NUMERICAL SOLUTION OF NONLINEAR ELLIPTIC PARTIAL DIFFERENTIAL EQUATIONS BY A GENERALIZED CONJUGATE GRADIENT METHOD

by

Paul Concus, Gene H. Golub and Dianne P. O'Leary

STAN-CS-76-585
DECEMBER 1976

COMPUTER SCIENCE DEPARTMENT
School of Humanities and Sciences
STANFORD UNIVERSITY

NUMERICAL SOLUTION OF NONLINEAR ELLIPTIC PARTIAL DIFFERENTIAL EQUATIONS BY A GENERALIZED CONJUGATE GRADIENT METHOD

by

Paul Concus
Lawrence Berkeley Laboratory
University of California
Berkeley, CA 94720

Gene H. Golub
Computer Science Department
Stanford University
Stanford, CA 94305

Dianne P. O'Leary
Department of Mathematics
University of Michigan
Ann Arbor, MI 48109

Also issued as Lawrence Berkeley Laboratory Report LBL-5572, University of California, Berkeley.

Research supported in part under Energy Research and Development Administration Grant E (04-3) 326 PA #30 and in part under National Science Foundation Grant MCS75-13497-A01.

ABSTRACT

We have studied previously a generalized conjugate gradient method for solving sparse positive-definite systems of linear equations arising from the discretization of elliptic partial-differential boundary-value problems. Here, extensions to the nonlinear case are considered. We split the original discretized operator into the sum of two operators, one of which corresponds to a more easily solvable system of equations, and accelerate the associated iteration based on this splitting by (nonlinear) conjugate gradients. The behavior of the method is illustrated for the minimal surface equation with splittings corresponding to nonlinear SSOR, to approximate factorization of the Jacobian matrix, and to elliptic operators suitable for use with fast direct methods. The results of numerical experiments are given as well for a mildly nonlinear example, for which, in the corresponding linear case, the finite termination property of the conjugate gradient algorithm is crucial.

0. Introduction

In earlier papers [8,9] we have discussed a generalized conjugate gradient iterative method for solving symmetric and nonsymmetric positive-definite systems of linear equations, with particular application to discretized elliptic partial differential boundary-value problems. The method consists of splitting the original coefficient matrix into the sum of two matrices, one of which is a symmetric positive-definite one that approximates the original and corresponds to a more easily solvable system of equations; the associated iteration based on this splitting is then accelerated using conjugate gradients. The conjugate gradient (cg) acceleration algorithm has a number of attractive features for linear problems, among which are: (a) not requiring an estimation of parameters, (b) taking advantage of the distribution of the eigenvalues of the iteration matrix, and (c) requiring fewer restrictions for optimal behavior than other commonly-used iteration methods, such as successive overrelaxation. Furthermore, cg is optimal among a large class of iterative algorithms in that for linear problems it reduces a particular error norm more than does any other of the algorithms for the same number of iterations.

In this paper we study an extension of the generalized conjugate gradient method to obtain solutions of systems of equations arising from elliptic partial-differential boundary value problems that are nonlinear. For such systems--which correspond to the minimization of convex nonquadratic functionals, as opposed to quadratic functionals for the linear case--optimality and orthogonality properties of cg need no

longer hold. Some algorithms for the nonquadratic case have been proposed [e.g., 10, 11, 14,16, 20, 22] that preserve one or more of the quadratic case properties of finite termination, monotonic decrease of the two-norm of the error, conjugacy of directions of search, and orthogonality of the residuals. The method we discuss only approximates these properties, but is found to be effective for solving the discrete nonlinear elliptic partial differential equations of primary concern in our study. The method is closely related to the one studied in [3] for solving mildly nonlinear equations using a particular splitting.

We discuss in Sec. 1 several nonlinear conjugate gradient algorithms and in Sec. 2 same convergence properties. In Sec. 3 possible splitting choices for the approximating operator are described. 'A test problem for the minimal surface equation is discussed in Sec. 4, and experimental results for several splittings are summarized in Sec. 5. In Sec. 6 are given experimental results for a test problem for a mildly nonlinear equation, for which, in the corresponding linear case, the finite termination property of cg is crucial.

Much of the work reported here comprises a portion of the last-named author's doctoral dissertation at Stanford University [23]. We wish to thank the Mathematics Research Center, University of Wisconsin - Madison for providing the first two authors the stimulating and hospitable surroundings in which portions of the manuscript were prepared. We thank H. Glaz for preparing the computer program and for carrying out the numerical experiments for the second test problem, and R. Hockney and D. Warner for suggesting the problems from which this test problem

was derived. We thank also R. Bank, B. Buzbee, P. Swarztrauber, and R. Sweet, who made available to us their excellent computer programs for solving separable elliptic equations with fast direct methods. The work reported here was supported in part by the U.S. Energy Research and Development Administration, by the Fannie and John Hertz Foundation, and by the National Science Foundation.

1. Nonlinear Conjugate Gradient Algorithms

In the linear case, the generalized conjugate gradient method [9] solves the $N \times N$ positive-definite system of equations

$$Ax = b \tag{1}$$

or, equivalently, minimizes the quadratic form

$$f(x) = \frac{1}{2} x^T A x - x^T b \ . \tag{2}$$

Let M be a positive-definite symmetric $N \times N$ matrix, chosen to approximate A. Then for symmetric A the algorithm, as described in [9] in its alternative form, is:

Let $x^{(0)}$ be a given vector and define arbitrarily $p^{(-1)}$.

For k = 0, 1, . . .

(i) Calculate the residual $r^{(k)} = b - A x^{(k)}$

and solve

$$Mz^{(k)} = r^{(k)} \ . \tag{3}$$

(ii) Compute the parameter

$$b_k^{(1)} = \frac{z^{(k)^T} r^{(k)}}{z^{(k-1)^T} r^{(k-1)}}, \qquad k \geq 1,$$

$$b_0^{(1)} = 0$$

and the new direction $p^{(k)} = z^{(k)} + b_k^{(1)} p^{(k-1)}$.

(iii> Compute the parameter

$$a_k^{(1)} = \frac{z^{(k)^T} r^{(k)}}{p^{(k)^T} A p^{(k)}},$$

. and the new iterate $x^{(k+1)} = x^{(k)} + a_k^{(1)} p^{(k)}$.

In place of the parameters $a_k^{(1)}$ and $b_k^{(1)}$, one may use instead equivalent ones [18,26], such as

$$a_k^{(2)} = \frac{p^{(k)^T} r^{(k)}}{p^{(k)^T} A p^{(k)}}$$

or

$$b_k^{(2)} = - \frac{z^{(k)^T} A p^{(k-1)}}{p^{(k-1)^T} A p^{(k-1)}}.$$

Instead of computing the residual $r^{(k)}$ explicitly for $k \geq 1$, as in (i), it is often advantageous to compute it recursively as $r^{(k)} = r^{(k-1)} - a_{k-1} A p^{(k-1)}$.

The effectiveness of the algorithm (i, ii, iii) is discussed in [9] for cases in which A is a sparse matrix arising from the discretization of an elliptic partial differential equation and M is one of several sparse matrices arising naturally from A.

For the nonlinear case, we consider solving the system of equations

$$g(x) = 0 \tag{4}$$

arising from minimizing $f(x)$, where $g(x)$ is the gradient of $f(x)$. (For the linear case (1,2), $g(x) \equiv Ax - b$; in either case, $g(x)$ is the negative of the residual.) We assume that the Jacobian matrix J of (4) is positive-definite and symmetric, and, as for the linear case, we are interested in those situations for which (4) is a discrete form of an elliptic partial differential equation and, correspondingly, J is sparse.

