Computing 19, 321—339 (1978)

Computing
© by Springer-Verlag 1978

Numerical Solution of Nonlinear Elliptic Partial Differential Equations by a Generalized Conjugate Gradient Method

P. Concus, Berkeley, California, **G. H. Golub**, Stanford, California, and **D. P. O'Leary**, Ann Arbor, Michigan

Received March 30, 1977

Abstract — Zusammenfassung

Numerical Solution of Nonlinear Elliptic Partial Differential Equations by a Generalized Conjugate Gradient Method. 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 mildy nonlinear example, for which, in the corresponding linear case, the finite termination property of the conjugate gradient algorithm is crucial.

Numerische Lösung von nichtlinearen elliptischen partiellen Differentialgleichungen mit einer verallgemeinerten Methode der konjugierten Gradienten. Wir haben früher eine verallgemeinerte Methode der konjugierten Gradienten studiert, um dünnbesetzte positiv definite Systeme von linearen Gleichungen zu lösen, die von der Diskretisierung von elliptischen partiellen Differential-Randwertproblemen herrühren. Wir betrachten hier die Verallgemeinerung auf den nichtlinearen Fall: Wir spalten den ursprünglichen diskretisierten Operator auf in eine Summe von zwei Operatoren. Einer von diesen Operatoren entspricht einem leicht lösbaren System von Gleichungen, und wir beschleunigen die aus dieser Spaltung hervorgehende Iteration mit (nichtlinearen) konjugierten Gradienten. Das Verhalten der Methode wird illustriert durch Anwendung auf die Minimalflächen-Gleichung, mit Spaltungen entsprechend dem nichtlinearen SSOR-Verfahren, der angenäherten Faktorisierung der Jacobi-Matrix, oder den elliptischen Operatoren, die sich für schnelle direkte Methoden eignen. Die Resultate von numerischen Experimenten für ein nur schwach nichtlineares Beispiel sind ebenfalls angegeben. Für den entsprechenden linearen Fall ist in diesem Fall die Konvergenz des konjugierten Gradienten-Algorithmus in einer endlichen Anzahl von Schritten wesentlich.

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 section 1 several nonlinear conjugate gradient algorithms and in section 2 some convergence properties. In section 3 possible splitting choices for the approximating operator are described. A test problem for the minimal surface equation is discussed in section 4, and experimental results for several splittings are summarized in section 5. In section 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

$$A\,x = b \tag{1}$$

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 $k=0, 1, \ldots$

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

$$M\,z^{(k)} = r^{(k)}. \tag{3}$$

(ii) Compute the parameter $b_k^{(1)}$, where $b_0^{(1)} = 0$ and

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

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

$$a_k^{(2)} = \frac{p^{(k)^T}\,r^{(k)}}{p^{(k)^T}\,A\,p^{(k)}} \quad \text{or} \quad b_k^{(2)} = -\frac{z^{(k)^T}\,A\,p^{(k-1)}}{p^{(k-1)^T}\,A\,p^{(k-1)}}.$$

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), $g(x) \equiv A\,x - b$; 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

$$M\,z^{(k)} = r^{(k)}. \tag{5}$$

(Nii) Compute $b_k = b_k^{(1)}$ or $b_k^{(2)}$, where $b_0 = 0$ and

$$b_k^{(1)} = \frac{z^{(k)^T}\,r^{(k)}}{z^{(k-1)^T}\,r^{(k-1)}}, \quad b_k^{(2)} = -\frac{z^{(k)}\;J\,(x^{(k)})\,p^{(k-1)}}{p^{(k-1)^T}\,J\,(x^{(k)})\,p^{(k-1)}}, \quad k \geq 1$$

and

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

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

$$a_k^{(1)} = \frac{z^{(k)^T}\,r^{(k)}}{p^{(k)^T}\,J\,(x^{(k)})\,p^{(k)}}, \quad a_k^{(2)} = \frac{p^{(k)^T}\,r^{(k)}}{p^{(k)^T}\,J\,(x^{(k)})\,p^{(k)}},$$

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 sections 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 section 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:

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

$$\begin{aligned} a_k &= \min\{\hat{a} : f(x^{(k)} + \hat{a}\, p^{(k)}) \leq f(x^{(k)} + a\, p^{(k)})\ \forall a \in (0, \infty)\} \\ &\equiv a_k^{\text{opt}} \end{aligned}$$

and if the iteration is restarted every α 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 spacer 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 non-increasing 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*

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

Then $f(x)$ is monotone non-increasing at the k-th 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,

$$p^{(k)^T} g(x^{(k)} + a\, p^{(k)})$$

will be a monotone increasing function of a that is negative at $a=0$ and is 0 at $a=a_k^{\text{opt}}$, 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

