, do:
   - expand the search space by solving
   $$A_0\, y_{i+1}^i = -\sum_{j=1}^{i+1} A_j\, y_{i+1-j}^i, \tag{39}$$
   - choose y_{i+1}^{i+1} in $\mathcal{A}_{i+1}(x)$, orthogonal to the vectors $\{y_0^i,\ y_1^i, \dots,\ y_i^i\}$ by computing
   $$\begin{aligned} a &= -\left(Y_i^* Y_i\right)^{-1} Y_i^*\, y_{i+1}^i,\\ y_{i+1}^{i+1} &= \, y_{i+1}^i + Y_i\, a, \end{aligned}$$
   where $Y_i = [y_0^i | y_1^i | \dots | y_i^i]$,
   - modify the other vectors: for $j = i, i-1, \dots, 0$, compute
   $$y_j^{i+1} = y_j^i + \alpha_i y_{j-1}^i + \cdots + \alpha_{i+1-j} y_0^i, \tag{40}$$
   with $a = (\alpha_0, \alpha_1, \dots, \alpha_i)^T$.
3. **Projection:** solve the problem
$$Y_k^* A(\lambda) Y_k \widetilde{w} = 0. \tag{41}$$
4. **Check:** let $\lambda, \widetilde{w}$ be a solution to (41). Define
$$v = Y_k \widetilde{w}. \tag{42}$$
If $||A(\lambda)v||_2/||v||_2$ is small, then λ, v is an approximate eigenpair of problem (1).

In the case of a linear eigenvalue problem $A(\lambda) = A - \lambda B$, the modification (40) is unnecessary because the vectors y_j^{i+1}, $0 \le j \le i$, are not used at the next step, and the algorithm reduces to the classical Arnoldi's method applied to $A^{-1}B$, except for the normalization of the vectors.

Remark. At the end of the ith step, only the vectors y_j^{i+1}, $0 \le j \le i+1$, need to be stored (notice that the vectors y_0^i do not change; they are all equal to x_0). However, if storage is not a problem, an alternative to this algorithm consists in using the orthogonal matrix $Z_k := [\, y_0^0/||y_0^0||_2 \,|\, y_1^1/||y_1^1||_2 \,|\, \dots \,|\, y_k^k/||y_k^k||_2 \,]$ in (41) and (42) instead of the matrix Y_k. It is also possible to compute the vectors y_{i+1}^{i+1} directly from the previous vectors y_j^j, $0 \le j \le i$ (that is, without storing the vectors y_j^i, $0 \le j < i$), which leads to a complete orthogonalization. The advantage of this strategy deserves to be investigated.

PROPOSITION 4.2. *Assume that the vectors $\{x_0,\ x_1, \dots,\ x_k\}$ are linearly independent. Then the vectors $\{y_0^k,\ y_1^k, \dots,\ y_k^k\}$ constructed above are also linearly independent and*
$$\operatorname{span}\{y_0^k,\ y_1^k, \dots,\ y_k^k\} = \mathcal{A}_k(x).$$

Proof. The result can be obtained by applying Lemma 4.1 at each step of the algorithm. □

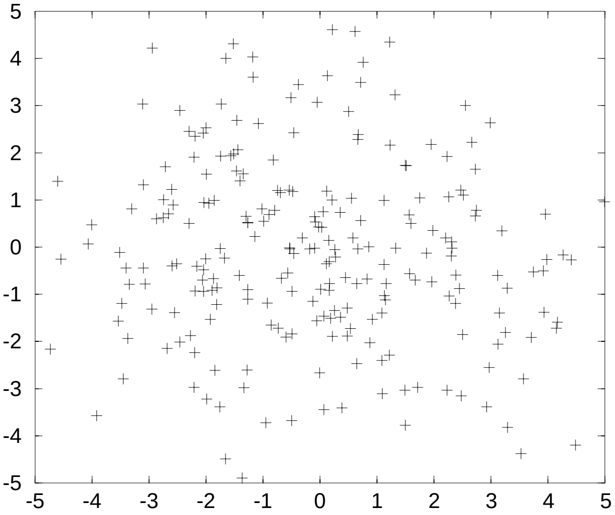


FIG. 1. *Generalized spectrum of* $A(\lambda)$.

5. A numerical example. In order to build confidence in Algorithms 2 and 3, we have experimented with them on a polynomial example which should be seen only as an illustration of the new approach. This example has been coded in MATLAB and executed on a DEC ALPHA workstation in about 15-decimal working precision. First we illustrate the convergence of the block variant (Algorithm 2). As one could expect, this method is not very stable. Next we compare the results of Algorithm 3 with the results obtained by using directly the original basis $\{x_0,\ x_1,\dots,\ x_k\}$, that is, without applying any orthogonalization. Finally we compare Algorithm 3 with Arnoldi's algorithm applied to the augmented linear eigenproblem (45).

We have searched the smallest eigenvalues of the cubic eigenproblem

$$A(\lambda)v = 0, \tag{43}$$

$$A(\lambda) = Q^*U(\lambda)Q, \tag{44}$$

where Q is a randomly chosen orthogonal 80×80 matrix, and $U(\lambda) = U_0 + U_1\lambda + U_2\lambda^2 + U_3\lambda^3$ is an 80×80 upper triangular matrix with (i,j) entries $u_{ij}(\lambda)$ satisfying $u_{ij}(\lambda) = 0$ if $j \geq i+2$. The roots of the polynomials $u_{ii}(\lambda)$ have been randomly chosen in the complex plane by using a centered normal distribution. The coefficients of the polynomials $u_{i\ i+1}(\lambda)$ have been randomly chosen on the real axis by also using a centered normal distribution. The generalized spectrum of $A(\lambda)$ is represented in Figure 1. As an initial vector b we took the vector with all coordinates equal to 1.

The approximate eigenproblems have been solved by using the QZ algorithm [13] in the following way. Let W be a matrix whose columns form a basis of the search space. The matrix $B(\lambda) = W^*A(\lambda)W$ reads

$$B(\lambda) = B_0 + B_1\lambda + B_2\lambda^2 + B_3\lambda^3,$$

with $B_j = W^*A_jW$, $A_j = Q^*U_jQ$. The determinant of $B(\lambda)$ is zero iff the determinant of the matrix $K + \lambda L$ is zero, where

$$K = \begin{pmatrix} B_0 & 0 & 0 \\ 0 & -I & 0 \\ 0 & 0 & -I \end{pmatrix}, \quad L = \begin{pmatrix} B_1 & B_2 & B_3 \\ I & 0 & 0 \\ 0 & I & 0 \end{pmatrix},$$

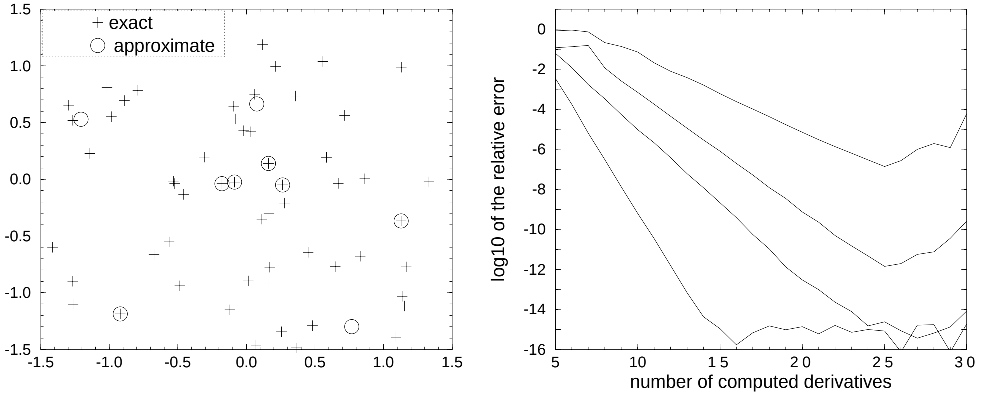


FIG. 2. *Convergence of the approximate eigenvalues with Algorithm* 2.

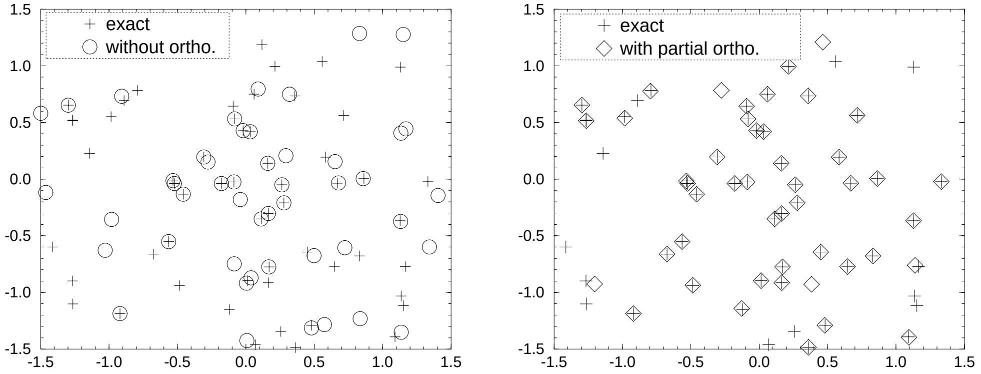


FIG. 3. *Approximate eigenvalues for* $k = 39$ *with Algorithm* 3.

and the QZ routine implemented in MATLAB is used for solving $\det(K + \lambda L) = 0$.

5.1. Results of Algorithm 2. The subspaces $\mathcal{T}_k(x)$ involved in the block variant have been chosen to be of dimension five. The results are given in Figure 2. On the left side, the approximate eigenvalues computed on $\mathcal{T}_{29}(x)$ are represented by circles, and the exact eigenvalues are represented by plus signs. On the right side is represented the relative error between the four smallest exact eigenvalues and the four approximate eigenvalues which are the closest to them. The instability of this method appears after 25 computations of derivatives.

5.2. Results of Algorithm 3. The performance of Algorithm 3 is shown in Figures 3 and 4. Figure 3 shows the approximate eigenvalues computed on the subspaces $\mathcal{A}_{39}(x)$. The circles correspond to the eigenvalues computed by directly using the basis $\{x_0,\ x_1,\ldots,\ x_{39}\}$, and the diamonds correspond to the eigenvalues computed by using Algorithm 3. The exact eigenvalues are still represented by plus signs. One can observe that Algorithm 3 allows more than 40 eigenvalues to be computed quite accurately within a search space of dimension 40.

Figure 4 shows the relative error between the seven smallest exact eigenvalues and the seven approximate eigenvalues which are the closest to them. The latter are computed by directly using the basis $\{x_0,\ x_1,\ldots,\ x_k\}\,, 4 \le k \le 39$, or by using

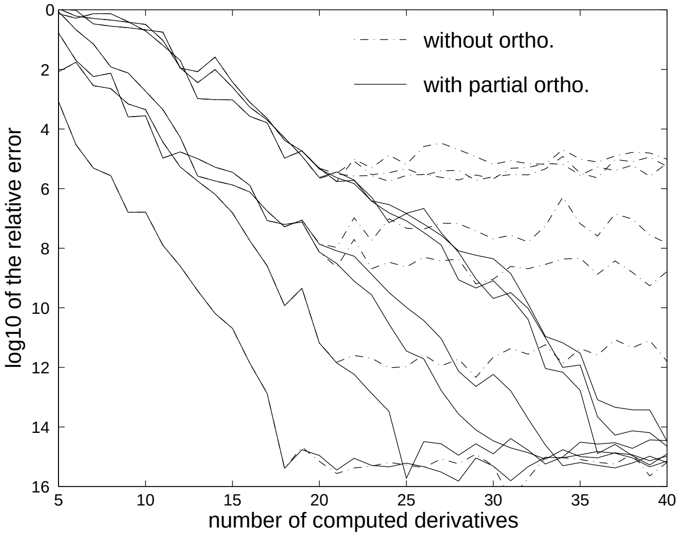


FIG. 4. *Relative errors on the seven smallest eigenvalues with Algorithm* 3.

Algorithm 3. One can observe the improving of stability brought by changing the basis of $\mathcal{A}_k(x)$.

5.3. Comparison with a classical method. If the matrices A_0 and A_3 have full rank, then the cubic eigenproblem (43) is equivalent to the augmented linear eigenproblem

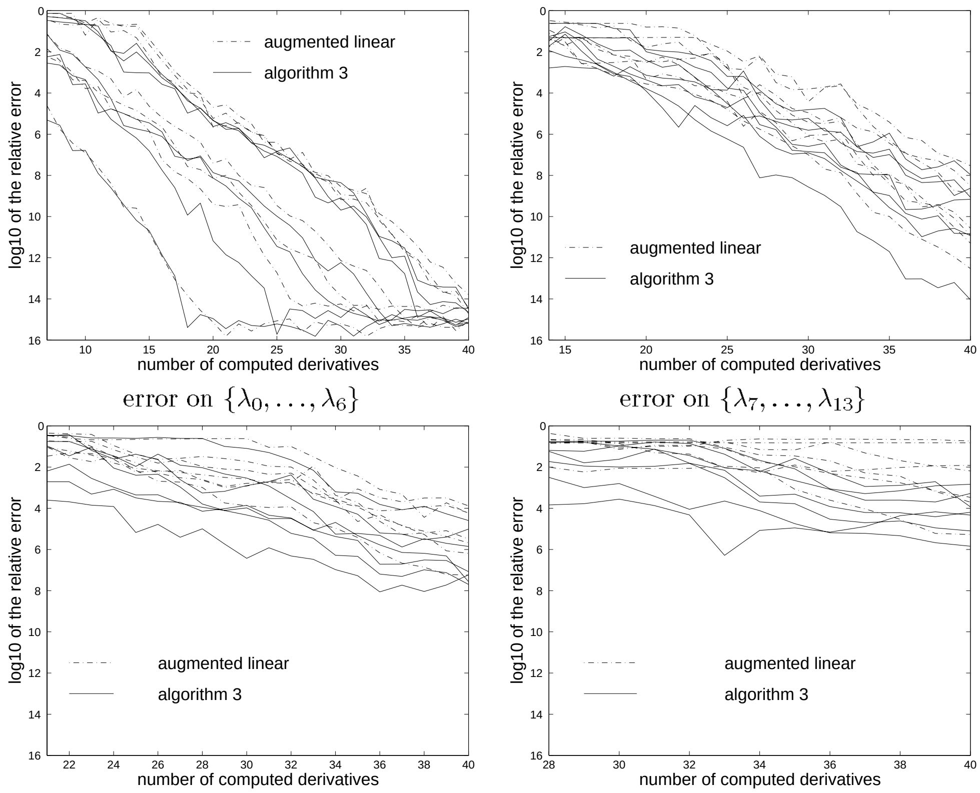
$$M^{-1}N\,v = \mu v, \quad \mu = \frac{1}{\lambda}, \tag{45}$$

with

$$M = \begin{pmatrix} A_0 & 0 & 0 \\ 0 & -I & 0 \\ 0 & 0 & -I \end{pmatrix}, \quad N = -\begin{pmatrix} A_1 & A_2 & A_3 \\ I & 0 & 0 \\ 0 & I & 0 \end{pmatrix}.$$

We have compared Algorithm 3 with Arnoldi's algorithm applied to (45), which uses the same LU factorization of the matrix A_0 as Algorithm 3 (39). Figure 5 shows the relative error on the 28 smallest eigenvalues. The comparison is made versus the number of computed derivatives, because if n is large, then in both methods most of the computational time is spent in solving a system of the form $A_0x = y$. Observe that in the case of our cubic eigenproblem, $k-1$ steps of Algorithm 3 provide $3k$ potential eigenvalues associated with an $n\times k$ search space, whereas Arnoldi's algorithm applied to (45) provides k potential eigenvalues associated with a $3n \times k$ search space. This may increase the number of accurate approximate eigenvalues computed by Algorithm 3 in the case where the eigenvectors associated with the smallest eigenvalues do not have full rank.

6. Final comments. The derivatives of the generating function can be considered very efficient tools for solving large nonlinear eigenproblems when a few eigenpairs are required. A thorough investigation of the error estimates remains to be performed, particularly in Algorithm 3. The difficulty comes from the fact that the projection



error on $\{\lambda_0, \ldots, \lambda_6\}$

error on $\{\lambda_7, \ldots, \lambda_{13}\}$

FIG. 5. *Comparison of Algorithm* 3 *with Arnoldi on the augmented linear problem.*

methods used in the linear case cannot be simply transposed to the nonlinear case. Finally, it would be interesting to incorporate some ideas of the Jacobi–Davidson method in the previous algorithm.

Acknowledgments. I thank the referees for their criticism and their useful suggestions, and I am very grateful to Prof. M. Masmoudi for the numerous and helpful discussions in the matter.

REFERENCES

[1] H. BAUMGÄRTEL, *Analytic Perturbation Theory for Matrices and Operators*, Birkhäuser-Verlag, Basel, Boston, MA, 1985.

[2] J. D. BELEY, C. BROUDISCOU, PH. GUILLAUME, M. MASMOUDI, AND F. THEVENON, *Application de la méthode des dérivées d'ordre elevé à l'optimisation des structures*, Revue Européenne Éléments Finis, (1996), pp. 537–567.

[3] C. BISCHOF AND A. GRIEWANK, *Computational differentiation and multidisciplinary design*, in Inverse Problems and Optimal Design in Industry, Philadelphia, PA, 1993, Teubner, Stuttgart, 1994, pp. 187–211.

[4] M. BORRI AND P. MANTEGAZZA, *Efficient solution of quadratic eigenproblems arising in dynamic analysis of structures*, Comput. Methods Appl. Mech. Engrg., 12 (1977), pp. 19–31.

[5] PH. GUILLAUME AND M. MASMOUDI, *Computation of high order derivatives in optimal shape design*, Numer. Math., 67 (1994), pp. 231–250.

[6] PH. GUILLAUME AND M. MASMOUDI, *Solution to the time-harmonic Maxwell's equations in a waveguide; use of higher-order derivatives for solving the discrete problem*, SIAM J. Numer. Anal., 34 (1997), pp. 1306–1330.

[7] J. C. Gilbert, G. Le Vey, and J. Masse, *La differentiation automatique de fonctions représentées par des programmes*, INRIA, Rapports de recherche 1557, Le Chesnay, France, 1991.

[8] A. Griewank, *On Automatic Differentiation*, in Mathematical Programming: Recent Developments and Applications, M. Iri and K. Tanabe, eds., Kluwer Academic Publishers, Dordrecht, 1989, pp. 83–108.

[9] T. Kato, *Perturbation Theory for Linear Operators*, Springer-Verlag, Berlin, 1980.

[10] G. Peters and J. H. Wilkinson, *Inverse iteration, ill-conditioned equations and Newton's methods*, SIAM Rev., 21 (1979), pp. 339–360.

[11] P. Lascaux and R. Théodor, *Analyse numérique matricielle appliquée à l'art de l'ingénieur*, Masson, Paris, Milan, Barcelone, 1994.

[12] J. Morgenstern, *How to compute fast a function and all its derivatives, a variation on the theorem of Baur-Strassen*, SIGACT News, 1985.

[13] C. B. Moler and G. W. Stewart, *An algorithm for generalized matrix eigenvalues problems*, SIAM J. Numer. Anal., 10 (1973), pp. 241–256.

[14] B. N. Parlett and H. C. Chen, *Use of an indefinite inner product for computing damped natural modes*, Technical report PAM-435, Center for Pure and Applied Mathematics, University of California at Berkeley, Berkeley, CA, 1988.

[15] W. Rudin, *Analyse réelle et complexe*, Masson, Paris, Milan, Barcelone, Bonn, 1992.

[16] Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, Algorithms Achit. Adv. Sci. Comput., Manchester University Press, Manchester, 1992.

[17] G. W. Stewart and J. G. Sun, *Matrix Perturbation Theory*, Academic Press, New York, 1990.

[18] A. Jennings, *Matrix Computations for Engineers and Scientists*, Wiley, New York, 1977.

[19] G. L. G. Sleijpen and H. A. van der Vorst, *A Jacobi–Davidson iteration method for linear eigenvalue problems*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 401–425.

[20] G. L. G. Sleijpen, A. G. L. Booten, D. R. Fokkema, and H. A. van der Vorst, *Jacobi–Davidson type methods for generalized eigenproblems and polynomial eigenproblems*, BIT, 36 (1996), pp. 595–633.

SIAM J. Matrix Anal. Appl.
Vol. 20, No. 3, pp. 596–605

ABSOLUTE SCHUR ALGEBRAS AND UNBOUNDED MATRICES*

PACHARA CHAISURIYA[†] AND SING-CHEONG ONG[‡]

Abstract. Let p, q, r be real numbers such that $p, q, r \geq 1$, and let $\mathfrak{B}$ be a Banach algebra. Let $\mathcal{B}(\ell^p, \ell^q)$ denote the set of all matrices which define bounded linear transformations from ℓ^p into ℓ^q. The set

$$\mathcal{S}^r(\mathfrak{B}) = \left\{ A = \left[a_{jk}\right] : a_{jk} \in \mathfrak{B} \text{ and } A^{[r]} = \left[\left\|a_{jk}\right\|^r\right] \in \mathcal{B}(\ell^p, \ell^q) \right\}$$

of infinite matrices over $\mathfrak{B}$, is shown to be a Banach algebra under the Schur product operation, and the norm $\|\!|A|\!\|_{p,q,r} = \|A^{[r]}\|^{1/r}$. For $r \geq 2$ and $\mathfrak{B} = \mathbb{C}$, the complex field, $\mathcal{S}^p = \mathcal{S}^p(\mathbb{C})$ contains the set $\mathcal{B}(\ell^p, \ell^q)$. For $r = 2$, $\mathcal{S}^2$ contains the bounded matrices $\mathcal{B}(\ell^p, \ell^q)$ as an ideal.

Key words. Schur product, Hadamard product, Schur multiplier norm

AMS subject classifications. Primary, 15A18; Secondary, 47A10

PII. S0895479897330005

1. Introduction and notation.

The *Schur product* (also known as *Hadamard product* or just *entrywise product*) of two matrices $A = [a_{jk}]$ and $B = [b_{jk}]$ of the same size, finite square or rectangular, or infinite bounded or unbounded (as linear transformation between ℓ^p spaces of pth power summable sequences with the standard basis), is defined by $A \bullet B = [a_{jk}b_{jk}]$; i.e., $A \bullet B$ is the matrix whose (j, k)-entry is the product of the (j, k)-entry of A and the (j, k)-entry of B. Areas such as matrix theory, complex analysis, operator theory, and operator algebras have made good use of results from the study of the Schur product and injected new problems in return. See [3, 1, 2, 4, 10, 11, 6, 5] for further reference to related literature. It is proved by Schur in [9] that the Schur product of two nonnegative (resp., positive) definite matrices is nonnegative (resp., positive) definite and that the ℓ^2-operator norm of the Schur product of two matrices is at most the product of their norms; i.e., the Schur product is submultiplicative with respect to operator norm on $\mathcal{B}(\ell^2)$, and hence the bounded matrices form a commutative Banach algebra under the usual sum, the Schur product, and the operator norm. Q. Stout studied the maximal ideal space structure of this algebra in his thesis [10]. L. Livshits [5] studied generalized Schur products of operator matrices. The product on the entries is the usual operator composition. G. Bennett [1] proved that all $\mathcal{B}(\ell^p, \ell^q)$, for $1 \leq p, q < \infty$, are also Banach algebras under the Schur product and operator norm on $\mathcal{B}(\ell^p, \ell^q)$.

From the results of Schur mentioned above, we see that the Schur product has some very desirable and pleasant properties which the usual product lacks. In this note we will unveil another pleasant aspect of this simple operation.

For a given matrix $A = [a_{jk}]$ (finite square or rectangular or infinite bounded or unbounded), and a positive real number r, we define the *absolute Schur rth power*

*Received by the editors September 8, 1997; accepted for publication (in revised form) by R. Brualdi March 20, 1998; published electronically March 2, 1999. This research was supported by a grant from the Ministry of University Affairs of Thailand.

http://www.siam.org/journals/simax/20-3/33000.html

[†] Department of Mathematics, Faculty of Science, Ramkhamhaeng University, Bangkok 10240, Thailand (pachara@ram1.ru.ac.th).

[‡] Department of Mathematics, Central Michigan University, Mount Pleasant, MI 48859. The research of this author was also supported by a Central Michigan University sabbatical leave and was done while visiting Ramkhamhaeng University (sing-cheong.ong@cmich.edu).

of A to be the matrix $A^{[r]} = [|a_{jk}|^r]$, for obvious reasons. Then we show that, for $1 \le r < \infty$, the set $\mathcal{S}^r$ of all matrices A such that $A^{[r]} \in \mathcal{B}(\ell^p, \ell^q)$ is a Banach algebra under the usual sum, the Schur product, and the norm $|||A|||_{p,q,r} = \|A^{[r]}\|_{p,q}^{1/r}$. For the case of $r = 2$, $\mathcal{S}^2$ contains the bounded matrices $\mathcal{B}(\ell^p, \ell^q)$ as an ideal. For $r \ge 2$, all $\mathcal{S}^r$ contain $\mathcal{B}(\ell^p, \ell^q)$ as a proper subset. We will work on the more general setting of matrices with entries from a fixed Banach algebra with identity. Since p and q will be fixed throughout, we will often suppress the subscripts and rely on the context to determine which one is intended.

It appears to us that the unbounded operators don't have much of the familiar and desirable structures. They are mostly dealt with individually. The pleasant structures of the $\mathcal{S}^p$ algebras and the multiplier algebra [1] may provide us with a reasonable way of studying the unbounded operators collectively as Schur algebras instead of individually.

This grew out of the simple Cauchy–Schwarz-type inequality proved in [7] and extended in [3, p. 340]. It has been observed by C.-K. Fong that the corresponding Hölder-type inequality can be proved by obvious modifications. We are grateful to Fong for this observation and for raising related questions.

2. Hölder and Minkowski-type inequalities.

For a scalar matrix $A = [a_{jk}]$, $\overline{A}$ denotes the matrix whose entries are the conjugates of the corresponding entries of A, i.e., $\overline{A} = [\overline{a_{jk}}]$. We state the following version of the Cauchy–Schwarz inequality [7, 3] before getting into the more general situation.

THEOREM 2.1. *For finite matrices A and B,*

$$\|A \bullet B\|_{2,2} \le \left\|A \bullet \overline{A}\right\|_{2,2}^{1/2} \left\|B \bullet \overline{B}\right\|_{2,2}^{1/2},$$

where the norm $\|\cdot\|_{2,2}$ is the operator norm induced by the Euclidean norm on the underlying space.

We note here that the finiteness of the size, and even boundedness, requirements on both A and B can be dropped, as we will see below. We will fix a Banach algebra $\mathfrak{B}$ with identity e.

DEFINITION 2.2. *For a given matrix $A = [a_{jk}]$ (finite or infinite) with entries from the Banach algebra $\mathfrak{B}$, and for any positive real number r, the* absolute Schur rth power of A *is the scalar matrix $A^{[r]} = [\|a_{jk}\|^r]$ with nonnegative entries. Denote by $\mathcal{M}(\mathfrak{B})$ the set of all (infinite) matrices with entries from the Banach algebra $\mathfrak{B}$.*

The following kind of generalized Schur product is first considered in [5] with bounded linear operators on a Hilbert space as entries.

DEFINITION 2.3. *For two matrices $A = [a_{jk}], B = [b_{jk}] \in \mathcal{M}(\mathfrak{B})$, the* Schur product of A and B *is the matrix $A \bullet B = [a_{jk}b_{jk}]$, where the multiplication of the entries is the multiplication of elements in $\mathfrak{B}$.*

We will make frequent use of the following well-known fact. The norm $\|A\|$ of a numerical matrix A will be the norm $\|A\|_{p,q}$ of A as a linear transformation from ℓ^p to ℓ^q if it is well defined and finite, and it is ∞ otherwise. We will sometimes include the subscripts p, q for clarity.

LEMMA 2.4. *Let $A = [a_{jk}]$, $B = [b_{jk}]$ be scalar matrices with nonnegative entries. If $a_{jk} \le b_{jk}$ for all j and k, then $\|A\| \le \|B\|$.*

To interpret the right-hand side of the following Hölder-type inequality, we will use the conventions $0 \cdot \infty = \infty \cdot 0 = 0$ and $\infty \cdot \infty = \alpha \cdot \infty = \infty \cdot \alpha = \infty$ for all positive α. The norm $\|A\|$ of a scalar matrix A is the operator norm of the linear transformation from ℓ^p to ℓ^q defined by the matrix A. Thus if $A \in \mathcal{M}(\mathfrak{B})$, $A^{[r]}$ is a scalar matrix, and $\|A^{[r]}\|$ is well defined (could be ∞).

THEOREM 2.5. *Let $A, B \in \mathcal{M}(\mathfrak{B})$. Then*

$$\left\|(A \bullet B)^{[1]}\right\| \leq \left\|A^{[1]} \bullet B^{[1]}\right\| \leq \left\|A^{[r]}\right\|^{1/r} \left\|B^{[r^*]}\right\|^{1/r^*}$$

for $1 < r < \infty$ and $\frac{1}{r} + \frac{1}{r^*} = 1$.

Proof. This is essentially the same proof that is given in [7]. For completeness we give a proof with the necessary modifications. With the above conventions, if either $\|A^{[r]}\|$ or $\|B^{[r^*]}\|$ is 0, then the matrix A or B is the zero matrix, and hence $A \bullet B = 0$. Thus the inequality clearly is satisfied. If either $\|A^{[r]}\|$ or $\|B^{[r^*]}\|$ is ∞, and the other is nonzero, then the right-hand side of the inequality is ∞, and hence the inequality holds. For the remaining case of both of these quantities being finite, we reason as follows. Let $A = [a_{jk}]$ and $B = [b_{jk}]$. For arbitrarily fixed positive integers J and K, and arbitrarily fixed unit vector $x = \{\xi_k\}_{k=1}^{\infty} \in \ell^p$,

$$\begin{aligned}
\sum_{j=1}^{J} \left[\sum_{k=1}^{K} \|a_{jk} b_{jk}\| \, |\xi_k|\right]^q &\leq \sum_{j=1}^{J} \left[\sum_{k=1}^{K} \|a_{jk}\| \, |\xi_k|^{1/r} \, \|b_{jk}\| \, |\xi_k|^{1/r^*}\right]^q \\
&\leq \sum_{j=1}^{J} \left[\sum_{k=1}^{K} \|a_{jk}\|^r \, |\xi_k|\right]^{q/r} \left[\sum_{k=1}^{K} \|b_{jk}\|^{r^*} \, |\xi_k|\right]^{q/r^*} \\
&\leq \left\{\sum_{j=1}^{J} \left[\sum_{k=1}^{K} \|a_{jk}\|^r \, |\xi_k|\right]^q\right\}^{1/r} \left\{\sum_{j=1}^{J} \left[\sum_{k=1}^{K} \|b_{jk}\|^{r^*} \, |\xi_k|\right]^q\right\}^{1/r^*} \\
&\leq \left\|\left(A^{[r]}\right)(|x|)\right\|_q^{q/r} \left\|\left(B^{[r^*]}\right)(|x|)\right\|_q^{q/r^*} \\
&\leq \left\|A^{[r]}\right\|_{p,q}^{q/r} \|x\|_p^{q/r} \left\|B^{[r^*]}\right\|_{p,q}^{q/r^*} \|x\|_p^{q/r^*} \\
&= \left\|A^{[r]}\right\|_{p,q}^{q/r} \left\|B^{[r^*]}\right\|_{p,q}^{q/r^*},
\end{aligned}$$

where $|x| = \{|\xi_k|\}_{k=1}^{\infty}$, which has the same ℓ^p-norm as x. Since J and K are arbitrary and the quantity $\|A^{[r]}\|_{p,q}^{q/r} \|B^{[r^*]}\|_{p,q}^{q/r^*}$ is independent of J and K, by taking the limits as $K \to \infty$ and then $J \to \infty$ we see that

$$\begin{aligned}
\sum_{j=1}^{\infty} \left[\left|\sum_{k=1}^{\infty} \|a_{jk} b_{jk}\| \, \xi_k\right|\right]^q \leq \sum_{j=1}^{\infty} \left[\sum_{k=1}^{\infty} \|a_{jk} b_{jk}\| \, |\xi_k|\right]^q &\leq \sum_{j=1}^{\infty} \left[\sum_{k=1}^{\infty} \|a_{jk}\| \, \|b_{jk}\| \, |\xi_k|\right]^q \\
&\leq \left\|A^{[r]}\right\|_{p,q}^{q/r} \left\|B^{[r^*]}\right\|_{p,q}^{q/r^*}.
\end{aligned}$$

Therefore

$$\begin{aligned}
\left\|(A \bullet B)^{[1]} x\right\|_q^q = \sum_{j=1}^{\infty} \left|\sum_{k=1}^{\infty} \|a_{jk} b_{jk}\| \, \xi_k\right|^q &\leq \sum_{j=1}^{\infty} \left[\sum_{k=1}^{\infty} \|a_{jk} b_{jk}\| \, |\xi_k|\right]^q \\
&\leq \left\|A^{[r]}\right\|_{p,q}^{q/r} \left\|B^{[r^*]}\right\|_{p,q}^{q/r^*}
\end{aligned}$$

and

$$\left\|\left(A^{[1]} \bullet B^{[1]}\right) x\right\|_q^q = \sum_{j=1}^{\infty} \left|\sum_{k=1}^{\infty} \|a_{jk}\| \, \|b_{jk}\| \, \xi_k\right|^q \le \sum_{j=1}^{\infty} \left[\sum_{k=1}^{\infty} \|a_{jk}\| \, \|b_{jk}\| \, |\xi_k|\right]^q$$
$$\le \left\|A^{[r]}\right\|_{p,q}^{q/r} \left\|B^{[r^*]}\right\|_{p,q}^{q/r^*}.$$

Since this is true for every unit vector x, we have

$$\left\|(A \bullet B)^{[1]}\right\|_{p,q} \le \left\|A^{[r]}\right\|_{p,q}^{1/r} \left\|B^{[r^*]}\right\|_{p,q}^{1/r^*}$$

and

$$\left\|A^{[1]} \bullet B^{[1]}\right\|_{p,q} \le \left\|A^{[r]}\right\|_{p,q}^{1/r} \left\|B^{[r^*]}\right\|_{p,q}^{1/r^*}$$

as asserted. The inequality $\|(A \bullet B)^{[1]}\|_{p,q} \le \|A^{[1]} \bullet B^{[1]}\|_{p,q}$ follows directly from the submultiplicativity of the norm on $\mathfrak{B}$ and Lemma 2.4. □

The case of $p = q = r = 2$ is an important special case. We actually started with that case. As expected, there is a corresponding Minkowski's inequality. Before stating the result, we introduce the necessary notation.

Definition 2.6. *Let r be a real number such that $1 \le r < \infty$, and let $A \in \mathcal{M}(\mathfrak{B})$ be a matrix such that $\left\|A^{[r]}\right\| < \infty$. We define the* absolute Schur r-norm of A *to be*

$$|||A|||_{p,q,r} = \left\|A^{[r]}\right\|^{1/r} := \left\|A^{[r]}\right\|_{p,q}^{1/r}.$$

For any matrix $A = [a_{jk}] \in \mathcal{M}(\mathfrak{B})$ with bounded entries (i.e., there exists a constant B such that $\|a_{jk}\| \le B$), the absolute Schur ∞-norm of A is defined to be

$$\|A\|_{\infty} = \sup\{\|a_{jk}\| : \; j, k = 1, 2, 3, \ldots\}.$$

For each r with $1 \le r < \infty$,

$$\mathcal{S}^r(\mathfrak{B}) = \left\{A \in \mathcal{M}(\mathfrak{B}) : \; |||A|||_{p,q,r} = \left\|A^{[r]}\right\|^{1/r} < \infty\right\}.$$

We don't know if the ∞-norm defined here is a good choice for this setup. It is just one that makes the following Minkowski-type inequality more complete. Another possible choice is the multiplier norm for the Schur multipliers. With this notation, the inequality in Theorem 2.5 can be restated as the following.

Corollary 2.7. *For any two matrices A and B in $\mathcal{M}(\mathfrak{B})$,*

$$\left\|(A \bullet B)^{[1]}\right\| \le \left\|A^{[1]} \bullet B^{[1]}\right\| \le |||A|||_{p,q,r} \, |||B|||_{p,q,r^*}$$

for $1 \le r \le \infty$ and $\frac{1}{r} + \frac{1}{r^*} = 1$. Moreover, if $A \in \mathcal{S}^r(\mathfrak{B}), B \in \mathcal{S}^{r^*}(\mathfrak{B})$, then $A \bullet B \in \mathcal{S}^1(\mathfrak{B})$.

Proof. Only the cases of $r = 1$ and $r = \infty$ need to be considered. Suppose that $A = [a_{jk}] \in \mathcal{S}^1(\mathfrak{B})$ and $B = [b_{jk}] \in \mathcal{S}^{\infty}(\mathfrak{B})$. Then

$$\begin{aligned}\left\|(A \bullet B)^{[1]}\right\| &= \|[\|a_{jk} b_{jk}\|]\| \\ &\le \|[\|a_{jk}\| \, \|b_{jk}\|]\| && \text{(by Lemma 2.4)} \\ &\le \|[\|a_{jk}\| \, \|B\|_{\infty}]\| && \text{(by Lemma 2.4 again)} \\ &= \|[\|a_{jk}\|]\| \, \|B\|_{\infty} \\ &= \|A\|_1 \, \|B\|_{\infty}\end{aligned}$$

as asserted. □

Now we are ready for the Minkowski's inequality.

THEOREM 2.8. *For given matrices A and B in $\mathcal{M}(\mathfrak{B})$ and for $1 \leq r \leq \infty$,*

$$|||A+B|||_{p,q,r} \leq |||A|||_{p,q,r} + |||B|||_{p,q,r}\,.$$

Proof. This is just a straightforward adaptation of the proof for the ℓ^p or L^p (Lebesgue spaces, see p. 64 in [8]) cases. If either of the two terms on the right-hand side of the inequality is ∞, then the inequality clearly holds. So we may assume that both of the quantities on the right are finite. For $1 < r < \infty$,

$$\begin{aligned}
\left\|(A+B)^{[r]}\right\| &\leq \left\|(A+B)^{[1]} \bullet (A+B)^{[r-1]}\right\| \leq \left\|\left(A^{[1]}+B^{[1]}\right) \bullet (A+B)^{[r-1]}\right\| \\
&\leq \left\|A^{[1]} \bullet (A+B)^{[r-1]}\right\| + \left\|B^{[1]} \bullet (A+B)^{[r-1]}\right\| \\
&\leq \left\|\left(A^{[1]}\right)^{[r]}\right\|^{1/r} \left\|\left((A+B)^{[r-1]}\right)^{[r^*]}\right\|^{1/r^*} \\
&\quad + \left\|\left(B^{[1]}\right)^{[r]}\right\|^{1/r} \left\|\left((A+B)^{[r-1]}\right)^{[r^*]}\right\|^{1/r^*} \\
&= \left\|A^{[r]}\right\|^{1/r} \left\|(A+B)^{[(r-1)r^*]}\right\|^{1/r^*} + \left\|B^{[r]}\right\|^{1/r} \left\|(A+B)^{[(r-1)r^*]}\right\|^{1/r^*} \\
&= \left(\left\|A^{[r]}\right\|^{1/r} + \left\|B^{[r]}\right\|^{1/r}\right) \left\|(A+B)^{[r]}\right\|^{1/r^*}.
\end{aligned}$$

For $r = 1$, this is just an immediate consequence of the triangle inequality for operator norm on $\mathfrak{B}$. Each entry $\|a_{jk} + b_{jk}\|$ of $(A+B)^{[1]}$ is not more than the corresponding entry $\|a_{jk}\| + \|a_{jk}\|$ of $A^{[1]} + B^{[1]}$. Therefore

$$\left\|(A+B)^{[1]}\right\| \leq \left\|A^{[1]} + B^{[1]}\right\| \leq \left\|A^{[1]}\right\| + \left\|B^{[1]}\right\|$$

by Lemma 2.4. The case of $r = \infty$ is even simpler than that of $r = 1$, and we omit the details. This completes the proof. □

PROPOSITION 2.9. *For every bounded scalar matrix $A \in \mathcal{B}(\ell^p, \ell^q)$,*

$$\|S_A\| \leq |||A|||_{p,q,2} \leq \|A\|\,,$$

where $\|S_A\|$ is the Schur multiplier norm of A and is defined by

$$\|S_A\| := \{\|A \bullet B\| : \ \|B\| \leq 1\}.$$

Proof. Note that for each $C \in \mathcal{B}(\ell^p, \ell^q)$,

$$|||C|||_{p,q,2} = \left\|C \bullet \overline{C}\right\|^{1/2} \leq \left\{\|C\| \left\|\overline{C}\right\|\right\}^{1/2} = \|C\|\,,$$

as C and $\overline{C}$ have the same norm. The second inequality follows. For each $B \in \mathcal{B}(\ell^p, \ell^q)$ with $\|B\| \leq 1$, we have $\|A \bullet B\| \leq |||A|||_{p,q,2}\, |||B|||_{p,q,2} \leq |||A|||_{p,q,2}\, \|B\| \leq |||A|||_{p,q,2}$. Thus $\|S_A\| \leq |||A|||_{p,q,2}$. □

REMARK 2.10. *We record here the following observations.*

(i) *It is proved in* [7] *that the equality holds in the second inequality in the proposition iff the operator norm and Schur multiplier norm of the matrix are equal. What are necessary and sufficient conditions for equality to hold in the first inequality?*

(ii) *From the analogy of p-norm and 2-norm, one is lead to conclude that the norm $|||\cdot|||_{2,2,2}$ is an inner product norm. It is not hard to find examples to show that the parallelogram law does not hold for $|||\cdot|||_{2,2,2}$, and hence it is not an inner product norm. (This has been observed by Leo Livshits.)*

3. The absolute Schur algebras over a Banach algebra. With the notation $A^{[r]} = [\|a_{jk}\|^r]$ for a given matrix $A = [a_{jk}] \in \mathcal{M}(\mathfrak{B})$, let

$$\mathcal{S}^r(\mathfrak{B}) = \left\{ A \in \mathcal{M}(\mathfrak{B}) : |||A|||_{p,q,r} = \left\| A^{[r]} \right\|^{1/r} < \infty \right\};$$

i.e., $\mathcal{S}^r(\mathfrak{B})$ is the set of all matrices $A = [a_{jk}]$ with a finite operator norm for the matrix $A^{[r]}$ as an element of $\mathcal{B}(\ell^p, \ell^q)$. Since we assume that $\mathfrak{B}$ is a Banach algebra with an identity, the set $\mathcal{B}(\ell^p, \ell^q)$ of scalar matrices can be regarded as a subset of $\mathcal{M}(\mathfrak{B})$. Moreover, the set $\mathcal{S}^2(\mathfrak{B})$ includes the set $\mathcal{B}(\ell^p, \ell^q)$ by the Schur–Bennett theorem. We will see that the inclusion is proper and that $\mathcal{S}^r(\mathfrak{B})$ is a Banach algebra under the usual sum, the Schur product, and the norm $|||\cdot|||_{p,q,r}$ defined above. For the special case of $\mathfrak{B} = \mathbb{C}$, we use $\mathcal{S}^r$ to denote $\mathcal{S}^r(\mathfrak{B})$.

PROPOSITION 3.1. *For $1 \leq r < r' \leq \infty$, $\mathcal{S}^r(\mathfrak{B}) \subsetneqq \mathcal{S}^{r'}(\mathfrak{B})$.*

Proof. Let $A \in \mathcal{S}^r(\mathfrak{B})$. Then the entries of A are bounded by some positive number. Multiply a suitable scalar $\alpha > 0$ to A, so that all entries αa_{jk} in αA are norm bounded by 1, $\alpha \|a_{jk}\| \leq 1$. Thus every entry $\alpha^{r'} \|a_{jk}\|^{r'}$ of $(\alpha A)^{[r']}$ is not more than the corresponding entry $\alpha^r \|a_{jk}\|^r$ of $(\alpha A)^{[r]}$. Thus

$$\alpha^{r'} \left\| A^{[r']} \right\| = \left\| (\alpha A)^{[r']} \right\| \leq \left\| (\alpha A)^{[r]} \right\| = \alpha^r \left\| A^{[r]} \right\| < \infty.$$

Thus $\|A^{[r']}\| \leq \alpha^{r-r'} \|A^{[r]}\| < \infty$, and hence $A \in \mathcal{S}^{r'}$. This shows that $\mathcal{S}^r(\mathfrak{B}) \subseteq \mathcal{S}^{r'}(\mathfrak{B})$. To see that the inclusion is proper, just take a sequence $\{\alpha_k\}$ of nonnegative numbers which is in $\ell^{r'}$ but not in ℓ^r (more explicitly, take $\alpha_k = [\frac{1}{k}]^{1/r}$). Then the matrix A with the first column the sequence $\{(\sqrt{\alpha_k})e\}$ (e the identity of $\mathfrak{B}$), and all other columns 0, is in $\mathcal{S}^r(\mathfrak{B})$ but not in $\mathcal{S}^{r'}(\mathfrak{B})$. □

Now we have the Riesz–Fischer-type theorem.

THEOREM 3.2. *For $1 \leq r \leq \infty$, the set $\mathcal{S}^r(\mathfrak{B})$ is a Banach algebra under the usual addition and scalar multiplication of matrices, the Schur product as the multiplication, and the norm $|||A|||_{p,q,r} = \|A^{[r]}\|^{1/r}$.*

Proof. We consider only the cases of $1 < r < \infty$ and leave the others for the reader. First we show that $\mathcal{S}^r(\mathfrak{B})$ is an algebra. Let $A,\ B \in \mathcal{S}^r(\mathfrak{B})$. Then

$$\begin{aligned} \left\| (A \bullet B)^{[r]} \right\| &\leq \left\| A^{[r]} \bullet B^{[r]} \right\| && \text{(by Lemma 2.4)} \\ &\leq \left\| A^{[r]} \right\| \left\| B^{[r]} \right\| && \text{(by the Schur–Bennett theorem [1])} \\ &< \infty. \end{aligned}$$

Thus $A \bullet B \in \mathcal{S}^r(\mathfrak{B})$ whenever $A,\ B \in \mathcal{S}^r(\mathfrak{B})$, hence $\mathcal{S}^r(\mathfrak{B})$ is an algebra. That the norm is submultiplicative also follows from these inequalities:

$$|||A \bullet B|||_{p,q,r} = \left\| (A \bullet B)^{[r]} \right\|^{1/r} \leq \left(\left\| A^{[r]} \right\| \left\| B^{[r]} \right\| \right)^{1/r} = |||A|||_{p,q,r} \, |||B|||_{p,q,r} \, .$$

The Minkowski-type inequality, Theorem 2.8, gives us the triangle inequality for the norm $|||\cdot|||_{p,q,r}$ and also the closure of the sum operation. All other conditions for a norm are easily verified.

The most important step here is the proof of completeness of $\mathcal{S}^r(\mathfrak{B})$. To this end, let $A_n = [a_{jk}^{(n)}]$, $n = 1, 2, \ldots$, be a sequence of matrices in $\mathcal{S}^r(\mathfrak{B})$ that form a Cauchy sequence in the norm $\vert\vert\vert\cdot\vert\vert\vert_{p,q,r}$. Then for each fixed j and k, we have

$$\begin{aligned}\left\| a_{jk}^{(n)} - a_{jk}^{(m)} \right\|^r &\le \left\| (A_n - A_m)^{[r]} \right\| \\ &= \vert\vert\vert A_n - A_m \vert\vert\vert_{p,q,r}^r \to 0, \quad \text{as } n,\ m \to \infty.\end{aligned}$$

We see that the sequence of (j,k)-entries $\{a_{jk}^{(n)}\}_{n=1}^{\infty}$ is a Cauchy sequence of elements in $\mathfrak{B}$. Since $\mathfrak{B}$ is complete, there is an element $a_{jk} \in \mathfrak{B}$ to which the sequence converges. Let $A = [a_{jk}]\,(\in \mathcal{M}(\mathfrak{B}))$. We will show that $A \in \mathcal{S}^r(\mathfrak{B})$ and that the sequence $\{A_n\}$ converges to A in the norm $\vert\vert\vert\cdot\vert\vert\vert_{p,q,r}$.

To see that $A \in \mathcal{S}^r(\mathfrak{B})$, let $x = \{\xi_k\} \in \ell^p$ be an arbitrary unit vector. Since $\{A_n\}$ is a Cauchy sequence under the norm $\vert\vert\vert\cdot\vert\vert\vert_{p,q,r}$, the sequence is bounded in the norm. Let

$$M = \sup\left\{ \vert\vert\vert A_n \vert\vert\vert_{p,q,r} :\ n = 1, 2, 3, \ldots \right\}.$$

Let J and K be arbitrary positive integers. For brevity of the notation, let $\widetilde{A}$ and $\widetilde{A}_n$ denote the numerical matrices which agree with $A^{[r]}$ and $A_n^{[r]}$ on the upper left $J \times K$ block and are 0 on all other entries. Let $\widetilde{x}$ denote the vector with the first K coordinates the same as that of x and all other entries 0. The entrywise convergence of A_n to A allows us to choose an N such that $\|\widetilde{A}_n - \widetilde{A}\|_{p,q} < 1$ for all $n \ge N$. Then

$$\begin{aligned}\sum_{j=1}^{J} \left[\sum_{k=1}^{K} \left| \|a_{jk}\|^r \xi_k \right| \right]^q &\le \left\| \widetilde{A} \right\|_{p,q}^q \le \left(\left\| \widetilde{A}_N \right\|_{p,q} + 1 \right)^q \le \left(\left\| A_N^{[r]} \right\|_{p,q} + 1 \right)^q \\ &\le \left[\vert\vert\vert A_N \vert\vert\vert_{p,q,r}^r + 1 \right]^q \le (M^r + 1)^q .\end{aligned}$$

Since this is true for every choice of J and K and the quantity $(M^r+1)^q$ is independent of J and K, by taking the limits as $K \to \infty$ and then as $J \to \infty$, we see that

$$\sum_{j=1}^{\infty} \left[\sum_{k=1}^{\infty} \left| (\|a_{jk}\|^r \xi_k) \right| \right]^q \le (M^r + 1)^q.$$

Thus

$$\left\| A^{[r]} x \right\|^q = \sum_{j=1}^{\infty} \left| \sum_{k=1}^{\infty} \|a_{jk}\|^r \xi_k \right|^q \le \sum_{j=1}^{\infty} \left[\sum_{k=1}^{\infty} \left| (\|a_{jk}\|^r \xi_k) \right| \right]^q \le (M^r + 1)^q.$$

That is, $A^{[r]}x \in \ell^q$ and $\|A^{[r]}x\| \le M^r + 1$. Since this is true for all unit vectors $x \in \ell^p$, we have that $A^{[r]} = [\|a_{jk}\|^r]$ defines a bounded linear transformation from ℓ^p to ℓ^q with operator norm $\le M^r + 1$. Thus $A \in \mathcal{S}^r(\mathfrak{B})$.

Now for the convergence, we reason as follows. Let $\epsilon > 0$ be given. Let

$$M = \sup\left\{ \vert\vert\vert A_n \vert\vert\vert_{p,q,r} :\ n = 1, 2, 3, \ldots \right\},$$

as in the preceding argument. Then, since the sequence $\{A_n\}$ is a Cauchy sequence under the norm $\vert\vert\vert\cdot\vert\vert\vert_{p,q,r}$, there exists an N such that

$$\vert\vert\vert A_n - A_m \vert\vert\vert_{p,q,r} < \frac{\epsilon}{6}$$

for all $n, m \geq N$. Let $n \geq N$. We will show that

$$\left\| \left[(A - A_n)^{[r]} \right] x \right\|_q < \epsilon^r \quad \text{for all} \quad x \in \ell^p \text{ with } \|x\|_p \leq 1.$$

To that end, let $x = \{\xi_k\} \in \ell^p$ be a vector of norm

$$\|x\|_p = \left(\sum_{k=1}^{\infty} |\xi_k|^p \right)^{1/p} \leq 1.$$

Then there exists, by the convergence of the series, a positive integer K (depending on x, of course) such that

$$\left\| P_K^\perp x \right\|_p = \left(\sum_{k=K+1}^{\infty} |\xi_k|^p \right)^{1/p} < \frac{\epsilon^r}{6 \left(M + \|A\|_{p,q,r} + 1 \right)^r},$$

where, for each positive integer ι, P_ι denotes the coordinate projection of ℓ^p onto the first ι coordinates; i.e.,

$$P_\iota \left[(\alpha_1, \alpha_2, \ldots) \right] = (\alpha_1, \ldots, \alpha_\iota, 0, 0, \ldots),$$

and for a coordinate projection Q, $Q^\perp = I - Q$ (I denotes the identity operator on ℓ^p). The matrix representation of P_ι with respect to the standard basis has the first ι diagonal entries 1's and all other entries (on or off diagonal) 0's. We will use the same notation for coordinate projections for both ℓ^p and ℓ^q and let the context determine which of the spaces is intended. Since $A \in \mathcal{S}^r(\mathfrak{B})$ from the preceding paragraph, and since $A - A_n \in \mathcal{S}^r(\mathfrak{B})$ for all n by Minkowski's inequality, we have

$$\left[(A - A_n)^{[r]} \right] P_K x \in \ell^q.$$

Thus there exists a J such that

$$\left\| P_J^\perp \left[(A - A_n)^{[r]} \right] P_K x \right\|_q < \frac{\epsilon^r}{6^r}.$$

By the construction of A, we may choose a $\nu \geq N$ (depending on J and K and hence on x) such that

$$\left\| P_J \left(A - A_\nu \right)^{[r]} P_K \right\|_{p,q} < \frac{\epsilon^r}{6^r},$$

because J and K are finite and $P_J \left(A - A_\nu \right)^{[r]} P_K$ has all entries 0 except the $J \times K$ rectangular block on the upper left corner. Now

$$\begin{aligned} \left\| \left[(A - A_n)^{[r]} \right] x \right\|_q &= \left\| \left(P_J + P_J^\perp \right) \left[(A - A_n)^{[r]} \right] \left(P_K + P_K^\perp \right) x \right\|_q \\ &\leq \left\| P_J \left[(A - A_n)^{[r]} \right] P_K x \right\|_q + \left\| P_J^\perp \left[(A - A_n)^{[r]} \right] P_K x \right\|_q \\ &\quad + \left\| P_J \left[(A - A_n)^{[r]} \right] P_K^\perp x \right\|_q + \left\| P_J^\perp \left[(A - A_n)^{[r]} \right] P_K^\perp x \right\|_q \end{aligned}$$

$$
\begin{aligned}
&\leq \left\| P_J \left[((A - A_\nu) + (A_\nu - A_n))^{[r]} \right] P_K x \right\|_q + \frac{\epsilon^r}{6} \\
&\quad + \left\| (A - A_n)^{[r]} \right\|_{p,q} \left\| P_K^\perp x \right\|_p + \left\| (A - A_n)^{[r]} \right\|_{p,q} \left\| P_K^\perp x \right\|_p \\
&\leq \left\| P_J \left\{ [(A - A_\nu) + (A_\nu - A_n)]^{[r]} \right\} P_K \right\|_{p,q} \left\| P_K x \right\|_p + \frac{\epsilon^r}{6} \\
&\quad + 2 \,|\!|\!| A - A_n |\!|\!|_{p,q,r}^r \left\| P_K^\perp x \right\|_p \\
&\leq |\!|\!| P_J \left(A - A_\nu \right) P_K + P_J \left(A_\nu - A_n \right) P_K |\!|\!|_{p,q,r}^r + \frac{\epsilon^r}{6} \\
&\quad + 2 \left(|\!|\!| A |\!|\!|_{p,q,r} + |\!|\!| A_n |\!|\!|_{p,q,r} \right)^r \left\| P_K^\perp x \right\|_p \\
&\leq \left[|\!|\!| P_J \left(A - A_\nu \right) P_K |\!|\!|_{p,q,r} + |\!|\!| P_J \left(A_\nu - A_n \right) P_K |\!|\!|_{p,q,r} \right]^r + \frac{\epsilon^r}{6} \\
&\quad + 2 \left(|\!|\!| A |\!|\!|_{p,q,r} + M \right)^r \left\| P_K^\perp x \right\|_p \\
&\leq \left[\frac{\epsilon}{3^{1/r} \cdot\ 2} + |\!|\!| A_\nu - A_n |\!|\!|_{p,q,r} \right]^r + \frac{\epsilon^r}{6} + \frac{2 \left(|\!|\!| A |\!|\!|_{p,q,r} + M \right)^r \epsilon^r}{6 \left(|\!|\!| A |\!|\!|_{p,q,r} + M + 1 \right)^r} \\
&\leq \left[\frac{\epsilon}{2 \cdot\ 3^{1/r}} + \frac{\epsilon}{2 \cdot\ 3^{1/r}} \right]^r + \frac{3\epsilon^r}{6} < \frac{5\epsilon^r}{6}.
\end{aligned}
$$

Since x is arbitrary with $\|x\| \leq 1$, we see that

$$
\left\| (A - A_n)^{[r]} \right\| \leq \frac{5\epsilon^r}{6}.
$$

Since $n \geq N$ is arbitrary, we have

$$
|\!|\!| A - A_n |\!|\!|_{p,q,r}^r \leq \frac{5\epsilon^r}{6} < \epsilon^r \ \text{ for all } n \geq N,
$$

which is equivalent to $|\!|\!| A - A_n |\!|\!|_{p,q,r} < \epsilon$ for all $n \geq N$. This proves that $\mathcal{S}^r(\mathfrak{B})$ is complete and finishes the proof of the theorem for $1 < r < \infty$. $\square$

4. Scalar absolute Schur algebras. The following result gives us a graded family of commutative Banach algebras (containing unbounded matrices) which contain all the bounded matrices. Recall that when $\mathfrak{B}$ is the complex field $\mathbb{C}$, $\mathcal{S}^r(\mathfrak{B})$ is denoted by $\mathcal{S}^r$ or $\mathcal{S}^r_{p,q}$. Note that for the unital Banach algebra $\mathfrak{B}$, each $\mathcal{S}^r$ can be considered as a subalgebra of $\mathcal{S}^r(\mathfrak{B})$.

Proposition 4.1. *Let* $r \geq 2$. *Then* $\mathcal{B}(\ell^p, \ell^q) \subsetneqq \mathcal{S}^r \subseteq \mathcal{S}^r(\mathfrak{B})$.

Proof. Let $A \in \mathcal{B}(\ell^p, \ell^q)$. Then $\overline{A} \in \mathcal{B}(\ell^p, \ell^q)$ and

$$
\left\| A^{[2]} \right\| = \left\| A \bullet \overline{A} \right\| \leq \|A\| \left\| \overline{A} \right\| = \|A\|^2 < \infty
$$

by the Schur–Bennett theorem [9, 1]. Thus $\mathcal{B}(\ell^p, \ell^q) \subseteq \mathcal{S}^2$. For $2 \leq r < \infty$, we have $\mathcal{S}^2 \subseteq \mathcal{S}^r$ by Proposition 3.1. Therefore $\mathcal{B}(\ell^p, \ell^q) \subseteq \mathcal{S}^r$. To see that the containment is proper, we note that the matrix A with the first column the sequence $\{\frac{1}{k^{1/q}}\}$, and all other columns 0, is in $\mathcal{S}^2$ ($A^{[2]}$ is a rank-1 matrix with the sum of the absolute values of the entries finite), but not in $\mathcal{B}(\ell^p, \ell^q)$ ($Ae_1 \notin \ell^q$, where e_1 is the first standard basis vector). Therefore $\mathcal{B}(\ell^p, \ell^q) \subsetneqq \mathcal{S}^2_{p,q}$. For $r > 2$, we already have $\mathcal{S}^r \supsetneqq \mathcal{S}^2$. Thus $\mathcal{B}(\ell^p, \ell^q) \subsetneqq \mathcal{S}^r$ for $r \geq 2$. $\square$

For the case of $p = 2$, $\mathcal{S}^2$ contains the bounded matrices as an ideal. We do not know if this is true for other values of $p > 2$.

Proposition 4.2. *The space, $\mathcal{B}(\ell^p, \ell^q)$, of bounded matrices is an ideal in $\mathcal{S}^2$.*

Proof. That $\mathcal{B}(\ell^p, \ell^q)$ is closed under the Schur product operation follows from the Schur–Bennett theorem [9, 1]. Let $A \in \mathcal{B}(\ell^p, \ell^q)$ and $B \in \mathcal{S}^2_{p,q}$. By Proposition 4.1, $A \in \mathcal{S}^2_{p,q}$. Thus $\|A^{[2]}\|, \|B^{[2]}\| < \infty$. Therefore

$$\|(A \bullet B)\|_{p,q} \leq \left\|A^{[2]}\right\|_{p,q}^{1/2} \left\|B^{[2]}\right\|_{p,q}^{1/2} < \infty,$$

by Theorem 2.5, and, consequently, $A \bullet B \in \mathcal{B}(\ell^p, \ell^q)$. □

It is easy to give examples of matrices that are in $\mathcal{S}^r_{2,2}$ for $1 < r < 2$ but not in $\mathcal{B}(\ell^2)$. Just as in the case of $p = 2$, let A be the matrix whose first column is the sequence $\{1/\sqrt{k}\}_{k=1}^{\infty}$ while all other columns are zero. Then the matrix A is not bounded as the ℓ^2-norm of the first column is

$$\sqrt{\sum_{k=1}^{\infty} \frac{1}{k}} = \infty,$$

whereas

$$\left\|A^{[p]}\right\| = \sqrt{\sum_{k=1}^{\infty} \frac{1}{k^p}} < \infty$$

as $p > 1$. The inclusion $\mathcal{S}^1 \subsetneqq \mathcal{B}(\ell^2)$ is well known; $\mathcal{S}^1$ is the set of *absolutely bounded matrices* (or *operators*). The arguments used to prove the proper inclusion here can be modified to give examples of matrices which are bounded (i.e., in $\mathcal{B}(\ell^2)$) but not in $\mathcal{S}^p$ for $1 < p < 2$. Therefore the algebras $\mathcal{S}^p$, $1 < p < 2$, and $\mathcal{B}(\ell^2)$ are not comparable.

Acknowledgment. The second author thanks the Mathematics Department of Ramkhamhaeng University and the first author for their hospitality during his visit.

REFERENCES

[1] G. Bennett, *Schur multipliers*, Duke Math. J., 44 (1977), pp. 603–639.

[2] C.-K. Fong, H. Radjavi, and P. Rosenthal, *Norms of matrices and operators*, J. Operator Theory, 18 (1987), pp. 99–113.

[3] R. A. Horn and C. R. Johnson, *Topics in Matrix Analysis*, Cambridge University Press, New York, 1991.

[4] W. Huang, C.-K. Li, and H. Schneider, *Norms and inequalities related to Schur products of rectangular matrices*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 334–347.

[5] L. Livshits, *Generalized Schur Product for Matrices with Operator Entries*, Ph.D. thesis, University of Toronto, Toronto, ON, 1991.

[6] R. Mathias and K. Okubo, *The induced norm of the Schur multiplication operator with respect to the operator radius*, Linear and Multilinear Algebra, 37 (1994), pp. 111–124.

[7] S.-C. Ong, *On the Schur multiplier norm of matrices*, Linear Algebra Appl., 56 (1984), pp. 45–55.

[8] W. Rudin, *Real and Complex Analysis*, 3rd. ed., McGraw–Hill, New York, 1987.

[9] J. Schur, *Bemerkungen Theorie der beschränken Bilinearformen mit unendlich vielen Veränder lichen*, J. Reine Angew. Math., 140 (1911), pp. 1–28.

[10] Q. Stout, *Schur Multiplication on $\mathcal{B}(\mathcal{H})$*, Ph.D. thesis, Indiana University, Bloomington, IN, 1977.

[11] Q. Stout, *Schur multiplication on $\mathcal{B}(\ell_p, \ell_q)$*, J. Operator Theory, 5 (1981), pp. 231–243.

SIAM J. MATRIX ANAL. APPL.
Vol. 20, No. 3, pp. 606–627

MODIFYING A SPARSE CHOLESKY FACTORIZATION*

TIMOTHY A. DAVIS[†] AND WILLIAM W. HAGER[‡]

Abstract. Given a sparse symmetric positive definite matrix $\mathbf{AA}^\mathsf{T}$ and an associated sparse Cholesky factorization $\mathbf{LDL}^\mathsf{T}$ or $\mathbf{LL}^\mathsf{T}$, we develop sparse techniques for obtaining the new factorization associated with either adding a column to $\mathbf{A}$ or deleting a column from $\mathbf{A}$. Our techniques are based on an analysis and manipulation of the underlying graph structure and on ideas of Gill et al. [*Math. Comp.*, 28 (1974), pp. 505–535] for modifying a dense Cholesky factorization. We show that our methods extend to the general case where an arbitrary sparse symmetric positive definite matrix is modified. Our methods are optimal in the sense that they take time proportional to the number of nonzero entries in $\mathbf{L}$ and $\mathbf{D}$ that change.

Key words. numerical linear algebra, direct methods, Cholesky factorization, sparse matrices, mathematical software, matrix updates

AMS subject classifications. 65F05, 65F50, 65-04

PII. S0895479897321076

1. Introduction.

This paper presents a method for updating and downdating the sparse Cholesky factorization $\mathbf{LDL}^\mathsf{T}$ or $\mathbf{LL}^\mathsf{T}$ of the matrix $\mathbf{AA}^\mathsf{T}$, where $\mathbf{A}$ is m-by-n. More precisely, we evaluate the Cholesky factorization of $\mathbf{AA}^\mathsf{T} + \sigma\mathbf{ww}^\mathsf{T}$, where either σ is $+1$ (corresponding to an update) and $\mathbf{w}$ is arbitrary or σ is -1 (corresponding to a downdate) and $\mathbf{w}$ is a column of $\mathbf{A}$. Both $\mathbf{AA}^\mathsf{T}$ and $\mathbf{AA}^\mathsf{T} + \sigma\mathbf{ww}^\mathsf{T}$ must be symmetric and positive definite. From this it follows that $m \le n$. The techniques we develop for the matrix $\mathbf{AA}^\mathsf{T}$ can be extended to determine the effects on the Cholesky factors of a general symmetric positive definite matrix $\mathbf{M}$ of any symmetric change of the form $\mathbf{M} + \sigma\mathbf{ww}^\mathsf{T}$ that preserves positive definiteness. The $\mathbf{AA}^\mathsf{T}$ case is simpler than the general case, which is why we discuss it first. Moreover, the techniques we develop for updating and downdating $\mathbf{AA}^\mathsf{T}$ are used in the algorithm for updating the general matrix $\mathbf{M}$. Our methods take into account the change in the sparsity pattern of $\mathbf{A}$ and $\mathbf{L}$ and are optimal in the sense that they take time proportional to the number of nonzero entries in $\mathbf{L}$ and $\mathbf{D}$ that change.

The importance of this problem has long been recognized [36], but prior sparse methods either are nonoptimal or do not consider changes to the sparsity pattern of $\mathbf{A}$ or $\mathbf{L}$. Both Law's sparse update method [27, 28] and the method of Chan and Brandwajn [6] are based on Bennett's method [3], which needs to be used with caution [20]. Law's symbolic update has nonoptimal asymptotic run time and can take more time than doing the symbolic factorization from scratch. The method of Row, Powell, and Mondkar [32] is for envelope-style factorization only and is also nonoptimal in both its numerical and its symbolic work. Neither of these approaches consider symbolic or numerical downdate. Chan and Brandwajn [6] consider the sparse numerical update and downdate, but with a fixed sparsity pattern.

*Received by the editors May 5, 1997; accepted for publication (in revised form) by S. Vavasis April 28, 1998; published electronically March 2, 1999.

http://www.siam.org/journals/simax/20-3/32107.html

[†] Department of Computer and Information Science and Engineering, University of Florida, Gainesville, FL 32611-6120 (davis@cise.ufl.edu, http://www.cise.ufl.edu/∼davis). The work of this author was supported by National Science Foundation grant DMS-9504974.

[‡] Department of Mathematics, University of Florida, Gainesville, FL 32611-6120 (hager@math.ufl.edu, http://www.math.ufl.edu/∼hager). The work of this author was supported by National Science Foundation grant DMS-9404431.

There are many applications of the techniques presented in this paper. In the linear program dual active set algorithm (LP DASA) [26], the $\mathbf{A}$ matrix corresponds to the basic variables in the current basis of the linear program, and in successive iterations, we bring variables in and out of the basis, leading to changes of the form $\mathbf{AA}^\mathsf{T} + \sigma \mathbf{ww}^\mathsf{T}$. Other application areas where the techniques developed in this paper are applicable include least-squares problems in statistics, the analysis of electrical circuits and power systems, structural mechanics, sensitivity analysis in linear programming, boundary condition changes in partial differential equations, domain decomposition methods, and boundary element methods. For a discussion of these application areas and others, see [25].

Section 2 introduces our notation. For an introduction to sparse matrix techniques, see [9, 13]. In section 3 we discuss the structure of the nonzero elements in the Cholesky factorization of $\mathbf{AA}^\mathsf{T}$, and in section 4 we discuss the structure associated with the Cholesky factors of $\mathbf{AA}^\mathsf{T} + \sigma \mathbf{ww}^\mathsf{T}$. The symbolic update and downdate methods provide the framework for our sparse version of Method C1 of Gill et al. [20] for modifying a dense Cholesky factorization. We discuss our sparse algorithm in section 5. Section 6 presents the general algorithm for modifying the sparse Cholesky factorization for *any* sparse symmetric positive definite matrix. A single update or downdate in the general case is more complicated and requires both a symbolic update and a symbolic downdate, based on the methods for $\mathbf{AA}^\mathsf{T}$ presented in section 4. The results of a numerical experiment with a large optimization problem from Netlib [8] are presented in section 7. Section 8 concludes with a discussion of future work.

2. Notation.

Throughout the paper, matrices are capital bold letters like $\mathbf{A}$ or $\mathbf{L}$, while vectors are lower-case bold letters like $\mathbf{x}$ or $\mathbf{v}$. Sets and multisets are in calligraphic style like $\mathcal{A}$, $\mathcal{L}$, or $\mathcal{P}$. Scalars are either lower-case Greek letters or italic style like σ, k, or m.

Given the location of the nonzero elements of $\mathbf{AA}^\mathsf{T}$, we can perform a *symbolic factorization* (this terminology is introduced by George and Liu in [13]) of the matrix to predict the location of the nonzero elements of the Cholesky factor $\mathbf{L}$. In actuality, some of these predicted nonzeros may be zero due to numerical cancellation during the factorization process. The statement "$l_{ij} \neq 0$" will mean that l_{ij} is *symbolically* nonzero. The main diagonals of $\mathbf{L}$ and $\mathbf{D}$ are always nonzero since the matrices that we factor are positive definite (see [35, p. 253]). The nonzero pattern of column j of $\mathbf{L}$ is denoted $\mathcal{L}_j$,

$$\mathcal{L}_j = \{i : l_{ij} \neq 0\},$$

while $\mathcal{L}$ denotes the collection of patterns:

$$\mathcal{L} = \{\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_m\}.$$

Similarly, $\mathcal{A}_j$ denotes the nonzero pattern of column j of $\mathbf{A}$,

$$\mathcal{A}_j = \{i : a_{ij} \neq 0\},$$

while $\mathcal{A}$ is the collection of patterns:

$$\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n\}.$$

The *elimination tree* can be defined in terms of a *parent map* π (see [29]). For any node j, $\pi(j)$ is the row index of the first nonzero element in column j of $\mathbf{L}$ beneath the diagonal element:

$$\pi(j) = \min \ \mathcal{L}_j \setminus \{j\},$$

where "min $\mathcal{X}$" denotes the smallest element of $\mathcal{X}$:

$$\min \ \mathcal{X} = \min_{i \in \mathcal{X}} \ i.$$

Our convention is that the min of the empty set is zero. Note that $j < \pi(j)$ except in the case where the diagonal element in column j is the only nonzero element. The inverse π^{-1} of the parent map is the *children multifunction*. That is, the children of node k are the set defined by

$$\pi^{-1}(k) = \{j : \pi(j) = k\}.$$

The *ancestors* of a node j, denoted $\mathcal{P}(j)$, are the set of successive parents:

$$\mathcal{P}(j) = \{j, \ \pi(j), \ \pi(\pi(j)), \ldots\} = \{\pi^0(j), \ \pi^1(j), \ \pi^2(j), \ldots\}.$$

Here the powers of a map are defined in the usual way: π^0 is the identity while π^i for $i > 0$ is the i-fold composition of π with itself. The sequence of nodes j, $\pi(j)$, $\pi(\pi(j))$, …, forming $\mathcal{P}(k)$, is called the *path* from j to the associated tree *root*. The collection of paths leading to a root form an *elimination tree*. The set of all trees is the *elimination forest*. Typically, there is a single tree whose root is m; however, if column j of $\mathbf{L}$ has only one nonzero element, the diagonal element, then j will be the root of a separate tree.

The number of elements (or size) of a set $\mathcal{X}$ is denoted $|\mathcal{X}|$, while $|\mathcal{A}|$ or $|\mathcal{L}|$ denotes the sum of the sizes of the sets they contain. Let the complement of a set $\mathcal{X}$ be denoted as $\mathcal{X}^c = \{x : x \notin \mathcal{X}\}$.

3. Symbolic factorization.

Any approach for generating the pattern set $\mathcal{L}$ is called *symbolic factorization* [10, 11, 12, 13, 34]. The symbolic factorization of a matrix of the form $\mathbf{AA}^\mathsf{T}$ is given in Algorithm 1 (see [14, 29]).

ALGORITHM 1 (symbolic factorization of $\mathbf{AA}^\mathsf{T}$).

```
$\pi(j) = 0$ for each $j$
for $j = 1$ to $m$ do
    $\mathcal{L}_j = \{j\} \cup \left( \bigcup_{c \in \pi^{-1}(j)} \mathcal{L}_c \setminus \{c\} \right) \cup \left( \bigcup_{\min \ \mathcal{A}_k = j} \mathcal{A}_k \right)$
    $\pi(j) = \min \ \mathcal{L}_j \setminus \{j\}$
end for
```

Algorithm 1 basically says that the pattern of column j of $\mathbf{L}$ can be expressed as the union of the patterns of each column of $\mathbf{L}$ whose parent is j and the patterns of the columns of $\mathbf{A}$ whose first nonzero element is j. The elimination tree, connecting each child to its parent, is easily formed during the symbolic factorization. Algorithm 1 can be done in $O(|\mathcal{L}| + |\mathcal{A}|)$ time[1] [14, 29].

Observe that the pattern of the parent of node j contains all entries in the pattern of column j except j itself [33]. That is,

$$\mathcal{L}_j \setminus \{j\} = \mathcal{L}_j \cap \{i : \pi(j) \leq i\} \subseteq \mathcal{L}_{\pi(j)}.$$

[1] Asymptotic complexity notation is defined in [7]. We write $f(n) = O(g(n))$ if there exist positive constants c and n_0 such that $0 \leq f(n) \leq cg(n)$ for all $n > n_0$.

Proceeding by induction, if k is an ancestor of j, then

$$\{i : i \in \mathcal{L}_j, k \le i\} \subseteq \mathcal{L}_k. \tag{3.1}$$

This leads to the following relation between $\mathcal{L}_j$ and the path $\mathcal{P}(j)$. The first part of this proposition, and its proof, are given in [33]. Our proof differs slightly from the one in [33]. We include it here since the same proof technique is exploited later.

PROPOSITION 3.1. *For each j, we have $\mathcal{L}_j \subseteq \mathcal{P}(j)$; furthermore, for each k and $j \in \mathcal{P}(k)$, $\mathcal{L}_j \subseteq \mathcal{P}(k)$.*

Proof. Obviously, $j \in \mathcal{P}(j)$. Let i be any given element of $\mathcal{L}_j$ with $i \ne j$. Since $j < i$, we see that the following relation holds for $l = 0$:

$$\pi^0(j) < \pi^1(j) < \cdots < \pi^l(j) < i. \tag{3.2}$$

Now suppose that (3.2) holds for some integer $l \ge 0$, and let k denote $\pi^l(j)$. By (3.1) and the fact that $k < i$, we have $i \in \mathcal{L}_k$, which implies that

$$i \ge \pi(k) = \pi(\pi^l(j)) = \pi^{l+1}(j).$$

Hence, either $i = \pi^{l+1}(j)$ or (3.2) holds with l replaced by $l+1$. Since (3.2) is violated for l sufficiently large, we conclude that there exists an l for which $i = \pi^{l+1}(j)$. Consequently, $i \in \mathcal{P}(j)$. Since each element of $\mathcal{L}_j$ is contained in $\mathcal{P}(j)$, we have $\mathcal{L}_j \subseteq \mathcal{P}(j)$. If $j \in \mathcal{P}(k)$ for some k, then j is an ancestor of k and $\mathcal{P}(j) \subseteq \mathcal{P}(k)$. Since we have already shown that $\mathcal{L}_j \subseteq \mathcal{P}(j)$, the proof is complete. □

As we will see, the symbolic factorization of $\mathbf{AA}^\mathsf{T} + \mathbf{ww}^\mathsf{T}$ can be obtained by updating the symbolic factorization of $\mathbf{AA}^\mathsf{T}$ using an algorithm that has the same structure as that of Algorithm 1, except that it operates only on nodes in the path $\mathcal{P}(j)$ (of the updated factors) for some node j. The symbolic update algorithm adds new entries to the nonzero pattern, which can be done with a simple union operation.

However, downdating is not as easy as updating. Once a set union has been computed, it cannot be undone without knowledge of how entries entered the set. We can keep track of this information by storing the elements of $\mathcal{L}$ as multisets rather than as sets. The multiset associated with column j has the form

$$\mathcal{L}_j^\sharp = \{(i, m(i,j)) : i \in \mathcal{L}_j\},$$

where the multiplicity $m(i,j)$ is the number of children of j that contain row index i in their pattern plus the number of columns of $\mathcal{A}$ whose smallest entry is j and that contain row index i. Equivalently,

$$m(i,j) = |\{k \in \pi^{-1}(j) : i \in \mathcal{L}_k\}| + |\{k : \min\ \mathcal{A}_k = j \text{ and } i \in \mathcal{A}_k\}|.$$

With this definition, we can undo a set union by subtracting multiplicities.

We now define some operations involving multisets. First, if $\mathcal{X}^\sharp$ is a multiset consisting of pairs $(i, m(i))$ where $m(i)$ is the multiplicity associated with i, then $\mathcal{X}$ is the set obtained by removing the multiplicities. In other words, the multiset $\mathcal{X}^\sharp$ and the associated base set $\mathcal{X}$ satisfy the relation

$$\mathcal{X}^\sharp = \{(i, m(i)) : i \in \mathcal{X}\}.$$

We define the addition of a multiset $\mathcal{X}^\sharp$ and a set $\mathcal{Y}$ in the following way:

$$\mathcal{X}^\sharp + \mathcal{Y} = \{(i, m'(i)) \ : \ i \in \mathcal{X} \text{ or } i \in \mathcal{Y}\},$$

where

$$m'(i) = \begin{cases} 1 & \text{if } i \notin \mathcal{X} \text{ and } i \in \mathcal{Y}, \\ m(i) & \text{if } i \in \mathcal{X} \text{ and } i \notin \mathcal{Y}, \\ m(i)+1 & \text{if } i \in \mathcal{X} \text{ and } i \in \mathcal{Y}. \end{cases}$$

Similarly, the subtraction of a set $\mathcal{Y}$ from a multiset $\mathcal{X}^\sharp$ is defined by

$$\mathcal{X}^\sharp - \mathcal{Y} = \{(i, m'(i)) \,:\, i \in \mathcal{X} \text{ and } m'(i) > 0\},$$

where

$$m'(i) = \begin{cases} m(i) & \text{if } i \notin \mathcal{Y}, \\ m(i)-1 & \text{if } i \in \mathcal{Y}. \end{cases}$$

The multiset subtraction of $\mathcal{Y}$ from $\mathcal{X}^\sharp$ cancels a prior addition. That is, for any multiset $\mathcal{X}^\sharp$ and any set $\mathcal{Y}$, we have

$$((\mathcal{X}^\sharp + \mathcal{Y}) - \mathcal{Y}) = \mathcal{X}^\sharp.$$

In contrast $((\mathcal{X} \cup \mathcal{Y}) \setminus \mathcal{Y})$ is equal to $\mathcal{X}$ if and only if $\mathcal{X}$ and $\mathcal{Y}$ are disjoint sets.

Algorithm 2 below performs a symbolic factorization of $\mathbf{AA}^\mathsf{T}$ with each set union operation replaced by a multiset addition. This algorithm is identical to Algorithm 1 except for the bookkeeping associated with multiplicities.

Algorithm 2 (symbolic factorization of $\mathbf{AA}^\mathsf{T}$ using multisets).

```
$\pi(j) = 0$ for each $j$
for $j = 1$ to $m$ do
    $\mathcal{L}_j^\sharp = \{(j, 1)\}$
    for each $c \in \pi^{-1}(j)$ do
        $\mathcal{L}_j^\sharp = \mathcal{L}_j^\sharp + (\mathcal{L}_c \setminus \{c\})$
    end for
    for each $k$ where min $\mathcal{A}_k = j$ do
        $\mathcal{L}_j^\sharp = \mathcal{L}_j^\sharp + \mathcal{A}_k$
    end for
    $\pi(j) = \min\ \mathcal{L}_j \setminus \{j\}$
end for
```

We conclude this section with a result concerning the relation between the patterns of $\mathbf{AA}^\mathsf{T}$ and the patterns of $\mathbf{AA}^\mathsf{T} + \mathbf{ww}^\mathsf{T}$.

Proposition 3.2. *Let $\mathcal{C}$ and $\mathcal{D}$ be the patterns associated with the symmetric positive definite matrices $\mathbf{C}$ and $\mathbf{D}$, respectively. Neglecting numerical cancellation, $\mathcal{C}_j \subseteq \mathcal{D}_j$ for each j implies that $(\mathcal{L}_C)_j \subseteq (\mathcal{L}_D)_j$ for each j, where $\mathcal{L}_C$ and $\mathcal{L}_D$ are the patterns associated with the Cholesky factors of $\mathbf{C}$ and $\mathbf{D}$, respectively.*

Proof. In [13, 31] it is shown that an edge (i, j) is contained in the undirected graph of the Cholesky factor of a symmetric positive definite matrix $\mathbf{C}$ if and only if there is a path from i to j in the undirected graph of $\mathbf{C}$ with each intermediate vertex of the path between 1 and min $\{i, j\}$. If $\mathcal{C}_j \subseteq \mathcal{D}_j$ for each j, then the paths associated with the undirected graph of $\mathbf{C}$ are a subset of the paths associated with the undirected graph of $\mathbf{D}$. It follows that $(\mathcal{L}_C)_j \subseteq (\mathcal{L}_D)_j$ for each j. □

Ignoring numerical cancellation, the edges in the undirected graph of $\mathbf{AA}^\mathsf{T}$ are a subset of the edges in the undirected graph of $\mathbf{AA}^\mathsf{T} + \mathbf{ww}^\mathsf{T}$. By Proposition 3.2, we conclude that the edges in the undirected graphs of the associated Cholesky factors satisfy the same inclusion.

4. Modifying the symbolic factors. Let $\overline{\mathbf{A}}$ be the modified version of $\mathbf{A}$. We put a bar over a matrix or a set or a multiset to denote its value after the update or downdate is complete. In an update, $\overline{\mathbf{A}}$ is obtained from $\mathbf{A}$ by appending the column $\mathbf{w}$ on the right, while in a downdate, $\overline{\mathbf{A}}$ is obtained from $\mathbf{A}$ by deleting the column $\mathbf{w}$ from $\mathbf{A}$. Hence, we have

$$\overline{\mathbf{A}}\,\overline{\mathbf{A}}^{\mathsf{T}} = \mathbf{A}\mathbf{A}^{\mathsf{T}} + \sigma \mathbf{w}\mathbf{w}^{\mathsf{T}},$$

where σ is either $+1$ and $\mathbf{w}$ is the last column of $\overline{\mathbf{A}}$ (update) or σ is -1 and $\mathbf{w}$ is a column of $\mathbf{A}$ (downdate). Since $\overline{\mathbf{A}}$ and $\mathbf{A}$ differ by at most a single column, it follows from Proposition 3.2 that $\mathcal{L}_j \subseteq \overline{\mathcal{L}}_j$ for each j during an update, while $\overline{\mathcal{L}}_j \subseteq \mathcal{L}_j$ during a downdate. Moreover, the multisets associated with the Cholesky factor of either the updated or downdated matrix have the structure described in the following theorem.

THEOREM 4.1. *Let k be the index associated with the first nonzero component of $\mathbf{w}$. For an update, $\mathcal{P}(k) \subseteq \overline{\mathcal{P}}(k)$. Moreover, $\overline{\mathcal{L}}_i^\sharp = \mathcal{L}_i^\sharp$ for all $i \in \overline{\mathcal{P}}(k)^c$. That is, $\overline{\mathcal{L}}_i^\sharp = \mathcal{L}_i^\sharp$ for all i except when i is k or one of the new ancestors of k. For a downdate, $\overline{\mathcal{P}}(k) \subseteq \mathcal{P}(k)$. Moreover, $\overline{\mathcal{L}}_i^\sharp = \mathcal{L}_i^\sharp$ for all $i \in \mathcal{P}(k)^c$. That is, $\overline{\mathcal{L}}_i^\sharp = \mathcal{L}_i^\sharp$ for all i except when i is k or one of the old ancestors of k.*

Proof. To begin, let us consider an update. We will show that each element of $\mathcal{P}(k)$ is a member of $\overline{\mathcal{P}}(k)$ as well. Clearly, k lies in both $\mathcal{P}(k)$ and $\overline{\mathcal{P}}(k)$. Proceeding by induction, suppose that

$$\pi^0(k), \pi^1(k), \pi^2(k), \ldots, \pi^j(k) \in \overline{\mathcal{P}}(k),$$

and define $l = \pi^j(k)$. We need to show that

$$\pi(l) = \pi(\pi^j(k)) = \pi^{j+1}(k) \in \overline{\mathcal{P}}(k)$$

to complete the induction. Since $l \in \overline{\mathcal{P}}(k)$, we have $l = \overline{\pi}^h(k)$ for some h. If $\overline{\pi}(l) = \pi(l)$, then

$$\overline{\pi}^{h+1}(k) = \overline{\pi}(\overline{\pi}^h(k)) = \overline{\pi}(l) = \pi(l) = \pi(\pi^j(k)) = \pi^{j+1}(k).$$

Since $\pi^{j+1}(k) = \overline{\pi}^{h+1}(k) \in \overline{\mathcal{P}}(k)$, the induction step is complete and $\pi^{j+1}(k) \in \overline{\mathcal{P}}(k)$.

If $\overline{\pi}(l) \neq \pi(l)$, then by Proposition 3.2, $\overline{\pi}(l) < \pi(l)$ and the following relation holds for $p = 1$:

$$\overline{\pi}^1(l) < \overline{\pi}^2(l) < \cdots < \overline{\pi}^p(l) < \pi(l). \tag{4.1}$$

Now suppose that (4.1) holds for some integer $p \geq 1$, and let q denote $\overline{\pi}^p(l)$. By Proposition 3.2, $\pi(l) \in \mathcal{L}_l \subseteq \overline{\mathcal{L}}_l$, and combining this with (4.1), $q = \overline{\pi}^p(l) < \pi(l) \in \overline{\mathcal{L}}_l$. It follows from (3.1) that $\pi(l) \in \overline{\mathcal{L}}_q$ for $q = \overline{\pi}^p(l)$. By the definition of the parent,

$$\overline{\pi}(q) = \overline{\pi}(\overline{\pi}^p(l)) = \overline{\pi}^{p+1}(l) \leq \pi(l) \in \overline{\mathcal{L}}_q.$$

Hence, either $\overline{\pi}^{p+1}(l) = \pi(l)$ or (4.1) holds with p replaced by $p + 1$. Since (4.1) is violated for p sufficiently large, we conclude that there exists an integer p such that $\overline{\pi}^p(l) = \pi(l)$, from which it follows that

$$\pi^{j+1}(k) = \pi(\pi^j(k)) = \pi(l) = \overline{\pi}^p(l) = \overline{\pi}^p(\overline{\pi}^h(k)) = \overline{\pi}^{p+h}(k) \in \overline{\mathcal{P}}(k).$$

Since $\pi^{j+1}(k) \in \overline{\mathcal{P}}(k)$, the induction step is complete and $\mathcal{P}(k) \subseteq \overline{\mathcal{P}}(k)$.

Suppose that $l \in \overline{\mathcal{P}}(k)^c$. It is now important to recall that k is the index of the first nonzero component of $\mathbf{w}$, the vector appearing in the update. Observe that l cannot equal k since $l \in \overline{\mathcal{P}}(k)^c$ and $k \in \overline{\mathcal{P}}(k)$. The proof that $\overline{\mathcal{L}}_l^\sharp = \mathcal{L}_l^\sharp$ is by induction on the height h defined by

$$h(l) = \max\{i : \pi^i(j) = l \text{ for some } j\}.$$

If $h(l) = 0$, then l has no children and the child loop of Algorithm 2 will be skipped when either $\overline{\mathcal{L}}_l^\sharp$ or $\mathcal{L}_l^\sharp$ are evaluated. And since $l \neq k$, the pattern associated with $\mathbf{w}$ cannot be added into $\overline{\mathcal{L}}_l^\sharp$. Hence, when $h(l) = 0$, the identity $\overline{\mathcal{L}}_l^\sharp = \mathcal{L}_l^\sharp$ is trivial. Now, assuming that for some $p \geq 0$ we have $\overline{\mathcal{L}}_l^\sharp = \mathcal{L}_l^\sharp$ whenever $l \in \overline{\mathcal{P}}(k)^c$ and $h(l) \leq p$, let us suppose that $h(l) = p + 1$. If $i \in \pi^{-1}(l)$, then $h(i) \leq p$. Hence, $\overline{\mathcal{L}}_i^\sharp = \mathcal{L}_i^\sharp$ for $i \in \pi^{-1}(l)$ by the induction assumption. And since $l \neq k$, the pattern of $\mathbf{w}$ is not added to $\overline{\mathcal{L}}_l^\sharp$. Consequently, when Algorithm 2 is executed, we have $\overline{\mathcal{L}}_l^\sharp = \mathcal{L}_l^\sharp$, which completes the induction step.

Now consider the downdate part of the theorem. Rearranging the downdate relation $\overline{\mathbf{A}}\,\overline{\mathbf{A}}^\mathsf{T} = \mathbf{A}\mathbf{A}^\mathsf{T} - \mathbf{w}\mathbf{w}^\mathsf{T}$, we have

$$\mathbf{A}\mathbf{A}^\mathsf{T} = \overline{\mathbf{A}}\,\overline{\mathbf{A}}^\mathsf{T} + \mathbf{w}\mathbf{w}^\mathsf{T}.$$

Hence, in a downdate, we can think of $\mathbf{A}$ as the updated version of $\overline{\mathbf{A}}$. Consequently, the second part of the theorem follows directly from the first part. $\square$

4.1. Symbolic update algorithm. We now present an algorithm for evaluating the new pattern $\overline{\mathcal{L}}$ associated with an update. Based on Theorem 4.1, the only sets $\overline{\mathcal{L}}_j$ that change are those associated with $\overline{\mathcal{P}}(k)$ where k is the index of the first nonzero component of $\mathbf{w}$. Referring to Algorithm 2, we can set $j = k$, $j = \overline{\pi}(k)$, $j = \overline{\pi}^2(k)$, and so on, marching up the path from k, and we can evaluate all the changes induced by the additional column in $\mathbf{A}$. In order to do the bookkeeping, there are at most four cases to consider:

Case 1: $j = k$. At the start of the new path, we need to add the pattern for $\mathbf{w}$ to $\mathcal{L}_j^\sharp$.

Case 2: $c \in \overline{\mathcal{P}}(k)$, $j \in \overline{\mathcal{P}}(k)$, $j > k$, and $c \in \overline{\pi}^{-1}(j) \cap \pi^{-1}(j)$. In this case, c is a child of j in both the new and the old elimination trees. Since the pattern $\overline{\mathcal{L}}_c$ may differ from $\mathcal{L}_c$, we need to add the difference to $\overline{\mathcal{L}}_j^\sharp$. Since j has a unique child on the path $\overline{\mathcal{P}}(k)$, there is at most one node c that satisfies these conditions. Also note that if

$$c \in \overline{\pi}^{-1}(j) \cap \pi^{-1}(j) \cap \overline{\mathcal{P}}(k)^c,$$

then by Theorem 4.1 $\overline{\mathcal{L}}_c = \mathcal{L}_c$ and hence this node c does not lead to an adjustment to $\overline{\mathcal{L}}_j^\sharp$ in Algorithm 2.

Case 3: $j \in \overline{\mathcal{P}}(k)$, $j > k$, and $c \in \overline{\pi}^{-1}(j) \setminus \pi^{-1}(j)$. In this case, c is a child of j in the new elimination tree, but not in the old tree, and the entire set $\overline{\mathcal{L}}_c$ should be added to $\overline{\mathcal{L}}_j^\sharp$ since it was not included in $\mathcal{L}_j^\sharp$. By Theorem 4.1, $\overline{\pi}(p) = \pi(p)$ for all $p \in \overline{\mathcal{P}}(k)^c$. Since $c \in \overline{\pi}^{-1}(j)$ but $c \notin \pi^{-1}(j)$, it follows that $\overline{\pi}(c) = j \neq \pi(c)$, and hence, $c \notin \overline{\mathcal{P}}(k)^c$ or, equivalently, $c \in \overline{\mathcal{P}}(k)$. Again, since each node on the path $\overline{\mathcal{P}}(k)$ from k has only one child on the path, there is at most one node c satisfying these conditions and it lies on the path $\overline{\mathcal{P}}(k)$.

Case 4: $j \in \overline{\mathcal{P}}(k)$, $j > k$, and $c \in \pi^{-1}(j) \setminus \overline{\pi}^{-1}(j)$. In this case, c is a child of j in the old elimination tree, but not in the new tree, and the set $\mathcal{L}_c$ should be subtracted from $\overline{\mathcal{L}}_j^\sharp$ since it was previously added to $\mathcal{L}_j^\sharp$. Since $\pi(p) = \overline{\pi}(p)$ for each $p \in \overline{\mathcal{P}}(k)^c$, the fact that $\pi(c) = j \neq \overline{\pi}(c)$ implies that $c \in \overline{\mathcal{P}}(k)$. In the algorithm that follows, we refer to nodes c that satisfy these conditions as *lost children*. A node j in the elimination tree can lose multiple children.

In each of the cases above, every node c that led to adjustments in the pattern was located on the path $\overline{\mathcal{P}}(k)$. To make these changes, Algorithm 3 (below) simply marches up the path $\overline{\mathcal{P}}(k)$ from k to the root making the adjustments enumerated in Cases 1 through 4 above. Consider a node c. If its parent changes, we have $\overline{\pi}(c) < \pi(c)$ by Proposition 3.2. Both the new parent $\overline{\pi}(c)$ and the old parent $\pi(c)$ are on the path $\overline{\mathcal{P}}(k)$. Node c is a new child of $\overline{\pi}(c)$ (Case 3), which is the next node in the path $\overline{\mathcal{P}}(k)$. Node c is a lost child of $\pi(c)$ (Case 4). That is, one node's new child is another node's lost child. If Algorithm 3 is at node $j = \overline{\pi}(c)$ and we notice a single new child c, we can place that node in a *lost-child-queue* for node $\pi(c)$ and process that queue when we come to the node $\pi(c)$ later in the path. We could instead modify node $\pi(c)$ the moment we find that it loses a child, but this could not be done in a simple left-to-right pass of the columns corresponding to the nodes in the path $\overline{\mathcal{P}}(k)$. As we will see in section 5, this will allow us to combine the symbolic and numeric algorithms into a single pass.

ALGORITHM 3 (symbolic update, add new column $\mathbf{w}$).

```
    Case 1: first node in the path
    $\mathcal{W} = \{i : w_i \neq 0\}$
    $k = \min\ \mathcal{W}$
    $\overline{\mathcal{L}}_k^\sharp = \mathcal{L}_k^\sharp + \mathcal{W}$
    $\overline{\pi}(k) = \min\ \overline{\mathcal{L}}_k \setminus \{k\}$
    $c = k$
    $j = \overline{\pi}(c)$
    while $j \neq 0$ do
        if $j = \pi(c)$ then
            Case 2: c is an old child of j, possibly changed
            $\overline{\mathcal{L}}_j^\sharp = \mathcal{L}_j^\sharp + (\overline{\mathcal{L}}_c \setminus \mathcal{L}_c)$
        else
            Case 3: c is a new child of j and a lost child of $\pi(c)$
            $\overline{\mathcal{L}}_j^\sharp = \mathcal{L}_j^\sharp + (\overline{\mathcal{L}}_c \setminus \{c\})$
            place c in lost-child-queue of $\pi(c)$
        end if
        Case 4: consider each lost child of j
        for each c in lost-child-queue of j do
            $\overline{\mathcal{L}}_j^\sharp = \overline{\mathcal{L}}_j^\sharp - (\mathcal{L}_c \setminus \{c\})$
        end for
        $\overline{\pi}(j) = \min\ \overline{\mathcal{L}}_j \setminus \{j\}$
        $c = j$
        $j = \overline{\pi}(c)$
    end while
    $\overline{\mathcal{L}}_j^\sharp = \mathcal{L}_j^\sharp$ and $\overline{\pi}(j) = \pi(j)$ for all $j \in \overline{\mathcal{P}}(k)^c$
end Algorithm 3
```

The time taken by this algorithm is given by the following lemma.

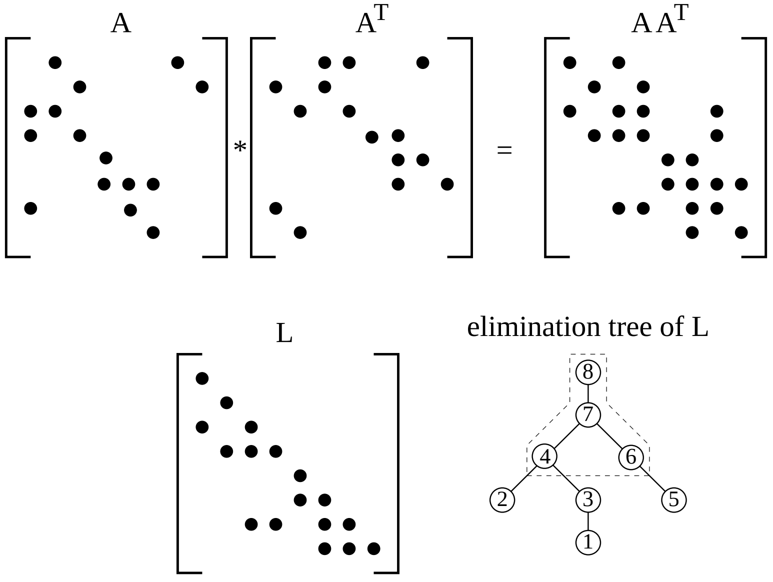


Fig. 4.1. *An example matrix, its factor, and its elimination tree.*

Lemma 4.2. *The time to execute Algorithm* 3 *is bounded above by a constant times the number of entries associated with patterns for nodes on the new path* $\overline{\mathcal{P}}(k)$*. That is, the time is*

$$O\left(\sum_{j\in\overline{\mathcal{P}}(k)} |\overline{\mathcal{L}}_j|\right).$$

Proof. In Algorithm 3, we simply march up the path $\overline{\mathcal{P}}(k)$ making adjustments to $\overline{\mathcal{L}}_j^\sharp$ as we proceed. At each node j, preparing column j for all set subtractions or additions takes $O(|\mathcal{L}_j|)$ time (a scatter operation), and it takes $O(|\overline{\mathcal{L}}_j|)$ time to gather the results at the end of step j. For each child of j, the set subtraction/addition adds time proportional to the size of the subtracted/added set. Each node is visited as a child c at most twice, since it falls into one or two of the four cases enumerated above (a node c can be a new child of one node and a lost child of another). If this work (proportional to $|\mathcal{L}_c|$ or $|\overline{\mathcal{L}}_c|$) is accounted to step c instead of j, the time to make the adjustment to the pattern is bounded above by a constant times either $|\mathcal{L}_j|$ or $|\overline{\mathcal{L}}_j|$. Since $|\mathcal{L}_j| \le |\overline{\mathcal{L}}_j|$ by Theorem 4.1, the proof is complete. □

In practice, we can reduce the execution time for Algorithm 3 by skipping over the current node j if it has no lost children and if its child c falls under Case 2 with $\overline{\mathcal{L}}_c = \mathcal{L}_c$. This check can be made in constant time, and if true, implies that $\mathcal{L}_j^\sharp = \overline{\mathcal{L}}_j^\sharp$.

We illustrate Algorithm 3 with an example. Figure 4.1 shows the nonzero pattern of an 8-by-8 matrix $\mathbf{A}$, the patterns of $\mathbf{A}\mathbf{A}^\mathsf{T}$ and its factor $\mathbf{L}$, and the elimination tree of $\mathbf{L}$. The highlighted portion of the elimination tree of $\mathbf{L}$ corresponds to the nodes in the path $\overline{\mathcal{P}}(k)$ if $\mathbf{A}\mathbf{A}^\mathsf{T}$ is updated with $\mathbf{w}\mathbf{w}^\mathsf{T}$, where the nonzero pattern of $\mathbf{w}$ is $\mathcal{W} = \{4, 6, 8\}$. The updated matrix $\overline{\mathbf{A}}\,\overline{\mathbf{A}}^\mathsf{T}$, its factor $\overline{\mathbf{L}}$, and the elimination tree of $\overline{\mathbf{L}}$ are shown in Figure 4.2. The new column $\mathbf{w}$ is appended to the original matrix $\mathbf{A}$.

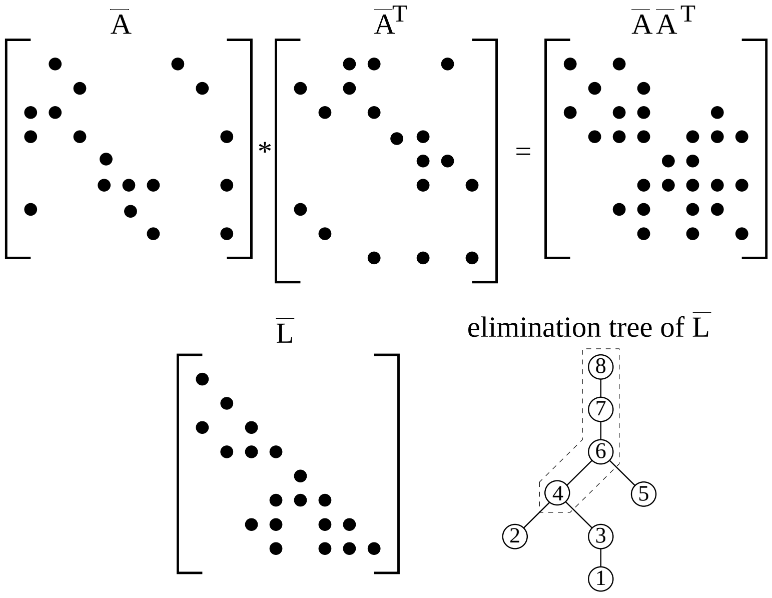


FIG. 4.2. *An example matrix after update.*

For this example, we have $\mathcal{W} = \{4, 6, 8\}$ and thus $k = 4$. The pattern of column 4 of $\overline{\mathbf{L}}$ falls under Case 1 and becomes $\overline{\mathcal{L}}_4 = \{4, 6, 7, 8\}$. The new parent of node 4 is node 6. At node 6, we find that $c = 4$ is a lost child of $\pi(4) = 7$ and a new child of node 6 (Case 3). The pattern of column 6 does not change, although its multiplicities do. The parent of node 6 is thus unchanged (node 7). Node 4 is placed in column 7's lost-child-queue. Node 6's lost-child-queue is empty. At column 7, we find that $c = 6$ is the old child of node 7 (Case 2), and node 4 is found in node 7's lost-child-queue (Case 4). This changes the multiplicities of column 7, but not its pattern. Finally, at node 8, nothing changes.

4.2. Symbolic downdate algorithm. Let us consider the removal of a column $\mathbf{w}$ from $\mathbf{A}$, and let k be the index of the first nonzero entry in $\mathbf{w}$. The symbolic downdate algorithm is analogous to the symbolic update algorithm, but the roles of $\mathcal{P}(k)$ and $\overline{\mathcal{P}}(k)$ are interchanged in accordance with Theorem 4.1. Instead of adding entries to $\mathcal{L}_j^\sharp$, we subtract entries; instead of lost-child-queues, we have new-child-queues; instead of walking up the path $\overline{\mathcal{P}}(k)$, we walk up the path $\mathcal{P}(k) \supseteq \overline{\mathcal{P}}(k)$.

ALGORITHM 4 (symbolic downdate, remove column $\mathbf{w}$).

```
Case 1: first node in the path
$\mathcal{W} = \{i : w_i \neq 0\}$
$k = \min\ \mathcal{W}$
$\overline{\mathcal{L}}_k^\sharp = \mathcal{L}_k^\sharp - \mathcal{W}$
$\overline{\pi}(k) = \min\ \overline{\mathcal{L}}_k \setminus \{k\}$
$c = k$
$j = \pi(c)$
while $j \neq 0$ do
    if $j = \overline{\pi}(c)$ then
```

```
                *Case* 2: *c is an old child of j, possibly changed*
                $\overline{\mathcal{L}}_j^\sharp = \mathcal{L}_j^\sharp - (\mathcal{L}_c \setminus \overline{\mathcal{L}}_c)$
            **else**
                *Case* 3: *c is a lost child of j and a new child of* $\overline{\pi}(c)$
                $\overline{\mathcal{L}}_j^\sharp = \mathcal{L}_j^\sharp - (\mathcal{L}_c \setminus \{c\})$
                place $c$ in new-child-queue of $\overline{\pi}(c)$
            **end if**
            *Case* 4: *consider each new child of j*
            **for** each $c$ in new-child-queue of $j$ **do**
                $\overline{\mathcal{L}}_j^\sharp = \overline{\mathcal{L}}_j^\sharp + (\overline{\mathcal{L}}_c \setminus \{c\})$
            **end for**
            $\overline{\pi}(j) = \min\ \overline{\mathcal{L}}_j \setminus \{j\}$
            $c = j$
            $j = \pi(c)$
        **end while**
        $\overline{\mathcal{L}}_j^\sharp = \mathcal{L}_j^\sharp$ and $\overline{\pi}(j) = \pi(j)$ for all $j \in \mathcal{P}(k)^c$
    **end Algorithm 4**
```

Similar to Algorithm 3, the execution time obeys the following estimate.

LEMMA 4.3. *The time to execute Algorithm 4 is bounded above by a constant times the number of entries associated with patterns for nodes on the old path $\mathcal{P}(k)$. That is, the time is*

$$O\left(\sum_{j\in\mathcal{P}(k)} |\mathcal{L}_j|\right).$$

We can skip over some nodes that do not change in a similar manner as Algorithm 3. Figures 4.1 and 4.2 can be viewed as an example of symbolic downdate, $\overline{\mathbf{A}}\,\overline{\mathbf{A}}^{\mathsf{T}} - \mathbf{w}\mathbf{w}^{\mathsf{T}} = \mathbf{A}\mathbf{A}^{\mathsf{T}}$, where $\mathbf{w}$ is the ninth column of $\overline{\mathbf{A}}$. The roles of $\mathbf{A}$ and $\overline{\mathbf{A}}$ are reversed.

5. The numerical factors.

When we add or delete a column in $\mathbf{A}$, we update or downdate the symbolic factorization in order to determine the location in the Cholesky factor of either new nonzero entries or old nonzero entries that are now zero. Knowing the location of the nonzero entries, we can update the numerical value of these entries. We first consider the case when $\mathbf{A}$ and $\mathbf{L}$ are dense and draw on the ideas of [20]. Then we show how the method extends to the sparse case.

5.1. Modifying a dense factorization.

Our algorithm to implement the numerical update and downdate is based on a modification of Method C1 in [20] for dense matrices. Although this algorithm is portrayed as a nonorthogonal algorithm in [20], it is equivalent, after a diagonal scaling, to the orthogonal algorithm of Pan [4, 30]. Hence, Method C1 should possess strong numerical stability properties comparable to those of Pan's method. Other dense update or downdate methods include [2, 5, 21, 22].

To summarize Gill et al.'s approach, we can write

$$\overline{\mathbf{A}}\,\overline{\mathbf{A}}^{\mathsf{T}} = \mathbf{A}\mathbf{A}^{\mathsf{T}}+\sigma\mathbf{w}\mathbf{w}^{\mathsf{T}} = \mathbf{L}\mathbf{D}\mathbf{L}^{\mathsf{T}}+\sigma\mathbf{w}\mathbf{w}^{\mathsf{T}} = \mathbf{L}(\mathbf{D}+\sigma\mathbf{v}\mathbf{v}^{\mathsf{T}})\mathbf{L}^{\mathsf{T}} = \mathbf{L}(\tilde{\mathbf{L}}\tilde{\mathbf{D}}\tilde{\mathbf{L}}^{\mathsf{T}})\mathbf{L}^{\mathsf{T}} = \overline{\mathbf{L}}\,\overline{\mathbf{D}}\,\overline{\mathbf{L}}^{\mathsf{T}},$$

where $\mathbf{v} = \mathbf{L}^{-1}\mathbf{w}$, $\overline{\mathbf{L}} = \mathbf{L}\tilde{\mathbf{L}}$, and $\overline{\mathbf{D}} = \tilde{\mathbf{D}}$. In Algorithm 5 below, we evaluate the new Cholesky factor $\overline{\mathbf{L}} = \mathbf{L}\tilde{\mathbf{L}}$ without forming $\tilde{\mathbf{L}}$ explicitly [20] by taking advantage of the special structure of $\tilde{\mathbf{L}}$. The product is computed column by column, moving from left to right. In practice, $\mathbf{L}$ can be overwritten with $\overline{\mathbf{L}}$.

Algorithm 5 (dense numeric update/downdate; Method C1, modified).

```
$\alpha = 1$
**for** $j = 1$ to $m$ **do**
    $\overline{\alpha} = \alpha + \sigma w_j^2 / d_j$
    $\gamma = w_j / (\overline{\alpha} d_j)$
    $\overline{d}_j = (\overline{\alpha}/\alpha) d_j$
    $\alpha = \overline{\alpha}$
    $\mathbf{w}_{j+1,\ldots,m} = \mathbf{w}_{j+1,\ldots,m} - w_j \mathbf{L}_{j+1,\ldots,m,j}$
    $\overline{\mathbf{L}}_{j+1,\ldots,m,j} = \mathbf{L}_{j+1,\ldots,m,j} + \sigma\gamma \mathbf{w}_{j+1,\ldots,m}$
**end for**
```

Note that Algorithm 5 also overwrites $\mathbf{w}$ with $\mathbf{v} = \mathbf{L}^{-1}\mathbf{w}$. In the jth iteration, w_j is simply equal to v_j. This observation is important to the sparse case discussed in the next section. Since σ is ± 1, the floating point operation count of Algorithm 5 is precisely $2m^2 + 5m$, counting multiplications, divisions, subtractions, and additions separately. As noted above, if we introduce a diagonal scaling in Algorithm 5, we obtain Pan's method [4, 30] for modifying the sparse $\mathbf{LL}^\mathsf{T}$ factorization, for which the corresponding operation count is $2.5m^2 + 6.5m$ plus an additional m square roots. Whether we use Algorithm 5 or Pan's algorithm to update the numerical factors, the symbolic algorithms are unchanged.

5.2. Modifying a sparse factorization. In the sparse case, $\mathbf{v} = \mathbf{L}^{-1}\mathbf{w}$ is sparse and its nonzero pattern is crucial. In Algorithm 5, we can essentially by-pass those executable statements associated with values of j for which the variable w_j vanishes. That is, when w_j vanishes, the values of α, d_j, $\mathbf{w}$, and column j of $\mathbf{L}$ are unchanged. Since w_j has been overwritten by v_j, it follows that when executing Algorithm 5, column j of $\mathbf{L}$ changes only when v_j does not vanish. The nonzero pattern of $\mathbf{v}$ can be found using the following lemma. The lemma is based on the directed graph $G(\mathbf{L}^\mathsf{T}) = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{1, 2, \ldots, m\}$ is the vertex set and $\mathcal{E} = \{(j, i) \mid i \in \mathcal{L}_j\}$ is the directed edge set.[2]

Lemma 5.1. *The nodes reachable from any given node k by path(s) in the directed graph $G(\mathbf{L}^\mathsf{T})$ coincide with the path $\mathcal{P}(k)$.*

Proof. If $\mathcal{P}(k)$ has a single element, the lemma holds. Proceeding by induction, suppose that the lemma holds for all k for which $|\mathcal{P}(k)| \le j$. Now, if $\mathcal{P}(k)$ has $j+1$ elements, then by the induction hypothesis, the nodes reachable from $\pi(k)$ by path(s) in the directed graph $G(\mathbf{L}^\mathsf{T})$ coincide with the path $\mathcal{P}(\pi(k))$. The nodes reachable in one step from k consist of the elements of $\mathcal{L}_k$. By Proposition 3.1, each of the elements of $\mathcal{L}_k$ is contained in the path $\mathcal{P}(k)$. If $i \in \mathcal{L}_k$, $i \ne k$, then $|\mathcal{P}(i)| \le j$. By the induction hypothesis, the nodes reachable from i coincide with $\mathcal{P}(i) \subseteq \mathcal{P}(k)$. The nodes reachable from k consist of the union of $\{k\}$ with the nodes reachable from $\mathcal{L}_k$. Since $k \in \mathcal{P}(k)$, it follows that the nodes reachable from k are contained in $\mathcal{P}(k)$. On the other hand, for each p, the element of $\mathbf{L}^\mathsf{T}$ in row $\pi^p(k)$ and column $\pi^{p+1}(k)$ is nonzero. Hence, all the elements of $\mathcal{P}(k)$ are reachable from k. Since the nodes in $\mathcal{P}(k)$ coincide with the nodes reachable from k by path(s) in the directed graph $G(\mathbf{L}^\mathsf{T})$, the induction step is complete. ∎

[2] Note that this definition of $G(\mathbf{L}^\mathsf{T})$ includes self-loops corresponding to the diagonal entries of $\mathbf{L}$.

Theorem 5.2. *During symbolic downdate* $\overline{\mathbf{A}}\,\overline{\mathbf{A}}^{\mathsf{T}} = \mathbf{A}\mathbf{A}^{\mathsf{T}} - \mathbf{w}\mathbf{w}^{\mathsf{T}}$ *(where* $\mathbf{w}$ *is a column of* $\mathbf{A}$*), the nonzero pattern of* $\mathbf{v} = \mathbf{L}^{-1}\mathbf{w}$ *is equal to the path* $\mathcal{P}(k)$ *in the (old) elimination tree of* $\mathbf{L}$ *where*

$$k = \min\,\{i : w_i \neq 0\}. \tag{5.1}$$

Proof. Let $\mathcal{W} = \{i : w_i \neq 0\}$. Theorem 5.1 of Gilbert [15, 16, 19] states that the nonzero pattern of $\mathbf{v}$ is the set of nodes reachable from the nodes in $\mathcal{W}$ by paths in the directed graph $G(\mathbf{L}^{\mathsf{T}})$. By Algorithm 1, $\mathcal{W} \subseteq \mathcal{L}_k$. Hence, each element of $\mathcal{W}$ is reachable from k by a path of length one, and the nodes reachable from $\mathcal{W}$ are a subset of the nodes reachable from k. Conversely, since $k \in \mathcal{W}$, the nodes reachable from k are a subset of the nodes reachable from $\mathcal{W}$. Combining these inclusions, the nodes reachable from k and from $\mathcal{W}$ are the same, and by Lemma 5.1, the nodes reachable from k coincide with the path $\mathcal{P}(k)$. □

Corollary 5.3. *During symbolic update* $\overline{\mathbf{A}}\,\overline{\mathbf{A}}^{\mathsf{T}} = \mathbf{A}\mathbf{A}^{\mathsf{T}} + \mathbf{w}\mathbf{w}^{\mathsf{T}}$*, the nonzero pattern of* $\mathbf{v} = \mathbf{L}^{-1}\mathbf{w}$ *is equal to the path* $\overline{\mathcal{P}}(k)$ *in the (new) elimination tree of* $\overline{\mathbf{L}}$ *where* k *is defined in* (5.1)*.*

Proof. Since $\mathbf{L}\mathbf{D}\mathbf{L}^{\mathsf{T}} = \overline{\mathbf{A}}\,\overline{\mathbf{A}}^{\mathsf{T}} - \mathbf{w}\mathbf{w}^{\mathsf{T}}$, we can view $\mathbf{L}$ as the Cholesky factor for the downdate $\overline{\mathbf{A}}\,\overline{\mathbf{A}}^{\mathsf{T}} - \mathbf{w}\mathbf{w}^{\mathsf{T}}$. Hence, we can apply Theorem 5.2, in effect replacing $\mathcal{P}$ by $\overline{\mathcal{P}}$. □

As a result of Theorem 5.2, a sparse downdate algorithm can skip over any column $j \in \mathcal{P}(k)^c$. Similarly, as a result of Corollary 5.3, a sparse update algorithm can skip over any column $j \in \overline{\mathcal{P}}(k)^c$. In both cases, the jth iteration of Algorithm 5 requires both the old and new nonzero patterns of the jth column of $\mathbf{L}$ (that is, $\mathcal{L}_j$ and $\overline{\mathcal{L}}_j$). These are computed in the symbolic update and downdate Algorithms 3 and 4. Finally, note that the symbolic and numeric algorithms have the same structure. They both iterate over the columns in the associated path. Therefore, Algorithms 3 and 5 can be combined to obtain a complete sparse update algorithm that makes just one pass of the matrix $\mathbf{L}$. The jth iteration of Algorithm 5 is placed just before the $j = \overline{\pi}(c)$ statement in Algorithm 3. Since $\mathcal{L}_j \subseteq \overline{\mathcal{L}}_j$ during a sparse update, the total time taken for Algorithms 3 and 5 is

$$O\left(\sum_{j \in \overline{\mathcal{P}}(k)} |\overline{\mathcal{L}}_j|\right).$$

Similarly, Algorithms 4 and 5 can be combined to obtain a complete sparse downdate algorithm. Since $\overline{\mathcal{L}}_j \subseteq \mathcal{L}_j$ during a sparse downdate, the time taken for Algorithms 4 and 5 is

$$O\left(\sum_{j \in \mathcal{P}(k)} |\mathcal{L}_j|\right).$$

This time can be much less than the $O(m^2)$ time taken by Algorithm 5 in the dense case.

6. Arbitrary symbolic and numerical factors. The methods we have developed for computing the modification to the Cholesky factors of $\mathbf{A}\mathbf{A}^{\mathsf{T}}$ corresponding to the addition or deletion of columns in $\mathbf{A}$ can be used to determine the effect on the Cholesky factors of a general symmetric positive definite matrix $\mathbf{M}$ of any symmetric change of the form $\mathbf{M} + \sigma \mathbf{w}\mathbf{w}^{\mathsf{T}}$ that preserves positive definiteness. We briefly describe how Algorithms 1 through 5 are modified for the general case.

Let $\mathcal{M}_j$ denote the nonzero pattern of the *lower triangular* part of $\mathbf{M}$:

$$\mathcal{M}_j = \{i \,:\, m_{ij} \neq 0 \text{ and } i \geq j\}.$$

The symbolic factorization of $\mathbf{M}$ [10, 11, 12, 13, 34] is obtained by replacing the union of $\mathcal{A}_k$ terms in Algorithm 1 with the set $\mathcal{M}_j$. With this change, $\mathcal{L}_j$ of Algorithm 1 is given by

$$\mathcal{L}_j = \{j\} \cup \left(\bigcup_{c \in \pi^{-1}(j)} \mathcal{L}_c \setminus \{c\} \right) \cup \mathcal{M}_j.$$

This leads to a change in Algorithm 2 for computing the multiplicities. The multiplicity of an index i in $\mathcal{L}_j$ becomes

$$m(i,j) = |\{k \in \pi^{-1}(j) : i \in \mathcal{L}_k\}| + (1 \text{ if } i \in \mathcal{M}_j, \text{ or } 0 \text{ otherwise}).$$

The loop involving the $\mathcal{A}_k$ terms in Algorithm 2 is replaced by the single statement $\mathcal{L}_j^\sharp = \mathcal{L}_j^\sharp + \mathcal{M}_j$. More precisely, we have

$$\left.\begin{array}{l} \textbf{for } \text{each } k \text{ where } \min \mathcal{A}_k = j \textbf{ do} \\ \qquad \mathcal{L}_j^\sharp = \mathcal{L}_j^\sharp + \mathcal{A}_k \\ \textbf{end for} \end{array}\right\} \quad \Rightarrow \quad \mathcal{L}_j^\sharp = \mathcal{L}_j^\sharp + \mathcal{M}_j.$$

Entries are removed or added symbolically from $\mathbf{A}\mathbf{A}^{\mathsf{T}}$ by the deletion or addition of columns of $\mathbf{A}$, and numerical cancellation is ignored. Numerical cancellation of entries in $\mathbf{M}$ should not be ignored, however, because this is the only way that entries can be dropped from $\mathbf{M}$. When numerical cancellation is taken into account, neither of the inclusions $\mathcal{M}_j \subseteq \overline{\mathcal{M}}_j$ nor $\overline{\mathcal{M}}_j \subseteq \mathcal{M}_j$ may hold. We resolve this problem by using a symbolic modification scheme with two steps: a symbolic update phase in which new nonzero entries in $\mathbf{M} + \sigma \mathbf{w}\mathbf{w}^{\mathsf{T}}$ are taken into account, followed by a separate symbolic downdate phase to handle entries that become numerically zero. Since each modification step now involves an update phase followed by a downdate phase, we attach (in this section) an overbar to quantities associated with the update and an underbar to quantities associated with the downdate.

Let $\mathcal{W}$ be the nonzero pattern of $\mathbf{w}$, namely, $\mathcal{W} = \{i \,:\, w_i \neq 0\}$. In the first symbolic phase, entries from $\mathcal{W}$ are symbolically added to $\mathcal{M}_j$ for each $j \in \mathcal{W}$. That is, if $i \notin \mathcal{M}_j$, but $i, j \in \mathcal{W}$ with $i > j$, then we add i to $\mathcal{M}_j$:

$$\overline{\mathcal{M}}_j = \mathcal{M}_j \cup \{i \in \mathcal{W} : i > j\}.$$

In the second symbolic phase, entries from $\mathcal{W}$ are symbolically deleted for each $j \in \mathcal{W}$:

$$\underline{\mathcal{M}}_j = \overline{\mathcal{M}}_j \setminus \{i \in \mathcal{W} : i > j, \;\; m_{ij} + \sigma w_i w_j = 0\}. \tag{6.1}$$

In practice, we need to introduce a drop tolerance t and replace the equality

$$m_{ij} + \sigma w_i w_j = 0$$

in (6.1) by the inequality $|m_{ij} + \sigma w_i w_j| \le t$. For a general matrix, the analogue of Theorem 4.1 is the following.

THEOREM 6.1. *If α is the first index for which $\mathcal{M}_\alpha \neq \overline{\mathcal{M}}_\alpha$, then $\mathcal{P}(\alpha) \subseteq \overline{\mathcal{P}}(\alpha)$. Moreover, $\overline{\mathcal{L}}_i^\sharp = \mathcal{L}_i^\sharp$ for all $i \in \overline{\mathcal{P}}(\alpha)^c$. If β is the first index for which $\overline{\mathcal{M}}_\beta \neq \underline{\mathcal{M}}_\beta$, then $\underline{\mathcal{P}}(\beta) \subseteq \overline{\mathcal{P}}(\beta)$. Moreover, $\underline{\mathcal{L}}_i^\sharp = \overline{\mathcal{L}}_i^\sharp$ for all $i \in \overline{\mathcal{P}}(\beta)^c$.*

In evaluating the modification in the symbolic factorization associated with $\mathbf{M} + \sigma \mathbf{w}\mathbf{w}^\mathsf{T}$, we start at the first index α where $\mathcal{M}_\alpha \neq \overline{\mathcal{M}}_\alpha$ and we march up the path $\overline{\mathcal{P}}(\alpha)$ making changes to $\mathcal{L}_j^\sharp$, obtaining $\overline{\mathcal{L}}_j^\sharp$. In the second phase, we start at the first index where $\overline{\mathcal{M}}_\beta \neq \underline{\mathcal{M}}_\beta$ and we march up the path $\overline{\mathcal{P}}(\beta)$ making changes to $\overline{\mathcal{L}}_j^\sharp$, obtaining $\underline{\mathcal{L}}_j^\sharp$. The analogue of Algorithm 3 in the general case differs only in the starting index (now α) and in the addition of the sets $\overline{\mathcal{M}}_j \setminus \mathcal{M}_j$ in each pass through the j-loop.

ALGORITHM 6a (symbolic update phase, general matrix).

```
Case 1: first node in the path
$\alpha = \min \{i : \mathcal{M}_i \neq \overline{\mathcal{M}}_i\}$
$\overline{\mathcal{L}}_\alpha^\sharp = \mathcal{L}_\alpha^\sharp + \overline{\mathcal{M}}_\alpha \setminus \mathcal{M}_\alpha$
$\overline{\pi}(\alpha) = \min\ \overline{\mathcal{L}}_\alpha \setminus \{\alpha\}$
$c = \alpha$
$j = \overline{\pi}(c)$
while $j \neq 0$ do
    $\overline{\mathcal{L}}_j^\sharp = \mathcal{L}_j^\sharp + \overline{\mathcal{M}}_j \setminus \mathcal{M}_j$
    if $j = \pi(c)$ then
        Case 2: c is an old child of j, possibly changed
        $\overline{\mathcal{L}}_j^\sharp = \overline{\mathcal{L}}_j^\sharp + (\overline{\mathcal{L}}_c \setminus \mathcal{L}_c)$
    else
        Case 3: c is a new child of j and a lost child of $\pi(c)$
        $\overline{\mathcal{L}}_j^\sharp = \overline{\mathcal{L}}_j^\sharp + (\overline{\mathcal{L}}_c \setminus \{c\})$
        place $c$ in lost-child-queue of $\pi(c)$
    end if
    Case 4: consider each lost child of j
    for each $c$ in lost-child-queue of $j$ do
        $\overline{\mathcal{L}}_j^\sharp = \overline{\mathcal{L}}_j^\sharp - (\mathcal{L}_c \setminus \{c\})$
    end for
    $\overline{\pi}(j) = \min\ \overline{\mathcal{L}}_j \setminus \{j\}$
    $c = j$
    $j = \overline{\pi}(c)$
end while
$\overline{\mathcal{L}}_j^\sharp = \mathcal{L}_j^\sharp$ and $\overline{\pi}(j) = \pi(j)$ for all $j \in \overline{\mathcal{P}}(\alpha)^c$
end Algorithm 6a
```

Similarly, the analogue of Algorithm 4 in the general case differs only in the starting index (now β) and in the subtraction of the sets $\overline{\mathcal{M}}_j \setminus \underline{\mathcal{M}}_j$ in each pass through the j-loop.

ALGORITHM 6b (symbolic downdate phase, general matrix).

```
Case 1: first node in the path
$\beta = \min \{i : \overline{\mathcal{M}}_i \neq \underline{\mathcal{M}}_i\}$
$\underline{\mathcal{L}}_\beta^\sharp = \overline{\mathcal{L}}_\beta^\sharp - \overline{\mathcal{M}}_\beta \setminus \underline{\mathcal{M}}_\beta$
$\underline{\pi}(\beta) = \min\ \underline{\mathcal{L}}_\beta \setminus \{\beta\}$
```

```
        $c = \beta$
        $j = \overline{\pi}(c)$
        while $j \neq 0$ do
            $\underline{\mathcal{L}}_j^\sharp = \overline{\mathcal{L}}_j^\sharp - \overline{\mathcal{M}}_j \setminus \underline{\mathcal{M}}_j$
            if $j = \underline{\pi}(c)$ then
                Case 2: c is an old child of j, possibly changed
                $\underline{\mathcal{L}}_j^\sharp = \underline{\mathcal{L}}_j^\sharp - (\overline{\mathcal{L}}_c \setminus \underline{\mathcal{L}}_c)$
            else
                Case 3: c is a lost child of j and a new child of $\underline{\pi}(c)$
                $\underline{\mathcal{L}}_j^\sharp = \underline{\mathcal{L}}_j^\sharp - (\overline{\mathcal{L}}_c \setminus \{c\})$
                place $c$ in new-child-queue of $\underline{\pi}(c)$
            end if
            Case 4: consider each new child of j
            for each $c$ in new-child-queue of $j$ do
                $\underline{\mathcal{L}}_j^\sharp = \underline{\mathcal{L}}_j^\sharp + (\underline{\mathcal{L}}_c \setminus \{c\})$
            end for
            $\underline{\pi}(j) = \min\ \underline{\mathcal{L}}_j \setminus \{j\}$
            $c = j$
            $j = \overline{\pi}(c)$
        end while
        $\underline{\mathcal{L}}_j^\sharp = \overline{\mathcal{L}}_j^\sharp$ and $\underline{\pi}(j) = \overline{\pi}(j)$ for all $j \in \overline{\mathcal{P}}(\beta)^c$
    end Algorithm 6b
```

Algorithm 5 is completely unchanged in the general case. It can be applied after the completion of Algorithm 6b so that we know the location of new nonzero entries in the Cholesky factor. It processes the submatrix associated with rows and columns in $\overline{\mathcal{P}}(k)$, where k is the index of the first nonzero element of $\mathbf{w}$. When $\mathbf{M}$ has the form $\mathbf{AA}^\mathsf{T}$ and when $\overline{\mathbf{M}}$ is found by either adding or deleting a column in $\mathbf{A}$, then assuming no numerical cancellations, Algorithm 6b can be skipped when we add a column to $\mathbf{A}$ since $\overline{\mathcal{M}}_j = \underline{\mathcal{M}}_j$ for each j. Similarly, when a column is removed from $\mathbf{A}$, Algorithm 6a can be skipped since $\overline{\mathcal{M}}_j = \mathcal{M}_j$ for each j. Hence, when Algorithm 6a followed by Algorithm 6b is applied to a matrix of the form $\mathbf{AA}^\mathsf{T}$, only Algorithm 6a takes effect during an update, while only Algorithm 6b takes effect during a downdate. Thus the approach we have presented in this section for an arbitrary symmetric positive definite matrix generalizes the earlier approach where we focus on matrices of the form $\mathbf{AA}^\mathsf{T}$.

7. Experimental results. We have developed Matlab codes to experiment with all the algorithms presented in this paper, including the algorithms of section 6 for a general symmetric, positive definite matrix. In this section, we present the results of a numerical experiment with a large sparse optimization problem from Netlib [8] in the context of the LP DASA [26]. The computer used for this experiment was a Model 170 UltraSparc, equipped with 256MB of memory and with Matlab Version 4.2c.

7.1. Experimental design. In the LP DASA, the columns of the matrix $\mathbf{A}$ in the product $\mathbf{AA}^\mathsf{T}$ are all chosen from among the columns of some fixed matrix $\mathbf{B}$. After a few updates or downdates, the system $\mathbf{AA}^\mathsf{T}\mathbf{x} = \mathbf{b}$ must be solved with a dense right-hand side $\mathbf{b}$.

The Cholesky factorization of the initial $\mathbf{AA}^\mathsf{T}$ (before any columns are added or deleted) is often preceded by a fill-reducing permutation $\mathbf{P}$ of the rows and columns

([1], for example). We can compute a permutation to reduce the fill for $\mathbf{BB}^\mathsf{T}$ since the Cholesky factors of $\mathbf{AA}^\mathsf{T}$ will be at least as sparse as those of $\mathbf{BB}^\mathsf{T}$ by Proposition 3.2, regardless of how the columns of $\mathbf{A}$ are chosen from the columns of $\mathbf{B}$. Based on the number of nonzeros in each column of the Cholesky factors of $\mathbf{BB}^\mathsf{T}$, we allocate a static storage structure that will always contain the Cholesky factors of each $\mathbf{AA}^\mathsf{T}$. This can lead to wasted space if the number of nonzeros in the Cholesky factors of $\mathbf{AA}^\mathsf{T}$ is far less than the number of nonzeros in the Cholesky factors of $\mathbf{BB}^\mathsf{T}$. Alternatively, we could store the Cholesky factor of the current $\mathbf{AA}^\mathsf{T}$ in a smaller space and reallocate storage during the updates and downdates, based on the changes in the nonzero patterns.

We selected an optimization problem from airline scheduling (DFL001). Its constraint matrix $\mathbf{B}$ is 6,071-by-12,230 with 35,632 nonzeros. We rescaled the variables so that the columns of $\mathbf{B}$ are all unit vectors, and we augmented $\mathbf{B}$ by appending on its right the matrix $10^{-6}\mathbf{I}$. This ensures that $\mathbf{BB}^\mathsf{T}$ is strictly positive definite. The matrix $\mathbf{BB}^\mathsf{T}$ has 37,923 nonzeros in its strictly lower triangular part. The Cholesky factor $\mathbf{L}_B$ has 1.49 million nonzeros (with a fill-minimizing permutation $\mathbf{P}_B$ of the rows of $\mathbf{B}$, described below) and requires 1.12 billion floating point operations and 115 seconds to compute (the LP DASA does not require this matrix; however, as noted above, this is an upper bound on the number of nonzeros that can occur during the execution of the LP DASA). This high level of fill-in in $\mathbf{L}_B$ is the result of the highly irregular nonzero pattern of $\mathbf{B}$. The matrix $\mathbf{A}_0$, corresponding to an optimal solution of the linear programming problem, has 5,446 columns taken from the original columns of $\mathbf{B}$ plus the 6,071 columns of the appended matrix $10^{-6}\mathbf{I}$.

We wrote a set of Matlab scripts that implements our complete sparse Cholesky update/downdate algorithm, discussed in section 4 and section 5. We first found $\mathbf{P}_B$, using 101 trials of Matlab's column multiple minimum degree ordering algorithm (`colmmd` [17]), 100 of them with a different random permutation of the rows of $\mathbf{B}$. We then took the best permutation found. The time for the initial Cholesky factorization of $\mathbf{AA}^\mathsf{T}$ is given by (see [13]) the following expression:

$$O\left(\sum_{j=1}^{m} |\mathcal{L}_j|^2\right),$$

which is $O(m^3)$ if $\mathbf{L}$ is dense. With our permutation $\mathbf{P}_B$, the factor $\mathbf{L}$ of $\mathbf{A}_0\mathbf{A}_0^\mathsf{T}$ has 831 thousand nonzeros and took 481 million floating point operations and 51 seconds to compute (using Matlab's `chol`). Following the method used in LP DASA, we added 10^{-12} to the diagonal to ensure positive definiteness (the columns of $\mathbf{B}$ are scaled to be unit vectors). We used the same permutation $\mathbf{P}_B$ for the entire experiment. The initial symbolic factorization took 15 seconds (Algorithm 2). It is this matrix and its factor that are required by the LP DASA.

We did not use Matlab's sparse matrix data structure since Matlab removes explicit zeros. Changing the nonzero pattern by a single entry can cause Matlab to make a new copy of the entire matrix. This would defeat the asymptotic performance of our algorithms. Instead, the column-oriented data structure we use for $\mathcal{L}$, $\mathcal{L}^\sharp$, and $\mathbf{L}$ consists of three arrays of length $|\mathbf{L}_B|$, an array of length m that contains indices to the first entry in each column, and an array of length m holding the number of nonzeros in each column. The columns are allocated so that each column can hold as many nonzeros as the corresponding column of $\mathbf{L}_B$ without reallocation.

Starting with the matrix $\mathbf{A}_0$, we added one column at a time until all 12,230 columns from the original $\mathbf{B}$ were present, and then we removed them one at a time (in a first-in first-out order) to obtain the starting $\mathbf{A}_0$. No linear programming solver does so much work, but this provides a simple contrived test of our methods under a wide range of conditions that could occur in practice. The average time and work required to modify the factors at each step was 3.5 seconds and 2.6 million floating point operations. By comparison, solving a linear system $\mathbf{LDL}^\mathsf{T}\mathbf{x} = \mathbf{b}$ with a dense right-hand side $\mathbf{b}$ (using our column-oriented data structure for $\mathbf{L}$) at each step took an average of 6.9 seconds and 5.0 million floating point operations (each solve takes $O(|\mathcal{L}|)$ time). Thus, modifying the factors takes about half the time and work as using the factors to solve a linear system.

The time taken for the entire update/downdate computation would be much smaller if our code was written in a compiled language. Solving one system $\mathbf{LDL}^\mathsf{T}\mathbf{x} = \mathbf{b}$ with a dense right-hand side (using the factorization of the matrix $\mathbf{A}_0$) takes 5.5 seconds using our column-oriented data structure, 1.3 seconds using a Matlab sparse matrix for $\mathbf{L}$, and 0.22 seconds using Fortran 77. Hence, for the DFL001 test problem, we expect that our computation (both symbolic and numerical) would take about a tenth of a second per update or downdate in Fortran 77, on average.

7.2. Numerical accuracy. In order to measure the error in the computed Cholesky factorization, we evaluated the difference $\|\mathbf{AA}^\mathsf{T} - \hat{\mathbf{L}}\hat{\mathbf{D}}\hat{\mathbf{L}}^\mathsf{T}\|_1$, where $\hat{\mathbf{L}}\hat{\mathbf{D}}\hat{\mathbf{L}}^\mathsf{T}$ is the computed Cholesky factorization. For the airline scheduling matrix of section 7, $\hat{\mathbf{L}}$ has up to 1.49 million nonzeros and it is impractical to compute the product $\hat{\mathbf{L}}\hat{\mathbf{D}}\hat{\mathbf{L}}^\mathsf{T}$ after each update. To obtain a quick and accurate estimate for $\|\mathbf{E}\|_1$, where $\mathbf{E} = \mathbf{AA}^\mathsf{T} - \hat{\mathbf{L}}\hat{\mathbf{D}}\hat{\mathbf{L}}^\mathsf{T}$, we applied the strategy presented in [23] (see [24, p. 139] for a symbolic statement of the algorithm) to estimate the 1-norm of a matrix. That is, we used a gradient ascent approach to compute a local maximum for the following problem:

$$\max\{\|\mathbf{Ex}\|_1 : \|\mathbf{x}\|_1 = 1\}.$$

Since $\hat{\mathbf{L}}$ is used multiple times in the following algorithm, we copied our data structure for $\hat{\mathbf{L}}$ into a Matlab sparse matrix. In exact arithmetic, Algorithm 7 computes a lower bound on the 1-norm of $\mathbf{E}$.[3]

ALGORITHM 7 (estimate 1-norm of an m-by-m matrix $\mathbf{E}$).

```
    $x_i = 1/m$ for $1 \le i \le m$
    $\rho = 0$ ($\rho$ is the current estimate for $\|\mathbf{E}\|$)
    while $\|\mathbf{Ex}\|_1 > \rho$ do
        $\rho = \|\mathbf{Ex}\|_1$
        for $i = 1$ to $m$ do
            $y_i = \phantom{-}1$ if $(\mathbf{Ex})_i \ge 0$
            $y_i = -1$ if $(\mathbf{Ex})_i < 0$
        end for
        $\mathbf{z} = \mathbf{E}^\mathsf{T}\mathbf{y}$
        $j = \arg\max \{|z_i| : i = 1 \text{ to } m\}$
        if $|z_j| \le \mathbf{z}^\mathsf{T}\mathbf{x}$ return
        $x_i = 0$ for $i = 1$ to $n$
        $x_j = 1$
    end while
end Algorithm 7
```

[3] Algorithm 7 is available in Mathwork's contributed m-files ftp site, ftp.mathworks.com, as the file pub/contrib/v4/linalg/normest1.m.

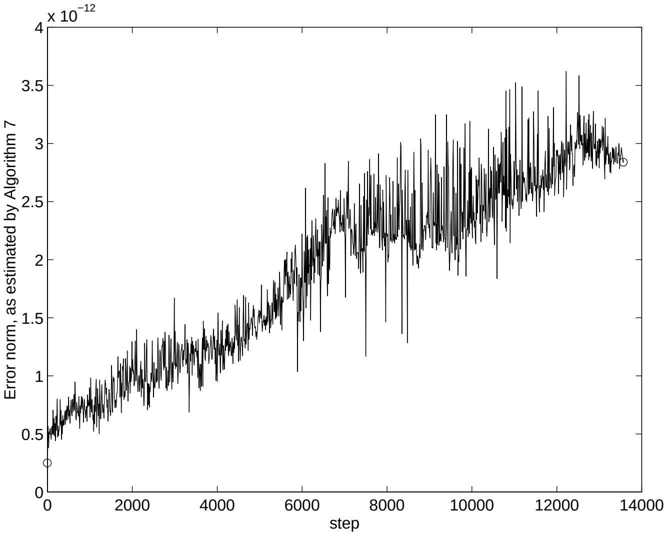


FIG. 7.1. *Estimated* 1*-norm of error in the* $\mathbf{LDL}^\mathsf{T}$ *factorization.*

To improve the accuracy of the 1-norm estimate, we used Algorithm 7 three times. In the second and third trials, a different starting vector $\mathbf{x}$ was used as described in [23]. Observe that Algorithm 7 only makes use of the product between the matrix $\mathbf{E}$ and a vector. This feature is important in the context of sparse matrices since $\mathbf{E}$ contains the term $\hat{\mathbf{L}}\hat{\mathbf{D}}\hat{\mathbf{L}}^\mathsf{T}$. It is impractical to compute the product $\hat{\mathbf{L}}\hat{\mathbf{D}}\hat{\mathbf{L}}^\mathsf{T}$, but it is practical to multiply $\hat{\mathbf{L}}\hat{\mathbf{D}}\hat{\mathbf{L}}^\mathsf{T}$ by a vector. For the airline scheduling matrix of section 7, the values for $\|\mathbf{E}\|_1$ initially, at step 6,784, and at the end were 2.5×10^{-13}, 2.4×10^{-12}, and 3.0×10^{-12}, respectively. The estimates obtained using Algorithm 7 were nearly identical at the same three steps (2.6×10^{-13}, 2.5×10^{-12}, and 2.9×10^{-12}, respectively). On the other hand, the times to compute $\|\mathbf{E}\|_1$ at the initial step and at step 6,784 were 119.4 and 266.4 seconds, while the times for three trials of Algorithm 7 were 8.3 and 13.5 seconds, respectively (excluding the time to construct the Matlab sparse matrix for $\hat{\mathbf{L}}$).

Our methods were quite accurate for this problem. After 6,784 updates and 6,784 downdates, or 13,568 changes in $\mathbf{A}$, the 1-norm of $\mathbf{E}$ increased by only a factor 12. Figure 7.1 shows the estimated value of $\|\mathbf{E}\|_1$ computed every 10 steps using Algorithm 7. The initial and final estimates are circled. The 1-norm of the matrix $\mathbf{AA}^\mathsf{T}$ increases from 458.0 initially to 1107.0 at iteration 6,784, then returns to 458.0 at iteration 13,568. Hence, the product of the computed Cholesky factors agrees with the product $\mathbf{AA}^\mathsf{T}$ to about 15 significant digits initially, while the products agree to about 14 significant digits after 13,568 modifications of $\mathbf{A}$.

7.3. Alternative permutations. Our methods are optimal in the sense that they take time proportional to the number of nonzero entries in $\mathbf{L}$ and $\mathbf{D}$ that change at each step. However, they are not optimal with respect to fill-in, since we assume a single initial permutation and no subsequent permutations. A fill-reducing ordering of $\mathbf{BB}^\mathsf{T}$ might not be the best ordering to use for all the $\mathbf{A}$ matrices. A simple pathological example is the m-by-n matrix $\mathbf{B}$, where $n = m(m-1)/2$ and the nonzero pattern of each column of $\mathbf{B}$ is a unique pair of integers from the set $\{1, 2, \ldots, m\}$.

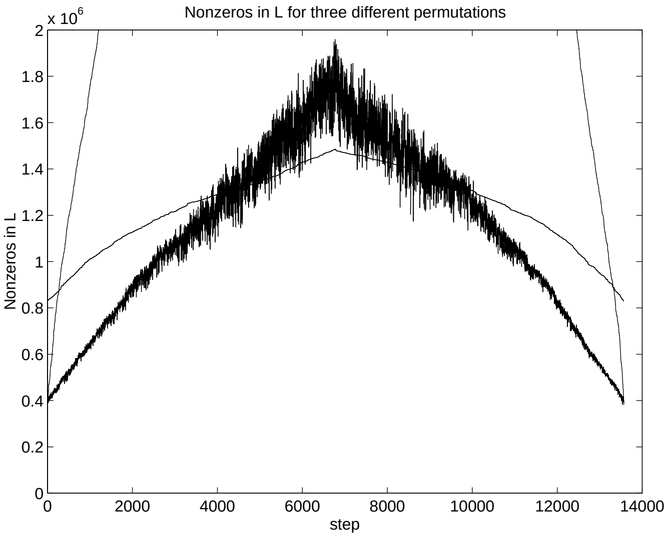


FIG. 7.2. *Nonzeros in* **L** *using three different permutations.*

In this case, every element of $\mathbf{BB}^\mathsf{T}$ is nonzero, while the nonzero pattern of $\mathbf{AA}^\mathsf{T}$ is arbitrary. As the matrix $\mathbf{A}$ changes, it might be advantageous to compute a fill-reducing ordering of $\mathbf{AA}^\mathsf{T}$ if the size of its factors grow "too large." A refactorization with the new permutation would then be required.

We found a fill-reducing permutation $\mathbf{P}_A$ of the starting matrix $\mathbf{A}_0\mathbf{A}_0^\mathsf{T}$ (again, the best of 101 trials of `colmmd`). This results in a factor $\mathbf{L}$ with 381 thousand nonzeros, requiring only 169 million floating point operations to compute. This is significantly less than the number of nonzeros (831 thousand) and floating point operations (481 million) associated with the fill-reducing permutation for $\mathbf{BB}^\mathsf{T}$. We also computed an ordering of $\mathbf{AA}^\mathsf{T}$ at each step, using `colmmd` just once, and then computed the number of nonzeros in the factor if we were to factorize $\mathbf{AA}^\mathsf{T}$ using this permutation ($\mathbf{P}_s$). Although it only takes about 1 second to compute the ordering [17] and symbolic factorization [18], it is not practical to use 100 random trials at each step.

Figure 7.2 depicts the nonzero counts of $\mathbf{L}$ for these three different permutations at each of the 13,568 steps. The fixed permutation $\mathbf{P}_B$ results in the smooth curve starting at 831 thousand and peaking at 1.49 million. The fixed permutation $\mathbf{P}_A$ results in a number of nonzeros in $\mathbf{L}$ that starts at 381 thousand and rises quickly, leaving the figure at step 1,206 and peaking at 7.4 million in the middle. It surpasses $\mathbf{P}_B$ at step 267. Using a permutation $\mathbf{P}_s$, computed at each step s, gives the erratic line in the figure, starting at 390 thousand and peaking at 1.9 million in the middle. These results indicate that it might be advantageous to start with the fixed permutation $\mathbf{P}_A$, use it for 267 steps, and then refactorize with the permutation $\mathbf{P}_s$ computed at step 267. This results in a new factor with only 463 thousand nonzeros. Near the center of the figure, however, $\mathbf{A}$ includes most of the columns in $\mathbf{B}$, and in this case the $\mathbf{P}_B$ permutation should be used.

8. Summary.

We have presented a new method for updating and downdating the factorization $\mathbf{LDL}^\mathsf{T}$ or $\mathbf{LL}^\mathsf{T}$ of a sparse symmetric positive definite matrix $\mathbf{AA}^\mathsf{T}$. Our experimental results show that the method should be fast and accurate in practice. Extensions to an arbitrary sparse symmetric positive definite matrix, $\mathbf{M}$, have been discussed. We mention additional extensions to our work that would be useful.

One drawback of our approach is the increase in storage. With the compressed pattern (a supernodal form of **L**) [13], the storage of **L** is dominated by the floating point values. In our storage scheme, we require two integers per floating point value (a row index and its multiplicity). Our methods almost double the storage for **L** (assuming 8-byte floating point values and 4-byte integers). A method based on a supernodal form could require less time and storage. However, supernodes would merge during update and split during downdate, which would be complicated to manage. Although implementing supernodes in the context of our current update and downdate methods is difficult, the numerical factorization can still be based on a supernodal method.

Some applications, such as local mesh refinement and coarsening and primal active set algorithms in optimization, require changes to the dimension of a symmetric positive definite matrix. These changes can be implemented using the techniques presented in this paper.

REFERENCES

[1] P. R. Amestoy, T. A. Davis, and I. S. Duff, *An approximate minimum degree ordering algorithm*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 886–905.

[2] R. H. Bartels, G. H. Golub, and M. A. Saunders, *Numerical techniques in mathematical programming*, in Nonlinear Programming, J. B. Rosen, O. L. Mangasarian, and K. Ritter, eds., Academic Press, New York, 1970, pp. 123–176.

[3] J. M. Bennett, *Triangular factors of modified matrices*, Numer. Math., 7 (1965), pp. 217–221.

[4] C. H. Bischof, C.-T. Pan, and P. T. P. Tang, *A Cholesky up- and downdating algorithm for systolic and SIMD architectures*, SIAM J. Sci. Comput., 14 (1993), pp. 670–676.

[5] N. A. Carlson, *Fast triangular factorization of the square root filter*, AIIA J., 11 (1973), pp. 1259–1265.

[6] S. M. Chan and V. Brandwajn, *Partial matrix refactorization*, IEEE Trans. on Power Systems, PWRS-1 (1986), pp. 193–200.

[7] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, The MIT Electrical Engineering and Computer Science Series, MIT Press, Cambridge, MA; McGraw–Hill, New York, 1990.

[8] J. J. Dongarra and E. Grosse, *Distribution of mathematical software via electronic mail*, Comm. ACM, 30 (1987), pp. 403–407.

[9] I. S. Duff, A. M. Erisman, and J. K. Reid, *Direct Methods for Sparse Matrices*, Oxford University Press, London, 1986.

[10] S. C. Eisenstat, M. C. Gursky, M. H. Schultz, and A. H. Sherman, *Yale sparse matrix package,* I: *The symmetric codes*, Internat. J. Numer. Methods Engrg., 18 (1982), pp. 1145–1151.

[11] 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. 139–162.

[12] A. George and J. W. H. Liu, *An optimal algorithm for symbolic factorization of symmetric matrices*, SIAM J. Comput., 9 (1980), pp. 583–593.

[13] A. George and J. W. H. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice–Hall, Englewood Cliffs, NJ, 1981.

[14] A. George, J. Liu, and E. Ng, *A data structure for sparse QR and LU factorizations*, SIAM J. Sci. Stat. Comput., 9 (1988), pp. 100–121.

[15] J. R. Gilbert, *Predicting Structure in Sparse Matrix Computations*, Tech. report CS-86-750, Computer Science Dept., Cornell Univ., Ithaca, NY, 1986.

[16] J. R. Gilbert, *Predicting structure in sparse matrix computations*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 62–79.

[17] J. R. Gilbert, C. Moler, and R. Schreiber, *Sparse matrices in MATLAB: Design and implementation*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 333–356.

[18] J. R. Gilbert, E. G. Ng, and B. W. Peyton, *An efficient algorithm to compute row and column counts for sparse Cholesky factorization*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 1075–1091.

[19] J. R. Gilbert and T. Peierls, *Sparse partial pivoting in time proportional to arithmetic operations*, SIAM J. Sci. Stat. Comput., 9 (1988), pp. 862–874.

[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] P. E. Gill and W. Murray, *Quasi-Newton methods for unconstrained optimization*, J. Inst. Math. Appl., 9 (1972), pp. 91–108.

[22] 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.

[23] W. W. Hager, *Condition estimates*, SIAM J. Sci. Stat. Comput., 5 (1984), pp. 311–316.

[24] W. W. Hager, *Applied Numerical Linear Algebra*, Prentice–Hall, Englewood Cliffs, NJ, 1988.

[25] W. W. Hager, *Updating the inverse of a matrix*, SIAM Rev., 31 (1989), pp. 221–239.

[26] W. W. Hager, *The LP dual active set algorithm,* in High Performance Algorithms and Software in Nonlinear Optimization, R. De Leone, A. Murli, P. M. Pardalos, and G. Toraldo, eds., Kluwer Academic Publishers, Norwell, MA, to appear.

[27] K. H. Law, *Sparse matrix factor modification in structural reanalysis*, Internat. J. Numer. Methods Engrg., 21 (1985), pp. 37–63.

[28] K. H. Law, *On updating the structure of sparse matrix factors*, Internat. J. Numer. Methods Engrg., 28 (1989), pp. 2339–2360.

[29] J. W. H. Liu, *The role of elimination trees in sparse factorization*, SIAM J. Matrix Anal. Appl., 11 (1990), pp. 134–172.

[30] C.-T. Pan, *A modification to the LINPACK downdating algorithm*, BIT, 30 (1990), pp. 707–722.

[31] D. J. Rose, R. E. Tarjan, and G. S. Lueker, *Algorithmic aspects of vertex elimination on graphs*, SIAM J. Comput., 5 (1976), pp. 266–283.

[32] D. G. Row, G. H. Powell, and D. P. Mondkar, *Solution of progressively changing equilibrium equations for nonlinear structures*, Comput. & Structures, 7 (1977), pp. 659–665.

[33] R. Schreiber, *A new implementation of sparse Gaussian elimination*, ACM Trans. Math. Software, 8 (1982), pp. 256–276.

[34] A. H. Sherman, *On the Efficient Solution of Sparse Systems of Linear and Nonlinear Equations*, Tech. report 46, Dept. of Computer Science, Yale Univ., New Haven, CT, 1975.

[35] G. Strang, *Linear Algebra and Its Applications*, Academic Press, New York, 1980.

[36] J. H. Wilkinson, *Linear algebra algorithms*, in Software for Numerical Mathematics, Proceedings of the Loughborough Univ. of Technology Conf. of the Inst. of Mathematics and its Appl., 1973, D. J. Evans, ed., Academic Press, New York, 1974, pp. 17–28.

SIAM J. MATRIX ANAL. APPL.
Vol. 20, No. 3, pp. 628–645

A NONSTANDARD CYCLIC REDUCTION METHOD, ITS VARIANTS AND STABILITY*

TUOMO ROSSI† AND JARI TOIVANEN†

Abstract. A nonstandard cyclic reduction method is introduced for solving the Poisson equation in rectangular domains. Different ways of solving the arising reduced systems are considered. The partial solution approach leads to the so-called partial solution variant of the cyclic reduction (PSCR) method, while the other variants are obtained by using the matrix rational polynomial factorization technique, including the partial fraction expansions, the fast Fourier transform (FFT) approach, and the combination of Fourier analysis and cyclic reduction (FACR) techniques. Such techniques have originally been considered in the standard cyclic reduction framework. The equivalence of the partial solution and the partial fraction techniques is shown. The computational cost of the considered variants is $\mathcal{O}(N \log N)$ operations, except for the FACR techniques for which it is $\mathcal{O}(N \log \log N)$. The stability estimate for the considered method is constructed, and the stability is demonstrated by numerical experiments.

Key words. separable matrix, fast Poisson solver, block cyclic reduction, partial solution problem

AMS subject classifications. 65F05, 65N22

PII. S0895479897317053

1. Introduction.

The fast direct methods for solving the Poisson problem already have a history of more than 30 years beginning in 1965 when Hockney [5] described the Fourier analysis method. In both [5] and [6], another method, called cyclic reduction, was discussed. The original cyclic reduction approach is unstable, but being combined with Fourier analysis in the FACR(l) algorithm [6], the unstability poses no serious restriction since in problems of practical size, typically, less than five steps of reduction process need to be performed. Stable, so-called Buneman's variants of cyclic reduction methods were introduced in [3], making the cyclic reduction a robust technique. The computational cost of Buneman's method is $\mathcal{O}(N \log N)$ floating point operations, and for the related FACR(l) algorithm with optimal choice of the number of reduction steps l it is $\mathcal{O}(N \log \log N)$ operations [13]. The approximate cyclic reduction method was introduced in [14]. Its computational cost is comparable to the optimal FACR algorithm when the solution is computed in the discretization accuracy, but it can be applied to a larger class of problems.

In the mid 1980s, another $\mathcal{O}(N \log N)$ method, utilizing the so-called partial solution technique [2], [9], was introduced in [17] and further developed in [8]. Because the method is based on the partial solution technique, it can easily be applied to the solution of elliptic problems which are discretized using nonuniform meshes. In [10],[1] the more efficient radix q variant ($q \geq 2$) of this method was considered and the method was generalized to the case of symmetric separable block band matrices. In

*Received by the editors February 18, 1997; accepted for publication (in revised form) by L. Reichel April 23, 1998; published electronically March 23, 1999. The work of the authors was supported by the Academy of Finland grant 34063.

http://www.siam.org/journals/simax/20-3/31705.html

†Laboratory of Scientific Computing, Department of Mathematics, University of Jyväskylä, PO Box 35, FIN-40351 Jyväskylä, Finland (Tuomo.Rossi@math.jyu.fi, Jari.Toivanen@math.jyu.fi).

[1] Here, a similar algorithm was called the divide and conquer method. Due to the ambiguity in this terminology, the authors prefer to use the name PSCR method instead.

radix q variants, the number of block equations is changed by the factor q between two consequent steps of the algorithm, whereas in the standard cyclic reduction method this factor is always two. The parallel implementation of this method, using a distributed memory parallel computer, is considered in [11]. In that paper, the method is called the partial solution variant of the cyclic reduction (PSCR) method due to its close relation to the cyclic reduction methodology.

The purpose of this paper is to develop a cyclic reduction-type formulation for the radix two variant of the PSCR method in order to show the close relationship between these two approaches. As a model problem, we consider the Dirichlet boundary value problem for the Poisson equation on the two-dimensional unit square. The model problem is discretized using the piecewise linear continuous finite elements on a triangulated uniform rectangular mesh with 2^k-1 interior mesh points in both directions. This leads to a linear system with a separable block tridiagonal matrix. We choose this problem because the relation between different methods can be clearly seen in this particular case. Usually, the PSCR method is formulated by means of hierarchical sequences of projection matrices [10], [11]. Using this formulation, the radix q variant of the PSCR method has been generalized to the cases of nonuniform meshes, arbitrary matrix dimensions, and block bandwidths. A combination of the PSCR method and FFT, leading to an $\mathcal{O}(N \log\log N)$ algorithm resembling the optimal FACR approach [13], was introduced in [7] for problems discretized using uniform meshes.

The cyclic reduction-type formulation leads to a sequence of problems which can be solved with a variety of different techniques. The main difference between our formulation and the classical one is the way in which the equations are eliminated. However, we are able to introduce variants of the classical rational polynomial factorization, partial fraction expansion, and FFT and FACR techniques to our nonstandard formulation. We also consider the partial solution approach which leads to the radix two variant of the PSCR method. We show that the partial fraction technique introduced in [16] essentially provides the partial solution version in the case of uniform meshes. We give only rough estimates of the computational costs of the different variants, while a more detailed estimate can be found in [11].

Finally, we present a detailed error analysis of the proposed method which, to our knowledge, appears to be a new result for the PSCR method. Our estimate shows that for the Poisson problem, the columnwise error cannot, asymptotically, grow faster than the same rate that the spectral radius of the inverse of the system matrix grows when the problem size increases. The validity of the stability estimate is verified by numerical experiments, which show that the variants given in this paper are essentially as stable as the well-known Buneman's second variant of the standard cyclic reduction method, which is implemented according to formulas given in [4].

2. A nonstandard cyclic reduction method. As a model problem, we consider the Dirichlet boundary value problem for the Poisson equation on the unit square. The piecewise linear finite element approximation on a uniform triangulated rectangular mesh with $n \times n$ interior nodes leads to the block tridiagonal linear system of the form

$$\begin{pmatrix} A & -T & & & \\ -T & A & -T & & \\ & -T & A & \ddots & \\ & & \ddots & \ddots & -T \\ & & & -T & A \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n-1} \\ u_n \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_{n-1} \\ f_n \end{pmatrix}, \tag{2.1}$$

where

$$A = \text{tridiag}\,\{-1 \;\; 4 \;\, -1\} \in \mathbb{R}^{n\times n}, \quad T = I \in \mathbb{R}^{n\times n}. \tag{2.2}$$

We assume that $n = 2^k - 1$ for some $k = 1, 2, \ldots$. We also assume that the matrices A and T commute, which is valid in our model problem. In the following formulas, we set $u_0 = u_{n+1} = 0$.

Let us consider three successive block rows in (2.1) for $i = 2, 4, \ldots, n-1$:

$$-Tu_{i-2} + Au_{i-1} - Tu_i = f_{i-1}, \tag{2.3a}$$

$$-Tu_{i-1} + Au_i - Tu_{i+1} = f_i, \tag{2.3b}$$

$$-Tu_i + Au_{i+1} - Tu_{i+2} = f_{i+1}. \tag{2.3c}$$

In the classical formulation of the cyclic reduction method, the block equations (2.3a) and (2.3c) are multiplied by the matrix T, and the second equation (2.3b) is multiplied by A. After that, the equations are summed up. This procedure eliminates the odd block rows from the system.

We propose an alternative way. We multiply the block equations (2.3a) and (2.3c) by TA^{-1} and add them to the second equation (2.3b). This leads to the reduced system of $2^{k-1} - 1$ block equations

$$\begin{aligned} -T^2A^{-1}u_{i-2} + (A - 2T^2A^{-1})u_i - T^2A^{-1}u_{i+2} &= f_i + TA^{-1}f_{i-1} + TA^{-1}f_{i+1}, \\ i &= 2, 4, \ldots, n-1. \end{aligned} \tag{2.4}$$

After solving the system of equations (2.4), we obtain the solution blocks u_i for $i = 2, 4, \ldots, n-1$. The unknown odd solution blocks can then be solved from the block diagonal system

$$Au_j = f_j + Tu_{j-1} + Tu_{j+1}, \quad j = 1, 3, \ldots, n-2, n. \tag{2.5}$$

The elimination (2.4) is repeated $k-1$ times until we have only one equation for the block $u_{(n+1)/2}$. To define this procedure recursively, we denote $A^{(0)} = A$, $T^{(0)} = T$, and $f^{(0)} = f$. Then, after r steps of elimination, $0 \le r \le k-1$, the reduced system has the block dimension $2^{k-r} - 1$, and it reads

$$\begin{pmatrix} A^{(r)} & -T^{(r)} & & & \\ -T^{(r)} & A^{(r)} & -T^{(r)} & & \\ & -T^{(r)} & A^{(r)} & \ddots & \\ & & \ddots & \ddots & -T^{(r)} \\ & & & -T^{(r)} & A^{(r)} \end{pmatrix} \begin{pmatrix} u_{2^r} \\ u_{2\cdot 2^r} \\ u_{3\cdot 2^r} \\ \vdots \\ u_{n-2^r+1} \end{pmatrix} = \begin{pmatrix} f^{(r)}_{2^r} \\ f^{(r)}_{2\cdot 2^r} \\ f^{(r)}_{3\cdot 2^r} \\ \vdots \\ f^{(r)}_{n-2^r+1} \end{pmatrix}, \tag{2.6}$$

where, for $r = 1, \dots, k-1$ and $i = 1, \dots, 2^{k-r}-1$,

$$A^{(r)} = A^{(r-1)} - 2\left(T^{(r-1)}\right)^2 \left(A^{(r-1)}\right)^{-1}, \tag{2.7}$$

$$T^{(r)} = \left(T^{(r-1)}\right)^2 \left(A^{(r-1)}\right)^{-1}, \tag{2.8}$$

$$f^{(r)}_{i\cdot 2^r} = f^{(r-1)}_{i\cdot 2^r} + T^{(r-1)} \left(A^{(r-1)}\right)^{-1} \left(f^{(r-1)}_{i\cdot 2^r - 2^{r-1}} + f^{(r-1)}_{i\cdot 2^r + 2^{r-1}}\right). \tag{2.9}$$

After $k-1$ steps of elimination, the resulting single block equation reads

$$A^{(k-1)} u_{2^{k-1}} = f^{(k-1)}_{2^{k-1}}, \tag{2.10}$$

from which we solve the middle block $u_{2^{k-1}}$. After we have solved the system (2.10), the rest of the unknown blocks are solved from the block diagonal systems of eliminated equations for $r = k-2, \dots, 0$. The system of the eliminated equations has the block dimension 2^{k-r-1}, and it is of the form

$$\begin{pmatrix} A^{(r)} & & & & \\ & A^{(r)} & & & \\ & & A^{(r)} & & \\ & & & \ddots & \\ & & & & A^{(r)} \end{pmatrix} \begin{pmatrix} u_{2^{r+1}-2^r} \\ u_{2\cdot 2^{r+1}-2^r} \\ u_{3\cdot 2^{r+1}-2^r} \\ \vdots \\ u_{n-2^r+1} \end{pmatrix} = \begin{pmatrix} g^{(r)}_{2^{r+1}-2^r} \\ g^{(r)}_{2\cdot 2^{r+1}-2^r} \\ g^{(r)}_{3\cdot 2^{r+1}-2^r} \\ \vdots \\ g^{(r)}_{n-2^r+1} \end{pmatrix}, \tag{2.11}$$

where

$$g^{(r)}_{j\cdot 2^{r+1}-2^r} = f^{(r)}_{j\cdot 2^{r+1}-2^r} + T^{(r)} \left(u_{(j-1)\cdot 2^{r+1}} + u_{j\cdot 2^{r+1}}\right), \quad j = 1, \dots, 2^{k-r-1}. \tag{2.12}$$

In the next sections, we consider how the right-hand sides $f^{(r)}$ and the solution vector u can be computed. First, we discuss the PSCR method involving the use of the partial solution technique. After that, we consider the rational polynomial factorization technique and its derivatives, the FFT approach, and the resulting FACR method.

3. The partial solution variant of the cyclic reduction method.

3.1. The reduction and backsubstitution stages. First, we shall formulate a lemma which will be needed in the subsequent two theorems.

LEMMA 3.1. *For the matrix $T^{(s)}$, $s = 1, \dots, k$, it holds that*

$$T^{(s)} = T^{(0)} \left(A^{(0)}\right)^{-1} T^{(0)} \left(A^{(1)}\right)^{-1} T^{(1)} \cdots \left(A^{(s-1)}\right)^{-1} T^{(s-1)}. \tag{3.1}$$

Proof. The result follows easily using induction and the definition of the matrices $T^{(r)}$ and from the fact that the matrices $A^{(r)}$ and $T^{(r)}$ commute. □

We denote in the following:

$$M^{(r)} = \begin{pmatrix} A & -T & & \\ -T & A & \ddots & \\ & \ddots & \ddots & -T \\ & & -T & A \end{pmatrix}, \tag{3.2}$$

where the block dimension of $M^{(r)}$ equals 2^r-1.

We shall first consider the computation of the vectors $f^{(r)}$ given by (2.9). The next result gives an equivalent way to compute them.

LEMMA 3.2. *Let $m = 2^r-1$. Then*

$$f^{(r)}_{i\cdot 2^r} = f^{(r-1)}_{i\cdot 2^r} + T\left(x^{(1)}_m + x^{(2)}_1\right), \tag{3.3}$$

where $x^{(1)}_m$ is the last block of length n of the vector $x^{(1)}$, and $x^{(2)}_1$ is the first block of length n of the vector $x^{(2)}$, where the vectors $x^{(k)}$, $k = 1, 2$, are given by the linear systems

$$M^{(r)}x^{(k)} = y^{(k)}. \tag{3.4}$$

The vectors $y^{(k)}$ have the block structures

$$y^{(k)}_j = \begin{cases} f^{(r-1)}_{i\cdot 2^r+(-1)^k\cdot 2^{r-1}}, & j = (m+1)/2, \\ 0, & \textit{otherwise.} \end{cases} \tag{3.5}$$

Proof. Let us apply the considered cyclic reduction method to solve a linear system of the form $M^{(r)}x = y$, where the only nonzero block of the vector y is $y_{(m+1)/2}$. Because of the sparse structure of the vector y, it follows that the solution block $x_{(m+1)/2}$ can be obtained from

$$A^{(r-1)}x_{(m+1)/2} = y_{(m+1)/2} \implies x_{(m+1)/2} = \left(A^{(r-1)}\right)^{-1} y_{(m+1)/2}.$$

We need the solution blocks x_1 and x_m. We shall consider the solution of the block x_1; the case of x_m is treated analogously. By using the backsubstitution formula (2.11) for the steps $j = r-2, \dots, 0$, and by following the first block equation, we notice that

$$Tx_1 = T^{(0)}x_1 = T^{(0)}\left(A^{(0)}\right)^{-1} T^{(0)}\left(A^{(1)}\right)^{-1} T^{(1)}\cdots\left(A^{(r-2)}\right)^{-1} T^{(r-2)}x_{(m+1)/2},$$

which, by using Lemma 3.1, gives

$$Tx_1 = T^{(r-1)}x_{(m+1)/2} = T^{(r-1)}\left(A^{(r-1)}\right)^{-1} y_{(m+1)/2}$$

from which the assertion follows. □

Hence, we have shown that in order to compute the vector $f^{(r)}$, it is sufficient to solve 2^{k-r} linear systems with the matrix $M^{(r)}$. These systems have a very special structure. The right-hand side vectors have only one nonzero block—the middle one, and the solution needs to be computed at most for two blocks of the solution vector, namely, the first and last ones. Thus, we have a partial solution problem for the linear system (3.4), and we show in the forthcoming section how to solve it efficiently by applying Fourier techniques.

The partial solution technique can be applied in the backsubstitution stage as well. At each level $k-1 \geq r \geq 0$, according to (2.11), we must solve 2^{k-r-1} systems

$$A^{(r)}u_{i\cdot 2^{r+1}-2^r} = f^{(r)}_{i\cdot 2^{r+1}-2^r} + T^{(r)}\left(u_{(i-1)\cdot 2^{r+1}} + u_{i\cdot 2^{r+1}}\right), \quad i = 1, \dots, 2^{k-r-1}.$$

Note that the above formulation also includes the single block equation which we solve when $r = k-1$. The following result gives us an equivalent way to compute the solution vector blocks.

Lemma 3.3. *Let $m = 2^{r+1}-1$. Then $u_{i\cdot 2^{r+1}-2^r} = x_{(m+1)/2}$, where $x_{(m+1)/2}$ is the middle block of length n of the solution vector x of the system $M^{(r+1)}x = y$. Here the vector y is defined as*

$$y_j = \begin{cases} Tu_{(i-1)\cdot 2^{r+1}}, & j = 1, \\ f^{(r)}_{i\cdot 2^{r+1}-2^r}, & j = (m+1)/2, \\ Tu_{i\cdot 2^{r+1}}, & j = m, \\ 0, & \textit{otherwise}, \end{cases} \tag{3.6}$$

except when $r = 0$, the only block y_1 is given by $y_1 = f^{(0)}_{2\cdot i-1} + T(u_{2\cdot(i-1)} + u_{2\cdot i})$.

Proof. Due to the linearity, we may write

$$x_{(m+1)/2} = x^{(1)}_{(m+1)/2} + x^{(2)}_{(m+1)/2} + x^{(3)}_{(m+1)/2},$$

where $x^{(k)}_{(m+1)/2}$ is the middle block of the solution $x^{(k)}$ of the system $M^{(r+1)}x^{(k)} = y^{(k)}$. The vectors $y^{(k)}$ are defined by

$$y^{(1)}_j = \begin{cases} f^{(r)}_{i\cdot 2^{r+1}-2^r}, & j = (m+1)/2, \\ 0, & textotherwise, \end{cases} \qquad y^{(2)}_j = \begin{cases} Tu_{(i-1)\cdot 2^{r+1}}, & j = 1, \\ 0, & \text{otherwise}, \end{cases}$$

$$y^{(3)}_j = \begin{cases} Tu_{i\cdot 2^{r+1}}, & j = m, \\ 0, & \text{otherwise}. \end{cases}$$

We already know, from the proof of Lemma 3.2, that

$$x^{(1)}_{(m+1)/2} = \left(A^{(r)}\right)^{-1} f^{(r)}_{i\cdot 2^{r+1}-2^r}.$$

Let us consider the solution of $M^{(r+1)}x^{(2)} = y^{(2)}$. The solution of $x^{(3)}$ can be treated similarly. We shall again apply the proposed cyclic reduction method in solving the middle block $x^{(2)}_{(m+1)/2}$.

Let us denote the arising right-hand side vectors by $g^{(j)}$ (starting with $g^{(0)} = y^{(2)}$). Due to the special structure of the vector $g^{(0)}$, we realize by following the update chain of the right-hand side vectors $g^{(j)}$ and by using Lemma 3.1 that

$$g^{(r)}_{(m+1)/2} = T^{(r-1)}\left(A^{(r-1)}\right)^{-1} \cdots T^{(0)}\left(A^{(0)}\right)^{-1} Tu_{(i-1)\cdot 2^{r+1}} = T^{(r)}u_{(i-1)\cdot 2^{r+1}}.$$

Hence, again, from the proof of Lemma 3.2, we obtain that

$$x^{(2)}_{(m+1)/2} = \left(A^{(r)}\right)^{-1} T^{(r)}u_{(i-1)\cdot 2^{r+1}}.$$

In a similar way, we can show that

$$x^{(3)}_{(m+1)/2} = \left(A^{(r)}\right)^{-1} T^{(r)}u_{i\cdot 2^{r+1}},$$

which completes the proof. □

Hence, each backsubstitution stage $k-1 \geq r \geq 0$ can be realized by solving 2^{k-r-1} partial solution problems with the matrix $M^{(r+1)}$.

3.2. Partial solution technique. Let us describe how we can apply the so-called partial solution technique [2], [9], in solving the arising linear systems with the matrices $M^{(r)}$ and $M^{(r+1)}$. We denote by I_s the identity $s \times s$ matrix and

$$K_s = \begin{pmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{pmatrix}_{s \times s} . \tag{3.7}$$

In our model case, it now follows that the matrix $M^{(r)}$ is separable, that is, it has the representation

$$M^{(r)} = I_{2^r-1} \otimes K_n + K_{2^r-1} \otimes I_n, \tag{3.8}$$

where $\otimes$ denotes the tensor product of matrices which in our case is defined as follows. For $A \in \mathbb{R}^{k \times k}$ and $B \in \mathbb{R}^{s \times s}$,

$$A \otimes B = \{A_{i,j} B\}_{i,j=1}^{k} \in \mathbb{R}^{ks \times ks}. \tag{3.9}$$

Let us consider the eigenvalue problem

$$K_{2^r-1} w_i = \lambda_i w_i, \quad i = 1, \ldots, 2^r - 1. \tag{3.10}$$

In this case, we know the explicit eigenvalues and orthonormal eigenvectors. They are given by

$$\lambda_i = 4 \left(\sin \left(\frac{i\pi}{2^{r+1}} \right) \right)^2, \quad (w_i)_j = \sqrt{2^{1-r}} \sin \left(\frac{ij\pi}{2^r} \right), \quad i, j = 1, \ldots, 2^r - 1. \tag{3.11}$$

We denote in the following:

$$W = [w_1 \; w_2 \ldots w_{2^r-1}] = W^T \in \mathbb{R}^{(2^r-1) \times (2^r-1)}. \tag{3.12}$$

In the reduction stage, we need to solve the linear systems

$$M^{(r)} x = y, \tag{3.13}$$

where the vector y has only one nonzero block, the middle one $y_{2^{r-1}}$, and we only need the blocks x_1 and x_{2^r-1} of the solution vector x. By multiplying (3.13) from the left by the matrix $W^T \otimes I_n$, and by denoting $\tilde{x} = (W^T \otimes I_n) x$, we obtain a block diagonal linear system

$$\begin{pmatrix} K_n + \lambda_1 I_n & & & \\ & K_n + \lambda_2 I_n & & \\ & & \ddots & \\ & & & K_n + \lambda_{2^r-1} I_n \end{pmatrix} \begin{pmatrix} \tilde{x}_1 \\ \tilde{x}_2 \\ \vdots \\ \tilde{x}_{2^r-1} \end{pmatrix} = \begin{pmatrix} \tilde{y}_1 \\ \tilde{y}_2 \\ \vdots \\ \tilde{y}_{2^r-1} \end{pmatrix}, \tag{3.14}$$

where $\tilde{y}_j = (w_j)_{2^{r-1}} y_{2^{r-1}}$, $j = 1, \ldots, 2^r - 1$. It is clear that $x = (W \otimes I_n) \tilde{x}$ and, thus, after solving the block diagonal system (3.14), the required solution blocks are obtained by

$$x_1 = \sum_{j=1}^{2^r-1} (w_j)_1 \tilde{x}_j, \quad x_{2^r-1} = \sum_{j=1}^{2^r-1} (w_j)_{2_r-1} \tilde{x}_j. \tag{3.15}$$

This special implementation of the classical method of separation of variables is called the partial solution method. Its computational cost is $\mathcal{O}(n(2^r - 1))$. The cost of computing the right-hand side vectors for the block diagonal system (3.14) is $n(2^r - 1)$ floating point operations since these vectors are simply obtained by scaling with a scalar. The cost of solving the arising $2^r - 1$ tridiagonal systems is $\mathcal{O}(n(2^r - 1))$. Finally, the cost computing both x_1 and x_{2^r-1} by (3.15) is $4n(2^r - 1)$, which leads to the given estimate.

The partial solution problems involving the matrix $M^{(r+1)}$ are solved in a similar manner. The required eigenvalues and eigenvectors can be obtained from (3.11) by replacing r by $r + 1$. In this case the right-hand side vectors have only three nonzero blocks, the first, the middle, and the last one. The sparse Fourier transforms are carried out using the eigenvectors w_j leading to $2^{r+1} - 1$ tridiagonal systems

$$(K_n + \lambda_j I_n)\tilde{x}_j = \tilde{y}_j, \quad j = 1, \ldots, 2^{r+1} - 1, \tag{3.16}$$

where

$$\tilde{y}_j = (w_j)_1 y_1 + (w_j)_{2^r} y_{2^r} + (w_j)_{2^{r+1}-1} y_{2^{r+1}-1}. \tag{3.17}$$

The required middle solution block x_{2^r} is then computed by

$$x_{2^r} = \sum_{j=1}^{2^{r+1}-1} (w_j)_{2^r} \tilde{x}_j. \tag{3.18}$$

The cost of this procedure is $\mathcal{O}(n(2^{r+1} - 1))$.

From these estimates it follows that the cost of each reduction and backsubstitution stage is $\mathcal{O}(n^2)$ floating point operations leading to the total cost of $\mathcal{O}(n^2 \log_2 n)$ operations.

It should be noted that the partial solution technique is not restricted to the case of uniform meshes. Furthermore, the dimensions of the partial solution problems can be arbitrary. These properties follow from the fact that we do not apply FFT in computing the required sparse Fourier transforms. This is the reason why the PSCR method can be quite easily generalized to treat more complicated situations.

4. Other variants.

4.1. Rational polynomial factorization technique. The reduction and backsubstitution stages (2.9) and (2.11) can also be computed in a traditional cyclic reduction manner using a rational polynomial factorization of the matrices $A^{(r)}$ and $T^{(r)}$. Let us define three matrix functions recursively as follows: Set $\alpha^{(0)}(A, T) = A$, $\tau^{(0)}(T) = T$, and $\beta^{(0)}(A, T) = I$. For $r = 1, \ldots, k - 1$, set

$$\begin{aligned} \alpha^{(r)}(A,T) &= \left(\alpha^{(r-1)}(A,T)\right)^2 - 2\left(\tau^{(r-1)}(T)\right)^2, \\ \tau^{(r)}(T) &= \left(\tau^{(r-1)}(T)\right)^2 = T^{2^r}, \\ \beta^{(r)}(A,T) &= \beta^{(r-1)}(A,T)\left(\alpha^{(r-1)}(A,T)\right)^{-1}. \end{aligned} \tag{4.1}$$

Then, it is straightforward to show that

$$A^{(r)} = \beta^{(r)}(A,T)\alpha^{(r)}(A,T) \quad \text{and} \quad T^{(r)} = \beta^{(r)}(A,T)\tau^{(r)}(T). \tag{4.2}$$

In [4], it has been shown that the polynomial $\alpha^{(r)}$ has the expansion

$$\alpha^{(r)}(A,T) = \prod_{j=1}^{2^r}(A - \theta(j,r)T), \quad \theta(j,r) = 2\cos\left(\frac{(2j-1)\pi}{2^{r+1}}\right), \tag{4.3}$$

and from this, it easily follows that

$$\beta^{(r)}(A,T) = \prod_{j=1}^{2^r-1}(A - \phi(j,r)T)^{-1}, \quad \phi(j,r) = 2\cos\left(\frac{j\pi}{2^r}\right). \tag{4.4}$$

In the reduction stage, we compute vectors of the form $x = T^{(r-1)}\left(A^{(r-1)}\right)^{-1} y$. By using the above formulas we obtain

$$\begin{aligned} x &= T^{(r-1)}\left(A^{(r-1)}\right)^{-1} y = \tau^{(r-1)}(T)\left(\alpha^{(r-1)}(A,T)\right)^{-1} y \\ &= \prod_{j=1}^{2^{r-1}}\left[(A - \theta(j,r-1)T)^{-1}T\right] y. \end{aligned} \tag{4.5}$$

This multiplication can be carried out in a standard cyclic reduction way by solving a sequence of tridiagonal linear systems. Furthermore, the reduction stage can be implemented in such a way that the LU decomposition of each matrix $A-\theta(j,r-1)T$ is computed only once at each reduction level r. This LU decomposition can then be used in solving all the arising 2^{k-r}–1 problems at level r.

In the backsubstitution stage, we have to compute the following type of vectors: $x = \left(A^{(r)}\right)^{-1}\left(y + T^{(r)}z\right)$. The vector x can be computed in a stable way as follows:

$$\begin{aligned} x &= \left(\alpha^{(r)}(A,T)\right)^{-1}\left(\beta^{(r)}\right)^{-1} y + \left(\alpha^{(r)}(A,T)\right)^{-1}\tau^{(r)}(T)z \\ &= \prod_{j=2}^{2^r}\left[(A - \theta(j,r)T)^{-1}(A - \phi(j-1,r)T)\right](A - \theta(1,r)T)^{-1}y \\ &+ \prod_{j=1}^{2^r}\left[(A - \theta(j,r)T)^{-1}T\right] z. \end{aligned} \tag{4.6}$$

The cost of the above algorithm can again be slightly reduced by computing the LU decomposition of each matrix $A - \theta(j,r)T$ only once. Then, all the systems involving this matrix in the 2^{k-r-1} problems of level r are solved using the LU decomposition.

The number of arithmetic operations which are needed in solving a tridiagonal $n \times n$ linear system is $\mathcal{O}(n)$. A straightforward analysis of the proposed variant shows that the number of tridiagonal systems which have to be solved is $\mathcal{O}(n(\log_2 n - 1))$. Hence, the cost of this variant is proportional to $\mathcal{O}(n^2 \log_2 n)$.

4.2. Rational polynomial approach using partial fractions. In this section, we describe how the partial fraction technique [16] is applied in computing (4.5) and (4.6). The basic idea is given by the next lemma [16].

LEMMA 4.1. *Let $p(t)$ and $q(t)$ be two polynomials satisfying the following:*

1. *p and q are relatively prime,*
2. *$\deg p < \deg q = n$,*

3. *the roots, $\alpha_1, \alpha_2, \ldots, \alpha_n$, of q are distinct.*
Then,

$$\frac{p(t)}{q(t)} = \sum_{j=1}^{n} \frac{c_j}{t-\alpha_j}, \quad \textit{where} \quad c_j = \frac{p(\alpha_j)}{q'(\alpha_j)}.$$

Let us consider the specific case $T = I$. Then, it follows that

$$\alpha^{(r)}(A, I) = C_{2^r}(A), \quad \beta^{(r)}(A, I) = \left(S_{2^r-1}(A)\right)^{-1}, \quad \tau^{(r)}(I) = I, \tag{4.7}$$

where $C_{2^r}(t)$ and $S_{2^r-1}(t)$ are the modified Chebyshev polynomials (see, for example, [1]) defined by

$$C_{2^r}(t) = \begin{cases} 2\cos\left(2^r \arccos(t/2)\right), & 0 \le t < 2, \\ 2\cosh\left(2^r \operatorname{arccosh}(t/2)\right), & t \ge 2, \end{cases} \tag{4.8}$$

$$S_{2^r-1}(t) = \begin{cases} \sin\left(2^r \arccos(t/2)\right) / \sin\left(\arccos(t/2)\right), & 0 \le t < 2, \\ \sinh\left(2^r \operatorname{arccosh}(t/2)\right) / \sinh\left(\operatorname{arccosh}(x/2)\right), & t \ge 2. \end{cases} \tag{4.9}$$

In the reduction stage (4.5), we need to perform matrix–vector multiplications of the form

$$x = p(A)q(A)^{-1}y, \quad \text{with} \quad p(A) = I, \quad q(A) = C_{2^{r-1}}(A). \tag{4.10}$$

Since

$$q'(t) = C'_{2^{r-1}}(t) = 2^{r-1} S_{2^{r-1}-1}(t), \tag{4.11}$$

we obtain, by substituting t with the roots $\alpha_j = \theta(j, r-1),\ j = 1, \ldots, 2^{r-1}$ given by (4.3), that the vector x in (4.5) can be computed in an additive manner by

$$x = 2^{1-r} \sum_{j=1}^{2^{r-1}} (-1)^{j-1} \sin\left(\frac{(2j-1)\pi}{2^r}\right) (A - \theta(j, r-1)I)^{-1} y. \tag{4.12}$$

A similar technique can be employed also in the backsubstitution stage (4.6). Obviously, the computation of the part $\left(A^{(r)}\right)^{-1} T^{(r)} z$ can be computed analogously to (4.12). Let us therefore consider the matrix–vector multiplication $\left(A^{(r)}\right)^{-1} y$. Hence, it suffices to study the expressions

$$p(A)q(A)^{-1}y, \quad \text{with} \quad p(A) = S_{2^r-1}(A), \quad q(A) = C_{2^r}(A). \tag{4.13}$$

It immediately follows from (4.11) that for all $j = 1, \ldots, 2^r$, we have $c_j = 2^{-r}$. This means that the vector x in (4.6) can be computed in an equivalent way by

$$x = 2^{-r} \sum_{j=1}^{2^r} (A - \theta(j, r)I)^{-1} \left(y + (-1)^{j-1} \sin\left(\frac{(2j-1)\pi}{2^{r+1}}\right) z \right). \tag{4.14}$$

THEOREM 4.2. *In the case $T = I$, the partial fraction technique is equivalent to the partial solution approach in the sense that it gives all the tridiagonal systems* (3.14), (3.16) *of the arising partial solution problems which have a nonzero solution.*

Table 4.1

The number of arising tridiagonal systems in the partial solution, rational polynomial factorization, and partial fraction variants, and in the Buneman's second variant of the standard cyclic reduction method.

Variant	No. of tridiagonal systems
part.sol.	$(2n+3)\log_2(n+1)-4n$
rat.fact.	$\frac{3}{2}(n+1)\log_2(n+1)-n$
part.frac.	$(n+1)\log_2(n+1)-n$
Buneman	$(n+1)\log_2(n+1)-n$

Proof. In the reduction stage of the partial solution approach, we compute the vector blocks x_1 and x_{2^r-1} of a given partial solution problem with the only nonzero right-hand side block being $y_{2^{r-1}}$. We shall only consider the computation of x_1. The case of x_{2^r-1} can be treated analogously. We have

$$\begin{aligned} x_1 &= \sum_{k=1}^{2^r-1} (w_k)_1 \tilde{x}_k = \sum_{k=1}^{2^r-1} (w_k)_1 (K_n + \lambda_k I_n)^{-1} (w_k)_{2^{r-1}} y_{2^{r-1}} \\ &= 2^{1-r} \sum_{k=1}^{2^r-1} \sin\left(\frac{k\pi}{2^r}\right) \sin\left(\frac{k\pi}{2}\right) (K_n + \lambda_k I_n)^{-1} y_{2^{r-1}} \\ &= 2^{1-r} \sum_{j=1}^{2^{r-1}} (-1)^{j-1} \sin\left(\frac{(2j-1)\pi}{2^r}\right) (K_n + \lambda_{2j-1} I_n)^{-1} y_{2^{r-1}}. \end{aligned}$$

Since

$$2\cos\left(\frac{(2j-1)\pi}{2^r}\right) = 2 - 4\left(\sin\left(\frac{(2j-1)\pi}{2^{r+1}}\right)\right)^2, \quad j = 1, \dots, 2^{r-1},$$

we obtain that

$$x_1 = 2^{1-r} \sum_{j=1}^{2^{r-1}} (-1)^{j-1} \sin\left(\frac{(2j-1)\pi}{2^r}\right) (A - \theta(j, r-1)I)^{-1} y_{2^{r-1}}.$$

It is easy to see in a similar manner that the backsubstitution stages are also equivalent. The only difference in the above analysis is the treatment of the nonzero middle block of the right-hand side vector. Since the middle components of the corresponding eigenvectors w_j are of the form $\sqrt{2^{-r}}\sin(j\pi/2)$, it follows that the contribution of the Fourier transforms is only a scaling factor 2^{-r}, which indeed appears in the term corresponding to the middle block in the formula (4.14). □

We have collected the numbers of the arising tridiagonal solves in Table 4.1 for the variants considered so far. For comparison, we also include the Buneman's second variant of the standard cyclic reduction method [3], [4], [13]. As can be seen, the computational cost of the radix two partial solution variant is about two times larger than that of the Buneman's method. However, a performance comparable to the Buneman's method can be achieved by using the partial fraction variant. The partial solution variant has a wider set of applications. For example, it is straightforward to apply it in the solution of problems discretized on nonuniform meshes. Its performance can be improved by increasing the radix of the algorithm which can be conveniently done by an alternative formulation using orthogonal projection matrices [10], [11]. In [11], it was observed that this approach leads to an algorithm which appears to be more efficient than the method considered in [12], [15].

4.3. FFT approach. An alternative way to solve the arising linear systems (2.9) and (2.11) is to apply the FFT. Because the matrices $A^{(r)}$ and $T^{(r)}$ commute, they have the same set of eigenvectors. Let us denote by $\Lambda^{(0)}$ and $\Gamma^{(0)}$ the diagonal matrices which contain the eigenvalues of A and T, respectively. Hence,

$$A = Q\Lambda^{(0)}Q^T, \quad T = Q\Gamma^{(0)}Q^T, \tag{4.15}$$

where Q is the orthogonal matrix whose column vectors are the eigenvectors of the matrices A and T. From the update formulas (2.7) and (2.8), we obtain

$$\begin{aligned} &A^{(r)} = Q\Lambda^{(r)}Q^T \iff \left(A^{(r)}\right)^{-1} = Q\left(\Lambda^{(r)}\right)^{-1}Q^T, \\ &T^{(r)} = Q\Gamma^{(r)}Q^T, \end{aligned} \tag{4.16}$$

where

$$\begin{aligned} \Lambda^{(r)} &= \Lambda^{(r-1)} - 2\left(\Lambda^{(r-1)}\right)^{-1}\left(\Gamma^{(r-1)}\right)^2, \\ \Gamma^{(r)} &= \left(\Lambda^{(r-1)}\right)^{-1}\left(\Gamma^{(r-1)}\right)^2. \end{aligned} \tag{4.17}$$

In the case of our model problem, we have the explicit formulas for the eigenvalues and eigenvectors of the matrices A and T. They are given by

$$\begin{aligned} \Lambda^{(0)}_{i,i} &= 2 + 4\left(\sin\left(\frac{i\pi}{2(n+1)}\right)\right)^2, \quad \Gamma^{(0)}_{i,i} = 1, \quad i = 1, \dots, n, \\ Q_{i,j} &= \sqrt{\frac{2}{n+1}}\sin\left(\frac{ij\pi}{n+1}\right), \quad i, j = 1, \dots, n. \end{aligned} \tag{4.18}$$

Due to the special structure of the eigenvectors, we can apply the fast discrete sine transform in the multiplication by the matrix Q. The use of the expansions (4.16) leads to the following algorithms for the reduction and backsubstitution stages. The computation of $x = T^{(r-1)}\left(A^{(r-1)}\right)^{-1}y$ in the reduction stage is done by

$$x = Q\Gamma^{(r-1)}\left(\Lambda^{(r-1)}\right)^{-1}Q^T y. \tag{4.19}$$

The operations of the form $x = \left(A^{(r)}\right)^{-1}\left(y + T^{(r)}z\right)$ in the backsubstitution stages can be computed by

$$x = Q\left(\Lambda^{(r)}\right)^{-1}\left(Q^T y + \Gamma^{(r)}Q^T z\right). \tag{4.20}$$

In both cases, the multiplications by the matrices Q and Q^T can be done using the FFT. It should be noted that the number of matrix–vector multiplications could be reduced by storing the Fourier coefficients from the previous levels. The asymptotic cost of this approach can be estimated by counting the number of matrix–vector multiplications by the matrices Q or Q^T. Since we apply FFT, the cost of one multiplication by either Q or Q^T is $\mathcal{O}(n \log_2 n)$. The total number of the considered multiplications is $5n$, and hence, the cost of this variant is $\mathcal{O}(n^2 \log_2 n)$.

4.4. FACR(l) approach. From the previous estimates we noticed that all steps of the polynomial factorization variant are approximately as expensive. However, in the FFT-based approach, the steps become cheaper as r increases. This suggests that we combine these methods in such a way that for small values of r we use the polynomial factorization and solve the intermediate part using an FFT-based approach. This is exactly what is done in the FACR(l)-type algorithms (see, for example, [6], [13]).

The resulting FACR(l) algorithm can be easily deduced. After $0 \leq l \leq k-1$ steps of the reduction stage, we have the system

$$\begin{pmatrix} A^{(l)} & -T^{(l)} & & \\ -T^{(l)} & A^{(l)} & \ddots & \\ & \ddots & \ddots & -T^{(l)} \\ & & -T^{(l)} & A^{(l)} \end{pmatrix} \begin{pmatrix} u_{2^l} \\ u_{2\cdot 2^l} \\ \vdots \\ u_{n-2^l+1} \end{pmatrix} = \begin{pmatrix} f_{2^l}^{(l)} \\ f_{2\cdot 2^l}^{(l)} \\ \vdots \\ f_{n-2^l+1}^{(l)} \end{pmatrix}, \tag{4.21}$$

which is transformed to n tridiagonal systems by applying the FFT to the blocks $f_j^{(l)}$ and by diagonalizing the matrices $A^{(l)}$ and $T^{(l)}$. These tridiagonal systems are of dimension $(2^{k-l}-1) \times (2^{k-l}-1)$ and they have the structure

$$\begin{pmatrix} a_j & -t_j & & \\ -t_j & a_j & \ddots & \\ & \ddots & \ddots & -t_j \\ & & -t_j & a_j \end{pmatrix} \begin{pmatrix} (x_{2^l})_j \\ (x_{2\cdot 2^l})_j \\ \vdots \\ (x_{n-2^l+1})_j \end{pmatrix} = \begin{pmatrix} (Q^T f_{2^l}^{(l)})_j \\ (Q^T f_{2\cdot 2^l}^{(l)})_j \\ \vdots \\ (Q^T f_{n-2^l+1}^{(l)})_j \end{pmatrix}. \tag{4.22}$$

Here $a_j = \Lambda_{j,j}^{(l)}$, $t_j = \Gamma_{j,j}^{(l)}$ and $(Q^T f_{i\cdot 2^l}^{(l)})_j$, $j = 1, \ldots, n$, denotes the jth component of the vector $Q^T f_{i\cdot 2^l}^{(l)}$, which is computed using the FFT. After all the n tridiagonal systems (4.22) are solved, the solution blocks are obtained with the FFT as $u_{i\cdot 2^l} = Qx_{i\cdot 2^l}$. The remaining solution blocks are then recovered performing the backsubstitution stages $l-1, \ldots, 0$.

Let us next study the optimal choice of l. We shall consider the case when the reduction and backsubstitution stages are computed with the partial solution variant. The rational polynomial factorization and partial fraction variants could also be used here and the computational cost would be of the same order. The cost of performing l steps of reduction and backsubstitution is of order $lc\, n^2$ operations, while the solution of the system (4.22) requires $2^{-l}(c n^2 + 2d n^2 \log_2 n)$ operations, where c and d are independent of n. Hence, the total cost is of order

$$lc\, n^2 + 2^{-l}(c n^2 + 2d\, n^2 \log_2 n), \tag{4.23}$$

which attains its minimum when

$$l = \log_2 \left(\frac{(c + 2d \log_2 n) \log 2}{c} \right) = \mathcal{O}(\log_2 \log_2 n). \tag{4.24}$$

By substituting this in the formula (4.23), we obtain that the cost is $\mathcal{O}(n^2 \log_2 \log_2 n)$.

5. Numerical stability.

We shall prove that the proposed variants of the nonstandard cyclic reduction method are linearly stable with respect to the size of the problem n^2 in the case when $T = I$, $A = A^T$, and the smallest eigenvalue of A is at least two. We shall use similar techniques as in [4] in our error analysis. The stability of the proposed methods is also demonstrated in numerical experiments.

5.1. The stability estimate.

Let $\|\cdot\|$ denote the spectral norm of a matrix and let $\lambda(A)$ denote the set of eigenvalues of the matrix A. Since we assumed that $\lambda \geq 2$ for all $\lambda \in \lambda(A)$, the polynomials are of the latter form in (4.8) and (4.9). To simplify the notation, we define

$$\theta = \operatorname{arccosh}(\lambda/2) = \log a, \quad a = \lambda/2 + \sqrt{(\lambda/2)^2 - 1}. \tag{5.1}$$

Now, according to (4.7),

$$\begin{aligned}\left\|\left(A^{(r)}\right)^{-1}\right\| &= \left\|\left(\beta^{(r)}(A,I)\alpha^{(r)}(A,I)\right)^{-1}\right\| = \left\|S_{2^r-1}(A)C_{2^r}(A)^{-1}\right\| \\ &\leq \max_{\lambda\in\lambda(A)}\left|\frac{S_{2^r-1}(\lambda)}{C_{2^r}(\lambda)}\right| = \max_{\lambda\in\lambda(A)}\left|\frac{\sinh(2^r\theta)}{2\sinh(\theta)\cosh(2^r\theta)}\right| \\ &= \max_{\lambda\in\lambda(A)}\left|\frac{e^\theta}{e^{2\theta}-1}\cdot\frac{e^{2^{r+1}\theta}-1}{e^{2^{r+1}\theta}+1}\right| = \max_{\lambda\in\lambda(A)}\left|\frac{a}{a^2-1}\cdot\frac{a^{2^{r+1}}-1}{a^{2^{r+1}}+1}\right| \\ &= \max_{\lambda\in\lambda(A)}\left|\frac{a\sum_{i=1}^{2^r}a^{2^{r+1}-2i}}{a^{2^{r+1}}+1}\right| = \max_{\lambda\in\lambda(A)}|q_1(a)|.\end{aligned} \tag{5.2}$$

In our case, $\lambda \in (2, 6)$, from which it follows that $a \in (1, 3 + 2\sqrt{2})$. The function $q_1(a) > 0$ decays when a increases. Hence,

$$\left\|\left(A^{(r)}\right)^{-1}\right\| \leq \lim_{a\to 1} q_1(a) = 2^{r-1}. \tag{5.3}$$

A similar analysis shows that

$$\left\|T^{(r)}\left(A^{(r)}\right)^{-1}\right\| \leq \max_{\lambda\in\lambda(A)}\left|C_{2^r}(\lambda)^{-1}\right| = \max_{\lambda\in\lambda(A)}\left|\frac{1}{a^{2^r}+a^{2^{-r}}}\right| = \max_{\lambda\in\lambda(A)}|q_2(a)|. \tag{5.4}$$

Again, we obtain that $q_2(a) > 0$, and

$$\left\|T^{(r)}\left(A^{(r)}\right)^{-1}\right\| \leq \lim_{a\to 1} q_2(a) = \frac{1}{2}. \tag{5.5}$$

Let us now consider the error accumulation in the considered cyclic reduction algorithm. Let $f_i^{(0)} = f_i$ denote the exact right-hand side vector blocks and let $f_{i,\varepsilon}^{(0)}$ be the floating point counterpart of $f_i^{(0)}$. Let $\varepsilon \geq 0$ denote the approximation error in the following sense. For every $i = 1, \ldots, n$, we have

$$\left\|f_i^{(0)} - f_{i,\varepsilon}^{(0)}\right\|_2 \leq \varepsilon. \tag{5.6}$$

Then, from the update formula (2.9) for the right-hand side vectors, we obtain that for $r = 1, \dots, k-1$,

$$\begin{aligned}\left\|f^{(r)}_{i\cdot 2^r} - f^{(r)}_{i\cdot 2^r,\varepsilon}\right\|_2 &\le \left\|f^{(r-1)}_{i\cdot 2^r} - f^{(r-1)}_{i\cdot 2^r,\varepsilon}\right\|_2 + \left\|T^{(r)}\left(A^{(r)}\right)^{-1}\right\| \\ &\quad\cdot \left(\left\|f^{(r-1)}_{i\cdot 2^r-2^{r-1}} - f^{(r-1)}_{i\cdot 2^r-2^{r-1},\varepsilon}\right\|_2 + \left\|f^{(r-1)}_{i\cdot 2^r+2^{r-1}} - f^{(r-1)}_{i\cdot 2^r+2^{r-1},\varepsilon}\right\|_2\right) \\ &\le \left\|f^{(r-1)}_{i\cdot 2^r} - f^{(r-1)}_{i\cdot 2^r,\varepsilon}\right\|_2 \\ &\quad+ \frac{1}{2}\left(\left\|f^{(r-1)}_{i\cdot 2^r-2^{r-1}} - f^{(r-1)}_{i\cdot 2^r-2^{r-1},\varepsilon}\right\|_2 + \left\|f^{(r-1)}_{i\cdot 2^r+2^{r-1}} - f^{(r-1)}_{i\cdot 2^r+2^{r-1},\varepsilon}\right\|_2\right). \end{aligned} \tag{5.7}$$

Hence,

$$\left\|f^{(r)}_{i\cdot 2^r} - f^{(r)}_{i\cdot 2^r,\varepsilon}\right\|_2 \le g^{(r)}(\varepsilon, \delta), \tag{5.8}$$

where

$$\begin{aligned}&g^{(0)}(\varepsilon,\delta) = \varepsilon, \\ &g^{(r)}(\varepsilon,\delta) = 2g^{(r-1)}(\varepsilon,\delta) + \delta = 2^r\varepsilon + (2^r-1)\delta, \quad r = 1,\dots,k-1.\end{aligned} \tag{5.9}$$

The positive parameter δ denotes the upper limit for the roundoff error resulting from the computations which are performed at each level r.

Let us analyze the backsubstitution stage. When $r = k-1$, we have

$$\begin{aligned}\left\|u_{(n+1)/2} - u_{(n+1)/2,\varepsilon}\right\|_2 &\le \left\|\left(A^{(k-1)}\right)^{-1}\right\| g^{(k-1)}(\varepsilon,\delta) \\ &\le 2^{k-2}g^{(k-1)}(\varepsilon,\delta) = \gamma^{(k-1)}(\varepsilon,\delta).\end{aligned} \tag{5.10}$$

For the levels $r = k-2, \dots, 0$, we obtain, according to the backsubstitution scheme (2.11), that

$$\begin{aligned}\left\|u_{j\cdot 2^{r+1}-2^r} - u_{j\cdot 2^{r+1}-2^r,\varepsilon}\right\|_2 &\le \left\|\left(A^{(r)}\right)^{-1}T^{(r)}\right\| 2\gamma^{(r+1)}(\varepsilon,\delta) \\ &\quad+ \left\|\left(A^{(r)}\right)^{-1}\right\| g^{(r)}(\varepsilon,\delta) + \delta \\ &\le \gamma^{(r+1)}(\varepsilon,\delta) + 2^{r-1}g^{(r)}(\varepsilon,\delta) + \delta = \gamma^{(r)}(\varepsilon,\delta).\end{aligned} \tag{5.11}$$

Hence, the maximum error of the solution blocks u_j, which are computed at the level $k-1 \ge r \ge 0$, is bounded from above with $\gamma^{(r)}(\varepsilon,\delta)$, where

$$\begin{aligned}&\gamma^{(k-1)}(\varepsilon,\delta) = 2^{k-2}g^{(k-1)}(\varepsilon,\delta), \\ &\gamma^{(r)}(\varepsilon,\delta) = 2^{k-2}g^{(k-1)}(\varepsilon,\delta) + \sum_{i=r}^{k-2}\left(2^{i-1}g^{(i)}(\varepsilon,\delta) + \delta\right), \quad r = k-2,\dots,0.\end{aligned} \tag{5.12}$$

Since the error accumulates most to the blocks which are computed at the level $r = 0$, we have proven the following result.

THEOREM 5.1. *Let f_i, $i = 1, \dots, n$ be the accurate right-hand side vector blocks and let $f_{i,\varepsilon}$ be the floating point counterpart of f_i, for which it holds that*

$$\|f_i - f_{i,\varepsilon}\|_2 \le \varepsilon.$$

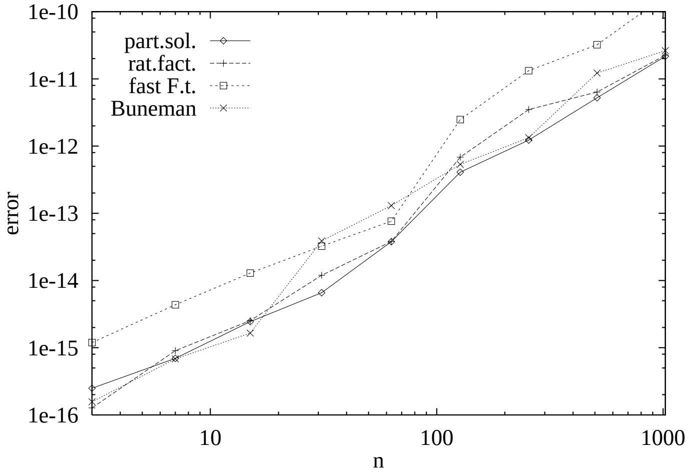


FIG. 5.1. *The stability of the partial solution, rational polynomial factorization, FFT, and Buneman's variants. Error is the maximum blockwise Euclidean norm of the error vector used in Theorem* 5.1.

Let u_i be the solution vector blocks corresponding to f_i, which are obtained using exact arithmetics, and let $u_{i,\varepsilon}$ be the solution vector blocks corresponding to $f_{i,\varepsilon}$, which are computed with floating point arithmetics. Then

$$\|u_i - u_{i,\varepsilon}\|_2 \leq \gamma^{(0)}(\varepsilon, \delta) \leq \frac{7}{48}(n+1)^2(\varepsilon + \delta), \quad i = 1, \ldots, n,$$

where δ denotes the upper bound of the roundoff error appearing in the computations at each level r.

It should be noted that when solving the model Poisson problem, the spectral radius of the inverse of the coefficient matrix is $\mathcal{O}(n^2)$ as n grows. According to Theorem 5.1, the blockwise error cannot, asymptotically, grow faster than $\mathcal{O}(n^2)$.

5.2. Numerical experiments demonstrating stability. In this section, we shall numerically demonstrate the stability properties of some of the proposed variants when solving the Poisson problems. We have tested all the different techniques proposed in this paper. All the considered methods were implemented using the formulas given earlier in this paper. Furthermore, for comparison, we also tested the well-known stable Buneman's second variant of the standard cyclic reduction method which was implemented according to formulas given in [4].

First, we generated a random solution vector u whose components were uniformly distributed to the interval $(0, 1)$. From the vector u, we computed the corresponding right-hand side vector f by multiplying u by the coefficient matrix A given by (2.1) and (2.2). We then recovered the solution with all the proposed methods and computed the norm of the blockwise error appearing in the error estimate of Theorem 5.1. All the computations were performed in double-precision arithmetics conforming to the IEEE standard.

Some of the test results are plotted in Figure 5.1. We chose the variants which use the partial solution method, the rational polynomial factorization, and the FFTs. The results of Buneman's method are also given for comparison. The missing FACR(l) method and the partial fraction approach behave in a similar manner to those shown in the figure.

As can be seen, the errors can be accurately estimated by Theorem 5.1. In particular, the methods presented in this paper appear to have similar stability properties in this experiment as does the classical Buneman's method.

6. Conclusions. In this paper, we have presented a nonstandard cyclic reduction method which, being combined with the partial solution technique, leads to the PSCR method. The cyclic reduction-type formulation also made it possible to adopt the other well-known techniques to the considered nonstandard method leading, for example, to the $\mathcal{O}(n^2 \log\log n)$ FACR variant. We showed that in the considered framework, the use of the partial fraction expansions for the rational polynomial factorizations is equivalent to the use of the partial solution technique in the case of uniform discretization meshes. The considered partial fraction variant is computationally as expensive as the Buneman's second variant of the standard cyclic reduction method, whereas the partial solution variant is approximately twice more expensive. However, the partial solution variant is applicable to a wider class of problems and, furthermore, its efficiency can be considerably improved by increasing its radix from two. Also, we modified the error analysis of the standard cyclic reduction technique to our nonstandard framework. According to the stability estimate and our numerical experiments, the accuracy of the PSCR method is comparable to the well-known Buneman's variants of the block cyclic reduction when solving the Poisson equation.

Acknowledgment. The authors wish to thank Professor Yuri A. Kuznetsov for introducing them to the PSCR method.

REFERENCES

[1] M. Abramowitz and I. A. Stegun, *Handbook of Mathematical Functions*, Dover, New York, 1968.

[2] A. Banegas, *Fast Poisson solvers for problems with sparsity*, Math. Comp., 37 (1978), pp. 441–446.

[3] O. Buneman, *A Compact Non-iterative Poisson Solver*, Technical report 294, Stanford University Institute for Plasma Research, Stanford, CA, 1969.

[4] B. L. Buzbee, G. H. Golub, and C. W. Nielson, *On direct methods for solving Poisson's equations*, SIAM J. Numer. Anal., 7 (1970), pp. 627–656.

[5] R. W. Hockney, *A fast direct solution of Poisson's equation using Fourier analysis*, J. Assoc. Comput. Mach., 12 (1965), pp. 95–113.

[6] R. W. Hockney, *The potential calculation and some applications*, Methods Comput. Phys., 9 (1970), pp. 135–211.

[7] Yu. A. Kuznetsov, *Matrix computational processes in subspaces*, in Computing Methods in Applied Sciences and Engineering, VI, R. Glowinski and J.-L. Lions, eds., North–Holland, Amsterdam, New York, 1984, pp. 15–31.

[8] Yu. A. Kuznetsov, *Numerical methods in subspaces*, in Vychisl. Processy i Sistemy II, G. I. Marchuk, ed., Nauka, Moscow, 1985, pp. 265–350 (in Russian).

[9] Yu. A. Kuznetsov and A. M. Matsokin, *On partial solution of systems of linear algebraic equations*, Soviet J. Numer. Anal. Math. Modelling, 4 (1989), pp. 453–468.

[10] Yu. A. Kuznetsov and T. Rossi, *Fast direct method for solving algebraic systems with separable symmetric band matrices*, East-West J. Numer. Math., 4 (1996), pp. 53–68.

[11] T. Rossi and J. Toivanen, *A parallel fast direct solver for block tridiagonal systems with separable matrices of arbitrary dimension*, SIAM J. Sci. Comput., to appear.

[12] P. N. Swarztrauber, *A direct method for the discrete solution of separable elliptic equations*, SIAM J. Numer. Anal., 11 (1974), pp. 1136–1150.

[13] P. N. Swarztrauber, *The methods of cyclic reduction, Fourier analysis and the FACR algorithm for the discrete solution of Poisson's equation on a rectangle*, SIAM Rev., 19 (1977), pp. 490–501.

[14] P. N. Swarztrauber, *Approximate cyclic reduction for solving Poisson's equation*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 199–209.

[15] P. N. Swarztrauber and R. A. Sweet, *Efficient FORTRAN subprograms for the solution of separable elliptic partial differential equations*, ACM Trans. Math. Software, 5 (1979), pp. 352–364.

[16] R. A. Sweet, *A parallel and vector variant of the cyclic reduction algorithm*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 761–765.

[17] P. S. Vassilevski, *Fast algorithm for solving a linear algebraic problem with separable variables*, C. R. Acad. Bulgare Sci., 37 (1984), pp. 305–308.

SIAM J. Matrix Anal. Appl.
Vol. 20, No. 3, pp. 646–666

LOGARITHMIC NORMS FOR MATRIX PENCILS[*]

INMACULADA HIGUERAS[†] AND BERTA GARCÍA-CELAYETA[†]

Abstract. We extend the usual concepts of least upper bound norm and logarithmic norm of a matrix to matrix pencils. Properties of these seminorms and logarithmic norms are derived. This logarithmic norm can be used to study the growth of the solutions of linear variable coefficient differential algebraic systems.

Key words. matrix pencil, logarithmic norm, Lyapunov stability, differential algebraic system

AMS subject classifications. 15A22, 34D20, 65L07

PII. S0895479897325955

1. Introduction.

We are interested in the growth of the solutions of the homogeneous linear time varying differential system

$$A(t)x'(t) + B(t)x(t) = 0 \tag{1.1}$$

with $A(t)$, $B(t)$ $n \times n$ matrices. If the matrix $A(t)$ is regular, then (1.1) is an ordinary differential equation (ODE); otherwise, (1.1) is a differential algebraic system (DAE). During recent years, DAEs have been deeply studied [7, 12, 2, 8, 3, 15]. One important characteristic of DAEs is that we cannot impose an initial condition of the form $x(t_0) = x_0$ for any $x_0 \in \mathbb{R}^n$; only some of them, called consistent initial conditions, are admissible.

The concept of logarithmic norm for a matrix was introduced in 1958 by Dahlquist and Lozinskij as a tool to study the growth of solutions to ODEs and the error growth in discretization methods for their approximate solution. For details see [5]. For a matrix A, the logarithmic norm is defined by

$$\mu[A] = \lim_{\Delta \to 0^+} \frac{\|I + \Delta A\| - 1}{\Delta}.$$

The norm here is generally assumed to be a least upper bound (l.u.b.) norm,

$$\|A\| = \max_{x \neq 0} \frac{\|Ax\|}{\|x\|}.$$

Properties of norms and logarithmic norms can be found in [11, 5, 4]. In the following lemmas we collect some well-known ones.

Lemma 1.1. *For an $n \times n$ matrix A with spectral radius $\rho(A)$ we have the following:*

(i) *$\rho(A) \leq \|A\|$ for any l.u.b. norm.*

(ii) *given $\varepsilon > 0$, there exists a norm such that $\|A\| \leq \rho(A) + \varepsilon$.*

[*] Received by the editors August 11, 1997; accepted for publication (in revised form) by V. Mehrmann May 20, 1998; published electronically March 23, 1999. This research was supported by the Gobierno de Navarra project “Métodos numéricos en la simulación de fenómenos evolutivos y en la aproximación de superficies” (O.F. 510/1994).

http://www.siam.org/journals/simax/20-3/32595.html

[†] Departamento de Matemática e Informática, Universidad Pública de Navarra, 31006 Pamplona, Spain (higueras@upna.es, berta@upna.es).

(iii) $\varepsilon = 0$ *is possible in* (ii) *if and only if no eigenvalue* λ *of* A *with* $|\lambda| = \rho(A)$ *is defect. (An eigenvalue is defect if it has a noncomplete set of eigenvectors.)*

LEMMA 1.2. *Let* A *and* B *be* $n \times n$ *matrices. Denote by* $\alpha(A)$ *the spectral abscissa of* A *(i.e., the maximal real part of the eigenvalues of* A*) and by* $\lambda_i[A]$ *the eigenvalues of* A*. Then we have the following:*

(i) $-\|A\| \leq -\mu[-A] \leq \mathrm{Re}\ (\lambda_i[A]) \leq \mu[A] \leq \|A\|,\ 1 \leq i \leq n.$

(ii) $\mu[cA] = |c|\mu[\,sgn\,(c)A]$ *for all* $c \in \mathbb{R}$.

(iii) $\mu[A + zI] = \mu[A] + \mathrm{Re}\ z$ *for all* $z \in C$.

(iv) $\max(\mu[A] - \mu[-B], -\mu[-A] + \mu[B]) \leq \mu[A + B] \leq \mu[A] + \mu[B]$.

