JOURNAL OF COMPUTATIONAL PHYSICS 26, 43–65 (1978)

The Incomplete Cholesky—Conjugate Gradient Method for the Iterative Solution of Systems of Linear Equations*

DAVID S. KERSHAW

University of California, Lawrence Livermore Laboratory, Livermore, California 94550

Received August 23, 1976; revised January 21, 1977

A new iterative method for the solution of systems of linear equations has been recently proposed by Meijerink and van der Vorst [1]. This method has been applied to real laser fusion problems taken from typical runs of the laser fusion simulation code LASNEX [2]. These same problems were also solved by various standard iteration methods. On a typical hard problem, the new method is about 8000 times faster than the point Gauss–Seidel method, 200 times faster than the alternating direction implicit method, and 30 times faster than the block successive overrelaxation method with optimum relaxation factor. The new method has two additional virtues. (1) Most of the algorithm is trivially vectorizable with a vector length equal to the full dimension of the system of linear equations. Thus, great savings are possible on vector machines. (2) The new method has a universal scope of application for solution of implicitly differenced partial differential equations. The only restrictions are that the matrix be symmetric and positive definite. The algorithm of Meijerink and van der Vorst which applied only to positive definite symmetric M-matrices is generalized to apply to positive definite symmetric matrices and further generalized to apply to nonsingular matrices arising from partial differential equations. A general description of the method is given. Numerical results are discussed and presented, and an explanation is given for the success of the method.

INTRODUCTION

A new iterative method for the solution of systems of linear equations has been recently proposed by Meijerink and van der Vorst [1]. This method has been applied to real laser fusion problems taken from typical runs of the laser fusion simulation code LASNEX [2]. These same problems were also solved by various standard iteration methods. On a typical hard problem, the new method is about 8000 times faster than the point Gauss–Seidel method, 200 times faster than the alternating direction implicit method, and 30 times faster than the block successive overrelaxation method with optimum relaxation factor. The new method has two additional virtues.

(1) Most of the algorithm is trivially vectorizable with a vector length equal to the full dimension of the system of linear equations. Thus, great savings are possible on vector machines.

* Research performed under the auspices of the U.S. Energy, Research and Development Administration under contract No. W7405-ENG-48.

0021-9991/78/0261–0043$02.00/0

(2) The new method has a universal scope of application for solution of implicitly differenced partial differential equations. The only restrictions are that the matrix be symmetric and positive definite. The algorithm of Meijerink and van der Vorst also required that the matrix be an M-matrix (i.e., $M_{ij} \leqslant 0$ for all $i \neq j$). A simple generalization of their algorithm discussed in Section 1 eliminates this requirement and still gives a very good approximate factorization in all applications we have tried. A large variety of physical problems lead to symmetric differential operators, and symmetric differencing of the partial differential equation is usually required for energy or particle number conservation. In the case of nonsymmetric differential operators, simple modifications of the method are still applicable and seem to work quite well (see Appendix A). The stability requirement that errors not grow in time leads to positive definite matrices. Methods like alternating direction implicit only apply to five-point coupling where the point (k, l) is coupled to $(k \pm 1, l)$ and $(k, l \pm 1)$. If the physics requires that (k, l) also couple to $(k \pm 1, l \pm 1)$ and $(k \pm 1, l \mp 1)$ (as is often the case), the methods become inapplicable; whereas the new method works just as well for this case. Methods like block successive overrelaxation only apply to cyclic, consistently ordered matrices. The new method has no such restrictions. In fact, the new method has been successfully applied to diffusion problems where the mesh points can be placed at arbitrary positions on the plane and arbitrarily ordered, and different points can have different numbers of neighbors. The corresponding matrix has nonzero entries on the diagonal and the other nonzero entries are sprinkled more or less randomly over the matrix.

In Section 1, a general description of the new method is given. In Section 2, the numerical results are presented. In Section 3, some explanations are given for the striking success of the method.

1. The Incomplete Cholesky—Conjugate Gradient Method

The conjugate gradient method was originally proposed in 1952 by Hestenes and Stiefel [3]. Excellent discussions of the method have recently appeared by Reid [4] and Concus *et al.* [5]. Given a system

$$Mx = y$$

of N linear equations where M is symmetric and positive definite, and an initial guess x_0 for the solution vector x, one could try to approximate x by letting

$$x_m = x_0 + \sum_{i=1}^{m} \alpha_i M^{i-1}(Mx_0 - y) \tag{1}$$

and choosing the α_i to minimize

$$\| x_m - x \|_M ,$$

where $\| z \|_M = (z, Mz)^{1/2}$.

Writing Eq. (1) in the form

$$x_m - x = (x_0 - x) + \sum_{i=1}^{m} \alpha_i M^i (x_0 - x)$$

it is obvious that the α_i that minimize $\| x_m - x \|_M$ may be found by orthonormalizing the set of vectors

$$M^i(x_0 - x); \qquad i = 1, 2, \ldots, m$$

in the $\| \ \|_M$ norm, thus obtaining an orthonormal set

$$u_i = \sum_{j=1}^{i} c_{ij} M^j (x_0 - x); \qquad i = 1, 2, \ldots, m$$

with $(u_i , Mu_j) = \delta_{ij}$, and then setting

$$(x_m - x) = (x_0 - x) - \sum_{i=1}^{m} (u_i , M(x_0 - x))\, u_i ,$$

or

$$x_m = x_0 - \sum_{i=1}^{m} (u_i , (Mx_0 - y))\, u_i . \tag{2}$$

If it were necessary to orthogonalize each new vector, $M^j (x_0 - x)$, with respect to each of the previous $(j - 1)$ vectors, the method would be very expensive. However, if instead of $M^j (x_0 - x)$ we take as our jth vector $M(x_{(j-1)} - x)$, which is just a linear combination of $M^j (x_0 - x)$ and the previous $(j - 1)$ vectors, then the jth vector is already orthogonal to the first $(j - 2)$ vectors, for

$$\begin{aligned} &(M^i(x_0 - x),\ M^2(x_{(j-1)} - x)), && \text{where} \quad i \leqslant j - 2, \\ &= (M^{i+1}(x_0 - x),\ M(x_{(j-1)} - x)), && \text{where} \quad i + 1 \leqslant j - 1, \end{aligned}$$

and $(x_{(j-1)} - x)$ was constructed orthogonal (in the M norm) to $M^l (x_0 - x)$, $l = 1, 2, \ldots, (j - 1)$. Therefore, one need only orthogonalize the jth vector with respect to $u_{(j-1)}$ and as one iterates, one need only store the last two orthonormal vectors $u_{(j-1)}$ and u_j.