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

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 $a < a_k^{\text{opt}}$ and this will give us a sufficient condition for convergence:

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

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

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

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 $x^{(k+1)}$:

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

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

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

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*

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

Then $f(x)$ is non-increasing 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 (section 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

$$x^{(k+1)} = x^{(k)} + z^{(k)}. \tag{7}$$

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 $J(x)$ 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

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

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

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

where $g(x)$ and $J(x)$ 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

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

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 $M\, z^{(k)} = r^{(k)}$ 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:

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

followed by

backward sweep:

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

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

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

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

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

We consider the domain $0 \leq x \leq 2$, $0 \leq y \leq 1$. If $s(x, y)$ is symmetric about $x = 1$, then the problem need only be solved on the unit square with the symmetry condition

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

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 $-2h^2$,

$$g_{i,j} \equiv \gamma_{\bar{i},\bar{j}}(2u_{i,j}-u_{i-1,j}-u_{i,j-1})+\gamma_{\bar{i}+1,\bar{j}}(2u_{i,j}-u_{i+1,j}-u_{i,j-1})$$
$$+\gamma_{\bar{i},\bar{j}+1}(2u_{i,j}-u_{i-1,j}-u_{i,j+1})+\gamma_{\bar{i}+1,\bar{j}+1}(2u_{i,j}-u_{i+1,j}-u_{i,j+1})=0, \qquad (12)$$
$$i=1,2,\ldots,n-1; \quad j=1,2,\ldots,n-1,$$

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

$$|\nabla u|^2_{\bar{i},\bar{j}}=\frac{1}{2h^2}[(u_{i,j}-u_{i-1,j})^2+(u_{i,j}-u_{i,j-1})^2$$
$$+(u_{i,j-1}-u_{i-1,j-1})^2+(u_{i-1,j}-u_{i-1,j-1})^2],$$
$$i=1,2,\ldots,n; \quad j=1,2,\ldots,n.$$

Along the symmetry boundary we obtain

$$g_{n,j}=\gamma_{\bar{n},\bar{j}}(2u_{n,j}-u_{n-1,j}-u_{n,j-1})+\gamma_{\bar{n},\bar{j}+1}(2u_{n,j}-u_{n-1,j}-u_{n,j+1})=0,$$
$$j=1,2,\ldots,n-1. \qquad (13)$$

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

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

a positive-definite symmetric matrix that is block tridiagonal, with each block being tridiagonal. For this test problem, the calculation of $\gamma'_{\bar{i},\bar{j}}\equiv d\gamma_{\bar{i},\bar{j}}/d|\nabla u|^2_{\bar{i},\bar{j}}$ 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 section 4 was solved numerically for the same boundary data as was considered in [6, 7],

$$v=0 \qquad \text{on } x=0 \text{ and } y=1,$$
$$v=\sin\frac{\pi x}{2} \qquad \text{on } y=0,$$

and the symmetry condition (11). The following algorithms discussed in sections 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 $M = \triangle$, $-2h^2$ times the discrete Laplace 5 point operator [$\gamma \equiv 1$ in (12, 13)].

IX. Conjugate gradient algorithm with $M = D^{1/2}(\triangle + \kappa I) D^{1/2}$, where $\triangle$ 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 number of iterations and operation counts required to decrease the residual to specified values. In table 1 are given approximate operation counts for various phases of the algorithms. 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 results were obtained originally in terms of the 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. 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

Tables 1a and 1b. *Cost factors per step for minimal surface algorithms*

Table 1a

Item	Cost
(1) Conjugate gradient overhead	$5N$
(2) SOR overhead	N
(3) Calculation of $\gamma_{\overline{ij}}$	$3N$
(4) Calculation of g (given $\gamma_{\overline{ij}}$)	$4N$
(5) Calculation of $\gamma'_{\overline{ij}}$ (given $\gamma_{\overline{ij}}$)	$2N$
(6) Calculation of J (given $\gamma_{\overline{ij}}$, $\gamma'_{\overline{ij}}$)	$20N$
(7) Calculation of only the tridiagonal portion of J ($8N$ for the diagonal only)	$12N$
(8) Factoring and solving tridiagonal system	$5N$
(9) Formation of Jp (given J)	$9N$

There are $n(n-1)=N$ unknowns. Costs consider only multiplications, divisions, and square roots and include only the highest order terms in N.

Table 1b

Algorithm	Item	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	Scaling	Total operations per point
	Cost	$5N$	$1N$	$3N$	$4N$	$2N$	$20N$	$12N$	$5N$	$9N$	N	
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-BSSOR		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.