The approximating matrix M for the linear case is chosen in [9] to be one of several positive-definite symmetric matrices approximating A naturally in some manner. For the nonlinear case, we consider related choices for M to approximate J, although sometimes M may not be linear, symmetric, or everywhere positive definite. We pattern after (i, ii, iii) the following algorithm (see also [3]).

Let $x^{(0)}$ be a given vector and define arbitrarily $p^{(-1)}$. For $k = 0,1,\ldots$

(Ni) Calculate

$$r^{(k)} = -g(x^{(k)})$$

and solve

$$Mz^{(k)} = r^{(k)} \tag{5}$$

.

(Nii) Compute $b_k = b_k^{(1)}$ or $b_k^{(2)}$,

where

$$b_k^{(1)} = \frac{z^{(k)^T} r^{(k)}}{z^{(k-1)^T} r^{(k-1)}} \quad , \quad k \geq 1$$

$$b_k^{(2)} = - \frac{z^{(k)^T} Jp^{(k-1)}}{p^{(k-1)^T} Jp^{(k-1)}} \quad ,$$

$$b_0 = 0 ,$$

and

$$p^{(k)} = z^{(k)} + b_k p^{(k-1)} .$$

(Niii) Compute $a_k = a_k^{(1)}$ or $a_k^{(2)}$,

where

$$a_k^{(1)} = \frac{z^{(k)^T} r^{(k)}}{p^{(k)^T} Jp^{(k)}}$$

$$a_k^{(2)} = \frac{p^{(k)^T} r^{(k)}}{p^{(k)^T} Jp^{(k)}}$$

and

$$x^{(k+1)} = x^{(k)} + a_k p^{(k)} .$$

The algorithm (i, ii, iii> for the linear case is generally iterated without any restarts (setting of b_k to zero for some value of $k > 0$); however the nonlinear algorithm (Ni, Nii, Niii) is usually restarted periodically to enhance convergence (see Secs. 2 and 5).

For some of the splittings we consider and in the presence of roundoff error, the numerator of $a_k^{(2)}$ may not be positive for some values of k. If it is not, then we find it convenient for these values of k to replace $p^{(k)}$ by its negative.

2. Convergence.

In the form (Ni, Nii, Niii) the algorithm of Sec. 1 cannot be guaranteed to converge. However, by introducing a line search to choose a_k so that f(x) is minimized along the line $x^{(k)} + a_k p^{(k)}$, by ensuring that M is positive definite, and by restarting the iteration periodically, convergence can be guaranteed. Convergence in this case can be shown by application of Zangwill's spacer step theorem [30], which states that if a closed algorithm with descent function f is applied infinitely often in the course of another algorithm that maintains the property

$$f(x^{(k+1)}) \leq f(x^{(k)})$$

for all k, and if

$$\{x: f(x) \leq f(x^{(0)})\}$$

is compact, then the composite algorithm converges.

We have the following:

Theorem 1. If the nonlinear conjugate gradient algorithm is modified to calculate a_k by

$$a_k = \min\{\hat{a}: f(x^{(k)} + \hat{a}p^{(k)}) \leq f(x^{(k)} + ap^{(k)}) \; \forall\, a \in (0,\infty)\}$$
$$\equiv a_k^{opt}$$

and if the iteration is restarted every a steps, then the algorithm is globally convergent (i.e., will converge from any initial point x_0) to x^* such that $f(x^*)$ is a minimum of $f(x)$ over E^N.

Proof. The sequence $\{f(x^{(k)})\}$ is monotone non-increasing, and every α steps we take a scaled steepest descent step. Since scaled steepest descent is a convergent algorithm, we can conclude by Zangwill's space step theorem that our algorithm converges.

This algorithm can be quite slow due to time consumed in line searches. In order to avoid a line search at every iteration, we impose additional constraints on the stepsize so that we can guarantee that f is monotone nonincreasing at each stage of the iteration. We have the following theorem:

Theorem 2. If the conjugate gradient iteration is restarted every α steps with the first step length in each cycle calculated by a line search, and if no conjugate gradient step causes an increase in the function $f(x)$, then the iteration will be globally convergent to x^* that minimizes $f(x)$.

Proof. By direct application of the spacer step theorem.

If the function $f(x^{(k)})$ is explicitly available, then we can accept our original definition of a_k if

$$f(x^{(k+1)}) \leq f(x^{(k)})$$

and do a line search if this test fails.

Lemma 1. Let a_k be chosen by the rule

$$a_k = \begin{cases} \ddot{a}_k^{(1)} \ (\text{or } a_k^{(2)}) & \text{if } f(x^{(k)} + a_k^{(1)} p^{(k)}) \leq f(x^{(k)}) \\ & (\text{or } f(x^{(k)} + a_k^{(2)} p^{(k)}) \leq f(x^{(k)})) \\ a_k^{opt} & \text{otherwise.} \end{cases}$$

Then $f(x)$ is monotone nonincreasing at the kth step.

If we have available only values of $g(x)$ and $J(x)$ at our iterates and not $f(x)$, we must make use of conditions that imply that f is decreasing.

Because f is convex,

$$p^{(k)T} g(x^{(k)} + ap^{(k)})$$

will be a monotone increasing function of a that is negative at $a = 0$ and is 0 at $a = a_k^{opt}$, the Point at which f attains its minimum on the line from $x^{(k)}$ in the direction $p^{(k)}$.

If a_{max} is chosen such that

$$a_{max} = \min\{a > a_k^{opt} : f(x^{(k)} + ap^{(k)}) = f(x^{(k)})\}$$

then we can deduce that $f(x^{(k+1)})$ will be less than $f(x^{(k)})$ if $0 < a < a_{max}$. Without further information (e.g., that obtained through a line search), we cannot calculate a_{max}. We can, however, easily verify whether $a < a_k^{opt}$ and this will give us a sufficient condition for convergence:

Lemma 2. Let a_k be chosen by the following rule:

$$a_k = \begin{cases} a_k^{(1)} \quad (\text{or } a_k^{(2)}) & \text{if } p^{(k)^T} g(x^{(k)} + \varepsilon_k^{(1)} p^{(k)}) \leq 0 \\ & (\text{or } p^{(k)^T} g(x^{(k)} + \varepsilon_k^{(2)} p^{(k)}) \leq 0) \\ a_k^{opt} & \text{otherwise.} \end{cases}$$

Then $f(x^{(k+1)}) \leq f(x^{(k)})$.

If we have information on the curvature of the function f, we can derive an alternate condition. Consider the Taylor series expansion of f at $x^{(k+1)}$:

$$(6) \qquad f(x^{(k+1)}) - f(x^{(k)}) = a_k g(x^{(k)})^T p^{(k)} + \frac{1}{2} a_k^2 p^{(k)^T} J(w) p^{(k)}$$

where w is a point between $x^{(k)}$ and $x^{(k+1)}$, and suppose we know that

$$0 < d \leq \lambda(J(x))$$

for all x in a convex set including all iterates. Then the right hand side of (6) can be guaranteed to be negative if

$$a_k < \frac{-2\, g(x^{(k)})^T p^{(k)}}{d\, p^{(k)T} p^{(k)}} .$$

This gives an alternate condition for convergence:

Lemma 3. Let a_k be chosen by the rule

$$a_k = \begin{cases} a_k^{(1)} \quad (\text{or } a_k^{(2)}) & \text{if } a_k^{(1)} < \dfrac{-2\, g(x^{(k)})^T p^{(k)}}{d\, p^{(k)T} p^{(k)}} \\ & (\text{or } a_k^{(2)} < \dfrac{-2\, g(x^{(k)})^T p^{(k)}}{d\, p^{(k)T} p^{(k)}}) \\ a_k^{opt} & \text{otherwise .} \end{cases}$$

Then f(x) is nonincreasing at that step.