This is accomplished *recursively* by the Hestenes and Stiefel algorithm: Let $r_0 = y - Mx_0$ and $p_0 = r_0$, then

$$a_i = (r_i , r_i)/(p_i , Mp_i), \tag{3a}$$

$$x_{(i+1)} = x_i + a_i p_i , \tag{3b}$$

$$r_{(i+1)} = r_i - a_i Mp_i , \tag{3c}$$

$$b_i = (r_{(i+1)} , r_{(i+1)})/(r_i , r_i), \tag{3d}$$

$$p_{(i+1)} = r_{(i+1)} + b_i p_i , \tag{3e}$$

$$i = 0, 1, 2, \ldots.$$

The p_i generated by this algorithm are just the (unnormalized) u_i of Eq. (2). That is

$$(p_i\,,\, Mp_j) = 0; \qquad i \neq j,$$

and the x_i of Eq. (3b) are just the x_i of Eq. (2). It is clear from Eq. (2) that when $m = N$, one has subtracted off the components of $(x - x_0)$ along a complete orthonormal set of vectors and therefore $x_N = x$. It is also clear from Eq. (2) that if the matrix M has only r distinct eigenvalues, then the $M^j(x_0 - x), j = 0, 1, 2,...$ all lie in an r-dimensional subspace, and so $x_r = x$, and the conjugate gradient method gets the exact answer after only r iterations. The whole process is completely analogous to the approximation of a function by a set of orthonormal polynomials, only the vector space is of finite rather than infinite dimension.

Just as with approximation of functions by orthonormal polynomials where one often gets quite good approximation with only a small number of polynomials, with the conjugate gradient method one often finds that $\| x_i - x \|/\| x \|$ is quite small even though $i \ll N$. This is especially true if the matrix M has many nearly degenerate or clustered eigenvalues which will be the case if M does not differ very much from the identity matrix. In real physics problems, the matrix M obtained by differencing a partial differential equation usually has a very large spread of eigenvalues with no approximate degeneracy. In the laser fusion problems studied typically

$$\lambda_{\max}/\lambda_{\min} \approx 10^4$$

with no multiple eigenvalues and eigenvalues evenly distributed between $\lambda_{\max}$ and $\lambda_{\min}$. Thus, for these problems, the conjugate gradient method in its simplest form (Eqs. (3a) through (3e)) does very poorly.

One way of solving the system of linear equations $Mx = y$ is Gaussian elimination and the most efficient form of Gaussian elimination when M is symmetric and positive definite is the Cholesky decomposition method [6] where one writes

$$M = LL^T \tag{4}$$

where L is lower triangular. Equation (4) uniquely determines L and since L is lower triangular, $L^{-1}z$ and $(L^T)^{-1}z$, where z is any vector, are easily calculated and so

$$x = (L^T)^{-1}\,(L^{-1}y) \tag{5}$$

solves the problem. Solving Eq. (4) for L, one finds it is determined column-by-column recursively by

$$L_{ii} = \left[M_{ii} - \sum_{k=1}^{(i-1)} L_{ik}^2\right]^{1/2}, \tag{6a}$$

$$L_{ji} = \left[M_{ji} - \sum_{k=1}^{(i-1)} L_{jk}L_{ik}\right]\Big/L_{ii}\,;$$

$$j = (i+1), (i+2),..., N. \tag{6b}$$

A simple modification of (4) in which square roots are avoided is as follows.

$$M = LDL^T, \tag{7a}$$

where off diagonal entries of D are zero and L and D are determined recursively column-by-column as follows.

$$\begin{aligned} L_{ji} &= M_{ji} - \sum^{(i-1)} L_{jk}L_{ik}D_{kk}, \\ &j = i, (i+1),..., N, \\ \text{and} \quad D_{ii} &= (L_{ii})^{-1}. \end{aligned} \tag{7b}$$

Unfortunately, for large sparse matrices, the calculation of L is both very time and very storage consuming because in calculating L, many of the elments of M which were zero become nonzero in L and so the sparseness of the original matrix is lost.

To avoid this problem, one can do incomplete Cholesky decomposition. One defines a sparsity pattern which is to be forced on L. That is, one chooses a set of matrix entries P which one is going to force to be zero in L. Then as one proceeds through algorithm (7b), whenever an L_{ij} turns up with

$$(i, j) \in P,$$

one sets it to zero and continues with the algorithm. Thus these elements of L are neither calculated nor stored. The simplest choice for P is

$$P = \{(i, j) \mid M_{ij} = 0;\ i, j = 1,..., N\},$$

that is, L is forced to have the same sparsity pattern as M. In Ref. [1] this choice is referred to as ICCG(0). This choice for P shall be used throughout the rest of this paper. In this way, an approximate inverse for M is obtained, i.e.,

$$M = LDL^T + E,$$

where E is a small error matrix whose nonzero entries all lie in the set P. As we proceed through algorithm (7b) it is crucial that all the L_{ii} be greater than zero. If $L_{ii} = 0$ then the algorithm breaks down, and if $L_{ii} < 0$, then LDL^T is not positive definite and the conjugate gradient method can no longer be used to get the exact solution as in Eqs. (8) and (9). Complete Cholesky decomposition always gives $L_{ii} > 0$ and Meijerink and van der Vorst showed that if M is an M-matrix ($M_{ij} \leqslant 0$ if $i \neq j$) incomplete Cholesky will always give $L_{ii} > 0$. Incomplete Cholesky decomposition of an arbitrary positive definite symmetric matrix will not always give $L_{ii} > 0$. A simple counterexample is the positive definite symmetric matrix

$$\begin{pmatrix} 3 & -2 & 0 & 2 \\ -2 & 3 & -2 & 0 \\ 0 & -2 & 3 & -2 \\ 2 & 0 & -2 & 3 \end{pmatrix}$$

Complete Cholesky (LDL^T, Eq. (7b)) gives $L_{44} = \frac{1}{3}$ while ICCG(0) gives $L_{44} = -5$. However, since we are not trying to get an exact decomposition, if an L_{ii} turns up such that $L_{ii} \leqslant 0$, we can simply set L_{ii} to some positive value and then go on with algorithm (7b). This will cause only the ith diagonal element of error matrix, E_{ii}, to become nonzero. All other nonzero entries of E will still lie in the set P. If $L_{ii} \leqslant 0$ rarely occurs (i.e., if the decomposition is mostly stable) this should work quite well [7]. Thus we have a positive definite, symmetric approximate factorization of M even when M is not an M-matrix. One must still ask if altering the diagonal elements of L in this way worsens the approximate LDL factorization of M. For a wide variety of problems we have encountered in our Laser Fusion work, this generalized LDL^T factorization is still a very good approximation as evidenced by the very rapid convergence of the conjugate gradient iterations [7]. Thus in approximate Cholesky factorization stability does not seem to be as crucial as in complete Cholesky decomposition. A few pivots can be unstable and one can still get quite a good approximate factorization as long as most of the pivots are stable. The work and storage involved in computing this approximate L is much less than that involved in computing the exact L.