(v) *If* $\|\cdot\|$ *is an inner product norm, then*

$$\mu[A] = \max_{x \neq 0} \frac{\langle Ax, x\rangle}{\langle x, x\rangle}.$$

This means that the logarithmic norm is the smallest one-sided Lipschitz constant.

(vi) *If* $\|\cdot\|$ *is an inner product norm, then the logarithmic norm is the smallest element of the set*

$$M = \{\theta / \|e^{At}\| \leq e^{\theta t}, t \geq 0\}. \tag{1.2}$$

Thus $\mu[A]$ *provides the optimal exponential bound for* $\|e^{At}\|$ *for* $t \geq 0$*. We get* $\|e^{At}\| \leq 1$ *for all* $t \geq 0$ *if and only if* $\mu[A] \leq 0$*.*

(vii) *The logarithmic norm can also be written as*

$$\mu[A] = \lim_{h \to 0^+} \frac{\ln \|e^{Ah}\|}{h}. \tag{1.3}$$

We briefly note here the connection between logarithmic norms and linear constant coefficient ODEs. If we consider the problem

$$x'(t) = Ax(t), \tag{1.4}$$

whose solution is $x(t) = e^{At}x(0)$, it can be proved that

$$\|x(t)\| \leq e^{\mu[A]t}\|x(0)\|, \qquad t \geq 0. \tag{1.5}$$

Thus $\|e^{At}\| \leq e^{\mu[A]t}$ and $\mu[A]$ is in the set M (1.2). Moreover, if $\mu[A] < 0$, (1.5) implies that the solution is asymptotically stable.

To construct the set M (1.2) and prove that $\mu[A]$ is in M, it has been used that the solution of the linear constant coefficient ODE (1.4) can be expressed in terms of the exponential of a matrix. In order to work in a similar way for matrix pencils, we need the solution of (1.1). We must also remember some concepts on matrix pencils and DAEs that we will use later. For details see [7, 3, 6] and the introduction in [16].

In the case of the linear constant coefficient DAE (1.1), in order to get uniqueness for the solution of initial value problems, the pencil (A, B) must be regular. This means that there exists $\lambda \in C$ such that $\det(\lambda A + B) \neq 0$. Otherwise, if $\det(\lambda A + B) = 0$ for all λ, the pencil is called singular. Regularity of the pencil is also needed to get uniqueness of the numerical solution for the DAE with the usual methods for ODEs.

Observe that if the pencil (A, B) is regular, then $\mathrm{Ker}\,(A) \cap \mathrm{Ker}\,(B) = \{0\}$. Otherwise, for a $v \in \mathrm{Ker}\,(A) \cap \mathrm{Ker}\,(B)$, we have $(\lambda A + B)v = 0$ for any λ, and the pencil is singular.

For a regular pencil, $\det(\lambda A + B)$ is a polynomial with degree less than or equal to n. The complex values λ such that $\det(\lambda A + B) = 0$ are called finite eigenvalues of the pencil. Observe that not every pencil has finite eigenvalues. For example, the pencil (A, B) for

$$A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \qquad B = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

has no finite eigenvalues. It is said that infinity is an eigenvalue of the pencil (A, B) if $\det(A) = 0$. The ODE (1.4) has only finite eigenvalues; the DAE (1.1) have infinite eigenvalues and may have finite ones. We will consider only the set of finite eigenvalues of the pencil and denote it by $\sigma(A, B)$, the finite spectrum of (A, B):

$$\sigma(A, B) = \{\lambda \in C / \det(\lambda A + B) = 0\}.$$

For $\lambda \in C$, the vectors $v \in C^n$ such that $(\lambda A + B)v = 0$ are called eigenvectors associated with the finite eigenvalue λ. Observe that for any eigenvector v associated with finite eigenvalues, we have $Av \neq 0$. The eigenvectors associated with the eigenvalue infinity are all the vectors in $\mathrm{Ker}\,(A)$.

The spectral radius, denoted by $\rho(A, B)$, is defined as

$$\rho(A, B) = \max\{|\lambda| / \lambda \in \sigma(A, B)\}.$$

For a regular matrix pencil (A, B), there are regular matrices P and Q such that

$$PAQ = \begin{pmatrix} I_r & \\ & N \end{pmatrix}, \qquad PBQ = \begin{pmatrix} J & \\ & I_{n-r} \end{pmatrix},$$

where N is a block diagonal $(n-r) \times (n-r)$ matrix formed by Jordan blocks associated with the zero eigenvalue, and J, an $r \times r$ matrix, is formed by Jordan blocks associated with the finite eigenvalues of the pencil. The index of nilpotency of the DAE (or index) is the order of nilpotency of the matrix N. The pencil $(\mathrm{diag}(I_r, N), \mathrm{diag}(J, I_{n-r}))$ is called the Weierstrass–Kronecker canonical form of (A, B). Observe that $\sigma(A, B) = \sigma(-J)$.

If k is the index of the pencil, we also have

$$\mathbb{R}^n = \mathrm{Ker}\,(\hat{A}^k) \oplus \mathrm{Img}\,(\hat{A}^k),$$

with $\hat{A} = (cA + B)^{-1}A$, $\hat{B} = (cA + B)^{-1}B$ for any c such that $cA + B$ is regular. It can be proved that $\hat{A}$ and $\hat{B}$ commute. The above expressions are independent of the constant c chosen to construct the matrices $\hat{A}$ and $\hat{B}$. If we want to note the specific value c used, we will write $\hat{A}_c$ and $\hat{B}_c$. To get the solution of the DAE, we decompose

$$x(t) = \hat{A}^D \hat{A} x(t) + (I - \hat{A}^D \hat{A}) x(t) = y(t) + z(t),$$

with A^D the Drazin inverse of the matrix A [3, p. 121]. One of the properties of the Drazin inverse is that $\mathrm{Img}\,(\hat{A}^D \hat{A}) = \mathrm{Img}\,(\hat{A}^k)$. In the homogeneous case, $z(t) = 0$ and the function $y(t)$ solves the differential problem

$$y'(t) = -\hat{A}^D \hat{B} y(t). \tag{1.6}$$

Thus $y(t) = e^{-\hat{A}^D \hat{B} t} y_0$, and the solution is given by

$$x(t) = e^{-\hat{A}^D \hat{B} t} y_0 = e^{-\hat{A}^D \hat{B} t} \hat{A}^D \hat{A} x_0$$

for any $x_0 \in \mathbb{R}^n$. Only one part of the solution, namely $y(t) = \hat{A}^D\hat{A}x(t)$, is given by the constant coefficient differential equation (1.6). Therefore, to know the behavior of the solution of the DAE we must study this ODE without forgetting the DAE context. For example, for (1.6) we have to consider initial conditions only in the lower dimension space $\operatorname{Img}(\hat{A}^D\hat{A})$. If we consider a solution for the problem (1.6), we get

$$\hat{A}^D\hat{A}y(t) = \hat{A}^D\hat{A}e^{-\hat{A}^D\hat{B}t}y_0 = \hat{A}^D\hat{A}y_0 + [e^{-\hat{A}^D\hat{B}t} - I]y_0 = (\hat{A}^D\hat{A} - I)y_0 + y(t),$$

where we have used that $\hat{A}^D\hat{A}\hat{A}^D = \hat{A}^D$. Thus the solution of any linear differential system with coefficient matrix $-\hat{A}^D\hat{B}$ and y_0 in $\operatorname{Img}(\hat{A}^D\hat{A})$ is in $\operatorname{Img}(\hat{A}^D\hat{A})$. Observe that the consistent initial condition is $x(0) = \hat{A}^D\hat{A}x_0 \in \operatorname{Img}(\hat{A}^D\hat{A})$, and the solution $x(t)$ is also in $\operatorname{Img}(\hat{A}^D\hat{A})$. Observe too that we get the same expression if we restrict to vectors x_0 in $\operatorname{Img}(\hat{A}^D\hat{A})$.

The rest of the paper is structured as follows. In section 2 we define a seminorm for a matrix pencil and derive some properties. In section 3 we define the logarithmic norm for a matrix pencil and get some properties. In section 4 we relate the qualitative behavior of the solutions of linear variable coefficient DAEs with the logarithmic norm of a pencil constructed from the original one. Some examples are given. Finally, in section 5 we give other characterizations of the logarithmic norm for some subspaces V.

2. Seminorms. Given a matrix pencil (A, B), we are looking for an extension of the l.u.b. norm for a matrix, such that a similar property to (i) in Lemma 1.1 holds, i.e., the spectral radius, whenever it exists, should be less than or equal to this number. In order to find this value, we proceed in a way similar to how it is done for l.u.b. norms. If v is an eigenvector of the pencil associated with the eigenvalue λ, we have

$$|\lambda|\|Av\| = \|Bv\|,$$

and as $Av \neq 0$, we can consider the eigenvector $w = v/\|Av\|$,

$$|\lambda| = \|Bw\|.$$

We can be tempted to bound this expression by

$$\sup_{v\in\mathbb{R}^n, \|Av\|=1} \|Bv\|,$$

but a simple example shows us that this supreme is not always finite.

Example 1. For A and B,

$$A = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}, \qquad B = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix},$$

the pencil is regular. If $v_n = (n, 1)^t$, then $\|Av_n\|_\infty = 1$, $\|Bv_n\|_\infty = n$, and the set $\{\|Bv\|/\|Av\| = 1\}$ is not bounded.

DEFINITION 2.1. *We consider a vectorial norm $\|\cdot\|$. Given a matrix pencil (A, B) and a set $V \subseteq \mathbb{R}^n$ such that $\big\{\|Bv\|/v \in V, \|Av\| = 1\big\}$ is nonempty and bounded, we define $\|A, B\|_V$ by*

$$\|A, B\|_V = \sup_{v\in V, \|Av\|=1} \|Bv\|. \tag{2.1}$$

We characterize the sets for which (2.1) exists.

Proposition 2.2. *Let $V \subseteq \mathbb{R}^n$ be a subspace such that $V \neq \{0\}$ and $V \cap \operatorname{Ker}(A) = \{0\}$. Then $\|A, B\|_V$ is well defined.*

Proof. Consider the linear, continuous, and injective function $f : V \to \mathbb{R}^n$, $v \hookrightarrow Av$. We can define $f^{-1} : f(V) \to V$, $w \hookrightarrow f^{-1}(w)$, which is also a continuous function.

As the set

$$T = \{w \in f(V)/\|w\| = 1\}$$

is compact in $f(V)$, the set

$$f^{-1}(T) = \{v \in V/\|Av\| = 1\}$$

is also a compact set in V and in $\mathbb{R}^n$. Therefore there exists

$$\sup_{v\in V, \|Av\|=1} \|Bv\| = \max_{v\in V, \|Av\|=1} \|Bv\|. \qquad \square$$

Definition 2.3. *We say that a subspace $V \neq \{0\}$ is an admissible subspace for $\|A, \cdot\|_V$ if $V \cap \operatorname{Ker}(A) = \{0\}$.*

Remark. We can also define $\|A, B\|_V$ for singular pencils (A, B). $\square$

In what follows we will always consider admissible subspaces V. There are other equivalent definitions to (2.1).

Proposition 2.4. *If V is an admissible subspace for $\|A, \cdot\|$, then*

$$\begin{aligned}\|A, B\|_V &= \sup_{\substack{v\in V\\ \|Av\|\leq 1}} \|Bv\| = \sup_{\substack{v\in V\\ \|Av\|<1}} \|Bv\| = \sup_{\substack{v\in V\\ \|Av\|\neq 0}} \frac{\|Bv\|}{\|Av\|}\\ &= \inf\{c/\|Bx\| \leq c\|Ax\|, \forall x \in V\}.\end{aligned}$$

Moreover, all the suprema are maxima.

Proof. The proof is straightforward. $\square$

Proposition 2.5. *Properties of* (2.1) are as follows:

(i) $\|Bv\| \leq \|A, B\|_V.\|Av\|$ *for all* $v \in V$.

(ii) *For any* α, $\beta \in C$, *with* $\alpha \neq 0$, *we have* $\|\alpha A, \beta B\|_V = \frac{|\beta|}{|\alpha|}\|A, B\|_V$.

(iii) $\|A, B + C\|_V \leq \|A, B\|_V + \|A, C\|_V$.

(iv) *If* $V \subseteq \operatorname{Ker}(B)$, *then* $\|A, B\|_V = 0$. *Thus* $\|A, B\|_V = 0$ *does not imply* $B = 0$. *The application*

$$P_{A,V} : \mathcal{L}(\mathbb{R}^n) \to \mathbb{R}, \qquad B \hookrightarrow P_{A,V}(B) = \|A, B\|_V$$

is a seminorm.

(v) *If V contains an eigenvector associated with any eigenvalue λ of (A, B), then*

$$|\lambda| \leq \|A, B\|_V. \tag{2.2}$$

(vi) *If V contains an eigenvector associated with any eigenvalue λ of (A, B) such that $|\lambda| = \rho(A, B)$, then*

$$\rho(A, B) \leq \|A, B\|_V. \tag{2.3}$$

(vii) *Let P be a nonsingular matrix; then $\|AP, BP\|_V = \|A, B\|_{PV}$, where $PV = \{Px/x \in V\}$. In particular, if A is not singular, then*

$$\|A, B\|_{\mathbb{R}^n} = \|I, BA^{-1}\|_{\mathbb{R}^n} = \|BA^{-1}\|.$$

Proof. The proof is immediate from the definition and the properties of vectorial norms. □

Remark. For any nonsingular matrix P, in general $\|PA, PB\|_V \neq \|A, B\|_V$. For example, for

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \qquad B = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \qquad P = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix},$$

and $V = \langle (1,1)^t \rangle$,

$$\|A, B\|_V = \frac{\|(2,1)^t\|}{\|(1,0)^t\|}, \qquad\qquad \|PA, PB\|_V = \frac{\|(2,3)^t\|}{\|(1,1)^t\|}. \qquad □$$

Remark. The definition depends on V. In general $\|A, B\|_V \neq \|A, B\|_W$ even if $\mathbb{R}^n = \operatorname{Ker}(A) \oplus V = \operatorname{Ker}(A) \oplus W$. Consider for example

$$A = \begin{pmatrix} 1 & & \\ & 1 & \\ & & 0 \end{pmatrix}, \qquad B = \begin{pmatrix} 1 & & \\ & 1 & \\ & & 1 \end{pmatrix}.$$

This is an index 1 DAE, with finite eigenvalues $\lambda = -1$ (double), $\operatorname{Ker}(A) = \langle (0,0,1)^t \rangle$. We consider $V_1 = \langle (1,0,0)^t, (0,1,0)^t \rangle$. Then

$$\max_{v \in V_1, \|Av\| \neq 0} \frac{\|Bv\|}{\|Av\|} = \max_{\substack{v=(v_1,v_2,0)^t \\ \|(v_1,v_2,0)^t\| \neq 0}} \frac{\|(v_1, v_2, 0)^t\|}{\|(v_1, v_2, 0)^t\|} = 1.$$

We consider now $V_2 = \langle (1,0,0)^t, (0,1,1)^t \rangle$. Then

$$\max_{v \in V_2, \|Av\| \neq 0} \frac{\|Bv\|}{\|Av\|} = \max_{\substack{v=(v_1,v_2,v_2)^t \\ \|(v_1,v_2,v_2)^t\| \neq 0}} \frac{\|(v_1, v_2, v_2)\|}{\|(v_1, v_2, 0)\|} = \begin{cases} \sqrt{2} \text{ in } \|\cdot\|_2, \\ 2 \text{ in } \|\cdot\|_1, \text{ and} \\ 1 \text{ in } \|\cdot\|_\infty. \end{cases} \quad □$$

Remark. Observe that for $A = I_n$ and $V = \mathbb{R}^n$, $\|I_n, B\|_{\mathbb{R}^n} = \|B\|$, but we can also use $\|I_n, B\|_V := \|B\|_V$ for any other lower dimension subspace. Then

$$\|Bx\| \leq \|B\|_V \|x\| \qquad \text{for all } x \in V,$$

and if ν is a constant such that

$$\|Bx\| \leq \nu \|x\| \qquad \text{for all } x \in V,$$

then $\|B\|_V \leq \nu$. □

Remark. If $V = \operatorname{Img}(C)$ and $BC = C$, then $\|I_n, B\|_V = 1$. This situation occurs for $V = \operatorname{Img}(A\hat{A}^D\hat{A})$. A simple computation gives $A\hat{A}^D(cA + B)^{-1}A\hat{A}^D\hat{A} = A\hat{A}^D\hat{A}\hat{A}^D\hat{A} = A\hat{A}^D\hat{A}$, and thus

$$\|A\hat{A}^D(cA + B)^{-1}\|_{\operatorname{Img}(A\hat{A}^D\hat{A})} = 1. \qquad □$$

For the Euclidean norm, $\|A\|_2 = \rho^{1/2}(A^t A)$. For matrix pencils, we get the following result.

Proposition 2.6. *Consider the $n \times s$ matrix S whose columns are a basis of V. Then*

$$\|A, B\|_{V,2} = \sqrt{\rho(S^t A^t AS, S^t B^t BS)}.$$

Proof. We compute

$$\|A, B\|^2_{V,2} = \max_{\substack{v\in V\\ v\neq 0}} \frac{\|Bv\|^2}{\|Av\|^2} = \max_{\substack{v\in V\\ v\neq 0}} \frac{v^t B^t Bv}{v^t A^t Av} = \max_{\substack{x\in\mathbb{R}^s\\ x\neq 0}} \frac{x^t S^t B^t BSx}{x^t S^t A^t ASx}.$$

Observe that $S^t A^t AS$ is a positive matrix because $V \cap \operatorname{Ker}(A) = 0$. Thus the pencil $(S^t A^t AS, S^t B^t BS)$ is positive. Therefore the eigenvalues are positive and, from [6],

$$\max_{\substack{x\in\mathbb{R}^s\\ x\neq 0}} \frac{x^t S^t B^t BSx}{x^t S^t A^t ASx} = \lambda_{max}(S^t A^t AS, S^t B^t BS) = \rho(S^t A^t AS, S^t B^t BS). \qquad \square$$

Given a nonsingular matrix T and a vectorial norm $\|\cdot\|$, we can define $\|x\|_T = \|Tx\|$. The corresponding subordinate matrix norm satisfies $\|A\|_T = \|TAT^{-1}\|$, and the logarithmic norm becomes $\mu_T(A) = \mu(TAT^{-1})$. It is said that the norms $\|\cdot\|$ and $\|\cdot\|_T$ are similar.

For matrix pencils we can define similar concepts. Given a vectorial norm $\|\cdot\|$ and nonsingular matrices T and $\tilde{T}$,

$$\|A, B\|_{T,V} = \max_{\substack{x\in V\\ x\neq 0}} \frac{\|Bx\|_T}{\|Ax\|_T} = \max_{\substack{x\in V\\ x\neq 0}} \frac{\|TBx\|}{\|TAx\|} = \max_{\substack{y\in \tilde{T}V\\ y\neq 0}} \frac{\|TB\tilde{T}^{-1}y\|}{\|TA\tilde{T}^{-1}y\|},$$

where $\tilde{T}V = \{\tilde{T}x/x \in V\}$. Thus

$$\|A, B\|_{T,V} = \|TA, TB\|_V = \|TA\tilde{T}^{-1}, TB\tilde{T}^{-1}\|_{\tilde{T}V}.$$

If V is $\tilde{T}$-invariant, then

$$\|A, B\|_{T,V} = \|TA\tilde{T}^{-1}, TB\tilde{T}^{-1}\|_V.$$

Observe that (2.3) is the analogous expression to Lemma 1.1(i). For the seminorm defined for the matrix pencil, equivalent results to Lemma 1.1(ii) can be proved.

Theorem 2.7. *Let (A, B) be a matrix pencil with spectral radius $\rho(A, B)$. Given $\varepsilon > 0$, there exist a vectorial norm $\|\cdot\|$ and a subspace V such that*

$$\rho(A, B) \leq \|A, B\|_V \leq \rho(A, B) + \varepsilon.$$

Moreover, $\varepsilon = 0$ is possible if and only if no eigenvalue λ of (A, B) with $|\lambda| = \rho(A, B)$ is defect.

Proof. We consider the Kronecker canonical form

$$PAQ = \begin{pmatrix} I_r & \\ & N \end{pmatrix}, \qquad PBQ = \begin{pmatrix} J & \\ & I_{n-r} \end{pmatrix}.$$

Suppose that $J = \operatorname{diag}(J_1, \ldots, J_k)$, with J_i Jordan blocks of size r_i associated with the eigenvalue λ_i. Consider $E = \{(x_1, 0)/x_1 \in \mathbb{R}^r, \|x_1\|_\infty \leq 1\}$ and the block diagonal matrix $\omega(\varepsilon) = \operatorname{diag}(\omega_1, \ldots, \omega_k)$ with each block $\omega_i = \operatorname{diag}(1, \varepsilon, \varepsilon^2, \ldots, \varepsilon^{r_i - 1})$, and

$$\tilde{\omega}(\varepsilon) = \begin{pmatrix} \omega(\varepsilon) & \\ & I_{n-r} \end{pmatrix}.$$

Thus

$$\begin{aligned}&\|\tilde{\omega}(\varepsilon)PAQ\tilde{\omega}(\varepsilon)^{-1}, \tilde{\omega}(\varepsilon)PBQ\tilde{\omega}(\varepsilon)^{-1}\|_{\infty,E}\\ &\quad = \left\| \begin{pmatrix} I_r & \\ & N \end{pmatrix}, \begin{pmatrix} \omega(\varepsilon)J\omega(\varepsilon)^{-1} & \\ & I_{n-r} \end{pmatrix} \right\|_{\infty,E} = \|\omega(\varepsilon)J\omega(\varepsilon)^{-1}\|_\infty.\end{aligned}$$

For a Jordan block J_i associated with the eigenvalue λ_i, we have

$$\omega_i(\varepsilon)J_i\omega_i(\varepsilon)^{-1} = \begin{pmatrix} \lambda_i & 0 & 0 & \cdots \\ \varepsilon & \lambda_i & 0 & \cdots \\ 0 & \varepsilon & \lambda_i & \cdots \\ . & . & . & . \end{pmatrix}.$$

Thus

$$\|\omega(\varepsilon)J\omega(\varepsilon)^{-1}\|_\infty = \rho(J) + \varepsilon = \rho(A,B) + \varepsilon.$$

We consider the subspace $V = Q\tilde{\omega}(\varepsilon)^{-1}E$. If Q_1 is the matrix formed by the first r columns in Q, then

$$V = Q\tilde{\omega}(\varepsilon)^{-1}E = \operatorname{Img}(Q_1\omega(\varepsilon)^{-1}).$$

From the Kronecker canonical form,

$$AQ_1\omega(\varepsilon)^{-1}x = P^{-1}\begin{pmatrix} \omega(\varepsilon)^{-1}x \\ 0 \end{pmatrix},$$

we get $\ker(A) \cap V = \{0\}$, and V is an admissible subspace for $\|A,\cdot\|_V$. If we consider the vectorial norm, $\|\cdot\|_{\infty,\tilde{\omega}(\varepsilon)P}$, then

$$\|A,B\|_{\infty,\tilde{\omega}(\varepsilon)P,V}$$

satisfies the required bound.

As V contains eigenvectors associated with all the finite eigenvalues of the pencil, we also get $\rho(A,B) \le \|A,B\|_V$.

If no eigenvalue λ of (A,B) with $|\lambda| = \rho(A,B)$ is defect, the we can choose ε such that $\|\omega(\varepsilon)J\omega(\varepsilon)^{-1}\|_\infty = \rho(J)$. □

3. Logarithmic norms.

With the seminorms above defined, we can define an extension of the logarithmic norm of a matrix A.

DEFINITION 3.1. *Let (A,B) be a matrix pencil in $\mathbb{R}^n$ and V an admissible subspace for $\|A,\cdot\|_V$. We define the logarithmic norm of the pencil as*

$$\mu_V[A,B] = \lim_{\Delta\to 0^+} \frac{\|A, A-\Delta B\|_V - 1}{\Delta}. \tag{3.1}$$

PROPOSITION 3.2. *The limit (3.1) exists for all matrix pencils (A,B) and all well-defined seminorms.*

Proof. Let $\theta \in (0,1)$. Then

$$\|A, A-\theta\Delta B\|_V = \|A, \theta(A-\Delta B) + (1-\theta)A\|_V \le \theta\|A, A-\Delta B\|_V + (1-\theta)\|A,A\|_V.$$

With the definition of seminorm given, $\|A,A\|_V = 1$; thus

$$\frac{\|A, A-\theta\Delta B\|_V - 1}{\theta\Delta} \le \frac{\theta\|A, A-\Delta B\|_V + (1-\theta) - 1}{\theta\Delta} = \frac{\|A, A-\Delta B\|_V - 1}{\Delta},$$

and consequently

$$h(\Delta) = \frac{\|A, A - \Delta B\|_V - 1}{\Delta}$$

is a nondecreasing function of Δ. From

$$\|A, A - \Delta B\|_V \leq \|A, -\Delta B\|_V + \|A, A\|_V = \Delta \|A, B\|_V + 1$$

and

$$1 = \|A, A\|_V = \|A, A - \Delta B + \Delta B\|_V \leq \|A, A - \Delta B\|_V + \Delta \|A, B\|_V$$

we get

$$-\|A, B\|_V \leq \frac{\|A, A - \Delta B\|_V - 1}{\Delta} \leq \|A, B\|_V, \tag{3.2}$$

and thus the limit (3.1) exists. □

Observe that for the pencil (I_n, B) and $V = \mathbb{R}^n$, (3.1) is $\mu[-B]$.

The next result states some properties of the logarithmic norm defined for matrix pencils.

Proposition 3.3. *The following are properties of the logarithmic norm:*

(i) $-\|A, B\|_V \leq \mu_V[A, B] \leq \|A, B\|_V$.

(ii) *For any α, β in $\mathbb{R}$, $\alpha \neq 0$,*

$$\mu_V[\alpha A, \beta B] = \frac{|\beta|}{|\alpha|} \mu_V[A,\ sgn\,(\alpha \cdot \beta) B].$$

(iii) $\mu_V[A, B + C] \leq \mu_V[A, B] + \mu_V[A, C]$.

(iv) *If V contains an eigenvector associated with any eigenvalue λ of (A, B), then* $\mathrm{Re}\,(\lambda) \leq \mu_V[A, B]$.

(v) $\mu_V[A, B + zA] = \mu_V[A, B] + \mathrm{Re}\, z$ *for all $z \in C$.*

(vi) *If $\|\cdot\|$ is an inner product norm, then*

$$\mu_V[A, B] = \max_{\substack{Ax \neq 0 \\ x \in V}} \frac{\langle Ax, -Bx \rangle}{\langle Ax, Ax \rangle}.$$

This means that the logarithmic norm is the smallest constant (one-sided Lipschitz constant) such that

$$\langle Ax, -Bx \rangle \leq \nu \langle Ax, Ax \rangle \qquad \text{for all } x \in V.$$

(vii) *If A is invertible, and $V = \mathbb{R}^n$, then*

$$\mu_V[A, B] = \mu_V[I_n, BA^{-1}] = \mu[-BA^{-1}].$$

(viii) $\|Bx\| \geq \max(-\mu_V[A, B], -\mu_V[A, -B])$ *for all $x \in V$.*

(ix) *For regular matrices T and $\tilde{T}$,*

$$\mu_{V,T}(A, B) = \mu_V(TA, TB) = \mu_{\tilde{T}V}(TA\tilde{T}^{-1}, TB\tilde{T}^{-1}),$$

and if V is $\tilde{T}$-invariant,

$$\mu_{V,T}(A, B) = \mu_V(TA\tilde{T}^{-1}, TB\tilde{T}^{-1}).$$

Proof. Part (i) follows from (3.2).

(ii) We have

$$\|\alpha A, \alpha A - \Delta\beta B\|_V = \left\|\alpha A, \alpha\left[A - \Delta\frac{\beta}{\alpha}B\right]\right\|_V = \left\|A, A - \Delta\frac{\beta}{\alpha}B\right\|_V.$$

Thus we can compute

$$\begin{aligned}\mu_V[\alpha A, \beta B] &= \lim_{\Delta\to 0^+}\frac{\|\alpha A, \alpha A - \Delta\beta B\|_V - 1}{\Delta} = \lim_{\Delta\to 0^+}\frac{\|A, A - \Delta\frac{\beta}{\alpha}B\|_V - 1}{\Delta}\\ &= \frac{|\beta|}{|\alpha|}\lim_{\Delta\to 0^+}\frac{\|A, A - \Delta\frac{|\beta|}{|\alpha|}\operatorname{sgn}(\alpha\cdot\beta)B\|_V - 1}{\Delta\frac{|\beta|}{|\alpha|}}\\ &= \frac{|\beta|}{|\alpha|}\mu_V[A, \operatorname{sgn}(\alpha\cdot\beta)B].\end{aligned}$$

(iii) Using the properties of the seminorm defined above we have

$$\begin{aligned}\mu_V[A, B + C] &= \lim_{\Delta\to 0^+}\frac{\|A, A - \Delta(B + C)\|_V - 1}{\Delta}\\ &\leq \lim_{\Delta\to 0^+}\left(\frac{\|A, \frac{1}{2}A - \Delta B\|_V - \frac{1}{2}}{\Delta} + \frac{\|A, \frac{1}{2}A - \Delta C\|_V - \frac{1}{2}}{\Delta}\right)\\ &= \lim_{\Delta\to 0^+}\left(\frac{\|A, A - 2\Delta B\|_V - 1}{2\Delta} + \frac{\|A, A - 2\Delta C\|_V - 1}{2\Delta}\right)\\ &= \mu_V[A, B] + \mu_V[A, C].\end{aligned}$$

(iv) We take any $\Delta > 0$. If v is an eigenvector of (A, B) with eigenvalue λ, then v is an eigenvector of $(A, A - \Delta B)$ with eigenvalue $-1 - \Delta\lambda$. By (2.2)

$$\frac{|\Delta\lambda + 1| - 1}{\Delta} \leq \frac{\|A, A - \Delta B\|_V - 1}{\Delta}. \tag{3.3}$$

When $\Delta \to 0^+$, the right-hand side of (3.3) tends to $\mu_V(A, B)$. A simple computation on the left-hand side gives

$$\frac{|1 + \Delta\lambda| - 1}{\Delta} = \frac{2\operatorname{Re}\lambda + \Delta(\operatorname{Re}\lambda)^2 + \Delta(\operatorname{Im}\lambda)^2}{\sqrt{(1 + \Delta\operatorname{Re}\lambda)^2 + (\Delta\operatorname{Im}\lambda)^2} + 1},$$

and thus, when $\Delta \to 0^+$, the left-hand side of (3.3) tends to $\operatorname{Re}(\lambda)$.

(v) A simple computation gives

$$\begin{aligned}\mu_V[A, B + zA] &= \lim_{\Delta\to 0^+}\frac{\|A, A + \Delta(B + zA)\|_V - 1}{\Delta}\\ &= \lim_{\Delta\to 0^+}\frac{|1 + z\Delta|\left\|A, A + \dfrac{\Delta}{|1 + z\Delta|}B\right\|_V - 1}{\Delta}\\ &= \lim_{\Delta\to 0^+}\left(\frac{\left\|A, A + \dfrac{\Delta}{|1 + z\Delta|}B\right\|_V - 1}{\dfrac{\Delta}{|1 + z\Delta|}} + \frac{1 - \dfrac{1}{|1 + z\Delta|}}{\dfrac{\Delta}{|1 + z\Delta|}}\right)\\ &= \mu_V[A, B] + \operatorname{Re} z.\end{aligned}$$

(vi) We can write

$$\begin{aligned}\frac{\|A, A-\Delta B\|_V - 1}{\Delta} &= \max_{\substack{x\in V\\ x\neq 0}} \frac{\|(A-\Delta B)x\| - \|Ax\|}{\Delta\|Ax\|}\\ &= \max_{\substack{x\in V\\ x\neq 0}} \frac{\|(A-\Delta B)x\|^2 - \|Ax\|^2}{\Delta\|Ax\|(\|(A-\Delta B)x\| + \|Ax\|)}\\ &= \max_{\substack{x\in V\\ x\neq 0}} \frac{\langle -Bx, Ax\rangle + \Delta/2\|Bx\|^2}{\|Ax\|^2 + 1/2\|Ax\|(\|(A-\Delta B)x\| - \|Ax\|)},\end{aligned}$$

and therefore

$$\mu_V[A,B] = \lim_{\Delta\to 0^+} \frac{\|A, A-\Delta B\|_V - 1}{\Delta} = \max_{\substack{x\in V\\ x\neq 0}} \frac{\langle -Bx, Ax\rangle}{\|Ax\|^2}.$$

(viii) For $\theta > 0$, and $x \in V$, we have

$$\begin{aligned}\|Bx\| = \frac{\|\theta Bx\|}{\theta} &= \frac{\|Ax - (A-\theta B)x\|}{\theta} \geq \frac{\|Ax\| - \|(A-\theta B)x\|}{\theta}\\ &\geq \frac{\|Ax\| - \|A, A-\theta B\|_V\|Ax\|}{\theta} = \frac{1 - \|A, A-\theta B\|_V}{\theta}\|Ax\|,\end{aligned}$$

and thus

$$\|Bx\| \geq -\mu_V[A,B]\cdot\|Ax\|.$$

In a similar way

$$\begin{aligned}\|Bx\| = \frac{\|-\theta Bx\|}{\theta} &= \frac{\|Ax - (A+\theta B)x\|}{\theta} \geq \frac{\|Ax\| - \|(A+\theta B)x\|}{\theta}\\ &\geq \frac{\|Ax\| - \|A, A+\theta B\|_V\|Ax\|}{\theta} = \frac{1 - \|A, A+\theta B\|_V}{\theta}\|Ax\|,\end{aligned}$$

and we get

$$\|Bx\| \geq -\mu_V[A,-B]\cdot\|Ax\|. \qquad \square$$

Remark. Given a Banach space W, the logarithmic derivative of the vector norm $\|\cdot\|$ is defined in [14] as

$$\mu(u,v) = \lim_{h\to 0^+} \frac{\|u+hv\| - \|u\|}{h\|u\|}$$

for any $u, v \in W$, $u \neq 0$. We have $\mu(u, Au) \leq \mu[A]$, and in the finite-dimensional case,

$$\sup_{u\neq 0} \mu(u, Au) = \mu[A].$$

For two linear mappings (A, B) of W into itself, we can define in a similar way its seminorm and its logarithmic norm. Observe that now both $\|A, B\|_V$ and $\mu_V[A, B]$ can be infinite if dim $V = \infty$. We can write $\mu(Au, -Bu)$ for u such that $Au \neq 0$, and with the properties of the seminorms above defined, it is immediate to prove

$$\mu(Au, -Bu) \leq \mu_V[A, B]$$

for any u such that $Au \neq 0$. In the finite-dimensional case, for admissible subspaces V we have

$$\max_{u \in V} \mu(Au, -Bu) = \mu_V[A, B]. \qquad \square$$

For matrices, $\mu_2[A] = \lambda_{max}((A + A^t)/2)$. For matrix pencils, we have the next result.

PROPOSITION 3.4. *Consider the $n \times s$ matrix S whose columns are a basis of V. Then*

$$\mu_{V,2}[A, B] = \lambda_{max} \left(S^t A^t A S, -S^t \frac{(B^t A + A^t B)}{2} S \right).$$

Proof. We compute

$$\begin{aligned} \mu_{V,2}[A, B] &= \max_{\substack{v \in V \\ v \neq 0}} \frac{-v^t \frac{(B^t A + A^t B)}{2} v}{v^t A^t A v} = \max_{\substack{x \in \mathbb{R}^s \\ x \neq 0}} \frac{-x^t S^t \frac{(B^t A + A^t B)}{2} S x}{x^t S^t A^t A S x} \\ &= \lambda_{max} \left(S^t A^t A S, -S^t \frac{(B^t A + A^t B)}{2} S \right). \qquad \square \end{aligned}$$

4. Growth of solutions. We are in position to study the growth of the solutions of the variable coefficient DAEs

$$A(t)x'(t) + B(t)x(t) = 0, \tag{4.1}$$

with $A(t)$ an $n \times n$ continuously differentiable matrix in $[t_0, \infty)$ and $B(t)$ an $n \times n$ continuous matrix in $[t_0, \infty)$.

We will need the right-hand differential operator D_+ that for a given function $z(t)$ is defined by

$$D_+ z(0) = \lim_{\tau \to 0^+} \frac{z(\tau) - z(0)}{\tau}.$$

Under some hypothesis, we can compute $D_+ \|z(t)\|$.

LEMMA 4.1 (see [5, Lemma 1.5.3]). *Consider the vector valued function $z(t) \in \mathbb{R}^n$ of the real variable t which has the right-hand derivative $\nu(t) = D_+ z(t)$. Then $\|z(t)\|$ has a right-hand derivative $D_+ \|z(t)\|$ and*

$$D_+ \|z(t)\| = \lim_{\Delta \to 0^+} \frac{\|z(t) + \Delta \nu(t)\| - \|z(t)\|}{\Delta}.$$

THEOREM 4.2. *Let V be an admissible subspace for $\|A(t), \cdot\|_V$ such that the solution $x(t)$ of the DAE* (4.1) *is in V. Then*

$$\|A(t)x(t)\| \leq e^{\int_{t_0}^t \mu_V[A(u), B(u) - A^{'}(u)] du} \cdot \|A(t_0)x(t_0)\|. \tag{4.2}$$

Proof. By definition

$$D_+ \|A(t)x(t)\| = \lim_{\Delta \to 0^+} \frac{\|A(t)x(t) + \Delta [A(t)x(t)]^{'}\| - \|A(t)x(t)\|}{\Delta}.$$

We compute

$$\begin{aligned}\|A(t)x(t)+\Delta[A(t)x(t)]^{'}\| &= \|[A(t)+\Delta A^{'}(t)-\Delta B(t)]x(t)\|\\ &\leq \|A(t),A(t)+\Delta A^{'}(t)-\Delta B(t)\|_V \cdot \|A(t)x(t)\|,\end{aligned}$$

where we have used (4.1). The fact that the solution is in V allows us to use property (i) in Proposition 2.5. Hence from

$$\begin{aligned}D_+\|A(t)x(t)\| &\leq \lim_{\Delta\to 0^+}\frac{\|A(t),A(t)+\Delta A^{'}(t)-\Delta B(t)\|_V\cdot\|A(t)x(t)\|-\|A(t)x(t)\|}{\Delta}\\ &= \mu_V[A(t),B(t)-A^{'}(t)]\cdot\|A(t)x(t)\|\end{aligned}$$

we get the differential inequality

$$D_+\|A(t)x(t)\| \leq \mu_V[A(t),B(t)-A^{'}(t)]\cdot\|A(t)x(t)\|,$$

which can be integrated to obtain (4.2). □

Remark. If $A(t)$ is regular, then

$$\mu_{\mathbb{R}}[A(t),B(t)-A'(t)] = \mu_{\mathbb{R}}[-(B(t)-A'(t))A(t)^{-1}].$$

We get the logarithmic norm of the matrix obtained when the change of variables $y(t) = A(t)x(t)$ is done. □

COROLLARY 4.3. *Let V be an admissible subspace for $\|A(t),\cdot\|_V$ such that the solution $x(t)$ of the DAE* (4.1) *is in V. If*

$$\int_{t_0}^{t}\mu_V[A(u),B(u)-A^{'}(u)]du < 0,$$

then $\|A(t)x(t)\|$ is exponentially decreasing.

In order to show that the solutions of the DAE are asymptotically stable, we must

1. find an admissible subspace V containing the solution of the DAE;
2. compute the logarithmic norm;
3. obtain the estimates of $x(t)$ from that of $A(t)x(t)$.

We study steps 1 and 2 for linear constant coefficient, index 1, and tractable with index 2 DAEs.

4.1. Linear constant coefficient DAEs. In the case of constant coefficient DAE, (4.2) is

$$\|Ax(t)\| \leq e^{\mu_V[A,B]t}\|Ax(0)\|, \qquad t \geq 0. \tag{4.3}$$

Observe that $x(0)$ must always be a consistent initial condition, i.e., $x(0) \in \operatorname{Img}(\hat{A}^D\hat{A})$. In this case, we know that the solution is in $\operatorname{Img}(\hat{A}^D\hat{A})$, and thus we will take $V = \operatorname{Img}(\hat{A}^D\hat{A})$. The next lemma shows that this subspace is admissible for $\|A,\cdot\|_V$.

LEMMA 4.4. *For $V = Img(\hat{A}^D\hat{A})$, we have $V \cap \operatorname{Ker}(A) = 0$.*

Proof. If $x \in \operatorname{Img}(\hat{A}^D\hat{A}) = \operatorname{Img}(\hat{A}^k)$ and $x \in \operatorname{Ker}(A)$, then $x \in \operatorname{Ker}(\hat{A}) \subseteq \operatorname{Ker}(\hat{A}^k)$. Thus $V \cap \operatorname{Ker}(A) = 0$. □

Inequality (4.3) shows that if $\mu_V[A,B] < 0$ for $V = \operatorname{Img}(\hat{A}^D\hat{A})$, then

$$\lim_{t\to\infty}\|Ax(t)\| = 0,$$

but this is equivalent to the asymptotic stability of the solution.

PROPOSITION 4.5. $\lim_{t\to\infty}\|Ax(t)\| = 0$ *if and only if* $\lim_{t\to\infty}\|x(t)\| = 0$.

Proof. Observe that $x(t) = \hat{A}^D\hat{A}x(t) = \hat{A}^D(cA+B)^{-1}Ax(t)$. □

4.2. Linear variable coefficient index 1 DAEs [7]. For index 1 DAEs,

$$\mathbb{R}^n = \operatorname{Ker}(A) \oplus S(t),$$

with $S(t) = \{x/B(t)x \in \operatorname{Img}(A(t))\}$. The solution is in $S(t)$. The subspace $V = S(t)$ is an admissible subspace for $\|A(t), \cdot\|_V$. With the help of the canonical projector onto $S(t)$ along $\operatorname{Ker}(A(t))$ [7],

$$P_s(t) = (A(t) + B(t)Q_s(t))^{-1}A(t),$$

we get

$$\begin{aligned}\|x(t)\| = \|P_s(t)x(t)\| &= \|(A(t) + B(t)Q_s(t))^{-1}A(t)x(t)\| \\ &\leq \|(A(t) + B(t)Q_s(t))^{-1}\| \cdot \|A(t)x(t)\|.\end{aligned}$$

Thus imposing conditions on $(A(t) + B(t)Q_s(t))^{-1}$, we can get the desired result.

Example 1 (see [10]). Consider the linear variable coefficient DAE (4.1) for

$$A(t) = \begin{pmatrix} 1 & (1+\varepsilon)t \\ 0 & 0 \end{pmatrix}, \qquad B(t) = \mu \begin{pmatrix} 1 & (1+\varepsilon)t \\ 1 & (1+\varepsilon)t - \varepsilon \end{pmatrix}.$$

This DAE has index 1 for $\varepsilon\mu \neq 0$. Thus $\mathbb{R}^2 = \operatorname{Ker}(A) \oplus S(t)$, with $S(t) = \{x/B(t)x \in \operatorname{Img}(A(t))\}$. The solution is in $S(t)$. The subspace $V = S(t)$ is an admissible subspace for $\|A(t), \cdot\|_V$. In this case,

$$\mu_V[A(t), B(t) - A'(t)] = \frac{1+\varepsilon}{\varepsilon} - \mu.$$

Thus for

$$\frac{1+\varepsilon}{\varepsilon} - \mu < 0,$$

we get that $\|A(t)x(t)\|$ is exponentially decreasing and

$$\lim_{t \to 0} \|A(t)x(t)\| = 0.$$

The canonical projector onto $S(t)$ along $\operatorname{Ker}(A(t))$ is

$$P_s(t) = (A(t) + B(t)Q_s(t))^{-1}A(t) = \frac{1}{\varepsilon} \begin{pmatrix} \varepsilon - t(1+\varepsilon) & (1+\varepsilon)(\varepsilon - t(1+\varepsilon))t \\ 1 & t(1+\varepsilon) \end{pmatrix},$$

and as

$$(A(t) + B(t)Q_s(t))^{-1} = \frac{1}{\varepsilon} \begin{pmatrix} \varepsilon - t(1+\varepsilon) & \frac{(1+\varepsilon)t}{\mu} \\ 1 & -\frac{1}{\mu} \end{pmatrix},$$

the solution is asymptotically stable.

We get the same sufficient condition that is obtained in [10], where for $x = (y, z)^t$, the essentially underlying ODE is computed,

$$z'(t) = \left[\frac{1+\varepsilon}{\varepsilon} - \mu\right] z, \qquad y = [\varepsilon - (1+\varepsilon)t]z,$$

and thus the solution is asymptotically stable if and only if

$$\frac{1+\varepsilon}{\varepsilon} - \mu < 0. \qquad \square$$

In [7], the asymptotical stability for nonlinear transferable DAEs is studied. Linear DAEs are called contractive [7, p. 78] if there is a constant $c \leq 0$ and a symmetric positive definite matrix S such that

$$\langle y, P(t)x \rangle_S \leq -c\|P(t)x\|^2 \tag{4.4}$$

holds for any x, y such that

$$A(t)y + (B(t) - A(t)P'(t))x = 0. \tag{4.5}$$

It is proved that this definition implies contractivity for $\|P(t)x(t)\|_S$, and thus if the canonical projector is continuously differentiable, as $x(t) = P_s x(t)$, asymptotical stability is obtained for the trivial solution. If the DAE is contractive in the sense of Griepentrog and März, in particular as the solutions $x(t)$ and $y(t) = (P(t)x(t))'$ satisfy (4.5), we get

$$\langle (P(t)x(t))', P(t)x(t) \rangle_S \leq -c\|P(t)x(t)\|^2. \tag{4.6}$$

It can be proved [7, p. 43] that

$$y(t) = (P'(t) - P(t)(I + P'(t))A_1^{-1}B(t))P(t)x(t),$$

and thus with the definition of logarithmic norm given in this paper, (4.6) is

$$\mu_V[I_n, -P'(t) + P(t)(I + P'(t))A_1^{-1}B(t)] < 0$$

for $V = \operatorname{Img}(P(t))$.

4.3. Tractable with index 2 DAEs [12]. For these problems $S(t) \cap \operatorname{Ker}(A(t)$ is nontrivial and $A_1(t) = A(t) + B(t)Q(t)$ is singular. If $Q_1(t)$ denotes a projector onto $\operatorname{Ker}(A_1(t))$, and $P_1(t) = I_n - Q_1(t)$, for tractable with index 2 DAEs it holds that the matrix $G_2(t) = A_1(t) + B(t)P(t)Q_1(t)$ is nonsingular. If we denote $\Pi(t) = (I_n - (QQ_1)' - QP_1A_2^{-1}B)PP_1$, the solution [9] is in $\operatorname{Img}\Pi(t)$. The subspace $V = \operatorname{Img}\Pi(t)$ is an admissible subspace for $\|A, \cdot\|_V$. As $A(t)x(t) = A_2(t)P_1(t)Px(t)$, and $PP_1(t)P = PP_1(t)$, we get

$$PP_1(t)x(t) = PA_2(t)^{-1}A_2(t)PP_1(t)x(t) = PA_2(t)^{-1}A(t)x(t).$$

Thus by imposing conditions on $PA_2(t)^{-1}$ we can get that $PP_1(t)x(t)$ is asymptotically stable. The solution $x(t)$ can be decoupled into

$$x(t) = Qx(t) + PP_1(t)x(t) + PQ_1(t)x(t).$$

For the homogeneous system [9, eqs. (1.10)–(1.12)], $PQ_1(t)x(t) = 0$ and

$$Qx(t) = -(QP_1(t)A_2(t)^{-1}B(t) + (QQ_1(t))')PP_1(t)x(t).$$

Therefore, imposing conditions on the matrix that multiplies $PP_1(t)x(t)$, we can derive asymptotical stability for the solution.

Example 2 (see [1]). Consider the linear variable coefficient DAE (4.1) for

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \qquad B(t) = \begin{pmatrix} -\lambda + \frac{1}{2-t} & 0 & \lambda(2-t) \\ \frac{1-\lambda}{2-t} & 1 & 1-\lambda \\ -(t+2) & 4-t^2 & 0 \end{pmatrix}.$$

For $t \neq 2$, this DAE is tractable with index 2 [12]. In this case,

$$\operatorname{Img} \Pi(t) = \langle ((2-t), 1, -2)^t \rangle.$$

The subspace $V = \operatorname{Img} \Pi(t)$ is an admissible subspace for $\|A, \cdot\|_V$, and some computation gives

$$\mu_V[A, B(t)] = \frac{-\lambda t^2 + 4\lambda t + t - 5\lambda - 2}{t^2 - 4t + 5}.$$

Thus, if $\lambda > 0$, the solution is asymptotically stable. The solution of the DAE with consistent initial condition $x(3) = (-e^{-3\lambda}, e^{-3\lambda}, -2e^{-3\lambda})^t$ is

$$x(t) = ((2-t)e^{-\lambda t}, e^{-\lambda t}, -2e^{-\lambda t})^t.$$

Example 3 (see [9]). Consider the linear variable coefficient DAE (4.1) for

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \qquad B(t) = \begin{pmatrix} \lambda & -\beta & -1 \\ \beta\eta(1-\eta t) - \eta & \lambda & -\eta t \\ 1-\eta t & 1 & 0 \end{pmatrix},$$

where β, λ, and $\eta \in \mathbb{R}$ are constant. This DAE is tractable with index 2 [12]. In this case

$$\operatorname{Img} \Pi(t) = \langle (1, \eta t - 1, -\beta(\eta t - 1))^t \rangle.$$

The subspace $V = \operatorname{Img} \Pi(t)$ is an admissible subspace for $\|A, \cdot\|_V$, and some computation gives

$$\mu_V[A, B(t)] = -\lambda - \frac{\eta - \eta^2 t}{2 - 2\eta t + \eta^2 t^2}.$$

Thus, if $\lambda > 0$, the solution is asymptotically stable. The solution of the DAE with consistent initial condition at $t = 0$ is

$$x(t) = (x_1^0 e^{-\lambda t}, x_1^0(\eta t - 1)e^{-\lambda t}, -x_1^0 \beta(\eta t - 1)e^{-\lambda t})^t. \qquad \square$$

In [13], the asymptotical stability of equilibrium points is studied for tractable with index 2 quasi-linear DAEs. The DAE is linearized and a linear constant coefficient DAE is studied. Projectors P and P_1 can be chosen in such a way that $x(t) = PP_1x(t)$, and it is proved that $u = PP_1x(t)$ solves a linear system $u' = Mu$ with M a matrix such that

$$\langle Mz, z\rangle_S \leq -c\|z\|_S^2 \tag{4.7}$$

for any $z \in \operatorname{Img}(PP1)$ and some symmetric positive definite matrix S. Contractivity for $\|PP_1x\|_S$ is obtained, and hence the trivial solution is asymptotically stable. With the logarithmic norm concept defined in this paper, (4.7) is $\mu_V[I, -M] \leq 0$ for $V = \operatorname{Img}(PP_1)$ and a norm similar to the Euclidean norm (recall property (vi) in Proposition 3.3).

4.4. An index 3 example. We consider the constrained linear mechanical system

$$\begin{pmatrix} I & & \\ & M & \\ & & 0 \end{pmatrix} \begin{pmatrix} q \\ v \\ \lambda \end{pmatrix}' + \begin{pmatrix} 0 & -I & 0 \\ K & C & G^t \\ G & 0 & 0 \end{pmatrix} \begin{pmatrix} q \\ v \\ \lambda \end{pmatrix} = 0, \tag{4.8}$$

where M, K, C are symmetric positive semidefinite matrices, M is positive definite on the null-space of G, and G is constant and has full row rank. We can derive the last equation twice to obtain the linear system

$$\begin{pmatrix} M & G^t \\ G & 0 \end{pmatrix} \begin{pmatrix} v' \\ \lambda \end{pmatrix} = - \begin{pmatrix} Kq + Cv \\ 0 \end{pmatrix}. \tag{4.9}$$

Under the above assumptions, this system has unique solution, and thus (4.8) is an index 3 DAE. If F_{12} denotes the (1,2)-block of the inverse matrix in (4.9), it holds that $GF_{12} = I$ and the solution of (4.8) is in

$$V = \{(x, y, z) \in \mathbb{R}^{2n+m} | x, y \in \ker(G), z = -F_{12}^t [Kx + Cy]\}.$$

It can be checked that V is an admissible subspace. If the norms used are monotonic,

$$\left\| \begin{pmatrix} I & & \\ & M & \\ & & 0 \end{pmatrix}, \begin{pmatrix} I & & \\ & M & \\ & & 0 \end{pmatrix} - \Delta \begin{pmatrix} 0 & -I & 0 \\ K & C - M' & G^t \\ G & 0 & 0 \end{pmatrix} \right\|_V$$
$$= \max_{(x,y) \in V^*} \frac{\left\| \begin{pmatrix} x \\ My \end{pmatrix} - \Delta \begin{pmatrix} -y \\ (I - \pi)[Kx + Cy] - M'y \end{pmatrix} \right\|}{\left\| \begin{pmatrix} x \\ My \end{pmatrix} \right\|}$$

with $V^* = \{(x, y) \in \mathbb{R}^{2n} | x, y \in \ker(G)\}$ and π the projector $G^t F_{12}^t$. Thus the logarithmic norm used in Theorem 4.2 is

$$\mu_{V^*} \left[\begin{pmatrix} I & \\ & M \end{pmatrix}, \begin{pmatrix} 0 & -I \\ (I - \pi)K & (I - \pi)C - M' \end{pmatrix} \right]. \tag{4.10}$$

If we denote by F a matrix whose m linearly independent columns expand $\ker(G)$, (4.10) is

$$\mu_{V^{**}} \left[\begin{pmatrix} F & \\ & MF \end{pmatrix}, \begin{pmatrix} 0 & -F \\ (I - \pi)KF & [(I - \pi)C - M']F \end{pmatrix} \right] \tag{4.11}$$

with $V^{**} = \mathbb{R}^{2m}$.

For the Euclidean norm, by Proposition 3.4, (4.10) is

$$\lambda \max \left[\begin{pmatrix} F^t F & \\ & F^t M^2 F \end{pmatrix}, \begin{pmatrix} 0 & \tilde{K}^t \\ \tilde{K} & \tilde{C} \end{pmatrix} \right],$$

where $\tilde{K}$ and $\tilde{C}$ are given by

$$\tilde{K} = -\frac{1}{2}(-F^t F + F^t M (I - \pi) K F),$$
$$\tilde{C} = -\frac{1}{2}(F^t M [(I - \pi)C - M']F) + F^t [C(I - \pi^t) - M'] M F).$$

A simple computation gives that the eigenvalues are the roots of

$$\det(\lambda^2 F^t M^2 F + \lambda\tilde{C} - \tilde{K}^t (F^t F)^{-1}\tilde{K}) = 0,$$

and thus the symmetric quadratic eigenvalue problem

$$\lambda^2 F^t M^2 F x + \lambda\tilde{C}x - \tilde{K}^t (F^t F)^{-1}\tilde{K}x = 0$$

must be studied. If we multiply the left-hand side by x^* and denote $M_x = x^* F^t M^2 F x$, $C_x = x^*\tilde{C}x$ and $K_x = x^*\tilde{K}x$, we get the equation

$$\lambda^2 M_x + \lambda C_x - K_x = 0,$$

which implies that the eigenvalues cannot be strictly negative; for the particular case in which $\tilde{K} = 0$ and $\tilde{C}$ are a positive semidefinite matrix, the logarithmic norm in (4.11) is zero.

If M is regular, then any subspace is admissible, and an upper bound for (4.10) is the usual logarithmic norm

$$\mu\left[-\begin{pmatrix} 0 & -M^{-1} \\ (I-\pi)K & [(I-\pi)C - M']M^{-1} \end{pmatrix}\right]. \tag{4.12}$$

For the constant coefficient case it is well known [5] that if the spectral abscissa of a matrix A is strictly negative, then there is an elliptic norm $\|\cdot\|_X$ such that $\mu_X[A] < 0$. Thus we must study the spectrum of the matrix in (4.12), whose characteristic polynomial is

$$\det(\lambda^2 M^2 + \lambda(I-\pi)C + (I-\pi)K) = 0,$$

and therefore a nonsymmetric quadratic eigenvalue problem must be studied.

5. Other characterizations of the logarithmic norm. If we consider the subspace $V = \mathrm{Img}\,(\hat{A}^D\hat{A})$, a similar definition to (1.3) can be given for the logarithmic norms defined above.

PROPOSITION 5.1. *For $V = Img(\hat{A}^D\hat{A})$,*

$$\begin{aligned}\mu_V&[A,B]\\ &= \lim_{h\to 0^+}\frac{\ln\|A, Ae^{-\hat{A}^D\hat{B}h}\|_{Img(\hat{A}^D\hat{A})}}{h}\\ &= \lim_{h\to 0^+}\frac{\ln\|Ae^{-\hat{A}^D\hat{B}h}\hat{A}^D(cA+B)^{-1}\|_{Img(A\hat{A}^D\hat{A})}}{h}.\end{aligned}$$

Proof. It is immediate to prove that

$$\|A, Ae^{-\hat{A}^D\hat{B}h}\|_{\mathrm{Img}\,(\hat{A}^D\hat{A})} = \|Ae^{-\hat{A}^D\hat{B}h}\hat{A}^D(cA+B)^{-1}\|_{\mathrm{Img}\,(A\hat{A}^D\hat{A})}.$$

We also have

$$\begin{aligned}\ln&\|Ae^{-\hat{A}^D\hat{B}h}\hat{A}^D(cA+B)^{-1}\|_{\mathrm{Img}\,(A\hat{A}^D\hat{A})}\\ &= \|A(I_n - h\hat{A}^D\hat{B})\hat{A}^D(cA+B)^{-1}\|_{\mathrm{Img}\,(A\hat{A}^D\hat{A})} - 1 + \vartheta(h^2),\end{aligned}$$

and in the above expression

$$
\begin{aligned}
&\|A\hat{A}^D(cA+B)^{-1} - hA\hat{A}^D\hat{B}\hat{A}^D(cA+B)^{-1}\|_{\mathrm{Img}\,(A\hat{A}^D\hat{A})}\\
&\quad = \max_{y\in\mathbb{R}^n, A\hat{A}^D\hat{A}y\neq 0} \frac{A\hat{A}^D(cA+B)^{-1}A\hat{A}^D\hat{A}y - hA\hat{A}^D\hat{B}\hat{A}^D(cA+B)^{-1}A\hat{A}^D\hat{A}y}{A\hat{A}^D\hat{A}y}\\
&\quad = \max_{y\in\mathbb{R}^n, A\hat{A}^D\hat{A}y\neq 0} \frac{A\hat{A}^D\hat{A}y - hA\hat{A}^D\hat{B}\hat{A}^D\hat{A}y}{A\hat{A}^D\hat{A}y}\\
&\quad = \max_{x\in\mathrm{Img}\,(\hat{A}^D\hat{A}), Ax\neq 0} \frac{Ax - hA\hat{A}^D\hat{B}x}{Ax} = \|A, A - hA\hat{A}^D\hat{B}\|_V
\end{aligned}
$$

for $V = \mathrm{Img}\,(\hat{A}^D\hat{A})$, where we have used $\hat{A}^D\hat{A}\hat{A}^D = \hat{A}^D$. Now as $\hat{A}^D$, $\hat{A}$, and $\hat{B}^D$ commute, $A\hat{A}^D\hat{B} = B\hat{A}^D\hat{A}$, which implies $A\hat{A}^D\hat{B} = B\hat{A}^D\hat{A}$. If we take the limit $h \to 0$ in the above expressions, we get the desired result. □

Finally, we study whether there is a similar set (1.2) related to the exponentials that appear in the solution of the DAE and the logarithmic norm.

As the solution of the DAE is given by

$$x(t) = e^{-\hat{A}^D\hat{B}t}\hat{A}^D\hat{A}x_0 = e^{-\hat{A}^D\hat{B}t}\hat{A}^D(cA+B)^{-1}Ax_0,$$

by Theorem 4.2 we have

$$\|Ax(t)\| = \|Ae^{-\hat{A}^D\hat{B}t}\hat{A}^D(cA+B)^{-1}Ax_0\| \leq e^{\mu_V[A,B]t}\|Ax_0\|,$$

and thus

$$\|Ae^{-\hat{A}^D\hat{B}t}\hat{A}^D(cA+B)^{-1}\|_{\mathrm{Img}\,(A\hat{A}^D\hat{A})} \leq e^{\mu_V[A,B]t} \tag{5.1}$$

for any c such that $cA+B$ is regular. Lemma 5.2 below ensures that the above expression is independent from the value c chosen.

Lemma 5.2. *For any α and β such that $\alpha A+B$ and $\beta A+B$ are regular, we have*

$$\hat{A}_\alpha^D(\alpha A+B)^{-1} = \hat{A}_\beta^D(\beta A+B)^{-1}.$$

Proof. The proof follows the lines of Theorem 9.2.2 in [3]:

$$
\begin{aligned}
&[(\alpha A+B)^{-1}A]^D(\alpha A+B)^{-1} = [((\beta A+B)^{-1}(\alpha A+B))^{-1}\hat{A}_\beta]^D(\alpha A+B)^{-1}\\
&\ = [(\alpha\hat{A}_\beta+\hat{B}_\beta)^{-1}\hat{A}_\beta]^D(\alpha A+B)^{-1} = \hat{A}_\beta^D(\alpha\hat{A}_\beta+\hat{B}_\beta)(\alpha A+B)^{-1} = \hat{A}_\beta^D(\beta A+B)^{-1},
\end{aligned}
$$

where we have used that if A and B commute, then $(AB)^D = B^DA^D = A^DB^D$, and if A is regular, $A^D = A^{-1}$. □

Observe that if A is invertible (i.e., we have an ODE), then

$$\hat{A}_c^D(cA+B)^{-1} = [(cA+B)^{-1}A]^D(cA+B)^{-1} = A^{-1},$$

and thus

$$Ax(t) = Ae^{-A^{-1}Bt}x_0 = Ae^{-A^{-1}Bt}A^{-1}Ax_0$$

and (5.1) is $\|Ae^{-A^{-1}Bt}A^{-1}\| \leq e^{\mu_V[A,B]t}$ with $V = \mathbb{R}^n$ and coincides with (1.5) for $A = I_n$.

From (5.1) we construct the set

$$\begin{aligned} M &= \{\theta / \|Ae^{-\hat{A}^D\hat{B}t}\hat{A}^D(cA+B)^{-1}\|_{\mathrm{Img}\,(A\hat{A}^D\hat{A})} \leq e^{\theta t}, t \geq 0\} \\ &= \{\theta / \|A, Ae^{-\hat{A}^D\hat{B}t}\|_{\mathrm{Img}\,(\hat{A}^D\hat{A})} \leq e^{\theta t}, t \geq 0\} \end{aligned}$$

and study its relationship with $\mu_V[A,B]$. We use the expression given by property (vi) in Proposition 3.3.

THEOREM 5.3. *Consider an index k pencil* (A,B). *Let* $\mu_V[A,B]$ *be the smallest constant such that*

$$\langle Ax, -Bx\rangle \leq \mu_V[A,B]\langle Ax, Ax\rangle \qquad \textit{for all } x \in V$$

with $V = \mathit{Img}(\hat{A}^D\hat{A}) = \mathit{Img}(\hat{A}^k)$. *Then* $\mu_V[A,B]$ *is the smallest element of the set* M.

Proof. By (5.1) we have $\mu_V[A,B] \in M$. Now let $\theta \in M$; then for any $x_0 \in V$ we have

$$\begin{aligned} \|Ae^{-\hat{A}^D\hat{B}t}x_0\| &= \|Ae^{-\hat{A}^D\hat{B}t}\hat{A}^D(cA+B)^{-1}Ax_0\| \\ &\leq \|Ae^{-\hat{A}^D\hat{B}t}\hat{A}^D(cA+B)^{-1}\|\|Ax_0\| \leq e^{\theta t}\|Ax_0\|. \end{aligned}$$

Let the function $\phi(t)$ be defined by

$$\phi(t) = \langle Ae^{-\hat{A}^D\hat{B}t}x_0, Ae^{-\hat{A}^D\hat{B}t}x_0\rangle = \langle Ae^{-\hat{A}^D\hat{B}t}\hat{A}^D\hat{A}x_0, Ae^{-\hat{A}^D\hat{B}t}\hat{A}^D\hat{A}x_0\rangle$$

and the function $\psi(t) = e^{2\theta t}\|Ax_0\|^2$. We have

$$\phi(t) \leq \psi(t), \qquad t \geq 0.$$

Observe that $\phi(0) = \psi(0)$. If D_+ denotes the right differential operator,

$$D_+\phi(0) = \lim_{\tau \to 0} \frac{\phi(\tau) - \phi(0)}{\tau},$$

then from the above inequalities we get

$$D_+\phi(0) \leq D_+\psi(0).$$

A simple computation using the fact that $x(t) = e^{-\hat{A}^D\hat{B}t}\hat{A}^D\hat{A}x_0$ is the solution for the problem $Ax'(t) + Bx(t) = 0$ gives

$$D_+\phi(0) = 2\langle Ax_0, -Bx_0\rangle.$$

Thus

$$\langle Ax_0, -Bx_0\rangle = \frac{1}{2}D_+\phi(0) \leq \frac{1}{2}D_+\psi(0) = \theta\|Ax_0\|^2 = \theta\langle Ax_0, Ax_0\rangle.$$

This can be done for any $x_0 \in V$. Thus $\mu_V[A,B] \leq \theta$. □

REFERENCES

[1] U.M. Ascher and L.R. Petzold, *Projected implicit Runge–Kutta methods for differential-algebraic equations*, SIAM J. Numer. Anal., 28 (1991), pp. 1097–1120.

[2] K.E. Brenan, S.L. Campbell, and L.R. Petzold, *Numerical Solution of Initial Value Problems in Differential Algebraic Equations*, North–Holland, New York, 1989.

[3] S.L. Campbell and C.D. Meyer, Jr., *Generalized Inverses of Linear Transformations*, Dover, New York, 1991.

[4] C. Desoer and H. Haneda, *The measure of a matrix as a tool to analyze computer algorithms for circuit analysis*, IEEE Trans. Circuit Theory, 19 (1972), pp. 480–486.

[5] K. Dekker and J.G. Verwer, *Stability of Runge–Kutta Methods for Stiff Nonlinear Differential Equations*, North–Holland, Amsterdam, 1984.

[6] F.R. Gantmacher, *The Theory of Matrices, Vol.* II, Chelsea Publishing Company, New York, 1989.

[7] E. Griepentrog and R. März, *Differential algebraic equations and their numerical treatment*, Teubner-Texte Math. 88, Teubner, Leipzig, 1986.

[8] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations* II: *Stiff and Differential-Algebraic Problems*, Springer, Berlin, 1991.

[9] M. Hanke, E. Izquierdo Macana, and R. März, *On asymptotics in case of linear index*-2 *differential-algebraic equations*, SIAM J. Numer. Anal. 35 (1998), pp. 1326–1346.

[10] M. Hanke and R. März, *On the Asymptotics in the Case of Differential Algebraic Equations*, Talk given in Oberwolfach, October 1995.

[11] P. Lascaux and R. Théodor, *Analyse numérique matricielle appliquée a l'art de l'ingénieur*, Masson, Paris, 1986.

[12] R. März, *Index*-2 *differential algebraic equations*, Results Math., 15 (1989), pp. 148–171.

[13] R. März, *On Quasilinear Index* 2 *Differential Algebraic Equations*, preprint 269, Fachbereich Mathematik, Humboldt Universität zu Berlin, Berlin, Germany, 1990.

[14] A. Neumaier, *Global and realistic bounds for the solution of dissipative differential equations, Part* I: *Theory*, Computing, 52 (1994), pp. 315–336.

[15] P.J. Rabier and W.C. Rheinboldt, *Time-dependent linear DAEs with discontinuous inputs*, Linear Algebra Appl., 247 (1996), pp. 1–29.

[16] G.W. Stewart, *On the sensitivity of the eigenvalue problem* $Ax = \lambda Bx$, SIAM J. Numer. Anal., 9 (1972), pp. 669–686.

SIAM J. MATRIX ANAL. APPL.
Vol. 20, No. 3, pp. 667–699

A GEOMETRIC APPROACH TO PERTURBATION THEORY OF MATRICES AND MATRIX PENCILS. PART II: A STRATIFICATION-ENHANCED STAIRCASE ALGORITHM*

ALAN EDELMAN[†], ERIK ELMROTH[‡], AND BO KÅGSTRÖM[‡]

Abstract. Computing the Jordan form of a matrix or the Kronecker structure of a pencil is a well-known ill-posed problem. We propose that knowledge of the closure relations, i.e., the stratification, of the orbits and bundles of the various forms may be applied in the staircase algorithm. Here we discuss and complete the mathematical theory of these relationships and show how they may be applied to the staircase algorithm. This paper is a continuation of our Part I paper on versal deformations, but it may also be read independently.

Key words. Jordan canonical form, Kronecker canonical form, staircase algorithm, matrix pencils, closure relations, stratification, quivers

AMS subject classifications. 65F15, 15A21, 15A22

PII. S0895479896310184

Notation.

A	A square matrix of size $n \times n$. I or I_n is the identity matrix.
A^T	The transpose of A.
$\text{vec}(A)$	An ordered stack of the columns of a matrix A from left to right.
$\det(A)$	Determinant of A.
$\mathcal{N}(A)$	The column nullspace of A.
$\text{diag}(A_1, \ldots, A_b)$	A block diagonal matrix with diagonal blocks A_i.
$A \otimes B$	The Kronecker product of two matrices A and B whose (i, j)th block element is $a_{ij}B$.
$A_1 \boxplus A_2$	A canonical form whose Segre characteristics are the sum of those of A_1 and A_2, or equivalently whose Weyr characteristics are the union of those of A_1 and A_2.
$A - \lambda B$	A matrix pencil of size $m \times n$. Also denoted P.
$A^{(k)} - \lambda B^{(k)}$	Deflated pencil at step k of a staircase algorithm.
m_k, s_k	m_k = dimension of nullspace of $A^{(k)}$, $m_k - s_k$ = dimension of common nullspace of $A^{(k)}$ and $B^{(k)}$.
λ_i	Eigenvalue of A or $A - \lambda B$. Also μ_i, α, β, γ, and δ are used.
$J_j(\lambda_i)$	Jordan block of size $j \times j$ associated with λ_i.
N_j	Jordan block of size $j \times j$ associated with the infinite eigenvalue (sometimes denoted $J_j(\infty)$).
L_j	Singular block of right (column) minimal index of size $j \times (j+1)$.
L_j^T	Singular block of left (row) minimal index of size $(j+1) \times j$.

*Received by the editors October 1, 1996; accepted for publication (in revised form) by J. Varah May 17, 1998; published electronically March 23, 1999.
http://www.siam.org/journals/simax/20-3/31018.html

[†]Department of Mathematics, Room 2-380, Massachusetts Institute of Technology, Cambridge, MA 02139 (edelman@math.mit.edu). The work of this author was supported by NSF grant DMS-9501278 and an Alfred P. Sloan Foundation Research Fellowship.

[‡]Department of Computing Science, Umeå University, S-901 87 Umeå, Sweden (elmroth@cs.umu.se, bokg@cs.umu.se). The work of the authors was supported by Swedish Research Council for Engineering Sciences grant TFR 222-95-34.

$\alpha^i \alpha^j \beta^k$	Compact notation for $J_i(\alpha) \oplus J_j(\alpha) \oplus J_k(\beta)$.
$\text{nrk}(A - \lambda B)$	Normal rank of $A - \lambda B$.
$\mathcal{O}(A)$	The orbit of A, i.e., the set of matrices similar to A.
$\mathcal{O}(A - \lambda B)$	The orbit of $A - \lambda B$, i.e., the set of matrix pencils equivalent to $A - \lambda B$.
$\overline{\mathcal{O}}(\cdot)$	The closure of an orbit.
$\mathcal{B}(A)$	The bundle of A, i.e., the set of matrices with the Jordan structure of A, but with the eigenvalues unspecified.
$\mathcal{B}(A - \lambda B)$	The bundle of $A - \lambda B$, i.e., the set of matrix pencils with the Kronecker structure of $A - \lambda B$, but with the eigenvalues unspecified.
$\overline{\mathcal{B}}(\cdot)$	The closure of a bundle.
$\mathcal{S} \oplus \mathcal{T}$	Direct sum of subspaces $\mathcal{S}$ and $\mathcal{T}$ of $\mathbf{R}^n$.
$\dim(\mathcal{S})$	Dimension of subspace $\mathcal{S}$. $\dim(S)$ denotes dimension of subspace spanned by the columns of S.
$\text{cod}(\mathcal{S})$	Codimension is the dimension of the subspace complementary to $\mathcal{S}$.
$\kappa(A)$	$\kappa = (k_1, k_2, \ldots)$ is the integer partition representing the Segre characteristics for an eigenvalue λ_i of A.
$\mu(A)$	Integer partition representing the Weyr characteristics for an eigenvalue λ_i of A.
κ'	The conjugate partition of κ, e.g., $\mu(A) = \kappa(A)'$.
$\mathcal{J}_{\mu_i}$	$\mathcal{J}_{\mu_i} = (j_1, j_2, \ldots)$ is the integer partition representing the Weyr characteristics of $A - \lambda B$ for the eigenvalue μ_i.
$\mathcal{R}$	$\mathcal{R} = (r_0, r_1, \ldots)$ is the integer partition representing the right singular structure of $A - \lambda B$.
$\mathcal{L}$	$\mathcal{L} = (l_0, l_1, \ldots)$ is the integer partition representing the left singular structure of $A - \lambda B$.
$\langle P_1, P_2 \rangle_1$	Inner product for Kronecker structures defined as $\dim\{V : A_2 V B_1^T = B_2 V A_1^T\}$, where $P_1 = A_1 - \lambda B_1$ and $P_2 = A_2 - \lambda B_2$.
$\langle P_1, P_2 \rangle_2$	Inner product for Kronecker structures defined as $\dim\{(U, V) : U P_1 = P_2 V\}$ (also denoted $\langle P_1, P_2 \rangle$).
$\mathbf{A}_m$, $\tilde{\mathbf{A}}_m$	The quivers $\mathbf{A}_m$ and $\tilde{\mathbf{A}}_m$.
$(x - y)_+$	$max(x - y, 0)$.

1. Introduction. The determination of the Jordan form of a matrix A with multiple defective or derogatory eigenvalues is an ill-posed problem in the presence of roundoff error [26]. The same is true for the Kronecker form of a matrix pencil $A - \lambda B$. Therefore modern numerical software such as GUPTRI [15, 16] regularizes the problem by allowing a tolerance for rank decisions so as to find a matrix pencil near $A - \lambda B$ with an interesting Kronecker (or Jordan) structure. These algorithms are known to occasionally fail, thereby accidentally producing wrong nearby structures. Failure appears to occur when the matrix or pencil is close to a manifold of interesting structures of higher codimension [13]. Motivating examples arise in control theory, where linear control systems have been found that the staircase algorithm can decide are easily controllable, but in fact these systems were nearly uncontrollable (see, e.g., [6]). Because of these occasional failures, we propose to make use of the mathematical knowledge of the *stratification* of the Jordan and Kronecker structures in order to enhance the staircase algorithm. This stratification, in effect, shows which structures

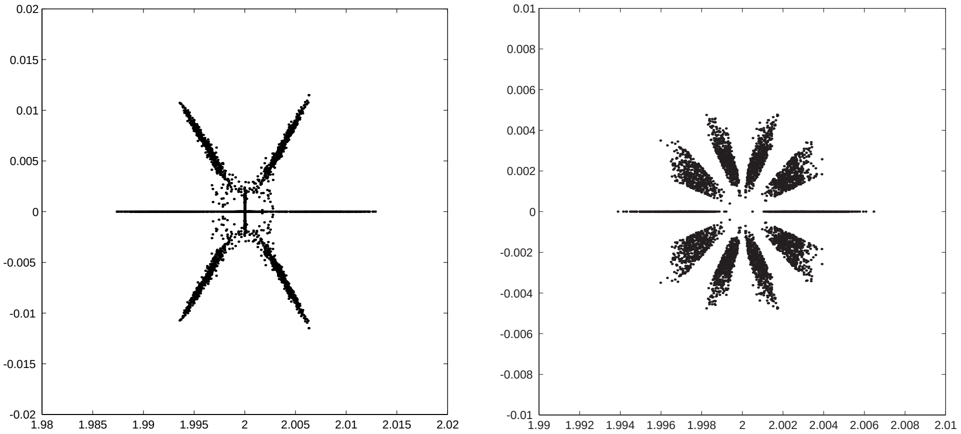


FIG. 1.1. *Perturbed eigenvalues of $J_3(2) \oplus J_2(2)$. Left: predominate structure. Right: nearby structure $J_5(2)$.*

are near other structures (in the sense of being in the closure) in the space of matrices. During the execution of a staircase algorithm the user or the program can be aware of the other nearby structures.

There are a number of ways to see the effects of nearby structures. Numerical experiments on random perturbations of 2×3 matrix pencils using GUPTRI are reported by Elmroth and Kågström [21, Table 3.1]. Assuming a fixed relative accuracy of the input data, the structures are studied as functions of the sizes of the perturbations. Even in the admittedly small 2×3 case, it becomes clear that there is an interesting combinatorial relationship among the possibilities, which we will investigate.

Consider the qualitative approach to the Jordan form proposed by Chaitin-Chatelin and Frayssé [10] using their example of the Ruhe matrix whose Jordan structure for eigenvalue $\lambda = 2$ is $J_3(2) \oplus J_2(2)$. The computed eigenvalues of roughly 1000 (real) normally distributed random perturbations of size $\sqrt{\epsilon}$ ($\epsilon = 2^{-52}$ is the usual IEEE double precision "epsilon") allow us to plot perturbed eigenvalues as in the picture to the left in Figure 1.1.

The six lines from the origin (four are fuzzy) and the smaller cross predict the Jordan structure $J_3(2) \oplus J_2(2)$. Besides this predominant structure, other structures may also appear [10]. We ran $50,000$ tests and filtered out those with roughly the predominant structure, thereby leaving around 1000 matrices whose perturbed eigenvalues appear to the right in Figure 1.1. This figure suggests that the structure $J_5(2)$ is somehow nearby. It turns out that $J_4(2) \oplus J_1(2)$ is also nearby but is much rarer. In 500,000 random tests none were found.

In addition to $J_4(2) \oplus J_1(2)$ and $J_5(2)$, one may wonder if we may have somehow missed other nearby structures. (We have not!) A more important question is if an algorithm such as GUPTRI or the qualitative approach suggest a certain Jordan structure, how can the user or a program be given the information of what structures are worth considering? The answer is that the staircase algorithm may be given expert knowledge of the combinatorial relationships among the various Jordan structures known as *strata.* In this paper we discuss these relationships and complete the mathematical theory needed not only for the Jordan eigenvalue case, but also for the Jordan bundle (see section 2.3) problem, the Kronecker structure problem, and the Kronecker bundle problem.

Before we introduce the form of the relationships among the strata, it is helpful to think about the information produced from one iteration to the next in the staircase algorithm. Suppose that we already have determined a subblock corresponding to a single eigenvalue. For simplicity suppose that the eigenvalue is zero so that we know from the beginning that we are trying to find the Jordan structure of a nilpotent matrix. It is well known that the most *generic* such matrix has the single Jordan block $J_n(0)$ as its Jordan form. Such matrices form a dense set within the set of nilpotent matrices. Therefore all we know at first is that the matrix is in the closure of the matrices similar to a single Jordan block. As the staircase algorithm to deduce the Jordan form proceeds, we gain more and more information about the matrix. What in fact happens at each point in the algorithm is that we learn that the matrix is in the closure of the set of matrices similar to some other Jordan form. Indeed one may view the algorithm as identifying a nested sequence of closures. If during the course of the algorithm the user is unhappy with any choice, he or she might wish to have the power to backtrack through the algorithm and be offered other choices. Alternatively at the end of the algorithm the user might wish to know what he or she has missed in what has been described eloquently by Hough [30, p. 270] (in the polynomial case) as the "thicket" of nearby structures.

Following Arnold, Gusein-Zade, and Varchenko [3, p. 41] (also see historical and rather technical discussions by Goresky and MacPherson [27, pp. 33, 37]), we say that we have a *stratified manifold* if it is the union of nonintersecting manifolds whose closure is the finite union of itself with strata of smaller dimensions (thereby defining stratified manifolds recursively). For matrices, the strata are orbits of similar matrices, or perhaps the union of such orbits (known as *bundles*). For pencils, the strata consist of strictly equivalent pencils (or bundles). The problem of stratification is to find the closure relations among the various orbits or bundles. These relations define a partial ordering on orbits or bundles. One structure covers another if its closure includes the closure of the other and there is no other structure in between.

While we are the first to propose the use of stratifications in an algorithm, some of the mathematical theory, at least for nilpotent matrices, goes back to 1961. It is known to the lie algebra community as the closure ordering [11] and to the algebra community as degenerations of modules over the $\tilde{\mathbf{A}}_0$ quiver (see section 5.2) [9]. Combinatorically it is trivial; it is the well-known dominance ordering on partitions. This is the case of relevance in an algorithm when the eigenvalues are well clustered so that we may shift all the blocks to be nilpotent.

When eigenvalues are not well clustered, we have to consider the bundle case as defined by Arnold [2]. We have not seen the closure relation for this case in the literature so we believe that our theorems are new. We show that testing the closure relation for bundles leads to an NP-complete problem; therefore it may be expensive to use a stratification-enhanced algorithm in the bundle case when more than only a few eigenvalues need to be clustered.

For orbits of matrix pencils, the closure ordering was published in a linear algebra journal by Pokrzywa [37] in 1986. A general unifying algebraic theory of degenerations has been obtained for quivers by several authors including Abeasis and Del Fra [1] and Bongartz [9]. Bongartz studied the pencil case in 1990, apparently unaware of Pokrzywa's work. In algebraic language, for a matrix or pencil orbit or bundle to cover another, it is necessary to have an extension. This condition is not sufficient; another new result in this paper is the necessary and sufficient conditions for covers. The Kronecker bundle case also seems to be new.

We summarize the main theorems in the box below. The closure decision problem is the question of how to test whether the closure of a given orbit or bundle is contained within the closure of another. The closely related covering relationship tells us which structures are covered by a given structure. The * indicates that to the best of our knowledge the results were either previously unknown or not strong enough for purposes of numerical computations. (For example, we extend Pokrzywa's results by providing both necessary and sufficient conditions for one Kronecker orbit to cover another.)

	Closure Decision Problem	Covering Relationship
Jordan Orbits	Theorem 2.2	Corollary 2.3
Jordan Bundles	Theorem 2.6, 2.7 *	Theorem 2.6 *
Kronecker Orbits	Theorem 3.1	Theorem 3.2 *
Kronecker Bundles	Theorem 3.3 *	Theorem 3.3 *

We begin our paper reviewing the combinatorics of integer partitions in section 2.1. We then discuss the nilpotent orbit case already known in section 2.2, providing our own simple proof in terms of the staircase form. Section 2.3 addresses the bundle case showing that the decision procedure is an NP-complete problem. Section 3 covers the more complicated Kronecker case. In section 3.1 we state the stratification theorems for both orbits and bundles. Examples are worked out in section 3.2. Some special cases that arise in applications are further explored in sections 3.3 and 3.4. The proofs of the theorems may be found in section 3.5. Section 4 provides some of the necessary details for using the theorems inside of the staircase algorithms yielding our so-called "stratification-enhanced staircase algorithm." Finally, section 5 covers some mathematical aspects of the problem and also provides a short exposition on the algebraic notation so as to narrow the gap between the numerical and algebraic communities.

2. Stratification of the Jordan canonical form.

When the user of a numerical algorithm is confident in the clustering of the eigenvalues, then the only question that may arise is, What is the Jordan structure corresponding to an individual eigenvalue? In that case, there is no loss of generality assuming the eigenvalue is 0; hence we are interested in the stratification of orbits of nilpotent matrices, the topic of section 2.2. When we are less confident in the clustering, we must consider the stratification of bundles as discussed in section 2.3. We start with some elementary combinatorial notions.

2.1. Integer partitions.

A *partition* κ of an integer n is a sequence of integers $(k_1, k_2, k_3, \ldots)$ such that $k_1 + k_2 + \cdots = n$ and $k_1 \geq k_2 \geq \cdots \geq 0$. We use standard vector operations and if m is a scalar we denote $(k_1 + m, k_2 + m, \ldots)$ as $\kappa + m$. The partitions of an integer form a *lattice* ([40] is a good undergraduate reference) under the *dominance ordering*: the dominance ordering on partitions (or integer sequences) specifies that $\kappa \geq \lambda$ if and only if $k_1 + \cdots + k_i \geq l_1 + \cdots + l_i$, for $i = 1, 2, \ldots$, and we say that $\kappa > \lambda$ if and only if $\kappa \geq \lambda$ and $\kappa \neq \lambda$. To say that we have a lattice means that for every pair of partitions one can find an upper bound and a lower bound, i.e., a partition that dominates the pair, and a partition that is dominated by the pair. In a lattice we say that κ *covers* λ if and only if $\kappa > \lambda$, but there is no μ such that $\kappa > \mu > \lambda$. In Figure 2.1, the covering relationship for all integer partitions of $n = 8$ is illustrated in a *Hasse diagram.* Notice that we have placed the most dominant partition at the bottom of the diagram, i.e., the diagram shows the reversed dominance ordering.

```
    11111111
        |
     2111111
        |
     221111
   /       \
22211     311111
 |  \       /
2222  32111
 |   /  |
3221    |
 |    41111
3311    |
 |   \  |
332    4211
 |   /  |
422     |
 |     5111
431     |
 |  \   |
44     521
 |  /      \
53         611
    \      /
       62
        |
       71
        |
        8
```

Fig. 2.1. *Covering relationship for all integer partitions of $n = 8$.*

One can easily illustrate the covering relationship (Figure 2.2) by placing n coins in a table with k_1 in the first column, k_2 in the second column, etc., corresponding to a Ferrer diagram. A partition κ_1 covers κ_2 if κ_2 may be obtained from κ_1 by moving a coin rightward *one* column, or downward *one* row, so long as the partition remains monotonic [11]. Or equivalently, κ_1 covers κ_2 if κ_1 may be obtained from κ_2 by moving a coin leftward one column, or upward one row, and keeping the monotonicity of the partition. We call these moves a *minimum rightward* and a *minimum leftward coin move*, respectively.

The final elementary notion that we need is the *conjugate partition*, which is the partition obtained by "transposing" the coins and is here denoted κ'. Figure 2.3 shows how (3,2,2,1) and (4,3,1) are conjugate partitions. Since transposing reverses the direction of coin moves, it is clear that $\kappa > \lambda$ if and only if $\kappa' < \lambda'$.

2.2. Stratification of nilpotent orbits. Consider two set of matrices; the first consists of the matrices similar to the nilpotent matrix A_1 and the second is the set similar to the nilpotent matrix A_2. When is the closure of the second set similar to that of the first? The closure is a mathematically precise way to discuss the vague idea of a Jordan form being "near" another Jordan form.

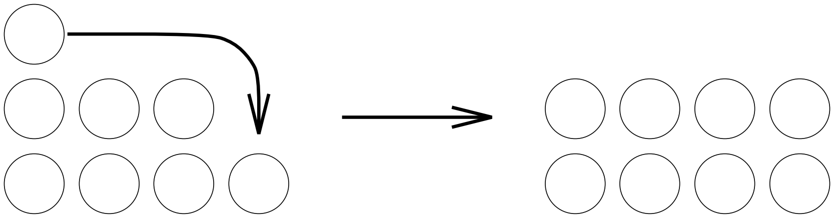


FIG. 2.2. *"Coin move" illustrates that* (3,2,2,1) *covers* (2,2,2,2).

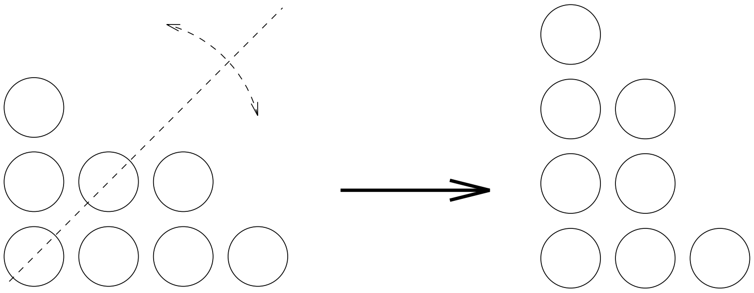


FIG. 2.3. *Transposing illustrates that* (3,2,2,1) *and* (4,3,1) *are conjugate partitions.*

Formally, in n^2-dimensional matrix space, consider the *orbits of matrices under similarity* transformations:

$$\mathcal{O}(A) \equiv \{SAS^{-1} : \det(S) \neq 0\}.$$

When is $\overline{\mathcal{O}}(A_1) \supseteq \overline{\mathcal{O}}(A_2)$? Trivially, if A_1 and A_2 are similar, then $\overline{\mathcal{O}}(A_1) = \overline{\mathcal{O}}(A_2)$. If $\overline{\mathcal{O}}(A_1) \supset \overline{\mathcal{O}}(A_2)$, then A_1 is "more generic" than A_2 or A_1 "degenerates" into A_2. In general, if an orbit $\mathcal{O}_1$ is more generic than an orbit $\mathcal{O}_2$, then dim $\mathcal{O}_1 >$ dim $\mathcal{O}_2$. However, this is not a sufficient condition for the closure of $\mathcal{O}_2$ to be a proper subset of the closure of $\mathcal{O}_1$.

Associated with every nilpotent matrix A is the partition $\kappa(A) = (k_1, k_2, k_3, \ldots)$ that lists in decreasing order the sizes of the Jordan blocks associated with A. The k_i are known as the *Segre characteristics*. The partition $\mu(A)$ that is conjugate to $\kappa(A)$ contains what are known as the *Weyr characteristics*. (See, for example, [13] or older textbooks for discussion.) The *staircase form* [26, 33] is obtained by applying a unitary similarity transformation that puts the nilpotent matrix A in the form illustrated in Figure 2.4 for a partition with four parts. Here, the $A_{i,i+1}$ blocks are of full column rank, the *'s are arbitrary, the "lower staircase" (below the $A_{i,i+1}$ blocks) consists of only zero entries, and m_i (= the number of principal vectors of grade i) are the Weyr characteristics. The Weyr characteristics are the block sizes that appear in the staircase form. The nilpotent A in Figure 2.4 has $m_1 - m_2, m_2 - m_3, m_3 - m_4$, and m_4 Jordan blocks $J_i(0)$ of size $i = 1, 2, 3$, and 4, respectively.

Strictly upper triangular matrices are associated with directed acyclic graphs by taking the *sparsity graph*, meaning that node i points to node j if the (i, j) entry is not 0. Conversely, one can start with a directed acyclic graph G and find the Jordan structure of a generic matrix with sparsity graph G by a procedure suggested by Gansner [22]: a *path* in G is a sequence of vertices connected by directed edges in the usual orientation. A *k-path* is a subset of vertices that can be partitioned into

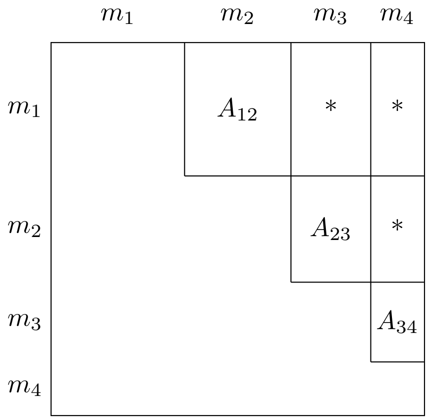


FIG. 2.4. *Example of a nilpotent A in staircase form.*

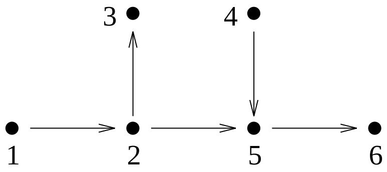


FIG. 2.5. *Digraph illustrating a $J_4 \oplus J_2$.*

k or fewer paths. Let s_1 denote the length of the longest path (1-path) in G, and inductively define s_j by letting $s_1+\cdots+s_j$ denote the size of a longest (most vertices) j-path. These s_i are the Segre characteristics associated with the digraph. The dual notion to longest k-paths is the *shortest k-truncated path.* Consider a partition of the vertices of the digraph into paths labeled $1,\ldots,l$. Let w_i denote the length of the ith path or k, whichever is smaller. The length of such a k-truncated path is $w_1+\cdots+w_l$; the smallest such sum is the length of the shortest k-truncated path. This gives a graph interpretation of the Weyr characteristics.

It would be a misconception that the size of the kth largest Jordan block can be found by looking at the longest path remaining after removing the longest $(k-1)$-path, since this may not be included in the longest k-path, as in the example in Figure 2.5. Here, the longest 1-path $(1, 2, 5, 6)$ of length four is not included in the longest 2-path $(1, 2, 3)$, $(4, 5, 6)$ of length six. (The Jordan normal form (JNF) of the generic matrix corresponding to the graph is $J_4 \oplus J_2$, not $J_4 \oplus J_1 \oplus J_1$.) By inspection we also see that this matrix cannot be put in staircase form by using only permutations. Even if the longest $(k-1)$-path is a subset of the longest k-path, it still may not necessarily be permuted into staircase form. The following is a graph characterization of the staircase form for nilpotent matrices.

THEOREM 2.1. *Denote the sources of a digraph as the* 1*-sources; deleting these sources we may denote the new sources as* 2*-sources, etc. A nilpotent matrix may be permuted to staircase form if and only if the k-sources form an antichain* (*i.e., no edges between them*) *and there is a matching between a subset of the* $(k-1)$*-sources and the k-sources.*

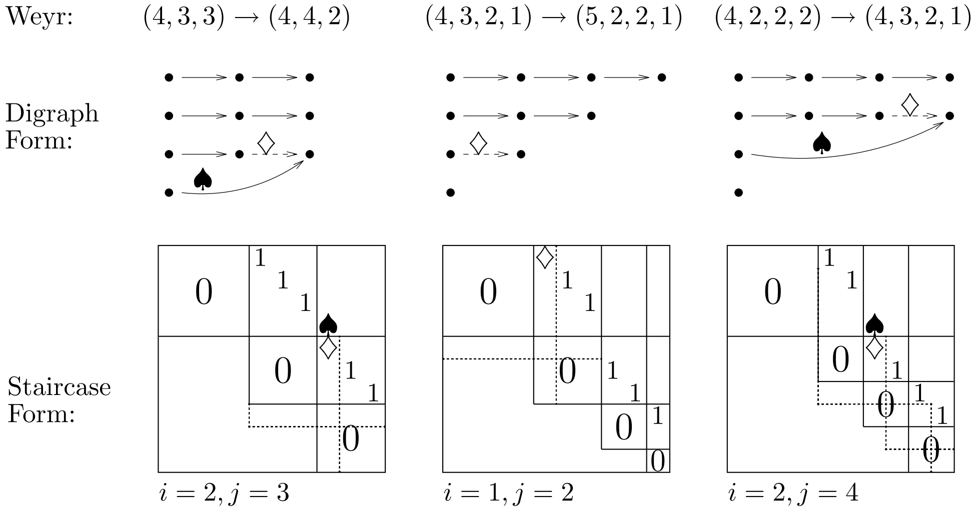


Fig. 2.6. *The ♢ goes from* 1 *to* 0*. The* ♠ = 1 *is introduced to preserve rank conditions. The solid lines indicate the original Jordan structure in staircase form. The dashed lines indicate the diagonal blocks of the final Jordan structure obtained when* ♢ = 0*.*

Proof. The nodes in the k-sources correspond to the kth diagonal block of the staircase form. The antichain corresponds to $A_{k,k} = 0$, and the matching condition is the full-rank condition on $A_{k,k+1}$. □

Our own version of the proof of the nilpotent stratification is quite short. Much of the proof may be understood by inspection of the staircase form or the digraph.

Theorem 2.2. *$\overline{\mathcal{O}}(A_1) \supseteq \overline{\mathcal{O}}(A_2)$ if and only if $\mu(A_2) \geq \mu(A_1)$, or equivalently $\kappa(A_1) \geq \kappa(A_2)$, where μ and κ denote the Weyr and Segre characteristics, respectively.*

Proof. We first remark that if $A_2 \in \overline{\mathcal{O}}(A_1)$, then $\mathcal{O}(A_2) \subseteq \overline{\mathcal{O}}(A_1)$ by taking similarity transformations. It then follows upon taking the closure of both sides that $\overline{\mathcal{O}}(A_2) \subseteq \overline{\mathcal{O}}(A_1)$. Therefore, to prove the "if" assertion, it suffices to assume that μ_2 covers μ_1 and to exhibit an A_1 and A_2 such that $\mu(A_i) = \mu_i$ for $i = 1, 2$ and $A_2 \in \overline{\mathcal{O}}(A_1)$.

For any μ_2 that covers μ_1, we have a "coin move" that decrements a column of size m_j and increments that of m_i, where $i < j$. The A_1 that we will pick with $\mu(A_1) = \mu_1$ has square identity matrices placed at the top of the superdiagonal blocks, except for $A_{j,j+1}$, where the identity matrix is placed at the bottom (if $j < j_{\max}$, the size of the largest Jordan block). We also place a 1 (denoted ♠ in Figure 2.6) in the first column of the super-super diagonal block $A_{i-1,i+1}$ in the last row (if $i > 1$). By zeroing the 1 in the first column of $A_{i,i+1}$ continuously, we effect a coin move that reduces m_j by one and increases m_i by one.

In Figure 2.6 we illustrate a few cases. The diamond (♢) moves continuously from 1 to 0. The spade (♠ = 1) is introduced if $i > 1$. Note the cascading effect when $j \neq i + 1$ in the equal blocks. The graph picture of the proof in terms of paths is also displayed in Figure 2.6. The deletion of the edge with the diamond corresponds to a coin move.

To prove the "only if" assertion, we assume that A_2 is a limit point of a continuous path Γ in $\mathcal{O}(A_1)$. We may continuously decompose every point on Γ into the staircase form corresponding to the partition $\mu(A_1)$. By the boundedness of the set of orthogonal matrices (compactness of the orthogonal group), although A_2, which is in the closure of $\mathcal{O}(A_1)$, may not be in $\mathcal{O}(A_1)$, it may also be put into the very same staircase form, although we may lose the full rank conditions on the $A_{i,i+1}$.

A rank one drop in one such $A_{i,i+1}$ corresponds to a leftward coin move. For example, if the rank of A_{23} in Figure 2.4 drops by one, then after an orthogonal similarity transformation, m_3 is decremented by one and m_2 (or possibly m_1 if $m_1 = m_2$) increments by one. (This assumes that the matrix consisting of A_{13} on top of A_{23} does not itself lose column rank, otherwise we have a more "long-distance" coin move obtainable by cascading short coin moves.) Therefore $\mu(A_2)$ dominates $\mu(A_1)$. □

From Theorem 2.2 and the definition of covering partitions we get the following obvious characterization for covering orbits of nilpotent matrices A_1 and A_2.

COROLLARY 2.3. *$\mathcal{O}(A_1)$ covers $\mathcal{O}(A_2)$ if and only if $\mu(A_2)$ can be obtained from $\mu(A_1)$ by a minimum leftward coin move.*

By reading the Hasse diagram in Figure 2.1 from top to bottom we get the stratification in terms of the Weyr characteristics. Reading the diagram from bottom to top, we get the closure hierarchy in terms of the Segre characteristics.

2.3. Stratification of Jordan bundles. Let $J_n(\alpha)$ denote a single $n \times n$ Jordan block with eigenvalue α. Our first example of a *bundle* as defined by Arnold [2, Sect. 5.3] is $\bigcup_\alpha \mathcal{O}(J_n(\alpha))$, the set of all matrices whose Jordan form consists of a single block. Notice that the bundle is the union of orbits. Here is the general definition. If two matrices have the same Jordan structure except that the distinct eigenvalues are different, we say they are in the same bundle. More precisely, let $w(\lambda_1), w(\lambda_2), \ldots, w(\lambda_p)$ be the Weyr characteristics of a matrix A with distinct eigenvalues $\lambda_1, \ldots, \lambda_p$. (Remember that $w(\lambda_i)$ is a partition of n_i, the algebraic multiplicity of the eigenvalue λ_i.) Another matrix B is said to be in the bundle $\mathcal{B}(A)$ if the distinct eigenvalues $\mu_1, \ldots, \mu_p$ of B may be ordered in such a way that the sequence of partitions $w(\mu_1), w(\mu_2), \ldots, w(\mu_p)$ is identical to that of A.

Let A_1 and A_2 be two nilpotent matrices. We define $A_1 \boxplus A_2$ to be the matrix in Jordan form (with the Jordan blocks ordered in decreasing order) whose Segre characteristics are the sums of those of A_1 and A_2, or equivalently whose Weyr characteristics are the union of those of A_1 and A_2. We point out that $A_1 \oplus A_2$ goes the other way: the Segre characteristics are the union, and the Weyr characteristics are the sum. For example, if $A_1 = J_3(0) \oplus J_1(0)$ and $A_2 = J_3(0) \oplus J_2(0) \oplus J_1(0)$, then $A_1 \oplus A_2 = 2J_3(0) \oplus J_2(0) \oplus 2J_1(0)$ and $A_1 \boxplus A_2 = J_6(0) \oplus J_3(0) \oplus J_1(0)$. We define an *extension* of A_1 and A_2 to be any matrix of the form

$$\begin{pmatrix} A_1 & X \\ 0 & A_2 \end{pmatrix},$$

where X could be any matrix of conforming size. For the example above, we get an extension with Jordan structure $A_1 \boxplus A_2$ by choosing x_{31} and x_{44} nonzero and all other elements in X (of size 4×6) as zero.

We have already pointed out that when $X = 0$, the Segre characteristics of the extension is the union of the Segre characteristics of A_1 and A_2. Therefore, an easy consequence of Theorem 2.2 is that $\mathcal{O}(A_1 \oplus A_2) \subset \overline{\mathcal{O}}(A_1 \boxplus A_2)$.

LEMMA 2.4. *The most generic extensions of A_1 and A_2 are in the orbit of $A_1 \boxplus A_2$.*

Proof. The easiest proof of this statement is obtained by assuming that A_1 and

A_2 are in Jordan form and by examining the longest k-paths in an extension. At most we can connect the longest path from the graph of A_1 to that of A_2, then the next longest of each, etc. Another proof may also be found in [25, Prop. 4.2.2]. □

If A is an extension of A_1 and A_2, then we have the obvious statement that

$$\mathcal{O}(A_1 \oplus A_2) \subseteq \overline{\mathcal{O}}(A) \subseteq \overline{\mathcal{O}}(A_1 \boxplus A_2). \tag{2.1}$$

The set of matrices A satisfying the relation (2.1) form a sublattice of the dominance ordering. Unfortunately, in general, this sublattice is not the set of extensions of A_1 and A_2. (An example is $A_1 = J_6(0)$ and $A_2 = J_4(0) \oplus J_2(0)$. In the lattice, $J_6(0) \oplus J_5(0) \oplus J_1(0)$ is between $A_1 \boxplus A_2$ and $A_1 \oplus A_2$, but it is not an extension of A_1 and A_2.) The characterization of the extensions (a further sublattice of this sublattice) is an open problem according to [25, p. 133], but it is not needed for our purposes.[1]

In the next lemma, we consider limit points of continuous paths $A(t)$ such that when $0 \le t < 1$, the path is contained in a bundle consisting of two distinct eigenvalues.

LEMMA 2.5. *Suppose $A(t)$ is similar to $A_1(t) \oplus A_2(t)$ for $0 \le t \le 1$, where $A_1(t) - \beta(t)I$ and $A_2(t) - \gamma(t)I$ are nilpotent, and for $0 \le t < 1$, $\beta(t) \neq \gamma(t)$, but when $t = 1$, $\beta(1) = \gamma(1) = 0$. In other words, $A_1(t)$ has the unique eigenvalue $\beta(t)$, $A_2(t)$ has the unique eigenvalue $\gamma(t)$, and these eigenvalues coalesce at 0 when $t = 1$. Then*

$$\overline{\mathcal{O}}(A(1)) \subseteq \overline{\mathcal{O}}(A_1(1) \boxplus A_2(1)).$$

Proof. We may find a continuous orthogonal similarity transformation $Q(t)$ such that

$$Q^T(t)A(t)Q(t) = \begin{pmatrix} A_1'(t) & X(t) \\ 0 & A_2'(t) \end{pmatrix},$$

and $A_1'(t)$ is similar to $A_1(t)$ and $A_2'(t)$ is similar to $A_2(t)$ for $0 \le t < 1$. Therefore, by Lemma 2.4,

$$\begin{pmatrix} A_1'(t) & X(t) \\ 0 & A_2'(t) \end{pmatrix} - \begin{pmatrix} \beta(t)I & \\ & \gamma(t)I \end{pmatrix} \in \overline{\mathcal{O}}(A_1(1) \boxplus A_2(1))$$

for $0 \le t < 1$. Letting $t \to 1$ shows that $A(1) \in \overline{\mathcal{O}}(A_1(1) \boxplus A_2(1))$, from which the result follows. □

The Jordan bundle stratification theorem follows below. Our results for the closure decision problem are also derived in [36, 17]. We believe the covering relationship is new.

THEOREM 2.6. *Suppose that we have two bundles $\mathcal{B}(A_1)$ and $\mathcal{B}(A_2)$ such that the former has at least as many distinct eigenvalues as the latter. Then $\overline{\mathcal{B}}(A_1) \supseteq \overline{\mathcal{B}}(A_2)$ if and only if it is possible to coalesce eigenvalues and apply the dominance ordering coin moves to the bundle defined by A_1 to reach that of A_2. Furthermore, a cover is obtained either by a minimal coin move* (*on the structure for one eigenvalue*) *or a generic extension* (*of the structures for two distinct eigenvalues assumed to coalesce*).

Proof. All that remains is to prove the minimality property of these covering relations. There are two natural quotient lattices of the bundle lattice. The first

[1] We prefer the use of the word "extension" rather than "completion" as used by [25] for consistency with the algebraic notion of extension of two modules.

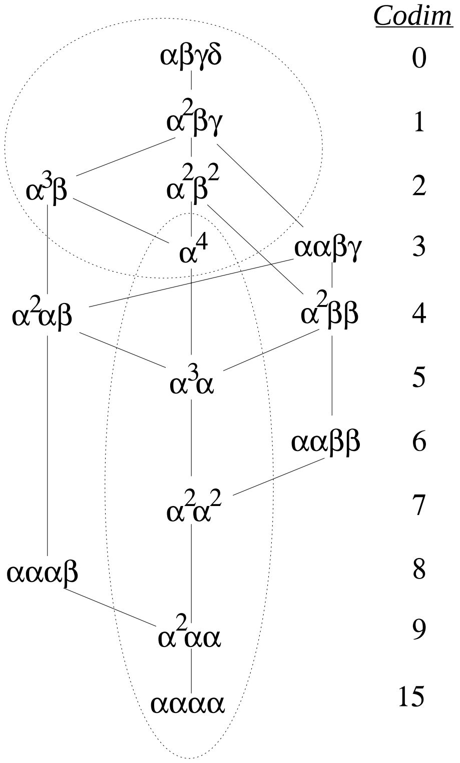


FIG. 2.7. *The bundle stratification for* 4×4 *matrices.*

counts the number of eigenvalues. The second is the partition of n obtained by taking the union of the partitions corresponding to all of the eigenvalues. Moving a coin in one partition moves down the first lattice but not the second. Coalescing two eigenvalues (forming the union of two partitions) moves down the second lattice but not the first. Therefore each operation cannot be obtained from the result of the other operation, so each is minimal. □

Figure 2.7 plots the bundle stratification for 4×4 matrices. Here, we use Arnold's compact notation for Jordan blocks: $\alpha^k \equiv J_k(\alpha)$. Circled in the top of the figure are those structures corresponding to coalescing eigenvalues. It is possible to gain an appreciation of the complicated manner of how these structures fit inside each other from the swallowtail diagram in Figure 2.8 [2], which shows the projection of these structures into three-dimensional space. The point α^4 is the swallowtail point, the curve $\alpha^3\beta$ are the two cusp edges coming out from the swallowtail, $\alpha^2\beta^2$ is the transversal intersection of the wings, and $\alpha^2\beta\gamma$ is the surface of the swallowtail. Everything outside the swallowtail is represented by $\alpha\beta\gamma\delta$. Any reliable numerical attempt to find the nearest structure of a certain particular form must somehow implicitly or explicitly deal with this kind of geometry. The circled structures in the lower part of the figure are those that correspond to the stratification of nilpotent orbits.

Figure 2.7 captures all distinct singularities of codimension 1 (α^2) and 2 (α^3, $\alpha^2\beta^2$) and two of the four distinct bundles of codimension 3 (α^4 and $\alpha\alpha$). The missing ones are $\alpha^3\beta^2$ and $\alpha^2\beta^2\gamma^2$.

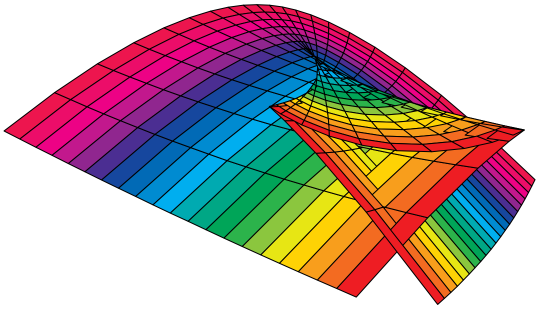


Fig. 2.8. *The swallowtail.*

Unfortunately, although the decision procedure for testing the closure relation for nilpotent matrices is trivial (all that is required is to test if one partition dominates the other), the corresponding procedure for bundles is an NP-complete problem. We speculate that this may capture some of the essence of the true difficulty associated with the clustering problem for perturbed eigenvalues. Another result that is slightly related was obtained by Gu [28], who showed that finding a well-conditioned similarity transformation to block-diagonalize a nonsymmetric matrix is an NP-hard problem.

Theorem 2.7. *Deciding whether a bundle is in the closure of another bundle is an NP-complete problem.*

Proof. Suppose that we have a matrix of dimension $n = km$ that has $3m$ distinct eigenvalues with multiplicities $k_1, \dots, k_{3m}$ with the property that $k/4 < k_i < k/2$ for each i. Consider the existence of a clustering of all of these eigenvalues into m triples so that the sum of the multiplicities of the three eigenvalues in each cluster is exactly k. This problem is the three-partition problem and is well known to be NP-complete [24]. □

The implication of Theorem 2.7 is that it is unlikely to find an efficient algorithm that solves all instances of the decision problem. However, it is still possible that there exist algorithms that can solve most practical cases efficiently.

3. Stratification of the Kronecker canonical form.

The notions of canonical form, orbits, bundles, and partitions extend to the matrix pencil case in a straightforward manner as follows. Any matrix pair (A, B), where A and B are $m \times n$ with real or complex entries, defines an *orbit (manifold) of strictly equivalent matrix pencils* in the $2mn$-dimensional space $\mathcal{P}$ of $m \times n$ pencils:

$$\mathcal{O}(A - \lambda B) = \{U^{-1}(A - \lambda B)V : \det(U)\det(V) \neq 0\}. \tag{3.1}$$

Let $P_1 = A_1 - \lambda B_1$, $P_2 = A_2 - \lambda B_2$ be two pencils (of possibly different sizes). A pair (U, V) satisfying $UP_1 = P_2V$ defines a homomorphism from P_1 to P_2 (see section 5.1). Let the dimension of such (U, V) be

$$\langle P_1, P_2 \rangle = \dim\{(U, V) : UP_1 = P_2V\}. \tag{3.2}$$

We also define $\mathrm{cod}(A - \lambda B)$ as the *codimension* of $\mathcal{O}(A - \lambda B)$, which is the dimension of the space complementary to the orbit, e.g., the dimension of the space normal to $\mathcal{O}(A - \lambda B)$ at the point $A - \lambda B$ [43, 13, 19]. It is known that $\mathrm{cod}(P) = \langle P, P \rangle - (m-n)^2$.

The *Kronecker canonical form* (KCF) (e.g., see [23]) for a pencil is the direct sum of the right singular, left singular, and regular structures, consisting of L_k blocks of size $k \times (k+1)$ for the *right singular structure* and L_k^T blocks for the *left singular structure*. The *regular structure* consists of Jordan blocks $J_k(\mu)$ corresponding to eigenvalue μ and N_k for the eigenvalue ∞. For short, we omit the word "singular" when clear from context. If $A - \lambda B$ is $m \times n$, where $m \neq n$, then for almost all A and B it will have the same KCF, depending only on m and n (the *generic case*; see section 3.3).

Define the *normal rank* of $A - \lambda B$ as

$$\operatorname{nrk}(A - \lambda B) = n - r_0 = m - l_0,$$

where r_0 and l_0 are the total number of right and left blocks, respectively, in the KCF of $A - \lambda B$.

As in the matrix case we consider two main problems in order to understand the stratification of matrix pencils. First, given two $m \times n$ matrix pencils $P_1 = A_1 - \lambda B_1$ and $P_2 = A_2 - \lambda B_2$ we want to have a procedure for deciding whether $\overline{\mathcal{O}}(P_1) \supseteq \overline{\mathcal{O}}(P_2)$. Second, we want to find a procedure for generating covering pencils, i.e., the closest neighboring orbits above or below in the closure hierarchy. These two problems are also investigated for bundles of pencils.

Because of the three basic structures (right, left, regular) in the Kronecker form, the complete characterization of all possible $m \times n$ Kronecker forms, their orbits, and bundles is somewhat more intricate than for the matrix case. We not only will need to define partitions for each eigenvalue, but in addition we will need partitions to define the right and left singular structures.

Let $\mathcal{R}(P)$ and $\mathcal{L}(P)$ denote the partitions for the right and left structures, respectively, of $P = A - \lambda B$ and let $\mathcal{J}_\mu(P)$ denote the partition for the Jordan structure corresponding to the eigenvalue μ (finite or infinite). When it is clear from context, we use the abbreviated notation $\mathcal{R}$, $\mathcal{L}$, and $\mathcal{J}_\mu$. The j_i's in a $\mathcal{J}$ partition are the Weyr characteristics for the eigenvalue μ, i.e., j_i is the number of $J_k(\mu)$ blocks of size $k \geq i$. Similarly, r_i of $\mathcal{R}$ (or l_i of $\mathcal{L}$) is the number of L_k (or L_k^T) blocks of size $k \geq i$.

We have shown in section 2 that the closure hierarchy of the set of $n \times n$ nilpotent matrices is completely determined by the dominance ordering of the integer n. Since the Kronecker structure for matrix pencils includes both Jordan blocks and singular blocks, a corresponding characterization involves integer sequences corresponding to each kind of block.

3.1. Stratification of Kronecker orbits and bundles. The decision procedure for the closure of orbits was derived by Pokrzywa [37] in 1986, was later reformulated by De Hoyos [12] in 1990, and was formulated differently by Bongartz [9] in 1990. In the following we give our formulation of De Hoyos's closure characterization. In section 5.2 we explain the algebraic and geometric connections between these approaches.

THEOREM 3.1 (see [37, 12, 9]). $\overline{\mathcal{O}}(P_1) \supseteq \overline{\mathcal{O}}(P_2)$ *if and only if the following relations hold:*

- $\mathcal{R}(P_1) + \operatorname{nrk}(P_1) \geq \mathcal{R}(P_2) + \operatorname{nrk}(P_2)$,
- $\mathcal{L}(P_1) + \operatorname{nrk}(P_1) \geq \mathcal{L}(P_2) + \operatorname{nrk}(P_2)$,
- $\mathcal{J}_{\mu_i}(P_1) + r_0(P_1) \leq \mathcal{J}_{\mu_i}(P_2) + r_0(P_2)$

for all $\mu_i \in \overline{\mathbf{C}}$, $i = 1, 2, \ldots$, *where* $\overline{\mathbf{C}} = \mathbf{C} \bigcup \{\infty\}$.

We remark that we could have used the more symmetric looking expression $\mathcal{J}_{\mu_i}(P_1) - \operatorname{nrk}(P_1) \leq \mathcal{J}_{\mu_i}(P_2) - \operatorname{nrk}(P_2)$ as item three in the theorem at the cost of having "negative coins." If $\operatorname{nrk}(P_1) = \operatorname{nrk}(P_2)$, we say that P_1 and P_2 are on the same level playing field, and the relations in Theorem 3.1 reduce to $\mathcal{R}(P_1) \geq \mathcal{R}(P_2)$, $\mathcal{L}(P_1) \geq \mathcal{L}(P_2)$, and $\mathcal{J}_{\mu_i}(P_1) \leq \mathcal{J}_{\mu_i}(P_2)$.

In a contemporary paper [7], Boley applies Theorem 3.1 and shows similar majorizing results of integer sequences associated with the KCF when a single row or column is appended to a matrix pencil. The application considered is adding a single input or output to a linear time-invariant dynamical system.

Now that we can test if one structure is more generic than another in the closure lattice, the next question we consider is the generation of all structures covered by a given pencil. Necessary conditions for two structures to be (closest) neighbors in the lattice are given in [37, 12, 9] and are used in an algorithm for computing the complete Kronecker structure hierarchy in [20]. We believe we are the first to give both necessary and sufficient conditions for neighbors in the lattice (which in addition gives an optimal algorithm for computing the complete hierarchy). We present these results in the form of coin moves associated with $\mathcal{R}$, $\mathcal{L}$, and $\mathcal{J}_{\mu_i}$ for different eigenvalues μ_i.

THEOREM 3.2. *$\mathcal{O}(P_1)$ covers $\mathcal{O}(P_2)$ if and only if P_2 can be obtained by applying one of the rules* (1)–(4) *to the integer partitions of P_1:*

(1) *Minimum rightward coin move in $\mathcal{R}$ (or $\mathcal{L}$).*

(2) *If the rightmost column in $\mathcal{R}$ (or $\mathcal{L}$) is one single coin, append that coin as a new rightmost column of some $\mathcal{J}_{\mu_i}$ (which may be empty initially).*

(3) *Minimum leftward coin move in any $\mathcal{J}_{\mu_i}$.*

(4) *Let k denote the total number of coins in all of the longest ($=$ lowest) rows from all of the $\mathcal{J}_{\mu_i}$. Remove these k coins, add one more coin to the set, and distribute $k+1$ coins to r_p, $p = 0, \ldots, t$ and l_q, $q = 0, \ldots, k-t-1$ such that at least all nonzero columns of $\mathcal{R}$ and $\mathcal{L}$ are given coins.*

Rules (1) *and* (2) *may not make coin moves that affect r_0 (or l_0).*

Notice that the restriction for rules (1) and (2) implies that the number of left and right blocks remains fixed, while rule (4) adds one new block of each kind. We also remark that rule (4) cannot be applied if the total number of nonzero columns in $\mathcal{R}$ and $\mathcal{L}$ is more than $k+1$. Rule (3) corresponds to the nilpotent case.

As in the matrix case we also consider stratification of bundles. Two pencils are in the same bundle if they have the same left and right singular structures and the same Jordan structure except that the distinct eigenvalues may be different. Of course, if $\mathcal{O}(P_1)$ covers $\mathcal{O}(P_2)$, then $\overline{\mathcal{B}}(P_1) \supset \overline{\mathcal{B}}(P_2)$, but the two bundles may not necessarily be covering, since there is a possibility of other structure changes from coalescing eigenvalues.

THEOREM 3.3. *$\mathcal{B}(P_1)$ covers $\mathcal{B}(P_2)$ if and only if P_2 can be obtained by applying one of the rules* (1)–(5) *to the integer partitions of P_1:*

(1) *Same as rule* 1 *in Theorem* 3.2.

(2) *Same as rule* 2 *in Theorem* 3.2, *except it is allowed only to start a new set corresponding to a new eigenvalue (i.e., no appending to nonempty sets).*

(3) *Same as rule* 3 *in Theorem* 3.2.

(4) *Same as rule* 4 *in Theorem* 3.2, *but apply only if there is just one eigenvalue in the KCF or if all eigenvalues have at least two Jordan blocks.*

(5) *Let any pair of eigenvalues coalesce, i.e., take the union of their sets of coins.*

The problem of deciding if the closure of the bundle of one pencil contains the bundle of another is NP-complete, just as for the matrix case (see Theorem 2.7). We postpone the proofs of Theorems 3.2 and 3.3 to section 3.5 and continue by illustrating the theorems with some examples. In section 4 an algorithmic implication of Theorem 3.2 is presented. We express different structure transitions in rules (1)–(4) in terms of the structure indices computed by a staircase algorithm.

$$P_1 = L_0 \oplus L_1 \oplus L_2 \oplus J_1(\mu_1) \oplus J_4(\mu_1) \oplus J_3(\mu_2) \oplus L_1^T \oplus L_3^T$$

	Structure blocks	Partition	Coins
Right	$L_0 \oplus L_1 \oplus L_2$	$\mathcal{R} = (3,2,1)$	○ ○ ○ ○ ○ ○
Left	$L_1^T \oplus L_3^T$	$\mathcal{L} = (2,2,1,1)$	○ ○ ○ ○ ○ ○
Regular eig=μ_1	$J_1(\mu_1) \oplus J_4(\mu_1)$	$\mathcal{J}_{\mu_1} = (2,1,1,1)$	○ ○ ○ ○ ○
eig=μ_2	$J_3(\mu_2)$	$\mathcal{J}_{\mu_2} = (1,1,1)$	○ ○ ○

FIG. 3.1. *Example Kronecker structure P_1 with corresponding partitions.*

3.2. Examples. To focus the reader's attention on how the covering theorem may be used in a numerical algorithm, we will examine two examples in detail. In the first example we take a particular pencil P_1 of size 17×18 and illustrate the application of some of the rules from Theorem 3.2. In the second example, we focus on a smaller case—the 2×3 pencils—and show the entire lattices for orbits and bundles. We also use this example to illustrate Theorem 3.1.

Our example pencil P_1 has KCF $L_0 \oplus L_1 \oplus L_2 \oplus J_1(\mu_1) \oplus J_4(\mu_1) \oplus J_3(\mu_2) \oplus L_1^T \oplus L_3^T$. We illustrate how the four rules in Theorem 3.2 can be used to find a pencil P_2 that is covered by P_1. The starting configuration for P_1 may be found in Figure 3.1 and the application of some of the rules is shown in Figure 3.2. We display KCF structures as both integer partitions and columns of "coins" ($\circ$). In Figure 3.2 we illustrate how each of the rules can be applied to P_1. (We append the notation a and b to rules (1) and (2) to denote application to the right or left structure, respectively.) The symbol $\bullet$ is used to denote a coin that will be moved in P_1's coin arrays or a coin that has been moved in P_2's coin arrays. Notice that some of the rules can also be applied to other combinations of blocks of P_1, i.e., the figure does not show all possible transitions that give a pencil P_2 that is covered by P_1. Each row of the figure shows how one of the rules may be applied to some of the blocks in the KCF. In the last column of Figure 3.2 (labeled "Block transitions") we record only the blocks that are involved in the application of the rule.

Recently, Elmroth and Kågström did a comprehensive study of the set of 2×3 pencils, including the stratification problem [21]. There are 9 possible bundles in this case. (From an algorithmic point of view, we may not want to bundle in the zero and infinite eigenvalue, in which case there are 20 bundles.) Fix the eigenvalues in the bundle to be γ and δ, with $\gamma \neq \delta$. The closure lattice corresponding to the orbits is shown in Figure 3.3. Following [21] we display the lattice with orbits (nodes) of the same codimension on the same horizontal level. The generic case (L_2) is at the highest level and the most degenerate pencil ($3L_0 \oplus 2L_0^T$ which is the 2×3 zero pencil) is at the lowest level. A pencil P_2 is in $\overline{\mathcal{O}}(P_1)$ if and only if there is a path from P_1 to P_2. The labels of the arcs correspond to which covering rules in Theorem 3.2 we have applied.

In Figure 3.4 we show the closure lattice corresponding to the bundles of 2×3 pencils. Here, a node represents the bundle consisting of all pencils with the displayed Kronecker structure, where the value of the eigenvalues γ and δ may vary, but $\gamma \neq \delta$. The arc labels show which of the rules in Theorem 3.3 we have applied. Since the

Rule	Partition	P_1	P_2	Block transitions
(1a)	$\mathcal{R}$	○ ○● ○○○	○ ○ ○○○●	$L_1 \oplus L_2 \longrightarrow L_0 \oplus L_3$
(1b)	$\mathcal{L}$	○● ○○○○	○ ○○○○●	$L_1^T \oplus L_3^T \longrightarrow L_0^T \oplus L_4^T$
(2a)	$\mathcal{R}$	○ ○○ ○○●	○ ○○ ○○	$L_2 \oplus J_4(\mu_1) \longrightarrow L_1 \oplus J_5(\mu_1)$
	$\mathcal{J}_{\mu_1}$	○ ○○○○	○ ○○○○●	
(2b)	$\mathcal{L}$	○○ ○○○●	○○ ○○○	$J_3(\mu_2) \oplus L_3^T \longrightarrow J_4(\mu_2) \oplus L_2^T$
	$\mathcal{J}_{\mu_2}$	○○○	○○○●	
(3)	$\mathcal{J}_{\mu_1}$	○ ○○○●	○● ○○○	$J_1(\mu_1) \oplus J_4(\mu_1) \longrightarrow J_2(\mu_1) \oplus J_3(\mu_1)$
(4)	$\mathcal{R}$	○ ○○ ○○○	● ○● ○○● ○○○	$J_4(\mu_1) \oplus J_3(\mu_2) \longrightarrow L_2 \oplus L_4^T$
	$\mathcal{L}$	○○ ○○○○	●● ○○●● ○○○○●	
	$\mathcal{J}_{\mu_1}$	○ ●●●●	○	
	$\mathcal{J}_{\mu_2}$	●●●		

FIG. 3.2. *Illustration of the covering rules in Theorem* 3.2 *starting with* P_1 *as in Figure* 3.1.

eigenvalues may vary in the bundles, the codimension for each Kronecker structure with regular part is one less for each eigenvalue compared to the orbit case (see Figure 3.3). For example, the codimension of $L_1 \oplus J_1(\gamma)$ is 2 in Figure 3.3 and 1 in Figure 3.4, since γ is an extra degree of freedom in the bundle case.

When comparing the closure hierarchies for bundles and orbits, we see that the bundle structure $L_0 \oplus J_2(\gamma)$ is found as the most generic one when γ and δ coalesce (rule (5)) in $L_0 \oplus J_1(\gamma) \oplus J_1(\delta)$, while these two structures are on different branches in the hierarchy for orbits (where the eigenvalues are assumed to be specified and therefore never may coalesce). This illustrates the restriction of rule (2) in Theorem 3.3.

Finally, we illustrate Theorem 3.1 by investigating the closure relations for the orbits of 2×3 pencils $P_1 = L_1 \oplus J_1(\gamma)$, $P_2 = L_0 \oplus J_2(\gamma)$, $P_3 = L_0 \oplus J_1(\gamma) \oplus J_1(\delta)$, and $P_4 = 2L_0 \oplus L_1^T$. From Table 3.1 we see that P_2, P_3, P_4 are in $\overline{\mathcal{O}}(P_1)$, P_4 is in P_1, P_2 and P_3, but neither P_2 or P_3 are in the closure of the other. To realize that P_1 and P_4 are closest neighbors to both P_2 and P_3 we have to apply Theorem 3.2 (see Figure 3.3).

3.3. Generic and full normal rank pencils.

The *generic Kronecker structure* for $A - \lambda B$ of size $m \times n$ with $d = n - m > 0$ is

$$\operatorname{diag}(L_\alpha, \dots, L_\alpha, L_{\alpha+1}, \dots, L_{\alpha+1}), \tag{3.3}$$

where $\alpha = \lfloor m/d \rfloor$, the total number of blocks is d, and the number of $L_{\alpha+1}$ blocks is m mod d (which is 0 when d divides m) [41, 13]. The same statement holds for $d = m - n > 0$ if we replace $L_\alpha, L_{\alpha+1}$ in (3.3) by $L_\alpha^T, L_{\alpha+1}^T$. Indeed, a generic

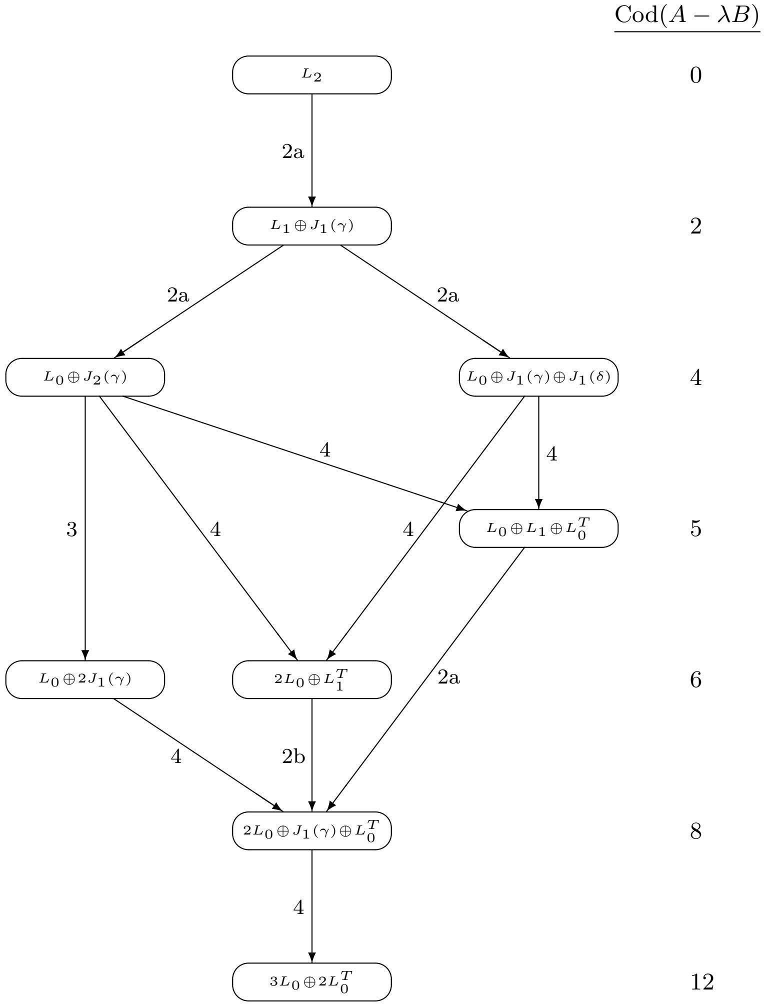


FIG. 3.3. *Closure hierarchy of orbits for the set of* 2×3 *pencils.*

nonsquare pencil $A - \lambda B$ is equivalent to one of the following two forms:

$$\begin{bmatrix} 0 & I_m \end{bmatrix} - \lambda \begin{bmatrix} I_m & 0 \end{bmatrix}, (m < n) \;\text{ and }\; \begin{bmatrix} 0 \\ I_n \end{bmatrix} - \lambda \begin{bmatrix} I_n \\ 0 \end{bmatrix}, (m > n).$$

Square pencils are generically regular, i.e., $\det(A - \lambda B) = 0$ if and only if λ is an eigenvalue. The generic singular pencils of size $n \times n$ have the Kronecker structures [43]:

$$\operatorname{diag}(L_j, L_{n-j-1}^T), \quad j = 0, \ldots, n-1.$$

All generic pencils have full normal rank, i.e., $\operatorname{nrk}(A - \lambda B) = \min(m, n)$. However, a pencil can have full normal rank without being generic. An $m \times n$ pencil with $m < n$ has full normal rank if and only if it has no L_k^T blocks. Similarly, if $m > n$ the pencil has full normal rank if and only if it has no L_k blocks. Finally, a square pencil $(m = n)$ has full normal rank if and only if it has no singular blocks.

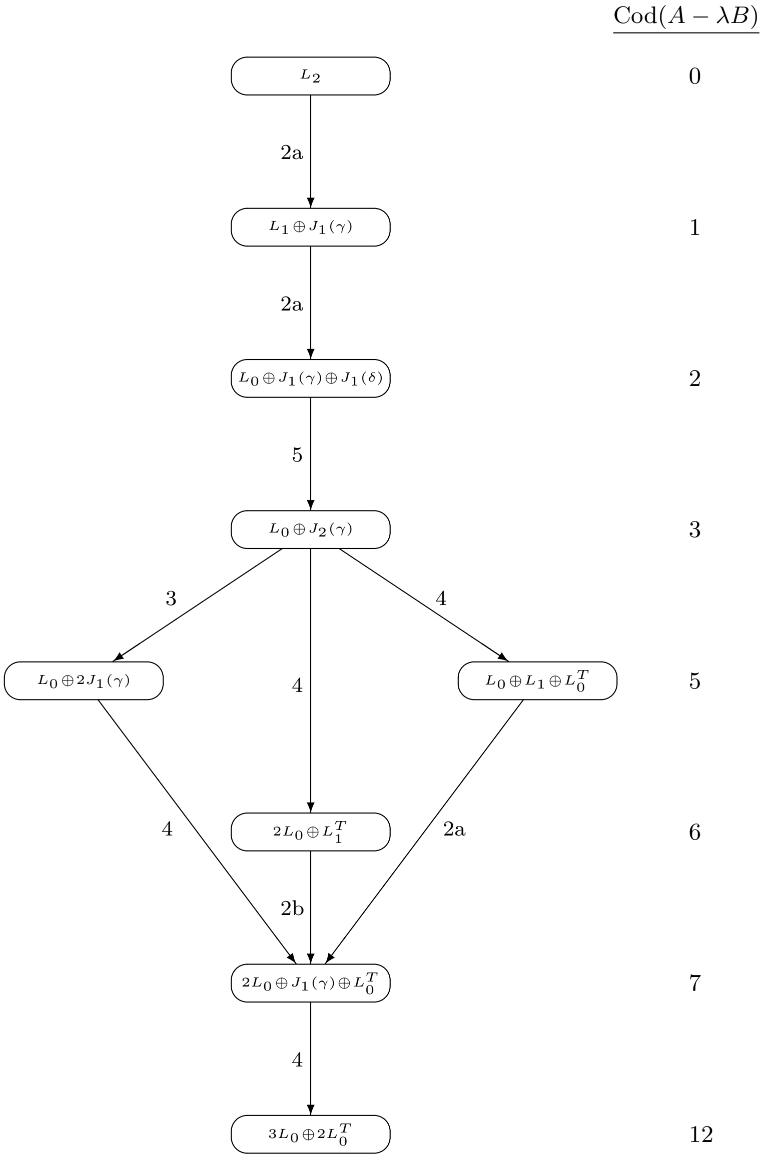

$\mathrm{Cod}(A-\lambda B)$
L_2
0
2a
$L_1 \oplus J_1(\gamma)$
1
2a
$L_0 \oplus J_1(\gamma) \oplus J_1(\delta)$
2
5
$L_0 \oplus J_2(\gamma)$
3
3
4
4
$L_0 \oplus 2J_1(\gamma)$
$L_0 \oplus L_1 \oplus L_0^T$
5
4
$2L_0 \oplus L_1^T$
2a
6
2b
$2L_0 \oplus J_1(\gamma) \oplus L_0^T$
7
4
$3L_0 \oplus 2L_0^T$
12

FIG. 3.4. *Closure hierarchy of bundles for the set of* 2×3 *pencils.*

Next we consider full normal rank pencils with only L_k or L_k^T blocks in their KCF. Let us assume that $m < n$; otherwise we can just perform the same process on the transposed pencil. The $\mathcal{R}$ partition corresponding to the generic pencil is $(r_0, r_1, \cdots, r_\alpha, r_{\alpha+1})$, where

$$r_0 = r_1 = \cdots = r_\alpha = d \quad \text{and } r_{\alpha+1} = m \bmod d$$

for $d = n - m$ and $\alpha = \lfloor m/d \rfloor$. Notice that $r_i = 0$ for $i > \alpha + 1$. Then we have the following corollary of Theorem 3.2.

COROLLARY 3.4. *The dominance ordering of* $\mathcal{R} = (r_0, r_1, \ldots, r_\alpha, r_{\alpha+1})$ *with* $r_0 = d = n - m > 0$ *kept fixed defines the closure hierarchy of the set of* $m \times n$ *matrix pencils with only* L_k *blocks.*

TABLE 3.1
Partitions for deciding closure relations between sample 2×3 *pencils.*

	$\mathcal{R}$	$\mathcal{L}$	$\mathcal{J}_\gamma$	$\mathcal{J}_\delta$	r_0	nrk$(\cdot)$
P_1	$(1,1,0,\ldots)$	$(0,\ldots)$	$(1,0,\ldots)$	$(0,\ldots)$	1	2
P_2	$(1,0,\ldots)$	$(0,\ldots)$	$(1,1,0,\ldots)$	$(0,\ldots)$	1	2
P_3	$(1,0,\ldots)$	$(0,\ldots)$	$(1,0,\ldots)$	$(1,0,\ldots)$	1	2
P_4	$(2,0,\ldots)$	$(1,1,0,\ldots)$	$(0,\ldots)$	$(0,\ldots)$	2	1

	$\mathcal{R}+\text{nrk}(\cdot)$	$\mathcal{L}+\text{nrk}(\cdot)$	$\mathcal{J}_\gamma+r_0$	$\mathcal{J}_\delta+r_0$
P_1	$(3,3,2,2,\ldots)$	$(2,2,2,2,\ldots)$	$(2,1,1,1,\ldots)$	$(1,1,1,1,\ldots)$
P_2	$(3,2,2,2,\ldots)$	$(2,2,2,2,\ldots)$	$(2,2,1,1,\ldots)$	$(1,1,1,1,\ldots)$
P_3	$(3,2,2,2,\ldots)$	$(2,2,2,2,\ldots)$	$(2,1,1,1,\ldots)$	$(2,1,1,1,\ldots)$
P_4	$(3,1,1,1,\ldots)$	$(2,2,1,1,\ldots)$	$(2,2,2,2,\ldots)$	$(2,2,2,2,\ldots)$

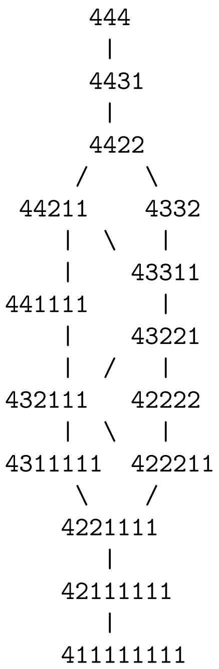


FIG. 3.5. *Covering relationship for* $\mathcal{R} = (4,4,4)$ *with* $r_0 = 4$ *kept fixed.*

We illustrate the corollary for the set of 8×12 pencils. The generic Kronecker structure is $4L_2$ ($d = 4, \alpha = 2, m \bmod d = 0$) which gives $\mathcal{R} = (4,4,4)$. The example is chosen so that parts of the dominance ordering for $n = 8$ can be reused (see Figure 2.1). The dominance ordering for $\mathcal{R}$ with r_0 kept fixed is displayed in Figure 3.5. Note the diagram is not symmetric since we are using only a sublattice from Figure 2.1.

A similar result holds for the set of $m \times n$ matrix pencils with a regular part of fixed Jordan structure besides L_j blocks in the KCF. The $\mathcal{R}$ partition corresponding to the most generic pencil with a $k \times k$ regular part is similar to (3.3), where now $d = m - n - k$ and $\alpha = \lfloor (m-k)/d \rfloor$. Since we can write P_1 as $P_1^{(1)} \oplus P_1^{(2)}$, where $P_1^{(1)}$ and $P_1^{(2)}$ correspond to the regular and right singular parts, respectively, we can apply Corollary 3.4 to $P_1^{(2)}$ which defines the $\mathcal{R}$ partition.

Corollary 3.5. *The dominance ordering of* $\mathcal{R} = (r_0, r_1, \ldots, r_\alpha, r_{\alpha+1})$ *with* $r_0 = n - m - k > 0$ *kept fixed defines the closure hierarchy of the set of* $m \times n$ *matrix pencils with a fixed Jordan structure* (*regular part*) *of size* $k \times k$ $(0 \leq k \leq m)$ *and* L_j *blocks only in the KCF.*

These sets of matrix pencils have important applications in linear systems theory. Let $A - \lambda B = [F, C] - \lambda[E, 0]$, where $E\dot{x}(t) = Fx(t) + Cu(t)$ is a linear system with m states and p controls. Then Corollary 3.4 gives the closure hierarchy for the the set of completely controllable systems with m states and p controls. Similarly, Corollary 3.5 gives the closure hierarchy for the sets of linear systems with k uncontrollable modes with fixed Jordan structure.

3.4. Interesting nearness problems. One motivation for our work on versal deformations (see Part I [19]) and stratifications of orbits and bundles (the present paper) was to get an improved understanding of important nearness problems, such as

- closest degenerate (nongeneric) pencil of a generic $A - \lambda B$,
- closest matrix pencil with a specified Kronecker structure,
- closest neighbors (covering pencils) of a given $A - \lambda B$.

Several of these problems have interesting applications in linear system theory. For example, if we add the restriction that the closest degenerate pencil to a generic $m \times n$ pencil (with $m < n$) should have a regular part, then the first problem corresponds to finding the closest uncontrollable system (see also section 3.3).

The closure hierarchy lattice gives one kind of answer to these nearness problems for equivalence orbits of pencils, where we use the codimension instead of the Euclidean distance. We make use of the theorem for covering pencils to prove the following statement.

Theorem 3.6. *Let* $m < n$. *Then the* $m \times n$ *pencils with regular part form a codimension* $n - m + 1$ *stratified submanifold of all pencils equal to the closure of the orbit of*

$$J_1(\gamma) \oplus A_1 - \lambda B_1, \quad \text{where} \quad A_1 - \lambda B_1 = \operatorname{diag}(L_{\bar{\alpha}}, \ldots, L_{\bar{\alpha}}, L_{\bar{\alpha}+1}, \ldots, L_{\bar{\alpha}+1}), \tag{3.4}$$

$\bar{\alpha} = \lfloor (m-1)/d \rfloor$, $d = n - m > 0$, *is the total number of* L *blocks, and the number of* $L_{\bar{\alpha}+1}$ *blocks is* $(m-1) \bmod d$. *Therefore, the nearest pencil with a regular part to a generic pencil is generically of the form* (3.4).

Proof. First, we can apply rule (1a) of Theorem 3.2 only until we get a single largest L_k block. Then we apply rule (2a) with the implication that $L_k \oplus \emptyset \rightarrow L_{k-1} \oplus J_1(\gamma)$. The codimension of $J_1(\gamma) \oplus A_1 - \lambda B_1$ is $n - m + 1$. □

Notably, $A_1 - \lambda B_1$ is the generic pencil of size $(m-1) \times (n-1)$ and $J_1(\gamma)$ is a Jordan block of size one with an unspecified eigenvalue γ.

We illustrate Theorem 3.6 for 7×12 pencils. In Figure 3.6 we show a sublattice corresponding to equivalence orbits of codimension ≤ 8. We see that $L_0 \oplus 2L_1 \oplus 3L_2$ has the least nonzero codimension $(= 3)$ (Corollary 3.4), and $4L_1 \oplus L_2 \oplus J_1(\gamma)$ has codimension 6 and is the most generic pencil with a regular part (Theorem 3.6). We remark that no structures with a regular part and codimension less than 8 can be found by following the "empty" arc from $L_0 \oplus 2L_1 \oplus L_2 \oplus L_3$, since application of rule (2a) here gives $L_0 \oplus 2L_1 \oplus L_2 \oplus L_3$ and $2L_0 \oplus 2L_2 \oplus L_3$, with codimensions 8 and 9, respectively.

Given a generic $m \times n$ pencil we can apply Theorem 4.2 in our Part I paper [19] to get lower bounds on the distance (measured in the Frobenius norm) to the closest nongeneric pencils of codimension $n - m + 1$.

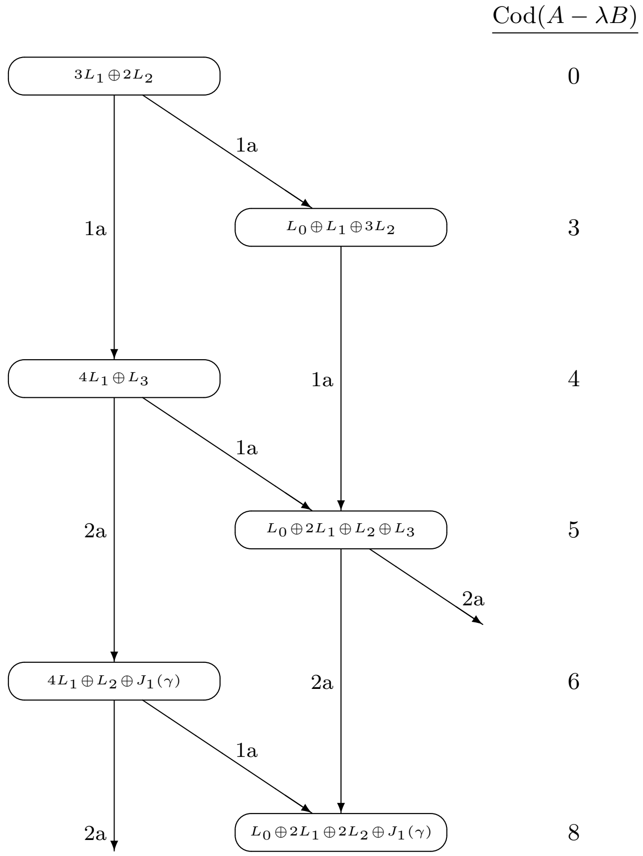

Cod$(A-\lambda B)$
$3L_1 \oplus 2L_2$
0
1a
1a
$L_0 \oplus L_1 \oplus 3L_2$
3
$4L_1 \oplus L_3$
1a
4
1a
2a
$L_0 \oplus 2L_1 \oplus L_2 \oplus L_3$
5
2a
$4L_1 \oplus L_2 \oplus J_1(\gamma)$
2a
6
1a
2a
$L_0 \oplus 2L_1 \oplus 2L_2 \oplus J_1(\gamma)$
8

FIG. 3.6. *Illustration of Theorem* 3.6. *The closure hierarchy shows that* $4L_1 \oplus L_2 \oplus J_1(\gamma)$ *is the most generic* 7×12 *structure with one eigenvalue.*

3.5. Proofs of Theorems 3.2 and 3.3. Given a pencil P_1, Pokrzywa's Lemma 5 [37] on necessary conditions for covering pencils exhibits a pencil P_2 such that $\overline{\mathcal{O}}(P_1) \supset \overline{\mathcal{O}}(P_2)$, but there may still exist another pencil P such that $\overline{\mathcal{O}}(P_1) \supset \overline{\mathcal{O}}(P) \supset \overline{\mathcal{O}}(P_2)$, i.e., Pokrzywa's rules do not guarantee a cover. The pencils found by his lemma, however, include all pencils P_2 that are covered by P_1; therefore the lemma includes the necessary conditions for covering pencils. We prove Theorem 3.2 by showing that we have included all possible restrictions to the rules without missing any links. For each rule in the proof we denote Pokrzywa's corresponding rule [37, Lem. 5] as (P1), (P2), etc., and consider them in terms of coin moves.

Proof of Theorem 3.2.

(1) (P1) is a rightward coin move in $\mathcal{R}$ (or $\mathcal{L}$) that is consistent with the columns being monotonically ordered. The restriction to a *minimum* rightward coin move precludes the possibility of reaching the same state with another sequence of moves.

(2) (P2) is a coin move from $\mathcal{R}$ (or $\mathcal{L}$) to $\mathcal{J}_{\mu_i}$ for any μ_i. In Theorem 3.2, the reason for moving the *rightmost* coin in $\mathcal{R}$ (or $\mathcal{L}$) is that if we move a coin c that is not the rightmost one, then the same partition can be found by a series of moves (move the coin c using rule (1) until it is in the rightmost

position and then move it to the $\mathcal{J}_{\mu_i}$ partition). This shows that we can generate the same partition with other partitions in between. Similarly, the reason for placing the coin in the *rightmost* position of $\mathcal{J}_{\mu_i}$ is that a partition obtained by placing it in any other position can be obtained by first placing it in the rightmost position and then applying rule (3).

(3) (P3) is a leftward coin move in $\mathcal{J}_{\mu_i}$ that is consistent with the columns being monotonically ordered. Our restriction to a *minimum* leftward coin move is obvious.

(4) (P4) is the removal of a row of coins from one or several eigenvalues. Add one more coin to these $\hat{k}$ coins and distribute all $\hat{k}+1$ coins from left to right in $\mathcal{R}$ and $\mathcal{L}$. There are several restrictions. By picking the *longest row* from each of the $\mathcal{J}_{\mu_i}$, we can always move coins back again (using rule (2)) in order to find the partitions we would have found by removing a shorter row. This is also the reason why we pick the longest row for *all eigenvalues*; if we want to, we can bring them back for all but one eigenvalue. The restriction that each column of $\mathcal{R}$ and $\mathcal{L}$ must have one coin each is required, since otherwise we could obtain the same sequence by first moving more coins to the $\mathcal{J}$ partitions using rule (2) and then applying rule (4).

It is obvious that these four rules are now minimal under these restrictions. □

The proof of Theorem 3.3 is based on the fact that if one orbit is covered by another, then also its bundle is in the closure of the bundle of the other, but the covering relation may be overruled by the possibility of eigenvalues coalescing.

Proof of Theorem 3.3. Rule (5) follows from the matrix case, and since the bundles are unions of orbits, all the covering relations for orbits are also valid for the bundle case, as long as the same operation cannot be performed in more than one step using rule (5). Since it is obvious that the operations corresponding to rules (1) and (3) cannot be done in more steps using rule (5), we only have to focus on rules (2) and (4).

Rule (2) in Theorem 3.2 allows a coin to be moved to any eigenvalue μ_i, but for bundles this can be done in two steps: move the coin to a new eigenvalue μ_j and apply rule (5) on μ_i and μ_j, i.e., append the coin from μ_j to the longest row of coins for μ_i (now $= \mu_j$).

The next question is whether rule (4) can be replaced with a sequence involving rule (5). If there is only one eigenvalue, rule (5) is not applicable, and if each eigenvalue has at least two Jordan blocks, then rule (5) must necessarily decrement the number of distinct eigenvalues while rule (4) does not. Otherwise if an eigenvalue has only one Jordan block, one may apply rule (5) before rule (4) to achieve the same result as a single application of rule (4). □

4. Empowering the staircase algorithm with stratifications.

The staircase algorithm is a powerful tool for computing the Kronecker structure of an $m \times n$ pencil $A - \lambda B$ [4, 8, 32, 35, 34, 41, 44]. The reduction of $A - \lambda B$ into *generalized Schur form* requires several applications of the staircase algorithm. Typically, the first application extracts the right structure and the Jordan structure of the zero eigenvalue using a finite sequence of orthogonal (unitary) equivalence transformations. This decomposition is called the *RZ*-staircase form (*RZ* for "right-zero").

In step k $(= 1, 2, \ldots)$ of the first phase, GUPTRI [15, 16] computes the *RZ* form by determining $m_k =$ dimension of the column nullspace of $A^{(k)}$ and $m_k - s_k =$ dimension of the common column nullspace of $A^{(k)}$ and $B^{(k)}$. Here, $A^{(1)} = A$ and $B^{(1)} = B$ and $(A^{(k)}, B^{(k)})$ for $k > 1$ correspond to the deflated matrix pair obtained after the equivalence transformation in step $k-1$. The structure indices (RZ-indices) display the Kronecker structure as follows:

$$m_k - s_k = \text{ number of } L_{k-1} \text{ blocks,} \quad s_k - m_{k+1} = \text{ number of } J_k(0) \text{ blocks.}$$

The Jordan structure associated with a finite but nonzero eigenvalue is obtained by applying the RZ-staircase algorithm to a shifted pencil. One way to find the left structure is to apply the same algorithm to the transposed pencil. Another way is to directly determine the sizes of the corresponding row nullspaces as done in the GUPTRI algorithm. Then by working on $B - \mu A$ we get the LI-staircase form and now $m_k - s_k$ and $s_k - m_{k+1}$, which are the number of L^T_{k-1} and N_k blocks, respectively, define the LI-indices. Applying the RZ-staircase algorithm to $B - \mu A$ gives the right structure and the Jordan structure of ∞ (RI-indices). Similarly, we can get the left structure and the Jordan structure of zero (LZ-indices) by applying the LI-staircase algorithm to $A - \lambda B$. All combinations (RZ, RI, LI, and LZ) are possible.

Knowing the RZ- and LI-indices we can easily extract the integer sequences (partitions) $\mathcal{R}, \mathcal{L}$, and $\mathcal{J}_{\mu_i}$ or the corresponding staircase indices (R, L, Z, and I). For example, the $\mathcal{R}$ and $\mathcal{J}_0$ partitions are obtained from the RZ-indices as

$$r_{i-1} = \sum_{k=i}^{\infty} m_k - s_k \quad \text{and} \quad j_i = \sum_{k=i}^{\infty} s_k - m_{k+1}.$$

4.1. Modified staircases and covering pencils. In the following we make an algorithmic application of Theorem 3.2. Given a pencil P_1 and the staircase indices defining its Kronecker structure, we want to find *all* pencils P_2 covered by P_1. We can therefore, for example, give the user a selection of choices or perhaps choose one automatically. The four rules in Theorem 3.2 correspond to different structure transitions. Rules (3) and (1) correspond to finding a covering orbit for nilpotent matrices (Corollary 2.3) and full normal rank pencils with only L (or L^T) blocks (Corollary 3.4), respectively. Rule (2) is applicable only if there exists a unique largest L_j (or L^T_j) block in P_1. Then the size of that block is decreased by one, while the size of the largest Jordan block (possibly 0×0) for one eigenvalue is increased by one. Rule (4) replaces the regular structure consisting of the largest Jordan blocks associated with all eigenvalues in P_1 by a generic square singular part. The new L and L^T blocks in P_2 must be at least as large as the corresponding largest singular blocks in P_1.

We assume that the left and right structures are captured only in the structure indices corresponding to one eigenvalue (possibly different for left and right structures). The remaining structure indices capture only Jordan structures, e.g., the RZ-indices associated with an eigenvalue μ_i reduces to Z-indices ($m_k = s_k$).

We propose that a nice user interface based on Algorithm 4.1 should be available to the user.

Algorithm 4.1. For all valid coin moves from column j to column k in the appropriate integer sequences ($\mathcal{R}, \mathcal{L}$, and $\mathcal{J}_{\mu_i}$) of rules (1)–(3) in Theorem 3.2, the staircase indices are adjusted as follows. (Remember that $\mathcal{R}$ and $\mathcal{L}$ start counting columns from 0 but $\mathcal{J}_{\mu_i}$ starts from 1.) We use a right arrow (→) to show how one block in the KCF is transferred to another.

(1a) Let m_k and s_k be RZ-indices (or RI-indices). Then $m_{j+1} := m_{j+1} - 1$, $s_j := s_j - 1$ ($L_j \rightarrow L_{j-1}$), $m_{k+1} := m_{k+1} + 1$, and $s_k := s_k + 1$ ($L_{k-1} \rightarrow L_k$).

(1b) Same as item (1a), where now m_k and s_k are LI-indices (or LZ-indices).

(2a) Let m_k and s_k be the RZ-indices (or RI-indices) associated with an eigenvalue $\mu \in \overline{\mathbf{C}}$ (RI-indices if $\mu = \infty$). Then $m_{j+1} := m_{j+1} - 1$, $s_j := s_j - 1$ ($L_j \to L_{j-1}$), $m_k := m_k + 1$, and $s_k := s_k + 1$ ($J_{k-1}(\mu) \to J_k(\mu)$).

(2b) Same as item (2a), where now m_k and s_k are the LZ-indices (or LI-indices) associated with $\mu \in \overline{\mathbf{C}}$.

(3) Let m_k and s_k be any of the staircase indices (RZ, RI, LI, or LZ) associated with $\mu \in \overline{\mathbf{C}}$. Then $m_j := m_j - 1$, $s_j := s_j - 1$ ($J_j(\mu) \to J_{j-1}(\mu)$), $m_k := m_k + 1$, $s_k := s_k + 1$ ($J_{k-1}(\mu) \to J_k(\mu)$).

For all valid coin moves in the appropriate integer sequences ($\mathcal{R}, \mathcal{L}$, and $\mathcal{J}_{\mu_i}$) of rule 4 in Theorem 3.2, the staircase indices are adjusted as follows.

(4) Each valid coin move is defined by k and t in the theorem and the following operations replace the selected $k \times k$ regular part with a generic square singular pencil ($J_{k_1}(\mu_i) \oplus J_{k_2}(\mu_1) \oplus \cdots \oplus J_{k_p}(\mu_p) \to L_t \oplus L_{k-t-1}^T$). Here, k_i is the size of the largest Jordan block of μ_i and $k = \sum k_i$.

- Repeat for all p eigenvalues μ_i: Let m_k and s_k be the RZ-indices (or RI-indices) of $\mu_i \in \overline{\mathbf{C}}$. Then $s_\iota := s_\iota - 1$, $m_\iota := m_\iota - 1$ for $\iota = 1, \dots, k_i$.
- Update RZ-indices (or RI-indices) with respect to new L_t block: $m_\iota := m_\iota + 1$ for $\iota = 1, \dots, t+1$, $s_\iota := s_\iota + 1$ for $\iota = 1, \dots, t$ (if $t > 0$).
- Update LZ-indices (or LI-indices) with respect to new L_{k-t-1}^T block: $m_\iota := m_\iota + 1$ for $\iota = 1, \dots, k-t$, $s_\iota := s_\iota + 1$ for $\iota = 1, \dots, k-t-1$ (if $k-t-1 > 0$).

Each valid application of any of the rules (1)–(4) results in a pencil P_2 such that $\mathcal{O}(P_1)$ covers $\mathcal{O}(P_2)$ and each P_2 is on a different branch in the closure lattice. Starting with a generic pencil, repeated applications of the stratification-enhanced algorithm will give us the complete closure hierarchy. Given m_k and s_k corresponding to any of the staircase forms of an arbitrary $m \times n$ pencil up to a certain point, the most generic object is the one where the remaining $\tilde{m} \times \tilde{n}$ pencil is generic. In other words, based on the information obtained up to this point, we know that the pencil is in the closure of the orbit corresponding to this situation. The values of $\tilde{m}$ and $\tilde{n}$ determine the KCF and the staircase indices of the remaining generic pencil (see section 3.3). Any application of the rules (1)–(4) will result in a less generic pencil. Note that the number of different orbits in the closure lattice is exponentially growing as a function of the problem size (m, n), so the algorithm is recursively applied only a few steps if m and n are large.

Similarly, given P_1 it is possible to characterize a pencil P_2 such that $\mathcal{O}(P_2)$ covers $\mathcal{O}(P_1)$. Of course, this will impose different prerequisites on and changes of P_1's structure indices. Moreover, algorithmic applications of Theorem 3.3 for finding covering bundles can be formulated similarly. The algorithmic details are omitted here.

For an illustration of the stratification-enhanced staircase algorithm we return to the examples in Figure 3.2. In Figure 4.1 we display staircase form transitions corresponding to different blocks of P_1 with KCF $L_0 \oplus L_1 \oplus L_2 \oplus J_1(\mu_1) \oplus J_4(\mu_1) \oplus J_3(\mu_2) \oplus L_1^T \oplus L_3^T$. Each of the six cases illustrates the structure index changes imposed by one of the rules of the algorithm. The diagonal blocks in the staircase forms of size $s_k \times m_k$ reveal the Kronecker structures of P_1 and P_2. In order to keep the picture small and clear, we display only the blocks that are directly affected by the transition from P_1 to P_2. These staircase forms correspond to the coin moves illustrated in Figure 3.2. Following the notation from Figure 2.6 the diamond ($\diamondsuit$ for A and $\diamondsuit_\lambda$ for B) is used to denote a matrix entry that the algorithm forces to zero and thereby

Blocks of $P_1 \longrightarrow$ Blocks of P_2

$$\text{(1a)}\qquad \left[\begin{array}{cc|cc|c} -\lambda & & 1 & & \\ & \diamondsuit_\lambda & \spadesuit_\lambda & 1 & \\ \hline & & & -\lambda & 1 \end{array}\right] \longrightarrow \left[\begin{array}{c|c|c|c} -\lambda & 1 & & \\ \hline 0 & \spadesuit_\lambda & 1 & \\ \hline & & -\lambda & 1 \end{array}\right]$$

$$\text{(1b)}\qquad \left[\begin{array}{c|c|cc} -\lambda & & & \\ \hline 1 & -\lambda & & \\ \hline & 1 & -\lambda & \\ & & \spadesuit & -\lambda \\ \hline & & \diamondsuit & \\ & & & 1 \end{array}\right] \longrightarrow \left[\begin{array}{c|c|c|c} -\lambda & & & \\ \hline 1 & -\lambda & & \\ \hline & 1 & -\lambda & \\ \hline & & \spadesuit & -\lambda \\ \hline & & 0 & \\ & & & 1 \end{array}\right]$$

$$\text{(2a)}\qquad \left[\begin{array}{cc|cc|cc|c} -\lambda & & 1 & & & & \\ & -\lambda & & 1 & & & \\ \hline & & & -\lambda & 1 & & \\ & & \diamondsuit_\lambda & & \spadesuit_\lambda & 1 & \\ \hline & & & & & -\lambda & 1 \\ \hline & & & & & & -\lambda \end{array}\right] \longrightarrow \left[\begin{array}{cc|cc|c|c|c} -\lambda & & 1 & & & & \\ & -\lambda & & 1 & & & \\ \hline & & & -\lambda & 1 & & \\ \hline & & 0 & & \spadesuit_\lambda & 1 & \\ \hline & & & & & -\lambda & 1 \\ \hline & & & & & & -\lambda \end{array}\right]$$

$$\text{(2b)}\qquad \left[\begin{array}{cc|cc|cc} \spadesuit & -\lambda & & & & \\ \hline \diamondsuit & & -\lambda & & & \\ & 1 & & -\lambda & & \\ \hline & & 1 & & -\lambda & \\ & & & 1 & & -\lambda \\ \hline & & & & 1 & \\ & & & & & 1 \end{array}\right] \longrightarrow \left[\begin{array}{c|c|cc|cc} \spadesuit & -\lambda & & & & \\ \hline 0 & & -\lambda & & & \\ & 1 & & -\lambda & & \\ \hline & & 1 & & -\lambda & \\ & & & 1 & & -\lambda \\ \hline & & & & 1 & \\ & & & & & 1 \end{array}\right]$$

$$\text{(3)}\qquad \left[\begin{array}{cc|c|c|c} -\lambda & & 1 & & \\ & -\lambda & & \spadesuit & \\ \hline & & -\lambda & \diamondsuit & \\ \hline & & & -\lambda & 1 \\ \hline & & & & -\lambda \end{array}\right] \longrightarrow \left[\begin{array}{cc|cc|c} -\lambda & & 1 & & \\ & -\lambda & & \spadesuit & \\ \hline & & -\lambda & 0 & \\ & & & -\lambda & 1 \\ \hline & & & & -\lambda \end{array}\right]$$

$$\text{(4)}\qquad \left[\begin{array}{c|c|c|c|ccc} -\lambda & 1 & & & & & \\ \hline & -\lambda & 1 & & & & \\ \hline & & \diamondsuit_\lambda & 1 & & & \\ \hline & & & -\lambda & 1 & & \\ \hline & & & & \tilde{\mu}_2-\lambda & 1 & \\ & & & & & \tilde{\mu}_2-\lambda & 1 \\ & & & & & & \tilde{\mu}_2-\lambda \end{array}\right] \longrightarrow \left[\begin{array}{c|c|c|cccc} -\lambda & 1 & & & & & \\ \hline & -\lambda & 1 & & & & \\ \hline & & 0 & 1 & & & \\ & & & -\lambda & 1 & & \\ & & & & \tilde{\mu}_2-\lambda & 1 & \\ & & & & & \tilde{\mu}_2-\lambda & 1 \\ & & & & & & \tilde{\mu}_2-\lambda \end{array}\right]$$

FIG. 4.1. *RZ- and LI-staircase forms of P_1 and P_2 displayed in Figure* 3.2.

changes the computed Kronecker structure from the KCF of P_1 to the KCF of P_2. The spade ($\spadesuit$ for A and $\spadesuit_\lambda$ for B) in P_1 is a nonzero entry that if not existing can be introduced by an equivalence transformation. If not introduced (i.e., the spade does not appear in P_2), then the KCF of P_2 is even less generic, which corresponds to further applications of the stratification-enhanced algorithm.

For cases (1a), (2a), (3), and (4) Figure 4.1 shows the RZ-staircase form of P_1 and P_2. Similarly, the LI-staircase form is displayed for cases (1b) and (2b). For completeness, we could have included the LI-staircase form for case (4) as well.

Rule	Indices	P_1						P_2						
(1a)	RZ	k	1	2	3	4	5	k	1	2	3	4	5	
		m_k	5	3	2	1	0	m_k	5	2	2	2	0	
		s_k	4	2	1	1	0	s_k	3	2	2	1	0	
(1b)	LI	k	1	2	3	4	5	k	1	2	3	4	5	6
		m_k	3	3	2	1	0	m_k	3	2	2	1	1	0
		s_k	3	2	2	0	0	s_k	2	2	2	1	0	0
(2a)	RZ	See (1a)						k	1	2	3	4	5	6
								m_k	5	3	1	1	1	0
								s_k	4	1	1	1	1	0
(2b)	LI	See (1b)						k	1	2	3	4	5	
								m_k	3	3	2	1	0	
								s_k	3	2	1	1	0	
(3)	RZ	See (1a)						k	1	2	3	4		
								m_k	5	4	2	0		
								s_k	4	3	1	0		
(4)	RZ	See (1a)						k	1	2	3	4		
								m_k	5	3	2	0		
								s_k	4	2	0	0		
	LI	See (1b)						k	1	2	3	4	5	6
								m_k	3	3	2	2	1	0
								s_k	3	2	2	1	0	0

FIG. 4.2. *Examples of structure index changes in the stratification-enhanced staircase algorithm.*

However, we see immediately that the last diagonal block of P_2 is an L_4^T block. Here $\tilde{\mu}_2$ corresponds to an eigenvalue $\mu_2 - \mu_1$ of the shifted pencil $A - (\lambda + \mu_1)B$. The sizes $(m_k \times s_k)$ of the diagonal blocks of these staircase forms reveal the changes in the local structure indices that result after applying rules (1)–(4) in the algorithm. For the RZ-staircase forms we start at the top left corner when listing m_k and s_k for $k = 1, \ldots$. Similarly, for the LI-staircase forms we start at the bottom right corner. More interesting are the corresponding changes in the global structure indices (RZ, LI, etc.) for these examples, which are displayed in Figure 4.2. Without loss of generality we have chosen $\mu_1 = 0$ and $\mu_2 = \infty$ in Figure 4.2.

Applying the GUPTRI algorithm in finite precision arithmetic means that all rank decisions for computing the structure indices are made with respect to a user supplied tolerance which reflects the relative accuracy of the data [15, 16]. Assuming a fixed accuracy of the input data it is possible to increase or decrease the tolerance for rank decisions such that a less generic or a more generic pencil, respectively, is computed. Alternatively, given a Kronecker structure computed by the staircase algorithm we can impose a more degenerate Kronecker structure by applying any of the applicable structure index changes. A stratification-enhanced GUPTRI algorithm can deliver an upper bound on the size of the distance from the pencil P_1 we started with to the pencil P_2 we imposed such that $\mathcal{O}(P_1)$ covers $\mathcal{O}(P_2)$. The other way around, we can start with a pencil P_1 and construct a more generic pencil P_2 by adding perturbations (whose sizes are dependent on the rank decision tolerance) such that $\mathcal{O}(P_2)$ covers $\mathcal{O}(P_1)$.

In infinite precision arithmetic we can always go upwards in the closure hierarchy by adding arbitrary small perturbations. This is normally not the case for going downwards in the hierarchy. See [21] for computable normwise bounds of the smallest perturbations for going downward (or upward) in the closure hierarchy of the set of 2×3 pencils.

5. The abstract algebra of matrix pencils.

We give a high-level view of the proofs of Theorem 3.1, mentioning a few new conjectures that we have solved. The Pokrzywa proof uses ordinary linear algebra notation; the algebraic notation by Bongartz would be foreign to many numerical readers. Moreover, we provide a quick introduction to narrow the gap between the algebra and numerical communities. The elegance in the algebraic approach is the unifying treatment obtained for the Jordan, Kronecker, echelon, and many other forms.

5.1. Closure relations, inner products, and codimension counts.

While the covering relationships might be thought of as combinatorial, the closure relations, which are statements about geometry, are derived mainly by algebraic techniques. As discussed earlier, there have been two independent derivations [37, 9] of the closure hierarchy. We suspected that the two very different looking proofs might be somehow "isomorphic," particularly since both count the dimension of the space of solutions to test homogeneous equations.

To be more precise, consider the two *inner products on Kronecker structures* for pencils $P_1 = A_1 - \lambda B_1$ and $P_2 = A_2 - \lambda B_2$,

$$\langle P_1, P_2 \rangle_1 = \dim\{V : A_2 V B_1^T = B_2 V A_1^T\},$$

defined by Pokrzywa [37], and the already defined (before without subscript (3.2))

$$\langle P_1, P_2 \rangle_2 = \dim\{(U, V) : U P_1 = P_2 V\}.$$

The inner product $\langle P_1, P_2 \rangle_2$ is used by Bongartz, who generalizes techniques of Abeasis and Del Fra [1], and by Riedtmann [38], who studied the dimension of the linear space of homomorphisms ($\dim \operatorname{Hom}(P_1, P_2) = \langle P_1, P_2 \rangle_2$ in our case) between path algebra modules (see section 5.2).

Using Kronecker products we can express the inner products as

$$\langle P_1, P_2 \rangle_1 = \dim\{x : T_1 x = 0\} \quad \text{and} \quad \langle P_1, P_2 \rangle_2 = \dim\{y : T_2 y = 0\},$$

where

$$T_1 = \left[\; B_1 \otimes A_2 - A_1 \otimes B_2 \;\right], \quad x = \operatorname{vec}(V),$$

and

$$T_2 = \begin{bmatrix} A_1^T \otimes I_m & -I_n \otimes A_2 \\ B_1^T \otimes I_m & -I_n \otimes B_2 \end{bmatrix}, \quad x = \begin{bmatrix} \operatorname{vec}(U) \\ \operatorname{vec}(V) \end{bmatrix}.$$

Thus, indeed we have that

$$\langle P_1, P_2 \rangle_1 = \dim \mathcal{N}(T_1) \quad \text{and} \quad \langle P_1, P_2 \rangle_2 = \dim \mathcal{N}(T_2),$$

where $\mathcal{N}(\cdot)$ denotes the nullspace of a matrix. It is clear that with either inner product, if $\overline{\mathcal{O}}(P_1) \supseteq \overline{\mathcal{O}}(P_2)$, then $\langle P_1, T\rangle \le \langle P_2, T\rangle$ for any test pencil T. Both Pokrzywa and Bongartz prove the converse, giving necessary and sufficient conditions. Furthermore, both observe that one need consider only the indecomposable blocks $T \in \{L_k, L_k^T, J_k\}$ as test pencils giving three sequences of conditions. Explicit formulas may be found in Demmel and Edelman (under the term "interaction") [13] and Beitia and Gracia [5, Thm. 4.5] although these papers did not make the connection to closure relations.

Since the inner product is bilinear, it is sufficient to display a table where the arguments are indecomposable blocks:

$\langle P_1, P_2\rangle_1$	L_k	L_k^T	$J_k(\gamma)$
L_j	$j+k+1$	$(k-j)_+$	k
L_j^T	$(j-k)_+$	0	0
$J_j(\gamma)$	j	0	$\min(j,k)$

$\langle P_1, P_2\rangle_2$	L_k	L_k^T	$J_k(\gamma)$
L_j	$(j+1-k)_+$	0	0
L_j^T	$j+k$	$(k+1-j)_+$	k
$J_j(\gamma)$	j	0	$\min(j,k)$

The inner product on Jordan structures corresponding to different eigenvalues is 0. Here, $J_k(\lambda)$ denotes a Jordan block for a finite or infinite eigenvalue. Notice from the tables that

$$\langle P, L_k\rangle_1 = \langle P^T, L_{k+1}\rangle_2, \quad \langle P, L_k^T\rangle_1 = \langle P^T, L_{k-1}^T\rangle_2, \quad \langle P, J_k\rangle_1 = \langle P^T, J_k\rangle_2.$$

This is not coincidence; we have shown that by eliminating the U from the Bongartz equation involving U and V, one obtains exactly the corresponding Pokrzywa relation, which the reader may notice is symmetric.

Demmel and Edelman were interested in $\operatorname{cod}(P) = \langle P, P\rangle_2 - (m-n)^2$ so as to understand the codimension of $\mathcal{O}(P)$ and the relation to the staircase algorithms for their computation. In our Part I paper [19] we observed that $\langle P, P\rangle_2 = \dim \mathcal{N}(T_2)$. Indeed, when $P = P_1 = P_2$, the tangent space of $P = A - \lambda B$ is the range of the Kronecker product block matrix T_2 [19].

Of course $\operatorname{cod}(P_1) \le \operatorname{cod}(P_2)$ if $\overline{\mathcal{O}}(P_1) \supseteq \overline{\mathcal{O}}(P_2)$, but the converse does not hold. We at first conjectured that a test pencil approach might work here. The conjecture was that if $\operatorname{cod}(P_1 \oplus T) \le \operatorname{cod}(P_2 \oplus T)$ for all test pencils T, then $\overline{\mathcal{O}}(P_1) \supseteq \overline{\mathcal{O}}(P_2)$. Unfortunately, this does not hold even for the Jordan case. We found a counterexample consisting of two matrices A_1 and A_2 with Segre characteristics $(5,1,1,1)$ and $(4,3,1)$, respectively. For this example $\operatorname{cod}(A_2 \oplus J_k) \le \operatorname{cod}(A_1 \oplus J_k)$ for every k, but there is no closure relationship between the orbits of the two matrices (see Figure 2.1).

5.2. Quiver representations and path algebra modules. Perhaps some numerical analysts find it unsatisfying to talk about the Jordan case and then proceed to "analogues" or "generalizations" to the Kronecker case. In fact, the notions of equivalent structures, closure relations, indecomposable blocks, etc., all are elements of an elaborate general theory of *quiver representations* and path algebra modules.

In this theory, the echelon form corresponds to an $\mathbf{A}_2$ quiver with graph consisting of a single arrow ($\bullet\longrightarrow\bullet$), the Jordan form is an $\tilde{\mathbf{A}}_0$ quiver whose graph is a loop ($\bullet\circlearrowleft$), and the Kronecker form is an $\tilde{\mathbf{A}}_1$ quiver with two arrows ($\bullet \rightrightarrows \bullet$). A quiver is really a synonym for a directed graph. We obtain a *representation* [18, 39] of the quiver if we associate vertices with vector spaces and arrows with linear maps between the spaces, i.e., matrices. If two vectors are connected tail to tip, then the matrices may be multiplied. A representation of the quiver with three arrows ($\bullet \longrightarrow \bullet \longrightarrow \bullet \longrightarrow \bullet$) is simply three matrices A, B, C that can be multiplied to form CBA. Two representations are said to be equivalent if one can be obtained from the other by a change of basis in the vector space. Thus we have defined similarity of square matrices, strict equivalence of pencils, and many other equivalences with one sentence.

Let E be the incidence matrix of the quiver defined so that E_{ij} is the number of arrows pointing from i to j. The matrix $B = E + E^T$ is independent of the orientation of the arrows. The diagonal elements of B count the number of loops at the node twice. Depending on whether the matrix $2I - B$ is positive definite, semidefinite, or indefinite, the graph is said to be *finite*, *tame*, or *wild.* The finite graphs are known as *Dynkin diagrams* [31]. They correspond to canonical forms built from finitely many blocks, e.g., there are three building blocks for the echelon form: matrices of dimension 1×1, 1×0, and 0×1. The tame quivers include the Jordan and Kronecker forms and are manageable. The wild quivers are more difficult.

We may now define the *path algebra of a quiver.* Formally, it is a vector space generated by elements called paths where multiplication also is defined. A path is simply a sequence of vertices that follow edges. A path of length l may be denoted $(a|\alpha_1, \dots, \alpha_l|b)$, where a and b are nodes, α_1 is an arrow pointing away from a, the successive arrows point towards each other, and the last arrow points toward b. Paths that connect may be multiplied in the obvious way

$$(a|\alpha_1, \dots, \alpha_l|b)(b|\beta_1, \dots, \beta_s|c) = (a|\alpha_1, \dots, \alpha_l, \beta_1, \dots, \beta_s|c).$$

Two paths that do not connect are defined to have product 0. Notice that the paths of length 0: $(a|a)$ are idempotent: $(a|a)^2 = (a|a)$ and the sum of all the length 0 paths (one for each node) is the identity.

The Kronecker pencil example is the path algebra for $\tilde{\mathbf{A}}_1$. With the arrows labeled e_1 and e_2, it may be thought of as the four-dimensional space of the form

$$\alpha(1|1) + \beta(1|e_1|2) + \gamma(1|e_2|2) + \delta(2|2)$$

with the path algebra multiplication table:

	$(1\|1)$	$(1\|e_1\|2)$	$(1\|e_2\|2)$	$(2\|2)$
$(1\|1)$	$(1\|1)$	$(1\|e_1\|2)$	$(1\|e_2\|2)$	0
$(1\|e_1\|2)$	0	0	0	$(1\|e_1\|2)$
$(1\|e_2\|2)$	0	0	0	$(1\|e_2\|2)$
$(2\|2)$	0	0	0	$(2\|2)$

We can write such an element as $(\alpha, \beta, \gamma, \delta)$. This algebra is isomorphic to the set of four-dimensional matrices

$$\left(\begin{array}{cc|c} \alpha & 0 & \beta \\ 0 & \alpha & \gamma \\ \hline 0 & 0 & \delta \end{array}\right)$$

with ordinary matrix multiplication. Let A and B be arbitrary $m \times n$ rectangular matrices. We say that vectors v of length $m + n$ form a module over the path algebra of $\tilde{\mathbf{A}}_1$. Define $(\alpha, \beta, \gamma, \delta)v$ to mean

$$\left(\begin{array}{c|c} \alpha I_m & \beta A + \gamma B \\ \hline 0 & \delta I_n \end{array}\right) v. \tag{5.1}$$

It is easy to check that the product

$$(\alpha_1, \beta_1, \gamma_1, \delta_1)(\alpha_2, \beta_2, \gamma_2, \delta_2)v$$

may be computed in either order giving the same answer. (One way requires multiplication in the path algebra; the other is ordinary matrix-vector multiplication.)

There is a one-to-one correspondence between equivalent pencils and modules over the path algebra of $\tilde{\mathbf{A}}_1$, and generally this holds between representations and modules over a path algebra of a quiver. Given two representations of a quiver, if we have linear maps U_i from the first to the second, we say that we have a homomorphism if the diagram composed of the two quivers and the connecting U_i's is commutative, as in the following example:

$$\begin{tikzcd}
\bullet \arrow[r, "A_1"] \arrow[d, "U_1"'] & \bullet \arrow[r, "B_1"] \arrow[d, "U_2"'] & \bullet \arrow[r, "C_1"] \arrow[d, "U_3"'] & \bullet \arrow[d, "U_4"'] \\
\bullet \arrow[r, "A_2"'] & \bullet \arrow[r, "B_2"'] & \bullet \arrow[r, "C_2"'] & \bullet
\end{tikzcd}$$

It is the dimension of the set of homomorphisms between two quivers (dim Hom) that is used explicitly by Bongartz and implicitly by Pokrzywa to obtain the closure relations.

The coin moves also have an algebraic interpretation. Pencils with only L_k blocks correspond to what algebraists call projective modules, while those with only L_k^T blocks are the injective modules. To denote that a matrix A is an extension of A_1 and A_2 (see section 2.3), algebraists write a short exact sequence:

$$0 \to A_1 \to A \to A_2 \to 0.$$

This generalizes to any quiver, and each coin move corresponds to some exact sequence.

This concludes our brief introduction to the algebraic language for these ideas. It is worthwhile to mention that not every equivalence relation in systems theory corresponds to a quiver. The set of matrix pairs (A, B) with A $m \times m$ and B $n \times n$ with the equivalence $(A, B) \sim (U^{-1}AU, U^{-1}B)$ does not correspond to a quiver representation, but if we add the matrix V: $(A, B) \sim (U^{-1}AU, U^{-1}BV)$ then we do have a (wild) quiver [29]. Similarly if we have the matrix quadruples often studied in systems theory [42, 14] with the equivalence relation

$$\begin{pmatrix} P & R \\ O & Q \end{pmatrix} \begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} P^{-1} & 0 \\ S & T \end{pmatrix} = \begin{pmatrix} A' & B' \\ C' & D' \end{pmatrix},$$

we do not have a quiver, but if we omit the matrices R and S, then once again we have a wild quiver.

Acknowledgments. The first author would like to thank Jim Demmel for suggesting this problem one New Year's Eve in Evans Hall (weeks before he met his wife in Evans Hall). We would like to thank Richard Stanley, who always knows the best references and pointers to the literature. We are indebted to Michel Goemans for his help with the proof of Theorem 2.7. Finally we thank Klaus Bongartz for very helpful email discussions, and the minters of Belgian, British, Swedish, French, and American coins.

REFERENCES

[1] S. Abeasis and A. Del Fra, *Degenerations for the representations of a quiver of type $\mathcal{A}_m$*, J. Algebra, 93 (1985), pp. 376–412.

[2] V. I. Arnold, *On matrices depending on parameters*, Russian Math. Surveys, 26 (1971), pp. 29–43.

[3] V. I. Arnold, S. M. Gusein-Zade, and A. N. Varchenko, *Singularities of Differentiable Maps, Vol.* I, Birkhäuser Boston, Cambridge, MA, 1985.

[4] T. Beelen and P. Van Dooren, *An improved algorithm for the computation of Kronecker's canonical form of a singular pencil*, Linear Algebra Appl., 105 (1988), pp. 9–65.

[5] M. A. Beitia and J.-M. Gracia, *Sylvester matrix equations for matrix pencils*, Linear Algebra Appl., 232 (1996), pp. 155–197.

[6] D. Boley, *Estimating the sensitivity of the algebraic structure of pencils with simple eigenvalue estimates*, SIAM J. Matrix Anal. Appl., 11 (1990), pp. 632–643.

[7] D. Boley, *The Algebraic Structure of Pencils and Block Toeplitz Matrices*, Report TR-96-048, Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN, 1996.

[8] D. Boley and P. Van Dooren, *Placing zeros and the Kronecker canonical form*, Circuits Systems Signal Process., 13 (1994), pp. 783–802.

[9] K. Bongartz, *On degenerations and extensions of finite dimensional modules*, Adv. Math., 121 (1996), pp. 245–287.

[10] F. Chaitin-Chatelin and V. Frayssé, *Lectures on Finite Precision Computations*, SIAM, Philadelphia, PA, 1996.

[11] D. H. Collingwood and W. M. McGovern, *Nilpotent Orbits in Semisimple Lie Algebras*, Van Nostrand Reinhold, New York, 1993.

[12] I. De Hoyos, *Points of continuity of the Kronecker canonical form*, SIAM J. Matrix Anal. Appl., 11 (1990), pp. 278–300.

[13] J. Demmel and A. Edelman, *The dimension of matrices (matrix pencils) with given Jordan (Kronecker) canonical forms*, Linear Algebra Appl., 230 (1995), pp. 61–87.

[14] J. Demmel and B. Kågström, *Accurate solutions of ill-posed problems in control theory*, SIAM J. Matrix. Anal. Appl., 9 (1988), pp. 126–145.

[15] J. Demmel and B. Kågström, *The generalized Schur decomposition of an arbitrary pencil $A - \lambda B$: Robust software with error bounds and applications, part* I*: Theory and algorithms*, ACM Trans. Math. Software, 19 (1993), pp. 160–174.

[16] J. Demmel and B. Kågström, *The generalized Schur decomposition of an arbitrary pencil $A - \lambda B$: Robust software with error bounds and applications, part* II*: Software and applications*, ACM Trans. Math. Software, 19 (1993), pp. 175–201.

[17] H. den Boer and Ph. A Thijsse, *Semi-stability of sums of partial multiplicities under additive perturbation*, Integral Equations Operator Theory, 3 (1980), pp. 23–42.

[18] V. Dlab and C. M. Ringel, *Indecomposable representations of graphs and algebras*, Mem. Amer. Math. Soc., 6 (1996), pp. 155–197.

[19] A. Edelman, E. Elmroth, and B. Kågström, *A geometric approach to perturbation theory of matrices and matrix pencils. Part* I*: Versal deformations*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 653–692.

[20] E. Elmroth, *On the Stratification of the Kronecker Canonical Form*, Report UMINF-95.14, Department of Computing Science, Umeå University, Umeå, Sweden, 1995.

[21] E. Elmroth and B. Kågström, *The set of* 2*-by-*3 *matrix pencils—Kronecker structures and their transitions under perturbations*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 1–34.

[22] E. R. Gansner, *Acyclic digraphs, Young tableaux and nilpotent matrices*, SIAM J. Alg. Discrete Methods, 2 (1981), pp. 429–440.

[23] F. Gantmacher, *The Theory of Matrices, Vols.* I, II, Chelsea, New York, 1959.

[24] M. R. Garey and D. S. Johnson, *Computers and Intractabilty, A Guide to the Theory of NP-Completeness*, Freeman and Company, San Francisco, 1979.

[25] I. Gohberg, P. Lancaster, and L. Rodman, *Invariant Subspaces of Matrices with Applications*, John Wiley, New York, 1986.

[26] G. H. Golub and J. H. Wilkinson, *Ill-conditioned eigensystems and the computation of the Jordan canonical form*, SIAM Rev., 18 (1976), pp. 578–619.

[27] M. Goresky and R. MacPherson, *Stratified Morse Theory*, Springer-Verlag, Berlin, New York, 1988.

[28] M. Gu, *Finding well-conditioned similarities to block-diagonalize non-symmetric matrices is NP-hard*, J. Complexity, 11 (1995), pp. 377–391.

[29] D. Hinrichsen and D. Prätzel-Wolters, *A wild quiver in linear systems theory*, Linear Algebra Appl., 91 (1987), pp. 143–175.

[30] D. G. Hough, *Explaining and Ameliorating the Ill Condition of Zeros of Polynomials*, Ph.D. thesis, Memo UCB/ERL M77/30, Electronics Research Lab, University of California, Berkeley, CA, 1977.

[31] V. G. Kac, *Montecatini lectures on invariant theory*, in Invariant Theory: Proceedings of the First 1982 session of the Centro Internazionale Matematico Estivo, Springer-Verlag, Berlin, 1983, Lecture Notes in Math., Vol. 996, Proceedings, Montecatini, Italy, 1982.

[32] B. Kågström, *RGSVD—An algorithm for computing the Kronecker structure and reducing subspaces of singular $A - \lambda B$ pencils*, SIAM J. Sci. Stat. Comp., 7 (1986), pp. 185–211.

[33] B. Kågström and A. Ruhe, *An algorithm for the numerical computation of the Jordan normal form of a complex matrix*, ACM Trans. Math. Software, 6 (1980), pp. 389–419.

[34] V. B. Khazanov and V. Kublanovskaya, *Spectral problems for matrix pencils, methods and algorithms* I, Sov. J. Numer. Anal. Math. Modelling, 3 (1988), pp. 337–371.

[35] V. Kublanovskaya, *AB-algorithm and its modifications for the spectral problem of linear pencils of matrices*, Numer. Math., 43 (1984), pp. 329–342.

[36] A. S Markus and E. É. Parilis, *The change of the Jordan structure of a matrix under small perturbations*, Linear Algebra Appl., 54 (1983), pp. 139–152.

[37] A. Pokrzywa, *On perturbations and the equivalence orbit of a matrix pencil*, Linear Algebra Appl., 82 (1986), pp. 99–121.

[38] C. Riedtmann, *Degenerations for representations of quivers with relations*, Ann. Sci. École Norm. Sup, 4 (1986), pp. 275–301.

[39] C. M. Ringel, *Tame algebras and integral quadratic forms*, Lecture Notes in Math., 1099, Springer-Verlag, Berlin, 1984.

[40] R. P. Stanley, *Enumerative Combinatorics*, Wadsworth & Brooks/Cole, Monterey, CA, 1986.

[41] P. Van Dooren, *The computation of Kronecker's canonical form of a singular pencil*, Linear Algebra Appl., 27 (1979), pp. 103–141.

[42] P. Van Dooren, *The generalized eigenstructure problem in linear system theory*, IEEE Trans. Automat. Control, AC-26 (1981), pp. 111–129.

[43] W. Waterhouse, *The codimension of singular matrix pairs*, Linear Algebra Appl., 57 (1984), pp. 227–245.

[44] J. H. Wilkinson, *Linear differential equations and Kronecker's canonical form*, in Recent Advances in Numerical Analysis, C. de Boor and G. Golub, eds., Academic Press, New York, 1978, pp. 231–265.

SIAM J. Matrix Anal. Appl.
Vol. 20, No. 3, pp. 700–719

EFFECTIVE METHODS FOR SOLVING BANDED TOEPLITZ SYSTEMS*

DARIO ANDREA BINI[†] AND BEATRICE MEINI[†]

Abstract. We propose new algorithms for solving $n \times n$ banded Toeplitz systems with bandwidth m. If the function associated with the Toeplitz matrix has no zero in the unit circle, then $O(n \log m + m \log^2 m \log\log \epsilon^{-1})$ arithmetic operations (ops) are sufficient to approximate the solution of the system up to within the error ϵ; otherwise the cost becomes $O(n \log m + m \log^2 m \log \frac{n}{m})$ ops. Here $m = o(n)$ and $n > \log \epsilon^{-1}$. Some applications are presented. The methods can be applied to infinite and bi-infinite systems and to block matrices.

Key words. banded matrices, Toeplitz matrices, displacement rank, cyclic reduction, Graeffe's method

AMS subject classifications. 65F05, 65F10, 15A23

PII. S0895479897324585

1. Introduction.

Let $\{a_k\}_{k \in \boldsymbol{Z}}$ be a sequence of real numbers such that $a_k = 0$ for $k < m_2$, and for $k > m_1$, $a_{m_1} a_{m_2} \neq 0$, where $\boldsymbol{Z}$ is the ring of integers, and $m_1, m_2 \in \boldsymbol{Z}$, $m_2 < 0 < m_1$. We may associate with $\{a_k\}_{k \in \boldsymbol{Z}}$ the function $\phi(z) = \sum_{i=m_2}^{m_1} a_i z^i$ and a matrix $T = (t_{i,j})$, having entries $t_{i,j} = a_{i-j}$. Let us use different notations for T according to its size. That is, we denote by $T_{\pm\infty} = (t_{i,j})_{i,j \in \boldsymbol{Z}}$ the bi-infinite matrix, by $T_\infty = (t_{i,j})_{i,j \geq 0}$ the infinite matrix, and by $T_n = (t_{i,j})_{i,j=1,\ldots,n}$ the matrix having size n. We leave the notation T for a matrix having an unspecified size. T is a banded matrix with bandwidth $m = \max\{m_1, -m_2\}$, and has the Toeplitz structure; i.e., its entries are constant along the diagonals.

In this paper we propose a new approach to solving the (block) banded Toeplitz systems $T\boldsymbol{x} = \boldsymbol{b}$ with bandwidth m, leading to an algorithm that can be used either as a direct algorithm or as an iterative one. For $n \times n$ systems the direct method, which can be applied with almost no restriction, has a sequential cost of $O(n \log m + t(m) \log \frac{n}{m})$ arithmetic operations (ops), where $t(m)$ is the arithmetic cost of solving an $m \times m$ Toeplitz-like system, i.e., $t(m) \in \{m^2, m \log^2 m, m \log m\}$ according to the algorithm adopted (compare [23], [1], [15]).

Under suitable conditions generally satisfied in most part of the applications, the direct algorithm turns into an iterative one, and its convergence speed is quadratic. In this way the number of ops needed to approximate n components of the solution up to within an $O(\epsilon)$ error is $O(n \log m + t(m) \log\log \epsilon^{-1})$.

The cost $O(t(m) \log\log \epsilon^{-1})$ is required for arriving at an approximate LU factorization of the (infinite) matrix obtained by means of a suitable permutation of rows and columns of T. The $O(n \log m)$ term is the cost of the back substitution stage of the algorithm. This nice convergence property allows us to approximate n consecutive components of the solution of infinite and bi-infinite systems with the same computational cost as well. In the parallel random access machine (PRAM) model of parallel computation, the algorithm costs $O(t_s(m) \log \frac{n}{m})$ steps with $\max\{t_p(m), n\}$

*Received by the editors July 16, 1997; accepted for publication (in revised form) by G. Golub May 26, 1998; published electronically April 14, 1999.
http://www.siam.org/journals/simax/20-3/32458.html

[†]Dipartimento di Matematica, Università di Pisa, Via F. Buonarroti 2, 56127 Pisa, Italy (bini@dm.unipi.it, meini@dm.unipi.it).

processors, where $t_s(m)$, $t_p(m)$ are the number of steps and of processors, respectively, needed to solve an $m \times m$ Toeplitz-like system.

1.1. The problem and its relevance. The problem of solving banded Toeplitz systems as $T\boldsymbol{x} = \boldsymbol{b}$ plays an important role in many applications. Example of these systems are frequently encountered in the numerical solution of ordinary differential equations with boundary values, where the banded Toeplitz matrix provides a finite difference discretization of the differential operator [31], and in the numerical solution of many Markov chains coming from the modeling of queueing problems, where the involved matrix defines the probabilities of transition from one state to the other [26], [25], [18]. Another indirect and more recent application is related to the solution of general Toeplitz systems, like the ones encountered in the problems in signal processing [16], where the banded Toeplitz matrix is used as a preconditioner for speeding up the convergence of preconditioned conjugate gradient techniques [17], [30], [29].

In all the above problems the dimension of the matrices involved is typically huge or, sometimes, may even be infinite [26]. Moreover, in multidimensional problems the banded Toeplitz structure is expressed in terms of blocks; i.e., the scalar entries a_{j-i} are replaced by square matrices.

This makes it particularly important to devise efficient and reliable algorithms for the solution of (block) banded (block) Toeplitz systems that maintain their efficiency in the case of matrices having a very large, or even infinite, size.

A certain attention has been devoted to this problem in the literature, especially concerning the analysis of the complexity in parallel models of computation, like the PRAM model. In the PRAM model it is assumed that a certain number of processors work concurrently; more specifically, at each time unit (parallel step) each processor performs a single arithmetic operation. In this model the cost of a computation is expressed in terms of the number of parallel steps and of the number of processors needed to carry out the computation.

Different direct parallel algorithms have been developed [20], [4], [6], [5]. They are based mainly on two different techniques: embedding the matrix into a slightly larger matrix that can be diagonalized by means of trigonometric transforms (FFT) and correcting the matrix by means of a low rank additive correction in order to find a new matrix that can be diagonalized by a trigonometric transform and applying the Morrison–Sherman–Woodbury formula.

The algorithms obtained in this way have a parallel cost of $O(\log n + \log^2 m)$ steps with nm^2 processors, and their sequential cost of $O(n \log n + m^2 n)$ ops is not inferior to the cost of Gaussian elimination, that is, $O(m^2 n)$ ops. Moreover most of these algorithms have problems of numerical stability.

Recently, a direct and fully parallelizable algorithm for inverting block Toeplitz matrices in block Hessenberg form, based on a doubling technique and on the use of the Morrison–Sherman–Woodbury formula, has been proposed in [11]. The algorithm, applied to solve the system $T_n\boldsymbol{x} = \boldsymbol{b}$, has a sequential cost of $O(t(m) \log \frac{n}{m} + n \log n)$ ops. However, this algorithm does not fully exploit the band structure of the system and requires a large storage.

Iterative algorithms, based on the preconditioned conjugate gradient (PCG) techniques, are fully parallelizable and have a sequential cost per iteration of $O(n \log n)$. There exists a wide literature concerning these techniques; see, for instance, [16], [15], [29], [32]. The number of PCG iterations needed in the worst case depends on the bandwidth m (compare [30] for the dependency of the number of iterations on the specific features of the function associated with the matrix). Due to the Axelsson–Lindskög theorem [2], the number of iterations sufficient to reach a superlinear convergence is proportional to m if a circulant [14], [32] or τ-class [6], [7], [28] preconditioner is used. Moreover, $O(mn \log n)$ ops are needed for the currently known preconditioners [30].

1.2. New results. The new approach that we present in this paper is based on the following idea. The matrix T is partitioned into $m \times m$ blocks so that we obtain a block Toeplitz block tridiagonal matrix with Toeplitz blocks. The technique of cyclic reduction (see [19]) is applied to the reblocked matrix T; that is, an odd-even permutation of the block rows and the block columns of T is followed by a step of block Gaussian elimination.

We prove that the block tridiagonal block Toeplitz structure is kept almost everywhere by the cyclic reduction step. In other words, the Schur complement that we obtain after applying the single step of block Gaussian elimination to the matrix with permuted block rows and block columns is still block tridiagonal and, except for the block in position (1,1), block Toeplitz.

Moreover, a sort of deterioration occurs in the inner Toeplitz structure of the blocks of T. In fact, the blocks of the Schur complement obtained just at the first step of cyclic reduction are not Toeplitz any more. In principle, this fact would imply that each block must be stored in m^2 memory locations and that managing with blocks would cost $O(m^3)$ ops. In this way cyclic reduction would require $O(m^2 n + m^3 \log \frac{n}{m})$ ops in order to solve an $n \times n$ banded Toeplitz system with bandwidth m.

However, even though the Toeplitz structure of the blocks is lost, a more general structure is maintained by the blocks of the Schur complement at each step of cyclic reduction. More precisely, the main result that we prove in section 3 is that each block defining the Schur complement is a Toeplitz-like matrix of displacement rank at most four. Each block can be represented as the sum of at most four products of pairs of lower and upper triangular Toeplitz matrices. Therefore it can be stored in $O(m)$ memory locations by representing the block by means of few vectors. Furthermore, performing multiplications between the blocks costs $O(m \log m)$ ops, and inverting a block costs $t(m) \in \{m^2, m \log^2 m\}$ ops.

Moreover we prove explicit relations which hold among the vectors defining the blocks of the Schur complement at two consecutive steps. These relations allow us to implement the cyclic reduction technique in $O(n \log m + t(m) \log \frac{n}{m})$ ops.

The algorithm can be easily extended to banded block Toeplitz systems. For blocks of size h the cost is increased by the factor h^3.

The cyclic reduction algorithm obtained in this way has a further nice feature. If the polynomial $a(z) = z^{-m_2}\phi(z) = \sum_{i=m_2}^{m_1} a_i z^{i-m_2}$ has m zeros inside (outside) the unit circle and if the blocks have bounded norm, we may prove that the lower (upper) diagonal blocks of the Schur complement obtained at the jth step of cyclic reduction tend to zero as γ^{2^j}, where $0 < \gamma < 1$ (double exponential convergence).

For stochastic matrices, like the ones modeling Markov chains that arise in quasi-birth-death (QBD) problems [26], the above conditions are satisfied [9], [10]. In the case where T is symmetric and the polynomial $a(z)$ has no zero of modulus 1, then the above property is satisfied simultaneously inside and outside the unit circle, and the boundedness of the blocks follows from the Cauchy interlace theorem. In this case we may prove that the constant γ that rules the double exponential convergence is the modulus of the zero of $a(z)$ in the open unit disc, closest to the unit circle.

This property generalizes and extends the results of [13] proved for tridiagonal Toeplitz matrices, where $m = 1$. The conditions under which the convergence occurs are much weaker than the conditions given in [22], [33].

For matrices associated with polynomials having some zeros of modulus 1, we provide a simple trick that enables us to move the zeros of modulus 1 slightly in or out of the unit disc. Then, constructing the normal equations leads us to a system having a banded matrix with a nearly Toeplitz structure, which satisfies the convergence theorem. This fact allows us to improve the robustness of our algorithm.

The algorithm proposed in this paper is numerically stable if applied to positive definite symmetric matrices. In fact, all the $m \times m$ linear systems involved at each step of cyclic reduction have condition numbers not greater than the condition number of the original matrix T.

Another interesting feature of the algorithm is its close relationship with polynomial computations. In fact, we show that applying the cyclic reduction algorithm to a banded Toeplitz matrix corresponds to applying the Graeffe iteration [27] to the polynomial $\alpha(z)$ having as zeros the mth powers of the zeros of $a(z)$. Moreover, this similarity between polynomial and Toeplitz computations enables us to extend the Graeffe iteration to matrix polynomials.

We have implemented in Fortran 90 different versions of our algorithm which include the cases of matrices with scalar and block entries, symmetric and unsymmetric matrices, and stochastic matrices.

The algorithms have been compared with the PCG method with circulant preconditioner. From the many numerical experiments performed on an Alpha workstation, our algorithms have proved to be robust, numerically stable, and fast. For matrices of size $n = 524288$ and bandwidth $m = 32$ our programs have been 16 times faster than the PCG method. For larger dimensions the conjugate gradient method could not be applied for lack of memory.

The programs are available at the URL http://www.dm.unipi.it/pages/bini/public_html/ric.html.

2. Preliminaries. Without loss of generality let us assume that $-m_2 \geq m_1$; moreover, let n be an integer multiple of m, i.e., $n = mq$, for an integer q. We refer to $\phi(z)$ as the generating function of the matrix T, and we call T_n an *n-section of T_∞*.

The matrix T can be partitioned into $m \times m$ blocks yielding a block tridiagonal Toeplitz matrix. In particular T_n has the following structure:

$$T_n = \begin{bmatrix} A_0 & A_{-1} & & & \bigcirc \\ A_1 & A_0 & A_{-1} & & \\ & \ddots & \ddots & \ddots & \\ & & A_1 & A_0 & A_{-1} \\ \bigcirc & & & A_1 & A_0 \end{bmatrix},$$

where

$$
\begin{aligned}
A_1 &= \begin{bmatrix} a_m & a_{m-1} & \dots & a_1 \\ 0 & a_m & \ddots & \vdots \\ \vdots & \ddots & \ddots & a_{m-1} \\ 0 & \dots & 0 & a_m \end{bmatrix}, \\
A_0 &= \begin{bmatrix} a_0 & a_{-1} & \dots & a_{-m+1} \\ a_1 & a_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & a_{-1} \\ a_{m-1} & \dots & a_1 & a_0 \end{bmatrix}, \\
A_{-1} &= \begin{bmatrix} a_{-m} & 0 & \dots & 0 \\ a_{-m+1} & a_{-m} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ a_{-1} & \dots & a_{-m+1} & a_{-m} \end{bmatrix},
\end{aligned}
\tag{2.1}
$$

and similarly T_∞ and $T_{\pm\infty}$.

In light of the above block structure we may associate with T the matrix polynomial $A(z) = A_{-1} + zA_0 + z^2 A_1$.

In this section we first recall the cyclic reduction algorithm, originally introduced for the numerical solution of the Poisson equation [19], used in [8] for computing the probability invariant vector of stochastic matrices, and here adjusted in order to solve the block tridiagonal block Toeplitz system $T\boldsymbol{x} = \boldsymbol{b}$. Then we introduce the concept of displacement rank, and we recall some properties that are needed for the design and analysis of our algorithm.

2.1. Cyclic reduction. Consider the system

$$T\boldsymbol{x} = \boldsymbol{b}, \tag{2.2}$$

and partition the vectors $\boldsymbol{x}$ and $\boldsymbol{b}$ into blocks $\boldsymbol{x}_k$, $\boldsymbol{b}_k$ of dimension m, respectively; that is, $\boldsymbol{x} = (\boldsymbol{x}_k)$, $\boldsymbol{b} = (\boldsymbol{b}_k)$. By performing an odd-even permutation of the block rows and block columns in (2.2) we find that

$$\begin{bmatrix} D_1^{(0)} & U^{(0)} \\ L^{(0)} & D_2^{(0)} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_+^{(0)} \\ \boldsymbol{x}_-^{(0)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_+^{(0)} \\ \boldsymbol{b}_-^{(0)} \end{bmatrix}, \tag{2.3}$$

where $\boldsymbol{x}_+^{(0)} = (\boldsymbol{x}_{2k})$, $\boldsymbol{x}_-^{(0)} = (\boldsymbol{x}_{2k-1})$, $\boldsymbol{b}_+^{(0)} = (\boldsymbol{b}_{2k})$, $\boldsymbol{b}_-^{(0)} = (\boldsymbol{b}_{2k-1})$;

$$D_1^{(0)} = D_2^{(0)} = \begin{bmatrix} A_0 & & \bigcirc \\ & A_0 & \\ \bigcirc & & \ddots \end{bmatrix}, \quad L^{(0)} = \begin{bmatrix} A_{-1} & & & \bigcirc \\ A_1 & A_{-1} & & \\ & \ddots & \ddots & \\ \bigcirc & & \ddots & \ddots \end{bmatrix},$$

$$U^{(0)} = \begin{bmatrix} A_1 & A_{-1} & & \bigcirc \\ & A_1 & A_{-1} & \\ & & \ddots & \ddots \\ \bigcirc & & & \ddots \end{bmatrix}.$$

Applying one step of block Gaussian elimination to the 2×2 block system (2.3) yields the equivalent system

$$\begin{cases} (D_2^{(0)} - L^{(0)} D_1^{(0)^{-1}} U^{(0)}) \boldsymbol{x}_-^{(0)} = \boldsymbol{b}_-^{(0)} - L^{(0)} D_1^{(0)^{-1}} \boldsymbol{b}_+^{(0)}, \\ \boldsymbol{x}_+^{(0)} = D_1^{(0)^{-1}} (\boldsymbol{b}_+^{(0)} - U^{(0)} \boldsymbol{x}_-^{(0)}). \end{cases} \tag{2.4}$$

Denoting

$$T^{(1)} = D_2^{(0)} - L^{(0)} D_1^{(0)^{-1}} U^{(0)} \tag{2.5}$$

the Schur complement of $D_2^{(0)}$, $\boldsymbol{x}^{(1)} = \boldsymbol{x}_-^{(0)}$, $\boldsymbol{b}^{(1)} = \boldsymbol{b}_-^{(0)} - L^{(0)} D_1^{(0)^{-1}} \boldsymbol{b}_+^{(0)}$, the above system is ultimately reduced to solving

$$T^{(1)} \boldsymbol{x}^{(1)} = \boldsymbol{b}^{(1)}. \tag{2.6}$$

It is interesting to observe that $T^{(1)}$ is a block tridiagonal matrix which, except for the northwest corner block, has the block Toeplitz structure, i.e.,

$$T^{(1)} = \begin{bmatrix} \widehat{A}^{(1)} & A_{-1}^{(1)} & & \bigcirc \\ A_1^{(1)} & A_0^{(1)} & A_{-1}^{(1)} & \\ & A_1^{(1)} & A_0^{(1)} & \ddots \\ \bigcirc & & \ddots & \ddots \end{bmatrix}.$$

In other words, $T^{(1)}$ is uniquely determined by the blocks $A_{-1}^{(1)}$, $A_0^{(1)}$, $A_1^{(1)}$, $\widehat{A}^{(1)}$. Once $\boldsymbol{x}^{(1)}$ has been computed or approximated, the solution of the original system (2.2) can be recovered by means of back substitution through (2.4). In order to compute the solution $\boldsymbol{x}^{(1)}$ we may cyclically apply the same reduction (cyclic reduction) to the system (2.6). In this way we obtain a sequence of systems $\{T^{(j)} \boldsymbol{x}^{(j)} = \boldsymbol{b}^{(j)}\}$, where $T^{(0)} = T$, and $T^{(j)}$ is a block tridiagonal matrix having, except for the northwest corner, the block Toeplitz structure. Each matrix $T^{(j)}$ is uniquely determined by the blocks $\widehat{A}^{(j)}$, $A_{-1}^{(j)}$, $A_0^{(j)}$, $A_1^{(j)}$.

For a finite banded Toeplitz system $T_n \boldsymbol{x} = \boldsymbol{b}$, where $n = mq$, $q = 2^p$, p is a positive integer, the cyclic reduction generates a sequence of systems $T_{n_j}^{(j)} \boldsymbol{x}^{(j)} = \boldsymbol{b}^{(j)}$, $j = 1, 2, \ldots, p$, of dimension $n_j = m2^{p-j}$.

For a bi-infinite block tridiagonal block Toeplitz system $T_{\pm\infty} \boldsymbol{x} = \boldsymbol{b}$, where $\boldsymbol{x} = (\boldsymbol{x}_i)_{i \in \boldsymbol{Z}}$, $\boldsymbol{b} = (\boldsymbol{b}_i)_{i \in \boldsymbol{Z}}$, the cyclic reduction generates a sequence of bi-infinite block tridiagonal block Toeplitz systems $T_{\pm\infty}^{(j)} \boldsymbol{x}^{(j)} = \boldsymbol{b}^{(j)}$, where $T_{\pm\infty}^{(j)}$ is the bi-infinite block tridiagonal block Toeplitz matrix defined by the $m \times m$ blocks $A_i^{(j)}$, $i = -1, 0, 1$.

For finite matrices of size $n = m2^p$, the cyclic reduction process is carried out in p steps. The solution $\boldsymbol{x}^{(p)}$ is computed by solving the $m \times m$ system $T^{(p)} \boldsymbol{x}^{(p)} = \boldsymbol{b}^{(p)}$, and the whole solution of (2.2) is computed by means of back substitution, recursively applying (2.4) extended to the generic jth step.

For infinite or bi-infinite matrices, under suitable conditions (see section 4), we may prove that the sequence of systems $T^{(j)} \boldsymbol{x}^{(j)} = \boldsymbol{b}^{(j)}$ converges to a system having a block (bi)diagonal structure. In this way, by choosing j large enough, we may approximate to any precision an arbitrary finite number of components of the solution $\boldsymbol{x}^{(j)}$. The back substitution stage completes the recovering of the solution of (2.2).

The block entries $A_i^{(j+1)}$, $i=-1,0,1$, $\widehat{A}^{(j+1)}$, obtained at step $j+1$ that define the matrix $T^{(j+1)}$, are related to the block entries $A_i^{(j)}$, $i=-1,0,1$, $\widehat{A}^{(j)}$, obtained at step j, by means of simple relations. For this purpose let us associate with $T^{(j)}$ the matrix polynomial

$$A^{(j)}(z)=A_{-1}^{(j)}+zA_0^{(j)}+z^2A_1^{(j)} \tag{2.7}$$

and express in compact form the block odd-even permutation and the Gaussian elimination of one step of cyclic reduction in terms of $A^{(j)}(z)$.

In fact from (2.5) we may easily deduce that

$$\begin{cases} A^{(j+1)}(z)=zA_0^{(j)}-(A_{-1}^{(j)}+zA_1^{(j)})A_0^{(j)^{-1}}(A_{-1}^{(j)}+zA_1^{(j)}), \\ \widehat{A}^{(j+1)}=\widehat{A}^{(j)}-A_{-1}^{(j)}A_0^{(j)^{-1}}A_1^{(j)}. \end{cases} \tag{2.8}$$

The above relations yield the matrix equations

$$\begin{cases} A_{-1}^{(j+1)} &= -A_{-1}^{(j)}A_0^{(j)^{-1}}A_{-1}^{(j)}, \\ A_0^{(j+1)} &= A_0^{(j)}-A_1^{(j)}A_0^{(j)^{-1}}A_{-1}^{(j)}-A_{-1}^{(j)}A_0^{(j)^{-1}}A_1^{(j)}, \\ A_1^{(j+1)} &= -A_1^{(j)}A_0^{(j)^{-1}}A_1^{(j)}, \\ \widehat{A}^{(j+1)} &= \widehat{A}^{(j)}-A_{-1}^{(j)}A_0^{(j)^{-1}}A_1^{(j)}, \end{cases} \tag{2.9}$$

which allow us to compute the sequence of blocks defining the matrices $T^{(j)}$, $j=1,2,\ldots$.

A direct inspection shows that the Toeplitz structure of the blocks $A_i^{(0)}$, $i=-1,0,1$, is no longer kept by the blocks $A_i^{(j)}$, $i=-1,0,1$, and $\widehat{A}^{(j)}$. For this property one might deduce that the initial structure cannot be exploited in order to increase the efficiency of the cyclic reduction algorithm.

However, we may prove that a more general structure, i.e., the displacement structure, is maintained by the blocks $A_i^{(j)}$, $i=-1,0,1$, and $\widehat{A}^{(j)}$. This fact allows us to devise a suitable FFT-based implementation of the cyclic reduction algorithm, thus reaching a dramatic reduction of the complexity bound.

For this purpose we need to rewrite the functional relation (2.8) for the matrix power series $A^{(j)}(z)$ in the following equivalent form:

$$A^{(j+1)}(z^2)=\left(\frac{A^{(j)}(z)^{-1}-A^{(j)}(-z)^{-1}}{2z}\right)^{-1}=-A^{(j)}(z)A_0^{(j)^{-1}}A^{(j)}(-z). \tag{2.10}$$

2.2. Displacement rank. The concept of displacement rank, introduced by Kailath, Vieira, and Morf in [24], is fundamental in devising and analyzing algorithms related to Toeplitz matrices. Several displacement operators have been introduced and analyzed in the literature; here we adopt an operator that seems to be particularly suitable for dealing with our problems.

Define the $m\times m$ *down-shift* matrix

$$Z=\begin{bmatrix} 0 & & & \\ 1 & 0 & & \\ & \ddots & \ddots & \\ & & 1 & 0 \end{bmatrix},$$

and consider the *displacement operator*

$$\Delta(H) = ZH - HZ$$

defined for any $m \times m$ matrix H. Moreover, denote by $L(\boldsymbol{w})$ the $m \times m$ lower triangular Toeplitz matrix defined by its first column $\boldsymbol{w}$.

Let us also introduce the following notation: the vectors $\boldsymbol{e}_1 = [1, 0, \ldots, 0]^T$, $\boldsymbol{e}_m = [0, \ldots, 0, 1]^T$ denote the first and the last column of the identity matrix of order m, respectively.

Observe that for a general Toeplitz matrix A the displacement $\Delta(A)$ is zero except for the entries in the first row and in the last column, from which the rank of $\Delta(A)$ is at most 2.

We say that the matrix H has *displacement rank* r if r is the rank of $\Delta(H)$, i.e., if there exist vectors $\boldsymbol{u}_i, \boldsymbol{v}_i$, $i = 1, \ldots, r$, satisfying the equation $\Delta(H) = \sum_{i=1}^{r} \boldsymbol{u}_i \boldsymbol{v}_i^T$.

According to this definition an $m \times m$ Toeplitz matrix A has displacement rank at most 2; more precisely it holds that

$$\Delta(A) = -\boldsymbol{e}_1 \boldsymbol{e}_m^T AZ + ZA\boldsymbol{e}_m \boldsymbol{e}_m^T.$$

A nice property of $\Delta(H)$ is that the displacement of the inverse matrix can be explicitly related to the displacement of the matrix itself; in fact, if H is nonsingular, then

$$\Delta(H^{-1}) = -H^{-1}\Delta(H)H^{-1}. \tag{2.11}$$

The following result has been proved in [3], [12].

THEOREM 2.1. *Let K be an $m \times m$ matrix such that $\Delta(K) = \sum_{i=1}^{r} \boldsymbol{u}_i \boldsymbol{v}_i^T$, where $\boldsymbol{u}_i$ and $\boldsymbol{v}_i$ are m-dimensional vectors. Then we have*

$$K = L(K\boldsymbol{e}_1) - \sum_{i=1}^{r} L(\boldsymbol{u}_i)L^T(Z\boldsymbol{v}_i).$$

The above result allows one to represent any matrix K as a sum of products of lower and upper triangular Toeplitz matrices defined by the first column of K and by the vectors $\boldsymbol{u}_i$, $\boldsymbol{v}_i$ associated with the displacement of K.

If the matrix K is nonsingular, then the above representation theorem can be applied to K^{-1} in light of (2.11), yielding the following result:

$$K^{-1} = L(K^{-1}\boldsymbol{e}_1) + \sum_{i=1}^{r} L(K^{-1}\boldsymbol{u}_i)L^T(ZK^{-T}\boldsymbol{v}_i).$$

We recall that multiplying an $m \times m$ Toeplitz matrix and a vector costs $O(m \log m)$ ops if the computation is performed by means of FFT [12]. In the PRAM model, the cost is $O(\log m)$ steps with m processors.