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.

Table 2. *Comparison of algorithms by number of iterations (and number of multiplications per mesh point) to obtain a residual* $\|r^{(k)}\|_2 \le EPS$; $h = 1/16$ *minimal surface equation*

Algorithm (Cost Factor)		ω	EPS $= 10^{-5}$	10^{-4}	10^{-3}	10^{-2}
I. BSOR-Newton (32)		1.1	182(5824)	139(4448)	96(3072)	53(1696)
		1.2	147(4704)	113(3616)	78(2496)	43(1376)
		1.3	118(3776)	90(2880)	48(1536)	35(1120)
		1.4	93(2976)	71(2272)	50(1600)	28 (896)
		1.5	70(2240)	54(1728)	38(1216)	22 (704)
		1.6	49(1568)	39(1248)	28 (896)	17 (544)
		1.7	33(1056)	26 (832)	20 (640)	14 (448)
		1.8	48(1536)	38(1216)	30 (960)	17 (544)
		1.9	102(3264)	80(2560)	60(1920)	37(1184)
II. BSSOR-Newton (59)		1.1	99(5841)	76(4484)	53(3127)	30(1770)
		1.2	83(4897)	63(3717)	44(2596)	25(1475)
		1.3	70(4130)	53(3127)	37(2183)	22(1298)
		1.4	59(3481)	45(2655)	32(1888)	18(1062)
		1.5	51(3009)	39(2301)	28(1652)	16 (944)
		1.6	45(2655)	35(2065)	25(1475)	14 (826)
		1.7	44(2596)	34(2006)	24(1416)	14 (826)
		1.8	52(3068)	41(2419)	29(1711)	17(1003)
III. Newton-BSSOR (57)		1.1	99(5643)	77(4389)	54(3078)	31(1767)
		1.2	83(4731)	64(3648)	45(2565)	26(1482)
		1.3	70(3990)	54(3078)	38(2166)	22(1254)
		1.4	60(3420)	46(2622)	33(1881)	19(1083)
		1.5	52(2964)	40(2280)	28(1596)	17 (969)
		1.6	46(2622)	36(2052)	25(1425)	15 (855)
		1.7	45(2565)	35(1995)	24(1368)	14 (798)
		1.8	52(2964)	40(2280)	28(1596)	17 (969)
		1.9	94(5358)	72(4104)	50(2850)	29(1653)
IV. CG (43)	$a^2\,b^1$		279(11997)	217(9331)	158(6794)	99(4257)
	$a^2\,b^2$		274(11782)	217(9331)	157(6751)	98(4214)
	$a^1\,b^1$		244(10492)	183(7869)	134(5762)	84(3612)
	$a^1\,b^2$		243(10449)	183(7869)	134(5762)	84(3612)
V. CG + BSSOR-Newton Restart 9 (102)	$a^2\,b^1$	1.1	23(2346)	22(2244)	20(2040)	16(1632)
		1.2	23(2346)	21(2142)	19(1938)	15(1530)
		1.3	22(2244)	21(2142)	18(1836)	14(1428)
		1.4	22(2244)	20(2040)	16(1632)	14(1428)
		1.5	21(2142)	19(1938)	16(1632)	13(1326)
		1.6	20(2040)	18(1836)	16(1632)	12(1224)
		1.7	20(2040)	18(1836)	15(1530)	12(1224)
		1.8	22(2244)	20(2040)	18(1836)	13(1326)
		1.9	no convergence			
Restart 13		1.65	16(1632)	13(1326)	11(1122)	9 (918)
V. CG + BSSOR-Newton $\omega = 1.7$ (102)	$a^2\,b^1$		20(2040)	18(1836)	15(1530)	12(1224)
	$a^2\,b^2$		15(1530)	13(1326)	11(1122)	8 (816)
	$a^1\,b^1$		17(1734)	15(1530)	13(1326)	10(1020)
	$a^1\,b^2$		17(1734)	14(1428)	12(1224)	10(1020)
VI. CG + Newton-BSSOR (71)	$a^1\,b^1$	1.1	24(1704)	22(1562)	21(1491)	16(1136)
		1.2	23(1633)	21(1491)	19(1349)	15(1065)
		1.3	22(1562)	21(1491)	19(1349)	15(1065)
		1.4	22(1562)	20(1420)	17(1207)	14 (994)
		1.5	21(1491)	18(1278)	16(1136)	14 (994)