For the five-point difference scheme used in my examples, storage for the complete Cholesky decomposition is KL^2 (where $k = 1, 2,..., K$ and $l = 1, 2,..., L$) whereas for the incomplete Cholesky decomposition storage it is only KL. For complete Cholesky the approximate number of multiplications is $\frac{1}{2}KL^3$ while for incomplete Cholesky it is $4KL$.

To get an exact solution, rewrite

$$Mx = y$$

as

$$[L^{-1}M(L^T)^{-1}]\,(L^T x) = (L^{-1})\,y, \tag{8}$$

(here we use the $M = LL^T$ decomposition, Eqs. (6a) and (6b)). If $(LL^T)^{-1}$ is an approximate inverse for M, then $L^{-1}M(L^T)^{-1}$ will be an approximate identity matrix and so by what was said earlier, the conjugate gradient method should converge very rapidly when applied to the matrix $L^{-1}M(L^T)^{-1}$. Substituting this matrix into Eqs. (3), after a little rearrangement, one obtains the modified algorithm:

Let $r_0 = y - Mx_0$ and $p_0 = (LL^T)^{-1}r_0$, then

$$a_i = (r_i\,,\,(LL^T)^{-1}r_i)/(p_i\,,\,Mp_i), \tag{9a}$$

$$x_{(i+1)} = x_i + a_i p_i\,, \tag{9b}$$

$$r_{(i+1)} = r_i - a_i\,Mp_i\,, \tag{9c}$$

$$b_i = (r_{(i+1)}\,,\,(LL^T)^{-1}r_{(i+1)})/(r_i\,,\,(LL^T)^{-1}r_i), \tag{9d}$$

$$p_{(i+1)} = (LL^T)^{-1}\,r_{(i+1)} + b_i p_i\,, \tag{9e}$$

$$i = 0,\ 1,\ 2,....$$

The success of the method will depend on how good an approximate inverse $(LL^T)^{-1}$ is. Physical intuition and experience tell us that for the diffusion equation

(and most other P.D.E.'s) M^{-1} (the Green's function) couples a given zone most strongly to its nearest neighbors and therefore neglecting the coupling to more distant neighbors should be a good approximation. Thus for matrices arising from P.D.E.'s we expect the ICCG method and its generalizations to work quite well. For arbitrary matrices, however, the neglect of coupling to distant neighbors is probably disastrous. As is shown in Sections 2 and 3, it is quite good for the problems we have studied.

2. Numerical Results

LASNEX [2], the Livermore laser fusion code, is a two-dimensional Lagrangian hydrodynamic code with diffusive radiation transport. For problems of interest (Δt) (the smallest practical time step) is often 10^3 to 10^4 times larger than τ (the characteristic relaxation time for an initial perturbation in radiation energy) so fully implicit differencing is an absolute necessity. The equation being solved is of the form

$$\partial f/\partial t = \underbrace{\nabla D\, \nabla f}_{\text{(transport term)}} + \underbrace{\gamma(f^e - f)}_{\text{(electron-radiation coupling term)}}$$

where D, γ, and f^e are given nonnegative functions of space and time. This is differenced in the form

$$\frac{f_{k,l}^{(m+1)} - f_{k,l}^{m}}{\Delta t} = (Rf)_{k,l}^{(m+1)} + (Sf)_{k,l}^{(m+1)} + \gamma_{k,l}[f_{k,l}^{e} - f_{k,l}^{(m+1)}],$$

$$k = 1, 2, \ldots, K \quad \text{and} \quad l = 1, 2, \ldots, L, \quad N = KL,$$

where

$$Rf = (R_1 - R_2)f,$$

$$Sf = (S_1 - S_2)f,$$

$$(R_1 f)_{k,l}^{(m+1)} = \alpha_{k,l} f_{(k+1),l}^{(m+1)} + \alpha_{(k-1),l} f_{(k-1),l}^{(m+1)},$$

$$(R_2 f)_{k,l}^{(m+1)} = [\alpha_{k,l} + \alpha_{(k-1),l}] f_{k,l}^{(m+1)},$$

$$(S_1 f)_{k,l}^{(m+1)} = \beta_{k,l} f_{k,(l+1)}^{(m+1)} + \beta_{k,(l-1)} f_{k,(l-1)}^{(m+1)},$$

$$(S_2 f)_{k,l}^{(m+1)} = [\beta_{k,l} + \beta_{k,(l+1)}] f_{k,l}^{(m+1)}.$$

This may be rewritten in matrix form as

$$Mf^{(m+1)} = y, \tag{10}$$

where

$$M = A - R - S,$$

$$(Af)_{k,l}^{(m+1)} = ((1/\Delta t) + \gamma_{k,l}) f_{k,l}^{(m+1)} = \sigma_{k,l} f_{k,l}^{(m+1)},$$

and

$$y = (1/(\Delta t)) f_{k,l}^{m} + \gamma_{k,l} f_{k,l}^{e}.$$

The matrix is clearly symmetric and $D \geqslant 0$ in the original partial differential equation, implies that $\alpha_{k,l}$ and $\beta_{k,l} \geqslant 0$. It is easy to show that this implies that M is positive definite.

Users of the code contributed typical problems on which they were currently working. For each problem, a time was chosen about halfway through the run, when all physics was in full swing, and M, f^m, and y (see Eq. (10)) were written from the radiation subroutine to a disc file. In addition, to facilitate future comparisons, a model problem was studied. Various iterative schemes were then used to calculate $f^{(m+1)}$ using f^m as an initial guess. For each scheme the work, or number of multiplications and divisions per iteration is given. The work is further broken down into recursive work where the calculation of the ith element depends on the previous calculation of the $(i-1)$st element or the $(i-K)$th element, and vector work where all N elements may be simultaneously calculated. For our problems the diffusion coefficient (and hence α and β) are arbitrary positive functions of position and often change by large amounts from one zone to the next. Therefore, R and S are strongly noncommutative and for both point SOR and ADI schemes there is no theoretical basis for choosing relaxation parameters to accelerate convergence. Therefore no acceleration parameters have been used for SOR and ADI.

(1) Point Gauss–Seidel or point successive overrelaxation with overrelaxation parameter $\omega = 1$, hereafter referred to as *GS*. Work is $2N$ vector and $3N$ recursive. Storage (besides the storage for M, $f^{(m+1)}$ and y) is 0.

(2) Alternating direction implicit defined by

$$(A - R) f_{(i-1/2)}^{(m+1)} = S f_{i}^{(m+1)} + y,$$

$$(A - S) f_{(i\ 1)}^{(m+1)} = R f_{(i+1/2)}^{(m+1)} + y,$$

where $f_i^{(m+1)}$ is the ith iteration approximation to $f^{(m+1)}$, referred to as ADI 1. Work is $8N$ vector and $4N$ recursive. Storage is $3N$.