3. Structural properties of cyclic reduction. In this section we prove that the matrix polynomials $A^{(j)}(z)$, $j = 0, 1, \ldots$, defined in (2.7), (2.10), have displacement rank at most 2 and provide explicit formulae relating the displacement of $A^{(j)}(z)$ to the displacement of $A^{(j+1)}(z)$. As a consequence we find that $A_{-1}^{(j)}$ and $A_1^{(j)}$ have displacement rank at most 2, and $A_0^{(j)}$ has displacement rank at most 4. Moreover, we show that the blocks $\widehat{A}^{(j)}$ have displacement rank at most 3.

These properties allow us to devise an algorithm, based on FFT, that performs a single cyclic reduction step in $O(m\log^2 m)$ arithmetic operations if super-fast algorithms for the inversion of $m\times m$ Toeplitz-like matrices are used.

We need the following result.

THEOREM 3.1. *For the matrix polynomial $A^{(j)}(z)$ generated by the cyclic reduction* (2.10) *we have*

$$\Delta(A^{(j)}(z)) = \boldsymbol{a}^{(j)}(z)\boldsymbol{u}^{(j)T}(z) - \boldsymbol{v}^{(j)}(z)\boldsymbol{c}^{(j)T}(z), \quad j\ge 0, \tag{3.1}$$

where the vectors $\boldsymbol{a}^{(j)}(z)$, $\boldsymbol{c}^{(j)}(z)$, $\boldsymbol{u}^{(j)}(z)$, $\boldsymbol{v}^{(j)}(z)$ have entries which are polynomials in z, and are defined by the following equations:

$$\begin{aligned}
&\boldsymbol{a}^{(j)}(z) = A^{(j)}(z)\boldsymbol{e}_1,\ \boldsymbol{c}^{(j)T}(z) = \boldsymbol{e}_m^T A^{(j)}(z),\ j\ge 0;\\
&\boldsymbol{u}^{(j)}(z^2)^T = (-\boldsymbol{u}^{(j-1)}(z)^T (A_0^{(j-1)})^{-1} A^{(j-1)}(-z)\\
&\qquad\qquad +\boldsymbol{u}^{(j-1)}(-z)^T (A_0^{(j-1)})^{-1} A^{(j-1)}(z))/(2z), \quad j\ge 1;\\
&\boldsymbol{v}^{(j)}(z^2) = (-A^{(j-1)}(-z)(A_0^{(j-1)})^{-1}\boldsymbol{v}^{(j-1)}(z)\\
&\qquad\qquad +A^{(j-1)}(z)(A_0^{(j-1)})^{-1}\boldsymbol{v}^{(j-1)}(-z))/(2z), \quad j\ge 1;\\
&\boldsymbol{u}^{(0)}(z)^T = z\boldsymbol{e}_m^T,\ \boldsymbol{v}^{(0)}(z) = z\boldsymbol{e}_1.
\end{aligned} \tag{3.2}$$

Proof. We prove relation (3.1) by induction on j. For $j=0$ a simple calculation shows that

$$\begin{aligned}
\Delta(A^{(0)}(z)) &= z(A^{(0)}(z)\boldsymbol{e}_1\boldsymbol{e}_m^T - \boldsymbol{e}_1\boldsymbol{e}_m^T A^{(0)}(z))\\
&= A^{(0)}(z)\boldsymbol{e}_1\boldsymbol{u}^{(0)T}(z) - \boldsymbol{v}^{(0)}(z)\boldsymbol{e}_m^T A^{(0)}(z).
\end{aligned}$$

Suppose that relation (3.1) holds for a fixed $j\ge 0$; we show that it holds for $j+1$. For simplicity let us denote $B = A^{(j)}(z)^{-1} - A^{(j)}(-z)^{-1}$. For the properties of the displacement operator (2.11) and for (2.10) we have

$$\begin{aligned}
\Delta(A^{(j+1)}(z^2)) &= 2z\Delta(B^{-1}) = -2zB^{-1}\Delta(B)B^{-1}\\
&= -2zB^{-1}(-A^{(j)}(z)^{-1}\Delta(A^{(j)}(z))A^{(j)}(z)^{-1}\\
&\quad\ + A^{(j)}(-z)^{-1}\Delta(A^{(j)}(-z))A^{(j)}(-z)^{-1})B^{-1};
\end{aligned}$$

by substituting relation (3.1) in the above equation we obtain

$$\begin{aligned}
\Delta(A^{(j+1)}(z^2)) &= 2zB^{-1}\boldsymbol{e}_1\boldsymbol{u}^{(j)}(z)^T A^{(j)}(z)^{-1}B^{-1} - 2zB^{-1}A^{(j)}(z)^{-1}\boldsymbol{v}^{(j)}(z)\boldsymbol{e}_m^T B^{-1}\\
&\quad - 2zB^{-1}\boldsymbol{e}_1\boldsymbol{u}^{(j)}(-z)^T A^{(j)}(-z)^{-1}B^{-1}\\
&\quad + 2zB^{-1}A^{(j)}(-z)^{-1}\boldsymbol{v}^{(j)}(-z)\boldsymbol{e}_m^T B^{-1}\\
&= A^{(j+1)}(z^2)\boldsymbol{e}_1\boldsymbol{u}^{(j+1)}(z^2)^T - \boldsymbol{v}^{(j+1)}(z^2)\boldsymbol{e}_m^T A^{(j+1)}(z^2). \qquad \square
\end{aligned}$$

By induction on j, from (3.2), it readily follows that $\boldsymbol{u}^{(j)}(z)$ and $\boldsymbol{v}^{(j)}(z)$ have components which are polynomials of degree at most 1. Therefore, since $\boldsymbol{a}^{(j)}(z)$ and $\boldsymbol{c}^{(j)}(z)$ have degree at most 2, we may denote

$$\begin{aligned}
\boldsymbol{u}^{(j)}(z) &= \boldsymbol{u}_{-1}^{(j)} + z\boldsymbol{u}_0^{(j)},\\
\boldsymbol{v}^{(j)}(z) &= \boldsymbol{v}_{-1}^{(j)} + z\boldsymbol{v}_0^{(j)},\\
\boldsymbol{a}^{(j)}(z) &= \boldsymbol{a}_{-1}^{(j)} + z\boldsymbol{a}_0^{(j)} + z^2\boldsymbol{a}_1^{(j)},\\
\boldsymbol{c}^{(j)}(z) &= \boldsymbol{c}_{-1}^{(j)} + z\boldsymbol{c}_0^{(j)} + z^2\boldsymbol{c}_1^{(j)}.
\end{aligned}$$

Moreover, by comparing the term of degree 3 in (3.1) we may deduce that, for any $j \geq 0$, it holds that

$$\boldsymbol{a}_1^{(j)} \boldsymbol{u}_0^{(j)T} = \boldsymbol{v}_0^{(j)} \boldsymbol{c}_1^{(j)T}. \tag{3.3}$$

In this way, from the above properties and from Theorem 3.1 we deduce explicit relations for the vectors $\boldsymbol{u}_i^{(j)}$, $\boldsymbol{v}_i^{(j)}$, $i = -1, 0$, $\boldsymbol{a}_i^{(j)}$, $\boldsymbol{c}_i^{(j)}$, $i = -1, 0, 1$, $j \geq 0$, defining $\Delta(A^{(j)}(z))$:

$$\begin{aligned}
\boldsymbol{u}_{-1}^{(j+1)T} &= \boldsymbol{u}_{-1}^{(j)T} - \boldsymbol{u}_0^{(j)T} A_0^{(j)^{-1}} A_{-1}^{(j)},\\
\boldsymbol{u}_0^{(j+1)T} &= -\boldsymbol{u}_0^{(j)T} A_0^{(j)^{-1}} A_1^{(j)},\\
\boldsymbol{v}_{-1}^{(j+1)} &= \boldsymbol{v}_{-1}^{(j)} - A_{-1}^{(j)} A_0^{(j)^{-1}} \boldsymbol{v}_0^{(j)},\\
\boldsymbol{v}_0^{(j+1)} &= -A_1^{(j)} A_0^{(j)^{-1}} \boldsymbol{v}_0^{(j)},\\
\boldsymbol{a}_{-1}^{(j+1)} &= -A_{-1}^{(j)} A_0^{(j)^{-1}} \boldsymbol{a}_{-1}^{(j)},\\
\boldsymbol{a}_0^{(j+1)} &= \boldsymbol{a}_0^{(j)} - A_{-1}^{(j)} A_0^{(j)^{-1}} \boldsymbol{a}_1^{(j)} - A_1^{(j)} A_0^{(j)^{-1}} \boldsymbol{a}_{-1}^{(j)},\\
\boldsymbol{a}_1^{(j+1)} &= -A_1^{(j)} A_0^{(j)^{-1}} \boldsymbol{a}_1^{(j)},\\
\boldsymbol{c}_{-1}^{(j+1)T} &= -\boldsymbol{c}_{-1}^{(j)^T} A_0^{(j)^{-1}} A_{-1}^{(j)},\\
\boldsymbol{c}_0^{(j+1)T} &= \boldsymbol{c}_0^{(j)T} - \boldsymbol{c}_{-1}^{(j)T} A_0^{(j)^{-1}} A_1^{(j)} - \boldsymbol{c}_1^{(j)T} A_0^{(j)^{-1}} A_{-1}^{(j)},\\
\boldsymbol{c}_1^{(j+1)T} &= -\boldsymbol{c}_1^{(j)T} A_0^{(j)^{-1}} A_1^{(j)},
\end{aligned} \tag{3.4}$$

where $\boldsymbol{u}_{-1}^{(0)} = \boldsymbol{v}_{-1}^{(0)} = \boldsymbol{0}$, $\boldsymbol{u}_0^{(0)} = \boldsymbol{e}_m$, $\boldsymbol{v}_0^{(0)} = \boldsymbol{e}_1$.

The above relations, together with Theorem 3.1, yield the following result.

THEOREM 3.2. *For the matrices $A_{-1}^{(j)}$, $A_0^{(j)}$, $A_1^{(j)}$, $j \geq 0$, generated by the cyclic reduction* (2.8) *we have*

$$\begin{aligned}
\Delta(A_{-1}^{(j)}) &= \boldsymbol{a}_{-1}^{(j)} \boldsymbol{u}_{-1}^{(j)T} - \boldsymbol{v}_{-1}^{(j)} \boldsymbol{c}_{-1}^{(j)T},\\
\Delta(A_0^{(j)}) &= \boldsymbol{a}_{-1}^{(j)} \boldsymbol{u}_0^{(j)T} + \boldsymbol{a}_0^{(j)} \boldsymbol{u}_{-1}^{(j)T} - \boldsymbol{v}_{-1}^{(j)} \boldsymbol{c}_0^{(j)T} - \boldsymbol{v}_0^{(j)} \boldsymbol{c}_{-1}^{(j)T},\\
\Delta(A_1^{(j)}) &= \boldsymbol{r}^{(j)} \boldsymbol{u}_0^{(j)T} - \boldsymbol{v}_0^{(j)} \widehat{\boldsymbol{c}}^{(j)T},
\end{aligned} \tag{3.5}$$

where

$$\begin{aligned}
\boldsymbol{r}^{(0)} &= \boldsymbol{a}_0^{(0)},\\
\boldsymbol{r}^{(j+1)} &= \boldsymbol{r}^{(j)} - A_1^{(j)} A_0^{(j)^{-1}} \boldsymbol{a}_{-1}^{(j)}, \quad j \geq 0,
\end{aligned} \tag{3.6}$$

and $\widehat{\boldsymbol{c}}^{(j)T} = \boldsymbol{e}_m^T \widehat{A}^{(j)}$.

Proof. By comparing the terms in z of degree 0 and 1 in relation (3.1) we obtain the expression for $\Delta(A_{-1}^{(j)})$ and $\Delta(A_0^{(j)})$, respectively. Similarly, by comparing the terms of degree 2 we have

$$\Delta(A_1^{(j)}) = \boldsymbol{a}_0^{(j)} \boldsymbol{u}_0^{(j)T} + \boldsymbol{a}_1^{(j)} \boldsymbol{u}_{-1}^{(j)T} - \boldsymbol{v}_{-1}^{(j)} \boldsymbol{c}_1^{(j)T} - \boldsymbol{v}_0^{(j)} \boldsymbol{c}_0^{(j)T}. \tag{3.7}$$

For $j = 0$ relation (3.5) immediately follows. For $j \geq 1$, from an inductive application of (3.4) we may deduce that $\boldsymbol{u}_{-1}^{(j)T} = -\sum_{i=0}^{j-1} \boldsymbol{u}_0^{(i)T} A_0^{(i)^{-1}} A_{-1}^{(i)}$ and $\boldsymbol{v}_{-1}^{(j)} = -\sum_{i=0}^{j-1} A_{-1}^{(i)} A_0^{(i)^{-1}} \boldsymbol{v}_0^{(i)}$, from which

$$\boldsymbol{a}_1^{(j)} \boldsymbol{u}_{-1}^{(j)T} = -\sum_{i=0}^{j-1} \boldsymbol{a}_1^{(j)} \boldsymbol{u}_0^{(i)T} A_0^{(i)^{-1}} A_{-1}^{(i)}. \tag{3.8}$$

On the other hand, from (2.9), it can be easily proved by induction on j that, for any $j \geq 1$ and $0 \leq i < j$, it holds that

$$A_1^{(j)} = (-1)^{j-i} \left(\prod_{h=i}^{j-1} A_1^{(j-h+i-1)} A_0^{(j-h+i-1)^{-1}} \right) A_1^{(i)}, \tag{3.9}$$

where $\prod_{h=h_1}^{h_2} X_{\sigma(h)} = X_{\sigma(h_1)} \cdots X_{\sigma(h_2)}$. Thus, from (3.8), (3.9), and (3.3), we have

$$\begin{aligned} \boldsymbol{a}_1^{(j)} \boldsymbol{u}_{-1}^{(j)T} &= - \textstyle\sum_{i=0}^{j-1} (-1)^{j-i} \left(\prod_{h=i}^{j-1} A_1^{(j-h+i-1)} A_0^{(j-h+i-1)^{-1}} \right) \boldsymbol{a}_1^{(i)} \boldsymbol{u}_0^{(i)T} A_0^{(i)^{-1}} A_{-1}^{(i)} \\ &= - \textstyle\sum_{i=0}^{j-1} (-1)^{j-i} \left(\prod_{h=i}^{j-1} A_1^{(j-h+i-1)} A_0^{(j-h+i-1)^{-1}} \right) \boldsymbol{v}_0^{(i)} \boldsymbol{c}_1^{(i)T} A_0^{(i)^{-1}} A_{-1}^{(i)}. \end{aligned} \tag{3.10}$$

By applying the recursive relation for $\boldsymbol{v}_0^{(j)}$, given in formula (3.4), it may be proved by induction on j that, for any $j \geq 1$, $0 \leq i < j$, it holds that

$$\boldsymbol{v}_0^{(j)} = (-1)^{j-i} \left(\prod_{h=i}^{j-1} A_1^{(j-h+i-1)} A_0^{(j-h+i-1)^{-1}} \right) \boldsymbol{v}_0^{(i)}, \tag{3.11}$$

from which, from (3.10) and (3.11), it follows that

$$\boldsymbol{a}_1^{(j)} \boldsymbol{u}_{-1}^{(j)T} = - \sum_{i=0}^{j-1} \boldsymbol{v}_0^{(j)} \boldsymbol{c}_1^{(i)T} A_0^{(i)^{-1}} A_{-1}^{(i)},$$

and, in (3.7), we obtain

$$\boldsymbol{a}_1^{(j)} \boldsymbol{u}_{-1}^{(j)T} - \boldsymbol{v}_0^{(j)} \boldsymbol{c}_0^{(j)T} = -\boldsymbol{v}_0^{(j)} \left(\boldsymbol{c}_0^{(j)T} + \sum_{i=0}^{j-1} \boldsymbol{c}_1^{(i)T} A_0^{(i)^{-1}} A_{-1}^{(i)} \right) = -\boldsymbol{v}_0^{(j)} \widehat{\boldsymbol{c}}^{(j)T}.$$

Similarly, it may be proved that

$$\boldsymbol{a}_0^{(j)} \boldsymbol{u}_0^{(j)T} - \boldsymbol{v}_{-1}^{(j)} \boldsymbol{c}_1^{(j)T} = \left(\boldsymbol{a}_0^{(j)} + \sum_{i=0}^{j-1} A_{-1}^{(i)} A_0^{(i)^{-1}} \boldsymbol{a}_1^{(i)} \right) \boldsymbol{u}_0^{(j)T} = \boldsymbol{r}^{(j)} \boldsymbol{u}_0^{(j)T}. \qquad \square$$

The above theorem, in light of the results of section 2.2, yields the following representations for the matrices $A_{-1}^{(j)}$, $A_0^{(j)}$, $A_1^{(j)}$.

THEOREM 3.3. *At each step j of cyclic reduction, the matrices $A_{-1}^{(j)}$, $A_0^{(j)}$, $A_1^{(j)}$, $A_0^{(j)^{-1}}$ can be rewritten as*

$$\begin{aligned} A_{-1}^{(j)} &= L(\boldsymbol{a}_{-1}^{(j)})(I - L^T(Z\boldsymbol{u}_{-1}^{(j)}) + L(\boldsymbol{v}_{-1}^{(j)}) L^T(Z\boldsymbol{c}_{-1}^{(j)})), \\ A_0^{(j)} &= L(\boldsymbol{a}_0^{(j)})(I - L^T(Z\boldsymbol{u}_{-1}^{(j)})) - L(\boldsymbol{a}_{-1}^{(j)}) L^T(Z\boldsymbol{u}_0^{(j)}) + L(\boldsymbol{v}_{-1}^{(j)}) L^T(Z\boldsymbol{c}_0^{(j)}) \\ &\qquad + L(\boldsymbol{v}_0^{(j)}) L^T(Z\boldsymbol{c}_{-1}^{(j)}), \\ A_1^{(j)} &= L(\boldsymbol{a}_1^{(j)}) - L(\boldsymbol{r}^{(j)}) L^T(Z\boldsymbol{u}_0^{(j)}) + L(\boldsymbol{v}_0^{(j)}) L^T(Z\widehat{\boldsymbol{c}}^{(j)}), \\ A_0^{(j)^{-1}} &= L(A_0^{(j)^{-1}} \boldsymbol{e}_1) + L^T(Z A_0^{(j)^{-T}} \boldsymbol{u}_{-1}^{(j)}) + L(A_0^{(j)^{-1}} \boldsymbol{a}_{-1}^{(j)}) L^T(Z A_0^{(j)^{-T}} \boldsymbol{u}_0^{(j)}) \\ &\qquad - L(A_0^{(j)^{-1}} \boldsymbol{v}_0^{(j)}) L^T(Z A_0^{(j)^{-T}} \boldsymbol{c}_{-1}^{(j)}). \end{aligned}$$

Proof. The proof readily follows from Theorems 3.2 and 2.1. $\square$

Similar properties hold for the matrices $\widehat{A}^{(j)}$.

THEOREM 3.4. *For the matrices $\widehat{A}^{(j)}$, $j \geq 0$, generated by the cyclic reduction* (2.8) *we have*

$$\Delta(\widehat{A}^{(j)}) = -\boldsymbol{e}_1\boldsymbol{c}_{-1}^{(0)T} + \boldsymbol{a}_{-1}^{(j)}\boldsymbol{u}_0^{(j)T} - \boldsymbol{v}_{-1}^{(j)}\widehat{\boldsymbol{c}}^{(j)T}, \tag{3.12}$$

from which the matrices $\widehat{A}^{(j)}$ can be rewritten as

$$\widehat{A}^{(j)} = L(\widehat{\boldsymbol{a}}^{(j)}) + L^T(Z\boldsymbol{c}_{-1}^{(0)}) - L(\boldsymbol{a}_{-1}^{(j)})L^T(Z\boldsymbol{u}_0^{(j)}) + L(\boldsymbol{v}_{-1}^{(j)})L^T(Z\widehat{\boldsymbol{c}}^{(j)}),$$

where $\widehat{\boldsymbol{a}}^{(j)} = \widehat{A}^{(j)}\boldsymbol{e}_1$, $\widehat{\boldsymbol{c}}^{(j)T} = \boldsymbol{e}_m^T\widehat{A}^{(j)}$.

Proof. We prove (3.12) by induction on j. For $j = 0$ the equation is obviously verified. Let us suppose that (3.12) holds for j, and let us prove it for $j+1$. From (2.8) and from the properties of displacement rank we have

$$\Delta(\widehat{A}^{(j+1)}) = \Delta(\widehat{A}^{(j)}) - \Delta(A_{-1}^{(j)}A_0^{(j)^{-1}}A_1^{(j)}). \tag{3.13}$$

From (2.11) it follows that

$$\begin{aligned}&\Delta(A_{-1}^{(j)}A_0^{(j)^{-1}}A_1^{(j)})\\ &= \Delta(A_{-1}^{(j)})A_0^{(j)^{-1}}A_1^{(j)} - A_{-1}^{(j)}A_0^{(j)^{-1}}\Delta(A_0^{(j)})A_0^{(j)^{-1}}A_1^{(j)} + A_{-1}^{(j)}A_0^{(j)^{-1}}\Delta(A_1^{(j)}).\end{aligned}$$

By substituting relations (3.5) in the above equation, it may be easily proved, by applying (3.4), that

$$\begin{aligned}&\Delta(A_{-1}^{(j)}A_0^{(j)^{-1}}A_1^{(j)})\\ &= -\boldsymbol{v}_{-1}^{(j+1)}\boldsymbol{c}_{-1}^{(j)T}A_0^{(j)^{-1}}A_1^{(j)} - \boldsymbol{a}_{-1}^{(j+1)}\boldsymbol{u}_0^{(j+1)T} + A_{-1}^{(j)}A_0^{(j)^{-1}}\boldsymbol{v}_{-1}^{(j)}\boldsymbol{c}_1^{(j)} + A_{-1}^{(j)}A_0^{(j)^{-1}}\boldsymbol{r}^{(j)}\boldsymbol{u}_0^{(j)},\end{aligned} \tag{3.14}$$

from which, by (3.13), (3.14), and from the inductive hypothesis, it follows that

$$\begin{aligned}&\Delta(\widehat{A}^{(j+1)})\\ &= -\boldsymbol{e}_1\boldsymbol{c}_{-1}^{(0)T} + \boldsymbol{a}_{-1}^{(j+1)}\boldsymbol{u}_0^{(j+1)T} - \boldsymbol{v}_{-1}^{(j+1)}\boldsymbol{e}_m^T(\widehat{A}^{(j)} - A_{-1}^{(j)}A_0^{(j)^{-1}}A_1^{(j)})\\ &+ (\boldsymbol{a}_{-1}^{(j)} - A_{-1}^{(j)}A_0^{(j)^{-1}}\boldsymbol{r}^{(j)})\boldsymbol{u}_0^{(j)T} - A_{-1}^{(j)}A_0^{(j)^{-1}}\boldsymbol{v}_{-1}^{(j)}\boldsymbol{c}_1^{(j)T}.\end{aligned}$$

On the other hand, by following the same lines of the proof of Theorem 3.2 and from (3.6), it may be verified that

$$(\boldsymbol{a}_{-1}^{(j)} - A_{-1}^{(j)}A_0^{(j)^{-1}}\boldsymbol{r}^{(j)})\boldsymbol{u}_0^{(j)T} = A_{-1}^{(j)}A_0^{(j)^{-1}}\boldsymbol{v}_{-1}^{(j)}\boldsymbol{c}_1^{(j)T}. \qquad \square$$

The vectors $\widehat{\boldsymbol{a}}^{(j)}$, $\widehat{\boldsymbol{c}}^{(j)}$ obtained at two subsequent stages of cyclic reduction are related by means of the equations

$$\begin{aligned}\widehat{\boldsymbol{a}}^{(j+1)} &= \widehat{\boldsymbol{a}}^{(j)} - A_{-1}^{(j)}A_0^{(j)^{-1}}\boldsymbol{a}_1^{(j)},\\ \widehat{\boldsymbol{c}}^{(j+1)T} &= \widehat{\boldsymbol{c}}^{(j)T} - \boldsymbol{c}_{-1}^{(j)T}A_0^{(j)^{-1}}A_1^{(j)}, \quad j \geq 0.\end{aligned} \tag{3.15}$$

Observe that all the above displacement properties still hold when the entries a_i of the blocks A_i, $i = -1, 0, 1$, of (2.1) are square matrices of dimension $h > 1$. In this case $\Delta(\cdot)$ is the block displacement operator, Z is the $m \times m$ block down-shift matrix, and the vectors $\boldsymbol{u}_i^{(j)}$, $\boldsymbol{v}_i^{(j)}$, $\boldsymbol{a}_i^{(j)}$, $\boldsymbol{c}_i^{(j)}$, $\boldsymbol{r}^{(j)}$, $\widehat{\boldsymbol{a}}_i^{(j)}$, $\widehat{\boldsymbol{c}}_i^{(j)}$ have m components, each component being a $h \times h$ matrix.

In the case of scalar entries a_i, the formulae displayed in Theorems 3.1–3.4 can be substantially simplified. In fact, from (2.10), we deduce the following.

PROPOSITION 3.5. *The matrix polynomials $A^{(j)}(z)$ and the vectors $\boldsymbol{r}^{(j)}(z)$, $\widehat{\boldsymbol{c}}^{(j)}(z)$ satisfy the following relations:*

$$A^{(j)}(z)^T = JA^{(j)}(z)J, \quad \widehat{\boldsymbol{c}}^{(j)}(z) = J\boldsymbol{r}^{(j)}(z),$$

where J denotes the permutation (reversion) matrix having unit entries along the antidiagonal.

From this, the vectors $\boldsymbol{u}^{(j)}(z)$, $\boldsymbol{v}^{(j)}(z)$, $\boldsymbol{a}^{(j)}(z)$, $\boldsymbol{c}^{(j)}(z)$ are related, as stated by the following.

PROPOSITION 3.6. *For the vectors $\boldsymbol{u}^{(j)}(z)$, $\boldsymbol{v}^{(j)}(z)$, $\boldsymbol{a}^{(j)}(z)$, $\boldsymbol{c}^{(j)}(z)$ we have $\boldsymbol{v}^{(j)}(z) = J\boldsymbol{u}^{(j)}(z)$, $\boldsymbol{c}^{(j)}(z) = J\boldsymbol{a}^{(j)}(z)$; thus*

$$\Delta(A^{(j)}(z)) = \boldsymbol{a}^{(j)}(z)\boldsymbol{v}^{(j)T}(z)J - \boldsymbol{v}^{(j)}(z)\boldsymbol{a}^{(j)T}(z)J.$$

In the case of symmetric matrices the formulae can be further simplified, since it can be easily verified that at each step j we have $A_1^{(j)} = A_{-1}^{(j)T}$, $A_0^{(j)} = A_0^{(j)T}$, $\widehat{A}^{(j)} = \widehat{A}^{(j)T}$. This simplification allows us to describe the algorithm in terms of the six vectors $\boldsymbol{a}_i^{(j)}$, $\boldsymbol{v}_i^{(j)}$, $i = -1, 0$, $\widehat{\boldsymbol{a}}^{(j)}$, $\boldsymbol{r}^{(j)}$.

The following algorithm enables us to perform a single cyclic reduction step by computing the blocks $\widehat{A}^{(j+1)}$, $A_i^{(j+1)}$, $i = -1, 0, 1$, given the blocks $\widehat{A}^{(j)}$, $A_i^{(j)}$, $i = -1, 0, 1$, by means of few FFTs of length $O(m)$. For the sake of simplicity we assume that $m = 2^s$ for a positive integer s and that the entries a_i are scalar numbers.

Algorithm.

Input: the vectors $\boldsymbol{r}^{(j)}$, $\widehat{\boldsymbol{a}}^{(j)}$, $\boldsymbol{v}_i^{(j)}$, $i = -1, 0$, $\boldsymbol{a}_i^{(j)}$, $i = -1, 0, 1$, defining the matrices $\widehat{A}^{(j)}$, $A_i^{(j)}$, $i = -1, 0, 1$, by means of Theorems 3.3 and 3.4 and Proposition 3.6;

Output: the vectors $\boldsymbol{r}^{(j+1)}$, $\widehat{\boldsymbol{a}}^{(j+1)}$, $\boldsymbol{v}_i^{(j+1)}$, $i = -1, 0$, $\boldsymbol{a}_i^{(j+1)}$, $i = -1, 0, 1$, defining the matrices $\widehat{A}^{(j+1)}$, $A_i^{(j+1)}$, $i = -1, 0, 1$, by means of Theorems 3.3 and 3.4 and Proposition 3.6;

Computation:

1. Compute the vectors $A_0^{(j)^{-1}}\boldsymbol{e}_1$, $A_0^{(j)^{-1}}\boldsymbol{a}_{-1}^{(j)}$, $A_0^{(j)^{-1}}\boldsymbol{v}_0^{(j)}$ $ZJA_0^{(j)^{-1}}\boldsymbol{v}_{-1}^{(j)}$, that fully define the six vectors representing the matrix $A_0^{(j)^{-1}}$, by solving four Toeplitz-like $m \times m$ systems.

2. Apply relations (3.4), (3.6), (3.15) in order to compute the vectors $\boldsymbol{r}^{(j+1)}$, $\widehat{\boldsymbol{a}}^{(j+1)}$, $\boldsymbol{v}_i^{(j+1)}$, $i = -1, 0$, $\boldsymbol{a}_i^{(j+1)}$, $i = -1, 0, 1$. By using the representation formula of $A_0^{(j)^{-1}}$ (see Theorem 3.3) and of the matrices $A_1^{(j)}$, $A_{-1}^{(j)}$, $\widehat{A}^{(j)}$, together with the results of section 2.2, this computation can be performed with a finite number of real FFTs of length $2m$.

The above results, integrated with the update of the right-hand side $\boldsymbol{b}^{(j)}$ by means of (2.4) and with a back substitution stage, lead to a direct algorithm for the solution of the finite system $T_n\boldsymbol{x} = \boldsymbol{b}$, where T_n is an $n \times n$ matrix, with $n = mq$, $q = 2^p$, p a positive integer.

The arithmetic cost of the latter algorithm can be easily estimated. In fact, denoting by $t(m)$ the cost of solving an $m \times m$ Toeplitz-like system, it follows that $O(n \log m + (m \log m + t(m)) \log \frac{n}{m})$ ops are sufficient to carry out the algorithm. The solution of an $m \times m$ Toeplitz-like system can be computed by extending the known algorithms for Toeplitz systems. Therefore, according to the specific properties of the matrices involved (symmetry, positive definiteness, conditioning), algorithms like the fast, super-fast, or iterative ones can be used for the computational cost $t(m)$ being $O(m^2)$, $O(m \log^2 m)$, or $O(m \log m)$, respectively.

The cyclic reduction algorithm has also nice properties with respect to parallel computations. In fact, consider the PRAM model, where we assume several processors working in parallel; at each step a processor performs an arithmetic operation, and the complexity of an algorithm is measured in terms of the number of parallel steps and of processors needed. Then the algorithm can be carried out in $O(t_s(m) \log \frac{n}{m})$ parallel steps, using at most $\max\{t_p(m), n\}$ processors, where $t_s(m)$, $t_p(m)$ denotes the number of parallel steps and of processors, respectively, needed to solve an $m \times m$ Toeplitz-like system. For well-conditioned matrices, where the PCG technique can be used, we have $t_s(m) = \log m$, $t_p(m) = m$.

4. Convergence properties. In this section we relate the polynomial $a(z) = z^m \sum_{k=m_2}^{m_1} z^k a_k$ associated with the matrix T with the polynomial $\alpha(z) = \det A(z)$. The results that we obtain are used in the next section to prove some useful convergence properties of the cyclic reduction algorithm.

Let us first recall the following result of [18] that relates the zero of $a(z)$ and $\alpha(z)$.

THEOREM 4.1. *For the polynomial $\alpha(z)$ it holds that*

$$\alpha(z^m) = \prod_{i=0}^{m-1} a(z\omega^i), \quad \omega = \cos\left(\frac{2\pi}{m}\right) + \boldsymbol{i} \sin\left(\frac{2\pi}{m}\right), \quad \boldsymbol{i}^2 = -1.$$

A consequence of the above theorem is that, if ξ is zero of the polynomial $a(z)$, then ξ^m is zero of $\alpha(z)$. Conversely, if η is zero of $\alpha(z)$, then there exists an mth root of η which is zero of $a(z)$.

Observe also that both the polynomials $a(z)$ and $\alpha(z)$ have degree $m+m_1$. This is trivial for $a(z)$; in order to prove it for $\alpha(z)$ it is sufficient to consider, in the expansion of $\det A(z)$, the term obtained by multiplying the entries on the m_1th diagonal below the main diagonal and the entries in the $(m - m_1)$th diagonal above the main one, yielding the z^{m+m_1} term.

It is surprising to observe that the cyclic reduction step is equivalent to the squaring step in the Graeffe algorithm, which is customarily used for factoring polynomials [27], [12]. In fact by taking determinants in both sides of (2.10) we obtain

$$\alpha^{(j+1)}(z^2) = -\alpha^{(j)}(z)\alpha^{(j)}(-z)/\det A_0^{(j)}.$$

An immediate consequence of the above relation is that the polynomials $\alpha^{(j)}(z)$ have degree $m + m_1$ and their zeros are explicitly given by

$$\xi_i^{m2^j}, \; i = 1, \ldots, m + m_1,$$

where

$$a(\xi_i) = 0, \quad i = 1, \ldots, m + m_1.$$

This property is very useful to prove some convergence results of the cyclic reduction algorithm.

First observe that the vector $\boldsymbol{\xi}^T = (1, \xi, \xi^2, \ldots)$ is such that $\boldsymbol{\xi}^T T_\infty = (\boldsymbol{w}^T, 0, \ldots)$, where $\boldsymbol{w}^T$ is a suitable vector of m components, for ξ ranging in the set of zeros of $a(z)$. Moreover, assume for simplicity that $a(z)$ has at least m distinct nonnull zeros $\xi_1, \ldots, \xi_m$, and consider the $m \times m$ Vandermonde matrix $V_m = (\xi_i^{j-1})_{i,j=1,m}$ made up with this set of zeros. It holds that

$$(V_m, DV_m, D^2V_m, \ldots)T_\infty = (W, O, \ldots), \quad D = \mathrm{Diag}(\xi_1^m, \ldots, \xi_m^m). \tag{4.1}$$

Observe that, for a polynomial $a(z)$ having multiple zeros, the above relation still holds if the matrix V_m is replaced by a generalized Vandermonde matrix. The homogeneous part of the above relation can be rewritten as

$$V_m A_{-1} + DV_m A_0 + D^2 V_m A_1 = O.$$

By applying the cyclic reduction to the system (4.1) we deduce that

$$V_m A_{-1}^{(j)} + D^{2^j} V_m A_0^{(j)} + D^{2^{j+1}} V_m A_1^{(j)} = O, \quad j = 0, 1, \ldots.$$

Multiplying the above relation on the left by V_m^{-1} we find

$$A_{-1}^{(j)} + F^{m2^j} A_0^{(j)} + F^{m2^{j+1}} A_1^{(j)} = O, \quad j = 0, 1, \ldots,$$

where $F = V_m^{-1} D V_m$ is the Frobenius matrix associated with the polynomial having zeros ξ_i, $i = 1, \ldots, m$.

The following result can be obtained from the above relations.

Theorem 4.2. *Let ξ_i, $i = 1, \ldots, m + m_1$, be the zeros of $a(z)$ ordered such that $|\xi_1| \le |\xi_2| \le \cdots \le |\xi_{m+m_1}|$. If $|\xi_m| < 1$ and if the blocks $A_i^{(j)}$ have norm bounded from above by a constant, then $\lim A_{-1}^{(j)} = O$. Moreover, if ξ_m is simple, then for any operator norm $||\cdot||$ it holds that $||A_{-1}^{(j)}|| = O(|\xi_m|^{m2^j})$; otherwise, for any $\epsilon > 0$ there exists an operator norm $||\cdot||$ such that $||A_{-1}^{(j)}|| = O((|\xi_m| + \epsilon)^{m2^j})$. Similarly, if $|\xi_{m_1+1}| > 1$ and the blocks have bounded norm, then $\lim A_1^{(j)} = O$. Moreover it holds that $||A_1^{(j)}|| = O(|\xi_{m_1+1}|^{-m2^j})$ for any operator norm or $||A_1^{(j)}|| = O((|\xi_{m_1+1}| - \epsilon)^{-m2^j})$ for a suitable norm according to the multiplicity of ξ_{m_1+1} as root of $a(z)$.*

Proof. For any $\epsilon > 0$ let $||\cdot||$ be an operator norm such that $||F|| < \rho(F) + \epsilon$ (compare [19]), where $\rho(F)$ denotes the spectral radius of F. Then we have

$$||A_{-1}^{(j)}|| \le ||F||^{m2^j} ||A_0^{(j)}|| + ||F||^{m2^{j+1}} ||A_1^{(j)}||,$$

from which, since $||A_0^{(j)}||, ||A_1^{(j)}||$ are bounded from above by a constant, it holds that

$$||A_{-1}^{(j)}|| = O((\rho(F) + \epsilon)^{m2^j}) = O((|\xi_m| + \epsilon)^{m2^j}).$$

If ξ_m is a simple root of $a(z)$, then there exists a norm $||\cdot||$ such that $||F|| = |\xi_m|$. Therefore, by applying the same argument as before, the above relation turns into $||A_{-1}^{(j)}|| = O(|\xi_m|^{m2^j})$. In order to prove that $||A_1^{(j)}|| = O((|\xi_{m_1+1}| - \epsilon)^{-m2^j})$ we apply a similar argument to the relation $A_{-1}^{(j)} + \hat{F}^{m2^j} A_0^{(j)} + \hat{F}^{m2^{j+1}} A_1^{(j)} = O$, where $\hat{F}$ is the Frobenius matrix associated with the polynomial having zeros $\xi_{m_1+1}, \ldots, \xi_{m_1+m}$. □

A nice consequence of Theorem 4.2 is that, if $T^{(0)} = T_\infty$ or $T^{(0)} = T_{\pm\infty}$, the sequence $T^{(j)}$ converges to a block (bi-)diagonal matrix. Thus, for $j = O(\log\log\epsilon^{-1})$, we may approximate an arbitrary finite number of components of the solution of the system $T^{(j)}\boldsymbol{x}^{(j)} = \boldsymbol{b}^{(j)}$, within the error ϵ. If our algorithm is applied to solve the finite system $T_n\boldsymbol{x} = \boldsymbol{b}$, where $n = m2^p$ and $p >> \log\log\epsilon^{-1}$, it is sufficient to perform $j = O(\log\log\epsilon^{-1})$ steps of cyclic reduction and then to compute the solution of $T^{(j)}\boldsymbol{x}^{(j)} = \boldsymbol{b}^{(j)}$, within the error ϵ, by solving a block (bi-)diagonal system. In this case the computational cost is $O(n\log m + t(m)\log\log\epsilon^{-1})$.

Conditions under which the blocks $A_i^{(j)}$ have bounded norm are given in [22], [33]. These conditions can be substantially relaxed in the real symmetric case where we require only that the polynomial $a(z)$ has no zeros of modulus 1. In this case the convergence speed depends on the modulus of the zero ξ closest to the unit circle. These properties are investigated with more details in the next section.

A very important situation where the blocks $A_i^{(j)}$ have bounded norm is the case $T_n = I - P_n$, with P_n a stochastic matrix. This case occurs in the analysis of Markov chains arising in queueing problems [9], [10].

4.1. The real symmetric positive definite case. Consider the case where the matrix T_n is real symmetric, so that for the blocks $A_i^{(j)}$ it holds that $A_{-1}^{(j)} = A_1^{(j)T}$ and $A_0^{(j)} = A_0^{(j)T}$, $j = 0, 1, \ldots$. Assume for simplicity that the polynomial $a(z)$ has no zero of modulus 1. Due to the symmetricity properties, the polynomial $a(z)$ is symmetric, that is, $a(z) = z^{2m}a(z^{-1})$, and its zeros occur in pairs $\xi_i = \xi_{2m-i+1}^{-1}$, $i = 1, \ldots, m$. Let us arrange the zeros ξ_i in such a way that $|\xi_1| \le |\xi_2| \le \cdots \le |\xi_m| < 1 < |\xi_{m+1}| \le \cdots \le |\xi_{2m}|$.

Due to the Szëgo theory on Toeplitz matrices [21], the matrices T_n, are either positive definite or negative definite for any n according to the sign that $a(z)$ takes over the unit circle. Throughout we assume, for simplicity T_n is positive definite. Moreover, the eigenvalues of the matrix T_n belong to the interval $[\min_{|z|=1}|a(z)|, \max_{|z|=1}|a(z)|]$.

It is useful to observe that the block tridiagonal matrices $T^{(j)}$ generated at each step of the cyclic reduction, being Schur complements in T_n, have eigenvalues belonging to the interval $[\min\lambda(T_n), \max\lambda(T_n)]$, where $\lambda(A)$ denotes the set of eigenvalues of the matrix A. Similarly the diagonal blocks

$$A_0^{(j)}, \quad \widehat{A}^{(j)}, \quad \begin{bmatrix} A_0^{(j)} & A_1^{(j)T} \\ A_1^{(j)} & A_0^{(j)} \end{bmatrix}$$

share the same properties due to the Cauchy interlace theorem [19]. Therefore it follows that the all the blocks $A_0^{(j)}$, $A_1^{(j)}$, $\widehat{A}^{(j)}$ have norm bounded from above by a constant.

These observations allow us to conclude with the following theorem.

Theorem 4.3. *Let T_n be a real symmetric matrix (Hermitian) matrix such that the associated polynomial $a(z)$ has no zeros in the unit circle. Then for the matrices $A_0^{(j)}$, $A_1^{(j)}$, $\widehat{A}^{(j)}$ generated by the cyclic reduction we have*

$$\lim A_1^{(j)} = O, \quad ||A_1^{(j)}|| \le \eta|\xi_m|^{m2^j}.$$

The above theorem can be used in order to devise an algorithm for the approximation of n components of the solution of the system $T\boldsymbol{x} = \boldsymbol{b}$ in $O((m\log m + t(m))\log\log\epsilon^{-1} + n\log m)$ ops provided that n is less than the size of T. More precisely, in light of Theorem 4.2, we may replace the term $\log\log\epsilon^{-1}$ with the expression

$\log_2 \log \theta\epsilon^{-1} - \log_2 \log |\xi_{m-1}|^{-m}$ for a suitable constant θ depending on the specific norm adopted in the proof of Theorem 4.2.

Another consequence of the boundedness of the eigenvalues of $T^{(j)}$ is that $\mathrm{cond}(T^{(j)}) \leq \mathrm{cond}(T)$, where cond is the condition number expressed in terms of the Euclidean norm. This means that the intermediate systems that are encountered in the cyclic reduction steps are not worse conditioned than the initial system.

4.2. The unsymmetric/indefinite case: Normal equations. If the polynomial $a(z)$ is not symmetric or if $a(\xi) = 0$ for some ξ of unit modulus, the convergence results of the previous section do not generally hold and the cyclic reduction technique cannot be used for solving the linear system $T_n\boldsymbol{x} = \boldsymbol{b}$ in an iterative fashion. Moreover, we cannot bound with a constant the condition number of the $m \times m$ blocks to be inverted at each step of cyclic reduction unless $a(z)$ is positive for $|z| = 1$. Therefore, the stability of the cyclic reduction technique is not ensured in this case.

However, with a simple trick we may reduce the general case to the positive definite one.

Let us first assume that $a(z)$ has zeros $\xi_1, \ldots \xi_{m+m_1}$ and that $|\xi_k| = 1$ for a given k. We may replace the system $T_n\boldsymbol{x} = \boldsymbol{b}$ with the systems

$$\begin{aligned} &DT_nD^{-1}\boldsymbol{y} = D\boldsymbol{b},\\ &\boldsymbol{y} = D\boldsymbol{x}, \end{aligned}$$

where $D = \mathrm{Diag}(1, \gamma, \gamma^2, \ldots, \gamma^{n-1})$ for a suitable γ. In this way the new matrix DT_nD^{-1} is still Toeplitz and is associated with the polynomial $a_\gamma(z) = a(\gamma z)$. The parameter γ can be chosen in such a way that $a_\gamma(z)$ has no zero of modulus 1. This can be done, deterministically, if information about the moduli of the zeros of $a(z)$ is known or by means of randomization with a random choice of γ in the range $[1, 1+\sigma]$ for a given positive σ, which guarantees with probability 1 that $a_\gamma(z)$ has no zero on the unit circle.

Indeed, if T_n is real symmetric, the new system is no longer symmetric. In order to keep the symmetricity property, we switch to the normal equations

$$\begin{aligned} &K_n\boldsymbol{y} = \boldsymbol{c},\\ &K_n = (DT_nD^{-1})^T(DT_nD^{-1}),\\ &\boldsymbol{c} = (DT_nD^{-1})^T D\boldsymbol{b}. \end{aligned} \tag{4.2}$$

It is easy to check that K_n has $4m + 1$ diagonals and is still Toeplitz except for the entries in the leading and in the trailing $m \times m$ submatrices. Moreover, for the Toeplitz part, the matrix is associated with the polynomial

$$q_\gamma(z) = \phi_\gamma(z)\phi_\gamma(z^{-1})z^{(m_1+m)}$$

having zeros $\xi_1, \ldots, \xi_{m+m_1}, \xi_1^{-1}, \ldots, \xi_{m_1+m}^{-1}$. More precisely $K_n = V_n - \mathrm{Diag}(R, O, S)$, where V_n is the banded Toeplitz matrix associated with $q_\gamma(z)$, $R = A_{-1}^T A_0^{-1}$, $S = A_1^T A_1$, and O denotes a null matrix of size $n - 2m$.

Partitioning K_n into $2m \times 2m$ blocks yields

$$K_n = \begin{bmatrix} \widehat{C} & B_{-1} & & & \\ B_1 & B_0 & B_{-1} & & \\ & \ddots & \ddots & \ddots & \\ & & B_1 & B_0 & B_{-1} \\ & & & B_1 & \widetilde{C} \end{bmatrix}. \tag{4.3}$$

Let us assume $n = 2mq$, where $q = 2^p - 1$ and p is a positive integer. In this way the cyclic reduction algorithm applied to the system (4.2) generates a sequence of systems whose matrices, of dimension $(2^{p-j} - 1)2m$, have the same structure of (4.3) and are defined by the $2m \times 2m$ blocks $\widehat{C}^{(j)}$, $\widetilde{C}^{(j)}$, $B_i^{(j)}$, $i = -1, 0, 1$, such that

$$\begin{cases} B_{-1}^{(j+1)} = -B_{-1}^{(j)} B_0^{(j)^{-1}} B_{-1}^{(j)}, \\ B_0^{(j+1)} = B_0^{(j)} - B_1^{(j)} B_0^{(j)^{-1}} B_{-1}^{(j)} - B_{-1}^{(j)} B_0^{(j)^{-1}} B_1^{(j)}, \\ B_1^{(j+1)} = -B_1^{(j)} B_0^{(j)^{-1}} B_1^{(j)}, \\ \widehat{B}^{(j+1)} = \widehat{B}^{(j)} - B_{-1}^{(j)} B_0^{(j)^{-1}} B_1^{(j)}, \\ \widehat{C}^{(j+1)} = \widehat{B}^{(j+1)} + \widehat{C} - B_0, \\ \widetilde{C}^{(j+1)} = J\widehat{B}^{(j+1)} J + \widetilde{C} - B_0. \end{cases}$$

In light of the results of the previous section the matrices $\widehat{C}^{(j)}$, $\widetilde{C}^{(j)}$, and $B_i^{(j)}$, $i = -1, 0, 1$, have displacement rank independent of j, and an algorithm similar to that of section 4.1 applies.

Observe that the condition number of $B_0^{(j)}$ is bounded by the condition number of V_n and thus depends on $\max_{|z|=1} q_\gamma(z) / \min_{|z|=1} q_\gamma(z)$. A similar bound cannot be proved for the blocks $\widehat{C}^{(j)}$, $\widetilde{C}^{(j)}$.

5. Numerical experiments and applications. We have implemented several versions of our algorithms in Fortran 90. The most general one deals with banded (unsymmetric) block Toeplitz matrices. Two more specific versions deal with the cases of (unsymmetric) scalar Toeplitz matrices and with symmetric scalar Toeplitz matrices. For the inversion of the diagonal blocks we used the conjugate gradient method with no preconditioning. We performed several numerical experiments; here we report the ones concerning the scalar symmetric case.

The test matrices that we have considered are defined by the function $\phi(z) = \sum_{i=-m}^{m} z^i + (a_0 - 1)$ for $m = 16, 32, 64$, $n = 2^h$, $h = 14, 15, \ldots, 22$, where the value of a_0 has been chosen differently in such a way that the polynomial $z^m \phi(z)$ has some zeros more or less close to the unit circle. The right-hand vector of the system has been chosen so that the system has the solution $\boldsymbol{x} = (1, \ldots, 1)^T$.

More specifically, for $m = 16$, we tested the algorithm with $a_0 = 8.2$. In this case, the polynomial $\phi(z)z^m$ has no zeros in the unit circle; the zeros closest to the unit circle are two pairs of modulus 1.003052 and 0.99695731. Their ratio, 1.006, is far enough from 1 to allow the double exponential convergence of the cyclic reduction algorithm.

For $m = 32$ we have chosen $a_0 = 15.1315$. With this value, the polynomial $\phi(z)z^m$ has no zeros of modulus 1. The zeros closest to the unit circle have modulus 1.00009 and 0.9999099, and their ratio has modulus 1.00018.

For $m = 64$ we set $a_0 = 25$. With this value the cyclic reduction algorithm does not converge since the polynomial $\phi(z)z^m$ has two pairs of complex zeros of modulus 1.

The algorithm has been compared with the preconditioned conjugate gradient method obtained by using the circulant preconditioner of Strang [32] on an Alpha workstation.

In Tables 5.1, 5.2, and 5.3 we report the time in seconds needed by the two algorithms ("*" denotes failure of the algorithm due to lack of memory), the maximum error in the solution, and the ratios between the times needed.

Table 5.1
$m = 16,\ a_0 = 8.2.$

	Cyclic reduction		PCG		
n	Time (s.)	Residual	Time (s.)	Residual	PCG/CR
8192	0.6	3. e -13	1.0	2. e -11	1.7
16384	1.0	3. e -13	2.6	2. e -11	2.6
32768	1.8	7. e -13	6.9	2. e -11	3.8
65536	3.5	7. e -13	16.0	2. e -11	4.6
131072	6.7	7. e -13	36.9	2. e -11	5.5
262144	13.3	7. e -13	132.0	2. e -11	9.9
524288	26.5	7. e -13	376.3	2. e -11	14.2
1048576	52.8	7. e -13	*	*	*
2097152	98.3	7. e -13	*	*	*

Table 5.2
$m = 32,\ a_0 = 15.1315.$

	Cyclic reduction		PCG		
n	Time (s.)	Residual	Time (s.)	Residual	PCG/CR
8192	0.7	2. e -11	1.3	1. e -12	1.9
16384	1.1	1. e -10	3.3	1. e -12	3.0
32768	1.8	2. e -10	7.9	1. e -12	4.3
65536	3.4	2. e -10	17.2	4. e -11	5.1
131072	6.4	2. e -10	38.9	4. e -11	6.1
262144	12.3	2. e -10	139.3	4. e -11	11.3
524288	24.3	2. e -10	395.9	4. e -11	16.3
1048576	48.2	2. e -10	*	*	*
2097152	95.6	2. e -10	*	*	*

Table 5.3
$m = 64,\ a_0 = 25.$

	Cyclic reduction		PCG		
n	Time (s.)	Residual	Time (s.)	Residual	PCG/CR
8192	0.8	2. e -11	1.0	3. e -12	1.3
16384	1.2	4. e -11	2.5	1. e -11	2.1
32768	2.1	1. e -11	6.6	2. e -11	3.1
65536	3.6	6. e -12	15.4	6. e -12	4.3
131072	6.8	2. e -10	37.0	1. e -11	5.4
262144	13.0	4. e -11	123.0	4. e -11	9.5
524288	25.4	4. e -11	328.0	6. e -12	12.9
1048576	49.8	1. e -9	*	*	*
2097152	99.2	8. e -11	*	*	*

The numerical experiments confirm that our algorithm is faster than the PCG method, more efficient in terms of memory use, and numerically stable.

REFERENCES

[1] G. S. Ammar and B. Gragg, *Numerical experience with superfast real Toeplitz solver*, Linear Algebra Appl., 121 (1989), pp. 185–206.

[2] O. Axelsson and G. Lindskög, *The rate of convergence of the preconditioned conjugate gradient method*, Numer. Math., 52 (1986), pp. 499–523.

[3] D. Bini, *On a Class of Matrices Related to Toeplitz Matrices*, Tech. Report 83-5, SUNY at Albany, Albany, NY, 1983.

[4] D. Bini, *Parallel solution of certain Toeplitz linear systems*, SIAM J. Comput., 13 (1984), pp. 268–276.

[5] D. Bini, *Matrix structures in parallel matrix computations*, Calcolo, 25 (1988), pp. 37–51.

[6] D. Bini and M. Capovani, *Spectral and computational properties of band symmetric Toeplitz matrices*, Linear Algebra Appl., 52 (1983), pp. 99–126.

[7] D. Bini and F. Di Benedetto, *A new preconditioner for the parallel solution of positive definite Toeplitz systems*, in Proceedings of the 2nd Annual SPAA, Crete, Greece, ACM Press, 1990, pp. 220–223.

[8] D. Bini and B. Meini, *On cyclic reduction applied to a class of Toeplitz-like matrices arising in queueing problems*, in Computations with Markov Chains, W. J. Stewart, ed., Kluwer Academic Publisher, Norwell, MA, 1995, pp. 21–38.

[9] D. Bini and B. Meini, *On the solution of a nonlinear matrix equation arising in queueing problems*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 906–926.

[10] D. Bini and B. Meini, *Improved cyclic reduction for solving queueing problems*, Numer. Algorithms, 15 (1997), pp. 57–74.

[11] D. Bini and B. Meini, *Inverting block Toeplitz matrices in block Hessenberg form by means of displacement operators: Application to queueing problems*, Linear Algebra Appl., 272 (1998), pp. 1–16.

[12] D. Bini and V. Pan, *Matrix and Polynomial Computations,* Vol. 1: *Fundamental Algorithms*, Birkhäuser, Boston, 1994.

[13] S. Bondeli and W. Gander, *Cyclic reduction for special tridiagonal systems*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 321–330.

[14] R. H. Chan, *Circulant preconditioners for Hermitian Toeplitz systems*, SIAM J. Matrix Anal. Appl., 10 (1989), pp. 542–550.

[15] R. H. Chan and M. K. Ng, *Conjugate gradient methods for Toeplitz systems*, SIAM Rev., 38 (1996), pp. 427–482.

[16] R. H. Chan and M. K. Ng, *Scientific applications of iterative Toeplitz solvers*, Calcolo, 33 (1996), pp. 249–267.

[17] R. H. Chan and P. T. P. Tang, *Fast band-Toeplitz preconditioners for Hermitian Toeplitz systems*, SIAM J. Sci. Comput., 15 (1994), pp. 164–171.

[18] H. R. Gail, S. L. Hantler, and B. A. Taylor, *Non-skip-free $M/G/1$ and $G/M/1$ type Markov chains*, Adv. in Appl. Probab., 29 (1997), pp. 733–758.

[19] G. Golub and C. van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, 1989.

[20] J. Grcar and A. Sameh, *On certain parallel Toeplitz linear system solvers*, SIAM J. Sci. Statist. Comput., 2 (1981), pp. 238–256.

[21] U. Grenander and G. Szegö, *Toeplitz Forms and Their Applications*, 2nd ed., Chelsea House, New York, 1984.

[22] D. Heller, *Some aspects of the cyclic reduction algorithm for block tridiagonal linear systems*, SIAM J. Numer. Anal., 13 (1976), pp. 484–496.

[23] T. Kailath and A. H. Sayed, *Displacement structure: Theory and applications*, SIAM Rev., 37 (1995), pp. 297–386.

[24] T. Kailath, A. Vieira, and M. Morf, *Inverses of Toeplitz operators, innovations, and orthogonal polynomials*, SIAM Rev., 20 (1978), pp. 106–119.

[25] M. F. Neuts, *Matrix-Geometric Solutions in Stochastic Models*, Johns Hopkins University Press, Baltimore, 1981.

[26] M. F. Neuts, *Structured Stochastic Matrices of $M/G/1$ Type and Their Applications*, Marcel Dekker, New York, 1989.

[27] M. A. Ostrowski, *Recherches sur la methode de Graeffe et les zeros des polynomes et des series de Laurent*, Acta Math., 72 (1940), pp. 99–257.

[28] S. Serra, *Preconditioning strategies for asymptotically ill-conditioned block Toeplitz systems*, BIT, 34 (1994), pp. 579–594.

[29] S. Serra, *Optimal, quasi-optimal and superlinear preconditioners for asymptotically ill-conditioned positive definite Toeplitz systems*, Math. Comp., 66 (1997), pp. 651–665.

[30] S. Serra Capizzano, *Toeplitz preconditioner constructed from linear approximation processes*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 446–465.

[31] J. Stoer and R. Bulirsch, *Introduction to Numerical Analysis*, Springer-Verlag, New York, 1980.

[32] G. Strang, *A proposal for Toeplitz matrix computations*, Stud. Appl. Math., 74 (1986), pp. 171–176.

[33] P. Yalamov and V. Pavlov, *Stability of the block cyclic reduction*, Linear Algebra Appl., 249 (1996), pp. 341–358.

SIAM J. MATRIX ANAL. APPL.
Vol. 20, No. 3, pp. 720–755

A SUPERNODAL APPROACH TO SPARSE PARTIAL PIVOTING[*]

JAMES W. DEMMEL[†], STANLEY C. EISENSTAT[‡], JOHN R. GILBERT[§], XIAOYE S. LI[¶], AND JOSEPH W. H. LIU[||]

Abstract. We investigate several ways to improve the performance of sparse LU factorization with partial pivoting, as used to solve unsymmetric linear systems. We introduce the notion of unsymmetric supernodes to perform most of the numerical computation in dense matrix kernels. We introduce unsymmetric supernode-panel updates and two-dimensional data partitioning to better exploit the memory hierarchy. We use Gilbert and Peierls's depth-first search with Eisenstat and Liu's symmetric structural reductions to speed up symbolic factorization.

We have developed a sparse LU code using all these ideas. We present experiments demonstrating that it is significantly faster than earlier partial pivoting codes. We also compare its performance with UMFPACK, which uses a multifrontal approach; our code is very competitive in time and storage requirements, especially for large problems.

Key words. sparse matrix algorithms, unsymmetric linear systems, supernodes, column elimination tree, partial pivoting

AMS subject classifications. 65F05, 65F50

PII. S0895479895291765

1. Introduction.

The problem of solving sparse symmetric positive definite systems of linear equations on sequential and vector processors is fairly well understood. Normally, the solution process is performed in two phases:

- symbolic determination of the nonzero structure of the Cholesky factor;
- numeric factorization and solution.

Elimination trees [35] and compressed subscripts [41] reduce the time and space for symbolic factorization. Supernodal [5] and multifrontal [15] elimination allow the use of dense vector operations for nearly all of the floating-point computation, thus reducing the symbolic overhead in numeric factorization. Overall, the Megaflop rates of modern sparse Cholesky codes are nearly comparable to those of dense solvers [37, 38, 39].

[*] Received by the editors September 11, 1995; accepted for publication (in revised form) by A. Greenbaum May 15, 1998; published electronically April 14, 1999.
http://www.siam.org/journals/simax/20-3/29176.html

[†] Computer Science Division, University of California, Berkeley, CA 94720 (demmel@cs.berkeley.edu). The research of the first and fourth authors was supported in part by NSF grant ASC–9313958, DOE grant DE–FG03–94ER25219, UT Subcontract ORA4466 from ARPA Contract DAAL03–91–C0047, DOE grant DE–FG03–94ER25206, and NSF Infrastructure grants CDA–8722788 and CDA–9401156.

[‡] Department of Computer Science, Yale University, P.O. Box 208285, New Haven, CT 06520-8285 (sce@cs.yale.edu). The research of this author was supported in part by NSF grant CCR-9400921.

[§] Xerox Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto, CA 94304 (gilbert@xerox.com). The research of this author was supported in part by the Institute for Mathematics and Its Applications, University of Minnesota, and in part by DARPA Contract DABT63-95-C0087.

[¶] National Energy Research Scientific Computing (NERSC) Center, Lawrence Berkeley National Lab, 1 Cyclotron Rd., Berkeley, CA 94720 (xiaoye@nersc.gov). The research of this author was done at UC Berkeley and as a summer intern and consultant at Xerox PARC.

[||] Department of Computer Science, York University, North York, ON, M3J 1P3, Canada, (joseph@cs.yorku.ca). The research of this author was supported in part by the Natural Sciences and Engineering Research Council of Canada under grant A5509.

```
for column $j = 1$ to $n$ do
    $f = A(:, j)$;
    Symbolic factor: determine which columns of $L$ will update $f$;
    for each updating column $r < j$ in topological order do
        Col-col update: $f = f - f(r) \cdot L(r + 1{:}n, r)$;
    end for;
    Pivot: interchange $f(j)$ and $f(m)$, where $|f(m)| = \max |f(j{:}n)|$;
    Separate $L$ and $U$: $U(1{:}j, j) = f(1{:}j)$;    $L(j{:}n, j) = f(j{:}n)$;
    Scale: $L(:, j) = L(:, j)/L(j, j)$;
    Prune symbolic structure based on column $j$;
end for;
```

FIG. 1.1. *Left-looking LU factorization with column-column updates.*

For unsymmetric systems, where pivoting is required to maintain numerical stability, progress has been less satisfactory. Recent research has concentrated on two basic approaches: submatrix-based methods and column-based (or row-based) methods. Submatrix methods typically use some form of Markowitz ordering with threshold pivoting, in which each stage's pivot element is chosen from the uneliminated submatrix by criteria that attempt to balance numerical quality and preservation of sparsity. Recent submatrix codes include Amestoy and Duff's symmetric pattern multifrontal code MUPS [2] and Davis and Duff's unsymmetric multifrontal code UMFPACK [7].

Column methods, by contrast, typically use ordinary partial pivoting.[1] The pivot is chosen from the current column according to numerical considerations alone; the columns may be preordered before factorization to preserve sparsity. Figure 1.1 sketches a generic left-looking column LU factorization.[2] Notice that the bulk of the numeric computation occurs in column-column (col-col) updates, or, to use BLAS terminology [13, 14], in sparse AXPYs.

In column methods, the preordering for sparsity is completely separate from the factorization, just as in the symmetric case. This is an advantage when several matrices with the same nonzero structure but different numerical values must be factored. However, symbolic factorization cannot be separated from numeric factorization, because the nonzero structures of the factors depend on the numerical pivoting choices. Thus column codes must do some symbolic factorization at each stage; typically this amounts to predicting the structure of each column of the factors immediately before computing it. (George and Ng [22, 23] described ways to obtain upper bounds on the structure of the factors based only on the nonzero structure of the original matrix.) A disadvantage of the column methods is that they do not reorder the columns dynamically, so there may be more fill.

An early example of such a code is Sherman's NSPIV [42] (which is actually a row code). Gilbert and Peierls [29] showed how to use depth-first search and topological ordering to get the structure of each factor column. This gives a column code that runs in total time proportional to the number of nonzero floating-point operations, unlike the other partial pivoting codes. Eisenstat and Liu [21] designed a pruning

[1] Row methods are exactly analogous to column methods, and codes of both sorts exist. We will use column terminology; those who prefer rows may interchange the terms throughout the paper.

[2] We use Matlab notation for integer ranges and submatrices: $r{:}s$ or $(r{:}s)$ is the range of integers $(r, r + 1, \ldots, s)$. If I and J are sets of integers, then $A(I, J)$ is the submatrix of A with rows whose indices are from I and with columns whose indices are from J. $A(:, J)$ abbreviates $A(1 : n, J)$. $nnz(A)$ denotes the number of nonzeros in A.

technique to reduce the amount of structural information required for the symbolic factorization, as we describe further in section 4. The result is that the time and space for symbolic factorization are typically reduced to a small fraction of the entire factorization.

In view of the success of supernodal techniques for symmetric matrices, it is natural to consider the use of supernodes to enhance the performance of unsymmetric solvers. Like the nonzero structure of the factors, the boundaries of the supernodes cannot be determined in advance; rather, they emerge depending on pivoting choices during the factorization.

In this paper, we generalize supernodes to unsymmetric matrices, and we give efficient algorithms for locating and using unsymmetric supernodes during a column-based LU factorization. We describe a new code called SuperLU that uses depth-first search and symmetric pruning to speed up symbolic factorization, and uses unsymmetric supernodes to speed up numeric factorization. The rest of the paper is organized as follows. Section 2 introduces the tools we use: unsymmetric supernodes, panels, and the column elimination tree. Section 3 describes the supernodal numeric factorization. Section 4 describes the supernodal symbolic factorization. In section 5, we present experimental results: we benchmark our code on several test matrices, we compare its performance to other column and submatrix codes, and we investigate its cache behavior in some detail. Finally, section 6 presents conclusions and open questions.

2. Unsymmetric supernodes.

The idea of a supernode is to group together columns with the same nonzero structure, so they can be treated as a dense matrix for storage and computation. Supernodes were originally used for (symmetric) sparse Cholesky factorization [5, 15]. In the factorization $A = LL^T$ (or $A = LDL^T$), a supernode is a range $(r{:}s)$ of columns of L with the same nonzero structure below the diagonal; that is, $L(r{:}s, r{:}s)$ is full lower triangular and every row of $L(s{:}n, r{:}s)$ is either full or zero.

Rothberg and Gupta [38, 39] and Ng and Peyton [37] analyzed the effect of supernodes in Cholesky factorization on modern uniprocessor machines with memory hierarchies and vector or superscalar hardware. All the updates from columns of a supernode are summed into a dense vector before the sparse update is performed. This reduces indirect addressing and allows the inner loops to be unrolled. In effect, a sequence of col-col updates is replaced by a supernode-column (sup-col) update. The sup-col update can be implemented using a call to a standard dense Level 2 BLAS matrix-vector multiplication kernel. This idea can be further extended to supernode-supernode (sup-sup) updates, which can be implemented using a Level 3 BLAS dense matrix-matrix kernel. This can reduce memory traffic by an order of magnitude, because a supernode in the cache can participate in multiple column updates. Ng and Peyton reported that a sparse Cholesky algorithm based on sup-sup updates typically runs 2.5 to 4.5 times as fast as a col-col algorithm. Indeed, supernodes have become a standard tool in sparse Cholesky factorization [5, 37, 38, 43].

To sum up, supernodes as the source of updates help because of the following:

1. The inner loop (over rows) has no indirect addressing. (Sparse Level 1 BLAS is replaced by dense Level 1 BLAS.)
2. The outer loop (over columns in the supernode) can be unrolled to save memory references. (Level 1 BLAS is replaced by Level 2 BLAS.)

Supernodes as the destination of updates help because of the following:

3. Elements of the source supernode can be reused in multiple columns of the destination supernode to reduce cache misses. (Level 2 BLAS is replaced by Level 3 BLAS.)

Supernodes in sparse Cholesky can be determined during symbolic factorization, before the numeric factorization begins. However, in sparse LU, the nonzero structure cannot be predicted before numeric factorization, so we must identify supernodes on the fly. Furthermore, since the factors L and U are no longer transposes of each other, we must generalize the definition of a supernode.

2.1. Definition of unsymmetric supernodes. We considered several possible ways to generalize the symmetric definition of supernodes to unsymmetric factorization. We define $F = L + U - I$ to be the *filled matrix* containing both L and U.

T1. Same row and column structures: A supernode is a range $(r{:}s)$ of columns of L and rows of U, such that the diagonal block $F(r{:}s, r{:}s)$ is full, and outside that block all the columns of L in the range have the same structure and all the rows of U in the range have the same structure. T1 supernodes make it possible to do sup-sup updates, realizing all three benefits.

T2. Same column structure in L*:* A supernode is a range $(r{:}s)$ of columns of L with triangular diagonal block full and the same structure below the diagonal block. T2 supernodes allow sup-col updates, realizing the first two benefits.

T3. Same column structure in L, full diagonal block in U*:* A supernode is a range $(r{:}s)$ of columns of L and U, such that the diagonal block $F(r{:}s, r{:}s)$ is full, and below the diagonal block the columns of L have the same structure. T3 supernodes allow sup-col updates, like T2. In addition, if the storage for a supernode is organized as for a two-dimensional (2-D) array (for Level 2 or 3 BLAS calls), T3 supernodes do not waste any space in the diagonal block of U.

T4. Same column structure in L and U*:* A supernode is a range $(r{:}s)$ of columns of L and U with identical structure. (Since the diagonal is nonzero, the diagonal block must be full.) T4 supernodes allow sup-col updates, and also simplify storage of L and U.

T5. Supernodes of A^TA*:* A supernode is a range $(r{:}s)$ of columns of L corresponding to a Cholesky supernode of the symmetric matrix A^TA. T5 supernodes are motivated by George and Ng's observation [22] that (with suitable representations) the structures of L and U in the unsymmetric factorization $PA = LU$ are contained in the structure of the Cholesky factor of A^TA. In unsymmetric LU, these supernodes themselves are sparse, so we would waste time and space operating on them. Thus we do not consider them further.

Figure 2.1 is a schematic of definitions T1 through T4.

Supernodes are only useful if they actually occur in practice. The occurrence of symmetric supernodes is related to the clique structure of the chordal graph of the Cholesky factor, which arises because of fill during the factorization. Unsymmetric supernodes seem harder to characterize, but they also are related to dense submatrices arising from fill. We measured the supernodes according to each definition for 126 unsymmetric matrices from the Harwell–Boeing sparse matrix test collection [17] under various column orderings. Table 2.1 shows, for each definition, the fraction of nonzeros of L that are not in the first column of a supernode; this measures how much row index storage is saved by using supernodes. Corresponding values for symmetric supernodes for the symmetric Harwell–Boeing structural analysis problems usually range from about 0.5 to 0.9. Larger numbers are better, indicating larger supernodes.

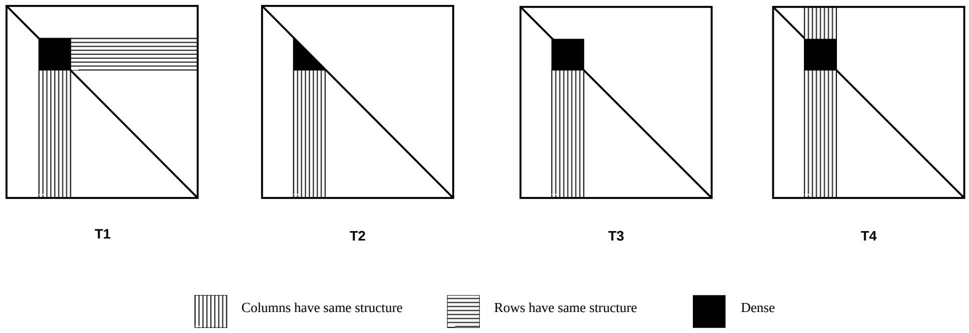


Fig. 2.1. *Four possible types of unsymmetric supernodes.*

Table 2.1
Fraction of nonzeros not in the first column of supernode.

	T1	T2	T3	T4
median	0.236	0.345	0.326	0.006
mean	0.284	0.365	0.342	0.052

We reject T4 supernodes as being too rare to make up for the simplicity of their storage scheme. T1 supernodes allow Level 3 BLAS updates, but as we will see in section 3.2 we can get most of their cache advantage with the more common T2 or T3 supernodes by using supernode-panel updates. Thus we conclude that either T2 or T3 is best by our criteria. Our code uses T2, which gives slightly larger supernodes than T3 at a small extra cost in storage (see section 2.2).

Figure 2.2 shows a sample matrix and the nonzero structure of its factors with no pivoting. Using definition T2, this matrix has four supernodes: $\{1,2\}$, $\{3\}$, $\{4,5,6\}$, and $\{7,8,9,10\}$. For example, in columns 4, 5, and 6 the diagonal blocks of L and U are full, and the columns of L all have nonzeros in rows 8 and 9. By definition T3, the matrix has five supernodes: $\{1,2\}$, $\{3\}$, $\{4,5,6\}$, $\{7\}$, and $\{8,9,10\}$. Column 7 fails to join $\{8,9,10\}$ as a T3 supernode because u_{78} is zero.

2.2. Storage of supernodes. A standard way to organize storage for a sparse matrix is a one-dimensional (1-D) array of nonzero values in column-major order, plus integer arrays giving row numbers and column starting positions. This is called *compressed column storage* and is also the scheme used in the Harwell–Boeing collection. We use this layout for both L and U, but with a slight modification: we store the entire square diagonal block of each supernode as part of L, including both the strict lower triangle of values from L and the upper triangle of values from U. We store this square block as if it were completely full (it is full in T3 supernodes, but its upper triangle may contain zeros in T2 supernodes). This allows us to address each supernode as a 2-D array in calls to BLAS routines. In other words, if columns $(r{:}s)$ form a supernode, then all the nonzeros in $F(r{:}n, r{:}s)$ are stored as a single dense 2-D array. This also lets us save some storage for row indices: only the indices of nonzero rows outside the diagonal block need be stored, and the structures of all columns within a supernode can be described by one set of row indices. This is similar to the effect of compressed subscripts in the symmetric case [41].

We represent the part of U outside the supernodal blocks with compressed column

$$\begin{pmatrix}
1 & \bullet & \bullet & & & \bullet & & & & \\
\bullet & 2 & & \bullet & & & & \bullet & & \\
 & & 3 & & & & \bullet & \bullet & & \\
 & & & 4 & \bullet & \bullet & & & \bullet & \\
 & & & \bullet & & & \bullet & & & \\
\bullet & & & & & 6 & & & \bullet & \\
 & & & & & & 7 & & \bullet & \bullet \\
\bullet & & \bullet & & & & & & & \\
 & & & \bullet & & \bullet & & & 9 & \\
 & & \bullet & & & & \bullet & & \bullet & 10
\end{pmatrix}
\qquad
\begin{pmatrix}
1 & \bullet & \bullet & & & \bullet & & & & \\
\bullet & 2 & \bullet & \bullet & & \bullet & & \bullet & & \\
 & & 3 & & & & \bullet & \bullet & & \\
 & & & 4 & \bullet & \bullet & & & \bullet & \\
 & & & \bullet & 5 & \bullet & \bullet & & \bullet & \\
\bullet & \bullet & \bullet & \bullet & \bullet & 6 & \bullet & \bullet & \bullet & \\
 & & & & & & 7 & & \bullet & \bullet \\
\bullet & \bullet & \bullet & \bullet & \bullet & \bullet & \bullet & 8 & \bullet & \bullet \\
 & & & \bullet & \bullet & \bullet & \bullet & \bullet & 9 & \bullet \\
 & & \bullet & & & & \bullet & \bullet & \bullet & 10
\end{pmatrix}$$

Original matrix A Factors $F = L + U - I$

FIG. 2.2. *A sample matrix and its LU factors. Diagonal elements a_{55} and a_{88} are zero.*

$$\begin{array}{c}
1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10
\end{array}
\begin{pmatrix}
s_1 & s_1 & u_3 & & & u_6 & & & & \\
s_1 & s_1 & u_3 & u_4 & & u_6 & & u_8 & & \\
 & & s_2 & & & & u_7 & u_8 & & \\
 & & & s_3 & s_3 & s_3 & & & u_9 & \\
 & & & s_3 & s_3 & s_3 & u_7 & & u_9 & \\
s_1 & s_1 & s_2 & s_3 & s_3 & s_3 & u_7 & u_8 & u_9 & \\
 & & & & & & s_4 & & s_4 & s_4 \\
s_1 & s_1 & s_2 & s_3 & s_3 & s_3 & s_4 & s_4 & s_4 & s_4 \\
 & & & s_3 & s_3 & s_3 & s_4 & s_4 & s_4 & s_4 \\
 & & s_2 & & & & s_4 & s_4 & s_4 & s_4
\end{pmatrix}$$

FIG. 2.3. *Supernodal structure (by definition* T2*) of the factors of the sample matrix.*

storage: the values are stored by columns, with a companion integer array the same size to store row indices; another array of n integers indicates the start of each column.

Figure 2.3 shows the structure of the factors in the example from Figure 2.2, with s_k denoting a nonzero in the kth supernode and u_k denoting a nonzero in the kth column of U outside the supernodal block. Figure 2.4 shows the storage layout. (We omit the indexing vectors that point to the beginning of each supernode and the beginning of each column of U.)

2.3. The column elimination tree. Since our definition requires the columns of a supernode to be contiguous, we should get larger supernodes if we bring together columns of L with the same nonzero structure. But the column ordering is fixed, for sparsity, before numeric factorization; what can we do?

In symmetric Cholesky factorization, one type of supernodes—the "fundamental" supernodes—can be made contiguous by permuting the matrix (symmetrically) according to a postorder on its elimination tree [4]. This postorder is an example of what Liu calls an equivalent reordering [35], which does not change the sparsity of the factor. The postordered elimination tree can also be used to locate the supernodes before the numeric factorization.

We proceed similarly for the unsymmetric case. Here the appropriate analogue of the symmetric elimination tree is the *column elimination tree*, or column etree

$$
\begin{array}{c|cc} & 1 & 2 \\ \hline & s_1 & s_1 \\ & s_1 & s_1 \\ 6 & s_1 & s_1 \\ 8 & s_1 & s_1 \end{array}
\qquad
\begin{array}{c|c} & 3 \\ \hline & s_2 \\ 6 & s_2 \\ 8 & s_2 \\ 10 & s_2 \end{array}
\qquad
\begin{array}{c|ccc} & 4 & 5 & 6 \\ \hline & s_3 & s_3 & s_3 \\ & s_3 & s_3 & s_3 \\ & s_3 & s_3 & s_3 \\ 8 & s_3 & s_3 & s_3 \\ 9 & s_3 & s_3 & s_3 \end{array}
\qquad
\begin{array}{cccc} 7 & 8 & 9 & 10 \\ \hline s_4 & 0 & s_4 & s_4 \\ s_4 & s_4 & s_4 & s_4 \\ s_4 & s_4 & s_4 & s_4 \\ s_4 & s_4 & s_4 & s_4 \end{array}
$$

Supernodal blocks (stored in column-major order)

row	1	2	2	1	2	3	5	6	2	3	6	4	5	6
	u_3	u_3	u_4	u_6	u_6	u_7	u_7	u_7	u_8	u_8	u_8	u_9	u_9	u_9

Nonzeros in columns of U outside supernodes

FIG. 2.4. *Storage layout for factors of the sample matrix, using* T2 *supernodes.*

for short. The vertices of this tree are the integers 1 through n, representing the columns of A. The column etree of A is the (symmetric) elimination tree of the column intersection graph of A, or equivalently the elimination tree of A^TA provided there is no cancellation in computing A^TA. See Gilbert and Ng [27] for complete definitions. The column etree can be computed from A in time almost linear in the number of nonzeros of A by a variation of an algorithm of Liu [35].

The following theorem states that the column etree represents potential dependencies among columns in LU factorization, and that (for strong Hall matrices) no stronger information is obtainable from the nonzero structure of A. Note that column i updates column j in LU factorization if and only if $u_{ij} \neq 0$.

THEOREM 2.1 (column etree [27]). *Let A be a square, nonsingular, possibly unsymmetric matrix, and let $PA = LU$ be any factorization of A with pivoting by row interchanges. Let T be the column etree of A.*

1. *If vertex i is an ancestor of vertex j in T, then $i \geq j$.*
2. *If $l_{ij} \neq 0$, then vertex i is an ancestor of vertex j in T.*
3. *If $u_{ij} \neq 0$, then vertex j is an ancestor of vertex i in T.*
4. *Suppose in addition that A is strong Hall (that is, A cannot be permuted to a nontrivial block triangular form). If vertex j is the parent of vertex i in T, then there is some choice of values for the nonzeros of A that makes $u_{ij} \neq 0$ when the factorization $PA = LU$ is computed with partial pivoting.*

Just as a postorder on the symmetric elimination tree brings together symmetric supernodes, we expect a postorder on the column etree to bring together unsymmetric supernodes. Thus, before we factor the matrix, we compute its column etree and permute the matrix columns according to a postorder on the tree. We now show that this does not change the factorization in any essential way.

THEOREM 2.2. *Let A be a matrix with column etree T. Let π be a permutation such that whenever $\pi(i)$ is an ancestor of $\pi(j)$ in T, we have $i \geq j$. Let P be the permutation matrix such that $\pi = P \cdot (1{:}\, n)^T$. Let $\bar{A} = PAP^T$.*

1. *$\bar{A} = A(\pi, \pi)$.*
2. *The column etree $\bar{T}$ of $\bar{A}$ is isomorphic to T; in particular, relabeling each node i of $\bar{T}$ as $\pi(i)$ yields T.*

3. *Suppose in addition that $\bar{A}$ has an LU factorization without pivoting, $\bar{A} = \bar{L}\bar{U}$. Then $P^T\bar{L}P$ and $P^T\bar{U}P$ are, respectively, unit lower triangular and upper triangular, so $A = (P^T\bar{L}P)(P^T\bar{U}P)$ is also an LU factorization.*

Remark 2.3. Liu [35] attributes to F. Peters a result similar to part 3 for the symmetric positive definite case, concerning the Cholesky factor and the (usual, symmetric) elimination tree.

Proof. Part 1 is immediate from the definition of P. Part 2 follows from Corollary 6.2 in Liu [35], with the symmetric structure of the column intersection graph of our matrix A taking the place of Liu's symmetric matrix A. (Liu exhibits the isomorphism explicitly in the proof of his Theorem 6.1.)

Now we prove part 3. We have $a_{\pi(i)\pi(j)} = \bar{a}_{ij}$ for all i and j. Write $L = P^T\bar{L}P$ and $U = P^T\bar{U}P$, so that $l_{\pi(i)\pi(j)} = \bar{l}_{ij}$ and $u_{\pi(i)\pi(j)} = \bar{u}_{ij}$. Then $A = LU$; we need show only that L and U are triangular.

Consider a nonzero $u_{\pi(i)\pi(j)}$ of U. In the triangular factorization $\bar{A} = \bar{L}\bar{U}$, element $\bar{u}_{ij}$ is equal to $u_{\pi(i)\pi(j)}$ and is therefore nonzero. By part 3 of Theorem 2.1, then, j is an ancestor of i in $\bar{T}$. By the isomorphism between $\bar{T}$ and T, this implies that $\pi(j)$ is an ancestor of $\pi(i)$ in T. Then it follows from part 1 of Theorem 2.1 that $\pi(j) \geq \pi(i)$. Thus every nonzero of U is on or above the diagonal, so U is upper triangular. A similar argument shows that every nonzero of L is on or below the diagonal, so L is lower triangular. The diagonal elements of L are a permutation of those of $\bar{L}$, so they are all equal to 1. □

Since the triangular factors of A are just permutations of the triangular factors of PAP^T, they have the same sparsity. Indeed, they require the same arithmetic to compute; the only possible difference is the order of updates. If addition for updates is commutative and associative, this implies that with partial pivoting (i, j) is a legal pivot in $\bar{A}$ if and only if $(\pi(i), \pi(j))$ is a legal pivot in A. In floating-point arithmetic, the different order of updates could conceivably change the pivot sequence. Thus we have the following corollary.

COROLLARY 2.4. *Let π be a postorder on the column etree of A, let P_1 be any permutation matrix, and let P_2 be the permutation matrix with $\pi = P_2 \cdot (1{:}n)^T$. If $P_1AP_2^T = LU$ is an LU factorization, then so is $(P_2^TP_1)A = (P_2^TLP_2)(P_2^TUP_2)$. In exact arithmetic, the former is an LU factorization with partial pivoting of AP_2^T if and only if the latter is an LU factorization with partial pivoting of A.*

This corollary states that an LU code can permute the columns of its input matrix by postorder on the column etree, and then fold the column permutation into the row permutation on output. Thus our SuperLU code has the option of returning either four matrices P_1, P_2, L, and U (with $P_1AP_2^T = LU$), or just the three matrices $P_2^TP_1$, $P_2^TLP_2$, and $P_2^TUP_2$, which are a row permutation and two triangular matrices. The advantage of returning all four matrices is that the columns of each supernode are contiguous in L, which permits the use of a Level 2 BLAS supernodal triangular solve for the forward-substitution phase of a linear system solver. The supernodes are not contiguous in $P_2^TLP_2$.

2.4. Relaxed supernodes. We observe that, for most matrices, the average size of a supernode is only about 2 to 3 columns (though a few supernodes are much larger). A large percentage of supernodes consists of only a single column, many of which are leaves of the column etree. Therefore we have devised a scheme to merge groups of columns at the fringe of the etree into *artificial supernodes* regardless of their row structures. A parameter r controls the granularity of the merge. Our merge rule is: node i is merged with its parent node j when the subtree rooted at j has at

```
for column $j = 1$ to $n$ do
    $f = A(:, j)$;
    Symbolic factorization: determine which supernodes of $L$ will update $f$;
    Determine whether $j$ belongs to the same supernode as $j - 1$;
    for each updating supernode $(r{:}s) < j$ in topological order do
        Apply supernode-column update to column $j$:
            $f(r{:}s) = L(r{:}s, r{:}s)^{-1} \cdot f(r{:}s)$;  /* Now $f(r{:}s) = U(r{:}s, j)$ */
            $f(s + 1{:}n) = f(s + 1{:}n) - L(s + 1{:}n, r{:}s) \cdot f(r{:}s)$;
    end for;
    Pivot: interchange $f(j)$ and $f(m)$, where $|f(m)| = \max |f(j{:}n)|$;
    Separate $L$ and $U$: $U(1{:}j, j) = f(1{:}j)$;    $L(j{:}n, j) = f(j{:}n)$;
    Scale: $L(j{:}n, j) = L(j{:}n, j)/L(j, j)$;
    Prune symbolic structure based on column $j$;
end for;
```

FIG. 3.1. *LU factorization with supernode-column updates.*

most r nodes. In practice, the best values of r are generally between 4 and 8 and yield improvements in running time of 5% to 15%.

Artificial supernodes are a special case of *relaxed supernodes*, which Duff and Reid [15] and Ashcraft and Grimes [4] have used in the context of multifrontal methods for systems with symmetric nonzero structure. They allow a small number of zeros in the structure of any supernode, thus relaxing the condition that the columns must have strictly nested structures. It would be possible to use this idea in the unsymmetric code as well, though we have not experimented with it. Relaxed supernodes could be constructed either on the fly (by relaxing the nonzero count condition described in section 4.3 below), or by preprocessing the column etree to identify small subtrees that we would merge into supernodes.

3. Supernodal numeric factorization.

Now we show how to modify the col-col algorithm to use sup-col updates and supernode-panel updates. This section describes the numerical computation involved in the updates. Section 4 describes the symbolic factorization that determines which supernodes update which columns and also the boundaries between supernodes.

3.1. Sup-col updates.

Figure 3.1 sketches the sup-col algorithm. The only difference from the col-col algorithm is that all the updates to a column from a single supernode are done together. Consider a supernode $(r{:}s)$ that updates column j. The coefficients of the updates are the values from a segment of column j of U, namely $U(r{:}s, j)$. The nonzero structure of such a segment is particularly simple: all the nonzeros are contiguous, and follow all the zeros (as proved in Corollary 4.2, which appears in section 4.1). Thus, if k is the index of the first nonzero row in $U(r{:}s, j)$, the updates to column j from supernode $(r{:}s)$ come from columns k through s. Since the supernode is stored as a dense matrix, these updates can be performed by a dense lower triangular solve (with the matrix $L(k{:}s, k{:}s)$) and a dense matrix-vector multiplication (with the matrix $L(s + 1{:}n, k{:}s)$). As described in section 4, the symbolic phase determines the value of k, that is, the position of the first nonzero in the segment $U(r{:}s, j)$.

The advantages of using sup-col updates are similar to those in the symmetric case [37]. Efficient Level 2 BLAS matrix-vector kernels can be used for the triangular solve and matrix-vector multiply. Furthermore, all the updates from the supernodal

```
for column $j = 1$ to $n$ step $w$ do
    Symbolic factor: determine which supernodes will update any of $L(:, j{:}j + w - 1)$;
    for each updating supernode $(r{:}s) < j$ in topological order do
        for column $jj = j$ to $j + w - 1$ do
            Apply supernode-column update to column $jj$;
        end for;
    end for;
    Inner factorization:
        Apply the sup-col algorithm on columns and supernodes within the panel;
end for;
```

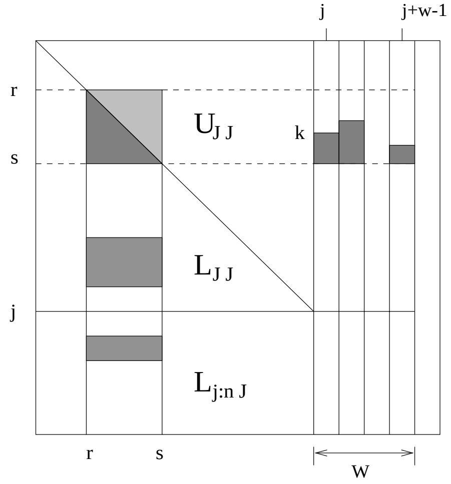


FIG. 3.2. *The supernode-panel algorithm, with columnwise blocking.* $J = 1{:}j - 1$.

columns can be collected in a dense vector before doing a single scatter into the target vector. This reduces the amount of indirect addressing.

3.2. Supernode-panel updates. We can improve the sup-col algorithm further on machines with a memory hierarchy by changing the data access pattern. The data we are accessing in the inner loop (lines 5–9 of Figure 3.1) include the destination column j and all the updating supernodes $(r{:}s)$ to the left of column j. Column j is accessed many times, while each supernode $(r{:}s)$ is used only once. In practice, the number of nonzero elements in column j is much less than that in the updating supernodes. Therefore, the access pattern given by this loop provides little opportunity to reuse cached data. In particular, the same supernode $(r{:}s)$ may be needed to update both columns j and $j{+}1$. But when we factor the $(j{+}1)$st column (in the next iteration of the outer loop), we will have to fetch supernode $(r{:}s)$ again from memory, instead of from cache (unless the supernodes are small compared to the cache).

3.2.1. Panels. To exploit memory locality, we factor several columns (say w of them) at a time in the outer loop, so that one updating supernode $(r{:}s)$ can be used to update as many of the w columns as possible. We refer to these w consecutive columns as a *panel* to differentiate them from a supernode; the row structures of these columns may not be correlated in any fashion, and the boundaries between panels may be different from those between supernodes. The new method requires rewriting the doubly nested loop as the triple loop shown in Figure 3.2. This is analogous to loop tiling techniques used in optimizing compilers to improve cache behavior for 2-D arrays with regular stride. It is also somewhat analogous to the sup-sup updates that Ng and Peyton [37] and Rothberg and Gupta [38] have used in symmetric Cholesky factorization.

The structure of each sup-col update is the same as in the sup-col algorithm. For each supernode $(r{:}s)$ to the left of column j, if $u_{kj} \neq 0$ for some $r \leq k \leq s$, then $u_{ij} \neq 0$ for all $k \leq i \leq s$. Therefore, the nonzero structure of the panel of U consists of dense column segments that are rowwise separated by supernodal boundaries, as in Figure 3.2. Thus, it is sufficient for the symbolic factorization algorithm to record only the first nonzero position of each column segment. As detailed in section 4.4, symbolic factorization is applied to all the columns in a panel at once, over all the updating supernodes, before the numeric factorization step.

In dense factorization, the entire supernode-panel update in lines 3–7 of Figure 3.2 would be implemented as two Level 3 BLAS calls: a dense triangular solve with w right-hand sides, followed by a dense matrix-matrix multiply. In the sparse case, this is not possible, because the different sup-col updates begin at different positions k within the supernode, and the submatrix $U(r{:}s, j{:}j + w - 1)$ is not dense. Thus the sparse supernode-panel algorithm still calls the Level 2 BLAS. However, we get similar cache benefits to those from the Level 3 BLAS, at the cost of doing the loop reorganization ourselves. Thus we sometimes call the kernel of this algorithm a "BLAS-$2\frac{1}{2}$" method.

In the doubly nested loop (lines 3–7 of Figure 3.2), the ideal circumstance is that all w columns in the panel require updates from supernode $(r{:}s)$. Then this supernode will be used w times before it is forced out of the cache. There is a trade-off between the value of w and the size of the cache. For this scheme to work efficiently, we need to ensure that the nonzeros in the w columns do not cause cache thrashing. That is, we must keep w small enough so that all the data accessed in this doubly nested loop fit in cache. Otherwise, the cache interference between the source supernode and the destination panel can offset the benefit of the new algorithm.

3.2.2. Outer and inner factorization. At the end of the supernode-panel update (line 7), columns j through $j + w - 1$ of L and U have received all their updates from columns to the left of j. We call this the *outer factorization.* What remains is to apply updates that come from columns within the panel. This amounts to forming the LU factorization of the panel itself (in columns $(j{:}j + w - 1)$ and rows $(j{:}n)$). This *inner factorization* is performed by the sup-col algorithm, almost exactly as shown in Figure 3.1. The inner factorization includes a columnwise symbolic factorization just as in the sup-col algorithm. The inner factorization also includes the supernode identification, partial pivoting, and symmetric structure reduction for the entire algorithm. Section 4 contains details of the inner factorization.

3.2.3. Reducing cache misses by rowwise blocking. Our first experiments with the supernode-panel algorithm showed speedups for some medium-sized problems of around 20–30%. However, the improvement for large matrices was often only a few percentage points. We now study the reasons and remedies for this.

To implement loops (lines 3–7 of Figure 3.2), we first expand the nonzeros of the panel columns of A into an n by w full working array, called the *sparse accumulator* [26] or SPA. This allows random access to the entries of the active panel. A temporary array stores the results of the BLAS operations, and the updates are scattered into the SPA. At the end of panel factorization, the data in the SPA are copied into storage for L and U. Although increasing the panel size w gives more opportunity for data

```
1.  for $j = 1$ to $n$ step $w$ do
2.      $\cdots$
3.      for each updating supernode $(r{:}s) < j$ in topological order do
4.          Apply triangular solves to $A(r{:}s, j{:}j + w - 1)$ using $L(r{:}s, r{:}s)$;
5.          for each row block $B$ in $L(s + 1{:}n, r{:}s)$ do
6.              for $jj = j$ to $j + w - 1$ do
7.                  Multiply $B \cdot U(r{:}s, jj)$, and scatter into SPA$(:, jj)$;
8.              end for;
9.          end for;
10.     end for;
11.     $\cdots$
12  end for;
```

FIG. 3.3. *The supernode-panel algorithm, with* 2*-D blocking.*

reuse, it also increases the size of the active data set that must fit into cache. The supernode-panel update loop accesses the following data:

- the nonzeros in the updating supernode $L(r{:}n, r{:}s)$;
- the SPA data structure, consisting of an n by w full array and a temporary store of size n.

By instrumenting the code, we found that the working sets of large matrices are much larger than the cache size. Hence, cache thrashing limits performance.

We experimented with a scheme suggested by Rothberg [39], in which the SPA has only as many rows as the number of nonzero rows in the panel (as predicted by symbolic factorization), and an extra indirection array of size n is used to address the SPA. Unfortunately, the cost incurred by double indirection is not negligible, and this scheme was not as effective as the 2-D blocking method we now describe.

We implemented a rowwise blocking scheme on top of the columnwise blocking in the supernode-panel update. The 2-D blocking adds another level of looping between the two loops in lines 3 and 4 of Figure 3.2. This partitions the supernodes (and the SPA structure) into block rows. Then each block row of the updating supernode is used for up to w partial matrix-vector multiplies, which are pushed all the way through into the SPA before the next block row of the supernode is accessed. The active data set accessed in the inner loops is thus much smaller than in the 1-D scheme. The 2-D blocking algorithm is organized as in Figure 3.3. The key performance gains come from the loops (lines 5–9), where each row block is reused as much as possible before the next row block is brought into the cache. The innermost loop is still a dense matrix-vector multiply, performed by a Level 2 BLAS kernel.

3.2.4. Combining 1-D and 2-D blocking. The 2-D blocking works well when the rectangular supernodal matrix $L(r{:}n, r{:}s)$ is large in both dimensions. If all of $L(r{:}n, r{:}s)$ fits in cache, then the rowwise blocking gives no benefit, but still incurs overhead for setting up loop variables, skipping the empty loop body, and so on. This overhead can be nearly 10% for some of the sparser problems. Thus we have devised a hybrid update algorithm that uses either the 1-D or 2-D partitioning scheme, depending on the size of each updating supernode. This decision is made at runtime, with the overhead limited to a one-line test. It turns out that this hybrid scheme works better than either 1-D or 2-D code for many problems. Therefore, this is the algorithm that we used in our code.

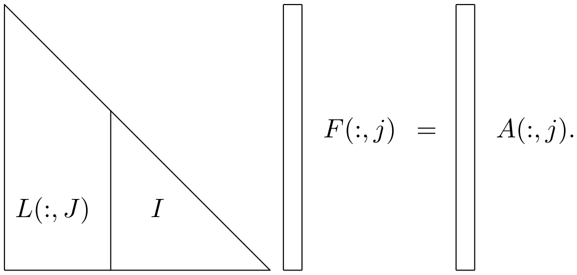
4. Symbolic factorization.

Symbolic factorization is the process that determines the nonzero structure of the triangular factors L and U from the nonzero structure of the matrix A. This in turn determines which columns of L update each column j of the factors (namely, those columns r for which $u_{rj} \neq 0$), and also determines which columns of L can be combined into supernodes.

Without numeric pivoting, the complete symbolic factorization can be performed before any numeric factorization. Partial pivoting, however, requires that the numeric and symbolic factorizations be interleaved. The sup-col algorithm performs symbolic factorization for each column just before it is computed, as described in section 4.1. The supernode-panel algorithm performs symbolic factorization for an entire panel at once, as described in section 4.4.

4.1. Column depth-first search.

From the numeric factorization algorithm, it is clear that the structure of column $F(:,j)$ depends on the structure of column $A(:,j)$ of the original matrix and on the structure of $L(:,J)$, where $J = 1{:}j{-}1$. Indeed, $F(:,j)$ has the same structure as the solution vector for the following triangular system [29]:



A straightforward way to compute the structure of $F(:,j)$ from the structures of $L(:,J)$ and $A(:,j)$ is to simulate the numerical computation. A less expensive way is to use the following characterization in terms of paths in the directed graph of $L(:,J)$.

For any matrix M, the notation $i \xrightarrow{M} j$ means that there is an edge from i to j in the directed graph of M, that is, $m_{ij} \neq 0$. Edges in the directed graph of M are directed from rows to columns. The notation $i \overset{M}{\Longrightarrow} j$ means that there is a directed path from i to j in the directed graph of M. Such a path may have length zero; that is, $i \overset{M}{\Longrightarrow} i$ always holds.

THEOREM 4.1 (see [24]). *f_{ij} is nonzero (equivalently, $i \xrightarrow{F} j$) if and only if $i \overset{L(:,J)}{\Longrightarrow} k \xrightarrow{A} j$ for some $k \leq i$.*

This result implies that the symbolic factorization of column j can be obtained as follows. Consider the nonzeros in $A(:,j)$ as a subset of the vertices of the directed graph $G = G(L(:,J)^T)$, which is the reverse of the directed graph of $L(:,J)$. The nonzero positions of $F(:,j)$ are then given by the vertices reachable by paths from this set in G. We use the graph of L^T here because of the convention that edges are directed from rows to columns. Since L is actually stored by columns, our data structure gives precisely the adjacency information for G. Therefore, we can determine the structure of column j of L and U by traversing G from the set of starting nodes given by the structure of $A(:,j)$.

The traversal of G determines the structure of $U(:,j)$, which in turn determines the columns of L that will participate in updates to column j in the numerical factorization. These updates must be applied in an order consistent with a topological ordering of G. We use depth-first search to perform the traversal, which makes it possible to generate a topological order (specifically, reverse postorder) on the nonzeros of $U(:, j)$ as they are located [29].

Another consequence of the path theorem is the following corollary. It states that if we divide each column of U into *segments*, one per supernode, then within each segment the column of U consists of a consecutive sequence of nonzeros. Thus we need only keep track of the position of the first nonzero in each segment.

COROLLARY 4.2. *Let* $(r{:}\,s)$ *be a supernode* (*of either type* T2 *or* T3) *in the factorization* $PA = LU$. *Suppose* u_{kj} *is nonzero for some* j *with* $r \leq k \leq s$. *Then* $u_{ij} \neq 0$ *for all* i *with* $k \leq i \leq s$.

Proof. Let $k \leq i \leq s$. Since $u_{kj} \neq 0$, we have $k \overset{L(:,J)}{\Longrightarrow} m \xrightarrow{A} j$ for some $m \leq k$ by Theorem 4.1. Now l_{ik} is in the diagonal block of the supernode, and hence is nonzero. Thus $i \xrightarrow{L(:,J)} k$, so $i \overset{L(:,J)}{\Longrightarrow} m \xrightarrow{A} j$, whence u_{ij} is nonzero by Theorem 4.1. □

4.2. Pruning the symbolic structure. We can speed up the depth-first search traversals by using a reduced graph in place of G, the reverse of the graph of $L(:, J)$. We have explored this idea in a series of papers [20, 21, 25]. Any graph H can be substituted for G, provided that $i \overset{H}{\Longrightarrow} j$ if and only if $i \overset{G}{\Longrightarrow} j$. The traversals are more efficient if H has fewer edges; but any gain in efficiency must be traded off against the cost of computing H.

An extreme choice of H is the elimination directed acyclic graph (*elimination dag*) [25], which is the transitive reduction of G, or the minimal subgraph of G that preserves paths. However, the elimination dag is expensive to compute. The *symmetric reduction* [20] is a subgraph that has (in general) fewer edges than G but more edges than the elimination dag, and that is much less expensive to compute. The symmetric reduction takes advantage of symmetry in the structure of the filled matrix F; if F is completely symmetric, it is just the symmetric elimination tree. The symmetric reduction of $L(:, J)$ is obtained by removing all nonzeros l_{rs} for which $l_{ts}u_{st} \neq 0$ for some $t < \min(r, j)$. Eisenstat and Liu [21] give an efficient method to compute the symmetric reduction during symbolic factorization and demonstrate experimentally that it significantly reduces the total factorization time when used in an algorithm that does col-col updates.

Our supernodal code uses symmetric reduction to speed up its symbolic factorization. Using the sample matrix in Figure 2.2, Figures 4.1 and 4.2 illustrate symmetric reduction in the presence of supernodes. We use S to represent the supernodal structure of $L(:, J)^T$ and R to represent the symmetric reduction of S. It is this R that we use in our code. Note that the edges of the graph of R are directed from columns of L to rows of L.

In the figures, the symbol "$\oslash$" indicates an entry in S that was pruned from R by symmetric reduction. The $(8, 2)$ entry was pruned due to the symmetric nonzero pair $(6, 2)$ and $(2, 6)$. The figure shows the current state of the reduced structure based on the first seven columns of the filled matrix.

It is instructive to follow this example through one more column to see how symbolic factorization is carried out in the reduced supernodal structure. Consider the symbolic step for column 8. Suppose a_{28} and a_{38} are nonzero. The other nonzeros in column 8 of the factor are generated by paths in the reduced supernodal structure (we show just one possible path for each nonzero):

$$8 \xrightarrow{A^T} 2 \xrightarrow{R} 6,$$

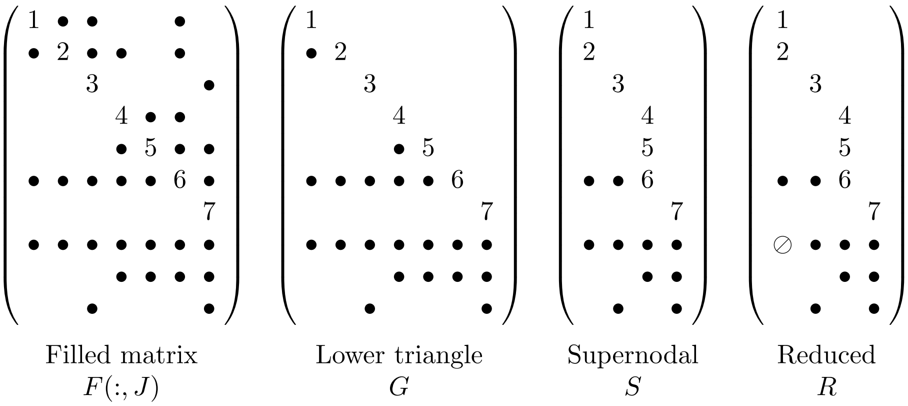


Fig. 4.1. *Supernodal and symmetrically reduced structures.*

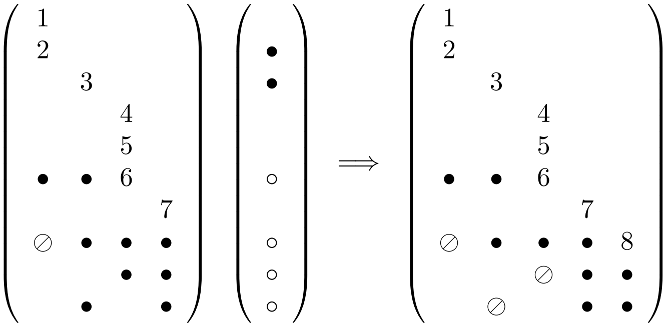


Fig. 4.2. *One step of symbolic factorization in the reduced structure.*

$$8 \xrightarrow{A^T} 3 \xrightarrow{R} 8,$$

$$8 \xrightarrow{A^T} 2 \xrightarrow{R} 6 \xrightarrow{R} 9,$$

$$8 \xrightarrow{A^T} 3 \xrightarrow{R} 10.$$

Figure 4.2 shows the reduced supernodal structure before and after column 8. In column 8 of A, the original nonzeros are shown as "•" and the fill nonzeros are shown as "∘". Once the structure of column 8 of U is known, more symmetric reduction is possible. The entry $l_{10,3}$ is pruned due to the symmetric nonzeros in l_{83} and u_{38}. Also, l_{96} is pruned by l_{86} and u_{68}. Figure 4.2 shows the new structure.

The supernodal symbolic factorization relies on the path characterization in Theorem 4.1 and on the path-preserving property of symmetric reduction. In effect, we use the modified path condition

$$i \xrightarrow{A^T} m \overset{R}{\Longrightarrow} j$$

on the symmetrically reduced supernodal structure R of $L(:, J)^T$.

4.3. Detecting supernodes. Since supernodes consist of contiguous columns of L, we can decide at the end of each symbolic factorization step whether the new column j belongs to the same supernode as column $j-1$.

For T2 supernodes, the test is straightforward. During symbolic factorization, we test whether $L(:,j) \subseteq L(:,j-1)$ (where the containment applies to the set of nonzero indices). At the end of the symbolic factorization step, we test whether $nnz(L(:,j)) = nnz(L(:,j-1))-1$. Column j joins column $(j-1)$'s supernode if and only if both tests are passed.

T3 supernodes also require the diagonal block of U to be full. To check this, it suffices to check whether the single element u_{rj} is nonzero, where r is the first column index of the supernode. This follows from Corollary 4.2: $u_{rj} \neq 0$ implies that $u_{ij} \neq 0$ for all $r \leq i \leq j$. Indeed, we can even omit the test $L(:,j) \subseteq L(:,j-1)$ for T3 supernodes. If $u_{rj} \neq 0$, then $u_{j-1,j} \neq 0$, which means that column $j-1$ updates column j, which implies $L(:,j) \subseteq L(:,j-1)$. Thus we have proved the following theorem.

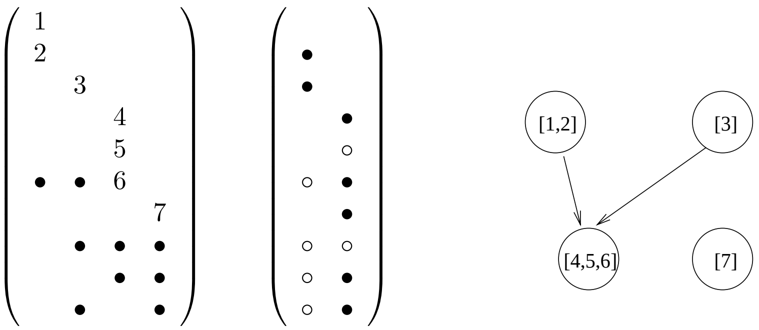
THEOREM 4.3. *Suppose a T3 supernode begins with column r and extends at least through column $j-1$. Column j belongs to this supernode if and only if $u_{rj} \neq 0$ and $nnz(L(:,j)) = nnz(L(:,j-1))-1$.*

For either T2 or T3 supernodes, it is straightforward to implement the relaxed versions discussed in section 2.4. Also, since the main benefits of supernodes come when they fit in the cache, we impose a maximum size for a supernode.

4.4. Panel depth-first search. The supernode-panel algorithm consists of an outer factorization (applying updates from supernodes to the active panel) and an inner factorization (applying sup-col updates within the active panel). Each has its own symbolic factorization. The outer symbolic factorization happens once per panel and determines two things: (1) a single column structure, which is the union of the structures of the panel columns, and (2) which supernodes update each column of the panel, and in what order. This is the information that the supernode-panel update loop in Figure 3.2 needs.

The inner symbolic factorization happens once for each column of the panel, interleaved column by column with the inner numeric factorization. In addition to determining the nonzero structure of the active column and which supernodes within the panel will update the active column, the inner symbolic factorization is also responsible for forming supernodes (that is, for deciding whether the active column will start a new supernode) and for symmetric structural pruning. The inner symbolic factorization is, therefore, exactly the sup-col symbolic factorization described above.

The outer symbolic factorization must determine the structures of columns j to $j+w-1$, i.e., the structure of the whole panel, and also a topological order for $U(1{:}j, j{:}j+w-1)$ en masse. To this end, we developed an efficient panel depth-first search scheme, which is a slight modification of the column depth-first search. The panel depth-first search algorithm maintains a single postorder depth-first search list for all w columns of the panel. Let us call this the PO list, which is initially empty. The algorithm invokes the column depth-search procedure for each column from j to $j+w-1$. In the column depth-first search, each time the search backtracks from a vertex, that vertex is appended to the PO list. In the panel depth-first search, however, the vertex is appended to the PO list *only if it is not already on the list.* This gives a single list that includes every position that is nonzero in any panel column, just once; and the entire list is in (reverse) topological order. Thus the order of updates specified by the list is acceptable for each of the w individual panel columns.



Reduced supernodal R $\quad$ $A(:, 8{:}9)$

Fig. 4.3. *The supernodal directed graph corresponding to* $L(1{:}7, 1{:}7)^T$.

We illustrate the idea in Figure 4.3, using the sample matrix from Figures 4.1 and 4.2, and a panel of width two. The first seven columns have been factored, and the current panel consists of columns 8 and 9. In the panel, nonzeros of A are shown as "•" and fill in F is shown as "○". The depth-first search for column 8 starts from vertices 2 and 3. After that search is finished, the panel postorder list is $PO = (6, 2, 3)$. Now the depth-first search for column 9 starts from vertices 6 and 7 (not 4, since 6 is the representative vertex for the supernode containing column 4). This depth-first search only appends 7 to the PO list, because 6 is already on the list. Thus, the final list for this panel is $PO = (6, 2, 3, 7)$. The postorder list of column 8 is $(6, 2, 3)$ and the postorder list of column 9 is $(6, 7)$, which are simply two sublists of the panel PO list. The topological order is the reverse of PO, or $(7, 3, 2, 6)$. In the loop of line 3 of Figure 3.2, we follow this topological order to schedule the updating supernodes and perform numeric updates to columns of the panel.

5. Evaluation.

In this section, we evaluate our algorithms using matrices from several applications and several sources. We compare the performance of SuperLU, our supernode-panel code, with its predecessors and with one other approach to sparse LU factorization.

5.1. Experimental setup.

Table 5.1 lists 23 matrices with some characteristics of their nonzero structures. Some of the matrices are from the Harwell–Boeing collection [17]. Many of the larger matrices are from the ftp site maintained by Tim Davis of the University of Florida.[3] Those matrices are as follows: Memplus is a circuit simulation matrix from Steve Hamm of Motorola. Rdist1 is a reactive distillation problem in chemical process separation calculations, provided by Stephen Zitney at Cray Research, Inc. Shyy161 is derived from a direct, fully coupled method for solving the Navier–Stokes equations for viscous flow calculations, provided by Wei Shyy of the University of Florida. Goodwin is a finite element matrix in a nonlinear solver for a fluid mechanics problem, provided by Ralph Goodwin of the University of Illinois at Urbana–Champaign. Venkat01, Inaccura, and Raefsky3/4 were provided by Horst Simon, then of NASA. Venkat01 comes from an implicit 2-D Euler solver for an unstructured grid in a flow simulation. Raefsky3 is from a fluid structure interaction turbulence problem. Raefsky4 is from a buckling problem for a container model.

[3] URL: http://www.cis.ufl.edu/~davis

Table 5.1

Characteristics of the test matrices. Structural symmetry s is the fraction of the nonzeros matched by nonzeros in symmetric locations. None of the matrices are numerically symmetric.

	Matrix	s	n	$nnz(A)$	$nnz(A)/n$
1	Memplus	.983	17758	99147	5.6
2	Gemat11	.002	4929	33185	6.7
3	Rdist1	.062	4134	9408	2.3
4	Orani678	.073	2529	90158	35.6
5	Mcfe	.709	765	24382	31.8
6	Lnsp3937	.869	3937	25407	6.5
7	Lns3937	.869	3937	25407	6.5
8	Sherman5	.780	3312	20793	6.3
9	Jpwh991	.947	991	6027	6.1
10	Sherman3	1.000	5005	20033	4.0
11	Orsreg1	1.000	2205	14133	6.4
12	Saylr4	1.000	3564	22316	6.3
13	Shyy161	.769	76480	329762	4.3
14	Goodwin	.642	7320	324772	44.4
15	Venkat01	1.000	62424	1717792	27.5
16	Inaccura	1.000	16146	1015156	62.9
17	Af23560	.947	23560	460598	19.6
18	Dense1000	1.000	1000	1000000	1000
19	Raefsky3	1.000	21200	1488768	70.2
20	Ex11	1.000	16614	1096948	66.0
21	Wang3	1.000	26064	177168	6.8
22	Raefsky4	1.000	19779	1316789	66.6
23	Vavasis3	.001	41092	1683902	41.0

Table 5.2

Machines used to compare various column LU codes. Column "#" is maximum number of instruction issues per clock cycle.

	Clock MHz	On-chip cache	External cache	#	Peak Mflops	dgemm Mflops	dgemv Mflops
IBM RS/6000-590	66.5	256 KB		6	266	250	235
SGI MIPS R8000	90	16 KB	4 MB	4	360	340	210
DEC Alpha 21164	300	8 KB-L1 96 KB-L2	4 MB	4	600	350	135
SUN UltraSparc-I	143	16 KB	512 KB	4	286	227	–

Af23560 is from solving an unsymmetric eigenvalue problem, provided by Zhaojun Bai of the University of Kentucky. Ex11 is from a three-dimensional (3-D) steady flow calculation in the SPARSKIT collection maintained by Youcef Saad at the University of Minnesota. Wang3 is from solving a coupled nonlinear PDE system in a 3-D ($30 \times 30 \times 30$ uniform mesh) semiconductor device simulation, as provided by Song Wang of the University of New South Wales, Sydney. Vavasis3 is an unsymmetric augmented matrix for a 2-D PDE with highly varying coefficients [44]. Dense1000 is a dense 1000×1000 random matrix.

The matrices are sorted in increasing order of $flops/nnz(F)$, the ratio of the number of floating-point operations to the number of nonzeros $nnz(F)$ in the factored matrix $F = U + L - I$. The reason for this order will be described in more detail in section 5.4.

This paper does not address the performance of column preordering for sparsity. We simply use the existing ordering algorithms provided by Matlab [26]. For all matrices, except 1, 15, and 21, the columns were permuted by Matlab's minimum degree ordering of A^TA, also known as "column minimum degree." However, this ordering produces excessive fill for matrices 1, 15, and 21, because it attempts only to minimize the upper bound on the actual fill, and the upper bounds are too loose in these cases. When these three matrices are symmetrically permuted by Matlab's symmetric minimum degree ordering on $A + A^T$, the amount of fill is much smaller than using column minimum degree ordering.

We conducted performance analysis on high-end workstations from four vendors (IBM, SGI, DEC, and SUN). Some characteristics of these machines are tabulated in Table 5.2. The instruction caches, if separate from the data cache, are not listed in the table. In most cases, the on-chip L1 caches are fairly small, so we use either the L2 cache or the off-chip cache as a reference. The DGEMM and DGEMV Mflop rates were measured using vendor-supplied BLAS libraries. (Exception: SUN does not supply a BLAS library, so we report the DGEMM speed from PHiPAC [6]. PHiPAC does not include DGEMV.) Our UltraSparc-I has less physical memory than the other machines, so some large problems could not be tested on this machine.

5.2. Performance of SuperLU on an IBM RS/6000-590. Table 5.3 presents the performance of SuperLU on this system. The CPU clock rate is 66.5 MHz. The processor has two branch units, two fixed-point units, and two floating-point units, which can all operate in parallel if there are no dependencies. Each FPU can perform two operations (a multiply and an add or subtract) in each cycle. Thus, the peak floating-point performance is 266 Mflops. The size of the main memory is 768 MB. SuperLU is implemented in C; we used the AIX xlc compiler with -O3 optimization. All floating-point computations are performed in double precision.

In the inner loops of our sparse code, we call the two Level 2 BLAS routines DTRSV (triangular solve) and DGEMV (matrix-vector multiply) provided in the IBM ESSL library [32], whose BLAS-3 matrix-matrix multiply routine (DGEMM) achieves about 250 Mflops when the dimension of the matrix is larger than 60 [1]. In our sparse algorithm, we find that DGEMV typically accounts for more than 80% of the floating-point operations. As shown in the second to last column of Table 5.3, this percentage is 95% higher than for many matrices. Our measurements reveal that for typical dimensions arising from the benchmark matrices, DGEMV achieves roughly 235 Mflops if the data are from cache. If the data are fetched from main memory, this rate can drop by a factor of 2 or 3.

The BLAS speed is clearly an upper bound on the overall factorization rate. However, because symbolic manipulation usually takes a nontrivial amount of time, this bound could be much higher than the actual sparse code performance. The last column in Table 5.3 presents the percentage of the total execution time spent in numeric computation. For matrices 1 and 2, the program spent less than 35% of its time in the numeric part. Compared to the others, these two matrices are sparser, have less fill, and have smaller supernodes, so our supernodal techniques are less applicable. Matrix 2 is also highly unsymmetric, which makes the symmetric structural reduction less effective. However, it is important to note that the execution times for these two matrices are small.

For larger and denser matrices such as 18–23, the algorithm achieves between 110 and 125 Mflops, which is about half of the machine peak. These matrices take much longer to factor, which could be a serious bottleneck in an iterative simulation process. Our techniques are successful in reducing the solution times for this type of problem.

Table 5.3
Performance of SuperLU on an IBM RS/6000-590.

				#flops	Time		% flops	% num
	Matrix	$nnz(F)$	$\frac{nnz(F)}{nnz(A)}$	(10^6)	(sec)	Mflops	dgemv	time
1	Memplus	140388	1.4	1.8	0.57	3.08	70	16
2	Gemat11	93370	2.8	1.5	0.27	5.64	82	33
3	Rdist1	338624	3.6	12.9	0.96	13.47	85	48
4	Orani678	280788	3.1	14.9	1.11	13.48	98	51
5	Mcfe	69053	2.8	4.1	0.24	17.42	96	54
6	Lnsp3937	427600	16.8	38.9	1.50	25.97	95	48
7	Lns3937	449346	17.7	44.8	1.65	27.16	96	48
8	Sherman5	249199	12.0	25.2	0.82	30.78	93	57
9	Jpwh991	140746	23.4	18.0	0.52	34.57	94	58
10	Sherman3	433376	21.6	60.6	1.37	44.24	85	56
11	Orsreg1	402478	28.5	59.8	1.21	49.42	87	50
12	Saylr4	654908	29.3	104.8	2.18	48.07	87	57
13	Shyy161	7634810	23.2	1571.6	25.42	61.83	88	57
14	Goodwin	3109585	9.6	665.1	12.55	52.99	92	63
15	Venkat01	12987004	7.6	3219.9	42.99	74.90	91	63
16	Inaccura	9941478	9.8	4118.7	67.73	60.81	96	64
17	Af23560	13986992	30.4	6363.7	75.91	83.83	92	73
18	Dense1000	1000000	1.0	666.2	5.68	117.28	93	72
19	Raefsky3	17544134	11.8	12118.7	107.60	112.62	94	77
20	Ex11	26207974	23.8	26814.5	247.05	108.54	95	81
21	Wang3	13287108	74.9	14557.5	116.58	124.86	96	81
22	Raefsky4	26678597	20.3	31283.4	263.13	118.89	97	83
23	Vavasis3	49192880	29.2	89209.3	786.94	113.36	98	80

For a dense 1000 × 1000 matrix, our code achieves 117 Mflops. This may be compared to 168 Mflops reported in the LAPACK manual [3] on a matrix of this size, and 236 Mflops reported in the online Linpack benchmark files [36].

5.3. Comparison with previous column LU algorithms. In this section, we compare the performance of SuperLU with three of its predecessors, including the column code GP by Gilbert and Peierls [29] (Figure 1.1), GP-Mod by Eisenstat and Liu [21] (section 4.2), and SupCol by Eisenstat, Gilbert, and Liu [19] (Figure 3.1). GP and GP-Mod were written in Fortran. SupCol was first written in Fortran, and later translated literally into C; no changes in algorithms or data structures were made in this translation. SuperLU is written in C. (Matlab contains C implementations of GP and GP-Mod [26], which we did not test here.)

For the Fortran codes, we use Fortran 77 compilers; for the C codes, we use ANSI C compilers. In all cases, we use highest possible optimization provided by each compiler. Both SupCol and SuperLU call Level 2 BLAS routines. For the RS/6000-590, we use the BLAS routines from IBM's ESSL library. For the DEC Alpha, we use the BLAS routines from DEC's DXML library. There are no vendor-supplied BLAS libraries on the Sparc, so we use our own routines implemented in C.

Tables 5.4 through 5.7 present the results of comparisons on the four machines. The blocking parameters w, t, and b (Figure 5.6) for SuperLU are chosen according to the size of the data cache (Table 5.2) and are reported in each comparison table. In all these tables, the column labeled "GP" gives the raw factorization time in seconds of GP. The numbers in each successive column are speedups achieved by the corresponding enhancement over GP. Thus, for example, a speedup of 2 means that the running time was half that of GP. The numbers in the last two rows of each table

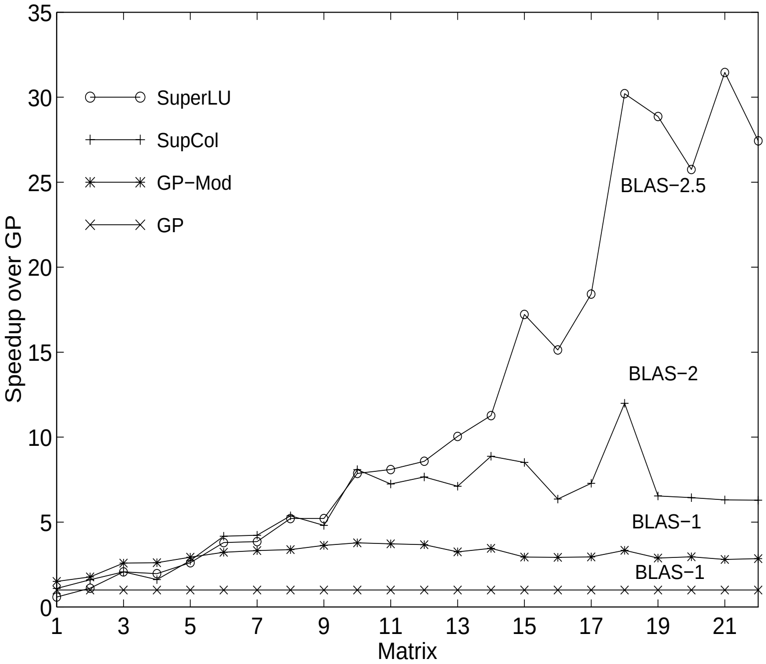


Fig. 5.1. *Speedups of each enhancement over GP on the MIPS R*8000.

show the average speedup and its standard deviation.

Figure 5.1 gives a visual look at the speedups for SuperLU over its predecessors on one of the workstations we experimented with (using data from Table 5.5). Of course, machines with different memory architectures would give different plots.

We make the following observations from these results:

- The symmetric structure pruning in GP-Mod is very effective in reducing the graph search time. This significantly decreases the symbolic factorization time in the GP code. It achieves speedup for all problems, on all machines. Its average speedup on the RS/6000-590 is 3.64, the highest among all the machines.
- Supernodes in SupCol restrict the search to the supernodal graph and allow the numeric kernels to employ dense BLAS-2 operations. The effects are not as dramatic as the pruning technique. For some matrices, such as 1–3, the runtimes are actually longer than GP-Mod. This is because supernodes are often small in the sparser matrices.
- Supernode-panel updates in SuperLU reduce the cache miss rate and exploit dense substructures in the factor F. For problems without much structure, the gain is often offset by various overheads. However, the advantage of SuperLU over SupCol becomes significant for larger or denser problems, or on machines with small cache, such as Alpha 21164, on which SuperLU achieves more than a factor of 2 speedup over SupCol for the six large matrices 18–23.

With more and more sophisticated techniques introduced, the added complications of the code increase the runtime overhead to some extent. This overhead can show up prominently in small or very sparse problems. The two supernodal codes are particularly sensitive to the characteristics of the problems. This can be seen from the

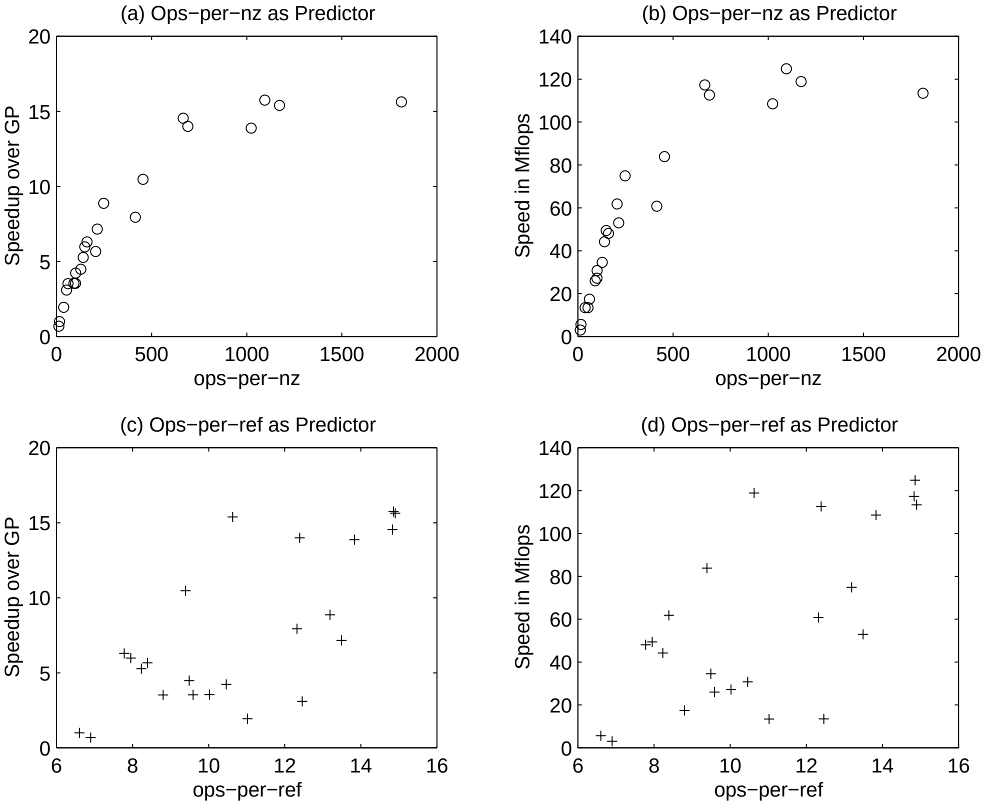


Fig. 5.2. *Matrix characteristics as predictors of performance.*

large standard deviations of their average speedups.

5.4. Understanding cache behavior and parameters. We now analyze the behavior of SuperLU in detail. We wish to understand when our algorithm is significantly faster than other algorithms. We would like performance-predicting variable(s) that depend on "intrinsic" properties of the problem, such as the sparsity structure, rather than algorithmic details and machine characteristics.

5.4.1. How much cache reuse can we expect? As discussed in section 3.2, the supernode-panel algorithm gets its primary gains from improved data locality by reusing a cached supernode several times. To understand how much cache reuse we can

hope for, we computed two statistics: *ops-per-nz* and *ops-per-ref*. After defining these statistics carefully, we discuss which more successfully measures reuse.

Ops-per-nz is simply calculated as $\#flops/nnz(F)$, the total number of floating-point operations per nonzero in the filled matrix F. If there were perfect cache behavior, i.e., one cache miss per data item (ignoring the effect of cache line size), then *ops-per-nz* would exactly measure the amount of work per cache miss. In reality, *ops-per-nz* is an upper bound on the reuse. Note that *ops-per-nz* depends only on the fact that we are performing Gaussian elimination with partial pivoting, not on implementation or machine details. *Ops-per-nz* is a natural measure of potential data reuse, because it has long been used to distinguish among the different levels of BLAS.

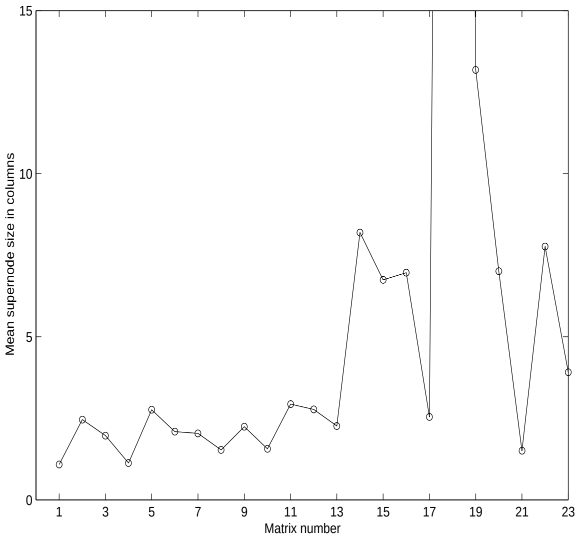


FIG. 5.3. *Mean number of columns per supernode.*

In contrast, *ops-per-ref* provides a lower bound on cache reuse and does depend on the details of the SuperLU algorithm. *Ops-per-ref* looks at each supernode-panel update separately and assumes that all the associated data are outside the cache before beginning the update. This pessimistic assumption limits the potential reuse to twice the panel size, $2\,w$.

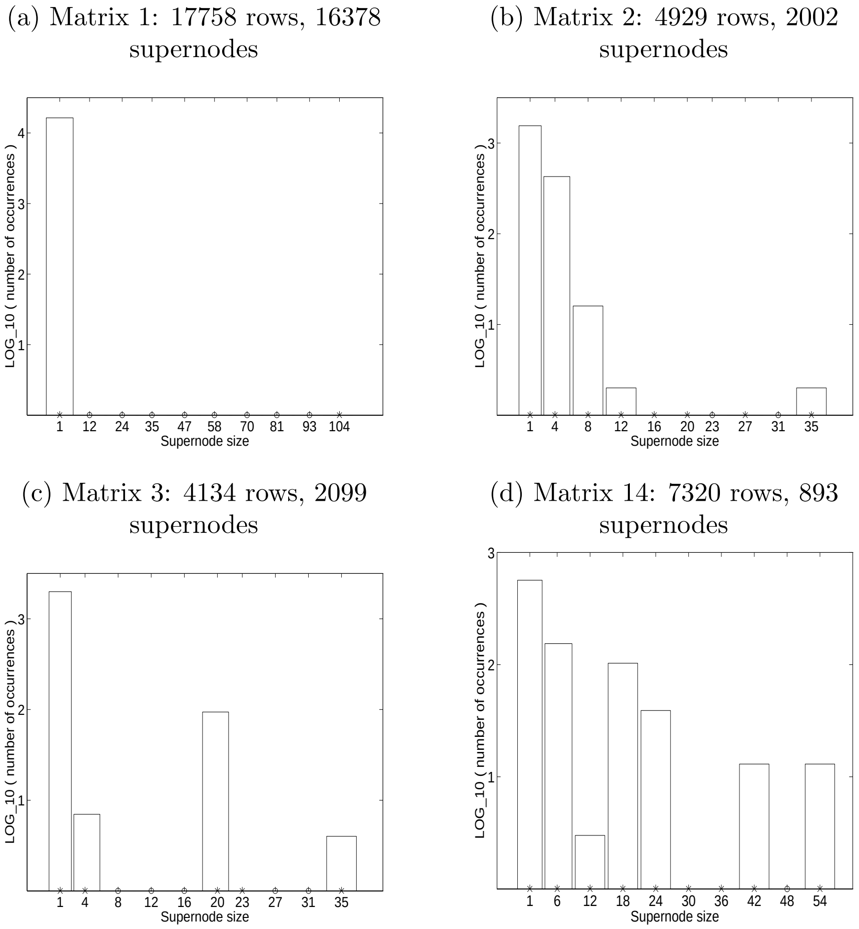
Now we describe how we compute the average *ops-per-ref* for the entire factorization. Consider a single update from supernode $(r\colon s)$ to panel $(j\colon j+w-1)$. We assume that the supernode entry is brought into cache from main memory exactly once for the entire supernode-panel update, if it is used at all. Thus, during a single supernode-panel update, each entry accessed in the updating supernode accounts for between 2 and $2\,w$ *operations per reference.* Define *kmin* to be the number of the first row containing a nonzero in the panel,

$$kmin = \min_{j \le jj < j+w} \{k \mid k = \min_{r \le i \le s} \{i \mid A(i, jj) \neq 0\}\} \;.$$

Then $nnz(L(r\colon n, kmin\colon s))$ entries of the supernode are referenced in the supernode-panel update. The dense triangular solve in column jj of the update takes $(s-k+1)\cdot(s-k)$ flops. The matrix-vector multiply takes $2\cdot(s-k+1)\cdot nnz(L(s+1\colon n, s))$ flops. (We count both additions and multiplications.) For all panel updates, we sum the memory reference counts and the flop counts, then divide the second sum by the first to arrive at an average *ops-per-ref. Ops-per-ref* ranges from 2 to $2\,w$, with larger values indicating better cache use.

Figure 5.2 plots these two statistics against the speedup SuperLU achieved over the col-col code GP and against SuperLU's raw execution rate. It is clear that (perhaps surprisingly) *ops-per-nz* is superior to *ops-per-ref* as a predictor of either of these measures of performance. This is good news, because *ops-per-nz* measures the best case reuse, and *ops-per-ref* the worst case. But neither statistic captures all the variation in the performance.

5.4.2. How large are the supernodes? The supernode size determines the


(a) Matrix 1: 17758 rows, 16378 supernodes

(b) Matrix 2: 4929 rows, 2002 supernodes

(c) Matrix 3: 4134 rows, 2099 supernodes

(d) Matrix 14: 7320 rows, 893 supernodes

Fig. 5.4. *Distribution of supernode size for four matrices.*

size of the matrix passed to matrix-vector multiply and other Level 2 BLAS routines. Figure 5.3 shows the average number of columns in the supernodes of the matrices after amalgamating the relaxed supernodes at the bottom of the column etree (section 2.4). The average size is usually quite small.

More important than average size is the distribution of supernode sizes. In sparse Gaussian elimination, more fill tends to occur in the later stages. Usually there is a large percentage of small supernodes corresponding to the leaves of the column etree, even after amalgamation. Larger supernodes appear near the root. In Figure 5.4 we plot the histograms of the supernode size for four matrices chosen to exhibit a wide range of behavior. In the figure, the number at the bottom of each bar is the smallest supernode size in that bin. The mark "o" at the bottom of a bin indicates zero occurrences; otherwise, a "$*$" is put at the bottom of a bin. Relaxed supernodes of granularity $r = 4$ are used. Matrix 1 has 16378 supernodes, all but one of which have less than 12 columns; the single large supernode, with 115 columns, is the dense submatrix at the bottom right corner of F. Matrix 14 has more supernodes distributed over a wider spectrum; it has 13 supernodes with 54 to 59 columns. This matrix shows greater speedups over the nonsupernodal codes.

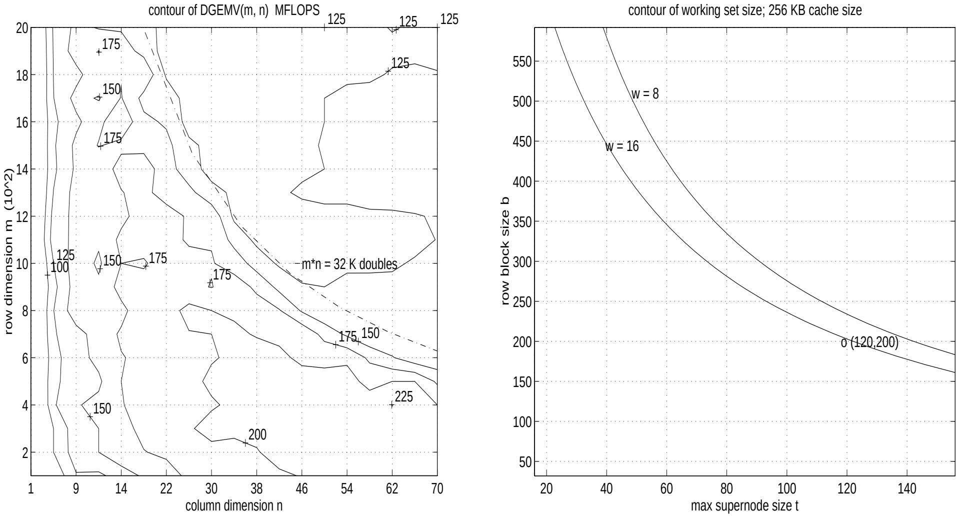


FIG. 5.5. (a) *Contour plot of* DGEMV *performance.* (b) *Contour plot of working set in* 2*-D algorithm.*

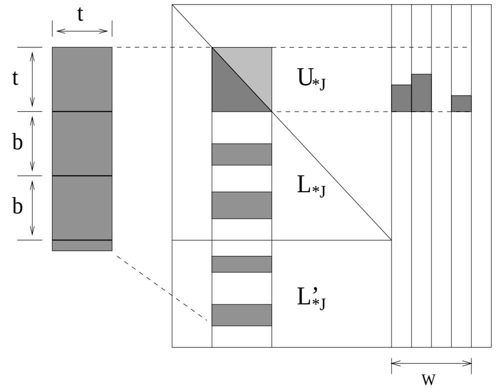


FIG. 5.6. *Parameters of the working set in the* 2*-D algorithm.*

5.4.3. Blocking parameters. In our hybrid algorithm (section 3.2.4), we need to select appropriate values for the parameters that describe the 2-D data blocking: panel width w, maximum supernode size t, and row block size b. The key considerations are that the active data we access in the inner loop (the *working set*) should fit into the cache, and that the matrices presented to the Level 2 BLAS routine DGEMV should be the sizes and shapes for which that routine is optimized. Here we describe in detail the methodology we used to choose parameters for the IBM RS/6000-590.

- **DGEMV optimization.** As indicated in the last column of Table 5.3, the majority of the floating-point operations are in the matrix-vector multiply. The dimensions (m, n) of the matrices in calls to DGEMV vary greatly depending on the supernode dimensions. Very often, the supernode is a tall and skinny matrix, that is, $m \gg n$. We measured the DGEMV Mflops rate for various m and n and present a contour plot in the (m, n) plane in Fig-

Table 5.4

*Speedups achieved by each enhancement over GP on the RS/*6000–590*. The blocking parameters for SuperLU are* $w = 8, t = 100$*, and* $b = 200$.

	Matrix	GP (Seconds)	GP-Mod	SupCol	SuperLU
1	Memplus	0.40	1.48	1.05	0.68
2	Gemat11	0.27	1.69	1.29	1.00
3	Rdist1	1.90	2.75	2.24	1.94
4	Orani678	13.86	3.55	2.98	3.10
5	Mcfe	1.55	3.44	3.52	3.52
6	Lnsp3937	7.11	3.39	3.86	3.54
7	Lns3937	7.77	3.39	3.85	3.55
8	Sherman5	3.98	3.43	4.57	4.23
9	Jpwh991	2.78	3.61	4.21	4.48
10	Sherman3	7.43	3.54	5.99	5.27
11	Orsreg1	8.73	3.64	5.86	5.98
12	Saylr4	17.51	3.67	5.99	6.30
13	Shyy161	163.14	3.65	6.46	5.67
14	Goodwin	90.63	3.84	6.46	7.16
15	Venkat01	355.50	3.86	8.33	8.87
16	Inaccura	544.91	4.17	7.24	7.94
17	Af23560	823.47	4.23	9.58	10.47
18	Dense1000	83.48	4.21	10.22	14.54
19	Raefsky3	1571.63	4.30	11.54	14.00
20	Ex11	3439.41	4.36	11.42	13.87
21	Wang3	1841.27	4.34	12.23	15.75
22	Raefsky4	3968.16	4.35	11.89	15.39
23	Vavasis3	12342.97	4.79	13.11	15.63
Mean			3.64	6.67	7.52
Std			0.79	3.69	5.04

ure 5.5(a). Each contour represents a constant Mflops rate. The dashed curve represents $mn = 32$K double reals, or a cache capacity of 256 Kbytes. In the optimum region, we achieve more than 200 Mflops; outside this region, performance drops either because the matrices exceed the cache capacity or because the column dimension n is too small.

- **Working set.** By studying the data access pattern in the inner loop of the 2-D algorithm, lines 6–8 in Figure 3.3, we find that the working set size is the following function of w, t, and b, as shown in Figure 5.6:

$$WS = \underbrace{b \times t}_{\text{supernode rows}} + \underbrace{(t + b) \times w}_{\text{DGEMV vectors}} + \underbrace{b \times w}_{\text{SPA rows}} .$$

  In Figure 5.5(b), we fix two w values and plot the contour lines for $WS = 32$K in the (t, b) plane. If the point (t, b) is below the contour curve, then the working set can fit in a cache of 32K double reals, or 256 Kbytes.

Based on this analysis, we use the following rules to set the parameters.

First we choose w, the width of the panel in columns. Larger panels mean more reuse of cached data in the outer factorization, but also mean that the inner factorization (by the sup-col algorithm) must be applied to larger matrices. We find empirically that the best choice for w is between 8 and 16. Performance tends to degrade for larger w.

Next we choose b, the number of rows per block, and t, the maximum number of columns in a supernode. Recall that b and t are upper bounds on the row and column

TABLE 5.5

Speedups achieved by each enhancement over GP on the MIPS R8000. The blocking parameters for SuperLU are $w = 16, t = 100$, and $b = 800$.

	Matrix	GP (Seconds)	GP-Mod	SupCol	SuperLU
1	MEMPLUS	0.42	1.51	1.10	0.59
2	GEMAT11	0.29	1.77	1.61	1.11
3	RDIST1	2.03	2.58	2.07	2.07
4	ORANI678	2.26	2.61	1.61	1.96
5	MCFE	0.60	2.93	2.73	2.61
6	LNSP3937	5.13	3.23	4.17	3.80
7	LNS3937	5.74	3.32	4.22	3.85
8	SHERMAN5	3.70	3.38	5.37	5.22
9	JPWH991	2.50	3.63	4.81	5.21
10	SHERMAN3	8.73	3.78	8.08	7.87
11	ORSREG1	8.18	3.72	7.24	8.10
12	SAYLR4	14.92	3.67	7.65	8.58
13	SHYY161	235.77	3.24	7.11	10.04
14	GOODWIN	103.66	3.45	8.87	11.27
15	VENKAT01	524.46	2.95	8.51	17.22
16	INACCURA	720.86	2.93	6.36	15.13
17	AF23560	1095.30	2.95	7.28	18.42
18	DENSE1000	113.28	3.34	11.99	30.21
19	RAEFSKY3	2263.80	2.88	6.54	28.87
20	EX11	5302.74	2.96	6.44	25.75
21	WANG3	2710.19	2.80	6.31	31.46
22	RAEFSKY4	6005.72	2.85	6.29	27.44
Mean			3.02	5.74	12.13
Std			0.57	2.75	10.48

dimensions of the call to DGEMV. We choose $t = 120$ and $b \approx 200$, which guarantees that the working set fits in cache (per Figure 5.5(b)), and that we can hope to be near the optimum region of DGEMV performance (per Figure 5.5(a)).

Recall that b is relevant only when we use rowwise blocking. This implies that the 2-D scheme adds overhead only if the updating supernode is small. In the actual code, the test for a large supernode is

if $ncol > 40$ **and** $nrow > b$ **then** the supernode is large,

where $nrow$ is the number of dense rows below the diagonal block of the supernode since $ncol$ is the number of dense columns of the supernode updating the panel. In practice, this choice usually gives the best performance.

The best choice of the parameters w, t, and b depends on the machine architecture and on the BLAS implementation, but it is largely independent of the matrix structure. Thus we do not expect each user of SuperLU to choose values for these parameters. Instead, our library code provides an inquiry function that returns the parameter values, much in the spirit of the LAPACK environment routine ILAENV. The machine-independent defaults will often give satisfactory performance. The methodology we have described here for the RS/6000 can serve as a guide for users who want to modify the inquiry function to give optimal performance for particular computer systems.

5.5. Comparison between SuperLU and multifrontal factorization. A number of codes for solving unsymmetric linear systems are available at the time of this writing, differing along several axes: emphasizing unsymmetric versus symmetric nonzero structure; using direct versus iterative methods; intended to be robust for

Table 5.6

Speedups achieved by each enhancement over GP on the Alpha 21164. *The blocking parameters for SuperLU are $w = 16, t = 50$, and $b = 100$.*

	Matrix	GP (Seconds)	GP-Mod	SupCol	SuperLU
1	Memplus	0.17	1.25	1.01	0.45
2	Gemat11	0.13	1.54	1.26	0.84
3	Rdist1	0.80	1.76	1.77	1.45
4	Orani678	0.92	1.74	1.47	1.45
5	Mcfe	0.24	1.71	2.01	1.85
6	Lnsp3937	2.09	1.93	2.61	2.27
7	Lns3937	2.33	1.94	2.59	2.27
8	Sherman5	1.50	1.92	3.13	3.00
9	Jpwh991	1.06	2.14	3.20	3.20
10	Sherman3	3.65	2.10	4.06	3.93
11	Orsreg1	3.41	2.07	3.87	3.91
12	Saylr4	6.73	2.05	4.01	4.34
13	Shyy161	102.19	1.81	3.97	4.58
14	Goodwin	46.18	1.92	3.84	4.90
15	Venkat01	235.01	1.71	4.08	7.00
16	Inaccura	333.24	1.72	3.48	6.07
17	Af23560	497.36	1.68	4.03	7.45
18	Dense1000	49.29	1.82	4.82	10.38
19	Raefsky3	1065.88	1.68	4.00	10.02
20	Ex11	1563.17	1.73	4.12	10.61
21	Wang3	1324.79	1.74	3.92	11.06
22	Raefsky4	2939.42	1.73	3.96	10.36
23	Vavasis3	9477.62	1.83	4.51	11.48
Mean			1.80	3.29	5.34
Std			0.20	1.10	3.69

general problems versus efficient for specific applications; and in the public domain versus subject to commercial restrictions. A comprehensive comparison of all the codes against all possible metrics would be valuable but is not the purpose of the present paper. Rather, to locate the performance of SuperLU in the constellation of linear solvers, we compare it in detail with one alternative: UMFPACK version 2.1 [7, 8, 9]. This is a modern code that, like SuperLU, emphasizes unsymmetric structure and robustness for general problems. (A recent report [30] compares SuperLU and GP with some unsymmetric iterative algorithms.)

UMFPACK uses a multifrontal algorithm. Where the outer loop of a left-looking algorithm like SuperLU is over columns (or panels of columns) of the factors being computed, the outer loop of a multifrontal algorithm is over pivots (or blocks of pivots) being eliminated. All the updates created when a block is eliminated are computed at once and stored as a dense *update matrix*. Before a block of pivots is eliminated, all the update matrices contributing to that block are summed into a *frontal matrix*. The elimination step can use Level 2 or Level 3 BLAS because the arithmetic is carried out on the dense frontal matrix. Some extra intermediate storage is needed to record update matrices that have not yet been assembled into frontal matrices, and some extra data movement is needed for the assembly. UMFPACK does not use a column preordering; rather, it chooses row and column pivots to balance considerations of stability and sparsity by using approximate Markowitz counts with a pivot threshold. In principle, the pivot threshold can lead to a less accurate solution than strict partial pivoting; in practice, the lost accuracy can usually be retrieved by iterative refinement of the solution. In principle, the freedom to choose both row and column pivots dy-

Table 5.7

*Speedups achieved by each enhancement over GP on the UltraSparc-*I*. The blocking parameters for SuperLU are $w = 8, t = 100$, and $b = 400$.*

	Matrix	GP (Seconds)	GP-Mod	SupCol	SuperLU
1	Memplus	0.36	1.17	1.08	0.58
2	Gemat11	0.23	1.27	1.16	0.93
3	Rdist1	1.53	1.69	1.56	1.46
4	Orani678	1.86	1.64	1.25	1.33
5	Mcfe	0.52	1.97	1.85	1.92
6	Lnsp3937	4.26	1.86	2.16	2.24
7	Lns3937	4.89	1.94	2.11	2.33
8	Sherman5	3.15	1.94	2.28	3.03
9	Jpwh991	2.32	2.18	2.47	3.09
10	Sherman3	7.73	2.01	2.84	3.59
11	Orsreg1	7.2	1.97	2.69	3.52
12	Saylr4	13.88	1.96	2.52	3.84
13	Shyy161	188.72	1.91	3.01	3.43
14	Goodwin	89.30	1.89	2.62	4.41
18	Dense1000	94.77	2.05	3.33	4.25
Mean			1.83	2.19	2.66
Std			0.28	0.69	1.22

namically could lead to sparser factors than strict partial pivoting; in practice, some matrices have sparser factors by one method and some by the other.

We compared UMFPACK and SuperLU on a group of 45 structurally unsymmetric matrices from a variety of applications, as described in Table 5.8. (This is a more comprehensive test set than the one we used in the earlier experiments with other left-looking codes described above.) We performed the experiments on the IBM RS/6000-590 described earlier. UMFPACK is written in Fortran; we compiled it with the AIX xlf compiler with -O3 optimization and linked it with the IBM ESSL library for BLAS calls. We used the parameter settings recommended by UMFPACK's authors [7].

UMFPACK does not include an initial column ordering step. For the initial column ordering in SuperLU, we ran Liu's multiple minimum degree algorithm [34] on the structure of $A^T A$. We report times for ordering and factorization separately. In applications where many matrices with the same nonzero structure but different values are factored, the cost of column ordering can be amortized over all the factorizations; in applications where only a single matrix is to be factored, preordering is part of the solution cost.

Table 5.9 gives time requirements and Table 5.10 gives memory requirements for the two codes on the matrices from the test set. The memory requirement we report includes only the memory actually used for the factorization, including working storage. Figures 5.7 and 5.8 summarize the comparison; each figure plots the relative time requirements against the relative space requirements for the two codes. Column preordering time is omitted in Figure 5.7 and included in Figure 5.8.

Neither code always dominates the other in either storage cost or time. Somewhat surprisingly, for 24 of the 45 matrices, the dynamic fill-reducing approach in UMFPACK seems to be less effective than the static preordering. SuperLU uses less memory for 60% of the matrices. When ordering time is not counted, SuperLU takes less time for 77% of the matrices. When ordering time is included, SuperLU takes less time for 44% of the matrices. For some matrices, such as memplus and orani678, the

Table 5.8

Characteristics of the unsymmetric matrices. StrSym is the fraction of nonzeros matched by nonzeros in symmetric locations. NumSym is the fraction of nonzeros matched by equal values in symmetric locations.

Matrix	n	$nnz(A)$	StrSym	NumSym	Discipline
cry10000	10000	49699	.9979	.2012	crystal growth simulation
memplus	17758	99147	.9830	.5864	circuit simulation
dw8192	8192	41746	.9699	.9320	square Dielectric waveguide
garon2	13535	390607	.9542	.6528	2D FEM, Navier-Stokes
jpwh_991	991	6027	.9469	.9469	circuit physics
af23560	23560	460598	.9465	.0511	eigenvalue problem
bramley2	17933	1021849	.9257	.0466	nonlinear CFD
bramley1	17933	1021849	.9254	.2806	nonlinear CFD
lnsp3937	3937	25407	.8686	.1272	fluid flow
lns_3937	3937	25407	.8686	.1272	fluid flow
watson5	1853	7803	.8590	.6170	circuit simulation
add32	4960	23884	.8310	.3979	computer component design
sherman5	3312	20793	.7802	.2882	petroleum engineering
mhd4800a	4800	102252	.7718	.2806	magnetohydrodynamics
shyy161	76480	329762	.7685	.3085	fluid flow
shyy41	4720	20042	.7664	.3113	fluid flow
olm5000	5000	19996	.7500	.5000	hydrodynamics
fs_541_2	541	4285	.7227	.1262	chemical kinetics
mcfe	765	24382	.7088	.0313	astrophysics
pores_2	1224	9613	.6613	.4689	petroleum engineering
goodwin	7320	324772	.6423	.0194	fluid mechanics
tols4000	4000	8784	.5935	.3642	aeroelasticity
utm5940	5940	83842	.5624	.0708	plasmas nuclear physics
bbmat	38744	1771722	.5398	.0224	structure engineering CFD
rw5151	5151	20199	.4902	.0000	Markov chain transition
psmigr_1	3140	543162	.4816	.0161	demography
gre_1107	1107	5664	.1954	.1954	discrete simulation
onetone2	36057	227628	.1482	.1020	nonlinear circuit
rdist3a	2398	61896	.1468	.0074	chemical engineering
onetone1	36057	341088	.0989	.0681	nonlinear circuit
orani678	2529	90158	.0728	.0023	economics
rdist1	4134	94408	.0620	.0034	chemical engineering
radfr1	1048	13299	.0600	.0066	chemical engineering
rdist2	3198	56934	.0491	.0037	chemical engineering
mahindas	1258	7682	.0302	.0166	economics
lhr04	4101	82682	.0159	.0010	chemical engineering
lhr01	1477	18592	.0085	.0013	chemical engineering
hydr1	5308	23752	.0040	.0006	chemical engineering
extr1	2837	11407	.0040	.0005	chemical engineering
west2021	2021	7353	.0039	.0006	chemical engineering
vavasis1	4408	95752	.0033	.0003	2D PDE
vavasis2	11924	306842	.0025	.0001	2D PDE
gemat11	4929	33108	.0017	.0003	electrical power
lhr71	70304	1528092	.0014	.0002	chemical engineering
vavasis3	41092	1683902	.0009	.0000	2D PDE

MMD ordering takes significantly more time than factorization. It should be noted that our current approach to ordering can be improved. For example, the column minimum degree algorithm used in Matlab [26] implements the minimum degree algorithm on A^TA without explicitly forming the structure of A^TA. In recent work, Davis, Gilbert, and Ng [10, 45] are investigating better minimum degree algorithms for unsymmetric matrices that we expect to improve both fill and runtime.

For 9 of the 13 problems whose dimensions are at least 10000, SuperLU outper-

Table 5.9

*Performance of SuperLU and UMFPACK on an IBM RS/*6000-590 *with* 768 *MB of memory. Numbers in parenthesis are factorization rate in Mflops. A "+" before a number indicates SuperLU outperforms UMFPACK. UMFPACK ran out of memory on* bbmat.

Matrix	Seconds (Mflops)				
	SuperLU			UMFPACK	
	order	factor			
cry10000	+0.35	+1.73	(28)	2.30	(31)
memplus	89.58	+1.65	(22)	1.94	(1)
dw8192	+0.38	+3.69	(28)	5.64	(53)
garon2	+3.86	+22.12	(58)	53.83	(97)
jpwh_991	0.13	0.53	(36)	0.25	(20)
af23560	+9.22	+62.11	(80)	224.89	(112)
bramley2	+23.01	+57.97	(93)	279.42	(101)
bramley1	+23.00	+58.10	(93)	274.46	(106)
lnsp3937	+0.40	+1.28	(21)	3.11	(36)
lns_3937	+0.40	+1.12	(24)	2.52	(31)
watson5	0.22	+0.13	(12)	0.20	(.4)
add32	0.24	+0.17	(.9)	0.36	(.3)
sherman5	0.24	+0.77	(29)	0.88	(33)
mhd4800a	+0.74	+0.58	(8)	7.10	(65)
shyy161	+2.33	+22.04	(47)	54.14	(63)
shyy41	+0.14	+0.49	(17)	0.69	(13)
olm5000	0.12	+0.14	(.6)	0.21	(.4)
fs_541_2	0.08	+0.05	(5)	0.12	(3)
mcfe	1.21	+0.22	(13)	0.25	(17)
pores_2	+0.10	+0.17	(14)	0.28	(9)
goodwin	+5.34	+10.31	(49)	22.37	(64)
tols4000	+0.06	+0.06	(.6)	0.23	(.1)
utm5940	+0.93	+3.02	(43)	4.23	(61)
bbmat	+185.77	+821.49	(54)	failed	
rw5151	+0.16	+1.24	(27)	1.97	(28)
psmigr_1	290.08	187.86	(89)	93.93	(100)
gre_1107	0.15	0.39	(23)	0.31	(20)
onetone2	3.00	+4.98	(26)	7.02	(26)
rdist3a	0.77	0.48	(24)	0.44	(17)
onetone1	15.77	48.00	(53)	38.87	(79)
orani678	73.41	+1.32	(24)	1.63	(4)
rdist1	1.02	0.76	(23)	0.74	(12)
radfr1	0.09	+0.10	(11)	0.11	(4)
rdist2	0.50	0.45	(16)	0.41	(8)
mahindas	0.69	+0.09	(7)	0.17	(2)
lhr04	2.65	+1.10	(15)	1.71	(9)
lhr01	0.46	+0.21	(11)	0.29	(7)
hydr1	+0.53	+0.26	(3)	0.81	(3)
extr1	+0.13	+0.13	(1)	0.32	(1)
west2021	+0.08	+0.08	(.9)	0.26	(.2)
vavasis1	12.49	7.50	(81)	4.40	(46)
vavasis2	76.51	38.06	(94)	37.09	(56)
gemat11	0.24	+0.26	(4)	0.39	(2)
lhr71	48.23	+23.12	(21)	40.74	(17)
vavasis3	1091.72	660.89	(98)	533.35	(122)

forms UMFPACK both in factorization time and in memory.

6. Remarks.

6.1. The rest of the SuperLU package.

In addition to the LU factorization kernel described in this paper, we have developed a suite of supporting routines to

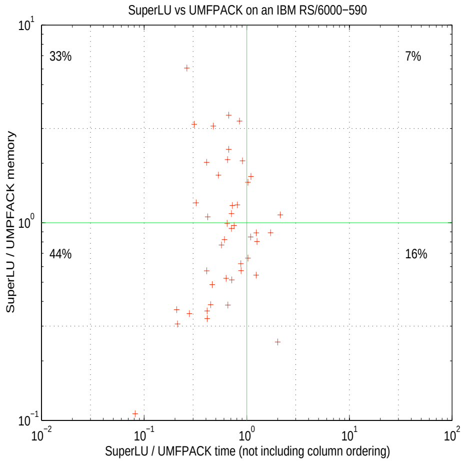
Table 5.10

*Performance of SuperLU and UMFPACK on an IBM RS/*6000-590 *with* 768 *MB of memory. A "+" before a number indicates SuperLU outperforms UMFPACK. UMFPACK ran out of memory on* bbmat.

Matrix	$nnz(F)$		Memory (MB)	
	SuperLU	UMF	SuperLU	UMF
cry10000	+651412	713789	+8.73	9.01
memplus	356842	157474	7.49	2.29
dw8192	+771299	1315910	+9.87	25.76
garon2	+4744645	8298907	+51.83	144.82
jpwh_991	144499	67703	1.71	1.56
af23560	+12950915	27979292	+136.35	392.83
bramley2	+12652411	29803064	+136.67	376.05
bramley1	+12684407	29355331	+137.04	445.65
lnsp3937	+367691	586976	+4.79	14.62
lns_3937	+373477	494120	+4.76	12.36
watson5	41152	17471	0.48	0.23
add32	40444	35660	1.45	0.47
sherman5	+238905	246358	+3.20	5.16
mhd4800a	+227469	1550890	+3.38	31.29
shyy161	+6695357	9524169	+83.29	145.83
shyy41	195685	167723	2.97	2.67
olm5000	40002	35004	1.47	0.42
fs_541_2	+16157	18458	0.30	0.28
mcfe	+63940	80595	+0.87	1.52
pores_2	+54028	54405	+0.83	1.01
goodwin	+2764580	5162683	+31.30	64.33
tols4000	8872	8784	0.97	0.16
utm5940	+994050	1225186	+11.46	22.31
bbmat	+49987183	failed	+538.76	failed
rw5151	+379538	440271	+5.12	9.78
psmigr_1	8709183	6369782	+88.75	355.98
gre_1107	110883	79048	+1.40	1.74
onetone2	+1270569	1337258	+20.92	22.34
rdist3a	254176	189852	+2.99	3.53
onetone1	+4676473	5145165	+56.47	103.99
orani678	804641	122079	7.99	6.49
rdist1	398008	277947	4.76	2.97
radfr1	50483	29699	0.72	0.35
rdist2	230084	151860	2.95	1.72
mahindas	23909	14126	0.54	0.31
lhr04	342084	313473	+4.62	4.65
lhr01	66863	60264	1.04	0.85
hydr1	+81335	111919	2.03	1.61
extr1	+34461	36836	0.97	0.48
west2021	19179	14615	0.63	0.20
vavasis1	1495345	915869	+16.21	18.21
vavasis2	5523041	4249821	+59.07	89.19
gemat11	87054	67149	1.93	0.82
lhr71	+7508569	8740127	+96.14	124.55
vavasis3	+39798599	41746313	+418.65	470.75

solve linear systems, all of which are available from Netlib.[4] The complete SuperLU package [12] includes column preordering for sparsity, based on Liu's multiple minimum degree algorithm [34] applied to the structure of A^TA. (As noted above, we plan to replace this with a new code that does not form A^TA.) SuperLU also includes condition number estimation, iterative refinement of solutions, and componentwise error

[4] URL: http://www.netlib.org/scalapack/prototype/



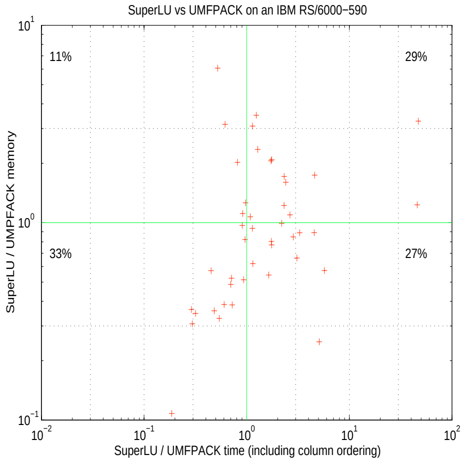


FIG. 5.7. *Compare SuperLU to UMFPACK, when MMD ordering time is not included.*

FIG. 5.8. *Compare SuperLU to UMFPACK, when MMD ordering time is included.*

bounds for the refined solutions. These are all based on the dense matrix routines in LAPACK [3]. In addition, SuperLU includes a Matlab mex-file interface, so that our factor and solve routines can be called as alternatives to those built into Matlab.

6.2. Effect of the matrix on performance. The supernodal approach reduces both symbolic and numeric computation time. But unsymmetric supernodes tend to be smaller than supernodes in symmetric matrices. The supernode-panel method is most effective for large problems with enough dense submatrices to use dense block operations and exploit data locality. In this regard, the dense 1000 × 1000 example illustrates the largest possible gains. Dense blocks are necessary for top performance in all modern factorization algorithms, whether left-looking, right-looking, multifrontal, or any other style.

Our goal has been to develop sparse LU software that works well for problems with a wide range of characteristics. It is harder to achieve high flop rates on problems that are very sparse and have no structure to exploit; it is easier on problems that are denser or become so during elimination. Fortunately, the "hard" matrices by this definition generally take many fewer floating-point operations than the "easy" ones, and hence take much less time to factor. Our combination of 1-D and 2-D blocking techniques gives a good performance compromise for all the problems we have studied, and with particularly good performance on the largest problems.

6.3. Effect of the computer system on performance. We have studied several characteristics of the computing platform that can affect the overall performance, including the Level 2 BLAS speed and the cache size. We showed how to systematically make a good choice of the blocking parameters in the code so as to maximize the speed of the numeric kernel, using the IBM RS/6000-590 as an example. We expect this methodology to be applicable to other systems (and BLAS implementations) as well.

6.4. Possible enhancements. We are considering several possible enhancements to the SuperLU code. One is to switch to a dense LU code at a late stage of the factorization. It would be difficult to implement this in a sup-col code, because that code is strictly left-looking, and only one column of the matrix is factored at a time. However, this would be much easier in the supernode-panel code. At the time we decide to switch, we simply treat the rest of the matrix columns (say, d of them) as one panel, and perform the panel update to $A(1{:}n, n-d+1{:}n)$. (One might want to split this panel up for better cache behavior.) Then the reduced matrix at the bottom right corner can be factored by calling an efficient dense code, for example, from LAPACK [3]. The dense code does not spend time on symbolic structure prediction and pruning, thus streamlining the numeric computation. We believe that, for large problems, the final dense submatrix will be big enough to make the switch beneficial. For example, for a 2-D $k \times k$ square grid problem ordered by nested dissection, the dimension of the final dense submatrix is $\frac{3}{2}k \times \frac{3}{2}k$; for a 3-D $k \times k \times k$ cubic grid, it is $\frac{3}{2}k^2 \times \frac{3}{2}k^2$, if pivots come from the diagonal. The Harwell library code MA48 [16, 18] employs such a switch to dense code, which has a significant and beneficial effect on performance.

To enhance SuperLU's performance on small and extremely sparse problems, it would be possible to make a choice at runtime whether to use supernode-panel, sup-col, or col-col updates. The choice would depend on the size of the matrix A and the expected properties of its supernodes; it might be based on an efficient symbolic computation of the density and supernode distribution of the Cholesky factor of $A^T A$ [28].

Could we make supernode-panel updates more effective by improving the similarity between the row structures of the columns in a panel? We believe this could be accomplished with a more sophisticated column permutation strategy. We could partition the nodes of the column etree into connected subtrees, grouping together nodes that have common descendants (and therefore the potential for updates from the same supernodes). Then the overall column order would be a two-level postorder, first within the subtrees (panels) and then among them. Again, it might be possible to use information about the Cholesky supernodes of $A^T A$ to guide this grouping.

We are also developing a parallel sparse LU algorithm based on SuperLU [11, 33]. In this context, we target large problems, especially those too big to be solved on a uniprocessor system. Therefore, we plan to parallelize the 2-D blocked supernode-panel algorithm, which has very good asymptotic behavior for large problems. The 2-D block-oriented layout has been shown to scale well for parallel sparse Cholesky factorization [31, 40].

Acknowledgments. We thank Rob Schreiber and Ed Rothberg for very helpful discussions during the development of SuperLU. Rob pushed us to find a way to do the supernode-panel update; Ed suggested the *ops-per-ref* statistic and also provided access to the SGI MIPS R8000. We are grateful to Tim Davis, Steve Vavasis, and Randy Bramley for making their matrices available to us. We thank the referees for their suggestions to improve the presentation of the paper. The Institute for Mathematics and Its Applications at the University of Minnesota provided the fertile environment in which this work began.

REFERENCES

[1] R. Agarwal, F. Gustavson, P. Palkar, and M. Zubair, *A performance analysis of the subroutines in the ESSL/LAPACK call conversion interface (CCI)*, IBM T.J. Watson Research Center, Yorktown Heights, NY, 1994.

[2] P. R. Amestoy and I. Duff, *Vectorization of a multiprocessor multifrontal code*, Int. J. Supercomputer Appl., 7 (1993), pp. 64–82.

[3] E. Anderson et al., *LAPACK User's Guide, 2nd ed.*, SIAM, Philadelphia, PA, 1995.

[4] C. Ashcraft and R. Grimes, *The influence of relaxed supernode partitions on the multifrontal method*, ACM Trans. Math. Software, 15 (1989), pp. 291–309.

[5] C. Ashcraft, R. Grimes, J. Lewis, B. Peyton, and H. Simon, *Progress in sparse matrix methods for large sparse linear systems on vector supercomputers*, Internat. J. of Supercomputer Applications, 1 (1987), pp. 10–30.

[6] J. Bilmes, K. Asanovic, J. Demmel, D. Lam, and C.-W. Chin, *Optimizing Matrix Multiply Using PHiPAC: A Portable, High-Performance, ANSI C Coding Methodology*, Tech. Report CS-96-326, Computer Science Department., University of Tennessee, Knoxville, TN, May 1996. (LAPACK Working Note #111).

[7] T. A. Davis, *User's Guide for the Unsymmetric-Pattern MultiFrontal Package (UMFPACK)*, Tech. Report TR-95-004, Computer and Information Sciences Department, University of Florida, Gainesville, FL, January 1995.

[8] T. A. Davis and I. S. Duff, *A combined unifrontal/multifrontal method for unsymmetric sparse matrices*, Tech. Report TR-95-020, Computer and Information Sciences Department, University of Florida, Gainesville, FL, 1995.

[9] T. A. Davis and I. S. Duff, *An unsymmetric-pattern multifrontal method for sparse LU factorization*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 140–158.

[10] T. A. Davis, J. R. Gilbert, E. Ng, and B. W. Peyton, *A column approximate minimum degree ordering algorithm*, Presented at Sixth SIAM Symposium on Applied Linear Algebra, Snowbird, UT, 1997.

[11] J. W. Demmel, J. R. Gilbert, and X. S. Li, *An Asynchronous Parallel Supernodal Algorithm for Sparse Gaussian Elimination*, Tech. Report UCB//CSD-97-943, Computer Science Division, University of California, Berkeley, CA, 1997.

[12] J. W. Demmel, J. R. Gilbert, and X. S. Li, *SuperLU User's Guide*, Tech. Report UCB//CSD-97-944, Computer Science Division, University of California, Berkeley, CA, 1997.

[13] D. Dodson and J. Lewis, *Issues relating to extension of the basic linear algebra subprograms*, ACM SIGNUM Newsletter, 20 (1985), pp. 2–18.

[14] J. J. Dongarra, J. D. Croz, S. Hammarling, and R. J. Hanson, *An extended set of basic linear algebra subroutines*, ACM Trans. Math. Software, 14 (1988), pp. 1–17, 18–32.

[15] I. Duff and J. Reid, *The multifrontal solution of indefinite sparse symmetric linear equations*, ACM Trans. Math. Software, 9 (1983), pp. 302–325.

[16] I. Duff and J. K. Reid, *The design of MA*48*, a code for the direct solution of sparse unsymmetric linear systems of equations*, ACM Trans. Math. Software, 22 (1996), pp. 187–226.

[17] I. S. Duff, R. Grimes, and J. Lewis, *Sparse matrix test problems*, ACM Trans. Math. Software, 15 (1989), pp. 1–14.

[18] I. S. Duff and J. K. Reid, *MA*48*, a Fortran Code for Direct Solution of Sparse Unsymmetric Linear Systems of Equations*, Tech. Report RAL–93–072, Rutherford Appleton Laboratory, Oxon, UK, 1993.

[19] S. C. Eisenstat, J. R. Gilbert, and J. W. Liu, *A supernodal approach to a sparse partial pivoting code*, in Householder Symposium 12, Los Angeles, CA, 1993.

[20] S. C. Eisenstat and J. W. H. Liu, *Exploiting structural symmetry in sparse unsymmetric symbolic factorization*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 202–211.

[21] S. C. Eisenstat and J. W. H. Liu, *Exploiting structural symmetry in a sparse partial pivoting code*, SIAM J. Sci. Statist. Comput., 14 (1993), pp. 253–257.

[22] J. A. George and E. Ng, *An implementation of Gaussian elimination with partial pivoting for sparse systems*, SIAM J. Sci. Statist. Comput., 6 (1985), pp. 390–409.

[23] J. A. George and E. Ng, *Symbolic factorization for sparse Gaussian elimination with partial pivoting*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 877–898.

[24] J. R. Gilbert, *Predicting structure in sparse matrix computations*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 62–79.

[25] J. R. Gilbert and J. W. H. Liu, *Elimination structures for unsymmetric sparse LU factors*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 334–352.

[26] J. R. Gilbert, C. Moler, and R. Schreiber, *Sparse matrices in Matlab: Design and implementation*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 333–356.

[27] J. R. Gilbert and E. Ng, *Predicting structure in nonsymmetric sparse matrix factorizations*, in Graph Theory and Sparse Matrix Computation, A. George, J. R. Gilbert, and J. W. H. Liu, eds., Springer–Verlag, New York, Berlin, 1993.

[28] J. R. Gilbert, E. G. Ng, and B. W. Peyton, *An efficient algorithm to compute row and column counts for sparse Cholesky factorization*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 1075–1091.

[29] J. R. Gilbert and T. Peierls, *Sparse partial pivoting in time proportional to arithmetic operations*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 862–874.

[30] J. R. Gilbert and S. Toledo, *An assessment of incomplete LU preconditioners for nonsymmetric linear systems*, manuscript, 1997.

[31] A. Gupta and V. Kumar, *Optimally scalable parallel sparse Cholesky factorization*, in Proc. 7th SIAM Conference on Parallel Processing for Scientific Computing, D.H. Bailey et. al., eds., SIAM, Philadelphia, 1995, pp. 442–447.

[32] *International Business Machines Corporation Engineering and Scientific Subroutine Library, Guide and Reference*, Version 2 Release 2, Order No. SC23-0526-01, 1994.

[33] X. S. Li, *Sparse Gaussian Elimination on High Performance Computers*, Tech. Report UCB//CSD-96-919, Computer Science Division, Ph.D. dissertation, University of California, Berkeley, CA, 1996.

[34] J. W. Liu, *Modification of the minimum degree algorithm by multiple elimination*, ACM Trans. Math. Software, 11 (1985), pp. 141–153.

[35] J. W. H. Liu, *The role of elimination trees in sparse factorization*, SIAM J. Matrix Anal. Appl., 11 (1990), pp. 134–172.

[36] *PDS: The performance database server*, *http://performance.netlib.org/performance/*, May 1995.

[37] E. G. Ng and B. W. Peyton, *Block sparse Cholesky algorithms on advanced uniprocessor computers*, SIAM J. Sci. Statist. Comput., 14 (1993), pp. 1034–1056.

[38] E. Rothberg and A. Gupta, *Efficient sparse matrix factorization on high-performance workstations—exploiting the memory hierarchy*, ACM Trans. Math. Software, 17 (1991), pp. 313–334.

[39] E. Rothberg and A. Gupta, *An evaluation of left-looking, right-looking and multifrontal approaches to sparse Cholesky factorization on hierarchical-memory machines*, Internat. J. High Speed Comput., 5 (1993), pp. 537–593.

[40] E. E. Rothberg and A. Gupta, *An efficient block-oriented approach to parallel sparse Cholesky factorization*, in Supercomput., 1993, pp. 503–512.

[41] A. H. Sherman, *On the Efficient Solution of Sparse Systems of Linear and Nonlinear Equations*, Ph.D. thesis, Yale University, New Haven, CT, 1975.

[42] 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.

[43] H. Simon, P. Vu, and C. Yang, *Performance of a Supernodal General Sparse Solver on the CRAY Y-MP:* 1.68 *GFLOPS with Autotasking*, Tech. Report TR SCA-TR-117, Boeing Computer Services, Seattle, WA, 1989.

[44] S. A. Vavasis, *Stable finite elements for problems with wild coefficients*, SIAM J. Numer. Anal., 33 (1996), pp. 890–916.

[45] S. I. Larimore, *An Approximate Minimum Degree Ordering Algorithm*, Tech. Report CISE-TR-98-016, Department of Computer and Information Science and Engineering, University of Florida, Gainesville, FL, 1998.

SIAM J. Matrix Anal. Appl.
Vol. 20, No. 3, pp. 756–776

AN ALGORITHM FOR CHECKING REGULARITY OF INTERVAL MATRICES[*]

CHRISTIAN JANSSON[†] AND JIRI ROHN[‡]

Abstract. Checking regularity (or singularity) of interval matrices is a known NP-hard problem. In this paper a general algorithm for checking regularity/singularity is presented which is not a priori exponential. The algorithm is based on a theoretical result according to which regularity may be judged from any single component of the solution set of an associated system of linear interval equations. Numerical experiments (with interval matrices up to the size $n = 50$) confirm that this approach brings an essential decrease in the amount of operations needed.

Key words. interval matrix, regularity, singularity, algorithm, branch-and-bound, linear interval equations, linear programming

AMS subject classifications. 65F30, 65G10, 90C90

PII. S0895479896313978

1. Introduction.

An interval matrix

$$A^I = [\underline{A}, \overline{A}] = \{A;\ \underline{A} \le A \le \overline{A}\}$$

(where $\underline{A}$, $\overline{A}$ are $n \times n$ matrices and the inequality is to be understood componentwise) is called regular if each $A \in A^I$ is nonsingular, and is said to be singular otherwise (i.e., if it contains a singular matrix). The problem of checking regularity naturally arises in solving linear interval equations, but it is also important in applications since some frequently used properties of interval matrices (such as positive definiteness, stability, or the P-matrix property) can be verified via checking regularity.

All presently known necessary and sufficient conditions for regularity of interval matrices (summed up in [17, Theorem 5.1]) exhibit exponential behavior: they require solving at least 2^n problems of some sort (such as computing determinants, solving linear equations, inverting matrices, checking the P-matrix property, etc.); hence they are of little use in practice except for examples of very low dimensions. The exponentiality inherent in all these conditions was explained by the result stating that checking regularity of interval matrices is an NP-hard problem (Poljak and Rohn [15], [16]; see also Nemirovskii [13]). Hence, in view of the present status of the complexity theory (the conjecture "P≠NP", Garey and Johnson [4]), there is only little hope that necessary and sufficient conditions verifiable in polynomial time may exist.

The present work was motivated by an attempt to construct an algorithm that would require an exponential number of operations only in the "worst case" examples, and would behave reasonably in the "average" ones. The algorithm presented in this paper, whose main idea is due to Jansson [7], is based on a necessary and sufficient condition of quite a different sort. First, it does not handle solely the interval matrix

[*] Received by the editors December 19, 1996; accepted for publication (in revised form) by U. Helmke July 31, 1998; published electronically April 20, 1999.
http://www.siam.org/journals/simax/20-3/31397.html

[†] Institute of Computer Science III, Technical University Hamburg-Harburg, Eißendorfer Str. 38, D-21071 Hamburg, Germany (jansson@tu-harburg.de).

[‡] Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic, and Institute of Computer Science, Academy of Sciences, Prague, Czech Republic (rohn@uivt.cas.cz). The work of this author was supported by the Czech Republic Grant Agency under grant 201/98/0222.

A^I, but it deals with the solution set $X(A^I, b) = \{x;\ Ax = b \text{ for some } A \in A^I\}$ of an interval linear system $A^I x = b$ with a specially preselected right-hand side b. Second, the necessary and sufficient condition says that for any component C of $X(A^I, b)$ (i.e., a maximal nonempty connected subset of $X(A^I, b)$), A^I is regular if and only if C is bounded. Hence, in order to check regularity, it is sufficient to take (better said, to generate) a *single* component of $X(A^I, b)$ and to check it for boundedness, which can be done by applying repeatedly a linear programming procedure. The fact that only one component is to be checked, together with a special choice of b aimed at minimizing the number of applications of the linear programming procedure, is decisive for achieving a big reduction in the amount of computations needed. As will be demonstrated later, we have been able to check in acceptable time regularity of interval matrices up to the size $n = 50$, which would be almost impossible via the classical necessary and sufficient conditions since $2^{50} \approx 10^{15}$.

The paper is organized as follows. To motivate the current research, in section 2 we first show how some properties of interval matrices (positive definiteness, stability, and the P-property) can be verified via checking regularity. The NP-hardness result is stated in section 3. In section 4 we derive a necessary and sufficient singularity condition of classical type; it not only demonstrates the inherent exponentiality of the problem, but also shows a way how to try to resolve it. The necessary and sufficient condition employed in the algorithm, formulated in terms of a component of the solution set as explained above, is proved in section 5. The algorithm is described in section 6, and the problem of the choice of an adequate right-hand side b is discussed in section 7. Some examples are given in the last section.

We shall use the following notations. The absolute value of a matrix $A = (a_{ij})$ is denoted by $|A| = (|a_{ij}|)$; the same notation applies to vectors as well. The set of all ± 1-vectors in R^n is denoted by Z; hence Z consists of 2^n vectors. For each $z \in R^n$ we denote

$$T_z := \begin{pmatrix} z_1 & 0 & \dots & 0 \\ 0 & z_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & z_n \end{pmatrix},$$

i.e., T_z is the diagonal matrix with diagonal vector z. For $x \in R^n$ we denote by $\operatorname{sgn} x$ the sign vector of x defined by

$$(\operatorname{sgn} x)_i = \begin{cases} 1 & \text{if } x_i \geq 0, \\ -1 & \text{if } x_i < 0 \end{cases}$$

$(i = 1, \dots, n)$. Hence, $\operatorname{sgn} x \in Z$ for each $x \in R^n$. Moreover, if $z = \operatorname{sgn} x$, then we have $|x| = T_z x$; we shall use this relation later to avoid use of absolute values.

2. Regularity and other properties.

2. Regularity and other properties. To motivate the current research, we shall state briefly here some results showing that a general method for checking regularity may be also employed for checking some other properties of interval matrices.

A real matrix A (not necessarily symmetric) is called positive definite if $x^T Ax > 0$ for each $x \neq 0$, stable if $\operatorname{Re} \lambda < 0$ for each eigenvalue λ of A, and P-matrix if all the principal minors of A are positive. An interval matrix A^I is said to have one of these properties if each $A \in A^I$ has this property.

Checking positive definiteness of interval matrices is needed, for example, in global optimization where branch-and-bound methods for nonlinear optimization problems may be accelerated and may be also made finite by using a so-called expansion scheme (see Jansson [6]) which is mainly based on proving positive definiteness of an interval matrix. The following result from [19] reduces checking positive definiteness to checking regularity.

Theorem 2.1. *An interval matrix $[\underline{A}, \overline{A}]$ is positive definite if and only if the interval matrix*

$$\left[\tfrac{1}{2}(\underline{A} + \underline{A}^T), \tfrac{1}{2}(\overline{A} + \overline{A}^T)\right]$$

is regular and contains at least one positive definite matrix.

The problem of checking stability of interval matrices has attracted the interest of many researchers in the last decade since it is closely connected to the problem of robust stability of a linear time-invariant system $\dot{x} = Ax$ under data perturbations (see Barmish [1], Lunze [9], and the survey paper by Mansour [10]). Here, reduction to regularity holds for symmetric interval matrices only. An interval matrix $A^I = [\underline{A}, \overline{A}]$ is called symmetric if both the bounds $\underline{A}, \overline{A}$ are symmetric; hence a symmetric interval matrix may contain nonsymmetric matrices. The following result again comes from [19].

Theorem 2.2. *A symmetric interval matrix A^I is stable if and only if it is regular and contains at least one symmetric stable matrix.*

The study of P-matrices is motivated, among other applications, by the fact that the linear complementarity problem

$$x^+ = Ax^- + b$$

has a unique solution $x = (x_i)$ for each right-hand side b if and only if A is a P-matrix (see Murty [11]). Here the vectors x^+ and x^- are defined by $x_i^+ = \max\{x_i, 0\}$ and $x_i^- = \max\{-x_i, 0\}$ for each i, so that $x = x^+ - x^-$. We have an analogous result [20].

Theorem 2.3. *A symmetric interval matrix A^I is a P-matrix if and only if it is regular and contains at least one symmetric P-matrix.*

These results give further motivation for studying the problem of checking regularity of interval matrices.

3. The NP-hardness result. At the end of the 1980s, existence of more than ten necessary and sufficient conditions for checking regularity of interval matrices, each of which required an exponential amount of time to check (summed up in [17, Theorem 5.1]), led to suspicion that the problem might be NP-hard. The corresponding statement was proved by Poljak and Rohn in a report [15] and later in a journal form [16]; it was independently proved by Nemirovskii [13]. In its most recent form [8], the result is stated as follows.

Theorem 3.1. *Checking regularity is NP-hard in the class of interval matrices of the form*

$$[A - E, A + E],$$

where A is a nonnegative symmetric positive definite rational matrix and E is the matrix of all ones.

This result shows that checking regularity of interval matrices is a problem at least as hard as the most difficult combinatorial problems in the class NP. Existence of a polynomial-time algorithm for checking regularity of interval matrices would imply, in view of the definition of NP-hardness (see Garey and Johnson [4]), polynomial-time solvability of all problems in the class NP, a possibility which at present cannot be fully excluded but in view of the experience with solving problems from the class NP gathered so far ought to be considered very unlikely. Nevertheless, polynomial-time algorithms may exist for special classes of interval matrices; see [2].

This shows that the problem we are interested in is in its full generality very difficult.

4. A necessary and sufficient condition. Some of the subsequent results (Theorems 4.2 and 5.1) can be derived from the Oettli–Prager theorem [14] for interval linear equations. In order to keep the paper self-contained, we prove them here in another way using the following result (Theorem 4.1) which is of independent interest. For the purpose of its formulation, for an interval matrix $A^I = [\underline{A}, \overline{A}]$ we introduce

$$A_c = \tfrac{1}{2}(\underline{A} + \overline{A})$$

(the center matrix) and

$$\Delta = \tfrac{1}{2}(\overline{A} - \underline{A})$$

(the radius matrix); then A^I can be written in the form

$$A^I = [A_c - \Delta, A_c + \Delta],$$

which we shall use from this point on. We have this result (for its formulation and proof, we refer to the notations introduced at the end of section 1).

Theorem 4.1. *Let $A^I = [A_c - \Delta, A_c + \Delta]$ be an $n \times n$ interval matrix and let $x \in R^n$. Then*

$$\{Ax;\, A \in A^I\} = [A_c x - \Delta|x|, A_c x + \Delta|x|].$$

Proof. If $A \in A^I$, then $|Ax - A_c x| = |(A - A_c)x| \le \Delta|x|$, which implies

$$Ax \in [A_c x - \Delta|x|, A_c x + \Delta|x|].$$

Conversely, let $b \in [A_c x - \Delta|x|, A_c x + \Delta|x|]$, so that $|A_c x - b| \le \Delta|x|$. Define

$$y_i = \begin{cases} (A_c x - b)_i/(\Delta|x|)_i & \text{if } (\Delta|x|)_i \ne 0, \\ 1 & \text{if } (\Delta|x|)_i = 0 \end{cases}$$

$(i = 1, \dots, n)$. Then $|y_i| \le 1$ and $(A_c x - b)_i = y_i(\Delta|x|)_i$ holds for each i; hence with $z = \operatorname{sgn} x$ we have $A_c x - b = T_y \Delta T_z x$ and thus $(A_c - T_y \Delta T_z)x = b$, where $A_c - T_y \Delta T_z$ belongs to A^I due to $|T_y \Delta T_z| \le \Delta$. Hence $b \in \{Ax;\, A \in A^I\}$. □

Returning to our problem, we obtain this characterization of singularity.

Theorem 4.2. *An interval matrix $A^I = [A_c - \Delta, A_c + \Delta]$ is singular if and only if the inequality*

$$|A_c x| \le \Delta|x| \tag{4.1}$$

has a nontrivial solution.

Proof. Obviously, A^I is singular if and only if

$$0 \in \{Ax;\, A \in A^I\}$$

holds for some $x \neq 0$, which in view of Theorem 4.1 is equivalent to

$$A_c x - \Delta|x| \leq 0 \leq A_c x + \Delta|x| \tag{4.2}$$

and thus also to

$$|A_c x| \leq \Delta|x|. \qquad \square$$

As we have seen in the proof, the left-hand-side absolute value in (4.1) can be removed (4.2), but the right-hand-side absolute value remains a problem. We can remove it only at the expense of the latent exponentiality in (4.1) becoming apparent, as it can be seen from the next theorem.

THEOREM 4.3. *An interval matrix $A^I = [A_c - \Delta, A_c + \Delta]$ is singular if and only if the linear programming problem*

$$\max\{z^T x;\ (A_c - \Delta T_z)x \leq 0,\ (A_c + \Delta T_z)x \geq 0,\ T_z x \geq 0\} \tag{4.3}$$

is unbounded for some $z \in Z$.

Proof. If A^I is singular, then by Theorem 4.2 there exists a vector $x \neq 0$ satisfying

$$-\Delta|x| \leq A_c x \leq \Delta|x|. \tag{4.4}$$

Setting $z = \operatorname{sgn} x$, we have $|x| = T_z x$; hence $T_z x \geq 0$ and from (4.4) it follows $(A_c - \Delta T_z)x \leq 0$ and $(A_c + \Delta T_z)x \geq 0$, which shows that x is a feasible solution of (4.3). Moreover, $z^T x = \sum_i |x_i| > 0$. Since αx is a feasible solution of (4.3) for each $\alpha > 0$, we can see that the linear programming problem (4.3) is unbounded.

Conversely, let (4.3) be unbounded for some $z \in Z$. Then there exists an x satisfying $(A_c - \Delta T_z)x \leq 0$, $(A_c + \Delta T_z)x \geq 0$, $T_z x \geq 0$, and $z^T x > 0$. Since $T_z x = |x|$, we obtain that (4.1) is satisfied for some $x \neq 0$; hence A^I is singular. $\square$

Let us now discuss this result. If we find out a $z \in Z$ for which (4.3) is unbounded, then A^I is proved to be singular and we are done. However, if A^I is regular, then we must inspect *all* the 2^n linear programming problems of type (4.3) (for each $z \in Z$) in order to be able to prove it. The cornerstone of the exponentiality is the fact that each linear programming problem (4.3) is feasible ($x = 0$ is always a feasible solution); hence none of the problems (4.3) can be a priori excluded. In this way we come to the basic idea: to replace the zero vector in the right-hand sides of the constraints $(A_c - \Delta T_z)x \leq 0$, $(A_c + \Delta T_z)x \geq 0$ by some nonzero vector b in order to make infeasible as many resulting linear programming problems as possible. We shall exploit and develop this idea in the next section.

5. Theoretical basis of the algorithm. Given an $n \times n$ interval matrix A^I and a vector $b \in R^n$ (we shall specify its choice later), consider the solution set of a system of interval linear equations defined by

$$X(A^I, b) = \{x;\ Ax = b \text{ for some } A \in A^I\}.$$

First we have this description.

THEOREM 5.1. *Let $A^I = [A_c - \Delta, A_c + \Delta]$ be an $n \times n$ interval matrix and let $b \in R^n$. Then*

$$X(A^I, b) = \{x;\ |A_c x - b| \leq \Delta|x|\}. \tag{5.1}$$

Proof. $x \in X(A^I, b)$ if and only if

$$b \in \{Ax;\ A \in A^I\}$$

holds, which according to Theorem 4.1 is equivalent to

$$A_cx - \Delta|x| \le b \le A_cx + \Delta|x|,$$

and thereby also equivalent to

$$|A_cx - b| \le \Delta|x|. \qquad \square$$

Next we shall remove the absolute values from (5.1).

THEOREM 5.2. *Let $A^I = [A_c - \Delta, A_c + \Delta]$ be an $n \times n$ interval matrix and let $b \in R^n$. Then*

$$X(A^I, b) = \bigcup_{z \in Z} X_z(A^I, b),$$

where

$$X_z(A^I, b) = \{x;\ (A_c - \Delta T_z)x \le b,\ (A_c + \Delta T_z)x \ge b,\ T_zx \ge 0\} \tag{5.2}$$

for $z \in Z$.

Proof. If $x \in X(A^I, b)$, then $|A_cx - b| \le \Delta|x|$ and we can easily see that x satisfies

$$(A_c - \Delta T_z)x \le b,\ (A_c + \Delta T_z)x \ge b,\ T_zx \ge 0 \tag{5.3}$$

for $z = \operatorname{sgn} x$; hence $x \in X_z(A^I, b)$. Conversely, if $x \in X_z(A^I, b)$ for some $z \in Z$, then from (5.3) we have $|A_cx - b| \le \Delta|x|$; hence $x \in X(A^I, b)$. $\square$

The set $X_z(A^I, b)$, defined by (5.2), is nothing else than the intersection of the solution set $X(A^I, b)$ with the orthant $\{x \in R^n;\ T_zx \ge 0\}$. Since $X_z(A^I, b)$ is described by a system of linear inequalities, it is a convex polytope; in particular, it is connected (let us recall that a set is called connected (sometimes, path-connected) if any two points of it can be connected by a curve which belongs entirely to the set; in case of a convex set the curve is simply the segment connecting the two points). However, $X(A^I, b)$, as the union of such sets, may be neither convex nor connected. Consider a one-dimensional example

$$[-1, 1]x_1 = [1, 1].$$

Here $X(A^I, b)$ consists of two disjoint unbounded connected sets

$$X_{-1}(A^I, b) = (-\infty, -1]$$

and

$$X_1(A^I, b) = [1, \infty).$$

We shall see later (Theorem 5.3) that such a situation may occur for singular interval matrices only (indeed, A_c is singular here).

A nonempty connected subset C of $X(A^I, b)$ is called a *component* of $X(A^I, b)$ if it has the property

$$C \subseteq D \subseteq X(A^I, b),\ D \text{ connected } \Rightarrow C = D,$$

i.e., if it is a maximal connected subset of $X(A^I, b)$ with respect to inclusion. Hence, each set $X_z(A^I, b), z \in Z$, being connected, must be entirely contained in a single component. This implies that each component C of $X(A^I, b)$ is of the form

$$C = \bigcup_{z \in Y} X_z(A^I, b) \tag{5.4}$$

for some $Y \subseteq Z$. The following theorem shows that regularity or singularity of A^I may be judged from *each* single component of $X(A^I, b)$. The result is due to Jansson [7], where it appeared in another formulation.

THEOREM 5.3. *Let A^I be an $n \times n$ interval matrix, let $b \in R^n$, and let C be an arbitrary component of $X(A^I, b)$. Then A^I is singular if and only if C is unbounded.*

Proof. Both implications are proved by contradiction. If A^I is regular, then $X(A^I, b)$, as the range of the continuous mapping $A \mapsto A^{-1}b$ on a compact set A^I, is connected and bounded; hence $C = X(A^I, b)$, which implies that C is bounded.

Conversely, let C be bounded. Since $C \neq \emptyset$ by definition, there exists an $x_0 \in C$ satisfying $A_0 x_0 = b$ for some $A_0 \in A^I$, and boundedness of C implies nonsingularity of A_0 (if A_0 were singular, then $A_0 \hat{x} = 0$ for some $\hat{x} \neq 0$ and C would contain an unbounded set $\{x_0 + \lambda \hat{x};\ \lambda \in R^1\}$, a contradiction). To prove that A^I is regular, assume to the contrary that A^I contains a singular matrix A_1. For each $t \in [0, 1]$ denote

$$A_t = A_0 + t(A_1 - A_0), \tag{5.5}$$

and let

$$\tau = \inf\{t \in [0, 1];\ A_t \text{ is singular}\}.$$

In view of continuity of the determinant, the infimum is achieved as minimum; hence A_τ is singular and $\tau \in (0, 1]$. For each $t \in [0, \tau)$, A_t is nonsingular; hence

$$x_t = A_t^{-1} b$$

is well defined and the mapping $s \mapsto x_s$, $s \in [0, t]$, defines a curve in $X(A^I, b)$ connecting x_0 with x_t. Hence $x_t \in C$ for each $t \in [0, \tau)$. Consider now the sequence of points $\{x_{t_m}\}$ with

$$t_m = (1 - \tfrac{1}{m})\tau, \tag{5.6}$$

$m = 1, 2, \ldots$. Then $\{x_{t_m}\}$ is bounded since C is bounded; hence it contains a convergent subsequence $\{x_{t_{m_k}}\}$, $x_{t_{m_k}} \rightarrow x^*$. Then $x^* \in C$ since C is closed due to (5.4), (5.2). As

$$A_{t_{m_k}} x_{t_{m_k}} = b$$

holds for each k, taking $k \rightarrow \infty$ we obtain in view of (5.5), (5.6) that

$$A_\tau x^* = b.$$

But since A_τ is singular, there exists an $\tilde{x} \neq 0$ with $A_\tau \tilde{x} = 0$; hence $A_\tau(x^* + \lambda \tilde{x}) = b$ for each $\lambda \in R^1$, which shows that C contains the unbounded set $\{x^* + \lambda \tilde{x};\ \lambda \in R^1\}$, a contradiction. Hence A^I must be regular; this concludes the second part of the proof. □

The result shows that if A^I is singular, then for each $b \in R^n$, *all* components of $X(A^I, b)$ are unbounded; if A^I is regular, then $X(A^I, b)$ has a single component which is equal to $X(A^I, b)$.

It remains to show how to check unboundedness of a component C in the form (5.4).

Theorem 5.4. *A component*

$$C = \bigcup_{z \in Y} X_z(A^I, b)$$

is unbounded if and only if the linear programming problem

$$\max\{z^T x;\ (A_c - \Delta T_z)x \leq b,\ (A_c + \Delta T_z)x \geq b,\ T_z x \geq 0\} \tag{5.7}$$

is unbounded for some $z \in Y$.

Proof. Obviously, C is unbounded if and only if $X_z(A^I, b)$ is unbounded for some $z \in Y$. If $X_z(A^I, b)$ is unbounded, then the optimization problem

$$\max\{|x_i|;\ (A_c - \Delta T_z)x \leq b,\ (A_c + \Delta T_z)x \geq b,\ T_z x \geq 0\} \tag{5.8}$$

is unbounded for some i, which implies that (5.7) is unbounded since

$$|x_i| \leq \sum_j |x_j| = z^T x$$

holds for each feasible solution x of (5.7), (5.8). Conversely, if (5.7) is unbounded, then (5.8) must be unbounded for some i, which means that the set $X_z(A^I, b)$ is unbounded. □

By comparing Theorems 4.3 and 5.4, we can see that the only (but essential) distinction made by the introduction of the right-hand side b is the difference in quantifiers "$z \in Z$" and "$z \in Y$," where cardinality of Y is equal to the number of the orthants intersected by the component C. Clearly, this number may be influenced by an appropriate choice of b. We shall discuss this question in section 7, but first we shall formulate an algorithm based on theoretical principles of this section.

6. The algorithm. The algorithm is based on Theorems 5.3 and 5.4 and constructs a component C of the form (5.4) implicitly in the following way. At the starting point, a list L of z's to be checked is initialized by setting $z := \operatorname{sgn}(A_c^{-1}b)$. In the current step of the algorithm, if for the current $z \in L$ the problem (5.7) is unbounded, then A^I is proved singular and the algorithm terminates; otherwise, if (5.7) is feasible, then we insert into L all the "neighboring" vectors from the set

$$N(z) = \{(z_1, \ldots, z_{j-1}, -z_j, z_{j+1}, \ldots, z_n)^T;\ 1 \leq j \leq n\}$$

that have not been checked yet; to be able to recognize it, we also keep a list K of vectors that have already passed the check. If L becomes empty at some stage, then the component of $X(A^I, b)$ containing the vector $A_c^{-1}b$ has been proved to be bounded, and A^I is regular by Theorem 5.3. In the following we give a formal description of the algorithm written in a pseudo-PASCAL code:

```
sing:=false;
if $A_c$ is singular then sing:=true
else
  $L := \emptyset$; $K := \emptyset$;
  select $b$; solve $A_c x = b$;
  $z := \operatorname{sgn} x$; insert $z$ into $L$;
  repeat
     remove an item $z$ from $L$;
```

```
        insert $z$ into $K$;
        **if** (5.7) is unbounded **then** *sing:=true*
        **else if** (5.7) is feasible **then** $L := L \cup (N(z) - (K \cup L))$
    **until** (*sing* or $L = \emptyset$);
**if** *sing* **then** {$A^I$ is singular} **else** {$A^I$ is regular}.
```

In order to simplify the proof of the main theorem, we first formulate an auxiliary result concerning the case when singularity was not detected during the algorithm. Denote by C_0 the component of $X(A^I, b)$ containing the point $x_c = A_c^{-1}b$ and let Y_0 be the set of ± 1-vectors that were inserted into L in the course of the algorithm. Then Y_0 has this property.

LEMMA 6.1. *If $x \in C_0$ and $T_{z_0}x \geq 0$ for some $z_0 \in Y_0$, then each $z \in Z$ with $T_z x \geq 0$ belongs to Y_0.*

Proof. For the purpose of the proof, denote the linear programming problem (5.7) by $P(z)$. Then x is a feasible solution of some $P(\overline{z})$. Since $T_{z_0}x = |x| = T_{\overline{z}}x$, we can see from the form of (5.7) that x is a feasible solution of $P(z_0)$. Let $T_z x \geq 0$, $z \neq z_0$. Denote

$$J = \{j;\ z_j \neq (z_0)_j\} = \{j_1, \ldots, j_m\}.$$

Since $T_{z_0}x \geq 0$ and $T_z x \geq 0$, it must be $x_j = 0$ for each $j \in J$. Set $z^0 = z_0$ and define vectors $z^k \in Z$, $k = 1, \ldots, m$, in the following way:

$$(z^k)_j = \begin{cases} (z^{k-1})_j & \text{if } j \neq j_k, \\ -(z^{k-1})_j & \text{if } j = j_k \end{cases} \tag{6.1}$$

($k = 1, \ldots, m$, $j = 1, \ldots, n$). We shall prove by induction on $k = 0, \ldots, m$ that $z^k \in Y_0$ and x is a feasible solution of $P(z^k)$. This is obvious for $k = 0$. If the assumption is true for some $k - 1 \geq 0$, then $z^{k-1} \in Y_0$ and $P(z^{k-1})$ is feasible; hence $N(z^{k-1}) - (K \cup L)$ was added to L in the respective step. Since $z^k \in N(z^{k-1})$ by (6.1), z^k was either already present in $K \cup L$, or newly added to L, in both cases $z^k \in Y_0$. Furthermore, since x is a feasible solution of $P(z^{k-1})$ and $T_{z^{k-1}}x = T_{z^k}x$ holds as $x_{j_k} = 0$, it is also a feasible solution of $P(z^k)$. This concludes the proof by induction; since $z^m = z$, we have $z \in Y_0$. □

As it can be seen, this detailed proof is a formalization of the following idea: if we take a path from x_c to $x \in C_0$, then the only change of signs occurs when the path passes through a point with one or more zero components; all the respective sign vectors are added to L in the course of the algorithm; hence they belong to Y_0.

Now we finally prove that the algorithm really performs the task for which it was designed.

THEOREM 6.2. *For each $n \times n$ interval matrix A^I and each $b \in R^n$, the algorithm in a finite number of steps checks regularity or singularity of A^I.*

Proof. First, only elements of the finite set Z are being inserted into L and no element may be reinserted; hence the algorithm terminates in a finite number of steps. If some problem (5.7) is proved unbounded, then A^I is singular by Theorem 5.4. Hence, we have only to prove that if A^I has not been found singular and if the list L becomes empty, then A^I is regular.

The component C_0 may be written in the form (5.4):

$$C_0 = \bigcup_{z \in Y} X_z(A^I, b),$$

where Y may be chosen so that $X_z(A^I, b) \neq \emptyset$ for each $z \in Y$. We shall prove that

$$Y \subseteq Y_0 \tag{6.2}$$

holds. Since $L = \emptyset$, this will imply that all $X_z(A^I, b)$, $z \in Y$ have been checked to be bounded; hence A^I is regular by Theorem 5.4.

To prove (6.2), take a $\overline{z} \in Y$. Choose an $x \in X_{\overline{z}}(A^I, b)$, so that $T_{\overline{z}}x \geq 0$. Since C_0 is connected and contains $x_c = A_c^{-1}b$, there exists a path from x_c to x contained entirely in C_0. In view of convexity of the sets $X_z(A^I, b), z \in Z$, the path may be chosen in a piecewise linear form $x^0x^1 \ldots x^m$, where $x^0 = x_c$, $x^m = x$, and the segment with endpoints x^i, x^{i+1} is always a part of a single orthant ($i = 0, \ldots, m-1$). We shall prove by induction on i that for each $i = 0, \ldots, m$, if $T_z x^i \geq 0$ for some $z \in Z$, then $z \in Y_0$. Since $z_c = \operatorname{sgn} x_c$ is inserted into L at the beginning of the main loop and $T_{z_c}x_c \geq 0$, the assertion for $x^0 = x_c$ follows from Lemma 6.1. Let the assertion be true for x^i, $i \geq 0$. Since the whole segment with endpoints x^i and x^{i+1} is a part of a single orthant, there exists a $\tilde{z} \in Z$ such that $T_{\tilde{z}}x^i \geq 0$ and $T_{\tilde{z}}x^{i+1} \geq 0$. Then $\tilde{z} \in Y_0$ by assumption concerning x^i; hence from $T_{\tilde{z}}x^{i+1} \geq 0$, $\tilde{z} \in Y_0$ we obtain from Lemma 6.1 that each $z \in Z$ with $T_z x^{i+1} \geq 0$ belongs to Y_0. This concludes the proof by induction. Hence, since $T_{\overline{z}}x^m = T_{\overline{z}}x \geq 0$, we have $\overline{z} \in Y_0$. This proves (6.2). □

In the form presented above, the algorithm suffers a serious setback. If a linear program (5.7) is found bounded (i.e., it has an optimal solution), then *all* the neighboring vectors $y \in N(z)$ are added to the list L (except those that have been already included). But we can see from the proof of Lemma 6.1 that we need only $N(z)$ to contain all the neighboring vectors y satisfying $X_y(A^I, b) \cap X_z(A^I, b) \neq \emptyset$. Hence vectors y with $X_y(A^I, b) \cap X_z(A^I, b) = \emptyset$ need not be included into $N(z)$, without affecting validity of Lemma 6.1 and of Theorem 6.2. In the next theorem we show how such vectors may be identified using information gathered from the current problem.

THEOREM 6.3. *Let for some $z \in Z$ the linear programming problem* (5.7) *have an optimal solution $\hat{x}$, and let $\hat{A} \in A^I$ be a nonsingular matrix satisfying $\hat{A}\hat{x} = b$. Then we have*

$$X_y(A^I, b) \cap X_z(A^I, b) = \emptyset$$

for each

$$y = (z_1, \ldots, z_{j-1}, -z_j, z_{j+1}, \ldots, z_n)^T \in N(z) \tag{6.3}$$

such that

$$2(z^T\hat{x})(|\hat{A}^{-1}|\Delta e)_j < |\hat{x}_j|, \tag{6.4}$$

where e is the vector of all ones.

Proof. Assume to the contrary that there exists an $x \in X_y(A^I, b) \cap X_z(A^I, b)$. Then in view of (6.3) it must be

$$x_j = 0. \tag{6.5}$$

Since $x \in X(A^I, b)$, there exists an $A \in A^I$ such that $Ax = b$. From the identity

$$\hat{A}(x - \hat{x}) = (\hat{A} - A)x$$

we have

$$|x - \hat{x}| = |\hat{A}^{-1}(\hat{A} - A)x| \leq 2|\hat{A}^{-1}|\Delta|x|. \tag{6.6}$$

But since $x \in X_z(A^I, b)$, for each $i \in \{1, \ldots, n\}$ there holds

$$|x_i| \le \sum_k |x_k| = z^T x \le z^T \hat{x};$$

hence

$$|x| \le (z^T \hat{x}) e,$$

and from (6.6) we obtain

$$|x - \hat{x}| \le 2(z^T \hat{x}) |\hat{A}^{-1}| \Delta e. \tag{6.7}$$

But from (6.4), (6.5), and (6.7) we have

$$|\hat{x}_j| = |x_j - \hat{x}_j| \le 2(z^T \hat{x})(|\hat{A}^{-1}|\Delta e)_j < |\hat{x}_j|,$$

which is a contradiction. Hence, $X_y(A^I, b) \cap X_z(A^I, b) = \emptyset$. □

This result shows that the algorithm will remain in force if we include into $N(z)$ only those vectors y of the form (6.3) which *do not* satisfy (6.4). A matrix $\hat{A}$ satisfying $\hat{A}\hat{x} = b$ may be constructed by means of the proof of Theorem 4.1.

In the general description of the algorithm we did not specify the order in which the items are to be removed from the list L. For an efficient implementation it is recommended, once a problem (5.7) has been solved, that one check immediately all the problems (5.7) with y of the form (6.3) for j's not satisfying (6.4), since each of these linear programming problems differs from the original one in one column and one coefficient of the objective function only, so that the previously computed simplex tableau may be easily updated for the new problem. Practical experience shows that in this implementation the algorithm requires $O(pn^4)$ operations, where p is the number of linear programming problems (5.7) solved in the course of the algorithm.

7. The choice of *b*. It is obvious that the performance of the algorithm depends heavily on the number of orthants intersected by the component C_0. It is therefore a natural idea to try to find a right-hand-side vector b such that C_0 would be entirely contained in a single orthant, preferably the nonnegative one. Unfortunately, it turns out that this is generally not possible, even in the case of regular interval matrices. The following example is due to Nedoma; we quote here the result only, referring an interested reader to [12] for a proof.

Example. Let $A^I = [A_c - \Delta, A_c + \Delta]$ with

$$A_c = \begin{pmatrix} 0 & 2 & 2 \\ 2 & 0 & 4 \\ 1 & 1 & 1 \end{pmatrix},$$

$$\Delta = \begin{pmatrix} 0 & \frac{3}{2} & \frac{3}{2} \\ \frac{3}{2} & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

Then A^I is regular and there does not exist a $b \in R^3$ satisfying

$$X(A^I, b) \subset (R^3_+)^0,$$

where $(R^3_+)^0$ is the interior of the nonnegative orthant in R^3.

Hence we must set the goal differently: to find a b such that $X(A^I, b)$ intersects a possibly small number of orthants. Let $x_c = A_c^{-1}b$ as before and let $x \in X(A^I, b)$. Then there exists a matrix $A \in A^I$ with $Ax = b$; hence

$$x = x_c + A_c^{-1}(A_c - A)x. \tag{7.1}$$

We shall try to choose b such that $|x_c| = |A_c^{-1}b|$ is componentwise as large as possible. Then, in view of (7.1), x_c and the perturbed solution x should stay in the same orthant, or differ only in a few signs.

Because $A_c^{-1}(\alpha b) = \alpha A_c^{-1}b$, it follows that $|A_c^{-1}b|$ can be made arbitrarily large. Therefore it is sufficient to look only at right-hand sides $b \in R^n$ with $\|b\|_\infty \leq 1$. This leads to the optimization problem

$$\max\{\gamma;\ |A_c^{-1}b| \geq \gamma e,\ -e \leq b \leq e\}. \tag{7.2}$$

This problem can be solved exactly by applying 2^n linear programming calls. But for construction of an appropriate right-hand side we need only a reasonably good approximation of the optimal solution. Therefore, we proceed in the following way. First, we look at the discrete problem

$$\max\{\gamma;\ |A_c^{-1}b| \geq \gamma e,\ b \in Z\}. \tag{7.3}$$

The solution of (7.3) should be a good approximation of the solution of (7.2). Since each component of $|A_c^{-1}b|$ must be greater than or equal to γ, we may assume that A_c^{-1} is row equilibrated, i.e., the ith row of A_c^{-1} is multiplied by the factor $1/\max_j|(A_c)^{-1}_{ij}|$, $i = 1, \dots, n$. Row scaling does not change regularity or singularity of an interval matrix A^I.

In order to calculate an approximation of an optimal solution of (7.3), we use a scheme which locally improves γ. We attempt to improve a current approximation b by looking for a superior solution b^{new} in an appropriate neighborhood of b. In the first step $b, b^{new} \in Z$ are called neighbored if they differ in exactly one entry. The neighbor b^{new} is accepted as a new approximation if the corresponding new objective value γ^{new} is better than the previous one. In the following we give a formal description (e_i denotes the ith column of the unit matrix):

```
$b := e$; $\gamma := \min_j |(A_c^{-1}b)_j|$;
for $k = 1, \dots, k_1$ do begin
  $i := k \bmod(n) + 1$;
  $b^{new} := b - 2b_i e_i$; $\gamma^{new} := \min_j |(A_c^{-1}b^{new})_j|$;
  if $\gamma^{new} > \gamma$ then begin $b := b^{new}$; $\gamma := \gamma^{new}$ end
end.
```

This algorithm starts with $b = e$, calculates k_1 neighbors, and compares the corresponding objective values. We have chosen $k_1 = n^2$. Because

$$A_c^{-1}b^{new} = A_c^{-1}b - 2b_i A_c{}^{-1}e_i,$$

the computational costs of the previous algorithm are $O(n^3)$ operations. Notice that the statement $b^{new} := b - 2b_i e_i$ simply changes the ith entry of b from -1 to 1 or conversely.

In the second step we improve the previously calculated right-hand side b by taking the same algorithm, but changing the neighborhood. Now $b, b^{new} \in Z$ are called neighbored if they differ in exactly two entries. The algorithm runs as follows:

```
for $k = 1, \dots, k_2$ do
for $l = 1, \dots, k_2$ do begin
  $i := k \operatorname{mod}(n) + 1$; $h := l \operatorname{mod}(n) + 1$;
  if $i \neq h$ then begin
    $b^{new} := b - 2b_i e_i - 2b_h e_h$; $\gamma^{new} := \min_j |(A_c^{-1} b^{new})_j|$;
    if $\gamma^{new} > \gamma$ then begin $b := b^{new}$; $\gamma := \gamma^{new}$ end
  end
end.
```

We have chosen $k_2 = n$, and a similar argument as above shows that this algorithm needs $O(n^3)$ operations.

The last calculated b, γ in the previous algorithm, denoted by $\hat{b}, \hat{\gamma}$, serve as an approximation of an optimal solution of the problem (7.3). Because we intended to solve the problem (7.2), we solve in the third step the linear programming problem

$$\max\{\gamma;\ T_z A_c^{-1} b \geq \gamma e,\ -e \leq b \leq e\},$$

where $z = \operatorname{sgn}(A_c^{-1}\hat{b})$. Since $\hat{b}, \hat{\gamma}$ is a feasible solution of this problem, it follows immediately that its optimal solution b^* satisfies

$$\min_j |(A_c^{-1} b^*)_j| \geq \min_j |(A_c^{-1}\hat{b})_j|,$$

and we can use b^* as the desired right-hand side for the main algorithm of section 6.

8. Numerical experiments. In this section results of some numerical experiments are described. To demonstrate how the algorithm works and what the computational costs of the method are, we have also added some examples which are not typical, or where the algorithm does not behave well. The computational costs are proportional to the number p of linear programming problems (5.7) solved in the course of the algorithm. Therefore, we display in the following tables the number p. The program of our algorithm is written in MATLAB and uses IEEE double floating point arithmetic.

We compare our algorithm with two sufficient regularity conditions. The first one (Beeck [3]) assures regularity of $A^I = [A_c - \Delta, A_c + \Delta]$ if

$$\varrho := \varrho(|A_c^{-1}|\Delta) < 1 \tag{8.1}$$

holds. The second one, by Rump [21], [22], states that A^I is regular if

$$\sigma := \frac{\sigma_{\max}(\Delta)}{\sigma_{\min}(A_c)} < 1, \tag{8.2}$$

where $\sigma_{\max}(\Delta), \sigma_{\min}(A_c)$ denote the maximal singular value of Δ and the minimal singular value of A_c, respectively. We shall see that in many cases both conditions are not satisfied, but A^I is regular. We shall also consider the radius of regularity of an interval matrix $A^I = [A_c - \Delta, A_c + \Delta]$ defined [16] by

$$r(A^I) = \inf\{\varepsilon \geq 0;\ [A_c - \varepsilon\Delta, A_c + \varepsilon\Delta] \text{ is singular}\}.$$

Hence A^I is regular if and only if $r(A^I) > 1$.