In general, each of the conditions in Lemma 1 through Lemma 3 is quite restrictive, but verifying any one is sufficient for descent at a given step. Thus an algorithm might incorporate facilities for testing each of the conditions successively if the preceding ones did not verify descent. This would keep the additional operational overhead for the algorithm low.

Notice that we do not need constraints on the b_k (other than $b_k \neq 0$), the parameters that determine the step direction, in order to guarantee convergence. In our numerical experiments (Sec. 5), we have observed that for the test problem considered here the algorithm is less sensitive to choices of the parameter b_k than to choices of a_k. It was found, on the other hand, for the problem studied in [25] with small N (~ 100), dense Jacobian, and exact line searches for a, that the cg algorithm could be quite sensitive to the choice for b.

3. Choice of Splitting Operator

We consider several choices for the splitting operator M. All the choices attempt to approximate the Jacobian J with an operator that is computationally easier to invert.

First, we consider choices related to the nonlinear block successive overrelaxation method, which has been found to be efficient for solving nonlinear elliptic equations [6,7,24,27]. This method obtains from the residual $r^{(k)}$ an increment $z^{(k)} = M^{-1} r^{(k)}$ that is added to $x^{(k)}$ to obtain a new approximation

$$x^{(k+1)} = x^{(k)} + z^{(k)} .$$

Equation (7) is the underlying first-order iteration that is accelerated by means of the conjugate gradient algorithm in (Ni, Nii, Niii).

Let x, $g(x)$, and J be subdivided into blocks, for example those corresponding to rows of points on a rectangular mesh for the finite difference approximation to a partial differential equation

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix}, \qquad g(x) = \begin{pmatrix} g_1 \\ g_2 \\ \vdots \\ g_m \end{pmatrix}, \qquad J = \begin{pmatrix} J_{11} & J_{12} & \cdots & J_{1m} \\ J_{21} & J_{22} & \cdots & J_{2m} \\ \vdots & & & \vdots \\ J_{m1} & J_{m2} & \cdots & J_{mm} \end{pmatrix}.$$

For standard discretization of elliptic equations, J has small block bandwidth, and its blocks are sparse. In two dimensions on a rectanglar mesh, for the nine-point discretization we shall consider here for the minimal surface equation, J is block tridiagonal with tridiagonal blocks [6].

We consider first the one-step block successive overrelaxation-Newton (BSOR-Newton) iteration [24]. For it, the $(k+1)$th approximation to x_j is obtained from the (k)th by

$$x_j^{(k+1)} = x_j^{(k)} - \omega J_{jj}^{-1} g_j, \qquad j = 1,2,\ldots,m, \tag{8}$$

where $g(x)$ and J are evaluated at the latest values for x, and ω is an acceleration parameter. If we partition the residual $r = -g(x)$ in the same manner as $g(x)$, then we can write (8) as

$$x_j^{(k+1)} = x_j^{(k)} + \omega J_{jj}^{-1} r_j, \qquad j = 1,2,\ldots,m$$

to correspond with our earlier notation. The banded, positive-definite system of equations

$$J_{jj} z_j = \omega r_j$$

can be solved efficiently in a numerically stable manner using Gaussian elimination, without pivoting, or Cholesky factorization.

For linear problems, BSOR is not suitable for use with conjugate gradient acceleration because its iteration matrix is not similar to a symmetric one. A symmetrized variant is suitable, however. This variant corresponds to ordering the equations alternately from blocks $j = 1$ to $j = m$ for one sweep and then from blocks $j = m$ to $j = 1$ for the next sweep; it is termed block symmetric SOR (BSSOR). For BSSOR the solution of $Mz^{(k)} = r^{(k)}$ reduces to the solution of

$$\frac{1}{\omega(2-\omega)} (D + \omega L) D^{-1} (D + \omega U) z^{(k)} = r^{(k)}, \quad 0 < \omega < 2,$$

where D is the block diagonal of A, L (U) is a strictly block lower (upper) triangular matrix, and $A = L + D + U$. Conjugate gradient acceleration has been found to be particularly effective for BSSOR because of the distribution of the eigenvalues of the iteration matrix [1,12,17].

For the nonlinear case we consider the correspondingly symmetrized variant of the one-step BSOR-Newton iteration, and we denote it by BSSOR-Newton.

We consider also another extension of the BSSOR method. For the case in which the calculation of the elements of J_{jj} in (8) is costly, the symmetric form of the one-step Newton-BSOR method [24] can be more efficient. This algorithm applies a back and forth sweep of BSSOR to the Newton iteration step

$$J(x^{(k)})z = -g^{(k)} = r^{(k)}$$

to obtain the increment $z^{(k)}$ in (7). As we did above for A, we write $J(x^{(k)}) = \bar{L} + \bar{D} + \bar{U}$, where $\bar{D}$ is the block diagonal of $J(x^{(k)})$ and $\bar{L}$ ($\bar{U}$) is strictly block lower (upper) triangular. Then for the choice of zero as initial approximation for z, and for z partitioned in the same manner as r, the back and forth BSSOR sweep is

forward sweep:

$$\tilde{z}_j^{(k)} = \omega J_{jj}^{-1}(r_j^{(k)} - [\bar{L}\tilde{z}^{(k)}]_j) , \qquad j = 1,2,\ldots,m$$

followed by

backward sweep:

$$z_j^{(k)} = \tilde{z}_j^{(k)} + \omega J_{jj}^{-1}(r_j^{(k)} - [\bar{L}\tilde{z}^{(k)} + \bar{D}\tilde{z}^{(k)} + \bar{U}z^{(k)}]_j) \qquad j = m,m-1,\ldots,1 .$$

Here J and r are evaluated at $x^{(k)}$. Note that the most recently obtained values of z are used in the computation of $[Jz]_j$ on the right hand sides.

Either BSSOR-Newton or Newton-BSSOR are reasonable choices for the operator M for the conjugate gradient iteration. When $x^{(k)}$ approaches the solution x^* the Jacobian approaches $J(x^*)$ so that M, which changes from iteration from iteration, approaches a limit also. As a possible alternative, one could fix M for a number of iterations by keeping J fixed at a value from an earlier iteration, updating only occasionally.

Another choice for M that we consider approximates the Jacobian matrix directly. We choose M to be the approximate sparse LDL^T (Cholesky) factorization of the Jacobian, as developed by Meijerink and van der Vorst for the solution of linear elliptic problems [21]. The matrix L is chosen with a sparsity pattern resembling that of the lower triangular part of J, and the elements are obtained systematically from J by enforcing the sparsity structure as the approximate factorization proceeds. For linear problems this splitting has been found to yield an iteration matrix with eigenvalues favorably distributed for conjugate gradient acceleration [21].

For "M" matrices, Meijerink and van der Vorst proved in [21] that the approximate factorization can be carried out in a stable manner. For the problems we consider, the Jacobian may not be such a matrix; however, we did not encounter difficulty in carrying out the approximate factorization for our test cases.

Finally, we consider approximating the Jacobian by a discretized separable operator, for which fast direct methods can be used [2,5,13,19]. For our test problems we consider as a choice for M the discrete Helmholtz operator, possibly scaled by the diagonal of the Jacobian.