(3) A more implicit version of ADI defined by

$$(A - R + S_2) f_{(i+1/2)}^{(m+1)} = S_1 f_{i}^{(m+1)} + y,$$

$$(A - S + R_2) f_{(i+1)}^{(m+1)} = R_1 f_{(i+1/2)}^{(m+1)} + y,$$

referred to as ADI 2. Work is $6N$ vector and $4N$ recursive. Storage is $3N$.

(4) Optimized block successive overrelaxation where the overrrelaxation parameter is given by

$$\omega_b = \frac{2}{[1 + (1 - \rho^2(J)])^{1/2}},$$

where $\rho(J)$ is the spectral radius of the associated block Jacobi matrix, J. In what follows, $\rho(J)$ was computed to high accuracy by the power method [8] and the optimum ω_b was put in for all iterations. Thus, our curves for this method (BSOR) may converge faster than practical BSOR where one iterates toward the correct ω_b as one iterates toward the solution vector. Work is $2N$ vector and $4N$ recursive. Storage is $2N$.

(5) The incomplete Cholesky–conjugate gradient method referred to as ICCG. Storage is $4N$.

The ICCG method will give different results depending on whether the matrix is set up with bandwidth K or L. Numerically it has been found to be quite advantageous to choose the bandwidth to always be the smaller of K and L. This choice minimizes the number of entries which would be nonzero in the exact decomposition but are set to zero in the incomplete decomposition. By using the LDL^T decomposition and choosing D so that the diagonal elements of L are all ones, we can eliminate N divisions per iteration and the work for ICCG(0) is $11N$ vector and $4N$ recursive.

The recursive work for the ICCG(0) method is the same as for BSOR or ADI and only the vector work is greater. Thus on vector (parallel processing) machines or by the use of Stacklib [9] on the CDC 7600, the vector operations are so much faster than the recursive ones that all methods take roughly the same CPU time per iteration. Storage is slightly greater for ICCG(0).

The exact (to 14 significant decimal digits) solution, $f^{(m+1)}$, was computed for all problems using double precision complete Cholesky decomposition and all iterative solutions were compared to this exact answer.

Three problems were chosen for presentation as representative. The diffusion equation was differenced on a mesh which was highly irregular in both shape and size in all three problems. The boundary of the region was also very irregular in all three problems. All problems had mixed Neumann

$$n \cdot \nabla f = 0,$$

and Dirchlet,

$$f = 0, \qquad \text{boundary conditions.}$$

All problems had regions which were well heated and regions which were quite cold.

Problem 1 was the most difficult of the three. It had 555 zones (dimension of $f^{(m+1)} = 555$). The α and β coefficients (off diagonal elements) of the matrix varied by 10^{27} over all the zones, while the σ coefficients (see Eq. (10)) varied by 10^{11} over all zones. There were regions of the problem where

$$\alpha \text{ and } \beta \text{ were } < 10^{-10}\sigma$$

so there was no transport and for these (k, l) the matrix was pure diagonal to ten-place accuracy. There were regions of the problem where

$$\alpha \text{ and } \beta \text{ were } > 10^6\sigma$$

and so for these (k, l) the matrix was pure Laplacian ($M_{ii} = -\sum_{(j\neq i)} M_{ij}$) to six-place accuracy. The largest and smallest eigenvalues of M were found by the power series method to be

$$\lambda_{\max} = 8.276 \times 10^3,$$
$$\lambda_{\min} = 4.156 \times 10^{-7},$$

so the condition of the matrix was

$$\lambda_{\max}/\lambda_{\min} = 2 \times 10^{10}.$$

The numerical results are shown in Fig. 1. The curve for ADI 1 is not shown in Fig. 1 because although ADI 1 was very slowly asymptotically convergent [$[\epsilon_{(i+1)}/\epsilon_i = 0.99832$] for this problem, on the first iteration ϵ increased to 2.177×10^4 and it took thousands of iterations to get ϵ back down onto the graph ($\epsilon = 0.1$). It is interesting to note that for this ill-conditioned problem, BSOR has a serious drawback. The overrelaxation parameter was optimized to give BSOR the best asymptotic rate of convergence and indeed, the asymptotic rate has been much improved over BSOR with no overrelaxation (which looks much like ADI 2). At the same time, however, optimized overrelaxation has made the initial convergence much worse so that for

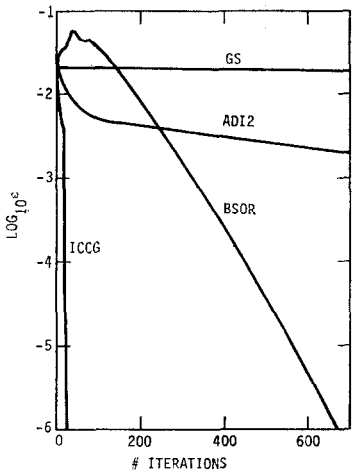

FIG. 1. Convergence curves for problem 1. $\epsilon = \| x^m - x^e \| / \| x^e \|$, where $\| x \|$ is the Euclidian norm, x^e is the exact solution vector, and x^m is the mth iteration approximation to x_i.

the first 130 iterations, the answer is worse than the initial guess. The number of iterations to achieve

$$\epsilon = 10^{-6}, \qquad \epsilon = \frac{\| f - f_{\text{exact}} \|}{\| f_{\text{exact}} \|}, \qquad \| x \| = (\Sigma x_i^2)^{1/2}$$

was,

ICCG — 25,
BSOR — 765,
ADI 2 — 4750,
ADI 1 — 10,200,
GS — 208,000.

Problem 2 was an "easy" problem. Variations in the coefficients were less severe with $10 > \alpha$ and $\beta > 10^{-16}$, $10 > \sigma > 10^{-6}$, over the whole problem. There were regions where

$$\alpha \text{ and } \beta \ll \sigma,$$

and so for these (k, l) the matrix was pure diagonal and there were regions where α and β were about as large as σ, but there was no region where α and $\beta \gg \sigma$ and so the matrix was nowhere pure Laplacian.

For this problem

$$\lambda_{\max} = 1.867 \times 10^3,$$
$$\lambda_{\min} = 2.47,$$
$$\text{condition} = \lambda_{\max}/\lambda_{\min} = 765.$$

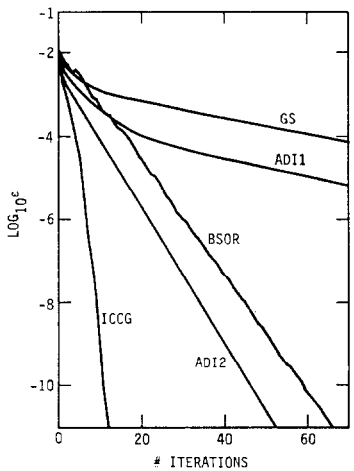

FIG. 2. Convergence curves for problem 2.