In what follows we present results of computations of eight examples. In all of them A_c is fixed and Δ is varied depending on a real parameter κ; in most cases

(Examples 2 and 4 through 7) we set $\Delta = \kappa|A_c|$. It can be seen from the tables that p grows, or does not decrease, until the radius of regularity is reached. It is typical for the method that singularity can be proved very fast (often with $p = 1$) for many examples because singular matrices usually occur in almost all orthants. Nevertheless, there exist also problems where p increases even in the singular case (Example 6).

Example 1. The following example can be viewed as a generalization of the two-dimensional Example 3.2 in [17], to the multidimensional case. The center matrix A_c has nonzero entries only in the main diagonal, and above and below the kth diagonals. We have chosen $n = 50, k = 48$, and A_c of the form

$$(A_c)_{ij} = \begin{cases} 50 & \text{if } i = j, \\ 100 & \text{if } j \geq i + 48, \\ -100 & \text{if } i \geq j + 48, \\ 0 & \text{otherwise} \end{cases}$$

$(i, j = 1, \ldots, 50)$. The radius matrix Δ is defined as follows:

$$(\Delta)_{ij} = \begin{cases} 40 & \text{if } i = j, \\ 0.01 + \kappa & \text{if } j \geq i + 48, \\ 0.01 + \kappa & \text{if } i \geq j + 48, \\ 0.01 & \text{otherwise} \end{cases}$$

$(i, j = 1, \ldots, 50)$. Hence, A^I has large intervals $[10, 90]$ on the diagonal, above and below the kth diagonals the widths of the intervals are very large (about 2κ), and for the zero entries of A_c we have small intervals $[-0.01, 0.01]$. Table 1 shows behavior of our algorithm and of the two sufficient regularity conditions for growing κ. We use an additional variable *reg* which is set to 1 if A^I is regular, and $reg = 0$ in case of singularity. We observe that with exception of $\kappa = 8$ both sufficient regularity conditions (8.1), (8.2) are not satisfied, but our algorithm proves regularity of A^I up to $\kappa = 96$, that is, the intervals above and below the kth diagonal are $[3.99, 196.01]$. In this case ϱ is about 3, σ is about 4, and $p = 6$. Change from regularity to singularity occurs between $\kappa = 96$ and $\kappa = 104$. In this region we performed some additional experiments with step size 0.5. For the values $\kappa = 96.5, 97, 97.5, 98$ our algorithm proved regularity with $p = 6$; for $\kappa = 99.5, 100, \ldots, 103.5$ singularity was proved with $p = 1$, but for the values $\kappa = 98.5$ and $\kappa = 99$ the algorithm run was stopped inconclusively after p reached 1000. For $\kappa \geq 104$ singularity of A^I was proved again with $p = 1$ (cf. the remark at the end of section 4). This shows that in a small region around $\kappa = r(A^I)$ the algorithm exhibits exponential behavior (which is a consequence of the NP-hardness result of section 3), whereas outside this region the computational costs indicated by p are very moderate, and much smaller in case of singularity than in that of regularity. This typical behavior (confirmed by many other experiments) constitutes the main advantage of our algorithm in contrast to necessary and sufficient regularity conditions [17, Theorem 5.1] which are exponential in n in *each* regularity case.

Example 2. Here the center matrix A_c is the Hilbert matrix of dimension $n = 7$. The 2-norm condition number of this matrix is about $4.7 \cdot 10^8$. The radius matrix is $\Delta = \kappa|A_c|$, i.e., we consider relative perturbations. Table 2 shows that in all cases $p = 1$ and ϱ characterizes the radius of regularity $r(A^I)$ because the inverse Hilbert matrix has a rank 1 sign pattern (see [23]), whereas σ underestimates this radius.

Example 3. This example is taken from [25, p. 442]. The center matrix is given

Table 1

κ	reg	ϱ	σ	p
8	1	0.9290	1.0595	3
16	1	1.1141	1.3183	3
24	1	1.2979	1.5772	4
32	1	1.4809	1.8361	5
40	1	1.6634	2.0950	6
48	1	1.8454	2.3539	6
56	1	2.0271	2.6128	6
64	1	2.2085	2.8716	6
72	1	2.3897	3.1305	6
80	1	2.5708	3.3894	6
88	1	2.7518	3.6483	6
96	1	2.9327	3.9072	6
104	0	3.1135	4.1661	1
112	0	3.2942	4.4250	1
120	0	3.4748	4.6838	1
128	0	3.6554	4.9427	1
136	0	3.8360	5.2016	1
144	0	4.0165	5.4605	1
152	0	4.1970	5.7194	1
160	0	4.3774	5.9783	1

Table 2

κ	reg	ϱ	σ	p
1.0e-09	1	0.1185	0.4754	1
2.0e-09	1	0.2369	0.9507	1
3.0e-09	1	0.3554	1.4261	1
4.0e-09	1	0.4738	1.9015	1
5.0e-09	1	0.5923	2.3768	1
6.0e-09	1	0.7108	2.8522	1
7.0e-09	1	0.8292	3.3276	1
8.0e-09	1	0.9477	3.8029	1
9.0e-09	0	1.0661	4.2783	1
1.0e-08	0	1.1846	4.7537	1
1.1e-08	0	1.3031	5.2290	1
1.2e-08	0	1.4215	5.7044	1
1.3e-08	0	1.5400	6.1798	1
1.4e-08	0	1.6585	6.6551	1
1.5e-08	0	1.7769	7.1305	1
1.6e-08	0	1.8954	7.6059	1
1.7e-08	0	2.0138	8.0812	1
1.8e-08	0	2.1323	8.5566	1

by

$$(A_c)_{ij} = \begin{cases} 11 - j & \text{if } j \geq i, \\ 10 - j & \text{if } j = i - 1, \\ 0 & \text{if } j < i - 1 \end{cases}$$

$(i, j = 1, \ldots, 10)$. The 2-norm condition number of A_c is about $2.8543 \cdot 10^7$. The coefficients of the radius matrix Δ are defined for all nonzero entries of A_c by $\Delta_{ij} = \kappa|(A_c)_{ij}|$, and are equal to 0.1 otherwise. Table 3 shows that in all cases $p = 1$, ϱ estimates very well the radius of regularity $r(A^I)$, whereas σ largely underestimates this radius.

Table 3

κ	reg	ϱ	σ	p
1.0e-08	1	0.1593	4.4539e+05	1
2.0e-08	1	0.2264	4.4539e+05	1
3.0e-08	1	0.2848	4.4539e+05	1
4.0e-08	1	0.3392	4.4539e+05	1
5.0e-08	1	0.3912	4.4539e+05	1
6.0e-08	1	0.4417	4.4539e+05	1
7.0e-08	1	0.4912	4.4539e+05	1
8.0e-08	1	0.5398	4.4539e+05	1
9.0e-08	1	0.5879	4.4539e+05	1
1.0e-07	1	0.6354	4.4539e+05	1
1.1e-07	1	0.6826	4.4539e+05	1
1.2e-07	1	0.7295	4.4539e+05	1
1.3e-07	1	0.7761	4.4539e+05	1
1.4e-07	1	0.8225	4.4539e+05	1
1.5e-07	1	0.8687	4.4539e+05	1
1.6e-07	1	0.9148	4.4539e+05	1
1.7e-07	1	0.9607	4.4539e+05	1
1.8e-07	0	1.0065	4.4539e+05	1
1.9e-07	0	1.0522	4.4539e+05	1
2.0e-07	0	1.0977	4.4539e+05	1

Example 4. This example is taken from [5, p. 41]. The ten-dimensional center matrix is orthogonal and given by

$$(A_c)_{ij} = (2/(n+1))^{1/2}\sin(ij\pi/(n+1)),$$

and the radius matrix is defined by

$$\Delta = \kappa|A_c|.$$

In contrast to the previous two examples we see from Table 4 that σ estimates regularity of A^I very well, whereas ϱ does not. (The reason for this behavior of ϱ and σ for orthogonal matrices is explained in Rump [21], [22].) The number p initially grows up to 249, and then p rapidly decreases to one in the case of singularity. Moreover, we see that comparing this matrix with the matrix of Example 3 yields types of matrices where both sufficient regularity conditions largely underestimate the radius of regularity, whereas our algorithm proves regularity or singularity of A^I with moderate computational costs.

Example 5. The following example is taken from [23]. The center matrix A_c is defined by

$$(A_c)_{ij} = \begin{cases} 1 & \text{if } j = i \text{ or } j = i-1, \\ (-1)^{n+1} & \text{if } i = 1 \text{ and } j = n, \\ 0 & \text{otherwise} \end{cases}$$

$(i, j = 1, \dots, n)$, and the radius matrix is given by

$$\Delta = \kappa|A_c|; \tag{8.3}$$

i.e., the coefficients of the center matrix are relatively perturbed. For this matrix Rump [24] showed that ϱ underestimates the radius of regularity of A^I by a factor which is equal to the dimension n. This is also demonstrated by Table 5, where $n = 10$.

Table 4

κ	reg	ϱ	σ	p
0.025	1	0.2199	0.0741	1
0.050	1	0.4398	0.1483	1
0.075	1	0.6597	0.2224	1
0.100	1	0.8795	0.2966	1
0.125	1	1.0994	0.3707	10
0.150	1	1.3193	0.4449	10
0.175	1	1.5392	0.5190	10
0.200	1	1.7591	0.5931	35
0.225	1	1.9789	0.6673	45
0.250	1	2.1988	0.7414	74
0.275	1	2.4187	0.8156	113
0.300	1	2.6386	0.8897	145
0.325	1	2.8585	0.9639	202
0.350	1	3.0784	1.0380	249
0.375	0	3.2982	1.1121	15
0.400	0	3.5181	1.1863	2
0.425	0	3.7380	1.2604	1
0.450	0	3.9579	1.3346	1

Table 5

κ	reg	ϱ	σ	p
0.1	1	1.000	0.639	5
0.2	1	2.000	1.279	53
0.3	1	3.000	1.918	61
0.4	1	4.000	2.557	118
0.5	1	5.000	3.196	118
0.6	1	6.000	3.836	118
0.7	1	7.000	4.475	118
0.8	1	8.000	5.114	118
0.9	1	9.000	5.753	118
1.0	0	10.000	6.393	2
1.1	0	11.000	7.032	1
1.2	0	12.000	7.671	1
1.3	0	13.000	8.310	1
1.4	0	14.000	8.949	1
1.5	0	15.000	9.589	1
1.6	0	16.000	10.228	1
1.7	0	17.000	10.867	1
1.8	0	18.000	11.506	1

Moreover, we see that σ also underestimates the radius of regularity. Here p grows up to 118 which is about n^2 in case of regularity, and $p \leq 2$ in all cases of singularity. Changing just one coefficient in the above example to $(A_c)_{1n} := n(-1)^{(n+1)}$, and defining Δ again by (8.3), we can see from Table 6 that $p \leq 10$ whereas both ϱ and σ underestimate the radius of regularity.

Example 6. The center matrix is the 10-dimensional matrix defined by

$$(A_c)_{ij} = \begin{cases} 1 & \text{if } j \geq i, \\ -1 & \text{if } j < i, \end{cases}$$

and the radius matrix is

$$\Delta = \kappa |A_c|.$$

TABLE 6

κ	reg	ϱ	σ	p
0.1	1	0.714	5.437	4
0.2	1	1.428	10.873	8
0.3	1	2.141	16.310	9
0.4	1	2.855	21.746	10
0.5	1	3.569	27.183	10
0.6	1	4.283	32.619	10
0.7	1	4.997	38.056	10
0.8	1	5.710	43.492	10
0.9	1	6.424	48.929	10
1.0	0	7.138	54.365	1
1.1	0	7.852	59.802	1
1.2	0	8.566	65.238	1
1.3	0	9.279	70.675	1
1.4	0	9.993	76.111	1
1.5	0	10.707	81.548	1
1.6	0	11.421	86.984	1
1.7	0	12.135	92.421	1
1.8	0	12.848	97.857	1

TABLE 7

κ	reg	ϱ	σ	p
0.02	1	0.2000	0.1975	1
0.04	1	0.4000	0.3951	1
0.06	1	0.6000	0.5926	1
0.08	1	0.8000	0.7902	1
0.10	1	1.0000	0.9877	55
0.12	0	1.2000	1.1852	123
0.14	0	1.4000	1.3828	124
0.16	0	1.6000	1.5803	69
0.18	0	1.8000	1.7778	81
0.20	0	2.0000	1.9754	20
0.22	0	2.2000	2.1729	20
0.24	0	2.4000	2.3705	20
0.26	0	2.6000	2.5680	21
0.28	0	2.8000	2.7655	21
0.30	0	3.0000	2.9631	21
0.32	0	3.2000	3.1606	21
0.34	0	3.4000	3.3581	2
0.36	0	3.6000	3.5557	2

This example is not typical and it shows a behavior which is contrary to most of our observations. As shown in Table 7, p is maximal in case of singularity, and moreover, in some singularity cases p increases for increasing κ.

Example 7. The following example is also not typical for most of our observations. It shows how the computational costs may change dramatically by altering slowly the dimension n. The center matrix is given by

$$(A_c)_{ij} = \begin{cases} 10 & \text{if } i < j, \\ 1 & \text{if } i = j, \\ -10 & \text{if } i > j, \end{cases}$$

and the radius matrix is

$$\Delta = \kappa |A_c|.$$

Table 8

κ	reg	ϱ	σ	p
0.005	1	0.3050	0.3050	1
0.010	1	0.6100	0.6100	1
0.015	1	0.9150	0.9150	1
0.020	0	1.2200	1.2200	1
0.025	0	1.5250	1.5250	1
0.030	0	1.8300	1.8300	1
0.035	0	2.1350	2.1350	1
0.040	0	2.4400	2.4400	1
0.045	0	2.7450	2.7450	1
0.050	0	3.0500	3.0500	1
0.055	0	3.3550	3.3550	1
0.060	0	3.6600	3.6600	1
0.065	0	3.9650	3.9650	1
0.070	0	4.2700	4.2700	1
0.075	0	4.5750	4.5750	1
0.080	0	4.8800	4.8800	1
0.085	0	5.1850	5.1850	1
0.090	0	5.4900	5.4900	1

Table 9

κ	reg	ϱ	σ	p
0.005	1	0.2231	0.1595	1
0.010	1	0.4461	0.3189	3
0.015	1	0.6692	0.4784	5
0.020	1	0.8923	0.6378	5
0.025	1	1.1154	0.7973	10
0.030	1	1.3384	0.9567	10
0.035	1	1.5615	1.1162	10
0.040	0	1.7846	1.2756	1
0.045	0	2.0076	1.4351	1
0.050	0	2.2307	1.5945	1
0.055	0	2.4538	1.7540	1
0.060	0	2.6769	1.9134	1
0.065	0	2.8999	2.0729	1
0.070	0	3.1230	2.2324	1
0.075	0	3.3461	2.3918	1
0.080	0	3.5691	2.5513	1
0.085	0	3.7922	2.7107	1
0.090	0	4.0153	2.8702	1

For $n = 7$ we get $p = 1$ and $\varrho = \sigma$ in all cases (Table 8). The results for $n = 8$ are displayed in Table 9.

Example 8. This example shows, for varying dimension n, results for interval matrices where the coefficients are normally distributed with mean 0 and variance 1. In detail, the center matrix is generated by $A_c = randn(n)$, $\kappa = 0.02 \cdot randn(n)$, and the radius matrix is given by $\Delta = \kappa \cdot randn(n)$. Here, $rand(n)$ is the MATLAB command for the normal random distribution, and for each test set the seed is set to 0. For dimension $n = 20$, we get the results in Table 10. Dimension $n = 30$ yields the results in Table 11, and $n = 40$ yields the results in Table 12. We can see that for most of these randomly generated examples $p = 1$, and in the worst cases p is bounded by n^2.

Table 10

κ	reg	ϱ	σ	p
1.5000e-03	1	2.0127e-01	2.5422e-01	1
3.4675e-03	1	2.5216e-01	2.9794e-01	1
5.7830e-03	1	6.1268e-01	8.9690e-01	10
3.8228e-03	0	4.2597e+00	6.6856e+00	1
2.1062e-02	1	1.1923e+00	1.3357e+00	54
8.0028e-03	0	2.8315e+00	3.9616e+00	1
1.9898e-02	0	2.0265e+00	2.4730e+00	1
5.7956e-03	1	1.0729e+00	1.5084e+00	17
4.6480e-04	1	3.5748e-02	4.7923e-02	1
1.0293e-02	1	9.1469e-01	1.1683e+00	20

Table 11

κ	reg	ϱ	σ	p
3.2528e-04	1	6.2265e-02	8.6597e-02	1
1.4271e-02	0	1.6529e+00	2.1586e+00	1
1.3444e-02	0	6.4642e+00	9.0395e+00	1
2.9834e-03	1	5.8821e-01	7.7027e-01	2
5.1537e-03	0	1.7320e+00	1.9743e+00	1
3.1523e-02	0	2.0272e+01	3.1276e+01	1
7.6462e-03	1	1.2505e+00	1.8812e+00	446
2.7498e-02	0	6.7397e+00	9.4327e+00	1
4.6382e-03	0	5.7781e+00	8.8484e+00	1
7.9532e-03	0	4.1317e+00	6.6051e+00	1

Table 12

κ	reg	ϱ	σ	p
1.0966e-02	0	6.6685e+00	1.0145e+01	1
2.4906e-03	1	3.0435e-01	3.2999e-01	1
1.7104e-03	1	2.6830e-01	2.5902e-01	1
6.5490e-03	0	2.1063e+00	3.3164e+00	1
1.4050e-02	0	2.4355e+00	2.8584e+00	1
1.0242e-02	0	5.6521e+00	8.2005e+00	1
2.4428e-03	0	3.8025e+00	5.4907e+00	1
1.0429e-02	0	2.4616e+00	2.8986e+00	1
4.8682e-03	1	1.0540e+00	1.4246e+00	986
1.5887e-02	0	4.1782e+00	5.8441e+00	1

9. Concluding remarks. As seen from the examples presented in the last section, the algorithm proved to be efficient, exhibiting very moderate numbers of calls of the linear programming procedure on the average. This behavior is to be ascribed to three features: (i) employing a new criterion of Theorem 5.3 which makes the algorithm not a priori exponential, (ii) using a fast heuristic algorithm for choosing a proper right-hand side b, and (iii) avoiding unnecessary calls of the linear programming procedure (Theorem 6.3).

Nevertheless, the computational work is still large compared to checking simple sufficient conditions (8.1) or (8.2). Therefore, for practical computations it is recommended that one use a hybrid algorithm which would first try the sufficient regularity conditions (8.1), (8.2), the sufficient singularity condition

$$\max_{ij}(|A_c^{-1}|\Delta)_{ij}(|A_c^{-1}|\Delta)_{ji} \geq 1$$

[23, Theorem 6.5], and the algorithm for finding a singular matrix in an interval matrix [18], and resort to the actual algorithm of this paper only if all the previous checks fail. In this way the algorithm can be made even more efficient.

Acknowledgments. The authors wish to thank the two referees for a number of constructive comments and suggestions that helped to improve the text of this paper.

REFERENCES

[1] B. R. Barmish, *New tools for robustness analysis*, in Proceedings of the 27th Conference on Decision and Control, Austin, TX, 1988, pp. 1–6.

[2] I. Bar-On, B. Codenotti, and M. Leoncini, *Checking robust nonsingularity of tridiagonal matrices in linear time*, BIT, 36 (1996), pp. 206–220.

[3] H. Beeck, *Zur Problematik der Hüllenbestimmung von Intervallgleichungssystemen*, in Interval Mathematics, K. Nickel, ed., Lecture Notes in Comput. Sci. 29, Springer-Verlag, Berlin, 1975, pp. 150–159.

[4] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP–Completeness*, Freeman, San Francisco, 1979.

[5] R. T. Gregory and D. L. Karney, *Collection of Matrices for Testing Computational Algorithms*, John Wiley & Sons, New York, 1969.

[6] C. Jansson, *On self-validating methods for optimization problems*, in Topics in Validated Computations, J. Herzberger, ed., North–Holland, Amsterdam, 1994, pp. 381–438.

[7] C. Jansson, *Calculation of exact bounds for the solution set of linear interval systems*, Linear Algebra Appl., 251 (1997), pp. 321–340.

[8] V. Kreinovich, A. Lakeyev, J. Rohn, and P. Kahl, *Computational Complexity and Feasibility of Data Processing and Interval Computations*, Kluwer, Dordrecht, The Netherlands, 1997.

[9] J. Lunze, *Robust Multivariable Feedback Control*, Prentice-Hall, Englewood Cliffs, NJ, 1989.

[10] M. Mansour, *Robust stability of interval matrices*, in Proceedings of the 28th Conference on Decision and Control, Tampa, FL, 1989, pp. 46–51.

[11] K. G. Murty, *Linear Complementarity, Linear and Nonlinear Programming*, Heldermann, Berlin, 1988.

[12] J. Nedoma, *Positively regular vague matrices*, Linear Algebra Appl., to appear.

[13] A. Nemirovskii, *Several NP-hard problems arising in robust stability analysis*, Math. Control Signals Systems, 6 (1993), pp. 99–105.

[14] W. Oettli and W. Prager, *Compatibility of approximate solution of linear equations with given error bounds for coefficients and right–hand sides*, Numer. Math., 6 (1964), pp. 405–409.

[15] S. Poljak and J. Rohn, *Radius of Nonsingularity*, Research report, KAM Series 88–117, Faculty of Mathematics and Physics, Charles University, Prague, Czechoslovakia, December 1988.

[16] S. Poljak and J. Rohn, *Checking robust nonsingularity is NP–hard*, Math. Control Signals Systems, 6 (1993), pp. 1–9.

[17] J. Rohn, *Systems of linear interval equations*, Linear Algebra Appl., 126 (1989), pp. 39–78.

[18] J. Rohn, *An algorithm for finding a singular matrix in an interval matrix*, J. Numer. Linear Algebra Appl., 1 (1992), pp. 43–47.

[19] J. Rohn, *Positive definiteness and stability of interval matrices*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 175–184.

[20] J. Rohn and G. Rex, *Interval P–matrices*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 1020–1024.

[21] S. M. Rump, *Validated solution of large linear systems*, in Computing Supplementum 9, R. Albrecht, G. Alefeld, and H. J. Stetter, eds., Springer-Verlag, Wien, 1993, pp. 191–212.

[22] S. M. Rump, *Verification methods for dense and sparse systems of equations*, in Topics in Validated Computations, J. Herzberger, ed., North–Holland, Amsterdam, 1994, pp. 63–135.

[23] S. M. Rump, *Bounds for the componentwise distance to the nearest singular matrix*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 83–103.

[24] S. M. Rump, *Almost sharp bounds for the componentwise distance to the nearest singular matrix*, Linear and Multilinear Algebra, 42 (1998), pp. 93–108.

[25] H. Rutishauser, *Lectures on Numerical Analysis*, Birkhäuser, Basel, Switzerland, 1990.

SIAM J. MATRIX ANAL. APPL.
Vol. 20, No. 3, pp. 777–799

CAUCHY-LIKE PRECONDITIONERS FOR TWO-DIMENSIONAL ILL-POSED PROBLEMS*

MISHA E. KILMER†

Abstract. Ill-conditioned matrices with block Toeplitz, Toeplitz block (BTTB) structure arise from the discretization of certain ill-posed problems in signal and image processing. We use a preconditioned conjugate gradient algorithm to compute a regularized solution to this linear system given noisy data. Our preconditioner is a Cauchy-like block diagonal approximation to a unitary transformation of the BTTB matrix. We show that the preconditioner has desirable properties when the kernel of the ill-posed problem is smooth: the largest singular values of the preconditioned matrix are clustered around one, the smallest singular values remain small, and the subspaces corresponding to the largest and smallest singular values, respectively, remain unmixed. For a system involving np variables, the preconditioned algorithm costs only $O(np(\lg n + \lg p))$ operations per iteration. We demonstrate the effectiveness of the preconditioner on three examples.

Key words. regularization, ill-posed problems, Toeplitz, Cauchy-like, preconditioner, conjugate gradient, normal equations, image processing, deblurring

AMS subject classifications. 65R20, 45L10, 94A12

PII. S0895479897319532

1. Introduction.

The two-dimensional integral equation

$$\int_{\alpha_l}^{\alpha_u} \int_{\beta_l}^{\beta_u} t(\alpha, \beta, \gamma, \delta) \hat{f}(\alpha, \beta) d\alpha d\beta = \hat{g}(\gamma, \delta)$$

is often used to describe the process by which data in signal and image processing applications is acquired. In optics, for example, t is called the *point spread function* and describes the response of the system or measuring device to a single point of light at coordinates (α, β) [13]. Thus if the values $\hat{f}(\alpha, \beta)$ represent light intensities reflected from a three-dimensional object, the integral equation might be used to model the blurring of that object when its picture is taken using a camera with a warped lens.

For simplicity, suppose quadrature is used to discretize the integral, and suppose p is the number of grid points α_j in the α direction and n is the number of grid points β_l in the β direction. The integral equation becomes a system of np linear equations of the form

$$T\hat{f} = \hat{g}, \tag{1}$$

where $\hat{f}$ is $np \times 1$ with entries $\hat{f}(\alpha_j, \beta_l)$, $1 \le j \le p$, $1 \le l \le n$. We note that many other discretization methods for the integral equation yield a system of np linear equations in which p and n have analogous definitions.

A Toeplitz matrix T_i is one whose elements are constant along diagonals; that is, the (k, j) entry in T_i is given by $t^{(i)}_{k-j}$. In applications, properties of the kernel,

*Received by the editors April 4, 1997; accepted for publication (in revised form) by D. Calvetti April 20, 1998; published electronically April 20, 1999. This work was supported by the National Science Foundation under grant CCR 95-03126.

http://www.siam.org/journals/simax/20-3/31953.html

†Applied Mathematics Program, University of Maryland, College Park, MD 20742 (mkilmer@ece.neu.edu). Current address: Electrical and Computer Engineering Department, Northeastern University, Boston, MA 02115.

discretization process, and a suitable ordering of unknowns can cause T to have a block Toeplitz structure in which each $p \times p$ block is Toeplitz. This structure arises, for example, by applying quadrature to a kernel t of the form $t(\alpha, \beta, \gamma, \delta) = t(\gamma - \alpha, \delta - \beta)$, ordering the unknowns $\hat{f}(\alpha_j, \beta_l)$ first by increasing j, then by increasing l. In this case, the (i, j) component in the (k, l) block is given by $(T_{kl})_{ij} = t_{i-j}^{(k-l)}$ for $1 \leq i, j \leq p$, $1 \leq k, l \leq n$. We then say that T is a block Toeplitz, Toeplitz block (BTTB) matrix.[1] For the remainder of the paper, we shall assume that n and p are powers of two, as is often the case in image processing applications.

Given $\hat{g}$ and T in (1), the discrete inverse problem is to recover $\hat{f}$. However, the continuous problem is generally ill-posed in the sense that small changes in $\hat{g}$ cause arbitrarily large changes in $\hat{f}$. Consequently, the matrix T will be ill-conditioned. Recovery of $\hat{f}$ is then complicated by the fact that noise e is also present in the measured data. That is, we have measured g rather than $\hat{g}$, where

$$Tf = \hat{g} + e = g. \tag{2}$$

Given the ill-conditioning of T, the exact solution, f, to (2) is not a reasonable approximation to $\hat{f}$. We instead seek an approximate solution f by solving a nearby, more well-posed problem. This method of approximating $\hat{f}$ is called *regularization*. We use a preconditioned conjugate gradient algorithm to compute such a regularized solution. A discussion of the methods of direct and iterative regularization techniques can be found in [19].

Iterative methods like conjugate gradients can take advantage of the well-known fact that matrix-vector products involving BTTB matrices with n blocks of size p can be computed in $O(np(\lg p + \lg n))$ operations [4]. Also, preconditioners for BTTB matrices which are block circulant (BC), circulant block (CB), or block circulant with circulant blocks (BCCB) have been found to be very efficient [4, 29, 1]. For example, if the preconditioner is determined to be block Toeplitz with circulant blocks (BTCB), applying the preconditioner can be reduced to solving p systems involving $n \times n$ Toeplitz matrices [4]. The application of other preconditioners which approximate the blocks by optimal fast transform-based matrices [4, 5, 26] can similarly be reduced to this form. However, for indefinite and/or ill-conditioned systems, the $O(n \lg^2 n)$ and $O(n^2)$ factorization algorithms for Toeplitz matrices can be numerically unstable; these algorithms can require as many as $O(n^3)$ operations in order to maintain stability [31, 15, 7].

To overcome this difficulty, we make use of the fact that because of their *displacement structure* (see [25, 9], for example), Toeplitz matrices are related to *Cauchy-like* matrices by fast unitary transformations [21, 9, 14]. The particular Cauchy-like matrices discussed in section 2 permit fast matrix-vector multiplication. An advantage of Cauchy-like matrices is that their inverses are also Cauchy-like, unlike Toeplitz matrices whose inverses are not generally Toeplitz. In addition, modified complete pivoting can be incorporated in the LDU factorization of a Cauchy-like matrix for a total cost of only $O(n^2)$.

In the course of this paper we develop a Cauchy-like preconditioner that can be used to filter noise and accelerate convergence of the conjugate gradient iteration to an approximate solution of (2) when T is BTTB. This preconditioner is the two-dimensional generalization of the preconditioner for Toeplitz matrices discussed in

[1] Although we assume here that T is square, we note that the preconditioner can be adjusted to the nonsquare case [27].

[28]. We begin with a discussion in section 2 of Cauchy-like matrices and some of their important properties. We discuss the regularizing properties of conjugate gradients and our choice of preconditioner in section 3. In section 4 we show that our preconditioner has desirable properties. Computational issues are the focus of section 5, where it is shown that each iteration can be completed in $O(pn(\lg p + \lg n))$ operations. Section 6 contains numerical results for several examples, and section 7 presents conclusions and future work.

2. Transformation from Toeplitz to Cauchy-like structure. A matrix C having the form

$$C = \left(\frac{a_i^T b_j}{\omega_i - \theta_j} \right)_{1 \le i,j \le n} \qquad (a_i, b_j \in \mathcal{C}^{\ell \times 1}; \omega_i, \theta_j \in \mathcal{C}; \omega_i \neq \theta_j) \tag{3}$$

is called a Cauchy-like, or generalized Cauchy, matrix. If $\ell = 1$ and $a_i^T b_j = 1$, then the matrix is said to be Cauchy. The matrix C can also be identified as the unique solution of the *displacement equation*

$$\Omega C - C\Theta = AB^T, \tag{4}$$

where

$$\Omega = \mathrm{diag}(\omega_1, \ldots, \omega_n), \Theta = \mathrm{diag}(\theta_1, \ldots, \theta_n), A = [a_1, \ldots, a_n]^T, B = [b_1, \ldots, b_n]^T.$$

The matrices A and B are called the *generators* of C with respect to Ω and Θ, and $\ell \le n$ is called the *displacement rank.* Notice that only the $2n\ell + 2n$ nonzero entries of A, B, Ω, Θ need to be stored to completely specify the entries of the matrix. Fortunately, certain properties of Cauchy-like matrices insure that LU factorizations of Cauchy-like matrices may be computed using only the matrices Ω, Θ, and the generators without ever forming the matrix C (see [9], for example).

One disadvantage of Toeplitz matrices is that permutations of Toeplitz matrices are not necessarily Toeplitz, so that incorporating pivoting into fast factorization schemes becomes difficult and expensive. However, because of (4), it is easy to show the following (see [21, 14], for example).

Property 1. *Row and column permutations of Cauchy-like matrices are Cauchy-like, as are leading principal submatrices.*

This property and the fact that Schur complements of Cauchy-like matrices are Cauchy-like [9] lead to fast algorithms for factoring Cauchy-like matrices which can pivot for stability [9, 14, 10, 11, 12]. We refer the reader to [12] for a detailed survey of types and applications of pivoting for structured matrices. Partial pivoting is usually sufficient to achieve stability for ill-conditioned matrices during factorization. However, our goal is to develop a preconditioner which clusters only a specific part of the spectrum (see section 3). We found in practice that we need a form of complete pivoting to provide us with the necessary rank revealing information.

We use the factorization algorithm (2.1) for Cauchy-like matrices given in [10] (see also [9, 11]) together with a pivoting scheme developed by Gu [14]. This form of the algorithm performs a fast $O(\ell n^2)$ variation of LU decomposition with modified complete pivoting. Recall that in complete pivoting, at every elimination step one chooses the largest element in the current submatrix as the pivot in order to reduce element growth. Gu proposes instead that one find an entry sufficiently large in magnitude by considering the largest 2-norm column of one of the generators that remains to be factored at each step. This algorithm computes the pivoted LU factorization ($C = PLUQ$, where P and Q are permutation matrices) [14, Alg. 2] using only the generators, which are easy to determine and to update (see section 5), and Gu shows that the algorithm, using his pivoting strategy, can be efficient and numerically stable. Although the Cauchy-like matrices of interest to us are full, they have displacement rank $\ell = 1$ or 2, which makes them both efficient to store using relation (4) and fast to factor. For our purposes it was convenient to set $D = \text{diag}(u_{11}, \ldots, u_{nn})$ and $U \leftarrow D^{-1}U$ to obtain the equivalent factorization $C = PLDUQ$.

We also exploit the following property of Cauchy-like matrices [21].

Property 2. *The inverse of a Cauchy-like matrix is Cauchy-like:*

$$C^{-1} = -\left(\frac{x_i^T w_j}{\theta_i - \omega_j}\right)_{1 \le i,j \le n} \qquad (x_i, w_j \in \mathcal{C}^{\ell \times 1}). \tag{5}$$

The generators X and W can be determined from the relations [21]

$$CX = A, \quad W^T C = B^T. \tag{6}$$

Thus, given the LU factorization of C, solving for X and W requires only $O(\ell n^2)$ operations and is stable when C is well-conditioned.

The third important property is that Toeplitz matrices also satisfy certain displacement equations [25, 9, 26] which allow them to be transformed via fast Fourier transforms into Cauchy-like matrices [21, 9].

Property 3. *Every Toeplitz matrix T satisfies an equation of the form*

$$R_1 T - T R_{-1} = AB^T, \tag{7}$$

where $A \in \mathcal{C}^{n \times \ell}$, $B \in \mathcal{C}^{n \times \ell}$, $1 \le \ell \le 2$, and

$$R_\delta = \begin{pmatrix} 0 & 0 & \ldots & 0 & \delta \\ 1 & 0 & \cdots & \cdots & 0 \\ 0 & 1 & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & 0 \end{pmatrix}.$$

The Toeplitz matrix T is unitarily related to a Cauchy-like matrix

$$C = FTS_0^* F^*$$

that satisfies the displacement equation

$$S_1 C - C S_{-1} = (FA)(B^T S_0^* F^*)\,, \tag{8}$$

where

$$\begin{aligned} S_1 &= \text{diag}(1, e^{\frac{2\pi i}{n}}, \ldots, e^{\frac{2\pi i}{n}(n-1)}), \\ S_{-1} &= \text{diag}(e^{\frac{\pi i}{n}}, \ldots, e^{\frac{(2n-1)\pi i}{n}}), \\ S_0 &= \text{diag}(1, e^{\frac{\pi i}{n}}, \ldots, e^{\frac{\pi i}{n}(n-1)}), \end{aligned}$$

and F is the normalized inverse discrete Fourier transform matrix defined by

$$F = \frac{1}{\sqrt{n}} \left[\exp\left(\frac{2\pi i}{n}(j-1)(k-1)\right)\right]_{1 \le j,k \le n}.$$

We note that Toeplitz matrices are unitarily related to Cauchy-like matrices through other fast transformations as well [14]. However, the particular relation in Property 3 can be exploited to determine an $O(n \lg n)$ stable algorithm for multiplication by the inverse of the Cauchy-like matrix [28].

Now suppose that T is Toeplitz block. Then the (k, l) block of T satisfies (7) with $A \equiv A_{kl}, B \equiv B_{kl}$. Thus each block of T is unitarily related to a Cauchy-like matrix as defined by Property 3 above. Therefore, T satisfies

$$(I \otimes F)T(I \otimes S_0^* F^*) = C,$$

where $\otimes$ denotes the Kronecker tensor product and C has Cauchy-like blocks C_{kl} with respective generators $FA_{kl}, (B_{kl}^T S_0^* F^*)^T$. Let G denote the rank $\leq 2p$ matrix with blocks $A_{kl}B_{kl}^T$, and note that G can be written as the outer product of two $N \times 2p$ matrices. Then C is itself Cauchy-like with displacement rank $\ell \leq 2p$ since it is the unique solution to the displacement equation (compare to (4))

$$(I \otimes S_1)C - C(I \otimes S_{-1}) = (I \otimes F)G(I \otimes S_0^* F^*). \tag{9}$$

Factorization algorithms which rely on C having small displacement rank will become expensive if p is large, requiring $O(p(np)^2)$ operations. Instead, we seek an approximation to C which will be block diagonal with Cauchy-like blocks. This approximation will still be a Cauchy-like matrix in the sense that it will solve (9) for a particular rank $\leq 2p$ matrix G; however, by considering the Cauchy-like blocks separately, we observe (refer to section 3.2) that a full factorization of the block diagonal matrix can be obtained stably in only $O(pn^2)$ operations.

3. Regularization and preconditioning.

If we were to solve the linear system (2) exactly, possibly by transforming the BTTB matrix T to a Cauchy-like matrix and factoring, the solution we would compute in this manner would be hopelessly contaminated with noise, as we now discuss. The analysis will be based on the following four assumptions:

1. The matrix T has been normalized so that its largest singular value is $O(1)$.
2. The uncontaminated data vector $\hat{g}$ satisfies the discrete Picard condition; i.e., the spectral coefficients of $\hat{g}$ decay in absolute value faster than the singular values [33, 18].
3. The additive noise is zero-mean white Gaussian. In this case, the components of the error e are independent random variables normally distributed with mean zero and variance ϵ^2.
4. The *noise level*, $\frac{\|e\|_2}{\|\hat{g}\|_2}$, is strictly less than one.

Let $T = U\Sigma V^T$ be the singular value decomposition of T and let f be the exact solution to the noisy system (2). The spectral coefficients of the exact solution $\hat{g}$ and noise e are $\zeta = U^T\hat{g}$ and $\eta = U^T e$, respectively. For the remainder of the paper we will assume that $N = pn$ is the dimension of T. Using (2), we observe that

$$f = \sum_{i=1}^{N} \frac{\zeta_i + \eta_i}{\sigma_i} v_i, \tag{10}$$

where v_i denotes the ith column of V and σ_i denotes the ith diagonal element of the diagonal matrix Σ.

Under the white noise assumption, the coefficients η_i are roughly constant in size, while the discrete Picard condition tells us that the ζ_i go to zero at least as fast as the singular values σ_i. Thus, components for which ζ_i is of the same order as η_i are obscured by noise.

By assumptions 2 and 4, there exists $\bar{m} > 0$ such that for all $i > \bar{m}$, the ζ_i are indeed indistinguishable from the η_i. Further, there exists $0 < m^* \leq \bar{m}$ such that for $i > m^*$ it is never the case that $|\zeta_i| \gg |\eta_i|$. We therefore choose to partition the columns of V into bases for the *upper, lower,* and *transition* subspaces as follows. We say that the *upper* subspace is the space spanned by the first m^* columns of V. Hence, the upper subspace corresponds to the largest m^* singular values. The *lower* subspace is the space spanned by the last $N - \bar{m}$ columns for V, i.e., those columns of V corresponding to the smallest singular values. Finally, the *transition* subspace is the space spanned by the remaining $\bar{m} - m^*$ columns of V. Since these columns correspond to the midrange singular values, the transition subspace is generally difficult to resolve unless there is a gap in the singular value spectrum.

Comparing the exact solution $\hat{f}$ of (1) to f in (10), we see that the greatest difference is in the magnitude of the components in the lower subspace. Thus we choose to use an iterative method called conjugate gradient for least squares (CGLS) which at early iterations produces a regularized solution with small components in the lower subspace and which resembles $\hat{f}$ in the upper subspace. An appropriate preconditioner will speed convergence to this approximate solution without adding components in the lower subspace.

3.1. Regularization by preconditioned conjugate gradients. The standard conjugate gradient (CG) method [22] is an iterative method for solving systems of linear equations for which the matrix is symmetric positive definite. If the matrix is not symmetric positive definite, one can use the CGLS algorithm [22], a variant of standard CG that solves the normal equations in factored form. If the discrete Picard condition holds, then CGLS acts as an iterative regularization method with the iteration index taking the role of the regularization parameter [8, 17, 19]. The spread and clustering of the singular values govern the speed and convergence of the algorithm [32]. Preconditioning is therefore often applied in an effort to cluster the singular values and thus to speed convergence.

When one wishes to solve a linear system exactly, it is desirable to choose the preconditioner to cluster the entire spectrum around some number > 0. Many preconditioners for systems involving matrices with Toeplitz and BTTB structure have been proposed in the literature (see [26, 4]) which provably cluster the spectrum under certain conditions; the interested reader is referred to the recent survey in [5] for an overview of these types of preconditioners. However, according to (10), we desire that the preconditioner cluster *only* the large singular values for which $|\zeta_i| \gg |\eta_i|$. Unfortunately, the indices for which this holds are difficult, if not impossible, to determine in advance. However, as we show in section 4, it is possible to choose a preconditioner that clusters most of the largest m^* singular values while leaving the small singular values, and with them, the lower subspace, relatively unchanged. Hence, the first few iterations of CGLS will quickly capture the solution lying within the upper subspace. Ideally, a modest number of subsequent iterations will provide some improvement over the transition subspace without significant contamination from the noise contained in the lower subspace.

3.2. The preconditioner. The (i, j) block of the given BTTB matrix T is the Toeplitz matrix T_{i-j}, and the (k, l) element of the (i, j) block is given by $(T_{i-j})_{kl} = t^{(i-j)}_{k-l}$. For each T_i, let us define H_i to be its T. Chan circulant approximation [6], so that the diagonals of H_i are given by

$$h_j^{(i)} = \begin{cases} \frac{(n-j)t_j^{(i)} + jt_{j-n}^{(i)}}{n}, & 0 \le j < n, \\ h_{n+j}^{(i)}, & 0 < -j < n. \end{cases}$$

The matrix H_i is the closest circulant matrix in the Frobenius norm to T_i [6]. Finally, we define H to be the BTCB matrix whose (i,j) block is the circulant matrix H_{i-j}. It was shown in [4] that H is the closest BTCB matrix to T in the Frobenius norm. The goal is to develop a preconditioner from an appropriately transformed version of the matrix H.

We define the matrices F and S_0 as in Property 3, with the dimension being either p or n as is appropriate in context. Since the matrices T and H are block Toeplitz, the matrices $(I \otimes F)T(I \otimes F^*)$ and $(I \otimes F)H(I \otimes F^*)$ with their (i,j) blocks given by $FT_{i-j}F^*$ and $FH_{i-j}F^*$, respectively, are also block Toeplitz.

Now since the H_i are circulant, they can be diagonalized by the matrix F [4]; therefore, for each (i,j), $FH_{i-j}F^*$ is diagonal. In section 1 we assumed that the unknowns are ordered first in the increasing α direction, then in order of increasing β. Let $\hat{P}$ be the $N \times N$ permutation matrix which reorders the unknowns in the increasing β direction first. Then

$$\tilde{T} = \hat{P}(I \otimes F)T(I \otimes F^*)\hat{P}^T \tag{11}$$

is a block matrix with Toeplitz $n \times n$ blocks while

$$\tilde{H} = \hat{P}(I \otimes F)H(I \otimes F^*)\hat{P}^T \tag{12}$$

is a block diagonal matrix with $n \times n$ Toeplitz blocks.

Since $\tilde{T}$ has size n Toeplitz blocks, $\tilde{T}$ is related to a Cauchy-like matrix $\tilde{C}$ as mentioned at the end of section 2:

$$\tilde{C} = (I \otimes F)\tilde{T}(I \otimes S_0^* F^*), \tag{13}$$

where F and S_0 now have dimension n. Each block of $\tilde{C}$ is Cauchy-like. Likewise, $\tilde{H}$ is related to a Cauchy-like matrix with Cauchy-like blocks:

$$\tilde{K} = (I \otimes F)\tilde{H}(I \otimes S_0^* F^*). \tag{14}$$

Since $\tilde{H}$ is block diagonal with Toeplitz blocks, $\tilde{K}$ is block diagonal with Cauchy-like blocks. Finally, we observe that solving $Tf = g$ must be equivalent to solving $\tilde{C}\tilde{f} = \tilde{g}$, where $\tilde{f} = (I \otimes FS_0)\hat{P}(I \otimes F)f$, $\tilde{g} = (I \otimes F)\hat{P}(I \otimes F)g$.

As mentioned previously, since $\tilde{C}$ is Cauchy-like, we could apply Gu's factorization algorithm directly to it; however, the cost of a full factorization would be $O(p(np)^2)$ operations. Fortunately, since each of the p blocks $\tilde{K}_{ii}$ is an $n \times n$ Cauchy-like matrix of displacement rank 2, to completely factor $\tilde{K}$, our approximation to $\tilde{C}$, requires only $O(pn^2)$ operations.

A factorization of $\tilde{K}_{ii}$ using a modified complete pivoting strategy may lead to an interchange of rows (specified by a permutation matrix P_i) and columns (specified by a permutation matrix Q_i). Let $P = \mathrm{diag}(P_1, \ldots, P_p)$ and $Q = \mathrm{diag}(Q_1, \ldots, Q_p)$. We will use an appropriate piece of the matrix $P^T\tilde{K}Q^T$, to be defined shortly, to precondition the matrix $P^T\tilde{C}Q^T$. First, we summarize the following sequence of transformations which leads to the development of the preconditioner:

1. Transform the matrices T and H to the Toeplitz block matrices $\tilde{T}$ and $\tilde{H}$ (cf. (11) and (12)). Note that $\tilde{H}$ is also block diagonal.
2. Transform the matrices $\tilde{T}$ and $\tilde{H}$ to Cauchy-like matrices with Cauchy-like blocks $\tilde{C}_{ij}, \tilde{K}_{ij}$, respectively (cf. (13) and (14)).
3. Permute the matrices $\tilde{C}$ and $\tilde{K}$ using the block diagonal permutation matrices P and Q:

$$\begin{aligned} C &= P^T \tilde{C} Q^T, \\ K &= P^T \tilde{K} Q^T. \end{aligned}$$

Note that since all the transformations are accomplished with unitary matrices, C and T have the same singular values, as do K and H.

Hence, setting $y = Q\tilde{f}$ and $z = P^T \tilde{g}$, we wish to solve the problem

$$Cy = z. \tag{15}$$

We choose a left preconditioner M, determined from K, so that $M^{-1}Cy = M^{-1}z$, and use CGLS to solve the corresponding normal equations

$$(M^{-1}C)^*(M^{-1}C)y = (M^{-1}C)^* M^{-1} z. \tag{16}$$

Recall from section 3.1 that we wish to design a preconditioner that clusters the largest m^* singular values while leaving the small singular values unchanged. Notice also that the singular values of K, our approximation to C, are simply the union of the singular values of the $K_{ii} = P_i^T \tilde{K}_{ii} Q_i^T$. Let Γ be the set of the largest m^* singular values of K. Then precisely m_i singular values of K_{ii} are in Γ, with $m^* = \sum_{i=1}^p m_i$. As a result of pivoting using Gu's pivoting scheme, the $m_i \times m_i$ leading submatrix of K_{ii} corresponds to the well-conditioned part of K while the rest contributes to the ill-conditioned part. Let $K_{ii} = L_{ii} D_{ii} U_{ii}$ and write this equation in block form, where the upper left blocks are $m_i \times m_i$:

$$\begin{bmatrix} K_{ii}^{(1)} & K_{ii}^{(2)} \\ K_{ii}^{(3)} & K_{ii}^{(4)} \end{bmatrix} = \begin{bmatrix} L_{ii}^{(1)} & 0 \\ L_{ii}^{(2)} & L_{ii}^{(3)} \end{bmatrix} \begin{bmatrix} D_{ii}^{(1)} & 0 \\ 0 & D_{ii}^{(2)} \end{bmatrix} \begin{bmatrix} U_{ii}^{(1)} & U_{ii}^{(2)} \\ 0 & U_{ii}^{(3)} \end{bmatrix}. \tag{17}$$

Here $L_{ii}^{(1)}, L_{ii}^{(3)}$ are lower triangular, $U_{ii}^{(1)}, U_{ii}^{(3)}$ are upper triangular, and $D_{ii}^{(1)}$ and $D_{ii}^{(2)}$ are diagonal. Then we define

$$M_i = \begin{bmatrix} L_{ii}^{(1)} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} D_{ii}^{(1)} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} U_{ii}^{(1)} & 0 \\ 0 & I \end{bmatrix} = \begin{bmatrix} K_{ii}^{(1)} & 0 \\ 0 & I \end{bmatrix}.$$

Finally, we choose as our preconditioner the matrix

$$M = \operatorname{diag}(M_1, \ldots, M_p).$$

Since leading principal submatrices of Cauchy-like matrices are Cauchy-like, M is a block diagonal matrix with Cauchy-like blocks each augmented by an identity of appropriate (possibly zero) dimension.

Let us compare our preconditioning scheme with the preconditioning method given in [17] (see also [16]) for the BTTB matrices of discrete ill-posed problems. In [17] the preconditioner is determined by forming the T. Chan BCCB approximant to T, computing its eigenvalues via two-dimensional fast Fourier transforms, and then replacing all the eigenvalues below a tolerance with ones. Therefore, our method is similar to their BCCB-based preconditioner in that we also rely on a rank revealing factorization to determine the appropriate cutoff which is used to form the preconditioner. We choose our cutoff tolerance in a manner similar to that given in [17]. However, our preconditioner is formed from a BTCB approximant to T, which requires approximating T only on one level, unlike the BCCB approximant, which requires approximating T on two levels.

The most notable difference is that we rely on a transformation to Cauchy-like matrices; therefore we may use a fast pivoted factorization scheme, rather than two-dimensional Fourier transforms, to generate the necessary rank revealing information. While the preconditioner in [17] requires $O(pn(\lg p + \lg n))$ operations to precompute, our preconditioner requires, in the worst case, $O(pb_w \lg p + pn \lg n + nm^*)$ operations to precompute, where b_w denotes the maximum block bandwidth of the matrix. However, in applications b_w is sometimes small compared to n, and when the blocks of T are symmetric, the number of operations required to initialize our preconditioner can be reduced to $O(pn \lg n + b_w p \lg p + m_s)$, where $m_s = n(m_1 + m_{p/2} + \sum_{i=1}^{p/2} m_i)$ when p is even. In some cases (when the dimension of the upper subspace is small, for example) we have observed that m_s is small relative to $pn \lg p$, which implies our preconditioner can be almost as cheap to precompute. Our preconditioner is competitive with the BCCB matrix in that it is stable to compute and can be applied in at most $O(pn \lg n)$, rather than $O(pn(\lg n + \lg p))$, operations. In the next section, we show that our preconditioner is just as effective as the one in [17] in clustering the large singular values around one. Further, we show that the small singular values remain small and that the upper and lower subspaces remain unmixed.

4. Properties of the preconditioner.

In this section we give theoretical results which show how successful our preconditioner is in filtering noise and accelerating convergence to a regularized solution.

4.1. Clustering.

Under the assumptions in section 3 for an ill-conditioned matrix C, in order for the first few iterations of CGLS to capture the solution corresponding to the largest m^* singular values, the preconditioner must cluster the majority of the m^* singular values while leaving the small singular values and lower subspace essentially unchanged. We show that the question of how well our preconditioner M clusters the singular values can be reduced to the question of how well K approximates C, or, equivalently, how well H approximates T.

We argue as follows. We first note that to show that the largest m^* singular values of $M^{-1}C$ cluster around one, it suffices to show that the smallest m^* singular values of $I - M^{-1}C$ cluster around zero. We denote the kth largest singular value of a matrix Z by $\sigma_k(Z)$ and the kth largest eigenvalue by $\lambda_k(Z)$.

Let $K - C = R$. Now $K = M + S$, where S is block diagonal with blocks

$$S_i = \begin{bmatrix} 0 & K_{ii}^{(2)} \\ K_{ii}^{(3)} & K_{ii}^{(4)} - I \end{bmatrix}.$$

Thus, $M - C = R - S$. We therefore obtain the equality

$$I - M^{-1}C = M^{-1}(R - S) = M^{-1}(K - C) - M^{-1}S. \tag{18}$$

Now let

$$Y_i = \begin{bmatrix} 0 & 0 \\ K_{ii}^{(3)} & K_{ii}^{(4)} \end{bmatrix} \text{ and } Z_i = \begin{bmatrix} 0 & K_{ii}^{(1)-1} K_{ii}^{(2)} \\ 0 & -I \end{bmatrix}.$$

Define E_S and E_M to be the block diagonal matrices

$$E_S = \operatorname{diag}(Y_1, \dots, Y_p) \text{ and } E_M = \operatorname{diag}(Z_1, \dots, Z_p).$$

Then $M^{-1}S = E_S + E_M$, where E_S and E_M each have rank $N - m^*$. From [23, Thm. 3.3.16, part a],

$$\sigma_{k+N-m^*}(M^{-1}S) \leq \sigma_k(E_S), \quad k = 1, \dots, m^*.$$

Applying [23, Thm. 3.3.16, parts a and d] to (18) with $2 \leq i + j \leq N + 1$ for $N - m^* + 1 \leq j \leq N$, we have

$$\begin{aligned}\sigma_{i+j-1}(I - M^{-1}C) &\leq \sigma_i(M^{-1}(K - C)) + \sigma_j(M^{-1}S)\\ &\leq \sigma_1(M^{-1})\sigma_i(K - C) + \sigma_j(M^{-1}S)\\ &\leq \sigma_1(M^{-1})\sigma_i(H - T) + \sigma_{j+m^*-N}(E_S).\end{aligned}$$

In particular,

$$\sigma_{i+N-m^*}(I - M^{-1}C) \leq \frac{\sigma_i(H - T)}{\sigma_N(M)} + \sigma_1(E_S), \quad i = 1, \dots, m^*. \tag{19}$$

Hence, under the assumptions that the preconditioner is well-conditioned and that the matrix E_S has sufficiently small elements, the clustering of the singular values of $I - M^{-1}C$ around zero depends on the clustering of the first m^* singular values of $H - T$ around zero. We now discuss two special cases for which $H - T$ has singular values clustered around zero. First, we will need the following lemma from [4].

Lemma 4.1 (R. Chan and Q. Jin). *Assume that the BTTB matrix T is symmetric. Let the entries of block T_i be denoted $t_{jk}^{(i)} = t_{|j-k|}^{(i)}$ for $1 \leq j, k \leq p$, $1 \leq i \leq n$. Assume that the generating sequence of T is absolutely summable, i.e.,*

$$\sum_{i=0}^{\infty} \sum_{j=0}^{\infty} |t_j^{(i)}| \leq J < \infty.$$

Then for all $\epsilon > 0$, there exist $j^*, k^* > 0$ such that for all $p > k^*$ and $n > 0$, at most nj^* eigenvalues of $H - T$ have absolute values exceeding ϵ.

Since $H - T$ is symmetric for symmetric BTTB matrices T, combining (19) with Lemma 4.1, we obtain the following.

Theorem 4.2. *Assume T is a symmetric BTTB matrix with an absolutely summable generating sequence. Then for all $\epsilon > 0$, there exists a $k^* > 0$ such that for all $p > k^*$ and $n > 0$, at most $nj^* + N - m^*$ singular values of $I - M^{-1}C$ exceed $\frac{\epsilon}{\sigma_N(M)} + \sigma_1(E_S)$.*

In other words, at least $m^* - nj^*$ singular values of $I - M^{-1}C$ are less than or equal to $\frac{\epsilon}{\sigma_N(M)} + \sigma_1(E_S)$, as desired.

Let us consider another special case for which we are guaranteed clustering. Let $C_{2\pi}$ denote the Banach space of all 2π-periodic, continuous, complex-valued functions equipped with the norm $\|\cdot\|_\infty$. This class of functions contains the Wiener class [2]. For all $h \in C_{2\pi}$, let the Fourier coefficients of h be defined by

$$\bar{t}_k = \frac{1}{2\pi} \int_{-\pi}^{\pi} h(\theta) e^{ik\theta} d\theta, \quad k = 0, \pm 1, \pm 2, \dots,$$

where $i = \sqrt{-1}$. Let $\bar{T}$ be the $p \times p$ complex Toeplitz matrix whose diagonals are given by $\bar{t}_k$, and let $\bar{H}$ be its T. Chan circulant approximation. Finally, let the BTTB matrix T be given as $T = \bar{R} \otimes \bar{T}$, where $\bar{R}$ is a nonsingular $n \times n$ matrix. A lemma proved by R. Chan and M. Yeung [3] will be useful.

LEMMA 4.3 (R. Chan and M. Yeung). *Let $h \in C_{2\pi}$. Then for all $\epsilon > 0$, there exist k^* and $j^* > 0$, such that for all $p > k^*$,*

$$\bar{T} - \bar{H} = \bar{U} + \bar{V},$$

where

$$\operatorname{rank}(\bar{U}) \leq j^* \quad \textit{and} \quad \|\bar{V}\|_2 \leq \epsilon.$$

Applying this lemma to T, we obtain the following result.

LEMMA 4.4. *Given $\epsilon > 0$, let k^*, j^*, $\bar{U}$, and $\bar{V}$ be defined as in Lemma 4.3 with $h \in C_{2\pi}$. Then*

$$\sigma_i(T - H) \leq \|\bar{R}\|_2 \epsilon, \quad nj^* + 1 \leq i \leq N.$$

Proof. Using [23, Lems. 3.3.16 and 4.2.15],

$$\sigma_i(T - H) \leq \sigma_i(\bar{R} \otimes \bar{U}) + \sigma_1(\bar{R} \otimes \bar{V}) \leq \sigma_i(\bar{R} \otimes \bar{U}) + \|\bar{R}\|_2 \epsilon, \quad i = 1, \ldots, N.$$

However, since $\bar{U}$ has rank j^*, the rank of $\bar{R} \otimes \bar{U}$ is nj^*, so that

$$\sigma_i(T - H) \leq \|\bar{R}\|_2 \epsilon, \quad nj^* + 1 \leq i \leq N. \qquad \square$$

We use Lemma 4.4 and (19) to deduce the following.

THEOREM 4.5. *Let the BTTB matrix T be defined as $T = \bar{R} \otimes \bar{T}$ for a given $n \times n$ nonsingular matrix $\bar{R}$, and let the entries of the $p \times p$ matrix $\bar{T}$ be given by $\bar{t}_{k-j}$ defined above with $h \in C_{2\pi}$. Then for all $\epsilon > 0$, there exist k^* and $j^* > 0$, such that for all $p > k^*$, at most $nj^* + N - m^*$ singular values of $I - M^{-1}C$ exceed $\frac{\|\bar{R}\|_2 \epsilon}{\sigma_N(M)} + \sigma_1(E_S)$.*

In both the aforementioned cases, assuming the values m_i were chosen appropriately, the preconditioner will cluster most of the m^* singular values of the preconditioned matrix around one when a sufficient number of the singular values of $T - H$ are small. As these special cases illustrate, a proof that our preconditioner is effective at clustering the large singular values is reduced to a proof that many of the singular values of $T - H$ are small for the given BTTB matrix T.

4.2. Unmixing results.

Recall that the transformation from the problem involving T to one involving C was accomplished using a sequence of unitary transforms. Thus, the singular values of T and C are the same, as is our definition of the upper, lower, and transition spaces in section 3. That is, we have changed the bases for the respective spaces, but we have not mixed them.

For the approximate solution generated by CGLS in early iterates to be essentially unaffected by noisy components in the lower subspace, we require that the preconditioner not mix the upper and lower subspaces. The following theorem tells the extent to which preconditioning by M mixes these subspaces.

THEOREM 4.6. *Let k be the dimension of the subspace corresponding to the smallest k singular values and let*

$$C = [Q_1\, Q_2\, Q_3] \begin{bmatrix} \Sigma_1 & 0 & 0 \\ 0 & \Sigma_2 & 0 \\ 0 & 0 & \Sigma_3 \end{bmatrix} \begin{bmatrix} V_1^* \\ V_2^* \\ V_3^* \end{bmatrix},$$

$$M^{-1}C = \left[\hat{Q}_1\, \hat{Q}_2\, \hat{Q}_3\right] \left[\begin{array}{ccc} \hat{\Sigma}_1 & 0 & 0 \\ 0 & \hat{\Sigma}_2 & 0 \\ 0 & 0 & \hat{\Sigma}_3 \end{array}\right] \left[\begin{array}{c} \hat{V}_1^* \\ \hat{V}_2^* \\ \hat{V}_3^* \end{array}\right]$$

be singular value decompositions with $V_3, \hat{V}_3 \in \mathcal{C}^{N\times k}$ *and* $V_1, \hat{V}_1 \in \mathcal{C}^{N\times m^*}$. *Then*

$$\|V_1^*\hat{V}_3\|_2 \leq \frac{\hat{\sigma}_{N-k+1}}{\sigma_{m^*}}(\max\{1, \max_i \|K_{ii}^{(1)}\|_2\}).$$

Proof. Using the decompositions, we have

$$\begin{aligned} V_1^*\hat{V}_3 &= (V_1^*C^{-1})M(M^{-1}C\hat{V}_3) \\ &= \Sigma_1^{-1}Q_1^*M\hat{Q}_3\hat{\Sigma}_3. \end{aligned}$$

Since Q_1^* has orthonormal columns, as does $\hat{Q}_3$, it follows that

$$\|V_1^*\hat{V}_3\|_2 \leq \frac{\hat{\sigma}_{N-k+1}}{\sigma_{m^*}}\|M\|_2 = \frac{\hat{\sigma}_{N-k+1}}{\sigma_{m^*}}(\max\{1, \max_i \|K_{ii}^{(1)}\|_2\}). \qquad \square$$

We note that if the preconditioner developed in [17] for their right preconditioning scheme is applied to the left rather than the right, a similar result can be obtained.

Next we show that $\hat{\sigma}_j \approx \sigma_j$ for σ_j corresponding to the last $N - m^*$ singular values, and thus $\hat{\sigma}_{N-k+1}$ is small. Hence, if M is well-conditioned, we are guaranteed that the upper and lower subspaces remain unmixed.

For given values of m_i, we first rewrite C in block form:

$$C = \left[\begin{array}{cccc} C_{11}^{(1)} & C_{11}^{(2)} & \dots & C_{1n}^{(2)} \\ C_{11}^{(3)} & C_{11}^{(4)} & \dots & C_{1n}^{(4)} \\ \vdots & \vdots & \vdots & \vdots \\ C_{n1}^{(1)} & C_{n1}^{(2)} & \dots & C_{nn}^{(2)} \\ C_{n1}^{(3)} & C_{n1}^{(4)} & \dots & C_{nn}^{(4)} \end{array}\right],$$

where $C_{ii}^{(1)}$ is $m_i \times m_i$ and $C_{ii}^{(4)}$ is $n - m_i \times n - m_i$. Likewise, we rewrite $M^{-1}C$:

$$M^{-1}C = \left[\begin{array}{cccc} K_{11}^{(1)-1}C_{11}^{(1)} & K_{11}^{(1)-1}C_{11}^{(2)} & \dots & K_{11}^{(1)-1}C_{1n}^{(2)} \\ C_{11}^{(3)} & C_{11}^{(4)} & \dots & C_{1n}^{(4)} \\ \vdots & \vdots & \vdots & \vdots \\ K_{nn}^{(1)-1}C_{n1}^{(1)} & K_{nn}^{(1)-1}C_{n1}^{(2)} & \dots & K_{nn}^{(1)-1}C_{nn}^{(2)} \\ C_{n1}^{(3)} & C_{n1}^{(4)} & \dots & C_{nn}^{(4)} \end{array}\right].$$

Define the $m^* \times N$ matrices E_{M_1}, E_{C_1} to contain the odd row-blocks of C and $M^{-1}C$, respectively, and let E_1 be the $N - m^* \times N$ matrix containing the even row-blocks of C (or $M^{-1}C$).

Under this partitioning, it easy to verify the relations

$$\begin{array}{lcl} (M^{-1}C)^*(M^{-1}C) & = & E_M + E, \\ C^*C & = & E_C + E, \end{array} \qquad (20)$$

where $E_M = E_{M_1}^*E_{M_1}$, $E_C = E_{C_1}^*E_{C_1}$, and $E = E_1^*E_1$. Since E_M and E_C have rank m^* and E has rank $N - m^*$, we obtain the following.

THEOREM 4.7. *The $(m^* + i)$th singular value of each of the matrices C and $M^{-1}C$ lies in the interval $[0, \sigma_i(E)]$ for $i = 1, \ldots, N - m^*$.*

Proof. Since the matrices in (20) are all Hermitian, we may apply Corollary IV.4.9 and problem 4, page 211, of [30] to obtain

$$\lambda_N(E) + \lambda_{m^*+i}(E_M) \leq \lambda_{m^*+i}((M^{-1}C)^*(M^{-1}C)) \leq \lambda_{m^*+1}(E_M) + \lambda_i(E)$$

and

$$\lambda_N(E) + \lambda_{m^*+i}(E_C) \leq \lambda_{m^*+i}(C^*C) \leq \lambda_{m^*+1}(E_C) + \lambda_i(E).$$

However, $\lambda_N(E) = 0$ and $\lambda_{m^*+1}(E_M) = 0 = \lambda_{m^*+1}(E_C)$, and thus

$$0 \leq \lambda_{m^*+i}((M^{-1}C)^*(M^{-1}C)) \leq \lambda_{m^*+1}(E_M) + \lambda_i(E) = \lambda_i(E)$$

and

$$0 \leq \lambda_{m^*+i}(C^*C) \leq \lambda_{m^*+1}(E_C) + \lambda_i(E) = \lambda_i(E).$$

The proof is completed by taking square roots. □

4.3. Properties of the factorization. The theorems in section 4.1 and section 4.2 show that the preconditioner will be effective under two restrictions. First, M, and hence each $K_{ii}^{(1)}$, must be well-conditioned. Second, the entries in E and E_S are required to be small. We now discuss to what extent these conditions hold for integral equation discretizations. It was discovered in [28] that if T satisfies the following property, then we expect these two conditions to hold. We refer the reader to [28] for details.

PROPERTY 4. *Suppose T is the BTTB matrix which results from the discretization of a smooth kernel t, normalized so the maximum element is one. For $l = 1, \ldots, p$, let $\tilde{K}_{ll} = F\tilde{H}_l S_0^* F^*$, where $\tilde{H}_l$ is an $n \times n$ Toeplitz matrix from the block diagonal of the matrix $\tilde{H}$ defined in section 3.2. Then for n sufficiently large, there exists an $\epsilon \ll 1$ and $m_l \ll n$ such that all the elements of $\tilde{K}_{ll}$ are less than ϵ in magnitude except for those located in the four corner blocks of total dimension $m_l \times m_l$.*

Consequently, each Cauchy-like block of $\tilde{K}$ can be permuted to contain the large elements in its upper left block. We have observed that when Gu's algorithm is applied to $\tilde{K}_{ll}$ described in Property 4, his pivoting strategy is such that $K_{ll}^{(1)}$ will contain the four corner blocks. Any pivoting strategy that yields this type of permutation will give a reasonable preconditioner for our scheme. We refer the interested reader to [14] for details of Gu's modified, complete pivoting strategy.

Thus, the components of the matrix E_S in section 4.1 are small, and therefore our preconditioner has the property that the largest singular values of the preconditioned matrix are clustered, provided that the singular values of $K - C$ are small. But if the singular values of $K - C$ are small, then the matrix E in section 4.2 necessarily has small singular values. Hence, the invariant subspace corresponding to the small singular values of C is not much perturbed by preconditioning. We therefore expect that the initial iterations of CGLS applied to the preconditioned system will produce a solution that is a good approximation to the noise-free solution $\hat{f}$.

5. Algorithmic issues. Our algorithm is as follows:

```
ALGORITHM 1. SOLVING $Tf = g$
1. Compute the generators for each submatrix $K_{ii}$ (see section 5.3).
2. For each $i$, determine the size $m_i$ of the partial factorization and
   factor $K_{ii} = P_i L_{ii} D_{ii} U_{ii} Q_i$.
3. Set $P = \mathrm{diag}(P_1, \ldots, P_p)$, $Q = \mathrm{diag}(Q_1, \ldots, Q_p)$, $z = P^T F \tilde{g}$.
4. For $i = 1, \ldots, n$, determine the generators of the $m_i \times m_i$ leading
   submatrix, $K_{ii}^{(1)}$, of $K_{ii}$ and let $M_i = \left[ \begin{smallmatrix} K_{ii}^{(1)} & 0 \\ 0 & I \end{smallmatrix} \right]$ (see section 5.3.).
5. Set $M = \mathrm{diag}(M_1, \ldots, M_p)$ and compute $M^{-1}$ as in section 5.3.
6. Compute an approximate solution $\tilde{y}$ to $M^{-1} C y = M^{-1} z$ using a
   few steps of CGLS where matrix-vector products involving $C =
   P^T \tilde{C} Q^T$ are formed without forming $C$ itself (see section 5.2).
7. The approximate solution in the original coordinate system is
   $f = S_0^* F^* Q^T \tilde{y}$.
```

A few comments about the algorithm are in order. First, the submatrices $K_{ii}^{(1)}$ and the matrix M are never actually formed; with only the easily determined generators of $K_{ii}^{(1)}$ and its factors, we can compute matrix-vector products with M^{-1} in $O(pn \lg n)$ operations (see section 5.4). Second, when to stop the CGLS iteration in order to get the best approximate solution is a well-studied but open question (for instance, see [20] and the references therein). We do not solve this problem, but we consider other algorithmic issues in the following subsections.

5.1. Determining the size of the $K_{ii}^{(1)}$.

As shown in section 4, the choices of the parameters m_i determine the number of clustered singular values in the preconditioned system. Since p partial factorizations of size m_i need to be computed, they influence the number of initialization operations. Most importantly, as Theorem 4.6 indicates, $m^* = \sum_{i=1}^{p} m_i$ influences the mixing of upper and lower subspaces. We use a simple heuristic in our numerical experiments. Given the noisy right-hand-side vector g, let G be the $n \times p$ matrix with entries given by $G_{kj} = g_{(k-1)n+j}$ for $1 \leq k \leq n$, $1 \leq j \leq p$, and let $\tilde{G}$ be its two-dimensional, normalized, inverse discrete Fourier transform. Then it is easy to show that the right-hand-side $\tilde{g}$ defined in section 3.2 results from stacking $\tilde{G}$ by columns. We sort the absolute values of $\tilde{g}$ and determine m^* to be the index of the value, *ctol*, for which the Fourier coefficients start to level off. This is presumed to be the noise level. Since $\tilde{G}$ requires $O(np(\lg p + \lg n))$ operations to compute, the cost involved in determining m^* is also $O(np(\lg p + \lg n))$ operations.

We choose the values m_i using two slightly different methods, which we now describe, and we compare the results in section 6. In the first approach, each value m_i is defined as the number of elements in the ith column of $\tilde{G}$ which are larger than *ctol*. We call this method of computing the values m_i the Fourier coefficient method. In the second approach, a full factorization is performed on each block $\tilde{K}_{ii}$ so that all the entries of each diagonal matrix D_{ii} are known. We set d to be the N-length vector comprised of the diagonals of the D_{ii}, sort the elements in decreasing magnitude, and set *dtol* to be the m^*th largest magnitude element. The value m_i is then defined to be the number of diagonal entries in D_{ii} which have magnitude greater than *dtol*. This is the d-selection method for computing the values m_i.

The latter approach appears to be the more expensive of the two, requiring $O(pn^2)$

operations to compute all values m_i. However, we found that the entries of D_{ii} decay nearly monotonically so that the values m_i could be similarly obtained by performing the steps of the factorization of each block in parallel. That is, the first step of the factorization is performed on all the blocks sequentially, then the second step on all the blocks, and so forth, up to step j. Block $\tilde{K}_{ii}$ ceases to be factored after step j when $|(D_{ii})_{jj}|$ is determined to be too small. Hence, with careful administration, all values m_i can be computed from the diagonal entries in D_{ii} as they accumulate in $O(n\sum_{i=1}^{p} m_i)$ operations, where the values m_i are almost identical to the values obtained in our second approach.

5.2. Matrix-vector products with *C*. Recall that C is related to the original BTTB matrix T through a sequence of fast unitary transforms and permutation matrices described in section 3.2. Since matrix-vector products involving T can be computed in $O(np(\lg n + \lg p))$ operations, matrix-vector products involving C can also be computed in $O(np(\lg n + \lg p))$ operations without ever having to compute the entries of C.

5.3. Computing the preconditioner. Since the entries of $\tilde{K}$ can be written in terms of the generators of each block, it is necessary to discuss how these generators are obtained. Because each block of H is circulant, the nonzero entries of $(I \otimes F)H(I \otimes F^*)$, which lie on the diagonals of each of its blocks, are the eigenvalues of each block of H. Let $\lambda_k^{(l)}$ for $1 \le k \le p$, $1 - n \le l \le n - 1$ denote the kth eigenvalue of block H_l. It is well known that the eigenvalues of a $p \times p$ circulant matrix can be computed by means of an FFT in $O(p \lg p)$ operations. Since there are at most $2n$ distinct H_l, all the $\lambda_k^{(l)}$ can be computed in at most $O(np \lg p)$ operations. The matrix $\hat{P}$ permutes these eigenvalues so that $\tilde{H} = \hat{P}(I \otimes F)H(I \otimes F^*)\hat{P}^T$ is a block diagonal matrix with p, $n \times n$ Toeplitz matrices $\tilde{H}_i$ on its diagonal. The diagonals of $\tilde{H}_i$ are given by $\tilde{t}_j^{(i)} = \lambda_i^j$, $1 - n \le j \le n - 1$, $i = 1, \dots, p$.

Now $\tilde{H}_i$ is Toeplitz, so it satisfies (7) with $A = A_i$ and $B = B_i$. Examination of (7) shows that the entries of the $n \times l$ matrices A_i and B_i are easily determined. The first column of A_i is the first unit vector, and the second column is given by

$$(21)\quad [0, \tilde{t}_{n-1}^{(i)}, \tilde{t}_{n-2}^{(i)}, \dots, \tilde{t}_{p-1}^{(i)}, \dots, \tilde{t}_{1}^{(i)}]^T + [\tilde{t}_{0}^{(i)}, \tilde{t}_{-1}^{(i)}, \tilde{t}_{-2}^{(i)}, \dots, \tilde{t}_{-(q-1)}^{(i)}, \dots, \tilde{t}_{-(n-1)}^{(i)}]^T.$$

The first column of B_i is

$$(22)\, [\tilde{t}_{-(n-1)}^{(i)}, \tilde{t}_{-(n-2)}^{(i)}, \dots, \tilde{t}_{-(q-1)}^{(i)}, \dots, \tilde{t}_{-1}^{(i)}, \tilde{t}_{0}^{(i)}]^T - [\tilde{t}_{1}^{(i)}, \tilde{t}_{2}^{(i)}, \dots, \tilde{t}_{p-1}^{(i)}, \dots, \tilde{t}_{n-1}^{(i)}, 0]^T$$

and the second column is the last unit vector. The generators for $\tilde{K}_{ii}$ are then $\tilde{A}_i \equiv FA_i$ and $\tilde{B}_i \equiv conj(FS_0)B_i$, where $conj(\cdot)$ denotes complex conjugation, with F and S_0 as described in Property 3. Since $\tilde{A}_i$ and $\tilde{B}_i$ can be computed by means of the inverse fast Fourier transform of size n, computing all p generator pairs requires a total of $O(np \lg n)$ operations.

By Property 3, $\tilde{K}_{ii}$ satisfies the displacement (4) with $\Omega = S_1$ and $\Theta = S_{-1}$. Therefore, using Property 1 $K_{ii}^{(1)}$ satisfies

$$\Omega_1 K_{ii}^{(1)} - K_{ii}^{(1)} \Theta_1 = A_i^{(1)} B_i^{(1)T},$$

where Ω_1 and Θ_1 are the leading principal submatrices of $P_i^T \Omega P_i$ and $Q_i \Theta Q_i^T$, respectively, and $A_i^{(1)}$ and $B_i^{(1)}$ contain the first m_i rows of $P_i^T \tilde{A}_i$ and $Q_i^T \tilde{B}_i$, respectively.

From Property 5, the expression for the entries of $K_{ii}^{(1)-1}$ is

$$K_{ii}^{(1)-1} = -\left(\frac{x_j^{(i)T} w_k^{(i)}}{\tilde{\theta}_j - \tilde{\omega}_k} \right)_{1 \le j,k \le n}, \tag{23}$$

where $\tilde{\theta}_j$ and $\tilde{\omega}_k$ are the elements of Θ and Ω that appear in Θ_1 and Ω_1, respectively, and, from (6), the vectors $x_j^{(i)}$ and $w_k^{(i)}$ are rows of $X_i^{(1)}$ and $W_i^{(1)}$ defined as

$$K_{ii}^{(1)} X_i^{(1)} = A_i^{(1)}, \;\; W_i^{(1)T} K_{ii}^{(1)} = B_i^{(1)T}.$$

Computing $X_i^{(1)}$ and $W_i^{(1)}$ costs $O(m_i^2)$ operations, given the factorization of $K_{ii}^{(1)}$ and the matrices $A_i^{(1)}$ and $B_i^{(1)}$. Since $M^{-1} = \mathrm{diag}(M_1^{-1}, \ldots, M_p^{-1})$, it takes $O(\sum_{i=1}^p m_i^2)$ operations to precompute M^{-1} given the matrices $\tilde{A}_i, \tilde{B}_i$ for $i = 1, \ldots, p$. As discussed at the beginning of this section, finding $\tilde{A}_i$ and $\tilde{B}_i$ requires $O(b_w p \lg p + pn \lg n)$ operations to precompute. The cost of computing the permutation matrices P_i and Q_i necessary to determine $A_i^{(1)}$ and $B_i^{(1)}$ is $O(nm_i)$ operations. Thus, the cost for precomputing M^{-1} is $O(b_w p \lg p + pn \lg n + +n \sum_{i=1}^n m_i)$ operations.

5.4. Applying the preconditioner. Since M^{-1} is block diagonal, to compute $M^{-1}r$ requires the p computations $K_{ii}^{(1)-1} r_i$ where r_i is the length m_i subvector of r beginning at index $ip + 1$. Using Algorithm 2 of [28], we compute each $K_{ii}^{(1)-1} r_i$ stably in $O(n \lg n)$ operations. Thus each application of the preconditioner costs at most $O(pn \lg n)$ operations. (If $p_m < p$ of the values m_i are nonzero, as we often found in practice, the cost reduces to $O(p_m n \lg n)$ operations.) Since matrix-vector products involving C can be computed in $O(np(\lg p + \lg n))$ operations, each iteration of CGLS costs $O(np(\lg p + \lg n))$ operations.

6. Numerical results.

In this section we summarize results of our algorithm on two test problems using Matlab and IEEE floating point double precision arithmetic. Our measure of success in filtering noise is the *relative error*, the 2-norm of the difference between the computed estimate f and the vector $\hat{f}$ corresponding to zero noise, divided by the 2-norm of $\hat{f}$. In each case, we apply the CGLS iteration with block Cauchy-like preconditioner with $m^* = \sum_{i=1}^p m_i$. The value $m^* = 0$ corresponds to no preconditioning.

6.1. Example 1. As mentioned in the introduction, BTTB matrices often arise in two-dimensional image processing problems. As an example, we consider the BTTB matrix $T = T_0 \otimes T_0$, where T_0 is the 32×32 Toeplitz matrix with diagonals (see [24])

$$h_k = \begin{cases} c \dfrac{\sin(\frac{k}{B_L})}{\frac{k}{B_L}}^2, & 0 \le |k| \le B_w, \\ 0 & \text{otherwise}, \end{cases}$$

where c is a normalization constant; $B_L = 2$; and B_w, the bandwidth of T_0, is set to 5. The condition number of T is approximately 1.6×10^8.

We then generate $\hat{f}$ by forming the image shown in Figure 1 and stacking it in columns. The image itself was created by truncating to radius 8 a two-dimensional Gaussian with standard deviation 30 centered at row 20, column 19, and multiplying the values by 40. A 3×3 spike of height 40 with upper-left corner at row 13, column 10, was also added. Next, we set $g = T\hat{f} + e$, where e is a normally distributed random

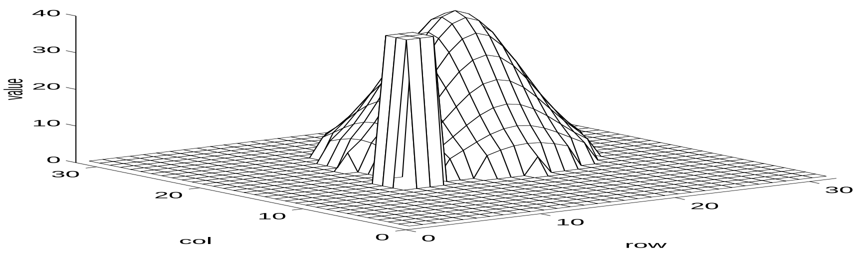


FIG. 1. *Original image, Example* 1.

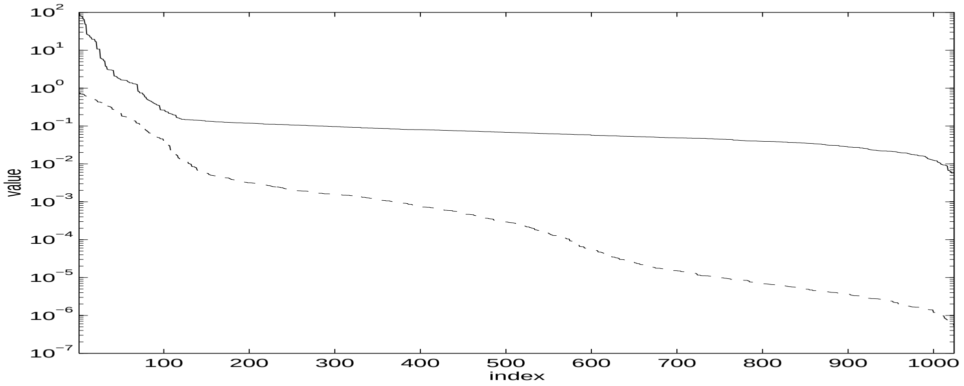


FIG. 2. *Solid line shows two-dimensional Fourier coefficients of the noisy data sorted in order of decreasing magnitude. Dashed line shows diagonal entries obtained during factorization, sorted in order of decreasing magnitude, Example* 1.

vector, generated with the Matlab randn function, scaled so that the noise level was 10^{-2}. The sorted absolute two-dimensional Fourier coefficients of G (cf. section 5.1) together with the vector d are shown in Figure 2.

The solid line in Figure 3 shows the convergence of CGLS in relative error for Example 2. With no preconditioning (i.e., $m^* = 0$) CGLS required 49 iterations to achieve its minimum relative error value of 2.54×10^{-1}. The dashed line in Figure 3 depicts the convergence of CGLS on the left-preconditioned system using our preconditioner with $m^* = 122$ and where the d-method for selecting the m_i is used. After seven iterations, a value of 2.59×10^{-1} was achieved. The dotted line in the figure shows the convergence behavior on the left-preconditioned system when $m^* = 109$ using the Fourier coefficient method of determining m_i. After six iterations, the minimum relative error of 2.66×10^{-1} was reached. In comparison, the dash-dotted line illustrates the optimal convergence behavior of the right-preconditioned scheme in [17], where the cutoff was determined to be 116 eigenvalues. This method achieves a minimum relative error value of 2.53×10^{-1} in seven iterations.

In fact, we note that the matrix in Example 1 is a special example of BTTB matrices arising from tensor products of Toeplitz matrices. In the case where $T = T_1 \otimes T_2,\ T_1 \neq T_2$, it is easy to show that the cost of precomputing our preconditioner is reduced to $O(nm_1 + p \lg p + n \lg n)$ operations. Likewise, the cost of precomputing the preconditioner in [17] reduces to $O(p \lg p + n \lg n)$ operations. (These costs do not include the cost of determining the cutoffs.)

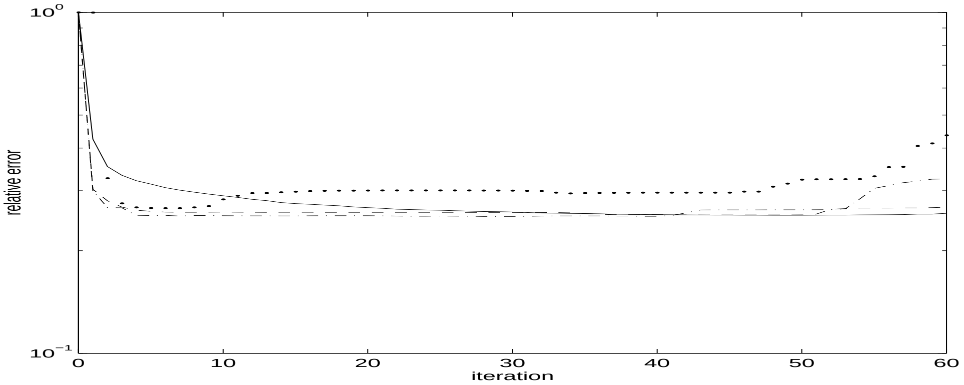


FIG. 3. *Relative error in computed solutions for Example* 1. *Solid line shows convergence for* $m^* = 0$*; dashed line shows convergence for preconditioner with* $m^* = 122$ *using d-selection method; dotted line shows convergence when* $m^* = 109$ *using Fourier coefficient selection method; dash-dotted line shows convergence for the preconditioning scheme* [17] *with cutoff at* 116 *eigenvalues.*

6.2. Example 2. In Example 1, the matrix T was symmetric and its block bandwidth was small relative to the number p of blocks. Since unsymmetry and larger bandwidth can be encountered in practice, we consider the nonsymmetric matrix with a larger block bandwidth as follows. We set $t_j^{(1)}$ and $t_j^{(2)}$ to be the length 32 vectors with entries given by

$$\begin{aligned} t_j^{(1)} &= c_1 e^{(-.1(1-j)^2)} \text{ if } j \leq 6 \text{ and 0 otherwise,} \\ t_j^{(2)} &= c_2 e^{(-.2(1-j)^2)} \text{ if } j \leq 11 \text{ and 0 otherwise,} \end{aligned}$$

where c_1 and c_2 were normalization constants, and we set $\bar{T}$ to be the Toeplitz matrix with first column t_1 and first row t_2 using the Matlab command $\bar{T} = toep(t^{(1)}, t^{(2)})$. We then generate a matrix H as shown in Example 1 with $B_L = 1$ and $B_w = 12$. Finally, we form T by tensor products: $T = H \otimes \bar{T}$. The condition number is 10^9.

For this example we took the exact solution $\hat{f}$ to be same as in Example 2. Next, we set $g = T\hat{f} + e$, where e is a normally distributed random vector scaled so that the noise level was 10^{-3}. The sorted absolute two-dimensional Fourier coefficients of G and the vector d are shown in Figure 4.

The relative error plot in Figure 5 shows that with no preconditioning, CGLS reaches its minimum relative error value of 1.05×10^{-1} at 121 iterations. However, using our preconditioner with $m^* = 576$ and the d-selection method for determining the m_i, a relative error value of 1.06×10^{-1} was reached in only eight iterations. When $m^* = 435$ and the Fourier coefficient selection method is used to determine our preconditioner, a relative error value of 1.44×10^{-1} was reached in nine iterations; after 30 iterations, the relative error was improved to 1.18×10^{-1}. In contrast, the preconditioned iterative scheme in [17] could do no better than 1.24×10^{-1} after 17 iterations (achieved when the cutoff was 574 eigenvalues).

6.3. Example 3. Finally, we consider an example for which T is not a tensor product of Toeplitz matrices. The entries of the 1024×1024 BTTB matrix T are given by

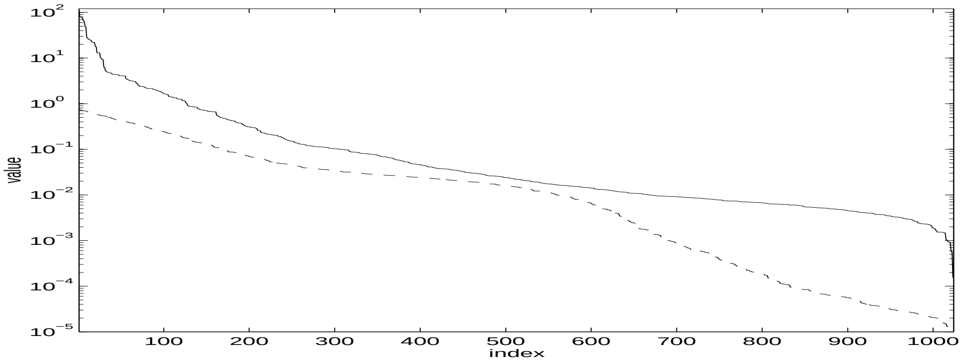


FIG. 4. *Solid line shows two-dimensional Fourier coefficients of the noisy data sorted in order of decreasing magnitude. Dashed line shows diagonal entries obtained during factorization, sorted in order of decreasing magnitude, Example* 2.

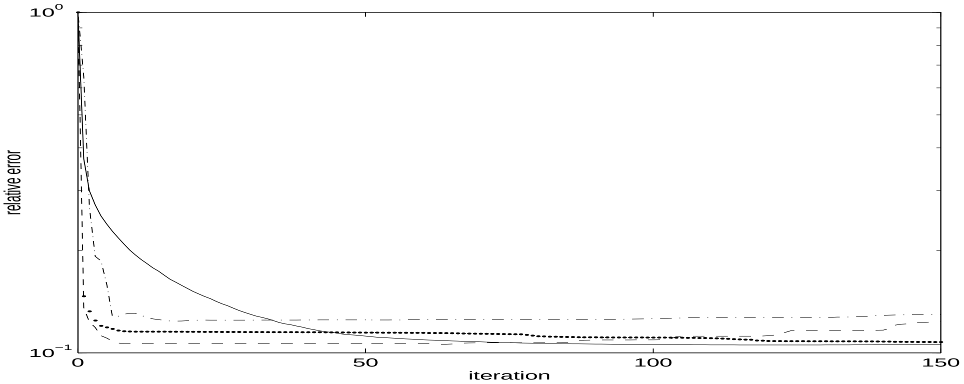


FIG. 5. *Relative error in computed solutions for Example* 2. *Solid line shows convergence for* $m^* = 0$*; dashed line shows convergence for preconditioner with* $m^* = 576$ *using d-selection method; dotted line shows convergence when* $m^* = 435$ *using Fourier coefficient selection method; dash-dotted line shows convergence for the preconditioning scheme* [17] *with cutoff at* 574 *eigenvalues.*

$$(T_{ij})_{kl} = \begin{cases} \exp(-.05(|i-j| + |k-l|)^2), & |i-j| < 9 \text{ and } |k-l| < 9, \\ 0 & \text{otherwise.} \end{cases}$$

The condition number of this matrix was 2.62×10^5.

Again, we took the same exact solution $\hat{f}$ as in the previous two examples. The noise level for this problem was 10^{-3}. In Figure 6 we plot both the sorted absolute two-dimensional Fourier coefficients of G and the sorted absolute values of the vector d.

With no preconditioning, CGLS requires 168 iterations to reach its minimum relative error of 1.137×10^{-1}, as illustrated in Figure 7. In comparison, we found that the preconditioned iterative scheme in [17] could not achieve a smaller relative error than 1.280×10^{-1} for any cutoff value in fewer than 180 iterations; this particular relative error was achieved in 54 iterations when the cutoff was 319 eigenvalues (see Figure 7). Setting $m^* = 307$ and using the d-selection method of determining the m_i gave the overall best relative error; a minimum relative error value of 1.136×10^{-1} was reached in 48 iterations. However, for $m^* = 224$, a minimum relative error value of 1.138×10^{-1} was reached in only 38 iterations (see Figure 7). In fact, we found that

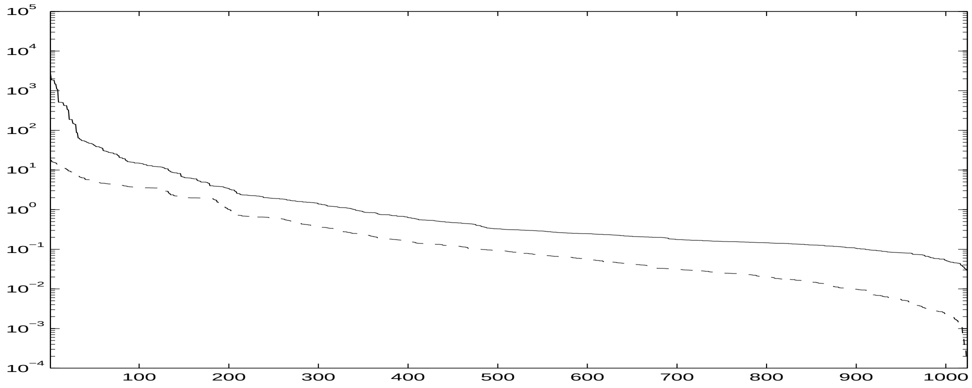


FIG. 6. *Solid line shows two-dimensional Fourier coefficients of the noisy data sorted in order of decreasing magnitude. Dashed line shows diagonal entries obtained during factorization, sorted in order of decreasing magnitude, Example* 3.

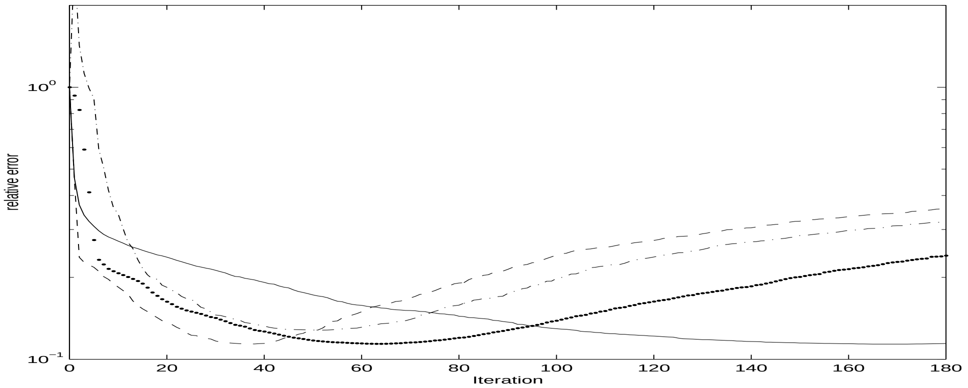


FIG. 7. *Relative error in computed solutions for Example* 3. *Solid line shows convergence for* $m^* = 0$*; dashed line shows convergence for preconditioner with* $m^* = 224$ *using d-selection method; dotted line shows convergence when* $m^* = 165$ *using Fourier coefficient selection method; dash-dotted line shows convergence for the preconditioning scheme* [17] *with cutoff at* 319 *eigenvalues.*

there was a large range of values for m^* (150 to 347) for which our preconditioned CGLS scheme (with the d-selection method of choosing m_i) gave solutions having near-minimal (< 1.15) relative error in 35 to 60 iterations. Using the Fourier selection method for choosing the m_i, our preconditioned iterative scheme took 64 iterations to reach a minimum relative error value of 1.139×10^{-1} when $m^* = 165$. For this selection scheme, we also found a large range of values m^* for which this preconditioner gave comparable relative error minima in 58 to 70 iterations.

6.4. Results summary. We conducted several other experiments comparing the effectiveness of our preconditioner with the effectiveness of the preconditioner found in [17] (see also [16]). The experiments, which we now summarize, were conducted using matrices of different sizes and structure, different original images, and various noise levels. First, we found that in cases where the dimension of the transition subspace was large relative to the dimension of the problem, while both preconditioners could be successful in speeding convergence in the first few iterations (i.e., the cutoffs could be chosen to cluster the largest singular values), it was unlikely that either preconditioned scheme could, within fewer iterations, produce solutions whose relative errors were comparable to those generated by unpreconditioned CGLS. We attribute this phenomena to the fact that both preconditioners mixed too much noise into early iterates by clustering too many singular values without being able to reconstruct some important components of the solution lying in the transition space.

As the ratio of block bandwidth to block size was increased, we found the BTCB approximation to T did a much better job than the BCCB approximation to T of approximating the midrange and small singular values of the matrix. We also found this to be true when the matrix was blockwise unsymmetric. Consequently, our preconditioner can show significant improvement for these types of problems over the preconditioner in [17] when the cutoff, determined by the noise level, is small enough to include some midrange singular values, as evidenced in Example 2. For larger noise levels, there was no consistent or significant advantage to using one preconditioner over the other. We therefore particularly recommend our preconditioner when T is block unsymmetric, has a ratio of block bandwidth to block size larger than, say, 1/8, and in other such cases when we expect that the T. Chan BCCB matrix approximation to T will fail to approximate T well on the block level.

Example 1 shows that when the block bandwidth is small relative to block size, the matrix is symmetric, and the dimension of the upper subspace is small relative to N, the optimal preconditioner in [17] can produce solutions with slightly smaller relative error in somewhat fewer iterations than our optimal preconditioner. It is important to remember that both preconditioners were sensitive to the choice of cutoff, so finding the optimal preconditioner is difficult in practice. Also, the cost to initialize our preconditioner in Examples 1 and 2 is of the same order of magnitude as the initialization cost of the preconditioner in [17].

In short, our preconditioner never performs much worse than the preconditioner in [17] and can perform much better in some cases.

7. Conclusions.

We have developed an efficient algorithm for computing regularized solutions to discrete ill-posed problems involving BTTB matrices. Our algorithm uses a unitary transform to transform the BTTB matrix and its BTCB approximant to Cauchy-like matrices whose blocks are Cauchy-like. It then iterates using the CGLS algorithm on the left-preconditioned system, where the preconditioner was determined using size m_i partial factorizations with pivoting on each of the p blocks of the transformed BTCB matrix. By exploiting properties of the transformation, we showed each iteration of CGLS costs $O(np(\lg n + \lg p))$ operations for a Cauchy-like system with p blocks of size n.

The theory developed in section 4 predicts that for many types of BTTB matrices, the preconditioner determined in the course of Gu's fast, modified, complete pivoting algorithm can be expected to cluster the largest singular values around one and to keep the small singular values small while leaving the upper and lower subspaces unmixed. Thus, CGLS produces a good approximation to the noise-free solution within a small number of iterations. Our results indicate that the algorithm is both efficient and practical with the truncation parameters m_i chosen using our second heuristic. Finally, the results show that our preconditioned method is competitive with the preconditioned method of [17] in terms of both the number of iterations required to reach a reasonable regularized solution and the amount of work performed per iteration.

We note that this preconditioner can also be applied in situations where the matrix T is only block Toeplitz and the blocks are not necessarily Toeplitz. In this case the diagonals of block H_i of the BTCB matrix H will be the T. Chan approximation to block T_i given by the formula (see [5], for instance)

$$h_l^{(i)} = \frac{1}{p} \sum_{j-k=l(\mod n)} T_{jk}^{(i)}, \quad l = 0, \ldots, p-1.$$

Acknowledgments. I would like to thank Dianne O'Leary for her helpful comments and guidance during the preparation of this paper. I am also grateful to Jim Nagy and a referee for providing suggestions on numerical examples and to both referees for their insightful remarks.

REFERENCES

[1] R. Chan, J. Nagy, and R. Plemmons, *FFT-based preconditioners for Toeplitz-block least squares problems*, SIAM J. Numer. Anal., 30 (1993), pp. 1740–1768.

[2] R. Chan, J. Nagy, and R. Plemmons, *Circulant preconditioned Toeplitz least squares iterations*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 80–97.

[3] R. Chan and M. Yeung, *Circulant preconditioners for Toeplitz matrices with positive continuous generating functions*, Math. Comp., 58 (1992), pp. 233–240.

[4] R. H. Chan and X.-Q. Jin, *A family of block preconditioners for block systems*, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 1218–1235.

[5] R. H. Chan and M. K. Ng, *Conjugate gradient methods for Toeplitz systems*, SIAM Rev., 38 (1996), pp. 427–482.

[6] T. Chan, *An optimal circulant preconditioner for Toeplitz systems*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 766–771.

[7] T. F. Chan and P. C. Hansen, *A lookahead Levinson algorithm for general Toeplitz systems*, IEEE Proc. Signal Processing, 40 (1992), pp. 1079–1090. Cited by [14].

[8] H. E. Fleming, *Equivalence of regularization and truncated iteration in the solution of ill-posed problems*, Linear Algebra Appl., 130 (1990), pp. 133–150.

[9] I. Gohberg, T. Kailath, and V. Olshevsky, *Fast Gaussian elimination with partial pivoting of matrices with displacement structure*, Math. Comp., 64 (1995), pp. 1557–1576.

[10] I. Gohberg and V. Olshevsky, *Fast algorithm for matrix Nehari problem*, in Proceedings of MTNS-93, Systems and Networks: Mathematical Theory and Applications, Invited and Contributed Papers, R. M. U. Helmke and J. Saures, eds., Vol. 2, Academy Verlag, 1994, pp. 687–690.

[11] I. Gohberg and V. Olshevsky, *Fast state space algorithms for matrix Nehari and Nehari-Takagi interpolation problems*, Integral Equations Operator Theory, 20 (1994), pp. 44–83.

[12] G. Golub and V. Olshevsky, *Pivoting for Structured Matrices, with Applications*, http://www-isl.stanford.edu/olshevsk (1997).

[13] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, Addison-Wesley, Reading, MA, 1992.

[14] M. Gu, *Stable and Efficient Algorithms for Structured Systems of Linear Equations*, Tech. rep. LBL-37690, Lawrence Berkeley Laboratory, Berkeley, CA, 1995.

[15] M. H. Gutknecht and M. Hochbruck, *Look-ahead Levinson and Schur Algorithms for Non-Hermitian Toeplitz Systems*, Tech. rep. IPS 93-11, IPS Supercomputing, ETH-Zurich, Switzerland, 1993.

[16] M. Hanke and J. Nagy, *Restoration of atmospherically blurred images by symmetric indefinite conjugate gradient techniques*, Inverse Problems, 12 (1996), pp. 157–173.

[17] M. Hanke, J. Nagy, and R. Plemmons, *Preconditioned iterative regularization for ill-posed problems*, in Numerical Linear Algebra and Sci. Computing, L. Reichel, A. Ruttan, and R. S. Varga, eds., de Gruyter, Berlin, 1993, pp. 141–163.

[18] P. C. Hansen, *The discrete Picard condition for discrete ill-posed problems*, BIT, 30 (1990), pp. 658–672.

[19] P. C. Hansen, *Rank Deficient and Discrete Ill-Posed Problems*, Ph.D. thesis, Technical University of Denmark, Odense, Denmark, 1995, UNIC Report UNIC-95-07.

[20] P. C. Hansen and D. P. O'Leary, *The use of the L-curve in the regularization of discrete ill-posed problems*, SIAM J. Sci. Comput., 14 (1993), pp. 1487–1503.

[21] G. Heinig, *Inversion of generalized cauchy matrices and other classes of structured matrices*, in Linear Algebra in Signal Processing, IMA Vol. Math. Appl. 69, Springer-Verlag, New York, 1994, pp. 95–114.

[22] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Res. Natl. Bur. Standards, 49 (1952), pp. 409–436.

[23] R. A. Horn and C. R. Johnson, *Topics in Matrix Analysis*, Cambridge University Press, Cambridge, UK, 1991.

[24] A. K. Jain, *An operator factorization method for restoration of blurred images*, IEEE Trans. Comput., C-26 (1977), pp. 1061–1071.

[25] T. Kailath, S. Kung, and M. Morf, *Displacement ranks of matrices and linear equations*, J. Math. Anal. Appl., 78 (1979), pp. 395–407.

[26] T. Kailath and V. Olshevsky, *Displacement structure approach to discrete-trigonometric-transform based preconditioners of the G.Strang and T.Chan types*, Calcolo, 33 (1996), pp. 191–208.

[27] M. E. Kilmer, *Regularization of Ill-Posed Problems*, Ph.D. thesis, University of Maryland, College Park, MD, 1997.

[28] M. E. Kilmer and D. P. O'Leary, *Pivoted Cauchy-like preconditioners for regularized solution of ill-posed problems*, SIAM J. Sci. Comput., to appear.

[29] T. Ku and C.-C. J. Kuo, *On the spectrum of a family of preconditioned block Toeplitz matrices*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 948–966.

[30] G. W. Stewart and J. G. Sun, *Matrix Perturbation Theory*, Academic Press, Boston, MA, 1990.

[31] D. R. Sweet, *The use of pivoting to improve the numerical performance of Toeplitz matrix algorithms*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 468–493.

[32] A. van der Sluis and H. van der Vorst, *The rate of convergence of conjugate gradients*, Numer. Math., 48 (1986), pp. 543–560.

[33] J. M. Varah, *Pitfalls in the numerical solution of linear ill-posed problems*, SIAM J. Sci. Statist. Comput., 4 (1983), pp. 164–176.

SIAM J. Matrix Anal. Appl.
Vol. 20, No. 3, pp. 800–819

ON SMOOTH DECOMPOSITIONS OF MATRICES[*]

LUCA DIECI[†] AND TIMO EIROLA[‡]

Abstract. In this paper we consider smooth orthonormal decompositions of smooth time varying matrices. Among others, we consider QR-, Schur-, and singular value decompositions, and their block-analogues. Sufficient conditions for existence of such decompositions are given and differential equations for the factors are derived. Also generic smoothness of these factors is discussed.

Key words. smooth orthonormal factorizations, QR factorization, Schur-decomposition, singular value decomposition, polar decomposition

AMS subject classifications. 15A, 65F, 65L

PII. S0895479897330182

1. Introduction.

For very good reasons, orthogonal matrices (unitary in the complex case) are the backbone of modern matrix computation. They can be computed stably, and they provide some of the most successful algorithmic procedures for a number of familiar tasks: finding orthonormal bases, solving least squares problems, eigenvalues and singular values computations, for example. The purpose of this work is to consider orthogonal decompositions for matrices depending on a real parameter. Thus we consider k times continuously differentiable matrix functions, i.e., $A \in \mathcal{C}^k(\mathbb{R}, \mathbb{C}^{m\times n})$, $k \geq 0$.

We consider a number of basic tasks, such as the QR-, Schur-, and singular value decomposition (SVD) of A, and their block-analogues. Of course, in general, the matrix $A(t)$ will have, say, an SVD at each given t, but we are interested in conditions and procedures guaranteeing that the factors involved are smooth. This is desirable in several situations, such as in updating techniques, perturbation theory, or continuations processes, or to compute moving frames, e.g., to an invariant curve in space (see [4], [13], [9], [14]). A key motivation for us was provided by techniques for computing Lyapunov exponents (e.g., see [2] or [5]), in which case A is a solution of a linear system; in particular, it is full rank for all t. (This turns out to be a convenient sufficient condition for some of the factorizations considered below.)

Fundamental theoretical results on decompositions of parameter dependent matrices A are given in the book by Kato [12]. There, the strongest results are obtained in case A is real analytic and Hermitian; then, for example, it has an analytic Schur-decomposition. Similarly, Bunse–Gerstner et alia have shown that a real analytic A admits an analytic SVD [3]. A different technique is used by Gingold and Hsieh in [6] to show that a real analytic (not necessarily Hermitian) matrix with only real eigenvalues admits an analytic Schur-decomposition (triangularization). However, we do not want to require analyticity of A; many of the problems which motivated our work arise from linearization of a differential equation around a computed trajectory, and A may represent either the fundamental solution of the linearized problem or the

[*] Received by the editors November 14, 1997; accepted for publication (in revised form) by V. Mehrmann August 11, 1998; published electronically April 27, 1999. This work was supported in part under NSF grant DMS-9625813 and Academy of Finland grant 26341.

http://www.siam.org/journals/simax/20-3/33018.html

[†] School of Mathematics, Georgia Institute of Technology, Atlanta, GA 30332 (dieci@math.gatech.edu).

[‡] Department of Mathematics, Joensuu University, FIN-80101, Joensuu, Finland (teirola@cc.joensuu.fi).

Jacobian matrix of the vector field. Thus we are interested in the case of A being a $\mathcal{C}^k$-matrix. Unfortunately, smoothness of A does not suffice for the existence of smooth decompositions for it. For example, in the case of a Hermitian matrix, it is well understood that a smooth Schur-decomposition, in general, does not exist.

In this paper, we first give some (strong) sufficient conditions for smooth decompositions to exist. Differential equations for the factors will be given in such cases. Seemingly, these conditions appear very restrictive. For example, for the QR factorization of a matrix A, one would need full rank of A; for the Schur factorization of a symmetric matrix or the SVD of a full rank matrix, one would need simple eigenvalues or singular values. We take three approaches to weaken our assumptions. First, we consider block-analogues of the standard decompositions. Also in this case differential equations are given. Second, we take a closer look at the type of singular behavior which can occur when, for example, A loses rank, or eigenvalues coalesce; this provides weaker sufficient conditions to guarantee the existence of smooth decompositions, usually with some loss of smoothness. Third, we consider what can be expected in the generic case.

Genericity considerations are common in the study of dynamical systems. In our context, they are a way to rigorously classify which properties of matrices depending on a real parameter are typical. The starting point is to consider the space of one-parameter functions of matrices. One then endows this space with a suitable topology and calls a property of the topological space *generic* if it holds for a set which is of the second Baire category. We refer to [1] and to [11] for the necessary background on the topic. A generic property implies that we can perturb a given function not satisfying this property into one which does satisfy it. A generic statement will provide a convenient starting point for computational approaches. For example, we will show that, generically, Hermitian matrix functions of one variable have simple eigenvalues. Therefore, an appropriate starting point for computational procedures to find smooth eigendecompositions of Hermitian matrices is to assume simple eigenvalues: this gives the differential equations of section 2. Likewise, the genericity results of section 4 will legitimate the differential equations' models put forward in section 2 also for QR factorization and SVD decompositions. We believe that a merit of this paper is to provide a general framework for the relevant decompositions based on the differential equations models. However, it is premature to say whether or not numerical solutions of these differential or differential-algebraic equations (DAEs) will lead to efficient algorithms of solutions for the problems under study.

This paper is organized as follows. In section 2 we consider QR-, Schur-, and block Schur factorizations, and SVD and block SVD decompositions. In this section, differential equations are derived for the decompositions under some nondegenericity assumptions.

In section 3 we extend existence results for smooth decompositions to the case in which some singular behavior is encountered. We consider QR factorizations for rank deficient matrices and Schur/SVD decompositions in the case of coalescing eigen/singular values.

In section 4 we study how smooth the factors can be expected to be in generic cases.

2. Smooth decompositions. In this section we consider $A \in \mathcal{C}^k(\mathbb{R}, \mathbb{C}^{m\times n})$, $k \geq 1$, and we give differential equations for the decompositions of interest. The differential equations for the QR factorization of a full rank matrix have been given before (e.g., see [5]); the differential equations for the SVD have already appeared elsewhere. (The earliest references of which we are aware are [8] and [16], with the former reference only for the case of a fundamental solution matrix.)

Remark 2.1. In 1965, Sibuya [15, Theorem 3 and Remark 3] established that a $\mathcal{C}^k$ ($k \geq 0$) matrix with disjoint groups of eigenvalues admits a $\mathcal{C}^k$ block-diagonalization; this result is the content of Remark 2.6 below. It seems possible to build on this result of Sibuya to establish some of the other results we give in section 2. However, the approach in [15] is not constructive and it seems hard to exploit it for devicing computational procedures. In contrast, the differential equations models are better for obtaining computational procedures (but, of course, one needs $k \geq 1$).

We will use the following simple results.

LEMMA 2.1. *Assume that in the linear system of equations $B(t)x(t) = b(t)$, $B(t)$ is invertible for all t and that $B, b \in \mathcal{C}^k$. Then, also $x \in \mathcal{C}^k$.*

Proof. This follows from repeated differentiation of $x(t) = B^{-1}(t)b(t)$, recalling that $\frac{d}{dt}B^{-1} = -B^{-1}\dot{B}B^{-1}$. □

LEMMA 2.2. *Let $Q_0 \in \mathbb{C}^{m\times n}$ be orthonormal.*[1] *Consider the differential equation*

$$\dot{Q} = S(Q,t)\ Q, \qquad Q(0) = Q_0, \tag{2.1}$$

where S is an $m \times m$-matrix function, locally Lipschitzian in Q and continuous in t, such that

$$Q^*\big[S(Q,t) + S(Q,t)^*\big]Q \ = \ 0$$

for every t and every orthonormal Q. Then (2.1) *has a unique solution Q and $Q(t)$ is orthonormal for all t.*

Proof. Existence and uniqueness are standard results. (Note that the set of orthonormal matrices is compact.) If Q is the solution of (2.1), then $Q^*\dot{Q} + \dot{Q}^*Q = Q^*(S + S^*)Q = 0$, i.e., Q^*Q is constant. □

Remark 2.2. Quite often, we will use Lemma 2.2 in the case $m = n$ with Q satisfying $\dot{Q} = Q\ H(Q,t)$, where H is a skew-Hermitian $n \times n$ matrix function (i.e., $H^*(Q,t) = -H(Q,t)$).

2.1. *QR* factorization. We begin by giving differential equations for the QR factorization of a full rank matrix $A \in \mathcal{C}^k(\mathbb{R}, \mathbb{C}^{m\times n})$, $m \geq n$. Suppose that $A(0) = Q(0)R(0)$ is a given QR factorization (i.e., $Q(0) \in \mathbb{C}^{m\times n}$ is orthonormal, and $R(0) \in \mathbb{C}^{n\times n}$ is upper triangular). We want to find, if possible, orthonormal Q and upper triangular R, both in $\mathcal{C}^k$, such that $A(t) = Q(t)R(t)$. Therefore, if feasible, differentiating $A = QR$ and $Q^*Q = I$ we get

$$\dot{A} = \dot{Q}R + Q\dot{R} \quad \text{and} \quad \dot{Q}^*Q + Q^*\dot{Q} = 0. \tag{2.2}$$

Hence $H := Q^*\dot{Q}$ is skew-Hermitian and from the first of (2.2) we get

$$\begin{aligned} \dot{R} &= Q^*\dot{A} - Q^*\dot{Q}R = Q^*\dot{A} - HR, \\ \dot{Q} &= \dot{A}R^{-1} - Q\dot{R}R^{-1} = (I - QQ^*)\dot{A}R^{-1} + QH. \end{aligned} \tag{2.3}$$

Since $\dot{R}$ has to be upper triangular, we require

$$\begin{aligned} (Q^*\dot{A})_{i,1} &= H_{i,1}R_{1,1}, & i \geq 2, \\ (Q^*\dot{A})_{i,2} &= H_{i,1}R_{1,2} + H_{i,2}R_{2,2}, & i \geq 3, \\ &\ \vdots \\ (Q^*\dot{A})_{n,n-1} &= H_{n,1}R_{1,n-1} + H_{n,2}R_{2,n-1} + \cdots + H_{n,n-1}R_{n-1,n-1}. \end{aligned} \tag{2.4}$$

[1] We call $Q \in \mathbb{C}^{m\times n}$ *orthonormal*, if $Q^*Q = I$, and *unitary*, if further $m = n$.

These are clearly solvable for $H_{i,j}$, $i > j$, assuming that $R_{1,1}, \ldots, R_{n-1,n-1}$ are nonzero. Then the skew-Hermitian property defines the part above the diagonal. Diagonal entries of H we set to purely imaginary values in such a way that the diagonal of $Q^*\dot{A} - HR$ becomes real.[2] This defines H as a smooth function of Q and R. Rewriting the differential equation for Q as

$$\dot{Q} = [(I - QQ^*)\dot{A}R^{-1}Q^* + QHQ^*]Q := S(Q,t)Q,$$

since $Q^*(S + S^*)Q = 0$, (2.3) and Lemmas 2.1 and 2.2 give the desired result as long as R stays invertible. On the other hand, if matrices Q and R satisfy the differential equations (2.3) and the initial condition $Q(0)R(0) = A(0)$, then they provide a QR factorization of A. Thus we get the following proposition.

PROPOSITION 2.3. *Any full column rank $\mathcal{C}^k$-matrix has a $\mathcal{C}^k$ QR-decomposition.*

Remark 2.3. The case in which A is a solution of a linear system, $\dot{A}(t) = B(t)A(t)$, leads to some simplifications in formulas (2.3). In fact, $\dot{A} = BQR$ gives

$$\begin{aligned} \dot{R} &= (Q^*BQ - H)R, \\ \dot{Q} &= (I - QQ^*)BQ + QH. \end{aligned} \tag{2.5}$$

2.2. Schur-decompositions. Next, we consider differential equations for Schur-decomposition. We begin with the case of a Hermitian matrix.

(a) Diagonalization of a Hermitian matrix. Denote by $\mathbb{C}_H^{n\times n}$ the set of Hermitian $n \times n$ matrices. Let $A \in \mathcal{C}^k(\mathbb{R}, \mathbb{C}_H^{n\times n})$. Suppose that a Schur-decomposition $A(0) = Q(0)D(0)Q(0)^*$ is given, i.e., $Q(0)$ is unitary and $D(0)$ is diagonal. We want to find, if possible, $Q, D \in \mathcal{C}^k$ unitary and diagonal, respectively, such that $Q^*(t)A(t)Q(t) = D(t)$ for all t. Here we write $D = \operatorname{diag}(d_1, \ldots, d_n)$. If feasible, we must have

$$\dot{D} = Q^*\dot{A}Q + DQ^*\dot{Q} + \dot{Q}^*QD,$$

or by letting $H = Q^*\dot{Q}$ and noticing that H is skew-Hermitian, $H^* = -H$, we get the following system for D and Q:

$$\begin{aligned} \dot{D} &= Q^*\dot{A}Q + DH - HD, \\ \dot{Q} &= QH. \end{aligned} \tag{2.6}$$

From this we then obtain $\dot{d}_i = (Q^*\dot{A}Q)_{ii}$. Notice that (2.6) is really a system of DAEs, and we wish to use the algebraic equations (the part relative to the off diagonal entries in $\dot{D}$) to determine H. Because of skewness, H_{ii}'s have to be purely imaginary, but otherwise arbitrary, e.g., zero. If the eigenvalues of A are distinct, then we get

$$H_{ij} = \frac{(Q^*\dot{A}Q)_{ij}}{d_j - d_i},\ i \neq j. \tag{2.7}$$

Thus from (2.6) and (2.7) we get the smooth Schur factorization of A if all eigenvalues of $A(t)$ are distinct for all t.

PROPOSITION 2.4. *Any Hermitian $\mathcal{C}^k$-matrix with simple eigenvalues is diagonalizable with a unitary $\mathcal{C}^k$-matrix .*

[2] With this choice of the diagonal of H we get the diagonal elements of R real if they are such for $R(0)$.

Under the previous assumptions, an obvious consequence of the Schur-decomposition of a positive definite matrix is that we can very simply obtain smooth square roots of the matrix, once the decomposition $A = QDQ^*$ is at hand. It suffices to consider square roots of the eigenvalues d_i.

Remark 2.4. A construction similar to the one above can be done for the eigen-decomposition of a general matrix $A \in \mathcal{C}^k(\mathbb{R}, \mathbb{C}^{n\times n})$ with distinct eigenvalues. Now we seek (given appropriate initial conditions $V(0)$ and $D(0)$) a smooth decomposition $A(t) = V(t)D(t)V(t)^{-1}$. By letting $P = V^{-1}\dot{V}$, we have

$$\begin{aligned} &\dot{d}_i = (V^{-1}\dot{A}V)_{ii}, && i = 1, \dots, n, \\ &\dot{V} = VP, \quad P_{ij} = \frac{(V^{-1}\dot{A}V)_{ij}}{d_j - d_i}, && i \neq j. \end{aligned} \tag{2.8}$$

The diagonal entries P_{ii} are not uniquely determined, and we may set them to 0. Another choice is to require the columns of V to have constant norms, which means $\mathrm{Re}(v_i^* \dot{v}_i) = 0$ giving the condition

$$\mathrm{Re}(v_i^* v_i P_{ii}) = -\sum_{j\neq i} \mathrm{Re}(v_i^* v_j \, P_{ji}).$$

Again, the factors V and D are as smooth as A.

(b) Schur-decomposition of a general matrix. Let $A \in \mathcal{C}^k(\mathbb{R}, \mathbb{C}^{n\times n})$. For given $U(0)$ and $R(0)$, respectively, unitary and upper triangular, such that $R(0) = U^*(0)A(0)U(0)$, we seek unitary U and triangular R as smooth as A, such that $R(t) = U^*(t)A(t)U(t)$ for all t. If feasible, we then must have

$$\dot{R} = U^*\dot{A}U + \dot{U}^*AU + U^*A\dot{U} = U^*\dot{A}U + (\dot{U}^*U)U^*AU + U^*AU(U^*\dot{U}).$$

Again we set $H := U^*\dot{U}$ and obtain the system

$$\begin{aligned} \dot{R} &= U^*\dot{A}U + RH - HR, \\ \dot{U} &= UH. \end{aligned} \tag{2.9}$$

Conditions that R is upper triangular and $H^* = -H$ bring this to a DAE system. We use the strictly lower triangular part in the first equation of (2.9) to determine H (except for its diagonal). This can be done by first finding the vector $(H_{21}, \dots, H_{n1})^*$, then $(H_{32}, \dots, H_{n2})^*$, etc., up to $H_{n,n-1}$. To find these vectors one needs to solve triangular systems which are easily seen to be nonsingular if $R_{ii}(t) \neq R_{jj}(t)$, $i \neq j$. Then, by solving (2.9), we get R and U as smooth as A. The entries $H_{ii}(t)$ are undetermined; we may set them to 0 (in any case, they must be purely imaginary).

Together with Remark 2.4 we get the following.

PROPOSITION 2.5. *Any $\mathcal{C}^k$-matrix with simple eigenvalues has a $\mathcal{C}^k$–Schur-decomposition and is diagonalizable with a $\mathcal{C}^k$-matrix.*

Clearly, the differential equation model above breaks down if some of the eigenvalues of $A(t)$ coalesce. Besides, also for simple eigenvalues, numerical difficulties can be expected in case two or more eigenvalues become close. In such cases, it may be better to compute the invariant subspaces relative to a cluster of eigenvalues.

(c) Block-Schur-decomposition. Consider $A \in \mathcal{C}^k(\mathbb{R}, \mathbb{C}^{n\times n})$. We want to find unitary Q and block upper triangular S, as smooth as A, such that $Q^*(t)A(t)Q(t) = S(t)$, where S is partitioned as

$$S = \begin{bmatrix} S_{11} & S_{12} \\ 0 & S_{22} \end{bmatrix}.$$

We start with a given decomposition $A(0) = Q(0)S(0)Q(0)^*$. Assume that $\lambda_1(t), \dots, \lambda_n(t) \in \Lambda(A(t))$ are continuous, such that $\Lambda_1(t) = \{\lambda_1(t), \dots, \lambda_m(t)\}$ and $\Lambda_2(t) = \{\lambda_{m+1}(t), \dots, \lambda_n(t)\}$ are disjoint for all t and $\Lambda(S_{jj}(0)) = \Lambda_j(0),\ j = 1, 2$.

Differentiating the relation $S = Q^*AQ$ and letting $H := Q^*\dot{Q}$ we obtain the system of DAEs

$$\begin{aligned} \dot{S} &= Q^*\dot{A}Q + SH - HS, \\ \dot{Q} &= QH, \\ S_{21} = 0&, \quad H^* = -H. \end{aligned} \tag{2.10}$$

This can be reduced to a system of differential equations (DEs) by eliminating the algebraic part. By rewriting the first equation of (2.10) in block form

$$\begin{aligned} \begin{bmatrix} \dot{S}_{11} & \dot{S}_{12} \\ 0 & \dot{S}_{22} \end{bmatrix} &= \begin{bmatrix} (Q^*\dot{A}Q)_{11} & (Q^*\dot{A}Q)_{12} \\ (Q^*\dot{A}Q)_{21} & (Q^*\dot{A}Q)_{22} \end{bmatrix} \\ &\quad + \begin{bmatrix} S_{11} & S_{12} \\ 0 & S_{22} \end{bmatrix} \begin{bmatrix} H_{11} & H_{12} \\ -H_{12}^* & H_{22} \end{bmatrix} - \begin{bmatrix} H_{11} & H_{12} \\ -H_{12}^* & H_{22} \end{bmatrix} \begin{bmatrix} S_{11} & S_{12} \\ 0 & S_{22} \end{bmatrix}, \end{aligned}$$

we realize that we must have

$$S_{22}H_{12}^* - H_{12}^*S_{11} = (Q^*\dot{A}Q)_{21}. \tag{2.11}$$

Thus, $H_{12} \in \mathbb{C}^{m\times(n-m)}$ is the unique solution of (2.11), since S_{11} and S_{22} have no common eigenvalue (e.g., see [7]). The blocks H_{11} and H_{22} are not uniquely determined, and we may set them both to 0 (in any case, they must be skew-Hermitian). Thus, taking $H = \left[\begin{smallmatrix} 0 & -H_{12}^* \\ -H_{12} & 0 \end{smallmatrix}\right]$ and the differential equations (2.10) for $Q, S_{11}, S_{12}, S_{22}$, we obtain the desired result.

The above procedure immediately generalizes to a block-Schur factorization in p blocks. That is, one seeks the factorization where S is partitioned as

$$S = \begin{bmatrix} S_{11} & S_{12} & \dots & S_{1p} \\ 0 & S_{22} & \dots & S_{2p} \\ & & \ddots & \vdots \\ 0 & 0 & & S_{pp} \end{bmatrix} \tag{2.12}$$

and each diagonal block is a square matrix. With obvious notation, block-partitioning $H = Q^*\dot{Q}$ conformally to S, and assuming that $\Lambda(S_{ii}(t)) \cap \Lambda(S_{jj}(t)) = \emptyset$ for all t and $i \neq j$, we obtain H by solving the system for the H_{ji}:

$$\begin{aligned} \sum_{k=i}^{p} S_{ik}H_{jk}^* - \sum_{k=1}^{j} H_{ki}^*S_{kj} = (Q^*\dot{A}Q)_{ij}, \\ j = 1, \dots, p-1, \quad i = j+1, \dots, p. \end{aligned} \tag{2.13}$$

The blocks H_{ii} are not uniquely determined and we may set them to 0. In summary, we have the following proposition.

PROPOSITION 2.6. *Any $\mathcal{C}^k$-matrix with disjoint groups of eigenvalues has a $\mathcal{C}^k$-block Schur-decomposition, the blocks corresponding to these groups.*

Remark 2.5. From a practical point of view, the strongest assumption we made is that the initial decomposition determines a correct blocking for all t.

In case A is Hermitian, the above block-Schur-decomposition gives a block diagonal form. Then (2.13) becomes $S_{ii}H_{ji}^* - H_{ji}^*S_{jj} = (Q^*\dot{A}Q)_{ij},\ i \neq j$. Here, since the initial conditions for S_{ii} are Hermitian, we will have $S_{ii}(t)$ Hermitian for all t, for every skew-Hermitian H_{ii}, because $\dot{S}_{ii} = (Q^*\dot{A}Q)_{ii} + S_{ii}H_{ii} - H_{ii}S_{ii}$.

Remark 2.6. Of course, a block diagonalization of general matrices can also be envisioned similarly to our previous remark, Remark 2.4 (see also [15]). That is, if feasible, we now seek block-diagonal

$$D = \begin{bmatrix} D_1 & & \\ & \ddots & \\ & & D_p \end{bmatrix}$$

and invertible V, both as smooth as A, such that $A(t) = V(t)D(t)V^{-1}(t)$. Differentiating this, letting $P = V^{-1}\dot{V}$, and partitioning P conformally to D, we get

$$\begin{aligned} \dot{D}_i &= (V^{-1}\dot{A}V)_{ii} + D_iP_{ii} - P_{ii}D_i, \quad i = 1, \dots, p, \\ 0 &= (V^{-1}\dot{A}V)_{ij} + D_iP_{ij} - P_{ij}D_j, \quad i \neq j, \\ \dot{V} &= VP\ . \end{aligned} \tag{2.14}$$

If $\Lambda(D_i(t)) \cap \Lambda(D_j(t)) = \emptyset,\ i \neq j$, the second of these equations can be solved for the P_{ij}. P_{ii} are not uniquely determined, and we may set them to 0, or require $V_i^*V_i$ to be constant, where $[V_1 \dots V_p]$ is the partitioning of V in column blocks. This amounts to

$$P_{ii}V_i^*V_i + V_i^*V_iP_{ii} = -\sum_{i \neq j}(P_{ji}^*V_j^*V_i + V_i^*V_jP_{ji}),$$

which is uniquely solvable for P_{ii}, since V_i has full rank.

2.3. Singular value decompositions. Next we consider the SVD of a matrix. Again, we also consider block analogs.

(a) Smooth SVD. This has been recently considered in [3] and [16] in the analytic case. We have a matrix $A \in \mathcal{C}^k(\mathbb{R}, \mathbb{C}^{m \times n})$, $m \geq n$, and look for $\mathcal{C}^k$ unitary U and V, and real "diagonal" Σ such that $A(t) = U(t)\Sigma(t)V^*(t)$. Here $\Sigma = \left[\begin{smallmatrix} S \\ 0 \end{smallmatrix}\right]$, $S = \operatorname{diag}(\sigma_1, \dots, \sigma_n)$. We assume that $0 \neq \sigma_i(t) \neq \sigma_j(t), i \neq j$ for all t. Let an initial SVD, $A(0) = U(0)\Sigma(0)V^*(0)$, be given. Differential equations for the factors are derived as follows. From $A = U\Sigma V^*$, we get

$$U^*\dot{A}V = (U^*\dot{U})\Sigma + \dot{\Sigma} + \Sigma(\dot{V}^*V).$$

Let us set

$$H := U^*\dot{U} \in \mathbb{C}^{m \times m}, \quad K := V^*\dot{V} \in \mathbb{C}^{n \times n},$$

both of which are skew-Hermitian. Therefore, we obtain the system of DAEs

$$\begin{aligned} \dot{\Sigma} &= U^*\dot{A}V - H\Sigma + \Sigma K, \\ \dot{U} &= UH, \\ \dot{V} &= VK \end{aligned} \tag{2.15}$$

with the requirement that Σ is real and diagonal. We proceed to eliminate the algebraic part, thereby reducing the above to a system of DEs. First we notice that the equations for the singular values are all decoupled as follows:

$$\dot{\sigma}_i = (U^*\dot{A}V)_{ii} - H_{ii}\sigma_i + \sigma_iK_{ii}, \quad i = 1, \dots, n.$$

Next, consider the algebraic part of the first equation in (2.15). For $i \neq j$, $i, j = 1, \ldots, n$, we must have

$$0 = (U^* \dot{A} V)_{ij} - H_{ij}\sigma_j + \sigma_i K_{ij},$$
$$0 = (U^* \dot{A} V)_{ji} - H_{ji}\sigma_i + \sigma_j K_{ji},$$

from which, using skewness of H and K, we get

$$H_{ij} = \frac{\sigma_j (U^* \dot{A} V)_{ij} + \sigma_i (U^* \dot{A} V)_{ji}}{\sigma_j^2 - \sigma_i^2},$$
$$K_{ij} = \frac{\sigma_j (U^* \dot{A} V)_{ji} + \sigma_i (U^* \dot{A} V)_{ij}}{\sigma_j^2 - \sigma_i^2},$$

where $i, j = 1, \ldots, n,\ i \neq j$. Also, it is easy to obtain

$$H_{ij} = -\bar{H}_{ji} = \frac{(U^* \dot{A} V)_{ij}}{\sigma_j}, \quad i = n+1, \ldots, m,\ j = 1, \ldots, n. \tag{2.16}$$

For $\dot{\sigma}_i$ to be real, the diagonal elements of H and K need to satisfy

$$(H_{ii} - K_{ii}) = \operatorname{Im}((U^* \dot{A} V)_{ii})/\sigma_i.$$

We can choose, e.g., $K_{ii} = 0$, which then determines H_{ii}.

Finally, the bottom right $(m-n) \times (m-n)$ block of H is not determined; one may thus set it to 0 (or any other skew matrix). These give the desired result in the following proposition.

PROPOSITION 2.7. *Any full column rank $\mathcal{C}^k$-matrix with distinct singular values has a $\mathcal{C}^k$ singular value decomposition.*

In case A is the solution of a linear system: $\dot{A}(t) = B(t)A(t)$ with $A(0)$ full rank, the above formulas simplify by $\dot{A} = BU\Sigma V^*$. The details are easy and one obtains

$$\dot{\sigma}_i = (U^* B U)_{ii}\sigma_i - H_{ii}\sigma_i + \sigma_i K_{ii}, \qquad i = 1, \ldots, n,$$

$$H_{ij} = \frac{\sigma_j^2 (U^* B U)_{ij} + \sigma_i^2 (U^* B U)_{ji}}{\sigma_j^2 - \sigma_i^2}, \qquad i, j = 1, \ldots, n,\ i \neq j,$$

$$H_{ij} = (U^* B U)_{ij}, \qquad i = n+1, \ldots, m,\ j = 1, \ldots, n,$$

$$K_{ij} = \sigma_i \sigma_j \frac{(U^* B U)_{ji} + (U^* B U)_{ij}}{\sigma_j^2 - \sigma_i^2}, \qquad i, j = 1, \ldots, n,\ i \neq j.$$

It should be noticed that the matrix V does not need to be computed if only information on the singular values is desired.

Remark 2.7. It should be clear that the U and V factors of the SVD of A are nothing but unitary Schur factors of the positive semidefinite matrices AA^* and A^*A, respectively. As usual, we can think of the square part of the Σ matrix as a square root of the block diagonal matrix $V^*(A^*A)V$.

(b) Block-SVD decomposition. Consider next a full rank matrix $A \in \mathcal{C}^k(\mathbb{R}, \mathbb{C}^{m \times n})$. We want to find unitary U, V, and "block-diagonal" Σ, as smooth as A, such that $U^*(t)A(t)V(t) = \Sigma(t)$. Here $\Sigma(t)$ is partitioned as

$$\Sigma = \begin{bmatrix} S_1 & 0 \\ 0 & S_2 \\ 0 & 0 \end{bmatrix},$$

where S_1, S_2 are Hermitian positive definite matrices. We assume that the (positive) singular values of A can be arranged into two disjoint groups, like the eigenvalues in the block Schur-decomposition above, and that initial conditions $U(0), \Sigma(0), V(0)$ are given. We proceed as usual, by differentiating the relation $U^*AV = \Sigma$. Denoting the skew-Hermitian matrices $H = U^*\dot{U}$ and $K = V^*\dot{V}$ we end up with the following equations

$$\begin{bmatrix} \dot{S}_1 & 0 \\ 0 & \dot{S}_2 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} (U^*\dot{A}V)_{11} & (U^*\dot{A}V)_{12} \\ (U^*\dot{A}V)_{21} & (U^*\dot{A}V)_{22} \\ (U^*\dot{A}V)_{31} & (U^*\dot{A}V)_{32} \end{bmatrix} + \begin{bmatrix} S_1 & 0 \\ 0 & S_2 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} K_{11} & K_{12} \\ -K_{12}^* & K_{22} \end{bmatrix}$$
$$- \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ -H_{12}^* & H_{22} & H_{23} \\ -H_{13}^* & -H_{23}^* & H_{33} \end{bmatrix} \begin{bmatrix} S_1 & 0 \\ 0 & S_2 \\ 0 & 0 \end{bmatrix}.$$

From the algebraic part in these equations we obtain

$$\begin{aligned} -H_{13}^*S_1 &= (U^*\dot{A}V)_{31}, \\ -H_{23}^*S_2 &= (U^*\dot{A}V)_{32}, \end{aligned}$$

and

$$\begin{bmatrix} H_{12} \\ K_{12} \end{bmatrix} S_2 - \begin{bmatrix} 0 & S_1 \\ S_1 & 0 \end{bmatrix} \begin{bmatrix} H_{12} \\ K_{12} \end{bmatrix} = \begin{bmatrix} (U^*\dot{A}V)_{12} \\ ((U^*\dot{A}V)_{21})^* \end{bmatrix},$$

and thus we can uniquely determine H_{13} and H_{23} if $0 \notin \Lambda(S_1)$, $0 \notin \Lambda(S_2)$, and H_{12}, K_{12} if $\Lambda(S_1) \cap \Lambda(S_2) = \emptyset$. Note that in this full rank case we can consider the "standard" block SVD with S_1 and S_2 positive definite, since singular values do not become 0. H_{33} is an arbitrary skew-Hermitian matrix, and we may set it to 0. For blocks $H_{ii}, K_{ii}, i = 1, 2$, we can reason as follows. The differential equations for S_1 and S_2 are

$$\dot{S}_i = (U^*\dot{A}V)_{ii} - H_{ii}S_i + S_iK_{ii}, \;\; i = 1, 2,$$

and we want S_1 and S_2 to be Hermitian. Thus, we must have

$$S_i(H_{ii} - K_{ii}) + (H_{ii} - K_{ii})S_i = (U^*\dot{A}V)_{ii} - (U^*\dot{A}V)_{ii}^*, \;\; i = 1, 2,$$

and hence $H_{ii} - K_{ii}, i = 1, 2$ is uniquely determined and skew-Hermitian if $0 \notin \Lambda(S_i)$ (for example, can take $K_{11} = K_{22} = 0$). In summary, we obtain the desired smoothness for the block SVD, as long as A is full rank and the blocks S_1 and S_2 have disjoint spectra. Hence the following proposition.

PROPOSITION 2.8. *Any full column rank $\mathcal{C}^k$-matrix with disjoint groups of singular values has a $\mathcal{C}^k$-block singular value decomposition.*

If a square A satisfies a linear system $\dot{A} = BA$, the above relations simplify to

$$\begin{aligned} (U^*BU)_{12}S_2 &= H_{12}S_2 - S_1K_{12}, \\ S_1(U^*BU)_{21}^* &= K_{12}S_2 - S_1H_{12}; \end{aligned} \tag{2.17}$$

and for $i = 1, 2$,

$$\begin{aligned} \dot{S}_i &= (U^*BU)_{ii}S_i - H_{ii}S_i + S_iK_{ii} \\ S_i(H_{ii} - K_{ii}) + (H_{ii} - K_{ii})S_i &= (U^*BU)_{ii}S_i - S_i(U^*BU)_{ii}^*. \end{aligned} \tag{2.18}$$

The V factor need not be computed if only the blocks S_1 and S_2 are desired.

(c) Polar factorization. With similar notation as above, we now consider differential equations for the polar factorization of a full rank $A \in \mathcal{C}^k(\mathbb{R}, \mathbb{C}^{m\times n})$. That is, we want to write $A(t) = Q(t)P(t)$, where Q is orthonormal, P is Hermitian positive definite, and Q, P are as smooth as A. Differentiating $A = QP$, using orthonormality of Q, and letting as usual $H = Q^*\dot{Q}$, we obtain

$$\begin{aligned} \dot{Q} &= QH, \\ \dot{P} &= Q^*\dot{A} - HP. \end{aligned} \tag{2.19}$$

Since we need $P = P^*$, we then must have

$$PH + HP = Q^*\dot{A} - \dot{A}^*Q. \tag{2.20}$$

Since P is positive definite, (2.20) has a unique solution, H. Thus, we obtain the desired result since H satisfying (2.20) is skew-Hermitian.

Remark 2.8. Using (2.19) and (2.20) to obtain a smooth polar factorization of A, in contrast to passing through the smooth SVD, is not hampered by the need of noncoalescing singular values.

3. Extensions.

3. Extensions. In this section, we extend the smooth factorizations to the case in which some singular behavior is encountered, such as rank deficiency, or eigenvalues coalescing.

3.1. Rank deficient *QR*. We begin with the case of the QR-factorization of a matrix $A \in \mathcal{C}^k(\mathbb{R}, \mathbb{C}^{m\times n})$. As we saw in section 2, if A has full rank, then it admits a QR-factorization where the Q and R factors are also $\mathcal{C}^k$. Next, we show that, under appropriate assumptions, A admits a QR factorization also in case it is rank deficient; however, some loss of differentiability may take place. Then we show that analytic (denoting $\mathcal{C}^\omega$) matrices have analytic QR factorizations.

Our proof is based on a careful analysis of the Gram–Schmidt orthogonalization process. The idea behind the proof is that in order for the function A to have a QR factorization (with some degree of smoothness), then, at points where loss of rank occurs, the matrix obtained by differentiating the columns of A (possibly, derivatives of different orders for different columns) must have full rank.

For the $\mathcal{C}^k$ case, the following example is helpful in understanding what we should expect.

Example 3.1. Let $A = \begin{bmatrix} t^k|t| \\ t^d \end{bmatrix}$, $d, k \in \mathbb{N}$. Then A is k times continuously differentiable. Depending whether $d \le k$ or $d = k+1$ we have the QR-decompositions

$$A(t) = \begin{bmatrix} t^{k-d}\,|t|\,/\sqrt{1+t^{2(k-d+1)}} \\ 1/\sqrt{1+t^{2(k-d+1)}} \end{bmatrix} t^d\sqrt{1+t^{2(k-d+1)}}, \quad A(t) = \begin{bmatrix} \operatorname{sgn}(t)/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix} t^d\sqrt{2},$$

respectively. In the first case Q is in $\mathcal{C}^{k-d}$, while in the latter it is not (and cannot be taken) continuous. Note that in the first case $A(t)^TA(t) = t^{2d}\,(t^{2(k-d+1)}+1)$; in particular, $\lim_{t\to 0}\frac{1}{t^{2d}}A(t)^TA(t) = 1 > 0$.

THEOREM 3.1. *Let $A \in \mathcal{C}^k(\mathbb{R}, \mathbb{C}^{m\times n})$, $m \ge n$, and assume $d \le k$ is such that*[3]

$$\limsup_{\tau\to 0} \frac{1}{\tau^{2d}} \det(A^*A)(t+\tau) > 0 \tag{3.1}$$

for every t. Given any QR-factorization $A(t_0) = Q(t_0)R(t_0)$ at a point t_0 where A has full rank, there exists a $\mathcal{C}^{k-d}$ QR-factorization of A satisfying this initial condition.

[3] This means that the limsup is either positive or $+\infty$.

If $R(t_0)$ has real diagonal, this QR-factorization becomes unique, if we require the diagonal of R to be real.

Proof. Write $A(t) = [a_1(t) \dots a_n(t)]$. If A has full rank at t, then $Q = [q_1 \cdots q_n]$ necessarily satisfies

$$\begin{aligned} q_1(t) &= \frac{\eta_1(t)}{|a_1(t)|}\, a_1(t), & P_1 &= I - q_1 q_1^*, \\ q_2(t) &= \frac{\eta_2(t)}{|P_1(t)a_2(t)|}\, P_1(t)a_2(t), & P_2 &= P_1 - q_2 q_2^*, \\ &\ \vdots \\ q_j(t) &= \frac{\eta_j(t)}{|P_{j-1}(t)a_j(t)|}\, P_{j-1}(t)a_j(t), & P_j &= P_{j-1} - q_j q_j^*. \end{aligned}$$

Here η_j's have to satisfy $|\eta_j(t)| = 1$ for all j; otherwise they can be chosen freely. Taking them smooth on intervals of full rank gives us Q and $R = Q^*A$ both $\mathcal{C}^k$ on these intervals. Furthermore, real η_j is equivalent to real $R_{j,j}$ there.

Our assumption implies that the points where A does not have full rank are isolated.

The details of the proof for the case in which there is some possible loss of smoothness are based on the following claim.

Claim. If $\hat{t}$ and $j \in \{0, 1, \dots, k-1\}$ are such that $\limsup_{\tau\to 0} \frac{\det(A^*A)(\hat{t}+\tau)}{\tau^{2j}} = 0$, then $\lim_{\tau\to 0} \frac{\det(A^*A)(\hat{t}+\tau)}{\tau^{2(j+1)}}$ exists and is finite.

Proof of the claim. Case $j = 0$: we can assume $\hat{t} = 0$, so that $\det(A(0)^*A(0)) = 0$. Applying a permutation to the columns of A, we may also assume that $A(0) = [\, a_1^0 \quad A_{2\dots n}^0 \,]$, where $a_1^0 = A_{2\dots n}^0 \beta$ for some $\beta \in \mathbb{C}^{n-1}$. Then

$$A(t) = [\, A_{2\dots n}^0\beta + t\, a_1^\Delta(t) \quad A_{2\dots n}^0 + t\, A_{2\dots n}^\Delta(t) \,],$$

where $[\, a_1^\Delta(t) \quad A_{2\dots n}^\Delta(t) \,] = \frac{1}{t}\, (A(t) - A(0))$. Then

$$\begin{aligned} \det(A(t)^*A(t)) &= \det\left(\begin{bmatrix} 1 & -\beta^* \\ 0 & I \end{bmatrix} \begin{bmatrix} \beta^* {A_{2\dots n}^0}^* + t\, a_1^\Delta(t)^* \\ {A_{2\dots n}^0}^* + t\, A_{2\dots n}^\Delta(t)^* \end{bmatrix} \right. \\ &\qquad \left. [\, A_{2\dots n}^0\beta + t\, a_1^\Delta(t) \quad A_{2\dots n}^0 + t\, A_{2\dots n}^\Delta(t) \,] \begin{bmatrix} 1 & 0 \\ -\beta & I \end{bmatrix} \right) \\ &= \det\left(\begin{bmatrix} t\, (a_1^\Delta(t)^* - \beta^* A_{2\dots n}^\Delta(t)^*) \\ {A_{2\dots n}^0}^* + t\, A_{2\dots n}^\Delta(t)^* \end{bmatrix} [\, t\, (a_1^\Delta(t) - A_{2\dots n}^\Delta(t)\beta) \quad A_{2\dots n}^0 + t\, A_{2\dots n}^\Delta(t) \,] \right) \\ &= t^2 \det(\tilde{A}(t)^*\tilde{A}(t)), \end{aligned}$$

where $\tilde{A}(t) = [\, a_1^\Delta(t) - A_{2\dots n}^\Delta(t)\beta \quad A_{2\dots n}(t) \,]$. Hence the claim for $j = 0$ follows. Since[4] $\tilde{A} \in \mathcal{C}^{k-1}$ the proof for general j is completed by obvious induction.

Let $\hat{t}$ be a point where A does not have full rank, and let $\hat{d}$ be the smallest integer for which

$$\limsup_{\tau\to 0} \frac{1}{\tau^{2\hat{d}}} \det(A^*A)(\hat{t}+\tau) > 0.$$

[4] Generally, if $f \in \mathcal{C}^k$ and $f(t) = f(0) + t\, f'(0) + \cdots + \frac{t^{d-1}}{(d-1)!}\, f^{(d-1)}(0) + \frac{t^d}{d!}\, g(t), d \le k$, then $g(t) = d \int_0^1 (1-s)^{d-1}\, f^{(d)}(ts)\, ds$, which shows that $g \in \mathcal{C}^{k-d}$.

Denote $A_j(t) = [\,a_1(t) \quad \dots \quad a_j(t)\,]$. Using Fischer's inequality,

$$\det([\,B_1 \quad B_2\,]^* [\,B_1 \quad B_2\,]) \le \det(B_1^* B_1) \det(B_2^* B_2),$$

and the claim above, we find for every $j = 1, \dots, n$ a $d_j \le \hat{d}$ such that

$$\lim_{\tau \to 0} \frac{1}{\tau^{2d_j}} \det(A_j^* A_j)(\hat{t} + \tau)$$

is finite and positive. Also $d_j \le d_{j+1}$. Set $e_1 = d_1$, $e_j = d_j - d_{j-1}$, $j > 1$.

Expand the first column of A:

$$a_1(\hat{t} + \tau) = \hat{a}_1 + \tau\, \hat{a}_1^{(1)} + \frac{\tau^2}{2!}\, \hat{a}_1^{(2)} + \cdots + \frac{\tau^{k-1}}{(k-1)!}\, \hat{a}_1^{(k-1)} + \frac{\tau^k}{k!}\, \tilde{a}_1^{(k)},$$

where $\hat{a}_1^{(i)} = \frac{d^i}{dt^i} a_1(t)\big|_{t=\hat{t}}$ and the entries of $\tilde{a}_1^{(k)}$ are the kth derivatives of elements of a_1 at some points between $\hat{t}$ and $\hat{t} + \tau$.

Now, since $\lim_{\tau \to 0} \frac{1}{\tau^{2d_1}} \left| a_1(\hat{t} + \tau) \right|^2$ is positive, we have $\hat{a}_1 = \hat{a}_1^{(1)} = \cdots = \hat{a}_1^{(d_1 - 1)} = 0$ and $\hat{a}_1^{(d_1)} \ne 0$, and

$$\begin{aligned} q_1(t) &= \eta_1(\hat{t} + \tau)\, \frac{\frac{\tau^{d_1}}{d_1!}\, \hat{a}_1^{(d_1)} + \cdots + \frac{\tau^k}{k!}\, \tilde{a}_1^{(k)}}{\left| \frac{\tau^{d_1}}{d_1!}\, \hat{a}_1^{(d_1)} + \cdots + \frac{\tau^k}{k!}\, \tilde{a}_1^{(k)} \right|} \\ &= \eta_1(\hat{t} + \tau) \operatorname{sgn}(\tau)^{d_1}\, \frac{\frac{1}{d_1!}\, \hat{a}_1^{(d_1)} + \cdots + \frac{\tau^{k-d_1}}{k!}\, \tilde{a}_1^{(k)}}{\left| \frac{1}{d_1!}\, \hat{a}_1^{(d_1)} + \cdots + \frac{\tau^{k-d_1}}{k!}\, \tilde{a}_1^{(k)} \right|}\ . \end{aligned}$$

Choosing $\eta_1(\hat{t} + \tau) \operatorname{sgn}(\tau)^{d_1}$ smooth, i.e., changing the sign of η_1 at $\hat{t}$ whenever d_1 is odd, we have q_1 in $\mathcal{C}^{k-d_1}$ in a neighborhood of $\hat{t}$.

For $j \ge 2$ note first that $I - P_{j-1}(t)$ is the orthogonal projection onto the range of $A_{j-1}(t)$ so that

$$\begin{aligned} &\det([\,A_{j-1} \quad a_j\,]^* [\,A_{j-1} \quad a_j\,]) \\ &\quad = \det([\,(I - P_{j-1})A_{j-1} \quad P_{j-1} a_j\,]^* [\,(I - P_{j-1})A_{j-1} \quad P_{j-1} a_j\,]) \\ &\quad = \det\left(\begin{bmatrix} A_{j-1}^* A_{j-1} & 0 \\ 0 & (P_{j-1} a_j)^* P_{j-1} a_j \end{bmatrix} \right) = \det(A_{j-1}^* A_{j-1})\, \| P_{j-1} a_j \|^2 \end{aligned}$$

and $\lim_{\tau \to 0} \frac{1}{\tau^{e_j}} \left\| (P_{j-1} a_j)(\hat{t} + \tau) \right\|$ is finite and positive. Since $P_{j-1} a_j \in \mathcal{C}^{k - d_{j-1}}$, it follows that

$$(P_{j-1} a_j)(\hat{t} + \tau) = \tau^{e_j} \hat{b}_j + O(\tau^{e_j + 1})$$

with $\hat{b}_j \ne 0$. Hence,

$$q_j(\hat{t} + \tau) = \eta_j(\hat{t} + \tau) \operatorname{sgn}(\tau)^{e_j}\, \frac{\hat{b}_j + O(\tau)}{|\hat{b}_j + O(\tau)|}\ .$$

Again changing the sign of η_j at $\hat{t}$ whenever e_j is odd, we have q_j and consequently also $P_j = P_{j-1} - q_j q_j^*$ in $\mathcal{C}^{k-d_j}$, in a neighborhood of $\hat{t}$.

This way we get the whole Q and hence also R in $\mathcal{C}^{k-d}$.

The uniqueness follows from the fact that the sign changes in η_j's are also necessary for smoothness. $\square$

Remark 3.1. If $m = n$, then in the previous theorem it suffices that the matrix A_{n-1} satisfies the assumptions. This is because the nth column of Q is then determined (up to η_n with $|\eta_n| = 1$) by the previous columns.

Remark 3.2. The matrix $A = \begin{bmatrix} t^k|t| & 1 \\ t^d & 0 \end{bmatrix}$, $d \le k+1$, (see Example 3.1) shows that no extra smoothness can be generally expected for the R factor.

In the case of a real analytic matrix we do not need any extra assumptions. (A different proof of this result is found in [6].)

THEOREM 3.2. *Any $A \in \mathcal{C}^\omega(\mathbb{R}, \mathbb{R}^{m\times n}), m \ge n$, has a $\mathcal{C}^\omega$ QR-decomposition. This is similar for $A \in \mathcal{C}^\omega(\mathbb{R}, \mathbb{C}^{m\times n})$.*

Proof. The proof is by induction with respect to n. Assume $n = 1$. If $A \equiv 0$, set $Q = e_1$, $R = 0$. Else all the zeroes of A^*A are of finite order and the proof of Theorem 3.1 gives us an analytic QR-decomposition. Assume the assertion is true for all $m \times n$ matrices with $m > n$. Then for analytic $A = [\, A_n \quad a_{n+1} \,]$ take an analytic QR-decomposition $A_n = Q_n R_n$ and set $r = Q_n^* a_{n+1}, b = a_{n+1} - Q_n r$. If $b \equiv 0$, take an analytic unit q_{n+1} such that $Q_n^* q_{n+1} \equiv 0$ and set $Q = [\, Q_n \quad q_{n+1} \,]$,$R = \begin{bmatrix} R_n & r \\ 0 & 0 \end{bmatrix}$. Else take q_{n+1} to be the analytic unit vector in the direction of b as in the proof of Theorem 3.1. □

3.2. Schur-decomposition of a Hermitian matrix with multiple eigenvalues. Consider again the Schur-decomposition of $A \in \mathcal{C}^k(\mathbb{R}, \mathbb{C}_H^{n\times n})$. As we saw in section 2.2, if the eigenvalues of A are simple, then A admits a smooth Schur-decomposition. Under reasonable assumptions, we show next that A admits a Schur-decomposition with some possible loss of smoothness also in case some of its eigenvalues coalesce.

From the book of Kato [12] we know that Hermitian real analytic matrices have analytic eigendecompositions. Also, it is true (a theorem of Rellich, see [13]) that $\mathcal{C}^1$ Hermitian matrices have $\mathcal{C}^1$ eigenvalues. Generally, however, even $\mathcal{C}^\infty$ Hermitian matrices don't have $\mathcal{C}^2$ eigenvalues or continuous Schur-decompositions, as the following example shows.

Example 3.2 (see [17]). Let α, $\beta > 0$ and

$$A(t) = \begin{bmatrix} e^{-(\alpha+\beta)/|t|} & e^{-\beta/|t|} \sin(1/t) \\ e^{-\beta/|t|} \sin(1/t) & -e^{-(\alpha+\beta)/|t|} \end{bmatrix}.$$

Then $A \in \mathcal{C}^\infty$. The eigenvalues are

$$\lambda_\pm(t) = \pm e^{-\beta/|t|} \left(e^{-2\alpha/|t|} + \sin^2(1/t)\right)^{\frac{1}{2}}.$$

These are in $\mathcal{C}^1$ but not in $\mathcal{C}^2$, if $\alpha \ge \beta$. Notice that $\lambda_+(t) - \lambda_-(t) = o(|t|^d)$for all d.

Here we show that the unitary diagonalization can be made smooth, provided that the order of coalescing of the eigenvalues is not more than k.

Denote by W_e the class of matrices depending on a real parameter, for which the eigenvalues do not have contacts of order higher than e. More precisely, if an $n \times n$ matrix $A \in W_e$ and $\lambda_1(t), \ldots, \lambda_n(t)$ are the continuous eigenvalues of $A(t)$, then for any t and $i \ne j$ we have

$$\liminf_{\tau\to 0} \frac{|\lambda_i(t+\tau) - \lambda_j(t+\tau)|}{|\tau^e|} \in (0, \infty]. \tag{3.2}$$

THEOREM 3.3. *Any Hermitian matrix $A \in \mathcal{C}^k \cap W_e$, $e \le k$, has a $\mathcal{C}^{k-e}$ Schur-decomposition.*

Proof. The proof is by induction on e. Proposition 2.4 gives the theorem for $e = 0$ (i.e., for distinct eigenvalues).

Assume now that the assertion is true for W_{e-1} matrices and let $A \in \mathcal{C}^k \cap W_e$. We show first that A has a $\mathcal{C}^{k-e}$-decomposition in a neighborhood of any $\hat{t}$. We may assume that $\hat{t} = 0$. Let λ_0 be a p-fold eigenvalue of $A(0)$ and let Q_0 be a $\mathcal{C}^k$ orthonormal $n \times p$ matrix obtained as one column block of a block–Schur-decomposition (see Prop. 2.6) corresponding to the eigenvalues close to λ_0. Thus Q_0 is defined in a neighborhood of 0, such that

$$Q_0(t)^* A(t) Q_0(t) = \lambda_0 \, I + t \, S_1(t),$$

where $S_1 \in \mathcal{C}^{k-1}$. The eigenvalues of $A(t)$ near λ_0 are of the form $\lambda_0 + t\,\mu(t)$, where $\mu(t)$ is an eigenvalue of $S_1(t)$. If $\tilde{\mu}(t)$ is another eigenvalue of $S_1(t)$, we have

$$\frac{|\mu(t) - \tilde{\mu}(t)|}{|t^{e-1}|} = \frac{|\lambda(t) - \tilde{\lambda}(t)|}{|t^e|},$$

where $\lambda(t), \tilde{\lambda}(t)$ are eigenvalues of $A(t)$. Hence $S_1 \in \mathcal{C}^{k-1} \cap W_{e-1}$, so that by induction hypothesis it has a $\mathcal{C}^{k-1-(e-1)} = \mathcal{C}^{k-e}$ Schur-decomposition. This is true for all blocks corresponding to different eigenvalues of $A(0)$ and hence for A.

The assumption implies that the points where $A(t)$ has multiple eigenvalues are isolated. Cover $\mathbb{R}$ with a countable set of intervals $I_j = (\alpha_j, \beta_j)$ such that on each of these A has a $\mathcal{C}^{k-e}$ Schur-decomposition $A(t) = U_j(t)\Lambda_j(t)U_j(t)^*$, $I_{j-1} \cap I_{j+1} = \emptyset$ for all $j \in \mathbb{Z}$, and A has simple eigenvalues on each (α_j, β_{j-1}).

Set $U = U_0$, $\Lambda = \Lambda_0$, on $[\beta_{-1}, \alpha_1]$, and $\Pi_0 = I$. For $j = 1, 2, \ldots$ let $\tau_j = \frac{1}{2}(\alpha_j + \beta_{j-1})$. Since $A(\tau_j)$ has simple eigenvalues, there exists a permutation matrix Π_j such that

$$\Pi_j^T \Lambda_j(\tau_j) \Pi_j = \Pi_{j-1}^T \Lambda_{j-1}(\tau_j) \Pi_{j-1}.$$

Take $D_j \in \mathcal{C}^{k-e}((\alpha_j, \alpha_{j+1}], \mathbb{C}^{n \times n})$, such that $D_j(t)$ is diagonal and unitary for all t, and

$$D_j(t) = \begin{cases} \Pi_j^T U_j(t)^* U_{j-1}(t) \Pi_{j-1} & \text{on } (\alpha_j,\, \frac{2}{3}\alpha_j + \frac{1}{3}\beta_{j-1}), \\ I & \text{on } (\frac{1}{3}\alpha_j + \frac{2}{3}\beta_{j-1},\, \alpha_{j+1}] \ . \end{cases}$$

Then $U_j \Pi_j D_j$ is smooth and equal to $U_{j-1}\Pi_{j-1}$ on the beginning part and $U_j \Pi_j$ on the final part of $(\alpha_j, \alpha_{j+1}]$. Set $U = U_j \Pi_j D_j$ and $\Lambda = \Pi_j \Lambda_j \Pi_j$ on this interval. This gives U and Λ smooth on $[\beta_{-1}, \alpha_{j+1}]$.

Similarly continue for $j = -1, -2, \ldots$. □

Our next task is to prove that in the situation of the previous theorem the eigenvalues are in fact $\mathcal{C}^k$ functions. For this we need to take a closer look at the functions involved in the Taylor remainder terms.

For a function $f \in \mathcal{C}^0([-T, T])$ set $m_e(f)(t) = t^e f(t)$.

For $k \geq e$ denote by $\mathcal{C}^k_e$ the set of $f \in \mathcal{C}^0([-T, T])$ for which $m_e(f) \in \mathcal{C}^k([-T, T])$. With the norm

$$\|f\|_{k,e} = \max_{\substack{0 \leq j \leq e \\ 0 \leq l \leq k-e+j}} \left\| m_j(f)^{(l)} \right\|_\infty ,$$

$\mathcal{C}^k_e$ becomes a Banach space.

Proposition 3.4. *If f, $g \in \mathcal{C}^k_e$, then*

$$\text{i)} \quad fg \in \mathcal{C}^k_e, \quad \text{ii)} \quad f(t) \neq 0 \, \forall t \implies 1/f \in \mathcal{C}^k_e, \quad \text{iii)} \quad f(t) > 0 \, \forall t \implies \sqrt{f} \in \mathcal{C}^k_e.$$

Furthermore, the operations i)–iii) *are continuous.*

Proof. Clearly $\mathcal{C}^{k+1}_{e+1} \subset \mathcal{C}^{k}_{e}$ with $\|f\|_{k,e} \le \|f\|_{k+1,e+1}$.

For $f \in \mathcal{C}^{k+1}_{e+1}$, $0 \le d \le e$, set $h_d(f)(t) = t^{-d} m_{d+1}(f)'(t)$. We show first that h_d is continuous $\mathcal{C}^{k+1}_{e+1} \to \mathcal{C}^{k}_{e}$. Take $0 \le j \le e$ and $0 \le l \le k-e+j$. If $j < d$, then by footnote 4

$$\begin{aligned}\left|m_j(h_d(f))^{(l)}(t)\right| &= \left|\frac{d^l}{dt^l}\left(t^{j-d} m_{d+1}(f)'(t)\right)\right| \\ &= \left|\frac{d^l}{dt^l}\frac{1}{(d-j-1)!}\int_0^1 (1-s)^{d-j-1} m_{d+1}(f)^{(d-j+1)}(ts)\,ds\right| \\ &\le C\left\|m_{d+1}(f)^{(d-j+1+l)}\right\|_{\infty} \le C\,\|f\|_{k+1,e+1}\end{aligned}$$

since $d+1 \le e+1$ and $d-j+1+l \le k+1-(e+1)+d+1$. If $j \ge d$, then

$$\begin{aligned}\left|m_j(h_d(f))^{(l)}(t)\right| &= \left|\frac{d^l}{dt^l} m_{j-d}\big(m_{d+1}(f)'\big)(t)\right| \\ &= \left|\frac{d^l}{dt^l}\big(m_{j+1}(f)'(t) - (j-d) m_j(f)(t)\big)\right| \\ &= \left|m_{j+1}(f)^{(l+1)}(t) - (j-d) m_j(f)^{(l)}(t)\right| \le C\,\|f\|_{k+1,e+1}\end{aligned}$$

since $j+1 \le e+1$ and $l+1 \le k+1-(e+1)+d+1$. Hence $\|h_j(f)\|_{k,e} \le C\,\|f\|_{k+1,e+1}$, for some constant C depending only on T, k, and e.

If $f, g \in \mathcal{C}^{k+1}_{e+1}$ and $0 \le j \le e$, then

$$\begin{aligned}m_{j+1}(fg)'(t) &= \frac{d}{dt}\big(t^{-j-1} m_{j+1}(f)(t) m_{j+1}(g)(t)\big) \\ &= (h_j(f) m_j(g))(t) + (m_j(f) h_j(g))(t) \\ &\qquad -(j+1)t^{-j}(m_j(f) m_j(g))(t) \\ &= \big(m_j(h_j(f)g) + m_j(f h_j(g)) - (j+1) m_j(fg)\big)(t)\ .\end{aligned} \tag{3.3}$$

Now, i) is trivially true for any k if $e = 0$. Assume it is true for k and e. Then (3.3) with $j = e$ shows that $m_{e+1}(fg)' \in \mathcal{C}^k$, i.e., $fg \in \mathcal{C}^{k+1}_{e+1}$.

This is similar for ii) and iii). If they are true for k, e, and if $f \in \mathcal{C}^{k+1}_{e+1}$ vanishes nowhere, then

$$m_{e+1}(1/f)'(t) = \frac{d}{dt}(t^{2e+2}/m_{e+1}(f)) = \left((2e+2)m_e(1/f) - m_e\left(h_e(f)\,\frac{1}{f}\,\frac{1}{f}\right)\right)(t).$$

So, by i) we get $m_{e+1}(1/f)' \in \mathcal{C}^k$ and $1/f \in \mathcal{C}^{k+1}_{e+1}$. Further, if f is positive, then differentiating $m_{e+1}(\sqrt{f})(t)^2 = t^{e+1} m_{e+1}(f)(t)$ gives

$$m_{e+1}(\sqrt{f})' = \frac{e+1}{2}\, m_e(\sqrt{f}) + \frac{1}{2} m_e(h_e(f)\, 1/\sqrt{f})$$

in $\mathcal{C}^k$ and $\sqrt{f} \in \mathcal{C}^{k+1}_{e+1}$.

Next, we prove continuity of the multiplication. We show inductively that

$$\|fg\|_{k,e} \le C\,\|f\|_{k,e}\,\|g\|_{k,e}\,, \tag{3.4}$$

which is trivially true for any k if $e = 0$. Assume it is true for k and e and let $f, g \in \mathcal{C}^{k+1}_{e+1}$. From (3.3) we get for $j+1 \le e+1,\ l+1 \le k+1-(e+1)+j+1$

$$\begin{aligned}\left\| m_{j+1}(fg)^{(l+1)} \right\|_\infty &\le \left\| m_j(h_j(f)g)^{(l)} \right\|_\infty + \left\| m_j(f h_j(g))^{(l)} \right\|_\infty + (j+1) \left\| m_j(fg)^{(l)} \right\|_\infty \\ &\le C\big(\|h_j(f)\|_{k,e} \|g\|_{k,e} + \|f\|_{k,e} \|h_j(g)\|_{k,e} + (j+1) \|f\|_{k,e} \|g\|_{k,e} \big) \\ &\le C \|f\|_{k+1,e+1} \|g\|_{k+1,e+1} .\end{aligned}$$

Thus, continuity of the multiplication follows from (3.4) by

$$\left\| \tilde f \tilde g - fg \right\|_{k,e} \le \left\| (\tilde f - f)\tilde g \right\|_{k,e} + \|f(\tilde g - g)\|_{k,e} \ .$$

Proofs of continuity of $f \to 1/f$ and $f \to \sqrt{f}$ are similar to that of multiplication and hence are omitted. □

With these tools we can show the following theorem.

THEOREM 3.5. *Under the assumptions of* Theorem 3.3*, the eigenvalues can be taken to be $\mathcal{C}^k$ functions.*

Proof. From the proof of Theorem 3.3 we get that the eigenvalues of $A(t)$ are of the form $\lambda_0 + t\,\mu_1(t)$, where $\mu_1(t)$ is an eigenvalue of $S_1(t)$ and $S_1 \in \mathcal{C}^k_1$. If $\mu_1(0)$ is a q-fold eigenvalue of $S_1(0)$, let $P_1(t)$ be the eigenprojector corresponding to the q eigenvalues of $S_1(t)$ close to $\mu_1(0)$. Then (see [12])

$$P_1(t) = -\frac{1}{2\pi i} \int_\Gamma (S_1(t) - \zeta I)^{-1} d\zeta,$$

where Γ is the boundary of a small disk containing only these eigenvalues. By i) and ii) of Proposition 3.4 $\zeta \to (S_1(\cdot) - \zeta I)^{-1}$ defines a continuous function $\Gamma \to \mathcal{C}^k_1([-T,T], \mathbb{C}^{q\times q})$ so that also $P_1 \in \mathcal{C}^k_1$. Take $Q_1(0)$, the columns of which form an orthonormal basis for the range of $P_1(0)$, and let $Q_1(t)R_1(t) = P_1(t)Q_1(0)$ be a QR-decomposition which, again by Proposition 3.4, is in $\mathcal{C}^k_1$. The columns of $Q_1(t)$ form an orthonormal basis for the $S_1(t)$-invariant subspace corresponding to the eigenvalues close to $\mu_1(0)$. Hence,

$$Q_1^*(t) S_1(t) Q_1(t) = \mu_1(0)\ I + t\ S_2(t),$$

where $S_2 \in \mathcal{C}^k_2$. Thus the eigenvalues of $A(t)$ are of the form $\lambda_0 + t\,\mu_1(0) + t^2\,\mu_2(t)$, where μ_2 is an eigenvalue of S_2. Continuing this way, we get that the eigenvalues of $A(t)$ are of the form

$$\lambda(t) = \lambda_0 + t\,\mu_1(0) + t^2\,\mu_2(0) + \cdots + t^e \mu_e(t)\ ,$$

where μ_e is an eigenvalue of $S_e \in \mathcal{C}^k_e$. By assumption, it is a simple eigenvalue. Then, as above, the corresponding eigenvector Q_e is in $\mathcal{C}^k_e$ and the same holds for $\mu_e(t) = Q_e(t)^* S_e(t) Q_e(t)$. Hence $m_e(\mu_e)$ and, consequently, also λ are in $\mathcal{C}^k$. □

3.3. SVD in the rank deficient case and with multiple singular values.

Here we consider a smooth singular value decomposition of $A \in \mathcal{C}^k(\mathbb{R}, \mathbb{C}^{m\times n})$, $m \ge n$, allowing now multiple singular values and/or loss of rank at some points. Parallel to sections 3.1 and 3.2, we assume that at any point the order of coalescing of the squares of the singular values is at most $e \le k$ (as in (3.2))

$$\liminf_{\tau\to 0} \frac{\left|\sigma_i^2(t+\tau) - \sigma_j^2(t+\tau)\right|}{|\tau^e|} \in (0,\infty] \text{for all } t,$$

and that for every t

$$\limsup_{\tau\to 0} \frac{1}{\tau^{2d}} \det(A^*A)(t+\tau) > 0. \tag{3.5}$$

We also assume that only one of the singular values can become zero, i.e., the rank of A is always at least $n-1$.

To get smooth decomposition one has to allow sign changes in the singular values as in the analytic case of [3]. The result might be more properly called a *signed* SVD.

Theorem 3.6. *With the above assumptions there exists a $\mathcal{C}^{k-\max(d,e)}$-singular value decomposition of A. Moreover, the singular values can be taken to be $\mathcal{C}^k$ functions.*

Proof. According to Theorem 3.3, we get that the Hermitian positive semidefinite matrix A^*A has a $\mathcal{C}^{k-e}$–Schur-decomposition

$$A^*A = \tilde{V}\Sigma^2\tilde{V}^* \ , \qquad \Sigma^2 = \begin{bmatrix} \sigma_1^2 & & \\ & \ddots & \\ & & \sigma_n^2 \end{bmatrix}.$$

Let $\hat{t}$ be a point where A loses rank. We can assume that σ_n^2 is the smallest eigenvalue near $\hat{t}$. Then, by the smooth block Schur result (Proposition 2.6) we can write $\tilde{V} = [\,\tilde{V}_1 \quad \tilde{v}_n\,]$, where $\tilde{v}_n \in \mathcal{C}^k$. Write also $\Sigma^2 = \left[\begin{smallmatrix} \Sigma_1^2 & \\ & \sigma_n^2 \end{smallmatrix}\right]$. Similarly, we can make the decomposition

$$AA^* = [\,U_1 \quad \tilde{U}_2\,] \begin{bmatrix} \Sigma_1^2 & \\ & B^*B \end{bmatrix} [\,U_1 \quad \tilde{U}_2\,]^* ,$$

where $U_1 \in \mathcal{C}^{k-e}$, $\tilde{U}_2 \in \mathcal{C}^k$, and $B = A^*\tilde{U}_2 \in \mathcal{C}^k$.

Since $w = A\tilde{v}_n$ is in $\mathcal{C}^k$ and $\left\|w(\hat{t}+\tau)\right\| \geq c\,|\tau|^{\hat{d}}$ for some $c > 0, \hat{d} \leq d$, we get $w(\hat{t}+\tau) = \tau^{\hat{d}} w_0(\tau)$, where $w_0 \in \mathcal{C}^k_{\hat{d}}$ and $w_0(0) \neq 0$. So, by Proposition 3.4, $s(\tau) = \|w_0(\tau)\|$ is in $\mathcal{C}^k_{\hat{d}}$. Hence, $\sigma_n = m_{\hat{d}}(s) \in \mathcal{C}^k$. In Σ_1 we can take the square roots of $\sigma_1^2, \dots, \sigma_{n-1}^2$ with any combination of signs, keeping them constant until a possible sign change of one that becomes zero.

Since $\|B\|^2 = \sigma_n^2$, we have

$$B(\hat{t}+\tau) = \tau^{\hat{d}}\hat{B}(\tau),$$

where $\hat{B} \in \mathcal{C}^{k-\hat{d}}$ and $\hat{B}(0)$ has rank one. Then we have the $\mathcal{C}^{k-\hat{d}}$ block Schur-decomposition:

$$W\hat{B}^*\hat{B}W^* = \begin{bmatrix} s^2 & 0 \\ 0 & 0 \end{bmatrix}.$$

Set $U = [\,U_1 \quad \tilde{U}_2W\,] = [\,U_1 \quad u_n \quad U_0\,] \in \mathcal{C}^{k-\max(\hat{d},e)}$. Then we want to find a matrix V such that $A = U \left[\begin{smallmatrix} \Sigma \\ 0 \end{smallmatrix}\right] V^*$. We look for it in the form: $V = \tilde{V}D$, where $D = \mathrm{diag}(d_1, \dots, d_n)$ satisfies $|d_j| = 1$ for all j. We need

$$\begin{bmatrix} U_1^* \\ u_n^* \end{bmatrix} A \ [\,\tilde{V}_1 \quad \tilde{v}_n\,] = \begin{bmatrix} \Sigma_1 & \\ & \sigma_n \end{bmatrix} \bar{D},$$

from which $\mathrm{diag}(\bar{d}_1, \dots, \bar{d}_{n-1}) = \Sigma_1^{-1} U_1^* A \tilde{V}_1 \in \mathcal{C}^{k-e}$. Finally, from $\sigma_n \bar{d}_n = u_n^* w$ we get $d_n \in \mathcal{C}^{k-\max(\hat{d},e)}$ by

$$\bar{d}_n(\hat{t}+\tau) = \frac{u_n(\hat{t}+\tau)^* w_0(\tau)}{s(\tau)} \ .$$

Hence $\Sigma \in \mathcal{C}^k$ and $U, V \in \mathcal{C}^{k-\max(d,e)}$. □

Remark 3.3. If in the previous theorem $A \in \mathcal{C}^k(\mathbb{R}, \mathbb{R}^{n\times n})$, then condition (3.5) is not needed and we get a $\mathcal{C}^{k-e}$ SVD. This is because U_1 and V_1 are in $\mathcal{C}^{k-e}$ and they determine (up to sign) uniquely the last columns u_n, $v_n \in \mathcal{C}^{k-e}$.

4. Genericity of smooth factorizations. In this section we discuss how smooth factors of a generic one-parameter family of matrices are. For example, we show that, generically, $\mathcal{C}^k$ matrices have $\mathcal{C}^k$ QR-decomposition and $\mathcal{C}^k$ singular value decomposition.

In what follows, for $\mathcal{C}^k(\mathbb{R}, \mathbb{C}^{m\times n})$ we take the Whitney topology (see [11]; it is called the "fine" topology in [1]). Also, recall that a *generic* property is one that holds for a set that contains a countable intersection of open and dense sets.

4.1. *QR*-decomposition. Here we show that the QR-decomposition is, generically, as smooth as the family.

Theorem 4.1. *A generic $A \in \mathcal{C}^k(\mathbb{R}, \mathbb{R}^{m\times n})$, $k \geq 1$, $m \geq n$, has a $\mathcal{C}^k$ QR-decomposition. This is similar for generic $A \in \mathcal{C}^k(\mathbb{R}, \mathbb{C}^{m\times n})$.*

Proof. The set V of $m \times n$ real matrices having rank $r < n$ is a stratified manifold, where each stratum has codimension $(m-r)(n-r)$ (see [11]). So, for $m > n$, the codimension is at least 2, and by the weak transversality theorem a generic one-parameter family does not meet this set (again, see [11]). So, it has full rank for all t. In the case $m = n$, for generic $A \in \mathcal{C}^k(\mathbb{R}, \mathbb{R}^{n\times n})$ we have that $[a_1 \dots a_{n-1}]$ is of full rank for all t, and by Remark 3.1 A has a $\mathcal{C}^k$ QR-decomposition. In the complex case, each stratum has (real) codimension $2(m-r)(n-r)$, and hence a generic one-parameter family has full rank for all t. □

4.2. Schur-decomposition. For a generic matrix $A \in \mathcal{C}^k(\mathbb{R}, \mathbb{C}^{n\times n})$, the eigenvalues are simple and we have a $\mathcal{C}^k$ Schur-decomposition (and $\mathcal{C}^k$ eigendecomposition). On the other hand, not even for a generic family of analytic *real* matrices we can expect smooth eigenvalues. For example, any smooth *real* perturbation of

$$A(t) = \begin{bmatrix} 0 & 1 \\ t & 0 \end{bmatrix}$$

will have a defective eigenvalue and nondifferentiable eigenvalues at some t near 0.

More interesting is the case of Hermitian matrices. For the $\mathcal{C}^k$ case, the next theorem shows that generically there is no loss of smoothness.

Theorem 4.2. *A generic $A \in \mathcal{C}^k(\mathbb{R}, \mathbb{R}_H^{n\times n})$, $k \geq 1$ (or complex Hermitian) has a $\mathcal{C}^k$ Schur-decomposition.*

Proof. We show that a generic one-parameter family of real symmetric matrices has simple eigenvalues for every t. The set W_0 of symmetric matrices with a double eigenvalue is the image of the map $(U, D) \to UDU^T$, where U is orthogonal and D is diagonal with $d_{11} = d_{22}$. This map is real analytic and proper (compact sets have compact preimages). Hence by [10], W_0 has a Whitney stratification. The dimension of the set of $n \times n$ orthogonal matrices is $n(n-1)/2$. If $\tilde{U} = U \begin{bmatrix} c & s & \\ -s & c & \\ & & I \end{bmatrix}$, where

$c^2 + s^2 = 1$, then $\tilde{U} D \tilde{U}^T = U D U^T$. Hence the maximum dimension of the strata of W_0 is

$$n(n-1)/2 - 1 + n - 1 = n(n+1)/2 - 2.$$

Thus W_0 has codimension two and by the weak transversality theorem a generic one-parameter family does not meet W_0. Similarly, now let W_0 be the set of complex Hermitian matrices with a double eigenvalue. This is the image of the map $(U, D) \to UDU^*$, with U unitary, and D real diagonal with $d_{11} = d_{22}$. By viewing this as the equivalent real map $\psi : (U_r,\ U_i,\ D) \to (U_r D U_r^T + U_i D U_i^T, U_i D U_r^T - U_r D U_i^T)$, where $U = U_r + iU_i^T$, similarly to the previous case one infers that W_0 admits a stratification. Now, the (real) dimension of the set of $n \times n$ unitary matrices is n^2. If we let $\tilde{U} = U \begin{bmatrix} V & 0 \\ 0 & \Phi \end{bmatrix} = \tilde{U}_r + i\tilde{U}_i$, where $V \in \mathbb{C}^{2\times 2}$ is unitary, and $\Phi = \operatorname{diag}(\eta_1, \ldots, \eta_{n-2})$, $|\eta_j| = 1$, then $\psi(\tilde{U}_r,\ \tilde{U}_i,\ D) = \psi(U_r,\ U_i,\ D)$. Hence the maximum dimension of the strata of W_0 is

$$n^2 + (n-1) - 4 - (n-2) \ = \ n^2 - 3,$$

and since the set of Hermitian matrices has real dimension n^2 we see that W_0 has codimension 3 and generically a one-parameter family does not meet it. □

Example 4.1. Let $A(t) = \left[\begin{smallmatrix} t & 0 \\ 0 & 0 \end{smallmatrix}\right]$. The perturbation $A_\varepsilon(t) = \left[\begin{smallmatrix} t & \varepsilon \\ \varepsilon & 0 \end{smallmatrix}\right]$ has simple eigenvalues $\frac{1}{2}(t \pm \sqrt{t^2 + 4\varepsilon^2})$ for every t. This example also shows that analytic symmetric matrices of *two* parameters do not necessarily have smooth eigenvalues.

4.3. Singular value decomposition. Here we show, in the $\mathcal{C}^k$ case, that generically the singular value decomposition is as smooth as the family.

THEOREM 4.3. *A generic* $A \in \mathcal{C}^k(\mathbb{R}, \mathbb{R}^{m\times n})$, $k \geq 1$ (*or complex valued*) *has a* $\mathcal{C}^k$ *singular value decomposition.*

Proof. We can assume that $m \geq n$; otherwise apply the following to A^T. We show first that generically a one-parameter family of real matrices has simple singular values for every t, i.e., we have $e = 0$ in Theorem 3.6.

Similarly to the previous proof, the set V_0 of real $m \times n$ matrices with a double singular value is the image of the proper analytic map $(U, \Sigma, V) \to U\Sigma V^T$, where $U \in \mathbb{R}^{m\times n}$ is orthonormal, Σ is diagonal with $\sigma_{11} = \sigma_{22}$, and V is orthogonal. As above,

$$U \begin{bmatrix} c & s & \\ -s & c & \\ & & I \end{bmatrix} \Sigma \begin{bmatrix} c & -s & \\ s & c & \\ & & I \end{bmatrix} V^T = U\Sigma V^T$$

so that the dimensions of the strata of V_0 do not exceed

$$mn - n(n+1)/2 + n - 1 + n(n-1)/2 - 1 = mn - 2.$$

Hence a generic one-parameter family does not intersect V_0. The complex case is handled similarly to Theorem 4.2.

If $m > n$, then as in the proof of Theorem 4.1 we generically get $d = 0$ for Theorem 3.6. This is also true for complex $n \times n$ matrices. For real $n \times n$ matrices we get the result by Remark 3.3. □

Acknowledgment. The authors are grateful to J. Guckenheimer for suggesting the study of generic smoothness of the factors.

REFERENCES

[1] V. I. Arnold, *Geometrical Methods in the Theory of Ordinary Differential Equations*, 2nd ed., Springer-Verlag, New York, 1988.

[2] G. Benettin, G. Galgani, L. Giorgilli, and J. M. Strelcyn, *Lyapunov exponents for smooth dynamical systems and for Hamiltonian systems; a method for computing all of them; Part* I: *Theory, Part* II: *Numerical applications*, Meccanica, 15 (1980), pp. 9–20, 21–30.

[3] A. Bunse-Gerstner, R. Byers, V. Mehrmann, and N. K. Nichols, *Numerical computation of an analytic singular value decomposition by a matrix valued function*, Numer. Math., 60 (1991), pp. 1–40.

[4] T. F. Coleman and D. C. Sorensen, *A note on the computation of and orthonormal basis for the null space of a matrix*, Math. Programming, 29 (1984), pp. 234–242.

[5] L. Dieci, R. D. Russell, and E. S. Van Vleck, *On the computation of Lyapunov exponents for continuous dynamical systems*, SIAM J. Numer. Anal., 34 (1997), pp. 402–423.

[6] H. Gingold and P. F. Hsieh, *Globally analytic triangularization of a matrix function*, Linear Algebra Appl., 169 (1992), pp. 75–101.

[7] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 2nd ed., The Johns Hopkins University Press, Baltimore, MD, 1989.

[8] J. M. Greene and J-S. Kim, *The calculation of Lyapunov spectra*, Physica D, 24 (1987), pp. 213–225.

[9] J. K. Hale, *Ordinary Differential Equations*, Krieger Publishing Co., Huntington, NY, 1980.

[10] R. M. Hardt, *Stratification of real analytic mappings and images*, Inventiones Math., 28 (1975), pp. 193–208.

[11] M. W. Hirsch, *Differential Topology*, Springer-Verlag, New York, 1976.

[12] T. Kato, *Perturbation Theory for Linear Operators*, 2nd ed., Springer-Verlag, Berlin, 1976.

[13] F. Rellich, *Perturbation Theory of Eigenvalue Problems*, Gordon and Breach, New York, 1969.

[14] W. Rheinboldt, *On the computation of multi-dimensional solution manifolds of parametrized equations*, Numer. Math., 53 (1988), pp. 165–181.

[15] Y. Sibuya, *Some global properties of matrices of functions of one variable*, Math. Ann., 161 (1965), pp. 67–77.

[16] K. Wright, *Differential equations for the analytical singular value decomposition of a matrix*, Numer. Math., 63 (1992), p. 283.

[17] W. Wasow, *On the spectrum of Hermitian matrix-valued functions*, Resultate der Math., 2 (1979), pp. 206–214.

SIAM J. MATRIX ANAL. APPL.
Vol. 20, No. 3, pp. 820–837

DECAY RATES OF THE INVERSE OF NONSYMMETRIC TRIDIAGONAL AND BAND MATRICES*

REINHARD NABBEN†

Abstract. It is well known that the inverse $C = [c_{i,j}]$ of an irreducible nonsingular symmetric tridiagonal matrix is given by two sequences of real numbers, $\{u_i\}$ and $\{v_i\}$, such that $c_{i,j} = u_i v_j$ for $i \leq j$. A similar result holds for nonsymmetric matrices A. There the inverse can be described by four sequences $\{u_i\}, \{v_i\}, \{x_i\}$, and $\{y_i\}$ with $u_i v_i = x_i y_i$. Here we characterize certain properties of A, i.e., being an M-matrix or positive definite, in terms of the u_i, v_i, x_i, and y_i. We also establish a relation of zero row sums and zero column sums of A and pairwise constant u_i, v_i, x_i, and y_i. Moreover, we consider decay rates for the entries of the inverse of tridiagonal and block tridiagonal (banded) matrices. For diagonally dominant matrices we show that the entries of the inverse strictly decay along a row or column. We give a sharp decay result for tridiagonal irreducible M-matrices and tridiagonal positive definite matrices. We also give a decay rate for arbitrary banded M-matrices.

Key words. decay rates, tridiagonal matrices, inverses of tridiagonal matrices

AMS subject classifications. 15A48, 15A57, 65F10

PII. S0895479897317259

1. Introduction.

In many mathematical problems which give rise to a linear system of equations the system matrix is tridiagonal or block tridiagonal. For example, the numerical solution of partial differential equations often leads to tridiagonal (one-dimensional problems) and block tridiagonal (higher dimensional problems) matrices. Therefore, these classes of matrices have been extensively studied. A review of this topic is given by Meurant [Meu] for symmetric matrices. One of the most important results is established by Gantmacher and Krein [GK2], who proved that the inverse of a symmetric irreducible tridiagonal matrix is given by two sequences, $\{u_i\}$ and $\{v_i\}$, of real numbers. A similar result holds for the inverse of a nonsymmetric irreducible tridiagonal matrix. There the inverse can be described elegantly by four sequences $\{u_i\}, \{v_i\}, \{x_i\}, \{y_i\}$, which satisfy $u_i v_i = x_i y_i$.

Considering (block) tridiagonal matrices and their inverses there are different kinds of problems. On one hand, one wants to find explicit formulas for the inverses of (block) tridiagonal matrices, and on the other hand, one wants to characterize matrices whose inverses are (block) tridiagonal. Here we continue considering tridiagonal matrices and their inverses. We characterize tridiagonal M-matrices and tridiagonal symmetric positive definite matrices in terms of the $\{u_i\}, \{v_i\}, \{x_i\}$, and $\{y_i\}$. Moreover, we explain the connection of zero row sums and zero columns of a tridiagonal matrix and pairwise constant u_i, v_i, x_i, and y_i.

It has been observed that for large classes of banded matrices the entries of the inverse tend to zero as $|i - j|$ becomes larger. The rate of decay is important to construct sparse approximations of the inverse as preconditioners. In [D] and [DMS] it is shown that the entries of the inverse of a symmetric positive matrix are bounded in an exponentially decaying manner along a row or column. For nonsymmetric tridiagonal M-matrices, which are strictly diagonally dominant by rows and by columns,

*Received by the editors January 23, 1997; accepted for publication (in revised form) by G. Golub March 18, 1998; published electronically May 7, 1999.
http://www.siam.org/journals/simax/20-3/31725.html

†Fakultät für Mathematik, Universität Bielefeld, Postfach 10 01 31, 33 501 Bielefeld, Federal Republic of Germany (nabben@mathematik.uni-bielefeld.de).

we establish that the entries of the inverse indeed decay along each row and column away from the diagonal. We also give a bound for this decay. This result generalizes a result by Concus, Golub, and Meurant [CGM] for symmetric matrices.

For arbitrary tridiagonal M-matrices or positive definite matrices the entries of the inverse do not strictly decay along a row in general. However, here we establish a sharp decay result related to two diagonal entries for the inverses of tridiagonal matrices. This result can be proved easily and continues the number of elegant results on tridiagonal matrices. For symmetric matrices the result is related to a result due to Vassilevski [Vas], who used Cauchy–Bunyakowski–Schwarz constants to describe the decay. Therefore, we establish a new proof of Vassilevski's theorem and obtain a new way to compute or estimate Cauchy–Bunyakowski–Schwarz constants for tridiagonal matrices. Moreover, we establish a decay rate for the inverses of banded M-matrices.

2. Inverses of tridiagonal matrices. One of the most important results on symmetric tridiagonal matrices is the result by Gantmacher and Krein [GK2] which describes the structure of the inverse of these matrices.

THEOREM 2.1. *Let $A \in \mathbb{R}^{n,n}$ be symmetric, irreducible, and nonsingular. Then A is tridiagonal if and only if $C = [c_{i,j}]$ is given by two sequences $\{u_i\}_{i=1}^n$, $\{v_i\}_{i=1}^n$ of numbers such that*

$$(2.1) \qquad C = \begin{bmatrix} u_1v_1 & u_1v_2 & \cdots & u_1v_n \\ u_1v_2 & u_2v_2 & \cdots & u_2v_n \\ \vdots & \vdots & \ddots & \vdots \\ u_1v_n & u_2v_n & \cdots & u_nv_n \end{bmatrix} \quad \textit{or} \quad c_{i,j} = \begin{cases} u_iv_j & i \le j, \\ u_jv_i & i \ge j. \end{cases}$$

Matrices of the form (2.1) can be described more elegantly as the Hadamard product (elementwise product) of a so-called *type D* and a *flipped type D matrix* (see [Mar] and [MNNST]), i.e.,

$$(2.2) \qquad C = \begin{bmatrix} u_1 & u_1 & \cdots & u_1 \\ u_1 & u_2 & \cdots & u_2 \\ \vdots & \vdots & \ddots & \vdots \\ u_1 & u_2 & \cdots & u_n \end{bmatrix} \circ \begin{bmatrix} v_1 & v_2 & \cdots & v_n \\ v_2 & v_2 & \cdots & v_n \\ \vdots & \vdots & \ddots & \vdots \\ v_n & v_n & \cdots & v_n \end{bmatrix}.$$

Since Gantmacher and Krein introduced matrices of the form (2.1) in 1937 [GK1] many names have been given to them. Originally, Gantmacher and Krein called them *matrices à un couple* while later on they were called *matrices factorisables* by Baranger and Duc-Jacquet in [BD]. In the Russian edition of [GK2] (1941) they are named *odnoparnyeh*, which means one pair(ed). In the German edition of [GK2] (1960) they are called *einpaarig*. Karlin named them *Green's matrices* in [K], which is also used by Barrett in [Ba], while Markham called them *matrices of a couple* in [Mar]. Ròzsa used in [R] the names *one-pair-matrices* and *separable matrices*.

In the study of inverses of band matrices Asplund [As] defined in 1958 a square matrix $A = [a_{i,j}]$ to be a *Green's matrix of order p* if all submatrices have rank $\le p$ if their elements belong to the part of $[a_{i,j}]$ for which $j + p > i$. Later on Ròzsa generalized this concept in [R] by introducing *Green's matrices of grade* $\{p, q\}$ which are Green's matrices of order p whose transposes are Green's matrices of order q. Obviously, a symmetric Green's matrix of grade $\{1, 1\}$ is a matrix of the form (2.1) and vice versa.

For nonsymmetric tridiagonal matrices $A \in \mathbb{R}^{n,n}$ with

$$A = \begin{bmatrix} a_1 & b_1 & & & \\ c_1 & a_2 & b_2 & & \\ & \ddots & \ddots & \ddots & \\ & & c_{n-2} & a_{n-1} & b_{n-1} \\ & & & c_{n-1} & a_n \end{bmatrix}, \tag{2.3}$$

Capovani [C1], [C2] proved the following theorem.

THEOREM 2.2. *Let $A \in \mathbb{R}^{n,n}$ be irreducible and nonsingular. Then A is tridiagonal if and only if there exist four sequences $\{u_i\}$, $\{v_i\}$, $\{x_i\}$, $\{y_i\}$, where $u_iv_i = x_iy_i$ for all i, such that $A^{-1} =: C = [c_{i,j}]$ is given by*

$$c_{i,j} = \begin{cases} u_iv_j & i \leq j, \\ x_iy_j & i \geq j. \end{cases} \tag{2.4}$$

As in the symmetric case matrices of the form (2.4) can be written nicely as the Hadamard product of two matrices:

$$C = \begin{bmatrix} u_1 & u_1 & \cdots & \cdots & u_1 \\ x_1 & u_2 & \cdots & \cdots & u_2 \\ x_1 & x_2 & u_3 & \cdots & u_3 \\ \vdots & \vdots & & \ddots & \vdots \\ x_1 & x_2 & \cdots & \cdots & u_n \end{bmatrix} \circ \begin{bmatrix} v_1 & v_2 & \cdots & \cdots & v_n \\ y_2 & v_2 & \cdots & \cdots & v_n \\ y_3 & y_3 & v_3 & \cdots & v_n \\ \vdots & \vdots & & \ddots & \vdots \\ y_n & y_n & \cdots & \cdots & v_n \end{bmatrix}. \tag{2.5}$$

Matrices of the form (2.4) are Green's matrices of grade $\{1, 1\}$ and they are also called $\{1, 1\}$ semiseparable by Rozsa in [R] if, in addition, $u_iv_j - x_iy_j \neq 0$ for $|i-j| = 1$.

Here we should note that the inverse of an irreducible tridiagonal matrix is given by three vectors, since $u_iv_i = x_iy_i$ for all i. Moreover, one can choose $u_1 = x_1 = 1$; thus the inverse of a tridiagonal matrix is given by $3n - 2$ parameters. However, to give a better illustration of the structure of the inverse we will use and consider four sequences.

Capovani [C1], [C2] also established equations for certain minors of matrices of the form (2.4) which we state here again in a different way. For $A \in \mathbb{R}^{n,n}$ let A_{ij} denote the $(n-1) \times (n-1)$ matrix obtained from A by deleting the ith row and the jth column. Moreover, for a matrix of the form (2.4) define

$$d_{i,j} := x_iy_j - v_iu_j, \qquad h_{i,j} := u_iv_j - y_ix_j. \tag{2.6}$$

Then one can show the following theorem.

THEOREM 2.3. *Let $C \in \mathbb{R}^{n,n}$ be a nonsingular matrix of the form* (2.4). *Then*

$$\det C = x_1y_n \prod_{i=1}^{n-1} d_{i+1,i} = u_1v_n \prod_{i=1}^{n-1} h_{i+1,i}, \tag{2.7}$$

$$\det C_{i,i+1} = \frac{\det C}{d_{i+1,i}} \qquad \det C_{i+1,i} = \frac{\det C}{h_{i+1,i}}, \tag{2.8}$$

$$\det C_{i,j} = 0 \quad \text{for} \;\; |i-j| > 1. \tag{2.9}$$

Here we should mention that Barrett gives in [Ba] another characterization of tridiagonal matrices.

THEOREM 2.4. *Let $C = [c_{i,j}] \in \mathbb{R}^{n,n}$ with nonzero entries $c_{2,2}, \dots, c_{n-1,n-1}$. Then C^{-1} is tridiagonal if and only if C has the triangle property, i.e.,*

$$c_{i,j} = \frac{c_{i,k}c_{k,j}}{c_{k,k}} \quad \textit{for all} \quad i < k < j, \quad i > k > j.$$

Note that none of these theorems includes the other. In Theorem 2.2 there is a restriction on irreducibility while in Theorem 2.4 there is a restriction of the diagonal entries of the inverse.

In the following we complete this section with some new results. We will characterize nonsingular tridiagonal M-matrices. A matrix $A = [a_{i,j}]$ is called a (nonsingular) M-matrix if $a_{i,j} \leq 0$ for $i \neq j$ and A^{-1} is a nonnegative matrix. We characterize nonsingular tridiagonal M-matrices in terms of the sequences $\{u_i\}, \{v_i\}, \{x_i\}$, and $\{y_i\}$ which give A^{-1}. To do so we define for $i = 1, \dots, n-1$

$$\alpha_i := \frac{u_i v_{i+1}}{u_{i+1} v_i}, \quad \beta_i := \frac{x_i y_{i+1}}{x_{i+1} y_i}. \tag{2.10}$$

THEOREM 2.5. *Let $A \in \mathbb{R}^{n,n}$ be irreducible and nonsingular. Then the following are equivalent:*

1. *A is a tridiagonal M-matrix,*
2. *A^{-1} is a matrix of the form (2.4), where all u_i, v_i, x_i, and y_i have the same sign and*

$$\alpha_i \beta_i < 1 \quad \textit{for all } i.$$

Proof. $A = [a_{i,j}]$ is given by

$$a_{i,j} = (-1)^{i+j} \frac{\det A^{-1}_{j,i}}{\det A^{-1}}.$$

Thus Theorem 2.3 yields

$$a_{i,i+1} = -\frac{1}{u_{i+1}v_i - x_i y_{i+1}}, \quad a_{i+1,i} = -\frac{1}{x_{i+1}y_i - u_i v_{i+1}}.$$

If A is an M-matrix we have

$$x_{i+1}y_i - u_i v_{i+1} > 0 \quad \text{and} \quad u_{i+1}v_i - x_i y_{i+1} > 0.$$

Therefore,

$$\frac{u_i v_{i+1}}{x_{i+1} y_i} < 1 \quad \text{and} \quad \frac{x_i y_{i+1}}{u_{i+1} v_i} < 1.$$

Hence $\alpha_i \beta_i < 1$.

If $\alpha_i \beta_i < 1$ we have

$$\frac{u_i v_{i+1}}{x_{i+1} y_i} \frac{x_i y_{i+1}}{u_{i+1} v_i} < 1.$$

Since $u_i v_i = x_i y_i$ for all i, i.e.,

$$\frac{x_i}{v_i} = \frac{u_i}{y_i}, \quad \text{and} \quad \frac{y_{i+1}}{u_{i+1}} = \frac{v_{i+1}}{x_{i+1}},$$

we obtain

$$\frac{u_i v_{i+1}}{x_{i+1} y_i} = \frac{x_i y_{i+1}}{u_{i+1} v_i}.$$

Therefore,

$$x_{i+1} y_i - u_i v_{i+1} > 0 \quad \text{and} \quad u_{i+1} v_i - x_i y_{i+1} > 0.$$

Since A^{-1} is a nonnegative matrix we then obtain that A is an M-matrix. □

Corollary 2.6. *Let $A \in \mathbb{R}^{n,n}$ be symmetric irreducible and nonsingular. Then the following are equivalent:*

1. *A is a tridiagonal M-matrix,*
2. *A^{-1} is a matrix of the form (2.4) given by the sequences $\{u_i\}$ and $\{v_i\}$ where all u_i, v_i and have the same sign and*

$$\frac{u_1}{v_1} < \cdots < \frac{u_n}{v_n}.$$

In [MNNST] another characterization of tridiagonal M-matrices is given. There it is used that for each nonsymmetric irreducible tridiagonal M-matrix A there exists a diagonal matrix D such that DA is symmetric. Here we can avoid this symmetrization.

For symmetric positive definite matrices we immediately obtain the following characterization.

Corollary 2.7. *Let $A \in \mathbb{R}^{n,n}$ be a nonsingular symmetric tridiagonal matrix. A is positive definite if and only if there exists a diagonal matrix $D = [d_{ij}]$ with $|d_{ii}| = 1$ such that*

$$(DAD)^{-1} = \begin{bmatrix} u_1 & u_1 & \cdots & u_1 \\ u_1 & u_2 & \cdots & u_2 \\ \vdots & \vdots & \ddots & \vdots \\ u_1 & u_2 & \cdots & u_n \end{bmatrix} \circ \begin{bmatrix} v_1 & v_2 & \cdots & v_n \\ v_2 & v_2 & \cdots & v_n \\ \vdots & \vdots & \ddots & \vdots \\ v_n & v_n & \cdots & v_n \end{bmatrix}, \tag{2.11}$$

where all u_i and v_i in (2.11) *have the same sign and*

$$0 < \frac{u_1}{v_1} < \frac{u_2}{v_2} < \cdots < \frac{u_n}{v_n}.$$

Proof. Since A is symmetric tridiagonal there exists a diagonal matrix $D = [d_{i,j}]$ with $|d_{i,i}| = 1$ such that DAD is a Z-matrix, i.e., all off-diagonal entries of DAD are nonpositive. Hence, with Theorem 2.5, DAD is a symmetric M-matrix, thus positive definite, if and only if (2.11) holds. □

These characterizations of tridiagonal M-matrices and tridiagonal positive definite matrices will be useful in the next section.

3. Decay rates. In this section we consider the decay of the elements of the inverse of tridiagonal and banded matrices. Several papers already established results on this topic. In [D] and [DMS] it is shown that the entries of the inverse of a banded symmetric positive matrix are bounded in an exponentially decaying manner along a row or column. In [EP] decay rates of inverses of certain M-matrices, i.e., those matrices whose factors in an LDU-factorization are bounded by diagonally dominant Toeplitz matrices, are given. As shown in [EP] this class of M-matrices contains the diagonally dominant M-matrices which are Toeplitz matrices.

Here we will give some decay results for arbitrary tridiagonal and banded M-matrices and we will show that the entries of a nonsymmetric tridiagonal diagonally dominant matrix indeed decay along a row and column.

A matrix $B = [b_{i,j}]$ is diagonally dominant by columns if

$$b_{i,i} \geq \sum_{j \neq i} |b_{j,i}| \quad \text{for all } i.$$

If

$$b_{i,i} \geq \sum_{j \neq i} |b_{i,j}| \quad \text{for all } i,$$

then B is called diagonally dominant by rows. If the above inequalities are strict, then B is called strictly diagonally dominant by columns or by rows, respectively.

For A as in (2.3) being a symmetric tridiagonal M-matrix which is diagonally dominant by columns (with $a_1 > b_1$ and $a_n > b_{n-1}$) Concus, Golub, and Meurant proved in [CGM] that the sequence $\{u_i\}$ is strictly increasing while $\{v_i\}$ is strictly decreasing. Thus the entries of A^{-1} indeed strictly decay along each row or column. Moreover, they proved that

$$(A^{-1})_{ij} \leq (A^{-1})_{ii} \rho^{|i-j|}, \tag{3.1}$$

where $1/\rho = \min_{k \geq 2}((a_k + b_{k-1})/b_k))$. Since A is diagonally dominant one has $\rho < 1$.

We will generalize this result to the nonsymmetric case. In the following we also assume for simplicity that A is a tridiagonal Z-matrix, i.e.,

$$A = \begin{bmatrix} a_1 & -b_1 & & & \\ -c_1 & a_2 & -b_2 & & \\ & \ddots & \ddots & \ddots & \\ & & -c_{n-2} & a_{n-1} & -b_{n-1} \\ & & & -c_{n-1} & a_n \end{bmatrix} \qquad b_i, c_i \geq 0. \tag{3.2}$$

The entries of the sequences $\{u_i\}, \{v_i\}, \{x_i\}$, and $\{y_i\}$, which give A^{-1}, can be computed as follows.

Lemma 3.1. *Let $A \in \mathbb{R}^{n,n}$ be an irreducible tridiagonal Z-matrix. Then*

$$\begin{aligned}
u_1 &= 1, \qquad u_2 = \frac{a_1}{b_1}, \\
u_i &= \frac{a_{i-1}u_{i-1} - c_{i-2}u_{i-2}}{b_{i-1}}, \quad i = 3, \ldots, n, \\
x_1 &= 1, \qquad x_2 = \frac{a_1}{c_1}, \\
x_i &= \frac{a_{i-1}x_{i-1} - b_{i-2}x_{i-2}}{c_{i-1}}, \quad i = 3, \ldots, n, \\
v_n &= \frac{1}{a_n u_n - c_{n-1}u_{n-1}}, \\
y_i &= \frac{u_i v_i}{x_i}, \quad i = n, \ldots, 1, \\
v_i &= \frac{1 + b_i x_i y_{i+1}}{a_i u_i - c_{i-1}u_{i-1}}, \quad i = n-1, \ldots, 1.
\end{aligned}$$

Proof. Multiplying the $(i-1)$th row of A with the ith column of A^{-1} gives u_i. Multiplying the ith row of A^{-1} with the $(i-1)$th column of A gives x_i. Multiplying the ith row of A with the ith column of A^{-1} gives v_i. Moreover, we use that $u_i v_i = x_i y_i$. □

Similar recurrence formulas for the entries of A^{-1} as in Lemma 3.1 can be found in [Be], [BC]. For the next theorem we define

$$\frac{1}{\rho_1} := \min_{i\geq 2} \frac{a_i - c_{i-1}}{b_i}, \qquad \frac{1}{\rho_2} := \min_{i\geq 2} \frac{a_i - b_i}{c_{i-1}},$$

$$\frac{1}{\rho_3} := \min_{i\geq 2} \frac{a_i - c_i}{b_{i-1}}, \qquad \frac{1}{\rho_4} := \min_{i\geq 2} \frac{a_i - b_{i-1}}{c_i}.$$

THEOREM 3.2. *Let $A \in \mathbb{R}^{n,n}$ be an irreducible tridiagonal M-matrix. If A is diagonally dominant by rows and if $a_1 > b_1$ and $a_n > c_{n-1}$, then $\{u_i\}$ is strictly increasing while $\{y_i\}$ is strictly decreasing. Moreover,*

$$\begin{aligned} (A^{-1})_{i,j} &\leq \rho_1^{j-i}(A^{-1})_{j,j} \quad \text{for } i<j,\\ (A^{-1})_{i,j} &\leq \rho_2^{i-j}(A^{-1})_{j,j} \quad \text{for } i>j. \end{aligned}$$

If A is diagonally dominant by columns and if $a_1 > c_1$ and $a_n > b_{n-1}$, then $\{x_i\}$ is strictly increasing while $\{v_i\}$ is strictly decreasing. Moreover,

$$\begin{aligned} (A^{-1})_{i,j} &\leq \rho_3^{j-i}(A^{-1})_{i,i} \quad \text{for } i<j,\\ (A^{-1})_{i,j} &\leq \rho_4^{i-j}(A^{-1})_{i,i} \quad \text{for } i>j. \end{aligned}$$

Proof. First assume that A is diagonally dominant by rows and $a_1 > b_1$, $a_n > b_{n-1}$. It is clear that $u_2 > u_1$. By the induction hypothesis and Lemma 3.1 we get

$$u_i = \frac{a_{i-1}u_{i-1} - c_{i-2}u_{i-2}}{b_{i-1}} > \frac{a_{i-1} - c_{i-2}}{b_{i-1}} u_{i-1} > u_{i-1}.$$

To obtain the desired result for the y_i we have to establish a recursive formula. Multiplying the ith row of A^{-1} with the ith column of A gives

$$y_i = \frac{1 + c_i u_i v_{i+1}}{a_i x_i - b_{i-1} x_{i-1}}.$$

Multiplying the $(i+1)$th row of A^{-1} with the ith column of A gives

$$-b_{i-1}x_{i-1} + a_i x_i - c_i x_{i+1} = 0.$$

Multiplying the $(i+1)$th row of A with the $(i+1)$th column of A^{-1} gives

$$-c_i u_i v_{i+1} + a_{i+1} x_{i+1} y_{i+1} - b_{i+1} x_{i+1} y_{i+2} = 1.$$

Hence

$$y_i = \frac{a_{i+1}x_{i+1}y_{i+1} - b_{i+1}x_{i+1}y_{i+2}}{c_i x_{i+1}} = \frac{a_{i+1}}{c_i} y_{i+1} - \frac{b_{i+1}}{c_i} y_{i+2}$$

and

$$y_{n-1} = \frac{a_n}{c_{n-1}} y_n.$$

We then have $y_{n-1} > y_n$ and by induction $y_i > y_{i+1}$. The decay rates follow immediately from the recursive formulas for the u_i and y_i and the special structure of A^{-1}.

If A is diagonally dominant by columns, $a_1 > c_1$ and $a_n > b_{n-1}$ we can obtain the desired results by considering A^T as above. □

For matrices which are not M-matrices but satisfy the other assumptions of Theorem 3.2 we obtain that the sequences of the absolute values strictly increase or decrease, respectively.

EXAMPLE 3.1. *Let A be*

$$A = \begin{bmatrix} 2 & -1 & 0 & 0 & 0 \\ -3 & 7 & -3 & 0 & 0 \\ 0 & -7 & 17 & -8 & 0 \\ 0 & 0 & -2 & 6 & -3 \\ 0 & 0 & 0 & -1 & 2 \end{bmatrix}.$$

The inverse of A is given by

$$A^{-1} = \begin{bmatrix} 0.6905 & 0.1270 & 0.0283 & 0.0504 & 0.0756 \\ 0.3809 & 0.2539 & 0.0567 & 0.1007 & 0.1511 \\ 0.1983 & 0.1322 & 0.1039 & 0.1847 & 0.2770 \\ 0.0881 & 0.0588 & 0.0462 & 0.3043 & 0.4565 \\ 0.0441 & 0.0294 & 0.0231 & 0.1522 & 0.7282 \end{bmatrix}.$$

A is diagonally dominant by rows but not by columns. We obtain for the vectors (or sequences) u, v, x, y:

$$\begin{aligned} u &= [1, 2, 3.6667, 6.0417, 9.6389]^T, \\ v &= [0.6905, 0.1270, 0.0283, 0.0504, 0.0756]^T, \\ x &= [1, 0.6667, 0.5238, 3.4524, 16.5238]^T, \\ y &= [0.6905, 0.3809, 0.1983, 0.0881, 0.0441]^T. \end{aligned}$$

As stated in Theorem 3.2 *the u_i are strictly increasing while the y_i are strictly decreasing. Thus the entries of A^{-1} strictly decay along a column, but not along a row, away from the diagonal. Moreover, $\rho_1 = 0.8$, while $\rho_2 = 7/9$.*

EXAMPLE 3.2. *Let A be*

$$A = toeplitz(-0.9, 5, -1.1) \in \mathbb{R}^{15,15}.$$

Here we compare our decay rate with the rate given in [EP]. *As mentioned above the rates in* [EP] *are established under some additional assumptions on A. The examples given there are all strictly diagonally dominant M-matrices which are Toeplitz matrices. The matrix A above is the matrix* (2) *in* [EP].

If one considers the inverse $C = [c_{i,j}]$ of A, given in [EP], *the almost exact decay rate to the right along a row is* 0.2295. *The almost exact rate to the left along a row is, however,* 0.1878. *The estimates given in* [EP] *are* 0.2295 *for the right and* 0.1877 *to the left. A short calculation gives $\rho_3 = 0.2683$ and $\rho_4 = 0.2308$. Thus the greater restriction in the class of matrices for which decay rates are estimated yields here the better decay estimates.*

Theorem 3.2 generalizes the result for symmetric tridiagonal matrices by Concus, Golub, and Meurant [CGM]. In the following we specify the above result. First we will explain what happens if the assumptions $a_1 > b_1$ and $a_n > c_{n-1}$, etc., are not fulfilled. Then we will establish results for matrices which are not diagonally dominant.

In the following we will explain the connection of zero row and column sums of A and some pairwise constant u_i, v_i, x_i, and y_i. Let for $A = [a_{ij}] \in \mathbb{R}^{n,n}$ and $i = 1, \dots, n$

$$r_i = \sum_{j=1}^{n} a_{ij} \qquad c_i = \sum_{j=1}^{n} a_{j,i}.$$

Then we have the following.

THEOREM 3.3. *Let A be an irreducible nonsingular tridiagonal matrix. Then there exist indices $s, t \in \{0, 1, \dots, n+1\}$ such that*

$$\begin{aligned} r_i &= 0 \quad \textit{for} \ \ i = 1, \dots, s, \\ r_i &= 0 \quad \textit{for} \ \ i = n-t, \dots, n \end{aligned}$$

if and only if

$$\begin{aligned} u_i &= u_{i+1} \quad \textit{for} \ \ i = 1, \dots, s, \\ y_{i-1} &= y_i \quad \textit{for}\ i = n-t, \dots, n. \end{aligned}$$

Moreover, there exist indices $\tilde{s}, \tilde{t} \in \{0, 1, \dots, n+1\}$ such that

$$\begin{aligned} c_i &= 0 \quad \textit{for} \ \ i = 1, \dots, \tilde{s}, \\ c_i &= 0 \quad \textit{for} \ \ i = n-\tilde{t}, \dots, n \end{aligned}$$

if and only if

$$\begin{aligned} x_i &= x_{i+1} \quad \textit{for} \ \ i = 1, \dots, \tilde{s}, \\ v_{i-1} &= v_i \quad \textit{for}\ i = n-\tilde{t}, \dots, n. \end{aligned}$$

Proof. The proof follows immediately from similar formulas for the u_i, v_i, x_i, and y_i as derived in Lemma 3.1 and in the proof of Theorem 3.2. □

Note that in Theorem 3.3 we do not assume A to be diagonally dominant or positive definite or to be a Z-matrix. For symmetric matrices Theorem 3.3 says that the first and last pairwise constant u_i and v_i gives zero row sums for the first and last rows in A and vice versa. Moreover, if $r_i = 0$ for $i = 1, \dots, s$ and $r_i = 0$ for $i = n-t, \dots, n$, then the inverse of A can be partitioned as

$$A^{-1} = C = \begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix}, \tag{3.3}$$

where $C_{11} \in \mathbb{R}^{s,s}$ and $C_{33} \in \mathbb{R}^{t+1,t+1}$. Here C_{11} is a flipped type D matrix while C_{33} is a type D matrix. C_{31} and C_{13} are flat matrices, i.e., all entries of these blocks are equal. Hence, there can be no strict decay away from the diagonal in these blocks. Moreover, the rows of C_{12} and the columns of C_{23} are flat. This structure is illustrated in the next example.

EXAMPLE 3.3.

$$A = \begin{bmatrix} 2 & -2 & 0 & 0 & 0 & 0 \\ -2 & 2 & 1 & 0 & 0 & 0 \\ 0 & 1 & 4 & -2 & 0 & 0 \\ 0 & 0 & -2 & 2 & -1 & 0 \\ 0 & 0 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix}.$$

Here we have $s = t = 1$. Thus all blocks $C_{i,j}$ are 2×2 blocks and the blocks $C_{1,3}$ and $C_{3,1}$ are flat:

$$10A^{-1} = \begin{bmatrix} -11 & -16 & 10 & 12 & 4 & 4 \\ -16 & -16 & 10 & 12 & 4 & 4 \\ 10 & 10 & 0 & 0 & 0 & 0 \\ 12 & 12 & 0 & 6 & 2 & 2 \\ 4 & 4 & 0 & 2 & 4 & 4 \\ 4 & 4 & 0 & 2 & 4 & 14 \end{bmatrix}.$$

Combining Theorem 3.3 with Theorem 3.2 we get the following.