4. First Test Problem

The first test problem, for which the above splittings are compared, is that of solving numerically the minimal surface equation on a rectangle. This problem was used previously for studying the behavior of nonlinear relaxation methods [6,7] and is of interest because of its strong nonlinearity. The minimal surface equation arises in finding a single-valued twice continuously differentiable function $v(x,y)$ that attains given values on the boundary of a region R and minimizes the area integral over R [15]. This equation is

$$\text{div}(\gamma \nabla v) = 0 \qquad \text{on } R , \tag{9}$$

where $\gamma = (1 + |\nabla v|^2)^{-1/2}$, with the boundary condition

$$v = s(x,y \qquad \text{on } \partial R . \tag{10}$$

We consider the domain

$$0 \leq x \leq 2, \qquad 0 \leq y \leq 1 .$$

If $s(x,y)$ is symmetric about $x = 1$, then the problem need only be solved on the unit square with the symmetry condition

$$\frac{\partial v}{\partial x} = 0 \qquad \text{on } x = 1. \tag{11}$$

We discretize (9,10,11) in the same manner as is done in [6]. A square mesh of size $h = 1/n$ is placed on the domain, and u_{ij} denotes the approximation to $v(x,y)$ at the mesh point $x = ih$, $y = jh$. Then at the interior points we obtain, after multiplication by $-2h^2$,

$$(12)\quad g_{i,j} \equiv \gamma_{\bar{i},\bar{j}}(2u_{i,j} - u_{i-1,j} - u_{i,j-1}) + \gamma_{\bar{i}+1,\bar{j}}(2u_{i,j} - u_{i+1,j} - u_{i,j-1})$$

$$+ \gamma_{\bar{i},\bar{j}+1}(2u_{i,j} - u_{i-1,j} - u_{i,j+1})$$

$$+ \gamma_{\bar{i}+1,\bar{j}+1}(2u_{i,j} - u_{i+1,j} - u_{i,j+1}) = 0\ ,$$

$$i = 1,2,\ldots,n-1; \quad j = 1,2,\ldots,n-1,$$

where $\gamma_{\bar{i},\bar{j}} = \gamma(|\nabla u|^2_{\bar{i},\bar{j}})$ approximates $\gamma(|\nabla v|^2)$ at $((i-1/2)h, (j-1/2)h)$, with

$$|\nabla u|^2_{\bar{i},\bar{j}} = \frac{1}{2h^2}\left[(u_{i,j} - u_{i-1,j})^2 + (u_{i,j} - u_{i,j-1})^2\right.$$

$$\left. + (u_{i,j-1} - u_{i-1,j-1})^2 + (u_{i-1,j} - u_{i-1,j-1})^2\right]$$

$$i = 1,2,\ldots,n; \quad j = 1,2,\ldots,n\ .$$

Along the symmetry boundary we obtain

$$(13)\qquad g_{n,j} = \gamma_{\bar{n},\bar{j}}(2u_{n,j} - u_{n-1,j} - u_{n,j-1})$$

$$+ \gamma_{\bar{n},\bar{j}+1}(2u_{n,j} - u_{n-1,j} - u_{n,j+1}) = 0 \quad , \qquad j = 1,2,\ldots,n-1.$$

In (12,13) we do not group together explicitly the coefficients of $u_{i,j}$ and of $u_{i\pm1,j\pm1}$, as is customary for the linear case, in order to emphasize that the problem is nonlinear and that the $\gamma_{\bar{i},\bar{j}}$ are themselves functions of the u_{ij}.

The Jacobian matrix J is given by

$$J = \left\{ \frac{\partial g_{i,j}}{\partial u_{k,\ell}} \right\} \quad ,$$

a positive-definite symmetric matrix that is block tridiagonal, with each block being tridiagonal. For this test problem, the calculation of $y'_{\bar{i},\bar{j}} \equiv d\gamma_{\bar{i},\bar{j}}/d|\nabla u|^2_{\bar{i},\bar{j}}$ and of J can be carried out with only a modest amount of computational effort in addition to that required for calculation of the $g_{i,j}$.

5. Experimental Results for the First Test Problem

The test problem of Sec. 4 was solved numerically for the same boundary data as was considered in [6,7],

$$v = 0 \qquad \text{on} \quad x = 0 \quad \text{and} \quad y = 1,$$

$$v = \sin \frac{\pi x}{2} \qquad \text{on} \quad y = 0 \ ,$$

and the symmetry condition (11). The following algorithms discussed in Sees. 2 and 3 were used:

I. One-step block SOR-Newton

II. One-step block SSOR-Newton

III. One-step block Newton-SSOR

IV. Conjugate gradient algorithm (with M the identity matrix)

V. Conjugate gradient algorithm with M the BSSOR-Newton operator.

VI. Conjugate gradient algorithm with M the Newton-BSSOR operator.

VII. Conjugate gradient algorithm with M the Meijerink-van der Vorst approximate sparse factorization of J, renewed every restart. The sparsity pattern of the approximate factor is chosen to be identical with that of the lower triangular part of J (the ICCG(0) variant [21]).

VIII. Conjugate gradient algorithm with $M = \Delta, -2h^2$ times the discrete Laplace 5 point operator [$\gamma \equiv 1$ in (12,13)].

IX. Conjugate gradient algorithm with $M = D^{1/2}(\Delta + \kappa I)D^{1/2}$, where Δ is the operator of VIII and D is the diagonal of J, renewed every iteration. κ is a constant chosen so that the average of three sample values of the diagonal of J equals the diagonal of M.

For the conjugate gradient algorithms each test used either $a^{(1)}$ or $a^{(2)}$ and either $b^{(1)}$ or $b^{(2)}$, with no line searches and none of the convergence safeguards developed in Lemmas 1-3 of Section 2.

The algorithms are compared in terms of operation counts required to decrease the residual to specified values. In Table 1 are given approximate operation counts for various phases of the iteration. In Tables 2 and 3 are given the results of experiments carried out in double precision with FORTRAN programs on the IBM 360/168 computer for grids with spacing $h = 1/16$ and $h = 1/32$, for initial approximation $u^{(0)} \equiv 0$. The supplemental tables give the results as obtained originally in terms of number of iterations, which were converted subsequently to multiplication counts by means of the cost factors. The cost factors relate to our test programs, which are not generally optimal but nevertheless should give a reasonable basis for comparison. Note that an iteration of one of the SOR algorithms requires half the number of operations of an iteration of the corresponding symmetric form, which requires both one forward and one backward sweep.

The columns, from left to right, in the tables correspond to successively larger values of $\|r^{(k)}\|_2$, the two-norm of the residual. The initial residual $\|r^{(0)}\|_2$ is approximately 0.47 for the coarser mesh and 0.34 for the finer one. (Recall that the residuals are for (12,13), which are obtained from (9) after multiplication by a factor proportional to h^2.)

From the tables, one observes that for this test problem the conjugate gradient algorithm with discrete Laplace operator splitting, with or without shift or Jacobian diagonal scaling,, produces an algorithm favorably competitive with nonlinear block SOR in terms of operation counts. On the larger problem, the conjugate gradient algorithm with one of the nonlinear BSSOR splittings is also faster than nonlinear BSOR.

TABLE 1

COST FACTORS PER STEP FOR MINIMAL SURFACE ALGORITHMS

(1) $n \times (n-1)$ unknowns, $n(n-1) = N$

(2) Costs consider only multiplications, divisions, and square roots and include only the highest order terms in N.