There were 1111 zones in the problem. The results are shown in Fig. 2. For this easier problem, the gap between the various methods narrowed but ICCG was still quite a bit faster. The number of iterations to achieve $\epsilon = 10^{-6}$ was

ICCG — 7,
BSOR — 25,
ADI 2 — 22,
ADI 1 — 110,
GS — 214.

Problem 3 was intermediate in difficulty. The dimension of the linear system was 595. The variation in coefficients was

$$10^{-5} > \alpha \quad \text{and} \quad \beta > 10^{-12},$$
$$10^{-4} > \sigma > 10^{-11}.$$

There were pure diagonal regions where

$$\alpha \text{ and } \beta < 10^{-3}\sigma$$

and pure Laplacian regions where

$$\sigma < 10^{-4} \qquad (\alpha \text{ or } \beta).$$

For this problem $\lambda_{\max} = 3.475 \times 10^{-4}$, $\lambda_{\min} = 8.785 \times 10^{-9}$, condition $= 3.96 \times 10^{4}$.
The results are shown in Fig. 3.

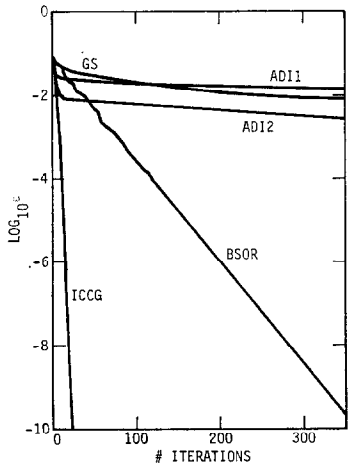

FIG. 3. Convergence curves for problem 3.

The number of iterations to achieve $\epsilon = 10^{-6}$ was

ICCG — 15,
BSOR — 201,
ADI 2 — 2817,
ADI 1 — 10861,
GS — 10076.

Since the exact matrix equations and initial guesses for the real physical problems 1 through 3 could not be given (it would take 3000 numbers to specify completely just the smallest of the three problems), we invented a model problem (problem #4) which is easy to specify exactly and which contains all the essential features of problems 1, 2, and 3, which made them so difficult. Thus the interested reader can make direct comparisons with our results.

A picture of the problem is shown in Fig. 9. For this problem $K = L = 50$. A corner has been cut out of the problem so that there are 1875 zones in the problem. Each zone has a $\rho_{k,l}$ value associated with it ($k = 1,..., K$; $l = 1,..., L$) and the value of $\rho_{k,l}$ for each zone is given in Fig. 9. In the notation of Eq. (10) the problem is specified by

$$\Delta t = 1$$

$$\gamma_{k,l} = 0 \qquad \text{for all } k \text{ and } l,$$

$$\alpha_{k,l} = 2(\rho_{k,l}^{-1} + \rho_{(k+1),l}^{-1})^{-1},$$

$$\beta_{k,l} = 2(\rho_{k,l}^{-1} + \rho_{k,(l+1)}^{-1})^{-1},$$

except on boundaries. All quantities are zone centered.

Along the $k = 1$ boundary marked N for Neumann boundary condition we have $\partial f/\partial \hat{n} = 0$, or $\alpha_{0,l} = 0$. Similarly, along the $k = 50$, $l = 1,..., 25$ boundary $\alpha_{50,l} = 0$. Along the $l = 1$ boundary, $\beta_{k,0} = 0$, and along the $l = 50$, $k = 1,..., 25$ boundary $\beta_{k,50} = 0$.

Along the $k = 25$, $l = 26,..., 50$ boundary marked D for Dirchlet boundary condition, we have $f^{m+1} = 0$ outside the problem and we choose

$$\alpha_{25,l} = \rho_{25,l},$$

$$f_{26,l}^{(m+1)} = 0.$$

Similarly along the $l = 25$, $k = 26,..., 50$ boundary

$$\beta_{k,25} = \rho_{k,25},$$

$$f_{k,26}^{(m+1)} = 0.$$

This specifies all boundary conditions. For f^m, the temperature at the end of the last time step we take

$$f^m_{k,l} = 10^{[(l-k+49)/24.5]},$$

and for our initial guess for f^{m+1} we guess it is zero everywhere. This completely specifies the model problem.

Our model problem has all the essential features of the real problems. It has mixed boundary conditions, regions where

$$\alpha \text{ and } \beta \gg \sigma,$$

and regions where

$$\alpha \text{ and } \beta \ll \sigma,$$

and α and β change by large factors between adjacent zones thus making R and S strongly noncommutative.

The largest and smallest eigenvalues of the matrix were

$$\lambda_{\max} = 7.855 \times 10^6,$$
$$\lambda_{\min} = 1,$$

giving a condition number of 7.855×10^6. The results are shown in Fig. 10. The number of iterations to achieve $\epsilon = 10^{-6}$ was

ICCG	29,
BSOR —	283,
ADI 2 —	9910,
ADI 1 —	255,000,
GS —	20,000.

Note that the curve for ADI 1 in Fig. 10 turns and goes very flat between the 500th and the 1000th iteration.

Figures 4, 5, and 6, compare ϵ_{norm}, the norm of the error vector divided by the norm of the solution, with $\epsilon_{\max}$, the maximum over all zones of the percentage error in $f^{(m+1)}$ in a given zone. For all methods except BSOR $\epsilon_{\max}$ follows ϵ_{norm} fairly closely. For BSOR, however, during the first 50 iterations $\epsilon_{\max}$ is rising while ϵ_{norm} is falling and on the 37th iteration, it reaches a maximum value of 2.18×10^4. Thus, with BSOR ϵ_{norm} is claiming a 17% error level while in certain zones $f^{(m+1)}$ is 2×10^4 times too large. This high noise level persists so that, e.g., on the 300th iteration ϵ_{norm} is 3.47×10^{-9} while $\epsilon_{\max} = 3.05 \times 10^{-2}$. For many physical applications $\epsilon_{\max}$ is required to be small and so the required number of iterations may be greatly increased.

The conjugate gradient (CG) method (Eqs. (3)) was also tried by itself. On problem 1, the method showed no appreciable convergence in the first 1000 iterations with

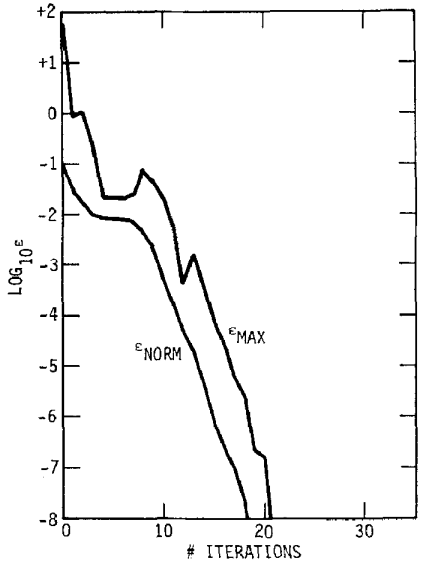

FIG. 4. Comparison of $\epsilon_{\max}$ and ϵ_{norm} for problem 3, ICCG, where $\epsilon_{\text{norm}} = \| x^m - x^e \| / \| x^e \|$, $\epsilon_{\max} = \text{Max}_i \mid x_i^m - x_i^e \mid / \mid x_i^e \mid$, where x_i is the ith component of the vector x.