Table 2. *(continued)*

Algorithm (Cost Factor)		ω	EPS $= 10^{-5}$	10^{-4}	10^{-3}	10^{-2}
		1.6	23(1633)	20(1420)	18(1278)	15(1065)
		1.7	26(1846)	23(1633)	20(1420)	18(1278)
		1.8	29(2059)	26(1846)	23(1633)	20(1420)
		1.9	31(2201)	28(1988)	26(1846)	21(1491)
VI. CG + Newton-BSSOR	$a^2 b^1$		22(1562)	20(1420)	17(1207)	14 (994)
$\omega = 1.4$ (71)	$a^2 b^2$		19(1349)	17(1207)	15(1065)	11 (781)
	$a^1 b^1$		26(1846)	23(1633)	20(1420)	18(1278)
	$a^1 b^2$		22(1562)	18(1278)	15(1065)	13 (923)
VII. CG + sparse LDL^T	$a^2 b^1$				26(1318)	21(1068)
(50 + 6 per restart)	$a^2 b^2$		30(1524)	27(1374)	21(1068)	16 (812)
	$a^1 b^1$		27(1374)	24(1218)	20(1018)	13 (662)
	$a^1 b^2$		28(1424)	25(1268)	19 (968)	15 (762)
VIII. CG + Laplacian (55)	$a^2 b^1$		19(1045)	19(1045)	16 (880)	11 (605)
	$a^2 b^2$		16 (880)	16 (880)	13 (715)	7 (385)
	$a^1 b^1$		15 (825)	15 (825)	13 (715)	7 (385)
	$a^1 b^2$		15 (825)	15 (825)	13 (715)	8 (440)
IX. CG + Laplacian + J	$a^2 b^1$		13 (754)	13 (754)	12 (696)	9 (522)
diagonal + shift (58)	$a^2 b^2$		11 (638)	11 (638)	9 (522)	7 (406)
	$a^1 b^1$		12 (696)	12 (696)	10 (580)	7 (406)
	$a^1 b^2$		13 (754)	13 (754)	11 (638)	9 (522)

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

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

Table 3. *Comparison of algorithms by number of iterations (and 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$=10^{-6}$	10^{-5}	10^{-4}	10^{-3}	10^{-2}
I. BSOR-Newton (32)	1.3		>350(>11200)	330(10560)	217(6944)	104(3328)
	1.4		>350(>11200)	262 (8384)	173(5536)	83(2656)
	1.5	341(10912)	272 (8704)	203 (6496)	134(4288)	65(2080)
	1.6	253 (8096)	202 (6464)	151 (4832)	101(3232)	50(1600)
	1.7	173 (5536)	139 (4448)	105 (3360)	71(2272)	37(1184)
	1.8	93 (2976)	77 (2464)	61 (1952)	45(1440)	27 (864)
	1.9	128 (4096)	103 (3296)	86 (2752)	64(2048)	36(1152)
II. BSSOR-Newton (59)	1.5	192(11328)	154 (9086)	115 (6785)	77(4543)	39(2301)
	1.6	154 (9086)	123 (7257)	93 (5487)	62(3658)	32(1888)
	1.7	123 (7257)	99 (5841)	75 (4425)	50(2950)	26(1534)
	1.75	112 (6608)	90 (5310)	68 (4012)	46(2714)	24(1416)
	1.8	106 (6254)	85 (5015)	64 (3776)	44(2596)	24(1416)
	1.85	107 (6313)	86 (5074)	66 (3894)	45(2655)	26(1534)
III. Newton-BSSOR (57)	1.4	238(13566)	191 (10887)	143 (8151)	96(5472)	48(2736)
	1.5	192(10944)	154 (8778)	115 (6555)	77(4389)	39(2223)
	1.6	153 (8721)	123 (7011)	92 (5244)	62(3534)	32(1824)
	1.7	123 (7011)	99 (5643)	75 (4275)	50(2850)	26(1482)
	1.8	104 (5928)	84 (4788)	63 (3591)	42(2394)	22(1254)
	1.9	127 (7239)	102 (5814)	77 (4389)	52(2964)	29(1653)
V. CG + BSSOR-Newton Restart 13, $a^2 b^2$ (102)	1.5	30 (3060)	29 (2958)	25 (2550)	20(2040)	17(1734)
	1.6	41 (4182)	39 (3978)	34 (3468)	30(3060)	27(2754)
	1.7	31 (3162)	28 (2856)	23 (2346)	19(1938)	16(1632)
	1.75	34 (3468)	31 (3162)	29 (2958)	27(2754)	23(2346)
	1.8	31 (3162)	28 (2856)	26 (2652)	21(2142)	19(1938)
	1.85	31 (3162)	28 (2856)	25 (2550)	21(2142)	18(1836)
	1.9	no convergence				
VI. CG + Newton-BSSOR $a^2 b^2$, Restart 13 (71)	1.4	36 (2556)	32 (2272)	30 (2130)	24(1704)	18(1278)
	1.5	32 (2272)	30 (2130)	27 (1917)	21(1491)	18(1278)
	1.6	35 (2485)	30 (2130)	25 (1775)	23(1633)	18(1278)
	1.7	33 (2343)	32 (2272)	30 (2130)	27(1917)	23(1633)
	1.8	37 (2627)	33 (2343)	30 (2130)	27(1917)	24(1704)
	1.9	no convergence				
IX. CG + Laplacian $a^2 b^1$		38 (2318)	35 (2135)	32 (1952)	29(1769)	23(1403)
+ J diagonal $a^1 b^1$ + shift Restart 16 (61)		39 (2379)	36 (2196)	33 (2013)	30(1830)	22(1342)
Restart 9 $a^2 b^1$		27 (1647)	25 (1525)	21 (1281)	19(1159)	15 (915)