(3) Conjugate gradient overhead is 5N.

(4) SOR Overhead is N.

(5) The cost of calculating γ_{ij} is 3N.

(6) The cost of forming g is 4N (given γ_{ij}).

(7) The cost of calculating γ'_{ij} is 2N (given γ_{ij}).

(8) 20N operations are needed to calculate J (given γ_{ij}, γ'_{ij}).

(9) 12N operations are needed to calculate only the tridiagonal portion of J. (8N for the diagonal only.)

(10) To factor and solve a tridiagonal system takes 5N operations.

(11) To form Jp takes 9N operations given J.

TABLE 1 Continued

Action cost	(3) 5N	(4) 1N	(5) 3N	(6) 4N	(7) 2N	(8) 20N	(9) 12N	(10) 5N	(11) 9N	Scaling N	Total Operations Per Point
I BSOR-Newton	--	1	2	1	2	--	1	1	--	--	32
II BSSOR-Newton	--	2	3	2	3	--	2	2	--	--	59
III Newton-BSSOR	--	2	1	1	1	1	--	2/1*	2	--	57
IV CG	1	--	1	1	1	1	--	--	1	--	43
V CG with BSSOR-Newton	1	--	1**	1**	1**	1**	--	--	1	59	73-102
VI CG with Newton-BSS OR	1	--	--	--	--	--	--	--	1	57	71
VII CG with sparse LDL^T	1	--	1	1	1	1	--	--	1	7	50
VIII CG with Laplacian	1	--	1	1	1	1	--	--	1	$3 \log_2 n$	$43 + 3 \log_2 n$
IX CG with Laplacian and J diagonal and shift	1	--	1	1	1	1	--	--	1	$3 + 3 \log_2 n$	$46 + 3 \log_2 n$

*2 solves and 1 factorization

**can be eliminated once we are near the solution

TABLE 2

COMPARISON OF ALGORITHMS BY NUMBER OF MULTIPLICATIONS PER MESH POINT TO OBTAIN A RESIDUAL $\|r^{(k)}\|_2 \leq$ EPS: h = 1/16 MINIMAL SURFACE EQUATION

Algorithm (Cost Factor)	ω	EPS = 10^{-5}	10^{-4}	10^{-3}	10^{-2}	10^{-1}
I BSOR-Newton (32)	1.1	5824	4448	3072	1696	320
	1.2	4704	3616	2496	1376	288
	1.3	3776	2880	1536	1120	224
	1.4	2976	2272	1600	896	192
	1.5	2240	1728	1216	704	192
	1.6	1568	1248	896	544	160
	1.7	1056	832	640	448	192
	1.8	1536	1216	960	544	288
	1.9	3264	2560	1920	1184	512
II BSSOR-Newton (59)	1.1	5841	4484	3127	1770	413
	1.2	4897	3717	2596	1475	354
	1.3	4130	3127	2183	1298	295
	1.4	3451	2655	1888	1062	295
	1.5	3009	2301	1652	944	236
	1.6	2655	2065	1475	826	236
	1.7	2596	2006	1416	826	236
	1.8	3068	2419	1711	1003	295
III New-ton-BSSOR (57)	1.1	5643	4384	3078	1767	456
	1.2	4731	3648	2565	1482	399
	1.3	3990	3078	2166	1254	399
	1.4	3420	2622	1881	1083	342
	1.5	2964	2280	1596	969	342
	1.6	2622	2052	1425	855	342
	1.7	2565	1995	1368	7%	342
	1.8	2964	2280	1596	969	344
	1.9	5358	4104	2850	1653	570
IV CG (43)	$a^{2}b^{1}$	11997	9331	6794	4257	1677
	$a^{2}b^{2}$	11782	9331	6751	4214	1763
	$a^{1}b^{1}$	10492	7869	5762	3612	1075
	$a^{1}b^{2}$	10444	7869	5762	3612	1548

TABLE 2 Continued

Algorithm (Cost Factor)		ω	EPS=10^{-5}	10^{-4}	10^{-3}	10^{-2}	10^{-1}
V CG+							
BSSOR-Newton		1.1	2346	2244	2040	1632	918
a^2b^1		1.2	2346	2142	1938	1530	918
Restart 9		1.3	2244	2142	1836	1428	918
(102)		1.4	2244	2040	1632	1428	918
		1.5	2142	1938	1632	1326	1020
		1.6	2040	1836	1632	1224	918
		1.7	2040	1836	1530	1224	1020
		1.8	2244	2040	1836	1326	1020
		1.9	no convergence				
Restart 13		1.65	1632	1326	1122	918	5 10
V CG+							
BSSOR-Newton	a^2b^1		2040	1836	1530	1224	1020
ω = 1.7	a^2b^2		1530	1326	1122	816	510
(102)	a^1b^1		1734	1530	1326	1020	612
	a^1b^2		1734	1428	1224	1020	612
VI CG +							
Newton-BSXOR		1.1	1704	1562	1491	1136	710
a^1b^1		1.2	1633	1441	1349	1065	634
(71)		1.3	1562	1441	1349	1065	639
		1.4	1562	1420	1207	994	639
		1.5	1441	1278	1136	994	639
		1.6	1633	1420	1278	1065	7 10
		1.7	1846	1633	1420	1278	781
		1.8	2059	1846	1633	1420	1207
		1.9	2201	1988	1846	1491	1207
VI CG +							
Newton-BSSOR	a^2b^1		1562	1420	1207	994	7 10
ω = 1.4	a^2b^2		1349	1207	1065	781	5%
(71)	a^1b^1		lb46	1633	1420	1278	781
	a^1b^2		1562	1278	1065	923	568

TABLE 2 Continued

Algorithm (Cost Factor)		EPS=10^{-5}	10^{-4}	10^{-3}	10^{-2}	10^{-1}
VII CG +	a^2b^1			1318	1068	562
sparse LDL^T	a^2b^2	1524	1374	1068	1112	462
(50 + 6 per	a^1b^1	1374	1218	1018	662	462
restart)	a^1b^2	1424	1268	968	762	462
VIII CG +	a^2b^1	1045	1045	880	605	220
Laplacian	a^2b^2	880	880	715	385	220
(55)	a^1b^1	825	825	715	385	220
	a^1b^2	825	825	715	440	220
IX CG +	a^2b^1	754	754	696	522	290
Laplacian +	a^2b^2	638	638	522	406	290
J diagonal	a^1b^1	696	696	580	406	232
+ shift	a^1b^2	754	754	638	522	290
(58)						

SUPPLEMENT TO TABLE 2

COMPARISON OF ALGORITHMS BY NUMBER OF ITERATIONS TO OBTAIN A RESIDUAL $\|r^{(k)}\|_2 \leq$ EPS,

h = 1/16

Algorithm	ω	EPS=10^{-5}	10^{-4}	10^{-3}	10^{-2}	10^{-1}
I BSOR-Newton	1.1	182	139	96	53	10
	1.2	147	113	78	43	9
	1.3	118	90	48	35	7
	1.4	93	71	50	28	6
	1.5	70	54	38	22	6
	1.6	49	39	28	17	5
	1.7	33	26	20	14	6
	1.8	48	38	30	17	9
	1.9	102	80	60	37	16
II BSSOR-Newton	1.1	94	76	53	30	7
	1.2	83	63	44	25	6
	1.3	70	53	37	22	5
	1.4	59	45	32	18	5
	1.5	51	39	28	16	4
	1.6	45	35	25	14	4
	1.7	44	34	24	14	4
	1.8	52	41	29	17	5
III Newton-BSSOR	1.1	94	77	54	31	8
	1.2	83	64	45	26	7
	1.3	70	54	38	22	7
	1.4	60	46	33	19	6
	1.5	52	40	28	17	6
	1.6	46	36	25	15	6
	1.7	45	35	24	14	6
	1.8	52	40	28	17	7
	1.9	94	72	50	29	10