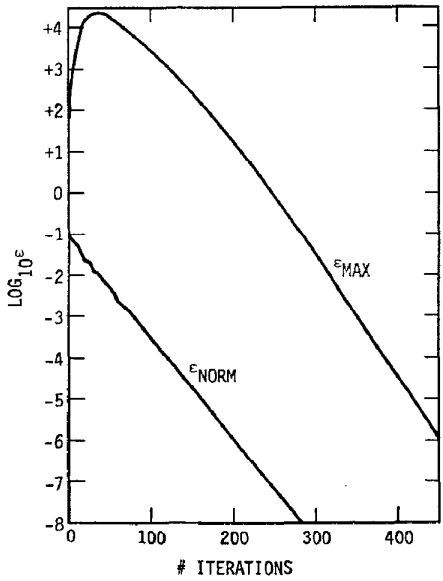

FIG. 5. Same for BSOR.

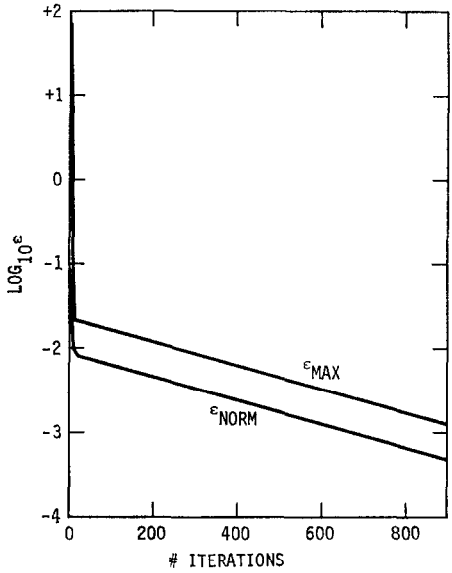

FIG. 6. Same for ADI 2.

both ϵ_{norm} and ϵ_{max} staying equal to their initial values. On problem 2, the CG method converged, but even slower than GS. On problem 3, ϵ_{norm} converged slowly (at about the same rate as GS) but meanwhile, ϵ_{max} increased to 10^6 and stayed large for 750 iterations after which it finally started to decrease. Thus, conjugate gradient without incomplete Cholesky was a disaster on problems 1 and 3 and was the slowest of all methods on problem 2.

The ICCG(3) method of Meijerink and van der Vorst [1] was also tried on our three problems. On all three problems it cut the number of iterations to achieve six-place accuracy in half but it increased the recursive work per iteration from $4N$ to $10N$ (vector work remains the same) and it took $3N$ more storage. Thus the gain in convergence rate was more than offset (on vector-oriented machines) by the loss in work per iteration and increased storage requirements.

3

Why is the new method so successful? Since conjugate gradient by itself does so badly on our test problems, we must analyze the effect of the imcomplete Cholesky decomposition. Even for a model problem such as

$$\nabla^2\phi = 0,$$

differenced on a regular square grid, it is extremely difficult to analytically compute the eigenvalues of $L^{-1}\,M(L^T)^{-1}$ and to see how close it is to the identity [10]. Instead,

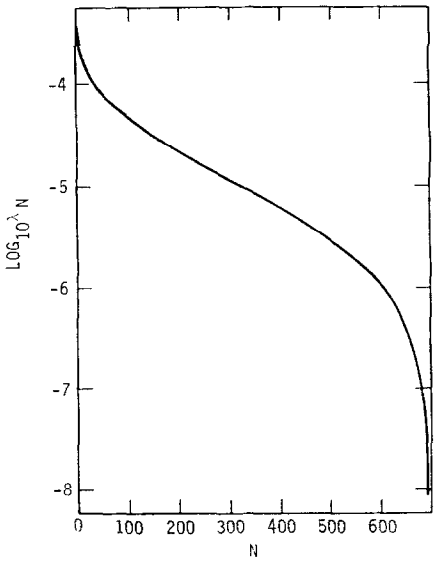

FIG. 7. The 695 eigenvalues of M for problem 3 arranged in order of decreasing magnitude.

it proved more profitable to use numerical methods for the determination of the eigenvalues. This was done for problem 3. All the 695 eigenvalues of M were calculated using BANDRD and TQL1 [11] and the results are plotted in Fig. 7. The largest and smallest eigenvalues of the original matrix M for this problem were

$$\lambda_{\max} = 3.4751 \times 10^{-4}$$

and

$$\lambda_{\min} = 8.7845 \times 10^{-9}$$

so the condition $= \lambda_{\max}/\lambda_{\min} = 3.956 \times 10^{4}$. The eigenvalues of M were all distinct and were spread evenly between $\lambda_{\max}$ and $\lambda_{\min}$. The largest 40 and smallest 40 eigenvalues of $L^{-1} M(L^T)^{-1}$ were calculated using the Lanczos method and checked with the power method [8]. Note that the ICCG(0) method is a Lanczos method and the coefficients a_i and b_i generated by Eqs. (9) are simply related to the tridiagonal matrix elements generated in the Lanczos method [12]. These are plotted in Fig. 8 and the largest 20 and smallest 20 are given in Table I. It is clear from Fig. 8 that $L^{-1} M(L^T)^{-1}$ is a very good approximate identity. Out of 695 eigenvalues, 640 of them lie between 1.15 and 0.85. This accounts for the great rapidity of convergence of the new method. An examination of Table I also shows why incomplete Cholesky decomposition followed by a linear iterative scheme such as H. L. Stone's scheme with $\alpha = 0$ [10] will not work nearly as well as ICCG. In this method, one writes

$$M = LL^T + E$$

where E is the error matrix. Then

$$Mx = y$$

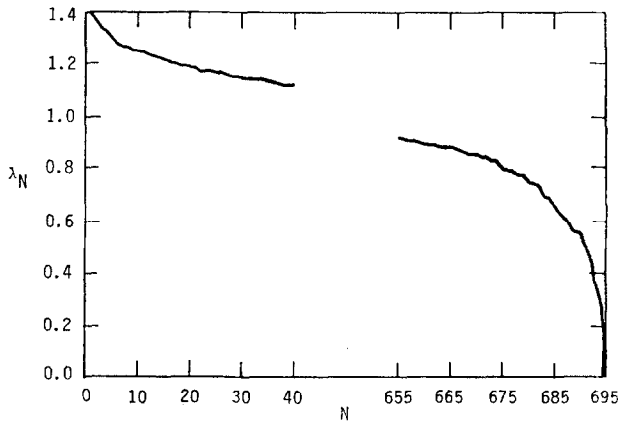

FIG. 8. Largest and smallest 40 eigenvalues of $L^{-1}M(L^T)^{-1}$ for problem 3, arranged in order of decreasing magnitude.