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,

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

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

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

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

$$\frac{\partial v}{\partial y} = 0 \text{ on } y=1 \text{ for } 0<x<1. \tag{17}$$

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

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

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 $(r^{(k)T}, z^{(k)})^{1/2} = \|r^{(k)}\|_{M^{-1}}$ was first reduced to less than EPS, for the initial approximation $u^{(0)}=0$. 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 was observed clearly in our experiments, most sharply for the coarsest mesh, for which contamination resulting from rounding errors had least opportunity to accumulate. For the finest mesh, a residual small enough for practical purposes occurred well before $2p+3$ iterations had been carried out.

Table 4. *Comparisons for second test problem*

			No. of iterations for $\|r\|_{M^{-1}} \leq$ EPS					
h	Problem	$2p+3$	EPS $=10^{-8}$	10^{-7}	10^{-6}	10^{-5}	10^{-4}	10^{-3}
1/16	I	13	12	12	11	11	9	8
	II	—	6	5	5	4	4	3
	III	13	20	20	18	16	13	13
1/32	I	25	21	19	18	16	14	13
	II	—	6	5	5	4	4	3
	III	25	30	28	26	23	20	17
1/64	I	49	34	30	28	23	22	19
	II	—	6	5	5	4	4	3
	III	49	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] Axelsson, O.: On preconditioning and convergence acceleration in sparse matrix problems. Report 74-10, Data Handling Division, CERN, Geneva (1974).

[2] Bank, R. E.: A FORTRAN implementation of the generalized marching algorithm. Trans. Math. Software. To appear.

[3] Bartels, R., Daniel, J. W.: A conjugate gradient approach to nonlinear elliptic boundary value problems in irregular regions. Proc. Conf. on the Numerical Solution of Differential Equations (Lecture Notes, Vol. 363), pp. 1—11. Berlin-Heidelberg-New York: Springer 1974.

[4] Bertsekas, D.: Partial conjugate gradient methods for a class of optimal control problems. IEEE Trans. Automat Control AC-19 1974, 209—217.

[5] Buzbee, B. L., Golub, G. H., Nielson C. W.: On direct methods for solving Poisson's equations. SIAM J. Numer. Anal. *7*, 627—656 (1970).

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

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

[8] Concus, P., Golub, G. H.: 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 (Lecture Notes in Economics and Math. Systems, Vol. 134), pp. 56—65. Berlin-Heidelberg-New York: Springer 1976.

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

[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*, 10—26 (1967).

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

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

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

[14] Fletcher, R., Reeves, C. M.: Function minimization by conjugate gradients. Computer J. *7*, 149—154 (1964).

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

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

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

[18] Hestenes, M., Stiefel, E.: Methods of conjugate gradients for solving linear systems.J. Res. Nat. Bur. Stand. *49*, 409—436 (1952).

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

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

[21] Meijerink, J. A., van der Vorst, H. A.: An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. Math. Comp. *31*, 148—162 (1977).

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

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

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

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

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

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

[28] Swartztrauber, P., Sweet, R.: 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] Underwood, R. R.: 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] Zangwill, W. I.: Nonlinear Programming, A Unified Approach. Englewood Cliffs, N. J.: Prentice-Hall 1969.

P. Concus
Lawrence Berkeley Laboratory
University of California
Berkeley, CA 94720, U. S. A.

D. P. O'Leary
Department of Mathematics
University of Michigan
Ann Arbor, MI 48109, U. S. A.

G. H. Golub
Computer Science Department
Stanford University
Stanford, CA 94305, U. S. A.