SUPPLEMENT TO TABLE 2 Continued

Algorithm		ω	EPS=10^{-5}	10^{-4}	10^{-3}	10^{-2}	10^{-1}
IV CG	a^2b^1		279	217	158	49	39
	a^2b^2		274	217	157	98	41
	a^1b^1		244	183	134	84	25
	a^1b^2		243	183	134	84	36
V CG+		1.1	23	22	20	16	9
BSSOR-Newton		1.2	23	21	19	15	9
a^2b^1		1.3	22	21	18	14	9
Restart 9		1.4	22	20	16	14	9
		1.5	21	19	16	13	10
		1.6	20	18	16	12	9
		1.7	20	18	15	12	10
		1.8	22	20	18	13	10
		1.9	no convergence				
Restart 13		1.65	16	13	11	9	5
V CG +							
BSSOR-Newton	a^2b^1		20	18	15	12	10
ω = 1.7	a^2b^2		15	13	11	8	5
	a^1b^1		17	15	13	10	6
	a^1b^2		17	14	12	10	6
-VI CG +		1.1	24	22	21	16	10
Newton - BSSOR		1.2	23	21	19	15	9
a^1b^1		1.3	22	21	19	15	9
		1.4	22	20	17	14	9
		1.5	21	18	16	14	9
		1.6	23	20	18	15	10
		1.7	26	23	20	18	11
		1.8	29	26	23	20	17
		1.9	31	28	26	21	17

SUPPLEMENT TO TABLE 2 Continued

Algorithm		EPS=10^{-5}	10^{-4}	10^{-3}	10^{-2}	10^{-1}
VI CG +	a^2b^1	22	20	17	14	10
Newton-BSSOR	a^2b^2	19	17	15	11	8
ω = 1.4	a^1b^1	26	23	20	18	11
	a^1b^2	22	18	15	13	8
VII CG +	a^2b^1			26	21	11
sparse LDL^T	a^2b^2	30	27	21	16	9
	a^1b^1	27	24	20	13	9
	a^1b^2	28	25	19	15	9
VIII CG +	a^2b^1	19	19	16	11	4
Laplacian	a^2b^2	16	16	13	7	4
	a^1b^1	15	15	13	7	4
	a^1b^2	15	15	13	8	4
IX CG +	a^2b^1	13	13	12	9	5
Laplacian	a^2b^2	11	11	9	7	5
+ J diagonal	a^1b^1	12	12	10	7	4
+ shift	a^1b^2	13	13	11	9	5

TABLE 3

COMPARISON OF ALGORITHMS BY NUMBER OF MULTIPLICATIONS PER MESH POINT TO OBTAIN A RESIDUAL $\|r^{(k)}\|_2 \leq$ EPS: h = 1/32 MINIMAL SURFACE EQUATION

Algorithm (Cost Factor)		ω	EPS=10^{-8}	10^{-7}	10^{-6}	10^{-5}	10^{-4}	10^{-3}	10^{-2}	10^{-1}
I	BSOR-Newton	1.3				>11200	105 60	6944	3328	192
	(32)	1.4				>11200	8384	5536	2656	160
		1.5		>11200	10912	8704	6496	4288	2080	160
		1.6	>11200	9696	8096	6464	4832	3232	1600	160
		1.7	7732	6624	5536	4448	3360	2272	1184	160
		1.8	4000	3328	2976	2464	1952	1440	864	192
		1.4	5408	4832	4096	3296	2752	2048	1152	640
II	BSSOR-Newton	1.5	15812	13570	11328	9086	678.5	4543	2301	236
	(59)	1.6	12626	10856	9086	72.57	5487	3658	1888	236
		1.7	lo148	8673	7257	5841	4425	2950	1534	236
		1.75	9204	7670	6608	5310	4012	2714	1416	236
		1.8	8673	7434	6254	5015	3776	2596	1416	236
		1.85	8791	7552	6313	5074	3894	2655	1534	295
III	Newton-BSSOR	1.4	19038	16302	13566	10887	8151	5472	2736	285
	(57)	1.5	15276	13110	10944	8778	6555	4389	2223	285
		1.6	12198	10431	8721	7011	5244	3534	1824	285
		1.7	9747	8379	7011	5643	4275	2850	1482	285
		1.8	8322	7125	5928	4788	3591	2394	1254	399
		1.9	10146	8721	7239	5814	4389	2964	1653	570

TABLE 3 Continued

	Algorithm (Cost Factor)	ω	EPS=10^{-8}	10^{-7}	10^{-6}	10^{-5}	10^{-4}	10^{-3}	10^{-2}	10^{-1}
V	CG +	1.5	3570	3366	3060	2958	2550	2040	1734	816
	BSSOR-Newton	1.6	4692	4488	4182	3978	3468	3060	2754	2346
	Restart 13	1.7	3570	3366	3162	2856	2346	1938	1632	1224
	a^2b^2	1.75	4182	3876	3468	3162	2958	2754	2346	1938
	(102)	1.8	3774	3468	3162	2856	2652	2142	1438	1428
		1.85	3978	3468	3162	2856	2550	2142	1836	1122
		1.9	no convergence							
VI	CG +	1.4	3053	2840	2556	2272	2130	1704	1278	852
	Newton-BSSOR	1.5	2627	2414	2272	2130	1917	1491	1278	1065
	a^2b^2	1.6	2769	2556	2485	2130	1775	1633	1278	710
	Restart 13	1.7	2911	2769	2343	2272	2130	1917	1633	1065
	(71)	1.8	3053	2840	2627	2343	2130	1917	1704	1278
		1.9	no convergence							
IX	CG + Laplacian	a^2b^1	2623	2440	2318	2135	1952	1769	1403	915
	+ J diagonal	a^1b^1	2623	2501	2379	2196	2013	1830	1342	915
	+ shift									
	Restart 16									
	(61)									
	Restart 9	a^2b^1	2074	1891	1647	1525	1281	1159	915	549

SUPPLEMENT TO TABLE 3

COMPARISON OF ALGORITHMS BY NUMBER OF ITERATIONS TO OBTAIN A RESIDUAL $\|r^{(k)}\|_2 <$ EPS,

h = 1/32

Algorithm		EPS=10^{-8}	10^{-7}	10^{-6}	10^{-5}	10^{-4}	10^{-3}	10^{-2}	10^{-1}
I BSOR-Newton	1.3				>350	330	217	104	6
	1.4				>350	262	173	83	5
	1.5		>350	341	272	203	134	65	5
	1.6	>350	303	253	202	151	101	50	5
	1.7	241	207	173	139	105	71	37	5
	1.8	125	104	93	77	61	45	27	6
	1.9	169	151	128	103	86	64	36	20
II BSSOR-Newton	1.5	268	230	192	154	115	77	39	4
	1.6	214	184	154	123	93	62	32	4
	1.7	172	147	123	99	75	50	26	4
	1.75	156	130	112	90	68	46	24	4
	1.8	147	126	106	85	64	44	24	4
	1.85	149	128	107	86	66	45	26	5
III Newton-BSSOR	1.4	334	286	238	191	143	96	48	5
	1.5	268	230	192	154	115	77	39	5
	1.6	214	183	153	123	92	62	32	5
	1.7	171	147	123	99	75	50	26	5
	1.8	146	125	104	84	63	42	22	7
	1.9	178	153	127	102	77	52	29	10