COROLLARY 3.4. *Let A be an irreducible tridiagonal M-matrix. If $r_i = 0$ for $i = 1, \ldots, s$ and $r_i = 0$ for $i = n - t, \ldots, n$ for $s, t \in \{0, 1, \ldots, n-1\}$, then the sequences $\{u_i\}$ and $\{y_i\}$ satisfy*

$$\begin{aligned} u_i &= u_{i+1} \text{ for } i = 1, \ldots, s, \\ u_i &< u_{i+1} \text{ else}, \\ y_{i-1} &= y_i \text{ for } i = n - t, \ldots, n, \\ y_{i-1} &> y_i \text{ else}. \end{aligned}$$

If $c_i = 0$ for $i = 1, \ldots, \tilde{s}$ and $c_i = 0$ for $i = n - \tilde{t}, \ldots, n$ for $\tilde{s}, \tilde{t} \in \{0, 1, \ldots, n-1\}$. Then the sequences $\{x_i\}$ and $\{v_i\}$ satisfy

$$\begin{aligned} x_i &= x_{i+1} \text{ for } i = 1, \ldots, \tilde{s}, \\ x_i &< x_{i+1} \text{ else}, \\ v_{i-1} &= v_i \text{ for } i = n - \tilde{t}, \ldots, n, \\ v_{i-1} &> v_i \text{ else}. \end{aligned}$$

EXAMPLE 3.4. *Consider the one-dimensional Laplacian with Dirichlet and Neumann boundary conditions:*

$$A = \begin{bmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & -1 & 2 & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 2 & -1 \\ & & & & -1 & 1 \end{bmatrix}.$$

Here $s = 0$ but $t = n - 1$. So we obtain a type D matrix as A^{-1}.

$$A^{-1} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 & 1 \\ 1 & 2 & 2 & \cdots & 2 & 2 \\ 1 & 2 & 3 & \cdots & 3 & 3 \\ \vdots & \vdots & \vdots & \ddots & & \\ 1 & 2 & 3 & & n-1 & n-1 \\ 1 & 2 & 3 & & n-1 & n \end{bmatrix}.$$

Hence, the entries do not decay away from the diagonal to the right along a row but to the left. This happens since the sequence $\{u_i\}$ is strictly increasing while the sequence $\{v_i\}$ is vanishing.

For M-matrices or symmetric positive definite tridiagonal or banded matrices, which are not diagonally dominant, the elements of A^{-1} do not decrease in general along a row or column. This can be seen in the following example.

Example 3.5.

$$A = \begin{bmatrix} 8 & -4 & 0 & 0 \\ -4 & 5 & -1 & 0 \\ 0 & -1 & 1 & -1 \\ 0 & 0 & -1 & 4 \end{bmatrix} \quad 10A^{-1} = \begin{bmatrix} 2.75 & 3 & 4 & 1 \\ 3 & 6 & 8 & 2 \\ 4 & 8 & 24 & 6 \\ 1 & 2 & 6 & 4 \end{bmatrix}.$$

However, we will establish a decay result for tridiagonal M-matrices and symmetric positive definite matrices in the following. These results are based on the characterizations of these matrices given in Theorem 2.5 and Corollary 2.7.

Theorem 3.5. *Let $A \in \mathbb{R}^{n,n}$ be a nonsingular irreducible tridiagonal matrix. Let $A^{-1} = C = [c_{ij}]$. Then for $k > 0$*

$$c_{i,i+k} = \left(\prod_{j=1}^{k} \alpha_{i+j}^{\frac{1}{2}} \right) (c_{i,i} c_{i+k,i+k})^{\frac{1}{2}}, \tag{3.4}$$

$$c_{i+k,i} = \left(\prod_{j=1}^{k} \beta_{i+j}^{\frac{1}{2}} \right) (c_{i,i} c_{i+k,i+k})^{\frac{1}{2}}, \tag{3.5}$$

where the α_i and β_i are as in (2.10).

Proof. Obviously, $c_{ii}, c_{i+k,i+k}$ and $c_{i,i+k}$ are given by

$$c_{ii} = u_i v_i, \quad c_{i+k,i+k} = u_{i+k} v_{i+k}, \quad c_{i,i+k} = u_i v_{i+k}.$$

Moreover,

$$\frac{u_i}{v_i} = \left(\prod_{j=1}^{k} \alpha_{i+j} \right) \frac{u_{i+k}}{v_{i+k}}.$$

Hence,

$$\begin{aligned} c_{i,i+k} &= u_i v_{i+k} \\ &= (u_i v_i u_{i+k} v_{i+k})^{\frac{1}{2}} \left(\frac{u_i v_{i+k}}{v_i u_{i+k}} \right)^{\frac{1}{2}} \\ &= \left(\prod_{j=1}^{k} \alpha_{i+j}^{\frac{1}{2}} \right) (c_{i,i} c_{i+k,i+k})^{\frac{1}{2}}. \end{aligned}$$

Similarly, we obtain (3.5). □

In general one cannot say anything about the α_i and β_i in (3.4) and (3.5). But for M-matrices and for symmetric positive definite matrices we have the following decay results.

Corollary 3.6. *Let $A \in \mathbb{R}^{n,n}$ be an irreducible tridiagonal M-matrix. Then for $A^{-1} = C = [c_{ij}]$ we have*

$$c_{i,i+k} c_{i+k,i} = \left(\prod_{j=1}^{k} (\alpha_{i+j} \beta_{i+j})^{\frac{1}{2}} \right) (c_{i,i} c_{i+k,i+k}), \tag{3.6}$$

where $\alpha_i\beta_i < 1$ for all i.

Proof. The proof follows immediately from Theorem 2.5 and Theorem 3.5. □

COROLLARY 3.7. *Let $A \in \mathbb{R}^{n,n}$ be an irreducible tridiagonal matrix which is symmetric positive definite. Then for $A^{-1} = C = [c_{ij}]$ we have*

$$c_{i,i+k} = \left(\prod_{j=1}^{k} \alpha_{i+j}^{\frac{1}{2}}\right) (c_{i,i}c_{i+k,i+k})^{\frac{1}{2}}, \tag{3.7}$$

where $\alpha_i < 1$ for all i.

Proof. With Theorem 2.7 we have for a symmetric positive definite tridiagonal matrix A that there exists a diagonal matrix D such that $DA^{-1}D$ is given by (2.11) with

$$0 < \frac{u_1}{v_1} < \frac{u_2}{v_2} < \cdots < \frac{u_n}{v_n}.$$

Thus we have $\alpha_i < 1$ for all i. Since (3.4) of Theorem 3.5 is independent of multiplying A from the left and right with the same diagonal matrix, we obtain (3.7). □

Note that Corollary 3.7 also includes the symmetric M-matrices. Since for M-matrices $\alpha_i\beta_i < 1$ for all i and $\alpha_i < 1$ for symmetric positive definite matrices, (3.6) and (3.7) give a sharp decay result for the entries of the inverse of tridiagonal matrices. Moreover, this result can be proved easily. The decay rates are given in terms of A^{-1}. However, the next lemma and the next theorem will show the relation of the α_i and β_i to some values determined directly from A.

For $k = 1, \ldots, n-1$ partition A as

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \tag{3.8}$$

where $A_{11} \in \mathbb{R}^{k,k}$ and $A_{22} \in \mathbb{R}^{n-k,n-k}$, and split A into $A = D_k - N_k$, where D_k is the block diagonal of A. If D_k is nonsingular we define

$$\rho_k = \rho(D_k^{-1}N_k). \tag{3.9}$$

We then have the following lemma.

LEMMA 3.8. *Let $A \in \mathbb{R}^{n,n}$ be tridiagonal and nonsingular. For $k = 1, \ldots, n-1$ let $A = D_k - N_k$ be as in (3.8). Moreover, for $A^{-1} = C = [c_{ij}]$ let*

$$\tilde{D}_k = \begin{bmatrix} c_{k,k} & 0 \\ 0 & c_{k+1,k+1} \end{bmatrix}, \quad \tilde{N}_k = -\begin{bmatrix} 0 & c_{k,k+1} \\ c_{k+1,k} & 0 \end{bmatrix}. \tag{3.10}$$

If D_k and $\tilde{D}_k$ are nonsingular, then with $J = [j_{st}] := D_k^{-1}N_k$ and

$$\hat{J}_k = \begin{bmatrix} 0 & j_{k,k+1} \\ j_{k+1,k} & 0 \end{bmatrix}$$

we have

$$\hat{J}_k = -(\tilde{D}_k^{-1}\tilde{N}_k)^T \tag{3.11}$$

and

$$\sigma(D_k^{-1}N_k) = \sigma(\hat{J}_k) \cup \{0\} = \sigma(\tilde{D}_k^{-1}\tilde{N}_k) \cup \{0\}. \tag{3.12}$$

Here $\sigma(T)$ denotes the spectrum of the matrix T.

Proof. We immediately get for $\hat{J}_k$

$$\hat{J}_k = \begin{bmatrix} 0 & -(A_{11}^{-1})_{kk}a_{k,k+1} \\ -(A_{22}^{-1})_{11}a_{k+1,k} & 0 \end{bmatrix}.$$

Now consider A partitioned as in (3.8). A^{-1} is given by

$$A^{-1} = \begin{bmatrix} (A/A_{22})^{-1} & -A_{11}^{-1}A_{12}(A/A_{11})^{-1} \\ -(A/A_{11})^{-1}A_{21}A_{11}^{-1} & (A/A_{11})^{-1} \end{bmatrix}.$$

Here (A/A_{11}) denotes the Schur complement of A, i.e., $(A/A_{11}) = A_{22} - A_{12}A_{11}^{-1}A_{21}$. We have

$$-A_{21}A_{11}^{-1} = -a_{k+1,k}\left[(A_{11})_k^{-T}, 0, \dots, 0\right]^T,$$

where $(A_{11})_k^{-1}$ is the last row of A_{11}^{-1}. Hence,

$$c_{k+1,k} = \left[-(A/A_{11})^{-1}A_{21}A_{11}^{-1}\right]_{1,k} = -(A/A_{11})_{1,1}^{-1} * (A_{11}^{-1})_{k,k} * a_{k,k+1}.$$

Thus

$$-\frac{c_{k+1,k}}{c_{k+1,k+1}} = (A_{11}^{-1})_{k,k}a_{k,k+1} = -j_{k,k+1}.$$

Similarly, we show that

$$-\frac{c_{k,k+1}}{c_{k,k}} = (A_{22}^{-1})_{1,1}a_{k+1,k} = -j_{k+1,k}.$$

Thus

$$\hat{J}_k = -(\tilde{D}^{-1}\tilde{N}_k)^T.$$

Equation (3.12) follows from (3.11) and the special structure of $J = D_k^{-1}N_k$. □

We then obtain the following theorem.

THEOREM 3.9. *Let $A \in \mathbb{R}^{n,n}$ be an irreducible tridiagonal M-matrix and let $A^{-1} = C = [c_{ij}]$. For $s = 1, \dots, n-1$ let ρ_s be as in (3.9). Then*

$$c_{i,i+k}c_{i+k,i} = \left(\prod_{l=0}^{k-1} \rho_{i+l}^2\right)(c_{i,i}c_{i+k,i+k}).$$

Proof. For all s (3.10) gives

$$\tilde{D}_s = \begin{bmatrix} u_sv_s & 0 \\ 0 & u_{s+1}v_{s+1} \end{bmatrix} = \begin{bmatrix} x_sy_s & 0 \\ 0 & x_{s+1}y_{s+1} \end{bmatrix},$$

$$\tilde{N}_k = -\begin{bmatrix} 0 & u_sv_{s+1} \\ x_sy_{s+1} & 0 \end{bmatrix};$$

thus

$$(\rho(\tilde{D}_s^{-1}\tilde{N}_k))^4 = \frac{u_sv_{s+1}x_sy_{s+1}}{u_sv_su_{s+1}v_{s+1}}\frac{x_sy_{s+1}u_sv_{s+1}}{x_sy_sx_{s+1}y_{s+1}} = \alpha_s\beta_s.$$

Hence with Lemma 3.8 we obtain

$$(\alpha_s\beta_s)^{\frac{1}{2}} = \rho_s^2. \qquad \square$$

Moreover, for symmetric positive definite matrices we obtain the following corollary.

Corollary 3.10. *Let $A \in \mathbb{R}^{n,n}$ be tridiagonal symmetric positive definite and let $A^{-1} = C = [c_{ij}]$. Then*

$$c_{i,i+k} = \left(\prod_{l=0}^{k-1} \rho_{i+l}\right) (c_{i,i}c_{i+k,i+k})^{\frac{1}{2}}.$$

A similar result is proved by Vassilevski [Vas] (see also [Ax, p. 368]). Vassilevski proved for tridiagonal symmetric positive definite matrices A and their inverses $C = [c_{ij}]$ that

$$c_{i,i+k} = \left(\prod_{l=0}^{k-1} \gamma_{i+l}\right) (c_{i,i}c_{i+k,i+k})^{\frac{1}{2}}, \tag{3.13}$$

where

$$\gamma_s = \sup_{v\in\mathbb{R}^{s,s}, w\in\mathbb{R}^{n-s,n-s}} \frac{v^T A_{12} w}{(v^T A_{11} v w^T A_{22} w)^{\frac{1}{2}}} \tag{3.14}$$

and A is partitioned as in (3.8) for all s. The constants γ_s are known as the Cauchy–Bunyakowski–Schwarz constants. Since A is positive definite, $\gamma_s < 1$. Originally in [Vas] the result (3.13) is obtained from a more general result for symmetric positive definite block tridiagonal matrices. There norms of the blocks are compared and the equality in (3.13) becomes an inequality ($\leq$).

The proof given in [Vas] is very long and technical. However, it is easy to prove that the ρ_s in Corollary 3.10 and the γ_s in (3.14) are the same. Thus, our approach for the nonsymmetric case gives even for the symmetric matrices a much simpler proof of Vassilevski's result.

Moreover, Theorem 3.10 and Lemma 3.8 give another way to compute or estimate the Cauchy–Bunyakowski–Schwarz constants.

Using comparison theorems for regular splittings we obtain from Theorem 3.10 and Theorem 3.9 the following corollary.

Corollary 3.11. *Let $A \in \mathbb{R}^{n,n}$ be tridiagonal and irreducible. Let $A^{-1} = C = [c_{ij}]$. Let $A = D - N$, where $D = diag(A)$, and let $\rho = \rho(D^{-1}N)$. If A is an M-matrix, then*

$$c_{i,i+k}c_{i+k,i} \leq \rho^{2k} c_{i,i}c_{i+k,i+k}. \tag{3.15}$$

If A is symmetric positive definite, then

$$c_{i,i+k} \leq \rho^k (c_{i,i}c_{i+k,i+k})^{\frac{1}{2}}. \tag{3.16}$$

Proof. The splittings $A = D_k - N_k$ of (3.8) are regular splittings, i.e., D_k^{-1} and N_k are nonnegative matrices. The same holds for the splitting $A = D - N$. Moreover, we have

$$N_k \leq N \quad \text{for all } k,$$

With Varga's comparison theorem for regular splittings [Var, p. 90] we obtain

$$\gamma_k = \rho(D_k^{-1}N_k) \leq \rho.$$

Since (3.16) is independent of scaling we obtain the result easily for symmetric positive definite matrices. □

Note that in both cases $\rho < 1$. Thus the spectral radius of the Jacobi iteration matrix is an upper bound for the decay of the entries of the inverse of a tridiagonal M-matrix or symmetric positive definite matrix.

EXAMPLE 3.6. *Consider the matrix A with*

$$A = \begin{bmatrix} 4 & -2 & 0 & 0 & 0 \\ -1.5 & 3 & -1.5 & 0 & 0 \\ 0 & -4 & 8 & -4 & 0 \\ 0 & 0 & -2.5 & 5 & -2.5 \\ 0 & 0 & 0 & -6 & 5 \end{bmatrix},$$

$$A^{-1} = C = \begin{bmatrix} 1 & 2 & 1 & 2 & 1 \\ 1.5 & 4 & 2 & 4 & 2 \\ 2 & 5.3333 & 3 & 6 & 3 \\ 2.5 & 6.6667 & 3.75 & 8 & 4 \\ 3 & 8 & 4.5 & 9.6 & 5 \end{bmatrix}.$$

The vectors u, v, x, y are as follows:

$$u = [1, 2, 3, 4, 5], \quad v = [1, 2, 1, 2, 1],$$
$$x = [1, 2.6667, 1.5, 3.2, 1.6667], \quad y = [1, 1.5, 2, 2.5, 3].$$

For the vectors $\alpha = [\alpha_i]$ and $\beta = [\beta_i]$ we get

$$\alpha = [1, 0.3333, 0.6666, 0.4], \quad \beta = [0.5625, 2.3704, 0.5859, 2.3040].$$

Thus one can verify Theorem 3.5. *To illustrate Corollary* 3.11 *we compare the matrices $\tilde{C} = C \circ C^T$ with the matrix built by the right-hand sides of* (3.15), *where $\rho = 0.99$:*

$$\tilde{C} = \begin{bmatrix} 1.0000 & 3.0000 & 2.0000 & 5.0000 & 3.0000 \\ 3.0000 & 16.0000 & 10.6667 & 26.6667 & 16.0000 \\ 2.0000 & 10.6667 & 9.0000 & 22.5000 & 13.5000 \\ 5.0000 & 26.6667 & 22.5000 & 64.0000 & 38.4000 \\ 3.0000 & 16.0000 & 13.5000 & 38.4000 & 25.0000 \end{bmatrix}$$
$$\leq \begin{bmatrix} 1.0000 & 3.9204 & 2.8818 & 7.5318 & 4.6137 \\ 3.9204 & 16.0000 & 11.7612 & 30.7391 & 18.8296 \\ 2.8818 & 11.7612 & 9.0000 & 23.5224 & 14.4089 \\ 7.5318 & 30.7391 & 23.5224 & 64.0000 & 39.2040 \\ 4.6137 & 18.8296 & 14.4089 & 39.2040 & 25.0000 \end{bmatrix}.$$

The decay rate of Corollary 3.11 *seems to be too pessimistic. However, the bounds on the entries reflect the oscillation of the entries of C and $\tilde{C}$ along a row.*

We have seen that the spectral radius of the Jacobi iteration matrix is an upper bound for the decay of the entries of the inverse of a tridiagonal M-matrix or symmetric positive definite matrix. In the following we will see that this is also true for banded M-matrices. A matrix $A = [a_{i,j}]$ is called a $2p+1$ banded matrix if $a_{i,j} = 0$ for $|i-j| > p$.

For nonnegative matrices the spectral radius is an eigenvalue and the related eigenvector is positive. So, if we split the M-matrix A into

$$A = D - N, \qquad \text{where } D = diag(A) \text{ and } N = D - A,$$

then $D^{-1}N$ is a nonnegative matrix. Thus, there exists a positive vector u with $D^{-1}Nu = \rho(D^{-1}N)u$ and a nonsingular diagonal matrix $\Delta = diag(\delta_i)$ with

$$\Delta e = u \quad \text{such that} \quad \Delta^{-1}D^{-1}N\Delta e = \rho(D^{-1}N)e, \tag{3.17}$$

where e is the vector of all ones. We then have the theorem below.

THEOREM 3.12. *Let A be a $2p+1$ banded M-matrix. Let $A^{-1} = C = [c_{s,t}]$. Then for any s, t with $s \in \{(i-1)p+2, \dots, ip+1\}$ and $t \in \{(j-1)p+2, \dots, jp+1\}$ ($i = 1$ if $s = 1$, $j = 1$ if $t = 1$) with $i \neq j$*

$$\delta_s^{-1}c_{s,t}\delta_t \leq \rho^{|i-j|}c_{t,t},$$
$$\delta_s^{-1}c_{s,t}\delta_t \leq \rho^{|i-j|}c_{s,s};$$

here $\rho = \rho(D^{-1}N)$ with $D = diag(A)$ and $N = D - A$.

Proof. First we split $\Delta^{-1}A\Delta$ into $\Delta^{-1}A\Delta = \Delta^{-1}D\Delta - \Delta^{-1}N\Delta$. We obtain

$$\Delta^{-1}D^{-1}N\Delta e = \rho(D^{-1}N)e. \tag{3.18}$$

We first assume that $s > t$. To find the tth column of $\Delta^{-1}C\Delta$ we have to solve $\Delta^{-1}A\Delta x = e_t$, where e_t is the zero vector except the tth entry which is 1. To do so we consider with $J := \Delta^{-1}D^{-1}N\Delta$

$$x^{(k)} = Jx^{(k-1)} + \Delta^{-1}D^{-1}\Delta e_j, \qquad x^{(0)} = 0. \tag{3.19}$$

If we define $\varepsilon^{(k)} = x - x^{(k)}$ we get

$$\varepsilon^{(k)} = J\varepsilon^{(k-1)} = J^k\varepsilon^{(0)} = J^k x.$$

Thus

$$||\varepsilon^{(k)}||_\infty \leq ||J^k||_\infty ||x||_\infty \leq ||J||_\infty^k ||x||_\infty = \rho(J)^k c_{tt}.$$

The last equality follows from (3.18) and $||x||_\infty = c_{tt}$ (see [FP]). With (3.19) we obtain for $p \neq 1$

$$x_l^{(i-j)} = 0 \quad \text{for } l \geq (i-1)p+2$$

and for $p = 1$

$$x_l^{(i-j)} = 0 \quad \text{for } l \geq i.$$

Now let

$$\varepsilon^{(k)} = \begin{bmatrix} \tilde{\varepsilon}^{(k)} \\ \hat{\varepsilon}^{(k)} \end{bmatrix},$$

where $\tilde{\varepsilon}^{(k)} \in \mathbb{R}^{(i-1)p+2}$. Similarly, we partition $x^{(k)}$ and x. Then

$$||\varepsilon^{(i-j)}||_\infty \geq ||\hat{\varepsilon}^{(i-j)}||_\infty = ||\hat{x} - \hat{x}^{(i-j)}||_\infty \geq ||\hat{x}||_\infty - ||\hat{x}^{(i-j)}||_\infty = ||\hat{x}||_\infty .$$

Thus

$$\delta_s^{-1} c_{s,t} \delta_t \leq ||\hat{x}||_\infty \leq \rho(J)^{(i-j)} c_{t,t}.$$

Similarly, we obtain by solving $x^T \Delta^{-1} A \Delta = e_s$

$$\delta_s^{-1} c_{s,t} \delta_t \leq \rho(J)^{(i-j)} c_{s,s}.$$

Similarly, we prove the case $s < t$. □

Theorem 3.12 can be extended easily to H-matrices A, i.e., matrices for which the comparison matrix $\mathcal{M}(A)$ with $(\mathcal{M}(A))_{ii} = |a_{ii}|, (\mathcal{M}(A))_{i,j} = -|a_{i,j}|$ for $i \neq j$, is an M-matrix. The decay rate is then the spectral radius of the Jacobi matrix of $\mathcal{M}(A)$. Moreover, Theorem 3.12 also can be formulated for sparse matrices, not only banded matrices, using the notation used by Meurant in [Meu].

We immediately get from Theorem 3.12 the following corollary.

COROLLARY 3.13. *Let A be a $2p+1$ banded M-matrix. Let $A^{-1} = C = [c_{s,t}]$. Then for any s, t with $s \in \{(i-1)p+2, \ldots, ip+1\}$ and $t \in \{(j-1)p+2, \ldots, jp+1\}$ ($i = 1$ if $s = 1$, $j = 1$ if $t = 1$) with $i \neq j$*

$$\begin{aligned} c_{s,t} c_{t,s} &\leq \rho^{2|i-j|} (c_{s,s} c_{t,t}), \\ c_{s,t} c_{t,s} &\leq \rho^{2|i-j|} c_{s,s}^2, \\ c_{s,t} c_{t,s} &\leq \rho^{2|i-j|} c_{t,t}^2. \end{aligned}$$

COROLLARY 3.14. *Let A be a $2p+1$ banded symmetric M-matrix. Let $A^{-1} = C = [c_{s,t}]$. Then for any s, t with $s \in \{(i-1)p+2, \ldots, ip+1\}$ and $t \in \{(j-1)p+2, \ldots, jp+1\}$ ($i = 1$ if $s = 1$, $j = 1$ if $t = 1$) with $i \neq j$*

$$\begin{aligned} c_{s,t} &\leq \rho^{|i-j|} (c_{s,s} c_{t,t})^{\frac{1}{2}}, \\ c_{s,t} &\leq \rho^{|i-j|} c_{s,s}, \\ c_{s,t} &\leq \rho^{|i-j|} c_{t,t}. \end{aligned}$$

The advantage of Corollaries 3.13 and 3.14 compared with a theorem by Meurant's [Meu, Theorem 4.13] is that we just need ρ and diagonal entries of A^{-1} to estimate the decay.

Acknowledgments. The author wishes to thank Ludwig Elsner and Shmuel Friedland as well as the referees for helpful comments.

REFERENCES

[As] E. ASPLUND, *Inverse of matrices $\{a_{i,j}\}$ which satisfy $a_{ij} =$ for $j > i+p$*, Math. Scand., 7 (1959), pp. 57–60.

[Ax] O. AXELSSON, *Iterative Solution Methods*, Cambridge University Press, London, 1994.

[Ba] W. W. BARRETT, *A theorem on inverses of tridiagonal matrices*, Linear Algebra Appl., 27 (1979), pp. 211–217.

[Be] R. BEVILACQUA, *Structural and computational properties of band matrices*, in *Complexity of Structured Computational Problems*, R. Bevilacqua, D. Bini, M. Capovani, G. Capriz, B. Codenotti, M. Leoncini, G. Resta, and P. Zellini, eds., Appl. Math. Monographs, Consiglio Nazionale delle Ricerche, Giardini Editori e Stampatori in Pisa, 1991, pp. 131–188.

[BC] R. Bevilacqua and M. Capovani, *Prorietà delle matrici a banda ad elementi e a blocchi,* Boll. Un. Mat. Ital., 13B (1976), pp. 844–861.

[BD] J. Baranger and M. Duc-Jacquet, *Matrices tridiagonales symétriques et matrices factorisables*, RIRO Ser. R-3, 5 (1971), pp. 61–66.

[C1] M. Capovani, *Sulla determinazione della inversa delle matrici tridiagonali a blocchi,* Calcolo, 7 (1970), pp. 295–303.

[C2] M. Capovani, *Su alcune proprietà delle matrici tridiagonali e pentadiagonali,* Calcolo, 8 (1971), pp. 149–159.

[CGM] C. Concus, G. H. Golub, and G. Meurant, *Block preconditioning for the conjugate gradient method*, SIAM J. Sci. Statist. Comp., 6 (1985), pp. 220–252.

[D] S. Demko, *Inverses of band matrices and local convergence of spline projections*, SIAM J. Numer. Anal., 14 (1977), pp. 616–619.

[DMS] S. Demko, W. F. Moss, P. W. Smith, *Decay rates for inverses of band matrices*, Math. Comp., 43 (1984), pp. 491–499.

[EP] V. Eijkhout and B. Polman, *Decay rates of inverses of banded M-matrices that are near to Toeplitz matrices,* Linear Algebra Appl., 109 (1988), pp. 247–277.

[FP] M. Fiedler and V. Pták, *Diagonally dominant matrices*, Czechoslovak Math. J., 47 (1967), pp. 420–433.

[GK1] F. R. Gantmacher and M. G. Krein, *Sur les matrices complètement non négatives et oscillatoires,* Compositio Math., 4 (1937), pp. 445–470.

[GK2] F. R. Gantmacher and M. G. Krein, *Oszillationsmatrizen, Oszillationskerne und kleine Schwingungen mechanischer Systeme*, Akademie Verlag, Berlin, 1960 (in Russian); first edition, 1941.

[K] S. Karlin, *Total Positivity*, Stanford University Press, Stanford, CA, 1968.

[Mar] T. L. Markham, *Nonnegative matrices whose inverses are M-matrices*, Proc. Amer. Math. Soc., 36 (1972), pp. 326–330.

[Meu] G. Meurant, *A Review on the inverse of symmetric tridiagonal and block tridiagonal matrices*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 707–728.

[MNNST] J. J. McDonald, R. Nabben, M. Neumann, H. Schneider, and M. Tsatsomeros, *Inverse tridiagonal Z-matrices,* Linear and Multilinear Algebra, to appear.

[R] P. Rózsa, *On the inverse of band matrices*, Integral Equations Operator Theory, 10 (1987), pp. 82–95.

[Var] R. S. Varga, *Matrix Iterative Analysis*, Prentice–Hall, Englewood Cliffs, NJ, 1962.

[Vas] P. S. Vassilevski, *On some ways of approximating inverses of banded matrices in connection with deriving preconditioners based on incomplete block factorizations*, Computing, 43 (1990), pp. 277–296.

SIAM J. Matrix Anal. Appl.
Vol. 20, No. 4, pp. 839–859

AN EFFICIENT ALGORITHM FOR A BOUNDED ERRORS-IN-VARIABLES MODEL[*]

S. CHANDRASEKARAN[†], G. H. GOLUB[‡], M. GU[§], AND A. H. SAYED[¶]

Abstract. We pose and solve a parameter estimation problem in the presence of bounded data uncertainties. The problem involves a minimization step and admits a closed form solution in terms of the positive root of a secular equation.

Key words. least-squares estimation, total least-squares, modeling errors, secular equation

AMS subject classifications. 15A06, 65F05, 65F10, 65F35, 65K10, 93C41, 93E10, 93E24

PII. S0895479896304678

1. Introduction.

Parameter estimation in the presence of data uncertainties is a problem of considerable practical importance, and many estimators have been proposed in the literature with the intent of handling modeling errors and measurement noise. Among the most notable is the total least-squares (TLS) method [1, 2, 3, 4], also known as orthogonal regression or errors-in-variables method in statistics and system identification [5]. In contrast to the standard least-squares problem, the TLS formulation allows for errors in the data matrix. Its performance may degrade in some situations where the effect of noise and uncertainties can be unnecessarily over-emphasized. This may lead to overly conservative results.

Assume $A \in \mathbf{R}^{m \times n}$ is a given full rank matrix with $m \geq n$, $b \in \mathbf{R}^m$ is a given vector, and consider the problem of solving the inconsistent linear system $A\hat{x} \approx b$ in the least-squares sense. The TLS solution assumes data uncertainties in A and proceeds to correct A and b by replacing them by their projections, $\hat{A}$ and $\hat{b}$, onto a specific subspace, and by solving the consistent linear system of equations $\hat{A}\hat{x} = \hat{b}$. The spectral norm of the correction $(A - \hat{A})$ in the TLS solution is bounded by the smallest singular value of $\begin{bmatrix} A & b \end{bmatrix}$. While this norm might be small for vectors b that are close enough to the range space of A, it need not always be so. In other words, the TLS solution may lead to situations in which the correction in A is unnecessarily large. Consider, for example, a situation in which the uncertainties in A are very small and, say, A is almost known exactly. Assume further that b is far from the range space of A. In this case, it is not difficult to visualize that the TLS solution will need to modify (A, b) into $(\hat{A}, \hat{b})$ and may therefore end up with an overly corrected approximant for A, despite the fact that A is almost exact.

These facts motivate us to introduce a new parameter estimation formulation with a bound on the size of the allowable correction to A. The solution of the new

[*] Received by the editors June 3, 1996; accepted for publication (in revised form) by S. Van Huffel July 22, 1998; published electronically July 9, 1999. The work of the first and fourth authors was supported in part by the National Science Foundation under award numbers MIP-9796147, CCR-9732376, and CCR-9734290, respectively.

http://www.siam.org/journals/simax/20-4/30467.html

[†] Department of Electrical and Computer Engineering, University of California, Santa Barbara, CA 93106 (shiv@ece.ucsb.edu).

[‡] Department of Computer Science, Stanford University, Stanford, CA 94305 (golub@sccon.stanford.edu).

[§] Department of Mathematics, University of California, Los Angeles, CA 90095 (mgu@math.ucla.edu).

[¶] Department of Electrical Engineering, University of California, Los Angeles, CA 90095 (sayed@ee.ucla.edu).

formulation turns out to involve the minimization of a cost function in an "indefinite" metric, in a way that is similar to more recent works on robust (or H^∞) estimation and filtering (e.g., [6, 7, 8, 9]). However, the cost function considered in our work is more complex and, contrary to robust estimation where no prior bounds are imposed on the size of the disturbances, the problem of this paper shows how to solve the resulting optimization problem in the presence of such constraints. A "closed" form solution to the new optimization problem is obtained in terms of the positive root of a secular equation.

The solution method proposed in this paper proceeds by first providing a geometric interpretation of the new optimization problem, followed by an algebraic derivation that establishes that the optimal solution can in fact be obtained by solving a related "indefinite" regularized problem. The regression parameter of the regularization step is further shown to be obtained from the positive root of a secular equation. The solution involves an SVD step and its computational complexity amounts to $O(mn^2+n^3)$, where n is the smaller matrix dimension. A summary of the problem and its solution is provided in section 4.7.

2. Problem statement.

Let $A \in \mathbf{R}^{m\times n}$ be a given matrix with $m \geq n$ and let $b \in \mathbf{R}^m$ be a given nonzero vector, which are assumed to be linearly related via an unknown vector of parameters $x \in \mathbf{R}^n$,

$$b = Ax + v \; . \tag{2.1}$$

The vector $v \in \mathbf{R}^m$ explains the mismatch between Ax and the given vector (or observation) b.

We assume that the "true" coefficient matrix is $A + \delta A$ and that we know only an upper bound on the perturbation δA:

$$\|\delta A\|_2 \;\leq\; \eta, \tag{2.2}$$

with η being known and where the notation $\|\cdot\|_2$ denotes the 2-induced norm of a matrix argument (i.e., its maximum singular value) or the Euclidean norm of a vector argument. We pose the following optimization problem.

PROBLEM 1. *Given $A \in \mathbf{R}^{m\times n}$, with $m \geq n$, $b \in \mathbf{R}^m$, and a nonnegative real number η, determine, if possible, an $\hat{x}$ that solves*

$$\min_{\hat{x}} \; \min\big\{\|\,(A+\delta A)\,\hat{x} - b\|_2 : \; \|\delta A\|_2 \leq \eta\big\}. \tag{2.3}$$

It turns out that the existence of a unique solution to this problem will require a fundamental condition on the data (A, b, η), which we describe in Lemma 3.1. When the condition is violated, the problem will become degenerate. In fact, such existence and uniqueness conditions also arise in other formulations of estimation problems (such as the TLS and H_∞ problems, which will be shown later to have some relation to the above optimization problem). In the H_∞ context, for instance, similar fundamental conditions arise, which when violated indicate that the problem does not have a meaningful solution (see, e.g., [6, 7, 8, 9]).

2.1. Intuition and explanation.

Before discussing the solution of the optimization problem we formulated above, it will be helpful to gain some intuition into its significance.

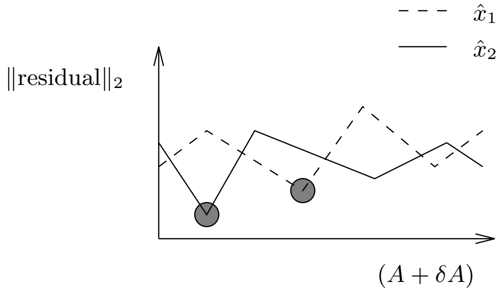


FIG. 2.1. *Two illustrative residual-norm curves.*

Intuitively, the above formulation corresponds to "choosing" a perturbation δA, within the bounded region, that would allow us to best predict the right-hand side b from the column span of $(A+\delta A)$. Comparing with the TLS formulation, we see that in TLS there is not an a priori bound on the size of the allowable perturbation δA. Still, the TLS solution finds the "smallest" δA (in a Frobenius norm sense) that would allow to estimate b from the column span of $(A+\delta A)$, viz., it solves the following problem [3]:

$$\min_{\delta A, \hat{x}} \left\| \begin{bmatrix} \delta A & (A+\delta A)\hat{x} - b \end{bmatrix} \right\|_{\mathrm{F}} .$$

Nevertheless, although small in a certain sense, the resulting correction δA need not satisfy an a priori bound on its size. The problem we formulated above explicitly incorporates a bound on the size of the allowable perturbations. We may further add that we have addressed a related estimation problem in the earlier work [10], where we have posed and solved a min-max optimization problem; it allows us to guarantee optimal performance in a worst-case scenario. Further discussion, from a geometric point of view, of this related problem and others, along with examples of applications in image processing, communications, and control, can be found in [11].

Returning to (2.3), we depict the situation in Figure 2.1. Any particular choice for $\hat{x}$ would lead to many residual norms,

$$\|\,(A+\delta A)\,\hat{x} - b\|_2 \ ,$$

one for each possible choice of δA. A second choice for $\hat{x}$ would lead to other residual norms, the minimum value of which need not be the same as the first choice. We want to choose an estimate $\hat{x}$ that minimizes the minimum possible residual norm.

2.2. A geometric interpretation. The optimization problem (2.3) admits an interesting geometric formulation that highlights some of the issues involved in its solution. We explain this by considering a scalar example. For the vector case, see [11].

Assume that we have a unit-norm vector b, $\|b\|_2 = 1$, and that A is simply a column vector, say a, with $\eta \neq 0$. Now problem (2.3) becomes

$$\min_{\hat{x}} \left(\min_{\|\delta a\|_2 \leq \eta} \|\,(a+\delta a)\,\hat{x} - b\|_2 \right) . \tag{2.4}$$

This situation is depicted in Figure 2.2. The vectors a and b are indicated in thick black lines. The vector a is shown in the horizontal direction and a circle of radius η around its vertex indicates the set of all possible vertices for $a+\delta a$.

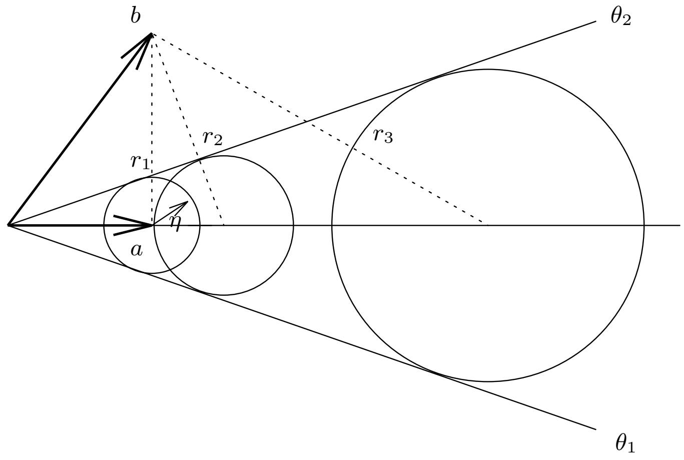


FIG. 2.2. *Geometric construction of the solution for a simple example.*

For any $\hat{x}$ that we pick, the set $\{(a+\delta a)\hat{x}\}$ describes a disc of center $a\hat{x}$ and radius $\eta\hat{x}$. This is indicated in the figure by the largest rightmost circle, which corresponds to a choice of a positive $\hat{x}$ that is larger than one. The vector in $\{(a+\delta a)\hat{x}\}$ that is the closest to b is the one obtained by drawing a line from b through the center of the rightmost circle. The intersection of this line with the circle defines a residual vector r_3 whose norm is the smallest among all possible residual vectors in the set $\{(a+\delta a)\hat{x}\}$.

Likewise, if we draw a line from b that passes through the vertex of a (which is the center of the leftmost circle), it will intersect the circle at a point that defines a residual vector r_1. This residual will have the smallest norm among all residuals that correspond to the particular choice $\hat{x} = 1$.

More generally, for any $\hat{x}$ that we pick, it will determine a circle and the corresponding smallest residual is obtained by finding the closest point on the circle to b. This is the point where the line that passes through b and the center of the circle intersects the circle on the side closer to b.

We need to pick an $\hat{x}$ that minimizes the smallest residual norm. The claim is that we need to proceed as follows: we drop a perpendicular from b to the upper tangent line denoted by θ_2. This perpendicular intersects the horizontal line in a point where we draw a new circle (the middle circle) that is tangent to both θ_1 and θ_2. This circle corresponds to a choice of $\hat{x}$ such that the closest point on it to b is the foot of the perpendicular from b to θ_2. The residual indicated by r_2 is the desired solution; it has the minimum norm among the smallest residuals.

3. An equivalent minimization problem.

To solve (2.3), we start by showing how to reduce it to an equivalent problem. For this purpose, we note that

$$\|\,(A+\delta A)\,\hat{x}-b\|_2 \;\geq\; |\;\|A\hat{x}-b\|_2 - \|\delta A\hat{x}\|_2\;|\,. \tag{3.1}$$

The lower bound on the right-hand side of the above inequality is a nonnegative quantity and, therefore, the least it can get is zero. This will in turn depend on how big or how small the value of $\|\delta A\|_2$ can be.

For example, if it happens that for all vectors $\hat{x}$ we always have

$$\eta\|\hat{x}\|_2 \;<\; \|A\hat{x}-b\|_2, \tag{3.2}$$

then we conclude, using the triangle inequality of norms, that

$$\|\delta A\hat{x}\|_2 \;\leq\; \|\delta A\|_2\|\hat{x}\|_2 \;\leq\; \eta\|\hat{x}\|_2 \;<\; \|A\hat{x}-b\|_2 \;.$$

It then follows from (3.1) that, under assumption (3.2), we obtain

$$\begin{aligned}\|\,(A+\delta A)\,\hat{x}-b\|_2 \;&\geq\; \|A\hat{x}-b\|_2 - \|\delta A\hat{x}\|_2\\ &\geq\; \|A\hat{x}-b\|_2 - \|\delta A\|_2\|\hat{x}\|_2\\ &\geq\; \|A\hat{x}-b\|_2 - \eta\|\hat{x}\|_2 \;.\end{aligned}$$

It turns out that condition (3.2) is the main (and only) case of interest in this paper, especially since we shall argue later that a degenerate problem arises when it is violated. For this reason, we shall proceed for now with our analysis under assumption (3.2) and shall postpone our discussion of what happens when it is violated until later in this section.

Now the lower bound in (3.1) is in fact achievable. That is, there exists a δA for which

$$\|\,(A+\delta A)\,\hat{x}-b)\|_2 \;=\; \|A\hat{x}-b\|_2 - \eta\|\hat{x}\|_2 \;.$$

To see that this is indeed the case, choose δA as the rank-1 matrix

$$\delta A^o = -\frac{(A\hat{x}-b)}{\|A\hat{x}-b\|_2}\;\frac{\hat{x}^T}{\|\hat{x}\|_2}\eta \;.$$

This leads to a vector $\delta A^o\hat{x}$ that is collinear with the vector $(A\hat{x}-b)$. (Note that $\hat{x}$ in the above definition for δA^o cannot be zero since otherwise (3.2) cannot be satisfied. Likewise, $A\hat{x}-b$ cannot be zero. Hence, δA^o is well defined.)

We are therefore reduced to the solution of the following optimization problem.

Problem 2. *Consider a matrix $A \in \mathbf{R}^{m\times n}$, with $m \geq n$, a vector $b \in \mathbf{R}^m$, and a nonnegative real number η, and assume that for all vectors $\hat{x}$ it holds that*

$$\eta\|\hat{x}\|_2 \;<\; \|A\hat{x}-b\|_2 \qquad (\textit{fundamental assumption}). \tag{3.3}$$

Determine, if possible, an $\hat{x}$ that solves

$$\min_{\hat{x}}\;(\|A\hat{x}-b\|_2 - \eta\|\hat{x}\|_2)\,. \tag{3.4}$$

3.1. Connections to TLS and $\mathbf{H}_\infty$-problems. Before solving problem (3.4), we elaborate on its connections with other formulations in the literature that also attempt, in one way or another, to take into consideration uncertainties and perturbations in the data.

First, cost functions similar to (3.4) but with squared distances, say

$$\min_{\hat{x}}\;\left(\|A\hat{x}-b\|_2^2 - \gamma\|\hat{x}\|_2^2\right) \tag{3.5}$$

for some $\gamma > 0$, often arise in the study of indefinite quadratic cost functions in robust or H^∞ estimation (see, e.g., the developments in [8, 9]). The major distinction between this cost and the one posed in (3.4) is that the latter involves *distance* terms and it will be shown to provide an automatic procedure for selecting a “regularization” factor that plays the role of γ in (3.5).

Likewise, the TLS problem seeks a matrix δA and a vector $\hat{x}$ that minimize the following Frobenius norm:

$$\min_{\delta A, \hat{x}} \left\| \begin{bmatrix} \delta A & (A+\delta A)\hat{x} - b \end{bmatrix} \right\|_{\mathrm{F}}^2 . \tag{3.6}$$

The solution of the above TLS problem is well known and is given by the following construction [4, p. 36]. Let $\{\sigma_1, \ldots, \sigma_n\}$ denote the singular values of A, with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$. Let also $\{\bar{\sigma}_1, \ldots, \bar{\sigma}_n, \bar{\sigma}_{n+1}\}$ denote the singular values of the extended matrix $\begin{bmatrix} A & b \end{bmatrix}$, with $\bar{\sigma}_i \geq 0$. If $\bar{\sigma}_{n+1} < \sigma_n$, then a unique solution $\hat{x}$ of (3.6) exists and is given by

$$\hat{x} = (A^T A - \bar{\sigma}_{n+1}^2 I)^{-1} A^T b. \tag{3.7}$$

For our purposes, it is more interesting to consider the following interpretation of the TLS solution (see, e.g., [9]). Note that the condition $\bar{\sigma}_{n+1} < \sigma_n$ ensures that $(A^T A - \bar{\sigma}_{n+1}^2 I)$ is a positive-definite matrix, since σ_n^2 is the smallest eigenvalue of $A^T A$. Therefore, we can regard (3.7) as the solution of the following optimization problem, with an indefinite cost function:

$$\min_{\hat{x}} \left(\|A\hat{x} - b\|_2^2 \; - \; \bar{\sigma}_{n+1}^2 \|\hat{x}\|_2^2 \right) .$$

This is a special form of (3.5) with a particular choice for γ. It again involves squared distances, while (3.4) involves *distance* terms and it will provide another choice of a γ-like parameter. In particular, compare (3.7) with the expression (4.4) derived further ahead for the solution of (3.4). We see that the new problem replaces $\bar{\sigma}_{n+1}^2$ with a new parameter α that will be obtained from the positive root of a secular (nonlinear) equation.

3.2. Significance of the fundamental assumption. We shall solve (3.4) in the next section. Here, we elaborate on the significance of the condition (3.3). Assume (3.3) is violated at some nonzero point $\hat{x}^{(1)}$, namely[1]

$$\eta \|\hat{x}^{(1)}\|_2 \;\geq\; \|A\hat{x}^{(1)} - b\|_2, \tag{3.8}$$

and define the perturbation

$$\delta A^{(1)} = (A\hat{x}^{(1)} - b) \frac{(\hat{x}^{(1)})^T}{\|\hat{x}^{(1)}\|_2^2}. \tag{3.9}$$

It is clear that $\delta A^{(1)}$ is a valid perturbation since, in view of (3.8), we have $\|\delta A^{(1)}\|_2 \leq \eta$. But this particular perturbation leads to

$$\begin{aligned} \left\| \left(A + \delta A^{(1)} \right) \hat{x}^{(1)} - b \right\|_2 &\geq \left| \; \|A\hat{x}^{(1)} - b\|_2 - \|\delta A^{(1)} \hat{x}^{(1)}\|_2 \; \right| \\ &\geq \|A\hat{x}^{(1)} - b\|_2 - \|A\hat{x}^{(1)} - b\|_2 = 0. \end{aligned}$$

That is, the lower limit of zero is achieved for $(\delta A^{(1)}, x^{(1)})$ and $x^{(1)}$ can be taken as a solution to (2.3). In fact, there are many possible solutions in this case. For example,

[1] If violation occurs for some zero $\hat{x}^{(1)}$ this means that we must necessarily have $b = 0$, which contradicts our assumption of a nonzero vector b.

once one such $x^{(1)}$ has been found, an infinite number of others can be constructed from it. To verify this claim, assume $x^{(1)}$ is a vector that satisfies (3.8), viz., it satisfies

$$\eta\|\hat{x}^{(1)}\|_2 \;=\; \|A\hat{x}^{(1)} - b\|_2 \;+\; \epsilon, \tag{3.10}$$

for some $\epsilon \geq 0$. Now assume we replace $x^{(1)}$ by $x^{(2)} = (x^{(1)} + \delta)$ for some vector δ to be determined so as to violate condition (3.3) and, therefore, also satisfy a relation of the form

$$\eta\|\hat{x}^{(2)}\|_2 \;\geq\; \|A\hat{x}^{(2)} - b\|_2 \;. \tag{3.11}$$

If such an $x^{(2)}$ can be found, then constructing the corresponding $\delta A^{(2)}$ as in (3.9) would also lead to a solution $(\delta A^{(2)}, x^{(2)})$.

Condition (3.11) requires a choice for the vector δ such that

$$\eta\|\hat{x}^{(1)} + \delta\|_2 \;\geq\; \|A(\hat{x}^{(1)} + \delta) - b\|_2. \tag{3.12}$$

But this can be satisfied by imposing the sufficient condition

$$\eta\|x^{(1)}\|_2 \;-\; \eta\|\delta\|_2 \;\;\geq\;\; \|A\hat{x}^{(1)} - b\|_2 \;+\; \|A\|_2\|\delta\|_2,$$

where the left-hand side is the smallest $\eta\|\hat{x}^{(1)} + \delta\|_2$ can get, while the right-hand side is the largest $\|A(\hat{x}^{(1)} + \delta) - b\|_2$ can get. Solving for $\|\delta\|_2$ we see that any vector δ that satisfies

$$\|\delta\|_2 \;\leq\; \frac{\epsilon}{\eta +\; \|A\|_2}$$

will lead to a new vector $\hat{x}^{(2)}$ that also violates (3.3). Consequently, given any single nonzero violation $\hat{x}^{(1)}$, many others can be obtained by suitably perturbing it.

We shall not treat the degenerate case in this paper (nor the case when (3.8) is violated only with equality). We shall instead assume throughout that the fundamental condition (3.3) holds. Under this assumption, the problem will turn out to always have a unique solution.

3.3. The fundamental condition for nondegeneracy.

The fundamental condition (3.3) needs to be satisfied for all vectors $\hat{x}$. This can be restated in terms of conditions on the data (A, b, η) alone. To see this, note that (3.3) implies, by squaring, that we must have

$$J(\hat{x}) \overset{\Delta}{=} \hat{x}^T(\eta^2 I - A^T A)\hat{x} + 2\hat{x}^T A^T b - b^T b \;< 0 \;\text{ for all }\; \hat{x}. \tag{3.13}$$

That is, the quadratic form $J(\hat{x})$ that is defined on the left-hand side of (3.13) must be negative for any value of the independent variable $\hat{x}$. This is possible only if

(i) the quadratic form $J(\hat{x})$ has a maximum with respect to $\hat{x}$, and

(ii) the value of $J(\hat{x})$ at its maximum is negative.

The necessary condition for the existence of a unique maximum (since we have a quadratic cost function) is

$$(\eta^2 I - A^T A) \;<\; 0, \tag{3.14}$$

which means that η should satisfy

$$\eta \;<\; \sigma_{min}(A). \tag{3.15}$$

Under this condition, the expression for the maximum point $\hat{x}_{max}$ of $J(\hat{x})$ is

$$\hat{x}_{max} = (A^TA - \eta^2 I)^{-1}A^Tb.$$

Evaluating $J(\hat{x})$ at $\hat{x} = \hat{x}_{max}$ we obtain

$$J(\hat{x}_{max}) = b^T\left[A(A^TA - \eta^2 I)^{-1}A^T \;-\; I\right]b.$$

Therefore, the requirement that $J(\hat{x}_{max})$ be negative corresponds to

$$b^T\left[I \;-\; A(A^TA - \eta^2 I)^{-1}A^T\right]b \;> 0. \tag{3.16}$$

Lemma 3.1. *Necessary and sufficient conditions in terms of* (A, b, η) *for the fundamental relation* (3.3) *to hold are the following:*

$$(\eta^2 I - A^TA) \;<\; 0 \qquad \Longleftrightarrow \qquad \eta \;<\; \sigma_{min}(A), \tag{3.17}$$

and

$$b^T\left[I \;-\; A(A^TA - \eta^2 I)^{-1}A^T\right]b \;> 0. \tag{3.18}$$

Note that for a well-defined problem of the form (2.3) we need to assume $\eta > 0$ which, in view of (3.17), means that A should be full rank so that $\sigma_{min}(A) > 0$. We therefore assume, from now on, that

A is full rank.

We further introduce the SVD of A:

$$A = U\begin{bmatrix}\Sigma\\ 0\end{bmatrix}V^T, \tag{3.19}$$

where $U \in \mathbf{R}^{m\times m}$ and $V \in \mathbf{R}^{n\times n}$ are orthogonal and $\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_n)$ is diagonal, with

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{n-1} \geq \sigma_n > 0$$

being the singular values of A. We further partition the vector U^Tb into

$$\begin{bmatrix}b_1\\ b_2\end{bmatrix} = U^Tb, \tag{3.20}$$

where $b_1 \in \mathbf{R}^n$ and $b_2 \in \mathbf{R}^{m-n}$.

While solving the minimization problem (3.4), we shall first assume that the two smallest singular values of A are distinct and, hence, satisfy

$$\sigma_n < \sigma_{n-1}.$$

Later, in section 4.6, we consider the case in which multiple singular values can occur.

4. Solving the minimization problem. To solve (3.4), we define the nonconvex cost function

$$\mathcal{L}(\hat{x}) = \|A\hat{x} - b\|_2 - \eta\|\hat{x}\|_2,$$

which is continuous in $\hat{x}$ and bounded from below by zero in view of (3.3). A minimum point for $\mathcal{L}(\hat{x})$ can occur only at ∞, at points where $\mathcal{L}(\hat{x})$ is not differentiable, or at points where its gradient, $\bigtriangledown\mathcal{L}(\hat{x})$, is 0. In particular, note that $\mathcal{L}(\hat{x})$ is not differentiable only at $\hat{x} = 0$ and at any $\hat{x}$ that satisfies $A\hat{x} - b = 0$. But points $\hat{x}$ satisfying $A\hat{x} - b = 0$ are excluded by the fundamental condition (3.3). Also, we can rule out $\hat{x} = \infty$ since

$$\lim_{\|\hat{x}\|_2 \to \infty} \mathcal{L}(\hat{x}) \;=\; \lim_{\|\hat{x}\|_2 \to \infty} \|A\hat{x}\|_2 - \eta\|\hat{x}\|_2 \;\geq\; (\sigma_n - \eta)\|\hat{x}\|_2 \;\to +\infty.$$

Now at points where $\mathcal{L}(\hat{x})$ is differentiable, the gradient of $\mathcal{L}$ is given by

$$\begin{aligned}\bigtriangledown\mathcal{L}(\hat{x}) &= \frac{1}{\|A\hat{x} - b\|_2} A^T \left(A\hat{x} - b\right) - \frac{\eta}{\|\hat{x}\|_2}\hat{x} \\ &= \frac{1}{\|A\hat{x} - b\|_2}\left(\left(A^T A - \alpha I\right)\hat{x} - A^T b\right),\end{aligned}$$

where we have introduced the positive real number

$$\alpha = \frac{\eta\|A\hat{x} - b\|_2}{\|\hat{x}\|_2}. \tag{4.1}$$

In view of the fundamental condition (3.3) we see that the value of α is necessarily larger than η^2,

$$\alpha > \eta^2. \tag{4.2}$$

Likewise, the Hessian of $\mathcal{L}$ is given by

$$\Delta\mathcal{L}(\hat{x}) = \frac{A^T A}{\|A\hat{x} - b\|_2} - \frac{\eta}{\|x\|_2} I - \frac{A^T(A\hat{x} - b)(A\hat{x} - b)^T A}{\|A\hat{x} - b\|_2^3} + \eta\frac{xx^T}{\|x\|_2^3}. \tag{4.3}$$

The critical points of $\mathcal{L}(\hat{x})$ (where the gradient is singular) satisfy

$$A^T(A\hat{x} - b) - \alpha\hat{x} = 0$$

or, equivalently,

$$\left(A^T A - \alpha I\right)\hat{x} = A^T b. \tag{4.4}$$

Equations (4.1) and (4.4) completely specify the stationary points of $\mathcal{L}(\hat{x})$. They provide two equations in the unknowns $(\alpha, \hat{x})$. We can use (4.4) to eliminate $\hat{x}$ from (4.1) and, hence, obtain an equation in α. Once we solve for α, we can then use (4.4) to determine the solution $\hat{x}$. The equation we obtain for α will in general be a nonlinear equation and the desired α will be a root of it. The purpose of the discussion in what follows is to show where the root α that corresponds to the global minimizer of $\mathcal{L}$ lies and how to find it.

We know from (4.2) that $\alpha > \eta^2$. We shall soon show that we need only to look for the solution α in the interval $(\eta^2, \sigma_{n-1}^2]$—see (4.5). (Further analysis later in the paper will in fact show that α lies within the smaller interval $(\eta^2, \sigma_n^2]$.) Hence, the coefficient matrix $(A^TA - \alpha I)$ in (4.4) is always nonsingular except for $\alpha = \sigma_{n-1}^2$ or $\alpha = \sigma_n^2$.

In summary, we see that the candidate solutions $\hat{x}$ to our minimization problem are the following:

1. $\hat{x} = 0$, which is a point at which $\mathcal{L}$ is not differentiable. We shall show that $\hat{x} = 0$ cannot be a global minimizer of $\mathcal{L}$.
2. Solution(s) $(\alpha, \hat{x})$ to (4.1) and (4.4) when $(A^TA - \alpha I)$ is invertible. In this case, we will see that α can lie only in the open interval (η^2, σ_n^2).
3. Solutions $(\alpha, \hat{x})$ to (4.1) and (4.4) when $(A^TA - \alpha I)$ is singular. We will see that this can happen only for the choices $\alpha = \sigma_{n-1}^2$ or $\alpha = \sigma_n^2$.

The purpose of the analysis in the sequel is to rule out all the possibilities except for one as a global minimum for $\mathcal{L}$. In loose terms, we shall show that in general a unique global minimizer $(\alpha, \hat{x})$ exists and that the corresponding α lies in the open interval (η^2, σ_n^2). Only in a degenerate case, the solution is obtained by taking $\alpha = \sigma_n^2$ and by solving (4.4) for $\hat{x}$. In other words, the global minimum will be obtained from the stationary points of $\mathcal{L}$, which is why we continue to focus on them.

The final statement of the solution is summarized in section 4.7.

4.1. Positivity of the Hessian matrix. We are of course interested only in those critical points of $\mathcal{L}$ that are candidates for local minima. Hence, the Hessian matrix at these points must be positive semidefinite.

Since $A^T(A\hat{x} - b) = \alpha\hat{x}$ at a critical point, we conclude from (4.3) that

$$\begin{aligned}\Delta\mathcal{L}(\hat{x}) &= \frac{1}{\|A\hat{x} - b\|_2}\left(A^TA - \alpha I\right) + \left(-\frac{\alpha^2}{\|A\hat{x} - b\|_2^3} + \frac{\eta}{\|\hat{x}\|_2^3}\right)\hat{x}\hat{x}^T \\ &= \frac{1}{\|A\hat{x} - b\|_2}\left(A^TA - \alpha I\right) + \frac{1}{\|A\hat{x} - b\|_2^3}\left(-\alpha^2 + \frac{\alpha^3}{\eta^2}\right)\hat{x}\hat{x}^T \\ &= \frac{1}{\|A\hat{x} - b\|_2}\left(A^TA - \alpha I\right) + \frac{\alpha^2}{\|A\hat{x} - b\|_2^3\eta^2}(\alpha - \eta^2)\hat{x}\hat{x}^T.\end{aligned}$$

Now observe that the second term is a symmetric rank-1 matrix that is also positive-semidefinite since $\alpha > \eta^2$. Hence, in view of the Cauchy interlacing theorem [3] the smallest eigenvalue of $\Delta\mathcal{L}(\hat{x})$ will lie between the two smallest eigenvalues of the matrix $\frac{1}{\|A\hat{x}-b\|_2}\left(A^TA - \alpha I\right)$. This shows that the value of α cannot exceed σ_{n-1}^2 since otherwise the two smallest eigenvalues of $\left(A^TA - \alpha I\right)$ will be nonpositive and the Hessian matrix will have a nonpositive eigenvalue.

The above argument explains why we need only to look for α in the interval

$$\eta^2 < \alpha \leq \sigma_{n-1}^2. \tag{4.5}$$

4.2. Solving for $\hat{x}$ and the secular equation. Given that we need only consider values of α in the interval $(\eta^2, \sigma_{n-1}^2]$, we can now solve for $\hat{x}$ using (4.1) and (4.4). Two cases should be considered since the coefficient matrix $(A^TA - \alpha I)$ may be singular for $\alpha = \sigma_n^2$ or $\alpha = \sigma_{n-1}^2$.

I. The case $\alpha \notin \{\sigma_n^2, \sigma_{n-1}^2\}$. From (4.4) we see that among the α's in the interval (4.5), as long as α is not equal to either σ_n^2 or σ_{n-1}^2, the critical point $\hat{x}$ associated with α is given uniquely by

$$\hat{x} = (A^TA - \alpha I)^{-1}A^Tb \qquad \text{for } \alpha \in (\eta^2, \sigma_{n-1}^2] \text{ and } \alpha \neq \sigma_n^2, \sigma_{n-1}^2.$$

Moreover, from (4.1) and (4.4) we see that

$$\mathcal{G}(\alpha) \triangleq \|\hat{x}\|_2^2 - \frac{\eta^2 \|A\hat{x} - b\|_2^2}{\alpha^2} = 0.$$

Substituting for $\hat{x}$ and using the SVD of A to simplify we obtain the equivalent expressions for $\hat{x}$ and $\mathcal{G}(\alpha)$:

$$\hat{x} = V(\Sigma^2 - \alpha I)^{-1} \Sigma b_1, \tag{4.6}$$

$$\mathcal{G}(\alpha) = b_1^T \left(\Sigma^2 - \eta^2 I\right) \left(\Sigma^2 - \alpha I\right)^{-2} b_1 - \frac{\eta^2 \|b_2\|_2^2}{\alpha^2}. \tag{4.7}$$

Clearly the roots of $\mathcal{G}(\alpha)$ that lie in the interval $(\eta^2, \sigma_{n-1}^2]$ will correspond to critical points that are candidates for local minima. Therefore we will later investigate the roots of $\mathcal{G}(\alpha)$.

II. The case $\boldsymbol{\alpha = \sigma_n^2}$ or $\boldsymbol{\alpha = \sigma_{n-1}^2}$. From (4.4) we see that $\alpha = \sigma_n^2$ or $\alpha = \sigma_{n-1}^2$ can correspond to a critical point $\hat{x}$ only if either $u_n^T b = 0$ (for $\alpha = \sigma_n^2$) or $u_{n-1}^T b = 0$ (for $\alpha = \sigma_{n-1}^2$). Here, $\{u_n, u_{n-1}\}$ denote the columns of U that correspond to $\{\sigma_n, \sigma_{n-1}\}$, i.e., the last two columns of U.

Here we show only how to solve for $\hat{x}$ when $\alpha = \sigma_n^2$. The technique for $\alpha = \sigma_{n-1}^2$ is similar.

From (4.4) it is clear that $\alpha = \sigma_n^2$ is a candidate for a critical point if and only if $u_n^T b = 0$. In this case the associated $\hat{x}$'s (there may be more than one) satisfy the equations

$$(A^T A - \sigma_n^2 I)\hat{x} = A^T b \tag{4.8}$$

and

$$\sigma_n^2 = \eta \frac{\sqrt{\|b_2\|_2^2 + \|\Sigma V^T \hat{x} - b_1\|_2^2}}{\|\hat{x}\|_2}. \tag{4.9}$$

Now we define $y \triangleq V^T \hat{x}$ and consider the following partitionings:

$$y = \begin{bmatrix} \bar{y} \\ y_n \end{bmatrix}, \quad b_1 = \begin{bmatrix} \bar{b}_1 \\ 0 \end{bmatrix}, \quad \Sigma = \begin{bmatrix} \bar{\Sigma} & 0 \\ 0 & \sigma_n \end{bmatrix}.$$

The quantities $\hat{x}$ and y define each other uniquely and $\|y\|_2 = \|\hat{x}\|_2$.

It follows from (4.8) that

$$\bar{y} = \left(\bar{\Sigma}^2 - \sigma_n^2 I\right)^{-1} \bar{\Sigma} \bar{b}_1. \tag{4.10}$$

Substituting this into (4.9) we have

$$\sigma_n^4 = \eta^2 \frac{\|b_2\|_2^2 + \|\bar{\Sigma}^2 \left(\bar{\Sigma}^2 - \sigma_n^2 I\right)^{-1} \bar{b}_1 - \bar{b}_1\|_2^2 + \sigma_n^2 y_n^2}{\|\bar{y}\|_2^2 + y_n^2}.$$

Solving for y_n^2 we obtain

$$\begin{aligned} y_n^2 &= -\sigma_n^2 \frac{\bar{b}_1^T \left(\bar{\Sigma}^2 - \eta^2 I\right) \left(\bar{\Sigma}^2 - \sigma_n^2 I\right)^{-2} \bar{b}_1 - \|b_2\|_2^2 \eta^2 \sigma_n^{-4}}{\sigma_n^2 - \eta^2} \\ &= -\frac{\sigma_n^2}{\sigma_n^2 - \eta^2} \, \bar{\mathcal{G}}(\sigma_n^2), \end{aligned}$$

where we introduced the function

$$\bar{\mathcal{G}}(\alpha) \triangleq \bar{b}_1^T \left(\bar{\Sigma}^2 - \eta^2 I\right) \left(\bar{\Sigma}^2 - \alpha I\right)^{-2} \bar{b}_1 - \frac{\eta^2 \|b_2\|_2^2}{\alpha^2}. \tag{4.11}$$

Comparing with the definition (4.7) for $\mathcal{G}(\alpha)$ we see that

$$\bar{\mathcal{G}}(\alpha) = \mathcal{G}(\alpha) - (u_n^T b)^2 \frac{\sigma_n^2 - \eta^2}{(\sigma_n^2 - \alpha)^2}. \tag{4.12}$$

Note that $\mathcal{G}(\alpha) = \bar{\mathcal{G}}(\alpha)$ if $u_n^T b = 0$, which is the case when $\alpha = \sigma_n^2$ is a possibility. Therefore the possible values of y_n are

$$y_n = \frac{\sigma_n}{\sqrt{\sigma_n^2 - \eta^2}} \left(\pm \sqrt{-\bar{\mathcal{G}}(\sigma_n^2)} \right). \tag{4.13}$$

It follows that

$$u_n^T b = 0 \quad \text{and} \quad \bar{\mathcal{G}}(\sigma_n^2) \leq 0 \tag{4.14}$$

are the necessary conditions for $\alpha = \sigma_n^2$ to correspond to a stationary double point at

$$\hat{x} \;=\; V \begin{bmatrix} \left(\bar{\Sigma}^2 - \sigma_n^2 I\right)^{-1} \bar{\Sigma} \bar{b}_1 \\ \frac{\sigma_n}{\sqrt{\sigma_n^2 - \eta^2}} \left(\pm \sqrt{-\bar{\mathcal{G}}(\sigma_n^2)} \right) \end{bmatrix}. \tag{4.15}$$

The global minimum. The purpose of the analysis in the following sections is to show that in general the global minimum is given by (4.6) with the corresponding α lying in the interval (η^2, σ_n^2). When a root of the secular equation $\mathcal{G}(\alpha)$ in (4.7) does not exist in the open interval (η^2, σ_n^2), we shall then show that the global minimum is given by (4.15).

4.3. The roots of the secular equation.

We have argued earlier in (4.5) that the roots α of $\mathcal{G}(\alpha)$ that may lead to global minimizers $\hat{x}$ can lie in the interval $(\eta^2, \sigma_{n-1}^2]$. We now determine how many roots can exist in this interval and later show that only the root lying in the subinterval (η^2, σ_n^2) corresponds to a global minimum when it exists. Otherwise, we have to use (4.15). The details are given below.

To begin with, we establish some properties of $\mathcal{G}(\alpha)$. From the nondegeneracy assumption (3.3) on the data it follows that $\mathcal{G}(\eta^2) < 0$. Moreover, from the expression (4.7) for $\mathcal{G}(\alpha)$ we see that it has a pole at σ_n^2 provided that $u_n^T b$ is not equal to zero, in which case

$$\lim_{\alpha \to \sigma_n^2} \mathcal{G}(\alpha) = +\infty. \tag{4.16}$$

Now observe from the expression for the derivative of $\mathcal{G}(\alpha)$,

$$\mathcal{G}'(\alpha) = 2 \left(b_1^T \left(\Sigma^2 - \eta^2 I\right) \left(\Sigma^2 - \alpha I\right)^{-3} b_1 \right) + \frac{\eta^2 \|b_2\|_2^2}{\alpha^3}, \tag{4.17}$$

that $\mathcal{G}'(\alpha) > 0$ for $0 < \alpha < \sigma_n^2$. We conclude from these facts that $\mathcal{G}(\alpha)$ has exactly one root in the open interval (η^2, σ_n^2). Actually, also since

$$\lim_{\alpha \to 0+} \mathcal{G}(\alpha) = -\infty,$$

we conclude that when $u_n^T b \neq 0$, $\mathcal{G}(\alpha)$ has exactly one root in the interval $(0, \sigma_n^2)$ and that this root lies in the subinterval (η^2, σ_n^2).

When $u_n^T b = 0$, the function $\mathcal{G}(\alpha)$ does not have a pole at σ_n^2 and, hence, (4.16) does not hold. However, by using the still-valid fact that $\lim_{\alpha \to 0+} \mathcal{G}(\alpha) = -\infty$ and that $\mathcal{G}'(\alpha) > 0$ over the larger interval $(0, \sigma_{n-1}^2)$, we conclude the following:

1. If $u_{n-1}^T b \neq 0$ then $\lim_{\alpha \to \sigma_{n-1}^2} \mathcal{G}(\alpha) = +\infty$ and, hence, $\mathcal{G}(\alpha)$ has a unique root in the interval $(0, \sigma_{n-1}^2)$.
2. If we also have $u_{n-1}^T b = 0$, then $\mathcal{G}(\alpha)$ can have at most one root in the interval $(0, \sigma_{n-1}^2)$. The root may or may not exist.

What about the interval $(\sigma_n^2, \sigma_{n-1}^2)$ when $u_n^T b \neq 0$? We now establish that $\mathcal{G}(\alpha)$ can have at most two roots in this interval. For this purpose, we first observe that both $\mathcal{G}(\alpha)$ and $\alpha^2 \mathcal{G}(\alpha)$ have the same number of roots in $(\sigma_n^2, \sigma_{n-1}^2)$ (since we added only a double root at 0). Next we compute the first derivative of $\alpha^2 \mathcal{G}(\alpha)$, obtaining

$$\frac{d}{d\alpha}\left(\alpha^2 \mathcal{G}(\alpha)\right) = 2\alpha b_1^T \left(\Sigma^2 - \eta^2 I\right) \Sigma^2 \left(\Sigma^2 - \alpha I\right)^{-3} b_1.$$

Using this we compute the second derivative obtaining

$$\frac{d^2}{d\alpha^2}\left(\alpha^2 \mathcal{G}(\alpha)\right) = 2 b_1^T \left(\Sigma^2 - \eta^2 I\right) \Sigma^2 \left(\Sigma^2 + 2\alpha I\right) \left(\Sigma^2 - \alpha I\right)^{-4} b_1.$$

It is clear that the second derivative is strictly positive for nonnegative α. From this we can conclude that $\alpha^2 \mathcal{G}(\alpha)$ and, hence, $\mathcal{G}(\alpha)$ have at most two zeros in $(\sigma_n^2, \sigma_{n-1}^2)$. We have therefore established the following result.

LEMMA 4.1. *The following properties hold for the function $\mathcal{G}(\alpha)$ defined in* (4.7)*:*

1. *When $u_n^T b \neq 0$, the function $\mathcal{G}(\alpha)$ has a single root in the interval (η^2, σ_n^2) and at most two roots in the interval $(\sigma_n^2, \sigma_{n-1}^2)$. We label them as*
$$\eta^2 < \alpha_1 < \sigma_n^2 < \alpha_2 \leq \alpha_3 < \sigma_{n-1}^2.$$
2. *When $u_n^T b = 0$ and $u_{n-1}^T b \neq 0$, the function $\mathcal{G}(\alpha)$ has a unique root in the interval (η^2, σ_{n-1}^2).*
3. *When $u_n^T b = 0$ and $u_{n-1}^T b = 0$, the function $\mathcal{G}(\alpha)$ has at most one root in the interval (η^2, σ_{n-1}^2).*

It is essential to remember that the roots α_2 and α_3 may not exist, though they must occur as a pair (counting multiplicity) if they exist.

We now show that α_3 cannot correspond to a local minimum if $\alpha_2 < \alpha_3$, i.e., if the two roots in the interval $(\sigma_n^2, \sigma_{n-1}^2)$ are distinct. Indeed, assume α_2 and α_3 exist. Then from the last lemma it must hold that $u_n^T b \neq 0$ and α_1 must also exist. Hence, we must have $\mathcal{G}'(\alpha_2) < 0$ and $\mathcal{G}'(\alpha_3) > 0$.

If we assume $\alpha_2 < \alpha_3$, we shall use the fact that $\mathcal{G}'(\alpha_3) > 0$ to show that α_3 cannot correspond to a local minimum solution $\hat{x}$. This will be achieved by showing that the determinant of the Hessian of $\mathcal{L}(\hat{x})$ at α_3 is negative. For this we note that

$$\begin{aligned}\det\left(\Delta \mathcal{L}(\hat{x})\right) &= \frac{\det(A^T A - \alpha I)}{\|A\hat{x} - b\|_2^n} \left(1 + \frac{\alpha^2(\alpha - \eta^2)}{\|A\hat{x} - b\|_2^2 \eta^2} \hat{x}^T \left(A^T A - \alpha I\right)^{-1} \hat{x}\right) \\ &= \frac{\det(A^T A - \alpha I)}{\|A\hat{x} - b\|_2^{n+2} \eta^2} \left(\|A\hat{x} - b\|_2^2 \eta^2 + \alpha^2(\alpha - \eta^2) \hat{x}^T (A^T A - \alpha I)^{-1} \hat{x}\right).\end{aligned}$$

We introduce for convenience the shorthand notation

$$\xi \triangleq \frac{\det(A^T A - \alpha I)}{\|A\hat{x} - b\|_2^{n+2} \eta^2}.$$

Then, using the SVD of A,

$$\begin{aligned}\det\left(\Delta\mathcal{L}(\hat{x})\right) &= \xi\left(\eta^2\|b_2\|_2^2 + \alpha^3 b_1^T\left(\Sigma^2-\eta^2 I\right)\left(\Sigma^2-\alpha I\right)^{-3} b_1\right)\\ &= \xi\frac{\alpha^3}{2}\mathcal{G}'(\alpha).\end{aligned}$$

Evaluating at $\alpha=\alpha_3$, and noting that $\det(A^TA-\alpha_3 I)<0$ and $\mathcal{G}'(\alpha_3)>0$, we conclude that $\det(\mathcal{L})$ at the $\hat{x}$ corresponding to α_3 is negative. Hence, α_3 cannot correspond to a local minimum.

4.4. Candidates for minima. We can now be more explicit about the candidates for global minimizers of $\mathcal{L}$, which we mentioned just prior to section 4.1:

1. $\hat{x}=0$, which corresponds to a point where $\mathcal{L}$ is not differentiable.
2. If α_1 exists, then the corresponding $\hat{x}$ is a candidate. Recall that α is guaranteed to exist if $u_n^Tb\neq 0$. It may or may not exist otherwise.
3. If $u_n^Tb\neq 0$ and α_2 exists, then the corresponding $\hat{x}$ is a candidate.
4. If $u_n^Tb=0$, then the $\hat{x}$ associated with $\alpha=\sigma_n^2$ is a candidate.
5. If $u_{n-1}^Tb=0$, then the $\hat{x}$ associated with $\alpha=\sigma_{n-1}^2$ is a candidate.

We shall show that item 2 is the global minimizer when α_1 exists. Otherwise, item 4 is the global minimizer.

We start by showing that $\hat{x}=0$ cannot be the global minimizer of $\mathcal{L}$. We divide our argument into two cases: $b_1=0$ and $b_1\neq 0$. When $b_1\neq 0$, we necessarily have $u_i^Tb\neq 0$ for some i between 1 and n. Let $z=(u_i^Tb/\sigma_i)Ve_i$. Then $\mathcal{L}(0)=\|b\|_2$, and

$$\mathcal{L}(z)=\|b-(u_i^Tb)u_i\| \;-\; \frac{\eta}{\sigma_i}|u_i^Tb|.$$

The term $b-(u_i^Tb)u_i$ is the error vector due to projecting b onto u_i. Hence,

$$\|b-(u_i^Tb)u_i\|^2=\|b\|^2-(u_i^Tb)^2,$$

and we obtain

$$\mathcal{L}(z)=\sqrt{\|b\|_2^2-(u_i^Tb)^2}-\frac{\eta}{\sigma_i}|u_i^Tb|.$$

We conclude that $\mathcal{L}(0)>\mathcal{L}(z)$, and $\hat{x}=0$ cannot correspond to a global minimum when $b_1\neq 0$.

Now we consider the case when $b_1=0$. Note that by the nondegeneracy assumption we must have $b_2\neq 0$. Now define again $y=V^T\hat{x}$. Then we can simplify $\mathcal{L}(\hat{x})$ to obtain

$$\mathcal{L}(\hat{x})=\sqrt{\|b_2\|_2^2+\|\Sigma y\|_2^2}-\eta\|y\|_2.$$

Choose $\hat{x}=\gamma Ve_n$, where $0<\gamma<\frac{2\|b_2\|_2\eta}{\sigma_n^2-\eta^2}$. The following sequence of inequalities then holds:

$$\begin{aligned}\gamma\quad &< \frac{2\|b_2\|_2\eta}{\sigma_n^2-\eta^2}\\ \Rightarrow\quad (\sigma_n^2-\eta^2)\gamma^2 &< 2\|b_2\|_2\eta\gamma\\ \Rightarrow\quad \|b_2\|_2^2+\sigma_n^2\gamma^2 &< \|b_2\|_2^2+\eta^2\gamma^2+2\|b_2\|_2\eta\gamma\\ \Rightarrow\sqrt{\|b_2\|_2^2+\|\Sigma\gamma e_n\|^2} &< \|b_2\|_2+\eta\gamma\\ \Rightarrow\quad \mathcal{L}(\gamma Ve_n) &< \mathcal{L}(0).\end{aligned}$$

Therefore $\hat{x} = 0$ can never be the global minimum.

4.5. Continuation argument. We are now ready to show that if α_1 exists, then the corresponding $\hat{x}$ in (4.6) gives the global minimum. Otherwise, the $\hat{x}$ in (4.15) that corresponds $\alpha = \sigma_n^2$ gives a double global minimum. The proof will be by continuation on the parameter $\beta \triangleq |u_n^T b|^2$.

We use (4.12) to write

$$\mathcal{G}(\alpha) = \bar{\mathcal{G}}(\alpha) + \beta \frac{\sigma_n^2 - \eta^2}{(\sigma_n^2 - \alpha)^2}. \tag{4.18}$$

We also recall from the definition of $\bar{\mathcal{G}}$ in (4.11) that it has a similar expression to that of $\mathcal{G}$ in (4.7), except that the pole of $\mathcal{G}$ at σ_n^2 has been extracted (as shown by (4.18)). Hence, the derivative of $\bar{\mathcal{G}}$ has a form similar to that of the derivative of $\mathcal{G}$ in (4.17) and we can conclude that $\bar{\mathcal{G}}'(\alpha) > 0$ over $(0, \sigma_{n-1}^2)$.

We continue our argument by considering separately the three cases: $\bar{\mathcal{G}}(\sigma_n^2) > 0$, $\bar{\mathcal{G}}(\sigma_n^2) = 0$, and $\bar{\mathcal{G}}(\sigma_n^2) < 0$.

I. $\boldsymbol{\bar{\mathcal{G}}(\sigma_n^2) > 0}$. In this case, and because of (4.14), $\alpha = \sigma_n^2$ cannot correspond to a global minimum. By further noting that $\bar{\mathcal{G}}'(\alpha) > 0$ over $(\sigma_n^2, \sigma_{n-1}^2)$ we conclude that $\bar{\mathcal{G}}(\alpha) > 0$ over $(\sigma_n^2, \sigma_{n-1}^2)$. Hence, using (4.18) we also have that $\mathcal{G}(\alpha) > 0$ over $(\sigma_n^2, \sigma_{n-1}^2)$. Moreover, σ_n^2 is either a pole of $\mathcal{G}(\alpha)$ (when $u_n^T b \neq 0$) or $\mathcal{G}(\sigma_n^2) = \bar{\mathcal{G}}(\sigma_n^2) > 0$ (when $u_n^T b = 0$). Now since $\mathcal{G}(\eta^2) < 0$ and $\mathcal{G}'(\alpha) > 0$ over (η^2, σ_n^2), we conclude that $\mathcal{G}$ has a unique root in (η^2, σ_n^2).

In this case the only contenders for global minima over $(0, \sigma_{n-1}^2]$ are $\alpha = \alpha_1$ and $\alpha = \sigma_{n-1}^2$. By an analysis similar to the one in section 4.2 for $\alpha = \sigma_n^2$ it can be shown that a necessary condition for $\alpha = \sigma_{n-1}^2$ to correspond to a global minimum is that

$$u_{n-1}^T b = 0, \quad \mathcal{G}(\sigma_{n-1}^2) - (u_{n-1}^T b)^2 \frac{\sigma_{n-1}^2 - \eta^2}{(\sigma_{n-1}^2 - \alpha)^2} \leq 0. \tag{4.19}$$

These two conditions imply that we must have $\mathcal{G}(\sigma_{n-1}^2) \leq 0$. This result is compatible with the fact that $\mathcal{G}(\alpha) > 0$ over $(\sigma_n^2, \sigma_{n-1}^2)$ only if $\mathcal{G}(\sigma_{n-1}^2) = 0$. This in turn implies from (4.18) that we must have $\bar{\mathcal{G}}(\sigma_{n-1}^2) \leq 0$. This is inconsistent with the facts $\bar{\mathcal{G}}(\sigma_n^2) > 0$ and $\bar{\mathcal{G}}'(\alpha) > 0$ over $(\sigma_n^2, \sigma_{n-1}^2)$.

Therefore, $\alpha = \sigma_{n-1}^2$ cannot correspond to a global minimum and we conclude that the only critical point we need to consider corresponds to the one associated with the unique root of $\mathcal{G}(\alpha)$ in (η^2, σ_n^2), which must naturally correspond to the global minimum.

In summary, the solution $\hat{x}$ in (4.6) that corresponds to α_1 is the global minimum in this case.

II. $\boldsymbol{\bar{\mathcal{G}}(\sigma_n^2) = 0}$. In this case, and because of (4.14), $\alpha = \sigma_n^2$ can correspond to a global minimum only if $\beta = 0$, in which case we also deduce from (4.18) that $\mathcal{G}(\sigma_n^2) = 0$ since $\mathcal{G} = \bar{\mathcal{G}}$. Hence, σ_n^2 is a root of $\mathcal{G}$. By using $\mathcal{G}(\eta^2) < 0$ and $\mathcal{G}'(\alpha) > 0$ over (η^2, σ_{n-1}^2) we conclude that $\mathcal{G}$ does not have any other root in (η^2, σ_{n-1}^2).

Therefore, when $\beta = 0$, the only contenders for global minima over $(0, \sigma_{n-1}^2)$ are $\alpha = \sigma_n^2$ and $\alpha = \sigma_{n-1}^2$. For $\alpha = \sigma_{n-1}^2$ to correspond to a global minimum we saw above that we must necessarily have $\mathcal{G}(\sigma_{n-1}^2) \leq 0$. This is inconsistent with $\mathcal{G}(\sigma_n^2) > 0$ and $\mathcal{G}'(\alpha) > 0$ over (η^2, σ_{n-1}^2). We thus obtain that the solution $\hat{x}$ in (4.15) that corresponds to σ_n^2 is the global minimizer.

What about the case $\beta \neq 0$? In this case, and because of (4.14), $\alpha = \sigma_n^2$ cannot correspond to a global minimum. By further noting that $\bar{\mathcal{G}}'(\alpha) > 0$ over $(\sigma_n^2, \sigma_{n-1}^2)$

we conclude that $\bar{\mathcal{G}}(\alpha) > 0$ over $(\sigma_n^2, \sigma_{n-1}^2)$. Hence, using (4.18) we also have that $\mathcal{G}(\alpha) > 0$ over $(\sigma_n^2, \sigma_{n-1}^2)$. Moreover, σ_n^2 is now a pole of $\mathcal{G}(\alpha)$ and since $\mathcal{G}(\eta^2) < 0$ and $\mathcal{G}'(\alpha) > 0$ over (η^2, σ_n^2) we conclude that $\mathcal{G}$ has a unique root in (η^2, σ_n^2). In this case the only contenders for global minima over $(0, \sigma_{n-1}^2)$ are $\alpha = \alpha_1$ and $\alpha = \sigma_{n-1}^2$. By an analysis similar to the one in case I, we can rule out σ_{n-1}^2.

In summary, we showed the following when $\bar{\mathcal{G}}(\sigma_n^2) \geq 0$:

1. When $u_n^T b \neq 0$, the solution $\hat{x}$ in (4.6) that corresponds to α_1 is the global minimum.
2. When $u_n^T b = 0$, the solution $\hat{x}$ in (4.15) that corresponds to σ_n^2 is the global minimum.

III. $\boldsymbol{\bar{\mathcal{G}}(\sigma_n^2) < 0}$. This is the most complex situation. Let ω be the largest number such that $\sigma_n^2 < \omega \leq \infty$ and $\mathcal{G}(\omega) = \infty$ if $\omega < \infty$ and $\mathcal{G}(\alpha)$ has no poles in the interval (σ_n^2, ω).

By the given conditions it is obvious from the form of $\mathcal{G}(\alpha)$ that it has two roots in $(\sigma_n^2, \omega]$ for sufficiently small β. Now we find the largest number δ such that for all β in the interval $(0, \delta]$ the function $\mathcal{G}(\alpha)$ has two roots (counting multiplicity) in $(\sigma_n^2, \omega]$. We claim that for $\beta > \delta$ there are no roots of $\mathcal{G}(\alpha)$ in $(\sigma_n^2, \omega]$. To see this we replace β in $\mathcal{G}(\alpha)$ by $\beta = \delta + \nu$, obtaining

$$\mathcal{G}(\alpha) = \bar{\mathcal{G}}(\alpha) + \delta \frac{\sigma_n^2 - \eta^2}{(\sigma_n^2 - \alpha)^2} + \nu \frac{\sigma_n^2 - \eta^2}{(\sigma_n^2 - \alpha)^2},$$

and observe that the term involving ν is strictly positive in the interval $(\sigma_n^2, \omega]$, for strictly positive ν.

We continue our analysis by considering separately two cases: $\beta = 0$ and $\beta \neq 0$.

IIIA. $\boldsymbol{\beta = 0}$. We will show that at $\beta = 0$ ($u_n^T b = 0$), the function $\mathcal{L}$ has a double global minimum at $\alpha = \sigma_n^2$, and that as β is increased this double root at $\alpha = \sigma_n^2$ bifurcates into the two roots α_1 and α_2, and that $\mathcal{L}(\alpha_1) < \mathcal{L}(\alpha_2)$ when $0 < \beta \leq \delta$.

When $\bar{\mathcal{G}}(\sigma_n^2) < 0$, the function $\bar{\mathcal{G}}(\alpha)$ has exactly one root in $(0, \omega]$. This is because $\bar{\mathcal{G}}(0) \to -\infty$ and $\bar{\mathcal{G}}'(\alpha) > 0$ in $(0, \omega]$. By using the same proof that we used earlier to show that α_3 cannot correspond to a local minimum we can establish that this root also cannot correspond to minimum. This leaves us with the double stationary points that we computed in (4.15) and which corresponded to $\alpha = \sigma_n^2$. It is an easy matter to verify, using the formula $\mathcal{L}(\hat{x})\eta = (\alpha - \eta^2)\|\hat{x}\|_2$, that both the stationary points yield the same value for $\mathcal{L}$.

IIIB. $\boldsymbol{0 < \beta \leq \delta}$. We now allow β to increase. Let $y(\beta)$ denote the stationary point $\hat{x}$ corresponding to $\alpha_1(\beta)$. Also, let $z(\beta)$ denote the stationary point $\hat{x}$ corresponding to $\alpha_2(\beta)$. It is easy to see from the form of $\mathcal{G}(\alpha)$ that

$$\lim_{\beta \to 0+} \alpha_1(\beta) = \sigma_n^2 = \lim_{\beta \to 0+} \alpha_2(\beta)$$

whenever $\bar{\mathcal{G}}(\sigma_n^2) < 0$. Now we will show that

$$\lim_{\beta \to 0+} |y_i(\beta)| = |y_i(0)| = |z_i(0)| = \lim_{\beta \to 0+} |z_i(\beta)|, \qquad 1 \leq i \leq n.$$

First we observe that the result is true for $i \neq n$ directly from formulas (4.4) and (4.10). Next we note that $\bar{\mathcal{G}}(\alpha)$ is continuous at $\alpha = \sigma_n^2$. Therefore

$$\lim_{\beta \to 0+} \frac{\sigma_n^2 - \eta^2}{(\sigma_n^2 - \alpha_1(\beta))^2} \beta + \bar{\mathcal{G}}(\sigma_n^2) = \lim_{\beta \to 0+} \left(\frac{\sigma_n^2 - \eta^2}{(\sigma_n^2 - \alpha_1(\beta))^2} \beta + \bar{\mathcal{G}}(\alpha_1(\beta)) \right)$$

$$
\begin{aligned}
&= 0\\
&= \lim_{\beta\to 0+} \frac{\sigma_n^2 - \eta^2}{(\sigma_n^2 - \alpha_2(\beta))^2}\beta + \bar{\mathcal{G}}(\sigma_n^2).
\end{aligned}
$$

Now using formula (4.13) it can be verified that

$$\lim_{\beta\to 0+} |y_n(\beta)| = |y_n(0)| = |z_n(0)| = \lim_{\beta\to 0+} |z_n(\beta)|.$$

Therefore $\mathcal{L}(y(\beta))$ and $\mathcal{L}(z(\beta))$ are continuous on the interval $[0,\infty)$, with $\mathcal{L}(y(0)) = \mathcal{L}(z(0))$. We now compute the derivative of $\mathcal{L}$ with respect to β at a stationary point. We have already observed that at a stationary point $\hat{x}$, corresponding to some α, the objective function $\mathcal{L}$ can be simplified $\mathcal{L}(\hat{x})\eta = (\alpha - \eta^2)\|\hat{x}\|_2$. To simplify the derivation we actually take the derivative of $\eta^2 L^2$, which can be expressed as

$$\eta^2\mathcal{L}^2(\hat{x}) = (\alpha - \eta^2)^2 b_1^T \Sigma^2 \left(\Sigma^2 - \alpha I\right)^{-2} b_1$$

for $\alpha \in (\sigma_n, \sigma_{n-1})$. Now we differentiate, obtaining

$$\frac{d}{d\beta}\left(\eta^2 L^2(\hat{x}(\beta))\right) = \frac{d\alpha}{d\beta}\left(\alpha - \eta^2\right) b_1^T \Sigma^2 \left(\Sigma^2 - \alpha I\right)^{-3}\left(\Sigma^2 - \eta^2 I\right) b_1 + \frac{\left(\alpha - \eta^2\right)^2}{\left(\sigma_n^2 - \alpha\right)^2}\sigma_n^2.$$

Now we obtain an expression for $d\alpha/d\beta$ by differentiating $\alpha^2\mathcal{G}(\alpha(\beta)) = 0$ with respect to β. Doing so we obtain

$$2\alpha b_1^T \Sigma^2 \left(\Sigma^2 - \eta^2 I\right)\left(\Sigma^2 - \alpha I\right)^{-3} b_1 \frac{d\alpha}{d\beta} + \alpha^2 \frac{\sigma_n^2 - \eta^2}{\left(\sigma_n^2 - \alpha\right)^2} = 0.$$

Solving this equation for the term involving $d\alpha/d\beta$ and substituting it into the above equation for $\frac{d}{d\beta}\left(\eta^2 L^2(x(\beta))\right)$ we obtain

$$
\begin{aligned}
\frac{d}{d\beta}\left(\eta^2 L^2(\hat{x}(\beta))\right) &= -\alpha\left(\alpha - \eta^2\right)\frac{\sigma_n^2 - \eta^2}{\left(\sigma_n^2 - \alpha\right)^2} + \sigma_n^2 \frac{\left(\alpha - \eta^2\right)^2}{\left(\sigma_n^2 - \alpha\right)^2}\\
&= \frac{\alpha - \eta^2}{\left(\sigma_n^2 - \alpha\right)^2}\left(\sigma_n^2\left(\alpha - \eta^2\right) - \alpha\left(\sigma_n^2 - \eta^2\right)\right)\\
&= \frac{\alpha - \eta^2}{\left(\sigma_n^2 - \alpha\right)^2}\left(\alpha\eta^2 - \sigma_n^2\eta^2\right)\\
&= \frac{\alpha - \eta^2}{\alpha - \sigma_n^2}\eta^2. \qquad (4.20)
\end{aligned}
$$

From this expression we can immediately conclude that the smaller root $\alpha_1(\beta)$ decreases the objective function $\mathcal{L}(y(\beta))$, as β increases from 0, and the larger root $\alpha_2(\beta)$ increases the value of the objective function $\mathcal{L}(z(\beta))$, as β increases. Since $\mathcal{L}(y(0)) = \mathcal{L}(z(0))$, we can now conclude that $\mathcal{L}(y(\beta)) \le \mathcal{L}(z(\beta))$ for all nonnegative β such that $\alpha_2(\beta) < \sigma_{n-1}^2$.

Therefore the choice for global minimum is between $y(\beta)$ and the critical points, if any, corresponding to $\alpha = \sigma_{n-1}^2$. As mentioned before, $\alpha = \sigma_{n-1}^2$ can correspond to a critical point only if condition (4.19) holds.

From the arguments in section 4.3 we know that $\mathcal{G}(\alpha)$ has at most two roots in (σ_n^2, ω). Therefore it follows that under condition (4.19), $\omega > \sigma_{n-1}^2$. This in turn implies that $\alpha_2(\beta_0) = \sigma_{n-1}^2$ for some $\beta_0 \in (0, \delta]$.

Using condition (4.19) and carrying out an analysis similar to that of (4.15), we can compute the critical point associated with $\alpha = \sigma_{n-1}^2$. From that it is easy to verify that

$$\lim_{\beta \to \beta_0 -} |z_i(\beta)| = |z_i(\beta_0)|, \qquad 1 \le i \le n,$$

where $z(\beta_0)$ denotes the critical point associated with $\alpha = \sigma_{n-1}^2$.

Now from the continuation argument for $\mathcal{L}$ it follows that $\mathcal{L}(y(\beta)) \le \mathcal{L}(y(0)) = \mathcal{L}(z(0)) \le \mathcal{L}(z(\beta_0))$. Therefore we do not need to consider $\alpha = \sigma_{n-1}^2$ as a possibility for the global minimum.

Furthermore, when $\beta > \delta$ we argued earlier that there are no roots of $\mathcal{G}(\alpha)$ in the interval $(\sigma_n^2, \omega]$. Also, from the above argument it follows that $\alpha = \sigma_{n-1}^2$ cannot correspond to a global minimum.

In summary, the $\hat{x}$ in (4.6) that corresponds to α_1 is the global minimizer.

4.6. Multiple singular values.

So far in the analysis we have implicitly ignored the possibility that $\sigma_n = \sigma_{n-1}$. We now discuss how to take care of this possibility. We need only to consider critical points α situated in the interval $(0, \sigma_n^2]$.

Let U_n denote a matrix with orthonormal columns that span the left singular subspace associated with the smallest singular value of A. If $\|U_n^T b\|_2 > 0$, then it follows from (4.4) that $\alpha = \sigma_n^2$ is not a possibility for a critical point. Furthermore $\mathcal{G}(\alpha)$ has a single root in (η^2, σ_n^2) and this must give the global minimum.

If $\|U_n^T b\|_2 = 0$, then either there exists a root of $\mathcal{G}(\alpha)$ in the interval (η^2, σ_n^2), which corresponds to the global minimum, or there is no such root and $\alpha = \sigma_n^2$ will give rise to multiple global minima, all of which can be calculated by the technique that led to (4.15). The only difference is that $\bar{y}$ in (4.10) will now denote the components of y associated with the right singular vectors perpendicular to the range space of V_n, where V_n is a matrix with orthonormal columns that span the right singular subspace of A corresponding to σ_n. The proofs of these statements are similar to the nonmultiple singular value case.

4.7. Statement of the solution of the optimization problem.

We collect in the form of a theorem the conclusions of our earlier analysis.

THEOREM 4.2. *Given $A \in \mathbf{R}^{m\times n}$, with $m \ge n$ and A full rank, a nonzero $b \in \mathbf{R}^m$, and a positive real number η satisfying $\eta < \sigma_{min}(A)$, assume further that*

$$b^T \left[I \;-\; A(A^TA - \eta^2 I)^{-1} A^T \right] b \; > 0.$$

The solution of the optimization problem,

$$\min_{\hat{x}} \left(\min_{\|\delta A\|_2 \le \eta} \| \left(A + \delta A\right) \hat{x} - b \|_2 \right), \tag{4.21}$$

can be constructed as follows:

- *Introduce the SVD of A,*

$$A = U \begin{bmatrix} \Sigma \\ 0 \end{bmatrix} V^T, \tag{4.22}$$

where $U \in \mathbf{R}^{m\times m}$ and $V \in \mathbf{R}^{n\times n}$ are orthogonal, and $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_n)$ is diagonal, with

$$\sigma_1 \ge \sigma_2 \ge \cdots \ge \sigma_n > 0$$

being the singular values of A.

- *Partition the vector $U^T b$ into*

$$\begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = U^T b, \tag{4.23}$$

where $b_1 \in \mathbf{R}^n$ and $b_2 \in \mathbf{R}^{m-n}$.

- *Introduce the secular function*

$$\mathcal{G}(\alpha) = b_1^T \left(\Sigma^2 - \eta^2 I\right) \left(\Sigma^2 - \alpha I\right)^{-2} b_1 - \frac{\eta^2}{\alpha^2} \|b_2\|_2^2. \tag{4.24}$$

- *Determine the unique positive root $\hat{\alpha}$ of $\mathcal{G}(\alpha)$ that lies in the interval (η^2, σ_n^2). If it does not exist, then set $\hat{\alpha} = \sigma_n^2$.*
- *Then*
  1. *If $\hat{\alpha} < \sigma_n^2$, the solution $\hat{x}$ is unique and is given by* (4.6) *or, equivalently,*

$$\hat{x} = (A^T A - \hat{\alpha} I)^{-1} A^T b.$$

  2. *If $\hat{\alpha} = \sigma_n^2$ and $\sigma_n < \sigma_{n-1}$, then two solutions exist that are given by* (4.15)*. Otherwise, if A has multiple singular values at σ_n, then multiple solutions exist and we can use the same technique that led to* (4.15) *to determine $\hat{x}$ as explained in the above section on multiple singular values.*

We can be more explicit about the uniqueness of solutions. Assume A has multiple singular values at σ_n and let U_n denote the matrix with singular vectors that spans the left singular subspace of A associated with these singular values:

1. *When $\|U_n^T b\| \neq 0$, the solution $\hat{x}$ is unique and it corresponds to a root $\hat{\alpha} < \sigma_n^2$ as shown above.*
2. *When $\|U_n^T b\| = 0$, then either an $\hat{\alpha}_1 < \sigma_n^2$ exists and the solution $\hat{x}$ is unique. Otherwise, $\hat{\alpha} = \sigma_n^2$ and multiple solutions $\hat{x}$ exist.*

5. Restricted perturbations.

We have so far considered the case in which all the columns of the A matrix are subject to perturbations. It may happen in practice, however, that only selected columns are uncertain, while the remaining columns are known precisely. This situation can be handled by the approach of this paper as we now clarify.

Given $A \in \mathbf{R}^{m \times n}$, we partition it into block columns,

$$A = \begin{bmatrix} A_1 & A_2 \end{bmatrix},$$

and assume, without loss of generality, that only the columns of A_2 are subject to perturbations while the columns of A_1 are known exactly. We then pose the following problem.

Given $A \in \mathbf{R}^{m \times n}$, with $m \geq n$ and A full rank, $b \in \mathbf{R}^m$, and a nonnegative real number η_2, determine $\hat{x}$ such that

$$\min_{\hat{x}} \min \left\{ \| \begin{bmatrix} A_1 & A_2 + \delta A_2 \end{bmatrix} \hat{x} - b \|_2 : \ \|\delta A_2\|_2 \leq \eta_2 \right\}. \tag{5.1}$$

If we partition $\hat{x}$ accordingly with A_1 and A_2, say,

$$\hat{x} = \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \end{bmatrix},$$

then we can write

$$\| \begin{bmatrix} A_1 & A_2 + \delta A_2 \end{bmatrix} \hat{x} - b \|_2 \ = \ \|(A_2 + \delta A_2)\hat{x}_2 - (b - A_1 \hat{x}_1)\|_2.$$

Assuming, for any vector $(\hat{x}_1, \hat{x}_2)$, the fundamental condition

$$\eta_2 \, \|\hat{x}_2\|_2 \; < \; \|A_2\hat{x}_2 - (b - A_1\hat{x}_1)\|_2 \; = \|Ax - b\|_2,$$

we can follow the argument at the beginning of section 3 to conclude that the minimum over δA_2 is achievable and is equal to

$$\|A_2\hat{x}_2 - (b - A_1\hat{x}_1)\|_2 \; - \; \eta_2\|\hat{x}_2\|_2.$$

In this way, statement (5.1) reduces to the minimization problem

$$\min_{\hat{x}_1,\hat{x}_2} \left(\left\| \begin{bmatrix} A_1 & A_2 \end{bmatrix} \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \end{bmatrix} - b \right\|_2 \; - \; \eta_2\|\hat{x}_2\|_2 \right). \tag{5.2}$$

This statement can be further reduced to the problem treated in Theorem 4.2 as follows. Introduce the QR decomposition of A, say

$$A = QR \; = \; Q \begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \\ 0 & 0 \end{bmatrix},$$

where we have partitioned R accordingly with the sizes of A_1 and A_2. Define

$$\begin{bmatrix} \bar{b}_{1A} \\ \bar{b}_{2A} \\ \bar{b}_2 \end{bmatrix} \; = \; Q^T b.$$

Then (5.2) is equivalent to

$$\min_{\hat{x}_1,\hat{x}_2} \left(\left\| \begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \end{bmatrix} - \begin{bmatrix} \bar{b}_{1A} \\ \bar{b}_{2A} \\ \bar{b}_2 \end{bmatrix} \right\|_2 \; - \; \eta_2\|\hat{x}_2\|_2 \right), \tag{5.3}$$

which can be further rewritten as

$$\min_{\hat{x}_1,\hat{x}_2} \left(\left\| \begin{bmatrix} R_{11}\hat{x}_1 + R_{12}\hat{x}_2 - \bar{b}_{1A} \\ R_{22}\hat{x}_2 - \bar{b}_{2A} \\ \bar{b}_2 \end{bmatrix} \right\|_2 \; - \; \eta_2\|\hat{x}_2\|_2 \right). \tag{5.4}$$

This shows that once the optimal $\hat{x}_2$ has been determined, the optimal choice for $\hat{x}_1$ is necessarily the one that annihilates the entry $R_{11}\hat{x}_1 + R_{12}\hat{x}_2 - \bar{b}_{1A}$. That is,

$$\hat{x}_1 = R_{11}^{-1} \left[\bar{b}_{1A} - R_{12}\hat{x}_2 \right]. \tag{5.5}$$

The optimal $\hat{x}_2$ is the solution of

$$\min_{\hat{x}_2} \left(\left\| \begin{bmatrix} R_{22} \\ 0 \end{bmatrix} \hat{x}_2 - \begin{bmatrix} \bar{b}_{2A} \\ \bar{b}_2 \end{bmatrix} \right\|_2 \; - \; \eta_2\|\hat{x}_2\|_2 \right). \tag{5.6}$$

This optimization is of the same form as the problem stated earlier in (3.4) with $\hat{x}$ replaced by $\hat{x}_2$, η replaced by η_2, A replaced by $\left[\begin{smallmatrix} R_{22} \\ 0 \end{smallmatrix} \right]$, and b replaced by $\left[\begin{smallmatrix} \bar{b}_{2A} \\ \bar{b}_2 \end{smallmatrix} \right]$.

Therefore, the optimal $\hat{x}_2$ can be obtained by applying the result of Theorem 4.2. Once $\hat{x}_2$ has been determined, the corresponding $\hat{x}_1$ follows from (5.5).

6. Conclusion. In this paper we have proposed and solved a new optimization problem for parameter estimation in the presence of data uncertainties. The problem incorporates a priori bounds on the size of the perturbations. It has a "closed" form solution that is obtained by solving an "indefinite" regularized least-squares problem with a regression parameter that is determined from the positive root of a secular equation.

Several extensions are possible. For example, weighted versions with uncertainties in the weight matrices are useful in several applications, as well as cases with multiplicative uncertainties and applications to filtering theory. Some of these cases, in addition to more discussion on estimation and control problems with bounded uncertainties, can be found in [11, 12, 13, 14].

Acknowledgment. The authors would like to thank one of the anonymous reviewers for pointing out a mistake in an earlier version of the paper.

REFERENCES

[1] G. H. Golub, *Some modified matrix eigenvalue problems*, SIAM Rev., 15 (1973), pp. 318–344.

[2] 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.

[3] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed., The Johns Hopkins University Press, Baltimore, MD, 1996.

[4] S. Van Huffel and J. Vandewalle, *The Total Least Squares Problem: Computational Aspects and Analysis*, SIAM, Philadelphia, PA, 1991.

[5] L. Ljung and T. Söderström, *Theory and Practice of Recursive Identification*, MIT Press, Cambridge, MA, 1983.

[6] J. C. Doyle, K. Glover, P. Khargonekar, and B. Francis, *State-space solutions to standard H_2 and H_∞ control problems*, IEEE Trans. Automat. Control, 34 (1989), pp. 831–847.

[7] P. Khargonekar and K. M. Nagpal, *Filtering and smoothing in an $H^\infty-$ setting*, IEEE Trans. Automat. Control, AC-36 (1991), pp. 151–166.

[8] B. Hassibi, A. H. Sayed, and T. Kailath, *Recursive linear estimation in Krein spaces—Part* I: *Theory*, IEEE Trans. Automat. Control, 41 (1996), pp. 18–33.

[9] A. H. Sayed, B. Hassibi, and T. Kailath, *Inertia conditions for the minimization of quadratic forms in indefinite metric spaces*, in Recent Developments in Operator Theory and Its Applications, Oper. Theory Adv. Appl. 87, I. Gohberg, P. Lancaster, and P. N. Shivakumar, eds., Birkhauser, Basel, Switzerland, 1996, pp. 309–347.

[10] S. Chandrasekaran, G. H. Golub, M. Gu, and A. H. Sayed, *Parameter estimation in the presence of bounded data uncertainties*, SIAM J. Matrix Anal. Appl., 19 (1998), pp. 235–252.

[11] A. H. Sayed, V. H. Nascimento, and S. Chandrasekaran, *Estimation and control in the presence of bounded data uncertainties*, Linear Algebra Appl., 284 (1998), pp. 259–306.

[12] S. Chandrasekaran, G. Golub, M. Gu, and A. H. Sayed, *Parameter estimation in the presence of bounded modeling errors*, IEEE Signal Processing Lett., 4 (1997), pp. 195–197.

[13] A. H. Sayed and S. Chandrasekaran, *Estimation in the presence of multiple sources of uncertainties with applications*, in Proceedings of the Asilomar Conference, Pacific Grove, CA, 1998, pp. 1811–1815.

[14] A. H. Sayed and V. H. Nascimento, *Design criteria for uncertain models with structured and unstructured uncertainties*, Robustness in Identification and Control, A. Garulli, A. Tesi, and A. Vicino, eds., Springer-Verlag, to appear.

SIAM J. MATRIX ANAL. APPL.
Vol. 20, No. 4, pp. 860–870

CUTPOINT DECOUPLING AND FIRST PASSAGE TIMES FOR RANDOM WALKS ON GRAPHS*

STEPHEN J. KIRKLAND† AND MICHAEL NEUMANN‡

Abstract. One approach to the computations for Markov chains, due to Meyer, is to break a problem down into corresponding computations for several related chains involving a smaller number of states. In this spirit, we focus on the mean first passage matrix associated with a random walk on a connected graph, and consider the problem of transforming the computation of that matrix into smaller tasks. We show that this is possible when there is a cutpoint in the graph and provide an explicit formula for the mean first passage matrix when this is the case.

Key words. Markov chains, first passage times, random walks, cutpoint graphs

AMS subject classifications. 15A51, 60J15, 15A48

PII. S0895479897318800

1. Introduction.

For a regular Markov chain, it is well known that the stationary distribution vector provides the long-term probability of the chain being in a particular state. However, for understanding the short-term behavior of the chain, a better and more meaningful measurement can be the *mean first passage time from state i to state j*, $e_{i,j}$, which is the expected number of steps taken to arrive for the first time in state j, given that the chain was initially in state i.

In this paper we consider random walks on a connected, undirected graph and discuss the problem of computing the corresponding mean first passage matrix E, that is, the matrix whose (i,j)th entry is $e_{i,j}$. We comment that random walks on undirected graphs occur in such applications as Markov chains on hypercubes, which are a type of *nearest-neighbor random walk*, and we refer to Karlin, Lindqvist, and Yao [7] and the references cited therein for quite a comprehensive description of such walks. Here we show that for such Markov chains the presence of cutpoints in the graph can be exploited to simplify the computation of the mean first passage matrix. This leads to a divide-and-conquer approach to the computation of the matrix in the case that the underlying graph is a tree. In this case, the computation of E can be broken into many smaller tasks of the same type which can be computed in parallel.

Suppose $T \in \mathbb{R}^{n,n}$ is nonnegative, irreducible, and row stochastic so that it is the transition matrix of an ergodic Markov chain. Let $Q = I - T$ and let $w = (w_1 \dots w_n)^T$ be the stationary distribution vector for the chain, viz., $w^T T = w^T$ and $\sum_{i=1}^n w_i = 1$. Then, according to Meyer [11], the mean first passage matrix E is given by

$$E \;=\; \left[I - Q^{\#} + JQ_d^{\#}\right] W^{-1}, \tag{1.1}$$

where $Q^{\#}$ is the *group* (*generalized*) *inverse of* Q, J is the $n \times n$ matrix of all ones, where for $Y \in \mathbb{R}^{n,n}$, Y_d denotes the $n \times n$ diagonal matrix whose diagonal entries are the corresponding diagonal entries of Y, and where W is the diagonal matrix

*Received by the editors March 17, 1997; accepted for publication (in revised form) by C. Meyer May 15, 1998; published electronically July 9, 1999.

http://www.siam.org/journals/simax/20-4/31880.html

†Department of Mathematics and Statistics, University of Regina, Regina, SK, Canada S4S 0A2. The research of this author was supported by NSERC grant OGP0138251.

‡Department of Mathematics, University of Connecticut, Storrs, CT 06269-3009 (neumann@math.uconn.edu). The research of this author was supported by NSF grant DMS–9306357.

whose diagonal entries are the corresponding entries of w. Considering (1.1), the two expensive items which are necessary to compute E are $Q^{\#}$ and w. (For methods for computing $Q^{\#}$ see Anstreicher and Rothblum [1] and Hartwig [6].)

The expense involved in the computation of w and $Q^{\#}$ serves in part as the motivation for our work here as we seek ways of reducing the cost of obtaining E. Our approach is similar in spirit to, and inspired by, the work of Meyer [12], who showed that the process of Perron complementation leads to the uncoupling of the chain so that the computation of w can be done by aggregating stationary distribution vectors of smaller chains.

To further introduce our results we need some terminology for undirected graphs. Suppose $\mathcal{G}$ is an undirected, loopless, connected graph on n vertices conveniently labeled $1, \ldots, n$. Let $A = A(\mathcal{G})$ be its adjacency matrix and let $D = D(\mathcal{G})$ be the diagonal matrix whose diagonal entries $d_1, \ldots, d_n$ are the corresponding vertex degrees, where by the *degree of a vertex* we mean the number of edges incident to it. The matrix $L = L(\mathcal{G}) = D - A$ is known as the *Laplacian* of $\mathcal{G}$. Note that L has zero row sums and nonpositive off-diagonal entries. It follows from Berman and Plemmons [3, Chap. 6] that L is a singular and irreducible M-matrix and that its proper principal submatrices of all orders are nonsingular. Letting $A(i,i)$ denote the principal submatrix of A of order $n-1$, obtained by deleting the ith row and column of A, we call $(A(i,i))^{-1}$ the *bottleneck matrix of* $\mathcal{G}$ *based at vertex* i. In the case that $\mathcal{G}$ is a tree, so that there is only one path between any two vertices, the (k,ℓ)th entry of $(A(i,i))^{-1}$ is the number of edges jointly on the path from k to i and on the path from ℓ to i (see [10] for a proof).

Consider now an undirected graph on vertices $1, \ldots, n$. We can associate a random walk with $\mathcal{G}$ by specifying that the probability of staying at vertex i is θ_i (which we always assume to be strictly less than 1), while the probability of moving to any of the vertices adjacent to i is $(1-\theta_i)/d_i$. Letting $P = \operatorname{diag}(\theta_1, \ldots, \theta_n)$, we find that the transition matrix for the random walk is given by

$$T = P + (I-P)D^{-1}A. \tag{1.2}$$

Observe that

$$I - T = (I-P)D^{-1}(D-A),$$

in which $D-A$ is the Laplacian matrix of the unweighted graph $\mathcal{G}$. In an earlier paper, [10] (but see also [9]), a formula for the group inverse of a Laplacian was developed in terms of the bottleneck matrix for a graph. Bottleneck matrices will also be a basic tool in the development of the results in this paper.

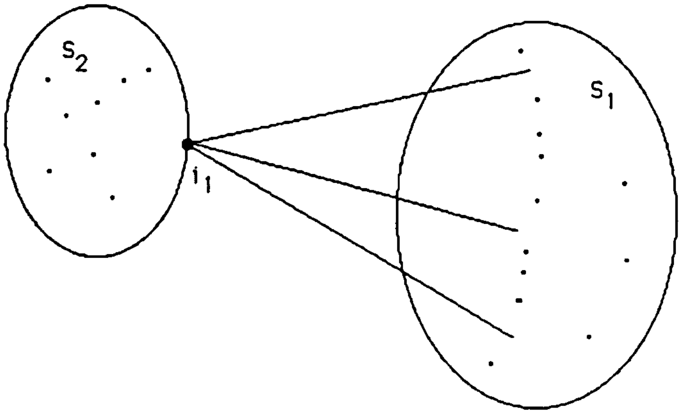
The plan of our paper is as follows. In section 2 we shall obtain several results in which the mean first passage matrix is computed for a random walk induced by an undirected connected graph which has a cutpoint. The cutpoint leads us to consider connected subgraphs associated with the connected components at the cutpoint. The portion of E corresponding to the states in each of the connected components can then proceed independently on each of these connected components, provided only that we adjust the probability of staying at the cutpoint. We find formulas for these modified probabilities and show how all of E can be computed from the connected components.

We devote section 3 to an example of obtaining the mean first passage matrix for a random walk associated with a tree on eight vertices. We illustrate how our results of section 2 can be exploited to reduce the number of computations necessary to obtain the matrix. We close the section with some suggestions on ways to implement the computation in parallel.

It follows from (1.1) that the mean first passage time from a state to itself is given by the reciprocal of the corresponding entry of the stationary distribution vector. Thus, our results in section 2 are directed toward computing the mean first passage times between different states. However, if one does wish to obtain the mean first passage between states and themselves, we exhibit in (2.3) that for random walks on graphs this task is straightforward.

2. Mean first passage times for a graph with a cutpoint. Recall that a vertex v of a connected graph $\mathcal{G}$ is a *cutpoint* if the graph $\mathcal{G} \setminus v$, obtained by deleting v and all edges incident with v, has two or more connected components. In this section we consider simplifications in computing the mean first passage matrix for a graph with a cutpoint. For such a graph, we have the following lemma concerning the bottleneck matrix.

Lemma 2.1. *Let $\mathcal{G}$ be a connected graph and partition its vertex set as $S_1 \cup S_2$ with $|S_1| = k$ and $|S_2| = n-k > 1$. Label the vertices of S_2 as $i_1, \ldots, i_{n-k}$. Let j be the number of edges from i_1 to S_1, say i_1 is adjacent to $\ell_1, \ldots, \ell_j \in S_1$, and suppose that any path from S_1 to S_2 goes through i_1: viz.*



Let B be the bottleneck matrix for $\mathcal{G}$ with destination vertex i_{n-k}. Then B can be written as

$$B = \left[\begin{array}{c|c} M_1 + \left(e_1^T M_2 e_1\right) J & \mathbf{1} e_1^T M_2 \\ \hline M_2 e_1 \mathbf{1}^T & M_2 \end{array}\right], \tag{2.1}$$

where M_2 is the bottleneck matrix for the subgraph induced by S_2 with destination vertex i_{n-k} and where

$$M_1 = \left(L_1 + \sum_{m=1}^{j} e_{\ell_m} e_{\ell_m}^T\right)^{-1} \tag{2.2}$$

with L_1 as the Laplacian for the graph induced by S_1.

Proof. We know that B is the inverse of the Laplacian of $\mathcal{G}$ with row i_{n-k} and column i_{n-k} deleted. That is,

$$B^{-1} = \left[\begin{array}{c|c} L_1 + \sum_{m=1}^{j} e_{\ell_m} e_{\ell_m}^T & -\left(\sum_{m=1}^{j} e_{\ell_m}\right) e_1^T \\ \hline -e_1 \left(\sum_{m=1}^{j} e_{\ell_m}\right)^T & M_2^{-1} + j e_1 e_1^T \end{array}\right].$$

From the fact that

$$M_1^{-1}\mathbf{1} = \sum_{m=1}^{j} e_{\ell_m},$$

we find that

$$M_1 \sum_{m=1}^{j} e_{\ell_m} = \mathbf{1}.$$

Using this last relation, a straightforward computation now shows that the product of the matrix in (2.1) and B^{-1} is, in fact, the identity. □

We next show how the bottleneck matrix can be used to obtain a formula for the first $n-1$ entries of the last column of the mean first passage matrix. This formula, in turn, will help facilitate much of our subsequent work. Consider a random walk on a graph $\mathcal{G}$ on vertices $1, \ldots, n$, with probability θ_j of staying at vertex j, for $1 \le j \le n$. We are interested in the mean first passage times to a vertex i which we shall take, without loss of generality, to be vertex n. Let η be given by $\sum_{i=1}^{n} d_i/(1-\theta_i)$ and let $y^T = (1/\eta)(d_1/(1-\theta_1) \ldots d_n/(1-\theta_n))$, where d_i is the degree of vertex i. Then note that

$$\begin{aligned} y^T T = \tfrac{1}{\eta}\mathbf{1}^T D(I-P)^{-1} T &= \tfrac{1}{\eta}(\mathbf{1}^T D(I-P)^{-1}P + \mathbf{1}^T D) \\ &= \tfrac{1}{\eta}\mathbf{1}^T D(I-P)^{-1} = y^T. \end{aligned} \tag{2.3}$$

Also note $y^T\mathbf{1} = 1$, showing that y is the Perron vector of T normalized so that its entries sum to 1. Thus, from the formula for the mean first passage matrix E in terms of $(I-T)^{\#}$ given in (1.1) and the partitioned formula for $(I-T)^{\#}$ given in [10, Proposition 2.2], it follows that the last column of E is given by

$$\begin{pmatrix} M\mathbf{1} \\ 1/y_n \end{pmatrix},$$

where M is the inverse of the principal submatrix of $I-T$ formed from its first $n-1$ rows and columns. It follows that $M = (\bar{D}-\bar{A})^{-1}\bar{D}(\bar{I}-\bar{P})^{-1}$, where the bar denotes truncation of the last row and column of an $n \times n$ matrix. Consequently, we find that

$$M\mathbf{1} = (\bar{D}-\bar{A})^{-1}\bar{D}(\bar{I}-\bar{P})^{-1}\mathbf{1} = B\begin{pmatrix} d_1/(1-\theta_1) \\ \vdots \\ d_{n-1}/(1-\theta_{n-1}) \end{pmatrix}, \tag{2.4}$$

where B is the bottleneck matrix for the graph $\mathcal{G}$ based at vertex n.

With the graph in the above lemma and its associated random walk, we now show that the mean first passage times from vertices in S_1 to vertices in S_2 admit an additive representation, with the terms in the summation being largely determined by one of the major parts of the graph.

THEOREM 2.2. *Let $\mathcal{G}$ be as in the statement of Lemma 2.1. For $1 \le i \le n$, let θ_i be the probability of staying at vertex i and consider the corresponding random walk on $\mathcal{G}$. If $\ell \in S_1$, then the mean first passage time from ℓ to i_{n-k} is the sum of the mean first passage time from ℓ to i_1 and the mean first passage time from i_1 to i_{n-k}.*

Proof. Let u be the $(n-1)$-vector whose entries are the mean first passage times from vertex $v \ne i_{n-k}$ to vertex i_{n-k}. Then, by (2.4), u can be written as

$$u = B \begin{bmatrix} \delta_1/(1-\theta_1) \\ \vdots \\ \delta_k/(1-\theta_k) \\ d_{i_1}/(1-\theta_{i_1}) \\ \vdots \\ d_{i_{n-k-1}}/(1-\theta_{i_{n-k-1}}) \end{bmatrix}, \tag{2.5}$$

where δ_i is the degree of vertex $i \in S_1$ and d_{i_m} is the degree of vertex $i_m \in S_2$. From Lemma 2.1 we further see that u is given by

$$\left(\frac{M_1 \begin{bmatrix} \delta_1/(1-\theta_1) \\ \vdots \\ \delta_k/(1-\theta_k) \end{bmatrix} + \left(e_1^T M_2 e_1\right) \left[\sum_{i=1}^k \delta_i/(1-\theta_i)\right] \mathbf{1} + \mathbf{1} e_1^T M_2 \begin{bmatrix} d_{i_1}/(1-\theta_{i_1}) \\ \vdots \\ d_{i_{n-k-1}}/(1-\theta_{i_{n-k-1}}) \end{bmatrix}}{\sum_{i=1}^k \delta_i/(1-\theta_i) M_2 e_1 + M_2 \begin{bmatrix} d_{i_1}/(1-\theta_{i_1}) \\ \vdots \\ d_{i_{n-k-1}}/(1-\theta_{i_{n-k-1}}) \end{bmatrix}} \right).$$

In particular, we see that the mean first passage time from i_1 to i_{n-k} is equal to

$$\left(e_1^T M_2 e_1\right) \left[\sum_{i=1}^k \delta_i/(1-\theta_i)\right] + e_1^T M_2 \begin{bmatrix} d_{i_1}/(1-\theta_{i_1}) \\ \vdots \\ d_{i_{n-k-1}}/(1-\theta_{i_{n-k-1}}) \end{bmatrix}. \tag{2.6}$$

Furthermore, the column $\hat{u}$ of the mean first passage times from vertices $v \ne i_1$ to vertex i_1 can be written as

$$\hat{B} \begin{bmatrix} \delta_1/(1-\theta_1) \\ \vdots \\ \delta_k/(1-\theta_k) \\ d_{i_2}/(1-\theta_{i_2}) \\ \vdots \\ d_{i_{n-k}}/(1-\theta_{i_{n-k}}) \end{bmatrix},$$

where $\hat{B}$ is the bottleneck matrix for $\mathcal{G}$ with destination vertex i_1. From this it follows that

$$\hat{u} = \left(\begin{array}{c} M_1 \begin{bmatrix} \delta_1/(1-\theta_1) \\ \vdots \\ \delta_k/(1-\theta_k) \end{bmatrix} \\ \hline \\ x \end{array} \right)$$

for some vector x, so that the mean first passage time from ℓ to i_1 is given by

$$e_\ell^T M_1 \begin{bmatrix} \delta_1/(1-\theta_1) \\ \vdots \\ \delta_k/(1-\theta_k) \end{bmatrix}.$$

The result now follows. □

In view of the above theorem, it is natural to ask whether we can compute the mean first passage times for any vertex in S_2 (other than i_{n-k}) to i_{n-k} just by considering S_2 alone. It turns out that with a modification of the probability of staying at vertex i_1, this is indeed the case, as is shown in our next theorem.

THEOREM 2.3. *Let $\mathcal{G}$ be as in Lemma 2.1. If $i_j \in S_2$ and $j \neq n-k$, then the mean first passage time from i_j to i_{n-k} is the same as that in the random walk on the subgraph of $\mathcal{G}$ induced by S_2, where the probability of staying at vertex i_j is θ_{i_j}, $2 \leq j \leq n-k$, and the probability of staying at vertex i_1 is*

$$\hat{\theta}_{i_1} = \frac{\sum_{i=1}^{k} \delta_i/(1-\theta_i) + \theta_{i_1} d_{i_1}/(1-\theta_{i_1}) + (d_{i_1} - \sigma_{i_1})}{\sum_{i=1}^{k} \delta_i/(1-\theta_i) + d_{i_1}/(1-\theta_{i_1})}, \tag{2.7}$$

where σ_{i_1} is the degree of i_1 in the subgraph induced by S_2.

Proof. From Lemma 2.1, for $1 \leq j < n-k$, the mean first passage time from i_j to i_{n-k} is the corresponding entry of (2.6). Now in the modified random walk on the subgraph induced by S_2, our mean first passage times are the elements of

$$M_2 \begin{bmatrix} \sigma_{i_1}/(1-\hat{\theta}_{i_1}) \\ d_{i_2}/(1-\theta_{i_2}) \\ \vdots \\ d_{i_{n-k-1}}/(1-\theta_{i_{n-k-1}}) \end{bmatrix}.$$

Observing that

$$\begin{bmatrix} \sigma_{i_1}/(1-\hat{\theta}_{i_1}) \\ d_{i_2}/(1-\theta_{i_2}) \\ \vdots \\ d_{i_{n-k-1}}/(1-\theta_{i_{n-k-1}}) \end{bmatrix} = \begin{bmatrix} d_{i_1}/(1-\theta_{i_1}) \\ \vdots \\ d_{i_{n-k-1}}/(1-\theta_{i_{n-k-1}}) \end{bmatrix} + \left[\sum_{i=1}^{k} \delta_i/(1-\theta_i) \right] e_1$$

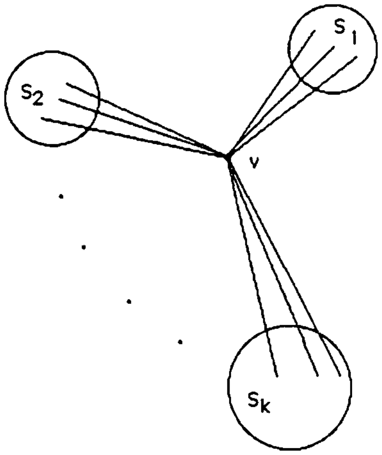
now yields the result. □

Since i_{n-k} represented an arbitrary vertex in vertex in $S_2 \setminus i_1$, we now have the following corollary.

COROLLARY 2.4. *The principal submatrix of the mean first passage matrix for $\mathcal{G}$ corresponding to the vertices in S_2 has the same off-diagonal parts as the mean* *first passage matrix for the random walk on the subgraph of $\mathcal{G}$ induced by S_2, with the probability $\hat{\theta}_{i_1}$ of staying at vertex i_1.*

For a random walk on a graph $\mathcal{G}$ with a cutpoint v, our next result shows how to express the mean first passage times for $\mathcal{G}$ in terms of mean first passage times for related random walks on the connected components of $\mathcal{G} \setminus v$.

Theorem 2.5. *Suppose that $\mathcal{G}$ is a graph with vertex v as a cutpoint. Let S_i, $1 \le i \le k$ denote subsets of vertices which induce the connected components of $\mathcal{G} \setminus v$, viz.*



Let d_j denote the degree of vertex j and let θ_j be the probability of staying at vertex j for each $j \in G$. Then the off-diagonal entries of the mean first passage matrix for $\mathcal{G}$ agree with the corresponding entries of

$$\left[\begin{array}{c|c|c|c|c} A_1 & a_1\mathbf{1}^T + \mathbf{1}b_2^T & \cdots\cdots & a_1\mathbf{1}^T + \mathbf{1}b_k^T & a_1 \\ \hline a_2\mathbf{1}^T + \mathbf{1}b_1^T & A_2 & \cdots\cdots & a_2\mathbf{1}^T + \mathbf{1}b_k^T & a_2 \\ \hline \vdots & \vdots & \ddots & \vdots & \vdots \\ \hline a_k\mathbf{1}^T + \mathbf{1}b_1^T & a_k\mathbf{1}^T + \mathbf{1}b_2^T & \cdots\cdots & A_k & a_k \\ \hline b_1^t & b_2^T & \cdots\cdots & b_k^T & * \end{array}\right], \tag{2.8}$$

where

$$\left[\begin{array}{c|c} A_i & a_i \\ \hline b_i^T & * \end{array}\right] \tag{2.9}$$

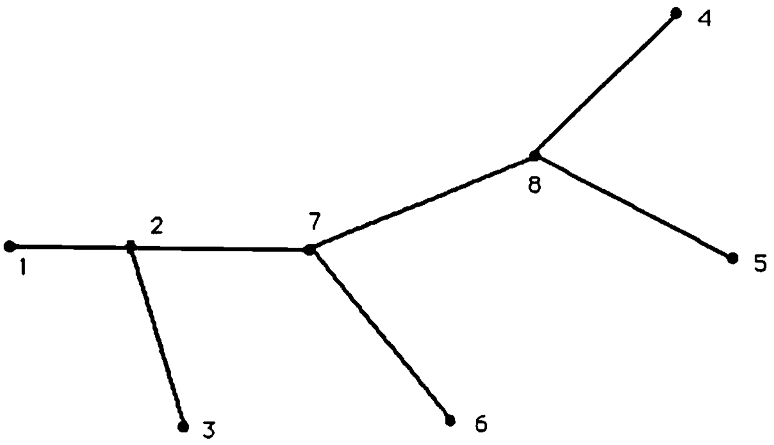
is the mean first passage matrix for the random walk on the subgraph induced by $S_i \cup \{v\}$ *with the probability*

$$\hat{\theta}_v(i) = \frac{\sum_{j \notin S_i \cup \{v\}} d_j/(1-\theta_j) + \theta_v d_v/(1-\theta_v) + (d_v - \tau_i)}{\sum_{j \notin S_i \cup \{v\}} d_j/(1-\theta_j) + d_v/(1-\theta_v)} \tag{2.10}$$

of staying at vertex v*, where* τ_i *is the degree of* v *in the subgraph induced by* $S_i \cup \{v\}$*.*

Proof. The formula for (2.9) follows from Corollary 2.4 when it is applied to $S_i \cup \{v\}$. If $\ell_1 \in S_i$ and $\ell_2 \in S_j$, $i \neq j$, then the mean first passage time from ℓ_1 to ℓ_2 is equal to the sum of the mean first passage time from ℓ_1 to v and the mean first passage time from v to ℓ_2, yielding the formula for the off-diagonal blocks of the mean first passage matrix. □

3. An example and computational remarks. As an example to illustrate our results in section 2, consider the following tree $\mathcal{G}$ on 8 vertices:



Taking the probabilities of staying at each vertex to be 0, we find that $\mathcal{G}$ induces a random walk whose transition matrix is given by

$$T = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/3 & 0 & 1/3 & 0 & 0 & 0 & 1/3 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1/3 & 0 & 0 & 0 & 1/3 & 0 & 1/3 \\ 0 & 0 & 0 & 1/3 & 1/3 & 0 & 1/3 & 0 \end{bmatrix}.$$

On computing E using Meyer's formula in (1.1) we find that

$$E = \left[\begin{array}{ccc|ccccc} 14 & 1 & 14 & 28 & 28 & 19 & 6 & 15 \\ 13 & 4\frac{2}{3} & 13 & 27 & 27 & 18 & 5 & 14 \\ 14 & 1 & 14 & 28 & 28 & 19 & 6 & 15 \\ \hline 28 & 15 & 28 & 14 & 14 & 19 & 6 & 1 \\ 28 & 15 & 28 & 14 & 14 & 19 & 6 & 1 \\ 23 & 10 & 23 & 23 & 23 & 14 & 1 & 10 \\ 22 & 9 & 22 & 22 & 22 & 13 & 4\frac{2}{3} & 9 \\ 27 & 14 & 27 & 13 & 13 & 18 & 5 & 4\frac{2}{3} \end{array}\right]. \tag{3.1}$$

Applying Theorem 2.3, let us now partition the vertex set into two subsets, $S_1 = \{1,2,3\}$ and $S_2 = \{4,5,6,7,8\}$, and regard $i_1 = 7$ and $i_5 = 8$. It is easily checked that formula (2.9) gives that

$$\hat{\theta}_1 = \frac{5+1}{5+3} = \frac{3}{4}.$$

Thus the transition matrix for the random walk on the subgraph of $\mathcal{G}$ induced by S_2 is given by

$$T_{S_2} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{8} & \frac{3}{4} & \frac{1}{8} \\ \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} & 0 \end{bmatrix}.$$

Using Meyer's formula in (1.1) once again (but now to compute the mean first passage matrix for the random walk whose transition matrix is given by T_{S_2}), we obtain:

$$E_{S_2} = \begin{bmatrix} 14 & 14 & 19 & 6 & 1 \\ 14 & 14 & 19 & 6 & 1 \\ 23 & 23 & 14 & 1 & 10 \\ 22 & 22 & 13 & 1\frac{3}{4} & 9 \\ 13 & 13 & 18 & 5 & 4\frac{2}{3} \end{bmatrix}. \tag{3.2}$$

Using the *flops* (floating point operations) counter in MATLAB, we have found that it takes 23,737 flops to compute E, but only 5,900 to compute E_{S_2}. This is a clear computational advantage if we are only interested in the mean first passage times within the states represented by S_2. We should mention that in computing (3.1) and (3.2) we did *not* compute the group inverse of a matrix by the most efficient method, as required in Meyer's formula, (1.1), as we were merely interested in the comparison of the number of flops for obtaining the two matrices.

Actually, let us perform a much more efficient computation of the entries of E by using (2.4) to compute entries 4–7 in the last column of E. This means having to solve a linear system where the coefficient matrix is the inverse of the appropriate bottleneck matrix. If we obtain these entries as a part of computing the entire column but the last entry, then the number of flops required is 980, whereas if we modify the probability of staying at vertex 7 from 0 to the ratio given by (2.7) and then compute the entire final column except the last entry of the mean first passage for the random walk of the subgraph of $\mathcal{G}$ induced by the vertex set S_2, then the number of flops drops to 210. We see that the ratio of the number of flops required to compute these last 4 entries did not deteriorate compared to the previous experiment in which we used Meyer's formula, but instead improved.

As an illustration of the utility of Theorem 2.5, we also computed E of (3.1) using the following procedure: We considered the branches at vertex 7 and used (1.1) to compute the mean first passage matrices for the modified random walks induced by the vertex sets $\{1, 2, 3, 7\}$, $\{4, 5, 8, 7\}$, and $\{6, 7\}$. We then applied Theorem 2.5 to compute E. The entire procedure used 6,405 flops, a significant improvement on the 23,737 flops necessary to compute E directly from (1.1).

We now discuss some strategies for computing E for a random walk on a tree. In order to make the computations efficient, we should look for an advantageous vertex in the tree and break the tree into branches at that vertex. For example, if we choose a vertex of high degree and then compute the mean first passage matrix for the modified random walk associated with each of its branches as described in Theorem 2.3, we can then compute all other off-diagonal entries of the mean first passage matrix using, for example, Theorem 2.5. Actually, once we have found a good vertex to partition the tree into branches, we may then continue the process of breaking the tree into small branches by seeking "good" vertices within each branch to split the branches themselves until we have reduced the problem to that of computing mean first passage matrices for small branches. Evidently much of the computation can proceed in parallel with this approach.

Another strategy for breaking up the tree might be to begin by searching for a *centroid* of the tree. This is a vertex v such that each branch at v contains at most $n/2$ vertices; there are at most two centroids in a tree, and if there are indeed two, they are necessarily adjacent. Having found a centroid, we can then break the branches away from the centroid and repeat the process within each branch until the initial tree has been split into sufficiently small branches. Computation of the mean first passage matrix can then be built up from the corresponding matrices on the small branches. We will investigate the computational efficiency of this and other suggestions in a separate report.

Acknowledgments. The authors would like to thank the referees for their careful reading of the original manuscript and for their suggestions.

REFERENCES

[1] K. M. Anstreicher and U. G. Rothblum, *Using Gauss–Jordan elimination to compute the index, generalized nullspace, and Drazin inverses*, Linear Algebra Appl., 85 (1987), pp. 221–239.

[2] A. Ben-Israel and T. N. Greville, *Generalized Inverses: Theory and Applications*, Academic Press, New York, 1973.

[3] A. Berman and R. J. Plemmons, *Nonnegative Matrices in the Mathematical Sciences*, SIAM, Philadelphia, 1994.

[4] S. L. Campbell and C. D. Meyer, Jr., *Generalized Inverses of Linear Transformations*, Dover Publications, New York, 1991.

[5] W. Feller, *An Introduction to Probability Theory and its Applications*, Vol. I, 2nd ed., John Wiley & Sons, New York, 1959.

[6] R. E. Hartwig, *A method for computing A^d*, Math. Japon., 26 (1981), pp. 37–43.

[7] S. Karlin, B. Lindqvist, and Y. Yao, *Markov chains on hypercubes: Spectral representation and several majorization relations*, Random Structures Algorithms, 4 (1993), pp. 1–36.

[8] J. G. Kemeny and J. L. Snell, *Finite Markov Chains*, Van Nostrand, Princeton, NJ, 1959.

[9] S. J. Kirkland, M. Neumann, and B. Shader, *Bounds on the subdominant eigenvalue involving group inverses with applications to graphs*, Czech. Math. J., 48 (1998), pp. 1–20.

[10] S. J. Kirkland, M. Neumann, and B. Shader, *Distances in weighted trees and group inverses of Laplacian matrices*, SIAM J. Matrix Anal. Appl., 18 (1997), pp. 827–841.

[11] C. D. Meyer, Jr., *The role of the group generalized inverse in the theory of finite Markov chains*, SIAM Rev., 17 (1975), pp. 443–464.

[12] C. D. Meyer, Jr., *Uncoupling the Perron eigenvector problem*, Linear Algebra Appl., 114/115 (1989), pp. 69–94.

[13] J. W. Moon, *Random walks on random trees*, J. Austral. Math. Soc., 15 (1973), pp. 42–53.

SIAM J. MATRIX ANAL. APPL.
Vol. 20, No. 4, pp. 871–885

PARTIAL ORDERS AND THE MATRIX R IN MATRIX ANALYTIC METHODS*

QI-MING HE†

Abstract. This paper studies the matrix R, which is the minimal nonnegative solution to a nonlinear matrix equation, raised in matrix analytic methods. Based on some partial orders defined on the transition matrix of Markov chains of GI/M/1 type, the monotonicity of the corresponding matrix R and its Perron–Frobenius eigenvalue is investigated. The results are useful in estimating tail probabilities of stationary distributions of Markov chains of GI/M/1 type and constructing upper bounds for the matrix R. Applications to the GI/MAP/1 queue are discussed as well.

Key words. matrix analytic methods, queueing theory, partial order, GI/MAP/1 queue, nonnegative matrix

AMS subject classifications. 60K25, 15A24

PII. S0895479897311214

1. Introduction.

Let $\{A_n, n \geq 0\}$ be a sequence of $m \times m$ nonnegative matrices whose summation matrix A is substochastic or stochastic. $\{A_n, n \geq 0\}$ is called a substochastic or stochastic sequence accordingly. Let R, an $m \times m$ matrix, be the minimal nonnegative solution to the equation

$$X = \sum_{n=0}^{\infty} X^n A_n. \tag{1.1}$$

Based on some partial orders defined on the set of all substochastic and stochastic sequences, this paper presents some characterizations of the matrix R. Applications to Markov chain theory, matrix analytic methods, and the GI/MAP/1 queue are explored.

The interest in the minimal nonnegative solution to (1.1) comes from the study of Markov chains of GI/M/1 type (see (2.1)), which often arise in the modeling of stochastic systems. A classical example is the GI/M/1 queue in which the embedded Markov chain for the queue length at arrival epochs has the GI/M/1-type structure. More complicated examples are Markov chains of GI/M/1 type with matrix transition blocks ($\{A_n, n \geq 0\}$) raised in the study of the GI/PH/1 and GI/MAP/1 queues (Neuts [16]). The matrix R is important since the stationary distribution of a Markov chain of GI/M/1 type, when it exists, has a matrix–geometric solution which can be expressed in terms of R and another constant vector. Early papers (see Purdue [20]) studied (1.1) and its minimal nonnegative solution. It has been proved that, under some conditions, the *Perron–Frobenius eigenvalue* (the eigenvalue with the largest real part) of R is less than one so that the stationary distribution of the Markov chain exists and is unique. More recently, Gail, Hantler, and Taylor [6] found all power bounded solutions of (1.1). Their work greatly extended our understanding of the solutions to (1.1). Although previous studies gained some insights into the minimal nonnegative solution of (1.1), there are still some important issues that need to be explored (see Neuts [19]). This paper addresses two of those issues.

*Received by the editors October 7, 1997; accepted for publication (in revised form) by M. Chu June 10, 1998; published electronically July 9, 1999.
http://www.siam.org/journals/simax/20-4/31121.html

†Department of Industrial Engineering, DalTech, Dalhousie University, Halifax, NS, Canada, B3J 2X4 (Qi-Ming.He@dal.ca).

The first issue is related to the comparison of the stationary distributions of Markov chains of GI/M/1 type. Similar to stochastic comparison of Markov chains (see Daley [5] and Keilson and Kester [9]), some partial orders shall be defined on the transition blocks ($\{A_n, n \geq 0\}$) of Markov chains of GI/M/1 type. When the transition blocks of two Markov chains of GI/M/1 type are partially ordered, the relationship between their corresponding matrix R shall be investigated. Since the stationary distributions of the two Markov chains of GI/M/1 type can be expressed in terms of their matrix R, the relationship between the two stationary distributions can be investigated accordingly. For example, by introducing the stochastically larger order, a general inequality for the matrix R under this partial order shall be proved. Implications of the results, especially to the stationary distribution, are then discussed (see section 4).

The second issue is related to the computation of the matrix R. Some algorithms have been developed for computing R (see Neuts [16], Latouche and Ramaswami [10], Ramaswami [21], and Ramaswami and Taylor [22]). Essentially, these algorithms generate a sequence of matrices which, usually nondecreasing, converges to R. That is, the resulting sequence converges to R from "below." An interesting problem is to find an algorithm which generates a sequence converging to R from "above." In this paper, some upper bounds of R and some schemes for computing R are presented. The proposed schemes generate sequences of matrices that have a Perron–Frobenius eigenvalue larger than that of R, and the sequence converges to R under certain conditions. The resulting sequence may not be decreasing, but its matrices are "above" the limiting matrix R in the sense of the Perron–Frobenius eigenvalue.

In this paper, the focus is on the matrix R. It is worthwhile to mention that the approach used in this paper can be used to study a dual problem of (1.1) raised from the study of Markov chains of M/G/1 type, i.e., the minimal nonnegative solution G to the equation

$$X = \sum_{n=0}^{\infty} A_n X^n. \tag{1.2}$$

The reason is that there is a one-on-one mapping between the minimal nonnegative solutions of (1.1) and (1.2) according to Asmussen and Ramaswami [2]. More studies of the matrix G can be found in Akar and Sohraby [1], Bini and Meini [4], Latouche and Stewart [11], and Lucantoni [13], where different algorithms for computing the matrix G are proposed and analyzed.

The rest of this paper is organized as follows. In section 2, some definitions and some classical results are presented. In section 3, a partial order defined on the set of substochastic and stochastic sequences is introduced. Section 4 focuses on the stochastically larger order and some applications to Markov chains of GI/M/1 type. Section 5 presents some inequalities of R and suggests a scheme for computing R. A few numerical examples are presented in section 5 as well. Section 6 discusses the moment generating order, functional monotonicity, and functional dominance. Section 7 is devoted to the GI/MAP/1 queue.

2. Preliminaries. In this section, a discrete Markov chain of GI/M/1 type, a simple algorithm for computing R, and some classical results on nonnegative matrices are introduced.

Markov chains of GI/M/1 type. A Markov chain P is of GI/M/1 type if it has a transition matrix

$$P = \begin{pmatrix} \overline{A}_0 & A_0 & & & \\ \overline{A}_1 & A_1 & A_0 & & \\ \overline{A}_2 & A_2 & A_1 & A_0 & \\ \vdots & \ddots & \ddots & \ddots & \ddots \end{pmatrix}, \tag{2.1}$$

where all the blocks in P are $m \times m$ matrices. The state space of P is $\{(n, j), n \geq 0, 1 \leq j \leq m\}$. The state set $\{(n, j), 1 \leq j \leq m\}$ is called the level n. One of the major features of the Markov chain P is that, for each transition, the first index n can increase at most by one. This feature determines a special structure of the stationary distribution of P. Let $\boldsymbol{\pi} = (\boldsymbol{\pi}_0, \boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \ldots)$ be the stationary distribution of P, where $\boldsymbol{\pi}_n$ is an m-dimensional vector for $n \geq 0$. When P is irreducible and positive recurrent, it has been found (see Neuts [16, Chapter 1]) that $\boldsymbol{\pi}_n = \boldsymbol{\pi}_0 R^n$, $n \geq 0$, where the vector $\boldsymbol{\pi}_0$ is the unique nonnegative solution to the equations

$$\mathbf{x} = \mathbf{x} \sum_{n=0}^{\infty} R^n \overline{A}_n \quad \text{and} \quad \mathbf{x}(\mathbf{I} - R)^{-1}\mathbf{e} = 1, \tag{2.2}$$

where $\mathbf{e}$ is the column vector with all components one, and $\mathbf{I}$ is the unit matrix. The solution $\boldsymbol{\pi}$ is called the *matrix–geometric solution.* Such a solution exists for (irreducible and positive recurrent) Markov chains of GI/M/1 type with more complicated boundary statues. In the context of the Markov chain P, R_{ij} $(1 \leq i, j \leq m)$ is interpreted as the mean total number of visits to state $(n + 1, j)$ before returning to level n, given that the Markov chain started in state (n, i). It is clear from the special structure of P that R_{ij} is the same for all positive n.

Continuous-time Markov chains of GI/M/1 type are defined similarly. The infinitesimal generator of a continuous-time Markov chain of GI/M/1 type has a similar structure to that of the matrix P in (2.1). It is worth mentioning that the results obtained in this paper also hold for Markov processes of GI/M/1 type with minor changes.

An algorithm for computing $\boldsymbol{R}$. A simple algorithm for computing the matrix R is as follows. Let $R[0] = 0$ and

$$R[k+1] = \sum_{n=0}^{\infty} (R[k])^n A_n, \qquad k \geq 0. \tag{2.3}$$

It is easy to see that $\{R[k], k \geq 0\}$ is nondecreasing and converges to R from below, when R is unique. Although there are other algorithms proposed for computing R for some special cases (see Latouche and Ramaswami [10] and Ramaswami and Taylor [22]), this algorithm is easy to use when $\{A_n, n \geq 0\}$ are known. In this paper, (2.3) will be used repeatedly in proving properties about the matrix R.

Nonnegative matrices. In this paper, some results of nonnegative matrices are used repeatedly. For convenience, those results are summarized here (see Gantmacher [7]). Assume that X is an irreducible nonnegative matrix. Let $sp(X)$ denote the Perron–Frobenius eigenvalue of X. Then $sp(X)$ is positive and its corresponding eigenvector has positive elements as well. $sp(X)$ is strictly increasing with respect to each element of X. For any two constants c_1 and c_2, if $\mathbf{u}c_1 \leq \mathbf{u}X \leq \mathbf{u}c_2$, where $\mathbf{u}$ is a nonzero nonnegative vector, then $c_1 \leq sp(X) \leq c_2$. For a stochastic or substochastic sequence $\{A_n, n \geq 0\}$, define

$$A^*(z) = \sum_{n=0}^{\infty} z^n A_n, \qquad z \in [0, 1). \tag{2.4}$$

If A, the summation matrix of $\{A_n, n \geq 0\}$, is irreducible, $A^*(z)$ is irreducible for all $0 < z < 1$. Denote by $\boldsymbol{\theta}(z)$ the eigenvector corresponding to $sp(A^*(z))$ with $\boldsymbol{\theta}(z)\mathbf{e} = 1$. $\boldsymbol{\theta}(z)$ is unique and positive. Also, $\log(sp(A^*(e^{-s})))$ is convex with respect to $s(> 0)$ (see Neuts [16, Chapter 1]). This last property implies that $sp(A^*(z))$, as a function of z, has at most one intersection with the linear function z in $[0, 1)$. This further implies that the eigenvector of $sp(R)$, denoted by $\boldsymbol{\theta}(sp(R))$ with $\boldsymbol{\theta}(sp(R))\mathbf{e} = 1$, is unique and positive.

3. A partial order and the matrix *R*. Partial ordering is an important tool in characterizing stochastic systems in applied probability (see Marshall and Olkin [14] as well as Shaked and Shanthikumar [24]). Various partial orders have been introduced and studied. For example, the majorization of vectors and matrices is discussed in Marshall and Olkin [14]. In Ridder [23], functional monotonicity and functional dominance are introduced with applications to matrix analytic methods.

Let M_m be the set of all sequences $\{A_n, n \geq 0\}$ of dimension m whose summation is an irreducible substochastic or stochastic matrix. In this section, a partial order is defined on M_m in order to study the monotonicity of the matrix R. The partial order is defined in such a way that the Perron–Frobenius eigenvalue of the matrix R can play an important role. Relationships between the matrix R and its corresponding eigenvalues of partially ordered sequences shall be derived.

Define a function ϕ (from $[0, 1)$ to $(0, \infty)$) as

$$\phi(x) = \sum_{n=0}^{\infty} \phi_n x^n, \tag{3.1}$$

where $\{\phi_n, n \geq 0\}$ are nonnegative and finite, $\phi_0 = 1$, and the summation converges for all x in $[0, 1)$.

Definition 3.1. *For $\{A_n, n \geq 0\}$ and $\{B_n, n \geq 0\}$ in M_m, if*

$$\sum_{i=0}^{n} \phi_{n-i} A_i \geq \sum_{i=0}^{n} \phi_{n-i} B_i \qquad \text{for all } n \geq 0, \tag{3.2}$$

then $\{A_n, n \geq 0\}$ is called smaller than $\{B_n, n \geq 0\}$ with respect to ϕ, denoted as $\{A_n, n \geq 0\} \leq_\phi \{B_n, n \geq 0\}$. It can be verified that (3.2) *indeed defines a partial order on M_m with transitivity and reflective properties. The following result shows why the partial order $\leq_\phi$ is useful in the study of the matrix R.*

Lemma 3.1. *Consider $\{A_n, n \geq 0\}$ and $\{B_n, n \geq 0\}$ in M_m satisfying $\{A_n, n \geq 0\} \leq_\phi \{B_n, n \geq 0\}$. Let R_a and R_b be the minimal nonnegative solutions to* (1.1) *corresponding to $\{A_n, n \geq 0\}$ and $\{B_n, n \geq 0\}$, respectively. (Indexes "a" and "b" shall be used to distinguish variables corresponding to the two sequences.) Assuming that $sp(R_a) \leq 1$, $\phi(R_a)$, and $\phi(R_b)$ are well defined, then*

$$\phi(R_a)R_a^n \geq \phi(R_a)R_b^n \quad \text{and} \quad \phi(R_b)R_a^n \geq \phi(R_b)R_b^n, \quad \text{for } n \geq 1, \tag{3.3}$$

and $sp(R_a) \geq sp(R_b)$. Furthermore, denote by $\boldsymbol{\theta}_a = \boldsymbol{\theta}(sp(R_a))$ and $\boldsymbol{\theta}_b = \boldsymbol{\theta}(sp(R_b))$, for $n \geq 1$,

$$\boldsymbol{\theta}_a R_a^n = (sp(R_a))^n \boldsymbol{\theta}_a \geq \boldsymbol{\theta}_a R_b^n \quad \text{and} \quad \boldsymbol{\theta}_b R_a^n \geq \boldsymbol{\theta}_b R_b^n = (sp(R_b))^n \boldsymbol{\theta}_b. \tag{3.4}$$

Proof. To prove the first inequality in (3.3), the fact that the nondecreasing sequence generated by (2.3) converges to R_b of $\{B_n, n \geq 0\}$ shall be used. First, it has

$$\phi(R_a)R_a = \phi(R_a)\sum_{n=0}^{\infty} R_a^n A_n \geq \phi(R_a)A_0 \geq \phi(R_a)B_0 = \phi(R_a)R_b[1], \tag{3.5}$$

where $A_0 \geq B_0 = R_b[1]$ by definition (see (2.3) and (3.2)). The commutativity of $\phi(R_a)$ and R_a yields, for $n \geq 1$,

$$\phi(R_a)R_a^n = R_a^{n-1}\phi(R_a)R_a \geq R_a^{n-1}\phi(R_a)R_b[1] \geq \cdots \geq \phi(R_a)(R_b[1])^n. \tag{3.6}$$

Suppose that (3.6) is true for k. For $k+1$, by induction, it has

$$\begin{aligned}\phi(R_a)R_a &= \phi(R_a)\sum_{n=0}^{\infty} R_a^n A_n = \sum_{n=0}^{\infty}\left(\sum_{i=0}^{\infty}\phi_i R_a^i\right)R_a^n A_n = \sum_{n=0}^{\infty} R_a^n\left(\sum_{i=0}^{n}\phi_i A_{n-i}\right)\\ &\geq \sum_{n=0}^{\infty} R_a^n\left(\sum_{i=0}^{n}\phi_i B_{n-i}\right) = \sum_{n=0}^{\infty}\phi(R_a)R_a^n B_n \geq \sum_{n=0}^{\infty}\phi(R_a)(R_b[k])^n B_n\\ &= \phi(R_a)\sum_{n=0}^{\infty}(R_b[k])^n B_n = \phi(R_a)R_b[k+1].\end{aligned} \tag{3.7}$$

The exchange of the summations in (3.7) is valid since the coefficients $\{\phi_n, n \geq 0\}$ and matrices involved are nonnegative. Similar to (3.6), it can be proved that $\phi(R_a)R_a^n \geq \phi(R_a)(R_b[k+1])^n$ for $n \geq 1$. Thus, $\phi(R_a)R_a^n \geq \phi(R_a)(R_b[k])^n$ holds for $n \geq 1$ and $k \geq 1$. Since $R_b[k]$ converges to R_b monotonically, the first inequality in (3.3) is obtained. Multiplying $\boldsymbol{\theta}_a$ on both sides of the first inequality in (3.3) yields

$$\boldsymbol{\theta}_a\phi(R_a)R_a^n = \phi(sp(R_a))\boldsymbol{\theta}_a(sp(R_a))^n \geq \phi(sp(R_a))\boldsymbol{\theta}_a R_b^n. \tag{3.8}$$

Since $\phi(x)$ is positive, $\boldsymbol{\theta}_a sp(R_a) \geq \boldsymbol{\theta}_a R_b$, which implies that $sp(R_a) \geq sp(R_b)$ and the first inequality of (3.4) holds.

To prove the second inequalities in (3.3) and (3.4), define the following sequence:

$$\underline{R}_a[1] = \sum_{n=0}^{\infty} R_b^n A_n \quad \text{and} \quad \underline{R}_a[k+1] = \sum_{n=0}^{\infty}(\underline{R}_a[k])^n A_n, \quad k \geq 1. \tag{3.9}$$

It can be proved that, for $n \geq 1$,

$$\phi(R_b)R_b^n \leq \phi(R_b)(\underline{R}_a[1])^n \quad \text{and} \quad \phi(R_b)R_b^n \leq \phi(R_b)(\underline{R}_a[k])^n, \quad k \geq 1. \tag{3.10}$$

Then it is only needed to prove that the sequence $\{\underline{R}_a[k],\ k \geq 1\}$ converges to R_a. By definition and induction,

$$\begin{aligned}&\boldsymbol{\theta}_a\underline{R}_a[1] = \sum_{n=0}^{\infty}\boldsymbol{\theta}_a R_b^n A_n \leq \boldsymbol{\theta}_a\sum_{n=0}^{\infty}(sp(R_a))^n A_n = sp(R_a)\boldsymbol{\theta}_a,\\ &\boldsymbol{\theta}_a(R_a[1])^n \leq (sp(R_a))^n\boldsymbol{\theta}_a,\\ &\boldsymbol{\theta}_a\underline{R}_a[k+1] = \sum_{n=0}^{\infty}\boldsymbol{\theta}_a(\underline{R}_a[k])^n A_n \leq \boldsymbol{\theta}_a\sum_{n=0}^{\infty}(sp(R_a))^n A_n = sp(R_a)\boldsymbol{\theta}_a,\\ &\boldsymbol{\theta}_a(\underline{R}_a[k+1])^n \leq (sp(R_a))^n\boldsymbol{\theta}_a, \quad n \geq 1.\end{aligned} \tag{3.11}$$

On the other hand, by (2.3) and (3.9), $\underline{R}_a[1] \geq A_0 = R_a[1]$. By induction, it can be proved that $\underline{R}_a[k] \geq R_a[k]$ for $k \geq 1$.

Since the sum of $\{A_n, n \geq 0\}$ is an irreducible matrix, $\boldsymbol{\theta}_a$ is positive. By (3.11), the sequence $\{\underline{R}_a[k], k \geq 1\}$ is uniformly bounded. Denote by $\underline{R}_a$ the limit matrix of a converging subsequence of $\{\underline{R}_a[k], k \geq 1\}$. Then $R_a \leq \underline{R}_a$ and $sp(R_a)\boldsymbol{\theta}_a \leq \boldsymbol{\theta}_a\underline{R}_a \leq sp(R_a)\boldsymbol{\theta}_a$. Thus, $sp(R_a)\boldsymbol{\theta}_a = \boldsymbol{\theta}_a\underline{R}_a$, which implies that $sp(\underline{R}_a) = sp(R_a)$. If $R_a < \underline{R}_a$, $\underline{R}_a - R_a$ is nonzero and nonnegative. Then $\boldsymbol{\theta}_a(\underline{R}_a - R_a)$ is nonzero and nonnegative since $\boldsymbol{\theta}_a$ is positive. This is a contradiction. Thus, $R_a = \underline{R}_a$. Since this is true for any converging subsequence of $\{\underline{R}_a[k], k \geq 1\}$, it concludes that the sequence $\{\underline{R}_a[k], k \geq 1\}$ converges to R_a. □

There is a variety of selections of the function $\phi(x)$. For applications to Markov chains of GI/M/1 type and queueing theory, functions of the form $(1-x)^{-k}$, $k \geq 0$, are of interest, especially when $k = 0$ and 1. When $k = 0$, $\phi(x) = 1$. $\{A_n, n \geq 0\} \leq_\phi \{B_n, n \geq 0\}$ implies that $A_n \geq B_n$ for all n. Lemma 3.1 implies that $R_a \geq R_b$. This result is useful, but the condition is too strong to be held by any two different stochastic sequences. Therefore, the focus of the next two sections will be on the case $\phi(x) = (1-x)^{-1}$. The partial order $\leq_\phi$ with $\phi(x) = (1-x)^{-1}$, in some sense, bears some analogy to the classical stochastically larger order (see Shaked and Shanthikumar [24]) and it is relatively easy to check whether or not two sequences in M_m are partially ordered.

4. Stochastically larger order and the matrix *R*. In this section, the focus is on the "stochastically smaller (larger)" order defined by $\phi(x) = (1-x)^{-1}$, and denoted by $\leq_{st}$ ($\geq_{st}$). This partial order is important not only because it has a scale case counterpart (see Shaked and Shanthikumar [24]) but also because it induces some useful results for the stationary distribution of Markov chains of GI/M/1 type. Furthermore, it suggests some schemes for computing R (see section 5).

For the stochastically smaller order, $\phi_n = 1$ for all n. If $\{A_n, n \geq 0\} \leq_{st} \{B_n, n \geq 0\}$, (3.2) becomes $A_0 + A_1 + \cdots + A_n \geq B_0 + B_1 + \cdots + B_n$ for all $n \geq 0$. When n goes to infinity, it leads to $A \geq B$, where A and B are the sums of $\{A_n, n \geq 0\}$ and $\{B_n, n \geq 0\}$, respectively. Lemma 3.1 leads to the following results of the matrix R.

THEOREM 4.1. *For $\{A_n, n \geq 0\}$ and $\{B_n, n \geq 0\}$ in M_m, assume that $\{A_n, n \geq 0\} \leq_{st} \{B_n, n \geq 0\}$. Then $sp(R_a)\boldsymbol{\theta}_a \geq \boldsymbol{\theta}_a R_b, \boldsymbol{\theta}_b R_a \geq sp(R_b)\boldsymbol{\theta}_b, sp(R_a) \geq sp(R_b)$, and*

(1) if $sp(R_a) < 1$, then $(\mathbf{I} - R_a)^{-1}R_a \geq (\mathbf{I} - R_a)^{-1}R_b$ and $(\mathbf{I} - R_b)^{-1}R_a \geq (\mathbf{I} - R_b)^{-1}R_b$;

(2) if $sp(R_b) < 1$, then $(\mathbf{I} - R_b)^{-1}R_a \geq (\mathbf{I} - R_b)^{-1}R_b$.

Proof. Consider $\{tA_n, n \geq 0\}$ and $\{tB_n, n \geq 0\}$ for $0 < t < 1$. Their minimal nonnegative solutions to (1.1) are $R_a(t)$ and $R_b(t)$, respectively. It is clear that $R_a(t)$, $R_b(t)$, $sp(R_b(t))$, and $sp(R_a(t))$ are nondecreasing in t and upper bounded by R_a, R_b, $sp(R_b)$, and $sp(R_a)$, respectively. It can be proved that $R_a(t)$, $R_b(t)$, $sp(R_b(t))$, and $sp(R_a(t))$ are continuous with respect to t when $sp(R_a) < 1$, $sp(R_b) < 1$, and $0 \leq t \leq 1$ (see He [8]). Matrices $(\mathbf{I} - R_a(t))^{-1}$ and $(\mathbf{I} - R_b(t))^{-1}$ are well defined since $sp(R_b(t)) \leq sp(R_a(t)) < 1$. Therefore, all conclusions in Lemma 3.1 hold for $t < 1$. Taking t to 1, the results are obtained. □

Results given in Theorem 4.1 have many applications. Two examples are shown next. First, recall that $R_a(R_b)$ is the mean number (in matrix form) of visits of the Markov chain $P_a(P_b)$ (see (2.1) for definition) to level $k+1$ before returning to level k, given that the process started in level k. Intuitively, when $\{A_n, n \geq 0\}$ is stochastically smaller than $\{B_n, n \geq 0\}$, Markov chain P_a is more likely to stay in level $k+1$ and higher. Therefore, the mean number of visits to level $k+1$ and higher should be larger. This intuition leads to the following corollary.

COROLLARY 4.2. *Assume that* $\{A_n, n \geq 0\} \leq_{st} \{B_n, n \geq 0\}$ *and* $sp(R_a) < 1$. *Then* $(\mathbf{I} - R_a)^{-1}R_a \geq (\mathbf{I} - R_b)^{-1}R_b$, *where* $(\mathbf{I} - R_a)^{-1}R_a$ *(or* $(\mathbf{I} - R_b)^{-1}R_b$*) is the mean total number of visits to level* $k+1$ *and higher before returning to level* k*, given that the Markov chain started in level* k*.*

Proof. By Theorem 4.1, $(\mathbf{I} - R_a)^{-1}R_a \geq (\mathbf{I} - R_a)^{-1}R_b$. Then, for all $n \geq 1$,

$$\begin{aligned} (\mathbf{I} - R_a)^{-1}R_b &= [\mathbf{I} + (\mathbf{I} - R_a)^{-1}R_a]R_b \geq R_b + (\mathbf{I} - R_a)^{-1}R_b^2 \\ &\geq R_b + R_b^2 + \cdots + R_b^n + (\mathbf{I} - R_a)^{-1}R_b^{n+1}. \end{aligned} \tag{4.1}$$

Since $sp(R_b) \leq sp(R_a) < 1$, R_b^n converges to zero. This implies that $(\mathbf{I} - R_a)^{-1}R_b^n$ converges to zero, which leads to the result.

The probabilistic interpretation for $(\mathbf{I} - R_a)^{-1}R_a$ (and $(\mathbf{I} - R_b)^{-1}R_b$) comes from the fact that R^n is the mean number of visits to level $k+n$ before returning to level k, given that the Markov chain started in level k (see Neuts [16, Chapter 1]). □

Note. When $sp(R_a) = 1$, the Markov chain P_a is transient. The mean number of visits to level $k+1$ and higher before returning to level k is infinity, given that the Markov chain started in level k.

Corollary 4.2 shows that the mean number of visits to *all higher levels* before the Markov chain returns to its current level is monotone with respect to the stochastically larger order. It is natural to ask if it is also true for *a higher level*; i.e., when $\{A_n, n \geq 0\} \leq_{st} \{B_n, n \geq 0\}$, is it true that $R_a \geq R_b$? The answer to this question is negative. A counterexample is given as follows.

Example 4.1. Define $\{A_n, n \geq 0\}$ as follows: $A_n = 0$, $n \geq 3$,

$$A_0 = \begin{pmatrix} 0 & 0.1 & 0.1 \\ 0.1 & 0 & 0.1 \\ 0 & 0 & 0.1 \end{pmatrix}, \quad A_1 = \begin{pmatrix} 0.1 & 0.05 & 0 \\ 0.1 & 0 & 0 \\ 0.1 & 0 & 0 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 0.2 & 0.2 & 0.25 \\ 0 & 0.5 & 0.2 \\ 0.15 & 0.55 & 0.1 \end{pmatrix}. \tag{4.2}$$

$\{B_n, n \geq 0\}$ is the same as $\{A_n, n \geq 0\}$ except $(B_0)_{1,2} = 0$ and $(B_1)_{1,2} = 0.15$. Apparently, $\{A_n, n \geq 0\} \leq_{st} \{B_n, n \geq 0\}$. However, $R_a \geq R_b$ is not true since

$$R_a = \begin{pmatrix} 0.0356 & 0.1269 & 0.1099 \\ 0.1347 & 0.0352 & 0.1097 \\ 0.0156 & 0.0106 & 0.1029 \end{pmatrix} \quad \text{and} \quad R_b = \begin{pmatrix} 0.0155 & 0.0112 & 0.1025 \\ 0.1341 & 0.0399 & 0.1060 \\ 0.0155 & 0.0112 & 0.1025 \end{pmatrix}. \tag{4.3}$$

Another application of Theorem 4.1 is the comparison of tail probabilities of stationary distributions of Markov chains P_a and P_b of GI/M/1 type. When $sp(R_a) < 1$, for instance, there is an approximation (see Neuts [17])

$$R_a^n = (sp(R_a))^n \mathbf{v}\boldsymbol{\theta}_a + o((sp(R_a))^n) \quad \text{as } n \to \infty, \tag{4.4}$$

where $\mathbf{v}$ and $\boldsymbol{\theta}_a$ are the right and left eigenvectors of R_a corresponding to $sp(R_a)$ and normalized by $\boldsymbol{\theta}_a\mathbf{e} = \boldsymbol{\theta}_a\mathbf{v} = 1$. Equation (4.4) implies that the stationary distribution $\boldsymbol{\pi} = (\boldsymbol{\pi}_0, \boldsymbol{\pi}_1, \boldsymbol{\pi}_2, \ldots)$ of Markov chain P_a has the following approximation (see section 2 for the definition):

$$\boldsymbol{\pi}_n = (sp(R_a))^n (\boldsymbol{\pi}_0\mathbf{v})\boldsymbol{\theta}_a + o((sp(R_a))^n) \quad \text{as } n \to \infty. \tag{4.5}$$

Therefore, when $\{A_n, n \geq 0\} \leq_{st} \{B_n, n \geq 0\}, sp(R_a) \geq sp(R_b)$ and the tail probabilities of P_a are larger than that of P_b. Again, this implies that P_a is more likely to be in higher levels than P_b, although the probabilities of such events are small.

Note. For any partial order $\leq_\phi$ defined by (3.1) and (3.2), $sp(R_a) \geq sp(R_b)$ if $\{A_n, n \geq 0\} \leq_\phi \{B_n, n \geq 0\}$. This implies that the same conclusion for tail probabilities holds for the partial order ϕ. Cases of interest include $\phi(x) = (1-x)^{-k}$ for $k \geq 0$. It is clear that the condition in (3.2) is weaker for large k than for small k, but they all imply the monotonicity of the Perron–Frobenius eigenvalue of the minimal nonnegative solution to (1.1).

5. Upper bounds of the matrix *R* and related issues. As was mentioned in section 1, upper bounds for the matrix R are useful, and it is even more interesting to develop algorithms generating sequences which converge to R from above. To address these issues, the stochastically larger order is explored further in this section. Although algorithms generating sequences which converge to R from above are not developed in this paper, some upper bounds are found and a scheme is developed. The scheme generates a sequence, whose matrices have Perron–Frobenius eigenvalues larger than $sp(R)$, converging to R under some conditions.

Consider $\{A_n, n \geq 0\}$ and $\{B_n, n \geq 0\}$ in $\mathbf{M}_m$. Define the following sequence for R_b:

$$(5.1) \qquad \hat{R}_b[1] = \sum_{n=0}^{\infty} R_a^n B_n, \quad \hat{R}_b[k+1] = \sum_{n=0}^{\infty} (\hat{R}_b[k])^n B_n \quad \text{for } k \geq 1.$$

When $\{A_n, n \geq 0\} \leq_{st} \{B_n, n \geq 0\}$, the sequence generated by using (5.1) is expected to converge to R_b monotonically. The sequence indeed converges to R_b under some conditions but, unfortunately, it is not always a monotone sequence.

Property 5.1. *Assume that* $\{A_n, n \geq 0\} \leq_{st} \{B_n, n \geq 0\}$ *and* $sp(R_a) < 1$. *Then*

$$(5.2) \qquad \hat{R}_b[k] \geq R_b[k] \quad \textit{and} \quad sp(R_b) \leq sp(\hat{R}_b[k]) \leq sp(R_a), \quad k \geq 1.$$

If the sequence generated by using (5.1) *converges, it converges to* R_b.

Proof. The first inequality in (5.2) is proved as follows:

$$(5.3) \qquad \hat{R}_b[1] \geq B_0 = R_b[1], \text{ by induction, } \hat{R}_b[k] \geq R_b[k].$$

To prove the second part, the following inequalities are proved first, for n, $k \geq 1$,

$$(5.4) \quad (\mathbf{I}-R_b)^{-1}R_b^n \leq (\mathbf{I}-R_b)^{-1}(\hat{R}_b[k])^n \quad \text{and} \quad (\mathbf{I}-R_a)^{-1}(\hat{R}_b[k])^n \leq (\mathbf{I}-R_a)^{-1}R_a^n.$$

By Theorem 4.1, it has

$$(\mathbf{I}-R_b)^{-1}R_b = \sum_{n=0}^{\infty}(\mathbf{I}-R_b)^{-1}R_b^n B_n \leq \sum_{n=0}^{\infty}(\mathbf{I}-R_b)^{-1}R_a^n B_n \leq (\mathbf{I}-R_b)^{-1}\hat{R}_b[1],$$

(5.5)

$$\begin{aligned}(\mathbf{I}-R_b)^{-1}R_b^n &= R_b^{n-1}(\mathbf{I}-R_b)^{-1}R_b \\ &\leq R_b^{n-1}(\mathbf{I}-R_b)^{-1}\hat{R}_b[1] \leq \cdots \leq (\mathbf{I}-R_b)^{-1}(\hat{R}_b[1])^n.\end{aligned}$$

By induction, the first part of (5.4) is proved. To prove the second part, first notice

$$\begin{aligned}(\mathbf{I}-R_a)^{-1}R_a &= \sum_{n=0}^{\infty} R_a^n\left(\sum_{i=0}^{n} A_i\right) \geq \sum_{n=0}^{\infty} R_a^n\left(\sum_{i=0}^{n} B_i\right)\\ &= (\mathbf{I}-R_a)^{-1}\sum_{n=0}^{\infty} R_a^n B_n = (\mathbf{I}-R_a)^{-1}\hat{R}_b[1],\\ (\mathbf{I}-R_a)^{-1}R_a^n &\geq (\mathbf{I}-R_a)^{-1}(\hat{R}_b[1])^n, \qquad n \geq 0.\end{aligned} \tag{5.6}$$

Using (5.1), the second part of (5.4) can be proved by induction. By (5.4), it is easy to see that the Perron–Frobenius eigenvalues of the matrices in the sequence generated by (5.1) are between $sp(R_a)$ and $sp(R_b)$. When the sequence generated by (5.1) converges, it converges to a nonnegative solution of (1.1) associated with the sequence $\{B_n, n \geq 0\}$. Since the minimal nonnegative solution of (1.1) with the largest eigenvalue less than one is unique (see Neuts [16, Theorem 1.3.3]), the sequence generated by (5.1) converges to the matrix R_b. □

The above result shows that the Perron–Frobenius eigenvalue plays an important role in generating new sequences which converge to R from “above,” since the most important feature of the initial matrix for (5.1) is that its Perron–Frobenius eigenvalue is larger than that of the matrix R_b. Following this idea, another sequence can be constructed as follows, for $0 < s < 1$:

$$\hat{R}_b[1] = \sum_{n=0}^{\infty} s^n B_n = B^*(s), \quad \hat{R}_b[k+1] = \sum_{n=0}^{\infty} (\hat{R}_b[k])^n B_n, \quad k \geq 1. \tag{5.7}$$

PROPERTY 5.2. *Assume that* $1 > s > sp(R_b)$. *For the sequence generated by* (5.7),

$$\hat{R}_b[k] \geq R_b[k], \qquad sp(R_b) \leq sp(\hat{R}_b[k]) \leq sp(B^*(s)) \leq s. \tag{5.8}$$

If the sequence converges, it converges to R_b.

Proof. By definition, it has

$$\hat{R}_b[1] = B^*(s) \geq R_b[1] = B_0 \quad \text{and} \quad \hat{R}_b[k] \geq R_b[k], \quad k \geq 1. \tag{5.9}$$

Recall that $\boldsymbol{\theta}(s)$ is the eigenvector of $B^*(s)$ corresponding to its Perron–Frobenius eigenvalue and $s \geq sp(B^*(s))$. Then

$$\begin{aligned}&\boldsymbol{\theta}(s)\hat{R}[1] = \boldsymbol{\theta}(s)B^*(s) = \boldsymbol{\theta}(s)sp(B^*(s)) \leq \boldsymbol{\theta}(s)s,\\ &\boldsymbol{\theta}(s)(\hat{R}[1])^n \leq \boldsymbol{\theta}(s)s^n, \qquad n \geq 0.\end{aligned} \tag{5.10}$$

By induction, it can be proved that

$$\begin{aligned}&\boldsymbol{\theta}(s)(\hat{R}[k])^n \leq \boldsymbol{\theta}(s)(sp(B^*(s)))^n \leq \boldsymbol{\theta}(s)s^n, \qquad k, n \geq 1,\\ &sp(\hat{R}[k]) \leq sp(B^*(s)) \leq s.\end{aligned} \tag{5.11}$$

Replacing $\boldsymbol{\theta}(s)$ by $\boldsymbol{\theta}_b$ and s by $sp(R_b)$ and changing the direction of inequalities in (5.10) and (5.11), the other inequality in (5.8) can be proved.

Using the same argument as in Property 5.1, when the sequence generated by (5.7) converges, it converges to R_b. □

Properties 5.1 and 5.2 show that all the matrices in the sequence generated by (5.1) or (5.7) have their largest eigenvalues in $[sp(R_b), sp(R_a)]$ or $[sp(R_b), s)$ with

$sp(R_a) < 1$ and $s < 1$. This suggests that the generated sequence should converge to R_b so that the convergence condition in Properties 5.1 and 5.2 can be dropped out. This conjecture is supported by numerical examples as well. However, the generated sequence may not be monotone, and a rigorous proof of its convergence is difficult to obtain. In fact, such a proof may involve the theory of multiple dimension contraction mappings and is beyond the scope of this paper. This is why it is assumed in Properties 5.1 and 5.2 that the generated sequence converges. The issue of convergence is left as an open problem.

The following property shows that an extra condition guarantees the monotonicity and convergence of a generated sequence.

PROPERTY 5.3. *For a nonnegative matrix* X *with* $0 < sp(X) < 1$, *define* $X[0] = X$, $X[k+1] = B_0 + X[k]B_1 + \cdots$ *for* $k \geq 0$. *If* $X[0] \geq X[1]$, *then* $X[k] \geq X[k+1]$, $k \geq 0$, *and* $\{X[k], k \geq 0\}$ *converges to* R_b.

Proof. If $X[0] \geq X[1]$, then $(X[0])^n \geq (X[1])^n$. Therefore, $X[1] \geq X[2]$ by definition. The rest of the proof is completed by induction and by using Theorem 1.3.3 in Neuts [16]. □

Usually, to calculate the matrix R associated with a sequence $\{A_n, n \geq 0\}$ in M_m, a sequence $\{B_n, n \geq 0\}$ in M_m is not given to facilitate the computation of R. One has to construct new sequences in M_m from $\{A_n, n \geq 0\}$ for such purposes. Thus, let us focus on a single sequence $\{A_n, n \geq 0\}$ in M_m. The issues of interest are (1) to find some upper bounds of R_a and (2) to find some initial matrices or values for the algorithms defined by (5.1) and (5.7), respectively. The idea is to consider a truncated sequence $\{A_0, A_1, \ldots, A_{N-1}, A_N + A_{N+1} + \cdots\}$ for a given $N(> 0)$. Denote the minimal nonnegative solution to (1.1) corresponding to the truncated sequence by $R_{(N)}$. It is easy to verify that the truncated sequence is stochastically smaller than $\{A_n, n \geq 0\}$. The computation of $R_{(N)}$ could be much easier than that of R_a, especially when N is small. For instance, when $N = 2$, an efficient algorithm was proposed in Latouche and Ramaswami [10] for computing $R_{(2)}$. Also, $R_{(N)}$ is expected to be an upper bound of R_a. Unfortunately, this is not true in general. A simple counterexample is obtained when $sp(R_a) = 1$ (which implies that $sp(R_{(N)}) = 1$). In fact, numerical results show that no upper bound can be easily found for R_a. However, if $sp(R_{(N)}) < 1$ for a small $N(\geq 2)$, a sequence converging to R_a from "above" in terms of the Perron–Frobenius eigenvalue can usually be generated.

Upper bounds. Consider $N = 2$; a simple upper bound for R_a is suggested. Let $\boldsymbol{\theta}_{(2)}$ be the eigenvector of $sp(R_{(2)})$, and $\boldsymbol{\theta}A = \boldsymbol{\theta}$ and $\boldsymbol{\theta}\mathbf{e} = 1$. By Theorem 4.1, $sp(R_{(2)})\boldsymbol{\theta}_{(2)} \geq \boldsymbol{\theta}_{(2)}R_a$. It is easy to prove that $\boldsymbol{\theta}R_a \leq \boldsymbol{\theta}$. Combining the two inequalities yields

$$(5.12) \qquad (R_a)_{i,j} \leq \min\left\{ sp(R_{(2)})\frac{(\boldsymbol{\theta}_{(2)})_j}{(\boldsymbol{\theta}_{(2)})_i}, \frac{(\boldsymbol{\theta})_j}{(\boldsymbol{\theta})_i} \right\}.$$

Notice that when $sp(R_{(2)}) = 1$, $\boldsymbol{\theta}_{(2)} = \boldsymbol{\theta}$. The upper bounds provided by (5.12) are useful for some state i where $(\boldsymbol{\theta})_i$ (or $(\boldsymbol{\theta}_{(2)})_i$) is small compared with other components. Another immediate result is that $sp(R_{(2)}) \geq sp(R_a)$.

A computational scheme. Using the idea of truncation, first find the smallest N for which $sp(R_{(N)}) < 1$. This is equivalent to finding the minimal number N_{min} such that

$$(5.13) \qquad \boldsymbol{\theta}\left[\sum_{n=0}^{N-1} nA_n + N\left(\sum_{n=N}^{\infty} A_n\right)\right]\mathbf{e} > 1.$$

This condition is given in Neuts [16, Chapter 1].

When N_{min} is small, R_{Nmin} can be computed efficiently using some existing algorithms. With R_{Nmin}, (5.1) can be used to generate a sequence which may converge to R from "above." Although the generated sequence is not always monotone, numerical results show that it is a decreasing sequence in most of the cases. The sequence generated by using (5.1) can be compared with the nondecreasing sequence generated by using (2.3) to determine when the iteration process for R should be stopped.

It is interesting to know whether or not $R_{(N)}$ is an upper bound of R_a when $sp(R_{(N)}) < 1$ or $sp(R_a) < 1$. The following two examples show that $R_{(N)} \geq R_a$ is generally untrue.

Example 5.1. Define the following stochastic sequence:

$$\begin{gathered}
A_0 = \begin{pmatrix} 0 & 0.2 & 0 & 0 \\ 0.1 & 0 & 0.1 & 0 \\ 0 & 0 & 0.5 & 0.1 \\ 0.1 & 0 & 0.1 & 0.5 \end{pmatrix}, \qquad A_1 = \begin{pmatrix} 0.1 & 0 & 0.2 & 0 \\ 0 & 0.2 & 0.1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \end{pmatrix}, \\
A_2 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0.1 & 0 & 0 & 0 \end{pmatrix}, \qquad A_3 = \begin{pmatrix} 0.2 & 0 & 0.3 & 0 \\ 0 & 0.2 & 0.3 & 0 \\ 0 & 0 & 0 & 0.4 \\ 0.1 & 0 & 0 & 0 \end{pmatrix}.
\end{gathered} \tag{5.14}$$

The matrix R for $\{A_0, A_1, A_2 + A_3\}$ is denoted as $R_{(2)}$, and $R_{(3)}$ for $\{A_0, A_1, A_2, A_3\}$. It is found that $sp(R_{(2)}) = 1 > sp(R_{(3)}) = 0.8988$, but $R_{(2)} > R_{(3)}$ is not true. This example shows that even when the transition matrix sequence is truncated to $N = 2$, the elements of the matrix R do not necessarily become larger. Therefore, $R_{(2)}$ may not be an upper bound of R_a in general.

Example 5.2. Define a stochastic sequence:

$$\begin{gathered}
A_0 = \begin{pmatrix} 0 & 0.1 & 0 & 0.2 \\ 0.1 & 0.1 & 0.1 & 0 \\ 0 & 0 & 0 & 0.1 \\ 0.1 & 0 & 0.1 & 0 \end{pmatrix}, \qquad A_1 = \begin{pmatrix} 0.05 & 0 & 0.05 & 0 \\ 0 & 0.2 & 0.05 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0.05 & 0 \end{pmatrix}, \\
A_2 = \begin{pmatrix} 0 & 0.05 & 0.05 & 0 \\ 0 & 0.1 & 0.05 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0.05 & 0 \end{pmatrix}, \qquad A_3 = \begin{pmatrix} 0 & 0.2 & 0.3 & 0 \\ 0 & 0.1 & 0.2 & 0 \\ 0 & 0 & 0 & 0.4 \\ 0.1 & 0 & 0 & 0 \end{pmatrix},
\end{gathered} \tag{5.15}$$

$$A_4 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \\ 0.1 & 0 & 0 & 0.5 \end{pmatrix}.$$

The matrix R for $\{A_0, A_1, A_2, A_3 + A_4\}$ is denoted as $R_{(3)}$, and $R_{(4)}$ for $\{A_0, A_1, A_2, A_3, A_4\}$. $sp(R_{(3)}) = 0.258442 > sp(R_{(4)}) = 0.25555$, but $R_{(3)} > R_{(4)}$ is not true.

6. Discussions of other partial orders. In sections 4 and 5, the focus was on the stochastically larger order. It has been shown that the Perron–Frobenius eigenvalue of the minimal nonnegative solution to (1.1) is monotone with the stochastically larger order but the matrix R itself is generally not. It is interesting to find out (1) are there weaker conditions which guarantee the monotonicity of the Perron–Frobenius eigenvalue and (2) what conditions guarantee the monotonicity of R with respect to some partial order? A thorough discussion of these two issues is beyond the scope of this paper. Only two examples are shown below.

Example 6.1. The moment generating order.

Definition 6.1. *$\{A_n, n \geq 0\}$ is smaller than $\{B_n, n \geq 0\}$ with respect to the moment generating order if $A^*(z) \geq B^*(z)$ for $z \in [0, 1)$. Denote this order as $\{A_n, n \geq 0\} \leq_m \{B_n, n \geq 0\}$. The moment generating order is weaker than the "$\leq_\phi$" order introduced in section 3 since $\phi(x)A^*(x) \geq \phi(x)B^*(x)$ implies $A^*(x) \geq B^*(x)$ for $x \in [0, 1)$ as $\phi(x) > 0$. If $\{A_n, n \geq 0\} \leq_m \{B_n, n \geq 0\}$,*

$$\boldsymbol{\theta}_a sp(R_a) = \boldsymbol{\theta}_a R_a = \boldsymbol{\theta}_a A^*(sp(R_a)) \geq \boldsymbol{\theta}_a B^*(sp(R_a)). \tag{6.1}$$

This implies that $sp(R_a) \geq sp(B^*(sp(R_a)))$. By the convexity of function $\log(sp(B^*(e^{-s})))$, $sp(R_a) \geq sp(R_b)$. Therefore, the moment generating order implies the monotonicity of the Perron–Frobenius eigenvalue of the matrix R.

Example 6.2. Functional monotone and functional dominance.

Let J be the matrix with all diagonal and underdiagonal elements 1 and all others zero. For vectors $\mathbf{u}$ and $\mathbf{v}$, if $\mathbf{u}J \leq \mathbf{v}J$, then $\mathbf{u}$ is *J-dominated* by $\mathbf{v}$, denoted by $\mathbf{u} \leq_J \mathbf{v}$. A matrix X is *J-dominated* by Y is $XJ \leq YJ$, denoted by $X \leq_J Y$. For a matrix X with m row vectors $\{\mathbf{x}_i\}$, if $\mathbf{x}_1 \leq_J \mathbf{x}_2 \leq_J \cdots \leq_J \mathbf{x}_m$, then X is *J-monotone* (see Ridder [23] for more details about monotone and dominance orders).

Definition 6.2. *A sequence $\{A_n, n \geq 0\}$ is J-dominated by $\{B_n, n \geq 0\}$ if $A_n \leq_J B_n$, for all n, denoted by $\{A_n, n \geq 0\} \leq_J \{B_n, n \geq 0\}$.*

Property 6.1. *For $\{A_n, n \geq 0\}$ and $\{B_n, n \geq 0\}$ in M_m, if A_n and B_n and J-monotone for all $n \geq 0$, then R_a and R_b are J-monotone. Furthermore, if $\{A_n, n \geq 0\} \leq_J \{B_n, n \geq 0\}$, then $R_a \leq_J R_b$.*

Proof. In Ridder [23], it has been proved that the J-monotone and J-dominance are closed under matrix multiplication and summation. The results are true for the sequences generated by using (2.3). Since R_a and R_b are limits of those sequences, the results follow. □

Note. The condition for J-domination is very strong since it imposes J-monotone on every pair of matrices $\{A_n, B_n\}$ of the two sequences. Nonetheless, J-domination is a partial order which guarantees the monotonicity of the matrix R. In addition, Property 6.1 finds a nice application in the GI/MAP/1 queue which shall be shown in the next section.

7. Applications to the GI/MAP/1 queue. This section considers a single server queueing system with a Markov arrival process (MAP) as its service process, general independent interarrival times, and a "first-come-first-served" service discipline. The MAP was first introduced in Neuts [15] (also see Lucantoni, Meier-Hellstern, and Neuts [12], and Neuts [18]) as a generalization of the phase-type renewal process. The MAP is defined on a finite irreducible Markov process (called the underlying Markov process) with m states and an irreducible infinitesimal generator D. In the MAP, the sojourn time in state i is exponentially distributed with parameter D_{ii}. At the end of the sojourn time in state i, a transition occurs to state j, $1 \leq j \leq m$, where the transition may or may not represent an arrival. Let D_0 be the (matrix) rate of transitions without an arrival and D_1 be the rate of transitions with an arrival. D_0 and D_1 are $m \times m$ matrices where D_0 has negative diagonal elements and nonnegative off-diagonal elements, and D_1 is a nonnegative matrix. Then $D = D_0 + D_1$. Using $\boldsymbol{\theta}$ to denote the stationary probability vector of the Markov process with the generator D, $\boldsymbol{\theta}$ satisfies $\boldsymbol{\theta} D = 0$ and $\boldsymbol{\theta}\mathbf{e} = 1$. The stationary service rate of the Markov arrival process is then $\lambda = \boldsymbol{\theta} D_1 \mathbf{e}$. The interarrival time between two consecutive customers is random with finite mean and variance. Let $F(t)$ be the distribution function of the interarrival time and denote the Laplace–Stieltjes transform (LST) of $F(t)$ by $f^*(s)$.

Consider the embedded Markov chain (q_n, J_n) at the nth arrival epoch, where q_n is the queue length just before the arrival and J_n is the phase of the service process at the arrival epoch. This embedded Markov chain (q_n, J_n) is of GI/M/1 type, since the increase of the queue length is at most one at a time. The one-step transition matrix of (q_n, J_n) is similar to P given in (2.1). Transition blocks $\{A_n, n \geq 0\}$ are defined in terms of D_0, D_1, and $F(t)$. The matrix R is then defined as the minimal nonnegative solution to (1.1) (see Neuts [19]). It can be proved that the matrix R is also the minimal nonnegative solution to the following exponential-form equation:

$$X = \int_0^\infty \exp\{t(D_0 + XD_1)\} dF(t). \tag{7.1}$$

Next, the results obtained in previous sections are applied to obtain some interesting results about the matrix R and the GI/MAP/1 queue. For convenience, the following analysis of R begins with (7.1) instead of (1.1). The discussion consists of two parts. The first part includes the results obtained by assuming the stochastically larger order on the interarrival times, and the second part consists of results obtained by imposing a special structure on the matrix representation (D_0, D_1) of the service process.

First, define a new matrix Q from R. Let $\xi > \max\{1, |(D_0)_{i,j}|\}$. By (7.1), it has

$$\begin{aligned} R &= \int_0^\infty \exp\{t(-\xi\mathbf{I} + \xi\mathbf{I} + D_0 + RD_1)\} dF(t) \\ &= \sum_{n=0}^\infty \int_0^\infty \frac{e^{-\xi t}(\xi t)^n}{n!} dF(t) \left[\mathbf{I} + \frac{D_0 + RD_1}{\xi}\right]^n \equiv a^*(Q), \end{aligned} \tag{7.2}$$

where $Q = \mathbf{I} + (D_0 + RD_1)/\xi$ and

$$a^*(z) = \sum_{n=0}^\infty z^n a_n, \quad \text{where } a_n = \int_0^\infty \frac{e^{-\xi t}(\xi t)^n}{n!} dF(t) \quad \text{and} \quad \sum_{n=0}^\infty a_n = 1. \tag{7.3}$$

For the matrix Q, it has

$$Q = K_0 + a^*(Q)K_1, \quad \text{where } K_0 = \mathbf{I} + \frac{D_0}{\xi} \text{ and } K_1 = \frac{D_1}{\xi}. \tag{7.4}$$

Thus, the matrix Q is the minimal nonnegative solution to (7.4), or, equivalently, Q is the minimal nonnegative solution to (1.1) with a sequence $\{K_0 + a_0K_1, a_1K_1, a_2K_1, \ldots\}$, which has a special structure in the sense that only two matrices K_0 and K_1 are involved. Let $K = K_0 + K_1$. K is an irreducible stochastic matrix. It is sometimes more convenient to deal with (7.4) than (7.1). The following relationship between R and Q ensures the equivalence of the studies of (7.1) and (7.4).

Lemma 7.1. *The minimal nonnegative solutions to* (7.1) *and* (7.4) *are* R *and* Q, *respectively. Let* $\eta(s)$ *be the largest eigenvalue of* $K(s) = K_0 + sK_1$. *Then* $\eta(s)$ *is increasing in* s *and*

$$(7.5)\quad sp(R) = \int_0^\infty \exp\{-tsp(Q)\}dF(t) = f^*(sp(Q)) \quad \textit{and} \quad sp(Q) = \eta(sp(R)).$$

Proof. The one-on-one relationship between R and Q is clear from (7.1), (7.2), and (7.3). The results of the Perron–Frobenius eigenvalues are then obtained easily from (7.2) and (7.4). $sp(R) < 1$ if and only if $sp(Q) < 1$ since $f^*(s)$ and $\eta(s)$ are strictly increasing. □

Now, the focus shifts from R and (7.1) to Q, (7.4), $a^*(s)$, and $K(s)$. The problem becomes simpler since only a function $a^*(s)$ and a linear matrix function $K(s)$ are involved. It is now easy to apply results obtained in sections 4, 5, and 6 to (7.4).

Theorem 7.2. *Consider two* GI/MAP/1 *queueing systems, labeled* "a" *and* "b," *respectively, with the same service process. If the interarrival times satisfy* $F_a(x) \leq_{st} F_b(x)$ *(the usual stochastic order, i.e.,* $F_a(x) \geq F_b(x)$ *for all* $x \geq 0$*; see Shaked and Shanthikumar* [24]*), then*

$$(7.6)\qquad sp(R_a) \geq sp(R_b) \quad \textit{and} \quad -(D_0 + R_aD_1)^{-1} \geq -(D_0 + R_bD_1)^{-1}.$$

Proof. If $F_a(x) \leq_{st} F_b(x)$, then $\{a_n\} \leq_{st} \{b_n\}$. By Definition 3.1, $\{K_0 + a_0K_1, a_1K_1, \ldots\} \leq_{st} \{K_0 + b_0K_1, b_1K_1, \ldots\}$. By Theorem 4.1, $sp(Q_a) \geq sp(Q_b)$. Since $\eta(s)$ is increasing, $sp(R_a) \geq sp(R_b)$. By Corollary 4.2, $(\mathbf{I} - Q_a)^{-1} \geq (\mathbf{I} - Q_b)^{-1}$, which implies (7.6). □

Intuitively, Theorem 7.2 shows that systems with stochastically smaller interarrival times (shorter interarrival times), are more likely to have a long queue since $sp(R_a) \geq sp(R_b)$ (recall the discussion in section 4). The matrix $D_0 + R_aD_1$ is the infinitesimal generator of the waiting time process in the queueing system (see Asmussen and Perry [3]). Theorem 7.2 shows that the mean waiting time is longer for the system with a stochastically smaller interarrival time.

The following theorem shows the monotonicity of the matrix R under the J-monotone order. It further implies the monotonicity of the corresponding stationary distribution of the queue length in the queueing system of interest.

Theorem 7.3. *Consider two* GI/MAP/1 *queues, labeled* "a" *and* "b," *respectively, with the same interarrival times. Their service processes have matrix representations* (D_0, D_1) *and* (C_0, C_1), *respectively. If* D_0, D_1, C_0, *and* C_1 *are* J*-monotone,* $D_0 \leq_J C_0$ *and* $D_1 \leq_J C_1$, *then* R_a *and* R_b *are* J*-monotone, and* $R_a \leq_J R_b$.

Proof. The theorem is proved by Property 6.1. □

Finally, a scheme for computing Q and R is proposed. The idea is to find an upper bound for $sp(Q)$ so that a sequence whose matrices have a Perron–Frobenius eigenvalue larger than $sp(Q)$ can be generated (see (5.7) and (7.4)). Another nondecreasing sequence can be generated by using (2.3). Compare the two sequences to determine when the iteration process for Q should be stopped. An upper bound for $sp(Q)$ can be found as follows: Let $t_0 = 0.5$.

(i) $s_n = a^*(t_n) = f^*(\xi(1 - t_n))$.

(ii) If $sp(K(s_n)) \geq t_n$, STOP; if $sp(K(s_n)) < t_n$, go to (iii).

(iii) $t_{n+1} = (1 + t_n)/2$, go to (i).

Then t_n gives an upper bound of $sp(Q)$. The matrix R can be obtained accordingly.

This scheme is feasible when $f^*(t)$ can be evaluated numerically. The Perron–Frobenius eigenvalue of the matrix $K(s)$ can be found without much difficulty since it is a linear function of s. This scheme might be useful in improving the accuracy for computing Q and R.

Acknowledgments. The author would like to thank two referees for their valuable comments and suggestions, and especially for pointing out a serious error in an earlier version of this paper. The author would also like to thank Dr. Yiqiang Zhao for proofreading this paper.

REFERENCES

[1] N. Akar and K. Sohraby, *An invariant subspace approach in $M/G/1$ and $G/M/1$ type Markov chains*, Comm. Statist. Stochastic Models, 13 (1997), pp. 381–416.

[2] S. Asmussen and V. Ramaswami, *Probabilistic interpretations of some duality results for the matrix paradigms in queueing theory*, Comm. Statist. Stochastic Models, 6 (1990), pp. 715–734.

[3] S. Asmussen and D. Perry, *On cycle maxima, first passage problems and extreme value theory for queues*, Comm. Statist. Stochastic Models, 8 (1992), pp. 421–458.

[4] D. Bini and B. Meini, *On the solution of a nonlinear matrix equation arising in queueing problems*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 906–926.

[5] D. J. Daley, *Stochastically monotone Markov chains*, Z. Wahrscheinlichkeitstheorieund Verw. Gebiete, 10 (1968), pp. 305–317.

[6] H. R. Gail, S. L. Hantler, and B. A. Taylor, *Solutions of the basic matrix equation for $M/G/1$ and $G/M/1$ type Markov chains*, Comm. Statist. Stochastic Models, 10 (1994), pp. 1–43.

[7] F. R. Gantmacher, *The Theory of Matrices*, Chelsea, New York, 1959.

[8] Q.-M. He, *Differentiability of the matrices R and G in the matrix analytic methods*, Comm. Statist. Stochastic Models, 11 (1995), pp. 123–132.

[9] J. Keilson and A. Kester, *Monotone matrices and monotone Markov processes*, Stochastic Process. Appl., 5 (1977), pp. 231–241.

[10] G. Latouche and V. Ramaswami, *A logarithmic reduction algorithm for quasi-birth-and-death processes*, J. Appl. Probab., 30 (1993), pp. 650–674.

[11] G. Latouche and G. W. Stewart, *Numerical methods for $M/G/1$ type queues*, in Computations with Markov Chains, W. J. Stewart, ed., Kluwer Academic Publishers, Norwell, MA, 1996, pp. 571–581.

[12] D. M. Lucantoni, K. S. Meier-Hellstern, and M. F. Neuts, *A single server queue with server vacations and a class of non-renewal arrival processes*, Adv. Appl. Prob., 22 (1990), pp. 676–705.

[13] D. M. Lucantoni, *New results on the single server queue with a batch Markovian arrival process*, Comm. Statist. Stochastic Models, 7 (1991), pp. 1–46.

[14] A. W. Marshall and I. Olkin, *Inequalities: Theory of Majorization and Its Applications*, Academic Press, New York, 1979.

[15] M. F. Neuts, *A versatile Markovian point process*, J. Appl. Probab., 16 (1979), pp. 764–779.

[16] M. F. Neuts, *Matrix-Geometric Solutions in Stochastic Models: An Algorithmic Approach*, The Johns Hopkins University Press, Baltimore, MD, 1981.

[17] M. F. Neuts, *The caudal characteristic curve of queues*, Adv. Appl. Prob., 18 (1986), pp. 221–254.

[18] M. F. Neuts, *Structured Stochastic Matrices of M/G/1 Type and Their Applications*, Marcel Dekker, New York, 1989.

[19] M. F. Neuts, *Matrix analytic methods in queueing theory*, in Advances in Queueing, Probab. Stochastics Ser., CRC Press, Boca Raton, FL, 1995.

[20] P. Purdue, *Non-linear matrix integral equation of Volterra type in queueing theory*, J. Appl. Probab., 10 (1973), pp. 644–651.

[21] V. Ramaswami, *Nonlinear matrix equations in applied probability—solution techniques and open problems*, SIAM Rev., 30 (1988), pp. 256–263.

[22] V. Ramaswami and P. G. Taylor, *Some properties of the rate operators in level dependent quasi-birth-and-death processes with a countable number of phases*, Comm. Statist. Stochastic Models, 12 (1996), pp. 143–164.

[23] A. Ridder, *Stochastic ordering of conditional steady-state probabilities*, Comm. Statist. Stochastic Models, 4 (1988), pp. 373–385.

[24] M. Shaked and J. G. Shanthikumar, *Stochastic Orders and Their Applications*, Academic Press, New York, 1994.

SPECIAL SECTION ON SPARSE AND STRUCTURED MATRICES AND THEIR APPLICATIONS

The field of sparse matrices is a broad and important area of the computational sciences that includes structured matrices and those with seemingly little or no structure. The relevance of the field is highlighted by the wide range of application areas that require the exploitation of matrix sparsity and structure in order to achieve a solution given real-world constraints on computing resources and/or time. Applications in which sparse matrices appear include structural analysis, computational fluid dynamics, economic modeling, financial analysis, numerical optimization, statistical modeling, power network analysis, electromagnetics, meteorology, medical imaging, data mining, and many more.

A number of significant advancements in sparse matrix computations have been made in recent years. These advances have led to new challenges as multidisciplinary problems are now ambitiously posed, and they symbolize the growth in the area and demonstrate the dependence of the future of many fields on sparse matrix methods. Moreover, they give rise to a host of new problems that have yet to be addressed by the sparse matrix community.

The Second SIAM Conference on Sparse Matrices, which was held in Coeur d'Alene, Idaho, October 9–11, 1996, was organized specifically to address some of the challenging issues in sparse matrix computations. This special issue of *SIAM Journal on Matrix Analysis and Applications* consists of some of the outstanding papers that were presented at the Sparse Matrix conference. Because of an error in scheduling, the paper titled "Using Generalized Cayley Transformations within an Inexact Rational Krylov Sequence Method" by Richard Lehoucq, which should have appeared in this section, was published in *SIAM Journal on Matrix Analysis and Applications*, volume 20, number 1.

We are grateful to Paul van Dooren, the Editor-in-Chief of *SIAM Journal on Matrix Analysis and Applications*, and the SIAM office for agreeing to publish a special section on sparse matrices. We would like to thank Roland Freund, Anne Greenbaum, Joseph Liu, and Zdenek Strakoš for serving on the Guest Editorial Board; their effort and cooperation in handling the papers were much appreciated. Finally, we would like to express thanks to all the authors who have submitted papers to the special section; it would not have become a reality without their support.

Esmond G. Ng
Lawrence Berkeley National Laboratory

Daniel J. Pierce
The Boeing Company

SIAM J. MATRIX ANAL. APPL.
Vol. 20, No. 4, pp. 889–901

THE DESIGN AND USE OF ALGORITHMS FOR PERMUTING LARGE ENTRIES TO THE DIAGONAL OF SPARSE MATRICES[*]

IAIN S. DUFF[†] AND JACKO KOSTER[‡]

Abstract. We consider techniques for permuting a sparse matrix so that the diagonal of the permuted matrix has entries of large absolute value. We discuss various criteria for this and consider their implementation as computer codes. We then indicate several cases where such a permutation can be useful. These include the solution of sparse equations by a direct method and by an iterative technique. We also consider its use in generating a preconditioner for an iterative method. We see that the effect of these reorderings can be dramatic although the best a priori strategy is by no means clear.

Key words. sparse matrices, maximum transversal, direct methods, iterative methods, preconditioning

AMS subject classifications. 65F05, 65F50

PII. S0895479897317661

1. Introduction.

We study algorithms for the permutation of a square unsymmetric sparse matrix $\boldsymbol{A}$ of order n so that the diagonal of the permuted matrix has large entries. This can be useful in several ways. If we wish to solve the system

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \tag{1.1}$$

where $\boldsymbol{A}$ is a nonsingular square matrix of order n and $\boldsymbol{x}$ and $\boldsymbol{b}$ are vectors of length n, then a preordering to place large entries on the diagonal can be useful whether direct or iterative methods are used for solution.

For direct methods, putting large entries on the diagonal suggests that pivoting down the diagonal might be more stable. There is, of course, nothing rigorous in this and indeed stability is not guaranteed. However, if we have a solution scheme like the multifrontal method of Duff and Reid [10], where a symbolic phase chooses the initial pivotal sequence and the subsequent factorization phase then modifies this sequence for stability, the modification required may be less than if the permutation were not applied.

For iterative methods, simple techniques like Jacobi or Gauss–Seidel converge more quickly if the diagonal entry is large relative to the off-diagonals in its row or column and techniques like block iterative methods can benefit if the entries in the diagonal blocks are large. Additionally, for preconditioning techniques, for example, for diagonal preconditioning or incomplete LU preconditioning, it is intuitively evident that large diagonals should be beneficial.

We consider more precisely what we mean by such permutations in section 2, and we discuss algorithms for performing them and implementation issues in section 3. We consider the effect of these permutations when using direct methods of solution in section 4 and their use with iterative methods in sections 5 and 6, discussing the effect

[*] Received by the editors February 28, 1997; accepted for publication (in revised form) by J. Liu January 9, 1998; published electronically July 9, 1999.
http://www.siam.org/journals/simax/20-4/31766.html

[†] Rutherford Appleton Laboratory, Chilton, Didcot, Oxon OX11 0QX, UK (isd@rl.ac.uk).

[‡] CERFACS, 42 Ave. G Coriolis, 31057 Toulouse Cedex, France (koster@cerfacs.fr). Current address: Rutherford Appleton Laboratory, Chilton, Didcot, Oxon OX11 0QX, UK.

on preconditioning in the latter section. Finally, we consider some of the implications of this current work in section 7.

Throughout, the symbol $|x|$ should be interpreted in context. If x is a scalar, the modulus is intended; if x is a set, then the cardinality, or number of entries in the set, is understood.

2. Permuting a matrix to have large diagonals.

2.1. Transversals and maximum transversals. We say that an $n \times n$ matrix $\boldsymbol{A}$ has a large diagonal if the absolute value of each diagonal entry is large relative to the absolute values of the off-diagonal entries in its row and column. We will be concerned with permuting the rows and columns of the matrix so the resulting diagonal of the permuted matrix has this property. That is, for the permuted matrix, we would like the ratio

$$\frac{|a_{jj}|}{\max_{i \neq j} |a_{ij}|} \tag{2.1}$$

to be large for all j, $1 \leq j \leq n$. Of course, it is not even possible to ensure that this ratio is greater than 1.0 for all j as the simple example $\binom{1\ \ 2}{2\ \ 3}$ shows. It is thus necessary to scale the matrix before computing the permutation. An appropriate scaling would be to scale the columns so that the largest entry in each column is 1.0. The algorithm that we describe in section 2.2 would then have the effect of maximizing (2.1).

For an arbitrary nonsingular $n \times n$ matrix, it is a necessary and sufficient condition that for a set of n entries to be permuted to the diagonal, no two can be in the same row and no two can be in the same column. Such a set of entries is termed a maximum transversal, a concept that will be central to this paper and which we now define more rigorously.

We let T denote a set of (at most n) ordered index pairs (i, j), $1 \leq i, j \leq n$, in which each row index i and each column index j appears at most once. T is called a *transversal* for matrix $\boldsymbol{A}$ if $a_{ij} \neq 0$ for each $(i, j) \in T$. T is called a *maximum* transversal if it has largest possible cardinality. $|T|$ is equal to n if the matrix is nonsingular. If indeed $|T| = n$, then T defines an $n \times n$ permutation matrix $\boldsymbol{P}$ with

$$\begin{cases} p_{ij} = 1 & \text{for } (i,j) \in T, \\ p_{ij} = 0 & \text{otherwise} \end{cases}$$

so that $\boldsymbol{P}^T \boldsymbol{A}$ is the matrix with the transversal entries on the diagonal.

In sparse system solution, a major use of transversal algorithms is in the first stage of permuting matrices to block triangular form. The matrix is first permuted by an unsymmetric permutation to make its diagonal zero-free, after which a symmetric permutation is used to obtain the block triangular form. An important feature of this approach is that the block triangular form does not depend on which transversal is found in the first stage [6]. A maximum transversal is also required in the generalization of the block triangular ordering developed by [17].

2.2. Bottleneck transversals. We will consider two strategies for obtaining a maximum transversal with large transversal entries. The primary strategy that we consider in this paper is to maximize the smallest value on the diagonal of the permuted matrix. That is, we compute a maximum transversal T such that for any other maximum transversal T_1 we have

$$\min_{(i,j) \in T_1} |a_{ij}| \;\leq\; \min_{(i,j) \in T} |a_{ij}|.$$

Transversal T is called a *bottleneck transversal*,[1] and the smallest value $|a_{ij}|$, $(i,j) \in T$, is called the *bottleneck value* of $\boldsymbol{A}$. Equivalently, if $|T| = n$, the smallest value on the diagonal of $\boldsymbol{P}^T\boldsymbol{A}$ is maximized over all permutations $\boldsymbol{P}$ and equals the bottleneck value of $\boldsymbol{A}$.

An outline of an algorithm that computes a bottleneck transversal T' for a matrix $\boldsymbol{A}$ is given below. We assume that we already have an algorithm for obtaining a maximum transversal and denote by $MT(A,T)$ a routine that returns a maximum transversal for a matrix $\boldsymbol{A}$, starting with the initial "guess" transversal T. We let $\boldsymbol{A}_\epsilon$ denote the matrix that is obtained by setting to zero in $\boldsymbol{A}$ all entries a_{ij} for which $|a_{ij}| < \epsilon$ (thus $\boldsymbol{A}_0 = \boldsymbol{A}$) and T_ϵ denote the transversal obtained by removing from transversal T all the elements (i,j) for which $|a_{ij}| < \epsilon$ (thus $T_0 = T$).

```
ALGORITHM BT.
    Initialization:
    Set $\epsilon min$ to zero and $\epsilon max$ to infinity.
    $M := MT(A, \emptyset)$;
    $T' := M$;
    while (there exist $i, j$ such that $\epsilon min < |a_{ij}| < \epsilon max$) do
    begin
        choose $\epsilon = |a_{ij}|$
        (we discuss how this is chosen later);
        $T := MT(A_\epsilon, T'_\epsilon)$;
        if $|T| = |M|$
        then
            $T' := T$;                    $(\star)$
            $\epsilon min := \epsilon$;
        else
            $\epsilon max := \epsilon$;
        endif
    end;
    Complete transversal for permutation
    (needed if matrix is structurally singular).
```

M is a maximum transversal for $\boldsymbol{A}$, and hence $|M|$ is the required cardinality of the bottleneck transversal T' that is to be computed. If $\boldsymbol{A}$ is nonsingular, then $|M| = n$. Throughout the algorithm, ϵmax and ϵmin are such that a maximum transversal of size $|M|$ does not exist for $\boldsymbol{A}_{\epsilon max}$ but does exist for $\boldsymbol{A}_{\epsilon min}$. At each step, ϵ is chosen in the interval $(\epsilon min, \epsilon max)$, and a maximum transversal for the matrix $\boldsymbol{A}_\epsilon$ is computed. If this transversal has size $|M|$, then ϵmin is set to ϵ; otherwise, ϵmax is set to ϵ. Hence, the size of the interval decreases at each step and ϵ will converge to the bottleneck value. After termination of the algorithm, T' is the computed bottleneck transversal and ϵ the corresponding bottleneck value. The value for ϵ is unique. The bottleneck transversal T' usually is not unique.

Algorithm BT makes use of algorithms for finding a maximum transversal. The currently known algorithm with best asymptotic bound for finding a maximum transversal is by Hopcroft and Karp [14]. It has a worst-case complexity of $\mathcal{O}(\sqrt{n}\tau)$, where τ is the number of entries in the matrix. An efficient implementation of this algorithm can be found in Duff and Wiberg [11]. The depth-first search algorithm implemented by Duff [7] in the Harwell Subroutine Library code `MC21` has a theoretically worst-case behavior of $\mathcal{O}(n\tau)$, but in practice it behaves more like $\mathcal{O}(n + \tau)$. Because this latter

[1] The term *bottleneck* has been used for many years in assignment problems, for example [12].

algorithm is far simpler, we concentrate on this in the following, although we note that it is relatively straightforward to modify and use the algorithm of Hopcroft and Karp [14] in a similar way.

A limitation of Algorithm BT is that it maximizes only the smallest value on the diagonal of the permuted matrix. Although this means that the other diagonal values are no smaller, they may not be maximal. Consider, for example, the 3 × 3 matrix

$$\begin{pmatrix} \delta & 1.0 & 1.0 \\ 1.0 & \delta & \\ & & \delta \end{pmatrix} \tag{2.2}$$

with δ close to zero. Algorithm BT applied to this matrix returns $\epsilon = \delta$ and either transversal $\{(1,1),(2,2),(3,3)\}$ or $\{(2,1),(1,2),(3,3)\}$. Clearly, the latter transversal is preferable. The modifications that we propose help to do this by choosing large entries when possible for the early transversal entries.

It is beneficial to first permute the matrix to block triangular form and then to use BT only on the blocks on the diagonal. This can be done since all entries in any maximum transversal must lie in these blocks. Furthermore, not only does this mean that BT operates on smaller matrices, but we also usually obtain a transversal of better quality inasmuch as not only is the minimum diagonal entry maximized but this is true for each block on the diagonal. Thus, for matrix (2.2) the combination of an ordering to block triangular form followed by BT would yield the preferred transversal $\{(2,1),(1,2),(3,3)\}$.

There are other possibilities for improving the diagonal values of the permuted matrix which are not the smallest. One is to apply a row scaling subsequent to an initial column scaling of the matrix $\boldsymbol{A}$. This will increase the numerical values of all the nonzero entries in those rows for which the maximum absolute numerical value is less than 1. A row scaling applied to the matrix (2.2) changes the coefficient a_{33} from δ to 1.0, and now Algorithm BT will compute $\{(2,1),(1,2),(3,3)\}$ as the bottleneck transversal of the matrix (2.2). Unfortunately, such a row scaling does not always help, as can be seen by the matrix

$$\begin{pmatrix} \delta & 1.0 & 1.0 \\ 1.0 & \delta & \\ 1.0 & & \delta \end{pmatrix}$$

with the maximum transversals $\{(1,1),(2,2),(3,3)\}$, $\{(2,1),(1,2),(3,3)\}$, and $\{(1,3),(2,2),(3,1)\}$, all legitimate bottleneck transversals. Indeed, the BT algorithm is very dependent on scaling. For example, the matrix $\left(\begin{smallmatrix} 1 & 2 \\ 3 & 4 \end{smallmatrix}\right)$ has bottleneck transversal $\{(2,1),(1,2)\}$ whereas, if it is row scaled to $\left(\begin{smallmatrix} 4 & 8 \\ 3 & 4 \end{smallmatrix}\right)$, the bottleneck transversal is $\{(1,1),(2,2)\}$.

Another possibility for improving the size of the diagonal values is to apply Algorithm BT repeatedly. Without loss of generality, suppose that, after application of BT, entry a_{nn} has the smallest diagonal value. Algorithm BT can then be applied to the $(n-1)\times(n-1)$ leading principal submatrix of $\boldsymbol{A}$, and this could be repeated until (after k steps) the $(n-k)\times(n-k)$ leading principal submatrix of $\boldsymbol{A}$ contains only 1s (on the assumption that the original matrix was row and column scaled). Obviously, this can be quite expensive, since Algorithm BT is applied $\mathcal{O}(n)$ times although we have a good starting point for the BT algorithm at each stage. We call this algorithm the successive bottleneck transversal algorithm. Because of this and because we have found that it usually gives little improvement over BT, we do not consider it further in this paper.

2.3. Maximum product transversals. An algorithm yielding the same transversal independent of scaling is to maximize the product of the moduli of entries on the diagonal, that is, to find a permutation σ so that

$$\left| \prod_{i=1}^{n} a_{i\sigma_i} \right| \tag{2.3}$$

is maximized. This is the strategy used for pivoting in full Gaussian elimination by Olschowka and Neumaier [16] and corresponds to obtaining a weighted bipartite matching. Olschowka and Neumaier [16] combine a permutation and scaling strategy. The permutation, as in (2.3), maximizes the product of the diagonal entries of the permuted matrix. (Clearly the product is zero if and only if the matrix is structurally singular.) The scaling transforms the matrix into a so-called $\boldsymbol{I}$-matrix, whose diagonal entries are all 1 and whose off-diagonal entries are all less than or equal to 1.