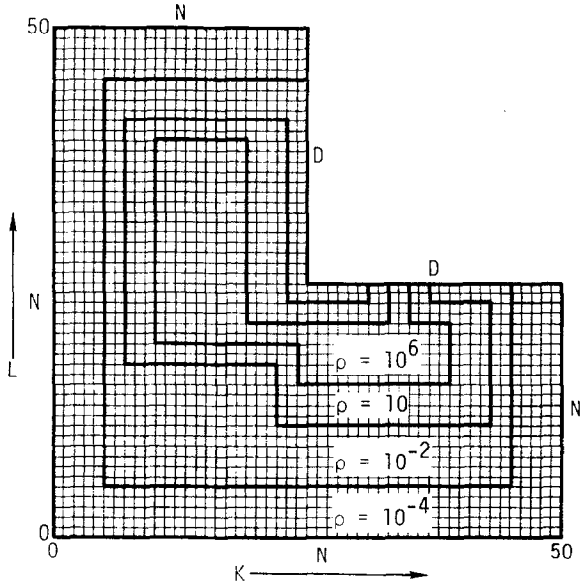

FIG. 9. The model problem with ρ values for each zone and boundary conditions (N or D).

can be rewritten as

$$L^{-1}\,M(L^T)^{-1}\,(L^T x) = [I + L^{-1}\,E(L^T)^{-1}]\,(L^T x) = L^{-1}y,$$

which leads to the iterative scheme

$$L^T x^{(m+1)} = L^{-1}y - L^{-1}\,Ex^m.$$

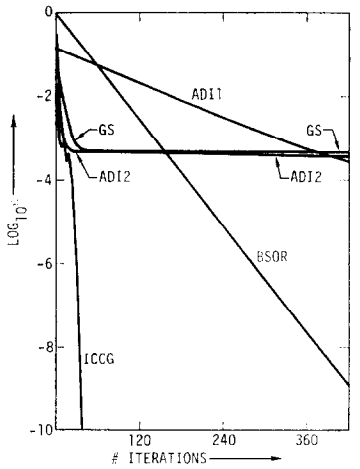

FIG. 10. Convergence curves for problem 4.

TABLE I

The Largest and Smallest 20 Eigenvalues of $L^{-1}M(L^T)^{-1}$ for Problem 3

1	1.3928	0.00466
2	1.3672	0.26209
3	1.3408	0.34010
4	1.3241	0.44114
5	1.3006	0.49221
6	1.2870	0.54889
7	1.2653	0.55108
8	1.2626	0.57347
9	1.2499	0.60203
10	1.2493	0.62470
11	1.2456	0.65454
12	1.2324	0.68270
13	1.2280	0.69131
14	1.2199	0.73451
15	1.2135	0.73781
16	1.2046	0.74406
17	1.1992	0.77077
18	1.1933	0.77163
19	1.1891	0.78438
20	1.1860	0.78767

Therefore, the asymptotic rate of convergence is given by

$$\rho = \max_{(i)} | 1 - \lambda_i |,$$

where λ_i are the eigenvalues of $L^{-1}M(L^T)^{-1}$. For problem 3

$$\lambda_{\min} = 4.6646 \times 10^{-3}$$
$$\lambda_{\max} = 1.3928,$$

so

$$\rho = 0.99534,$$

which means about 2900 iterations are required to reduce the worst eigenvector by a factor of 10^6. By contrast, the conjugate gradient method first reduces the components of the residual, $y - Mx_i$ along the eigenvectors with the most extreme eigenvalue to very small values. This is just why the Lanczos method is able to get good approximations to the extreme eigenvalues and their eigenvectors after only a few iterations. The residual then lies almost entirely in the subspace of the eigenvectors with less extreme eigenvalues and the iteration then proceeds as if the most extreme eigenvectors and eigenvalues were not present [12]. Thus, although the condition of $L^{-1}M(L^T)^{-1}$ is

$$\lambda_{\max}/\lambda_{\min} = 298.59,$$

which is not very much better than the condition of M, conjugate gradient immediately eliminates the eigenvector with the smallest eigenvalue and since the next smallest eigenvalue is

$$\lambda_{\min} = 0.26209$$

the effective condition number after the first few iterations is

$$\lambda_{\max}/\lambda_{\min} = 5.314,$$

a very large reduction in condition number.

Thus, the success of the new method is due to two factors.

(1) Incomplete Cholesky decomposition modifies the original system of linear equations to a new system whose matrix is very close to the identity matrix except for a few extreme eigenvalues.

(2) The conjugate gradient method applied to this modified system of linear equations quickly eliminates the few extreme eigenvalues and eigenvectors and ends up solving the linear system in an invariant subspace where the matrix is almost the identity matrix.

CONCLUSION

The incomplete Cholesky-conjugate gradient method is highly recommended. On fairly well-conditioned problems, it was faster than any of the standard methods.

On very ill-conditioned problems, it was enormously faster than any of the standard methods.

Most of the algorithm is trivially vectorizable leading to great savings on the coming generation of parallel processing (vector) machines.

The algorithm has a much wider range of applicability than most of the standard iterative methods.

APPENDIX A: Generalization of the ICCG Method to Arbitrary Nonsingular Sparse Matrices Arising from Partial Differential Equations

Consider a matrix equation,

$$Ax = y$$

where A is any nonsingular sparse matrix. Since A need not be symmetric or positive definite we shall do an incomplete LU decomposition. As before we choose a set of entries P and as we proceed through the standard LU algorithm [6] (without pivoting) we force all the entries in P to be zero in L and U. We use the form of the algorithm in which all diagonal elements of L are 1. The U_{ii} can now be positive or negative but if $U_{ii} = 0$ the algorithm breaks down. In this case, as before, we simply set U_{ii} to a nonzero value and go on with the algorithm thus introducing one nonzero entry on the diagonal of the error matrix $E = A - LU$. Often partial differential equations give rise to diagonally dominant matrices in which case exact LU decomposition (without pivoting) gives all good pivots and probably for this case incomplete LU decomposition will only rarely give $U_{ii} = 0$.