SUPPLEMENT TO TABLE 3 Continued

	Algorithm		ω	ESP=10-8	10-7	10-6	10^{-5}	10-4	10^{-3}	10^{-2}	10^{-1}
V	CG+		1.5	35	33	30	29	25	20	17	8
	BSSOR-Newton		1.6	46	44	41	39	34	30	27	23
	Restart 13		1.7	35	33	31	28	23	19	16	12
		a^2b^2	1.75	41	38	34	31	29	27	23	19
			1.8	37	34	31	28	26	21	19	14
			1.85	39	34	31	28	25	21	18	11
			1.9	no convergence							
VI	CG +		1.4	43	40	36	32	30	24	18	12
	Newton-BSSOR		1.5	37	34	32	30	27	21	18	15
		a^2b^2	1.6	39	36	35	30	25	23	18	10
	Restart 13		1.7	41	39	33	32	30	27	23	15
			1.8	43	40	37	33	30	27	24	18
			1.9	no convergence							
IX	CG + Laplacian	a^2b^1		43	40	38	35	32	29	23	15
	+ J diagonal	a^1b^1		43	41	39	36	33	30	22	15
	+ shift										
	Restart 16										
	Restart 9	a^2b^1		34	31	27	25	21	19	15	9

As has been observed also for linear cases [1,12,17], the symmetric SOR algorithms accelerated by cg are less sensitive to the choice of relaxation parameter ω than are the corresponding unaccelerated SOR algorithms.

Of course, as with any higher order method, the storage requirements of cg are greater than those of the basic unaccelerated iteration. It should be noted also, that for nonrectangular domains more operations would be required to obtain the solution of (5) for cases VIII and IX.

The results for initial approximations other than $u^{(0)} \equiv 0$ are not included in the tables: however, there were indications in our experiments that poorer initial approximations could result in divergence for some of the methods, without the safeguards of Section 2, as would be the case also for the unaccelerated nonlinear SOR methods [6,7]. In the experiments, the algorithms exhibited some sensitivity to the length of the conjugate gradient cycle between restarts. Restarting every 9 iterations, which is the case reported in Table 2, seemed effective for the coarser grid. For the finer grid 13 to 16 iterations were better.

Limitations of time prevented us from investigating Case VII for the finer grid and from investigating either a variant of the LDL^T approximate factorization allowing one more subdiagonal nonzero band in L (analogous to ICCG(3) in [21]) or a variant utilizing block techniques developed recently in [29]. Either of these variants might yield results superior to those reported for Case VII, as they have been found generally to be more efficient for linear problems.

We conclude from these experiments that the generalized conjugate gradient algorithm, with modifications to ensure convergence, holds promise of being favorably competitive with relaxation techniques for solving strongly nonlinear elliptic problems.

6. Second Test Problem

For the second test problem we consider a mildly nonlinear equation arising from the theory of semiconductor devices,

$$-v_{xx} - v_{yy} + (1 - e^{-5x})e^{v} = 1 . \tag{14}$$

Equation (14) is to be solved on the unit square subject to the boundary conditions

$$\begin{aligned} &\text{on } x = 0: && v = 0 \\ &\text{on } x = 1: && v = 1 \\ &\text{on } y = 0: && \partial v/\partial y = 0 \\ &\text{on } y = 1: && \begin{cases} \partial v/\partial y = 0 & \text{for } 0 < x \le a < 1/2 \\ v = -1 & \text{for } a < x < 1-a \\ \partial v/\partial y = 0 & \text{for } 1-a \le x < 1 . \end{cases} \end{aligned} \tag{15}$$

Of particular interest is the mixed boundary condition on the edge $y = 1$, as it would preclude the immediate use of one of the basic fast direct algorithms for solving (5) if M were chosen to be a discrete Helmholtz operator with boundary conditions (15).

We place a uniform mesh of width h on the unit square and denote the approximation to $v(x,y)$ at the mesh point $x = ih$, $y = jh$ by $u_{i,j}$. Then at an interior point we obtain, using the standard five-point discretization,

$$(16) \qquad \frac{1}{h^2}\left(-u_{i,j-1} - u_{i-1,j} + 4u_{i,j} - u_{i+1,j} - u_{i,j+1}\right) + \left(1-e^{-5x_i}\right)\exp(u_{i,j}) = 1.$$

At the Neumann boundary points the difference equations specialize in the usual manner, as in Section 4.

We choose for M the equivalent discretization of the Helmholtz operator H

$$Hv \equiv -v_{xx} - v_{yy} + \kappa v ,$$

but with the boundary condition along $y = 1$ in (15) replaced by

$$(17) \qquad \frac{\partial v}{\partial y} = 0 \qquad \text{on} \quad y = 1 \quad \text{for} \quad 0 < x < 1.$$

This permits the use of standard fast direct methods for carrying out the numerical solution of $Mz = r$. Also, we augment the system (16) with the equations

$$(18) \qquad \frac{4}{h^2} u_{i,j} + \left(1 - e^{-5x_i}\right)\exp(u_{i,j}) = -\frac{4}{h^2} + \left(1-e^{-5x_i}\right)\exp(-1)$$

for the Dirichlet points on $y = 1$, so that the Jacobian of the augmented system and M have the same rank. The constant κ is chosen to be 1,

a value that is meant to approximate $(1 - e^{-5x})e^{v}$ on the square, so that M approximates, in this manner, the Jacobian of the augmented system (16,18).

This choice for M does not approximate the Jacobian well in norm, because of the differing boundary conditions on y = 1. However, because the number of mesh points at which the boundary conditions differ is small, a corresponding linear problem--say with $(1 - e^{-5x})e^{v}$ in (14) replaced by v

$$-v_{xx} - v_{yy} + v = 1 , \tag{19}$$

with corresponding replacements in (16) and (18), and with $\kappa = 1$ in M--will converge completely in only a moderate number of iterations. At most $2p + 3$ iterations are required in this (linear) case to reach the solution (in the absence of round-off errors), where p is the number of Dirichlet boundary points on y = 1, because of the finite termination property of cg [9]. For our test problem, our interest is in obtaining an indication of the degree to which the introduction of a mild nonlinearity alters the convergence rate from that for the corresponding linear problem (see also [4]).

In Table 4 are given the observed number of iterations at which the residual norm $(r^{(k)},z^{(k)})^{1/2} = \|r^{(k)}\|_{M^{-1}}$ was first reduced to less than EPS, for the initial approximation $u^{(0)} \equiv 0$. The value of a was taken to be 5/16, and the problems were solved using a FORTRAN program on a CDC 7600 computer for mesh spacings h = 1/16, 1/32, 1/64.

The parameters $a^{(1)}$, $b^{(1)}$ were used, and there was no restarting. The solution of $Mz = r$ was carried out at each iteration using either the program GMA with marching parameter $K = 2$ [2] or the program package from NCAR [28]. (These two programs give slightly different rounding errors; we observed no important difference between them in their effect on the cg iterations.) The initial residual norms $\|r^{(0)}\|_{M^{-1}}$ were of the order of 10^2 for $h = 1/16$ and 10^3 for $h = 1/64$.