Maximizing the product of the diagonal entries of $\boldsymbol{A}$ is equivalent to minimizing the sum of the diagonal entries of a matrix $\boldsymbol{C} = (c_{ij})$ that is defined as follows (we here assume that $\boldsymbol{A} = (a_{ij})$ denotes an $n \times n$ nonnegative nonsingular matrix):

$$c_{ij} = \begin{cases} \log \overline{a}_j - \log a_{ij}, & a_{ij} \neq 0, \\ 0, & a_{ij} = 0, \end{cases}$$

where $\overline{a}_j = \max_k(a_{kj})$ is the maximum absolute value in column j of matrix $\boldsymbol{A}$.

Minimizing the sum of the diagonal entries can be stated in terms of an assignment problem and can be solved in $\mathcal{O}(n^3)$ time for full $n \times n$ matrices or in $\mathcal{O}(n\tau \log n)$ time for sparse matrices with τ entries. A bipartite weighted matching algorithm is used to solve this problem. Applying this algorithm to $\boldsymbol{C}$ produces vectors u, v and a transversal T, all of length n, such that

$$\begin{cases} u_i + v_j = c_{ij}, & (i,j) \in T, \\ u_i + v_j \leq c_{ij}, & (i,j) \notin T. \end{cases}$$

If we define

$$\begin{aligned} \boldsymbol{D}_1 &= diag(d_1^1, d_2^1, \ldots, d_n^1), \quad d_i^1 = \exp(u_i), \\ \boldsymbol{D}_2 &= diag(d_1^2, d_2^2, \ldots, d_n^2), \quad d_j^2 = \exp(v_j)/\overline{a}_j, \end{aligned}$$

then the scaled matrix $\boldsymbol{B} = \boldsymbol{D}_1 \boldsymbol{A} \boldsymbol{D}_2$ is an $\boldsymbol{I}$-matrix. We do not do this scaling in our experiments; unlike Olschowka and Neumaier, we use a *sparse* bipartite weighted matching, whereas they considered only full matrices.

The worst-case complexity of this algorithm is $\mathcal{O}(n\tau \log n)$. This is similar to BT, although in practice it sometimes requires more work than BT. We have programmed this algorithm, without the final scaling. We have called it algorithm MPD (for maximum product on diagonal) and compare it with BT and MC21 in the later sections of this paper. Note that on the matrix

$$\begin{pmatrix} 2/\epsilon & & 1 \\ & 2/\epsilon & \\ 1 & & \epsilon \end{pmatrix}$$

the MPD algorithm obtains the transversal $\{(1,1), (2,2), (3,3)\}$ whereas, for example, for Gaussian elimination down the diagonal the transversal $\{(1,3), (2,2), (3,1)\}$ would be better. Additionally, the fact that scaling does influence the choice of bottleneck transversal could be deemed a useful characteristic.

3. Implementation of the BT algorithm. We now consider implementation details of Algorithm BT from the previous section. We will also illustrate its performance on some matrices from the Harwell–Boeing Collection [8] and the collection of Davis [5]. A code implementing the BT algorithm will be included in a future release of the Harwell Subroutine Library [15].

When we are updating the transversal at stage ($\star$) of Algorithm BT we can easily accelerate the algorithm described in section 2 by computing the value of the minimum entry of the transversal, viz.

$$\min_{(i,j)\in T} |a_{ij}|, \tag{3.1}$$

and then setting ϵmin to this value rather than to ϵ. The other issue, crucial for efficiency, is the choice of ϵ at the beginning of each step. If, at each step, we choose ϵ close to the value of ϵmin, then it is highly likely that we will find a maximum transversal, but the total number of steps required to obtain the bottleneck transversal can be very large. In the worst case, we could require $\tau - n$ steps when the number of nonzero entries in $\boldsymbol{A}_\epsilon$ reduces by only one at each iteration.

The algorithm converges faster if the size of the interval $(\epsilon min, \epsilon max)$ reduces significantly at each step. It therefore would appear sensible to choose ϵ at each step so that the interval is split into two almost-equal subintervals, that is, $\epsilon \approx (\epsilon min + \epsilon max)/2$. However, if most of the nonzero values in $\boldsymbol{A}$ that have a magnitude between ϵmin and ϵmax are clustered near one of these endpoints, the possibility exists that only a few nonzero values are discarded and the algorithm again will proceed slowly. To avoid this, ϵ should be chosen as the median of the nonzero values between ϵmin and ϵmax.

We now consider how a transversal algorithm like `MC21` can be modified to implement Algorithm BT efficiently. Before doing this, it is useful to describe briefly how `MC21` works. Each column of the matrix is searched in turn (called an original column) and either an entry in a row with no transversal entry currently in the row is found and this is made a transversal entry (a *cheap assignment*), or there is no such entry and so the search moves to a previous column whose transversal entry is in one of the rows with an entry in the original column. This new column is then checked for a cheap assignment. If one exists, then this cheap assignment and the entry in the original column in the row of the old transversal entry replace that as transversal entries, thereby extending the length of the transversal by 1. If there is no cheap assignment, the search continues to other columns in a depth-first search fashion until a chain or *augmenting path* of the form

$$\{(a_{j_1,j}), (a_{j_1,j_2}), (a_{j_3,j_2}), \ldots, (a_{i,j_k})\}$$

is found, where there are no transversal entries in row i and every even member of the path is a transversal entry. The assignment is made in column j and the transversal is extended by 1 by replacing all transversal entries in the augmenting path with the odd members of this path.

Transversal selection algorithms like `MC21` do not take into account the numerical values of the nonzero entries. However, it is clear that Algorithm BT will converge faster if T is chosen so that the value of its minimum entry is large. We do this by noting that, when constructing an augmenting path, there are often several candidates for a cheap assignment or for extending the path. `MC21` makes an arbitrary choice and we have modified it so the candidate with largest absolute value is chosen. Note that this is a local strategy and does not guarantee that augmenting paths with the highest values will be found.

The second modification aims at exploiting information obtained from previous steps of Algorithm BT. Algorithm BT repeatedly computes a maximum transversal $T = MT(A_\epsilon, T'_\epsilon)$. The implementation of `MC21` in the Harwell Subroutine Library computes T from scratch, so we have modified it so that it can start with a partial transversal. This can be achieved easily by holding the set of columns which contain entries of the partial transversal and performing the depth search through that set of columns.

Of course, there are many ways to implement the choice of ϵ. One alternative is to maintain an array PTR (of length τ) of pointers, such that the entries in the first part of PTR point to those entries in $\boldsymbol{A}$ that form matrix $\boldsymbol{A}_{\epsilon_{max}}$, the first two parts of PTR point to the entries that form $\boldsymbol{A}_{\epsilon_{min}}$, and the elements in the third part of PTR point to all the remaining (smaller) entries of $\boldsymbol{A}$. A new value for ϵ can then be chosen directly ($\mathcal{O}(1)$ time) by picking the numerical value of an entry that is pointed to by an element of the second part of PTR. After the assignment in Algorithm BT to either ϵ_{min} or ϵ_{max}, the second part of PTR has to be permuted so that PTR again can be divided into three parts. An alternative is to do a global sort (using a fast sorting algorithm) on all the entries of $\boldsymbol{A}$, such that the elements of PTR point to the entries in order of decreasing absolute value. Then again PTR can be divided into three parts as described in the previous alternative. By choosing (in $\mathcal{O}(1)$ time) ϵ equal to the numerical value of the entry pointed to by the median element of the second part of PTR, ϵ will divide the interval $(\epsilon_{min}, \epsilon_{max})$ into parts of close-to-equal size. Both alternatives have the advantage of being able to choose the new ϵ quickly, but they require $\mathcal{O}(\tau)$ extra memory and (repeated) permutations of the pointers.

We prefer an approach that is less expensive in memory and that matches our transversal algorithm better. Since `MC21` always searches the columns in order, we facilitate the construction of the matrices $\boldsymbol{A}_\epsilon$, by first sorting the entries in each column of the matrix $\boldsymbol{A}$ by decreasing absolute value. For a sparse matrix with a well-bounded number of entries in each column, this can be done in $\mathcal{O}(n)$ time. The matrix $\boldsymbol{A}_\epsilon$ is then implicitly defined by an array LEN of length n with LEN[j] pointing to the first entry in column j of matrix $\boldsymbol{A}$ whose value is smaller than ϵ, which is the position immediately after the end of column j of matrix $\boldsymbol{A}_\epsilon$. Since the entries of a column of $\boldsymbol{A}_\epsilon$ are contiguous, the repeated modification of ϵ by Algorithm BT, which redefines matrix $\boldsymbol{A}_\epsilon$, corresponds to simply changing the pointers in the array LEN.

The actual choice of ϵ at phase ($\star$) in Algorithm BT is done by selecting in matrix $\boldsymbol{A}_{\epsilon_{min}}$ an entry that has an absolute value X such that $\epsilon_{min} < X \leq \epsilon_{max}$. The columns of $\boldsymbol{A}_{\epsilon_{min}}$ are searched until such an entry is found and ϵ is set to its absolute value. This search costs $\mathcal{O}(n)$ time since, for each column, we have direct access to the entries with absolute values between ϵ_{min} and ϵ_{max} through the pointer array LEN.

As mentioned before, by choosing ϵ carefully we can speed up Algorithm BT considerably. Therefore, instead of choosing an arbitrary entry from the matrix to define ϵ, we can choose a number (k, say) of entries lying between ϵ_{min} and ϵ_{max} at random, sort them by absolute value, and then set ϵ to the absolute value of the median element.[2] In our implementation we used $k = 10$.

[2] This is a technique commonly used to speed up sorting algorithms like quicksort.

Table 3.1

Times for transversal algorithms. The order of matrix is n and the number of entries is τ.

Matrix	n	τ	Time in secs		
			MC21	BT	MPD
MAHINDAS	1258	7682	0.01	0.01	0.02
ORANI678	2529	90158	0.02	0.10	0.13
RDIST1	4134	94408	0.02	0.18	0.37
GEMAT11	4929	33185	0.01	0.03	0.04
GOODWIN	7320	324784	0.27	2.26	1.82
ONETONE1	36057	341088	2.67	0.70	0.61
ONETONE2	36057	227628	2.63	0.53	0.42
TWOTONE	120750	1224224	60.10	6.95	2.17
LHR02	2954	37206	0.04	0.14	0.17
LHR14C	14270	307858	0.28	1.13	3.32
LHR71C	70304	1528092	1.86	9.00	37.73
AV41092	41092	1683902	35.72	10.81	65.13

The set of matrices that we used for our experiments are unsymmetric matrices taken from the sparse matrix collections [9] and [5]. Table 3.1 shows the order, number of entries, and time to compute a bottleneck transversal for each matrix. All matrices are initially row and column scaled. By this we mean that the matrix is scaled so that the maximum entry in each row and in each column is 1.

The machine used for the experiments in this and the following sections is a 166 MHz Sun ULTRA-2. The algorithms are implemented in Fortran 77.

4. The solution of equations by direct methods. MCSPARSE, a parallel direct unsymmetric linear system solver developed by Gallivan, Marsolf, and Wijshoff [13], uses a reordering to identify a priori large- and medium-grain parallelism and to reorder the matrix to bordered block triangular form. Their ordering uses an initial nonsymmetric ordering that enhances the numerical properties of the factorization, and subsequent symmetric orderings are used to obtain a bordered block triangular matrix [19]. The nonsymmetric ordering is effectively a modified version of MC21. During each search phase, for both a cheap assignment and an augmenting path, an entry a_{ij} is selected only if its absolute value is within a bound α, $0 \leq \alpha \leq 1$, of the largest entry in column j. Instead of taking the first entry that is found by the search that satisfies the threshold, the algorithm scans all of the column for the entry with the largest absolute value.

The algorithm starts off with an initial bound $\alpha = 0.1$. If a maximum transversal cannot be found, then the values in each column are examined to determine the maximum value of the bound that would have allowed an assignment to take place for that column. The new bound is then set to the minimum of the bound estimates from all the failed columns and the algorithm is restarted. If a bound less than a preset limit is tried and a transversal is still not found, then the bound is ignored and the code finds any transversal. In our terminology (assuming an initial column scaling of the matrix) this means that a maximum transversal of size n is computed for the matrix $\boldsymbol{A}_\alpha$.

In the multifrontal approach of Duff and Reid [10], later developed by Amestoy and Duff [2], an analysis is performed on the structure of $\boldsymbol{A} + \boldsymbol{A}^T$ to obtain an ordering that reduces fill-in under the assumption that all diagonal entries will be numerically suitable for pivoting. The approximate minimum degree (AMD) algorithm

Table 4.1
Number of delayed pivots in factorization from MA41. *A dash indicates that* MA41 *requires a real working space larger than* 25 *million words (of* 8 *bytes).*

Matrix	Transversal algorithm used			
	None	MC21	BT	MPD
GEMAT11	-	76	0	0
ONETONE1	-	16261	255	100
ONETONE2	40916	8310	214	100
GOODWIN	536	1622	358	53
LHR02	3432	388	211	56
LHR14C	-	7608	3689	169
LHR71C	-	35354	-	2643
AV41092	-	10151	2143	1730

Table 4.2
Number of entries in the factors from MA41.

Matrix	Transversal algorithm used			
	None	MC21	BT	MPD
GEMAT11	-	127819	78589	78161
ONETONE1	-	10359161	6957229	4715085
ONETONE2	14082683	2875603	2167523	2169903
GOODWIN	1263104	2673318	1791112	1282004
LHR02	2298550	333450	394724	235048
LHR14C	-	3111142	4596028	2164392
LHR71C	-	18786982	-	11599556
AV41092	-	16226036	15140098	14110336

of Amestoy, Davis, and Duff [1] is used to obtain the ordering to reduce the fill-in. The numerical factorization is guided by an assembly tree. At each node of the tree, some steps of Gaussian elimination are performed on a dense submatrix whose Schur complement is then passed to the parent node in the tree where it is assembled (or summed) with Schur complements from the other children and original entries of the matrix. If, however, numerical considerations prevent us from choosing a pivot then the algorithm can proceed, but now the Schur complement that is passed to the parent is larger and usually more work and storage will be needed to effect the factorization.

The logic of first permuting the matrix so that there are large entries on the diagonal, before computing the ordering to reduce fill-in, is to try to reduce the number of pivots that are delayed in this way, thereby reducing storage and work for the factorization. We show the effect of this in Table 4.1, where we can see that even using MC21 can be very beneficial although the BT algorithm can show significant further gains, and sometimes the use of MPD can cause further significant reduction in the number of delayed pivots. We should add that the numerical accuracy of the solution is sometimes slightly improved by these permutations and, in all cases, good solutions were found.

In Table 4.2, we show the effect of this on the number of entries in the factors. Clearly this mirrors the results in Table 4.1 and shows the benefits of the transversal selection algorithms. This effect is seen in Table 4.3, where we can sometimes observe a dramatic reduction in time for solution when preceded by a permutation. In all

TABLE 4.3
Time (in seconds on Sun ULTRA-2) for `MA41` *for solution of system.*

Matrix	Transversal algorithm used			
	None	`MC21`	BT	MPD
GEMAT11	-	0.28	0.20	0.20
ONETONE1	-	225.71	83.11	44.22
ONETONE2	81.45	17.05	9.48	11.54
GOODWIN	3.64	14.63	6.00	3.56
LHR02	24.85	1.07	1.29	0.58
LHR14C	-	12.66	29.17	5.88
LHR71C	-	148.07	-	43.33
AV41092	-	226.20	184.68	155.70

cases, an AMD ordering has been used on the permuted matrix.

In addition to allowing selection of the pivots chosen by the analysis phase, the multifrontal code `MA41` will do better on matrices whose structure is symmetric or nearly so. The transversal orderings in some cases increase the symmetry of the resulting reordered matrix. This is particularly apparent when we have a very sparse system with many zeros on the diagonal. In that case, the reduction in number of off-diagonal entries in the reordered matrix has an influence on the symmetry. Notice that, in this respect, the more sophisticated transversal algorithms may actually cause problems since they could reorder a symmetrically structured matrix with a zero-free diagonal, whereas `MC21` will leave it unchanged.

5. The solution of equations by iterative methods. A large family of iterative methods, the so-called stationary methods, has the iteration scheme

$$\boldsymbol{M}\boldsymbol{x}^{(k+1)} = \boldsymbol{N}\boldsymbol{x}^{(k)} + \boldsymbol{b}, \tag{5.1}$$

where $\boldsymbol{A} = \boldsymbol{M} - \boldsymbol{N}$ is a *splitting* of $\boldsymbol{A}$ and $\boldsymbol{M}$ is chosen such that a system of the form $\boldsymbol{M}\boldsymbol{x} = \boldsymbol{y}$ is easy to solve. If $\boldsymbol{M}$ is invertible, (5.1) can be written as

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{M}^{-1}\boldsymbol{N}\boldsymbol{x}^{(k)} + \boldsymbol{M}^{-1}\boldsymbol{b} = (\boldsymbol{I} - \boldsymbol{M}^{-1}\boldsymbol{A})\boldsymbol{x}^{(k)} + \boldsymbol{M}^{-1}\boldsymbol{b}\ . \tag{5.2}$$

We have

$$\rho(\boldsymbol{M}^{-1}\boldsymbol{N}) \leq ||\boldsymbol{M}^{-1}\boldsymbol{N}||_\infty \leq ||\boldsymbol{M}^{-1}||_\infty ||\boldsymbol{N}||_\infty,$$

where ρ is the spectral radius, so that if $||\boldsymbol{M}^{-1}||_\infty ||\boldsymbol{N}||_\infty < 1$, convergence of the iterates $\boldsymbol{x}^{(k)}$ to the solution $\boldsymbol{A}^{-1}\boldsymbol{b}$ is guaranteed for arbitrary $\boldsymbol{x}^{(0)}$. In general, the smaller $||\boldsymbol{M}^{-1}||_\infty ||\boldsymbol{N}||_\infty$, the faster the convergence. Thus an algorithm that makes entries in $\boldsymbol{M}$ large and those in $\boldsymbol{N}$ small should be beneficial.

The simplest method of this type is the Jacobi method, corresponding to the splitting $\boldsymbol{M} = \boldsymbol{D}$ and $\boldsymbol{N} = -(\boldsymbol{L} + \boldsymbol{U})$, where $\boldsymbol{D}$ denotes the diagonal, $\boldsymbol{L}$ the strictly lower triangular part, and $\boldsymbol{U}$ the strictly upper triangular part of the matrix $\boldsymbol{A}$. However, this is not a particularly current or powerful method so we conduct our experiments using the block Cimmino implementation of Arioli et al. [3], which is equivalent to using a block Jacobi algorithm on the normal equations. In this implementation, the subproblems corresponding to blocks of rows from the matrix are solved by a direct method similar to that considered in the previous section. For similar reasons, it can

Table 5.1
Number of iterations of block Cimmino algorithm on MAHINDAS.

Acceleration +	Transversal algorithm			
# block rows	None	MC21	BT	MPD
CG(1)				
2	324	267	298	295
4	489	383	438	438
8	622	485	532	524
16	660	572	574	574
CG(4)				
2	148	112	130	133
4	212	190	199	194
8	261	235	232	233
16	281	245	253	253
CG(8)				
2	80	62	72	75
4	117	105	109	108
8	140	133	127	130
16	151	142	137	136

be beneficial to increase the magnitude of the diagonal entries through unsymmetric permutations.

We show the effect of this in Table 5.1, where we see that the number of iterations for the solution of the problem MAHINDAS ($n = 1258$, $\tau = 7682$). The convergence tolerance was set to 10^{-12}. The transversal selection algorithm was followed by a reverse Cuthill–McKee algorithm to obtain a block tridiagonal form. The matrix was partitioned in 2, 4, 8, and 16 block rows and the acceleration used was a block CG algorithm with block sizes of 1, 4, and 8.

In every case, the use of a transversal algorithm accelerates the convergence of the method, sometimes by a significant amount. However, the use of the algorithms to increase the size of the diagonal entries does not usually help convergence further. The convergence of block Cimmino depends on angles between subspaces which are not so strongly influenced by the diagonal entries.

6. Preconditioning. In this section, we consider the effect of using a permutation induced by our transversal algorithms prior to solving a system using a preconditioned iterative method. We consider preconditionings corresponding to incomplete factorizations of the form ILU(0), ILU(1), and ILUT and study the convergence of the iterative methods GMRES(20), BiCGSTAB, and QMR. We refer the reader to a standard text like that of Saad [18] for a description and discussion of these methods. Since the diagonal of the permuted matrix is "more dominant" than the diagonal of the original matrix, we would hope that such permutations would enhance convergence.

We show the results of some of our runs in Table 6.1. The maximum number of iterations was set to 1000 and the convergence tolerance to 10^{-9}. It is quite clear that the reorderings can have a significant effect on the convergence of the preconditioned iterative method. In some cases, the method will converge only after the permutation; in others it greatly improves the convergence. It would appear from the results in Table 6.1 and other experiments we have performed that the more sophisticated MPD transversal algorithm generally results in the greatest reduction

Table 6.1
Number of iterations required by some preconditioned iterative methods.

Matrix and method		Transversal algorithm MC21	 BT	 MPD
IMPCOL E				
ILU(0)	GMRES(20)	-	16	15
	BiCGSTAB	123	21	11
	QMR	101	26	17
ILU(1)	GMRES(20)	59	15	11
	BiCGSTAB	98	16	8
	QMR	72	19	12
ILUT	GMRES(20)	8	7	8
	BiCGSTAB	9	5	5
	QMR	10	8	8
MAHINDAS				
ILU(0)	GMRES(20)	-	-	179
	BiCGSTAB	-	-	39
	QMR	-	-	55
ILU(1)	GMRES(20)	-	-	69
	BiCGSTAB	-	-	26
	QMR	-	-	34
ILUT	GMRES(20)	-	-	15
	BiCGSTAB	-	-	11
	QMR	-	-	17
WEST0497				
ILU(0)	GMRES(20)	-	60	19
	BiCGSTAB	-	78	22
	QMR	-	63	23
ILU(1)	GMRES(20)	-	79	15
	BiCGSTAB	-	82	15
	QMR	-	82	18
ILUT	GMRES(20)	-	15	10
	BiCGSTAB	-	15	7
	QMR	-	17	12

in the number of iterations, although the best method will depend on the overall solution time including the transversal selection algorithm.

7. Conclusions and future work. We have described algorithms for obtaining transversals with large entries and have indicated how they can be implemented showing that resulting programs can be written for efficient performance.

While it is clear that reordering matrices so that the permuted matrix has a large diagonal can have a very significant effect on solving sparse systems by a wide range of techniques, it is somewhat less clear that there is a universal strategy that is best in all cases. We have thus started experimenting with combining the strategies mentioned in this paper and, particularly for the block Cimmino approach, with combining our unsymmetric ordering with a symmetric ordering. One example that we plan to study is a combination with the symmetric TPABLO ordering [4].

It is possible to extend our techniques to orderings that try to increase the size of not just the diagonal but also the immediate sub and super diagonals and then use the resulting tridiagonal part of the matrix as a preconditioner.

One can also build other criteria into the weighting for obtaining a bipartite matching, for example, to incorporate a Markowitz count so that sparsity would also be preserved by the choice of the resulting diagonal as a pivot.

Finally, we noticed in our experiments with MA41 that one effect of transversal selection was to increase the structural symmetry of unsymmetric matrices. We are thus exploring further the use of ordering techniques that more directly attempt to increase structural symmetry.

Acknowledgments. We are grateful to Patrick Amestoy, Michele Benzi, and Daniel Ruiz for their assistance with the experiments on the direct methods, the preconditioned iterative methods, and the block iterative methods, respectively. We would also like to thank Alex Pothen for some early discussions on bottleneck transversals and John Reid and Jennifer Scott for comments on a draft of this paper.

REFERENCES

[1] P. R. Amestoy, T. A. Davis, and I. S. Duff, *An approximate minimum degree ordering algorithm*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 886–905.

[2] P. R. Amestoy and I. S. Duff, *Vectorization of a multiprocessor multifrontal code*, Internat. J. Supercomputer Appl., 3 (1989), pp. 41–59.

[3] M. Arioli, I. Duff, J. Noailles, and D. Ruiz, *A block projection method for sparse matrices*, SIAM J. Sci. Statistical Comput., 13 (1992), pp. 47–70.

[4] M. Benzi, H. Choi, and D. Szyld, *Threshold ordering for preconditioning nonsymmetric problems*, in Proc. Workshop on Scientific Computing '97, Hong Kong, G. H. Golub, ed., Lecture Notes in Comput. Sci., Springer-Verlag, Berlin, New York, 1997, pp. 159–165.

[5] T. A. Davis, *University of Florida Sparse Matrix Collection*, http://www.cise.ufl.edu/~davis, ftp://ftp.cise.ufl.edu/pub/faculty/davis (1997).

[6] I. S. Duff, *MA28—A Set of Fortran Subroutines for Sparse Unsymmetric Linear Equations*, Tech. report AERE R8730, Her Majesty's Stationery Office, London, UK, 1977.

[7] I. S. Duff, *The design and use of a frontal scheme for solving sparse unsymmetric equations*, in Numerical Analysis, Proc. 3rd IIMAS Workshop, Lecture Notes in Math. 909, J. P. Hennart, ed., Springer-Verlag, Berlin, New York, pp. 240–247.

[8] I. S. Duff, R. G. Grimes, and J. G. Lewis, *Sparse matrix test problems*, ACM Trans. Math. Software, 15 (1989), pp. 1–14.

[9] I. S. Duff, R. G. Grimes, and J. G. Lewis, *Users' Guide for the Harwell-Boeing Sparse Matrix Collection (Release* I*)*, Tech. report RAL 92-086, Rutherford Appleton Laboratory, Oxon, UK, 1992.

[10] I. S. Duff and J. K. Reid, *The multifrontal solution of indefinite sparse symmetric linear systems*, ACM Trans. Math. Software, 9 (1983), pp. 302–325.

[11] I. S. Duff and T. Wiberg, *Remarks on implementation of $\mathcal{O}(n^{1/2}\tau)$ assignment algorithms*, ACM Trans. Math. Software, 14 (1988), pp. 267–287.

[12] D. Fulkerson, I. Glicksberg, and O. Gross, *A Production Line Assignment Problem*, Tech. report RM-1102, Rand Corporation, 1953.

[13] K. A. Gallivan, B. A. Marsolf, and H. A. G. Wijshoff, *Solving large nonsymmetric sparse linear systems using MCSPARSE*, Parallel Comput., 22 (1996), pp. 1291–1333.

[14] J. E. Hopcroft and R. M. Karp, *An $n^{(5/2)}$ algorithm for maximum matchings in bipartite graphs*, SIAM J. Comput., 2 (1973), pp. 225–231.

[15] HSL, *Harwell Subroutine Library. A Catalogue of Subroutines (Release* 12*)*, AEA Technology, Harwell Laboratory, Oxfordshire, UK, 1996.

[16] M. Olschowka and A. Neumaier, *A new pivoting strategy for Gaussian elimination*, Linear Algebra Appl., 240 (1996), pp. 131–151.

[17] A. Pothen and C. Fan, *Computing the block triangular form of a sparse matrix*, ACM Trans. Math. Software, 16 (1990), pp. 303–324.

[18] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing, New York, 1996.

[19] H. A. G. Wijshoff, *Symmetric Orderings for Unsymmetric Sparse Matrices*, Tech. report CSRD 901, Center for Supercomputing Research and Development, University of Illinois, Urbana, IL, 1989.

SIAM J. MATRIX ANAL. APPL.
Vol. 20, No. 4, pp. 902–914

PERFORMANCE OF GREEDY ORDERING HEURISTICS FOR SPARSE CHOLESKY FACTORIZATION*

ESMOND G. NG† AND PADMA RAGHAVAN‡

Abstract. Greedy algorithms for ordering sparse matrices for Cholesky factorization can be based on different metrics. Minimum degree, a popular and effective greedy ordering scheme, minimizes the number of nonzero entries in the rank-1 update (degree) at each step of the factorization. Alternatively, minimum deficiency minimizes the number of nonzero entries introduced (deficiency) at each step of the factorization. In this paper we develop two new heuristics: modified minimum deficiency (MMDF) and modified multiple minimum degree (MMMD). The former uses a metric similar to deficiency while the latter uses a degree-like metric. Our experiments reveal that on the average, MMDF orderings result in 21% fewer operations to factor than minimum degree; MMMD orderings result in 15% fewer operations to factor than minimum degree. MMMD requires on the average 7–13% more time than minimum degree, while MMDF requires on the average 33–34% more time than minimum degree.

Key words. sparse matrix ordering, minimum degree, minimum deficiency, greedy heuristics

AMS subject classifications. 65F05, 65F50

PII. S0895479897319313

1. Introduction.

It is well known that ordering the rows and columns of a matrix is a crucial step in the solution of sparse linear systems using Gaussian elimination. The ordering can drastically affect the amount of fill introduced during factorization and hence the cost of computing the factorization [8, 14]. When the matrix is symmetric and positive definite, the ordering step is independent of the numerical values and can be performed prior to numerical factorization. An ideal choice, for example, is an ordering that introduces the least fill. However, the problem of computing such an optimal ordering is NP-complete [24]. Consequently, almost all ordering algorithms are heuristic in nature. Examples include reverse Cuthill–McKee [6, 7, 9], automatic nested dissection [10], and minimum degree [20].

A greedy ordering heuristic numbers columns successively by selecting at each step a column with the optimal value of a metric. In the minimum degree algorithm of Tinney and Walker [23] the metric is the number of operations in the rank-1 update associated with a column in a right-looking, sparse Cholesky factorization. The algorithm can be stated in terms of vertex eliminations in a graph representing the matrix [19, 20]. In this framework, the number of operations in the rank-1 update is proportional to the square of the degree of a vertex; consequently, implementations use the degree as the metric. Efficient implementations of minimum degree are due to George and Liu [12, 13]. The minimum degree algorithm with multiple eliminations

*Received by the editors March 28, 1997; accepted for publication (in revised form) by J. Liu March 25, 1998; published electronically July 9,1999. This work was supported in part by the Defense Advanced Research Projects Agency under contracts DAAL03-91-C-0047, ERD9501310, and Xerox-MP002315, by the Applied Mathematical Sciences Research Program, Office of Energy Research, U.S. Department of Energy under contract DE-AC05-96OR22464 with Lockheed Martin Energy Research Corp., and by the National Science Foundation under grants NSF-ASC-94-11394 and NSF-CDA-9529459.

http://www.siam.org/journals/simax/20-4/31931.html

†Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831-6367 (ngeg@ornl.gov).

‡107 Ayres Hall, Department of Computer Science, The University of Tennessee, Knoxville, TN 37996-1301 (padma@cs.utk.edu).

(MMD), due to Liu [17], has become very popular in the last decade. Multiple independent vertices are eliminated at a single step in MMD to reduce the ordering time. More recently, Amestoy, Davis, and Duff [2] have developed the approximate minimum degree (AMD) algorithm. AMD uses an approximation to the degree to further reduce the ordering time without degrading the quality of orderings produced. Berman and Schnitger [5] have analytically shown that the minimum degree algorithm can, in some rare cases, produce a poor ordering. However, experience has shown that the minimum degree algorithm and its variants are effective heuristics for generating fill-reducing orderings. In fact, only some very recent separator-based schemes [4, 15, 16] have outperformed MMD for certain classes of sparse matrices. Two of these new schemes are hybrids of a separator-based scheme and a greedy ordering strategy such as the minimum degree algorithm.

A greedy ordering heuristic that was also proposed by Tinney and Walker [23], but has largely been ignored, is the minimum deficiency (or minimum fill) algorithm. The minimum deficiency algorithm minimizes the number of fill entries *introduced* at each step of sparse Cholesky factorization (or *deficiency* in graph terminology). Although the metrics look similar, the minimum deficiency and minimum degree algorithms are different. For example, the deficiency could well be zero even when the degree is not. There are two reasons why the minimum deficiency algorithm has not become as popular as the minimum degree algorithm [8]. First, the minimum deficiency algorithm is typically much more expensive than the minimum degree algorithm. Second, it was believed that the quality of minimum deficiency orderings is not much better than that of minimum degree orderings [8]. Recent results by Rothberg [21] (and also by us [18]) demonstrate that minimum deficiency leads to significantly better orderings than minimum degree. However, current implementations of the minimum deficiency algorithm are slower than MMD by more than an order of magnitude [18, 21].

In this paper, we develop two greedy heuristics that produce better orderings than minimum degree; they result in orderings that require 15–21% fewer operations in the factorization step than MMD. The heuristics are variants of minimum deficiency and minimum degree requiring on the average 7–34% more time than MMD. In section 2, we provide background material and introduce notation to help describe our heuristics. In section 3 we develop our MMDF and MMMD heuristics. We also show that the two heuristics can be implemented using the update mechanism in the AMD scheme of Amestoy, Davis, and Duff [2]. In section 4 we provide empirical results on the performance of MMDF and MMMD. Section 5 contains some concluding remarks. The remaining part of this section describes recent related work.

Related work. Rothberg has investigated metrics for greedy ordering schemes based on approximations to the deficiency [21]. His work and our work [18] were done independently of each other.[1] In [21] Rothberg

- shows that the minimum deficiency algorithm is significantly superior to MMD in terms of the quality of the orderings; on the average, relative to MMD, minimum deficiency orderings produce 14% fewer nonzeros in the Cholesky factor and require 28% fewer operations to compute the Cholesky factor;
- develops three approximate minimum fill (AMF) heuristics based on approximations to the deficiency;

[1] Raghavan and Rothberg presented their results independently at the 2nd SIAM Conference on Sparse Matrices in 1996.

- concludes that heuristic AMF1 (renamed AMF in [22]) is the best among the three; on the average, AMF1 orderings require 14% fewer operations in the factorization step than MMD orderings.

In our earlier report [18], we

- establish that many of the techniques used in efficient implementations of the minimum degree algorithm (namely, indistinguishable vertices, mass elimination, and outmatching) also apply to the minimum deficiency algorithm;
- corroborate Rothberg's empirical results, establishing the superior performance of the minimum deficiency metric;
- develop our MMDF and MMMD heuristics;
- show that MMDF (MMMD) orderings require 17% (15%) fewer operations in the factorization step than MMD (on the average).

It is difficult to compare the results in [18] and [21] directly because the test suites used in the two papers are substantially different. The aggregate measures reported in the two papers are also different. Moreover, they were based on performance data obtained from different sets of initial numberings.

More recently, in a revision of [21], Rothberg and Eisenstat have developed two new metrics for greedy ordering schemes [22]. In [22] Rothberg and Eisenstat

- develop two heuristics, approximate minimum mean fill (AMMF) and average minimum increase in neighbor degree (AMIND);
- show that AMMF orderings require 22% (median) to 25% (geometric mean) fewer operations in the factorization step than MMD orderings; AMIND orderings require 20% (median) to 21% (geometric mean) fewer operations to factor than MMD orderings.

This paper is a shorter version of [18]. The test suite in this paper is substantially different from that in the original paper. In an attempt to compare the performance of our heuristics with those of Rothberg and Eisenstat (AMF, AMMF, and AMIND, [21, 22]), we now use nearly the same test suite as in their reports. Four of the matrices in [21] and [22] are proprietary, and therefore are not available to us. To report performance relative to MMD, we had earlier used the "median of ratios" over seven initial orderings (six random orderings and the ordering in which the matrix was given to us). In this paper, we use the "ratio of medians" over 11 random initial orderings (as in [21, 22]). As we will see later in the paper, our MMDF and MMMD heuristics produce better orderings than MMD. The MMDF and MMMD orderings are very competitive with those produced by AMF, AMMF, and AMIND. What we see as interesting is the development of five different metrics that can produce orderings that are significantly better than those produced by MMD. We had commented in our earlier report [18] that there could well be other relatively inexpensive greedy strategies that outperform the ones known at that time. The performance of newer schemes AMMF and AMIND [22] supports our prediction; AMMF seems to be slightly better than our MMDF. As we discuss in section 5, we still believe that there may well be other greedy metrics that perform better than the five developed so far.

2. Implementing greedy ordering heuristics. The efficient implementation of greedy ordering schemes is based on a compact realization of the graph model of Cholesky factorization [19]. In this section, we provide a brief description of elimination graphs and quotient graphs, and introduce an example, together with some notation used to describe our greedy heuristics. We also describe minimum degree and minimum deficiency schemes using quotient graphs.

We use terminology common in sparse matrix factorization throughout. The reader is referred to the book by George and Liu [14] for details.

Elimination graphs and quotient graphs. Sparse Cholesky factorization can be modeled using elimination graphs [19]. Let G denote an elimination graph. At the beginning, G is initialized to G_0, the graph of a sparse symmetric positive definite matrix [14]. At each step, a vertex and its incident edges are removed from G. If x is the vertex removed, edges are added to G so that the neighbors of x become a clique. Thus cliques are formed as the elimination proceeds.

A quotient graph [11, 14] is a compact representation of an elimination graph. It requires no more space than that for G_0 [14]. Unlike the elimination graph, vertices are not explicitly removed and neither are cliques formed explicitly. Instead vertices are grouped in "supervertices" and marked as "eliminated" or "uneliminated."

Let G denote the current elimination graph. Let S be the set of vertices that have been eliminated. Consider the subgraph induced by S in G_0. This subgraph will contain one or more connected components (which are also called domains). In the quotient graph, the vertices in each connected component are coalesced into an *eliminated supervertex*. Note that the cliques created by the elimination process in G are easy to identify in a quotient graph. Each such clique contains all (uneliminated) neighbors of an eliminated supervertex.

It is well known that as the elimination proceeds, some (uneliminated) vertices will become *indistinguishable* from each other; that is, they share essentially the same adjacency structure in the current elimination graph G. Now each set of indistinguishable vertices is coalesced into an *uneliminated supervertex* in the quotient graph.[2] Observe that all vertices in an uneliminated supervertex have the same degree (or deficiency) and hence can be "mass-eliminated" when the degree (or deficiency) becomes minimum [14, 18]. Furthermore, vertices in an uneliminated supervertex form a clique.

Thus a quotient graph can be viewed as a graph containing two kinds of supervertices: uneliminated supervertices and eliminated supervertices. Each uneliminated supervertex is a clique of indistinguishable vertices of the corresponding elimination graph G. Each eliminated supervertex is a subset of the vertices that have been eliminated from the original graph G_0. Vertices in the set of uneliminated supervertices adjacent to the same eliminated supervertex in the quotient graph form a clique in the elimination graph. For simplicity, we will say that these uneliminated supervertices form a "clique" in the quotient graph. Two uneliminated supervertices are said to be "neighbors" in the quotient graph when there is an edge between them or they are adjacent to the same eliminated supervertex in the quotient graph; the vertices belonging to one uneliminated supervertex are adjacent to those of the other supervertex in the corresponding elimination graph. We follow the convention in [14] in our implementation. When an eliminated supervertex X is formed, edges between its uneliminated neighbors are subsumed.

An example and some notation. Figure 1 contains an example of a quotient graph. Eliminated supervertices are represented by rectangles and uneliminated supervertices are represented by circles. Assume that an uneliminated supervertex has been selected according to the greedy criterion, and the quotient graph has been transformed. This gives a new eliminated supervertex in the quotient graph. Denote the new eliminated supervertex by X; in the remaining part of this paper X will be

[2] Most implementations only coalesce sets of indistinguishable vertices that can be identified inexpensively; identifying all maximal sets of indistinguishable vertices is too expensive.

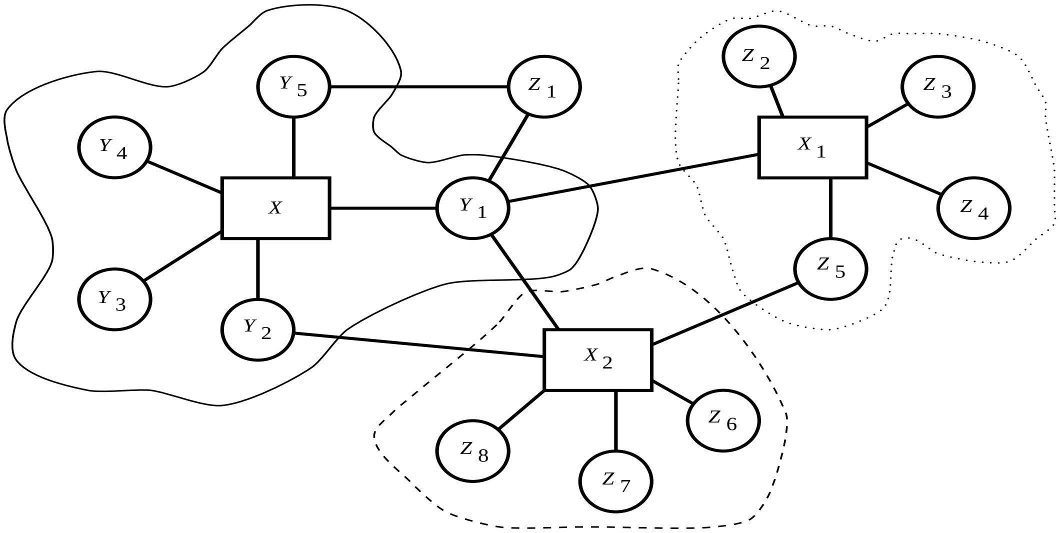


Fig. 1. *An example of a quotient graph; X is the most recently eliminated supervertex. Supervertices $\{Y_1, Y_2, Y_3, Y_4, Y_5\}$ enclosed in a curve form a clique; other partial cliques used in MMDF are shown using dotted curves.*

referred to as the "most recently eliminated supervertex." Let Y_1, Y_2, Y_3, Y_4, and Y_5 (enclosed by a solid curve) denote the uneliminated supervertices that are adjacent to X. Note that Y_1, Y_2, Y_3, Y_4, and Y_5 form a clique in the quotient graph. Using our convention, Z_1, Z_2, …, Z_8 are neighbors of Y_1. The eliminated supervertices X_1 and X_2 identify two other cliques: $\{Y_1, Z_2, Z_3, Z_4, Z_5\}$ and $\{Y_1, Y_2, Z_5, Z_6, Z_7, Z_8\}$. Observe that these three cliques are not disjoint. Uneliminated supervertices that are enclosed by a dotted curve (such as Z_2, Z_3, Z_4, and Z_5) form what we call a "partial" clique; these "partial" cliques will be used to describe our heuristics in the next section.

If V is a supervertex (either uneliminated or eliminated) in the quotient graph, we define $\mathcal{N}_1(V)$ as the set of uneliminated supervertices that are neighbors of V. We use $\mathcal{N}_2(V)$ to denote the set of uneliminated supervertices that are neighbors of those in $\mathcal{N}_1(V)$. We use $\deg(V)$ to denote the degree of an uneliminated supervertex V; $\deg(V)$ is the sum of $|V| - 1$ and the total number of vertices in all supervertices in $\mathcal{N}_1(V)$.

Minimum degree and deficiency schemes. Recall that a greedy heuristic needs a *metric* $d(v)$ for selecting the next supervertex to eliminate. Examples of $d()$ are the degree (in minimum degree) and the deficiency (in minimum deficiency). In terms of elimination graphs, a greedy heuristic has the following structure: select a vertex that minimizes $d()$, eliminate it from the current elimination graph, form the next elimination graph, and update the value of the metric for each vertex affected by the elimination. A greedy scheme can also be described in terms of quotient graphs: select an uneliminated supervertex that minimizes $d()$, create a new quotient graph, and update the value of the metric for each uneliminated supervertex affected by the elimination.

In the minimum deficiency heuristic, updating the deficiency after one step of elimination may be significantly more time consuming than updating the degree in the minimum degree algorithm. Consider the example in Figure 1 where X is the most recently eliminated supervertex. With minimum degree (MMD and AMD) only the uneliminated supervertices in $\mathcal{N}_1(X)$ need a degree update. However, with minimum deficiency, we need to update the deficiency of not only supervertices in $\mathcal{N}_1(X)$, but also some of the supervertices belonging to $\mathcal{N}_2(X)$. Any supervertex in $\mathcal{N}_2(X)$ that is a neighbor of two or more supervertices in $\mathcal{N}_1(X)$ may need a deficiency update. With respect to Figure 1, the deficiency of Z_1 (a neighbor of Y_1 and Y_5) will change if Y_1 and Y_5 are not adjacent prior to the elimination of X. Similarly, we may have to update the deficiency of each of Z_5, Z_6, Z_7, and Z_8 (each supervertex is a neighbor of both Y_1 and Y_2).

Rothberg showed that the true minimum deficiency algorithm (true local fill in [21]) produces significantly better orderings than MMD; a true minimum deficiency ordering, on the average, results in 28% fewer operations in computing the Cholesky factor than an MMD ordering. We obtained similar results in [18]. Our implementation of the minimum deficiency algorithm was on the average slower than MMD by two orders of magnitude [18]. Let X be the most recently eliminated supervertex. Using the deficiency as the metric but restricting updates, as in MMD, to uneliminated supervertices in $\mathcal{N}_1(X)$ leads to orderings that are inferior to true minimum deficiency but still significantly better than MMD. This was observed by Rothberg [21] and later corroborated by Ng and Raghavan [18]. However, even such a restricted scheme is more than 40 times slower than MMD [18]. In the next section we describe two relatively inexpensive but effective heuristics based on modifications to the deficiency and degree.

3. Modified minimum deficiency and minimum degree heuristics.

We now describe two heuristics based on approximations to the deficiency and the degree. Both metrics can be implemented using either the update mechanism in MMD or the faster scheme in AMD.

Our first heuristic MMDF is based on a deficiency-like metric. Consider the example in Figure 1, where X is the most recently eliminated supervertex. We update the values of the metric $d()$ of uneliminated vertices in $\mathcal{N}_1(X) = \{Y_1, Y_2, Y_3, Y_4, Y_5\}$ just as in MMD. Consider updating $d(Y_1)$ in Figure 1. An upper bound δ on the deficiency of Y_1 can be obtained in terms of the degree of Y_1. The true deficiency of Y_1 is obtained by subtracting from the upper bound the number of edges that are present before Y_1 is eliminated. Identifying *all* such edges requires examining the uneliminated supervertices in $\mathcal{N}_1(Y_1)$ and $\mathcal{N}_2(Y_1)$. However, *some* of these edges can be identified easily because in the quotient graph representation, uneliminated supervertices connected to a common eliminated supervertex form a clique. Using notation introduced earlier, $\mathcal{N}_1(Y_1) = \{Y_2, Y_3, Y_4, Y_5\} \cup \{Z_1, Z_2, \ldots, Z_8\}$ is the set of uneliminated neighbors of Y_1. The elements of $\mathcal{N}_1(Y_1)$ can be grouped into a set of disjoint "partial" cliques K.[3] The obvious member of K is $\{Y_2, Y_3, Y_4, Y_5\}$, consisting of the uneliminated neighbors of the eliminated supervertex X. The other partial cliques depend on the order in which the neighbors of Y_1 are examined. Without loss of generality, assume $\{Z_1\}$ forms the second clique. Likewise let $\{Z_2, Z_3, Z_4, Z_5\}$ be the next partial clique we examine. Finally, $\{Z_6, Z_7, Z_8\}$ is the fourth disjoint partial clique; we do not include Z_5 because it was visited and included in the third partial clique. The metric $d(Y_1)$ is defined as $\delta - C - ct$; δ is an upper bound on deficiency, C is the sum of contributions from partial cliques, and ct is the correction term. At initialization, $d(Y_1)$ is set to the upper bound δ. We define δ, C, and ct below.

[3] The exact partitioning of the elements of $\mathcal{N}_1(Y_1)$ into disjoint partial cliques depends on the data structure in the implementation of quotient graphs, and on the order in which the elements are visited and marked.

- δ : The upper bound δ is based on the external degree [17]: $\delta = \text{edeg}(Y_1)[\text{edeg}(Y_1) - 1]$, where $\text{edeg}(Y_1) = \deg(Y_1) - |Y_1|$. This upper bound favors larger supervertices. If two supervertices have the same degree, the larger supervertex will have a smaller upper bound on deficiency since its external degree is smaller.
- C : Let K be the set of disjoint partial cliques as described above. We define $C = \sum_{W \in K} |W| * [|W| - 1]$, where W is a partial clique in K; the size of W is the total number of vertices in all uneliminated supervertices that constitute W.
- ct : The correction term ct takes into account contributions missed because (1) partial cliques in K are forced to be disjoint, and (2) cliques such as $\{Z_1, Y_5\}$ are not detected because we do not examine $\mathcal{N}_2(Y_1)$. Our heuristic value of ct is $2\,\text{edeg}(Y_1) * |Y_1|$. The rationale for the choice of ct is as follows. Assume that each supervertex in $\mathcal{N}_1(Y_1)$ is connected to one other supervertex (in $\mathcal{N}_1(Y_1)$) and that the associated contribution has been missed. Assume further that the size of Y_1 is representative of the sizes of supervertices in $\mathcal{N}_1(Y_1)$; then the contributions that have been missed equal $\sum_{V \in \mathcal{N}_1(Y_1), V \neq Y_1} 2\,|V||Y_1|$, which simplifies to $ct = 2\,\text{edeg}(Y_1) * |Y_1|$.

The greedy ordering process repeatedly involves operations such as (1) finding and deleting vertices with a minimal value of the metric, and (2) altering values of the metric for arbitrary vertices. In short, an implementation of a priority-queue [1] data structure is required. In the minimum degree algorithm, the $d()$ is the degree of a vertex and is bounded above by n, the order of the matrix. Now the priority-queue is implemented as a set of n bins with vertices in each bin stored as a linked list. In a deficiency-based algorithm, $d()$ can be as large as n^2. We therefore use a heap [1] to maintain the values of the metric.

We would like to emphasize that MMDF is a heuristic. We see the correction term as an approximation to edges missed because we restrict our attention to partial cliques that are disjoint. In our experiments we found that small multiples of the correction term behaved just as well if not better. However, the heuristic performs poorly if the correction term is absent.

Our second heuristic MMMD attempts to use a metric that is bound by a small multiple of the degree. As indicated by the name it is a close variant of the minimum degree algorithm with multiple eliminations. The goal is to come up with an ordering algorithm that has a time complexity similar to that of MMD. Consider Figure 1, and once again assume X is the most recently eliminated supervertex. For the supervertex Y_1, MMMD uses the metric $d(Y_1) = 2\text{edeg}(Y_1) - U$, where U is the size of the largest partial clique in the set K (described above). More precisely, $U = \max_{W \in K} |W|$. At initialization we simply use $2\,\text{edeg}(Y_1)$. Note that MMMD differs from MMD only in the definition of the metric. The metric in MMMD tries to take into account contributions from the largest clique that contains Y_1.

Our implementations of MMDF and MMMD are based on modifications of Liu's MMD code. In particular, the supervertices in the set K of Y_1 are visited in the same order as that in Liu's MMD code. Hence, we expect the update cost of MMDF and MMMD for each supervertex to be similar to that of Liu's MMD.

Approximate MMDF and MMMD. We now briefly outline how "approximate" versions of the two schemes can be implemented using the faster update mechanism in the AMD scheme of Amestoy, Davis, and Duff [2]. Our description of AMD is admittedly brief and is in terms of our notation. The reader is referred to [2] for details.

To describe approximate-MMDF, we once again consider Figure 1 and the metric for Y_1, an uneliminated supervertex adjacent to the most recently eliminated supervertex X. The upper bound is now calculated using the approximate external degree of AMD. The correction term can also be easily calculated in terms of this approximate external degree and the size of supervertex Y_1. The main difference is in how K is constructed, and hence the term C. Now the set K corresponds to the cliques used in AMD to compute an approximation to the degree. AMD uses the sizes of certain cliques of supervertices in the set $\mathcal{N}_1(Y_1) = \{Y_2, Y_3, Y_4, Y_5\} \cup \{Z_1, Z_2, \dots, Z_6\}$. With respect to the example in Figure 1, the cliques used in AMD are: $\{Y_2, Y_3, Y_4, Y_5\}$, $\{Z_1\}$, $\{Z_2, Z_3, Z_4, Z_5\}$, $\{Z_5, Z_6, Z_7, Z_k\}$. The first clique is the one formed by elimination of X; the remaining cliques have no overlap with this clique. However, the remaining cliques may have supervertices in common. MMDF based on MMD forces the partial cliques to be disjoint. On the other hand, approximate-MMDF relaxes this restriction; i.e., it uses the cliques in AMD, and these cliques may have common uneliminated supervertices. The approximation to C (the contribution to the deficiency from the partial cliques) is computed using the clique sizes used in AMD (for the approximation to the degree). Approximate-MMMD is similar; it also uses the cliques in AMD.

It should be noted that in AMD, when a supervertex is shared by exactly two cliques, the approximation to the degree is exact. In our approximate-MMDF, because of the correction term, it is unlikely that our approximation to $d(Y_1)$ is exactly the same as that in MMDF even when Y_1 is shared by only two cliques. This comment also applies to our approximate-MMMD.

Relation to other deficiency-like schemes. We would like to note that MMDF is similar to AMF3, proposed by Rothberg [21]; it differs mainly in the way in which the partial cliques are constructed, as well as in the definition of the correction term. AMF, AMMF, and AMIND are three other heuristics developed by Rothberg and Eisenstat [21, 22]. The deficiency-like metrics in AMF, AMMF, and AMIND use only edges in the most recently formed clique while the metric in MMDF takes into account edges in as many cliques as we can "easily identify." AMIND also uses a term which is similar to our correction term in MMDF. MMMD is similar to AMF in that it uses only the size of a single clique but it differs in the sense that it uses a degree-like metric.

4. Performance of MMDF and MMMD.

We now report on the performance of MMDF and MMMD. We use a set of 36 test matrices in our empirical study. Our test suite is a subset of the one used by Rothberg and Eisenstat [21, 22]; their test suite contains four other matrices that are proprietary and hence are not available to us. Our MMD code is Liu's Fortran implementation converted to C. Our MMDF and MMMD heuristics are built using the MMD code. MMDF differs from MMD in the metric update as well in the use of heaps to store and retrieve the metric. Furthermore, unlike MMD, MMDF does not allow "multiple eliminations." MMMD is nearly identical to MMD and differs only in the metric update portion. All our experiments were performed on a Sun Ultra Sparc-2 workstation.

The quality of greedy orderings can vary depending on the initial numbering. For each test matrix, we use 11 different random initial numberings for MMD, MMDF, and MMMD. We consider two measures of the quality of ordering: the number of nonzeros in the Cholesky factor, and the number of floating-point operations required to compute the Cholesky factor. We also report execution times for MMD, MMDF, and MMMD. The characteristics of the test matrices and the quality of MMD orderings

Table 1
Performance of MMD; $|L|$ and operations are mean and median values over 11 *initial orderings.*

Problem	rank	$\lvert A\rvert$	Mean			Median		
			Time	$\lvert L\rvert$	Operations	Time	$\lvert L\rvert$	Operations
		(10^3)	(secs)	(10^3)	(10^6)	(secs)	(10^3)	(10^6)
3DTUBE	45330	3256.9	10.07	31739	42157	10.08	31842	42101
bcsstk15	3948	117.8	0.61	651	167	0.61	653	168
bcsstk16	4884	290.3	0.42	734	143	0.42	737	143
bcsstk17	10974	428.6	0.96	1130	197	0.97	1135	197
bcsstk18	11948	140.1	1.02	645	132	1.01	644	131
bcsstk23	3134	45.2	0.62	461	144	0.62	455	139
bcsstk25	15439	252.2	1.90	1530	341	1.88	1537	342
bcsstk29	13992	619.5	1.69	1767	443	1.70	1744	425
bcsstk30	28924	2043.5	2.91	3844	930	2.89	3856	935
bcsstk31	35588	1181.4	4.78	5311	2547	4.78	5263	2485
bcsstk32	44609	2014.7	5.32	5200	1078	5.25	5228	1096
bcsstk33	8738	591.9	1.30	2644	1301	1.32	2640	1302
bcsstk35	30237	1480.4	2.80	2737	389	2.80	2734	389
bcsstk36	23052	1166.2	1.57	2757	610	1.56	2761	609
bcsstk37	25503	1166.5	1.82	2838	558	1.77	2834	560
bcsstk38	8032	363.5	1.28	750	123	1.26	747	122
bcsstk39	46772	2089.3	8.18	7651	2187	8.12	7645	2191
bikker2	173160	854.9	43.27	101268	197828	43.27	100870	195802
cfd1	70656	1899.0	19.44	39724	46751	19.38	39959	46935
cfd2	123440	3211.3	36.93	89902	178101	36.84	90610	178166
copter2	55476	759.9	10.81	14247	12852	10.79	14183	12620
crystk01	4875	320.7	0.46	1081	336	0.46	1083	338
crystk02	13965	982.5	1.85	6107	4342	1.84	6124	4340
crystk03	24696	1775.8	3.91	13950	13167	3.92	13944	13051
flap	51537	1010.7	5.75	5613	1882	5.77	5630	1905
ford2	100196	544.7	8.64	2438	303	8.57	2448	309
gearbox	153746	9234.1	40.50	53012	57817	40.52	52973	57328
msc10848	10848	1240.6	1.08	2032	572	1.07	2028	576
msc23052	23052	1177.9	1.58	2757	612	1.57	2748	608
pwt	36519	326.1	2.55	1764	223	2.51	1768	225
sphere6	16386	114.7	0.57	794	133	0.57	796	133
struct1	46949	2329.5	4.15	5075	1302	4.13	5068	1299
struct2	73752	3670.9	7.74	9853	3867	7.74	9810	3817
struct3	53570	1227.3	6.35	5333	1231	6.36	5309	1216
struct4	4350	242.1	2.10	2299	1857	2.07	2248	1756
troll	213453	1198.5	42.66	61482	156149	42.37	61171	153228

are reported in Table 1. We report mean and median values over 11 initial random numberings for MMD in Table 1.

Table 2 shows the performance of MMDF and MMMD relative to that of MMD. The relative measure is computed as the ratio of the medians over 11 initial random numberings. We also present the geometric mean and the median over all test matrices in the last two lines of the table. The execution time of MMMD is slightly more (7–13%) than that of MMD, while MMDF requires on average an overhead of 33–34% over MMD. Our experiments indicate that avoiding the use of heaps in MMDF (as suggested by a referee) will reduce about a third of this overhead.[4]

[4] One of the referees has suggested using a data structure similar to that in MMD, i.e., use n bins for $d()$ values. The last bin is used to store those $d()$'s that are larger than $n - 1$. On the occasions when the minimum value of $d()$'s is greater than or equal to n, a linear search is used.

Table 2

Performance of MMDF and MMMD relative to MMD. For each problem we report the ratio of median values over 11 *initial random orderings.*

Problem	Ordering time		$\|L\|$		Operations	
	MMDF	MMMD	MMDF	MMMD	MMDF	MMMD
3DTUBE	1.50	0.99	0.87	0.86	0.79	0.77
bcsstk15	0.77	0.93	0.89	0.92	0.76	0.83
bcsstk16	1.64	1.50	0.93	0.93	0.84	0.85
bcsstk17	1.21	1.11	1.00	0.98	0.97	0.95
bcsstk18	1.04	1.06	0.90	0.93	0.76	0.82
bcsstk23	0.79	1.05	0.87	0.93	0.77	0.88
bcsstk25	1.06	1.10	0.87	0.92	0.72	0.83
bcsstk29	1.08	1.05	0.94	0.95	0.82	0.83
bcsstk30	1.41	1.17	0.94	0.97	0.83	0.93
bcsstk31	1.36	1.18	0.92	0.95	0.81	0.88
bcsstk32	1.64	1.36	0.96	0.97	0.85	0.89
bcsstk33	1.10	1.17	0.90	0.89	0.77	0.76
bcsstk35	1.64	1.34	1.02	1.00	0.98	0.98
bcsstk36	1.68	1.47	1.01	1.00	0.98	0.96
bcsstk37	1.72	1.50	0.98	0.98	0.89	0.91
bcsstk38	1.02	1.21	0.99	1.00	0.94	1.00
bcsstk39	1.59	1.09	1.08	1.02	1.32	1.08
bikker2	2.02	1.24	0.86	0.86	0.69	0.71
cfd1	1.26	1.01	0.79	0.82	0.62	0.69
cfd2	1.25	0.99	0.80	0.80	0.65	0.69
copter2	1.19	1.05	0.80	0.86	0.66	0.75
crystk01	1.33	1.22	0.91	0.89	0.80	0.77
crystk02	1.23	1.04	0.86	0.83	0.72	0.68
crystk03	1.25	1.04	0.89	0.80	0.79	0.64
flap	1.50	1.06	0.93	0.91	0.81	0.78
ford2	1.30	1.06	0.92	0.95	0.73	0.81
gearbox	1.58	1.13	0.91	1.00	0.77	1.17
msc10848	1.56	1.33	0.95	0.96	0.85	0.88
msc23052	1.69	1.52	1.00	0.99	0.91	0.94
pwt	1.48	1.03	0.94	0.97	0.85	0.92
sphere6	1.74	1.04	0.89	0.96	0.76	0.93
struct1	1.72	1.08	0.95	0.96	0.85	0.90
struct2	1.80	1.06	0.97	0.98	0.94	0.93
struct3	1.28	1.05	0.95	0.96	0.87	0.90
struct4	0.73	0.99	0.80	0.84	0.65	0.70
troll	1.13	1.04	0.80	0.85	0.65	0.73
g-mean	1.33	1.13	0.91	0.92	0.80	0.84
median	1.34	1.07	0.92	0.95	0.80	0.86

Results for variants of MMDF and MMMD are summarized in Table 3. Approximate-MMDF (approximate-MMMD) performs just as well as MMDF (MMD); the geometric mean and the median remain unchanged. For MMDF, MMMD, and their approximate counterparts, adding Ashcraft's initial compression step [3] improves the performance slightly (1% on the average). The approximate versions of MMDF and MMMD are based on our implementation of AMD, which are not as efficient as the implementation in [2]. For example, our implementations do not include features such as "aggressive absorption" and the use of extra storage to repack quotient graph segments for improved cache-access. Likewise, in our implementation of Ashcraft's compression step, our hashing mechanism is not very efficient. We therefore do not provide timing results for these experiments. We expect the overheads of approximate-

Table 3

Summary of performance of MMDF and MMMD variants relative to MMD. The geometric-mean and median over all problems in the test suite are based on the ratio of median values over 11 *initial random orderings for each problem.*

Method	$\lvert L\rvert$		Operations	
	g-mean	median	g-mean	median
Without initial compression				
MMDF	.91	.92	.80	.80
approximate-MMDF	.91	.92	.80	.80
MMMD	.92	.95	.84	.86
approximate-MMMD	.92	.95	.84	.86
With initial compression				
MMDF	.90	.90	.79	.79
approximate-MMDF	.90	.91	.79	.79
MMMD	.92	.95	.84	.85
approximate-MMMD	.92	.95	.84	.86

Table 4

Summary of operation counts to factor for MMDF, MMMD, AMF, AMMF, and AMIND relative to MMD (with initial compression).

Measure	Ng–Raghavan		Rothberg–Eisenstat		
	MMDF	MMMD	AMF	AMMF	AMIND
g-mean	.79	.84	.85	.75	.79
median	.79	.85	.85	.78	.80

MMDF (approximate-MMMD) relative to AMD (with or without initial compression) to be the same as that of MMDF (MMMD) relative to MMD.

5. Conclusions.

We have developed two new greedy heuristics: MMDF and MMMD. Both of these schemes produce orderings that are better than MMD orderings. When initial compression is not performed, the first scheme MMDF produces orderings that require approximately 20% fewer floating-point operations for factorization than MMD, while the second scheme MMMD generates orderings that incur 14% fewer operations for factorization than MMD. The reductions are slightly better when initial compression is performed. MMDF uses a deficiency-like metric, i.e., a metric whose value is a quadratic function of the degree. The execution time of MMDF is approximately 1.3 times that of MMD. On the other hand, MMMD uses a degree-like metric, which is bounded above by twice the value of the degree. The implementation of MMMD is the same as that of MMD, but for the difference in the choice of the metric. The slight increase in ordering time for MMMD is primarily due to the differences in the orderings computed by the two schemes.[5] Finally, there is no change in the quality of the orderings when MMDF (MMMD) is implemented using the AMD framework of Amestoy, Davis, and Duff [2].

For completeness, Table 4 summarizes the performance of our schemes, as well as those in [22]. It appears that the performance of MMMD and AMF1 [21] and AMF [22] are similar. Likewise, MMDF and AMIND [22] produce orderings of similar quality. AMMF [22] appears to be slightly better than MMDF. Relative to MMD, AMMF orderings require 25% (geometric mean) to 22% (median) fewer operations for factorization, while MMDF (with compression) orderings require 21% (geometric mean and median) fewer operations for factorization.

[5] The cost of maintaining the quotient graph depends on the adjacency structure and the elimination sequence.

Our work is an attempt to understand factors affecting the performance of greedy ordering heuristics. We tried several metrics that are close to those in MMDF and MMMD. Many of these had average operation counts for factorization similar to those reported for MMDF and MMMD, while others varied substantially. We also experimented with a variant of MMDF that did update the metric for "neighbors of neighbors" as in true minimum deficiency. Surprisingly, the operation counts were on the average higher by 3–4% for this variant. The performance of true minimum deficiency shows that deficiency is a better metric than the degree. However, we surmise that the improved performance of our heuristics is from the complicated interplay of the metric and the greedy process and not necessarily from accurately modeling the true deficiency. We conjecture that there could well be other relatively inexpensive greedy strategies that significantly outperform the ones known so far.

Acknowledgments. The authors would like to thank the referees and the editor for many thoughtful suggestions, which substantially improved the presentation of the material.

REFERENCES

[1] A. Aho, J. Hopcroft, and J. Ullman, *The Design and Analysis of Computer Algorithms*, Addison–Wesley, Reading, MA, 1974.

[2] P. Amestoy, T. A. Davis, and I. S. Duff, *An approximate minimum degree ordering algorithm*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 886–905.

[3] C. Ashcraft, *Compressed graphs and the minimum degree algorithm*, SIAM J. Sci. Comput., 16 (1995), pp. 1404–1411.

[4] C. Ashcraft and J. W.-H. Liu, *Robust Orderings of Sparse Matrices Using Multisection*, Technical report ISSTECH-96-002, Boeing Computer Services, Seattle, WA, 1996.

[5] P. Berman and G. Schnitger, *On the performance of the minimum degree ordering for Gaussian elimination*, SIAM J. Matrix Anal. Appl., 11 (1990), pp. 83–88.

[6] E. Cuthill, *Several strategies for reducing bandwidth of matrices*, in Sparse Matrices and Their Applications, D. J. Rose and R. A. Willoughby, eds., Plenum Press, New York, 1972, pp. 157–166.

[7] E. Cuthill and J. McKee, *Reducing the bandwidth of sparse symmetric matrices*, in Proceedings 24th ACM National Conference, August 1969, ACM, New York, pp. 157–172.

[8] I. Duff, A. Erisman, and J. Reid, *Direct Methods for Sparse Matrices*, Oxford University Press, Oxford, England, 1987.

[9] A. George, *Computer Implementation of the Finite Element Method*, Ph.D. thesis, Dept. of Computer Science, Stanford University, Stanford, CA, 1971.

[10] A. George and J. W.-H. Liu, *An automatic nested dissection algorithm for irregular finite element problems*, SIAM J. Numer. Anal., 15 (1978), pp. 1053–1069.

[11] A. George and J. W.-H. Liu, *A quotient graph model for symmetric factorization*, in Sparse Matrix Proceedings 1978, I. S. Duff and G. W. Stewart, eds., SIAM, Philadelphia, PA, 1979, pp. 154–175.

[12] A. George and J. W.-H. Liu, *A fast implementation of the minimum degree algorithm using quotient graphs*, ACM Trans. Math. Software, 6 (1980), pp. 337–358.

[13] A. George and J. W.-H. Liu, *A minimal storage implementation of the minimum degree algorithm*, SIAM J. Numer. Anal., 17 (1980), pp. 282–299.

[14] A. George and J. W.-H. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice Hall, Englewood Cliffs, NJ, 1981.

[15] A. Gupta, *Fast and Effective Algorithms for Graph Partitioning and Sparse Matrix Ordering*, Technical report RC-20496, IBM Research Division, T.J. Watson Research Center, Yorktown Heights, NY, 1996.

[16] B. Hendrickson and E. Rothberg, *Improving the run time and quality of nested dissection ordering*, SIAM J. Sci. Comput., 20 (1999), pp. 468–489.

[17] J. W.-H. Liu, *Modification of the minimum degree algorithm by multiple elimination*, ACM Trans. Math. Software, 11 (1985), pp. 141–153.

[18] E. G.-Y. Ng and P. Raghavan, *Performance of Greedy Ordering Heuristics for Sparse Cholesky Factorization*, manuscript, 1997.

[19] S. Parter, *The use of linear graphs in Gaussian elimination*, SIAM Rev., 3 (1961), pp. 119–130.

[20] D. Rose, *A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations*, in Graph Theory and Computing, R. C. Read, ed., Academic Press, New York, 1972, pp. 183–217.

[21] E. Rothberg, *Ordering Sparse Matrices Using Approximate Minimum Local Fill*, manuscript, 1996.

[22] E. Rothberg and S. Eisenstat, *Node selection strategies for bottom-up sparse matrix ordering*, SIAM J. Matrix Anal. Appl., 19 (1998), pp. 682–695.

[23] W. Tinney and J. Walker, *Direct solution of sparse network equations by optimally ordered triangular factorization*, Proc. IEEE, 55 (1967), pp. 1801–1809.

[24] M. Yannakakis, *Computing the minimum fill-in is NP-complete*, SIAM J. Alg. Discrete Methods, 2 (1981), pp. 77–79.

SIAM J. Matrix Anal. Appl.
Vol. 20, No. 4, pp. 915–952

AN ASYNCHRONOUS PARALLEL SUPERNODAL ALGORITHM FOR SPARSE GAUSSIAN ELIMINATION[*]

JAMES W. DEMMEL[†], JOHN R. GILBERT[‡], AND XIAOYE S. LI[§]

Abstract. Although Gaussian elimination with partial pivoting is a robust algorithm to solve unsymmetric sparse linear systems of equations, it is difficult to implement efficiently on parallel machines because of its dynamic and somewhat unpredictable way of generating work and intermediate results at run time. In this paper, we present an efficient parallel algorithm that overcomes this difficulty. The high performance of our algorithm is achieved through (1) using a graph reduction technique and a supernode-panel computational kernel for high single processor utilization, and (2) scheduling two types of parallel tasks for a high level of concurrency. One such task is factoring the independent panels in the disjoint subtrees of the column elimination tree of A. Another task is updating a panel by previously computed supernodes. A scheduler assigns tasks to free processors dynamically and facilitates the smooth transition between the two types of parallel tasks. No global synchronization is used in the algorithm. The algorithm is well suited for shared memory machines (SMP) with a modest number of processors. We demonstrate 4- to 7-fold speedups on a range of 8 processor SMPs, and more on larger SMPs. One realistic problem arising from a 3-D flow calculation achieves factorization rates of 1.0, 2.5, 0.8, and 0.8 gigaflops on the 12 processor Power Challenge, 8 processor Cray C90, 16 processor Cray J90, and 8 processor AlphaServer 8400.

Key words. sparse Gaussian elimination, unsymmetric linear systems, supernodes, parallelism, dynamic scheduling and load balancing

AMS subject classifications. 65F50, 65F05

PII. S0895479897317685

1. Introduction.

In earlier work with Eisenstat and Liu, we described a publically released sequential software library, SuperLU, to solve unsymmetric sparse linear systems using Gaussian elimination with partial pivoting [6]. This left-looking, blocked algorithm includes symmetric structural reduction for fast symbolic factorization, and supernode-panel updates to achieve better data reuse in cache and floating-point registers. Here, we assume the reader has prior knowledge of SuperLU and the material upon which SuperLU is based [6, 25].

[*] Received by the editors February 28, 1997; accepted for publication (in revised form) by E. Ng January 30, 1998; published electronically July 9, 1999. The research of J. Demmel and X. Li was supported in part by NSF grant ASC-9313958, DOE grant DE-FG03-94ER25219, UT subcontract ORA4466 from ARPA contract DAAL03–91–C0047, DOE grant DE-FG03-94ER25206, and NSF Infrastructure grants CDA-8722788 and CDA-9401156. This manuscript has been authored by a contractor of the U.S. Government under contract DE-AC03-76SF00098. The U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes. Copyright is owned by SIAM to the extent not limited by these rights.

http://www.siam.org/journals/simax/20-4/31768.html

[†] Computer Science Division, University of California, Berkeley, CA 94720 (demmel@cs.berkeley.edu).

[‡] Xerox Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto, CA 94304 (gilbert@parc.xerox.com). The research of this author was supported in part by the Institute for Mathematics and Its Applications at the University of Minnesota and in part by DARPA contract DABT63-95-C0087.

[§] National Energy Research Scientific Computing (NERSC) Center, Lawrence Berkeley National Lab, MS 50F, 1 Cyclotron Rd., Berkeley, CA 94720 (xiaoye@nersc.gov). The work of this author was supported in part by the Office of Computational and Technology Research, Division of Mathematical, Information, and Computational Sciences of the U.S. Department of Energy under contract 76SF00098.

In this paper we study an efficient parallel algorithm based on SuperLU. The primary objective of this work is to achieve good efficiency on shared memory systems with a modest number of processors (for example, between 10 and 20). In addition to measuring the efficiency of our parallel algorithm on these machines, we also study a theoretical upper bound on performance of this algorithm. The efficiency of the algorithm has been demonstrated on several shared memory parallel machines. When compared to the best sequential runtime of SuperLU, the parallel algorithm typically achieved 4- to 7-fold speedups on 8 processor platforms for large sparse matrices.

The rest of the paper is organized as follows. In section 2 we review the sequential SuperLU algorithm. Section 3 presents the sources and the characteristics of the test matrices. Section 4 presents the parallel machines used in our study. In section 5 we describe several design choices we have made in parallelization, including how to find parallelism, how to define individual tasks, and memory management for supernodes. Section 6 sketches the high-level parallel scheduling algorithm. In section 7, we present the parallel performance achieved with the test matrices on a number of platforms. Both time and space efficiency will be illustrated. We also quantify the sources of the overhead in parallelization and give a thorough analysis of their impact on performance. In the end of this section we establish a PRAM (parallel random-access machine) model to predict an upper bound on speedups attainable by the proposed algorithm. Finally, section 8 draws conclusions and suggests future research.

2. Overview of sequential algorithm in SuperLU. Figure 2.1 sketches the supernode-panel factorization algorithm used in SuperLU. A *supernode* is defined to be a range $(r\colon s)$ of columns of L with the triangular block just below the diagonal being full, and with the same row structure below this block. We store a supernode as a rectangular block, including the triangle of U in rows and columns r through s (see Figure 2.2). This allows us to address each supernode as a 2-D array in calls to BLAS routines [7, 8] and to get high performance. To increase the average size of supernodes (and hence performance), we merge groups of consecutive columns (usually no more than four columns) at the fringe of the column elimination tree (section 5.1) into *relaxed* supernodes regardless of their row structures. A *panel* is a block of w consecutive columns in the matrix that are updated simultaneously by a supernode using calls to the BLAS. The row structures of the columns in a panel may not be correlated in any fashion, and the boundaries between panels may be different from those between supernodes. Each panel factorization (outer loop in Figure 2.1) consists of three distinct steps: (1) the symbolic factorization to determine the nonzero structure, (2) the numerical updates by supernodes, and (3) the factorization of each column in the panel. The pivot selection, detection of the supernode boundary, and symmetric structure reduction (to reduce the cost of later symbolic factorization steps) are all performed in the inner factorization step. Both panel and column symbolic steps use depth-first search (DFS). A further refinement, a 2-D supernode partitioning (defined by the blocking parameters t and b in Figure 2.2), enhances performance for large matrices and machines with small caches. A more detailed description of SuperLU is in the paper [6].

We conducted extensive performance evaluation for SuperLU on several recent superscalar architectures. For large sparse matrices, SuperLU achieves up to 40% of the peak floating-point performance on both IBM RS/6000-590 and MIPS R8000. It achieves nearly 25% peak on the DEC Alpha 21164. Li [25] presented and analyzed the performance results in more detail.

```
**for** column $j = 1$ **to** $n$ **step** $w$ **do**
  $F(:, j : j + w - 1) = A(:, j : j + w - 1)$;
  (1) *Predict the nonzero structure of panel* $F(:, j : j + w - 1)$*:*
      Determine which supernodes will update any of $F(:, j: j + w - 1)$;
  (2) *Update panel* $F(:, j : j + w - 1)$ *using previous supernodes:*
      **for** each updating supernode $(r: s) < j$ in topological order **do**
        • Triangular solve:
          $U(r : s, j : j + w - 1) = L(r : s, r : s) \backslash F(r : s, j : j + w - 1)$;
        • Matrix update:
          $F(s + 1 : n, j : j + w - 1) = F(s + 1 : n, j : j + w - 1)$
                                $- L(s + 1 : n, r : s) \cdot U(r : s, j : j + w - 1)$;
      **end for** $(r : s)$;
  (3) *Inner factorization for each column in the panel:*
      **for** column $jj = j$ **to** $j + w - 1$ **do**
        • Supernode-column update for column $F(j : n, jj)$;
        • Row pivoting for column $F(jj : n, jj)$;
        • Determine whether $jj$ belongs to the same supernode as $jj - 1$;
        • Symmetric structure pruning;
      **end for** $jj$;
**end for** $j$;
```

FIG. 2.1. *The supernode-panel factorization algorithm.*

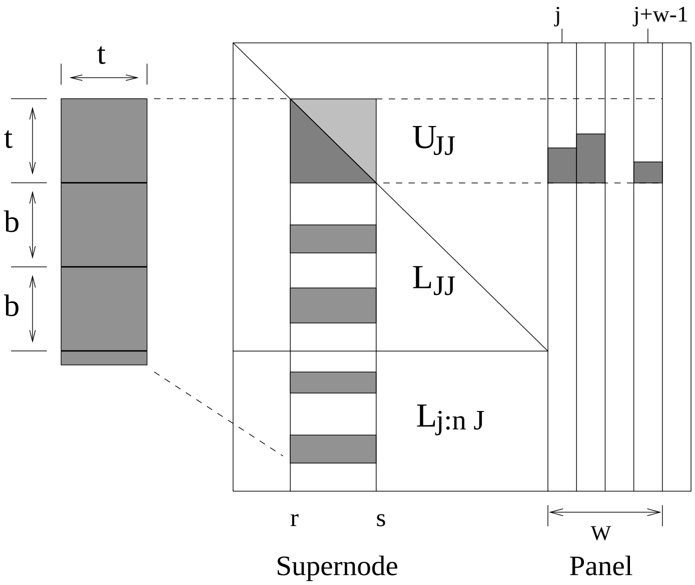


FIG. 2.2. *Illustration of a supernode-panel update.* $J = 1: j - 1$.

3. Test matrices. To evaluate our algorithms, we have collected matrices from various sources, with their characteristics summarized in Table 3.1.

Some of the matrices are from the Harwell-Boeing collection [9]. Many of the larger matrices are from the ftp site maintained by Tim Davis of the University of

Table 3.1

Characteristics of the test matrices. Structural symmetry s is defined to be the fraction of the nonzeros matched by nonzeros in symmetric locations. None of the matrices are numerically symmetric. $nnz(A)$ is the number of nonzeros in A. $F = L + U - I$ is the filled matrix, where I is an identity matrix.

	Matrix	s	n	$nnz(A)$	$\frac{nnz(A)}{n}$	$nnz(F)$	#flops/$nnz(F)$
1	Memplus	.983	17758	99147	5.6	140388	12.5
2	Gemat11	.002	4929	33185	6.7	93370	16.3
3	Rdist1	.062	4134	9408	2.3	338624	38.1
4	Orani678	.073	2529	90158	35.6	280788	53.3
5	Mcfe	.709	765	24382	31.8	69053	59.9
6	Lnsp3937	.869	3937	25407	6.5	427600	91.1
7	Lns_3937	.869	3937	25407	6.5	449346	99.7
8	Sherman5	.780	3312	20793	6.3	249199	101.3
9	Jpwh_991	.947	991	6027	6.1	140746	127.7
10	Sherman3	1.000	5005	20033	4.0	433376	139.8
11	Orsreg_1	1.000	2205	14133	6.4	402478	148.6
12	Saylr4	1.000	3564	22316	6.3	654908	160.0
13	Shyy161	.769	76480	329762	4.3	7634810	205.8
14	Goodwin	.642	7320	324772	44.4	3109585	213.9
15	Venkat01	1.000	62424	1717792	27.5	12987004	247.9
16	Inaccura	1.000	16146	1015156	62.9	9941478	414.3
17	Af23560	.947	23560	460598	19.6	13986992	454.9
18	Dense1000	1.000	1000	1000000	1000	1000000	666.2
19	Raefsky3	1.000	21200	1488768	70.2	17544134	690.7
20	Ex11	1.000	16614	1096948	66.0	26207974	1023.1
21	Wang3	1.000	26064	177168	6.8	13287108	1095.5
22	Raefsky4	1.000	19779	1316789	66.6	26678597	1172.6
23	Vavasis3	.001	41092	1683902	41.0	49192880	1813.5

Florida.[1] Those matrices are as follows. Memplus is a circuit simulation matrix from Steve Hamm of Motorola. Rdist1 is a reactive distillation problem in chemical process separation calculations, provided by Stephen Zitney of Cray Research, Inc. Shyy161 is derived from a direct, fully coupled method for solving the Navier–Stokes equations for viscous flow calculations, provided by Wei Shyy of the University of Florida. Goodwin is a finite element matrix in a nonlinear solver for a fluid mechanics problem, provided by Ralph Goodwin of the University of Illinois at Urbana-Champaign. Venkat01, Inaccura, and Raefsky3/4 were provided by Horst Simon, then of NASA and currently at NERSC. Venkat01 comes from an implicit 2-D Euler solver for an unstructured grid in a flow simulation. Raefsky3 is from a fluid structure interaction turbulence problem. Raefsky4 is from a buckling problem for a container model. Af23560 is from solving an unsymmetric eigenvalue problem, provided by Zhaojun Bai of the University of Kentucky. Ex11 is from a 3-D steady flow calculation in the SPARSKIT collection maintained by Yousef Saad at the University of Minnesota. Wang3 is from solving a coupled nonlinear PDE system in a 3-D (30 × 30 × 30 uniform mesh) semiconductor device simulation, as provided by Song Wang of the University of New South Wales, Sydney. Vavasis3 is an unsymmetric augmented matrix for a 2-D PDE with highly varying coefficients [32]. Dense1000 is a dense 1000 × 1000 random matrix.

This paper does not address the performance of column preordering for sparsity. We simply use the existing ordering algorithms provided by Matlab [18]. For all matrices except 1 (Memplus), 15 (Venkat01), and 21 (Wang3), the columns were

[1] URL: http://www.cis.ufl.edu/∼davis

Table 4.1
Characteristics of the parallel machines used in our study.

Machine	Processor	CPUs	Bus Bandwidth	Read Latency	Memory Size	Programming Model
Sun SPARCcenter 2000	SuperSPARC	4	500 MB/s	1200 ns	196 MB	Solaris thread
SGI Power Challenge	MIPS R8000	16	1.2 GB/s	252 ns	2 GB	Parallel C
DEC AlphaServer 8400	Alpha 21164	8	1.6 GB/s	260 ns	4 GB	pthread
Cray PVP	C90	8	245.8 GB/s	96 ns	640 MB	microtasking
Cray PVP	J90	16	51.2 GB/s	330 ns	640 MB	microtasking

permuted by Matlab's minimum degree ordering of A^TA, also known as "column minimum degree" ordering. However, this ordering produces a tremendous amount of fill for matrices 1, 15, and 21, because it only attempts to minimize the upper bound on the actual fill and the upper bounds are too loose in these cases. We found that when these three matrices were symmetrically permuted by Matlab's symmetric minimum degree ordering on $A + A^T$, the amount of fill is much smaller than using column minimum degree ordering. The last column in Table 3.1 shows the number of nonzeros in matrix F when using these column preorderings.

The matrices are sorted in increasing order of $flops/nnz(F)$, the ratio of the number of floating-point operations to the number of nonzeros $nnz(F)$. This "figure of merit" gives the maximum potential data reuse, as described by Demmel et al. [6]. Thus, we expect uniprocessor performance to increase with increasing $flops/nnz(F)$.

4. Shared memory multiprocessor systems used for testing. We evaluated the parallel algorithm on several commercially popular machines, including the Sun SPARCcenter 2000 [31], SGI Power Challenge [30], DEC AlphaServer 8400 [12], and Cray C90/J90 [33, 34]. Table 4.1 summarizes the configurations and several key parameters of the five parallel systems. In the column "Bus Bandwidth" we report the effective or sustainable bandwidth to main memory. In "Read Latency" we report the minimum amount of time it takes a processor to fetch a piece of data from main memory into a register in response to a load instruction.

The last column in the table shows the programming model used to enable multiprocessing. All the systems provide lightweight multithreading or multitasking libraries. Synchronization and context switching of the threads are accomplished rapidly at the user level, without entering the OS kernel. For P processors, we usually create P (logical) threads for the scheduling loop `Slave_worker()` (Figure 6.1). Scheduling these threads on available physical processors is done by the operating system or runtime library. Thread migration between processors is usually invisible to us. The program is easily portable to multiple platforms. The source codes on different machines differ only in thread spawning and locking primitives.

Table 4.2 summarizes the characteristics of the individual processors in the parallel machines, including the clock speed, the cache size, the peak Mflop rate, and the DGEMM and DGEMV peak Mflop rates. Most DGEMM and DGEMV Mflop rates were measured using vendor-supplied BLAS libraries. When the vendors do not provide a BLAS library, we report the results from PHiPAC [4], with an asterisk (*) beside such a number. For some machines, PHiPAC is often faster than the vendor-supplied DGEMM.

Table 4.2
Some characteristics of the processors used in the parallel systems.

	Clock MHz	On-chip Cache	External Cache	#Flops/ 1 cycle	Peak Mflops	DGEMM Mflops	DGEMV Mflops
MIPS R8000	90	16 KB	4 MB	4	360	340	210
Alpha 21164	300	8 KB-L1 96 KB-L2	4 MB	2	600	350	135
SuperSPARC	50	16 KB	1 MB	1	50	45*	–
C90	240	–	–	4	960	900	890
J90	100	–	–	2	200	190	167

Table 5.1
Differences between the parallel algorithm and serial SuperLU.

Construct	Parallel algorithm
Panel	Restricted so it does not contain branchings in the etree (section 5.2)
Supernode	Restricted to be a fundamental supernode in the etree (section 5.3)
Supernode storage	Use either static or dynamic upper bound (section 5.3)
Pruning & DFS	Use both $G(L^T)$ and pruned $G(L^T)$ to avoid locking (section 5.4)

5. Parallel strategies.

5. Parallel strategies. In this section, we present crucial design choices we have made to parallelize SuperLU, such as how we shall exploit both coarse and fine levels of parallelism, how we shall define the individual tasks, and how we shall deal with the issue of dynamic memory growth.

In order to make the parallel algorithm efficient, we need to make nontrivial modifications to serial SuperLU. All these changes are summarized in Table 5.1 and discussed in the subsections below. These show that the parallel algorithm is not a straightforward parallelization of the serial one and illustrate the program complications arising from parallelization.

5.1. Parallelism. We exploit two sources of parallelism in the sparse LU factorization. The coarse level parallelism comes from the sparsity of the matrix and is exposed to us by the *column elimination tree* (or *column etree* for short) of A. The vertices of this tree are the integers 1 through n, representing the columns of A. The column etree of A is the (symmetric) elimination tree of A^TA provided there is no cancellation in computing A^TA. More specifically, if L_c denotes the Cholesky factor of A^TA, then the parent of vertex j is the row index i of the first nonzero entry below the diagonal of column $L_c(:,j)$. The column etree can be computed from A in time almost linear in the number of nonzeros of A by a variation of an algorithm of Liu [26].

Theorem 5.1 (column elimination tree [19]). *Let A be a square, nonsingular, possibly unsymmetric matrix, and let $PA = LU$ be any factorization of A with pivoting by row interchanges. Let T be the column elimination tree of A.*

1. *If vertex i is an ancestor of vertex j in T, then $i \geq j$.*
2. *If $l_{ij} \neq 0$, then vertex i is an ancestor of vertex j in T.*
3. *If $u_{ij} \neq 0$, then vertex j is an ancestor of vertex i in T.*
4. *Suppose in addition that A is strong Hall (that is, it cannot be permuted to a nontrivial block triangular form). If vertex j is the parent of vertex i in T, then there is some choice of values for the nonzeros of A that makes $u_{ij} \neq 0$ when the factorization $PA = LU$ is computed with partial pivoting.*

Since column i updates column j in LU factorization if and only if $u_{ij} \neq 0$, part 3 of Theorem 5.1 implies that the columns in different subtrees do not update one another. Furthermore, the columns in independent subtrees can be computed without referring to any common memory because the columns they depend on have completely disjoint row indices [20, Theorem 3.2]. It has been shown in a series of studies [14, 15, 19, 20] that the column etree gives the information about all potential dependencies.

In general we cannot predict the nonzero structure of U precisely before the factorization because the pivoting choices and hence the exact nonzero structure depend on numerical values. The column etree can overestimate the true column dependencies. An example is

$$A = \begin{pmatrix} 1 & \bullet & & \\ \bullet & 2 & \bullet & \\ \bullet & & 3 & \bullet \\ \bullet & & & 4 \end{pmatrix},$$

in which the Cholesky factor L_c of $A^T A$ is symbolically full, so the column etree is a single chain. But if the numerical values are such that row 4 is selected as the pivot row at the first step of elimination, column 1 will update neither column 2 nor column 3. Despite the possible overestimate, part 4 of Theorem 5.1 says that if A is strong Hall, this dependency is the strongest information obtainable from the structure of A alone.

Having studied the parallelism arising from different subtrees, we now turn our attention to the dependent columns, that is, the columns having ancestor-descendant relations. When the elimination process proceeds to a stage where there are more processors than independent subtrees, we need to make sure all processors work cooperatively on dependent columns. Thus the second level of parallelism comes from *pipelining* the computations of the dependent columns.

Consider a simple situation with only two processors. Processor 1 gets a task *Task* 1 containing column j, processor 2 gets another task *Task* 2 containing column k, and node j is a descendant of node k in the etree. The (potential) dependency says only that *Task* 2 cannot finish its execution before *Task* 1 finishes. However, processor 2 can start *Task* 2 right away with the computations not involving column j; this includes performing the symbolic structure prediction and accumulating the numeric updates using the finished columns that are descendants in the etree. After processor 2 has finished this part of the computation, it has to wait for *Task* 1 to finish. (If *Task* 1 is already finished at this moment, processor 2 does not waste any time waiting.) Then processor 2 will predict the new fills and perform numeric updates that may result from the finished columns in *Task* 1. In this way, both processors do useful work concurrently while still preserving the precedence constraint. Note that we assume the updates can be done in any order. This could give different (indeed, nondeterministic) numerical results from run to run.[2]

Although this pipelining mechanism is complicated to implement, it is essential to achieve higher concurrency. This is because, in most problems, a large percentage of the computation occurs at upper levels of the etree, where there are fewer branches than processors. An extreme example is a dense matrix, the etree of which is a single chain. In this case, the parallel SuperLU "reduces to" a pipelined column-oriented dense LU algorithm.

[2] In order to guarantee determinism, we must statically assign the tasks to processors. The performance cost we pay for determinism may be load imbalance and reduced parallelism. We are considering adding an option that guarantees determinism, in order to help debug codes calling parallel SuperLU.

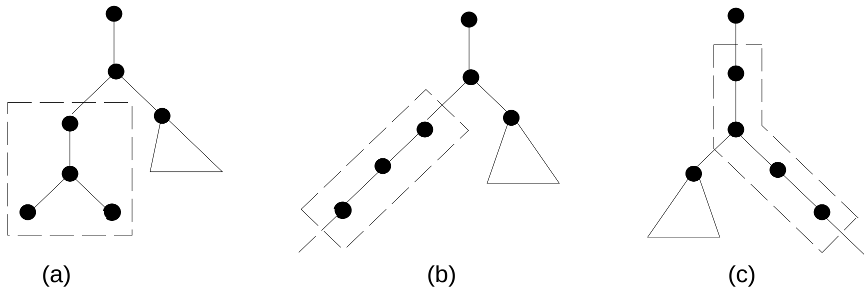


FIG. 5.1. *Panel definition.* (a) *a relaxed supernode at the bottom of the column etree;* (b) *consecutive columns from part of one branch of the etree;* (c) *consecutive columns from more than one branch of the etree.*

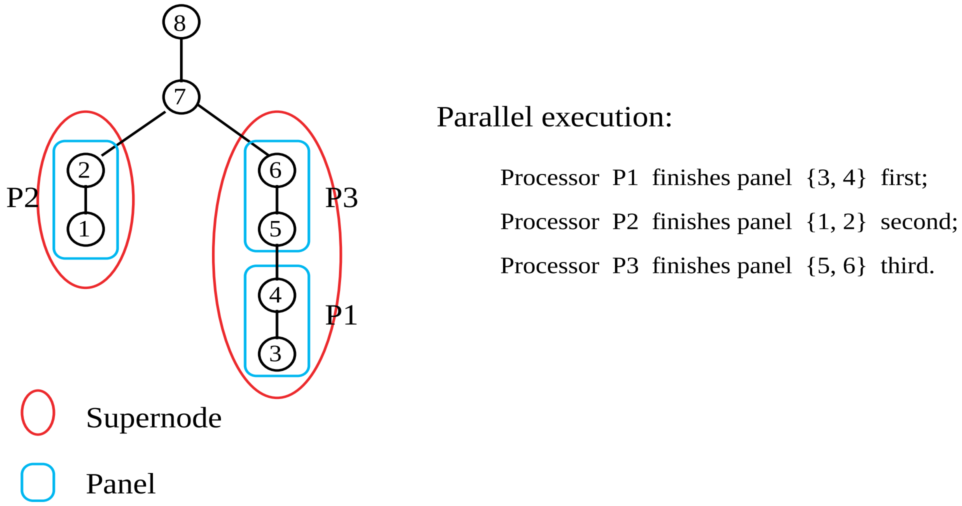
5.2. Panel tasks. As studied by Demmel et al. [6], the introduction of supernodes and panels makes the computational kernels highly efficient. To retain the serial algorithm's ability to reuse data in cache and registers, we treat the factorization of one panel as a unit task to be scheduled; it computes the part of U and the part of L for all columns within this panel. Choosing a panel as scheduling unit affords the best granularity on the SMPs we targeted and requires only modest changes to the serial code [6]. The alternative, blocking the matrix by rows and columns [23, 29], introduces too much synchronization overhead to make it worthwhile on SMPs with modest parallelism.

A panel task consists of two distinct subtasks. The first corresponds to the outer factorization, which accumulates the updates from the descendant supernodes. The second subtask is to perform the panel's inner factorization. We exploit parallelism within the first subtask, but not the second.

Since the parallel algorithm uses the column etree as the main scheduling tool, it is worth studying the relationship between the panels and the structure of the column etree. We assume that the columns of the matrix are ordered according to a postorder on the column etree. We expect a postorder on the column etree to bring together unsymmetric supernodes, just as a postorder on the symmetric etree brings together symmetric supernodes. Pictorially, panels can be classified into three types, depending on where they are located in the etree, as illustrated in Figure 5.1.

In the pipelining algorithm, panels of type (c) complicate the record-keeping if a processor owning this panel needs to wait for, and later perform, the updates from the busy panels down the etree. To simplify this, we imposed two restrictions. We first restricted the definition of panels so that type (c) panels do not occur. We will let a panel stop before a node (column) that has more than one child in the etree. That is, every branching node necessarily starts a new panel. Second, we make sure that all busy descendant panels always form one path in the etree. Then the processor waiting for these busy panels can simply walk up the path in the etree starting from the most distant busy descendant.

With this restricted definition of panels, there will be more panels of smaller sizes. The question arises of whether this will hurt performance. We studied the distribution of floating-point operations on different panel sizes for all of our test matrices and observed that usually more than 95% of the floating-point operations are performed in the panels of largest size, and these panels tend to occur at a few topmost levels of the etree. Thus, panels of small sizes normally do not represent much



Parallel execution:

Processor P1 finishes panel {3, 4} first;

Processor P2 finishes panel {1, 2} second;

Processor P3 finishes panel {5, 6} third.

FIG. 5.2. *A snapshot of parallel execution.*

computation. On uniprocessors, we see almost identical performance using the original and the new definitions of panels. Therefore, we believe that this restriction on panels simplifies and accelerates the parallel scheduling algorithm with little performance loss on individual processors.

5.3. Supernode storage using nonzero column counts in *QR* factorization. It is important to store the columns of a supernode *consecutively* in memory, so that we can call BLAS routines directly into place without paying the cost of copying the columns into contiguous memory. Although this contiguity is easy to achieve in a sequential code, it poses problems in the parallel algorithm.

Consider the scenario of parallel execution depicted in Figure 5.2. According to the order of the finishing times specified in the figure, the panel consisting of columns {3,4} will be stored in memory first, followed by panel {1,2}, and then followed by panel {5,6}. The supernode {3,4,5,6} is thus separated by the panel {1,2} in memory. The major difficulty comes from the fact that the supernodal structure emerges dynamically as the factorization proceeds, so we cannot statically calculate the amount of storage required by each supernode. Another difficulty is that panels and supernodes can overlap in several different ways.

One immediate solution would be not to allow any supernode to cross the boundary of a panel. In other words, the leading column of a panel would always be treated as the beginning of a new supernode. Thus a panel could possibly be subdivided into more than one supernode, but not vice versa. In such circumstances, the columns of a supernode would always be contiguous in memory because they would be assigned to a single processor by the scheduler. Each processor would simply store a (partial) ongoing supernode in its local temporary store and copy the whole supernode into the global data structure as soon as it was finished.

This restriction on supernodes would mean that the maximum size of supernodes would be bounded by the panel size. As discussed in section 7.5 (see also Li [25]), for best per-processor efficiency and parallelism, we would like to have large supernodes but relatively small panels. These conflicting demands make it hard to find one good size for both supernodes and panels. We conducted an experiment with this scheme

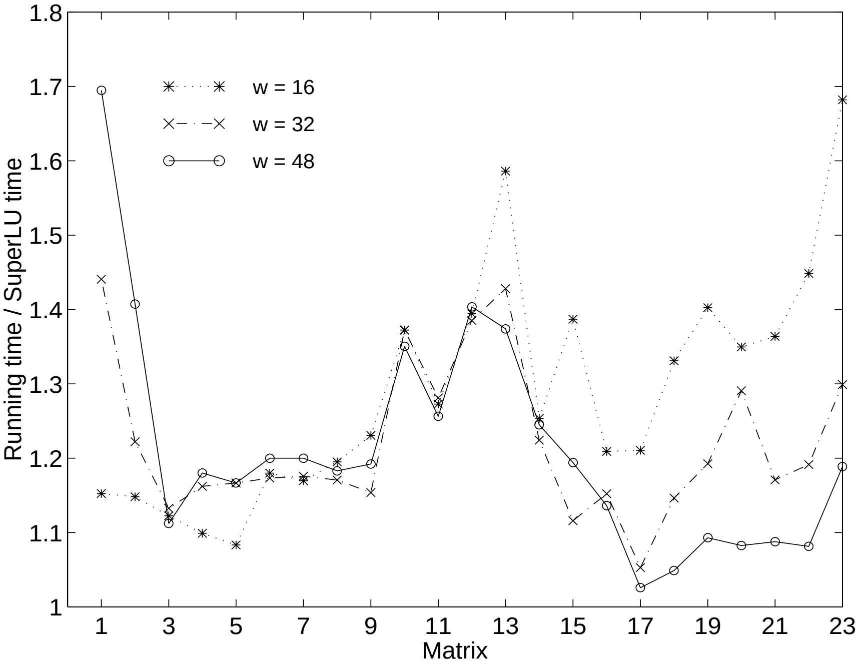


FIG. 5.3. *The sequential runtime penalty for requiring that a leading column of a panel also starts a new supernode. The times are measured on the RS*/6000-590.

for the sequential algorithm. Figure 5.3 shows the uniprocessor performance loss for various panel sizes (i.e., maximum sizes of supernodes). For large matrices, say matrices 12–21, the smaller panels and supernodes result in severe performance loss. For example, when $w = 16$, the slowdown can be as large as 20% to 68%. Even for large panel sizes, such as $w = 48$, the slowdown is still between 5% and 20%. However, in the parallel algorithm, such large panels give rise to too large a task granularity and severely limit the level of concurrency. We therefore feel that this simple solution is not satisfactory. Instead, we seek a solution that does not impose any restriction on the relation between panels and supernodes and that allows us to vary the size of panels and supernodes independently in order to better trade off concurrency and single-processor efficiency.

Our second and preferred solution is to preallocate space that is an upper bound on the actual storage needed by each supernode in the L factor, *irrespective of the numerical pivoting choices*. Then there will always be space to store supernode columns as they are computed. We now describe how we preallocate enough (but not too much) space.

After Gaussian elimination with partial pivoting, we can write $A = P_1 L_1 P_2 L_2 \cdots P_{n-1} L_{n-1}\ U$, where P_i is an elementary permutation matrix representing the row interchange at step i, and L_i is a unit lower triangular matrix with its ith column containing the multipliers at step i. We now define L as the unit lower triangular matrix whose ith column is the ith column of L_i, such that $L - I = \sum_i (L_i - I)$.[3] We

[3] This L is different from the $\hat{L}$ in $PA = \hat{L}U$. Both L and $\hat{L}$ contain the same nonzero values, but in different positions. In this section, L is used as a data structure for storing $\hat{L}$.

shall make use of the following structure containment property in our storage scheme. Here we quote only the result without proof.

THEOREM 5.2 (see [14, 16]). *Consider the QR factorization $A = QR$ using Householder transformations. Let H be the symbolic Householder matrix consisting of the sequence of Householder vectors used to represent the factored form of Q. In other words, we assume that no entries of H or R are zero because of numerical cancellation. If A is a nonsingular matrix with nonzero diagonal, and L and U are the triangular factors of A represented as above, then $Struct(L) \subseteq Struct(H)$, and $Struct(U) \subseteq Struct(R)$.*

In what follows, we describe how this upper bound can facilitate our storage management for the L supernodes. First, we need a notion of *fundamental* supernode, which was introduced by Ashcraft and Grimes [3] for symmetric matrices. In a fundamental supernode, every column except the last (numbered highest) is an only child in the elimination tree. Liu, Ng, and Peyton [27] gave several reasons why fundamental supernodes are appropriate, one of which is that the set of fundamental supernodes is the same regardless of the particular etree postordering. For consistency, we now also impose this restriction on the supernodes in L and H. For convenience, let S_L denote the fundamental supernodes in the L factor and S_H denote the fundamental supernodes in the symbolic Householder matrix H. We shall omit the word "fundamental" when it is clear.

Our code breaks the L supernode at the boundary of an H supernode, forcing the L supernode to be contained in the H supernode. In fact, if we use fundamental L supernodes and ignore numerical cancellation (which we must do anyway for symmetric pruning), we can show that an L supernode is always contained in an H supernode [21].

Our objective is to allocate storage based on the number of nonzeros in S_H, so that this storage is sufficiently large to hold S_L. Figure 5.4 illustrates the idea of using S_H as a bound. Two supernodes in S_L from different branches of the etree will go to their corresponding memory locations of the enclosing supernodes in S_H. Even if an H supernode breaks into multiple L supernodes, those L supernodes will all lie on one path in the column etree. Thus an L supernode from a different subtree cannot interrupt the storage for a supernode as in Figure 5.2. Since the panels (and hence the supernodes) within an H supernode are finished in order of increasing column numbers, the columns of each S_L supernode are contiguous in the storage of the S_H supernode.

To determine the storage for S_H, we need an efficient algorithm to compute the column counts $nnz(H_{*j})$ for H. We also need to identify the first vertex of each supernode in S_H. Then the number of nonzeros in each supernode is simply the product of the column count of the first vertex and the number of columns in the supernode.

Finding the first vertex and computing the column count can be done using a variant of the QR-column-count algorithm by Gilbert, Ng, and Peyton [21]. The modified QR-column-count algorithm takes $Struct(A)$ and the postordered T as inputs, and computes $nnz(H_{*j})$ and S_H. The complexity of the algorithm is $O(m\ \alpha(m, n))$, where $m = nnz(A)$ and $\alpha(m, n)$ is the slowly growing inverse of Ackermann's function coming from disjoint set union operations. In practice, it is as fast as computing the column etree T [25, Table 5.2]. In both the etree and QR-column-count algorithms, the disjoint set union operations are implemented using path halving and no union by rank (see [22] for details).

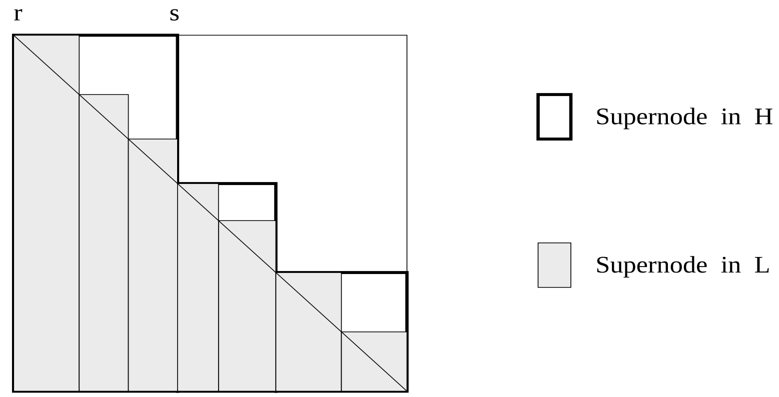


FIG. 5.4. *Bound of the L supernode storage using the supernodes in H.*

One remaining issue is what we should do if the static storage given by an upper bound structure is much too generous than actually needed. We developed a dynamic prediction scheme as a fallback for this situation. In this scheme, we still use the supernode partition S_H. Unlike the static scheme, which uses the column counts $nnz(H_{*j})$, we dynamically compute the column count for the first column of each supernode in S_H as follows. When a processor obtains a panel that includes the first column of some supernode $H(:, r : s)$ in S_H, the processor invokes a search procedure on the directed graph $G(L(:, 1 : r-1)^T)$, using the nonzeros in $A(:, r : s)$, to determine the union of the row structures in the submatrix $(r : n, r : s)$. We use the notation $D(r : n, r : s)$ to denote this structure. It is true that

$$(5.1)\quad Struct((\hat{L}+U)(r : n, r : s)) \subseteq Struct(D(r : n, r : s)) \subseteq Struct(H(r : n, r : s)) \ .$$

The search procedure is analogous to (but simpler than) the panel symbolic step (Figure 2.1, step (1)); now we want only to determine the count for the column $D(r : n, r)$, without the nonzero structure or the topological order of the updates. Then we use the product of $nnz(D(r : n, r))$ and $s - r + 1$ to allocate storage for the L supernodes within columns r through s. Since $nnz(L(r : n, r)) \leq nnz(D(r : n, r)) \leq nnz(H(r : n, r))$, the dynamic storage bound so obtained is usually tighter than the static bound.

The storage utilizations for the supernodes in S_L are tabulated in Table 5.2. The utilization is calculated as the ratio of the actual number of nonzeros in the supernodes of L to the number of nonzeros in the supernodes of H. When collecting this data, the maximum supernode size t was set to 64. For most matrices, the storage utilizations using the static bound by H are quite high; they are often greater than 70% and are over 85% for 14 out of the 21 problems. However, in the static scheme, the storage utilizations for matrices 1 (MEMPLUS), 15 (VENKAT01), and 21 (WANG3) are only 4%, 11%, and 14%. The dynamic scheme overcomes those low utilizations. For the three matrices above, the utilizations in the dynamic scheme are 68%, 74%, and 89%. These percentage utilizations are quite satisfactory. For other problems, the dynamic approaches also result in higher utilizations.

The runtime overhead associated with the dynamic scheme is usually between 2% and 15% on the single processor RS/6000-590. From these experiments, we conclude that the static scheme using H often gives a tight enough storage bound for S_L. For some problems, the dynamic scheme must be employed to achieve better storage

Table 5.2

Supernode storage utilization, using static and dynamic upper bounds. The number tabulated is the ratio of the number of nonzeros in supernodes of L to that in the prediction H.

	Matrix	Static	Dynamic
1	Memplus	.04	.68
2	Gemat11	.85	.90
3	Rdist1	.72	.73
4	Orani678	.56	.90
5	Mcfe	.73	.89
6	Lnsp3937	.84	.92
7	Lns_3937	.86	.94
8	Sherman5	.92	.96
9	Jpwh_991	.88	.94
10	Sherman3	.89	.91
11	Orsreg_1	.90	.92
12	Saylr4	.89	.92
13	Shyy161	.91	.92
14	Goodwin	.95	.98
15	Venkat01	.11	.74
16	Inaccura	.96	.99
17	Af23560	.95	.97
18	Dense1000	1.00	1.00
19	Raefsky3	.99	.99
20	Ex11	.99	1.00
21	Wang3	.14	.89
22	Raefsky4	.99	.99
23	Vavasis3	.95	.98

utilization. Then the program will suffer from a certain amount of slowdown. Our code tries the static scheme first and switches to the dynamic scheme only if the static scheme requests more space than is available.

5.4. Nonblocking pruning and DFS. The idea of symmetric pruning [10, 11] is to use a graph G' with fewer edges than the graph G of L^T to represent the structure of L. Traversing G' gives the same reachable sets as traversing G, but is less expensive. As shown by Eisenstat and Liu [11], this technique significantly reduces the symbolic factorization time.

In the sequential algorithm, in addition to the adjacency structure for G, there is another adjacency structure to represent the reduced graph G'. For each supernode, since the row indices are the same among the columns, we only store the row indices of the first column of G and the row indices of the last column of G'. (If we used only one adjacency list for each supernode, since pivoting may have reordered the rows so that the pruned and unpruned rows are intermingled in the original row order, it would be necessary to reorder all of L and A to account for it.)

Figure 5.5 illustrates the storage layout for the adjacency lists of G and G' of a sample matrix. Array `Lsub[*]` stores the row subscripts. `G_ptr[*]` points to the beginning of each supernode in array `Lsub[*]`. `G'_ptr[*]` points to the pruned location of each supernode in array `Lsub[*]`. Using `G_ptr` and `G'_ptr` together, we can locate the adjacency list for each supernode in G'. This matrix has four supernodes: $\{1,2\}$, $\{3\}$, $\{4,5,6\}$, and $\{7,8,9,10\}$. The adjacency lists for G and G' are interleaved by supernodes in the global memory `Lsub[*]`. The storage for the adjacency structure of G' is reclaimed at the end of the factorization.

The pruning procedure works on the adjacency lists in G'. Each adjacency list of a supernode (actually only the last column in the supernode) is pruned at the position

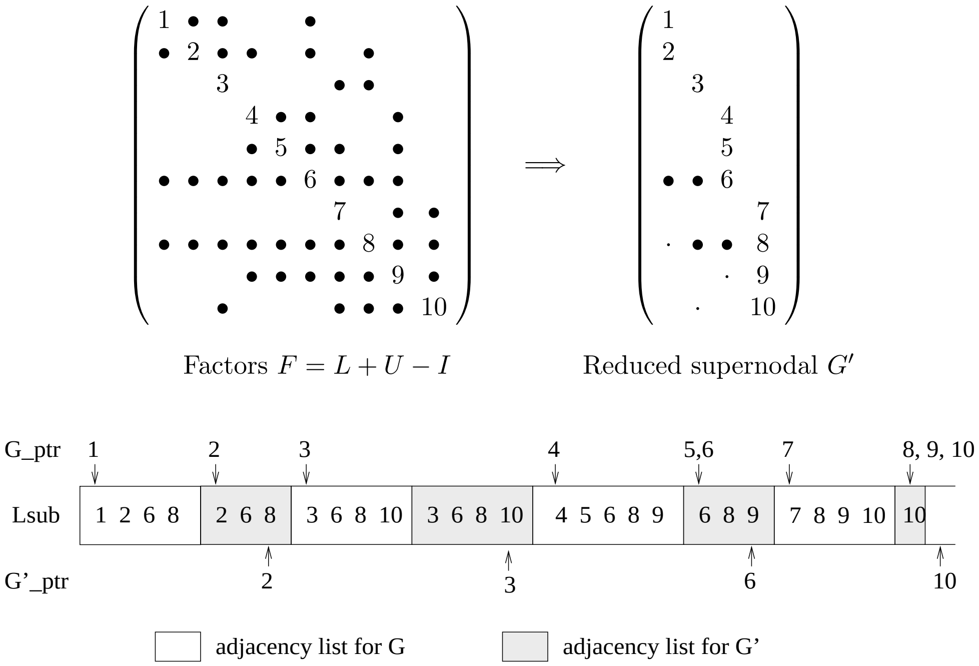


FIG. 5.5. *Storage layout for the adjacency structures of G and G'.*

of the first symmetric nonzero pair in the factored matrix F, as indicated by the small "·" in the figure. Both panel DFS and column DFS traverse the adjacency structure of G', as given by `G'_ptr[*]` in Figure 5.5.

In the parallel algorithm, contention occurs when one processor is performing DFS using the adjacency list in G' of column j (a READ operation), while another processor is pruning the structure of column j and thus reordering the row indices in the list (a MODIFY operation). There are two possible solutions to avoid this contention. The first solution is to associate one mutually exclusive (mutex) lock with each adjacency list of G'. A processor acquires the lock before it prunes the list and releases the lock thereafter. Similarly, a processor uses the lock when performing DFS on the list. Although the critical section for pruning can be short, the critical section for DFS may be very long, because the list must be locked until the entire DFS starting from all nodes in the list is completed. During this period, all the other processors attempting to prune the list or to traverse the list will be blocked. Therefore this approach may incur too much overhead, and the benefit of pruning may be completely offset by the cost of locking.

We now describe a better algorithm that is free from locking. We will use both graphs G' and G to facilitate the DFS. Recall that each adjacency list is pruned only *once* during the factorization. We will associate with each list a status bit indicating whether or not it is pruned. Once a list is pruned, all the subsequent traversals of the list involve only READ operations and hence do not require locking. If the search procedure reaches a list of G' that has not yet been pruned, we will direct the search procedure to traverse the list of the corresponding column in G rather than G'. So, when the search algorithm reaches column j, it does the following:

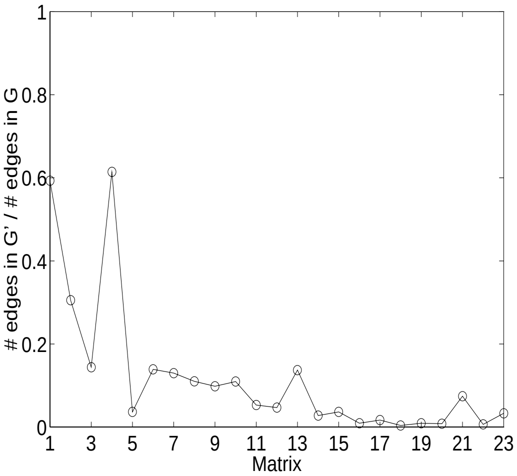


FIG. 5.6. *Number of edges in G' versus number of edges in G.*

```
if column $j$ has been pruned then
    continue search from nodes in the $G'$-list of column $j$;
else
    continue search from nodes in the $G$-list of column $j$;
endif
```

This scheme prevents us from using one minor working-storage optimization from the sequential algorithm: sequential SuperLU uses separate G- and G'-lists for supernodes with two or more columns but overlaps the lists for singleton supernodes. The parallel code must use both lists for every supernode.

Since G' is a subgraph of G, the DFS in the parallel code may traverse more edges than those in the sequential code. This is because in the parallel algorithm, a supernode may be pruned later than in the sequential algorithm. However, because of the effectiveness of symmetric reduction, very often the search still uses the pruned list in G'. So it is likely that the time spent in the occasional extra search in the G-lists is much less than that when using the locking mechanism. Figure 5.6 shows the relative size of the reduced supernodal graph G', and Figure 5.7 shows the fraction of searches that use the G'-lists. The numbers in both figures are collected on a single processor Alpha 21164.

6. The asynchronous scheduling algorithm. Having described the parallel strategies, we are now in a position to describe the parallel factorization algorithm. Several methods have been proposed to perform sparse Cholesky factorization [13, 24, 28] and sparse *LU* factorization [2, 5, 17, 20] on SMPs. A common practice is to organize the program as a self-scheduling loop, interacting with a global pool of tasks that are ready to be executed. Each processor repeatedly takes a task from the pool, executes it, and puts new ready task(s) in the pool. This pool-of-tasks approach has the merit of balancing work load automatically even for tasks with large variance in granularity. There is no notion of *ownership* of tasks or submatrices by processors—the assignment of tasks to processors is completely dynamic, depending on the execution speed of the individual processors. Our scheduling algorithm employs this model as well. This is in contrast to some implementations of sparse Cholesky,

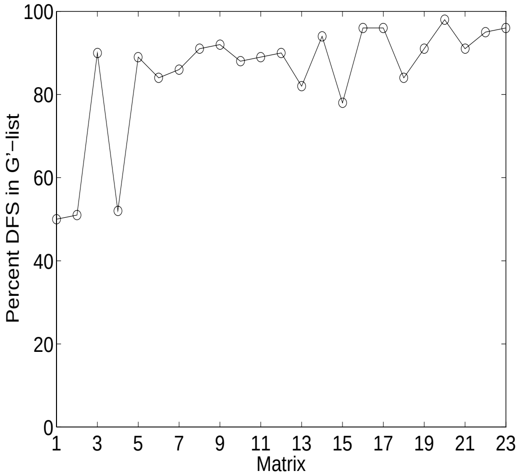


Fig. 5.7. *Percent of individual search steps that take place in adjacency lists in G'.*

```
Slave_worker()
newpanel = NULL;
while ( there are more panels ) do
    oldpanel := newpanel;
    newpanel := Scheduler( oldpanel, queue );
    if ( newpanel is a relaxed supernode ) then
        relaxed_supernode_factor( newpanel );
    else
        panel_symbolic_factor( newpanel );
            • Determine which supernodes will update panel newpanel;
            • Skip all BUSY panels/supernodes;
        panel_numeric_factor( newpanel );
            • Accumulate updates from the DONE supernodes, updating newpanel;
            • Wait for the BUSY supernodes to become DONE, then predict
              new fills and accumulate more updates to newpanel;
        inner_factorization( newpanel ); /* independent from other processors */
            • Supernode-column update within the panel;
            • Row pivoting;
            • Detect supernode boundary;
            • Symmetric structure pruning;
    end if;
end while;
```

Fig. 6.1. *The parallel scheduling loop to be executed on each processor.*

which can schedule work to processors carefully and cheaply ahead of time [23]. The dynamic nature of partial pivoting prevents us from doing this.

Our scheduling approach used some techniques from the parallel column-oriented algorithm developed by Gilbert [20]. Figure 6.1 sketches the top level scheduling loop. Each processor executes this loop until its termination criterion is met, that is, until all panels have been factorized.

The parallel algorithm maintains a central priority queue of tasks (panels) that are ready to be executed by any free processor. The content of this task queue can be accessed and altered by any processor. At any moment during the elimination, a panel is tagged with a certain state, such as READY, BUSY, or DONE. Every processor repeatedly asks the scheduler (at line 4) for a panel task in the queue. The `Scheduler()` routine implements a priority-based scheduling policy described below. The input argument *oldpanel* denotes the panel that was just finished by this processor. The output argument *newpanel* is a newly selected panel to be factorized by this processor. The selection preference is as follows:

(1) The scheduler first checks whether all the children of *oldpanel*'s parent panel, say, *parent*, are DONE. If so, *parent* now becomes a new leaf and is immediately assigned to *newpanel* on the same processor.

(2) If *parent* still has unfinished children, the scheduler next attempts to take from the queue a panel which can be computed without pipelining, that is, a leaf panel.

(3) If no more leaf panels exist, the scheduler will take a panel that has some BUSY descendant panels currently being worked by other processors. Then the new panel must be computed by this processor in a pipelined fashion.

One might argue that (1) and (2) should be reversed in priority. Choosing to eliminate the immediately available parent first is primarily concerned with locality of reference. Since a just-finished panel is likely to update its parent or other ancestors in the etree, it is advantageous to schedule its parent and other ancestors on the same processor.

To implement the priority scheme above, the task queue is initialized with the leaf panels, that is, the relaxed supernodes, which are marked as READY. Later on, `Scheduler()` may add more panels at the tail of the queue. This happens when all the children of *newpanel*'s parent, *parent*, are BUSY; *parent* is then enqueued and is marked as eligible for pipelining. By rule (1), some panel in the middle of the queue may be taken when all its children are DONE. This may happen even before all the initial leaf panels are finished. All the intermediate leaf panels are taken in this way. By rules (2) and (3), `Scheduler()` removes tasks from the head of the queue.

It is worth noting that the executions of different processors are completely asynchronous. There is no global barrier; the only synchronization occurs at line 13 in Figure 6.1, where a processor stalls when it waits for some BUSY updating supernode to finish. As soon as this BUSY supernode is finished, all the processors waiting on this supernode are awakened to proceed. This type of synchronization is commonly referred to as *event notification.* Since the newly finished supernode may produce new fills to the waiting panels, the symbolic mechanism is needed to discover and accommodate these new fills.

7. Parallel performance.

We now evaluate the performance of the parallel algorithm. The organization of this section is as follows. Section 7.1 summarizes the observed speedups on various platforms, relative to serial SuperLU. Section 7.2

Table 7.1

Time notation used to evaluate performance of the parallel algorithm.

Notation	Meaning
T_s	SuperLU best serial time
T'_s	SuperLU serial time with smaller blocking tuned for parallel code
T_1	Execution time of the parallel algorithm on one processor
T_P	Parallel execution time on P processors
T_I	Total idle time of all processors

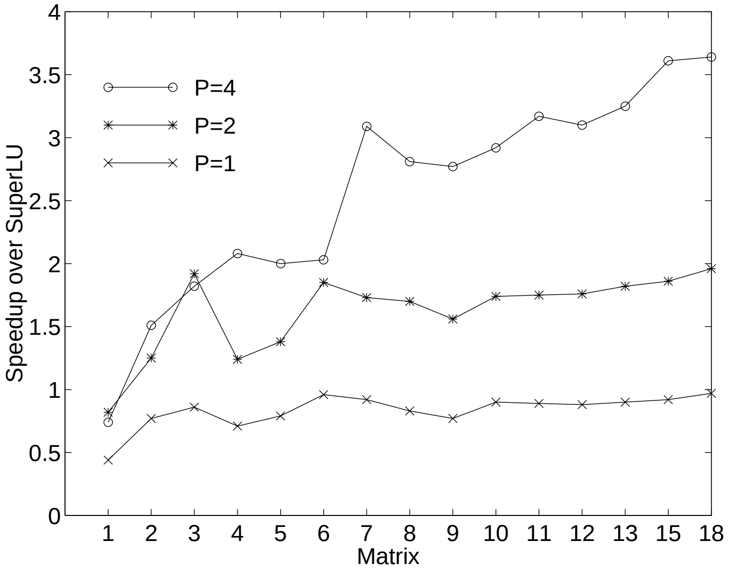


FIG. 7.1. *Speedup on a* 4*-CPU Sun SPARCcenter* 2000*.*

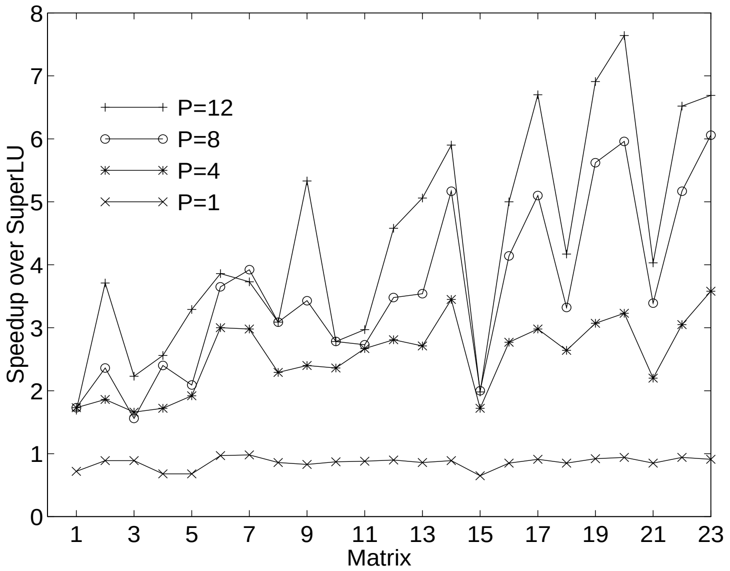


FIG. 7.2. *Speedup on a* 12*-CPU SGI Power Challenge.*

quantifies parallel overhead and its impact on performance. Section 7.3 gives the statistics of load balance. Section 7.4 studies the space efficiency of the algorithm. In the performance evaluation, we timed various parts of the algorithms. Table 7.1 lists our time notation.

7.1. Speedup summary. Figures 7.1 through 7.5 report the speedups of the parallel algorithm on the five platforms, with number of threads "P" varied. Because of memory limits we could not test all problems on the SPARCcenter 2000. The speedup is measured against the best sequential runtime achieved by SuperLU on a

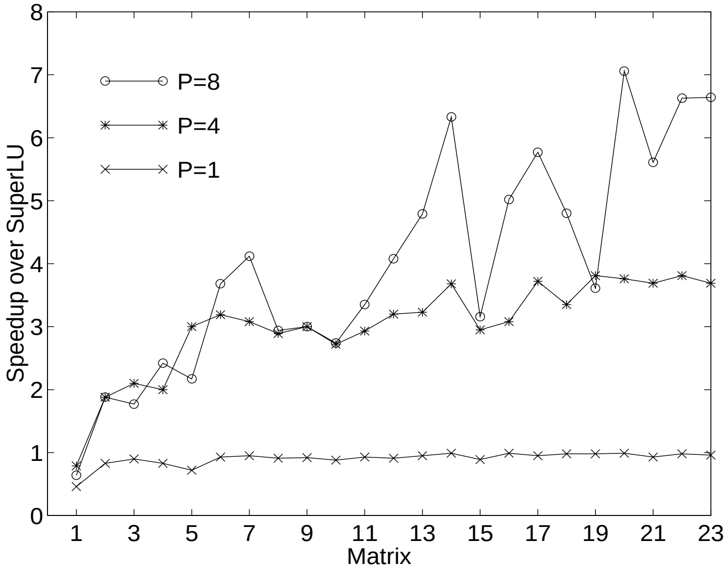


Fig. 7.3. *Speedup on an* 8-*CPU DEC AlphaServer* 8400.

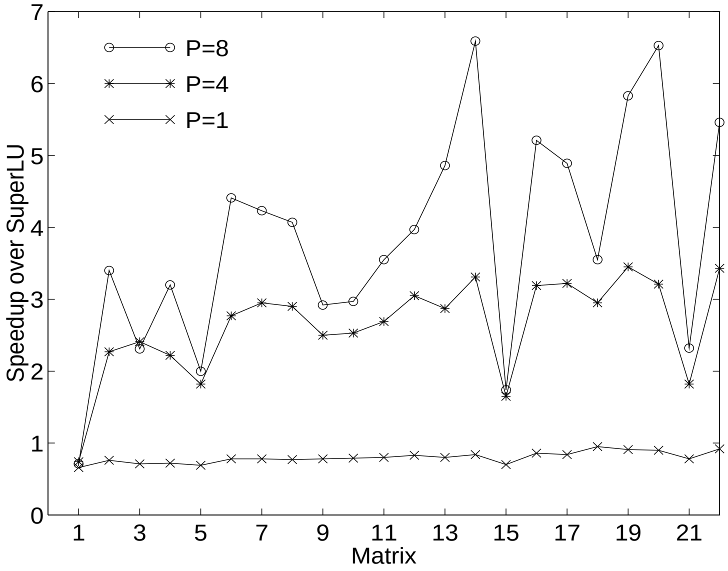


Fig. 7.4. *Speedup on an* 8-*CPU Cray C*90.

single processor of each parallel machine; that is, $speedup = T_s/T_P$.

In each figure, the bottom curve labeled "$P = 1$" illustrates the overhead in the parallel code when compared to the serial SuperLU, using the same blocking parameters. The structure of the parallel code, when run on a single processor, does not differ much from sequential SuperLU, except that a global task queue and various locks are involved. The extra work in the parallel code is purely integer arithmetic. In order to achieve a higher degree of concurrency, the panel size (w) and maximum size

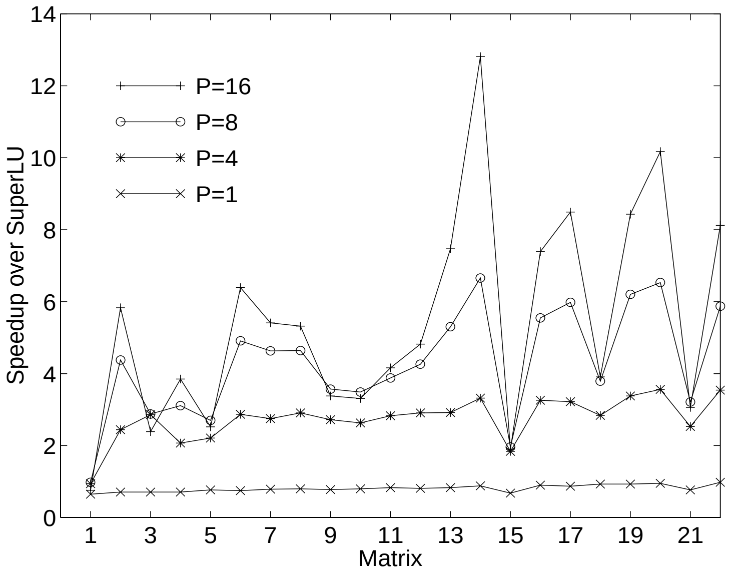


FIG. 7.5. *Speedup on a* 16-*CPU Cray J*90.

of a supernode (*maxsup*) for "$P > 1$" are set smaller than those used for "$P = 1$".[4]

We also tabulate these speedup figures in the appendix (Tables A.1 through A.5), where the last two columns in each table show the factorization time and Megaflop rate, corresponding to the largest number of processors used.

7.2. Impact of overhead on parallel efficiency. The parallel algorithm experiences some overhead, which mainly comes from three sources: the reduced per-processor efficiency due to smaller granularity of unit tasks, accessing critical sections via locks, and orchestrating the dependent tasks via event notification. The purpose of this section is to understand how much time is spent in each part of the algorithm and explain the speedups we saw in section 7.1.

7.2.1. Decreased per-processor performance due to smaller blocking. The first overhead is due to the necessity of reducing the blocking parameters in order to achieve more concurrency. Recall that two blocking parameters affect performance: panel size (w) and maximum size of a supernode (*maxsup*). For better per-processor performance, we prefer larger values. On the other hand, the large granularity of unit tasks limits the degree of concurrency.

On the Cray J90, this trade-off is not so important, because $w = 1$ is good for the sequential algorithm. We therefore also use $w = 1$ in the parallel algorithm. When varying the value of *maxsup*, we find that performance is quite robust in the range between 16 and 64.

On the Power Challenge and AlphaServer 8400, we observe more dramatic differences with varied blockings. Figures 7.6 and 7.7 illustrate this loss of efficiency for several large problems on single processors of the two machines. In this experiment, the parallel code is run on a single processor with two different settings of w and *maxsup*. Figure 7.6 shows, on a single processor Power Challenge, the ratio of the runtime with the best blocking for 1 CPU ($w = 24, maxsup = 64$) to the runtime

[4] Both w and *maxsup* denote the size in number of columns.

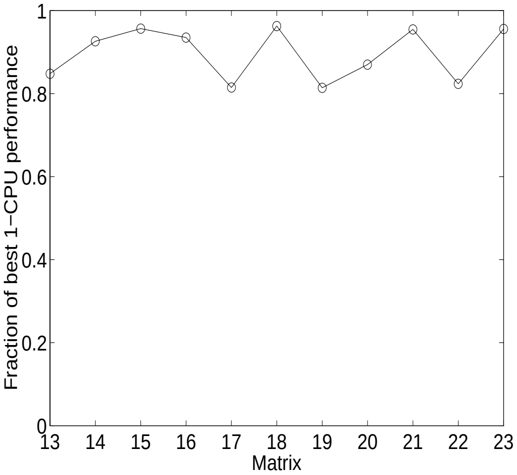


FIG. 7.6. *T_s/T'_s for serial SuperLU on* 1*-CPU Power Challenge.*

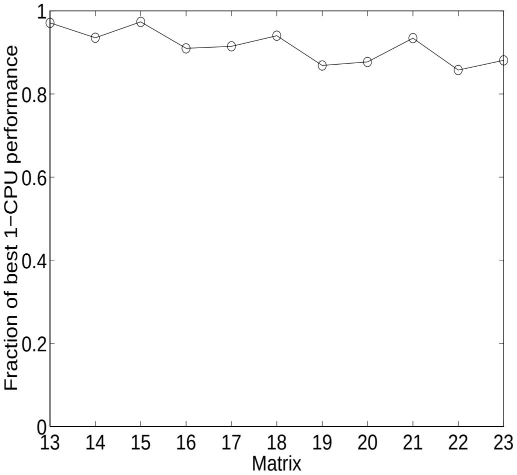


FIG. 7.7. *T_s/T'_s for serial SuperLU on* 1*-CPU AlphaServer* 8400.

with the best blocking for 12 CPUs ($w = 12, maxsup = 48$). Figure 7.7 shows the analogous ratio for the 8-CPU AlphaServer 8400. On the Power Challenge, the blocking used for best parallel performance achieves only 81% uniprocessor efficiency for matrices 17 (AF23560) and 19 (RAEFSKY3). The corresponding lowest number on the AlphaServer 8400 is 86% for matrix 22 (RAEFSKY4).

7.2.2. Accessing critical sections. Several places in the program must be protected by mutual exclusion. In Table 7.2, we roughly count the number of times the program acquires and relinquishes various locks. Note that the total number of lockings performed is independent of the number of processors. Since we want to allow different processors to enter different critical sections simultaneously, we use five mutex variables to guard the five critical regions.

To see how much cost is associated with locking, in Table 7.3 we measured the time it takes to acquire and relinquish a lock on several platforms, with different numbers

Table 7.2
Number of lockings performed.

Critical section	Counts
Call `Scheduler()`	Number of panels (approx.)*
Allocate storage for row indices of L (`Lsub`)	Number of supernodes
Allocate storage for L supernodes (S_L)	Number of supernodes
Allocate storage for a column of U (`Usub/Uval`)	Number of columns
Increment supernode number `nsuper`	Number of supernodes

*Here we assume that `Scheduler()` returns a new panel upon each call.

Table 7.3
Time in microseconds (cycles) to perform a single lock and unlock.

Machine	$P = 1$	$P = 4$	P=8
SPARCcenter 2000	1.63 (82)	4.34 (217)	4.36 (218)
Power Challenge	1.13 (102)	1.98 (179)	2.02 (182)
AlphaServer 8400	0.98 (294)	2.26 (678)	2.71 (814)
Cray C90	1.34 (323)	1.09 (261)	1.40 (336)
Cray J90	2.67 (267)	4.17 (417)	4.42 (442)

of threads P. The figure in parentheses is the number of clock cycles. In this small benchmark code, the critical section is simply one statement, to increment a counter. The locking and unlocking are placed around this statement. The measurement is done in a tight loop with many iterations. When there is more than one thread, the time increases slightly, but not linearly in the number of threads.

The uniprocessor slowdown is partly due to the overhead incurred by using these locks, when there are no other processors competing for the locks. By multiplying the time for a single lock/unlock in Table 7.3 by the number of the lockings performed in Table 7.2, we can estimate the locking overhead. As a concrete example, let us consider a medium-size matrix, number 13 (Shyy161), on a single processor Cray J90. Since the sequential code performance is 26 Mflops, each lock/unlock is equivalent to roughly 69 floating-point operations. When the factorization is performed with panel size $w = 1$, the total number of lock acquisitions is 237004, which, when multiplied by 2.67 microseconds, results in about 0.64 seconds. This is less than 3% of the entire factorization time (24.85 seconds). We observe that this percentage is typical for large matrices (also the bottom curve in Figure 7.8). The locking overhead also varies with machines. For example, it is higher on the Cray J90 than on the Power Challenge or the AlphaServer 8400.

7.2.3. Coordinating dependent tasks. The third source of overhead is due to insufficient parallelism in the pipelined executions of the dependent panels. Dependent panels are those that have an ancestor-descendant relation in the column etree. When a processor factoring a panel needs an update from a BUSY descendant panel, this processor simply spins, waiting for that panel to finish, as shown at line 13 in the scheduling loop of Figure 6.1. During the spin wait the processor does nothing useful. The total amount of spin wait time observed is significant in some cases, especially with larger numbers of processors. For example, for matrix 16 (Inaccura), on the 12-CPU Power Challenge, about 40% of the parallel runtime is spent spinning. The corresponding number for the dense matrix is about 58%. The dense matrix is the worst one, because the factorization of all panels must be carried out in pipelined fashion.

Figure 7.8 depicts the locking overhead (section 7.2.2) and the spinning due to

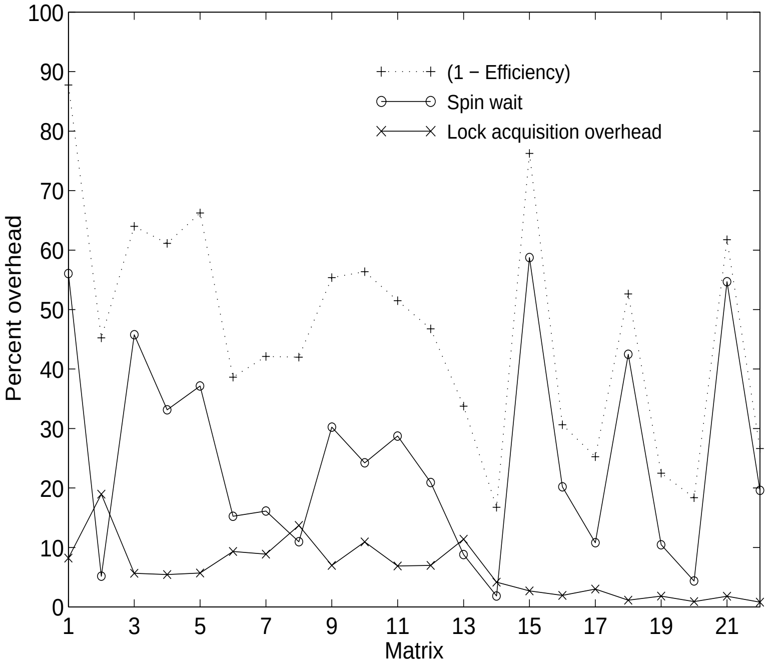


FIG. 7.8. *Parallel overhead in percent on an* 8*-CPU Cray J*90.

dependencies on the 8-CPU Cray J90. The locking overhead also includes the possible contention from the 8 processors. In this figure, we also plot the inefficiency (i.e., 1-*efficiency*) of the parallel algorithm. For most matrices, the spinning overhead due to dependencies is much higher than the overhead from lock acquisition.

7.2.4. Putting all overheads together. In this subsection we evaluate the effect of the combined overheads on the parallel efficiency. In summary, the overheads include

Overhead (1): reduced uniprocessor performance due to smaller blocking;
Overhead (2): accessing critical sections;
Overhead (3): idle time (from spin wait in the panel pipeline and in the top-level scheduling loop).

Overhead (1) affects only uniprocessor performance. Overhead (2) decreases both uniprocessor performance of the parallel code and parallel performance. Compared with the serial execution, the parallel execution experiences more contention for locks. But Table 7.3 and Figure 7.8 indicate that runtime does not increase significantly because of contention. Therefore, we may model (2) as only adding overhead to the uniprocessor execution. Overhead (3) exists only in the parallel computations.

We now analyze the relationships among the various times defined in Table 7.1. All the times are measured independently. In particular, T_I is obtained by timing two kinds of idle periods on each processor and summing over all processors: one is the spin wait in the panel update pipeline, and the other is when a processor calls `Scheduler()` (line 4 in Figure 6.1) and fails to get a panel from the scheduler. We found that, for the test matrices and the numbers of processors being considered, failure from the scheduler rarely occurs. So most of the idle time is due to pipeline waiting. The following relation holds for the parallel runtime:[5]

$$P\,T_P \approx T_1 + T_I\ . \tag{7.1}$$

We now compute the observed efficiency (E_{actual}) as follows:

$$E_{actual} = \frac{T_s}{P\,T_P}\ . \tag{7.2}$$

Since T_P, T_1, and T_I are obtained from different runs of the program, the left-hand side and the right-hand side of (7.1) may not match well. For the purpose of checking, we also compute the following quantity:

$$E_{check} = \frac{T_s}{T_1 + T_I}\ . \tag{7.3}$$

The closeness of E_{check} to E_{actual} indicates the accuracy of the timings, as shown in Tables 7.4 and 7.5.

In order to understand the impact of the overheads discussed in previous subsections on the parallel efficiency, we introduce two parameters α_1 and α_p, which are calculated based on T_s, T_1, T_P, and T_I as follows:

$$\alpha_1 = \frac{T_1 - T_s}{T_1} = 1 - \frac{T_s}{T_1}\ , \tag{7.4}$$

$$\alpha_p = \frac{T_I}{P\,T_P}\ . \tag{7.5}$$

Both α_1 and α_p are in the range $[0, 1)$; α_1 measures the overhead that degrades the uniprocessor performance, while α_p measures the overhead in the parallel execution. The smaller α_1 and α_2 are, the more efficient is the parallel algorithm. Since

$$(1-\alpha_1)\cdot(1-\alpha_p) = \frac{T_s}{T_1}\cdot\frac{P\,T_P - T_I}{P\,T_P} \approx \frac{T_s}{T_1}\cdot\frac{T_1}{P\,T_P} = E_{actual}\ ,$$

we can use

$$E_{est} = (1-\alpha_1)\cdot(1-\alpha_p) \tag{7.6}$$

as an estimate for the actual efficiency.

In Tables 7.4 and 7.5, we report E_{actual}, E_{check}, E_{est}, α_1, and α_p obtained on two of the parallel machines.

Cray J90. In the first two columns of Table 7.4, we compare the estimated efficiency E_{est} in (7.6) with the actually observed efficiency E_{actual} in (7.2). The estimated and observed efficiencies are very close. Their differences are mostly within 5%, except for matrices 13 and 20, which have 15% and 9% differences. For these two matrices, E_{actual} and E_{check} differ significantly, indicating that some overhead is not reflected in T_1 or T_I.

As mentioned in section 7.2.1, the uniprocessor performance on the J90 does not degrade much with smaller *maxsup*; that is, Overhead (1) does not exist, so $T_s' = T_s$. Therefore, T_s/T_1 can be from the bottom curve in Figure 7.5. We gathered the

[5] In the absence of errors in the individual time measurement, equality should hold.

Table 7.4
Efficiencies and overheads on a 16-*CPU Cray J*90.

		Efficiency			Overhead		Balance
	Matrix	E_{actual}	E_{check}	E_{est}	α_1	α_p	B
13	Shyy161	.47	.59	.63	.17	.23	.66
14	Goodwin	.80	.79	.79	.12	.10	.97
15	Venkat01	.12	.13	.17	.32	.74	.99
16	Inaccura	.46	.47	.48	.10	.46	.97
17	Af23560	.53	.55	.57	.13	.34	.93
18	Dense1000	.25	.26	.30	.07	.67	.99
19	Raefsky3	.53	.56	.58	.07	.37	.96
20	Ex11	.64	.70	.73	.05	.23	.98
21	Wang3	.19	.19	.22	.23	.71	.99
22	Raefsky4	.51	.53	.55	.02	.43	.97

Table 7.5
Efficiencies and overheads on a 12-*CPU Power Challenge.*

		Efficiency			Overhead		Balance
	Matrix	E_{actual}	E_{check}	E_{est}	α_1	α_p	B
13	Shyy161	.42	.54	.58	.27	.20	.70
14	Goodwin	.49	.57	.61	.18	.25	.87
15	Venkat01	.17	.20	.27	.38	.56	.91
16	Inaccura	.42	.45	.47	.21	.40	.88
17	Af23560	.56	.58	.59	.26	.20	.93
18	Dense1000	.35	.35	.34	.18	.58	.92
19	Raefsky3	.58	.62	.63	.25	.16	.95
20	Ex11	.64	.73	.74	.18	.09	.98
21	Wang3	.34	.36	.38	.19	.52	.93
22	Raefsky4	.54	.63	.65	.23	.15	.95
23	Vavasis3	.56	.68	.71	.14	.17	.97

statistics for α_p on 16 processors, as shown in Table 7.4. For most problems, the pipeline spin waiting, as measured by α_p, is the primary cause of inefficiency. This is particularly evident for matrices 15, 18, and 21, for which processors are idle 74%, 67%, and 71% of the time. This explains the low speedups achieved for these matrices.

Power Challenge. On a cache-based machine, the uniprocessor performance loss of the parallel code is a combination of lockings and less efficient cache utilization. Therefore, T_s/T_1 equals the product of the numbers from the bottom curve in Figure 7.2 (T'_s/T_1) and Figure 7.6 (T_s/T'_s). Compared to the J90, we observe that α_1 is much larger, because the cache plays an important role on the Power Challenge. In fact, for matrices 13, 17, 19, 20, and 22, uniprocessor performance loss is more severe than the parallel overhead, α_p.

Again, for matrices 15, 18, and 21, the spin wait time is the major bottleneck; the processors are idle more than 50% of the time. We found that E_{est} and E_{actual} did not match as well as they did on the J90. For matrices 13, 14, and 23, the gaps are 16%, 12%, and 15%. The corresponding gaps between E_{check} and E_{actual} are large as well. This again indicates some overhead not accounted for in T_1 or T_I. We need further study to fully understand this phenomenon.

7.3. Load balance. As mentioned earlier, our dynamic scheduling approach can automatically balance the work load. One way to measure the load balance is as follows. Let f_i denote the number of floating-point operations performed on processor

Table 7.6

Working storage requirement as compared with the storage needed for L and U. The blocking parameter settings are $w = 8$, $t = 100$, and $b = 200$.

	Matrix	LU storage (MB)	Fraction of LU storage $P = 1$	$P = 8$
1	Memplus	16.27	.23	1.51
2	Gemat11	1.15	.89	5.92
3	Rdist1	3.70	.23	1.54
4	Orani678	4.77	.11	.73
5	Mcfe	0.88	.18	1.26
6	Lnsp3937	4.93	.16	1.10
7	Lns_3937	7.04	.12	.77
8	Sherman5	2.75	.25	1.66
9	Jpwh_991	1.58	.13	.88
10	Sherman3	4.68	.22	1.47
11	Orsreg_1	4.23	.11	.72
12	Saylr4	6.98	.10	.70
13	Shyy161	80.01	.19	1.31
14	Goodwin	34.25	.04	.30
15	Venkat01	566.09	.02	.15
16	Inaccura	106.06	.03	.21
17	Af23560	145.02	.03	.22
18	Dense1000	9.90	.02	.14
19	Raefsky3	183.65	.02	.16
20	Ex11	277.59	.01	.08
21	Wang3	459.14	.01	.07
22	Raefsky4	271.28	.02	.10
23	Vavasis3	521.75	.02	.11

i, and let P denote the number of processors. We define the load balance B as

$$B = \frac{\sum_i (f_i)}{P \ \max_i (f_i)} \ . \tag{7.7}$$

In other words, B equals the average work load divided by the maximum work load. It is readily seen that $0 < B \leq 1$, and higher B indicates better load balance. If load imbalance is the sole overhead in a parallel program, the parallel execution time is simply the execution time of the slowest processor, whose work load is highest.

We should note that the load balance measured by (7.7) is an accurate measure of work distribution only under the condition that each floating-point operation takes the same amount of time. This is not the case in practice, but the large values of B shown in the last columns of Tables 7.4 and 7.5 still show that good load balance was achieved in terms of flop counts. Matrix 13 (Shyy161) is an exception.

7.4. Working storage requirement. The parallel algorithm may require more working storage than the sequential one. Multiple threads share heap storage, static storage, and code, all residing in main memory. Each thread, upon execution, is allocated a private stack and has its own register set. Our program does not use many stack variables, so the stack size for each thread need not be very large. All working storage is allocated via `malloc()` from the heap. The working storage consists of two parts, where one part is shared among all threads, and another part is local to each thread. The shared working storage is mainly used to facilitate the central scheduling activity and memory management. It includes

- one integer array of size p used as the task queue, where p is the total number of panels;

- one bit vector of size n to mark whether a column is busy;
- four integer arrays of size n to record the status of each panel;
- one integer array of size n to record a column's most distant busy column down the etree during pipelining;
- three integer arrays of size n to implement the storage layout for supernodes (section 5.3).

The local working storage used by each thread is very similar to that used by sequential SuperLU, that is, all that is necessary to factorize one panel. It includes

- eight integer arrays of size n to perform the panel and column DFS;
- one n-by-w integer array to keep track of the position of the first nonzero of each supernodal segment in U;
- one n-by-w integer array to temporarily store the row subscripts of the nonzeros filled in the panel;
- one n-by-w real array used as the sparse accumulator [18];
- one scratch space of size $(t + b) \times w$ to help BLAS calls. See Figure 2.1 for the definition of t, b, and w.

This amount of local storage should be multiplied by P, where P is the number of threads created. Thus the working storage grows affinely with respect to P, and this algorithm, albeit efficient, is hard to scale up from a memory point of view.

To put this in perspective, Table 7.6 compares the working storage requirement with the actual LU storage. The last two columns report the amount of working storage as a fraction of the total LU storage in megabytes, for 1 and 8 threads. It is clear that for $P = 8$, the working storage requirement can be comparable to the LU storage for small problems. For large problems, working storage is typically 10% to 20% of the LU storage. Matrix 13 (Shyy161) is exceptionally bad: it is a matrix of medium size for which the required working storage is more than LU storage. Since we would not use multiple processors on the small problems anyway, the overall working storage requirement is quite small.

7.5. A PRAM model to predict optimal speedup. Given a matrix with a fixed column ordering, we want to establish a performance model to estimate the maximum speedup attainable by our algorithm and, indeed, to determine the limitations of algorithms based on partitioning a matrix by columns, and using a column as a scheduling unit.

Because of various precedence constraints in the algorithm, some parts of the work must be finished before other parts can start. Thus, the completion time of the parallel algorithm is constrained by the amount of work that must be done *serially*, i.e., the critical path. Our objective here is to give a lower bound on parallel completion time.

In the model we make the following simplifying assumptions: (1) The work includes only floating-point operations, and each floating-point operation takes one unit of time. (2) There is an infinite number of processors. Whenever a task is ready, there will be a free processor to execute this task immediately. (3) Accessing memory and communication are free. (4) We ignore various overheads associated with the actual implementation of the scheduling algorithm and the synchronizations. This model gives an optimistic estimate; therefore, we can use it to prove upper bounds on the performance of the parallel algorithm on a real machine.

The left-looking LU factorization algorithm can be modeled by a data structure called a directed acyclic graph (DAG), in which edges are directed from groups of the etree vertices representing supernodes to groups of the etree vertices representing panels. (Panels and supernodes can overlap in arbitrary fashion.) Each node in

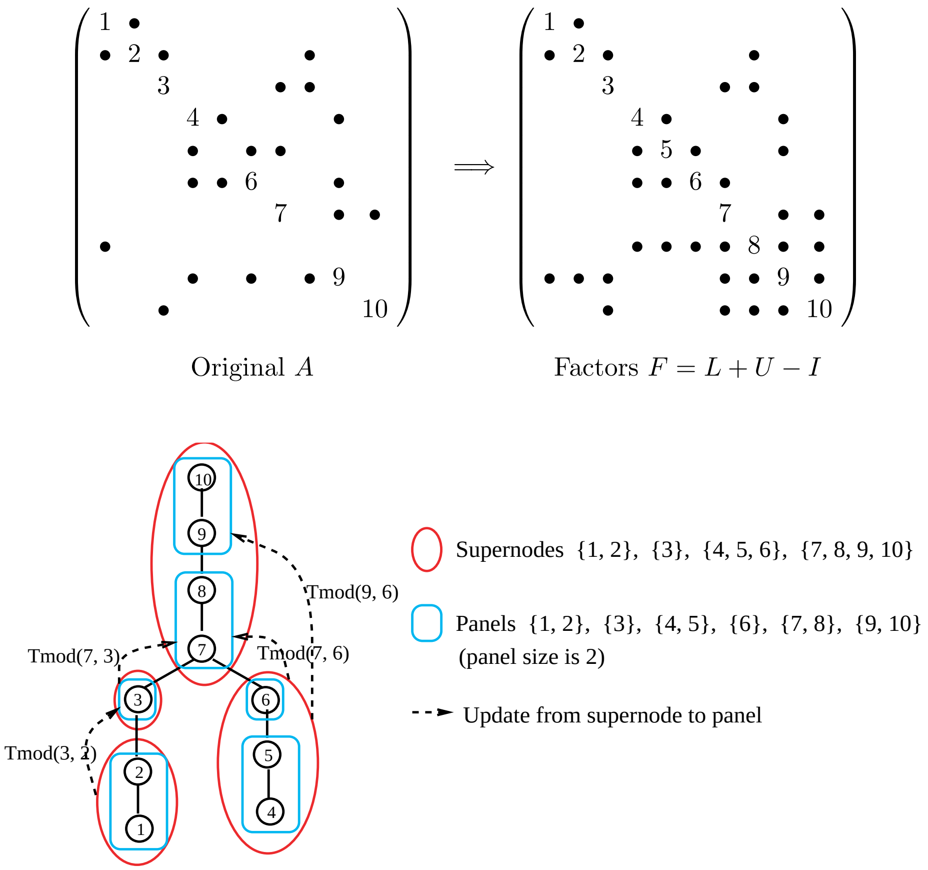

Original A
⟹
Factors $F = L + U - I$
Supernodes {1, 2}, {3}, {4, 5, 6}, {7, 8, 9, 10}
Panels {1, 2}, {3}, {4, 5}, {6}, {7, 8}, {9, 10}
(panel size is 2)
Update from supernode to panel
Tmod(9, 6)
Tmod(7, 3)
Tmod(7, 6)
Tmod(3, 2)

FIG. 7.9. *An example of computational DAG to model the factorization.*

the DAG corresponds to the computation of a panel. An edge directed from s to p corresponds to an update of panel p by supernode s. The edges also represent precedence relations between the updating supernodes and the destination panels. Figure 7.9 illustrates such a DAG for a 10-by-10 matrix.

In presenting our model, we employ the following notation:

- $T_{mod}(p, d) :=$ the task of updating panel p by a descendant supernode d;
- $T_{div}(p) :=$ the task of performing the inner factorization of panel p;
- $tmod(p, d) :=$ time taken by task $T_{mod}(p, d)$;
- $tdiv(p) :=$ time taken by task $T_{div}(p)$;
- $EST(p) :=$ earliest possible starting time of $T_{div}(p)$;
- $EFT(p) :=$ earliest possible finishing time of $T_{div}(p)$.

All times are expressed in units of floating-point operations. It is clear that for any panel p the following relation holds: $EFT(p) = EST(p) + tdiv(p)$.

According to our scheduling algorithm, each panel task is assigned to a single processor. A panel task for panel p consists of the following two types of subtasks:

$$T_{panel}(p) := \{T_{mod}(p, d) \mid d \in \mathcal{D}\} \cup \{T_{div}(p)\} ,$$

where $\mathcal{D}$ is the set of descendant supernodes that update the destination panel p.

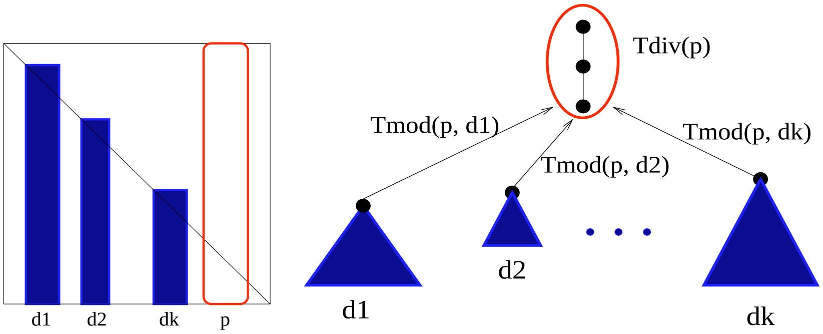


FIG. 7.10. *Tasks associated with panel p.*

Figure 7.10 shows the part of the DAG associated with a particular panel p.

Each T_{mod} and T_{div} is an indivisible task and is carried out sequentially on one processor. Clearly, T_{div} cannot start until all the T_{mod}'s have finished. By looking at the precedence relations of these two types of tasks, we can determine the runtime of $T_{panel}(p)$ on processor P. We will try to schedule these tasks as early as possible, in order to derive the minimum parallel execution time.

We first look at the tasks associated with one particular panel p, as shown in Figure 7.10. Suppose there are k descendant supernodes to update panel p, and that all the times $\{EFT(d), d \in \mathcal{D}\}$ have been computed. We schedule the tasks $\{T_{mod}(p,d), d \in \mathcal{D}\}$ to processor P in the order of $T_{mod}(p,1), \ldots, T_{mod}(p,k)$, such that

$$EFT(1) \leq EFT(2) \leq \cdots \leq EFT(k) .$$

Here, $EFT(i)$ is the finishing time of the last column of supernode i, because a supernode i cannot update any ancestor panel before its last column is completed. We call this scheduling policy Sched-A. Then we can compute $EST(p)$ and $EFT(p)$ as follows.

1. Run the following to get the completion times of the T_{mod}'s:

```
    $t = 0$;
    for $i = 1$ to $k$
        $t = \max\ \{\ t,\ EFT(i)\ \} + tmod(i)$;
    endfor;
```

2. Set $EST(p) = t$ and $EFT(p) = t + tdiv(p)$.

Now we will give an informal argument for the optimality of the parallel runtime resulting from Sched-A.

THEOREM 7.1. *For panel p, scheduling the T_{mod}'s by Sched-A gives the shortest completion time.*

Proof. Processor P requires at least $\sum_{i=1}^{k} tmod(p,i)$ units of time to finish all the updates to panel p. Now suppose another scheduling strategy Sched-B starts with a task $T_{mod}(p,i), i \neq 1$. Due to the precedence constraint, $T_{mod}(p,i)$ cannot start until after time $EFT(i)$ ($\geq EFT(1)$). That means processor P will be idle during the period of $LAG := EFT(i) - EFT(1)$. Thus the amount of time to finish all the T_{mod}'s will be at least $LAG + \sum_{i=1}^{k} tmod(p,i)$.

On the other hand, in Sched-A, at least some $T_{mod}(p,j), j < i$ have been scheduled in the time period LAG. Hence the amount of work left after time $EFT(i)$ is less

Table 7.7
Optimal speedup predicted by the model, and the column etree height.

		$maxsup = 32$			$maxsup = 64$			
	Matrix	$w = 4$	$w = 8$	$w = 16$	$w = 4$	$w = 8$	$w = 16$	$height/n$
1	Memplus	4.8	3.6	2.8	2.9	2.5	2.1	0.95
2	Gemat11	7.3	5.3	4.1	6.4	4.9	3.6	0.06
3	Rdist1	4.6	3.2	2.1	4.6	3.2	2.1	0.99
4	Orani678	42.2	28.4	16.6	42.2	28.4	16.6	0.64
5	Mcfe	6.6	4.3	2.6	6.6	4.3	2.6	0.67
6	Lnsp3937	23.2	15.4	9.7	23.2	15.4	9.7	0.25
7	Lns_3937	24.1	15.8	9.6	22.9	15.3	9.6	0.27
8	Sherman5	15.8	11.4	7.5	14.0	10.7	7.2	0.20
9	Jpwh_991	13.4	9.7	6.4	11.3	8.3	6.0	0.46
10	Sherman3	12.7	9.7	7.0	8.2	6.9	5.5	0.20
11	Orsreg_1	14.4	11.0	7.5	9.2	7.8	5.9	0.34
12	Saylr4	19.8	16.1	11.0	13.1	11.4	8.6	0.29
13	Shyy161	47.9	36.2	24.1	28.1	23.8	18.1	0.04
14	Goodwin	97.4	71.3	43.6	83.4	63.4	40.1	0.19
15	Venkat01	22.0	20.2	17.0	14.3	14.2	13.1	0.73
16	Inaccura	62.6	43.5	26.0	44.5	33.6	22.2	0.45
17	Af23560	70.9	55.3	37.2	41.4	35.7	27.4	0.20
18	Dense1000	33.1	23.7	18.4	18.2	14.9	12.7	1.00
19	Raefsky3	140.2	110.6	80.8	80.4	69.6	56.5	0.21
20	Ex11	106.7	83.5	58.2	61.6	53.2	41.7	0.35
21	Wang3	57.6	43.4	29.4	34.3	28.9	22.1	0.94
22	Raefsky4	99.1	77.1	52.0	56.3	48.5	37.3	0.33
23	Vavasis3	176.5	133.9	90.7	106.2	89.5	68.2	0.18

than the work left when using Sched-B. Sched-A will have an earlier finishing time than Sched-B. □

We are now ready to simulate parallel computation for the whole factorization. To begin with, the ESTs of the leaf panels in the column etree are initialized to zero. Various times can be computed successively from the bottom of the etree to the top. By applying the argument above inductively to all the panels in the DAG, with leaf panels as the basis, we can show that EFT(root panel) gives the minimum execution time. The (predicted) optimal speedup can then be computed by

$$\text{Predicted speedup} = \frac{\text{Total flops}}{EFT(\text{root panel})} .$$

There are several points worth noting in this model. First, because of numerical pivoting, we do not know the computational DAG in advance of the factorization; rather, the DAG is built incrementally as the factorization proceeds. Also, the floating-point operations associated with all the tasks are calculated on the fly. So this model gives an a posteriori estimate. Second, for each panel computation, the scheduling method of Sched-A requires sorting the EFT's of all the descendant supernodes that will update this panel. The cost associated with this sorting is prohibitively high, and so this method cannot be used to schedule panel updates in practice. Nevertheless, this gives us an upper bound on the theoretically attainable speedup.

In our algorithm, two parameters control task granularity: the panel size w determines the amount of work in a T_{div} task, and both w and the maximum supernode size $maxsup$ determine the amount of work in a T_{mod} task. Any large supernode of size exceeding $maxsup$ (such as in a dense matrix) is divided into smaller ones so that they fit into a cache.

Table 7.7 reports the predicted speedups when varying w and $maxsup$. For a fixed value of $maxsup$, the simulated speedups decrease with increasing w. For sequential

SuperLU we find empirically that the best choice for w is between 8 and 16, depending on matrices and architectures. In the parallel setting, a smaller w—say, between 4 and 8—seems to give the best overall performance. This embodies an interesting trade-off between available concurrency and per-processor efficiency.

We now compare the results when fixing w but varying *maxsup*. In sparser matrices, such as matrices 1–10, the actual sizes of supernodes may be much smaller than *maxsup*. The performance for such matrices are not so sensitive to *maxsup*. However, for larger and denser matrices, a larger value of *maxsup* results in poorer speedup.

Finally we note that the speedups for small matrices are very low, even with small values of w and *maxsup*. Fortunately, for large matrices such as 13–21, the predicted speedups are greater than 20 when $w = 8$ and $maxsup = 32$. These matrices perform more than one billion floating-point operations in the factorization. It is these matrices that require parallel processing power. The current column-oriented algorithm is well suited for most of the commercially popular SMPs, because the number of processors on these systems is usually below 20.

The height of the column etree can also be used as a crude prediction of the parallel performance. The height of a node i is defined as

$$height(i) = \begin{cases} 0 & \text{if } i \text{ is a leaf node,} \\ 1 + \max\{\, height(j) \mid j \in child(i)\} & \text{otherwise.} \end{cases}$$

The height of the etree is the height of the root, which represents the longest path in the etree. The computation of all the nodes along this path must be performed in succession. Therefore, the length of the critical path constrains performance. The last column of Table 7.7 shows the height of the etree over total numbers of nodes n in the etree. The larger $height/n$ is, the larger the fraction of panels will be factorized in pipelined manner, resulting in poor parallelism and more synchronizations. For example, $height/n$ for matrices 1, 3, 15, and 21 is rather large. This is consistent with the lower predicted speedups. However, we must note that the etree height alone is not an accurate measure of parallelism. For example, both dense matrix (18) and a tridiagonal matrix have $height/n = 1.00$, but the former possesses much more concurrency than the later.

The actual speedups achieved are much lower than the upper bounds predicted by the PRAM model (Figures 7.1 through 7.5). This is because the model does not capture the details of the machines and the implementation, such as cache behavior, synchronization, etc. However, we do see a similar shape of speedup curves. For example, the model predicts that matrices 15 and 21 have lower speedups compared with the other large matrices. In reality, these two matrices perform worse than the others. The poor performance is primarily due to two factors: (1) The column etree is tall and contains substantial false dependencies. (2) The dynamic algorithm is needed to allocate memory for the supernodes (section 5.3), because the static upper bound on supernode storage is too large for these two problems (Table 5.2).

8. Conclusions.

We have designed and implemented a parallel algorithm for shared memory multiprocessors of modest size. The efficiency of the algorithm has been demonstrated on several parallel machines. Figure 8.1 shows the speedups on 8 processors of three parallel machines. Figures 8.2 through 8.5 summarize the factorization rate in Megaflops for six large matrices, with increasing numbers of processors. We believe these large problems are the primary candidates to be solved on parallel machines. In fact, the largest one in our test suite takes a little more than 0.5 GBytes

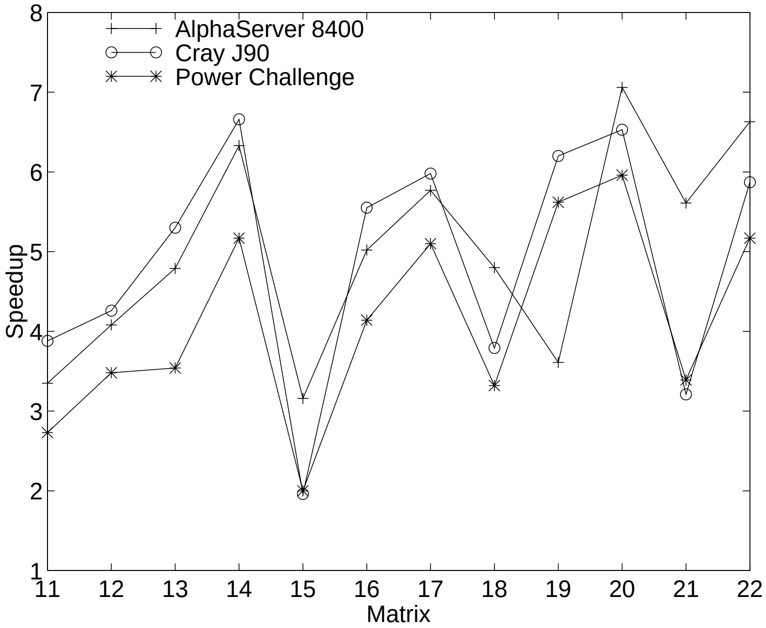


FIG. 8.1. *Speedups on* 8 *processors of the Power Challenge, the AlphaServer* 8400*, and the Cray J*90.

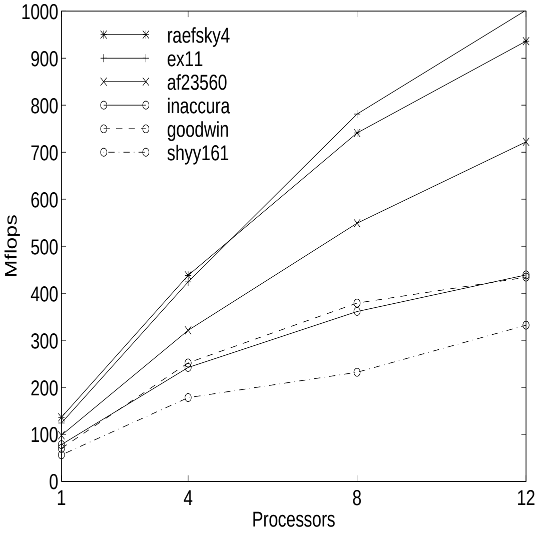


FIG. 8.2. *Mflop rate on an SGI Power Challenge.*

memory, far less than most parallel machines offer. Our algorithm is expected to work well for even larger problems.

For a realistic problem arising from a 3-D flow calculation (matrix 20, Ex11), on the 12-CPU Power Challenge, the 8-CPU Cray C90, the 16-CPU J90, and the 8-CPU AlphaServer 8400, our parallel algorithm achieves 23%, 33%, 25%, and 17% peak floating-point performance. The respective Mflop rates are 1002, 2583, 831, and 781.

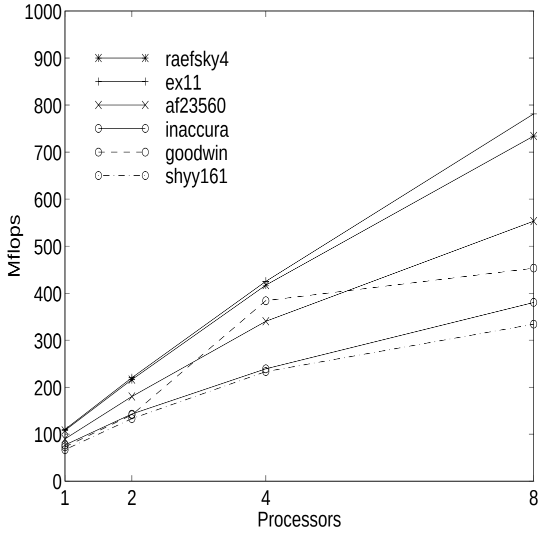


FIG. 8.3. *Mflop rate on a DEC AlphaServer* 8400.

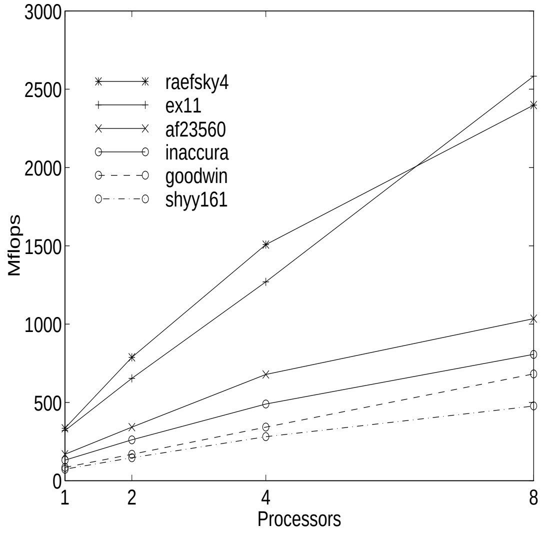


FIG. 8.4. *Mflop rate on a Cray C*90.

These are the fastest results for the unsymmetric *LU* factorization on these powerful high-performance machines. Previous results showed much lower factorization rates because the machines used were relatively slow and the computational kernel in the earlier parallel algorithms was based on Level 1 BLAS. The closest work is the parallel symmetric pattern multifrontal factorization by Amestoy and Duff [1], also on SMPs. However, that approach may result in too many nonzeros and so may be inefficient for unsymmetric pattern sparse matrices.

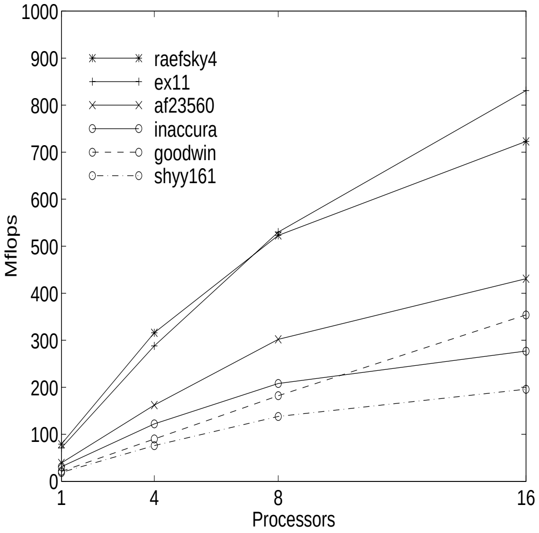


FIG. 8.5. *Mflop rate on a Cray J90.*

Another contribution is to provide detailed performance analysis and modeling for the underlying algorithm. In particular, we identified the three main factors limiting parallel performance: (1) contention for accessing critical sections, (2) processors sitting idle due to pipeline waiting, and (3) the need to sacrifice some per-processor efficiency in order to gain more concurrency. Which factor plays the most significant role depends on the relative performance of integer and floating-point arithmetic in the underlying architecture.

We have developed a theoretical model to analyze our parallel algorithm and predict the optimally attainable speedup. When comparing the theoretical prediction (Table 7.7) with the actual speedups (Figure 8.1), we find that there exists a discrepancy between the two. This is because our hypothetical machine and the optimal scheduling used in the model do not capture all the details of a real machine with real scheduling. Nevertheless, we do see a similar behavior in the predicted and actual speedups. That is, for the matrices with lower predicted speedups, such as 11, 15, 18, and 21, the actual speedups are also lower. The model is a useful tool to help identify the inherently sequential problems with bad column orderings. The model also suggests that the panel-wise parallel algorithm, although efficient on small scale SMPs, cannot effectively utilize more than 50 processors.

We plan to expand this research in several directions. We will study a more scalable algorithm for larger parallel machines. That algorithm will likely partition the matrix by both rows and columns, and schedule blocks of submatrices onto processors. This will potentially increase parallelism, and reduce the panel update pipeline waiting time. In the framework of SuperLU, both serial and parallel, we will investigate incomplete *LU* factorizations, which can be used as a class of preconditioners for unsymmetric sparse iterative solvers.

Appendix A. Performance of the parallel algorithm.

A.1. On the Sun SPARCcenter 2000.

Table A.1
Speedup, factorization time, and Mflop rate on a 4*-CPU SPARCcenter* 2000.

	Matrix	$P = 1$	$P = 2$	$P = 4$	Seconds	Mflops
1	Memplus	0.44	0.82	0.74	2.35	1
2	Gemat11	0.77	1.25	1.51	0.47	3
3	Rdist1	0.86	1.92	1.82	1.71	8
4	Orani678	0.71	1.24	2.08	1.98	8
5	Mcfe	0.79	1.38	2.00	0.45	9
6	Lnsp3937	0.96	1.85	2.03	2.26	18
7	Lns3937	0.92	1.73	3.09	2.41	19
8	Sherman5	0.83	1.70	2.81	1.26	20
9	Jpwh991	0.77	1.56	2.77	0.84	22
10	Sherman3	0.90	1.74	2.92	2.77	22
11	Orsreg1	0.89	1.75	3.17	2.27	27
12	Saylr4	0.88	1.76	3.10	4.17	25
13	Shyy161	0.90	1.82	3.25	59.55	26
15	Goodwin	0.92	1.86	3.61	20.50	33
18	Dense1000	0.97	1.96	3.64	16.39	41
	Mean speedup	0.83	1.62	2.64		
	Std deviation	0.13	0.32	0.83		

A.2. On the SGI Power Challenge.

Table A.2
Speedup, factorization time, and Mflop rate on a 12*-CPU SGI Power Challenge.*

	Matrix	$P = 1$	$P = 4$	$P = 8$	$P = 12$	Seconds	Mflops
1	Memplus	0.72	1.73	1.73	1.69	0.42	4
2	Gemat11	0.89	1.86	2.36	3.71	0.07	22
3	Rdist1	0.89	1.66	1.56	2.23	0.44	32
4	Orani678	0.68	1.72	2.40	2.56	0.45	33
5	Mcfe	0.68	1.92	2.09	3.29	0.07	59
6	Lnsp3937	0.97	3.00	3.65	3.86	0.35	122
7	Lns3937	0.98	2.98	3.92	3.73	0.40	117
8	Sherman5	0.86	2.29	3.09	3.09	0.23	111
9	Jpwh991	0.83	2.40	3.43	5.33	0.09	205
10	Sherman3	0.87	2.36	2.78	2.78	0.40	157
11	Orsreg1	0.88	2.67	2.73	2.97	0.34	180
12	Saylr4	0.90	2.81	3.48	4.58	0.38	284
13	Shyy161	0.86	2.71	3.54	5.06	4.64	332
14	Goodwin	0.89	3.45	5.17	5.90	1.56	433
15	Venkat01	0.65	1.72	2.00	1.98	15.37	209
16	Inaccura	0.85	2.77	4.14	5.00	9.53	438
17	Af23560	0.91	2.98	5.10	6.70	8.87	722
18	Dense1000	0.85	2.64	3.32	4.17	0.90	740
19	Raefsky3	0.92	3.07	5.62	6.91	11.35	1070
20	Ex11	0.94	3.23	5.96	7.64	26.95	1046
21	Wang3	0.85	2.20	3.39	4.03	21.37	681
22	Raefsky4	0.94	3.05	5.17	6.52	33.57	936
23	Vavasis3	0.91	3.58	6.06	6.69	105.06	862
	Mean speedup	0.86	2.56	3.59	4.37		
	Std deviation	0.09	0.59	1.36	1.73		

A.3. On the DEC AlphaServer 8400.

Table A.3
Speedup, factorization time, and Mflop rate on an 8*-CPU DEC AlphaServer* 8400.

	Matrix	$P = 1$	$P = 2$	$P = 4$	$P = 6$	$P = 8$	Seconds	Mflops
1	Memplus	0.46	0.79	0.79	0.78	0.64	0.59	3
2	Gemat11	0.83	1.63	1.88	1.88	1.88	0.08	20
3	Rdist1	0.90	1.98	2.10	1.77	1.77	0.31	40
4	Orani678	0.83	1.29	2.00	2.33	2.42	0.26	57
5	Mcfe	0.72	1.80	3.00	2.17	2.17	0.06	66
6	Lnsp3937	0.93	1.94	3.19	3.68	3.68	0.25	159
7	Lns3937	0.95	1.83	3.08	3.81	4.12	0.25	187
8	Sherman5	0.91	1.89	2.89	2.94	2.94	0.17	151
9	Jpwh991	0.92	1.89	3.00	3.30	3.00	0.11	178
10	Sherman3	0.88	1.83	2.72	2.74	2.74	0.34	180
11	Orsreg1	0.93	1.88	2.93	3.35	3.35	0.26	231
12	Saylr4	0.91	1.98	3.20	3.78	4.08	0.38	276
13	Shyy161	0.95	1.93	3.23	4.21	4.79	4.66	334
14	Goodwin	0.99	1.98	3.68	5.39	6.33	1.49	453
15	Venkat01	0.89	1.92	2.95	3.04	3.16	10.62	303
16	Inaccura	0.99	1.83	3.08	4.15	5.02	10.94	380
17	Af23560	0.95	1.98	3.72	5.03	5.77	11.58	553
18	Dense1000	0.98	1.86	3.35	4.32	4.80	0.99	675
19	Raefsky3	0.98	1.98	3.81	3.16	3.61	28.65	422
20	Ex11	0.99	1.98	3.76	5.56	7.06	34.23	781
21	Wang3	0.93	1.98	3.69	4.75	5.61	21.36	682
22	Raefsky4	0.98	1.98	3.81	5.44	6.63	42.79	734
23	Vavasis3	0.96	1.97	3.69	5.28	6.64	124.24	724
	Mean speedup	0.92	1.74	2.89	3.59	4.01		
	Std deviation	0.13	0.28	0.81	1.31	1.77		

A.4. On the Cray C90.

Table A.4
Speedup, factorization time, and Mflop rate on an 8*-CPU Cray C*90.

	Matrix	$P = 1$	$P = 2$	$P = 4$	$P = 6$	$P = 8$	Seconds	Mflops
1	Memplus	0.66	0.75	0.74	0.72	0.71	1.24	2
2	Gemat11	0.76	1.36	2.27	3.09	3.40	0.10	15
3	Rdist1	0.71	1.98	2.41	2.41	2.31	0.48	34
4	Orani678	0.72	1.24	2.22	2.91	3.20	0.41	37
5	Mcfe	0.69	1.25	1.82	2.00	2.00	0.10	43
6	Lnsp3937	0.78	1.51	2.77	2.84	4.41	0.27	151
7	Lns3937	0.78	1.51	2.95	3.97	4.23	0.30	156
8	Sherman5	0.77	1.49	2.90	3.59	4.07	0.15	170
9	Jpwh991	0.78	1.52	2.50	3.18	2.92	0.12	164
10	Sherman3	0.79	1.48	2.53	2.97	2.97	0.29	214
11	Orsreg1	0.80	1.53	2.69	3.25	3.55	0.22	278
12	Saylr4	0.83	1.58	3.05	3.85	3.97	0.33	318
13	Shyy161	0.80	1.50	2.87	3.87	4.86	3.29	477
14	Goodwin	0.84	1.65	3.31	4.83	6.59	0.99	682
15	Venkat01	0.70	1.28	1.65	1.73	1.74	14.04	229
16	Inaccura	0.86	1.70	3.19	4.38	5.21	5.18	807
17	Af23560	0.84	1.63	3.22	4.56	4.89	6.24	1035
18	Dense1000	0.95	1.86	2.95	3.30	3.55	0.71	943
19	Raefsky3	0.91	1.74	3.45	4.77	5.83	6.17	1977
20	Ex11	0.90	1.65	3.21	5.02	6.53	10.37	2583
21	Wang3	0.78	1.48	1.82	2.31	2.32	14.62	996
22	Raefsky4	0.92	1.80	3.43	4.60	5.46	13.13	2399
	Mean speedup	0.80	1.53	2.63	3.42	3.85		
	Std deviation	0.08	0.27	0.67	1.11	1.55		

A.5. On the Cray J90.

Table A.5
Speedup, factorization time, and Mflop rate on a 16-*CPU Cray J*90.

	Matrix	$P = 1$	$P = 4$	$P = 8$	$P = 12$	$P = 16$	Seconds	Mflops
1	Memplus	0.65	0.94	0.98	0.97	0.76	3.67	1
2	Gemat11	0.71	2.44	4.38	5.25	5.83	0.18	8
3	Rdist1	0.71	2.86	2.88	2.71	2.39	1.53	10
4	Orani678	0.71	2.07	3.11	3.82	3.85	1.13	13
5	Mcfe	0.77	2.21	2.70	2.70	2.52	0.29	15
6	Lnsp3937	0.75	2.87	4.91	6.21	6.39	0.66	62
7	Lns3937	0.79	2.75	4.63	5.41	5.41	0.83	58
8	Sherman5	0.80	2.91	4.64	5.07	5.32	0.41	63
9	Jpwh991	0.78	2.72	3.57	3.68	3.38	0.37	49
10	Sherman3	0.80	2.63	3.49	3.42	3.31	0.96	66
11	Orsreg1	0.83	2.83	3.88	4.22	4.16	0.70	89
12	Saylr4	0.81	2.91	4.26	4.82	4.82	0.99	108
13	Shyy161	0.83	2.92	5.30	6.94	7.47	8.06	196
14	Goodwin	0.88	3.32	6.66	10.02	12.81	1.94	354
15	Venkat01	0.68	1.84	1.96	1.98	1.90	47.34	68
16	Inaccura	0.90	3.26	5.55	6.64	7.39	15.09	277
17	Af23560	0.87	3.22	5.98	7.55	8.49	15.05	431
18	Dense1000	0.93	2.84	3.79	3.92	3.91	2.61	256
19	Raefsky3	0.93	3.38	6.20	7.69	8.43	19.03	641
20	Ex11	0.95	3.56	6.53	9.47	10.17	32.48	831
21	Wang3	0.77	2.53	3.21	3.14	3.06	50.42	288
22	Raefsky4	0.98	3.54	5.87	7.36	8.12	43.54	723
	Mean speedup	0.81	2.75	4.29	5.13	5.45		
	Std deviation	0.09	0.60	1.51	2.38	2.97		

Acknowledgments. Ed Rothberg not only provided us access to the SGI Power Challenge, but also helped improve performance of our algorithm. We thank Esmond Ng for correspondences on the issues of nonzero structure prediction, which helped design the memory management scheme discussed in section 5.3. We thank Kathy Yelick for suggestions on the presentation of the performance analysis section. We thank Osni Marques, Peter Tang, and the anonymous referees for their suggestions on improving the presentation of the material.

REFERENCES

[1] P. R. Amestoy and I. S. Duff, *MUPS: A Parallel Package for Solving Sparse Unsymmetric Sets of Linear Equations*, Technical report, CERFACS, Toulouse, France, 1994.

[2] P. R. Amestoy, *Factorization of Large Unsymmetric Sparse Matrices Based on a Multifrontal Approach in a Multiprocessor Environment*, Technical report TH/PA/91/2 and Ph.D. thesis, CERFACS, Toulouse, France, February 1991.

[3] C. Ashcraft and R. Grimes, *The influence of relaxed supernode partitions on the multifrontal method*, ACM Trans. Math. Software, 15 (1989), pp. 291–309.

[4] J. Bilmes, K. Asanovic, J. Demmel, D. Lam, and C.-W. Chin, *Optimizing Matrix Multiply Using PHiPAC: A Portable, High-performance, ANSI C Coding Methodology*, Technical report CS-96-326, Computer Science Dept., University of Tennessee, Knoxville, TN, May 1996 (LAPACK Working Note 111).

[5] T. A. Davis and P.-C. Yew, *A nondeterministic parallel algorithm for general unsymmetric sparse LU factorization*, SIAM J. Matrix Anal. Appl., 11 (1990), pp. 383–402.

[6] J. W. Demmel, S. C. Eisenstat, J. R. Gilbert, X. S. Li, and J. W. H. Liu, *A supernodal approach to sparse partial pivoting*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 720–755.

[7] J. Dongarra, J. Du Croz, S. Hammarling, and R. J. Hanson, *An extended set of FORTRAN basic linear algebra subprograms*, ACM Trans. Math. Software, 14 (1988), pp. 1–17.

[8] J. Dongarra, J. Du Croz, I. S. Duff, and S. Hammarling, *A set of level* 3 *basic linear algebra subprograms*, ACM Trans. Math. Software, 16 (1990), pp. 1–17.

[9] I. S. Duff, R. Grimes, and J. Lewis, *Sparse matrix test problems*, ACM Trans. Math. Software, 15 (1989), pp. 1–14.

[10] S. C. Eisenstat and J. W. H. Liu, *Exploiting structural symmetry in unsymmetric sparse symbolic factorization*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 202–211.

[11] S. C. Eisenstat and J. W.H. Liu, *Exploiting structural symmetry in a sparse partial pivoting code*, SIAM J. Sci. Comput., 14 (1993), pp. 253–257.

[12] D. M. Fenwick, D. J. Foley, W. B. Gist, S. R. VanDoren, and D. Wissel, *The AlphaServer* 8000 *series: High-end server platform development*, Digital Tech. J., 7 (1995), pp. 43–65.

[13] A. George, M. T. Heath, J. Liu, and E. Ng, *Solution of sparse positive definitive systems on a shared-memory multiprocessor*, Internat. J. Parallel Programming, 15 (1986), pp. 309–325.

[14] A. George, J. Liu, and E. Ng, *A data structure for sparse QR and LU factorizations*, SIAM J. Sci. Stat. Comput., 9 (1988), pp. 100–121.

[15] A. George and E. Ng, *An implementation of Gaussian elimination with partial pivoting for sparse systems*, SIAM J. Sci. Stat. Comput., 6 (1985), pp. 390–409.

[16] A. George and E. Ng, *Symbolic factorization for sparse Gaussian elimination with partial pivoting*, SIAM J. Sci. Stat. Comput., 8 (1987), pp. 877–898.

[17] A. George and E. Ng, *Parallel sparse Gaussian elimination with partial pivoting*, Ann. Oper. Res., 22 (1990), pp. 219–240.

[18] J. R. Gilbert, C. Moler, and R. Schreiber, *Sparse matrices in Matlab: Design and implementation*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 333–356.

[19] J. R. Gilbert and E. G. Ng, *Predicting structure in nonsymmetric sparse matrix factorizations*, in Graph Theory and Sparse Matrix Computation, A. George, J. R. Gilbert, and J. W. H. Liu, eds., Springer-Verlag, New York, 1993, pp. 107–139.

[20] J. R. Gilbert, *An Efficient Parallel Sparse Partial Pivoting Algorithm*, Technical report CMI 88/45052-1, Christian Michelsen Institute, Bergen, Norway, 1988.

[21] J. R. Gilbert, E. G. Ng, and B. W. Peyton, *Computing row and column counts for sparse QR factorization*, talk presented at Fifth SIAM Conference on Applied Linear Algebra, Snowbird, UT, June 1994; journal version in preparation.

[22] J. R. Gilbert, E. G. Ng, and B. W. Peyton, *An efficient algorithm to compute row and column counts for sparse Cholesky factorization*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 1075–1091.

[23] A. Gupta and V. Kumar, *Optimally scalable parallel sparse Cholesky factorization*, in Proc. 7th SIAM Conference on Parallel Processing for Scientific Computing, SIAM, Philadelphia, PA, 1995, pp. 442–447.

[24] A. Gupta, E. Rothberg, E. Ng, and B. W. Peyton, *Parallel sparse Cholesky factorization algorithms for shared-memory multiprocessor systems*, in Advances in Computer Methods for Partial Differential Equations–VII, R. Vichnevetsky, D. Knight, and G. Richter, eds., IMACS, New Brunswick, NJ, 1992, pp. 622–628.

[25] X. S. Li, *Sparse Gaussian Elimination on High Performance Computers*, Technical report UCB//CSD-96-919 and Ph.D. thesis, Computer Science Division, University of California, Berkeley, CA, September 1996.

[26] J. W. H. Liu, *The role of elimination trees in sparse factorization*, SIAM J. Matrix Anal. Appl., 11 (1990), pp. 134–172.

[27] J. W. H. Liu, E. G. Ng, and B. W. Peyton, *On finding supernodes for sparse matrix computations*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 242–252.

[28] E. Ng and B. W. Peyton, *A supernodal Cholesky factorization algorithm for shared-memory multiprocessors*, SIAM J. Sci. Comput., 14 (1993), pp. 761–769.

[29] E. Rothberg, *Performance of panel and block approaches to sparse Cholesky factorization on the iPSC*/860 *and Paragon multicomputers*, SIAM J. Sci. Comput., 17 (1996), pp. 699–713.

[30] *SGI Power Challenge*, Technical report, Silicon Graphics, Mountain View, CA, 1995.

[31] *SPARCcenter* 2000 *architecture and implementation*, Technical white paper, Sun Microsystems, Inc., Mountain View, CA, November 1993.

[32] S. A. Vavasis, *Stable finite elements for problems with wild coefficients*, SIAM J. Numer. Anal., 33 (1996), pp. 809–916.

[33] *The Cray C*90 *series*, Cray Research, Inc., Eagen, MN, http://www.cray.com/swpubs/

[34] *The Cray J*90 *series*, Cray Research, Inc., Eagen, MN, http://www.cray.com/swpubs/

SIAM J. Matrix Anal. Appl.
Vol. 20, No. 4, pp. 953–969

AN OBJECT-ORIENTED APPROACH TO THE DESIGN OF A USER INTERFACE FOR A SPARSE MATRIX PACKAGE*

ALAN GEORGE[†] AND JOSEPH LIU[‡]

Abstract. The authors designed and implemented a sparse matrix package called Sparspak in the late 1970s. One of the important features of that package is an interface which shields the user from the complicated calling sequences common to most sparse matrix software. The implementation of the package was challenging because the relatively primitive but widely available Fortran 66 language was used. Modern programming languages such as Fortran-90 and C++ have important features which facilitate the design of flexible and "user-friendly" interfaces for software packages. These features include dynamic storage allocation, function name overloading, user-defined data types, and the ability to hide functions and data from the user. This article describes the redesign of the Sparspak user interface using Fortran-90 and C++, outlining the reasons for its various features and highlighting similarities and differences in the features and capabilities of the two languages. The two new implementations of Sparspak have been named Sparspak-90 and Sparspak++.

Key words. sparse matrix software, object-oriented numerical software, user interfaces

AMS subject classifications. 65F10, 65F50

PII. S0895479897317739

1. Introduction.

Sparspak is a sparse matrix package that was designed and implemented by the authors in the late 1970s.[1] One of the important features of the package is a user interface which shields the user from the complicated calling sequences common to most sparse matrix software available at that time. A description of the interface together with motivation for its design can be found in [4].

This paper describes the design and some general implementation issues of the new version of Sparspak. The software is being implemented in C++ and Fortran-90, and the two implementations are referred to as Sparspak++ and Sparspak-90 throughout this paper. An "object-oriented" approach to the design has been adopted, reflecting widely accepted software engineering practice [1, 8].

Software for solving sparse systems of equations involves relatively complicated data structures which are not provided as standard data types in the language in which the software is implemented. Solving a sparse system of equations usually consists of a number of individual steps typically involving different types of data structures; sometimes data used at one step can be discarded at the end of it and the storage released for later use. Sometimes the amount of storage required is not known until at least some of the computation has been completed.

When Sparspak was implemented, Fortran 66 and its successor, Fortran 77, were used almost exclusively for scientific computing. Neither language provides user-defined data types nor dynamic storage allocation. Consequently, the subroutines and functions implementing sparse matrix software tend to have long argument lists,

*Received by the editors February 27, 1997; accepted for publication (in revised form) by E. Ng January 30, 1998; published electronically July 9, 1999. This work was supported by Natural Sciences and Engineering Research Council of Canada grants OGP 000811 and OGP 005509.

http://www.siam.org/journals/simax/20-4/31773.html

[†] Department of Computer Science, University of Waterloo, Waterloo, ON, N2L 3G1, Canada (jageorge@sparse.uwaterloo.ca).

[‡] Department of Computer Science, York University, Toronto, ON, M3J 1P3, Canada (joseph@cs.yorku.ca).

[1] Dr. Esmond Ng, currently at the Mathematical Sciences Section, Oak Ridge National Laboratory, collaborated with the authors on a second release of the package in 1984.

most of which have little or no relevance for the user of the software. One of the main roles of the interface is to provide the user with simple subroutines whose argument lists, to the extent possible, include only information specific to the problem; that is, information that the user would inevitably know. The interface provides for the allocation of storage and creation of data structures using the more primitive data types available in the language. In addition, the interface ensures that its subroutines are called in the correct order, provides uniform error message handling, and collects timing and storage statistics.

There are several motivations for revisiting the interface design. First, Sparspak has been in use for more than 15 years, and the users of the package have provided useful feedback concerning both strong points and shortcomings of the interface. In addition, Fortran has evolved substantially, so that the current standard, Fortran-90 [7], contains modern programming language features that make implementation of a user interface far more convenient and effective. Collaterally, Fortran is no longer the exclusive choice of scientific programmers; C [6] and C++ [11] are becoming increasingly popular for implementing scientific software. Both of these languages have features that are desirable in connection with implementing an interface for a sparse matrix package. Finally, experience suggests that the package should cater to a broader audience. When Sparspak was designed, it was assumed that its users would be primarily engineers and scientists whose main interest would be solving sparse problems arising in their applications. That is, it would serve as a utility in various scientific applications and application packages. However, the package also has been used extensively by researchers in sparse matrix computation. Thus, there is a motivation to redesign it so that it can also serve as a research test bed for experts in the field of sparse matrix computation. A corollary of this objective is that the interface should be flexible enough to support techniques for dealing with a wide spectrum of sparse matrix problems. The original package was restricted to positive definite problems, and to least squares problems which could be solved using techniques largely adapted from those used for positive definite problems. This restriction simplified the design of the user interface.

An outline of the paper follows. Section 2 contains a brief review of the basic steps that are performed in connection with solving large sparse systems of equations, along with various contexts in which sparse systems are solved. This establishes the capabilities that one would expect to find in a reasonably comprehensive sparse matrix solver, *regardless of whether it even has an interface.* This in turn provides a framework for identifying the features that we believe an ideal user interface should possess in order that users, sophisticated and otherwise, are able to conveniently exploit the capabilities of the package. Section 3 describes the basic elements of the design of the package along with a description of the major *objects* that are integral parts of it. Section 4 contains some examples of how the package might be used, with particular focus on how users with varying levels of knowledge or sophistication can use the package. This section provides a context in which to demonstrate how the design objectives from section 2 are achieved, with comments about how the design, together with various programming language features, promotes that objective. The last section contains concluding comments about how various programming language features aid in the implementation of the design, drawing attention to the similarities and differences between C++ and Fortran-90, at least with respect to this application.

2. Design considerations. Loosely speaking, a user interface is something that allows a user to access the capabilities of a system. In the present context, the interface is simply a layer of software which shields the user from the complications associated with sparse matrix software, yet allows one to use that software in a natural and convenient way to solve sparse systems of equations. Naturally, the design of the interface is conditioned by the capabilities of the underlying sparse matrix software, so an essential first step is to describe what is assumed to be those capabilities.

For definiteness, the problem to be solved will be denoted by

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b},$$

where $\boldsymbol{A}$ is an $n \times n$ sparse coefficient matrix, and the method to be used is Gaussian elimination.[2] A triangular factorization of the matrix is computed, followed by the solution of two triangular systems in order to obtain the solution $\boldsymbol{x}$.

There is no general "best method" for solving sparse systems of equations. Even if one restricts the basic algorithm to Gaussian elimination, the way it is best implemented often depends on the characteristics of the given sparse linear system. Therefore, a sparse matrix package should accommodate a variety of methods and allow for convenient inclusion of methods yet to be developed. This should be possible with minimal disruption to the interface.

Sparse systems arise in a variety of contexts. Sometimes many problems having the same structure must be solved, and sometimes many problems differing only in their right-hand sides must be solved. Also, the way in which their structure and numerical values become available is highly variable. The package should be able to handle these situations efficiently and the interface should make it convenient and natural for the user to exploit that capability.

The discussion above suggests the capabilities that one would expect in a reasonably comprehensive sparse matrix package. In particular, there would be software available to handle sparse positive definite systems, sparse symmetric indefinite systems, and general sparse unsymmetric systems. These include software for ordering algorithms, software for creating appropriate data structures, and software for implementing the actual numerical routines. In what follows, we regard the software that does all these tasks, together with the user interface, as the *package*.

In general terms, the interface should support the following objectives:

- *Usability.* The intellectual overhead in learning to use the package should be low.
- *Versatility.* The package should be flexible; that is, it should be convenient to use in a wide variety of situations.
- *Layered accessibility.* The package should serve users having different levels of expertise in sparse matrix computation, ranging from the casual user to the sparse matrix researcher.
- *Extensibility.* The package should be designed so that it can be extended easily to new methods as they are developed and also to other classes of problems, such as sparse least squares problems and sparse nonlinear systems of equations. Ideally, such extensions should cause minimal or no disruption to the basic structure of the interface.
- *User control.* The interface should provide useful feedback, and the amount of such feedback should be under the control of the user.

[2] The authors' ultimate objective is to provide a package which deals with more general problems, such as over- and underdetermined systems. Solving such systems may involve algorithms other than Gaussian elimination, such as orthogonal factorization. However, for purposes of presentation, we restrict our attention to square systems, since their requirements are diverse enough that meeting the needs of these problems also provides the flexibility required to deal with more general classes.

3. Basic elements of the design. This section describes some of the essential ingredients of the package, with particular emphasis on its user interface. This provides a context in which to explain how many of the design objectives outlined in the previous section have been achieved.

3.1. C++ and Fortran-90 classes. For illustration purposes, some Fortran-90 source codes are included with the description. However, C++ examples could have as easily been included, and the discussion is largely language-independent.[3] The main features used to achieve the design objectives are available in both languages, although the details differ somewhat.

One of the key features used is the ability to encapsulate data structures and functions or subroutines that act upon them together as a single entity. In C++ these are *classes* and each class contains member variables and member functions.[4] Fortran-90 does not really have an analogue of a C++ class. It supports the feature of derived (user-defined) data types. However, unlike member functions in C++ classes, there is no facility to explicitly bind functions or subroutines to a data type.

In Sparspak-90, *modules* are used to support features similar to classes in C++. A Fortran-90 module can contain declarations of derived data types and procedures. A programming style has been adopted in Sparspak-90 such that a module contains a derived type together with routines that act on that data type. In other words, a logical association of a set of routines and a derived type is provided through membership in a module. Furthermore, the binding of these routines with the derived type can be achieved in a natural way by including an object of the derived type as an argument in each of the routines. This organization allows one to treat (and think about) Fortran-90 modules and C++ class definitions in the same way.

In what follows, the term "class" will be used. For C++ programmers, the meaning will be immediate. For Fortran-90 programmers, this should be interpreted as a module that defines a data type, together with a collection of routines that act on instances of that data type.

Instances of a class are called *objects*. A program may have several objects of the same class having different names. From the perspective of a typical user, use of the package involves creating and using two types of objects: **Problem** objects and **Solver** objects. These are described in the sections that follow.

3.2. The problem class. Regardless of the level of user sophistication, one task is fundamental and ubiquitous: the user must communicate the sparse matrix problem to the package, and the user interface should make this as convenient as possible. The task is complicated by the variety of ways in which the problem may materialize, as well as by transformations that the user might want to apply to the input. The different ways include the following:

- The structure of the problem and its numerical values may become available simultaneously or at differing times.
- There may be many systems to be solved, differing only in their numerical values.
- There may be numerous problems that differ only in their right-hand sides; these may be available all at once, or each may be the result of computations involving previous right-hand side(s) in a sequence. The latter circumstance

[3] The corresponding C++ examples can be found online at http://www.cs.yorku.ca/~joseph/.

[4] Member variables are sometimes referred to as instance variables, and member functions are sometimes referred to as "methods."

arises naturally if the package is being used in connection with solving a system of nonlinear equations.

- Given the matrix $\boldsymbol{A}$, it may be desirable to generate a right-hand side corresponding to a given solution or, given the structure of $\boldsymbol{A}$, it may be desirable to assign numerical values giving $\boldsymbol{A}$ certain properties (random, symmetric, positive definite, diagonally dominant, etc.). Such capabilities can be useful in developing and testing software.

The list above is far from exhaustive but serves to illustrate the variety of situations which the package should be able to handle. With these considerations in mind, it seems natural and compelling that the package should contain a class that can be viewed as the "problem repository." This class is given the name **Problem**, since objects of this class contain the numerical values and structure of the coefficient matrix, its right-hand side(s), corresponding solution vectors, and related information pertaining to the problem.

The class **Problem** has various member routines which operate on its data. Roughly speaking, these routines fall into four categories:

1. Procedures which provide for input of structural and numerical values, such as **InAij**, **InRow**, **InColumn**, and **InRhs**.
2. Procedures which adjust the input, such as **ReplaceAij**, **MakeGSymmetric**, and **MakeASymmetric**. (The latter two procedures make the problem structurally and numerically symmetric, respectively.)
3. Procedures which retrieve and/or display information, such as **GetRhs**, **GetSolution**, and **PrintSolution**.
4. Procedures which provide information about the problem, such as **IsASymmetric**, **IsGSymmetric**, and **IsAijPresent**. The first two procedures determine whether the matrix is numerically and structurally symmetric, respectively, while the latter two determine whether the ijth element of the matrix is present.

An important aspect of the **Problem** class is its general purpose design. It can represent sparse matrix problems with square or rectangular matrices, with symmetric or unsymmetric structures, and with symmetric or unsymmetric numerical values. The general nature of its design allows easy extension of the package in the future to handle more general types of sparse problems, thus promoting one of the design objectives, namely, that the package should be easily extendible.

In order to provide concreteness to the description above, Figure 1 contains a Fortran-90 subroutine which generates an $n \times n$ tridiagonal matrix problem. The example provides an initial opportunity to elaborate on several design features of Sparspak++ and Sparspak-90. Extensive use is made of the programming language feature known as *function name overloading*, which is available in both Fortran-90 and C++. That is, different routines are allowed to have the same names, provided that their parameter lists ("signatures") differ. The compiler detects which routines should be called by matching up the types and number of parameters in the routines. Thus, routines which perform essentially the same role, even though they may employ different internal data structures or operate on different objects, can still have the same name. This name overloading capability is an important way in which the intellectual overhead in learning and using the package is reduced.

Referring to the example in Figure 1, an instance of the problem class is initialized by calling the function **Construct**. *All* objects in the package are initialized by calling

```
subroutine MakeTriDiagProblem( p, n )
    use Sparspak90
    integer :: n, i
    type (Problem) :: p

    call Construct(p, "TriDiagProblem")
    do i = 1, n-1
        call InAij(p, i+1, i, -1D0); call InAij(p, i, i, 4D0)
        call InAij(p, i, i+1, -1D0); call InRhs(p, i, 1D0)
    end do
    call InAij(p, n, n, 4D0); call InRhs(p, n, 1D0)
end subroutine MakeTriDiagProblem
```

FIG. 1. *Tridiagonal test problem generator.*

a function whose name is **Construct**.[5] Of course, the routine that is actually executed will depend on the type of the first member of the argument list (which the compiler can determine). The key point is that the user has to remember only one function name in connection with creating new objects, regardless of their type or class.

Another example demonstrating the convenience of overloading is illustrated by the function **InAij** in the example in Figure 1. As noted in the previous section, the structure and the numerical values of the system of equations can arrive at different times and in different aggregations. If the user does not know the value of a_{ij} but wishes to communicate the fact that the ijth element of $\boldsymbol{A}$ is nonzero, the function **InAij** is still used but the last parameter is omitted.[6] Analogously, overloaded input routines **InRow**, **InColumn**, and **InSubMatrix** are available in the event that the nonzeros (or perhaps only their positions in the matrix) become available by rows or columns or as submatrices. Similar remarks apply to **InRhs**. The right-hand side $\boldsymbol{b}$ can be input one element at a time, as shown in the example in Figure 1, or as a subvector with an accompanying list of subscripts, or all at once. In all cases a function with the same name is called. Thus, the user must remember only a small number of function names to use **Problem** objects.

The call to the routine **Construct** in the example in Figure 1 illustrates another design feature. All objects that are created during the use of Sparspak are given names so that if they later generate error messages or other information, they can "identify themselves." Each class has a character string member variable called **objectName**, and the printing of object information always includes its **objectName** for identification. Since there may be numerous objects of the same type present in an executing program, this self-identification is important in helping the user understand output from his or her program. The explicit naming of each object created is optional; if the user omits the parameter, a name for the object is created automatically. Each class has a function **ReName** to make it possible for the user to change the name of an object if that is desirable. The setup allows users to assign unique names to objects for identification.

[5] This statement and the rest of the paragraph applies only to Sparspak-90. In C++, *constructors*, which play the same role as **Construct**, are called automatically whenever an object of a class is instantiated.

[6] Fortran-90 supports the notion of optional arguments, so this feature can be achieved in Fortran-90 by using an optional argument rather than subroutine name overloading.

```
program SimpleExample
    use Sparspak90
    type (Problem) :: p
    type (SparseSpdSolver) :: s

    call MakeTriDiagProblem(p, 5)   ! create test problem
    call Construct(s, p)            ! create solver object
    call Solve (s, p)               ! instruct s to solve p
    call PrintSolution(p)           ! print the solution
    call Destruct(p); call Destruct(s)  ! release storage
end program SimpleExample
```

FIG. 2. *Simple use of the package.*

Another noteworthy feature is that each class has standard member procedures **Save** and **Restore** for saving and restoring the contents of an object to and from an external file.

3.3. The solver class. The second major type of object that the typical user of the package will employ is a “solver” object. Loosely speaking, a solver object accepts a problem object as input and produces a solution to the problem. The package contains numerous different “solvers” for sparse systems of equations. That is, in addition to a class of type **Problem** in the package, there are solver classes, each consisting of a particular data structure, ordering algorithm, and numerical factorization and substitution routines; together, these implement a particular overall approach to solving a sparse system. For example, for symmetric positive definite systems, there are several effective algorithms for finding a low fill ordering; there are several efficient methods for storing Cholesky factors; and there are several efficient ways of implementing the factorization using the same data structure (left-looking, right-looking, multifrontal [3, 5, 9]). Various solvers result from selecting different combinations of these options.

The multitude of solvers is necessary because problems vary in several dimensions. They may or may not be square; if square, they may or may not be structurally symmetric. If they are structurally symmetric, they may or may not be numerically symmetric. Regardless of either shape or symmetry, row and/or column interchanges may be necessary to ensure numerical stability. In addition, for any particular combination of problem attributes above, there may be more than one approach that will solve the problem. Of course, a solver that assumes no special features will cope with them all but not as efficiently as one that exploits special features that a problem may possess.

3.4. Coarse structure of Sparspak. At a low (coarse) level of resolution, the package can be regarded as consisting of just two fundamental types of classes, namely, **Problem** and **xxxSolver**, where **xxx** denotes one of the numerous possibilities mentioned above. For example, Sparspak contains a solver for symmetric positive definite problems that reorders the problem to reduce fill. This solver class has the name **SparseSpdSolver**, standing for Sparse Symmetric positive-definite Solver. A simple example showing its use is given in Figure 2, where the subroutine in Figure 1 is used to create a small symmetric positive definite tridiagonal problem.

The package also contains a solver for symmetric positive definite problems that orders the problem so that it has a small envelope. This solver class has the name **EnvSpdSolver**, which stands for Envelope-reducing Symmetric positive definite Solver.

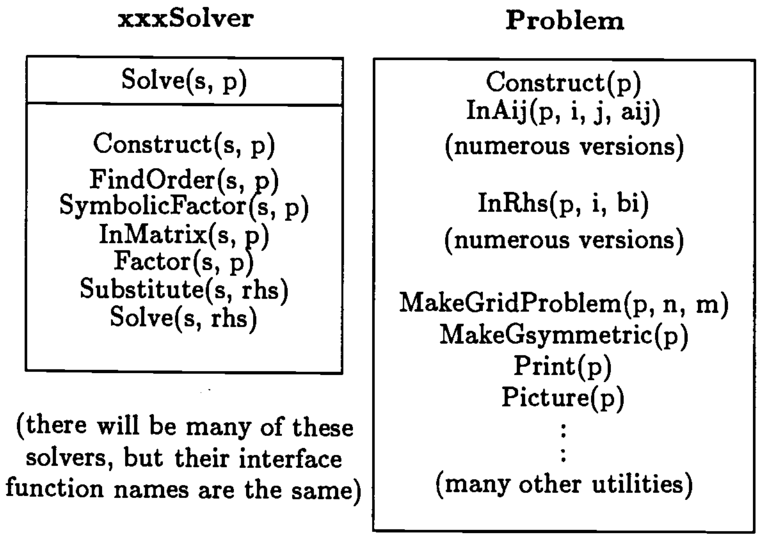


FIG. 3. *Coarse structure of Sparspak.*

The only change necessary in the program in Figure 2 in order to use this solver would be in line 4, where **SparseSpdSolver** would be changed to **EnvSpdSolver**. The use of function name overloading for **Solve** and **Destruct** means that no other changes are required; of course different procedures will be invoked. The compiler detects which procedures should be called by matching up the types and numbers of parameters in the procedures.

"Behind the scenes," **Solve** invokes subroutines which carry out the four standard steps of solving a sparse positive definite system. These subroutine names are also overloaded; they are invoked by the same names, regardless of the actual sparse positive definite system solver class being used:

1. **FindOrder**: find an appropriate ordering;
2. **SymbolicFactor**: symbolic factorization;
3. **Factor**: numerical factorization;
4. **Substitute**: numerical forward and backward substitution.

Before numerical factorization, **Solve** will also invoke **InMatrix** to extract the numerical matrix values from a given matrix **Problem** and place them into the internal data structure. The subroutine name **InMatrix** is also overloaded, so that all **Solve** subroutines use the same name, although the subroutine that it invokes depends on the actual solver, since the data structures that store the Cholesky factor will generally differ across solvers.

Thus, from the perspective of a typical user, Sparspak can be viewed as shown in Figure 3. Two types of objects are involved in its use, namely, problem objects and solver objects.

Figure 3 provides focus for an important point about the solver objects. As the example in Figure 2 illustrates, a solver module can be used to solve a problem using a single call to the subroutine **Solve**. That routine in turn invokes other subroutines mentioned above to perform the various steps in solving a symmetric positive definite system. These are the subroutines listed in the lower subbox in the box labeled

xxxSolver. However, these routines are also accessible *directly* by the user; a user who wishes to control when various steps of the computation are performed, or wishes to execute only some of the steps, can invoke these "second-level" procedures directly. The solvers contain private state variables (not accessible by the user) that ensure these subroutines are called in the correct order and that issue warnings or fatal error messages if they are called out of sequence.

It was noted earlier that there is no need for a function **Construct** in C++ because constructors are automatically called whenever a new object is created. Similarly, when objects disappear in C++ as a result of leaving the scope in which the object was declared, *destructors* are automatically invoked. This is not the case in Fortran-90, so Sparspak-90 has **Destruct** routines to explicitly release storage used by objects when they are no longer needed.

4. Meeting the design objectives. Section 2 described a number of design objectives, and section 3 provided a general description of the design of Sparspak. The objective of this section is to provide some examples which illustrate how the design of the previous section allows the objectives to be met. Throughout this section, references will be made to the five objectives enumerated in section 2.

One objective of the redesign of Sparspak is that it be able to cater to the needs of a wide variety of users, from the casual user who may know little about sparse matrix technology to the sparse matrix researcher who might use the package as a "toolbox" containing functionality useful in advancing the field. To that end, the examples in this section illustrate how the package might be employed by users with varying levels of sophistication or need.

At a very coarse level, usage of the package can be divided into two categories: *standard usage* and *research usage*. The distinction is based on the objects the user declares and manipulates. Of course this distinction is somewhat arbitrary, but it is useful in understanding the design and usage of the package. Standard users are those who are aware of the coarse structure of the package, as described in the previous section. Of course, within this group of users there will still be substantial variation in the level of sophistication, but the objects that are manipulated by the user will be generally limited to **Problem** objects and **Solver** objects. Some will use only a few of the interface routines available and make no use of options that might be available, while others will make extensive use of them. Research users, on the other hand, will declare and manipulate some or all of the more basic objects within the package, such as graphs, orderings, and so on. These objects will be used as building blocks for the researcher's own software development efforts.

4.1. Standard usage of the package. A simple example of basic usage of the package was introduced in Figure 2. The user instantiates an object of the **Problem** class, "loads" it with a problem, creates a **Solver** object, and then uses it to solve the problem. If the user does not know that the problem has any special characteristics (or that it may not have any), then the solver chosen will be a general one that assumes no special properties.

For purposes of this discussion, suppose the problem to be solved is symmetric and positive definite and that an envelope method is to be used to solve the problem. Also, suppose one step of iterative refinement is to be performed on the solution. These computations would be performed as shown in Figure 4. Here a built-in test problem generator **MakeGridProblem** is used. The package provides a number of subroutines to generate standard problems for testing software.

```
program refine
    use Sparspak90
    type (Problem) :: p
    type (EnvSpdSolver) :: s
    real (double), dimension(36) :: x, res

    call MakeGridProblem(p, 6, 6, "9pt") ! create test problem
    call Construct(s, p)                ! create solver object
    call Solve(s, p)                    ! solve for initial solution
    call ComputeResidual(p, res )       ! compute the residual
    call PrintVector(res, "residual")   ! print the residual
    call Solve(s, res)                  ! solve for correction
    call GetSolution(p, x);             ! retrieve the solution
    x = x + res                         ! compute improved solution
    call PrintVector(x, "refined solution") ! print solution
    call Destruct(s); call Destruct(p)  ! release storage
end program refine
```

FIG. 4. *Sample user program—one step of iterative refinement.*

The function name **Solve** is overloaded within each solver. In the example, the first call to it involves the solver and problem objects **s** and **p**, and the solution is put in the problem object **p**. In the second call to **Solve**, its arguments are the solver and a vector **rhs** containing the right-hand side; its contents are replaced by the solution.

Just as in the example in Figure 2, a change of only one text string in the program (**EnvSpdSolver**) is all that is needed to change the solver being used. All other function names would remain unchanged.

The second example in this section demonstrates how two different methods for solving a problem might be compared in terms of storage requirements and execution times. Such quantities are automatically captured within each solver. Suppose a user wants to compare the performance of the envelope and sparse solvers applied to a 50×50 grid problem obtained using a nine-point difference operator. One such program to do this is shown in Figure 5. Note again the use of name overloading: **Factor**, **PrintStats**, and **Destruct** invoke different subroutines, depending on the types of their parameters.

It was noted at the end of the previous section that interface routines for solver objects can be invoked at two levels. The user may choose to invoke only the **Solve** function; in that case, the appropriate routines will be called in the required order, and a solution will be produced. However, a user may wish to select various options that might be available within a solver and/or explicitly invoke some of the steps. An example appears in Figure 6. A grid problem is generated, just as in the example in Figure 5. Then the problem is factored using an envelope solver, where the default ordering is overridden by a random ordering. (The function **RandomOrdering** is a built-in function in Sparspak that generates a random ordering.) Then the same problem is factored using a fill-reducing solver, but the default ordering subroutine is overridden by a user-supplied subroutine called **MyOrdering**. To be able to implement such a subroutine, of course, the user must know the interface for such ordering subroutines.