Thus we obtain an approximate inverse $(LU)^{-1}$ for A and we rewrite the original equation as

$$L^{-1} AU^{-1} (Ux) = L^{-1}y. \tag{8'}$$

We use a modified form of the conjugate gradient method which applies to any nonsingular matrix M. Equation (1) is modified to read

$$x_m = x_0 + \sum_{i=1}^{m} \alpha_i (M^T M)^{i-1} M^T (Mx_0 - y) \tag{1'}$$

and α_i is chosen to minimize $\| x_m - x \|$.

Note that because we are expanding in powers of $(M^T M)$ instead of powers of M we are now able to minimize $x_m - x$ in the Euclidean norm. This is different from the standard method of Hestenes and Stiefel [3] for arbitrary nonsingular matrices which minimizes $\| Mx_m - y \|$. This requires less work per iteration and should give faster convergence than the standard algorithm. It is well known (see, [4, Sect. 4.3]) that for the positive definite symmetric case minimizing in the $(z, M^{\mu} z)$ norm works best when μ is as small as possible.

Equations (3) now become

$$r_0 = y - Mx_0 \quad \text{and} \quad p_0 = M^T r_0$$

$$a_i = (r_i, r_i)/(p_i, p_i), \tag{3'a}$$

$$x_{i+1} = x_i + a_i p_i, \tag{3'b}$$

$$r_{i+1} = r_i - a_i Mp_i, \tag{3'c}$$

$$b_i = (r_{i+1}, r_{i+1})/(r_i, r_i), \tag{3'd}$$

$$p_{i+1} = M^T r_{i+1} + b_i p_i, \tag{3'e}$$

$$i = 0, 1, 2, \ldots.$$

Combining Eqs. (3') and Eq. (8') (i.e., $M = L^{-1}AU^{-1}$) after some algebra we obtain the algorithm

$$r_0 = y - Ax_0 \quad \text{and} \quad p_0 = (U^TU)^{-1} A^T(LL^T)^{-1} r_0$$

$$a_i = \frac{(r_i, (LL^T)^{-1} r_i)}{(p_i, U^TUp_i)}, \tag{9'a}$$

$$x_{i+1} = x_i + a_i p_i, \tag{9'b}$$

$$r_{i+1} = r_i - a_i Ap_i, \tag{9'c}$$

$$b_i = \frac{(r_{i+1}, (LL^T)^{-1} r_{i+1})}{(r_i, (LL^T)^{-1} r_i)}, \tag{9'd}$$

$$p_{i+1} = (U^TU)^{-1} A^T(LL^T)^{-1} r_{i+1} + b_i p_i, \tag{9'e}$$

$$i = 0, 1, 2, \ldots.$$

How good is this approximate LU factorization? It has been tried on mirror machine $2D$ phase-space diffusion problems (in LLL's magnetic confinement fusion program) where the analytic operator and its finite difference analog were neither symmetric nor positive definite. The results were very encouraging with the code being speeded up by a factor of 10 on stiff problems [13]. This algorithm was also tried on the problems presented in this paper and it required 1.5 to 2.5 times as many interations to converge to six-place accuracy as did the ICCG algorithm (Eqs. (9)), and about 50% more work per iteration. Thus the generalized conjugate gradient method was only 1.5 to 2.5 times slower than standard conjugate gradient (for positive definite symmetric matrices) on our problems.

So in the applications tried so far the approximate LU factorization combined with generalized conjugate gradient method seems to work quite well.

ACKNOWLEDGMENTS

I wish to thank Gene Golub for bringing the ICCG method to my attention, Garry Rodrigue for helpful discussions, and Yu-Li Pan, John White, and John Lindl for the "typical" LASNEX problems they donated. I wish to thank the reviewers for calling to my attention the possibility of L_{ii} being $\leqslant 0$ when M is not an M-matrix and also for their suggestion that I include a model problem.

REFERENCES

1. 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.* January (1977), to appear.
2. G. B. ZIMMERMAN, "Numerical Simulation of the High Density Approach to Laser Fusion," Lawrence Livermore Laboratory preprint, UCRL-74811, Livermore, Ca., 1973.
3. M. R. HESTENES AND E. STIEFEL, Methods of conjugate gradients for solving linear systems, *Nat. Bur. Standards J. Res.* **49** (1952), 409–436.
4. J. K. REID, On the method of conjugate gradients for the solution of large sparse systems of linear equations, *in* "Proceedings of the Conference on Large Sparse Systems of Linear Equations," Academic Press, New York, 1971.
5. P. CONCUS, G. H. GOLUB, AND D. P. O'LEARY, "A Generalized Conjugate Gradient Method for the Numerical Solution of Elliptic Partial Differential Equations," Lawrence Berkeley Laboratory Pub. LBL-4604, Berkeley, Ca., 1975.
6. L. FOX, Practical solution of linear equations and inversion of matrices, *Nat. Bur. Standards Appl. Math. Ser.* **39** (1954), 1–54.
7. The ICCG(0) method has been used in production in LASNEX with matrices generated by a nine-point difference approximation to the diffusion operator. Because the mesh consists of arbitrary quadrilaterals, the matrix (which is positive definite and symmetric) is not an M-matrix and has many positive off diagonal elements. Nevertheless, in running hundreds of problems for hundreds of time steps only very rarely did a $L_{ii} \leqslant 0$ turn up. When it did we put $L_{ii} = \sum_{j=1}^{(i-1)} |L_{ij}| + \sum_{j=(i+1)}^{N} |L_{ji}|$ and continued with the algorithm. Convergence was achieved just as rapidly in these cases as in nearby timesteps where no $L_{ii} \leqslant 0$ had turned up.
8. J. H. WILKINSON, "The Algebraic Eigenvalue Problem," Oxford Univ. Press, London, 1965; see Chapter 9, Section 2, for the power method and Chapter 6, Section 5, for a discussion of the Lanczos method.
9. F. H. MCMAHON, L. J. SLOAN, AND G. A. LONG, "Stacklibe—A Vector Function Library of Optimum Stack-Loops for the CDC 7600," Lawrence Livermore Laboratory publication UCID 30083, Livermore, Ca., 1976.
10. This problem is discussed by H. L. STONE, Iterative solution of implicit approximations of multidimensional partial differential equations, *SIAM J. Numer. Anal.* **5** (1968), 530–558, in Section 4 of his article. I have discussed Stone's method only for $\alpha = 0$ because for problems of our generality (arbitrary variations in diffusion coefficient) there is no theoretical basis for choosing α and furthermore, his ad hoc formula (26) for α must be in error because the left-hand side is dimensionless while the right-hand side has dimensions of length squared. Anyone wishing to compare my results with SIP can try it on the model problem (prob. #4) with α's of his own choosing.
11. J. H. WILKINSON AND C. REINSCH, "Linear Algebra," Vol. II, Chapters II/8 and II/3 of the "Handbook for Automatic Computation," Springer–Verlag, New York, 1971.
12. See Ref. [5, pp. 12, 13].
13. T. A. CUTLER, private communication.