Problem I is the discretized linear problem (15, 19) augmented with the equations

$$\frac{4}{h^2} u_{i,j} + u_{i,j} = -\frac{4}{h^2} - 1$$

for the Dirichlet boundary points on $y = 1$, with M as described above with $\kappa = 1$. Problem II is the discretized nonlinear equation (14) with the same boundary condition (17) on $y = 1$ as that for M and with the same M as for Problem I. Problem III combines the boundary condition of Problem I with the nonlinearity of Problem II; it is the discretized nonlinear problem (14, 15, 18), again with the same M.

The number p of special boundary points for Problems I and III is given in Table 4 for each of the mesh spacings. The finite termination behavior of cg for the linear problem can be observed clearly for the coarsest mesh; for the finer meshes some contamination resulting from rounding errors occurs. For the finest mesh, a residual small enough for practical purposes occurs well before $2n + 3$ iterations have been carried out.

TABLE 4

COMPARISONS FOR SECOND TEST PROBLEM

h	Problem	2p + 3	No. of iterations for $\|r\|_M^{-1} \leq$ EPS							
			ESP = 10^{-10}	10^{-9}	10^{-8}	10^{-7}	10^{-6}	10^{-5}	10^{-4}	10^{-3}
$\frac{1}{16}$	I	13	13	12	12	12	11	11	9	8
	II	--	7	6	6	5	5	4	4	3
	III	13	24	23	20	20	18	16	13	13
$\frac{1}{32}$	I	25	26	23	21	19	18	16	14	13
	II	--	7	6	6	5	5	4	4	3
	III	25	36	34	30	28	26	23	20	17
$\frac{1}{64}$	I	49	43	36	34	30	28	23	22	19
	II	--	7	6	6	5	5	4	4	3
	III	49	57	51	49	43	39	35	31	26

The results for Problem II indicate that convergence is rapid for this choice of M when the mild nonlinearity is present and the mixed boundary conditions on $y = 1$ are absent. As one would expect for this case the number of iterations to reach a given residual is essentially independent of mesh size. The results for Problem III indicate that with the mixed boundary condition on $y = 1$, the convergence rate for the mildly nonlinear case is slowed moderately from that for the linear case, Problem I. One could likely improve the results for Problems I and III in terms of number of iterations by choosing κ to be, instead of a--constant, the sum of a function in x and one in y, which would still permit the use of fast direct methods. We did not include such choices in our experiments, however. We repeated some of our experiments for an initial approximation $u^{(0)}$ equal to pseudo-random numbers in [0,1] and found no substantial difference from the results of Table 4.

REFERENCES

[1] 0. Axelsson, "On preconditioning and convergence acceleration in sparse matrix problems," Report 74-10, Data Handling Division, CERN, Geneva (1974).

[2] R. E. Rank, "Marching algorithms for elliptic boundary value problems," Doctoral Thesis, Harvard University (1975).

[3] R. Bartels and J. W. Daniel, "A conjugate gradient approach to nonlinear elliptic boundary value problems in irregular regions," Proc. Conf. on the Numerical Solution of Differential Equations, Springer-Verlag Lecture Notes 363, (1974), l-11.

[4] D. Bertsekas, "Partial conjugate gradient methods for a class of optimal control problems," IEEE Trans. Automat Control AC-19, (1974), 209-17.

[5] B. L. Buzbee, G. H. Golub, and C. W. Nielson, "On direct methods for solving Poisson's equations,' SIAM J. Numer. Anal. 7 (1970), 627-56.

[6] P. Concus, 'Numerical solution of the minimal surface equation,' Math. Comp. 21 (1967), 340-350.

[7] P. Concus, 'Numerical solution of the minimal surface equation by block nonlinear successive overrelaxation," Information Processing 68, Proc. IFIP Congress 1968, North-Holland, Amsterdam (1969), 153-158.

[8] P. Concus and G. H. Golub, "A generalized conjugate gradient method for nonsymmetric systems of linear equations," Proc. Second International Symposium on Computing Methods in Applied Sciences and Engineering, IRIA, Paris, Dec. 1975, Springer-Verlag Lecture Notes (to appear).

[9] P. Concus, G. H. Golub, and D. P. O'Leary, "A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations," Sparse Matrix Computations, J. R. Bunch and D. J. Rose, Eds., Academic Press, New York (1976), 309-332.

[10] Daniel, J. W., "The conjugate gradient method for linear and nonlinear operator equations," Ph.D. Thesis, Stanford University and SIAM J. Numer. Anal. 4 (1967), 10-26.

[11] Dixon, L. C. W., "Conjugate gradient algorithms: quadratic termination without linear searches," J. Inst. Maths. Applics. 15 (1975), 9-18.

[12] L. W. Ehrlich, "On some experience using matrix splitting and conjugate gradient" (abstract), SIAM Review 18 (1976), 801.

[13] D. Fischer, G.H. Golub, 0. Hald, C. Leiva, and 0. Widlund, 'On Fourier-Toeplitz methods for separable elliptic problems,' Math. Comp. 28 (1974), 349-368.

[14] R. Fletcher and C.M. Reeves, "Function minimization by conjugate gradients," Computer J. 7 (1964), 149-54.

[15] G.E. Forsythe and W.R. Wasow, "Finite-difference Methods for Partial Differential Equations," Wiley, New York (1960).

[16] D. Goldfarb, "A conjugate gradient method for nonlinear programming," Princeton University Press, Thesis (1966).

[17] L. Hayes, D.M. Young, and E. Schleicher, "The use of the accelerated SSOR method to solve large linear systems" (abstract), SIAM Review 18 (1976), 808.

[18] M. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems," J. Res. Nat. Bur. Stand. 49 (1952), 409-36. --

[19] R.W. Hockney, "The potential calculation and some applications," Methods in Computational Physics 9, B. Adler, S. Fernbach and M. Rotenberg eds., Academic Press, New York, (1969), 136-211.

[20] M. Lenard, "Practical convergence conditions for restarted conjugate gradient methods," MRC Report 1373, University of Wisconsin (December 1973).

[21] 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," Tech. Report TR-1, Acad. Comp. Ctr., Utrecht, The Netherlands (1976).

[22] L. Nazareth, "A conjugate direction algorithm without line searches," J. Opt. Th. Applic. (to appear).

[23] D.P. O'Leary, "Hybrid conjugate gradient algorithms," Doctoral dissertation, Computer Science Department, Stanford University, Report No. STAN-CS-76-548 (1976).

[24] J.M. Ortega and W.C. Rheinboldt, "Iterative Solution of Nonlinear Equations in Several Variables," Academic Press, New York (1970).

[25] M.J.D. Powell, "Restart procedures for the conjugate gradient method," Report C.S.S. 24, AERE, Harwell, England (1975).

[26] J.K. Reid, "On the method of conjugate gradients for the solution of large sparse systems of linear equations," Large Sparse Sets of Linear Equations, J. K. Reid, ed., Academic Press, New York, (1971), 231-254.

[27] S. Schecter, "Relaxation methods for convex problems," SIAM J. Numer. Anal. 5 (1968), 601-612.

[28] P. Swarztrauber and R. Sweet, "Efficient FORTRAN subprograms for the solution of elliptic partial differential equations," Report No. NCAR-TN/IA-109, National Center for Atmospheric Research, Boulder, co (1975).

[29] R. R. Underwood, "An approximate factorization procedure based on the block Cholesky decomposition and its use with the conjugate gradient method," Report No. NEDO-11386, General Electric Co., Nuclear Energy Systems Div., San Jose, CA. (1976).

[30] W.I. Zangwill, "Nonlinear Programming, A Unified Approach," Prentice-Hall, Englewood Cliffs, N.J., (1969).