Thus, some users of Sparspak will at times use the solver objects at this next level of resolution, explicitly invoking subroutines which execute the individual steps of the solution process. Some will select various options that may be available to adjust the

```
program compare
    use Sparspak90
    type (Problem) :: p
    type (EnvSpdSolver) :: envSolver
    type (SparseSpdSolver) :: sparseSolver

    call MakeGridProblem(p, 50, 50, "9pt") ! test problem

    call Construct(envSolver, p)    ! create solver objects
    call Construct(sparseSolver, p)
    call Solve(envSolver, p)        ! solve problem using each method
    call Solve(sparseSolver, p)

    call PrintStats(envSolver)      ! print performance statistics
    call PrintStats(sparseSolver)

    call Destruct(p)                ! release storage
    call Destruct(envSolver); call Destruct(sparseSolver)
end program compare
```

FIG. 5. *Sample user program—comparing two solvers.*

```
program myorder
    use Sparspak90
    type (Problem) :: p
    type (EnvSpdSolver) :: envSolver
    type (SparseSpdSolver) :: sparseSolver
    interface
        subroutine MyOrdering( g, order)
            use Sparspak90
            type (Ordering) :: order
            type (Graph) :: g
        end subroutine MyOrdering
    end interface

    call MakeGridProblem(p, 25, 25, "9pt")   ! test problem

    call Construct(envSolver, p)    ! create solver objects
    call Construct(sparseSolver, p)

    call Factor (envSolver, p, RandomOrdering)
    call PrintStats(envSolver, "random ordering: envelope solver")

    call Factor(sparseSolver, p, MyOrdering)
    call PrintStats(sparseSolver, "My ordering: sparse solver")

    call Destruct(p)              ! release storage
    call Destruct(envSolver); call Destruct(sparseSolver)
end program myorder
```

FIG. 6. *Using options in the package.*

```
program SimpleResearchUsage
    use Sparspak90
    type (Problem) :: p
    type (Grid) :: grd
    type (Graph) :: g
    type (Ordering) :: rcmOrder, mmdOrder

    call Construct(grd, 4, 4)    ! create 4 by 4 grid object
    call MakeGridProblem(p, grd, "9pt" )

    call Construct(g, p)    ! create graph object from p
    call Picture(g)         ! draw incidence matrix of the graph
    call Print(g)           ! print the adjacency lists of g

    call RCM(g, rcmOrder)   ! find RCM ordering of g
    call Picture(g, rcmOrder)! draw incidence matrix:RCM-reordered graph

    call MMD(g, mmdOrder)   ! find MMD ordering of g
    call Picture(g, mmdOrder)! draw incidence matrix:MMD-reordered graph

    call Destruct(g); call Destruct(p); call Destruct(grd) ! release storage
    call Destruct(rcmOrder); call Destruct(mmdOrder)
end program SimpleResearchUsage
```

FIG. 7. *Simple research-level usage of the package.*

behavior of the algorithms executed at each step. However, the basic objects that the user manipulates remain the same, and the details of data structures and algorithms are hidden from the user. The user needs only to be concerned with the "essential ingredients" of the problem.

The example in Figure 6 illustrates another design feature of Sparspak. Heavy use is made of optional arguments. Many of the subroutines allow the user to include a text string as a parameter to aid in making output from the package understandable. In the example, such a string is included in the calls to the subroutine **PrintStats**.

4.2. Research usage of the package. At this level of usage of the package, the user will be aware of some or perhaps all of the major classes or modules, along with the various routines that operate on objects from these classes. These include such things as graphs, orderings, elimination trees, grids, and so on. A simple example of usage at this level is given in Figure 7. The comments in the program in Figure 7 provide an explanation of what the program does.

The example in Figure 7 illustrates a number of additional features of Sparspak. Essentially all objects have the capability of printing themselves; that is, there is a subroutine called **Print** (heavily overloaded) that will print the contents of an object. For example, if the argument is a graph, the adjacency lists of the graph will be printed. If the object is a problem object, a listing of the nonzeros in the matrix and their positions will be printed, and similarly for other objects. Solver objects print detailed information about their data structures and orderings. These are mainly useful to sparse matrix researchers and for instructional purposes.

Similarly, many objects are able to draw a "picture" of themselves. For example, it is often useful to be able to see the structure of matrices and graphs, and there are **Picture** routines which provide such pictures. Indeed, even the solvers in Sparspak have **Picture** routines which provide pictures of the data structures used. Again, such routines are useful for sparse matrix researchers as well as helpful in connection with teaching about sparse matrix methods. As usual, the name is overloaded so that the user needs only to remember one name, regardless of what object is being pictured.

In cases where it makes sense to print or provide a picture of a permuted form of the object, such as when the object is a graph or a problem, then an ordering object can be (optionally) provided to the print and picture routines. This option is used in the example in Figure 7, where a picture of the graph under the two different orderings found is printed.

4.3. Low intellectual overhead. The examples in the previous subsections make it evident that learning how to use the standard capabilities of the package is relatively simple. Of course, if one wants to utilize the many options available, or use the basic building blocks within the package, more must be learned. However, learning to use the basic power and functionality of the software requires little intellectual investment. This has been achieved through the following:

1. *Simple structure.* The typical user really needs to know about only two kinds of objects, namely, problem objects that contain information about a sparse system of equations, and solver objects which accept a problem as input and produce a solution as output.
2. *Name overloading.* The user must remember only a few function names in order to use the package. Functions which perform essentially the same task, regardless of the underlying data structure and algorithms being used, are invoked by the same name.
3. *Information hiding.* Intricate data structures used in storing sparse factors and detailed implementation of numerical routines are hidden from the user.

4.4. Flexibility of the package. The examples presented above illustrate how the package can cope with a wide variety of circumstances. The features that facilitate its flexibility are as follows:

1. Object-oriented design, which allows multiple problem and solver objects to be manipulated in one program.
2. There are a variety of methods available for solving a sparse system. The various scenarios (multiple systems with the same structure, multiple systems differing only in the right-hand side, etc.) can be handled in a natural way, as illustrated by the example in Figure 4.

The example in Figure 6 illustrates how optional parameters allow the user to replace the standard subroutines that find the ordering. Although only a modest number of solution methods exist in the current package, the design allows the simple extension of the package to include additional new methods.

4.5. Serving different users. One of the design objectives was to be able to cater to the needs of users with varying levels of expertise. The examples in the previous subsections demonstrate that the design allows this to occur in a more or less seamless fashion.

At a basic level, the user creates one or more problem objects and one or more solver objects. Depending on the context, relatively few member routines of those objects might be invoked. An example of such usage is given in Figure 2.

On the other hand, the context might require slightly more functionality: One or more of the second-level routines in the solver (**FindOrder, SymbolicFactor**, etc.)

might be invoked. An example of this somewhat more sophisticated use of the package is given in Figure 4. Other examples of the usage of these second-level routines in a solver are given in Figures 5 and 6.

Finally, the individual basic objects upon which the package is based can also be employed by the user. This is illustrated by the example in Figure 7.

4.6. User-package communications. Messages from the package are grouped into three categories: error messages, warning messages, and information messages. The extent to which these messages are printed is under the control of the user through the value of a **messageLevel** variable. If it is set to zero, all messages are suppressed. Setting it to one allows fatal errors to be printed; setting it to two allows warning messages to be printed as well; and setting it to three permits all messages to be printed. The message level can be reset during execution.

Sparspak allows multiple objects of the same class in the same program. As noted earlier, to provide clear association of messages with objects, each object has a string variable **objectName** for its identification. All messages printed are accompanied by the corresponding **objectName**. This allows the user to relate package messages to the objects that printed them.

To facilitate the use of the package by researchers, a debugging facility has been included in Sparspak, using ideas borrowed in part from a debugging facility provided in the Nachos system [2]. The user can select one of four debugging levels which control the amount of debugging information printed. Setting the level to zero will suppress all debugging information, with levels one, two, and three specifying increasing amounts of information. Additionally, debugging messages can be grouped into *categories* through the use of debug flags. For example, if a researcher is developing a new ordering subroutine, only messages associated with orderings and graphs may be desired. By setting the debug flag to “og,” only messages with “o” (ordering) or “g” (graph) in their argument list will be printed. In this way, a software developer can choose to print only messages within specified categories as well as control the amount of information through setting the debug level. These parameters can be reset at any time in the user program so that the user can choose different types and levels of debugging information at selective portions of his or her code.

In Sparspak++, “conditional include” has been used for all the debugging statements. In a production environment, the user can simply choose to exclude all such debugging statements in compiling the code. On the other hand, for research development of new algorithms, the package should be compiled including these statements.

Unfortunately, Fortran-90 has no such conditional include facility, so a much less elegant solution has been used. All statements associated with debugging have a trailing comment of the form **!DEBUG**, so that a simple text stream processor can strip out the debugging statements prior to compilation when a “production version” of the package is desired.

5. C++ and Fortran-90: Concluding remarks. The development of the Sparspak++ and Sparspak-90 versions of Sparspak is an ongoing project. This article has described the objectives of the revision to Sparspak, which represent a substantial broadening over those of the original Sparspak package. In particular, supporting a much larger problem class and catering to the needs of a broad spectrum of users are prime objectives. The previous sections have provided the essential features of the design, along with a rationale for it. Also included was a collection of sample programs and commentary about them illustrating the way the design, together with various language features, allows the objectives enunciated in section 2 to be met.

Both C++ and Fortran-90 have a number of features that have been critical in achieving the design objectives set for Sparspak. The ability to overload function names is immensely helpful in keeping low the number of names a user must remember and is also helpful in understanding the structure of the package.

Another important feature of both languages is the ability to create "objects." Although Fortran-90 is not as versatile as C++ in this respect, it is possible with a simple programming protocol to have Fortran-90 modules and C++ classes serve more or less identical roles in the implementations of Sparspak-90 and Sparspak++. This feature is important in meeting one of the design goals, namely, the ability to simultaneously have a number of sparse matrix problems in the process of being solved.

In terms of providing a friendly and versatile interface to the user, one can easily conclude that these two programming language features are among the most important.

An important difference between the two languages that had a significant effect on the *ease of implementation* was support of *inheritance*. Briefly, in C++ new classes can be created from existing *base* classes by "subclassing" them. That is, one can define new classes as extensions of existing classes by declaring additional variables and introducing additional member functions. The new class thus defined *inherits* all the variables and functions from the base class. For example, the underlying structure of Sparspak++ involves the definition of an **SpdSolver** class containing basic objects and functions common to all symmetric positive definite direct solvers. Subclasses of **SpdSolver** will inherit this set of member variables and member functions from **SpdSolver**. Other functions and their interfaces may be declared in **SpdSolver**, but the actual implementations of these member functions will depend on the solution methods and will appear in subclasses of **SpdSolver**. There are currently two subclasses of **SpdSolver**: **EnvSpdSolver** for the envelope solution method and **SparseSpdSolver** for the sparse solution method. This feature has the important and well-known advantage of promoting the reuse of software and making it more manageable. Only one base class has to be maintained, even though it might be used in numerous other classes.

Unfortunately, Fortran-90 does not support the notion of inheritance, so the benefits of the strategy described above are less convenient to realize. There are modules containing the user-defined data types and the subroutines that implement each basic approach; these correspond to the subclasses of **SpdSolver** in the C++ implementation. For example, **EnvSpdBase** is one such Sparspak-90 "class" and another is **SparseSpdBase**. The corresponding solver **EnvSpdSolver** is created in a module with user-defined data type **EnvSpdSolver** having as one of its components **EnvSpdBase**. Thus, **EnvSpdBase** can be viewed as a subclass of **EnvSpdSolver** in the sense of C++. Similarly, a **SparseSpdSolver** is created in a module with user-defined data type **SparseSpdSolver** having as one of its components **SparseSpdBase**. This technique is known as *composition* in the software engineering literature.

At first glance, the advantage of software commonality appears to be lost, since there are two "base classes," namely, **EnvSpdSolver** and **SparseSpdSolver**. However, these solver classes are virtually identical. They differ only in the name of their user-defined data type and in the name of one of its components. Name overloading allows all subroutine calls to be otherwise identical. To change the **EnvSpdSolver** module to the **SparseSpdSolver** solver requires only a simple global text change. Thus, in reality only one solver module needs to be maintained. The use of makefiles and stream editor scripts allows the various solvers to be generated automatically, and effective reuse of common software is achieved.

The Sparspak++ implementation makes extensive use of a standard C++ library, called the Standard Template Library (STL) [10]. This has been a very substantial aid in implementing many of the fundamental classes in Sparspak++. For example, the STL vector and list classes are used extensively in creating the Sparspak++ Graph class.[7] There does not appear to be anything similar available in Fortran-90; such a library would be very valuable, although it is not obvious that the language supports the generality required to allow such a library to be implemented.

Not surprisingly, Fortran-90 has some powerful features for numerical computation that are absent in C++. Perhaps the most notably useful are various array operations. The so-called colon notation, allowing a subarray to be referenced, is highly useful. Also, array-valued subscripts are extremely useful in gather-scatter operations and in applying permutations to arrays.

There are several features within C++ and absent in Fortran-90 that are potentially useful, although their utility has not been fully explored at the time of writing. One is *dynamic binding*, that is, the ability to establish links to routines at execution time. A need for this feature has not yet been encountered, but it may be in the future. The other notable feature in C++ that is almost certain to be useful is the ability to “throw” and “catch” exceptions, such as floating-point overflow, divide-by-zero, etc. When such exceptions happen many levels down within the procedure hierarchy, it is often necessary to pass such information up the hierarchy to a level where the exception can be addressed. The throw-and-catch facility available in C++ obviates the need to have error flags appear in the argument lists of the routines which either raise the exception, service it, or simply transmit it from one call level to another.

Finally, there are some differences in the way function name overloading is done in the two languages and in the way function parameters are handled. However, the capabilities are comparable and did not lead to a real distinction between the two languages in terms of ease of implementation.

Acknowledgments. The authors would like to thank the referees and Dr. John Lewis for many perceptive comments that have greatly improved the layout, notation, and general readability of the paper.

REFERENCES

[1] G. Booch, *Object-Oriented Analysis and Design with Applications*, Benjamin/Cummings Publishing Co., Inc., Redwood City, CA, 1994.

[2] W. A. Christopher, S. J. Procter, and T. E. Anderson, *The nachos instructional operating system*, in Proceedings of USENIX Winter 1993 Technical Conference, The USENIX Association, Berkeley, CA, 1993, pp. 479–488.

[3] I. S. Duff, A. M. Erisman, and J. K. Reid, *Direct Methods for Sparse Matrices*, Oxford University Press, Oxford, UK, 1987.

[4] 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.

[5] A. George and J. W.-H. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice–Hall, Englewood Cliffs, NJ, 1981.

[7] One might anticipate that the use of such a library might exact performance penalties. However, C++ allows one to specify that a function be *inline*. This means that short utility functions that are frequently called can instead have their code bodies included directly in the calling program. The function is not “called,” and all overhead associated with the function call is eliminated.

[6] B. W. Kernighan and D. M. Ritchie, *The C Programming Language*, Prentice–Hall Software Series, Prentice–Hall, Englewood Cliffs, NJ, 1978.

[7] M. Metcalf and J. Reid, *Fortran* 90 *Explained*, Oxford Science Publications, Oxford, UK, 1990.

[8] B. Meyer, *Object-Oriented Software Construction*, Prentice–Hall, Englewood Cliffs, NJ, 1997.

[9] E. G. Ng and B. W. Peyton, *Block sparse Cholesky algorithms on advanced uniprocessor computers*, SIAM J. Sci. Comput., 14 (1993), pp. 1034–1056.

[10] A. Stepanov and M. Lee, *The Standard Template Library*, Technical Report HPL-94-34, Hewlett-Packard Laboratories, Palo Alto, CA, April 1994.

[11] B. Stroustrup, *The C++ Programming Language*, Addison–Wesley, Reading, MA, 1986.

SIAM J. Matrix Anal. Appl.
Vol. 20, No. 4, pp. 970–986

TOWARD AN EFFECTIVE SPARSE APPROXIMATE INVERSE PRECONDITIONER*

WEI-PAI TANG†

Abstract. Sparse approximate inverse preconditioners have attracted much attention recently, because of their potential usefulness in a parallel environment. In this paper, we explore several performance issues related to effective sparse approximate inverse preconditioners (SAIPs) for the matrices derived from PDEs. Our refinements can significantly improve the quality of existing SAIPs and/or reduce the cost of computing them. For the test problems from the Harwell–Boeing collection and some other applications, the performance of our preconditioners can be comparable or superior to incomplete LU (ILU) preconditioners with similar preconditioning cost.

Key words. approximate inverse, globally coupled local inverse, ILU preconditioner, exponential decay

AMS subject classifications. 15A09, 15A23, 65F10, 65F50, 65Y05

PII. S0895479897320071

1. Introduction. The use of preconditioned Krylov space methods has been proven to be a competitive solution technique for a wide range of large sparse matrix problems. It is commonly acknowledged now that a high-quality preconditioner holds the key to fast convergence. One of the most popular choices for the preconditioner is the ILU preconditioner. In most cases, level one fill (ILU(1)) [16] is enough to yield a good preconditioner. This surprisingly simple technique has provided a robust preconditioner for many rather challenging applications such as semiconductor device simulation, groundwater contamination, oil reservoir simulation, etc. However, the successful use of ILU preconditioners depends on dealing with several issues:

- *Potential zero pivot* during the factorization process;
- *Negative pivot* may occur, even if A is positive definite;
- *Instability*, if the inverses of L and U are very large [23];
- *Ordering*—the quality of the preconditioner is sensitive to the ordering of the unknowns [15, 16, 21];
- *Parallel implementations*—it is not a trivial task to write a high-performance parallel implementation of ILU preconditioners. It is even more difficult to construct an effective portable (between different architectures) implementation.

Recently, interest in a SAIP has emerged [8, 11, 14, 31, 32, 33, 35, 38, 39, 40]. The motivation of this renewed interest is largely from parallel processing. Instead of using ILU to approximate the matrix A, we seek a sparse approximation of A^{-1}. The forward and backward solution process in the ILU preconditioning step is replaced by a simple (sparse) matrix–vector product operation. Its parallel implementation is

*Received by the editors April 14, 1997; accepted for publication (in revised form) by E. Ng February 2, 1998; published electronically July 9, 1999. This work was supported by the Natural Sciences and Engineering Research Council of Canada, the Information Technology Research Centre, which is funded by the Province of Ontario, and NASA under contract NAS 2-96027 between NASA and the Universities Space Research Association (USRA).

http://www.siam.org/journals/simax/20-4/32007.html

†Department of Computer Science, University of Waterloo, Waterloo, ON N2L 3G1 Canada (wptang@bz.uwaterloo.ca). This work was done while the author was at the Research Institute for Advanced Computer Science (RIACS), NASA Ames Research Center.

straightforward and can be effectively implemented. In addition, several other potential problems with using ILU preconditioners are solved.

The idea of using a sparse approximation of A^{-1} as a preconditioner [3, 4] was proposed only slightly later than ILU was suggested [24]. The original crude approach was far less effective than the latter, and hence did not gain popularity. To construct a good, but sparse approximation $M \approx A^{-1}$, a key issue is the sparsity pattern of M. Initial approaches failed to provide a robust technique to determine an effective sparsity pattern. To reduce the cost of preconditioning, M has to be as sparse as possible. Unfortunately, this goal has to be balanced with the quality of the preconditioner. The search for an optimal sparsity pattern would be a much more expensive proposition than the solution of $Ax = b$ itself. Effective heuristics are required. The success of several new methods relies on their elegant schemes in determining the sparsity pattern and the solution of the approximation [5, 14, 11, 33, 39]. Their new insight to this "old" idea has offered new promise. Some difficult problems can be effectively solved using these new techniques.

There are two kinds of approaches to constructing sparse approximate inverses. The first and usually more effective one is the factorized sparse approximate inverse [5, 6, 39]. The second approach is to construct a sparse matrix as the solution of

$$\min_M \|AM - I\|_F, \tag{1.1}$$

subject to constraints on the number and position of the nonzeros entries of M. The Frobenius norm is particularly useful for parallel implementation. Notice that

$$\|AM - I\|_F^2 = \sum_{j=1}^{n} \|Am_j - e_j\|_2^2,$$

where m_j and e_j are the jth column of M and I, respectively. Thus solving (1.1) leads to solving n independent least squares problems,

$$\min_{m_j} \|Am_j - e_j\|_2, \qquad j = 1, \ldots, n, \tag{1.2}$$

which can be done in parallel.

A good comparison on these two different approaches can be found in [6]. We will concentrate on the Frobenius norm approach in this paper, since much of this work was completed before the new results were available. In addition, the techniques discussed here are in general also useful.

The Frobenius norm approach avoids many of the potential difficulties with ILU preconditioners and becomes a useful complementary approach to ILU. Currently, the average performance of this type of preconditioner cannot match the performance of the ILU type of preconditioner. The potential to improve them further will be explored in this note. A comparison of approximate inverse preconditioners and ILU(0) on Harwell–Boeing matrices can be found in [29].

In the next section, some limitations of a sparse approximation to a dense inverse are discussed. Section 3 presents some heuristic techniques which can improve the performance of this method. For a variety of problems derived from PDEs and some difficult problems from the Harwell–Boeing collection, the performance of our enhanced SAIP can be comparable or superior to an ILU preconditioner with similar preconditioning cost,[1] where the measure of cost is the number of nonzero entries in

[1] We refer the preconditioning cost only for the application of the preconditioner.

$$A = \begin{bmatrix} 0.96 & -1 & & & & & \\ -1 & 2 & -1 & & & & \\ & -1 & 2 & -1 & & 0 & \\ & & \ddots & \ddots & \ddots & & \\ & & & \ddots & \ddots & \ddots & \\ & & 0 & & -1 & 2 & -1 \\ & & & & & -1 & 2 \end{bmatrix}_{20\times 20} .$$

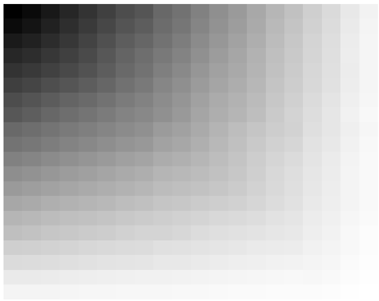


FIG. 2.1. *A counter example of decay.*

the preconditioner. It should also be noted that these techniques are also useful for any other approximate sparse inverse preconditioners.

2. Two issues. The inverse of a sparse matrix is generally full. The construction of a good sparse approximation is commonly based on the assumption that the majority of elements in the inverse are very small. This is often a questionable assumption. In particular, a problem displaying global coupling will necessarily have a dense inverse. This suggests an intrinsic conflict between a good approximation and a sparse approximation. A good approximation requires adequate global information in the preconditioner. However, the sparsity in the approximate inverse leads to compact support which inhibits the exchange of global information.

2.1. Inverse and decay. The basis for the SAIP approach is the assumption that the majority of elements in the inverse which we are to approximate are small. Its theoretical foundation is the decay estimate of the inverse elements of a sparse matrix. Many papers discuss this issue (e.g., [18, 19, 20, 22, 41, 48]). Matrices derived from PDEs typically exhibit this desirable decay feature in their inverse. However, the decay results presented there can often be misleading. For example, a typical estimate might be: If a banded matrix A is positive definite (or an M-matrix) the following bound can be established:

$$|b_{ij}| \leq \mathcal{C}\gamma^{|i-j|}, \tag{2.1}$$

where $\gamma < 1$, $\mathcal{C} > 0$ and b_{ij} is the element of A^{-1} at the (i, j)th location. Intuitively, when the distance $|i - j|$ is large, the size of the element b_{ij} will be very small, since $\gamma < 1$. This is known as the decay phenomenon. The statement (2.1), while correct, may cause the following facts to be overlooked.

While it is true that $\gamma^k \to 0$ when $k \to \infty$, very large *constant* $\mathcal{C}$ may lead to unacceptable slow decay, or none at all, as illustrated by the following examples. The matrix A on the left of Fig. 2.1 is a banded symmetric, positive definite matrix. The gray level picture of its inverse (Fig. 2.1) appears on the right . The darker the pixel in the image, the larger the element of the inverse. It can be seen that only the upper right part of the inverse displays decay. In the lower left part, the inverse elements are actually getting bigger when the distance $|i - j|$ increases! Is the estimate (2.1) wrong? No, it is correct. Since the constant $\mathcal{C}$ is so large, even an increasing trend can appear under this estimate.

Actually, a well-conditioned matrix can exhibit nondecay behavior. On the left of Fig. 2.2 is a "nice" M-matrix which is (row) diagonally dominant. The gray level

$$A_2 = \begin{bmatrix} 53.1 & -32 & & & & \\ -21 & 52.1 & -31 & & 0 & \\ & -21 & 51.1 & -30 & & \\ & & \ddots & \ddots & \ddots & \\ & 0 & & \ddots & \ddots & \ddots \\ & & & -21 & 35.1 & -14 \\ & & & & -21 & 34.1 \end{bmatrix}_{20\times 20} .$$

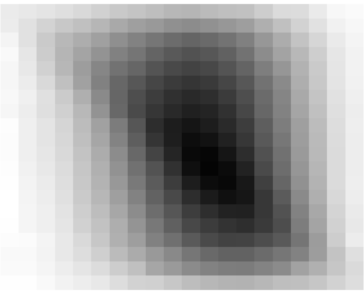
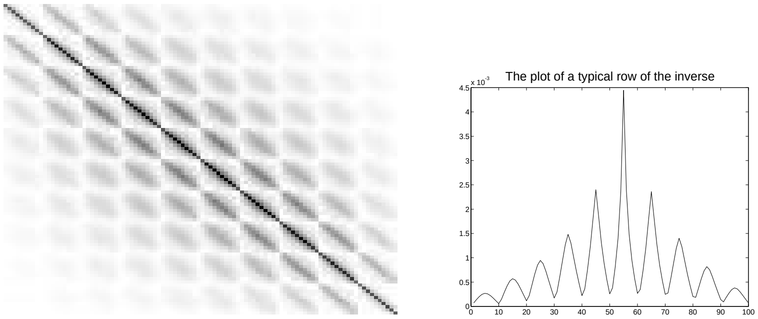


FIG. 2.2. *Another counter example of decay.*



(a) *The image of the inverse Laplacian.*

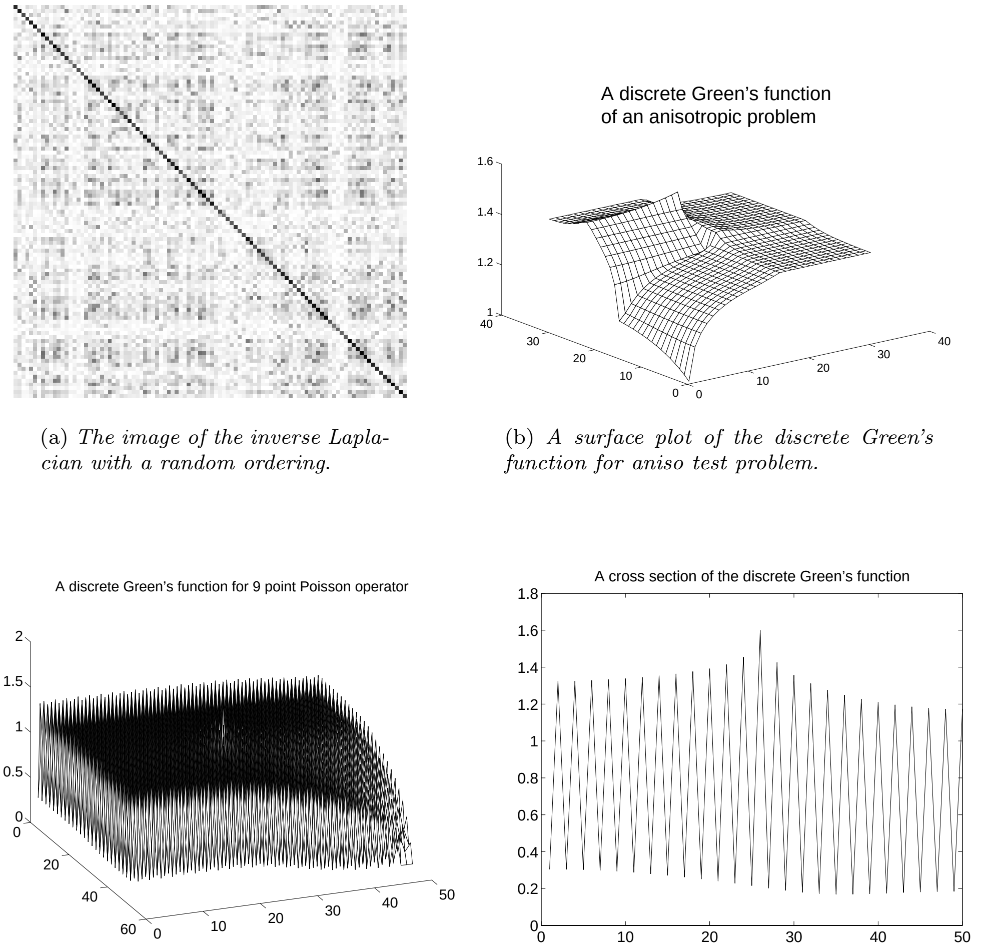
(b) *A cross section of the inverse Laplacian.*

FIG. 2.3. *One row of the inverse.*

image of its inverse is presented beside the matrix. Again, note that the elements in a row of the inverse of A_2 do not decay. For the first few rows, the size of the elements are increasing first when the column index increases. Interestingly enough, the elements in any column decay away from the diagonal "exponentially." Notice that this matrix is not column diagonally dominant. This example is specially designed to demonstrate the following often overlooked result: row diagonally dominant ensures only columnwise decay in the inverse and vice versa [48].

The second important issue in designing an effective SAIP algorithm is the pattern of the decay. It has been observed that the elements of some inverse matrices decay in an oscillatory pattern [48]. For example, the image of the inverse of a typical matrix derived from the model problem on an equally spaced rectangular grid using a 5-point stencil is presented in Fig. 2.3(a). A cross section of the inverse (a row) is also presented in Fig. 2.3(b). The estimate in (2.1) is obviously not useful for identifying the sparsity pattern in the approximation. The real challenging issue is how to extract the true decay pattern from the seemingly complicated image and use it to guide the selection of a sparsity pattern for a SAIP.

The complication in Fig. 2.3(b) is caused by the conflict between the construction of a matrix and the topology of the original problem. The matrix structure is essentially only helpful for operators in one-dimensional (topologically) space. For a sparse matrix problem derived from a higher dimensional grid, an artificial linear ordering



(a) *The image of the inverse Laplacian with a random ordering.*

(b) *A surface plot of the discrete Green's function for aniso test problem.*

(c) *Discrete Green's function.*

(d) *A cross section of the surface.*

FIG. 2.4. *A cross section of the surface.*

has to be imposed on the unknowns. These unknowns are physically distributed in a higher dimensional space. The distance between two unknowns in a matrix structure in this case has nothing to do with the distance in the solution domain of the original problem. The latter governs the physical influences between two unknowns. If a random ordering is chosen, the inverse matrix of the discrete Laplacian can look chaotic (see Fig. 2.4(a)). No useful conclusion can be abstracted from this picture, since the distance between any two unknowns doesn't have any physical meaning.

Any meaningful discussion of the decay should relate to the topological distance in the physical space of the problem rather than the index difference in a matrix. The template operator [48, 49] is a more natural structure. Using a template operator, a row of the inverse will be presented as a discrete Green's function on the physical solution domain. We can compare the discrete Green's function with the Green's function for the original PDE problem graphically using the template operator. The earliest discussion on this concept can be found in the classical book [26, pp. 315–318]. A more recent discussion can also be found in [41]. The surface plot of a discrete Green's function for the Laplace operator on a square grid is presented in Fig. 2.5(a). It is a very natural imitation of the Green's function in PDEs for this problem. All the mystery of the oscillatory decay disappears. A monotonic decay picture clearly relates the decay to the physical distance rather than to the indices in the matrix. In designing an effective sparsity pattern for SAIP, it is also important to use the physical (or graphical) distance as the criterion for determining the sparsity pattern of SAIP.

In addition, for example, discrete Green's functions do not display decay for many anisotropic (second-order) elliptic PDEs.[2] Fig. 2.4(b) is the discrete Green's function of a typical test case: aniso (see test problems) [12, 16]. No good sparse approximation can be obtained. Regular SAIP algorithms perform poorly for this kind of problem (see Table 3.1).

Another interesting example is from unsteady incompressible Navier–Stokes problems. The Poisson equation is solved at each time step. The discretization scheme is a 9-point stencil on a curvilinear grid. If we plot the discrete Green's function and one of its cross sections at the middle of the solution domain (with a 50 × 50 grid), the plots are extremely oscillatory (see Fig. 2.4(c,d)). No decay is displayed. Most of the elements in the inverse are far away from zero.

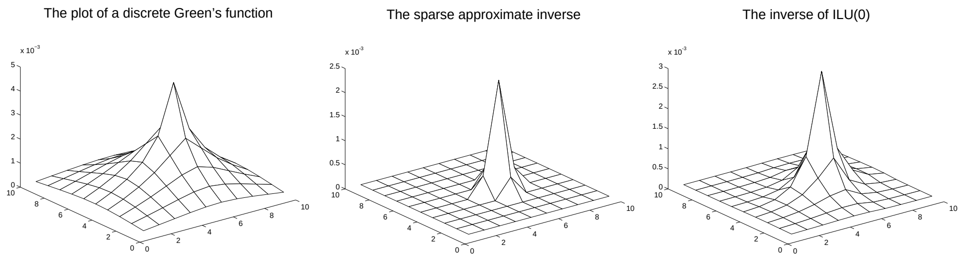
2.2. Global coupling. From the discussion above, it is clear that a good approximation cannot solely depend on the decay or on having a majority of the elements in the inverse be small. In particular, a sparse approximation implies some sort of compact support. Therefore, it cannot contain any global information in the approximation when a problem is globally coupled. Even if the discrete Green's function decays, the rate of the decay may not be fast enough to be represented by a few nonzero elements. Other techniques should be used in conjunction with the SAIP methods to yield competitive performance. Here is a simple benchmark for the model problem.[3] In the table below we compare the SPAI(0.4) [33] preconditioner with ILU(0) in order to reduce the residual norm by 10^{-8}. Both preconditioners have a similar number of nonzero elements. We used Bi-CGSTAB when the SPAI(0.4) preconditioner is used. The number of applications of the preconditioners (i.e., $Mx = p$, where M is the preconditioner) are compared. The number of nonzeros of the corresponding preconditioner is indicated in the parentheses. To ensure fairness of the comparison, we did not take advantage of symmetry in ILU(0).

	SPAI(0.4)	ILU(0)
Grid size	Numbers of applications (nonzeros in M)	
32 × 32	68 (4624)	25 (4992)
64 × 64	160 (19472)	63 (20224)

If we examine the surface plots of the ILU(0)s inverse and SPAI(0.4) and compare them to the original discrete Green's function, the reasons for differing performance are clear. Three plots for the same row (or grid node) are presented (see Fig. 2.5):

[2] When a problem is anisotropic, the decay pattern should also be guided by the anisotropy of the problem [12].

[3] We refer to the model problem as the Poisson equation on a unit square with Dirichlet boundary conditions.



(a) *The true discrete Green's function.*

(b) *The SAIP approximation.*

(c) *The discrete function of ILU (0) factorization.*

FIG. 2.5. *Discrete Green's function.*

- Left: the true discrete Green's function of the model problem at the middle of the solution region;
- Middle: the corresponding approximate inverse SPAI(0.4) for the same position;
- Right: the discrete Green's function of the ILU(0) factorization, namely, the same row of the ILU(0)s inverse.

The true inverse has global support while the SAIP can only collect the information from its few neighbors. Although the incomplete factorization has only compact support, its inverse still has a (weak) global base. That explains the very different performance.

3. The refinement techniques. From the previous discussion and our numerical testing, the following issues can be identified for the current techniques:

- Cost in identifying a good sparsity pattern and the approximation—this cost problem comes from two sources. The first is the computation of a good sparsity pattern for the approximation. The second source is the fast growth of the number of nonzero elements in the approximation when the accuracy of the approximation is tightened.
- Lack of global information in the approximation.
- The effectiveness of the compression—the sparse approximation of an inverse can be viewed as a compressed image. The information contained in the approximation for the inverse is not very effective if you compare Fig. 2.5(a) and Fig. 2.5(b).

We propose several approaches which can improve the performance of a sparse approximate inverse preconditioner.

3.1. Local inverse approximation. Both adaptive search technique [33] and dropping approach [11] are more computationally expensive than ILU factorization due to the dynamic nature of the computation. For the former approach, the least squares problems (1.2) are solved by the QR factorization. Algorithms are derived to determine the profitable positions of the nonzero entries adaptively. This dynamic approach makes the parallel implementation expensive. For Chow and Saad's approach, standard iterative methods (e.g., GMRES) are used to find an approximate solution to $Am_j = e_j$, and a dropping strategy is applied to m_j to control the amount of fill-in. The idea is to let the Krylov subspace build up the sparsity pattern gradually and then the nonzero entries are selected automatically by size.

We propose a simpler approach to make the computation cheaper. When this technique is used in conjunction with a global coupling method, the quality of the SAIP can be comparable with ILU preconditioners.

A sparse matrix can be represented by a digraph $\mathcal{G} = (\mathcal{O}, \mathcal{E})$ [28]. Define $\mathcal{L}_k(o_i)$, the k-level neighbor set of node o_i, as the set of nodes which are a distance $k+1$ or less from o_i. The 0-level neighbor set contains all the nodes which directly connect to node o_i. Similarly, define $\mathcal{W}_k(o_i)$, the kth-wavefront of node o_i, as the set of nodes which are a distance $k+1$ from node o_i. It can be shown that the elements in a discrete Green's function decay in a wavefront fashion for many problems [48]. In particular, $\mathcal{L}_k(o_i)$ includes the k most influential wavefronts. That is the motivation to choose $\mathcal{L}_k(o_i)$ for the locations to approximate the discrete Green's function at node o_i. The computation of these locations is also cheap. The submatrix[4]

$$A(\mathcal{L}_k(o_i), \mathcal{L}_k(o_i))$$

is defined as the k-level local matrix of node o_i. Using the level concept to define the sparsity pattern is widely used for ILU preconditioning [34, 16].

The local approximation of an inverse is to use the discrete Green's function of the local matrix $A(\mathcal{L}_k(o_i), \mathcal{L}_k(o_i))$ for node o_i as the approximation of the discrete Green's function of node o_i for the original matrix. More precisely, we solve a small problem for each node o_i:

$$A^T(\mathcal{L}_k(o_i), \mathcal{L}_k(o_i))x = e_{o_i},$$

where e_{o_i} is a unit basis vector with one in the o_i position.[5] Inject the elements of x to the corresponding locations of the k-level neighbor set in a zero row of the target approximation inverse. To compute the local discrete Green's function is an easy task and can be implemented in parallel effectively.

We call this approximate inverse a k-level local inverse preconditioner, or simply, a local inverse preconditioner. The level 0 approach was used by Benson originally. Unfortunately, the results were disappointing. However, when we use the k-level local matrix in conjunction with a global coupling, the result is more promising. In [36], the adjacency graph of A^k $k = 1,\ 2,\ 3$ is used for the sparsity pattern for a factorized approximate inverse. This is equivalent to our 1-level approach.

3.2. Globally coupled local inverse approximation. A promising technique which can bring in more global information is the multilevel approach. We consider an algebraic two-level technique to improve the quality of the approximation.

For the graph $\mathcal{G} = (\mathcal{O}, \mathcal{E})$ of a sparse matrix A, a set of global nodes (coarse grid nodes) is first selected. We group the indexes for global nodes at the end of a permutation vector. The original matrix can be reordered in a 2×2 block form:

$$A = \begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix},$$

[4] The MATLAB notation is adopted for extracting a submatrix from a given matrix A.

[5] Note that the local matrix can be singular. Since many nonzero elements have been dropped to form the local matrix, a similar technique used in the shifted ILU method is adapted to ensure that $A(\mathcal{L}_k(o_i), \mathcal{L}_k(o_i))$ is not singular. Specifically, the values of the dropped elements can be (or partially) added to the corresponding diagonal elements. For the test problems from PDEs, this remedy is actually never activated. Another remedy is to replace the above linear system by a least squares problem. Therefore, a local approximation of the discrete Green's function is still possible.

where the second row corresponds to the coarse (global) grid nodes. In [11], Chow and Saad also discussed the approaches using block form to improve the SAIP.

We tested several algorithms for choosing global nodes. The best performance comes from a generalized red-black ordering [15]. A brief description of the algorithm is as follows:

```
Set all nodes as unmarked.
While unmarked node set is not empty
    Pick an unmarked node as red (coarse grid) node and mark it.
    Mark all its unmarked neighbors as black (fine grid) nodes.
end
```

The inverse of this reordered block matrix is

$$A^{-1} = \begin{bmatrix} A_1^{-1}(I + A_2 S^{-1} A_3 A_1^{-1}) & -A_1^{-1} A_2 S^{-1} \\ -S^{-1} A_3 A_1^{-1} & S^{-1} \end{bmatrix},$$

where $S = A_4 - A_3 A_1^{-1} A_2$ is the well-known Schur complement. We compute the k-level local inverse $\widetilde{A_1^{-1}}$ of A_1; use this approximation to compute an approximation $\widetilde{S}$ of the Schur complement S. Then, compute the k-level local inverse $\widetilde{S^{-1}}$ of $\widetilde{S}$. The k-level globally coupled local inverse preconditioner (GCLI(k)) is then

$$M = \widetilde{A^{-1}} = \begin{bmatrix} \widetilde{A_1^{-1}}(I + A_2 \widetilde{S^{-1}} A_3 \widetilde{A_1^{-1}}) & -\widetilde{A_1^{-1}} A_2 \widetilde{S^{-1}} \\ -\widetilde{S^{-1}} A_3 \widetilde{A_1^{-1}} & \widetilde{S^{-1}} \end{bmatrix}. \tag{3.1}$$

For most of the PDE problems, A_1 is a diagonal matrix; hence, the computation of S is simple. Given a vector $x = (y^T, z^T)^T$, y contains the elements for the fine grid nodes (black), while z contains the elements for the coarse grid nodes (red). The preconditioning step Mx can be described as

$$Mx = M \begin{bmatrix} y \\ z \end{bmatrix} = \begin{bmatrix} u - \widetilde{A_1^{-1}} A_2 w \\ w \end{bmatrix},$$

where $u = \widetilde{A_1^{-1}} y$ and $w = -\widetilde{S^{-1}}(A_3 u - z)$. The process can be interpreted as one sweep of a two-level iteration. The vector u is the approximation on the fine grid, which is "projected" to the coarse grid for computing a new right-hand side $A_3 u - z$. Then the solution w on the coarse grid is used to refine the results on the fine grid in the second application of $\widetilde{A_1^{-1}}$ for $\widetilde{A_1^{-1}} A_2 w$.

Tables 3.1, 3.2, and 3.3 present the numerical results for some typical problems which arise in PDEs. A detailed description of these problems is presented in the next section. A description of the matrices from the Harwell–Boeing collection can be found in many references, for example, in [29, 33]. The advantages of using a simpler two-level approach are clear if one compares the performance of three different preconditioning techniques. The GCLI(k) shows competitive performance to ILU(k).

Table 3.2 presents the results for the same problems on a finer grid. To save space, we did not list the results for GCLI(1), since the improvement is similar. Notice that the algebraic two-level approach can be recursively applied to the Schur complement as well. Our numerical testing indicates that another 30% speed up can be obtained, on average. The detailed list is not presented. Of course, the improvement brought by a multilevel approach is not "cost-free." The original simple preconditioning step

Table 3.1
Performance comparison.

Problem		ILU(0)	SPAI(0.4)	GCLI(0)
	Size	Number of iterations (nonzero in M)		
Laplacian	861	22 (4930)	58 (4089)	20 (3977)
Variab.	861	19 (4930)	60 (3874)	17 (3977)
Disc.	900	22 (5280)	46 (4935)	30 (3880)
Indef.	861	21 (4930)	47 (3678)	27 (3977)
Aniso	961	33 (5642)	160(4805)	98 (4561)
		ILU(1)	SPAI(0.2)	GCLI(1)
	Size	Number of iterations (nonzero in M)		
Laplacian	861	15 (6498)	34(13338)	13 (9801)
Variab.	861	10 (6498)	19(13972)	10 (9801)
Disc.	900	13 (6963)	27(16513)	23 (9391)
Indef.	861	14 (6498)	24(14669)	22 (9801)
Aniso	961	24 (7442)	68(14551)	68 (11280)

Table 3.2
Performance comparison for finer grids.

Problem		ILU(0)	SPAI(0.4)	GCLI(0)
	Size	Number of iterations (nonzero in M)		
Laplacian	2401	35 (14210)	84 (11809)	44 (11620)
Variab.	2401	49 (14210)	75 (10833)	36 (11620)
Disc.	2500	51 (14800)	87 (14692)	53 (11620)
Indef.	2401	42 (14210)	73 (10855)	50 (11620)

Table 3.3
Anisotropic problems.

Problem		ILU(1)	SPAI(0.2)	Two(0.2)
	Size	Iterations (nonzero in M)		
Aniso	961	24 (7442)	68(14551)	38 (11310)
Stones	961	24 (7994)	94(17790)	41 (11190)
Sherman 4	1104	11 (7130)	22(11550)	17 (5237)

(a matrix–vector multiplication) is now replaced by several matrix–vector multiplications. The multilevel approach will complicate the parallel programming task and increase the cost in communications. However, the return from the investment is rewarding.

For many anisotropic problems, the GCLI(k) may cause unnecessary introduction of many nonzeros in the sparse approximation. Some of these nonzero entries do not contribute much to the quality of the approximation. We also tried to combine the adaptive search technique with the two-level method, namely, the sparse approximation of the matrix S^{-1} is computed by SPAI(α), where the control parameter $\alpha = 0.2$ is used. Table 3.3 presents numerical testing for some anisotropic problems. Two(0.2) is the combination of the two-level approach and the adaptive search method (SPAI(0.2)) [33].

3.3. A priori dropping technique. The technique of dropping small elements is often used after some approximation is formed. It can play an important role in reducing the growth of the nonzeros in an approximation. This technique is adopted in both ILU and SAIP. When the problem is anisotropic, however, the elements of its inverse usually decay extremely slowly or even have no small values at all. This often

Table 3.4
A priori dropping technique.

	ILU(0)	ILU(1)	Grote(0.2)	Two(0.2)	Drop(0.2)
	Number of iterations (nonzeros in the preconditioner)				
Orsirr 1	23 (7888)	10 (13562)	19 (13973)	12 (12050)	16 (3206)
Orsirr 2	23 (7856)	10 (11922)	19 (12137)	12 (10180)	15 (4653)
Orsreg 1	24 (16338)	12 (27060)	23 (30673)	10 (21080)	13 (12070)
Pores 2	27 (21710)	8 (50680)	27 (27598)	25 (28100)	53 (4399)
Pores 3	21 (4006)	10 (5844)	42 (24487)	41 (17110)	44 (9438)

leads to failure in determining an effective sparse approximation if a regular adaptive technique or discarding technique is used. When anisotropy is strong, it is beneficial to drop the small elements in the original matrix prior to computing the approximate inverse. Using the notation in MATLAB, the approximated original matrix is defined by

$$\tilde{A} = A. * (\mathrm{abs}(A) > \varepsilon),$$

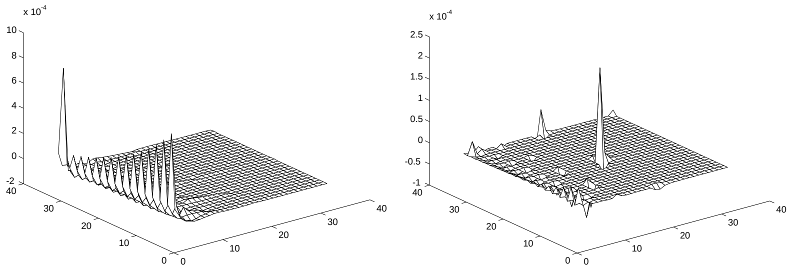
where ε is the threshold.[6] A similar idea is also adopted in algebraic multigrid [43], in the solution of dense systems [1, 37], and in [36].

We compute the approximate inverse of the approximated original matrix $\tilde{A}$ as the preconditioner. When many small elements are dropped, the fast growth of the number of nonzeros in SAIP is dramatically reduced. For the adaptive search techniques [33], the number of probes is also reduced and the resulting approximation has many fewer nonzeros. When the adaptive search technique and dropping small elements are combined with the two-level approach, the improvement for some problems is significant. For example, we choose the three oil reservoir simulation matrices: Orsirr 1, Orsirr 2, Orsreg 1, and Pores 2, Pores 3 from the Harwell–Boeing collection. These matrices result from a three-dimensional irregular grid and are very anisotropic. In Table 3.4 we present a performance comparison between the three different approaches, where Drop(0.2) drops the small elements before applying a two-level Grote–Huckle approach with control parameter $\alpha = 0.2$. The computational cost in both computing and applying the sparse approximate preconditioner is significantly reduced.

3.4. Wavelet compression. Each of the discrete Green's functions of a matrix (see Fig 2.5(a)) can be considered as a smooth surface. A compact approximation cannot represent the surface well. We also remark that even if the discrete Green's function displays decay behavior, the rate of decay may not be enough for the sparse approximate inverse to have a good convergence performance. However, the smooth surface in the discrete Green's function can be converted into many very small wavelet coefficients. Specifically, we apply a wavelet transform to compress the piecewise smooth discrete Green's function and then apply the standard techniques (e.g., Grote and Huckle's implementation) to construct a sparse approximate inverse in the wavelet space. Having a majority of small elements allows us to aggressively discard most of these small elements and maintain the quality of the approximations. Intuitively, we can use the same storage to recover a more "authentic" inverse.

Two discrete Green's functions of the Laplacian in the wavelet space are displayed in Fig. 3.1. The most significant contribution in these discrete Green's functions is

[6] For a matrix with a very different scale, the threshold should relate to the largest element in a given row, and different rows should use different thresholds for discarding the small elements.



(a) *At the corner of the domain.* (b) *At the center of the domain.*

FIG. 3.1. *Discrete Green's functions in wavelet space.*

very compact. Fig. 3.1(a) corresponds to a grid node near the corner, while Fig. 3.1(b) is for the center grid node of the rectangular region.

The fast wavelet-based approximate inverse can be constructed as follows. Given an orthogonal wavelet function in the continuous space, there corresponds an orthogonal matrix W that transforms vectors from the standard basis to the wavelet basis. Furthermore, if v is a vector of smoothly varying numbers (with possibly local singularities), its wavelet representation $\tilde{v} = Wv$ will have mostly small entries. We can also represent two-dimensional transforms by W. Let A be a matrix in the standard basis. Then $\tilde{A} = WAW^T$ is the representation of A in the wavelet basis. This wavelet representation $\tilde{A}$ is also called the standard form of A [7]. Assuming A^{-1} is piecewise smooth, our idea is to apply a wavelet transform to compress A^{-1} and then use it as a preconditioner. At first glance, this seems impossible since we do not even have A^{-1}. Note that

$$\widetilde{A^{-1}} \;\equiv\; WA^{-1}W^T \;=\; (WAW^T)^{-1} \;=\; \tilde{A}^{-1},$$

where W is an orthogonal wavelet transform matrix. Therefore, we can first transform A to its wavelet basis representation $\tilde{A}$ and then apply, for example, Grote and Huckle's method to find an approximate inverse for $\tilde{A}$, which is the preconditioner that we want to compute. In other words, we do not need to form A^{-1} but are still able to compute its transform.

We shall next show how we utilize the wavelet transform in the least squares approach. Consider (1.1) again. Let W be an orthogonal wavelet transform matrix; i.e., $\tilde{x} = Wx$ is the vector x in the wavelet basis. (Note that W can be 1-level or full $\log_2 n$-level wavelet transform matrix.) Then

$$(3.2)\qquad \begin{aligned}\min_M \|AM - I\|_F &= \min_M \|WAW^TWMW^T - I\|_F \\ &= \min_{\tilde{M}} \|\tilde{A}\tilde{M} - I\|_F,\end{aligned}$$

where $\tilde{A} = WAW^T$ and $\tilde{M} = WMW^T$ are the representations of A and M in the wavelet basis, respectively. Thus, our n least squares problems become

$$(3.3)\qquad \min_{\tilde{m}_j} \|\tilde{A}\tilde{m}_j - e_j\|_2, \qquad j = 1, 2, \ldots, n.$$

Note that since A is sparse, so is $\tilde{A}$ (but probably denser than A). Because of the wavelet basis representation, if M is piecewise smooth, we would expect $\tilde{M}$, neglecting

Table 3.5
Wavelet sparse approximate inverse.

Test Problem	Wavelet SPAI	SPAI(0.4)	SPAI(0.2)	ILU(0)
(64 × 64)	Number of iterations (nonzeros in the preconditioner)			
Laplacian	50 (6616)	160 (19472)	63 (69688)	57 (20224)
Variab.	66 (6616)	>200 (18618)	129 (73936)	93 (20224)
Helical	68 (6616)	>200 (19472)	126 (69628)	89 (20224)

Table 3.6
A comparison of scaling for Pores 2.

	ILU(0)	ILU(1)	Grote(0.2)	Two(0.2)	Drop(0.2)
	Number of iterations (nonzeros in the preconditioner)				
Scaling	27 (21710)	8 (50680)	27 (27598)	25 (28100)	53 (4399)
No scaling	26 (10837)	12 (21289)	57 (73099)	74 (31350)	78 (27840)

small entries, to be sparse too. Therefore, the sparse solution of (3.3) would hopefully give rise to a more effective approximate inverse than the original approach without the wavelet transform.

The numerical testing and comparisons for three test problems arising from PDEs are presented in Table 3.5. Wavelet compression demonstrates its effectiveness in compression. Very few nonzero entries (one-third of the ILU(0) or one-tenth of the SPAI(0.2)) are used to approximate the inverse in wavelet space and the quality of the preconditioner is superior to ILU(0). A detailed analysis and more experimental results are presented in [10].

We have presented several approaches which can improve the performance of SAIP. Many other techniques similar to those used in the ILU preconditioner may also be beneficial. For example, (block) diagonal scaling is often a very useful technique. In Table 3.6, block row scaling is applied to matrices Pores 2 and Pores 3. With the scaling, the performance of these same techniques are very different compared to without scaling. The benefits of scaling were discussed in several papers (for example, [25]). Table 3.6 is a comparison for test problem Pores 2.

In summary, there are many techniques which can be used toward a more effective approximate inverse preconditioner. The choice should depend on the characteristics of the application. The techniques we discussed here are also useful for other approximate inverse preconditioners.

4. Conclusion.

In this paper, we have discussed several implementation issues which may enhance the performance of a sparse approximation inverse preconditioner. For a specific application, if the proper technique is adopted, the improvement can be significant.

Appendix A. Test problems.

A detailed description of the test problems is presented here. The iterative methods, preconditioned CG and Bi-CGSTAB, are used for computations [50].

Helical spring (Helical). If a single turn of a helical spring of small angle α and radius R is deformed into a plane ring under the influence of an axial load, the stress-function Φ can be shown to satisfy the differential equation [13]

$$\Phi_{xx} + \Phi_{yy} + \frac{3}{R-y}\Phi_x - 2G\lambda = 0,$$

and it vanishes on the boundary Γ, where Γ is the boundary of the cross section in the (x, y) plane which contains the axis of the spring. G is the modulus of rigidity and $\lambda = \sin\alpha\cos\alpha/R$.

The special case we considered is one in which the problem has rectangular cross section $\Omega = (-.5, .5) \times (-1, 1)$ and $R = 5$. The problem has an exact solution [42]:

$$\Phi = (1 - y^2)(1 - 4x^2)(5 - y^3)(0.0004838y + 0.0010185).$$

Variable coefficient problem (Variab). This problem has continuous variable coefficients for second-order derivative terms. The unknown is defined on a unit square with homogeneous Dirichlet boundary condition. It satisfies the equation

$$((1 + x^2)u_x)_x + u_{yy} + (\tan y)^3 u_y = -100x^2.$$

Discontinuous coefficient problem (Disc). In this test, the following equation was considered:

$$(K_x u_x)_x + (K_y u_y)_y + u_x + u_y = \sin(\pi xy).$$

The unknown vanishes on the boundary of the solution domain $\Omega = (0, 1) \times (0, 1)$, where

$$K_x = K_y = \begin{cases} 1, & [0, .5] \times [0, .5], \\ 10^{+3}, & [.5, 1] \times [0, .5], \\ 10^{-3}, & [0, .5] \times [.5, 1], \\ 1, & [.5, 1] \times [.5, 1]. \end{cases}$$

In general, discontinuous coefficient problems are relatively difficult to solve.

Stone's third problem (Stones). This anisotropic and discontinuous coefficient test problem is widely used in the oil industry as a benchmark problem [47]. The equation

$$\frac{\partial}{\partial x}\left(K_x \frac{\partial P}{\partial x}\right) + \frac{\partial}{\partial y}\left(K_y \frac{\partial P}{\partial y}\right) = -q \tag{A.1}$$

was discretized on the unit square using a finite difference technique. A 33 × 33 grid was used (see Fig. A.1), and a harmonic average was used for discontinuities in K_x and K_y [2]. Let x_i and y_i, $i = 1, \ldots, 33$ be the grid nodes on the x-axis and y-axis, respectively. The values of K_x, K_y, and q are

$$(K_x, K_y) = \begin{cases} (1, 100) & \text{if } (x_i, y_j) \in A, \quad 14 \le i \le 30,\ 1 \le j \le 16, \\ (100, 1) & \text{if } (x_i, y_j) \in B, \quad 5 \le i \le 12,\ 5 \le j \le 12, \\ (0, 0) & \text{if } (x_i, y_j) \in C, \quad 12 \le i \le 19,\ 21 \le j \le 28, \\ (1, 1) & \text{if } (x_i, y_j) \in D, \quad \text{elsewhere.} \end{cases} \tag{A.2}$$

$$\begin{aligned} &q_1(3, 3) = 1.0, \quad q_2(3, 27) = 0.5, \quad q_3(23, 4) = 0.6, \\ &q_4(14, 15) = -1.83, \quad q_5(27, 27) = -0.27. \end{aligned} \tag{A.3}$$

Indefinite PDE (Indef). The problem tested was

$$\begin{aligned} Lu = &-\left[\left(1 + \frac{1}{2}\sin(50\pi x)\right) u_x\right]_x - \left[\left(1 + \frac{1}{2}\sin(50\pi x)\sin(50\pi y)\right) u_y\right]_y \\ &+ 20\sin(10\pi x)\cos(10\pi y)u_x - 20\cos(10\pi x)\sin(10\pi y)u_y + cu, \end{aligned} \tag{A.4}$$

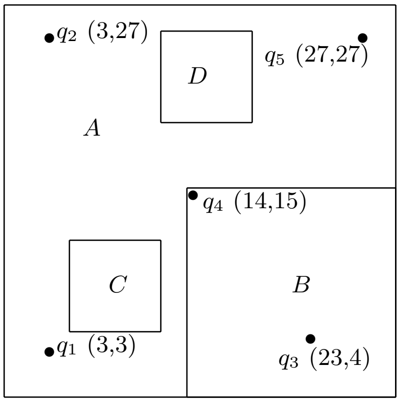


FIG. A.1. *Stone's third problem.*

where $u = \exp(xy)\sin(\pi x)\sin(\pi y)$ is defined on a unit square. This testing problem is taken from the paper of Cai, Gropp, and Keyes [9]. The sign of the coefficient c in (A.4) has a crucial effect on this problem. The difficulty of the linear system depends on both the mesh widths $\triangle x, \triangle y$ and the magnitude of c. For a fixed grid size, the larger the magnitude of the negative value c, the more difficult the problem will be. In this work, the cases for $c = -20$ and -70 were tested.

Anisotropic PDE (Aniso). The PDE is the same as (A.1). The value distributions of K_x and K_y are

$$(K_x, K_y) = \begin{cases} (100,1) & \text{if } 0 \le x \le 1/2, \quad 0 \le y \le 1/2, \\ (1,100) & \text{if } 1/2 \le x \le 1, \quad 0 \le y \le 1/2, \\ (100,1) & \text{if } 1/2 \le x \le 1, \quad 1/2 \le y \le 1, \\ (1,100) & \text{if } 0 \le x \le 1/2, \quad 1/2 \le y \le 1. \end{cases}$$

REFERENCES

[1] G. ALLEON, M. BENZI, AND L. GIRAUD, *Sparse approximate inverse preconditioning for dense linear systems arising in computational electromagnetics*, Numer. Algorithms, 16 (1997), pp. 1–15.

[2] A. BEHIE AND P. A. FORSYTH, *Comparison of fast iterative methods for symmetric systems*, IMA J. Numer. Anal., 3 (1983), pp. 41–63.

[3] M. W. BENSON, *Iterative Solution of Large Scale Linear Systems*, thesis, Lakehead University, Thunder Bay, Ontario, Canada, 1973.

[4] M. BENSON AND P. FREDERICKSON, *Iterative solution of large sparse linear systems arising in certain multidimensional approximation problems*, Utilitas Math., 22 (1982), pp. 127–140.

[5] M. BENZI, C. D. MEYER, AND M. TUMA, *A sparse approximate inverse preconditioner for the conjugate gradient method*, SIAM J. Sci. Comput., 17 (1996), pp. 1135–1149.

[6] M. BENZI AND M. TUMA, *Numerical experiments with two approximate inverse preconditioners*, BIT, 38 (1998), pp. 234–241.

[7] G. BEYLKIN, R. COIFMAN, AND V. ROKHLIN, *Fast wavelet transforms and numerical algorithms* I, Comm. Pure Appl. Math., 44 (1991), pp. 141–184.

[8] R. BRIDSON AND W.-P. TANG, *Ordering anisotropy and factored sparse approximate inverses*, SIAM J. Sci. Comput., submitted.

[9] X.-C. CAI, W. D. GROPP, AND D. E. KEYES, *A comparison of some domain decomposition algorithms for nonsymmetric elliptic problems*, in Proceedings Fifth International Symposium on Domain Decomposition Methods for Partial Differential Equations, T. F. Chan, D. E. Keyes, G. Meurant, J. S. Scroggs, and R. G. Voigt, eds., SIAM, Philadelphia, PA, 1992, pp. 224–235.

[10] T. Chan, W.-P. Tang, and W. Wan, *Wavelet sparse approximate inverse preconditioners*, BIT, 37 (1997), pp. 644–650.

[11] E. Chow and Y. Saad, *Approximate inverse techniques for block-partitioned matrices*, SIAM J. Sci. Comput., 18 (1997), pp. 1657–1675.

[12] S. S. Clift and W.-P. Tang, *Weighted graph based ordering techniques for preconditioned conjugate gradient methods*, BIT, 35 (1995), pp. 30–47.

[13] L. Collatz, *The Numerical Treatment of Differential Equations*, 3rd ed., Springer-Verlag, New York, 1966.

[14] J. Cosgrove, J. Diaz, and A. Griewank, *Approximate inverse preconditionings for sparse linear systems*, Internat. J. Comput. Math., 44 (1992), pp. 91–110.

[15] D. F. D'Azevedo, P. A. Forsyth, and W.-P. Tang, *Ordering methods for preconditioned conjugate gradient methods applied to unstructured grid problems*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 944–961.

[16] D. F. D'Azevedo, P. A. Forsyth, and W.-P. Tang, *Towards a cost effective ILU preconditioner with high level fill*, BIT, 32 (1992), pp. 442–463.

[17] D. F. D'Azevedo, P. A. Forsyth, and W.-P. Tang, *Drop tolerance preconditioning for incompressible viscous flow*, Internat. J. Computer Math., 44 (1992), pp. 301–312.

[18] C. de Boor, *Dichotomies for band matrices*, SIAM J. Numer. Anal., 17 (1980), pp. 894–907.

[19] S. Demko, *Inverses of band matrices and local convergence of spline projections*, SIAM J. Numer. Anal., 14 (1977), pp. 616–619.

[20] S. Demko, W. F. Moss, and P. W. Smith, *Decay rates for inverse of band matrices*, Math. Comp., 43 (1984), pp. 491–499.

[21] I. S. Duff and G. A. Meurant, *The effect of ordering on preconditioned conjugate gradients*, BIT, 29 (1989), pp. 635–657.

[22] V. Eijkhout and B. Polman, *Decay rates of inverses of banded m-matrices that are near to Toeplitz matrices*, Linear Algebra Appl., 109 (1988), pp. 247–277.

[23] H. C. Elman, *A stability analysis of incomplete LU factorizations*, Math. Comp., 47 (1986), pp. 191–217.

[24] D. J. Evans, *The use of pre-conditioning in iterative methods for solving linear equations with symmetric positive definite matrices*, J. Inst. Math. Appl., 4 (1968), pp. 295–314.

[25] Q. Fan, P. A. Forsyth, J. R. F. McMacken, and W.-P. Tang, *Performance issues for iterative solvers in device simulation*, SIAM J. Sci. Comput., 17 (1996), pp. 100–117.

[26] G. E. Forsythe and W. R. Wasow, *Finite Difference Methods for Partial Differential Equations*, John Wiley, New York, 1960.

[27] R. W. Freund, G. H. Golub, and N. N. Nachtigal, *Iterative solution of linear systems*, Acta Numerica, (1992), pp. 57–100.

[28] A. George and J. Liu, *Computer Solution of Large Sparse Positive Definite Systems*, Prentice–Hall, Englewood Cliffs, NJ, 1981.

[29] N. I. M. Gould and J. A. Scott, *On Approximate-Inverse Preconditioners*, Technical report RAL-TR-95-026, Computing and Information System Dept., Atlas Center, Rutherford Appleton Laboratory, Osfordshire OX11 0QX, England, 1995.

[30] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 2nd ed., The Johns Hopkins University Press, Baltimore, MD, 1989.

[31] M. Grote and H. Simon, *Parallel preconditioning and approximate inverses on the connection machine*, in Proceeding of the Scalable High Performance Computing Conference (SHPCC), April 1992, Williamsburg, VA, IEEE Computer Science Press, Piscataway, NJ, 1992, pp. 76–83.

[32] M. Grote and H. Simon, *Parallel preconditioning and approximate inverses on the connection machine*, in Sixth SIAM Conference on Parallel Processing for Scientific Computing II, R. Sincovec et al., eds., SIAM, Philadelphia, PA, 1993, pp. 519–523.

[33] M. Grote and T. Huckle, *Parallel preconditioning with sparse approximate inverses*, SIAM J. Sci. Comput., 18 (1997), pp. 838–854.

[34] I. Gustafsson, *A class of first order factorization methods*, BIT, 18 (1978), pp. 142–156.

[35] T. Huckle and M. Grote, *A New Approach to Parallel Preconditioning with Sparse Approximate Inverses*, Technical report SCCM-94-03, Dept. of Scientific Computing/Computational Mathematics, Stanford University, Stanford, CA, 1994.

[36] I. E. Kaporin, *New convergence results and preconditioning strategies for the conjugate gradient method*, Numer. Linear Algebra Appl., 1 (1994), pp. 179–210.

[37] L. Y. Kolotilina, *Explicit preconditioning of systems of linear algebraic equations with dense matrices*, J. Soviet Math., 43 (1988), pp. 2566–2573.

[38] L. Kolotilina, A. Nikishin, and A. Yeremin, *Factorized sparse approximate inverse (FSAI) preconditionings for solving* 3*D FE systems on massively parallel computers* II, in Iterative Methods in Linear Algebra, Proc. of the IMACS International Symposium, Brussels, 1991, R. Beauwens and P. Groen, eds., 1992, pp. 311–312.

[39] L. Y. Kolotilina and A. Y. Yeremin, *Factorized sparse approximate inverse preconditionings* I. *Theory*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 45–58.

[40] J. Lifshitz, A. Nikishin, and A. Yeremin, *Sparse approximate inverse (FSAI) preconditionings for solving* 3*D CFD problems on massively parallel computers*, in Iterative Methods in Linear Algebra, Proc. of the IMACS International Symposium, Brussels, 1991, R. Beauwens and P. Groen, eds., 1992, pp. 83–84.

[41] G. Meurant, *A review on the inverse of symmetric tridiagonal and block tridiagonal matrices*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 707–728.

[42] J. Rice and R. Boisvert, *Solving Elliptic Problems Using ELLPACK*, Springer-Verlag, New York, 1985.

[43] J. W. Ruge and K. Stüben, *Algebraic multigrid (AMG)*, in Multigrid Methods, Frontiers in Applied Mathematics 3, S. F. McCormick, ed., SIAM, Philadelphia, PA, 1987, pp. 73–130.

[44] Y. Saad, *Krylov subspace methods on supercomputers*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 1200–1232.

[45] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[46] H. D. Simon, *Incomplete LU preconditioners for conjugate-gradient-type iterative methods*, Paper No. SPE 13533, in Proceedings of the 1985 Reservoir Simulation Symposium, Dallas, TX, 1985.

[47] H. L. Stone, *Iterative solution of implicit approximations of multidimensional partial differential equations*, SIAM J. Numer. Anal., 5 (1968), pp. 530–558.

[48] W.-P. Tang, *Schwarz Splitting and Template Operators*, Ph.D. thesis, Computer Science Dept., Stanford University, Stanford, CA, 1987.

[49] W.-P. Tang, *Template Operators and Exponential Decay*, Technical report CS-90-46, Dept. of Computer Science, University of Waterloo, Ontario, Canada, 1990.

[50] H. A. Van Der Vorst, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Comput., 13 (1992), pp. 631–644.

SIAM J. MATRIX ANAL. APPL.
Vol. 20, No. 4, pp. 987–1006

SCALABLE PARALLEL PRECONDITIONING WITH THE SPARSE APPROXIMATE INVERSE OF TRIANGULAR MATRICES[*]

ARNO C. N. VAN DUIN[†]

Abstract. In this paper an approach is proposed for preconditioning large general sparse matrices. This approach combines the scalability of the application of explicit preconditioners with the preconditioning efficiency of incomplete factorizations. Several algorithms resulting from this approach are presented. Both the preconditioning efficiency and the cost of applying this preconditioner are tested. The experiments indicate that this technique offers the ability to make efficient use of parallel computers with a convergence rate comparable to that of the underlying incomplete factorization.

Key words. triangular matrices, preconditioning, iterative methods, parallel computation, sparse matrices, sparsity, numerical linear algebra

AMS subject classifications. 65F10, 65F35, 65F50, 65Y05

PII. S0895479897317788

1. Introduction.

The problem of finding the solution vector x of a linear system

$$Ax = b, \tag{1.1}$$

where A is a general sparse $n \times n$ matrix and b an n-dimensional vector, arises in many application areas. The use of an iterative method is imperative when both time and memory constraints prohibit the use of a direct solver. In practice, basic iterative schemes [13] like Jacobi, Richardson, Gauss–Seidel, and SOR, as well as Krylov subspace methods like CGS [25], BiCGSTAB [28], GMRES [24], and QMR [12], show poor convergence when they are applied without a preconditioner. Applying such a (left) preconditioner P^{-1} to the system (1.1) means mathematically that the system

$$\tilde{A}x = \tilde{b} \tag{1.2}$$

is solved, where $\tilde{A} = P^{-1}A$ and $\tilde{b} = P^{-1}b$. This preconditioner is then efficient only when the time needed to solve (1.2) (including the time needed to set up the preconditioner) is (far) less than the time needed to solve the unpreconditioned system (1.1). This can be achieved by choosing a P^{-1} that can be constructed easily and that can be applied without much extra cost. This is why P is often diagonal, triangular, or orthogonal. Incomplete factorizations have proven to be quite efficient preconditioners in a great number of cases [18]. Unfortunately those preconditioners do not allow for a large degree of parallelism usually, neither in setting up the preconditioner nor in applying the preconditioner. A method to gain parallelism in applying this type of preconditioner, i.e., triangular solves, is by explicitly inverting the triangular matrices. The big disadvantage of explicitly inverting sparse triangular matrices is that the resulting triangular matrix is not necessarily sparse, making the preconditioner more expensive; both setting up the preconditioner and applying the preconditioner require

[*]Received by the editors March 4, 1997; accepted for publication (in revised form) by E. Ng May 7, 1998; published electronically July 9, 1999.

http://www.siam.org/journals/simax/20-4/31778.html

[†]Department of Computer Science, Leiden University, P.O. Box 9512, 2300 RA Leiden, The Netherlands (arno@cs.leidenuniv.nl).

more operations. A way to overcome this problem is by making the explicit inverse sparser ("sparsification"); see [1, 9, 27, 26]. Due to this sparsification the resulting matrix is not the exact inverse of the triangular matrix but an approximation of it. In this article several ways to compute such approximate inverses of triangular matrices are proposed.

2. Inverting triangular matrices.

Without loss of generality it is assumed that the triangular matrices have a unit diagonal, as in LDU factorization. Only upper triangular matrices are considered; lower triangular matrices can be treated likewise. Each of these matrices U can be written as

$$U = I + \sum_{i=1}^{n-1} e_i u_i^T, \tag{2.1}$$

in which u_i^T contains the strictly upper triangular part of the ith row of U, so $u_i(j) = 0$ for all $j \leq i$.

U can also be expressed in product form:

$$U = \prod_{i=n-1}^{1} (I + e_i u_i^T). \tag{2.2}$$

This follows from the fact that, for $j \leq k$, $e_k u_k^T e_j u_j^T = 0$, since the jth entry of u_k $(= u_k^T e_j)$ equals zero for all $j \leq k$.

The inverses of the terms in (2.2) are easy to construct:

$$(I + e_i u_i^T)^{-1} = I - e_i u_i^T, \tag{2.3}$$

and therefore,

$$U^{-1} = \prod_{i=1}^{n-1} (I - e_i u_i^T). \tag{2.4}$$

Since U^{-1} is also upper triangular, it is also possible to write U^{-1} in the same form as (2.1):

$$U^{-1} = I + \sum_{i=1}^{n-1} e_i \hat{u}_i^T. \tag{2.5}$$

Combining this with (2.4), an expression for $\hat{u}_i^T$, the strictly upper triangular part of the ith row of U^{-1} is obtained:

$$\hat{u}_i^T = -u_i^T \prod_{j=i+1}^{n-1} (I - e_j u_j^T). \tag{2.6}$$

No data resulting from the calculation of any other $\hat{u}_j$ is needed for the calculation of $\hat{u}_i$; i.e., there are no true dependencies between the calculation of $\hat{u}_i$ and $\hat{u}_j$ $(i \neq j)$, so all $\hat{u}_i$ can be calculated in parallel. On a shared memory computer this will work fine, since the only concurrent accesses to the same memory location will be read accesses. On a distributed memory machine, where both the triangular factor U and its approximate inverse $\hat{U}$ are distributed, there is some communication overhead.

By doing one all-to-all broadcast, each processor can get its own local copy of the complete U and no further communication is required. When this approach is not feasible (because there is not enough memory with each processor to store the complete U), a common approach is to use $p - 1$ ($p =$ number of processors) steps to send the rows of U along all processors. In the first step, processors 1 to $p - 1$ send their block of rows to the next processor; processors 2 to p receive these rows and update their own rows. In step i, processors i to $p - 1$ send the block of rows of U they received in step $i - 1$ to the next processor. These rows are processed and then sent on. To accommodate for the load imbalance, p is usually larger than the number of physical processors. How to distribute the matrices in order to minimize the communication overhead and balance the computation is not trivial, but it is beyond the scope of this article. The interested reader is referred to [15, 5] and the references therein.

Algorithm 1. Sparse product algorithm.

```
for $i = 1$ to $n - 1$
    $\hat{u}_i^T = -u_i^T$
    $j$ is first nonzero position in $\hat{u}_i^T$
    while $j < n$ do
        $\alpha = -\hat{u}_i^T e_j$
        $\hat{u}_i^T = \hat{u}_i^T + \alpha u_j^T$ (sparse daxpy)
        $j$ is next nonzero position in $\hat{u}_i^T$
    endwhile
endfor
```

Algorithm 1 is a sparse algorithm to compute the rows of U^{-1}. Sparsity is used twice: j will loop over the nonzero entries only, and the vector update (termed daxpy in the algorithm) can be performed using sparse techniques.

With each product in (2.6), $\hat{u}_i$ can get more nonzero entries. This has the effect that U^{-1} is less sparse than U (for realistic problems U^{-1} is nearly a dense matrix), and therefore, apart from the fact that it might be impossible to store the explicit inverse, multiplying with U^{-1} is computationally more expensive than backward solving with U. This makes it in general cheaper (faster) to perform a backward solve instead of inverting U and perform a matrix-vector multiplication. Even when the backward solve needs to be done a lot of times, e.g., because U^{-1} is used as a preconditioner for some iterative solving procedure, it is usually not worthwhile to invert U explicitly; the gain achieved by using a (possibly parallel) matrix-vector multiplication is annihilated by the extra work caused by fill. Nevertheless, if the fill can be controlled and the sparsity preserved, explicitly inverting U can be cost effective. Of course the number of nonzeros of U^{-1} is fixed, making it impossible to design an algorithm that computes the exact inverse of U that has less nonzeros than the matrix resulting from Algorithm 1. But on the other hand the exact inverse is not really needed since U^{-1} is used only to precondition the system, and an approximate inverse of U can be just or almost as effective as a preconditioner for the system as the exact inverse. This article discusses a number of ways to come up with an approximate inverse of U. The preconditioning effectiveness is compared with that of the exact inverse, and the increase in parallel performance is examined.

3. Drop strategies. In order to preserve the sparsity of the preconditioner, an approximate inverse $\hat{U}$ of U is used. This approximate inverse of U can be calculated in several ways. All of these ways have in common that all rows can be computed in parallel. The standard approach is to compute a matrix $\hat{U}$ for which $\|I - \hat{U}U\|_F$ is small in some sense; see, e.g., [7, 6, 14]. Another approach is to truncate formula (2.6). There are a number of ways to do this, and each will result in a different approximate inverse. As with Gaussian elimination, several fill drop strategies can be used to control the "fill." In this section these strategies are discussed.

3.1. Numerical fill dropping. The numerical fill drop strategies known from incomplete factorizations can be adapted into Algorithm 1 in two ways. The first method is to drop according to the final value of an entry, i.e., on α. When α gets below a certain threshold ε, the corresponding element is dropped, and the vector update that otherwise would have been executed and could have caused more fill is not performed. Algorithm 2 uses this method of numerical dropping.

Algorithm 2. Numerical fill drop strategy I: On final value.

```
for $i = 1$ to $n - 1$
    $\hat{u}_i^T = -u_i^T$
    $j$ is first nonzero position in $\hat{u}_i^T$
    while $j < n$ do
        $\alpha = -\hat{u}_i^T e_j$
        if $|\alpha| > \varepsilon$ then
            $\hat{u}_i^T = \hat{u}_i^T + \alpha u_j^T$ (sparse daxpy)
        else
            $\hat{u}_i^T(j) = 0$
        endif
        $j$ is next nonzero position in $\hat{u}_i^T$
    endwhile
endfor
```

The second method is to drop fill as soon as it occurs, i.e., within the sparse vector update. A possible implementation of such a vector update is given in Algorithm 3. In the implementation described, this strategy is applied on top of the dropping on final value.

A problem with these numerical fill drop strategies is that there is no direct control over the number of fills. Since the values of the entries are unknown until they have been calculated, the only way the number of fills can be predicted is by computing the preconditioner. One way to obtain some kind of control is by putting a (for each row (possibly) different) restriction on the allowed number of nonzero entries, an ILUT-like adaptation (see [22]).

In Figure 3.1 it is shown what the resulting matrix looks like when applied to the incomplete Cholesky factor with no fill of the reverse Cuthill–McKee [8] reordered matrix nos3 from the Harwell–Boeing set [11].

3.2. Positional fill dropping. As with numerical fill dropping, the situation is similar to that of incomplete Gaussian elimination. There are three methods for positional fill dropping. The first method uses a predetermined set S of positions, and

Algorithm 3. Numerical fill drop strategy II: Within vector update.

```
for all nonzeros in $u_j^T$
    $k$ is column index of this nonzero
    $u$ is value of this nonzero
    $d = \alpha * u$
    if position $k$ not filled in $\hat{u}_i^T$ then
        if $|d| > \varepsilon$ then
            register this fill
            $\hat{u}_i^T(k) = d$
        endif
    else
        $\hat{u}_i^T(k) = \hat{u}_i^T(k) + d$
    endif
endfor
```

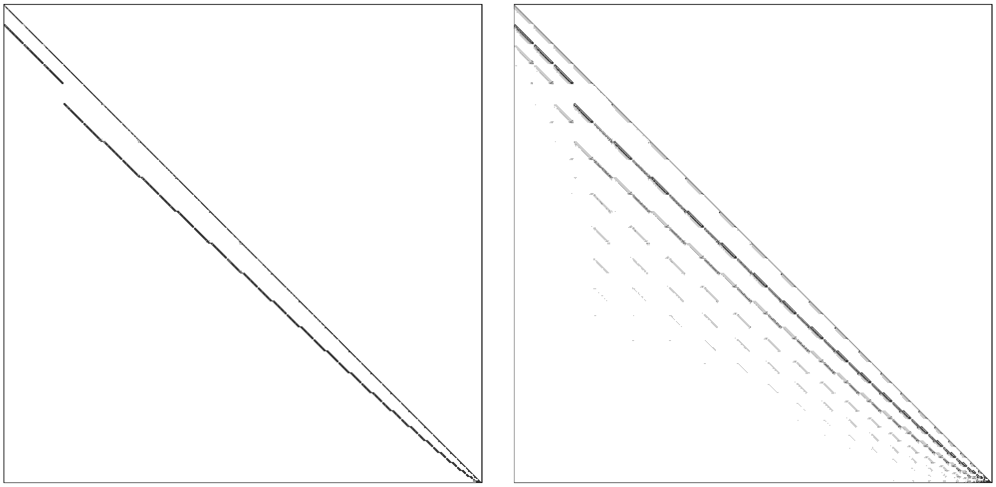


Fig. 3.1. *The Cholesky factor of nos3 and its approximate inverse with approximately twice as many nonzero elements.*

$\hat{u}_i^T(k)$ is only calculated when $(i, k) \in S$. Algorithm 4 uses this strategy. A difficulty with this strategy is how to choose the set S.

The second method reduces this problem to the choice of one parameter. It uses a rewritten form of (2.4), which can also be obtained by working out the Neumann series expansion of (2.1):

$$\prod_{i=1}^{n-1}(I - e_i u_i^T) = I - \sum_{j_1=1}^{n-1} e_{j_1} u_{j_1}^T + \sum_{j_2=1}^{n-2}\left(e_{j_2} u_{j_2}^T \sum_{j_1=j_2+1}^{n-1} e_{j_1} u_{j_1}^T \right) - \sum_{j_3=1}^{n-3}\left(e_{j_3} u_{j_3}^T \sum_{j_2=j_3+1}^{n-2}\left(e_{j_2} u_{j_2}^T \sum_{j_1=j_2+1}^{n-1} e_{j_1} u_{j_1}^T \right)\right) + \cdots . \tag{3.1}$$

Algorithm 4. Positional fill drop strategy I: On predetermined position.

```
for $i = 1$ to $n-1$
    for all $k$ for which $(i,k) \in S$ and $u_i^T(k) \neq 0$
        $\hat{u}_i^T(k) = -u_i^T(k)$
    endfor
    for all $j$ for which $(i,j) \in S$ and $\hat{u}_i^T(j) \neq 0$
        $\alpha = -\hat{u}_i^T(j)$
        for all $k$ for which $(i,k) \in S$ and $u_j^T(k) \neq 0$
            $\hat{u}_i^T(k) = \hat{u}_i^T(k) + \alpha u_j^T(k)$
        endfor
    endfor
endfor
```

By truncating (3.1), a new fill drop criterion is obtained. The first two terms of (3.1) are for free. The number of extra terms is denoted by the subscript m. The subscripted $\hat{U}$ is used to denote after which term (3.1) is truncated, so

$$\begin{aligned} \hat{U}_0 &= I - \sum_{j_1=1}^{n-1} e_{j_1} u_{j_1}^T, \\ \hat{U}_1 &= I - \sum_{j_1=1}^{n-1} e_{j_1} u_{j_1}^T + \sum_{j_2=1}^{n-2} e_{j_2} u_{j_2}^T \sum_{j_1=j_2+1}^{n-1} e_{j_1} u_{j_1}^T, \\ &\text{etc.} \end{aligned} \tag{3.2}$$

Algorithm 5 calculates $\hat{U}_m$.

A disadvantage of this truncation is now apparent: vector updates with u_k^T for the computation of $\hat{u}_i^T$ will be performed (worst case) m times. The third method accumulates these updates. This is done by using the notion of fill levels [10, 17, 19]. All fill with a fill level higher than some predefined value is dropped. The fill level of an entry is defined as follows. The positions of all nonzeros in U are defined to have fill level zero. All fill that is caused by a vector update due to an entry at a position with fill level zero are defined to have fill level one (i.e., when U has a nonzero at position (i,k) associated with $\hat{u}_i^T e_k$ in Algorithm 1, all fill that is generated due to the resulting vector update has fill level one). Likewise higher fill levels are defined. The positions of all structural zeros of $\hat{U}$ are defined to have fill level $+\infty$.

Using the definitions given above, the following initialization function is obtained:

$$level_{ij} = \begin{cases} 0 & \text{if } u_i^T(j) \neq 0, \\ +\infty & \text{if } u_i^T(j) = 0, \end{cases} \tag{3.3}$$

and the function to update the fill levels is

$$level_{ik} = \min(level_{ij} + 1, level_{ik}), \tag{3.4}$$

in which i is the index of the row vector of $\hat{U}$ that is constructed, j is the index of the vector that is added to $\hat{u}_i^T$, and k is the index within the corresponding vector update

Algorithm 5. Positional fill drop strategy II: On truncation level.

```
for $i = 1$ to $n - 1$
    $\hat{u}_i^T = -u_i^T$
    for all nonzeros $j_1$ in $u_i^T$
        $\alpha_1 = -u_i^T e_{j_1}$
        for all nonzeros $j_2$ in $u_{j_1}^T$
            $\alpha_2 = -u_{j_2}^T e_{j_2} \alpha_1$
            :
            for all nonzeros $j_m$ in $u_{j_{m-1}}^T$
                $\alpha_m = -u_{j_2}^T e_{j_2} \alpha_{m-1}$
                $\hat{u}_i^T = \hat{u}_i^T + \alpha_m u_{j_m}^T$ (sparse daxpy)
            endfor
            :
            $\hat{u}_i^T = \hat{u}_i^T + \alpha_2 u_{j_2}^T$ (sparse daxpy)
        endfor
        $\hat{u}_i^T = \hat{u}_i^T + \alpha_1 u_{j_1}^T$ (sparse daxpy)
    endfor
endfor
```

Algorithm 6. Positional fill drop strategy III: On fill level.

```
for $i = 1$ to $n - 1$
    $\hat{u}_i^T = -u_i^T$
    $j$ is first nonzero position in $\hat{u}_i^T$
    while $j < n$ do
        if $level_{ij} \leq m$ then
            $\alpha = -\hat{u}_i^T e_j$
            $\hat{u}_i^T = \hat{u}_i^T + \alpha u_j^T$ (sparse daxpy)
            update fill levels
        else
            $\hat{u}_i^T(j) = 0$
        endif
        $j$ is next nonzero position in $\hat{u}_i^T$
    endwhile
endfor
```

loop. The algorithm thus obtained is Algorithm 6. This algorithm can be viewed as a special case of Algorithm 4, where $\mathcal{S}$ is equal to the nonzero pattern of U^m.

Another variant is to use the fill levels of the incomplete factorization as initial fill levels for the approximate inverse. So the function given in (3.3) becomes

$$level_{ij} = \begin{cases} iclevel_{ij} & \text{if } u_i^T(j) \neq 0, \\ +\infty & \text{if } u_i^T(j) = 0 \end{cases} \tag{3.5}$$

with $iclevel_{ij}$ being the final fill level of the entry at position (i, j) of the incomplete factorization.

3.3. Hybrid drop strategy. The numerical drop strategies presented in section 3.1 and the positional drop strategies presented in section 3.2 can also be combined into hybrid fill drop strategies. One of the possible strategies is to use both numerical dropping on final value and fill level dropping. This leads to Algorithm 7: a combination of Algorithm 2 and Algorithm 6. Another type of hybrid drop strategies is to use any of the algorithms presented in this section to determine the positions of the nonzero elements of $\hat{U}$ and to use a standard approximate inverse algorithm to calculate the values of these nonzero elements.

ALGORITHM 7. Hybrid fill drop strategy I: On fill level and final value.

```
for $i = 1$ to $n - 1$
    $\hat{u}_i^T = -u_i^T$
    $j$ is first nonzero position in $\hat{u}_i^T$
    while $j < n$ do
        $\alpha = -\hat{u}_i^T e_j$
        if $|\alpha| > \varepsilon$ and $level_{ij} \leq m$ then
            $\hat{u}_i^T = \hat{u}_i^T + \alpha u_j^T$ (sparse daxpy)
            update fill levels
        else
            $\hat{u}_i^T(j) = 0$
        endif
        $j$ is next nonzero position in $\hat{u}_i^T$
    endwhile
endfor
```

3.4. Set-up cost. In order to assess the cost of the construction of the approximate inverse using any of the algorithms presented, a few assumptions are made. The first assumption is that each row of U has roughly the same number of nonzeros. Thus, the cost of each sparse vector update is the same. The second assumption is that this also holds for L. This assumption is needed to estimate the cost of the incomplete factorization. Let nnz_U denote the number of off-diagonal nonzeros in the matrix U. Define likewise nnz_L and $nnz_{\hat{U}}$. The third assumption is that the discarding of fill happened after it had been created. In that case each off-diagonal nonzero in L caused a sparse vector update with a row from U. This also holds for the off-diagonal nonzeros of $\hat{U}$. The cost (in terms of number of floating point operations) for the back-substitution ($C_{substitute}$), for the incomplete factorization ($C_{factorize}$), and for the approximate inversion ($C_{app.invert}$), can now be expressed as

$$\begin{aligned} C_{factorize} &= \mathcal{O}\left(nnz_L * \frac{nnz_U}{n}\right), \\ C_{app.invert} &= \mathcal{O}\left(nnz_{\hat{U}} * \frac{nnz_U}{n}\right), \\ C_{substitute} &= \mathcal{O}(nnz_U). \end{aligned}$$

The relative extra cost of calculating $\hat{U}$ compared with the cost of the incomplete factorization and the cost of back-substitution is

$$\frac{C_{app.invert}}{C_{factorize}} = \mathcal{O}\left(\frac{nnz_{\hat{U}}}{nnz_L}\right),$$
$$\frac{C_{app.invert}}{C_{substitute}} = \mathcal{O}\left(\frac{nnz_{\hat{U}}}{n}\right).$$

Under the assumption that L has about the same number of nonzeros as U, the extra cost for the (parallelizable) approximate inversion relative to the cost for the decomposition is governed by the factor of extra nonzeros in $\hat{U}$ compared with U.

3.5. Asymmetry. The drop strategies presented have in common that they can be used both in a column-wise manner and in a row-wise manner. In general, the resulting preconditioner will not be the same. To illustrate this, consider the following matrix and its column-wise and row-wise approximate inverses, where the entry at position (2,4) is dropped:

$$\begin{pmatrix} 1 & a & b & c \\ 0 & 1 & d & 0 \\ 0 & 0 & 1 & e \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \begin{array}{l} \overset{\text{column-wise}}{\rightsquigarrow} \begin{pmatrix} 1 & -a & -b+ad & -c+be \\ 0 & 1 & -d & 0 \\ 0 & 0 & 1 & -e \\ 0 & 0 & 0 & 1 \end{pmatrix} \\ \overset{\text{row-wise}}{\rightsquigarrow} \begin{pmatrix} 1 & -a & -b+ad & -c+be-ade \\ 0 & 1 & -d & 0 \\ 0 & 0 & 1 & -e \\ 0 & 0 & 0 & 1 \end{pmatrix}. \end{array}$$

For fill level dropping, the pattern but not the values for the row-wise and column-wise approximate inverse will be the same. When one of the numerical dropping strategies is applied, even the patterns can be different. This can be seen from the example above: since the elements at position (1,4) differ, $-c + be$ can be below the threshold while $-c + be - ade$ is not, or vice versa.

In the experiments presented in the following sections, the row-wise algorithms were used.

4. Numerical results. In this section the algorithms from section 3 are tested on the Poisson problem on a unit square, discretized on a 32×32 grid. An incomplete Cholesky decomposition using fill levels 0, 1, 2, 3, and 4 was calculated and used to precondition the system. The effectiveness of this preconditioner is given in Table 4.1. For these experiments the right-hand side was generated using a constant one solution vector, and the initial guess equaled the zero vector. Convergence was reached when the 2-norm of the residual was a factor 10^9 smaller than the 2-norm of the initial residual.

The inverse of the preconditioner was approximated using each of the described algorithms. The approximate inverse is denoted with the term ICI. The results are presented in the next subsections.

4.1. Preconditioning efficiency. As an example of the standard approximate inverse approach the SPAI algorithm [14] is used. In SPAI there are three parameters: (a) $\varepsilon =$ the desired accuracy of ICI (i.e., $\|e_i - \hat{U}Ue_i\|_2 < \varepsilon$); (b) `nmax`, the maximum number of elements in each row of ICI; and (c) `nadd`, the maximum number of elements added to the sparsity pattern before a new least squares problem is solved. In the

Table 4.1

Number of CG-iterations needed for convergence and number of nonzeros of the incomplete Cholesky factorization as a function of the fill-level for the test problem.

level	#iter	nnz
0	32	3008
1	24	3969
2	20	4899
3	15	6728
4	12	8495

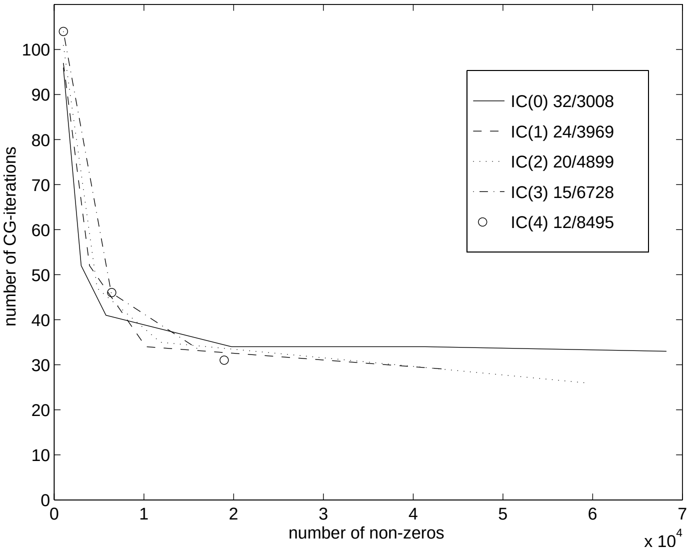


Fig. 4.1. *Preconditioning efficiency of the SPAI approximate inverse of various decompositions.*

experiments, only ε is varied; `nmax` (100) and `nadd` (1) are kept constant (for larger problems a larger value for `nadd` (e.g., 10) is more efficient [14]).

In Figure 4.1 the number of iterations needed for CG to converge as a function of the number of nonzeros in the SPAI approximate inverse of the various incomplete Cholesky decompositions is presented. The number of nonzeros grows with the desired accuracy of the approximate inverse. The results for IC(0) (for which U has 3008 nonzeros and needs 32 CG-iterations for convergence) are also presented in Table 4.2. What can be seen from Figure 4.1 is that the gain in preconditioning efficiency obtained due to higher fill level in the Cholesky decomposition cannot be achieved with the SPAI approximate inverse of that decomposition unless a large number of nonzeros in the approximate inverse is allowed.

Next, the two methods of numerical dropping described in section 3.1 were applied to the test problem. The thresholds used are 0.9, 0.5, 0.2, 0.1, 1e-2, 1e-3, and 1e-5. The results are depicted in Figure 4.2. For this problem, dropping on final value is better than dropping on intermediate value. Third, the two variants of positional dropping of Algorithm 6 were applied to the test problem, and the results are depicted in Figure 4.3. The other two strategies (Algorithms 4 and 5) were not tested.

Finally the result of the experiments with the hybrid dropping of Algorithm 7 on the test problem are presented in Figure 4.4. Only the results for approximating the inverse of IC(2) are presented. The approximations of the other decompositions show

TABLE 4.2
Number of CG-iterations and number of nonzeros of the SPAI approximate inverse of IC(0) for several thresholds.

ε	#it.	#nnz
1.0	96	1024
0.5	96	1024
0.2	52	3008
0.1	41	5769
0.01	34	19754
0.001	34	41308
0.0001	33	68228

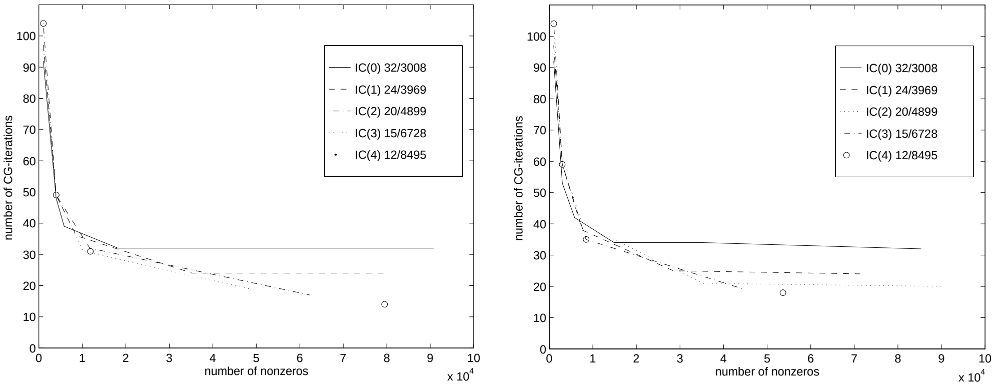


FIG. 4.2. *Preconditioning efficiency of the approximate inverse with numerical dropping (left: final value, right: intermediate value) for various decompositions.*

a similar behavior. The results of the experiments with the second hybrid dropping algorithm show that given the sparsity pattern generated with the positional dropping strategy using fill levels (Algorithm 6), a better approximation of the inverse of the decomposition does not necessarily result in a better preconditioner.

These experiments show that, for higher levels, relatively more nonzeros are needed in order to come close to the preconditioning efficiency of the triangular solve. So, the gain in preconditioning efficiency relative to the number of nonzeros is better for the triangular solve than it is for its approximate inverse.

4.2. Comparing the approximate inverses. In Figure 4.5 the results for the test problem preconditioned with the different approximations of IC(2) are plotted. For this problem the numerical dropping on final value strategy gives the best precondition efficiency/number of nonzeros ratio (the lowest line in the graph).

The preconditioner was also tested on some symmetric matrices from the Harwell–Boeing set. The algorithm used is the ILUT-like extended version of Algorithm 2 ICI(F,ε), with fixed threshold parameter $\varepsilon = 0.01$ and various values for the maximum number of nonzeros (F is the ratio between the number of nonzeros in the approximate inverse and the Cholesky factor). The results are presented in Table 4.3. With this threshold the convergence rate within two iterations of that of IC(0) could be reached for all matrices, albeit that for nos3 ICI(∞,0.01) requires 16.5 times as much nonzeros as the Cholesky factor.

For problems of considerably larger size, it is probably worthwhile to view the triangular matrices as a block triangular matrix and only invert the diagonal blocks

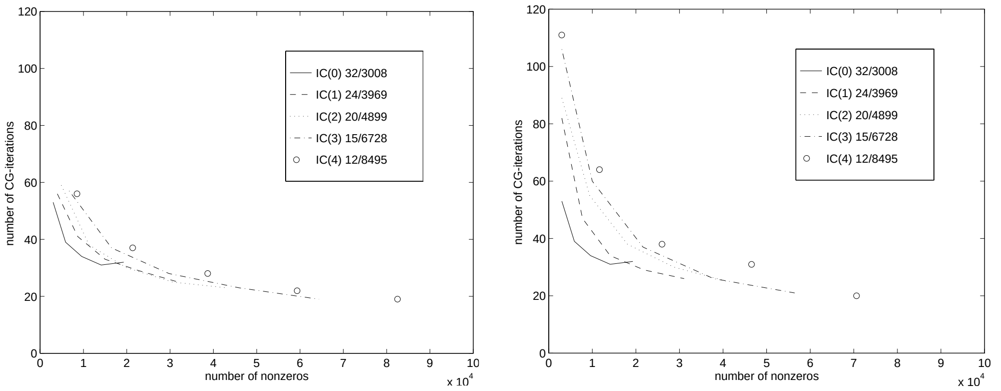


FIG. 4.3. *Preconditioning efficiency of the approximate inverse with positional dropping (left: new levels, right: passed levels) for various decompositions.*

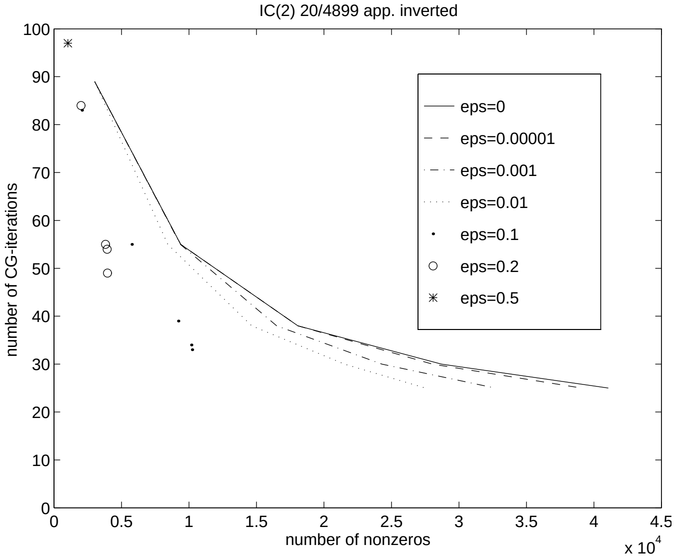


FIG. 4.4. *Preconditioning efficiency of the approximate inverse of* IC(2) *with hybrid dropping.*

approximately. This is also suggested in [20], where it is shown that using the exact inverse of the diagonal blocks can already result in a performance gain of a factor three on an Alliant FX/8 with 8 processors, but was less effective than level scheduling on that platform. However, as long as both the diagonal blocks and the off-diagonal blocks contain enough nonzeros for the associated matrix-vector multiplications to be performed efficiently, approximately inverting only the diagonal blocks instead of the whole triangular factor will not have a large degrading effect on the obtainable speed-up. In this case, the application of the preconditioner consists of a series of sequential steps, which are themselves easily parallelizable.

4.3. Comparing triangular solve with matrix-vector multiplication. An important aspect of the preconditioner other than its efficiency in decreasing the number of iterations needed for convergence is the time needed to apply the preconditioner.

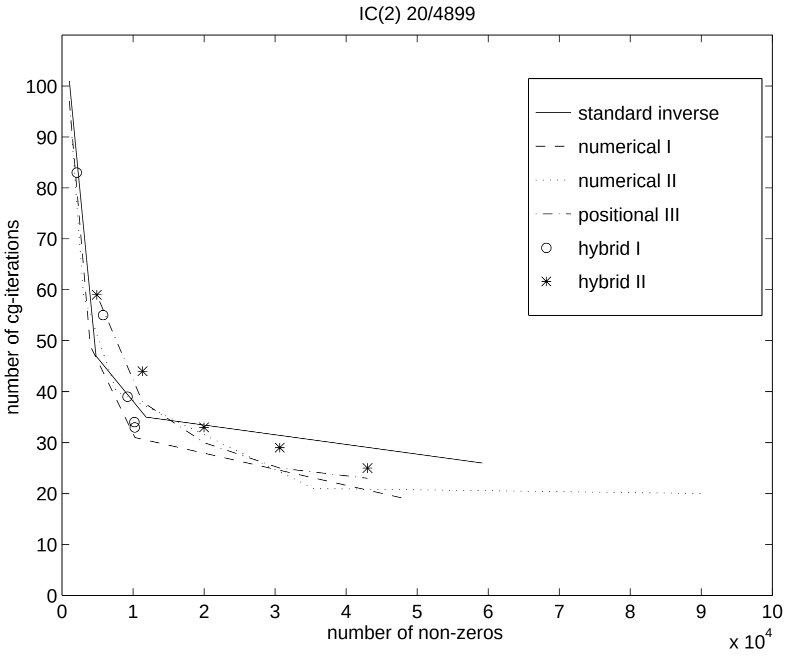


FIG. 4.5. *Preconditioning efficiency of the approximate inverses of IC*(2).

TABLE 4.3

Number of CG-iterations needed for convergence for ICI with numerical dropping ($\varepsilon = 0.01$) *and limits on the maximum number of nonzeros.*

Matrix	IC(0)	ICI(F, 0.01)					ICI(∞, 0.01)	
		F=1.0	F=1.5	F=2.0	F=3.0	F=4.0	Iter.	F
494_bus	41	‡	‡	252	89	53	43	5.3
662_bus	50	161	98	70	61	54	50	6.4
685_bus	62	130	85	71	62	63	62	5.9
1138_bus	76	‡	‡	‡	240	162	78	6.4
gr_30_30	25	42	38	32	30	28	25	7.8
lund_a	17	134	132	115	83	63	17	7.1
lund_b	10	63	48	42	30	23	10	6.0
nos3	46	79	70	63	57	52	45	16.5
nos4	23	52	43	36	30	26	23	4.9
nos5	47	86	58	53	48	48	47	8.9
nos6	29	41	36	31	30	29	29	3.9
nos7	32	47	42	38	34	33	30	7.6
poisson	32	53	48	39	42	40	32	6.0

‡ means no convergence within 300 iterations. The last column specifies the number of iterations when ICI with no limit on the number of nonzeros would be used: when F=7.0, ICI with no limit would require 7.0 times the number of nonzeros of IC(0)

Therefore the time needed to do a triangular solve is compared with the time needed to multiply a vector with a triangular matrix on several computing platforms: an HP700/9000 workstation, a CRAY-C90, and a CM-5. The triangular matrices used were randomly generated with dimension and number of nonzeros per row as driving parameters. The dimension was varied from 100 to 25600 and the average number of off-diagonal nonzeros from 1 to 64.

Timings for the HP workstation are presented in Figure 4.6. Since the number of

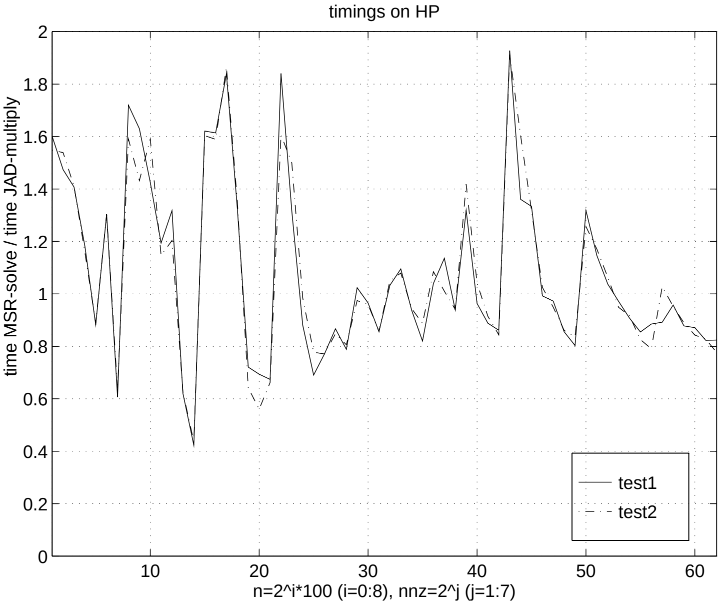


FIG. 4.6. *Timings to do a triangular solve relative to the time needed for matrix-vector multiplication on an HP workstation.*

floating point operations is the same, one would expect that neither one of the two would be clearly better. The experiments show that both methods outrun each other in about half of the cases. The subroutines used were AMUXJ and LDSOL from SPARSKIT2 [21].

For the CRAY-C90, timings are presented in Figure 4.7. Due to the jagged diagonal storage, the matrix vector multiply can make better use of the vector capabilities than the triangular solve. One would expect that this difference would decrease with growing number of nonzeros per row, but in these experiments this effect is not apparent even for 64 nonzeros per row.

In order to be able to specify beforehand how many nonzeros are allowed in the approximate inverse, given the dimension of the matrix, the number of nonzeros in the incomplete factorization, and the speed-up desired, the time needed to perform a matrix vector multiplication for a given dimension is modeled by a linear function of the number of nonzeros. From Figure 4.7 this seems to be a reasonable assumption. The same is assumed for the triangular solve with the average number of nonzeros per row being larger than 10. Let t_{JAD} denote the time needed to perform one matrix vector multiplication, and z_{JAD} the number of nonzeros per row in the approximate inverse. This leads to the following function:

$$t_{\mathrm{JAD}}(n, z_{\mathrm{JAD}}) = a_{\mathrm{JAD}}(n) z_{\mathrm{JAD}} + b_{\mathrm{JAD}}(n),$$

where $a_{\mathrm{JAD}}(n)$ and $b_{\mathrm{JAD}}(n)$ are the model parameters sought.

Likewise for the triangular solve:

$$t_{\mathrm{MSR}}(n, z_{\mathrm{MSR}}) = a_{\mathrm{MSR}}(n) z_{\mathrm{MSR}} + b_{\mathrm{MSR}}(n).$$

The least squares solution of the coefficients as a function of n is given in Figure 4.8. Again linear functions are used as a model for these coefficients, using least

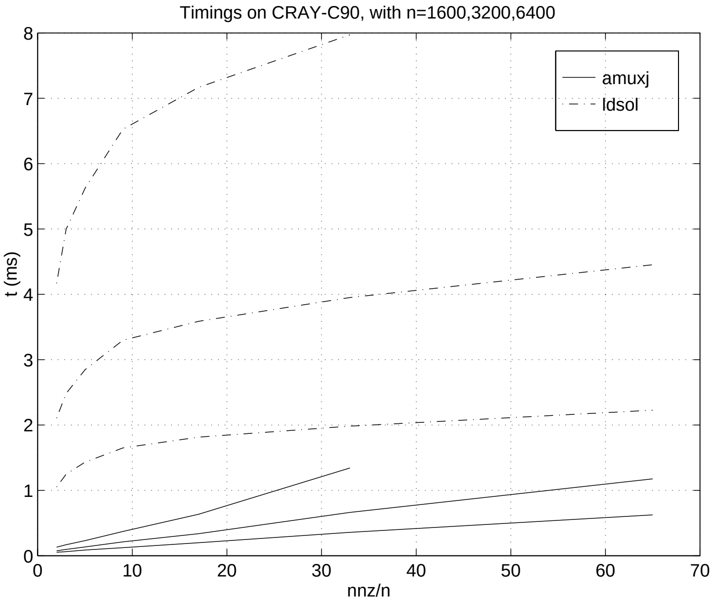


FIG. 4.7. *Timings to do a triangular solve compared to the time needed for matrix-vector multiplication on a CRAY-C*90.

squares to find the coefficients. With the found parameters, it is now possible to give an estimate for the number of nonzeros that can be allowed in the approximate inverse.

Let f be the desired speed-up factor that one wants to achieve on the CRAY-C90; then setting $ft_{\text{JAD}} = t_{\text{MSR}}$ leads to

$$F(n, f, z_{\text{MSR}}) = \frac{z_{\text{JAD}}}{z_{\text{MSR}}} = \frac{a_{\text{MSR}}(n)}{f \cdot a_{\text{JAD}}(n)} + \frac{b_{\text{MSR}}(n)}{f \cdot a_{\text{JAD}}(n) z_{\text{MSR}}} - \frac{b_{\text{JAD}}(n)}{a_{\text{JAD}}(n) z_{\text{MSR}}}.$$

For $n > 1000$, F is independent of n. In Figure 4.9, F is plotted as a function of f and z_{MSR}. In the previous section it was shown that the number of nonzeros needed to get close to the preconditioning efficiency of the incomplete factorization is at least a factor of two or three larger than the number of nonzeros in the incomplete factor. From this figure it can be seen that if the original factor contains 20 nonzeros per row and the desired speed-up is a factor three, then the approximate inverse may not contain more than three times as many nonzeros.

For the implementation on the CM-5, the global execution model as provided by CM Fortran was used. For the sparse matrix vector multiply a built-in subroutine, SPARSE_MATVEC_MULT, was available. For the triangular solve, the routine LDSOL was converted to CM Fortran. The timings are presented in Table 4.4. The CM-5 used had 1024 processors of which 512 were used for these experiments.

Although more efficient implementations of triangular solve on a CM-5 probably exist, the speed-up obtainable by, for instance, level scheduling is usually in order of 4 to 8, which is quite modest with respect to the number of processors. On this type of machine the excellent scalability of the approximate inverse preconditioner clearly pays off.

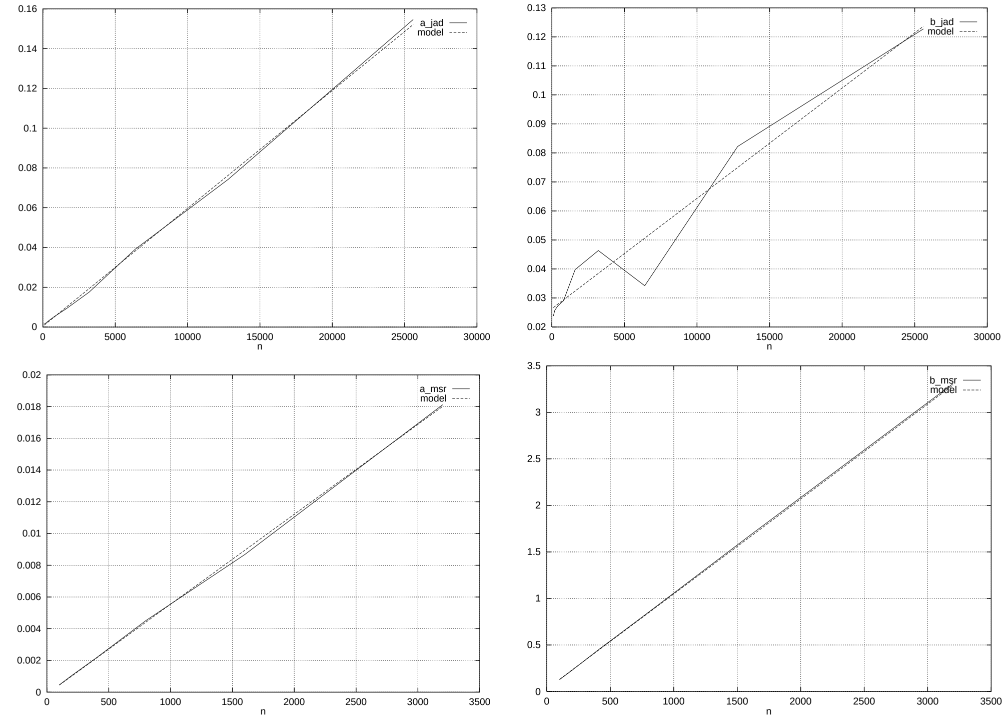


Fig. 4.8. *The values of the coefficients and the linear functions used as a model.*

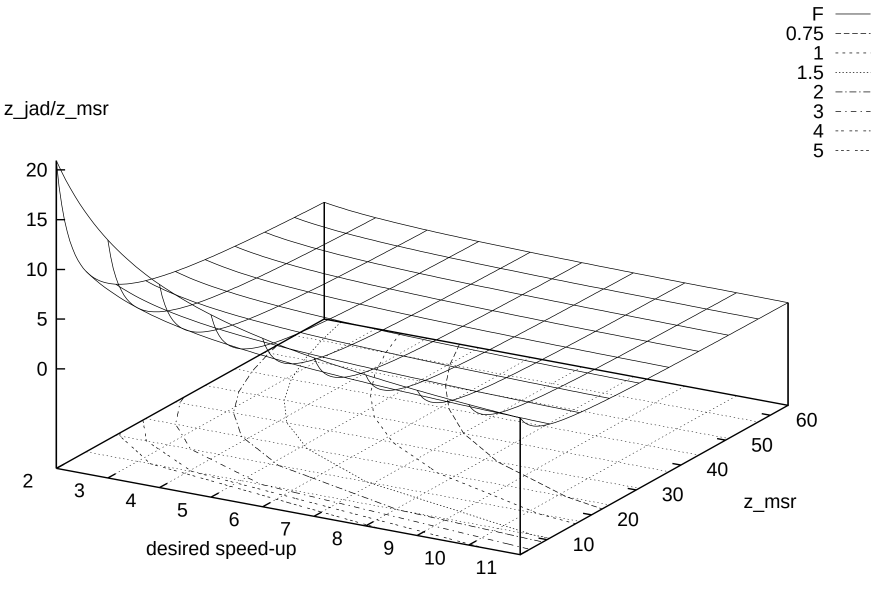


Fig. 4.9. *The factor of allowed nonzeros in the approximate inverse as a function of the number of nonzeros in the incomplete factorization and the desired speed-up of the matrix-vector multiplication over the triangular solve.*

Table 4.4

Timings on a CM-5 with 512 processors for sparse matrix-vector multiplication, inner product wise triangular solve, and vector update-wise triangular solve.

n	nnz/n	multiply	solve (dot)	solve (daxpy)
800	2	0.003	0.443	0.134
	4	0.003	0.445	0.197
	8	0.004	0.452	0.335
	16	0.005	0.474	0.604
	32	0.007	0.541	1.138
	64	0.011	0.677	2.226
1600	2	0.003	0.886	0.260
	4	0.003	0.892	0.396
	8	0.005	0.926	0.661
	16	0.006	1.032	1.199
	32	0.009	1.259	2.286
	64	0.014	1.624	4.436
3200	2	0.004	1.818	0.524
	4	0.004	1.829	0.786
	8	0.006	2.022	1.327
	16	0.008	2.417	2.417
	32	0.012	3.052	4.565
	64	0.021	4.469	8.899
6400	2	0.005	3.883	1.049
	4	0.006	5.138	1.613
	8	0.006	5.535	2.708
	16	0.008	6.285	4.904
	32	0.013	7.661	9.284
	64	0.021	10.399	18.198

5. Related work.

In his thesis [9], Dağ investigates the partitioned inverse of the triangular matrix

$$U^{-1} = \prod_{i=1}^{n} (I - e_i u_i^T) = \prod_{k=1}^{K} U_k, \tag{5.1}$$

where U_k is the product of a set of consecutive elementary matrices $U_k = (I - e_i u_i^T) \ldots (I - e_j u_j^T)$ such that there is no or little fill-in. When U is not so sparse, e.g., when U stems from a complete factorization, the number of terms in (5.1) can be quite moderate (see [2]). Unfortunately this is not true when U is as sparse as it is when an incomplete factorization was used. Three algorithms for discarding fill ("sparsification") are presented: (1) numerical dropping on exact value ($SW^x ILU_i$), (2) positional dropping on fill-level ($IW_j ILU_i$), (3) numerical dropping on exact value after fill-level dropping ($SIW_j^x ILU_i$). $IW_j ILU_i$ with one partition is equal to Algorithm 6. $SW^x ILU_i$ is similar to Algorithm 2; likewise $SIW_j^x ILU_i$ is similar to Algorithm 7. The difference is that $SW^x ILU_i$ and $SIW_j^x ILU_i$ drop elements using a threshold after all elements (of one column or row) are calculated, thus introducing more work than Algorithms 2 and 7. The resulting approximate inverse ($\tilde{U}$) is closer to the exact inverse in the sense that $\|U^{-1} - \tilde{U}\|_F \leq \|U^{-1} - \hat{U}\|_F$, but this does not imply that $\|I - U\tilde{U}\|_F \leq \|I - U\hat{U}\|_F$.

The resulting preconditioners are tested on a shared memory machine (a Sequent Symmetry). The timings show that using multiple partitions ($K > 1$) reduces the required CPU time. This is due to the effect that for $K > 1$ the preconditioner is a better approximation of U^{-1}, while at the same time full use can be made (still) of the 14 available processors.

Another approach is to calculate a factored approximate inverse directly from A instead of doing some kind of incomplete factorization first. Kolotilina and Yeremin [16] calculate an approximate inverse with fixed sparsity pattern of the complete Cholesky factor L without actually calculating L. Benzi, Meyer, and Tůma [3] and Benzi and Tůma [4] perform an incomplete (bi)conjugation process resulting in a preconditioner that is largely independent of the ordering of the matrix [4]. The incomplete biconjugation process contains less parallelism than in the algorithms proposed in this article, but so does the incomplete factorization. Their algorithm stays within reasonable limits from the incomplete factorization time with on average equal preconditioning performance with the same number of nonzeros. More nonzeros, however, does not necessarily mean better or equal convergence with their method.

6. Conclusions and future work. Ideally one would like to use the exact inverse as a preconditioner. In this way one can calculate the influence of the right-hand side to the solution at all nodes in one step. Unfortunately it is too expensive to calculate the exact inverse, and it is also too expensive to store the exact inverse. Instead of using the exact inverse, a complete factorization can be used. The resulting factors can be viewed as sparse indirect representation of the exact inverse. Although this is much cheaper than calculating the exact inverse, it is (usually) still too costly to be feasible. There are two ways to save on computing time and storage: (i) calculate a sparse approximation of the exact inverse and (ii) calculate sparse approximations of the factors. The first way leads to an explicit preconditioner, the second to an incomplete decomposition. If one calculates the explicit preconditioner that corresponds to the incomplete factorization, one notices that the number of nonzeros is much larger than the sum of the number of nonzeros of the factors. For this reason one expects the incomplete factorization to yield a better preconditioner than a sparse approximate inverse with the same sparsity. For tridiagonal matrices this is more than true, since the incomplete factorization is exact, whereas the approximate inverse is not. For some application areas, however, sparse approximations can be competitive preconditioners; see, e.g., [23].

The method proposed in this article combines the advantages of being an explicit preconditioner with the preconditioning efficiency of the underlying incomplete factorization. The algorithms developed show how the (mainly) sequential process involved with the triangular solve steps of an incomplete factorization can be turned into a scalable parallel process by using the sparse approximate inverse of the triangular factors. The algorithms have in common with the Frobenius norm approach that all columns (or rows) of the preconditioner can be calculated in parallel. The amount of work associated with each algorithm strongly depends on the sparsity of the triangular matrix, but the experiments show that a preconditioner efficiency close to that of the underlying incomplete factorization can be achieved.

The method can be extended in a way similar to that used by [9] to come up with products of sparse triangular matrices. In that case the aim would be to use as many matrices as possible such that each matrix gives rise to a scalable parallel matrix vector multiplication and at the same time the total number of floating point operations does not grow (much). In this way it is possible to approximate the inverse of the factor better with the same sparsity.

The assumption that an incomplete factorization has already been computed leads to several disadvantages of this method. The first disadvantage is that only the application and part of the setting up of the preconditioner contains scalable parallelism. When the time required for calculating the incomplete factorization is relatively small, e.g., because it can be used many times, this method remains applicable. A major disadvantage is that if the incomplete factorization does not exist or is unstable, this method is simply not applicable. A third disadvantage is that, apart from the parameters for the incomplete factorization, parameters for the approximate inversion need to be chosen. Finding the optimal parameters might turn out to be hard. Another disadvantage is that it is very unlikely that the resulting preconditioner will do a better job in reducing the number of iterations than the underlying incomplete factorization. Therefore, the challenge will be to find an explicit preconditioner in the form of products of sparse matrices, such that the combined sparsity stays within certain limits and each matrix contains enough nonzeros to make efficient use of the parallel processing capability when used in a matrix vector product, while at the same time each of these matrices can be computed in parallel using the original matrix A instead of some (hard to parallelize) incomplete factorization.

Acknowledgments. The author would like to thank Henk van der Vorst and Harry Wijshoff for their helpful comments and suggestions.

REFERENCES

[1] F. Alvarado and H. Dağ, *Sparsified and incomplete sparse factored inverse preconditioners*, in Copper Mountain Conference on Iterative Methods, Preliminary Proceedings, Vol. I, April 9–14, 1992.

[2] F. L. Alvarado and R. Schreiber, *Optimal parallel solution of sparse triangular systems*, SIAM J. Sci. Comput., 14 (1993), pp. 446–460.

[3] M. Benzi, C. D. Meyer, and M. Tůma, *A sparse approximate inverse preconditioner for the conjugate gradient method*, SIAM J. Sci. Comput., 17 (1996), pp. 1135–1149.

[4] M. Benzi and M. Tůma, *A sparse approximate inverse preconditioner for nonsymmetric linear systems*, SIAM J. Sci. Comput., 19 (1998), pp. 968–994.

[5] R. H. Bisseling, *Sparse matrix computations on bulk synchronous parallel computers*, in Proceedings ICIAM'95, Issue 1. G. Alefeld, O. Mahrenholtz, and R. Mennicken, eds., Numerical Analysis, Scientific Computing, Computer Science, Akademie Verlag, Berlin, 1996, pp. 127–130.

[6] E. Chow and Y. Saad, *Approximate inverse techniques for block-partitioned matrices*, SIAM J. Sci. Comput., 18 (1997), pp. 1657–1675.

[7] E. Chow and Y. Saad, *Approximate inverse preconditioners via sparse-sparse iterations*, SIAM J. Sci. Comput., 19 (1998), pp. 995–1023.

[8] E. Cuthill and J. McKee, *Reducing the bandwidth of sparse symmetric matrices*, in Proceedings 24th National Conference, Association for Computing Machinery, Brandon Press, Princeton, NJ, 1969, pp. 157–172.

[9] H. Dağ, *Iterative Methods and Parallel Computation for Power Systems*, Ph.D. thesis, University of Wisconsin-Madison, December 1995.

[10] E. F. d'Azevedo, P. A. Forsyth, and W. P. Tang, *Towards a cost-effective ILU preconditioner with high level fill*, BIT, 32 (1992), pp. 442–463.

[11] I. S. Duff, R. G. Grimes, and J. C. Lewis, *Sparse matrix test problems*, ACM Trans. Math. Software, 15 (1989), pp. 1–14.

[12] R. Freund and N. Nachtigal, *QMR: A quasi-minimal residual method for non-Hermitian linear systems*, Numer. Math., 60 (1991), pp. 315–339.

[13] G. H. Golub and C. F. van Loan, *Matrix Computations*, The Johns Hopkins University Press, London, 3rd edition, 1996.

[14] M. J. Grote and T. Huckle, *Parallel preconditioning with sparse approximate inverses*, SIAM J. Sci. Comput., 18 (1997), pp. 838–853.

[15] M. T. Heath, E. Ng, and B. W. Peyton, *Parallel algorithms for sparse linear systems*, SIAM Rev., 33 (1991), pp. 420–460.

[16] L. Yu. Kolotilina and A. Yu. Yeremin, *Factorized sparse approximate inverse preconditionings* I. *Theory*, SIAM J. Matrix Anal. Appl., 14 (1993), pp. 45–58.

[17] 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.

[18] J. A. Meijerink and H. A. 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.

[19] D. J. Rose, *A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations*, in Graph Theory and Computing, R. C. Read, ed., Academic Press, New York, London, 1972, pp. 183–217.

[20] Y. Saad, *Krylov subspace methods on supercomputers*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 1200–1232.

[21] Y. Saad, *SPARSKIT: A Basic Tool Kit for Sparse Matrix Computations*, Technical report TR-1029, CSRD, University of Illinois at Urbana-Champaign, Urbana, IL, 1990.

[22] Y. Saad, *ILUT: A dual threshold incomplete LU-factorization*, Numer. Linear Algebra Appl., 1 (1994), pp. 387–402.

[23] Y. Saad, *Preconditioned Krylov subspace methods for CFD applications*, in Solution Techniques for Large-Scale CFD Problems, W. G. Habashi, ed., Wiley, New York, 1995, pp. 139–158.

[24] Y. Saad and M. H. Schultz, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[25] P. Sonneveld, *CGS, A fast Lanczos-type solver for nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 10 (1989), pp. 36–52.

[26] R. Suda, *Large scale circuit analysis by preconditioned relaxation methods*, in Proc. PCG'94, Keio University, 1994, pp. 189–205.

[27] R. Suda, *New Iterative Solvers for Parallel Circuit Simulation*, Ph.D. thesis, Department of Information Sciences, University of Tokyo, Tokyo, Japan, 1996.

[28] H. A. van der Vorst, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 631–644.

SIAM J. Matrix Anal. Appl.
Vol. 20, No. 4, pp. 1007–1026

MATRIX RENUMBERING ILU: AN EFFECTIVE ALGEBRAIC MULTILEVEL ILU PRECONDITIONER FOR SPARSE MATRICES[*]

E. F. F. BOTTA[†] AND F. W. WUBS[†]

Abstract. In this paper a multilevel-like ILU preconditioner is introduced. The ILU factorization generates its own ordering during the elimination process. Both ordering and dropping depend on the size of the entries. The method can handle structured and unstructured problems. Results are presented for some important classes of matrices and for several well-known test examples. The results illustrate the efficiency of the method and show in several cases near grid independent convergence.

Key words. multilevel methods, preconditioning, ILU, dropping strategies, Krylov-subspace methods

AMS subject classifications. 65F10, 65N06

PII. S0895479897319301

1. Introduction.

Solving large sparse systems of equations continues to be a major research area. This attention is caused by the fact that solving such equations forms the bottleneck in many practical problems. For really large systems direct methods become too expensive in CPU time and storage requirements, and therefore an iterative approach is needed. In particular the use of preconditioned CG-type methods has proved to be very competitive. It is also widely recognized that the quality of the preconditioner determines the success of the iterative method. With a proper preconditioner the choice of the CG-like accelerator is not that critical.

The preconditioner presented in this paper is a special multilevel-like incomplete factorization. In this introduction we briefly describe the various incomplete decomposition approaches available today and their relation to the approach presented here.

The history of ILU factorizations is amongst others described in [15]. Moreover, historical notes are to be found in the textbooks of Axelsson [1], Hackbusch [31], and Saad [50]. The first roots of the approach lie in the 1960s [13, 42, 43] and since then the method has become applicable to a wide class of problems. Furthermore, analyses for important classes of matrices could be made. Today, ILU factorizations are an important tool for solving large-scale problems.

Classical ILU approach. The classical approach is to allow only fill entries in the L and U factors, where the original matrix A has nonzeros. This simple approach allows for a very efficient implementation by Eisenstat [25] and is still very popular.

As observed by Dupont, Kendall, and Rachford [23], an important improvement in the convergence of the classical approach can be obtained by lumping the dropped elements onto the diagonal. With this modification, the factorization, called MILU, is made exact for a constant vector. For a more general matrix A Gustafsson [30] found a similar result. For many second-order elliptic problems, the preconditioning with the classical ILU gives asymptotically the same condition number as with diagonal scaling, i.e., $O(h^{-2})$. After this simple modification this improves to $O(h^{-1})$. For M-matrices the existence of ILU factorizations can be proved [35], but this is not the case

[*] Received by the editors March 31, 1997; accepted for publication (in revised form) by R. Freund January 19, 1998; published electronically July 9, 1999.
http://www.siam.org/journals/simax/20-4/31930.html

[†] Department of Mathematics, University of Groningen, P.O. Box 800, 9700 AV Groningen, The Netherlands (E.F.F.Botta@math.rug.nl, F.W.Wubs@math.rug.nl).

for MILU factorizations. Here, the factorization may break down [24]. The subject of existence is studied by Beauwens and Quenon [5] and Notay [38, 39]. Relaxed forms of MILU have been introduced to prevent the problem of breakdown and bad conditioning [40]. Problems occur at positions in the matrix where due to lumping the diagonal becomes very small.

The classical approach is an example of *drop-by-position*, which can be generalized by also allowing fill at other user-specified positions in the L and U factors. In general it is difficult to determine where to allow fill. Hence, matrix-dependent approaches are to be favored.

Matrix-dependent approaches. One may distinguish two approaches: one in which only the nonzero structure of the original matrix determines the fill pattern, and another in which the values of the coefficients also are involved.

An example of the first approach is the one based on the level of fill. The levels are defined recursively. Entries belonging to the original nonzero structure of the matrix are defined to have a level of fill of zero. Fills in the LU factors caused by entries of level k have level $k + 1$. For many problems the size of elements decreases with the level number and in practice the number of levels is kept low (see [50, section 10.3.3]). This approach appears to be rather successful [17]. However, there can still be a lot of fill of which most entries are very small.

In the second approach, called *drop-by-size* method (or incomplete factorization by value, e.g., [1]) such small elements are dropped according to a dropping rule. Let us briefly review some of the drop-by-size strategies.

In the ILU method, proposed by Saad [50, section 10.4.1], the factors L and U are constructed row by row, with L unit lower triangular. When constructing row i, first the 2-norm of the ith row of the original matrix is calculated. Multiplied by a user specified tolerance, this is used as a drop tolerance for this row and the multipliers used in the construction of this row. The fill in L and U is further limited by keeping, besides the diagonal, only the p largest elements in the L and U parts of the row.

Axelsson and Munksgaard [3] and Axelsson [1] follow the standard construction of the L and U factors. Elements in the part of the matrix still to be factorized at step k of the construction are dropped if

$$|a_{ij}^{(k)}| \leq \varepsilon |a_{ii}^{(k)} a_{jj}^{(k)}|^{1/2}.$$

For block matrices the dropping condition is generalized to

$$||A_{ij}^{(k)}|| \leq \varepsilon \{||A_{ii}^{(k)}|| ||A_{jj}^{(k)}||\}^{1/2}.$$

This approach is strongly based on symmetric positive definite systems.

D'Azevedo, Forsyth, and Tang [18] base the dropping rule on the maximum values in rows i and j of the original matrix, i.e.,

$$|a_{ij}^{(k)}| < \varepsilon \min(||a_{i*}||_\infty, ||a_{j*}||_\infty).$$

This condition is applied successfully to nonsymmetric problems. Nevertheless, one should be careful when the maximum does not occur at the diagonal position and as a result large multipliers may come across.

Ordering strategies. In any (I)LU factorization the ordering may have a significant impact on the amount of fill. Finding an optimal ordering is difficult (NP-hard), but over the years several successful heuristics have evolved. Some well-known examples are (reverse) Cuthill–McKee, minimum degree, and nested dissection (see, e.g., [21]). For a variety of orderings the effect on preconditioned CGs has been studied by Duff and Meurant [22]. It turns out that orderings can have a dramatic influence on the convergence and that the norm of the residual matrix $R = A - LU$ is useful as an estimate for the quality of the preconditioning. It appears that for ILU(0) the simple row ordering and the spiral ordering perform very well.

When more drop-tolerance controlled fill is allowed, the situation is different. Now the red-black ordering and the alternating diagonal ordering are very competitive. In the matrix renumbering ILU (MRILU) method, described in section 2, this type of ordering is generated automatically for each new Schur complement during the elimination process.

In D'Azevedo, Forsyth, and Tang [18] a near-optimal ordering is constructed during the elimination process. The basic idea is to choose in each step of a level-of-fill approach the next pivot node that minimizes the Frobenius norm of a discarded fill matrix. Hence in some sense the norm of the residual matrix is minimized. For ILU(0) on a standard Laplace problem this leads to an ordering very similar to the spiral ordering and with ILU(1) a generalized red-black ordering evolves. This agrees nicely with the results of Duff and Meurant [22], who also show that in case of anisotropy the ordering becomes even more critical. In such cases the method designed by D'Azevedo, Forsyth, and Tang constructs an ordering that is different from the isotropic case, leading to a better convergence. An interesting analysis for modified block incomplete factorizations is given by Magolu [36]

Repeated red-black orderings have been studied extensively. Brand [12], and separately Axelsson and Eijkhout [2], analyzed this approach. Notay and Ould Amar [41] showed recently a bound for the condition number of the preconditioned matrix $O(N^{0.153})$, where N is the total number of unknowns. Hence the condition number comes close to the optimal value one. In [8] we considered a fixed red-black ordering and combined this with a dropping strategy. It appeared that grid-independent convergence is possible if used in a CG process.

MRILU as presented in this paper determines the ordering by itself. At every block step in the process we determine an independent set, i.e., a set of unknowns not directly connected in the graph. The unknowns associated with an independent set can be eliminated simultaneously. This concept is stretched to the case of a weak coupling. This is very similar to the work done by Saad [50] and Saad and Zhang [51] in ILU with multielimination (ILUM); however, there each elimination is exact and at a later stage the resulting system is solved by a simple iteration scheme. It appears that in our approach again grid-independent convergence is possible even for unstructured problems. An interesting side effect of red-black orderings is their attractiveness for implementation on supercomputers. This was the basic argument for Saad to develop ILUM. In [10] we constructed such an implementation for a shared-memory parallel computer using block slicing to distribute the work over the processors and jagged-diagonal storage to speed up the vector processing.

The above orderings ask for a matrix close to an M-matrix and are not very suitable for many other matrices. The (Navier–) Stokes problem, for example, leads to matrices that differ largely from an M-matrix. The Stokes problem can be written as a system with a symmetric indefinite matrix. It has negative and positive eigenvalues. The Navier–Stokes equation is a perturbation of the Stokes equation but is not symmetric. Another example is the convection-diffusion equation with strong convection and discretized by central differences. In [27, sect. 4.4] an overview of direct methods for this type of problem is given.

The symmetric structure of the matrix is destroyed by partial or complete pivoting and therefore we always employ diagonal pivoting (reordering). A form relevant to our approach is the diagonal pivoting method of Bunch and Parlett [14]. In this method the notion of a pivot is extended to 2×2 blocks that are used when, due to large elements outside the diagonal, choosing a 1×1 pivot on the diagonal is no longer stable.

Relation to algebraic multigrid. Grid-independent convergence has become almost synonymous with multigrid. The algorithms in the multilevel world can be classified into two groups: parallel subspace correction (PSC) and successive subspace correction (SSC) [61]. Some papers use the corresponding terminology from domain decomposition approaches: additive Schwarz and multiplicative Schwarz methods. The two types are similar to the Jacobi and Gauss–Seidel methods. The first is more suited for parallel computations whereas the second is more attractive on sequential computers. The classical V-cycle multigrid is an SSC method (see [61]), whereas the Bramble–Pasciak–Xu (BPX) method [11] is a PSC method. In the classical approaches the sequence of nested subspaces is given and together with the corresponding orthogonal projections the coarse grid matrices are constructed by means of the Galerkin approach. One way to improve the robustness of this classical approach is to use matrix-dependent projections [19, 44, 45]. As a next step the subspaces can be determined by the matrix. This resulted in the so-called algebraic multigrid (AMG). Although these steps were first done in the multiplicative context they can also be applied in the additive form (see [28]).

The approach we follow is connected with AMG as introduced by Ruge and Stüben [47]. In their approach two steps in particular are important. The first step is the selection of coarse grid points. The coarse grid points should be distributed uniformly over the grid such that the matrix can be ordered in a 2×2 block partition with the property that the coupling between points in A_{22} is weak. The second step is the construction of an interpolation formula, again using the matrix entries. Once this interpolation operator with a weighting of unity is settled, the coarse grid operator can be constructed by a Galerkin approach. Thereafter the construction is started again at the coarse grid. In the iteration process smoothing at each level has to be applied to get rid of high-frequency errors. In [16] an AMG method is presented that can handle nonsymmetric problems. Here, a special interpolation formula is derived that can handle positive off-diagonal elements and is accurate for linear functions. Moreover, the coarse grid operator is computed more accurately by an approximate elimination. This results in a rather robust method.

In a certain sense our approach may be viewed as such an AMG method. However, we don't use smoothing and the prolongation and restriction operators appear in a natural way during the decomposition. As with AMG only one algorithm can handle various problems, which is not the case for the geometrically oriented multigrid algorithms.

In the following we will assume that the matrices originate from a partial differential equation (PDE) without any specific scaling of columns. If possible we start with a symmetric matrix; otherwise a row scaling is performed such that a reasonable degree of normality is obtained [17]. In the case of a *system* of PDEs the unknowns should be properly scaled by the user. Note that dropping strategies are sensitive to scaling. Scaling is a difficult job and can best proceed on a problem-by-problem basis; see [27] for further discussion.

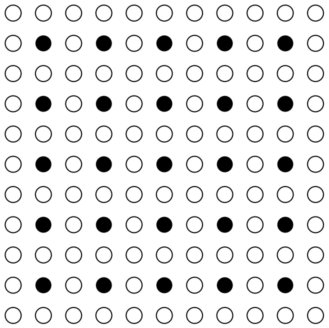


FIG. 1. *Hierarchical multigrid ordering in NGILU.*

2. MRILU. MRILU generalizes an earlier method called nested grids ILU (NGILU) [8]. For a better understanding of MRILU we will therefore start with a short description of this method. NGILU uses a multilevel approach as in multigrid and combines a hierarchical multigrid ordering of the unknowns with an easy-to-construct ILU factorization using a special drop tolerance technique. The construction of the incomplete factorization is illustrated by a brief discussion of the first reduction step. Suppose we start with the system $Sx = b$ on the grid given in Figure 1. On this grid we distinguish between points/unknowns $\circ$ and $\bullet$, where the dots are the points from a next-coarser grid. This allows the system to be partitioned as

$$\begin{pmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{pmatrix} \begin{pmatrix} \circ \\ \bullet \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}. \tag{1}$$

For many problems it can be shown that S_{11} is well-conditioned. Think of S_{11} as the matrix that remains when the unknowns in the dots are prescribed. In that case we will get rid of the smallest eigenvalues and the corresponding smooth eigenvectors (low frequencies). For this well-conditioned S_{11} we can easily construct a sparse incomplete factorization $\tilde{S}_{11}$ and use it with a drop tolerance ε to construct the incomplete factorization

$$\tilde{S} = \begin{pmatrix} \tilde{S}_{11} & \tilde{S}_{12} \\ \tilde{S}_{21} & \tilde{S}_{22} \end{pmatrix} = \begin{pmatrix} \tilde{S}_{11} & O \\ \tilde{S}_{21} & S^{(2)} \end{pmatrix} \begin{pmatrix} I & \tilde{S}_{11}^{-1}\tilde{S}_{12} \\ O & I \end{pmatrix}.$$

Here the Schur complement

$$S^{(2)} = \tilde{S}_{22} - \tilde{S}_{21}\tilde{S}_{11}^{-1}\tilde{S}_{12}$$

corresponds with the reduced system obtained after elimination of the unknowns in the circles. Now we repeat the process for this reduced system on the coarser grid given by the dots in Figure 1. We continue this approach until we obtain a Schur complement small enough to be solved with some standard method and we finally arrive at the incomplete factorization

$$A = LU + R,$$

where R is called the residual matrix. To obtain grid-independent convergence, it is essential that the drop tolerance ε decreases as we go to higher levels (normally by a factor of 4 or 5 in two dimensions). If we consider, just as an illustration, the extreme case that the residual matrix has only nonzeros in the first left upper part R_{11}, the vector Rx contains only components of the first level. With this type of preconditioner all low-frequency errors are eliminated immediately and the iterative method has only to remove high-frequency errors with a wavelength in the order of the mesh size.

This approach can be combined with Gustafsson's modification. It is also advantageous to use an appropriate ordering of the unknowns within each level. For a five-point stencil we can use a red-black ordering. This offers the possibility of starting in the first step with an exact reduction of about half of the unknowns. This will always be the starting point and the preconditioning and acceleration are actually performed for this reduced problem.

Some results of NGILU are given in the next section. For many problems NGILU leads to fast, smooth, and (almost) grid-independent convergence. However, shortcomings of NGILU are the need for a structured grid and, for supercomputers, the sequential nature of the forward and backward substitution when solving $LUx = b$. It also turns out that the method does not perform well on stretched grids, a property it shares with a number of popular iterative methods [9].

The problem with NGILU is that the numbering is based entirely on the grid and does not take into account the size of the nonzero elements of the matrix. For an efficient factorization it is crucial that the upper left blocks S_{11} are always well-conditioned. The difficulties with the stretched grid are caused by the fact that there this condition is violated.

These shortcomings have been overcome in MRILU, the generalized version of NGILU. Here the renumbering is not made beforehand on the basis of the underlying grid or the sparsity pattern [21] but is determined during the construction of an incomplete block factorization using the sparsity pattern and the magnitude of the elements. The method is related to ILUM of Saad [48, 49]. During the factorization we guarantee by construction that the diagonal blocks to be inverted are strongly diagonally dominant, i.e., the coefficients of S_{11} satisfy

$$\sum_{i \neq k} |s_{ik}| \leq \varepsilon |s_{ii}| \text{ with } \varepsilon < 1.$$

By taking ε small enough, we can approximate S_{11} by a diagonal matrix. This not only simplifies the construction of the next Schur complement but also leads to more potential parallelism. The unknowns belonging to S_{11} can be eliminated simultaneously. This also means that the ordering within this level is no longer relevant.

For general matrices the renumbering can be constructed by a greedy algorithm. In the implementation we keep track of the absolute sum of columns belonging to the points selected for S_{11}. For each following candidate we now can easily verify whether it can be added to the (near) independent set selected thus far.

In this way each Schur complement S is partitioned as

$$\begin{pmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{pmatrix}$$

and the strongly diagonally dominant S_{11} is replaced by a diagonal matrix.

To limit the number of nonzeros, small elements will be dropped during the construction. It is difficult to study the existence and stability of ILU factorizations for general matrices and even more difficult to study the effect of dropping on the eigenvalues of the preconditioned system. Therefore dropping criteria as used in practice are based on heuristics.

Dropping an element s_{ij} and, in the modified approach, compensating for this dropping by adding s_{ij} to s_{ii} means a modification of row i and column j. Whether this dropping is acceptable is usually measured in terms of s_{ii} and s_{jj}. For stretched grids or strongly varying coefficients the diagonal of the next Schur complement can be significantly smaller than the original diagonal of S_{22}. Relatively small modifications of the diagonal in S_{22} may be large compared to this new diagonal. The diagonal of the next Schur complement can become even smaller in the steps to follow. This is the main reason that the factorization can easily break down (cf. section 1). To anticipate these problems we would prefer dropping criteria based on the nontrivial diagonal D in the final factorization. The diagonal of S_{11} becomes a (slightly modified) part of D and this explains that, just as with NGILU, dropping elements outside S_{11} is much more critical. We limit the dropping outside S_{11} to S_{12} and S_{21} and approximate D within S_{22} by the diagonal of the next Schur complement (temporarily calculated without dropping outside S_{11}). For dropping we now demand that on each row or column i the sum of the absolute values of all discarded elements, including the values belonging to earlier levels, is smaller than $\varepsilon|d_i|$, with d_i the "updated" diagonal element on row i. When dropping entries in S_{21} on a certain row, that same row might also be accessed for dropping on higher levels. Therefore we limit the row space for dropping within S_{21} even further and multiply the remaining space by the number of columns in S_{21} divided by the dimension of S. Finally, the available space for lumping on rows of S_{21} should not be consumed too fast and therefore only entries smaller than a quarter of this space are dropped. Similar restrictions are made for the dropping within columns of S_{12}.

For many problems it turns out that the d_i become more or less constant during the block factorization; see also [33]. This offers the possibility to simplify the dropping strategy and use a constant d_i just from the start. In the next section we refer to it as the simplified dropping strategy.

The above approach can be generalized to a block approach. To decide whether entries in a certain block row can be dropped, we multiply this block row from the left with the inverse of the corresponding diagonal block. This can be simplified somewhat by considering only the absolute maximum in each column of this inverse. The dropping in columns can be handled in a similar way using multiplication from the right.

Theoretical results for this type of preconditioner or the related AMG-based approach are rare, as is also observed in [28]. In general it is restricted to existence of the decomposition and analysis of special cases. For the case of a symmetric diagonally dominant M-matrix some theoretical results with respect to the NGILU method can be found in [56, 8]. For MRILU it is shown in [7] that on all levels the constructed Schur complements are again symmetric diagonally dominant M-matrices. Similar results can be found in [4, 46].

3. Results.

In the first two problems the typical behavior of NGILU is illustrated by a comparison to other standard methods. In the variety of problems that follows, the results of MRILU are compared with those of several other methods. We will focus our attention on the solution process.

Of course the cost for solving depends upon the quality of the preconditioner, hence on the cost for its construction. For one of the examples described in section 3.4, i.e., the Poisson equation on a finite element mesh with 37,791 unknowns, detailed

Table 1
Results on a finite element mesh for various values of ε.

ε	Flops prec.	Flops solving	Flops	Rel. fill	Levels	It.
0.7	152	1699	1851	0.77	12	45
0.6	174	1057	1231	0.86	13	27
0.5	194	727	921	0.95	14	18
0.4	223	546	769	1.1	16	13
0.3	284	501	785	1.3	17	11
0.2	368	441	809	1.6	19	9
0.15	481	424	905	1.9	21	8
0.1	764	363	1127	2.4	24	6
0.03	1684	311	1995	3.6	35	4
0.01	3279	293	3572	5.1	46	3
0.003	6231	246	6477	6.9	59	2
0.001	10221	296	10517	8.7	69	2
0	55562	749	56311	54.0		1

information about what happens when going from a rough preconditioner ($\varepsilon = 0.8$) toward a complete factorization ($\varepsilon = 0$) is presented in Table 1.

The table shows (in flops per unknown) the cost for the construction of the preconditioner, the cost for solving, and the total cost. It further shows the fill (number of nonzeros in the incomplete LU factorization divided by the nonzeros in the original system), the number of levels, and, finally, the number of CG iterations needed for a decrease in the 2-norm of the residual of the preconditioned system by at least a factor of 10^6. The results for ($\varepsilon = 0$) are from a standard Cholesky factorization. The behavior shown in Table 1 is typical for MRILU. The results presented in this section are obtained for a choice of ε such that the cost for construction of the preconditioner and the cost for solving are of the same order. Note that this choice is not very critical with respect to the total number of flops. The number of iterations is normally of the order of 10. The presented results for solving can be somewhat improved by using a smaller value of ε. This certainly can be done in time-dependent or nonlinear systems where the preconditioner can be used several times and the cost of its construction is therefore amortized over several solves. For more detailed information with respect to the costs and a comparison with other methods we refer to [6]. Unless denoted otherwise we will use Bi-CGSTAB as accelerator.

3.1. Poisson equation on a uniform grid. For a Poisson equation with constant coefficients discretized on a uniform grid it may be expected from the discussion in the previous section that the hierarchical multigrid ordering will work well. For Neumann boundary conditions on all sides of the unit square, the results for a number of standard methods and NGILU are given in Figure 2. The figure shows the number of flops per unknown necessary to improve the preconditioned residual by six digits versus the number of unknowns. In modified incomplete Cholesky CG (MICCG) small perturbations are applied to the diagonal as described in [32], i.e., before the factorization all diagonal elements are multiplied by $1 + \zeta h^2$. The choice $\zeta = 10$ is almost optimal in the present case. With standard incomplete Cholesky CG (ICCG) the number of flops per unknown grows very strongly with the problem size. Perturbed MICCG does a much better job, but NGILU is by far the best and shows grid independent convergence.

For this simple standard problem special-purpose solvers, e.g., solvers based on FFT or cyclic reduction, are an order of magnitude faster than NGILU(0.2); see [6]. Methods such as the BPX method [11] or the multilevel filtering technique [34], which

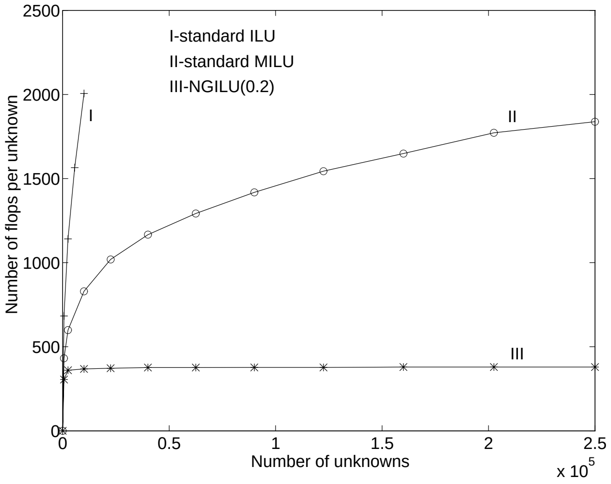


FIG. 2. *Neumann problem $-\Delta u = f$ on a uniform grid.*

are designed for constant coefficient problems (but also can operate on nonconstant coefficient cases), work well on this problem. From [34] we took the result of the standard BPX method for the corresponding problem with 12,000 unknowns (a point very near the left of our plot). To gain 6 digits 750 flops are needed, about twice as many as for NGILU(0.2). According to the bound for the condition number of the preconditioned matrix $O(\log h^{-1})$, this gap will increase with the number of unknowns. For this problem the multilevel filtering technique (an improvement of the BPX method) using an appropriate filter is about as efficient as NGILU(0.2).

3.2. An aquifer problem. As a less trivial example we take a simplified aquifer problem as described in [57]. The nonsymmetric system of linear equations is a result of the discretization of the following steady convection-diffusion equation:

$$-\frac{\partial}{\partial x}\left(a(x,y)\frac{\partial u}{\partial x}\right) - \frac{\partial}{\partial y}\left(a(x,y)\frac{\partial u}{\partial y}\right) + 2e^{2(x^2+y^2)}\frac{\partial u}{\partial x} = \begin{cases} 100 & \text{in center,} \\ 0 & \text{elsewhere.} \end{cases}$$

The diffusion equation coefficient function $a(x, y)$ and the Dirichlet boundary conditions are shown in Figure 3. Figure 4 shows the convergence behavior of Bi-CGSTAB on a 201 × 201 grid for ILU, MILU, and NGILU. The standard MILU factorization breaks down in this case. It is essential to use MILU with a small drop tolerance [55]. Note the smooth convergence behavior of NGILU, which is very favorable; see [57, 53, 52].

3.3. Poisson equation on a severely stretched grid. Investigating the flow in a driven cavity [58] leads to a solution where a lot of action occurs near the boundaries.[1] To get enough resolution near the boundaries, a grid is used, as given in Figure 5. Here we consider the solution of the homogeneous Poisson equation with zero Neumann boundary conditions on an exponentially stretched grid, where the ratio of maximum and minimum mesh size is given by $h_{max}/h_{min} = 100$. The results are obtained for a nonzero starting vector and the stopping criterion $u_{max}^{(n)} - u_{min}^{(n)} <$

[1] See http://www.math.rug.nl/˜veldman/cfd-gallery.html.

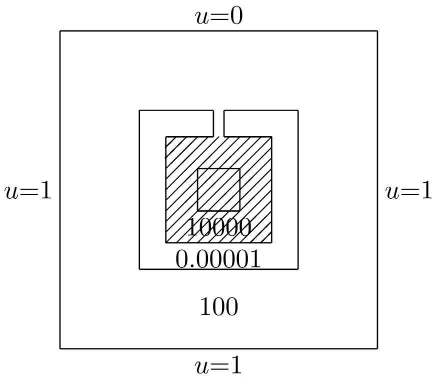


FIG. 3. *Diffusion coefficient $a(x, y)$.*

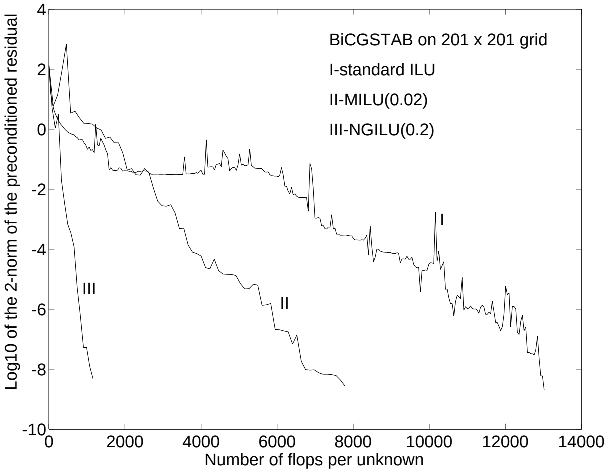


FIG. 4. *Convergence behavior on an aquifer problem.*

$10^{-6}(u^{(0)}_{max} - u^{(0)}_{min})$. This type of problem may cause the convergence of standard iterative methods (including multigrid) to deteriorate [9]. In Figure 6 matrix renumbering ICCG (MRICCG), the symmetric version of MRILU (with the simplified dropping strategy), is compared to the standard methods SOR, ICCG, and MICCG. The last one again uses a proper perturbation of the diagonal, and for both ICCG and MICCG the efficient Eisenstat implementation is used [25]. Moreover, in MICCG a small perturbation is applied to the diagonal. The figure shows for an $M \times M$ grid the number of flops per unknown as a function of M. Clearly MRILU outperforms the other methods. Moreover, it shows a convergence nearly independent of the grid. All methods show their typical behavior. For SOR it is known that for the optimal overrelaxation factor the convergence is $1 - O(1/M)$ (see [62]) and for ICCG a similar behavior holds. This means that the work is linear in M. For MICCG the amount of work is about $\sqrt{M}$. This is nicely reflected in the results.

The stretching of the grid may also be interpreted as anisotropy. We applied MRILU to many other types of anisotropic problems, among which are the two-dimensional ones described in [29] and [34]. The behavior is always the same as for

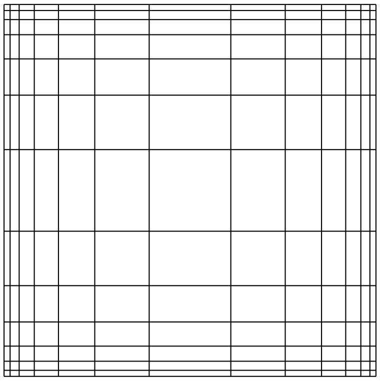


FIG. 5. *Exponentially stretched grid.*

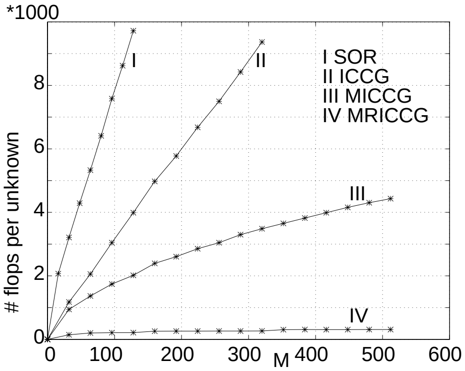


FIG. 6. *Comparison with standard iterative methods on a stretched grid.*

the problem described in this section and about 300 flops per grid point are needed. This means that an order of magnitude in efficiency is easily gained with respect to the results in those papers. Similar observations are made in [28].

3.4. Poisson equation on a finite element mesh. MRILU can handle arbitrary sparsity patterns. Here we will show results on a finite element mesh as given in Figure 7. This grid is generated by a finite element package. On this grid $-\Delta u = f$ has been solved with Dirichlet boundary conditions. As stopping criterion we demand that the 2-norm of the residual of the preconditioned system is decreased by at least a factor of 10^6. Again we compare MRILU, with the simplified dropping strategy, to some standard methods: a direct solver (a frontal method), ILU, and MILU. The results are displayed in Figure 8, where M denotes the square root of the total number of unknowns. As one may expect the direct solver is applicable only to small problems. However, one should note that in the case of a repeated solve direct methods may be attractive. For ILU we see again its predicted linear behavior. We see the same behavior for MILU because the perturbation is difficult to choose on a general mesh. MRILU shows again its nearly grid-independent convergence. Hence, it performs equally well on structured and unstructured grids. For a comparison with other advanced methods on this problem see [6].

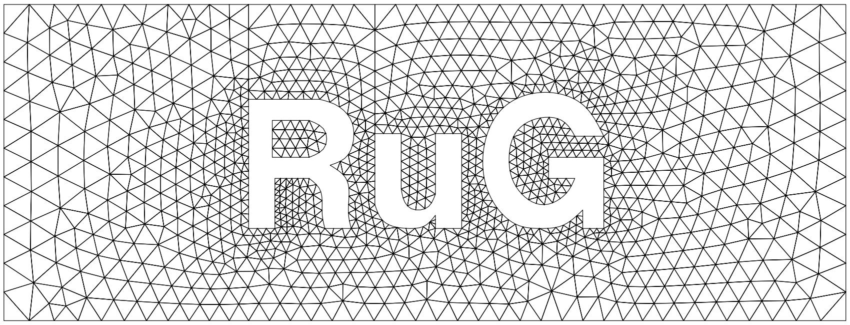


Fig. 7. *Finite element grid.*

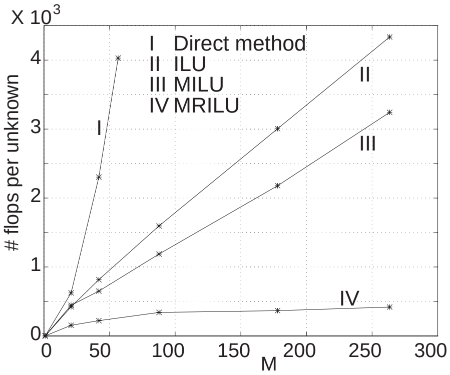


Fig. 8. *Comparison on a finite element grid.*

3.5. Centrally discretized convection-diffusion problem. In this section we will show how MRILU can be applied to convection-dominated problems. We consider the model problem

$$-u_{xx} - u_{yy} + au_x + bu_y = f, \qquad a, b \gg 1. \tag{2}$$

In many fluid flow problems convection plays a central role. However, numerically there are some difficulties associated with the terms modeling this phenomenon. For a sufficient accuracy one usually needs at least a second-order discretization. A simple one is the central discretization

$$u_x \approx (u_{j+1} - u_{j-1})/(2\Delta x). \tag{3}$$

In strongly convection dominated flows this results in weak coupling of odd and even points, which may result in the occurrence of so-called 2-Δx wiggles. For that reason artificial diffusion often is used to restore the coupling.

On sufficient fine grids, nevertheless, central differences are the most accurate [58, 59], but we have to face some problems concerning the coefficient matrix. In strongly convection dominated flows this matrix may be far from an M-matrix. The diagonal is weak with respect to the off-diagonal elements. This is especially difficult for incomplete decompositions. In complete decompositions partial pivoting usually is used to overcome this problem. However, this destroys the structure in the matrix.

It is also possible to precondition on the basis of the so-called one-sided discretization of the convection-diffusion problem. (For more details on these defect correction type approaches see [20].)

We prefer the use of a block form, as will be explained below. In one dimension there is much similarity between our approach and nested dissection or cyclic reduction. Therefore we will introduce our block form by that method.

One-dimensional cyclic reduction. Consider the stencil

$$[1 \qquad \varepsilon \quad -1]$$

as a simplification of the convection-diffusion stencil in one dimension. Eliminating the unknowns at the odd points of the grid results in a system for unknowns at the even points for which the stencil is given by

$$[-1/\varepsilon \qquad \varepsilon + 2/\varepsilon \quad -1/\varepsilon].$$

To obtain this stencil, multipliers $\pm 1/\varepsilon$ have to be used. In practical computations ε may become as small as 0.01, resulting in large multipliers. However, after this unusual step a symmetric positive definite system occurs which can be solved by standard approaches. Elman and Golub [26] show how such an approach can be utilized.

One-dimensional block cyclic reduction. Suppose we keep unknowns of two subsequent grid points together. We then get a block stencil of the form

$$\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \varepsilon & -1 \\ 1 & \varepsilon \end{bmatrix} \begin{bmatrix} 0 & 0 \\ -1 & 0 \end{bmatrix}.$$

After elimination of the odd pairs we obtain the following reduced system for the even pairs:

$$\begin{bmatrix} 0 & \frac{1}{1+\varepsilon^2} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \varepsilon + \frac{\varepsilon}{1+\varepsilon^2} & -1 \\ 1 & \varepsilon + \frac{\varepsilon}{1+\varepsilon^2} \end{bmatrix} \begin{bmatrix} 0 & 0 \\ \frac{-1}{1+\varepsilon^2} & 0 \end{bmatrix}.$$

In this case the multipliers used are

$$\begin{bmatrix} \frac{-1}{1+\varepsilon^2} & \frac{\varepsilon}{1+\varepsilon^2} \\ 0 & 0 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 0 & 0 \\ \frac{-\varepsilon}{1+\varepsilon^2} & \frac{-1}{1+\varepsilon^2} \end{bmatrix}.$$

Observe that all elements in the multiplier are now less than one. Hence, the reduction is stable. Furthermore, observe that the diagonal elements of the central block have become larger. As the process is repeated these elements keep growing. (In fact they double as long as they are small with respect to the other elements in the block.) This is a favorable situation which says that the coupling of the unknowns within a pair will become stronger during the reduction and eventually will fix the coupling between odd and even points in the back substitution. Observe that this block approach is applicable for any ε and thus also in cases where it is not necessary.

The extension of this block approach to two dimensions is done as follows. Pairs are chosen on the basis of the dominant flow direction. These pairs are maintained during the whole process. (Note that one could form new pairs for each new Schur complement. This gives slightly better results but is of course also more expensive.)

Table 2
Convection-diffusion problem with $a = 10{,}000$, $b = 1000$.

Problem	Fill	Rel.fill	It.	Flops
S_{CR}	Low	2.0	35	2390
$\kappa = 2417$	Mod.	2.6	11	911
	High	2.8	5	451
S_{BlCR}	Low	0.9	8	435
$\kappa = 18$	Mod.	1.4	4	268
	High	1.8	3	230

Convection-diffusion with constant coefficients. We will present some results for the convection-diffusion problem (2) with $a = 10{,}000$, $b = 1000$. The domain is the unit square; we take Dirichlet conditions at $x = 0$, $y = 0$, and Neumann conditions at $x = 1$, $y = 1$, and improve the preconditioned residual by 6 digits. The number of equidistant grid points in each direction is 32. We choose a convection-dominated flow (mesh Peclet number about 150) and compare the standard approach without blocks, resulting in exactly the same Schur complement as the one used in [26] with the block approach. The results are displayed in Table 2. The first column gives the condition number of the first Schur complement in both approaches and one observes a dramatic difference. The second column gives a fuzzy indication of the fill. This is made more explicit in the third column, where the fill needed for the decomposition relative to that of the original matrix is given. The fourth column shows the number of iterations and the last column gives the flop count for the solution process. Note the relation between the bad conditioning and the iteration. In the pointwise approach far more fill is needed to get an acceptable number of iterations.

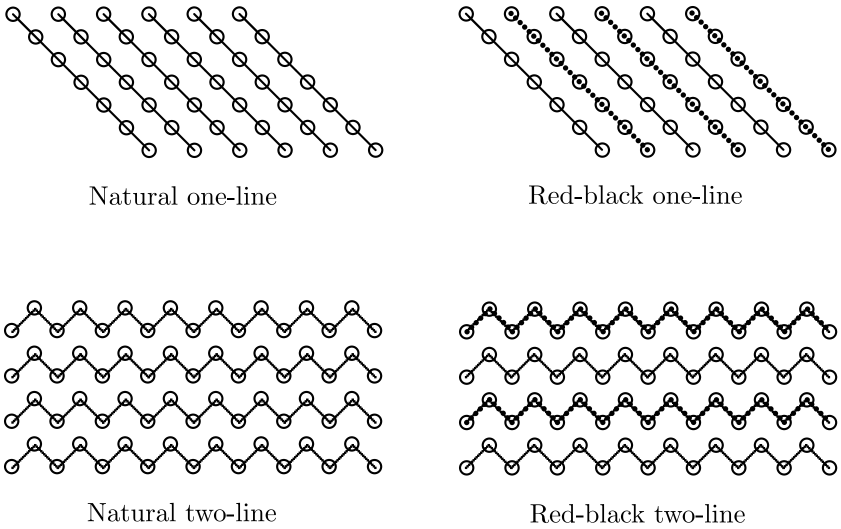
Comparison on a rotating flow. For a convection-diffusion problem posed by Morton [37] we will compare our result with those of Elman and Golub [26, Tables 6.1 and 6.2]. In this problem the coefficients of the first derivatives vary and can be interpreted as velocities belonging to a rotating flow. For this problem Elman and Golub use two methods: block Gauss–Seidel and GMRES combined with block ILU. For the choice of the blocks four variants are considered. They are displayed in Figure 9. Our comparison will be expressed in flop counts. In Table 3 the results are shown when the residual is improved by six digits starting with a random vector. Here (2,M–) denotes that a block size of 2 has been used with the unmodified approach (M+ for modified). For $1/\varepsilon = 100$ the matrix is almost triangular, which explains the good convergence for all methods. For $1/\varepsilon = 1000$ the block variant performs much better than the other methods.

3.6. SHERMAN problems.

Recently, Chapman, Saad, and Wigton [17] showed results for incomplete decompositions on the SHERMAN problems 2, 3, and 5.[2] In this section we compare these results with those of MRILU. (For problems 1 and 4 MRILU performs analogously.) In Table 4 these problems are described briefly. In the comparison we adopted results from [17] of GMRES(50) with various preconditioners:

- ILUT: threshold drop tolerance and fill number;
- ILUD: Gustafsson's modification variant of ILUT; drop tolerance only;
- ILU(k): dropping strategy based on "level of fill."

[2] SHERMAN problems are available as part of the Harwell–Boeing Collection at http://math.nist.gov:80/MatrixMarket. Users should be aware that the block size in SHERMAN2 is 6 instead of the given number 5.



Natural one-line

Red-black one-line

Natural two-line

Red-black two-line

FIG. 9. *Blocks used in ILU and Gauss–Seidel.*

It is interesting to compare our approach with these methods. The methods are quite similar but MRILU adds the matrix-dependent ordering and the use of blocks. A random right-hand side is used with a zero starting vector. The residual is improved by eight digits. The results can be found in Table 5. For the SHERMAN2 problem one observes that ILUD has serious convergence problems. It is well known (see section 1) that a modified ILU factorization is more sensitive to convergence problems than its unmodified ILU variant. Here the unmodified ILUT performs reasonably only with high fill. This is because one solves a problem with block size 6 by a pointwise method. A pointwise version of MRILU has similar difficulties with this problem.

The SHERMAN3 problem looks like a Poisson problem. The matrix is almost symmetrical and it turns out that here MRILU can also be combined with CG. Compared to Bi-CGSTAB, this doubles the number of iterations but also halves the number of matrix vector multiplications. Therefore the number of flops is comparable.

The ordering produced by MRILU and the use of blocks for SHERMAN2 clearly has a favorable influence on the results. An order of magnitude is easily gained for a comparable fill.

3.7. The incompressible Navier–Stokes equations. In this section some preliminary results for the two-dimensional incompressible Navier–Stokes equations are presented. There is some resemblance to the AMG approach followed by Webster [60].

The incompressible Navier–Stokes equations read

$$\begin{aligned} u_t &= -uu_x - vu_y - p_x + \frac{1}{\mathrm{Re}}(u_{xx} + u_{yy}), \\ v_t &= -uv_x - vv_y - p_y + \frac{1}{\mathrm{Re}}(v_{xx} + v_{yy}), \end{aligned} \tag{4}$$

$$u_x + v_y = 0$$

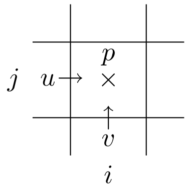
Table 3
Comparison on a rotating flow.

31 × 31 grid							
		$1/\varepsilon$					
Method		10		100		1000	
	Ordering	r.fill	flops	r.fill	flops	r.fill	flops
Gauss–Seidel	Nat. one-line		1098		243		>1350
	RB one-line		1071		261		>1350
	Nat. two-line		1026		234		>1350
	RB two-line		999		234		>1350
GMRES/ILU	Nat. one-line		260		182		858
	RB one-line		1071		702		1924
	Nat. two-line		364		260		2262
	RB two-line		624		676		1846
	Fill	(1,M+)		(1,M+)		(2,M–)	
Bi-CGSTAB/MRILU	Low	1.0	303	1.1	201	1.5	482
	Mod.	1.4	238	1.5	165	2.4	434
	High	1.9	198	1.8	136	3.8	321
63 × 31 grid							
Gauss–Seidel	Nat. 1-line		>1350		198		>1350
	Nat. 2-line		1161		198		>1350
GMRES/ILU	Nat. 1-line		442		260		>3900
	Nat. 2-line		520		312		1664
	Fill	(1,M+)		(1,M+)		(2,M–)	
Bi-CGSTAB/MRILU	Low	1.1	333	1.1	210	1.5	417
	Mod.	1.5	271	1.5	170	2.3	335
	High	2.0	237	1.8	154	3.4	307

Table 4
Short description of SHERMAN problems.

Problem	Order	Nonzeros	Description
SHERMAN2	1080	23094	Thermal simulation, steam injection
SHERMAN3	5005	20033	Black oil, IMPES simulation
SHERMAN5	3312	20793	Fully implicit black oil simulator

with u and v the horizontal and vertical velocity, p the pressure, and Re the Reynolds number. They are discretized using a finite-difference formulation on a uniform staggered grid:



The diffusion term is discretized using the standard second-order central scheme. For the convection term a first-order upwind scheme is used. The equations are solved simultaneously; therefore we can keep together the unknowns belonging to a single grid cell in a vector $w_{i,j} = (u_{i-\frac{1}{2},j}, v_{i,j-\frac{1}{2}}, p_{i,j})$ and denote the dicretized system with the stencil

$$\begin{array}{ccc} NW & N & \\ W & C & E \\ & S & SE, \end{array}$$

Table 5
Flop count per unknown for SHERMAN problems 2, 3, *and* 5.

Method	Fill	SHERMAN2			SHERMAN3			SHERMAN5		
		r.fill	it.	flops	r.fill	it.	flops	r.fill	it.	flops
ILUD	Low	0.8	n.c.	n.c.	1.0	77	8150	1.4	36	3905
	High	1.6	76	15127	1.6	32	2907	4.0	17	1744
ILUT	Low	0.7	145	25504	0.9	216	25174	1.0	30	2754
	High	1.5	12	1661	2.1	46	5634	1.9	21	1772
ILU(k)	k=0	1.0	45	8171	1.0	233	27316	1.0	36	3714
	k=2	3.0	7	1327	3.1	50	6746	3.1	19	1810
		(6,M–)			(1,M+)			(1,M+)		
MRILU	Low	0.4	4	634	1.0	19	747	0.6	8	380
	Mod.	0.5	3	490	1.3	12	518	0.8	6	315
	High	0.7	2	391	1.9	9	450	1.1	4	230

Table 6
Results for the incompressible Navier–Stokes equations.

Problem	Fill	Rel.fill	It.	Flops
Steady state	Low	2.7	14	1723
	Mod.	3.4	11	1557
	High	4.1	9	1471
Time dependent	Low	1.1	13	1040
	Mod.	1.7	11	1067
	High	2.1	9	982

where the coefficients are 3×3 matrices.

In Table 6 we show results of the solution of a typical linear system coming about in the Newton iteration in the steady state driven cavity problem; see [54] for more detail. The problem is discretized on a 33×33 grid (3267 unknowns) and the preconditioned residual is improved by six digits. For the time-dependent problem the time step is such that a Courant number of approximately 1 is obtained. One observes that the steady state problem asks for a much higher fill than the time-dependent problem.

4. Conclusions.

Preconditioning techniques combined with CG-like iterations methods provide powerful tools for solving large sparse systems of equations. Motivated by the success of the nested grid method NGILU, we developed the more general MRILU method, which also can be used on unstructured grids.

An attractive property of this method is that its structure is simple: it is merely an ILU factorization. The essential ingredients are the ordering, which is carried out during the factorization process, and the dropping, which has to be done carefully in order not to destroy the convergence. Both ordering and dropping are based on the size of the entries of the matrix.

Convergence behavior is observed that is nearly independent of the mesh size, an attractive property for very large problems. The method has been applied successfully to symmetric, nonsymmetric, and indefinite problems. For many of the problems shown in this paper the method decreases the number of flops needed to get a prescribed accuracy by an order of magnitude in comparison with other advanced iteration methods. Implementations on high-performance computers are possible due to the high degree of independence in the L and U factors.

The analysis of such a general method is difficult and will be the subject of future research. It may be expected that this will lead to further improvements of the method.

Acknowledgment. The authors wish to thank Prof. Henk van der Vorst for providing the code for the discretization of the aquifer problem.

REFERENCES

[1] O. Axelsson, *Iterative Solution Methods*, Cambridge University Press, London, 1994.

[2] O. Axelsson and V. Eijkhout, *The nested recursive two-level factorization method for nine-point difference matrices*, SIAM J. Sci. Statist. Comput., 12 (1991), pp. 1373–1400.

[3] O. Axelsson and N. Munksgaard, *Analysis of incomplete factorizations with fixed storage locations*, in Preconditioning Methods, Analysis and Applications, D.J. Evans, ed., Gordon and Breach, London, 1983, pp. 219–241.

[4] R.E. Bank and C. Wagner, *Multilevel ILU Decomposition*, Numer. Math, to appear.

[5] R. Beauwens and L. Quenon, *Existence criteria for partial matrix factorizations in iterative methods*, SIAM J. Numer. Anal., 13 (1976), pp. 615–643.

[6] E.F.F. Botta, K. Dekker, Y. Notay, A. van der Ploeg, C. Vuik, F.W. Wubs, and P.M. de Zeeuw, *How fast the Laplace equation was solved in* 1995, Appl. Numer. Math., 24 (1997), pp. 439–455.

[7] E.F.F. Botta and A. van der Ploeg, *Preconditioning techniques for matrices with arbitrary sparsity patterns*, in Proceedings of the Ninth International Conference on Finite Elements in Fluids, New Trends and Applications, University of Padova, Venice, 1995, pp. 989–998.

[8] E.F.F. Botta, A. van der Ploeg, and F.W. Wubs, *Nested grids ILU-decomposition (NGILU)*, J. Comput. Appl. Math., 66 (1996), pp. 515–526.

[9] E.F.F. Botta and F.W. Wubs, *The convergence behaviour of iterative methods on severely stretched grids*, Internat. J. Numer. Methods Engrg., 36 (1993), pp. 3333–3350.

[10] E.F.F. Botta, F.W. Wubs, and A. van der Ploeg, *A fast linear-system solver for large unstructured problems on a shared-memory computer*, in Proceedings of the Conference on Algebraic Multilevel Iteration Methods with Applications, O. Axelsson and B. Polman, eds., University of Nijmegen, Nijmegen, The Netherlands, 1996, pp. 105–116.

[11] J. Bramble, J. Pasciak, and J.-C. Xu, *Parallel multilevel preconditioners*, Math. Comp., 55 (1990), pp. 1–22.

[12] Cl.W. Brand, *An incomplete-factorization preconditioning using red-black ordering*, Numer. Math., 61 (1992), pp. 433–454.

[13] N.I. Buleev, *A numerical method for the solution of two-dimensional and three-dimensional equations of diffusion*, Mat. Sb., 51 (1960), pp. 227–238.

[14] J.R. Bunch and B.N. Parlett, *Direct methods for solving symmetric indefinite systems of linear equations*, SIAM J. Numer. Anal., 8 (1971), pp. 639–655.

[15] T.F. Chan and H.A. van der Vorst, *Approximate and incomplete factorizations*, Parallel Numerical Algorithms, ICASE/LaRC Interdiscip. Ser. Sci. Eng. 4, Kluwer Academic Publishers, Dordrecht, the Netherlands, 1997, pp. 167–202.

[16] Q. Chang, Y.S. Wong, and H. Fu, *On the algebraic multigrid method*, J. Comput. Phys., 125 (1996), pp. 279–292.

[17] A. Chapman, Y. Saad, and L. Wigton, *High-order ILU Preconditioners for CFD Problems*, Technical report UMSI 96/14, University of Minnesota, Minneapolis, 1996.

[18] E.F. D'Azevedo, P.A. Forsyth, and W.P. Tang, *Towards a cost-effective ILU preconditioner with high level fill*, BIT, 32 (1992), pp. 442–463.

[19] P.M. de Zeeuw, *Matrix-dependent prolongations and restrictions in a blackbox multigrid solver*, J. Comput. Appl. Math., 3 (1990), pp. 1–27.

[20] J.-A. Désidéri and P.W. Hemker, *Convergence analysis of the defect-correction iteration for hyperbolic problems*, SIAM J. Sci. Comput., 16 (1995), pp. 88–118.

[21] I.S. Duff, A.M. Erisman, and J.K. Reid, *Direct Methods for Sparse Matrices*, Monogr. Numer. Anal., Oxford University Press, New York, 1986.

[22] I.S. Duff and G.A. Meurant, *The effect of ordering on preconditioned conjugate gradients*, BIT, 29 (1989), pp. 635–657.

[23] 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.

[24] V. Eijkhout, *Beware of unperturbed modified incomplete factorizations*, in Iterative Methods in Linear Algebra, R. Beauwens and P. de Groen, eds., North–Holland, Amsterdam, 1992, pp. 583–591.

[25] S.C. Eisenstat, *Efficient implementation of a class of preconditioned conjugate gradient methods*, SIAM J. Sci. Statist. Comput., 2 (1981), pp. 1–4.

[26] H.C. Elman and G.H. Golub, *Line iterative methods for cyclically reduced discrete convection-diffusion problems*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 339–363.

[27] G.H. Golub and C.F. van Loan, *Matrix Computations*, 3rd ed., The Johns Hopkins University Press, Baltimore, MD, 1996.

[28] T. Grauschopf, M. Griebel, and H. Regler, *Additive multilevel preconditioners based on bilinear interpolation, matrix dependent geometric coarsening and algebraic multigrid coarsening for second order elliptic PDEs*, Appl. Numer. Math., 23 (1997), pp. 63–95.

[29] M. Griebel and P. Oswald, *Tensor-product-type subspace splittings and multilevel iterative methods for anisotropic problems*, Adv. Comput. Math., 4 (1995), pp. 171–206.

[30] I. Gustafsson, *A class of* 1*st order factorization methods*, BIT, 18 (1978), pp. 142–156.

[31] W. Hackbusch, *Iterative Solution of Large Sparse Linear Systems of Equations*, Appl. Math. Sci. 95, Springer-Verlag, New York, 1994.

[32] E.F. Kaasschieter, *Preconditioned conjugate gradients for solving singular systems, Iterative methods for the solution of linear systems*, J. Comput. Appl. Math., 24, (1988), pp. 265–275.

[33] S. Knapek, *Matrix-dependent multigrid homogenization for diffusion problems*, SIAM J. Sci, Comput., 20 (1999), pp. 515–533.

[34] C.-C.J. Kuo, T.F. Chan, and C. Tong, *Multilevel filtering elliptic preconditioners*, SIAM J. Matrix Anal. Appl., 11 (1990), pp. 403–429.

[35] 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.

[36] M.-M. Magolu, *Ordering strategies for modified block incomplete factorizations*, SIAM J. Sci. Comput., 16 (1995), pp. 378–399.

[37] K.W. Morton, *Generalised Galerkin methods for steady and unsteady problems*, in Numerical Methods for Fluid Dynamics, K.W. Morton and M.J. Baines, eds., Academic Press, New York, 1982, pp. 1–32.

[38] Y. Notay, *Solving positive (semi)definite linear systems by preconditioned iterative methods*, in Preconditioned Conjugate Gradient Methods, O. Axelsson and L.Y. Kolotilina, eds., Lecture Notes in Math. 1457, Springer-Verlag, Berlin, 1990, pp. 105–125.

[39] Y. Notay, *Conditioning analysis of modified block incomplete factorizations*, Linear Algebra Appl., 154/156 (1991), pp. 711–722.

[40] Y. Notay, *DRIC: A dynamic version of the RIC method*, J. Numer. Linear Algebra Appl., 1 (1994), pp. 511–532.

[41] Y. Notay and Z. Ould Amar, *A nearly optimal preconditioning based on recursive red-black orderings*, J. Numer. Linear Algebra Appl., 4 (1997), pp. 369–391.

[42] T.A. Oliphant, *An implicit numerical method for solving two-dimensional time-dependent diffusion problems*, Quart. Appl. Math., 19 (1961), pp. 221–229.

[43] T.A. Oliphant, *An extrapolation process for solving linear systems*, Quart. Appl. Math., 20 (1962), pp. 257–267.

[44] A. Reusken, *Multigrid with matrix-dependent transfer operators for a singular perturbation problem*, Computing, 50 (1993), pp. 199–212.

[45] A. Reusken, *Fourier analysis of a robust multigrid method for convection-diffusion equations*, Numer. Math., 71 (1995), pp. 365–398.

[46] A. Reusken, *On the approximate cyclic reduction preconditioner*, SIAM J. Sci. Comput., to appear.

[47] J.W. Ruge and K. Stüben, *Algebraic multigrid*, in Multigrid Methods, S.F. McCormick, ed., Frontiers Appl. Math. 3, SIAM, Philadelphia, PA, 1987, pp. 73–130.

[48] Y. Saad, *Highly Parallel Preconditioners for General Sparse Matrices*, Recent Advances in Iterative Methods, IMA Vol. Math. Appl. 60, Springer-Verlag, New York, 1994, pp. 165–199.

[49] Y. Saad, *ILUM: A multi-elimination ILU preconditioner for general sparse matrices*, SIAM J. Sci. Comput., 17 (1996), pp. 830–847.

[50] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS, Boston, MA, 1996.

[51] Y. Saad and J. Zhang, *BILUM: Block Versions of Multi-Elimination and Multi-Level ILU Preconditioner for General Sparse Linear Systems*, Technical report UMSI 97-126, University of Minnesota, Minneapolis, 1997.

[52] G.L.G. Sleijpen and H.A. van der Vorst, *Maintaining convergence properties of BiCGstab methods in finite precision arithmetic*, Numer. Algorithms, 10 (1995), pp. 203–223.

[53] G.L.G. Sleijpen and H.A. van der Vorst, *Reliable updated residuals in hybrid Bi-CG methods*, Computing, 56 (1996), pp. 141–163.

[54] G. Tiesinga, *Block preconditioned BiCGstab*(2) *for solving the Navier-Stokes equation*, Z.

Angew. Math. Mech., 76 (1996), pp. 563–564.

[55] A. van der Ploeg, *Preconditioning techniques for large sparse, non-symmetric matrices with arbitrary sparsity patterns*, in Iterative Methods in Linear Algebra, R. Beauwens and P. de Groen, eds., North–Holland, Amsterdam, 1992, pp. 173–179.

[56] A. van der Ploeg, *Preconditioning for Sparse Matrices with Applications*, Ph.D. thesis, University of Groningen, Groningen, The Netherlands, 1994.

[57] H.A. van der Vorst, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 631–644.

[58] R.W.C.P. Verstappen and A.E.P. Veldman, *A fourth-order finite volume method for direct numerical simulation of turbulence at higher Reynolds numbers*, in Computational Fluid Dynamics '96, J. Periaux, et al., eds., John Wiley, New York, 1996, pp. 1073–1079.

[59] R.W.C.P. Verstappen and A.E.P. Veldman, *Direct numerical simulation of turbulence at lower costs*, J. Engrg. Math., 32 (1997), pp. 143–159.

[60] R. Webster, *An algebraic multigrid solver for Navier-Stokes problems in the discrete second-order approximation*, Internat. J. Numer. Methods Fluids, 22 (1996), pp. 1103–1123.

[61] J. Xu, *Iterative methods by space decomposition and subspace correction*, SIAM Rev., 34 (1992), pp. 581–613.

[62] D.M. Young, *Iterative Solution of Large Linear Systems*, Academic Press, New York, 1971.

SIAM J. Matrix Anal. Appl.
Vol. 20, No. 4, pp. 1027–1037

SPARSE MATRIX COMPUTATIONS ARISING IN DISTRIBUTED PARAMETER IDENTIFICATION[*]

CURTIS R. VOGEL[†]

Abstract. A penalized least squares approach known as Tikhonov regularization is commonly used to estimate distributed parameters in partial differential equations. The application of quasi-Newton minimization methods then yields very large linear systems. While these systems are not sparse, sparse matrices play an important role in gradient evaluation and Hessian matrix-vector multiplications. Motivated by the spectral structure of the Hessian matrices, a preconditioned conjugate gradient method is introduced to efficiently solve these linear systems. Numerical results are also presented.

Key words. distributed parameter identification, regularization, conjugate gradient iteration, preconditioning

AMS subject classifications. 65F10, 65N22

PII. S0895479897317703

1. Introduction.

Parameter identification means the estimation of coefficients in a differential equation for observations of the solution. By a *distributed parameter* we mean a coefficient which is not simply a constant but is a function of position and/or time. Distributed parameter identification problems arise in a number of applications. Important examples include the estimation of elastic parameters from seismic observations and the determination of aquifer characteristics from groundwater flow observations. A simple mathematical model for groundwater flow, which will serve to illustrate numerical techniques to be presented in this paper, is the partial differential equation (PDE)

$$-\nabla \cdot (\kappa \nabla u) = f(\mathbf{x}), \quad \mathbf{x} \in \Omega. \tag{1.1}$$

In groundwater flow applications, u represents fluid pressure, $\kappa(\mathbf{x})$ is the (spatially dependent) hydraulic conductivity, and $\vec{v} = -\kappa \nabla u$ is the fluid flow field. Provided the fluid is incompressible, $f(\mathbf{x})$ represents the fluid gain or loss rate, e.g., due to injecting or pumping out fluid from wells in the aquifer. The parameter of interest, $q(\mathbf{x}) = \log(\kappa(\mathbf{x}))$, is known as the log conductivity and is to be estimated from observations of the solution $u(\mathbf{x})$ to the PDE (1.1). Adopting notation from [1], we represent the parameter-dependent PDE by

$$\mathcal{A}(q)u = f \tag{1.2}$$

and the observations of the solution by

$$u_{obs} = \mathcal{C}u. \tag{1.3}$$

The *parameter-to-observation map* is the composition

$$\mathcal{F}(q) \overset{\text{def}}{=} \mathcal{C}\, u(q) = \mathcal{C}\mathcal{A}(q)^{-1} f. \tag{1.4}$$

[*] Received by the editors March 3, 1997; accepted for publication (in revised form) by R. Freund March 2, 1998; published electronically July 9, 1999. This work was partially supported by NSF grant DMS-9622119 and Air Force Office of Scientific Research grant F49620-96-1-0456.

http://www.siam.org/journals/simax/20-4/31770.html

[†] Department of Mathematical Sciences, Montana State University, Bozeman, MT 59717-0240 (vogel@math.montana.edu).

The measured data, which are inexact, are modeled by

$$z = \mathcal{F}(q) + \eta. \tag{1.5}$$

The error term η accounts for factors such as measurement errors and inadequacies of the mathematical model.

To estimate the parameter q given the data z, one must somehow "solve" the operator equation

$$\mathcal{F}(q) = z. \tag{1.6}$$

Obvious numerical difficulties are presented by the nonlinearity of the operator $\mathcal{F}$ and the large number of unknowns in the discretization of (1.6). A more subtle but very serious difficulty is the ill-posedness of (1.6), i.e., the lack of continuous dependence of the parameter q on the data z.

To overcome ill-posedness, one must apply regularization. Intuitively, this means replacing the ill-posed problem (1.6) with a "nearby" well-posed problem such as the penalized least squares minimization

$$\min_q \frac{1}{2}||\mathcal{F}(q) - z||^2 + \alpha \mathcal{J}_{reg}(q). \tag{1.7}$$

This approach is known as *Tikhonov regularization* in the inverse problems community [15, 9]. Here $\mathcal{J}_{reg}$ is the regularization, or penalty, functional. Besides imposing stability, it serves to penalize "unreasonable," e.g., highly oscillatory, estimates for the parameter q. A popular choice in distributed parameter identification is the squared H^1 seminorm,

$$\mathcal{J}_{reg}(q) = \frac{1}{2}\int_\Omega |\nabla q|^2 = \frac{1}{2}\int_\Omega \sum_i \left(\frac{\partial q}{\partial x_i}\right)^2. \tag{1.8}$$

Recently, the total variation (TV) functional

$$\mathcal{J}_{reg}(q) = \int_\Omega |\nabla q| = \int_\Omega \sqrt{\sum_i \left(\frac{\partial q}{\partial x_i}\right)^2} \tag{1.9}$$

has been applied (see [4, 8, 17]). This has the advantage of better recovering "blocky," possibly discontinuous, coefficients $q(\mathbf{x})$. The *regularization parameter* α in (1.7) is a small positive number which quantifies the trade-off between the goodness of fit to the data and stability.

We will address numerical linear algebraic issues in the solution of problems such as (1.7) which arise in regularized distributed parameter identification. Quasi-Newton methods for handling the nonlinearity yield a sequence of approximates $q^{\nu+1} = q^\nu + s$, with

$$\mathcal{H}^\nu s = -g(q^\nu), \quad \nu = 0, 1, \ldots, \tag{1.10}$$

where the approximate Hessian takes the form

$$\mathcal{H}^\nu = \mathcal{H}_{ls} + \alpha \mathcal{L}. \tag{1.11}$$

The $\mathcal{L}$ in (1.11) arises from regularization functionals such as (1.8) or (1.9) and is a symmetric, nonnegative diffusion operator; cf. [16]. In case (1.8), $\mathcal{L}$ is the negative

Laplacian. Standard discretization techniques, e.g., finite difference or finite element methods, then yield large sparse positive semidefinite matrices. Unfortunately, when these same discretization techniques are applied, the discretization of the least squares term $\mathcal{H}_{ls}$ is *not* sparse. It is symmetric, and with certain quasi-Newton schemes (e.g., Gauss–Newton linearization of the least squares term in (1.7)), it is positive definite. Moreover, it has eigenvalues which cluster at zero, which means that it has limited useful spectral content. In addition, using adjoint, or costate, techniques [5, 18], one can compute the action of the least squares Hessian $\mathcal{H}_{ls}$ on a vector very quickly with sparse matrix methods. This suggests the use of iterative methods such as conjugate gradients (CG) to solve (1.10). Since CG convergence tends to be slow, the issue of preconditioning is important. We will present a preconditioner based on the operator $\mathcal{L}$ which is effective unless the regularization parameter α becomes very small.

This paper is organized as follows. Section 2 contains an example which illustrates the important role played by sparse matrices in gradient evaluation and Hessian matrix-vector multiplications. While this example involves a specific 2-D boundary value problem, the structure and ideas presented here carry over to more general problems. The matrix computations outlined in this section are the discrete analogue of continuous adjoint approaches for computing derivatives in parameter identification and control theory. In section 3, we discuss the spectral properties of the Hessian (1.11), and we present a preconditioned conjugate gradient (PCG) method for its inversion. Numerical results are presented in the final section.

2. Gradient and Hessian matrix-vector evaluation.

We first briefly discuss standard numerical optimization techniques used in parameter identification [1]. Assume a discretization of (1.2)–(1.7),

$$\min_{\mathbf{q}\in\mathbb{R}^{n_q}} \frac{1}{2}||\mathbf{F}(\mathbf{q})-\mathbf{z}||^2 + \alpha J_{reg}(\mathbf{q}), \tag{2.1}$$

where

$$\mathbf{F}(\mathbf{q}) = C\mathbf{u}(\mathbf{q}) = CA(\mathbf{q})^{-1}\mathbf{f}. \tag{2.2}$$

Let n, m, and n_q, respectively, denote the lengths of the vectors $\mathbf{u}$, $\mathbf{z}$, and $\mathbf{q}$. Then $A(\mathbf{q})$ is an $n \times n$ matrix, while C is $m \times n$. Standard secant methods such as BFGS are ineffective for nonlinear least squares problems. Instead, one typically employs a Gauss–Newton approximation (see [7, p. 221]) to the least squares fit-to-data term in (2.1):

$$\begin{aligned} J_{ls}(\mathbf{q}+\mathbf{s}) &\overset{\text{def}}{=} \frac{1}{2}||\mathbf{F}(\mathbf{q}+\mathbf{s})-\mathbf{z}||^2 \\ &\approx \frac{1}{2}||\mathbf{F}'(\mathbf{q})\mathbf{s}+\mathbf{r}(\mathbf{q})||^2. \end{aligned} \tag{2.3}$$

Here the prime ($'$) denotes differentiation with respect to $\mathbf{q}$, and $\mathbf{r}(\mathbf{q}) = \mathbf{F}(\mathbf{q}) - \mathbf{z}$ denotes the least squares residual. Differentiating (2.3) with respect to $\mathbf{s}$ and evaluating at $\mathbf{s} = \mathbf{0}$, one obtains the true least squares gradient,

$$\mathbf{g}_{ls}(\mathbf{q}) = \mathbf{F}'(\mathbf{q})^T\mathbf{r}(\mathbf{q}). \tag{2.4}$$

Taking the second derivative with respect to $\mathbf{s}$ in (2.3) yields the Gauss–Newton approximation to the least squares Hessian,

$$H_{ls}(\mathbf{q}) = \mathbf{F}'(\mathbf{q})^T\mathbf{F}'(\mathbf{q}). \tag{2.5}$$

This matrix is $n_q \times n_q$ and symmetric. Unlike the true least squares Hessian, which contains the additional term $\mathbf{F}''(\mathbf{q})^T \mathbf{r}(\mathbf{q})$, it is also guaranteed to be positive semidefinite. From (2.2) and the fact that $A(\mathbf{q})^{-1}A(\mathbf{q}) = I$,

$$\mathbf{F}'(\mathbf{q}) = -CA(\mathbf{q})^{-1}A'(\mathbf{q})A(\mathbf{q})^{-1}\mathbf{f} = -CA(\mathbf{q})^{-1}A'(\mathbf{q})\mathbf{u}(\mathbf{q}). \tag{2.6}$$

Since $A(\mathbf{q})$ is an $n \times n$ matrix, its derivative $A'(\mathbf{q})$ is a *tensor* of size $n \times n_q \times n$.

To illustrate additional structure, consider a 2-D version of (1.1) on the open unit square, $\Omega = \{(x,y) : 0 < x < 1,\ 0 < y < 1\}$, with homogeneous Dirichlet boundary conditions

$$\begin{aligned} -\frac{\partial}{\partial x}\left(\exp(q(x,y))\frac{\partial u}{\partial x}\right) - \frac{\partial}{\partial y}\left(\exp(q(x,y))\frac{\partial u}{\partial y}\right) = f(x,y), \quad (x,y) \in \Omega,\\ u(x,y) = 0, \quad (x,y) \in \partial\Omega. \end{aligned} \tag{2.7}$$

Applying cell-centered finite difference (CCFD) discretization [19] on a uniform $n_x \times n_x$ grid with lexicographical ordering of the $n = n_x^2$ coefficients of $\mathbf{u}$, one obtains a discrete analogue of (1.2),

$$A(\mathbf{q})\mathbf{u} = \mathbf{f}. \tag{2.8}$$

In the case of uniform CCFD discretization of (2.7), one can express

$$\begin{aligned} A(\mathbf{q}) &= B_x^T D_x(\mathbf{q}) B_x + B_y^T D_y(\mathbf{q}) B_y \\ &= B_{div} D(\mathbf{q}) B_{grad}, \end{aligned} \tag{2.9}$$

where $B_{div} = B_{grad}^T$ is block upper bidiagonal and $D(\mathbf{q})$ is diagonal with diagonal entries consisting of $\kappa = \exp(q)$ evaluated at the cell boundary midpoints. The matrix $A(\mathbf{q})$ is symmetric positive definite and block tridiagonal. For an illustration of the sparsity structure, see [13, p. 55]. With other finite difference and finite element discretizations on a uniform grid, this sparsity structure is retained. The cost of storing and inverting $A(\mathbf{q})$ using direct methods which utilize sparsity is $\mathcal{O}(n_x^3) = \mathcal{O}(n^{3/2})$. If state-of-the-art sparse iterative techniques for elliptic PDEs (e.g., multigrid methods) are used, the storage and computational requirements are typically much smaller. It should be noted that with other geometries or boundary conditions or more general diffusion operators, $A(\mathbf{q})$ need not be symmetric, but its sparsity structure is much the same.

If one assumes observations of $u(x,y)$ at the n cell centers, then the observation operator C is the $n \times n$ identity matrix. More generally, with m observations and (local) interpolation, C is $m \times n$ and sparse.

Next, we discuss the efficient evaluation of both the least squares gradient and Hessian matrix-vector products. The fact that $A(\mathbf{q})^{-1}$ is, in general, a full matrix makes it impractical to assemble the $m \times n_q$ matrix $\mathbf{F}'(\mathbf{q})$; cf. (2.6). Fortunately, all that is required is the *action* of $\mathbf{F}'(\mathbf{q})$ and its transpose on vectors. In particular, the ith component of the least squares gradient in (2.4) is

$$\begin{aligned} [\mathbf{g}_{ls}(\mathbf{q})]_i \overset{\text{def}}{=} \frac{\partial J_{ls}}{\partial q_i} &= \left(\frac{\partial \mathbf{F}}{\partial q_i}\right)^T \mathbf{r}(\mathbf{q}) \\ &= -\left(C\,A(\mathbf{q})^{-1}\frac{\partial A}{\partial q_i}\mathbf{u}(\mathbf{q})\right)^T \mathbf{r}(\mathbf{q}) \\ &= -\left(\frac{\partial D}{\partial q_i}B_{grad}\mathbf{u}(\mathbf{q})\right)^T B_{div}^T A(\mathbf{q})^{-T} C^T \mathbf{r}(\mathbf{q}). \end{aligned} \tag{2.10}$$

The last equality follows from (2.9). If $D(\mathbf{q}) = \mathrm{diag}(\exp(q_i))$, then $\frac{\partial D}{\partial q_i}$ has only one nonzero entry, which is $\exp(q_i)$ in position i, i.

To evaluate the gradient, one first obtains the solution $\mathbf{u}(\mathbf{q})$ to the discrete linear system (2.8) and then computes the residual $\mathbf{r}(\mathbf{q}) = C\mathbf{u}(\mathbf{q}) - \mathbf{z}$. Next, one obtains the solution $\mathbf{y}(\mathbf{q})$ to the discrete adjoint system

$$A(\mathbf{q})^T \mathbf{y} = -C^T \mathbf{r}(\mathbf{q}). \tag{2.11}$$

After computing $\tilde{\mathbf{u}} = B_{grad}\mathbf{u}$ and $\tilde{\mathbf{y}} = B_{div}^T \mathbf{y}$, one computes the gradient components $(\frac{\partial D}{\partial q_i}\tilde{\mathbf{u}})^T \tilde{\mathbf{y}}$ for $i = 1, \ldots, n+1$. Note that these are all sparse matrix computations and that two sparse matrix inversions (one for $A(\mathbf{q})$ and one for its transpose) are required to obtain the gradient vector.

Matrix-vector multiplication with the Gauss–Newton approximation to the Hessian matrix (cf. (2.5)) can be carried out in a similar manner. Given a vector $\mathbf{v} \in \mathbb{R}^{n+1}$,

$$\begin{aligned}
[H_{ls}\mathbf{v}]_i &= \left(CA(\mathbf{q})^{-1} \frac{\partial A}{\partial q_i} \mathbf{u}(\mathbf{q}) \right)^T CA(\mathbf{q})^{-1} \left(\sum_{j=1}^{n+1} \frac{\partial A}{\partial q_j} v_j \right) \mathbf{u}(\mathbf{q}) \\
&= \left(\frac{\partial D}{\partial q_i} B_{grad} \mathbf{u}(\mathbf{q}) \right)^T \\
&\quad \times B_{div}^T A(\mathbf{q})^{-T} C^T C A(\mathbf{q})^{-1} B_{div} \left(\sum_{j=1}^{n+1} v_j \frac{\partial D}{\partial q_j} \right) B_{grad} \mathbf{u}(\mathbf{q}) \\
&= \left(\frac{\partial D}{\partial q_i} \tilde{\mathbf{u}} \right)^T B_{div}^T A(\mathbf{q})^{-T} C^T C A(\mathbf{q})^{-1} B_{div} \left(\sum_{j=1}^{n+1} v_j \frac{\partial D}{\partial q_j} \right) \tilde{\mathbf{u}}. \qquad (2.12)
\end{aligned}$$

Given that one has $\tilde{\mathbf{u}}$ from the gradient computation, this requires inversion of sparse matrices $A(\mathbf{q})$ and $A(\mathbf{q})^T$, plus some additional sparse matrix and dot product computations. Similar adjoint techniques may be used to apply the true Hessian. See for example [14].

Finally, we examine the structure of the matrices arising in the discretization of the regularization operator in (1.7). In general the gradient of $\mathcal{J}_{reg}$ takes the form

$$\mathcal{J}_{reg}'(q) = \mathcal{L}(q) q, \tag{2.13}$$

where $\mathcal{L}(q)$ is a positive semidefinite diffusion operator with a diffusion coefficient which may depend on q. The associated boundary conditions are "natural," or homogeneous Neumann. The true Hessian of the regularization term is

$$\begin{aligned}
\mathcal{J}_{reg}''(q) &= \mathcal{L}(q) + (\mathcal{L}''(q))q \\
&\approx \mathcal{L}(q). \qquad (2.14)
\end{aligned}$$

The above approximate Hessian is symmetric and positive semidefinite and corresponds to the "lagged diffusivity" fixed point iteration introduced in [16]. In the case of H^1 regularization (1.8), $\mathcal{L}$ is the negative Laplacian, which is independent of q. In the case of TV (1.9),

$$\mathcal{L}(q)v = -\nabla \cdot \left(\frac{1}{|\nabla q|} \nabla v \right). \tag{2.15}$$

Provided that the same discretization methods are applied, the sparsity structure of the resulting $n_q \times n_q$ matrix $L(\mathbf{q})$ will be much the same as $A(\mathbf{q})$, e.g., block tridiagonal. The spectral properties of $L(\mathbf{q})$ and $A(\mathbf{q})$ should also be quite similar, except that $L(\mathbf{q})$ has a nontrivial null space consisting of constant vectors. Note, however, that $H_{ls}(\mathbf{q})$ and $L(\mathbf{q})$ have drastically different structures. Due to the presence of $A(\mathbf{q})^{-1}$ in (2.6), $H_{ls}(\mathbf{q})$ is a full matrix with eigenvalues which cluster at zero.

3. Solution of linear systems.

In many parameter identification applications where minimization problems such as (1.7) are solved, quasi-Newton schemes for handling the nonlinearity are rapidly convergent. The computational bottleneck is the solution of the linear systems (1.10), with (approximate) Hessian matrices of the form

$$H = H_{ls} + \alpha L, \tag{3.1}$$

where α is a small positive parameter. L is a discretization of a diffusion operator, e.g., the negative Laplacian, with natural boundary conditions. With standard discretization methods, it is symmetric, positive semidefinite, and sparse, and its null space, null(L), consists of the constant vectors.

The least squares Hessian matrix H_{ls} is harder to characterize. From a spectral standpoint, it has much in common with "blurring" matrices arising in image reconstruction (see [3]). It is the discretization of a compact operator and hence has eigenvalues which cluster at zero. In 2-D, the blurring matrix is block Toeplitz and can be applied quickly using fast Fourier transforms (FFTs). FFTs can also be used to construct effective preconditioners for systems (3.1) arising in image deblurring [3, 17]. In parameter identification, H_{ls} has no such regular structure, but the adjoint approaches sketched in the previous section can be used to compute Hessian matrix-vector products. The construction of effective preconditioners is more difficult in this case, since the explicit computation of components of H_{ls} is impractical.

Given the compactness of the continuous least squares Hessian, perhaps the most obvious approach is to precondition with L, converting (3.1) to *standard form* [11]

$$\tilde{H} = \tilde{H}_{ls} + \alpha I, \quad \tilde{H}_{ls} \quad \text{a discretization of a compact operator.} \tag{3.2}$$

In this case, the rate of convergence of PCG is known to be asymptotically superlinear [6] as the discretization level $h \to 0$. Since L has a nontrivial null space, a preliminary transformation is required. Consider the decomposition

$$\mathbf{s} = a\mathbf{v}_0 + \mathbf{s}^\perp, \tag{3.3}$$

where $\mathbf{s}^\perp \in null(L)^\perp$ and $\mathbf{v}_0 \in null(L)$ with $||\mathbf{v}_0|| = 1$. From the linearizations (2.3) and (2.14), one obtains the quadratic penalized least squares functional

$$Q_\alpha(a, \mathbf{s}^\perp) = ||a\mathbf{F}'(\mathbf{q})\mathbf{v}_0 + \mathbf{F}'(\mathbf{q})\mathbf{s}^\perp + \mathbf{r}||^2 + \alpha(\mathbf{s}^\perp)^T L\mathbf{s}^\perp, \tag{3.4}$$

whose minimizer satisfies the block system

$$\begin{bmatrix} \mathbf{v}_0^T\mathbf{w}_0 & \mathbf{w}_0^T \\ \mathbf{w}_0 & \mathbf{F}'(\mathbf{q})^T\mathbf{F}'(\mathbf{q}) + \alpha L \end{bmatrix} \begin{bmatrix} a \\ \mathbf{s}^\perp \end{bmatrix} = \begin{bmatrix} -\mathbf{v}_0^T\mathbf{g} \\ -\mathbf{g} \end{bmatrix} \tag{3.5}$$

with $\mathbf{w}_0 = \mathbf{F}'(\mathbf{q})^T\mathbf{F}'(\mathbf{q})\mathbf{v}_0$. Taking the Schur complement, one solves the reduced system

$$\overline{H}\,\mathbf{s}^\perp = -\overline{\mathbf{g}}, \tag{3.6}$$

where

$$\overline{H} = \mathbf{F}'(\mathbf{q})^T\mathbf{F}'(\mathbf{q}) - \frac{\mathbf{w}_0\mathbf{w}_0^T}{\mathbf{v}_0^T\ \mathbf{w}_0} + \alpha L \overset{\text{def}}{=} \overline{H}_{ls} + \alpha L,$$

$$\overline{\mathbf{g}} = \mathbf{g} - \frac{\mathbf{v}_0^T\mathbf{g}}{\mathbf{v}_0^T\ \mathbf{w}_0}\mathbf{w}_0.$$

One then sets

$$a = \frac{-\mathbf{v}_0^T\mathbf{g} - \mathbf{w}_0^T\mathbf{s}^{\perp}}{\mathbf{v}_0^T\ \mathbf{w}_0}$$

and computes $\mathbf{s}$ using (3.3).

One can show that $null(\overline{H}) = null(L)$ and that $\overline{\mathbf{g}} \in null(L)^{\perp}$. Hence, the reduced system (3.6) is consistent and has a unique pseudoinverse solution in $null(L)^{\perp}$. This pseudoinverse solution can be obtained using CG, given an initial guess in $null(L)^{\perp}$. One can precondition the reduced system (3.6) with $L^{\perp}$, the restriction of L to $null(L)^{\perp}$, obtaining the desired transformation of the reduced Hessian to the form (3.2).

It should be noted that the above approach is in principle very similar to that of Hanke and Hansen [11]. However, in terms of practical implementation it is quite different in that it does not require a (not necessarily square) matrix B for which

$$\mathbf{s}^T L\mathbf{s} = ||B\mathbf{s}||^2. \tag{3.7}$$

Without a factorization $L = B^T B$, it is not possible to transform the problem to allow the direct application of conjugate gradient least squares (CGLS) or LSQR or some other variant of CG specially designed for least squares problems.

4. Numerical experiments. Consider the 2-D diffusion equation (2.7) with slightly different boundary conditions, $u = 0$ on the left ($x = 0$) and right ($x = 1$) boundary edges, and no flux ($u_y = 0$) boundary conditions on the top and bottom edges. Corresponding to a point source at $(x_0, y_0) = (1/2, 1/2)$, we took $f(x, y) = \delta(x - x_0, y - y_0)$, the Dirac distribution. This yields a solution u with a logarithmic singularity at (x_0, y_0). The diffusion coefficient $\kappa(x, y)$ was taken to be piecewise smooth with a curved interface between regions of low diffusivity on the left and higher diffusivity on the right, and is shown in the upper left subplots of Figures 4.1 and 4.2. The parameter of interest, $q(x, y) = \log(\kappa(x, y))$, is also piecewise smooth. For a discussion of an alternative numerical approach to a very similar model problem, see [10].

All computations were performed using MATLAB [12]. We applied uniform CCFD discretization to the PDE (2.7). The solution $u(x, y)$ and the parameter $q(x, y)$ are defined by their values at the cell centers. With $n_x = 64$ cells on a side, the vectors $\mathbf{u}$ and $\mathbf{q}$ are both of length $n = 4096$. The resulting discretization error is insignificant compared to the simulated measurement error. Linear interpolation was used to obtain values of q at the cell interface midpoints. Due to the logarithmic singularity in u, simulated data were generated only at cell centers *outside* a neighborhood of radius of .05 of the source point (x_0, y_0). Hence, the matrix C, which is induced by the piecewise constant CCFD representation of the solution u, is diagonal with 1's at diagonal entries corresponding to nodes outside this neighborhood, and 0's elsewhere. Most of the diagonal entries are 1's. We generated simulated data according to the

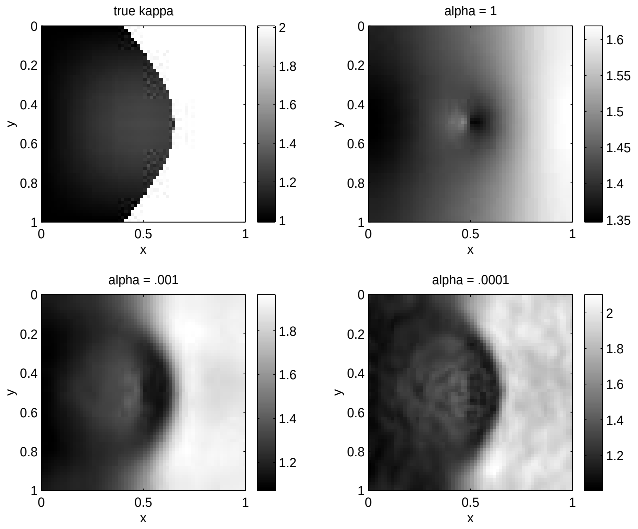


FIG. 4.1. *Grayscale plots showing true diffusion coefficient $\kappa = \exp(q(x, y))$ (upper left corner) and reconstructions obtained for $\alpha = 1$ (upper right), $\alpha = 10^{-3}$ (lower left), and $\alpha = 10^{-4}$ (lower right).*

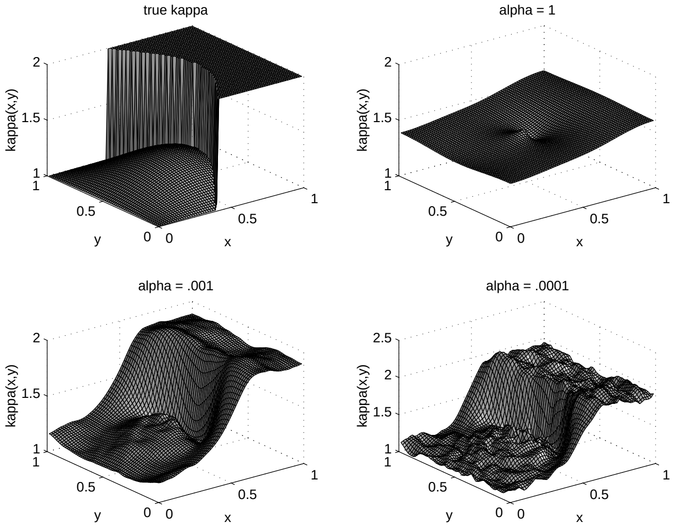


FIG. 4.2. *Surface plots showing true diffusion coefficient $\kappa = \exp(q(x, y))$ (upper left corner) and reconstructions obtained for $\alpha = 1$ (upper right), $\alpha = 10^{-3}$ (lower left), and $\alpha = 10^{-4}$ (lower right).*

discrete version of (1.5), with additive Gaussian error whose mean was zero and whose standard deviation was selected so that the noise-to-signal ratio $||\eta||/||C\mathbf{u}|| = 0.01$.

Figures 4.1 and 4.2 show the reconstructions obtained by solving the penalized least squares minimization problem (1.7) with TV penalty (cf. (1.9)), for various values of the regularization parameter α. Note the increasing amount of detail as α decreases. Note also for small α the onset of spurious oscillations due to the inversion of noise in the data.

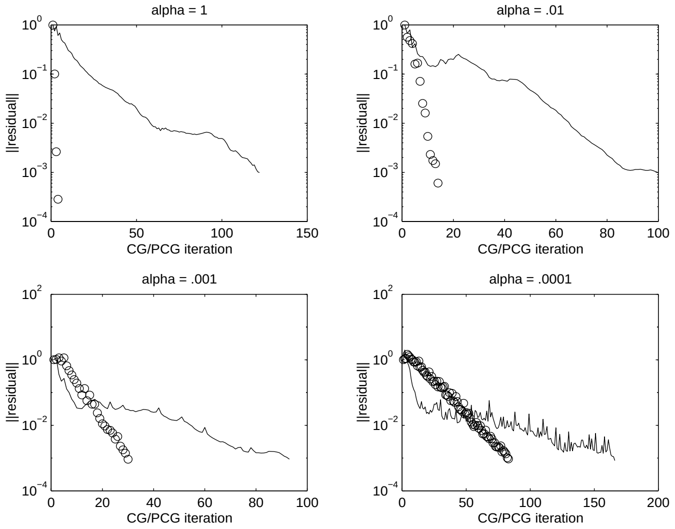


Fig. 4.3. *Convergence history for CG iterations (solid lines) and PCG iterations (circles), showing norm of the residual,* $||H\mathbf{s}^k + \mathbf{g}||$ *vs. iteration count* k*. The upper left subplot corresponds to* $\alpha = 1$*; the upper right subplot corresponds to* $\alpha = 10^{-2}$*; the lower left subplot corresponds to* $\alpha = 10^{-3}$*; and the lower right subplot corresponds to* $\alpha = 10^{-4}$*.*

To estimate the parameter q, we employed a crude "continuation in α" approach, solving minimization problems (2.1) for a decreasing sequence, $\alpha = 10^{-k}, k = 0, 1, 2, 3, 4$. For each fixed value of α, we applied a quasi-Newton iteration with the value of $\mathbf{q}$ from the previous α as the initial guess. At least for relatively large α, this eliminated the need for "globalization" (e.g., line search or trust region) to guarantee convergence, and it required relatively few total quasi-Newton iterates. An a posteriori scheme (see [11]) can then be applied to determine the "best" value of α. Given an optimality criterion for α, one may instead estimate α "on the fly" rather than use a predetermined sequence of values for the regularization parameter. See, for example, [2]. Note that the parameterization of the diffusivity, $\kappa = \exp(q)$, eliminated the need for a nonnegativity constraint on κ.

The quasi-Newton scheme consists of Gauss–Newton linearization of the least squares functional (cf. (2.5)) and "lagged diffusivity" linearization of the TV penalty term; cf. (2.14). The convergence rate for each α is linear but quite rapid. For each value of α, we took five quasi-Newton iterations. Each quasi-Newton iteration required the solution of a linear system of the form (3.1). To solve these systems, we applied CG with and without preconditioning. Figure 4.3 provides a comparison between plain CG and PCG and shows the effects of decreasing the regularization parameter α. The results shown in this figure correspond to the fifth quasi-Newton iteration in each case but are essentially independent of quasi-Newton iteration count. The preconditioner is as described in the previous section. The performance of plain CG varied little with α. On the other hand, PCG performed very well for larger values of α, and the performance deteriorated as α decreased. For $\alpha = 1$, to attain a three order of magnitude decrease in the residual norm, PCG required less than 1/30 as many iterations as plain CG. This ratio decreased to about 1/7 for $\alpha = 10^{-2}$ and 1/3 for $\alpha = 10^{-3}$. For $\alpha = 10^{-4}$, PCG required about half as many iterations as plain CG.

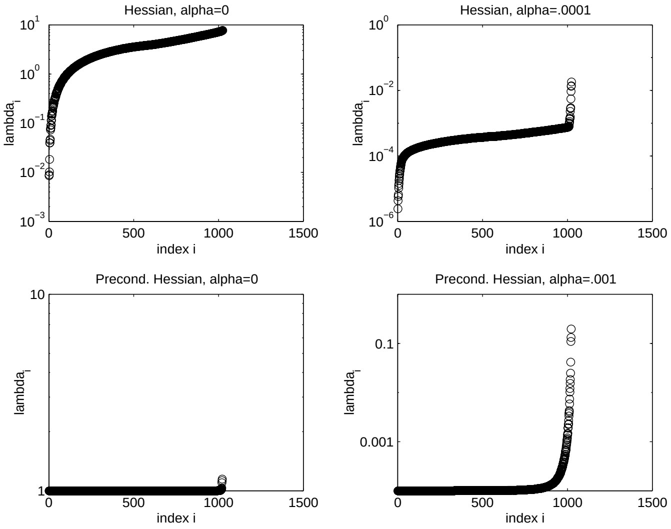


FIG. 4.4. *Eigenvalue distributions for the Hessian (top row) and preconditioned Hessian (bottom row of subplots) for* $\alpha = 1$ *(left subplots) and* $\alpha = 10^{-3}$ *(right subplots).*

The cost of each CG iteration is dominated by the Hessian matrix-vector multiplication. From (2.12), this is dominated by the cost of solving one linear system involving the block tridiagonal matrix $A(\mathbf{q})$ and one linear system involving its transpose. Each PCG iteration requires an additional inversion of $L^{\perp}$, as described at the end of section 3. Assuming the cost of inverting $L^{\perp}$ is comparable to that of inverting $A(\mathbf{q})$ and assuming that the dominant costs arise in solving linear systems, one plain CG iteration should be about two-thirds as expensive as one PCG iteration. This rough cost analysis ignores a substantial amount of overhead in the Hessian matrix-vector multiplications; cf. (2.12). Consequently, from Figure 4.3 one observes a computational cost advantage for PCG for all values of $\alpha \leq 10^{-4}$, but the advantage becomes less pronounced as α decreases. Obviously, when applying a "continuation in α" approach, this particular preconditioner can drastically reduce total computational cost, since several systems (3.1) with relatively large α must be solved.

We close this section with an explanation for the deterioration in PCG performance as α decreases. The analysis in [6] shows that CG is superlinearly convergent for Hilbert space operators equations in which the operator is a compact perturbation of the identity. Unfortunately, this analysis is qualitative and gives no indication of the effects of varying parameters such as α. Looking at the continuous version of (3.1), we see that the Hessian is the sum of a symmetric compact operator, with eigenvalues clustering at zero, and a positive scalar multiple of a diffusion operator, with eigenvalues clustering at positive infinity. Assuming that boundary conditions are imposed so that L is invertible (it is not, but this technical difficulty is overcome by the decomposition (3.3)) our PCG scheme would yield a transformed operator of the form

$$\tilde{H} = L^{-1/2} H_{ls} L^{-1/2} + \alpha I. \tag{4.1}$$

Since $L^{-1/2}$ is now compact, $L^{-1/2} H_{ls} L^{-1/2}$ is again compact and has eigenvalues which decay to zero more rapidly than those of H_{ls}. Consequently, the eigenvalues of $\tilde{H}$ cluster at α; cf. (3.2). This is clearly seen in the bottom two subplots in Figure 4.4.

Unfortunately, as can be seen in the lower right subplot, there are a relatively large number of eigenvalues which are significantly greater than α and are not clustered. Note that the eigenvalue computations were performed on much smaller systems of order $32^2 = 1024$. While the results for the order 64^2 system may be somewhat different, the qualitative behavior is the same.

REFERENCES

[1] H. T. Banks and K. Kunisch, *Estimation Techniques for Distributed Parameter Systems*, Birkhauser, Boston, 1989.

[2] P. Blomgren and T. F. Chan, *Modular Solvers for Constrained Image Restoration Problems*, UCLA CAM Report 97-52, Department of Mathematics, Univ. California, Los Angeles, 1997.

[3] R. H. Chan, T. F. Chan, and C. K. Wong, *Cosine transform based preconditioners for total variation minimization problems in image processing*, in Iterative Methods in Linear Algebra II, Proc. 2nd IMACS Internat. Symposium on Iterative Methods in Linear Algebra, Bulgaria, June 1995, IMACS Ser. Comput. Appl. Math. 3, S. Margenov and P. Vassilevski, eds., pp. 311–329.

[4] T. F. Chan, G. H. Golub, and P. Mulet, *A nonlinear primal-dual method for total variation-based image restoration*, in ICAOS '96, 12th Internat. Conf. on Analysis and Optimization of Systems: Images, Wavelets, and PDE's, Paris, June 26–28, 1996, Lecture Notes in Control and Inform. Sci. 219, M. Berger, R. Deriche, I. Herlin, J. Jaffre, and J. Morel, eds., Springer-Verlag, New York, 1996, pp. 241–252.

[5] G. Chavent and P. Lemonnier, *Identification de la non-linearité d'une equation parabolique quasilineare*, Appl. Math. Optim., 1 (1974), pp. 121–162.

[6] J. W. Daniel, *The conjugate gradient method for linear and nonlinear operator equations*, SIAM J. Numer. Anal., 4 (1967), pp. 10–26.

[7] J. E. Dennis, Jr. and R. B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1983.

[8] D. C. Dobson and F. Santosa, *Recovery of blocky images from noisy and blurred data*, SIAM J. Appl. Math., 56 (1996), pp. 1181–1198.

[9] C. W. Groetsch, *Inverse Problems in the Mathematical Sciences*, Friedr. Vieweg & Sohn, Braunschweig, 1993.

[10] M. Hanke, *A regularizing Levenberg-Marquardt scheme, with applications to inverse groundwater filtration problems*, Inverse Problems, 13 (1997), pp. 79–95.

[11] M. Hanke and P. C. Hansen, *Regularization methods for large-scale problems*, Surveys Math. Indust., 3 (1993), pp. 253–315.

[12] MATLAB, The MathWorks, Inc., Natick, MA, 1997.

[13] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing Co., Boston, MA, 1996.

[14] F. Santosa and W. W. Symes, *Computation of the Hessian for least-squares solutions of inverse problems in reflection seismology*, Inverse Problems, 4 (1988), pp. 211–233.

[15] A. N. Tikhonov, *Regularization of incorrectly posed problems*, Soviet Math. Dokl., 4 (1963), pp. 1624–1627.

[16] C. R. Vogel and M. E. Oman, *Iterative methods for total variation denoising*, SIAM J. Sci. Comput., 17 (1996), pp. 227–238.

[17] C. R. Vogel and M. E. Oman, *A fast, robust algorithm for total variation based reconstruction of noisy, blurred images*, IEEE Trans. Image Process., 7 (1998), pp. 813–824.

[18] C. R. Vogel and J. G. Wade, *Analysis of costate discretizations in parameter estimation for linear evolution equations*, SIAM J. Control Optim., 33 (1995), pp. 227–254.

[19] A. Weiser and M. F. Wheeler, *On convergence of block-centered finite differences for elliptic problems*, SIAM J. Numer. Anal., 25 (1988), pp. 351–375.

SIAM J. MATRIX ANAL. APPL.
Vol. 20, No. 4, pp. 1038–1059

BLOCK STATIONARY METHODS FOR NONSYMMETRIC CYCLICALLY REDUCED SYSTEMS ARISING FROM THREE-DIMENSIONAL ELLIPTIC EQUATIONS*

CHEN GREIF† AND JAMES VARAH‡

Abstract. We consider a three-dimensional convection-diffusion model problem and examine systems of equations arising from performing one step of cyclic reduction on an equally spaced mesh, discretized using the seven-point operator. We present two ordering strategies and analyze block splittings of the resulting matrices. If the matrices are consistently ordered relative to a given partitioning, Young's analysis for the block Gauss–Seidel and block SOR methods can be applied. We compare partitionings for which this property holds with ones where the matrices do not have Property A yet still give rise to an efficient solution process. Bounds on convergence rates are derived and the work involved in solving the systems is estimated.

Key words. cyclic reduction, stationary methods, three-dimensional problems, convection-diffusion

AMS subject classifications. 65F10, 65N22

PII. S0895479897317715

1. Introduction.

Consider the three-dimensional (3D) convection-diffusion equation with constant coefficients

$$-(u_{xx}+u_{yy}+u_{zz})+\sigma u_x+\tau u_y+\mu u_z=p(x,y,z) \tag{1.1}$$

on the unit cube $\Omega=[0,1]\times[0,1]\times[0,1]$, subject to Dirichlet-type boundary conditions. We focus on applying seven-point finite difference discretizations, for example centered differences to the diffusive terms, and centered differences or first-order upwind approximations to the convective terms. Let us define n and h so that n^3 is the number of unknowns and $h=1/(n+1)$ is the mesh size, and let F denote the corresponding difference operator, after scaling by h^2, so that for a gridpoint $u_{i,j,k}$ not next to the boundary we have

$$\begin{aligned} F\,u_{i,j,k} &= a\,u_{i,j,k}+b\,u_{i,j-1,k}+c\,u_{i-1,j,k}\\ &\quad +d\,u_{i+1,j,k}+e\,u_{i,j+1,k}+f\,u_{i,j,k-1}+g\,u_{i,j,k+1}. \end{aligned} \tag{1.2}$$

If we denote the mesh Reynolds numbers by

$$\beta=\frac{\sigma h}{2},\quad \gamma=\frac{\tau h}{2},\quad \delta=\frac{\mu h}{2}, \tag{1.3}$$

then the values of the components of the computational molecule are given by

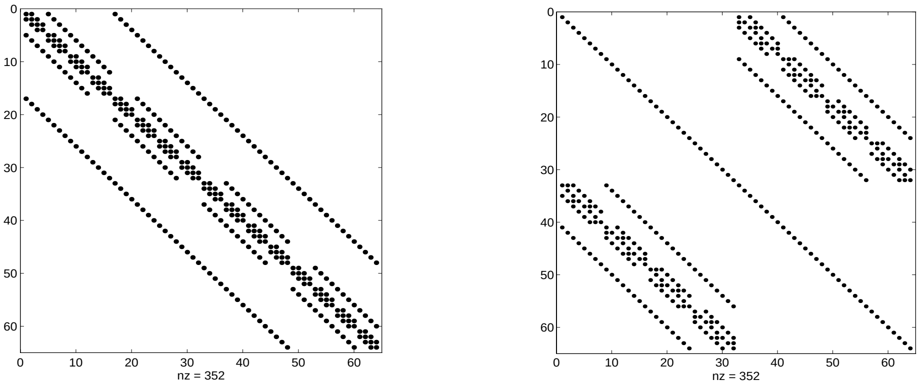
$$\begin{aligned} &a=6,\quad b=-1-\gamma,\quad c=-1-\beta,\quad d=-1+\beta,\\ &e=-1+\gamma,\quad f=-1-\delta,\quad g=-1+\delta \end{aligned} \tag{1.4}$$

*Received by the editors March 3, 1997; accepted for publication (in revised form) by Z. Strakoš February 27, 1998; published electronically July 9, 1999.

http://www.siam.org/journals/simax/20-4/31771.html

†SCCM Program, Gates Building, Stanford University, Stanford, CA 94305 (greif@sccm.stanford.edu).

‡Department of Computer Science, University of British Columbia, Vancouver, BC, V6T 1Z4, Canada (varah@cs.ubc.ca).



(**a**) lexicographic ordering (**b**) red/black ordering

FIG. 1.1. *Sparsity patterns of the matrices corresponding to two possible orderings of the unknowns*

if centered difference approximations of the first derivatives are used and by

$$a = 6 + 2(\beta + \gamma + \delta), \quad b = -1 - 2\gamma, \quad c = -1 - 2\beta,$$
$$d = -1, \quad e = -1, \quad f = -1 - 2\delta, \quad g = -1 \tag{1.5}$$

if backward first-order accurate schemes are used.

The sparsity pattern of the underlying matrix depends on the ordering of the unknowns. In Fig. 1.1 the sparsity patterns associated with two possible ordering strategies are illustrated. The natural lexicographic ordering in (a) is one where the unknowns are numbered rowwise and then planewise. The red/black ordering in (b) means we color the gridpoints using two colors, in a checkerboard fashion, and then number all the points that correspond to one of the colors first.

As is evident from Fig. 1.1(b), if we split the matrix into four blocks of the same size, we can see that the two diagonal blocks are diagonal matrices. This means that the matrix has Property A [24]. A cheap and relatively simple process of elimination of all the points that correspond to one color (say, red) leads to a smaller system of equations, whose associated matrix is the Schur complement of the original matrix, and is still fairly sparse. This procedure amounts to performing one step of cyclic reduction. Notice that in general both the original and the reduced matrices are nonsymmetric.

The cyclic reduction step can be repeated until a small system of equations is obtained, which can then be solved directly. This procedure is called *complete* cyclic reduction. It has been studied in several papers, mainly for symmetric systems arising from two-dimensional (2D) self-adjoint elliptic problems. A general overview of the algorithm and a list of references can be found in [12]. Early papers that present and analyze the algorithm are those of Hockney [17], Buneman [2], and Buzbee, Golub, and Nielson [4]. Buzbee et al. [3] use cyclic reduction for solving the Poisson equation on irregular regions; Concus and Golub [6] discuss 2D nonseparable cases. Application of cyclic reduction to matrices with arbitrary dimensions is done by Sweet [20], [21]. Detyna [7] presents a fast $O(n^2)$ algorithm and discusses its stability and efficiency.

One step of cyclic reduction for symmetric positive definite systems is analyzed by Hageman and Varga in [16] and later by Hageman, Luk, and Young [15], where it is shown that the reduced solver generally converges faster than the unreduced solver.

In [1], Axelsson and Gustafsson use cyclic reduction in conjunction with the conjugate gradient method. Elman and Golub have conducted an extensive investigation for 2D elliptic non-self-adjoint problems [8], [9], [10] and have shown that one step of cyclic reduction leads to systems with several valuable properties, such as symmetrizability for a large set of the underlying PDE coefficients, which is effectively used to derive bounds on the convergence rates of iterative solvers, and fast convergence.

Preliminary analysis for the non-self-adjoint 3D model problem (1.1) has been done by the authors in [14], where one step of 3D cyclic reduction has been described in detail, and a block Jacobi solver has been analyzed, which is based on a certain block splitting (referred to as 1D splitting throughout this paper), in conjunction with what we called a two-plane ordering strategy.

The computational molecule of the reduced operator consists of 19 points, located on 5 parallel planes. Let R denote the reduced difference operator, after scaling by ah^2. Then for an interior gridpoint, $u_{i,j,k}$, we have

$$\begin{aligned} R\,u_{i,j,k} = {} & (a^2-2be-2cd-2fg)\,u_{i,j,k} - f^2\,u_{i,j,k-2} - 2ef\,u_{i,j+1,k-1} \\ & -2cf\,u_{i-1,j,k-1} - 2df\,u_{i+1,j,k-1} - 2bf\,u_{i,j-1,k-1} - e^2\,u_{i,j+2,k} \\ & -2de\,u_{i+1,j+1,k} - c^2\,u_{i-2,j,k} - d^2\,u_{i+2,j,k} - 2bc\,u_{i-1,j-1,k} \\ & -b^2\,u_{i,j-2,k} - 2eg\,u_{i,j+1,k+1} - 2cg\,u_{i-1,j,k+1} - 2ce\,u_{i-1,j+1,k} \\ & -2bd\,u_{i+1,j-1,k} - 2dg\,u_{i+1,j,k+1} - 2bg\,u_{i,j-1,k+1} - g^2\,u_{i,j,k+2}\ . \end{aligned} \tag{1.6}$$

The following results hold for any ordering strategy. See [14] for the proofs, which have been obtained by using the techniques of Elman and Golub [8], [9].

THEOREM 1.1. *The reduced matrix can be symmetrized by a real diagonal similarity transformation if and only if the products $bcde$, $befg$, and $cdfg$ are positive.*

THEOREM 1.2. *If be, cd, $fg > 0$, then both the reduced matrix and the symmetrized reduced matrix are diagonally dominant M-matrices.*

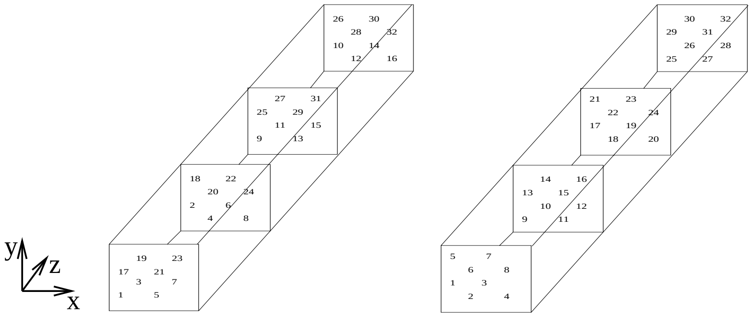
In this paper our purpose is to extend the analysis initiated in [14] and examine block stationary methods as solvers for the reduced system. In section 2 we present the ordering strategies that are examined. In section 3 two block splittings are presented, and bounds on convergence rates are derived. In section 4 we analyze the reduced system in the context of consistently ordered matrices. In section 5 the amount of computational work involved in solving the linear systems is estimated, a comparison of the reduced system with the unreduced system is conducted, and some numerical results which validate our analysis and illustrate the fast convergence of the reduced system are given. Finally, in section 6 we conclude.

2. Orderings for the reduced system. We consider two ordering strategies for the reduced grid. The two-plane ordering has been described in detail in [14]. It corresponds to ordering the unknowns by gathering blocks of $2n$ gridpoints from two horizontal lines and two adjacent planes. This ordering strategy is depicted in Fig. 2.1(a). In the figure, the numbers are the indices of the gridpoints, which are to be expressed below by ℓ in (2.1) and (2.15).

The connection between the index of a gridpoint, ℓ, and its coordinate values $(i,j,k) = (\frac{x}{h}, \frac{y}{h}, \frac{z}{h})$ is given below. The term fix is borrowed from MATLAB and means rounding to the nearest integer toward zero.

$$i = \text{fix}\{[(\ell-1) \bmod (2n)]/2\} + 1, \tag{2.1a}$$

$$j = \begin{cases} 2\cdot[\text{fix}(\frac{\ell-1}{n^2})+1], & \ell \bmod 4 = 0 \text{ or } 1, \\ 2\cdot\text{fix}(\frac{\ell-1}{n^2})+1, & \ell \bmod 4 = 2 \text{ or } 3, \end{cases} \tag{2.1b}$$



(a) two-plane **(b)** two-line

FIG. 2.1. *Two suggested ordering strategies for the reduced grid (in the figure the unreduced grid is of size* $4 \times 4 \times 4$*).*

$$k = \begin{cases} 2 \cdot \text{fix}(\frac{(\ell-1) \bmod n^2}{2n}) + 1, & \ell \ \text{ odd,} \\ 2 \cdot [\text{fix}(\frac{(\ell-1) \bmod n^2}{2n}) + 1], & \ell \ \text{ even.} \end{cases} \tag{2.1c}$$

See [14] for specification of the matrix entries.

An alternative to the two-plane ordering is a straightforward generalization to three dimensions of the two-line ordering used by Elman and Golub in [9]. It is illustrated in Fig. 2.1(b). The reduced matrix for this ordering strategy is block pentadiagonal:

$$S = \text{penta}[S_{j,j-2}, S_{j,j-1}, S_{j,j}, S_{j,j+1}, S_{j,j+2}]. \tag{2.2}$$

Each $S_{i,j}$ is $(n^2/2) \times (n^2/2)$ and is a combination of $\frac{n}{2}$ uncoupled matrices, each of size $n \times n$.

The diagonal matrices $\{S_{j,j}\}$ are themselves block tridiagonal. Each submatrix is of size $n \times n$ and its diagonal block is

$$S_{j,j}^{(0)} = \begin{cases} \text{penta}(-c^2, \ -2bc \cdot E_{01} - 2ce \cdot E_{10}, \ *, \ -2bd \cdot E_{01} - 2de \cdot E_{10}, \ -d^2), & j \ \text{ odd,} \\ \text{penta}(-c^2, -2bc \cdot E_{10} - 2ce \cdot E_{01}, *, -2bd \cdot E_{10} - 2de \cdot E_{01}, \ -d^2), & j \ \text{ even,} \end{cases}$$

where $E_{10} = (1, 0, 1, 0, \dots)$, $E_{01} = (0, 1, 0, 1, \dots)$, and $*$ stands for the value along the main diagonal of the matrix, which is given by $a^2 - 2cd - 2be - 2fg$ for gridpoints not next to the boundary. See [14] for specification of the main diagonal's values associated with gridpoints next to the boundary.

For the superdiagonal and the subdiagonal blocks of the matrices $S_{j,j}$ we have the following irregular tridiagonal structure, which depends on whether j is even or odd. The superdiagonal matrices are given by

$$S_{j,j}^{(1)} = \begin{pmatrix} -e^2 & -2de & & & & & \\ & -e^2 & & & & & \\ & -2ce & -e^2 & -2de & & & \\ & & & \ddots & & & \\ & & & & \ddots & & \\ & & & & -2ce & -e^2 & -2de \\ & & & & & & -e^2 \end{pmatrix} \tag{2.3}$$

if j is odd or

$$S_{j,j}^{(1)} = \begin{pmatrix} -e^2 & & & & & & \\ -2ce & -e^2 & -2de & & & & \\ & & -e^2 & & & & \\ & & & \ddots & & & \\ & & & & \ddots & & \\ & & & & & -e^2 & \\ & & & & & -2ce & -e^2 \end{pmatrix} \tag{2.4}$$

if j is even.

The subdiagonal matrices are

$$S_{j,j}^{(-1)} = \begin{pmatrix} -b^2 & & & & & & \\ -2bc & -b^2 & -2bd & & & & \\ & & -b^2 & & & & \\ & & & \ddots & & & \\ & & & & \ddots & & \\ & & & & & -b^2 & \\ & & & & & -2bc & -b^2 \end{pmatrix} \tag{2.5}$$

if j is odd or

$$S_{j,j}^{(-1)} = \begin{pmatrix} -b^2 & -2bd & & & & & \\ & -b^2 & & & & & \\ & -2bc & -b^2 & -2bd & & & \\ & & & \ddots & & & \\ & & & & \ddots & & \\ & & & & -2bc & -b^2 & -2bd \\ & & & & & & -b^2 \end{pmatrix} \tag{2.6}$$

if j is even.

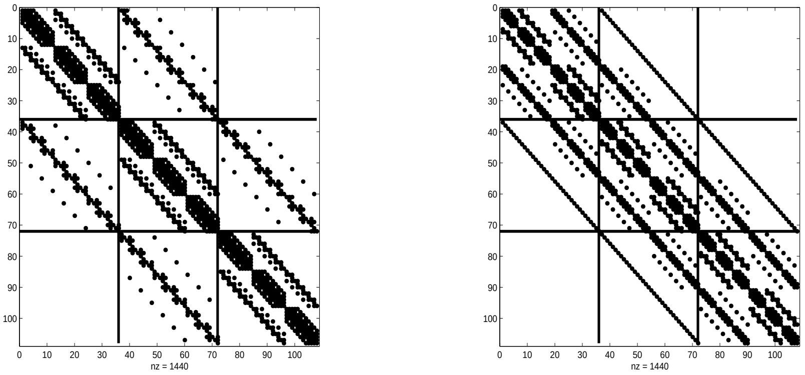
The superdiagonal and the subdiagonal blocks of S, $S_{j,j\pm1}$ are block tridiagonal:

$$S_{j,j-1}^{(-1)} = \begin{cases} \operatorname{diag}(-2bf \cdot E_{01}), & j \text{ odd}, \\ \operatorname{diag}(-2bf \cdot E_{10}), & j \text{ even}, \end{cases} \tag{2.7}$$

$$S_{j,j-1}^{(0)} = \begin{cases} \operatorname{tri}(-2cf,\ -2bf \cdot E_{10} - 2ef \cdot E_{01},\ -2df), & j \text{ odd}, \\ \operatorname{tri}(-2cf,\ -2bf \cdot E_{01} - 2ef \cdot E_{10},\ -2df), & j \text{ even}, \end{cases} \tag{2.8}$$

$$S_{j,j-1}^{(1)} = \begin{cases} \operatorname{diag}(-2ef \cdot E_{10}), & j \text{ odd}, \\ \operatorname{diag}(-2ef \cdot E_{01}), & j \text{ even}, \end{cases} \tag{2.9}$$

$$S_{j,j+1}^{(-1)} = \begin{cases} \operatorname{diag}(-2bg \cdot E_{01}), & j \text{ odd}, \\ \operatorname{diag}(-2bg \cdot E_{10}), & j \text{ even}, \end{cases} \tag{2.10}$$



(**a**) two-plane (**b**) two-line

FIG. 2.2. *Sparsity patterns of the reduced matrices associated with the two ordering strategies (the matrices correspond to* $6 \times 6 \times 6$ *grids). Each square corresponds to an* $n^2 \times n^2$ *block (*n^2 *gridpoints form two coupled planes in the reduced grid).*

$$S^{(0)}_{j,j+1} = \begin{cases} \mathrm{tri}(-2cg,\ -2bg \cdot E_{10} - 2eg \cdot E_{01},\ -2dg), & j \text{ odd}, \\ \mathrm{tri}(-2cg,\ -2bg \cdot E_{01} - 2eg \cdot E_{10},\ -2dg), & j \text{ even}, \end{cases} \tag{2.11}$$

$$S^{(1)}_{j,j+1} = \begin{cases} \mathrm{diag}(-2eg \cdot E_{10}), & j \text{ odd}, \\ \mathrm{diag}(-2eg \cdot E_{01}), & j \text{ even}. \end{cases} \tag{2.12}$$

Finally, the matrices $S_{j,j-2}$ and $S_{j,j+2}$ are diagonal:

$$S_{j,j-2} = \mathrm{diag}(-f^2), \qquad j = 3, \ldots, n\ ; \tag{2.13}$$

$$S_{j,j+2} = \mathrm{diag}(-g^2), \qquad j = 1, \ldots, n-2. \tag{2.14}$$

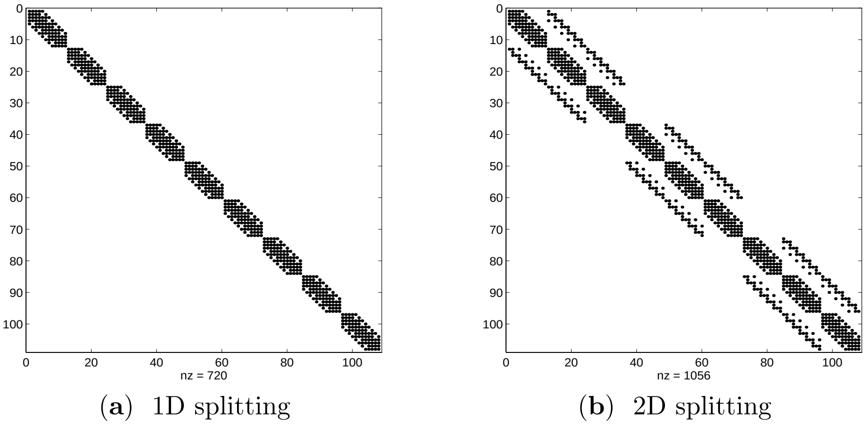
The connection between the gridpoint's index and its coordinate values is given by

$$i = [(\ell - 1) \bmod n] + 1, \tag{2.15a}$$

$$k = \mathrm{fix}\left(\frac{\ell - 1}{n^2/2}\right) + 1, \tag{2.15b}$$

$$j = \begin{cases} 2 \cdot (\mathrm{fix}((\mathrm{fix}(\ell - (k-1) \cdot (n^2/2)) - 1)/n)) + 1, & k \text{ odd}, \\ 2 \cdot (\mathrm{fix}((\mathrm{fix}(\ell - (k-1) \cdot (n^2/2)) - 1)/n)) + 2, & k \text{ even}. \end{cases} \tag{2.15c}$$

The sparsity patterns of the matrices corresponding to two-plane ordering and two-line ordering are depicted in Fig. 2.2.



(**a**) 1D splitting (**b**) 2D splitting

FIG. 3.1. *Sparsity patterns of the block diagonal matrices associated with the block Jacobi splitting, for the two suggested block splittings, using the two-plane ordering strategy.*

3. Block splittings and bounds on convergence rate. For the two ordering strategies presented in section 2 the matrices can be expressed as block tridiagonal, of the form

$$S = \text{tri}[S_{j,j-1}, S_{j,j}, S_{j,j+1}] \ . \tag{3.1}$$

S is an $(n^3/2) \times (n^3/2)$ matrix. In the case of two-plane ordering, each block $S_{i,j}$ is of size $n^2 \times n^2$ and is block tridiagonal with respect to $2n \times 2n$ blocks. In the case of two-line ordering, each block $S_{i,j}$ is of size $(n^2/2) \times (n^2/2)$ and is block tridiagonal with respect to $n \times n$ blocks.

In solving the reduced system using a stationary method, various splittings are possible. We consider two obvious ones, based on dimension. We use the term 1D splitting for a splitting which is based on partitioning the matrix into $O(n)$ blocks ($2n \times 2n$ blocks for the two-plane ordering and $n \times n$ blocks for the two-line ordering). A 2D splitting is one which is based on partitioning the matrix into $O(n^2)$ blocks ($n^2 \times n^2$ blocks for the two-plane ordering and $(n^2/2) \times (n^2/2)$ blocks for the two-line ordering). Notice that the 1D splitting for both ordering strategies is essentially associated with blocks of gridpoints that are x-oriented. However, the 2D splitting for the two-line ordering corresponds to x-y oriented planes of gridpoints, whereas for the two-plane ordering it corresponds to x-z oriented planes of gridpoints. (These observations can be deduced by referring to Fig. 2.1.) Different orientations can be obtained by simply reordering the unknowns so that the roles of x, y, and z are interchanged.

The sparsity patterns of the block diagonal parts of the splittings associated with the block Jacobi scheme are depicted in Fig. 3.1.

We now compare the orderings. We have the following useful result.

THEOREM 3.1. *If be, cd, $fg > 0$, then for the* 1*D splitting the spectral radius of the Jacobi iteration matrix associated with two-plane ordering is smaller than the spectral radius of the iteration matrix associated with the two-line ordering.*

Proof. By Theorem 1.2 the matrices are M-matrices. Each ordering strategy produces a matrix which is merely a symmetric permutation of a matrix associated with the other ordering. Suppose $S_1 = M_1 - N_1$ is a 1D splitting of the two-plane ordering matrix and $S_2 = M_2 - N_2$ is a 1D splitting for the two-line ordering. There exists a permutation matrix P such that $P^T S_2 P = S_1$. Consider the splitting $P^T S_2 P = P^T M_2 P - P^T N_2 P$. It is straightforward to show by examining the matrix entries that $P^T N_2 P \geq N_1$. The latter are both nonnegative matrices; therefore by [23, Thm. 3.15] it follows that $0 < \rho(M_1^{-1} N_1) < \rho(P^T M_2^{-1} N_2 P) = \rho(M_2^{-1} N_2) < 1$. □

The same result applies to 2D splitting, provided that the orientation of the planes of gridpoints is identical for both ordering strategies. The proof for this is identical to the proof of Theorem 3.1.

The results indicated in Theorem 3.1 can be observed in Fig. 3.2. It is interesting to observe that the superiority of the two-plane ordering carries over to the case $be,\ cd,\ fg < 0$, which corresponds to the region of mesh Reynolds numbers larger than 1 (for which the PDE is considered convection-dominated). We remark, however, that the amount of computational work per each iteration is somewhat higher for the system which corresponds to two-plane ordering. In Fig. 3.2 a few cross sections of mesh Reynolds numbers are examined. For example, graph (a) corresponds to flow with the same velocity in $x,\ y$, and z directions. Graph (b) corresponds to flow only in x and y directions, and no convection in z direction, and so on. (See (1.3) for definitions of $\beta,\ \gamma$, and δ.)

We now derive bounds on convergence rates. Below we shall attach the subscripts 1 and 2 to matrices associated with the 1D splitting and 2D splitting, respectively. Since two-plane ordering gives rise to a more-efficient solution procedure than two-line ordering, we focus on it.

Denote the two splittings for the block Jacobi scheme by $S = D_1 - C_1 = D_2 - C_2$. In [14] we have shown that if be, cd, and fg have the same sign then a real diagonal nonsingular symmetrizer can be found, and thus (since the symmetrizer is diagonal) the sparsity patterns of the original nonsymmetric matrix and the symmetrized matrix are identical. Let us attach the hat sign to a matrix to denote application of the similarity transformation that symmetrizes it. That is, for a given matrix X and a diagonal symmetrizer Q, $Q^{-1} X Q$ is to be denoted by $\hat{X}$.

The matrices $\hat{D}_1^{-1} \hat{C}_1$ and $\hat{D}_2^{-1} \hat{C}_2$ are similar to the original iteration matrices $D_1^{-1} C_1$ and $D_2^{-1} C_2$, respectively, and thus have the same spectral radii. Following Elman and Golub's strategy [8], [9], the symmetric matrix can be handled more easily as far as computing the spectral radius is concerned, since we can use the following:

$$\rho(D_i^{-1} C_i) = \rho(\hat{D}_i^{-1} \hat{C}_i) \leq ||\hat{D}_i^{-1}||_2 ||\hat{C}_i||_2 = \frac{\rho(\hat{C}_i)}{\lambda_{min}(\hat{D}_i)}, \qquad i = 1, 2. \tag{3.2}$$

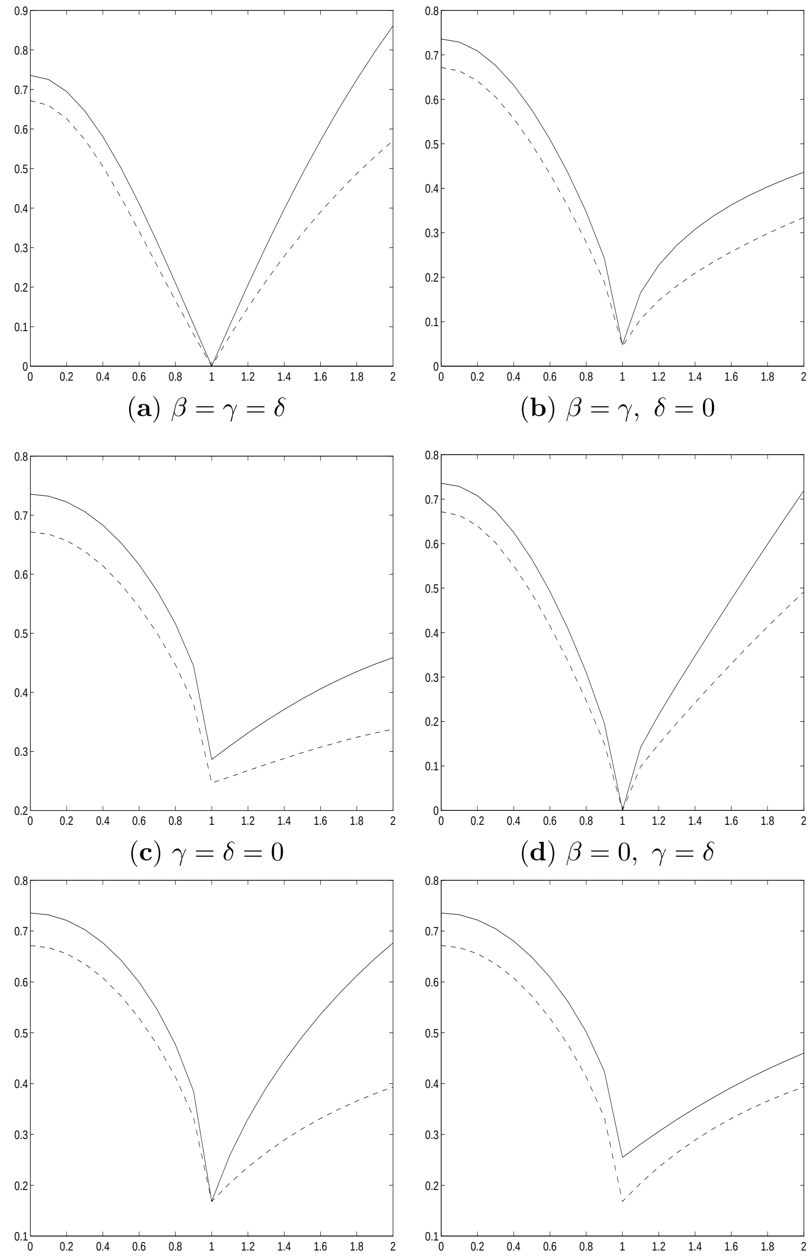
The results presented below are for the case $be,\ cd,\ fg > 0$, using two-plane ordering. These conditions are equivalent to $|\beta|,\ |\gamma|,\ |\delta| < 1$ if centered differences are used to discretize the convective terms. No restriction on the magnitude of the mesh Reynolds numbers is imposed if upwind differences are used. For these values tight bounds for the spectral radius of the iteration matrix can be obtained.

For $\hat{D}_1$ the minimal eigenvalue has been found in [14, Thm. 3.8], and the relevant part of this theorem is quoted below.

Proposition 3.2. *The minimal eigenvalue of $\hat{D}_1$ is*

$$\eta = a^2 - 2be - 2fg - 2\sqrt{befg} - 4(\sqrt{bcde} + \sqrt{cdfg}) \cdot \cos(\pi h) - 4cd\ \cos^2(\pi h)\ . \tag{3.3}$$

A lower bound for $\hat{D}_2 - \hat{D}_1$ is given by the following proposition.



(a) $\beta = \gamma = \delta$ **(b)** $\beta = \gamma,\ \delta = 0$

(c) $\gamma = \delta = 0$ **(d)** $\beta = 0,\ \gamma = \delta$

(e) $\beta = \gamma = 0$ **(f)** $\beta = \delta = 0$

FIG. 3.2. *Spectral radii of iteration matrices versus mesh Reynolds numbers for the block Jacobi scheme, using* 1*D splitting and centered difference discretization. The broken lines correspond to two-plane ordering. The solid lines correspond to two-line ordering.*

PROPOSITION 3.3. *The minimal eigenvalue of $\hat{D}_2 - \hat{D}_1$ is bounded from below by $-\xi$, where*

$$\xi = 2fg \cos\left(\frac{\pi}{\frac{n}{2}+1}\right) + \sqrt{4befg + 16 \cdot cdfg \cdot \cos^2(\pi h) + 16\sqrt{bcde} \cdot fg \cdot \cos(\pi h)}. \tag{3.4}$$

TABLE 3.1

Comparison between the computed spectral radius and the bound, for the 2D splitting, with $\beta = \gamma = \delta = 0.5$.

Scheme	Upwind			Centered		
n	ρ	bound	ratio	ρ	bound	ratio
4	0.265	0.430	1.62	0.203	0.309	1.52
6	0.411	0.530	1.29	0.297	0.368	1.24
8	0.499	0.583	1.17	0.350	0.398	1.14
10	0.553	0.615	1.11	0.381	0.415	1.09
12	0.588	0.634	1.08	0.400	0.425	1.06
14	0.611	0.647	1.06	0.413	0.432	1.05

The proof for this part follows from [14, Lem. 3.10], where it is shown that the spectral radius of $\hat{D}_2 - \hat{D}_1$ is bounded by ξ.

Combining Propositions 3.2 and 3.3, and applying Rayleigh quotients to the matrices $\hat{D}_1$ and $\hat{D}_2 - \hat{D}_1$, we obtain the following lemma.

LEMMA 3.4. *The minimal eigenvalue of $\hat{D}_2$ is bounded from below by $\eta - \xi$, where η and ξ are the expressions given in* (3.3) *and* (3.4).

The bound for $\hat{C}_2$ can be obtained by combining [14, Lems. 3.11–3.13], as follows.

LEMMA 3.5. *The spectral radius of the matrix $\hat{C}_2$ is bounded by*

$$\phi = 4\sqrt{befg} + 4\sqrt{bcde} \cdot \cos\frac{\pi}{n+1} + 2be\cos\left(\frac{\pi}{\frac{n}{2}+1}\right). \tag{3.5}$$

Finally, Lemmas 3.4 and 3.5 lead to the following theorem.

THEOREM 3.6. *The spectral radii of the iteration matrices $D_1^{-1}C_1$ and $D_2^{-1}C_2$ are bounded by $\frac{\phi+\xi}{\eta}$ and $\frac{\phi}{\eta-\xi}$, respectively, where η, ξ, and ϕ are defined in* (3.3), (3.4), *and* (3.5), *respectively.*

COROLLARY 3.7. *If be, cd, $fg > 0$ then the block Jacobi iteration converges for both the 1D and 2D splittings.*

Proof. For this we can use Varga's result on M-matrices [23, Thm. 3.13]. Alternatively, Taylor expansions of the bounds given in Theorem 3.6 are given by

$$\rho(D_1^{-1}C_1) \le \frac{\phi+\xi}{\eta} = 1 - \left(\frac{10}{9}\pi^2 + \frac{1}{6}\mu^2 + \frac{1}{6}\tau^2 + \frac{1}{6}\sigma^2\right)h^2 + o(h^2) \tag{3.6}$$

and

$$\rho(D_2^{-1}C_2) \le \frac{\phi}{\eta-\xi} = 1 - \left(2\pi^2 + \frac{3}{10}\mu^2 + \frac{3}{10}\tau^2 + \frac{3}{10}\sigma^2\right)h^2 + o(h^2) \tag{3.7}$$

and thus are smaller than 1. □

In Table 3.1 we give some indication on the quality of the bound for the 2D splitting. Results with a similar level of accuracy have been obtained and presented in [14] for the 1D splitting. As can be observed, the bounds are tight and become tighter as n increases, which suggests that they are asymptotic to the spectral radii.

We now discuss other stationary methods, namely, Gauss–Seidel and SOR. Relative to a given partitioning, if the reduced matrix is consistently ordered, then it is straightforward to apply Young's analysis, and the bounds in Theorem 3.6 can be used for estimating the rate of convergence of the Gauss–Seidel and SOR schemes. The reader is referred to [19, Defs. 4.3 and 4.4] for definitions of Property A and consistent ordering. As stated in [19], a matrix that is consistently ordered has Property

A; conversely, a matrix with Property A can be permuted so that it is consistently ordered. We mentioned in the introduction that the matrix of the unreduced system has Property A. For the reduced system, we have the following observations.

PROPOSITION 3.8. *The reduced matrix associated with two-line ordering, S_L, does not have Property A relative to* 1*D or* 2*D partitionings.*

Proof. Let $S_{i,j}$ denote the (i,j)th $n \times n$ block of S_L, and let Q be an $(n^2/2) \times (n^2/2)$ matrix, whose entries satisfy $q_{i,j} = 1$ if $S_{i,j} \neq 0$ and $q_{i,j} = 0$ otherwise. Let T be an $n \times n$ matrix, such that $t_{i,j} = 1$ if the (i,j)th $(n^2/2) \times (n^2/2)$ block submatrix of S is nonzero, and $t_{i,j} = 0$ otherwise. Clearly, T is a pentadiagonal matrix and thus does not have Property A. Since Q can be referred to as a partitioning of T into $(n/2) \times (n/2)$ blocks, it also does not have Property A. □

PROPOSITION 3.9. *The reduced matrix associated with two-plane ordering, S_P, does not have Property A relative to* 1*D partitioning.*

Proof. Let $S_{i,j}$ denote the (i,j)th $2n \times 2n$ block of S_P, and let Q be an $(n^2/4) \times (n^2/4)$ matrix, whose entries satisfy $q_{i,j} = 1$ if $S_{i,j} \neq 0$ and $q_{i,j} = 0$ otherwise. It is straightforward to see that the nonzero pattern of Q is identical to that of the matrix associated with using a nine-point operator for a 2D grid. Since the latter does not have Property A relative to partitioning into 1×1 matrices, the result follows. □

On the other hand, we have the following proposition.

PROPOSITION 3.10. *The reduced matrix associated with two-plane ordering, S_P, has Property A and, moreover, is consistently ordered relative to* 2*D partitioning.*

Proof. The matrix is block tridiagonal relative to this partitioning [24]. □

For the SOR scheme we have the following result, which is completely analogous to Elman and Golub's result for the 2D problem [9, Thm. 4].

THEOREM 3.11. *Let $\mathcal{L}_\omega$ denote the block SOR operator associated with* 2*D splitting and using two-plane ordering. If either be, cd, fg > 0 or < 0, then the choice $\omega^* = \frac{2}{1+\sqrt{1-\rho^2(D_2^{-1}C_2)}}$ minimizes $\rho(\mathcal{L}_\omega)$ with respect to ω, and $\rho(\mathcal{L}_{\omega^*}) = \omega^* - 1$.*

The proof of this theorem is essentially identical to the proof of Elman and Golub in [9, Thm. 4] and follows from Young [24, Chap. 14, Sects. 5.2 and 14.3]. The algebraic details on how to pick the signs of the diagonal symmetrizer so that the symmetrized block diagonal part of the splitting is a diagonally dominant M-matrix are omitted. That $\rho(D_2^{-1}C_2) < 1$ is known by Corollary 3.7. The reduced matrix is consistently ordered by Proposition 3.10.

A way to approximately determine an optimal relaxation parameter for the case be, cd, $fg > 0$ is to replace $\rho(D_2^{-1}C_2)$ by the bound for it (given in Theorem 3.6) in the expression for ω^* in Theorem 3.11. If the bound for the block Jacobi scheme is tight, then the estimate of ω^* is fairly accurate.

PROPOSITION 3.12. *Suppose be, cd, $fg > 0$. For the system associated with* 2*D splitting and for h sufficiently small, the choice*

$$\tilde{\omega}^* = \frac{2(\eta - \xi)}{\eta - \xi + \sqrt{(\eta - \xi)^2 - \phi^2}} \tag{3.8}$$

approximately minimizes $\rho(\mathcal{L}_\omega)$. The spectral radius of the iteration matrix is approximately $\tilde{\omega}^* - 1$.

The Taylor expansion of the estimate for the optimal relaxation parameter is given by

$$\tilde{\omega}^* = 2 - \frac{2}{5}\sqrt{100\pi^2 + 15(\tau^2 + \mu^2 + \sigma^2)} \cdot h + O(h^2). \tag{3.9}$$

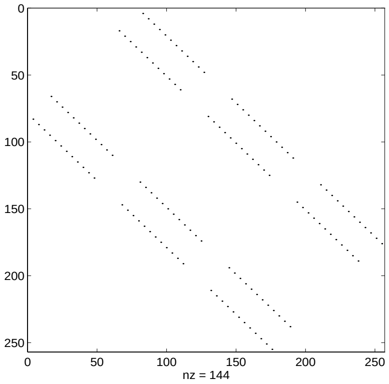


FIG. 4.1. *The sparsity pattern of the matrix* C_d.

From (3.9) it follows that the estimated asymptotic rate of convergence of the block SOR scheme is approximately the second term in (3.9) (with the negative sign removed) and is thus $O(h)$.

4. Near-Property A for 1D splitting of the two-plane matrix. Although the matrix associated with two-plane ordering does not have Property A relative to the 1D partitioning, some interesting observations can be made: As before, let $\{S_{i,j}\}$ denote the $n^2 \times n^2$ blocks of the reduced matrix. Each block $S_{i,j}$ is a block tridiagonal matrix relative to $2n \times 2n$ blocks. We attach superscripts to mark how far a block diagonal is from the main block diagonal, and we define

$$C_d = -\left(S^{(-1)}_{j,j+1} + S^{(1)}_{j,j+1} + S^{(-1)}_{j,j-1} + S^{(1)}_{j,j-1}\right). \tag{4.1}$$

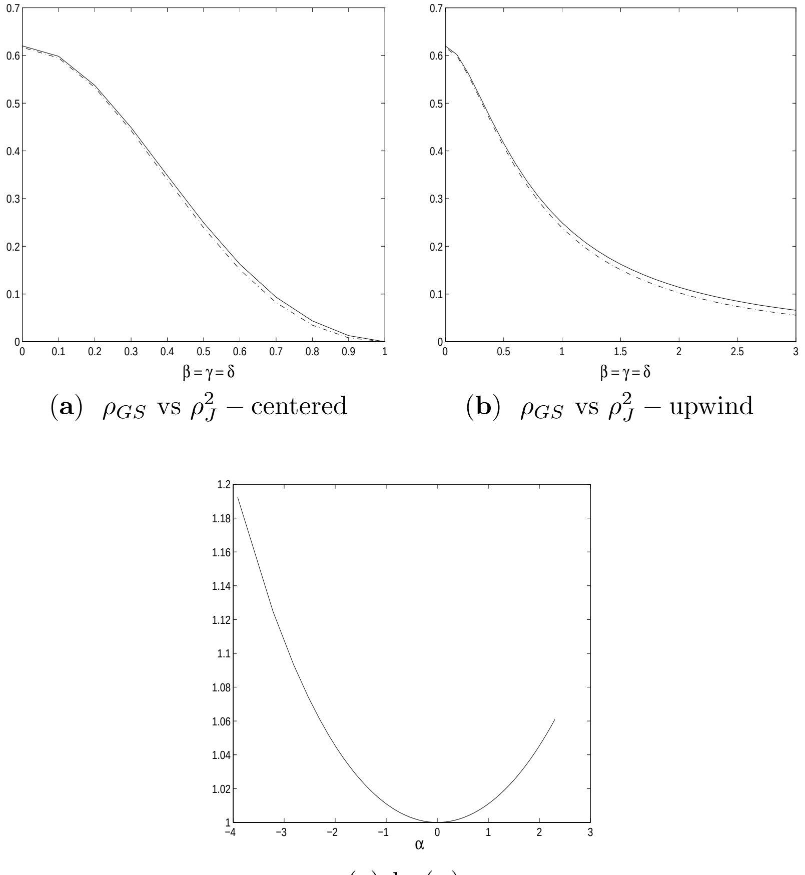
See [14] for specification of the entries of these matrices. As in section 3 (with a slight change in notation), let $S_P = D - C$ be the 1D splitting of the matrix, and define $\tilde{C}$ so that $S_P = D - (\tilde{C} + C_d)$. The matrix $D - \tilde{C}$ has Property A, but S_P does not. Let us examine the matrix that prevents S_P from having Property A, namely, C_d. It is an extremely sparse matrix, and the magnitude of the nonzero values in this matrix is bounded by 2 if $be,\ cd,\ fg > 0$. The nonzero pattern of C_d is depicted in Fig. 4.1.

We wish to estimate how far the reduced matrix S_P is from having block Property A, relative to the 1D partitioning. Let us denote the upper part and the lower part of C_d by U_d and L_d, respectively, and let $\tilde{U}$ and $\tilde{L}$ be the upper part and lower part of $\tilde{C}$, respectively. Then the spectral radius of the block Gauss–Seidel matrix satisfies

$$\rho_{GS} = \rho((D - \tilde{L} - L_d)^{-1}(\tilde{U} + U_d)) \ \le \frac{||(D - \tilde{L})^{-1}\tilde{U}||_2 + ||(D - \tilde{L})^{-1}U_d||_2}{1 - ||(D - \tilde{L})^{-1}L_d||_2}\ . \tag{4.2}$$

$||(D - \tilde{L})^{-1}\tilde{U}||_2$ is significantly larger than the other norms in the above inequality, which means that the spectral radius of the Gauss–Seidel iteration matrix associated with the two-plane ordering can be estimated by replacing the two-plane matrix by $D - \tilde{C}$, which does have Property A and thus is easier to analyze. Alternatively, the following observation has been obtained by numerical experiments:

$$\rho(D^{-1}C) \approx \rho(D^{-1}\tilde{C}) + \rho(D^{-1}C_d). \tag{4.3}$$



(**a**) ρ_{GS} vs ρ_J^2 – centered (**b**) ρ_{GS} vs ρ_J^2 – upwind

(**c**) $h_B(\alpha)$

FIG. 4.2. *"Near Property A" for the* 1*D splitting.*

Young's analysis can be applied directly to both $D - \tilde{C}$ and $D - C_d$ (both have Property A), and thus an approximate relationship between the eigenvalues of the block Jacobi iteration matrix and the eigenvalues of the block Gauss–Seidel iteration matrix can be obtained.

For $be,\ cd,\ fg > 0$ we have observed that the spectral radius of the block Jacobi iteration matrix satisfies

$$\rho_J^2 \approx \rho_{GS}. \tag{4.4}$$

The first two graphs in Fig. 4.2 illustrate this phenomenon numerically. The broken lines in graphs (a) and (b) correspond to the square of the spectral radius of the iteration matrix associated with block Jacobi, for a 256 × 256 matrix. The solid lines correspond to the spectral radius of the block Gauss–Seidel iteration matrix. As can be seen, the curves are almost indistinguishable. This phenomenon becomes more dramatic as the systems become larger.

Some analysis can be done using Varga's work on extensions of the theory of p-cyclic matrices [22], [23, Sect. 4.4]. (In this paper we are concerned only with $p = 2$.) Recall [23, Def. 4.2], which defines a set S of matrices as follows. The square matrix $B \in S$ if B satisfies the following properties:

1. $B \geq 0$ with zero diagonal entries.
2. B is irreducible and convergent, i.e., $0 < \rho(B) < 1$.
3. B is symmetric.

If be, cd, $fg > 0$, the reduced matrix S_P is a diagonally dominant M-matrix which can be symmetrized, and $\hat{D}^{1/2}$ is well defined. Define $\tilde{S} = \hat{D}^{-1/2}\hat{S}\hat{D}^{-1/2} = I - \hat{D}^{-1/2}\hat{C}\hat{D}^{-1/2}$.

Applying block Jacobi to the original reduced system is analogous to applying point Jacobi to $\tilde{S}$, in the sense that the spectra of the iteration matrices associated with both systems are identical. The iteration matrix associated with $\tilde{S}$ is $B = \hat{D}^{-1/2}\hat{C}\hat{D}^{-1/2}$. Showing that the matrix B belongs to the set S defined above is easy and is omitted. Let L be the lower part of B. Define $M_B(\theta) = \theta L + L^T, \theta \geq 0$, and $m_B(\theta) = \rho(M_B(\theta))$. Let

$$h_B(\ln\theta) = \frac{m_B(\theta)}{\rho(B)\theta^{1/2}}, \quad \theta > 0. \tag{4.5}$$

Then we have [23, Thm. 4.7] (with a slight modification so as to match the terminology used in this paper), as follows.

THEOREM 4.1. *Let $B \in S$. Then, $h_B(\alpha) \equiv 1$ if and only if B is consistently ordered.*

In some sense $h_B(\ln\theta)$ measures the departure of the matrix B from having block Property A. For matrices that are not consistently ordered, the following result applies [23, Thm. 4.6].

THEOREM 4.2. *If $B \in S$, then either $h_B(\alpha) \equiv 1$ for all real α, or $h_B(\alpha)$ is strictly increasing for $\alpha \geq 0$. Moreover, for any $\alpha \neq 0$,*

$$1 \leq h_B(\alpha) \leq \cosh\left(\frac{\alpha}{2}\right). \tag{4.6}$$

Figure 4.2(c) demonstrates how close the function h_B is to 1 for the reduced matrix when 1D partitioning is used and provides another way to illustrate the near-Property A of the matrix. In the figure, the function h_B is computed for a symmetrized block Jacobi 256×256 matrix, where $\beta = \gamma = \delta = 0.5$.

We can now analyze the Gauss–Seidel and SOR schemes. Recall [23, Thm. 4.8] (slightly modified), as follows.

THEOREM 4.3. *Let $\mathcal{L}_{B,\omega}$ denote the SOR iteration matrix. If $B \in S$ then the Gauss–Seidel iteration matrix, which corresponds to the case $\omega = 1$, satisfies*

$$\rho^2(B) \leq \rho(\mathcal{L}_{B,1}) < \frac{\rho(B)}{2 - \rho(B)}, \tag{4.7}$$

with equality possible only if B is consistently ordered.

This is a sharpened form of the Stein–Rosenberg theorem [23]. Applying this theorem to our reduced matrix, we have the following theorem.

THEOREM 4.4. *If the bound for the block Jacobi iteration matrix tends to the actual spectral radius as $h \to 0$, then the spectral radius of the block Gauss–Seidel iteration matrix coincides with the square of the bound for the spectral radius of the block Jacobi iteration matrix up to $O(h^2)$ terms.*

Proof. Since the iteration matrix B has the same spectral radius as $D^{-1}C$, where $S_P = D - C$, we can use the bound for the 1D iteration matrix, which was presented in Theorem 3.6. For simplicity of notation, denote this bound by Φ. Clearly, since

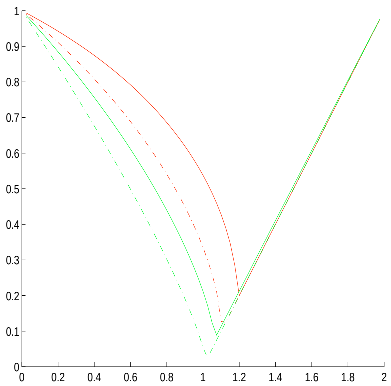


FIG. 4.3. *Spectral radius of the SOR iteration matrix versus the relaxation parameter. The uppermost curve corresponds to* $1D$ *splitting for the unreduced system, and then we have, in order,* $2D$ *splitting for the unreduced system,* $1D$ *splitting for the reduced system, and* $2D$ *splitting for the reduced system.*

$0 < \rho(B) \leq \Phi$,

$$\frac{\rho(B)}{2-\rho(B)} \leq \frac{\Phi}{2-\Phi}. \tag{4.8}$$

Since Φ has a Taylor expansion of the form $1 - ch^2 + O(h^2)$, by (3.6), it follows that $\frac{\Phi}{2-\Phi}$ and Φ^2 have the same Taylor expansion up to $O(h^2)$ terms, of the form $1 - 2ch^2 + O(h^3)$. Indeed, in terms of the PDE coefficients,

$$\frac{\Phi}{2-\Phi} = 1 - \left(\frac{20}{9}\pi^2 + \frac{1}{3}\sigma^2 + \frac{1}{3}\tau^2 + \frac{1}{3}\mu^2\right) h^2 + O(h^3), \tag{4.9}$$

and the same for Φ^2. It has been shown that the bound for $\rho(B)$ is extremely tight as $h \to 0$, and so we can replace the spectral radii by the bounds for the spectral radii in Theorem 4.3 to obtain the desired result. □

The actual meaning of this result is that for systems of equations that are large enough, the matrix nearly has Property A relative to 1D partitioning, at least as far as the convergence properties of the block Gauss–Seidel scheme are concerned. Since the solution process for small mesh Reynolds numbers is more efficient for the 1D splitting, compared to the 2D splitting, as we shall see in section 5, it was our aim to overcome the difficulty of not being able to apply Young's analysis directly.

For the block SOR scheme, the upper bound for the spectral radius is given in [23, Thm. 4.9] as $\sqrt{\omega^* - 1}$ and is not tight. However, it is numerically evident that the bound for the Jacobi iteration matrix can be effectively used to estimate the optimal SOR parameter. In Fig. 4.3 we can observe that the behavior for the 1D splitting is qualitatively identical to the behavior of two-cyclic consistently ordered matrices. Here we present results for centered difference discretization of the problem with $\beta = \gamma = \delta = 0.5$. The reduced matrix is 256×256. In the figure we also present the behavior of the SOR iteration matrix of the unreduced system.

5. Computational work and numerical experiments. Having done some analysis, in this section we examine which of the 1D and 2D solvers is more efficient overall and show that the reduced system is superior to the unreduced system.

5.1. Aspects of computational work. If $be,\ cd,\ fg > 0$, then by [23, Thm. 3.15] or by (3.6) and (3.7), it is evident that the spectral radius of the iteration matrix associated with the 2D splitting is smaller than that of the 1D iteration matrix. However, inverting D_1 involves less computational work than inverting D_2. We now compare these two solution procedures.

We begin with the block Jacobi scheme. Asymptotically, there is a fixed ratio of 1.8 between the rate of convergence of the two splittings (see (3.6) and (3.7)). In rough terms, this number characterizes the ratio between number of iterations until convergence for the two solvers.

As far as the computational work per iteration is concerned, if $D_1 = L_1U_1$ and $D_2 = L_2U_2$ are the LU decompositions of the matrices of the systems that are to be solved in each iteration, we can assume that the number of operations per iteration is approximately the number of nonzeros in $L_i + U_i$ plus the number of nonzeros in the other part of the splitting. In order to avoid costly fill-in using Gaussian elimination for D_2 (whose band is sparse), we use instead a technique of inner-outer iterations.

Let k_1 and k_2 denote the number of iterations for the schemes associated with the 1D splitting and the 2D splitting, respectively. Let us also define cost functions as follows: $c_1(n)$ and $c_2(n)$ represent the overall number of floating point operations for each of the solvers, and $c_{in}(n)$ represents the cost of the inner solve. Then

$$c_1(n) \approx [nz(L_1 + U_1) + nz(S - D_1)] \cdot k_1 = [10n^3 - 19n^2 + 4n] \cdot k_1, \tag{5.1a}$$

$$c_2(n) \approx [c_{in}(n) + nz(S - D_2)] \cdot k_2 = [c_{in}(n) + 3n^3 - 8n^2 + 4n] \cdot k_2. \tag{5.1b}$$

The term $nz(X)$ stands for the number of nonzeros of a matrix X, and S stands for the reduced matrix.

PROPOSITION 5.1. *For n large enough, the scheme associated with the 2D splitting is cheaper than the one associated with the 1D splitting only if $c_{in}(n) < 15n^3$.*

Proof. If n is large enough we can use the relation $\frac{k_1}{k_2} = 1.8$ and refer only to the leading power of n in the expressions for $c_1(n)$ and $c_2(n)$. So doing, it follows that

$$\frac{c_1(n)}{c_2(n)} \approx \frac{18n^3}{c_{in}(n) + 3n^3}, \tag{5.2}$$

and the result stated in the proposition readily follows. □

What is left now is to examine the amount of work involved in solving the inner system of equations. A natural choice of a splitting for this system is $D_2 = D_1 - (D_1 - D_2)$. It is straightforward to show the following by Propositions 3.2 and 3.3.

PROPOSITION 5.2. *If block Jacobi based on the splitting $D_2 = D_1 - (D_1 - D_2)$ is used, then the spectral radius of the inner iteration matrix, namely, $I - D_1^{-1}D_2$, is bounded by $\frac{\xi}{\eta}$, where η and ξ are defined in* (3.3) *and* (3.4).

For considering methods that are faster than block Jacobi for the inner system, we have the following useful result.

PROPOSITION 5.3. *The inner matrix is block consistently ordered relative to 1D partitioning.*

Proof. The inner matrix is block tridiagonal relative to this partitioning. □

We are now ready to prove the main result of this subsection.

PROPOSITION 5.4. *If $be,\ cd,\ fg > 0$, then if 1D splitting is used in solving the inner system, the cost of solving it is higher than $15n^3$ floating point operations, for block Jacobi as well as block Gauss–Seidel and block SOR, and thus, for n large enough and the methods considered in this paper, the 1D solver is faster than the 2D solver.*

Proof. The Taylor expansion of the bound in Proposition 5.2 is

$$\frac{\xi}{\eta} = \frac{4}{9} - \left(\frac{43}{81}\pi^2 + \frac{65}{648}\mu^2 + \frac{29}{648}\tau^2 + \frac{25}{324}\sigma^2\right) h^2 + o(h^2). \tag{5.3}$$

For h small enough, we can simply examine the leading term: The bound is approximately $\frac{4}{9}$ if block Jacobi is used, and since by Proposition 5.3 the matrix is consistently ordered, Young's analysis shows that the spectral radius is approximately $\frac{16}{81}$ if block Gauss–Seidel is used and approximately 0.055 if block SOR with the optimal relaxation parameter is used. For both of these schemes each iteration costs about $7n^3$ floating point operations. Since reducing the inital error by a factor of 10^m takes roughly $-m/\log_{10}\rho$ iterations, where ρ is the spectral radius of the associated iteration matrix, it follows that even for the block SOR scheme with the optimal relaxation parameter, which is the fastest scheme considered here, after two iterations the error is reduced only by a factor of approximately $10^{2.5}$, which is obviously far from satisfactory. Thus the iteration count is larger than 2, and the cost of inner solve is larger than $15n^3$ floating point operations. □

We remark that an inexact inner solve can also be considered (see, for example, Elman and Golub's paper on inexact Uzawa algorithms [11]), but this is beyond the scope of this work.

It is our conclusion that the solver associated with 1D splitting is more efficient than the one associated with the 2D splitting if upwind differences are used or if centered differences with mesh Reynolds numbers smaller than 1 in magnitude are used.

5.2. Comparison with the unreduced system. One step of cyclic reduction results in a more complicated difference operator compared to the original, unreduced system, and a grid which is more difficult to handle as far as ordering of the unknowns is concerned. Moreover, the unreduced matrix is block consistently ordered relative to *both* 1D and 2D splittings (we refer to the straightforward one-line and one-plane partitionings as the basis for 1D and 2D splittings in case of the unreduced system) and thus Young's analysis can be easily applied. One could ask, therefore, what the advantages of using cyclic reduction are. In this subsection we illustrate the superiority of the reduced system over the unreduced system.

We start with the block Jacobi scheme. For the unreduced system we shall refer to natural lexicographic ordering of the unknowns, so that the lines of gridpoints are x-oriented and the planes are x-y oriented. We start with quoting the following result, given in [14, Sects. 2 and 4].

Lemma 5.5. *The spectral radius of the block Jacobi scheme associated with the* 1*D splitting for the unreduced system is*

$$\frac{2\sqrt{be}\cdot\cos(\pi h) + 2\sqrt{fg}\cdot\cos(\pi h)}{a + 2\sqrt{cd}\cdot\cos(\pi n h)}\ . \tag{5.4}$$

The Taylor expansion of (5.4) *about* h = 0 *is given by*

$$1 - \left(\frac{3}{4}\pi^2 + \frac{1}{16}\sigma^2 + \frac{1}{16}\tau^2 + \frac{1}{16}\mu^2\right)\ h^2 + o(h^2). \tag{5.5}$$

In [14] we have shown that the spectrum of the iteration matrix of the unreduced system can be found by a sequence of diagonalizations and permutations that form a similarity transformation of the matrix into a matrix whose associated iteration matrix is easy to analyze, as far as its spectrum is concerned. The reader is referred to the proof of [14, Thm. 2.1] for full details. For the 2D splitting a similar procedure can be applied. The technique we have used is similar to the one presented in [14] and the algebraic details are omitted.

LEMMA 5.6. *The spectral radius of the block Jacobi iteration matrix associated with* 2*D splitting is given by*

$$\frac{2\sqrt{fg}\cdot\cos(\pi h)}{a+2\sqrt{cd}\cdot\cos(\pi nh)+2\sqrt{be}\cdot\cos(\pi nh)}, \tag{5.6}$$

and its Taylor expansion about $h=0$ *is*

$$1-\left(\frac{3}{2}\pi^2+\frac{1}{8}\sigma^2+\frac{1}{8}\tau^2+\frac{1}{8}\mu^2\right)\ h^2+o(h^2). \tag{5.7}$$

The same type of analysis that has been done in the previous section, comparing the 1D splitting to the 2D splitting for the reduced system, is possible for the unreduced system. Below we sketch the main details: Suppose inner-outer iterations are used in solving the scheme associated with the 2D splitting. Denote, again, this splitting for the inner system as $D_2 = D_1 - (D_1 - D_2)$ (D_1 and D_2 are now different than the ones defined in section 3). Then we have the following proposition.

PROPOSITION 5.7. *Consider the unreduced system. Suppose be, cd, fg* > 0, *n is sufficiently large, and* 1*D splitting is used in solving the inner system. Then for the stationary methods considered in this paper, the* 1*D solver is faster than the* 2*D solver.*

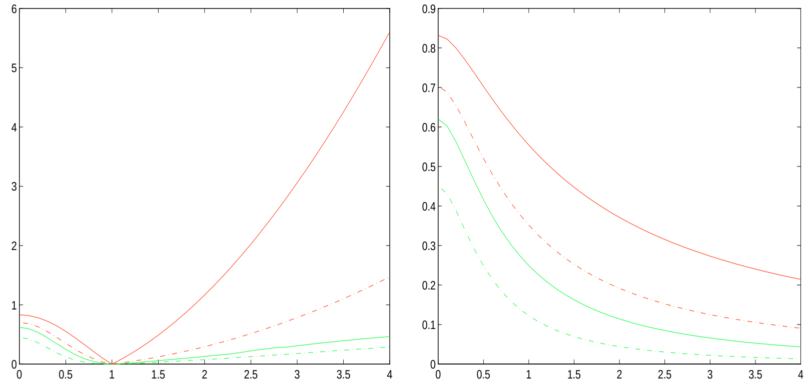
Proof. The ratio between the asymptotic rate of convergence between the 1D solver and the 2D solver is 2. The number of nonzeros of the whole matrix is approximately $7n^3$, the number of nonzeros of D_1 is approximately $3n^3$, and the number of nonzeros of D_2 is approximately $5n^3$. Since the spectral radii for the two splittings are available, we can find the spectral radius for the iteration matrix of the inner system. Its Taylor expansion is given by $\frac{1}{2}-(\frac{3}{8}\pi^2+\frac{1}{16}\sigma^2+\frac{1}{32}\tau^2)h^2+o(h^2)$. Defining cost functions analogous to the ones defined in section 3 for the reduced system, and using the same line of argument, we have

$$\frac{c_1(n)}{c_2(n)} \approx \frac{14n^3}{c_{in}(n)+2n^3}, \tag{5.8}$$

and from this it follows that only if $c_{in}(n) < 12n^3$ the 2D solver is more efficient. However, as in Proposition 5.4, this means at most two iterations of the inner solve can be performed, which is not enough for the required accuracy. □

Since the 1D splitting for both the reduced and the unreduced systems gives rise to a more efficient solve, we compare these two systems, focusing on this splitting. See also [14, Sect. 4]. The LU decomposition for the solution of the system in each iteration is done once and for all (see [12] for operation count) and its cost is negligible in comparison with the amount of work done in the iterative process.

Each iteration in the reduced system costs about $10n^3$ floating point operations, whereas each iterate for the unreduced system costs approximately $7n^3$ floating point operations per iteration. Hence, the amount of computational work per iteration is cheaper for the unreduced system by a factor of about 10/7. However, using the asymptotic formulas (3.6) and (5.5), it is evident that the number of iterations required for the unreduced system is larger than that required for the reduced system, and in



(**a**) ρ_{gs} *− centered* (**b**) ρ_{gs} *− upwind*

FIG. 5.1. *Comparison between the spectral radii of the Gauss-Seidel iteration matrices of the reduced and unreduced systems. The uppermost curve corresponds to* 1*D splitting for the unreduced system, and then we have, in order,* 2*D splitting for the unreduced system,* 1*D splitting for the reduced system, and* 2*D splitting for the reduced system.*

the worst case, the ratio between the work required for solving the reduced system versus the unreduced system is roughly $(10/7) \cdot (27/40)$, which is $27/28$ and is still smaller than 1, thus the reduced solver is more efficient. If the convective terms are nonzero, then this ratio becomes smaller, and in practice we have observed substantial savings, as is illustrated in the test problem discussed in section 5.3.

Moving from comparing the block Jacobi scheme for both the reduced and the unreduced systems to comparing Gauss–Seidel and SOR is straightforward if Young's analysis can be used. In section 4 we showed that even though the reduced matrix is not consistently ordered relative to 1D partitioning, it is *nearly* consistently ordered. In general, convergence analysis for the Jacobi scheme does not always indicate the behavior of the Gauss–Seidel and SOR schemes. Nevertheless, for two-cyclic consistently ordered matrices (or matrices that are nearly so) the strong connections between the spectra of the Jacobi iteration matrix and the Gauss–Seidel and SOR iteration matrices [24] allow us to conclude that once the superiority of the reduced system over the unreduced system has been shown for Jacobi, this superiority is carried over to the other stationary schemes. Indeed, our numerical experiments verify this observation, as is illustrated in section 5.3.

In Fig. 5.1 the superiority of the reduced system over the unreduced system for the Gauss–Seidel scheme is illustrated numerically. The graphs were created for a small 512-point grid. It is interesting to notice that the reduced 1D Gauss–Seidel iteration matrix is well behaved (i.e., its spectral radius is significantly smaller than 1), even for the convection-dominated case, when centered differences are used. Convergence does not occur when the block Jacobi scheme with the same values of mesh Reynolds numbers is used. We have no bounds on convergence rates for this range of mesh Reynolds numbers and thus cannot explain this phenomenon analytically.

The superiority of the reduced system is evident also for the SOR scheme (see Fig. 4.3). Notice that for the SOR scheme it is difficult to determine the optimal relaxation parameter when be, cd, and fg are negative.

We end this subsection with a remark regarding the case of convection-dominated equations. Our convergence analysis does not cover the case of mesh Reynolds num-

Table 5.1

Comparison between iteration counts for the reduced and unreduced system, for different values of mesh Reynolds numbers. N/C marks no convergence after 2,000 *iterations. GS = Gauss–Seidel.*

System		Reduced				Unreduced			
PDE coeff. ($\sigma = \tau = \mu$)		10	20	100	1000	10	20	100	1000
Jacobi	centered	393	173	53	N/C	1030	444	N/C	N/C
GS	centered	188	77	14	322	492	198	N/C	N/C
SOR	centered	36	25	-	-	61	38	-	-
Jacobi	upwind	455	239	75	43	1194	620	179	89
GS	upwind	219	111	27	10	574	287	63	16
SOR	upwind	39	27	18	9	66	45	24	11

bers that are greater than 1 in magnitude in conjunction with centered difference discretization. Since the numerical solution might be oscillatory when a centered difference scheme is used [18], analysis for this case is of less interest. Nevertheless, Fourier analysis based on Chan and Elman's technique [5], which shows that when one of the mesh Reynolds numbers tends to ∞ the scheme still converges, is presented in [13].

5.3. Test problem. Consider (1.1), where the right-hand side is such that the solution for the continuous problem is $u(x, y, z) = \sin(\pi x) \cdot \sin(\pi y) \cdot \sin(\pi z)$ and the domain is the unit cube. The Dirichlet boundary conditions in this case are zero. The performance of the solvers for this specific problem well represents the performance for other test problems that we have examined.

We have taken the zero vector as our initial guess and have used $||r_i||_2/||r_0||_2 < 10^{-10}$ as a stopping criterion (here r_i denotes the residual at the ith iterate). The program stopped if the stopping criterion was not satisfied after 2,000 iterations. Our numerical experiments were executed on an SGI Origin 2000, which has four parallel 195 MHZ processors, 512 MB RAM, and 4MB cache. The program was written in MATLAB 5.

In the experiments that are presented, the 1D solver is used. In Table 5.1, the grid is of size $32 \times 32 \times 32$. The matrix of the underlying system of equations is of size $32,768 \times 32,768$. In the table, iteration counts for the Jacobi scheme and the Gauss–Seidel scheme are presented for four values of the PDE coefficients and for two discretization schemes.

The PDE coefficients referred to in Table 5.1 are specified in (1.1). For the values of these coefficients in the table, namely 10, 20, 100, and 1,000, the corresponding values of the mesh Reynolds numbers are 0.1515, 0.3030, 1.515, and 15.15. Notice that the last two are larger than 1, and so for these values we have no analytical way of knowing the optimal relaxation parameter and the experiments for these values were not performed.

The following observations can be made.

1. Overall the reduced solver is substantially faster than the unreduced solver. There are cases where the reduced solver converges whereas the unreduced solver does not. We remark that in all cases that were examined, the CPU time for the reduced solver was less (much less in most cases) than the CPU time for the unreduced system.
2. For $\sigma = 50$ convergence is faster than for $\sigma = 10$. This illustrates a phenomenon, which is supported by the analysis and holds also for the two-dimensional case [9], that for small-enough mesh Reynolds numbers, the

"more nonsymmetric" systems converge faster than the "close to symmetric" ones (close in the sense of PDE coefficients close to zero).

3. The upwind difference scheme converges more slowly than the centered difference scheme when the mesh Reynolds numbers are small in magnitude, but convergence is extremely fast for large mesh Reynolds numbers. This applies to both the reduced and the unreduced systems and follows from the fact that as the PDE coefficients grow larger, the underlying matrix is more diagonally dominant when upwind schemes are used.

6. Concluding remarks. We have presented ordering strategies for a cyclically reduced matrix arising from discretizing a 3D model problem with constant coefficients. We have derived bounds on convergence rates for block stationary schemes associated with what we called 1D splitting or 2D splitting. We have compared the amount of work involved in solving the system with the suggested splittings. In general, the 1D splitting gives rise to more-efficient solvers. Since the matrices associated with this splitting are not consistently ordered, we have analyzed their departure from block Property A and have shown that, in fact, these matrices are nearly block consistently ordered. We have shown, both analytically and numerically, that one step of cyclic reduction results in a system which is easier to solve, compared to the original, unreduced system.

Acknowledgments. We would like to thank the referees for their helpful comments, which substantially improved this manuscript.

REFERENCES

[1] O. Axelsson and I. Gustafsson, *On the use of preconditioned conjugate gradient methods for red-black order five-point difference schemes*, J. Comput. Phys., 35 (1980), pp. 284–289.

[2] O. Buneman, *A compact non-iterative Poisson solver*, Report 294, Stanford University Institute for Plasma Research, Stanford, CA, 1969.

[3] B. L. Buzbee, F. W. Dorr, J. A. George, and G. H. Golub, *The direct solution of the discrete Poisson equation on irregular regions*, SIAM J. Numer. Anal., 8 (1971), pp. 722–736.

[4] B. L. Buzbee, G. H. Golub, and C. W. Nielson, *On direct methods for solving Poisson's equations*, SIAM J. Numer. Anal., 7 (1970), pp. 627–656.

[5] T. F. Chan and H. C. Elman, *Fourier analysis of iterative methods for elliptic problems*, SIAM Rev., 31 (1989), pp. 20–49.

[6] P. Concus and G. H. Golub, *Use of fast direct methods for the efficient numerical solution of nonseparable elliptic equations*, SIAM J. Numer. Anal., 10 (1973), pp. 1103–1120.

[7] E. Detyna, *Point cyclic reductions for elliptic boundary-value problems* I: *The constant coefficient case*, J. Comput. Phys., 33 (1979), pp. 204–216.

[8] H. C. Elman and G. H. Golub, *Iterative methods for cyclically reduced non-self-adjoint linear systems*, Math. Comp., 54 (1990), pp. 671–700.

[9] H. C. Elman and G. H. Golub, *Iterative methods for cyclically reduced non-self-adjoint linear systems* II, Math. Comp., 56 (1991), pp. 215–242.

[10] H. C. Elman and G. H. Golub, *Line iterative methods for cyclically reduced discrete convection-diffusion problems*, SIAM J. Sci. Stat. Comput., 13 (1992), pp. 339–363.

[11] H. C. Elman and G. H. Golub, *Inexact and preconditioned Uzawa algorithms for saddle point problems*, SIAM J. Numer. Anal., 31 (1994), pp. 1645–1661.

[12] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed., Johns Hopkins University Press, Baltimore, MD, 1996.

[13] C. Greif, *Analysis of Cyclic Reduction for the Numerical Solution of Three-Dimensional Convection-Diffusion Equations*, Ph.D. thesis, University of British Columbia, Vancouver, BC, 1998.

[14] C. Greif and J. M. Varah, *Iterative solution of cyclically reduced systems arising from discretization of the three-dimensional convection diffusion equation*, SIAM J. Sci. Comput., (1998), pp. 1918–1940.

[15] L. A. Hageman, F. T. Luk, and D. M. Young, *On the equivalence of certain iterative acceleration methods*, SIAM J. Numer. Anal., 17 (1980), pp. 852–873.

[16] L. A. Hageman and R. S. Varga, *Block iterative methods for cyclically reduced matrix equations*, Numer. Math., 6 (1964), pp. 106–119.

[17] R. W. Hockney, *A fast direct solution of Poisson's equation using Fourier analysis*, J. Assoc. Comput. Mach., (1965), pp. 95–113.

[18] K. W. Morton, *Numerical Solution of Convection-Diffusion Problems*, 1st ed., Chapman and Hall, London, 1996.

[19] Y. Saad, *Iterative Methods for Sparse Linear Systems*, PWS Publishing Company, Boston, 1996.

[20] R. A. Sweet, *A generalized cyclic reduction algorithm*, SIAM J. Numer. Anal., 11 (1974), pp. 506–520.

[21] R. A. Sweet, *A cyclic reduction algorithm for solving block tridiagonal systems of arbitrary dimension*, SIAM J. Numer. Anal., 14 (1977), pp. 706–720.

[22] R. S. Varga, *p-Cyclic matrices: A generalization of the Young-Frankel successive overrelaxation scheme*, Pacific J. Math., 9 (1959), pp. 617–628.

[23] R. S. Varga, *Matrix Iterative Analysis*, Prentice–Hall, Englewood Cliffs, NJ, 1962.

[24] D. M. Young, *Iterative Solution of Large Linear Systems*, Academic Press, New York, 1971.

SIAM J. Matrix Anal. Appl.
Vol. 20, No. 4, pp. 1060–1082

ABLE: AN ADAPTIVE BLOCK LANCZOS METHOD FOR NON-HERMITIAN EIGENVALUE PROBLEMS[*]

ZHAOJUN BAI[†], DAVID DAY[‡], AND QIANG YE[§]

Abstract. This work presents an adaptive block Lanczos method for large-scale non-Hermitian Eigenvalue problems (henceforth the ABLE method). The ABLE method is a block version of the non-Hermitian Lanczos algorithm. There are three innovations. First, an adaptive blocksize scheme cures (near) breakdown and adapts the blocksize to the order of multiple or clustered eigenvalues. Second, stopping criteria are developed that exploit the semiquadratic convergence property of the method. Third, a well-known technique from the Hermitian Lanczos algorithm is generalized to monitor the loss of biorthogonality and maintain semibiorthogonality among the computed Lanczos vectors. Each innovation is theoretically justified. Academic model problems and real application problems are solved to demonstrate the numerical behaviors of the method.

Key words. non-Hermitian matrices, eigenvalue problem, spectral transformation, Lanczos method

AMS subject classifications. 65F15, 65F10

PII. S0895479897317806

1. Introduction.

A number of efficient numerical algorithms for solving large-scale matrix computation problems are built upon the Lanczos procedure, a procedure for successive reduction of a general matrix to a tridiagonal form [28]. In the 1970s and '80s, great progress in mathematical and numerical analysis was made on applying the Lanczos algorithm for solving large sparse Hermitian eigenvalue problems. Today, a Lanczos-based algorithm has been accepted as the method of choice to large sparse Hermitian eigenvalue problems arising in many computational science and engineering areas.

Over the last decade there has been considerable interest in Lanczos-based algorithms for solving *non*-Hermitian eigenvalue problems. The Lanczos algorithm without rebiorthogonalization is implemented and applied to a number of application problems in [12]. Different schemes to overcome possible failure in the non-Hermitian Lanczos algorithm are studied in [38, 17, 53]. A Lanczos procedure with look-ahead scheme is available in QMRPACK [18]. Theoretical studies of breakdown and instability can be found in [21, 36, 23, 6]. Error analyses of the non-Hermitian Lanczos procedure implemented in finite precision arithmetic are presented in [2, 14].

Despite all this progress, a number of unresolved issues, some of which are related to the use of nonorthogonal basis and hence its conditional stability property, obstruct a robust and efficient implementation of the non-Hermitian Lanczos procedure. These issues include

- how to distinguish *copies* of converged Rayleigh–Ritz values from multiple or clustered eigenvalues,

[*] Received by the editors March 4, 1997; accepted for publication (in revised form) by Z. Strakos February 27, 1998; published electronically July 9, 1999. This work was supported in part by NSF grant ASC-9313958, DOE grant DE-FG03-94ER25219, and a research grant from NSERC of Canada.
http://www.siam.org/journals/simax/20-4/31780.html

[†] Department of Mathematics, University of Kentucky, Lexington, KY 40506 (bai@ms.uky.edu).

[‡] MS 1110, Sandia National Laboratories, PO Box 5800, Albuquerque, NM 87185 (dday@cs.sandia.gov).

[§] Department of Applied Mathematics, University of Manitoba, Winnipeg, MB R3T 2N2, Canada (ye@gauss.amath.umanitoba.ca).

- how to treat a (near) breakdown to preserve the stringent numerical stability requirements on the Lanczos procedure for eigenvalue problems in finite precision arithmetic,
- how to explain and take advantage of the observed semiquadratic convergence rate of the Lanczos procedure, and
- how to extend the understanding of the Hermitian Lanczos algorithm with semiorthogonality [46] and the non-Hermitian Lanczos algorithm with semibiorthogonality [14] to the *block* non-Hermitian Lanczos algorithm.

In the adaptive block Lanczos method for large-scale non-Hermitian Eigenvalue problems (ABLE method) proposed in this work, we address each of these issues as follows:

- A block version of the Lanczos procedure is implemented. Several nontrivial implementation issues are addressed. The blocksize adapts to be at least the order of multiple or clustered eigenvalues, and the linear independence of the Lanczos vectors is maintained. This accelerates convergence in the presence of multiple or clustered eigenvalues.
- The blocksize also adapts to cure (near) breakdowns. The adaptive blocking scheme proposed here enjoys the theoretical advantage that any exact breakdown can be cured with fixed augmentation vectors. In contrast, the prevalent look-ahead techniques require an arbitrary number of augmentation vectors to cure a breakdown and may not be able to cure all breakdowns [38, 17, 36].
- An asymptotic analysis of the second-order convergence of the Lanczos procedure is presented and utilized in the stopping criteria.
- A scheme to monitor the loss of the biorthogonality and maintain semibiorthogonality is developed in the adaptive block Lanczos procedure.

The ABLE method is a generalization of the block Hermitian Lanczos algorithm [10, 19, 39, 22] to the non-Hermitian case. For general application codes that represent their matrices as out-of-core, block algorithms substitute matrix block multiplies and block solvers for matrix vector products and simple solvers [22]. In other words, higher level BLAS are used in the inner loop of block algorithms. This decreases the I/O costs essentially by a factor of the blocksize.

We will demonstrate numerical behaviors of the ABLE method using several numerical examples from academic model problems and real application problems. There are many competitive methods for computing large sparse non-Hermitian eigenvalue problems, namely, the simultaneous iteration method [5, 50, 15], Arnoldi's method [1, 42], the implicitly restarted Arnoldi method [48, 29], block Arnoldi [43, 45], the rational Krylov subspace method [40], Davidson's method [13, 44], and the Jacobi–Davidson method [47, 8]. In particular, ARPACK [31], an implementation of the implicitly restarted Arnoldi method, is gaining acceptance as a standard piece of mathematical software for solving large-scale eigenvalue problems. A comparative study of simultaneous iteration-based methods and Arnoldi-based methods is presented in [30]. It is beyond the scope of this paper to compare our ABLE method with the rest of the methods. However, a comprehensive comparison study is certainly a part of our future work.

The rest of this paper is organized as follows. In section 2, we present a basic block non-Hermitian Lanczos algorithm, discuss its convergence properties, and review how to maintain biorthogonality among Lanczos vectors computed in finite precision arithmetic. An adaptive block scheme to cure (near) breakdown and adapt the blocksize to the order of multiple or clustered eigenvalues is described in section 3. In section 4, we

```
1. Choose starting $n \times p$ vectors $P_1$ and $Q_1$ so that $P_1^T Q_1 = I$
2. $R_1 = (P_1^T A)^T$ and $S_1 = AQ_1$
3. For $j = 1, 2, \ldots\ldots$
   3.1. $A_j = P_j^T S_j$
   3.2. $R_j := R_j - P_j A_j^T$ and $S_j := S_j - Q_j A_j$
   3.3. Compute the QR decompositions: $R_j = P_{j+1} B_{j+1}^T$ and $S_j = Q_{j+1} C_{j+1}$
   3.4. Compute the singular value decomposition: $P_{j+1}^T Q_{j+1} = U_j \Sigma_j V_j^H$
   3.5. $B_{j+1} := B_{j+1} U_j \Sigma_j^{1/2}$ and $C_{j+1} := \Sigma_j^{1/2} V_j^H C_{j+1}$
   3.6. $P_{j+1} := P_{j+1} \bar{U}_j \Sigma_j^{-1/2}$ and $Q_{j+1} := Q_{j+1} V_j \Sigma_j^{-1/2}$
   3.7. $R_{j+1} = (P_{j+1}^T A - C_{j+1} P_j^T)^T$ and $S_{j+1} = AQ_{j+1} - Q_j B_{j+1}$
```

FIG. 2.1. *Basic block non-Hermitian Lanczos algorithm.*

model the loss of biorthogonality among the Lanczos vectors in finite precision arithmetic and present an efficient algorithm for maintaining semibiorthogonality among the computed Lanczos vectors. The complete ABLE method is presented in section 5. In section 6, we briefly discuss how a spectral transformation is used to solve a generalized eigenvalue problem using the ABLE method. Numerical experiments are reported in section 7.

2. A block non-Hermitian Lanczos algorithm.

In this section we present a block implementation of the non-Hermitian Lanczos algorithm and discuss its convergence properties for solving non-Hermitian eigenvalue problems. An adaptive block non-Hermitian Lanczos algorithm (see section 5) builds into this algorithm features presented in the intervening sections.

2.1. A basic block Lanczos algorithm.

The basic block non-Hermitian Lanczos procedure presented in Figure 2.1 is a variation of the original Lanczos procedure as proposed by Lanczos [28]. Given an n by n matrix A and initial n by p block vectors P_1 and Q_1, two sequences of n by p block vectors $\{P_j\}$ and $\{Q_j\}$, called Lanczos vectors, are generated such that for $j = 1, 2, \ldots,$

$$\begin{aligned}
\operatorname{span}\{\, P_1^T,\, P_2^T,\, \ldots,\, P_j^T \,\} &= \boldsymbol{\mathcal{K}}_j(\, P_1^T,\, A\,) := \operatorname{span}\{\, P_1^T,\, P_1^T A,\, P_1^T A^2,\, \ldots,\, P_1^T A^{j-1} \,\},\\
\operatorname{span}\{\, Q_1,\, Q_2,\, \ldots,\, Q_j \,\} &= \boldsymbol{\mathcal{K}}_j(\, Q_1,\, A\,) := \operatorname{span}\{\, Q_1,\, AQ_1,\, A^2 Q_1,\, \ldots,\, A^{j-1} Q_1 \,\},
\end{aligned}$$

where $\boldsymbol{\mathcal{K}}_j(Q_1, A)$ and $\boldsymbol{\mathcal{K}}_j(P_1^T, A)$ are right and left Krylov subspaces. The block vectors $\{P_j\}$ and $\{Q_j\}$ are constructed so that they are biorthonormal. Together these properties determine the computed quantities up to a scaling. Several nontrivial practical issues are resolved in the implementation presented in Figure 2.1.

The basic block Lanczos iteration implements the three-term recurrences

$$B_{j+1} P_{j+1}^T = P_j^T A - A_j P_j^T - C_j P_{j-1}^T, \tag{2.1}$$

$$Q_{j+1} C_{j+1} = AQ_j - Q_j A_j - Q_{j-1} B_j. \tag{2.2}$$

The procedure can be also viewed as a successive reduction of an $n \times n$ non-Hermitian matrix A to a block tridiagonal form. If we let

$$\boldsymbol{\mathcal{P}}_j = [\, P_1, P_2, \ldots, P_j \,], \qquad \boldsymbol{\mathcal{Q}}_j = [\, Q_1, Q_2, \ldots, Q_j \,],$$

and

$$T_j = \begin{bmatrix} A_1 & B_2 & & \\ C_2 & A_2 & \ddots & \\ & \ddots & \ddots & B_j \\ & & C_j & A_j \end{bmatrix}, \tag{2.3}$$

then the three-term recurrences (2.1) and (2.2) have the matrix form

$$\boldsymbol{\mathcal{P}}_j^T A = T_j \boldsymbol{\mathcal{P}}_j^T + E_j B_{j+1} P_{j+1}^T, \tag{2.4}$$

$$A \boldsymbol{\mathcal{Q}}_j = \boldsymbol{\mathcal{Q}}_j T_j + Q_{j+1} C_{j+1} E_j^T, \tag{2.5}$$

where E_j is a tall thin matrix whose bottom square is an identity matrix and which vanishes otherwise. Furthermore, the computed Lanczos vectors P_j and Q_j satisfy the *biorthonormality*

$$\boldsymbol{\mathcal{P}}_j^T \boldsymbol{\mathcal{Q}}_j = I. \tag{2.6}$$

When the blocksize $p = 1$, this is just the unblocked non-Hermitian Lanczos algorithm discussed in [20, p. 503].

Remark 1. For a complex matrix A we still use the transpose $\cdot^T$ instead of the conjugate transpose $\cdot^H$. If A is complex symmetric and $P_1 = Q_1$, then (2.4) is the transpose of (2.5), and it is necessary to compute only one of these two recurrences provided that a complex symmetric scaling scheme is used at step 3.4 in Figure 2.1.

Remark 2. The above block Lanczos algorithm can breakdown prematurely if $R_j^T S_j$ is singular (see step 3.6 in Figure 2.1). We will discuss this issue in section 3.

Remark 3. Many choices of the $p \times p$ nonsingular scaling matrices B_{j+1} and C_{j+1} satisfy $R_j^T S_j = B_{j+1} C_{j+1}$. The one presented here involves a little more work (computing singular value decomposition (SVD) of $P_{j+1}^T Q_{j+1}$), but it maintains that the local basis vectors in P_{j+1} and Q_{j+1} are orthogonal and at the same time biorthogonal to each other. Furthermore, the singular values provide principal angles between the subspaces spanned by P_{j+1} and Q_{j+1}, which is a measure of the quality of the bases constructed (see Remark 4 below).

An alternative scaling maintains the unit length of all Lanczos vectors. This scaling scheme for the unblocked Lanczos algorithm is used in [17, 36, 14]. In this case the Lanczos algorithm determines a pencil $(\widehat{T}_j, \Omega_j)$, where $\widehat{T}_j$ is tridiagonal and Ω_j is diagonal. It can be shown that the tridiagonal T_j determined by the above unblocked ($p = 1$) Lanczos algorithm and this pencil are related by

$$T_j = \pm |\Omega_j|^{1/2} \widehat{T}_j |\Omega_j|^{1/2}.$$

The Lanczos vectors are also related, up to sign, by a similar scaling.

Remark 4. The condition numbers of the Lanczos vectors $\boldsymbol{\mathcal{P}}_j$ and $\boldsymbol{\mathcal{Q}}_j$ can be monitored by the diagonal matrices $\Sigma_1, \Sigma_2, \ldots, \Sigma_j$. Recall that the condition number of the rectangular matrix $\boldsymbol{\mathcal{Q}}_j$ is defined by

$$\operatorname{cond}(\boldsymbol{\mathcal{Q}}_j) \overset{\text{def}}{=} \|\boldsymbol{\mathcal{Q}}_j\|_2 \|\boldsymbol{\mathcal{Q}}_j^\dagger\|_2 = \frac{\|\boldsymbol{\mathcal{Q}}_j\|_2}{\sigma_{\min}(\boldsymbol{\mathcal{Q}}_j)},$$

where $\sigma_{\min}(\boldsymbol{\mathcal{Q}}_j) = \min_{\|\mathbf{x}\|_2=1} \|\boldsymbol{\mathcal{Q}}_j \mathbf{x}\|_2$. To derive the bound for $\operatorname{cond}(\boldsymbol{\mathcal{Q}}_j)$ observe from step 3.6 in Figure 2.1 that $\|P_i^T\|_2 = \|Q_i\|_2 = \|\Sigma_i^{-1/2}\|_2$. Then for a unit vector v such that $\|\boldsymbol{\mathcal{P}}_j^T\|_2 = \|\boldsymbol{\mathcal{P}}_j^T v\|_2$,

$$\|\boldsymbol{\mathcal{P}}_j^T\|_2^2 \;=\; \|\boldsymbol{\mathcal{P}}_j^T v\|_2^2 \;=\; \sum_{i=1}^{j} \|P_i^T v\|_2^2 \;\le\; \sum_{i=1}^{j} \frac{1}{\min(\Sigma_i)},$$

where $\min(\Sigma_i)$ denotes the smallest diagonal element of Σ_i. The latter bound also applies to $\boldsymbol{\mathcal{Q}}_j$. Furthermore, we note that the biorthonormality condition (2.6) implies that $\|\boldsymbol{\mathcal{P}}_j\|_2 \; \sigma_{\min}(\boldsymbol{\mathcal{Q}}_j) \ge 1$. Therefore,

$$\operatorname{cond}(\boldsymbol{\mathcal{Q}}_j) \;\le\; \|\boldsymbol{\mathcal{P}}_j\|_2 \|\boldsymbol{\mathcal{Q}}_j\|_2 \;\le\; \sum_{i=1}^{j} \frac{1}{\min(\Sigma_i)}.$$

The bound applies to $\operatorname{cond}(\boldsymbol{\mathcal{P}}_j)$ by the similar argument. This generalizes and slightly improves a result from [36].

Remark 5. This implementation is a generalization of the block Hermitian Lanczos algorithms of Golub and Underwood [19] and Grimes, Lewis, and Simon [22] to the non-Hermitian case. A simple version of the block non-Hermitian Lanczos procedure has been studied in [3]. Other implementations of the basic block non-Hermitian Lanczos procedure have been proposed for different applications in [7].

2.2. Eigenvalue approximation. To extract approximate the eigenvalues and eigenvectors of A, we solve the eigenvalue problem of the $jp \times jp$ block tridiagonal matrix T_j after step 3.3 in Figure 2.1. Each eigentriplet (θ, w^H, z) of T_j,

$$w^H T_j = \theta w^H \quad \text{and} \quad T_j z = z\theta,$$

determines a *Rayleigh–Ritz triplet*, (θ, y^H, x), where $y^H = w^H \boldsymbol{\mathcal{P}}_j^T$ and $x = \boldsymbol{\mathcal{Q}}_j z$. Rayleigh–Ritz triplets approximate eigentriplets of A.

To assess the approximation, (θ, y^H, x), of an eigentriplet of the matrix A, let s and r denote the corresponding left and right residual vectors. Then by (2.4) and (2.5), we have

$$s^H = y^H A - \theta y^H = (w^H E_j) B_{j+1} P_{j+1}^T, \tag{2.7}$$

$$r = Ax - x\theta = Q_{j+1} C_{j+1} (E_j^T z). \tag{2.8}$$

Note that a remarkable feature of the Lanczos algorithm is that the residual norms $\|s^H\|_2$ and $\|r\|_2$ are available without explicitly computing y^H and x. There is no need to form y^H and x until their accuracy is satisfactory.

The residuals determine a backward error bound for the triplet. The biorthogonality condition, (2.6), applied to the definition of x and y^H yields

$$P_{j+1}^T x = 0 \qquad \text{and} \qquad y^H Q_{j+1} = 0. \tag{2.9}$$

From (2.8) and (2.7), we have the following measure of the backward error for the Rayleigh–Ritz triplet (θ, y^H, x):

$$y^H (A - F) = \theta y^H \qquad \text{and} \qquad (A - F)x = x\theta,$$

where the backward error matrix F is

$$F = \frac{rx^H}{\|x\|_2^2} + \frac{ys^H}{\|y\|_2^2} \tag{2.10}$$

and $\|F\|_F^2 = \|r\|_2^2/\|x\|_2^2 + \|s^H\|_2^2/\|y\|_2^2$. That is, the left and right residual norms bound the distance to the nearest matrix to A with eigentriplet (θ, y^H, x). In fact it has been shown that F is the smallest perturbation of A such that (θ, y^H, x) is an eigentriplet of $A - F$ [25]. The computed Rayleigh–Ritz value θ is a $\|F\|_F$-pseudo eigenvalue of the matrix A [51].

If we write $A = B + F$, where $B = A - F$, then a first-order perturbation analysis indicates that there is an eigenvalue λ of A such that

$$|\lambda - \theta| \leq \mathrm{cond}(\theta)\|F\|_2,$$

where $\mathrm{cond}(\theta) = \|y^H\|_2 \ \|x\|_2/|y^H x|$ is the condition number of the Rayleigh–Ritz value θ [20]. This first-order estimate is very often pessimistic because θ is a *two-sided* or generalized Rayleigh quotient [34]. A second-order perturbation analysis yields a more realistic error estimate, which should be used as a stopping criterion. Global second-order bounds for the accuracy of the generalized Rayleigh quotient may be found in [49] and [9]. Here we derive an asymptotic bound.

Recall that (θ, y^H, x) is an eigentriplet of $B = A - F$ and that $y^H F = s^H$ and $Fx = r$. Assume that B has distinct eigenvalues $\{\theta_i\}$ and the corresponding normalized left and right eigenvectors $\{y_i^H, x_i\}$ $(\|y_i^H\|_2 = \|x_i\|_2 = 1)$. Let us perturb $\theta = \theta(0)$ toward an eigenvalue λ of A using the implicit function $\theta(t) = \theta(B + tE)$ for $E = F/\|F\|_2$. Under classical results from function theory [26], it can be shown that in a neighborhood of the origin there exist differentiable $\theta(t)$, $y^H(t)$, and $x(t)$ $(\|y^H(t)\|_2 = \|x(t)\|_2 = 1)$ such that

$$y^H(t)(B + tE) = \theta(t)y^H(t) \quad \text{and} \quad (B + tE)x(t) = x(t)\theta(t). \tag{2.11}$$

Next expand $\theta(t)$ about $t = 0$:

$$\lambda = \theta(\|F\|_2) = \theta(0) + \theta'(0)\|F\|_2 + \frac{1}{2}\theta''(0)\|F\|_2^2 + O(\|F\|_2^3).$$

By differentiating (2.11) with respect to t, and setting $t = 0$, we obtain

$$\theta'(0) = \frac{1}{\|F\|_2}\frac{y^H F x}{y^H x}.$$

Note that from (2.10), $y^H F x = y^H r + s^H x$. Substitute (2.7), (2.8), and (2.9) to find $y^H F x = 0$. This implies the stationarity property $\theta'(0) = 0$. Differentiate (2.11) with respect to t twice, and set $t = 0$, and there appears

$$\theta''(0) = \frac{2}{\|F\|_2}\frac{s^H}{y^H x}x'(0).$$

Now the standard eigenvector sensitivity analysis gives

$$x'(0) = \sum_{\theta_i \neq \theta} \frac{y_i^H E x}{(\theta - \theta_i)y_i^H x_i} x_i.$$

See, for example, Golub and Van Loan [20, p. 345]. From the above two formulas, up to the second order of $\|F\|_2$, we obtain

$$|\lambda - \theta| \leq \frac{\|s^H\|_2\,\|r\|_2}{\mathrm{gap}(\theta, B)} \left(\frac{1}{|y^H x|} \sum_{\theta_i \neq \theta} \frac{1}{|y_i^H x_i|} \right). \tag{2.12}$$

Here $\mathrm{gap}(\theta, B) = \min_{\theta_i \neq \theta} |\theta - \theta_i|$. Note that the term in the parentheses involves the condition numbers of the eigenvalues $\{\theta_i\}$ of B.

The bound (2.12) shows that the accuracy of the Rayleigh–Ritz value θ is proportional to the product of the left and right residuals and the inverse of the gap in eigenvalues of B. We call this the *semiquadratic convergence*. Since $\mathrm{gap}(\theta, B)$ is not computable, we use the $\mathrm{gap}(\theta, T_j)$ to approximate $\mathrm{gap}(\theta, B)$ when $\|F\|_2$ is small. From (2.10) and (2.12), we advocate accepting θ as an approximate eigenvalue of A if

$$\min \left\{ \|s^H\|_2, \|r\|_2, \frac{\|s^H\|_2\ \|r\|_2}{\mathrm{gap}(\theta, T_j)} \right\} \leq \tau_c, \tag{2.13}$$

where τ_c is a given accuracy threshold. Note that for ill-posed problems, small residuals (backward errors) do not imply high eigenvalue accuracy (small forward error). In this case, the estimate is optimistic. In any case, since both the left and right approximate eigenvectors are available, the approximate eigenvalue condition numbers are readily computable. This detects ill-posedness in an eigenvalue problem. See numerical example 5 in section 7.

It is well known that for Hermitian matrices, the Lanczos algorithm reveals first the outer and well-separated eigenvalues [35]. In the block Hermitian Lanczos algorithm with blocksize p, the outer eigenvalues and the eigenvalue clusters of order up to p that are well separated from the remaining spectra converge first [19, 41]. This qualitative understanding of convergence has been extended to the block Arnoldi algorithm for non-Hermitian eigenproblems in [42, 24].

2.3. Maintaining the biorthogonality of the Lanczos vectors.

The quantities computed in the block Lanczos algorithm in the presence of finite precision arithmetic have different properties than the corresponding exact quantities. The biorthogonality property, (2.6), fails to hold, and the columns of the matrices $\boldsymbol{\mathcal{P}}_j$ and $\boldsymbol{\mathcal{Q}}_j$ are spanning sets but not bases. The loss of linear independence in the matrices $\boldsymbol{\mathcal{P}}_j$ and $\boldsymbol{\mathcal{Q}}_j$ computed by the three-term recurrence is coherent; as a Rayleigh–Ritz triplet converges to an eigentriplet of A, copies of the Rayleigh–Ritz values appear. At this iteration, $\boldsymbol{\mathcal{Q}}_j$ is singular because it maps a group of right eigenvectors of T_j to an eigenvector of A.

For example, in a Lanczos run of 100 iterations, one may observe 5 copies of the dominant eigenvalue of A among the Rayleigh–Ritz values. This increases the number of iterations required to complete a given task. As a partial remedy, we advocate maintaining *local biorthogonality* to ensure the biorthogonality among consecutive Lanczos vectors in the three-term recurrences [14]. Local biorthogonality is maintained as follows. After step 3.2 in Figure 2.1,

$$\begin{aligned} R_j &:= R_j - P_j(Q_j^T R_j), \\ S_j &:= S_j - Q_j(P_j^T S_j). \end{aligned}$$

Repeating this inner loop increases the number of floating point operations in a Lanczos iteration. However, no new data transfer is required, and without repetition the local biorthogonality would normally be swamped. The cost effectiveness seems indisputable.

Another limitation of simple implementations of the three-term recurrences is that the multiplicity of an eigenvalue of A is not related in any practical way to the multiplicity of a Rayleigh–Ritz value. To reveal the multiplicity or clustering of an eigenvalue it typically suffices to explicitly enforce (2.6). This variation has been called a Lanczos algorithm with *full rebiorthogonalization* [38]. It is maintained by incorporating a variant of the Gram–Schmidt process called the two-sided modified Gram–Schmidt biorthogonalization (TSMGS) [36]. After step 3.6 in Figure 2.1, we biorthogonalize P_{j+1} and Q_{j+1} in place against all previous Lanczos vectors $\boldsymbol{\mathcal{P}}_j = [\,P_1, P_2, \ldots, P_j\,]$ and $\boldsymbol{\mathcal{Q}}_j = [\,Q_1, Q_2, \ldots, Q_j\,]$:

```
for $i = 1, 2, \ldots, j$
    $P_{j+1} := P_{j+1} - P_i(Q_i^T P_{j+1})$
    $Q_{j+1} := Q_{j+1} - Q_i(P_i^T Q_{j+1})$
end
```

Maintaining full biorthogonality substantially increases the cost per iteration of the Lanczos algorithm. To be precise, at Lanczos iteration j, an additional $8p^2jn$ flops is required. More importantly all the computed Lanczos vectors are accessed at each iteration. This is very often the most costly part of a Lanczos run, although there are cases where the matrix-vector multiplications may be the dominating factor. A less-costly alternative to full biorthogonality is presented in section 4.

3. An adaptive block Lanczos algorithm.

In this section, we present an adaptive block scheme. This algorithm has the flexibility to adjust the blocksize to the multiplicity or the order of a cluster of desired eigenvalues. In addition, the algorithm can be used to cure (near) breakdowns.

3.1. Augmenting the Lanczos vectors.

In a variable block Lanczos algorithm, at the jth iteration, P_j and Q_j have p_j columns, respectively. At the next $(j+1)$th iteration, the number of columns of the Lanczos vectors P_{j+1} and Q_{j+1} can be increased by k as follows.

First note that for any n by k matrices $\widehat{P}_{j+1}$ and $\widehat{Q}_{j+1}$, the basic three-term recurrences (2.1) and (2.2) also hold with augmented $(j+1)$th Lanczos vectors $[P_{j+1}\ \widehat{P}_{j+1}]$ and $[Q_{j+1}\ \widehat{Q}_{j+1}]$:

$$\begin{bmatrix} B_{j+1} & 0 \end{bmatrix} \begin{bmatrix} P_{j+1}^T \\ \widehat{P}_{j+1}^T \end{bmatrix} = P_j^T A - A_j P_j^T - C_j P_{j-1}^T$$

and

$$\begin{bmatrix} Q_{j+1} & \widehat{Q}_{j+1} \end{bmatrix} \begin{bmatrix} C_{j+1} \\ 0 \end{bmatrix} = AQ_j - Q_j A_j - Q_{j-1} B_j.$$

Provided that

$$\begin{bmatrix} P_{j+1} & \widehat{P}_{j+1} \end{bmatrix}^T \begin{bmatrix} Q_{j+1} & \widehat{Q}_{j+1} \end{bmatrix} \tag{3.1}$$

is *nonsingular*, the Lanczos procedure continues as before under the substitutions

$$P_{j+1} \leftarrow \begin{bmatrix} P_{j+1} & \widehat{P}_{j+1} \end{bmatrix}, \quad Q_{j+1} \leftarrow \begin{bmatrix} Q_{j+1} & \widehat{Q}_{j+1} \end{bmatrix}$$

with proper normalization and

$$B_{j+1} \leftarrow \left[\begin{array}{cc} B_{j+1} & 0 \end{array}\right], \quad C_{j+1} \leftarrow \left[\begin{array}{c} C_{j+1} \\ 0 \end{array}\right].$$

The only other constraint on $\widehat{P}_{j+1}$ and $\widehat{Q}_{j+1}$ is that they satisfy the biorthogonality condition among the Lanczos vectors; i.e., it is required that

$$\widehat{P}_{j+1}^T \boldsymbol{\mathcal{Q}}_j = 0 \quad \text{and} \quad \boldsymbol{\mathcal{P}}_j^T \widehat{Q}_{j+1} = 0.$$

As a consequence, the adaptive block scheme has the same governing equations and the same resulting Rayleigh–Ritz approximation properties as the basic block Lanczos method described in section 2.

Before we turn to the usage of the adaptive block scheme, we discuss the choice of the increment vectors $\widehat{P}_{j+1}^T$ and $\widehat{Q}_{j+1}$. Ideally we would like to choose augmentations so that the resulting matrix $P_{j+1}^T Q_{j+1}$ is well conditioned. To be precise we want the smallest singular value of $P_{j+1}^T Q_{j+1}$ to be larger than the given threshold τ_b, say $\tau_b = 10^{-8}$ in double precision. However, there may not exist $\widehat{P}_{j+1}^T$ and $\widehat{Q}_{j+1}$ such that the given threshold τ_b is satisfied. A natural choice to choose $\widehat{P}_{j+1}$ and $\widehat{Q}_{j+1}$ in practice is to biorthogonalize a pair of random n by k vectors to the previous Lanczos vectors. In other words, the vectors $\widehat{P}_{j+1}$ and $\widehat{Q}_{j+1}$ are computed by applying TSMGS (see section 2.3) to a pair of random n by k vectors. The construction is repeated a few times (say, 3 at most) if necessary to ensure that the smallest singular value of (3.1) is larger than a threshold. We observe that this works well in practice.

3.2. Adaptive blocking for clustered eigenvalues. If A has an eigenvalue of multiplicity greater than the blocksize, then the Rayleigh–Ritz values converge slowly to this group of eigenvalues [12, 19, 22, 3]. In some applications, information about multiplicity is available a priori and then the blocksize can be chosen accordingly. But when this information is not available, it is desirable to adjust the blocksize using the information obtained during the iteration.

In any variable block implementation of the Lanczos algorithm in which the biorthogonality of the computed Lanczos vectors is maintained, it is advantageous to increase the blocksize to the order of the largest cluster of Rayleigh–Ritz values, $\{\theta_i\}$. The adaptive block scheme proposed in section 3.1 offers such flexibility.

The cluster of Rayleigh–Ritz values about θ_i is the set of all θ_k such that

$$|\theta_i - \theta_k| \le \eta \max(|\theta_i|, |\theta_k|), \tag{3.2}$$

where η is a user-specified clustering threshold. The order of the largest cluster of Rayleigh–Ritz values is computed whenever we test for convergence, and the blocksize is increased to the order of the largest cluster.

3.3. Adapting the blocksize to treat breakdown. A second reason to increase the blocksize is to overcome a breakdown in the block Lanczos algorithm. Recall from section 2.1 that breakdown occurs when $R_j^T S_j$ is singular. There are two cases:

I. Either R_j or S_j is rank deficient.

II. Both R_j and S_j are not rank deficient but $R_j^T S_j$ is.

Exact breakdowns are rare, but near breakdowns (i.e., $R_j^T S_j$ has singular values close to 0) do occur. In finite precision arithmetic this can cause numerical instability.

In case I, if S_j vanishes in step 3.2 of Figure 2.1 of the basic block Lanczos algorithm, an invariant subspace is detected. To restart the Lanczos procedure choose

Q_{j+1} to be any vector such that $\boldsymbol{\mathcal{P}}_j^T Q_{j+1} = 0$. If S_j is just (nearly) rank deficient, then after the QR decomposition of S_j, $S_j = Q_{j+1}C_{j+1}$, we biorthogonalize Q_{j+1} to the previous left Lanczos vectors $\boldsymbol{\mathcal{P}}_j$. This also effectively expands the Krylov subspace and continues the procedure. Rank deficiency of R_j is treated similarly. Note that in this case, the blocksize is not changed. This generalizes the treatment suggested by Wilkinson for the unblocked Lanczos procedure [52, p. 389].

Case II is called a *serious breakdown* [52]. Let us first examine the case of exact breakdown. Let $R_j = P_{j+1}B_{j+1}^T$ and $S_j = Q_{j+1}C_{j+1}$ be the QR decompositions of R_j and S_j. In this case, $P_{j+1}^T Q_{j+1}$ is singular. Suppose that $P_{j+1}^T Q_{j+1}$ has the SVD

$$P_{j+1}^T Q_{j+1} = U \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix} V^H,$$

where Σ is nonsingular if it exists (Σ may be 0 by 0). Let us see how to augment P_{j+1} and Q_{j+1} so that $P_{j+1}^T Q_{j+1}$ is nonsingular. For clarity, drop the subscript $j+1$ and partition $P\bar{U}$ and QV into

$$P\bar{U} = \begin{bmatrix} P_{(1)} & P_{(2)} \end{bmatrix} \qquad \text{and} \qquad QV = \begin{bmatrix} Q_{(1)} & Q_{(2)} \end{bmatrix}.$$

Here the number of columns of $P_{(2)}$ and $Q_{(2)}$ is the number of zero singular values of $P^T Q$. Let the augmented Lanczos vectors be

$$P := \begin{bmatrix} P_{(1)} & P_{(2)} & \widehat{P} \end{bmatrix} \qquad \text{and} \qquad Q := \begin{bmatrix} Q_{(1)} & Q_{(2)} & \widehat{Q} \end{bmatrix},$$

where

$$\widehat{P} = (I - \Pi_j)^T \bar{Q}_{(2)} \qquad \text{and} \qquad \widehat{Q} = (I - \Pi_j)\bar{P}_{(2)}.$$

$\Pi_j = \boldsymbol{\mathcal{Q}}_j \boldsymbol{\mathcal{P}}_j^T$ is the oblique projector. The biorthogonality condition (2.6) and then the orthonormality of the columns of $\begin{bmatrix} P_{(1)} & P_{(2)} \end{bmatrix}$ yield

$$P_{(1)}^T \widehat{Q} = P_{(1)}^T (I - \Pi_j)\bar{P}_{(2)} = P_{(1)}^T \bar{P}_{(2)} = 0$$

and

$$P_{(2)}^T \widehat{Q} = P_{(2)}^T (I - \Pi_j)\bar{P}_{(2)} = P_{(2)}^T \bar{P}_{(2)} = I.$$

Similarly, $\widehat{P}^T Q_{(1)} = 0$ and $\widehat{P}^T Q_{(2)} = I$. Therefore, we have

$$P^T Q = \begin{bmatrix} \Sigma & 0 & 0 \\ 0 & 0 & I \\ 0 & I & \widehat{P}^T \widehat{Q} \end{bmatrix},$$

which is nonsingular for any $\widehat{P}^T \widehat{Q}$. Therefore, we conclude that exact breakdowns are always curable by the adaptive blocksize technique.

However, for the near breakdown case, the situation is more complicated. The above choice may not succeed in increasing the smallest singular value of $P_{j+1}^T Q_{j+1}$ above a specific given threshold, τ_b. The difficulty involves the fact that the norms of $\widehat{P}$ and $\widehat{Q}$ can be large because of the use of oblique projector Π_j. In our implementation, we have chosen $\widehat{P}$ and $\widehat{Q}$ by dualizing a pair of random n by k vectors to the previous

Lanczos vectors as described in section 3.1. The increment to the blocksize is the number of singular values of $P_{j+1}^T Q_{j+1}$ below a specified threshold.

Another scheme for adjusting the block size to cure (near) breakdown is the *look-ahead* strategy [38, 17]. In the look-ahead Lanczos algorithm, the spans of the columns of $\boldsymbol{\mathcal{P}}_j$ and $\boldsymbol{\mathcal{Q}}_j$ remain within $\boldsymbol{\mathcal{K}}(P_1^T, A)$ and $\boldsymbol{\mathcal{K}}(Q_1, A)$, respectively. Specifically, P_{j+1}^T and Q_{j+1} are augmented by

$$\widehat{P} = (I - \Pi_j)^T A^T P_{j+1} = A^T P_{j+1} - P_j C_{j+1}^T$$

and

$$\widehat{Q} = (I - \Pi_j) A Q_{j+1} = A Q_{j+1} - Q_j B_{j+1}.$$

If $[P_{j+1}\ \widehat{P}^T]\ [Q_{j+1}\ \widehat{Q}]$ is not (nearly) singular, then one step of look-ahead is successful and P_{j+1} and Q_{j+1} are obtained from P and Q, respectively, after normalization. Since

$$\operatorname{span}(\boldsymbol{\mathcal{Q}}_{j+1}) = \operatorname{span}(\boldsymbol{\mathcal{Q}}_j, [Q_{j+1}, \widehat{Q}])$$

and

$$\begin{aligned}\operatorname{span}(\boldsymbol{\mathcal{Q}}_{j+2}) &= \operatorname{span}(\boldsymbol{\mathcal{Q}}_j, [Q_{j+1}, \widehat{Q}], A[Q_{j+1}, \widehat{Q}])\\ &= \operatorname{span}(\boldsymbol{\mathcal{Q}}_{j+1}, A^2 Q_{j+1}),\end{aligned}$$

Q_{j+2} has no more columns than Q_{j+1} prior to augmentation. That is, the block size doubles at step $j+1$ only and then returns to the ambient block size at the following step $j+2$. It may be necessary to repeatedly augment the $(j+1)$th Lanczos block-vectors [36]. In contrast, we have shown that the adaptive strategy has the property that an exact breakdown is cured in using a fixed number of augmentation vectors. Moreover, to reveal clustered eigenvalues and to eliminate a potential source of slow convergence, we store $\boldsymbol{\mathcal{P}}_j$ and $\boldsymbol{\mathcal{Q}}_j$ and maintain biorthogonality (see section 4). We have found the adaptive block scheme to be a viable alternative to look-ahead strategies here.

4. Maintaining semibiorthogonality.

In this section we present a form of *limited* rebiorthogonalization that is more efficient than the full rebiorthogonalization described in section 2.3. This method extends the block Hermitian Lanczos algorithm with *partial* reorthogonalization to the non-Hermitian case [22]. Instead of maintaining full biorthogonality (section 2.3), only *semibiorthogonality* is maintained at each iteration; i.e., for $j \geq 1$,

$$d_{j+1} = \max\left(\frac{\|\boldsymbol{\mathcal{P}}_j^T Q_{j+1}\|_1}{\|\boldsymbol{\mathcal{P}}_j\|_1 \|Q_{j+1}\|_1}, \frac{\|\boldsymbol{\mathcal{Q}}_j^T P_{j+1}\|_1}{\|\boldsymbol{\mathcal{Q}}_j\|_1 \|P_{j+1}\|_1} \right) \leq \sqrt{\epsilon}, \tag{4.1}$$

where ϵ is the roundoff error unit. This generalizes the definition of semibiorthogonality for the unblocked Lanczos algorithm [14]. We will show that semibiorthogonality requires less computation and data transfer to maintain than full biorthogonality. In particular, $\boldsymbol{\mathcal{P}}_j$ and $\boldsymbol{\mathcal{Q}}_j$ are accessed only at certain iterations.

In section 4.1 we show how to monitor the loss of numerical biorthogonality without significantly increasing the number of floating point operations in the Lanczos recurrences. In section 4.2 we show how best to correct the loss of biorthogonality.

4.1. Monitoring the loss of biorthogonality. When the Lanczos algorithm is implemented in finite precision arithmetic, the computed quantities can be modeled by perturbed three-term recurrences:

$$B_{j+1}P_{j+1}^T = P_j^T A - A_j P_j^T - C_j P_{j-1}^T - F_j^T, \tag{4.2}$$

$$Q_{j+1}C_{j+1} = AQ_j - Q_j A_j - Q_{j-1}B_j - G_j, \tag{4.3}$$

where F_j and G_j represent the roundoff error introduced at iteration j. By applying the standard model of the rounding errors committed in floating point arithmetic [52], it can be shown that to first order in roundoff errors there holds

$$\begin{aligned} \|F_j\|_F &\le \mathbf{u}(\|A\|_1\,\|P_j\|_1 + \|A_j\|_1\,\|P_j\|_1 + \|C_j\|_1\,\|P_{j-1}\|_1), \\ \|G_j\|_F &\le \mathbf{u}(\|A\|_1\,\|Q_j\|_1 + \|A_j\|_1\,\|Q_j\|_1 + \|B_j\|_1\,\|Q_{j-1}\|_1), \end{aligned}$$

where $\mathbf{u}$ is a constant multiple of the roundoff error unit ϵ. The governing equations for the computed quantities are

$$\boldsymbol{\mathcal{P}}_j^T A = T_j \boldsymbol{\mathcal{P}}_j^T + E_j B_{j+1} P_{j+1}^T + \boldsymbol{\mathcal{F}}_j^T, \tag{4.4}$$

$$A\boldsymbol{\mathcal{Q}}_j = \boldsymbol{\mathcal{Q}}_j T_j + Q_{j+1}C_{j+1}E_j^T + \boldsymbol{\mathcal{G}}_j, \tag{4.5}$$

where the matrices $\boldsymbol{\mathcal{F}}_j = [\,F_1,\, F_2, \ldots, F_j\,]$ and $\boldsymbol{\mathcal{G}}_j = [\,G_1,\, G_2, \ldots, G_j\,]$ are such that

$$\max(\|\boldsymbol{\mathcal{F}}_j\|_F, \|\boldsymbol{\mathcal{G}}_j\|_F) \le \mathbf{u}(\|A\|_1 + \|T_j\|_1)\max(\|\boldsymbol{\mathcal{P}}_j\|_F, \|\boldsymbol{\mathcal{Q}}_j\|_F). \tag{4.6}$$

A detailed analysis for the unblocked case can be found in [2, 14].

Now we use this model of rounding errors in the Lanczos process to quantify the propagation of the loss of biorthogonality from iteration to iteration. The biorthogonality of the $(j+1)$th Lanczos vectors to the previous Lanczos vectors can be measured using the short vectors

$$X_j = \boldsymbol{\mathcal{P}}_j^T Q_{j+1} \qquad \text{and} \qquad Y_j = P_{j+1}^T \boldsymbol{\mathcal{Q}}_j.$$

In the following, we show that these vectors satisfy perturbed three-term recurrences which we can use to efficiently monitor the biorthogonality loss.

The recurrence for X_j is derived as follows. Note that

$$\boldsymbol{\mathcal{P}}_j^T Q_j = \begin{bmatrix} X_{j-1} \\ 0 \end{bmatrix} + E_j. \tag{4.7}$$

Let $W_{ij} = P_i^T Q_j$. Multiply (4.3) by $\boldsymbol{\mathcal{P}}_j^T$ on the left, substitute in (4.4) × Q_j, and there appears

$$\begin{aligned} X_j C_{j+1} = {} & T_j \boldsymbol{\mathcal{P}}_j^T Q_j - \boldsymbol{\mathcal{P}}_j^T Q_j A_j - \boldsymbol{\mathcal{P}}_j^T Q_{j-1} B_j \\ & + E_j B_{j+1} W_{j+1,j} + \boldsymbol{\mathcal{F}}_j^T Q_j - \boldsymbol{\mathcal{P}}_j^T G_j. \end{aligned} \tag{4.8}$$

Substitute (4.7) above and (2.3), the definition of T_j, and simplify to find

$$T_j \boldsymbol{\mathcal{P}}_j^T Q_j - \boldsymbol{\mathcal{P}}_j^T Q_j A_j = T_j \begin{bmatrix} X_{j-1} \\ 0 \end{bmatrix} - \begin{bmatrix} X_{j-1} \\ 0 \end{bmatrix} A_j + E_{j-1} B_j. \tag{4.9}$$

In addition, we have the identity

$$\boldsymbol{\mathcal{P}}_j^T Q_{j-1} = \begin{bmatrix} X_{j-2} \\ 0 \\ 0 \end{bmatrix} + E_{j-1} + W_{j,j-1} E_j. \tag{4.10}$$

Substituting (4.9) and (4.10) into (4.8) finally yields

$$\begin{aligned} X_j C_{j+1} = T_j \begin{bmatrix} X_{j-1} \\ 0 \end{bmatrix} - \begin{bmatrix} X_{j-1} \\ 0 \end{bmatrix} A_j - \begin{bmatrix} X_{j-2} \\ 0 \\ W_{j,j-1} \end{bmatrix} B_j \\ + E_j B_{j+1} W_{j+1,j} + O(\mathbf{u}_j), \end{aligned} \tag{4.11}$$

where $O(\mathbf{u}_j)$ represents the local rounding error term $\boldsymbol{\mathcal{F}}_j^T Q_j - \boldsymbol{\mathcal{P}}_j^T G_j$ and

$$\mathbf{u}_j = \mathbf{u}(\|T_j\|_1 + \|A\|_1) \max(\|\boldsymbol{\mathcal{P}}_j\|_F, \|\boldsymbol{\mathcal{Q}}_j\|_F).$$

The similar analysis of the left Lanczos vectors yields

$$\begin{aligned} B_{j+1} Y_j = [\,Y_{j-1},\ 0\,] T_j - A_j [\,Y_{j-1},\ 0\,] - C_j [\,Y_{j-2},\ 0,\ W_{j-1,j}\,] \\ + W_{j,j+1} C_{j+1} E_j + O(\mathbf{u}_j). \end{aligned} \tag{4.12}$$

Equations (4.11) and (4.12) model the propagation of the loss of the numerical biorthogonality among Lanczos vectors from iteration to iteration. The following algorithm implements these recurrence relations to monitor the biorthogonality loss. Note that the scalar parameter $\hat{d}_{j+1}$ is our measure of the biorthogonality. When $\hat{d}_{j+1} > \sqrt{\epsilon}$, then TSMGS is invoked to recover biorthogonality as described in the next section.[1]

Algorithm for monitoring the loss of biorthogonality.

```
Initially, when $j = 1$, we set $X_1 = 0$, $Y_1 = 0$, $d_1 = \mathbf{u}$, compute
$X_2 = P_1^T Q_2$, $Y_2 = P_2^T Q_1$, and let $W_1^{(l)} = Y_2$, $W_1^{(r)} = X_2$. When
$j > 1$.
   1. $W_2^{(l)} = P_{j+1}^T Q_j$
   2. $X_3 = T_j \begin{bmatrix} X_2 \\ 0 \end{bmatrix} - \begin{bmatrix} X_2 \\ 0 \end{bmatrix} A_j - \begin{bmatrix} X_1 \\ W_1^{(l)} \end{bmatrix} B_j + \begin{bmatrix} 0 \\ B_{j+1} W_2^{(l)} \end{bmatrix}$
   3. $X_3 := (X_3 + F_j) C_{j+1}^{-1}$
   4. $X_1 = \begin{bmatrix} X_2 \\ 0 \end{bmatrix}; \quad X_2 = X_3$
   5. $W_2^{(r)} = P_j^T Q_{j+1}$
   6. $Y_3 = \begin{bmatrix} Y_2 & 0 \end{bmatrix} T_j - A_j \begin{bmatrix} Y_2 & 0 \end{bmatrix} - C_j \begin{bmatrix} Y_1 & W_1^{(r)} \end{bmatrix} + \begin{bmatrix} 0 & W_2^{(r)} C_{j+1} \end{bmatrix}$
   7. $Y_3 := B_{j+1}^{-1} (Y_3 + F_j^T)$
   8. $Y_1 = \begin{bmatrix} Y_2 & 0 \end{bmatrix}; \quad Y_2 = Y_3$
   9. $W_1^{(l)} = W_2^{(l)}; \quad W_1^{(r)} = W_2^{(r)}$
  10. $\hat{d}_{j+1} = \max(\|X_2\|_1 / (\|\boldsymbol{\mathcal{P}}_j\|_1 \|Q_{j+1}\|_1), \|Y_2\|_\infty / (\|\boldsymbol{\mathcal{Q}}_j\|_1 \|P_{j+1}\|_1))$
```

The matrix F_j is a random matrix scaled to have norm $\mathbf{u}_j$ to simulate the roundoff errors in the three-term recurrences. The number of floating point operations per iteration of the monitoring algorithm is $2j^2 + O(n)$, where the $2j^2$ is for the multiplications by T_j in steps 2 and 6 above and the n comes from the "inner products" of block Lanczos vectors in steps 1 and 5 above. If the block tridiagonal structure of T_j is taken in account, then the cost is just $O(n)$. Therefore the cost of the monitoring algorithm is not significant, as promised.

[1] To economize on storage there is a subtle change of notation in the following monitoring algorithm. At Lanczos iteration j, the vectors X_{j-1}, X_j, and X_{j+1} are denoted X_1, X_2, and X_3, and the previous X_k are not stored. Similar conventions apply to $\{Y_i\}$ and $\{W_{i,k}\}$.

4.2. Correcting the loss of biorthogonality. When action is required to maintain semibiorthogonality (4.1), TSMGS (see section 2.3) is invoked to rebiorthonormalize or *correct* the candidate Lanczos vectors P_{j+1} and Q_{j+1}. Recall from (4.11) that the sequence $\{X_j\}$ satisfies a perturbed three-term recurrence. Correcting Q_{j+1} annihilates the $O(\sqrt{\epsilon})$ matrix X_j, but at the next Lanczos iteration X_{j+1} will be a multiple of the nearly $O(\sqrt{\epsilon})$ matrix X_{j-1}. Instead, as Q_{j+1} is corrected to maintain semibiorthogonality, we also correct Q_j; in this way the biorthogonality of the following Lanczos vectors can deteriorate gradually. The similar comments hold for the left Lanczos vectors. There is a much better way to do this than to apply TSMGS at consecutive iterations to the pairs of P_j and Q_j and P_{j+1} and Q_{j+1}, respectively. Instead, as the columns of $\boldsymbol{\mathcal{P}}_j$ and $\boldsymbol{\mathcal{Q}}_j$ are transferred from slow storage to the computational unit to correct P_{j+1} and Q_{j+1}, the previous Lanczos vectors P_j and Q_j also can be *retroactively* corrected. This halves the amount of data transfer required.

RETROACTIVE TSMGS. Biorthogonalize P_j, P_{j+1}, Q_j, and Q_{j+1} against the previous Lanczos vectors *in place*.

```
for $i = 1, 2, \dots, j-1$
    $P_j := P_j - P_i(Q_i^T P_j)$
    $P_{j+1} := P_{j+1} - P_i(Q_i^T P_{j+1})$
    $Q_j := Q_j - Q_i(P_i^T Q_j)$
    $Q_{j+1} := Q_{j+1} - Q_i(P_i^T Q_{j+1})$
end
$P_{j+1} := P_{j+1} - P_j\,(Q_j^T P_{j+1})$
$Q_{j+1} := Q_{j+1} - Q_j(P_j^T Q_{j+1})$
```

We do *not* update the QR decompositions and SVDs computed in the basic Lanczos algorithm after retroactive TSMGS for the same technical reasons discussed in section 6.3 of [14] for the unblocked Lanczos algorithm.

5. The ABLE method.

5. The ABLE method. In summary, the ABLE method presented in Figure 5.1 incorporates an adaptive blocking scheme (section 3) into the basic block Lanczos algorithm (section 2) and maintains the local and semibiorthogonality of Lanczos vectors (section 4). Specifically, we have the following:

- At step 3.3, we suggest the use of (2.13) in section 2.2 as the stopping criterion. Then, at the end of a Lanczos run, we compute the residual norms $\|s^H\|_2$ and $\|r\|_2$ corresponding to the converged Rayleigh–Ritz triplets (θ, y^H, x). See (2.7) and (2.8) in section 2.2. Note that the theory in section 2.2 is based on the exact biorthogonality. When only semibiorthogonality is maintained, $\theta'(0)$ is no longer zero. However, using (4.4), (4.5), and semibiorthogonality (4.1), it is easy to see that $\theta'(0)$ is still in the magnitude of $\sqrt{\epsilon}$. Thus, as far as $\|F\|_2$ is not too small (not less than $O(\sqrt{\epsilon})$), the second term in the expansion for λ still dominates the first term $\theta'(0)\|F\|_2$, and therefore, (2.13) would be valid. (Specifically, $y^H r \sim \sqrt{\epsilon}\|r\|_2$ and the first term in the expansion satisfies $\theta'(0)\|F\|_2 \sim \frac{1}{|y^H x|}\sqrt{\epsilon}\|F\|_2$.)
- At step 3.4, (3.2) is used to compute the order of the largest cluster as described in section 3.2.
- For step 3.7, see section 3.3 for an explanation.
- At step 3.9, τ_b is a threshold for breakdown. $\min(\Sigma)$ is the smallest singular value of the matrix $P_{j+1}^T Q_{j+1}$. If there is (near) breakdown and/or the order of the largest cluster of the converged Rayleigh–Ritz values is larger than the blocksize, then the blocks are augmented as described in section 3.1.

```
1. Choose starting vectors $P_1$ and $Q_1$ so that $P_1^T Q_1 = I$
2. $R = (P_1^T A)^T$ and $S = AQ_1$
3. For $j = 1, 2, \ldots$ until convergence
   3.1. $A_j = P_j^T S$
   3.2. $R := R - P_j A_j^T$ and $S := S - Q_j A_j$
   3.3. Compute the eigen-decomposition of $T_j$, and test for convergence
   3.4. Find the largest order $\delta$ of the clustering of converged Rayleigh–Ritz
        values
   3.5. Local biorthogonality: $R := R - P_j(Q_j^T R)$ and $S := S - Q_j(P_j^T S)$
   3.6. Compute the QR decompositions: $R = P_{j+1} B_{j+1}^T$ and $S = Q_{j+1} C_{j+1}$
   3.7. If $R$ or $S$ (or both) is rank deficient, apply TSMGS to biorthogonalize
        $P_{j+1}$ and $Q_{j+1}$ against the previous Lanczos vectors
   3.8. Compute the SVD: $P_{j+1}^T Q_{j+1} = U \Sigma V^H$
   3.9. Increase blocksize if $\min(\Sigma) < \tau_b$ and/or $\delta > p_j$
  3.10. $B_{j+1} := B_{j+1} U \Sigma^{1/2}$ and $C_{j+1} := \Sigma^{1/2} V^H C_{j+1}$
  3.11. $P_{j+1} := P_{j+1} \bar{U} \Sigma^{-1/2}$ and $Q_{j+1} := Q_{j+1} V \Sigma^{-1/2}$
  3.12. Monitor the loss of biorthogonality, and correct if necessary
  3.13. $R = (P_{j+1}^T A - C_{j+1} P_j^T)^T$ and $S = AQ_{j+1} - Q_j B_{j+1}$
```

Fig. 5.1. *ABLE method.*

- Algorithms for monitoring the loss of biorthogonality and maintaining semibiorthogonality at step 3.12 are described in sections 4.1 and 4.2.

The non-Hermitian Lanczos algorithm is also called the two-sided Lanczos algorithm because both the operations

$$X^T A \quad \text{and} \quad AX$$

are required at each iteration. A is referenced only as a rule to compute these matrix-vector products. Because of this feature, the algorithm is well suited for large sparse matrices or large structured dense matrices for which matrix-vector products can be computed cheaply. The efficient implementation of these products depends on the data structure and storage format for the A matrix and the Lanczos vectors.

If no Lanczos vectors are saved, the three-term recurrences can be implemented using only six block vectors of length n. To maintain the semibiorthogonality of the computed Lanczos vectors $\boldsymbol{\mathcal{P}}_j$ and $\boldsymbol{\mathcal{Q}}_j$, it is necessary to store these vectors in core or out-of-core memory. This consumes a significant amount of memory. The user must be conscious of how much memory is needed for each application. For very large matrices it may be best to store the Lanczos vectors out-of-core. After each Lanczos iteration, save the current Lanczos vectors to an auxiliary storage device. The Lanczos vectors are recalled in the procedure TSMGS for rebiorthogonalization and when the converged Rayleigh–Ritz vectors are computed at the end of a Lanczos run.

A block Lanczos algorithm is ideal for application codes that represent A out-of-core. The main cost of a Lanczos iteration, with or without blocks, is accessing A. Block algorithms compute the matrix block vectors product with only one pass over the data structure defining A, with a corresponding savings of work.

The most time-consuming steps in a Lanczos run are to

1. apply the matrix A (from the left and the right),
2. apply retroactive TSMGS to maintain semibiorthogonality, and

3. solve the eigenproblem for the block tridiagonal matrix T_j when j increases.
Items 1 and 2 have been addressed already (see the above and section 4.2). For item 3, we presently use the QR algorithm for T_j. We note that it is not necessary to solve the eigenproblem for T_j at each Lanczos iteration. A way to reduce such cost is to solve the eigenvalue problem for T_j only after a correction iteration has been made to maintain semibiorthogonality. This technique utilizes the connection between the loss of the biorthogonality and convergence [33, 35, 2].

6. A spectral transformation ABLE method.

In this section we briefly discuss how to use the ABLE method to compute some eigenvalues of the generalized eigenvalue problem

$$Kx = \lambda Mx \tag{6.1}$$

nearest an arbitrary complex number, σ. We assume that $K - \sigma M$ is nonsingular and that it is feasible to solve the linear system of equations with coefficient matrix $K - \sigma M$. The reward for solving this linear system of equations is the rapid convergence of the Lanczos algorithm. In section 7 we apply the ABLE method to such a generalized eigenvalue problem arising in magneto-hydro-dynamics (MHD).

We apply a popular shift-and-invert strategy to the pair (K, M) with shift σ [16]. In this approach, the ABLE method is applied with

$$A = (K - \sigma M)^{-1}M. \tag{6.2}$$

The eigenvalues, μ, of A are $\mu = 1/(\lambda - \sigma)$. The outer eigenvalues of A are now the eigenvalues of (K, M) nearest to σ. This spectral transformation also generally improves the separation of the eigenvalues of interest from the remaining eigenvalues of (K, M), a very desirable property.

When we apply the ABLE method to the matrix $A = (K - \sigma M)^{-1}M$, the governing equations become

$$\boldsymbol{\mathcal{P}}_j^T (K - \sigma M)^{-1} M = T_j \boldsymbol{\mathcal{P}}_j^T + E_j B_{j+1} P_{j+1}^T, \tag{6.3}$$

$$(K - \sigma M)^{-1} M \boldsymbol{\mathcal{Q}}_j = \boldsymbol{\mathcal{Q}}_j T_j + Q_{j+1} C_{j+1} E_j^T. \tag{6.4}$$

If (θ, w^H, z) is an eigentriplet of T_j, then from the above governing equations (6.3) and (6.4), the triplet

$$\left(\tilde{\lambda},\, \tilde{y}^H,\, \tilde{x}\right) := \left(\sigma + \frac{1}{\theta},\, w^H \boldsymbol{\mathcal{P}}_j^T (K - \sigma M)^{-1},\, \boldsymbol{\mathcal{Q}}_j z\right)$$

is an approximate eigentriplet of the matrix pair (K, M). The corresponding left and right residuals are

$$\begin{aligned} s^H &= \tilde{y}^H K - \tilde{\lambda} \tilde{y}^H M = -\frac{1}{\theta} w^H E_j B_{j+1} P_{j+1}^T, \\ r &= K\tilde{x} - \tilde{\lambda} M \tilde{x} = -\frac{1}{\theta} (K - \sigma M) Q_{j+1} C_{j+1} E_j^T z. \end{aligned}$$

The matrix-vector products $Y = [(K - \sigma M)^{-1}M]X$ and $Z^T = X^T[(K - \sigma M)^{-1}M]$ required in the inner loop of the algorithm can be performed by first computing the LU factorization of $K - \sigma M = LU$ and then solving the linear systems of equations $LUY = MX$ and $Z^T = X^T(LU)^{-1}M$ for Y and Z^T, respectively.

If K and M are real, and the shift σ is complex, one can still keep the Lanczos procedure in real arithmetic using a strategy proposed by Parlett and Saad [37].

In many applications, M is symmetric positive definite. In this case, one can avoid factoring M explicitly by preserving M-biorthogonality among the Lanczos vectors [16, 35, 11, 22]. Numerical methods for the case in which M is symmetric indefinite are discussed in [22, 32].

7. Summary of numerical examples. This section summarizes our numerical experience with the ABLE method. We have selected test eigenvalue problems from real applications to demonstrate the major features of the ABLE method. Each numerical example illustrates a property of the ABLE method. All test matrices presented here can be found in the test matrix collection for non-Hermitian eigenvalue problems [4].

The ABLE method has been implemented in Matlab 4.2 with sparse matrix computation functions. All numerical experiments are performed on a SUN Sparc 10 workstation with IEEE double precision floating point arithmetic. The tolerance value τ_c for the stopping criterion (2.13) is set to be 10^{-8}. The clustering threshold (3.2) is $\eta = 10^{-6}$. The breakdown threshold is $\tau_b = 10^{-8}$.

Example 1. The block algorithm accelerates convergence in the presence of multiple and clustered eigenvalues. When the desired eigenvalues are known in advance to be multiple or clustered, we should initially choose the blocksize as the expected multiplicity or the cluster order. For example, the largest eigenvalue of the 656 × 656 Chuck matrix has multiplicity 2. If we use the unblocked ABLE method, then at iteration 20 the converged Rayleigh–Ritz values,

$$\begin{gathered} 5.502378378875370e+00, \\ 1.593971696766128e+00, \end{gathered}$$

approximate the two largest *distinct* eigenvalues. But the multiplicity is not yet revealed. However, if we use the ABLE method with initial blocksize 2, then at iteration 7 the converged Rayleigh–Ritz values are

$$\begin{gathered} 5.502378378347202e+00, \\ 5.502378378869873e+00. \end{gathered}$$

Each computed Rayleigh–Ritz value agrees to 10 to 12 decimal digits compared with the one computed by the dense QR algorithm.

Example 2. Full biorthogonality is very expensive to maintain in terms of floating point operations and memory access. Based on our experience, maintaining semibiorthogonality is a reliable and much less expensive alternative. Our example is a 2500 × 2500 block tridiagonal coefficient matrix obtained by discretizing the two-dimensional model convection-diffusion differential equation

$$\begin{aligned} -\Delta u + 2p_1 u_x + 2p_2 u_y - p_3 u = f \quad &\text{in} \quad \Omega, \\ u = 0 \quad &\text{on} \quad \partial\Omega \end{aligned}$$

using finite differences, where Ω is the unit square $\{(x, y) \in \mathbf{R}^2, 0 \leq x, y \leq 1\}$. The eigenvalues of the coefficient matrix can be expressed analytically in terms of the parameters p_1, p_2, and p_3. In our test run, we choose $p_1 = 0.5$, $p_2 = 2$, and $p_3 = 1$. For this set of parameters, all eigenvalues of the resulting matrix A are positive real and distinct. With full biorthogonality, at iteration 132, the two largest

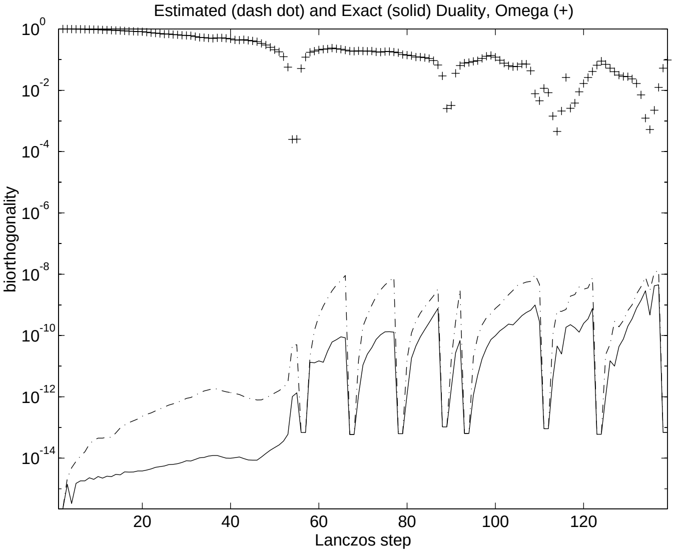


FIG. 7.1. *The exact (solid line) and estimated (dash-dot line) biorthogonality of the Lanczos vectors and the smallest singular values (+) of* $P_{j+1}^TQ_{j+1}$.

eigenvalues are converged. If we use the ABLE method with semibiorthogonality, at iteration 139, the same two largest eigenvalues are converged to the same accuracy. The difference is that only 8 corrections of biorthogonality loss are invoked to maintain semibiorthogonality, compared to 132 corrections for full biorthogonality.

In Figure 7.1 the solid and dotted lines display the exact and estimated biorthogonality of the computed Lanczos vectors, and the "+"-points concern breakdown and are the smallest singular values of $P_j^TQ_j$. The solid line plots

$$d_{j+1} = \max\left(\frac{\|\boldsymbol{\mathcal{P}}_j^TQ_{j+1}\|_1}{\|\boldsymbol{\mathcal{P}}_j\|_1\|Q_{j+1}\|_1}, \frac{\|\boldsymbol{\mathcal{Q}}_j^TP_{j+1}\|_1}{\|\boldsymbol{\mathcal{Q}}_j\|_1\|P_{j+1}\|_1}\right)$$

for $j = 1, 2, \ldots, 132$. Each sharp decrease corresponds to a correction. The dotted line plots the estimate, $\hat{d}_{j+1}$, of this quantity computed by the monitoring algorithm of section 4.2. Correction iterations are taken when the dotted line increases to the threshold $\sqrt{\epsilon}$, where ϵ denotes the machine precision. The observation that the solid line is below the dotted line indicates that the monitoring algorithm is prudent. A near breakdown occurs if the smallest singular value of $P_{j+1}^TQ_{j+1}$ is less than the breakdown threshold, but this is not the case in this example.

Example 3. As mentioned before, when we know the multiplicity of the eigenvalues in advance, we should choose the appropriate blocksize, otherwise the adaptive scheme presented in section 3 can dynamically adjust the blocksize to accelerate convergence. This smooths the convergence behavior to clusters of eigenvalues. For example, we apply the ABLE method with initial blocksize 1 to the 656 × 656 Chuck matrix. At iteration 24, the double eigenvalue is detected and the blocksize is doubled.

Example 4. Exact breakdowns are rare but near breakdowns are not. In general, we can successfully cure the near breakdowns. For example, when the ABLE method is applied to the 882 × 882 Quebec Hydro matrix from the application of numerical

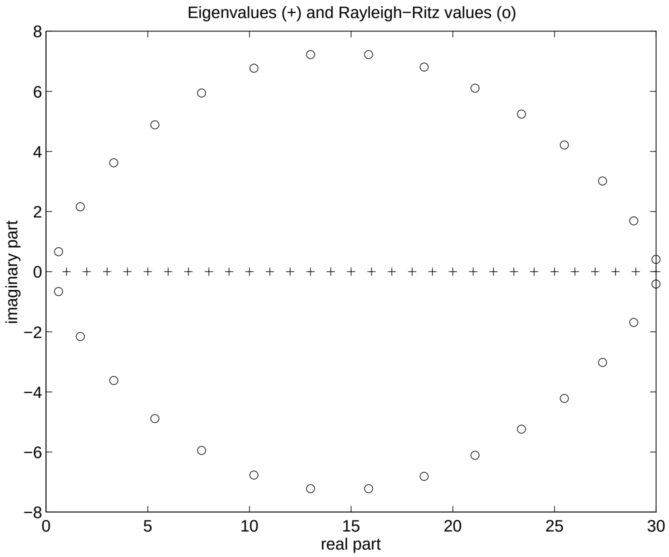


FIG. 7.2. *Spectra (+) and pseudospectra (○) of* 30 *by* 30 *Wilkinson bidiagonal matrix.*

methods for power systems simulations, four near breakdowns are cured. At iteration 37, the four leading eigenvalues are converged.

In the further investigation of this example, we found that the breakdowns are solely caused by the bad balancing of the entries of the matrix A. If we balance the matrix first (say, using the balance function available in Matlab), then the breakdown does not occur for the balanced matrix. The balancing of a large sparse matrix is a subject of further study.

Example 5. One of the attractive features of the ABLE method is that condition numbers of the approximate eigenvalues can be readily computed at the end of the ABLE method. This makes it possible to detect ill-posed eigenvalue problems. Our example is the 30 by 30 Wilkinson bidiagonal matrix [52, p. 90],

$$A = \begin{bmatrix} 30 & 30 & & & \\ & 29 & 30 & & \\ & & \ddots & \ddots & \\ & & & 2 & 30 \\ & & & & 1 \end{bmatrix}.$$

In the ABLE method with blocksize 1, all the residual errors after 30 iterations indicate convergence but the Rayleigh–Ritz values do not approximate exact eigenvalues; see Figure 7.2. This is understandable since all corresponding condition numbers $\mathrm{cond}(\theta_i)$ are of the order 10^{11} to 10^{13}. The eigenvalue problem for the Wilkinson matrix is ill-posed and the "converged" Rayleigh–Ritz values are pseudospectra.

Example 6. In this example, we apply the spectral transformation ABLE method to a generalized eigenvalue problem

$$Kx = \lambda Mx \tag{7.1}$$

arising from MHD [27, 11], where K is non-Hermitian and M is Hermitian positive definite. The interesting part of the spectrum in MHD problems is not the outer part of the spectrum but an internal branch, known as the Alfvén spectrum. We need to use a spectral transformation technique to transfer the interesting spectrum to the outer part. In section 6, we have outlined a general approach. Now, we show the numerical results of this general approach for the MHD test problem. Both K and M are 416 by 416 block tridiagonal matrices with 16 by 16 blocks. To two significant digits, there holds

$$\|K\|_1 = 3100 \quad \text{and} \quad \|M\|_1 = 2.50,$$

but the estimated condition number of M is 5.05×10^9; M is quite ill conditioned. The computational task is to calculate the eigenvalues close to the shift $\sigma = -0.3 + 0.65i$ [8].

We ran the unblocked spectral transformation ABLE method. After only 30 iterations, 10 Rayleigh–Ritz values are converged; their accuracy ranges from 10^{-8} to 10^{-12}, compared with the eigenvalues computed by the QZ algorithm. The following table lists the 10 converged Rayleigh–Ritz values θ_i and the corresponding left and right residual norms, where

$$\text{Res-L}_i = \frac{\|y_i^H K - \theta_i y_i^H M\|_2}{\max(\|K\|_1, \|M\|_1)}, \qquad \text{Res-R}_i = \frac{\|K x_i - \theta_i M x_i\|_2}{\max(\|K\|_1, \|M\|_1)},$$

and (y_i^H, x_i) are the normalized approximate left and right eigenvectors of (K, M) (i.e., $\|y_i^H\|_2 = \|x_i\|_2 = 1$):

i	θ_i	Res-L$_i$	Res-R$_i$
1	$-2.940037576164888e-01 + 5.871546479737660e-01i$	$3.82e-12$	$6.59e-11$
2	$-2.381814888866186e-01 + 5.914958688660595e-01i$	$2.66e-11$	$4.46e-11$
3	$-3.465530921874517e-01 + 5.468970786348115e-01i$	$1.23e-11$	$2.76e-10$
4	$-3.780991425908282e-01 + 5.071655448857557e-01i$	$6.18e-11$	$3.98e-10$
5	$-2.410301845692590e-01 + 5.238090347100917e-01i$	$9.81e-11$	$4.32e-10$
6	$-1.989292783177918e-01 + 5.900118523050361e-01i$	$5.34e-11$	$8.55e-11$
7	$-2.045328538082208e-01 + 5.678048139549289e-01i$	$5.97e-11$	$1.12e-10$
8	$-3.092857309948118e-01 + 4.687528684165645e-01i$	$5.23e-09$	$2.59e-08$
9	$-1.749780170739634e-01 + 5.920044440850396e-01i$	$5.62e-10$	$9.58e-10$
10	$-1.573456542107287e-01 + 5.976613227972810e-01i$	$5.98e-09$	$9.63e-09$

In addition, six other Rayleigh–Ritz values range in accuracy from 10^{-5} to 10^{-7}. Figure 7.3 shows Alfvén spectrum computed by the QZ algorithm (+) and the Rayleigh–Ritz values (○) computed by the spectral transformation ABLE method.

Three corrections to maintain semibiorthogonality were taken at iterations 13, 20, and 26. The convergence history of the Rayleigh–Ritz values are shown in the following table, where j is the Lanczos iteration and k is the number of converged Rayleigh–Ritz values at the jth iteration:

j	≤ 14	15–18	19	20–22	23–24	25–26	27–28	29	30
k	0	1	2	3	4	7	8	9	10

Moreover, at Lanczos iteration 45, the entire Alfvén branch of spectra of the MHD test problem are revealed: 20 Rayleigh–Ritz values converged, and 12 other Rayleigh–Ritz values range in accuracy from 10^{-7} up to 10^{-5}. No copies of eigenvalues are observed.

Acknowledgments. The authors would like to thank the referees for their valuable comments on the manuscript.

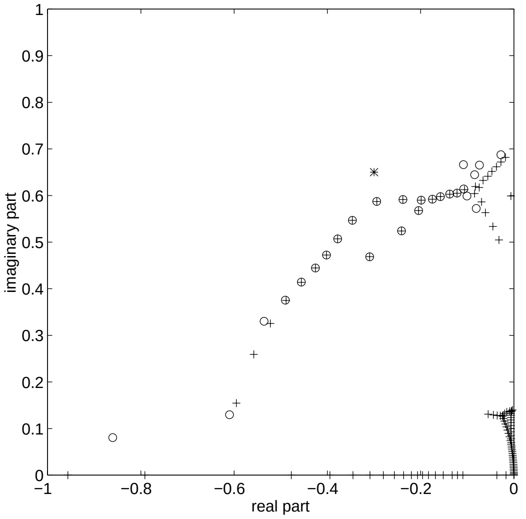


FIG. 7.3. *The Alfvén spectra of the MHD test problem. "+" denotes the eigenvalues computed by the QZ algorithm. "○" are the Rayleigh–Ritz values computed by* 30 *steps of the spectral transformation ABLE method. "∗" is the shift* $\sigma = -0.3 + 0.65i$.

REFERENCES

[1] W. E. ARNOLDI, *The principle of minimized iteration in the solution of the matrix eigenproblem*, Quart. Appl. Math., 9 (1951), pp. 17–29.

[2] Z. BAI, *Error analysis of the Lanczos algorithm for the nonsymmetric eigenvalue problem*, Math. Comp., 62 (1994), pp. 209–226.

[3] Z. BAI, *A spectral transformation block nonsymmetric Lanczos algorithm for solving sparse non-Hermitian eigenproblems*, in Proc. Fifth SIAM Conference on Applied Linear Algebra, J. G. Lewis, ed., SIAM, Philadelphia, PA, 1994, pp. 307–311.

[4] Z. BAI, D. DAY, D. DEMMEL, AND J. DONGARRA, *A Test Matrix Collection for Non-Hermitian Eigenvalue Problems*, available online from http://math.nist.gov/MatrixMarket.

[5] Z. BAI AND G. W. STEWART, *SRRIT: A Fortran subroutine to calculate the dominant invariant subspace of a nonsymmetric matrix*, ACM Trans. Math. Software, 23 (1997), pp. 494–513.

[6] D. BOLEY, S. ELHAY, G. H. GOLUB, AND M. H. GUTKNECHT, *Nonsymmetric Lanczos and Finding Orthogonal Polynomials Associated with Indefinite Weights*, Numerical Analysis report NA-90-09, Stanford University, Palo Alto, CA, 1990.

[7] D. BOLEY AND G. GOLUB, *The nonsymmetric Lanczos algorithm and controllability*, Systems Control Lett., 16 (1991), pp. 97–105.

[8] J. G. L. BOOTEN, H. A. VAN DER VORST, P. M. MEIJER, AND H. J. J. TE RIELE, *A Preconditioned Jacobi-Davidson Method for Solving Large Generalized Eigenvalue Problems*, Technical report NM-R9414, Dept. of Numerical Math, CWI, Amsterdam, the Netherlands, 1994.

[9] F. CHATELIN, *Eigenvalues of Matrices*, John Wiley, Chichester, England, 1993.

[10] J. CULLUM AND W. E. DONATH, *A block Lanczos algorithm for computing the q algebraically largest eigenvalues and a corresponding eigenspace of large sparse real symmetric matrices*, in Proc. 1974 IEEE Conference on Decision and Control, Phoenix, AZ, 1974, pp. 505–509.

[11] J. CULLUM, W. KERNER, AND R. WILLOUGHBY, *A generalized nonsymmetric Lanczos procedure*, Comput. Phys. Comm., 53 (1989), pp. 19–48.

[12] J. CULLUM AND R. WILLOUGHBY, *A practical procedure for computing eigenvalues of large sparse nonsymmetric matrices*, in Large Scale Eigenvalue Problems, J. Cullum and R. Willoughby, eds., North-Holland, Amsterdam, 1986.

[13] E. R. DAVIDSON, *The iteration calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices*, Comput. Phys., 17 (1975), pp. 87–94.

[14] D. Day, *Semi-Duality in the Two-Sided Lanczos Algorithm*, Ph.D. thesis, University of California, Berkeley, CA, 1993.

[15] I. Duff and J. Scott, *Computing selected eigenvalues of sparse nonsymmetric matrices using subspace iteration*, ACM Trans. Math. Software, 19 (1993), pp. 137–159.

[16] T. Ericsson and A. Ruhe, *The spectral transformation Lanczos method for the numerical solution of large sparse generalized symmetric eigenvalue problem*, Math. Comp., 35 (1980), pp. 1251–1268.

[17] R. W. Freund, M. H. Gutknecht, and N. M. Nachtigal, *An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices*, SIAM J. Sci. Comput., 14 (1993), pp. 137–158.

[18] R. W. Freund, N. M. Nachtigal, and J. C. Reeb, *QMRPACK User's Guide*, Technical report ORNL/TM-12807, Oak Ridge National Laboratory, Oak Ridge, TN, 1994.

[19] G. Golub and R. Underwood, *The block Lanczos method for computing eigenvalues*, in Mathematical Software III, J. Rice, ed., Academic Press, New York, 1977, pp. 364–377.

[20] G. Golub and C. Van Loan, *Matrix Computations*, 2nd ed., Johns Hopkins University Press, Baltimore, MD, 1989.

[21] W. B. Gragg, *Matrix interpretations and applications of the continued fraction algorithm*, Rocky Mountain J. Math., 5 (1974), pp. 213–225.

[22] R. Grimes, J. Lewis, and H. Simon, *A shifted block Lanczos algorithm for solving sparse symmetric generalized eigenproblems*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 228–272.

[23] M. H. Gutknecht, *A completed theory of the unsymmetric Lanczos process and related algorithms, Parts* I *and* II, SIAM J. Matrix Anal. Appl., Part I, 13 (1992), pp. 594–639, Part II, 15 (1994), pp. 15–58.

[24] Z. Jia, *Generalized block Lanczos methods for large unsymmetric eigenproblems*, Numer. Math., 80 (1998), pp. 239–266.

[25] W. Kahan, B. N. Parlett, and E. Jiang, *Residual bounds on approximate eigensystems of nonnormal matrices*, SIAM J. Numer. Anal., 19 (1982), pp. 470–484.

[26] T. Kato, *Perturbation Theory for Linear Operators*, 2nd ed., Springer-Verlag, Berlin, 1980.

[27] W. Kerner, *Large-scale complex eigenvalue problems*, J. Comput. Phys., 85 (1989), pp. 1–85.

[28] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, J. Res. Natl. Bur. Stand, 45 (1950), pp. 225–280.

[29] R. Lehoucq, *Analysis and Implementation of an Implicitly Restarted Arnoldi Iterations*, Ph.D. thesis, Rice University, Houston, Texas, 1995.

[30] R. Lehoucq and J. A. Scott, *An Evaluation of Software for Computing Eigenvalues of Sparse Nonsymmetric Matrices*, Preprint MCS-P547-1195, Argonne National Laboratory, Argonne, IL, 1996.

[31] R. Lehoucq, D. Sorensen, and C. Yang, *ARPACK Users' Guide: Solution of Large Scale Eigenvalue Problems by Implicitly Restarted Arnoldi Methods*, SIAM, Phildelphia, PA, 1998.

[32] K. Meerbergen and A. Spence, *Implicitly Restarted Arnoldi with Purification for the Shift-Invert Transformation*, report tw 225, Dept. of Comput. Sci., Katholieke Universiteit Leuven, Belgium, 1995.

[33] C. Paige, *The Computation of Eigenvalues and Eigenvectors of Very Large Sparse Matrices*, Ph.D. thesis, London University, London, UK, 1971.

[34] B. Parlett, *The Rayleigh quotient algorithm iteration and some generalizations for nonnormal matrices*, Math. Comp., 28 (1974), pp. 679–693.

[35] B. Parlett, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, NJ, 1980.

[36] B. Parlett, *Reduction to tridiagonal form and minimal realizations*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 567–593.

[37] B. Parlett and Y. Saad, *Complex shift and invert strategies for real matrices*, Linear Algebra Appl., 88/89 (1987), pp. 575–595.

[38] B. N. Parlett, D. R. Taylor, and Z. A. Liu, *A look-ahead Lanczos algorithm for unsymmetric matrices*, Math. Comp., 44 (1985), pp. 105–124.

[39] A. Ruhe, *Implementation aspects of band Lanczos algorithms for computation of eigenvalues of large sparse symmetric matrices*, Math. Comp., 33 (1979), pp. 680–687.

[40] A. Ruhe, *Rational Krylov, a Practical Algorithm for Large Sparse Nonsymmetric Matrix Pencils*, Computer Science Division UCB/CSD-95-871, University of California, Berkeley, CA, 1995.

[41] Y. Saad, *On the rates of convergence of the Lanczos and block Lanczos methods*, SIAM J. Numer. Anal., 17 (1980), pp. 687–706.

[42] Y. Saad, *Variations on Arnoldi's method for computing eigenelements of large unsymmetric*

matrices, Linear Algebra Appl., 34 (1980), pp. 269–295.

[43] Y. Saad, *Numerical Methods for Large Eigenvalue Problems*, Halsted Press (division of John Wiley), New York, 1992.

[44] M. Sadkane, *Block-Arnoldi and Davidson methods for unsymmetric large eigenvalue problems*, Numer. Math., 64 (1993), pp. 195–211.

[45] M. Sadkane, *A block Arnoldi-Chebyshev method for computing the leading eigenpairs of large sparse unsymmetric matrices*, Numer. Math., 64 (1993), pp. 181–193.

[46] H. Simon, *The Lanczos algorithm with partial reorthogonalization*, Math. Comp., 42 (1984), pp. 115–142.

[47] G. L. G. Sleijpen and H. A. van der Vorst, *A Jacobi-Davidson iteration method for linear eigenvalue problems*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 401–425.

[48] D. Sorensen, *Implicit application of polynomial filters in a k-step Arnoldi method*, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 357–385.

[49] G. W. Stewart, *Error and perturbation bounds for subspaces associated with certain eigenvalue problems*, SIAM Rev., 15 (1973), pp. 727–764.

[50] W. J. Stewart and A. Jennings, *Algorithm* 570 *LOPSI: A simultaneous iteration algorithm for real matrices*, ACM Trans. Math. Software, 7 (1981), pp. 230–232.

[51] L. N. Trefethen, *Pseudospectra of matrices*, in Numerical Analysis 1991, Dundee, Scotland, Longman Sci. Tech., Harlow, 1992.

[52] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford University Press, Oxford, UK, 1965.

[53] Q. Ye, *A breakdown-free variation of the nonsymmetric Lanczos algorithms*, Math. Comp., 62 (1994), pp. 179–207.

SIAM J. Matrix Anal. Appl.
Vol. 20, No. 4, pp. 1083–1098

THE *BR* EIGENVALUE ALGORITHM*

G. A. GEIST[†], G. W. HOWELL[‡], AND D. S. WATKINS[§]

Abstract. The *BR* algorithm, a new method for calculating the eigenvalues of an upper Hessenberg matrix, is introduced. It is a bulge-chasing algorithm like the *QR* algorithm, but, unlike the *QR* algorithm, it is well adapted to computing the eigenvalues of the narrow-band, nearly tridiagonal matrices generated by the look-ahead Lanczos process. This paper describes the *BR* algorithm and gives numerical evidence that it works well in conjunction with the Lanczos process. On the biggest problems run so far, the *BR* algorithm beats the *QR* algorithm by a factor of 30–60 in computing time and a factor of over 100 in matrix storage space.

Key words. eigenvalue computation, *QR* algorithm, unsymmetric Lanczos process

AMS subject classifications. 65F15, 15A18

PII. S0895479897317077

1. Introduction.

One of the most economical tools for probing the spectrum of a large, sparse, nonsymmetric matrix is the look-ahead Lanczos algorithm [4], [7], [8]. A subproblem that arises at least twice and perhaps repeatedly in a look-ahead Lanczos run is that of calculating the eigenvalues of a nearly tridiagonal auxiliary matrix that is generated by the algorithm. After m Lanczos steps, the auxiliary matrix is $m \times m$. If m is small, one can calculate the eigenvalues cheaply by the standard method, the *QR* algorithm. However, as m grows large, this step can become a bottleneck, since the cost of applying the *QR* algorithm grows approximately as m^3. The *QR* algorithm does not make use of all of the structure of the auxiliary matrix; it exploits and preserves the upper Hessenberg form, but it neither exploits nor preserves the many zeros above the main diagonal. It is therefore natural to look for an algorithm that does a better job of preserving the structure.

In this paper we introduce the *BR* algorithm, which attempts to exploit and preserve all of the structure of the auxiliary matrix. It turns out that with rare exceptions (in our experience) it does succeed in preserving enough of the structure to run significantly faster than the *QR* algorithm. Since it stores the matrix in a banded data structure, it also requires significantly less storage space than the *QR* algorithm. In our best runs we have been able to cut computer time by a factor of 60 and matrix storage space by a factor of more than 100.

The *BR* algorithm is a member of the family of *GR* algorithms [18], [16]. It is an implicit *GR* algorithm, which makes it a bulge-chasing algorithm [17]. Bulge-chasing algorithms operate on matrices that have been reduced to upper Hessenberg form (or some comparable form). Each iteration consists of an initial transformation that

[*] Received by the editors February 18, 1997; accepted for publication (in revised form) by A. Greenbaum March 23, 1998; published electronically July 9, 1999.

http://www.siam.org/journals/simax/20-4/31707.html

[†] Mathematical Sciences Section, Oak Ridge National Laboratory, Box 2008, Bldg. 6012, Oak Ridge, TN 37831-6367 (gst@ornl.gov).

[‡] Department of Applied Mathematics, Florida Institute of Technology, 150 W. University Boulevard, Melbourne, FL 32901 (howell@zach.fit.edu).

[§] Department of Pure and Applied Mathematics, Washington State University, Pullman, WA 99164-3113 (watkins@wsu.edu). Mailing address: 6835 24th Avenue NE, Seattle, WA 98115-7037.
The work of this author was supported by the National Science Foundation under grant DMS-9403569.

disturbs the upper Hessenberg form, followed by a sequence of transformations that restore the upper Hessenberg form. Thus algorithms that transform a matrix to upper Hessenberg form lie at the center of this subject.

In this paper we restrict our attention to real matrices. Similar algorithms can be developed for complex matrices.

2. Algorithms that transform a matrix to upper Hessenberg form. It is well known [19], [6], [15] that every $m \times m$ matrix can be transformed to upper Hessenberg form by an orthogonal similarity transformation in $O(m^3)$ operations. *Upper Hessenberg* means that $a_{ij} = 0$ if $i > j + 1$. The transformation is effected in $m-2$ major steps as follows: In the first step a reflector (Householder transformation) acting on rows 2 through m transforms entries $(3,1), \ldots, (m,1)$ to zero. When the transformation is applied on the right (i.e., to columns 2 through m), the first column is untouched, so the zeros are preserved. In the second step a reflector acting on rows 3 through m creates zeros in entries $(4,2), \ldots, (m,2)$, and so on. In general the kth step produces the desired zeros in the kth column. A venerable implementation of this procedure is the code ORTHES from EISPACK [1]. A more modern implementation, which applies the reflectors in blocks, is the code DGEHRD from LAPACK [3].

If one does not insist upon orthogonal transformations, one can use other means to create the zeros. For example, the EISPACK algorithm ELMHES [1] uses Gaussian elimination transformations with pivoting. Thus, on the first step, the largest (in magnitude) entry in positions $(2,1), \ldots, (m,1)$ is identified. If it is not already in position $(2,1)$, it is moved there by a row interchange. (The similarity transformation is completed by performing the corresponding column interchange.) Then appropriate multiples of the second row are subtracted from rows 3 through m to create zeros in positions $(3,1), \ldots, (m,1)$. Multiples of columns 3 through m are added to column 2 to complete the similarity transformation.

If one uses Gaussian elimination transformations, one can often create additional zeros above the main diagonal. In fact it has long been known [19] that almost any matrix can be transformed to tridiagonal form. The first step of such a transformation can be accomplished as follows. Once the zeros have been created in the first column, if the entry in the $(1,2)$ position is nonzero, it can be used as a pivot for column operations that create zeros in the first row. That is, the appropriate multiples of the second column are subtracted from columns 3 through m to produce zeros in positions $(1,3), \ldots, (1,m)$. The similarity tranformation is completed by adding multiples of rows 3 through m to row 2. The important point is that these operations do not disturb the previously created zeros in column 1. However, we do not have the luxury of pivoting. A single row and column interchange at the beginning of each step determines the pivots for both the row elimination and the column elimination. Furthermore, if the $(1,2)$ pivot entry happens to be zero, the row elimination will not be possible. More important, if the $(1,2)$ entry is very close to zero, the eliminations will require extremely large multipliers, and the transformation will be unstable.

There have been numerous attempts to stabilize the reduction to tridiagonal form, none of which was entirely successful. Recently Howell, Geist, and Diaa [10] and Howell, Geist, and Rowan [11] developed a compromise strategy that reduces the matrix to upper Hessenberg form and also introduces as many zeros above the diagonal as possible. The resulting matrix is somewhere between tridiagonal and full upper Hessenberg. Since it is typically a banded upper Hessenberg matrix, the algorithm is called BHESS.

The BHESS strategy is roughly as follows. For details see [10], [11]. On the kth step BHESS does a column elimination to make zeros in positions $(k+2, k), \ldots, (m, k)$. It also attempts to create zeros in the kth row or some previous row that was not eliminated on an earlier step. Thus it will attempt to create zeros in positions $(j, k+2), \ldots, (j, m)$, where j designates a row such that $j \leq k$ and the $(j, k+2), \ldots, (j, m)$ entries are not all zero already. The row elimination will be carried out if the multipliers that would be used for the column and row eliminations are not too big on average. The precise meaning of "too big" depends on a user-specified error tolerance τ. If more than one row qualifies for elimination, the qualifier with the smallest j is eliminated. A row and column interchange that minimizes the maximum multiplier for row and column eliminations together is performed. This is a compromise. Once the row and column to be eliminated have been determined, the product of row and column pivots is invariant. Thus pivoting to maximize the column pivot will have the effect of minimizing the row pivot and vice versa. Rather than optimizing one or the other, BHESS chooses pivots that do as well as possible for rows and columns together.

If no row qualifies for elimination, only the column elimination is done. The maximal pivot is chosen.

3. Bulge-chasing algorithms. This is a brief review. For details see [18] and [17]. Given an upper Hessenberg matrix A, an iteration of a double-shift GR algorithm can be performed on A as follows. First two shifts σ_1 and σ_2 are chosen. Most commonly these are taken to be the eigenvalues of the 2×2 submatrix at the lower right-hand corner of A. Then the vector $x = (A - \sigma_1)(A - \sigma_2)e_1$ is formed. Since A is upper Hessenberg, only the first three components of x are nonzero. If either $\sigma_2 = \overline{\sigma}_1$ or σ_1 and σ_2 are real, then x will be real. A nonsingular matrix G_0 is constructed so that $G_0 e_1$ is proportional to x. For example, G_0 can be a reflector or a Gaussian elimination transformation, with or without pivoting. In any event, G_0 should have the form $G_0 = \operatorname{diag}\{M, I_{n-3}\}$, where M is 3×3. The similarity transformation $A \leftarrow G_0^{-1} A G_0$ disturbs the upper Hessenberg form; the new matrix has a bulge extending to position $(4, 1)$, as illustrated here in the 7×7 case:

$$\begin{bmatrix} * & * & * & * & * & * & * \\ * & * & * & * & * & * & * \\ * & * & * & * & * & * & * \\ * & * & * & * & * & * & * \\ & & & * & * & * & * \\ & & & & * & * & * \\ & & & & & * & * \end{bmatrix}.$$

The rest of the iteration consists of returning the matrix to upper Hessenberg form by clearing out the columns one at a time, as in the algorithms discussed in the previous section. Each step removes a row from the left-hand side of the bulge and adds a new row to the bottom. Thus the bulge is "chased" from the upper left-hand corner downward along the subdiagonal until it disappears off the bottom of the matrix. This completes an iteration.

It is shown in [17] that this procedure amounts to an iteration of a GR algorithm, regardless of which method for reduction to upper Hessenberg form is used. Thus the convergence theory of [18] is applicable. Under mild hypotheses repeated bulge-chasing iterations will cause the matrix to converge to block upper triangular form, revealing the eigenvalues.

If all of the transformations in the bulge chase are orthogonal (e.g., reflectors, as in ORTHES), each iteration amounts to a step of the QR algorithm. If Gauss transforms (with or without pivoting) are used, the iteration amounts to a step of the LR algorithm (with or without pivoting).

4. The *BR* algorithm. One can equally well use BHESS to chase the bulge. This is the primary idea behind the BR algorithm. We can expect (or at least hope) that iterative use of BHESS will successively narrow the bandwidth of the matrix as it makes progress toward finding the eigenvalues. If the matrix has a narrow bandwidth to begin with, as is the case in the look-ahead Lanczos process, repeated application of BHESS can be expected to keep it narrow.

It has to be noted, however, that when BHESS is used to chase the bulge in a narrow-band matrix, there are two mutually antagonistic forces at work. We have already seen that the algorithm is constantly trying to create new zeros above the main diagonal. This tends to reduce the bandwidth. On the other hand, the algorithm does a certain amount of pivoting for stability. These row and column interchanges tend to widen the band. It is not clear a priori which of these forces will prevail in the long run.

Our first experiments were disappointing. We used BHESS to perform an initial reduction to narrow-band Hessenberg form. We then applied BHESS as a bulge-chasing algorithm on the narrow-band matrix. It turned out that over the course of many iterations the bandwidth tended to grow until the matrix became full upper Hessenberg. On inspecting the intermediate matrices we found that some extremely large numbers were building up in the part of the matrix above the main diagonal. These made row eliminations difficult. A mechanism for preventing this growth was needed.

After some experimentation we found that if a balancing operation is performed before each iteration, the undesired element growth is prevented. Our balancing operation is described as follows. For $i = 1, 2, \ldots, n$ multiply the ith row of the matrix by d_i^{-1} and the ith column by d_i, where d_i is chosen so that the ith row and column will have equal 1-norm after rescaling. The operations are performed in serial. That is, the scaling factor d_i is determined after the first $i-1$ rows and columns have been rescaled by $d_1, \ldots, d_{i-1}$. The entire scaling operation amounts to a similarity transformation $A \leftarrow D^{-1}AD$, where $D = \operatorname{diag}\{d_1, \ldots, d_n\}$.

An important difference between our scaling operation and the classical balancing routine [1], [3] is that our routine makes only one sweep through the matrix, whereas the classical routine iterates until each row's norm is nearly equal to that of the corresponding column. Our routine makes no such guarantee; the equality between row and column norms that is established for the first rows and columns will be upset by the rescaling of the later rows and columns. Nevertheless, we have found that this single sweep does a good job of shifting excess weight from the upper part of the matrix to the lower part.

The inclusion of balancing improved performance dramatically. A more refined strategy, which we use in our current code, is to balance only a small portion of the matrix at a time. Thus we balance the first few rows (20 or so), then run the bulge through that part of the matrix, then we balance some more and run the bulge further, and so on.

Further improvements can be made by refining the elimination strategy. BHESS was developed for reducing a full matrix to a sparse form. The strategy it uses does not consider that the matrix may already have contained many zeros to begin with.

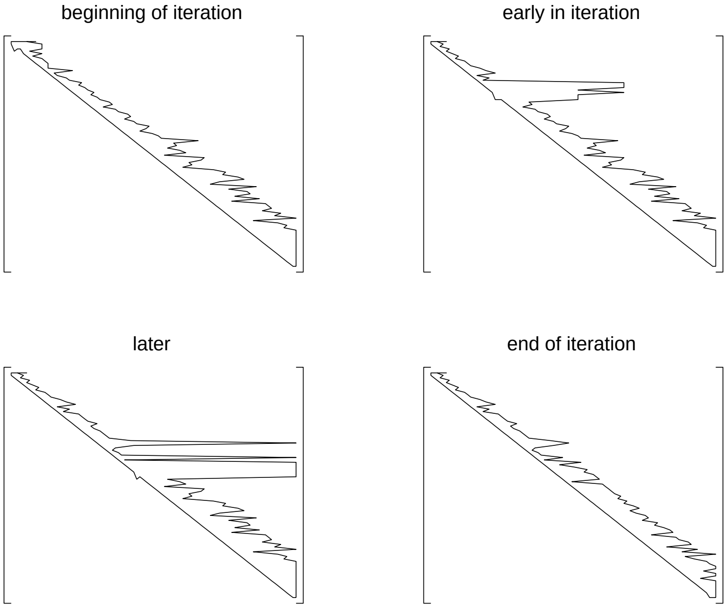


FIG. 4.1. *Spike in profile of narrow-band matrix during bulge chase by BHESS.*

Thus when BHESS is applied to a narrow-band matrix, an elimination in row i may cause the destruction of many previously existing zeros in other rows. For example, suppose that at step i the entry in position $(m, i+1)$ is used as a pivot to create zeros in positions $(m, i+2), \ldots, (m, i+b)$. Then multiples of column $i+1$ are subtracted from columns $i+2, \ldots, i+b$. These operations can result in destruction of zeros in any other row p for which the $(p, i+1)$ entry is nonzero. More important, multiples of rows $i+2, \ldots, i+b$ must then be added to row $i+1$ to complete the similarity transformation. If the nonzero part of any of these rows protrudes far past the nonzero part of row $i+1$, the bandwidth in row $i+1$ will be widened considerably by these operations. This effect compounds itself from one step to the next and can result in the creation of a huge spike of nonzeros in the profile of what had been a narrow-band matrix, as shown in Figure 4.1. By the end of the bulge chase, the narrow-band form has been restored, but the cost of restoration is unacceptable. Spikes of this type occur frequently at low tolerances (e.g., $\tau = 3$) and also occasionally at larger tolerances (e.g., $\tau = 15$).

In order to minimize the destruction of previously created zeros during the bulge chase, we made the following modifications of the elimination strategy. We keep track of the position of the last nonzero entry in each row. That is, for each i we keep a record of α_i such that the entry in position (i, α_i) is the last nonzero in row i. At step i we find the first row $m \le i$ for which the entries in positions $(m, i+1), \ldots, (m, n)$ are not all zero already. We consider performing an elimination *only* in that row. If an elimination in row m is permissible according to the multiplier tolerance test, then it will be perfomed only if it does not result in too great an increase in α_{i+1}.

If it is decided not to eliminate a row, the column elimination is done without pivoting if all of the multipliers are smaller than the tolerance τ. Otherwise the maximal pivot is chosen.

Our code also incorporates several other features that are standard in QR codes,

Table 5.1
Performance of BR algorithm for varying τ.

	Time (sec)			Maximum
	Average	Std. dev.	Maximum	error
QR algorithm	6.09	0.21	6.45	1.3×10^{-11}
BR with $\tau = 1$	16.16	2.18	19.36	1.0×10^{-9}
BR with $\tau = 3$	2.78	0.48	3.78	3.5×10^{-9}
BR with $\tau = 10$	1.24	0.44	2.45	2.3×10^{-7}
BR with $\tau = 30$	0.91	0.07	0.99	6.0×10^{-5}
BR with $\tau = 100$	0.85	0.06	0.94	1.4×10^{-3}
BR with $\tau = 300$	0.83	0.05	0.90	6.1×10^{-1}

e.g., exceptional shifts when the iterations seem to be stalled and exploitation of pairs of consecutive small subdiagonal entries.

5. Performance of the *BR* algorithm.

5.1. Experiments with random matrices. In this paper we are mainly advocating the use of the BR algorithm on the nearly tridiagonal matrices produced by the look-ahead Lanczos process. Nevertheless, we shall begin by reporting on some experiments with full matrices. The numbers reported in this subsection were obtained using an early version of the algorithm that does a complete rebalance between bulge chases and no rebalancing during a bulge chase.

We constructed random matrices with known eigenvalues by the following procedure. First a random block upper triangular matrix $T + N$ is constructed. T is the block diagonal part. It has 1×1 and 2×2 blocks; its eigenvalues are obvious, and these are the eigenvalues of the matrix. The eigenvalues and the entries of N are normally distributed with mean zero and standard deviations σ_T and σ_N, respectively. The ratio σ_T/σ_N is adjusted to control the ratio $\delta = \| N \|_F / \| T \|_F$, which is a measure of departure from normality. We also control the number of complex eigenvalues. We then produce a matrix A with the same eigenvalues by applying a random orthogonal similarity transformation (uniform with respect to Haar measure on the orthogonal group) by the method of Stewart [14].

We generated numerous matrices of this type. Table 5.1 shows some results for 500×500 matrices with 50 real and 450 complex eigenvalues and a departure from normality $\delta \approx 1$. This ratio produces matrices whose eigenvalues are well conditioned. Each matrix was reduced to upper Hessenberg form by the LAPACK code DGEHRD, and its eigenvalues were calculated by the LAPACK multishift QR code DHSEQR. This is (on average) the fastest QR code we know of for modern cache-based RISC workstations, such as the DEC AlphaStation 500/333 that was used for these experiments. Each matrix was also reduced to banded upper Hessenberg form by BHESS, and its eigenvalues were calculated by BR with various choices of the user-specified tolerance τ. The times are the average, standard deviation, and maximum over 10 matrices. Each error is the maximum error over all 10 matrices.

We observe that for all choices of τ, BR is less accurate than QR. If the greatest possible precision is needed, BR is not the algorithm of choice. We see that as τ is increased from 1 to 300, there is a tradeoff between reduced computing time and increased error. For $\tau \geq 10$ the times look quite good, but at $\tau = 1$ BR is unacceptably slow. In the latter case the tolerance is too small to allow the preservation of a narrow-band form. The matrix eventually fills out to full upper Hessenberg form after having spent a lot of time creating and trying to defend zeros above the diagonal. At $\tau = 3$

Table 5.2
Accuracy as a function of departure from normality.

Departure from normality	Maximum error QR	BR $(\tau = 15)$
0.0	2.3×10^{-14}	1.9×10^{-9}
0.5	2.7×10^{-14}	4.7×10^{-10}
1.0	1.6×10^{-13}	6.6×10^{-9}
1.5	2.4×10^{-11}	1.6×10^{-5}
2.0	1.1×10^{-8}	9.1×10^{-4}
2.5	8.4×10^{-7}	6.1×10^{-1}

the algorithm does much better at creating and preserving zeros, but the production of new nonzeros is still significant. (More evidence of this will be given in Table 5.3.) The intermediate values $\tau = 10$ and $\tau = 30$ compute reasonably accurate eigenvalues very quickly.

Since the BR algorithm uses Gaussian elimination and may allow multipliers that are much greater than one, depending on the tolerance, there is no guarantee of backward stability. Indeed, it is not hard to make the algorithm fail. Table 5.2 shows the maximum error accrued in calculating the eigenvalues of a 300×300 matrix with 270 complex eigenvalues, varying the departure from normality $\delta = \|N\|_F / \|T\|_F$ from 0 to 2.5.

For both the QR and the BR algorithm the maximum error grows as the departure from normality grows. Since the QR algorithm is backward stable, the growth in error can be attributed almost entirely to increasing ill conditioning of the eigenvalues. The BR results also reflect the increasing ill conditioning, but they are consistently about five decimal places worse than the QR results. As a consequence, at $\delta = 2.5$ the QR algorithm is still delivering accurate eigenvalues, whereas BR is no longer resolving all eigenvalues well.

These results apply specifically to the class of randomly generated matrices described above. Experiments on other classes of matrices have shown that the departure from normality is generally a poor indicator of the accuracy of the BR algorithm.

Getting back to the question of computing times, we can learn more by considering matrices of various sizes, as in Table 5.3. The matrices used for these tests were randomly generated with independent entries normally distributed with mean zero and variance one. Thus the exact eigenvalues are not known. Matrices generated in this way have a departure from normality very close to 1, and their eigenvalues tend to be well conditioned. In every case we compared the eigenvalues generated by BR with those generated by QR. The maximum "error" ranged from 10^{-11} for small matrices and small tolerances to 10^{-5} for the largest matrices and tolerances.

The times in Table 5.3 are times to calculate eigenvalues of matrices that have been reduced to upper Hessenberg form by either DGEHRD or BHESS. We see that with $\tau \geq 9$ BR is very much faster than QR on large matrices. Let us consider the trends. We expect the computing time of QR to be $O(n^3)$, based on the reasoning that each iteration takes $O(n^2)$ work and at least one iteration will be required for each eigenvalue or pair of eigenvalues. This expectation can be checked numerically by making a log-log plot of the computing time as a function of matrix size n. Doing so, we find that the plot is nearly a straight line. If the slope is k, then the computing time is $O(n^k)$. In fact the slope of the least squares straight line fit to the QR data is 2.88, which is close to the expected slope 3. Now how do the BR times grow with n?

Table 5.3

Random matrices reduced to upper Hessenberg form.

	QR time	BR time					
n		$\tau = 3$	$\tau = 6$	$\tau = 9$	$\tau = 12$	$\tau = 15$	$\tau = 18$
107	.10	.07	.06	.06	.06	.06	.06
142	.21	.16	.10	.10	.10	.10	.11
190	.45	.31	.21	.18	.18	.18	.18
253	.98	.54	.37	.37	.34	.36	.33
337	2.13	1.53	.75	.62	.55	.53	.56
450	4.79	3.11	1.56	1.21	1.09	1.06	1.04
600	11.92	6.68	6.26	2.48	2.23	1.97	1.82
800	29.26	19.74	13.04	9.49	4.55	3.39	3.19
1067	79.04	74.07	21.76	17.85	17.49	9.20	7.36

Table 5.4

Observed computational complexity $O(n^k)$.

	QR exponent	BR exponents					
		$\tau = 3$	$\tau = 6$	$\tau = 9$	$\tau = 12$	$\tau = 15$	$\tau = 18$
k	2.88	2.91	2.69	2.48	2.34	2.12	2.03

If the bandwidth is very small ($O(1)$) and stays small throughout each iteration, then an iteration will require $O(n)$ work. Thus the total work for $O(n)$ iterations should be $O(n^2)$.

Taking least squares fits to the log-log plots for BR, we obtain the slopes given in Table 5.4. For $\tau = 15$ and $\tau = 18$ we obtain the desired results. The situation is not so good for $\tau = 6$ and $\tau = 3$, for which we get slopes of 2.7 and 2.9, respectively. Thus with $\tau = 3$ we are seeing almost $O(n^3)$ behavior. This shows that at these values of τ, the algorithm is not doing a good job of keeping the band narrow. These results suggest that successful use of BR will require use of a fairly liberal value of τ. From now on we will stick with $\tau = 15$.

The numbers in Table 5.3 are encouraging. Before moving on to the look-ahead Lanczos process, let us inject one more set of discouraging numbers. The tests reported in Table 5.3 used matrices generated by filling the entire array with random numbers, then reducing the matrix to upper Hessenberg form. If one instead builds matrices by simply filling the upper Hessenberg part with random numbers and skipping the reduction step, one gets very different results, as shown in Table 5.5.

These matrices have a high departure from normality (e.g., $\delta \approx 12$ when $n = 600$). Their eigenvalues are very badly conditioned, but they are fairly well conditioned with respect to the nearly upper Hessenberg perturbations that occur during the execution of bulge chasing algorithms. Thus both the QR and the BR algorithm are able to compute the eigenvalues quite accurately. Here the issue is not accuracy but computing time. While the QR times are not much different than they were in Table 5.3, the BR times with $\tau = 15$ are much worse. Indeed they are no better than the QR times. Matrices of this type have so much weight above the main diagonal that it is difficult to get them into a narrow-band form. As a consequence BR ends up doing as much work as QR.

5.2. Experiments with sparse matrices. We performed numerous experiments in which we used the BR algorithm to calculate the eigenvalues of the nearly tridiagonal matrices generated by the look-ahead Lanczos process. These are upper

Table 5.5
Random upper Hessenberg matrices.

n	QR time	BR time $\tau = 15$
253	1.03	0.77
337	2.33	2.05
450	5.21	5.22
600	12.00	12.02

Hessenberg, so we can apply BR directly to them; there is no need to preprocess them by BHESS. We modified the code DULAL from the package QMRPACK by Freund and Nachtigal [5]. In that code the eigenvalue computation is done by the QR algorithm in a standard array data structure. We switched to a banded storage scheme and replaced QR with BR. In many cases we ran both QR and BR for comparison purposes. DULAL uses the EISPACK code HQR, but we substituted the LAPACK code DHSEQR, which is usually faster than HQR on the DEC AlphaStation 500/333 and similar machines.

Another change we made was in the balancing strategy of BR. Instead of rebalancing the whole matrix once before each bulge chase, we rebalance small sections of the matrix during the bulge chase, as described earlier. That is, we balance the first few rows, then we run the bulge through that part of the matrix, then we balance some more and run the bulge further, and so on. This change reduces the growth of the bandwidth significantly. This is an important improvement, since the matrix now has to be kept within the confines of a banded data structure.

In the experiments reported below, the matrices produced by the look-ahead Lanczos process were always very nearly tridiagonal. If the look-ahead feature is not used at all, a tridiagonal matrix is formed. Each time the look-ahead is used, a small bulge on the upper side of the band is formed. In our experiments no more than four look-aheads were needed in any given run. The upper bandwidths of the resulting matrices never exceeded 3. A more typical upper bandwidth was 2, and in many cases it was 1 (tridiagonal), indicating that no look-ahead steps had been needed.

Convection-diffusion matrices. Our first examples are matrices obtained by discretizing a three-dimensional convection-diffusion operator

$$Lu = -u_{xx} - u_{yy} - u_{zz} + c(u_x + u_y + u_z)$$

on the domain $\Omega = (0, 1)^3$ with $u = 0$ on $\partial\Omega$. The standard second-order centered finite-difference approximations were used.

With a mesh size $h = \frac{1}{40}$ in each direction, we obtain a matrix of order $39^3 = 59319$. Each row has seven nonzero entries. We chose the convection coefficient $c = 8$ to get a Péclet number $\frac{ch}{2} = 0.1$. We ran m look-ahead Lanczos steps and calculated the eigenvalues of the $m \times m$ narrow-band matrix for various choices of m ranging from about 100 to 6000. In Figure 5.1 we display the time to generate the matrix (plus symbols), the time to calculate its eigenvalues by the QR algorithm (cross symbols), and the time to calculate the eigenvalues by the BR algorithm (circle symbols) with $\tau = 15$.

We see that BR is cheaper than QR for all m in the range that we studied, but for small matrices it does not matter which method we use. Both algorithms can calculate the eigenvalues in a small fraction of the time it takes to generate the

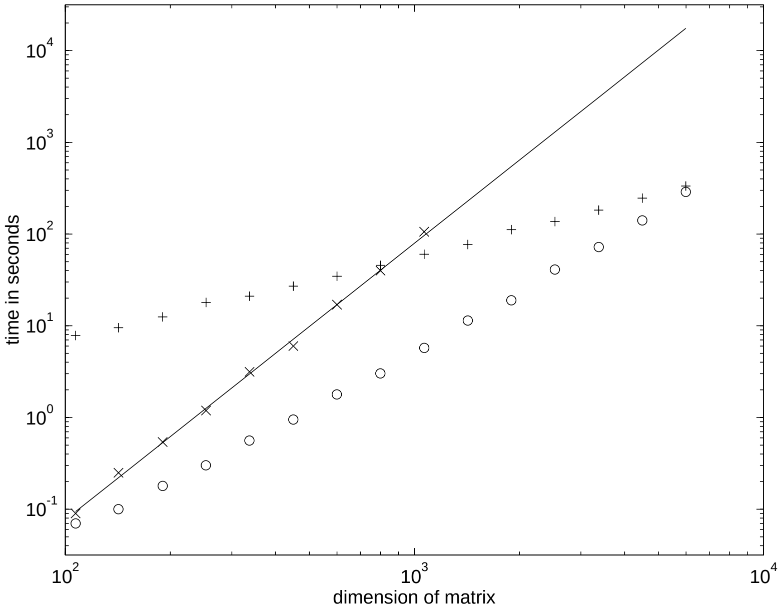


FIG. 5.1. *Time to generate $m \times m$ a narrow-band matrix (plus symbols) from convection-diffusion operator and calculate its eigenvalues by the QR algorithm (crosses) and by the BR algorithm with $\tau = 15$ (circles).*

matrix. As the matrix dimension increases, the time spent computing eigenvalues rapidly becomes significant, especially if the QR algorithm is used. The line that best fits the Lanczos times has slope 0.93, whereas the line that fits the QR times (the solid line in Figure 5.1) has slope 3.01. For values of m above 1100 we did not do the QR calculation, because it would have taken too long. The line that best fits the BR times has slope 2.09, so we can go to much larger matrices before the BR computation time becomes significant. At $m = 5993$, the largest run shown in Figure 5.1, the Lanczos algorithm took 5.6 minutes to generate the matrix, and the BR algorithm took 4.8 minutes to calculate the eigenvalues. If we had used the QR algorithm to calculate the eigenvalues, it would have taken about 4.8 *hours*, so the BR algorithm saves us a factor of 60 on that part of the calculation.

Storage space is also a consideration. Just to store the 5993×5993 matrix in the standard array format, which QR needs, would require 287 megabytes. On the run under consideration here we stored the matrix in a 5993×50 array, which allows enough room to store a band of 46 diagonals above the main diagonal. This array occupies 2.4 megabytes of memory. The maximum number of diagonals actually used in this computation was 37, so we could have gotten away with a bit less storage space.

For dimensions below 1100 we were able to compare the computed eigenvalues from the QR and BR calculations. We sorted the eigenvalues and paired them off. In no case did the relative difference between the QR and BR values exceed 5.5×10^{-6}. The matrices typically had a high departure from normality, ranging from 4.7 to 23.1, with typical values around 15.

On the matrices with dimensions above 1100 it was not possible to make the comparison, but we did do some spot checking. On each matrix we took a sample

Table 5.6
Convection-diffusion operators with various Péclet numbers.

Péclet number	QR time	BR time $\tau = 15$	Max. rel. "error"	Max. upper bandwidth (BR)
0.1	5.38	0.95	2.4×10^{-8}	8
0.2	6.54	0.90	5.2×10^{-8}	12
0.4	6.46	1.07	7.2×10^{-4}	14
0.6	4.39	***	***	**
1.5	5.35	***	***	**
2.0	5.55	***	***	**
3.0	4.78	0.85	4.2×10^{-6}	7
6.0	5.02	0.85	1.3×10^{-8}	7

of about $\sqrt{m}$ computed eigenvalues and refined them using the generalized Rayleigh quotient iteration code GIRI [13]. In no case did the relative difference between the original computed value and the refined value exceed 1.2×10^{-5}.

We have focused on how well BR calculates the eigenvalues of the $m \times m$ narrow-band matrix. Typically only a few of these will be good approximations to the eigenvalues of the large sparse matrix. The question of which ones are "good" is difficult and obviously important. We are ignoring it here, because our objective is simply to study how well the BR algorithm does its assigned task.

Harder Péclet numbers. One can make the BR algorithm fail on convection-diffusion problems by making the Péclet number closer to 1. When it is exactly 1, all of the eigenvalues of the convection diffusion operator coalesce into a single highly defective eigenvalue. The Jordan canonical form consists of one gigantic Jordan block, and the eigenvalue is catastrophically ill conditioned. For Péclet numbers near 1 the eigenvalues are distinct but crowded, and they are all ill conditioned.

Table 5.6 lists the outcomes of runs with a variety of Péclet numbers. In these experiments a coarser grid with $h = 1/20$ was used. The dimension of the convection-diffusion matrix was thus $19^3 = 6859$. The look-ahead Lanczos process was run for 450 steps (taking about 2.5 seconds), and the eigenvalues of the resulting 450×450 matrix were calculated by both the QR and the BR algorithm.

For Péclet numbers far from 1, the BR algorithm returns good results in about one sixth the time as QR. For three values nearer 1, BR returned without calculating the eigenvalues, because it needed more space than was allocated. That is, the bandwidth blew up. We had allocated enough room for a band of 42 diagonals above the main diagonal. The numbers in the last column of Table 5.6 show that this was far more than enough room for those runs that were successful. For the unsuccessful runs, 42 diagonals was far less than enough. On a subsequent attempt we increased the maximum bandwidth to 120, but that still was not enough. By taking the bandwidth large enough, we would eventually be able to make the code work. However, our experiences with the earlier version of the code suggest that bandwidth blowups are a sign of trouble that should not be ignored. They cause a severe increase in computing time, and the computed eigenvalues are likely to be inaccurate. We suggest that BR be used with a modest maximum bandwidth. If it cannot solve the problem within the allocated space, it probably will not be able to solve the problem economically or accurately.

Another way to deal with bandwidth blowups is to increase the tolerance τ. When we set τ at 100, BR succeeded for all three of the Péclet numbers for which it had

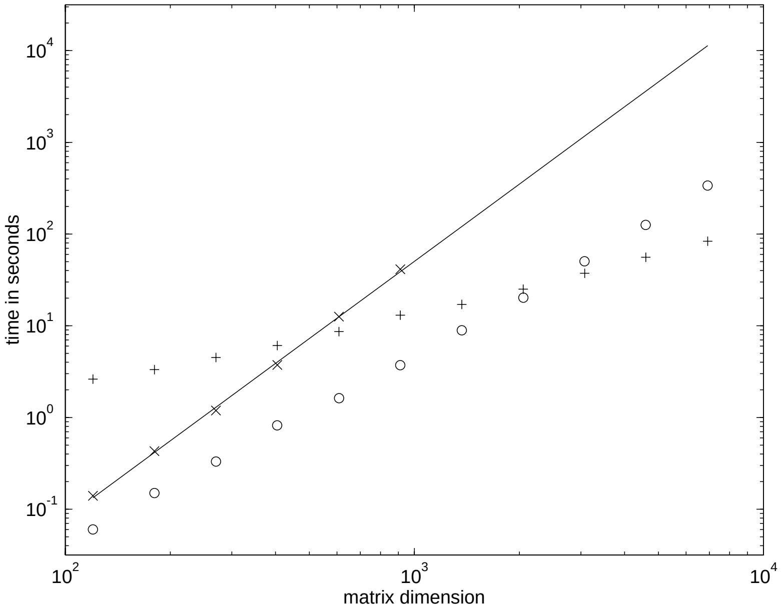


FIG. 5.2. *Time to generate $m \times m$ a narrow-band matrix (plus symbols) from pentadiagonal Toeplitz operator and calculate its eigenvalues by the QR algorithm (crosses) and by the BR algorithm with $\tau = 15$ (circles).*

failed at $\tau = 15$. It also succeeded for Péclet numbers 0.9 and 1.1. Execution times were under one second. However, the results agreed with those computed by QR to only about one decimal place. This is partly due to loss of accuracy caused by taking such a high tolerance. However, the values computed by QR should not necessarily be accepted as correct, since most of the eigenvalues are extremely ill conditioned.

Pentadiagonal Toeplitz matrices. Our next example is the pentadiagonal Toeplitz matrix of dimension 25000 with symbol $(1, -10, 0, 10, 1)$. This is one of the matrices in the collection of Higham [9]. We did a series of Lanczos runs with m in the range from 100 to 7000 and calculated the eigenvalues of the resulting matrix by the BR algorithm. For $m < 1000$ cases, we also computed the eigenvalues by the QR algorithms and compared the results of the two computations. In no case did the relative difference exceed 6×10^{-8}. For the larger cases, spot checks using Rayleigh quotient iteration suggested that the eigenvalues are correct to at least six decimal places. The times are given in Figure 5.2.

The timings are generally similar to those for the convection-diffusion operator, but in this case the trend for the QR times was better than before: The slope of the QR time line is only 2.80. The slope of the BR line is 2.09, just as it was for the convection-diffusion operator. The slope of the Lanczos time line was 0.86. On the largest run we had $m = 6920$. The look-ahead Lanczos algorithm took 1.4 minutes to generate the matrix, and the BR algorithm took 5.6 minutes to calculate its eigenvalues. The projected time for the QR algorithm is 3.1 hours, some 33 times longer.

We did numerous tests on other pentadiagonal Toeplitz matrices, with comparable results.

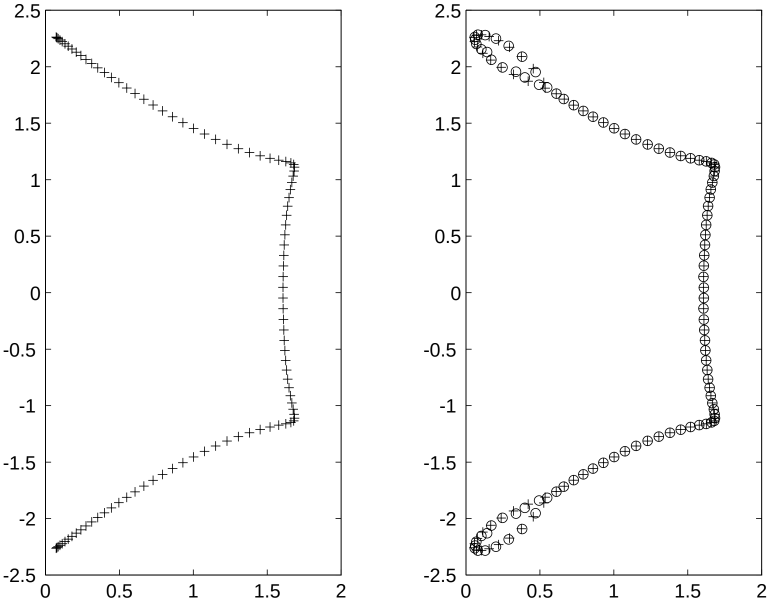


FIG. 5.3. *Spectrum of* 100×100 *Grcar matrix (left), and computed spectrum of transposed Grcar matrix (right) by the* QR *algorithm (pluses) and the* BR *algorithm (circles) with* $\tau = 15$.

Tolosa matrix. We performed the same sequence of experiments on a Tolosa matrix of dimension 40,000, which we obtained from the collection of Bai et al. [2]. The results were nearly identical to those shown in Figure 5.2, except that we were unable to get BR (with tolerance $\tau = 15$) to run for matrices above about 3000, because the bandwidth would not stay within the constraint that we had imposed (46 bands above the main diagonal). We remedied this by increasing τ to 30. We were then able to run m up to 7000 with modest bandwidths.

Other matrices. We also experimented with several other matrices from the collection of Bai et al. [2], including the Brusselator wave model of a chemical reaction, the Ising model for ferromagnetic materials, and the Navier–Stokes matrix of dimension 23560 from Mahajan, Dowell, and Bliss [12]. In all cases the BR algorithm was able to calculate the eigenvalues of the narrow-band matrices produced by look-ahead Lanczos quickly and with reasonable accuracy.

Upper Hessenberg examples. A few of the standard test matrices are narrow-band upper Hessenberg matrices to begin with. We can apply the BR algorithm directly to these matrices, without having to preprocess them by BHESS or look-ahead Lanczos.

The Grcar matrices, which are included in the collections of Bai et al. [2], Higham [9], and others, are a well-known family of matrices with ill-conditioned eigenvalues. They are upper Hessenberg Toeplitz matrices with bandwidth 5, $a_{i,i-1} = -1$, and $a_{ij} = 1$ for $j = i, \ldots, i+3$. We used the QR and BR algorithms to calculate the eigenvalues of the 100×100 Grcar matrix. The results agreed to about 10 decimal places. The spectrum is shown in the left panel of Figure 5.3.

The results are good because the eigenvalues are not sensitive to Hessenberg or near-Hessenberg perturbations of the matrix. The sensitivity can be brought out

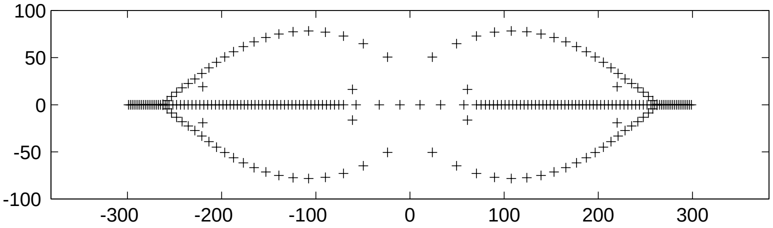


Fig. 5.4. *Incorrect spectrum of* 300×300 *Clement matrix computed by the QR algorithm.*

by transposing the matrix. We used the LAPACK code DGEHRD to reduce the transposed matrix to upper Hessenberg form, then we applied the QR algorithm. The resulting computed spectrum is given by plus symbols in the right panel of Figure 5.3. We also used BHESS to reduce the transposed matrix to banded upper Hessenberg form, and then we applied the BR algorithm. The results are given by circle symbols in the figure. We observe that both QR and BR failed to resolve the eigenvalues at the ends of the spectrum accurately, and both failed by about the same amount.

The Clement matrices, which are also in the collection of Higham, are $n \times n$ tridiagonal matrices with $a_{i,i-1} = i-1$, $a_{ii} = 0$, and $a_{i,i+1} = n-i$. Thus the 4×4 Clement matrix is

$$\begin{bmatrix} 0 & 1 & & \\ 3 & 0 & 2 & \\ & 2 & 0 & 3 \\ & & 1 & 0 \end{bmatrix}.$$

It is diagonally similar to a symmetric matrix, so its eigenvalues are all real; in fact they are the integers $\lambda_j = n+1-2j$, $j = 1, \ldots, n$. The bad scaling makes the eigenvalues ill conditioned. We calculated the eigenvalues of the 300×300 Clement matrix by the BR algorithm and got the correct eigenvalues to 10 decimal places. Presumably the BR algorithm's repeated rebalancing was helpful here. When we used the QR algorithm on the same matrix, we got the computed spectrum shown in Figure 5.4. This is clearly wrong; some of the eigenvalues have imaginary parts greater than 50. We tried balancing the matrix beforehand by the LAPACK routine DGEBAL, but that did not help. In fact, the numbers used to generate Figure 5.4 were from a run in which DGEBAL had been used.

This is an amusing comparison but, of course, it is unfair. The “right” way to calculate the eigenvalues of this matrix is to symmetrize it and then to apply any of the several fast, reliable algorithms for calculating the eigenvalues of a symmetric, tridiagonal matrix.

We modified the Clement matrix to disguise its underlying symmetry. This was accomplished by making a similarity transformation $A \to S^{-1}AS$, where S is the upper triangular, block-diagonal matrix $S = \mathrm{diag}\{U, U, U, \ldots\}$ with

$$U = \begin{bmatrix} 1 & & \\ & 1 & 1 \\ & & 1 \end{bmatrix}.$$

The resulting modified Clement matrix is still upper Hessenberg but has upper bandwidth 2, instead of 1. The QR algorithm (with preprocessing by DGEBAL) performed even worse on this matrix. In the 200×200 case it produced computed eigenvalues with imaginary parts as large as 50. The computed spectrum was similar in appearance to the spectrum shown in Figure 5.4. In contrast, the BR algorithm was able to compute the correct eigenvalues to 10 decimal places.

One can build modified Clement matrices with arbitrarily thick bands by applying further similarity transformations with matrices like S. We performed a few experiments along those lines with results similar to what we have reported here.

6. Concluding remarks. We have introduced the BR algorithm and shown that it does a good job of computing the eigenvalues of the narrow-band upper Hessenberg matrices produced by the look-ahead Lanczos process. Although the BR algorithm can sometimes fail, our experience has been that failures are rare. The BR algorithm is usually much faster than the QR algorithm, and it needs much less storage space. Experiments suggest that its execution time is little more than $O(m^2)$ as the matrix size m becomes large, provided that a liberal multiplier tolerance (e.g., $\tau = 15$) is used. Since large multipliers are allowed, we cannot claim that the algorithm is stable. Thus the raw output of the algorithm should not be accepted as accurate spectrum without further testing. In the context of the Lanczos process such further testing is carried out routinely, since it is also necessary to decide which of the eigenvalues of the narrow-band matrix are indeed good approximations to eigenvalues of the original large matrix.

REFERENCES

[1] B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow, Y. Ikebe, V. C. Klema, and C. B. Moler, *Matrix Eigensystem Routines—EISPACK Guide*, 2nd ed., Springer-Verlag, New York, 1976.

[2] Z. Bai, D. Day, J. Demmel, and J. Dongarra, *A Test Matrix Collection for Nonhermitian Eigenvalue Problems*, Tech. Rep., University of Kentucky, Lexington, KY, 1996. Available by anonymous ftp from ftp.ms.uky.edu. from the directory pub/misc/bai.

[3] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen, *LAPACK Users' Guide*, SIAM, Philadelphia, 1992.

[4] R. W. Freund, M. H. Gutknecht, and N. M. Nachtigal, *An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices*, SIAM J. Sci. Comput., 14 (1993), pp. 137–158.

[5] R. W. Freund, N. M. Nachtigal, and J. C. Reeb, *QMRPACK Users' Guide*, Tech. Rep. ORNL/TM-12807, Oak Ridge National Laboratory, Oak Ridge, TN, http://www.epm.ornl.gov/∼santa/ (1994).

[6] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed., Johns Hopkins University Press, Baltimore, 1996.

[7] M. H. Gutknecht, *A completed theory of the unsymmetric Lanczos process and related algorithms, Part* I, SIAM J. Matrix Anal. Appl., 13 (1992), pp. 594–639.

[8] M. H. Gutknecht, *A completed theory of the unsymmetric Lanczos process and related algorithms, Part* II, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 15–58.

[9] N. J. Higham, *The Test Matrix Toolbox for MATLAB*, Tech. Rep. 276, University of Manchester, Manchester, England, http://www.ma.man.ac.uk/∼higham/testmat.html (1995).

[10] G. W. Howell, G. A. Geist, and N. Diaa, *Gaussian Reduction to a Similar Near-tridiagonal Hessenberg Form: Algorithm BHESS*, preprint, 1995.

[11] G. W. Howell, G. A. Geist, and T. Rowan, *Error Analysis of Reduction to Similar Banded Hessenberg Form*, Tech. Rep. ORNL/TM-13344, Oak Ridge National Laboratory, Oak Ridge, TN, March 1998.

[12] A. Mahajan, E. H. Dowell, and D. Bliss, *Eigenvalue calculation procedure for an Euler/Navier-Stokes solver with applications to flows over airfoils*, J. Comput. Phys., 97 (1991), pp. 398–413.

[13] G. Schrauf, *Algorithm* 696*, An inverse Rayleigh iteration for complex band matrices*, ACM Trans. Math. Software, 17 (1991), pp. 335–340.

[14] G. W. Stewart, *The efficient generation of random orthogonal matrices with an application to condition estimators*, SIAM J. Numer. Anal., 17 (1980), pp. 403–409.

[15] D. S. Watkins, *Fundamentals of Matrix Computations*, John Wiley and Sons, New York, 1991.

[16] D. S. Watkins, *QR-like algorithms—an overview of convergence theory and practice*, in The Mathematics of Numerical Analysis, Lectures in Applied Math. 32, J. Renegar, M. Shub, and S. Smale, eds., American Mathematical Society, Providence, RI, 1996, pp. 879–893.

[17] D. S. Watkins and L. Elsner, *Chasing algorithms for the eigenvalue problem*, SIAM J. Matrix Anal. Appl., 12 (1991), pp. 374–384.

[18] D. S. Watkins and L. Elsner, *Convergence of algorithms of decomposition type for the eigenvalue problem*, Linear Algebra Appl., 143 (1991), pp. 19–47.

[19] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, England, 1965.