SIAM J. SCI. STAT. COMPUT.
Vol. 13, No. 1, pp. 30–46, January 1992

002

PRECONDITIONED ITERATIVE METHODS FOR HOMOTOPY CURVE TRACKING*

COLIN DESA†, KASHMIRA M. IRANI†, CALVIN J. RIBBENS†, LAYNE T. WATSON†, AND HOMER F. WALKER‡

Abstract. Homotopy algorithms are a class of methods for solving systems of nonlinear equations that are globally convergent with probability one. All homotopy algorithms are based on the construction of an appropriate homotopy map and then the tracking of a curve in the zero set of this homotopy map. The fundamental linear algebra step in these algorithms is the computation of the kernel of the homotopy Jacobian matrix. Problems with large, sparse Jacobian matrices are considered. The curve-tracking algorithms used here require the solution of a series of very special systems. In particular, each $(n+1) \times (n+1)$ system is in general nonsymmetric but has a leading symmetric indefinite $n \times n$ submatrix (typical of large structural mechanics problems, for example). Furthermore, the last row of each system may by chosen (almost) arbitrarily. The authors seek to take advantage of these special properties. The iterative methods studied here include Craig's variant of the conjugate gradient algorithm and the SYMMLQ algorithm for symmetric indefinite problems. The effectiveness of various preconditioning strategies in this context are also investigated, and several choices for the last row of the systems to be solved are explored.

Key words. globally convergent, homotopy algorithm, nonlinear equations, preconditioned conjugate gradient, homotopy curve tracking, sparse matrix, matrix splitting, bordered matrix

AMS(MOS) subject classifications. 65F10, 65F50, 65H10, 65K10

1. Introduction.

The fundamental problem motivating this work is to solve a nonlinear system of equations $F(x) = 0$, where $F : E^n \to E^n$ is a C^2 map defined on real n-dimensional Euclidean space E^n. The homotopy approach to solving $F(x) = 0$ is to construct a continuous map $H(\lambda, x)$, the "homotopy," deforming a simple function $s(x)$ to the given function $F(x)$ as λ varies from 0 to 1. Starting from the easily obtained solution to $H(0, x) = s(x) = 0$, the essence of a homotopy algorithm is to track solutions of $H(\lambda, x) = 0$ until a solution of $H(1, x) = F(x) = 0$ is obtained. The theoretical and implementational details of such algorithms are nontrivial; see Rheinboldt and Burkardt [24] and Watson, Billups, and Morgan [36] for summaries of significant recent progress in this area.

Homotopies are a traditional part of topology and only recently have begun to be used for practical numerical computation. The homotopies considered here are sometimes called "artificial-parameter generic homotopies," in contrast to natural-parameter homotopies, where the homotopy variable is a physically meaningful parameter. In the latter case, the resulting homotopy zero curves must be dealt with as they are, bifurcations, ill-conditioning, and all. Therefore, curve tracking becomes the main focus of the problem-solving effort. Our artificial-parameter generic homotopies require that the homotopy zero curves obey strict smoothness conditions. These conditions generally will not hold if the homotopy parameter represents a physically meaningful quantity, but they can always be obtained via certain generic constructions using an artificial (i.e., nonphysical) homotopy parameter.

* Received by the editors April 5, 1990; accepted for publication (in revised form) December 5, 1990.

† Department of Computer Science, Virginia Polytechnic Institute & State University, Blacksburg, Virginia 24061. The work of these authors was supported in part by Department of Energy grant DE-FG05-88ER25068 and Air Force Office of Scientific Research grant 89-0497.

‡ Department of Mathematics and Statistics, Utah State University, Logan, Utah 84322. The work of this author was supported in part by Department of Energy grant DE-FG02-86ER25018 and National Science Foundation grant DMS-0088995.

The objective is to solve a "parameter-free" system of equations, $F(x) = 0$. What this means is that one can design the homotopy to have paths with nice properties. Thus extra attention is devoted to constructing the homotopy, and the curve-tracking algorithm can be limited to a well-behaved class of curves. The goal of using these artificial-parameter homotopies is to solve fixed-point and zero-finding problems with homotopies whose zero curves do not have bifurcations or other singular or ill-conditioned behavior. The mathematical software package HOMPACK [36] used here for comparative purposes is designed for artificial-parameter generic homotopies.

The theory and algorithms for functions $F(x)$ with small, dense Jacobian matrices $DF(x)$ are well developed [33], [32]. In this paper we focus on large, sparse $DF(x)$, a class of problems about which much less is known. Solving large sparse nonlinear systems of equations via homotopy methods involves sparse rectangular linear systems of equations. The sparsity suggests the use of iterative solution methods. Preconditioning techniques are used to make the iterative methods more efficient. In this paper we are particularly interested in problems where the Jacobian matrix $DF(x)$ is symmetric. Such problems are common, for example, in structural mechanics, where the Jacobian matrix is often the Hessian of a potential energy function.

Section 2 describes the homotopy approach to zero-finding problems and outlines the curve-tracking algorithm used in this paper. Section 3 discusses the linear algebra details of homotopy curve tracking. Several algorithmic possibilities are presented. Section 4 presents numerical results from the application of the various algorithms to three test problems. Some general conclusions from these results are drawn in § 5.

2. Homotopy algorithm. The philosophy of artificial-parameter homotopy algorithms is to create homotopies whose zero curves are well behaved, with Jacobian matrices that are well conditioned, and that reach a solution for almost all choices of a parameter. These homotopies may be used to solve fixed-point and zero-finding problems.

In this paper we concentrate on the zero-finding problem $F(x) = 0$, where $F : E^n \to E^n$ is a C^2 map. The theoretical basis for the homotopy algorithm can be summarized as follows (see [7] for details). Suppose there exists a C^2 map

$$\rho : E^m \times [0,1) \times E^n \to E^n$$

such that

(a) the $n \times (m+1+n)$ Jacobian matrix $D\rho(a,\lambda,x)$ has rank n on the set

$$\rho^{-1}(0) = \{\, (a,\lambda,x) \mid a \in E^m, 0 \le \lambda < 1, x \in E^n, \rho(a,\lambda,x) = 0 \,\},$$

and for any fixed $a \in E^m$, letting $\rho_a(\lambda,x) = \rho(a,\lambda,x)$,

(b) $\rho_a(0,x) = \rho(a,0,x) = 0$ has a unique solution x_0,

(c) $\rho_a(1,x) = F(x)$,

(d) ${\rho_a}^{-1}(0)$ is bounded.

Then for almost all $a \in E^m$ there exists a zero curve γ of ρ_a along which the Jacobian matrix $D\rho_a$ has rank n, emanating from $(0,x_0)$ and reaching a zero $\bar{x}$ of F at $\lambda = 1$. Furthermore, γ does not intersect itself and is disjoint from any other zeros of ρ_a. Thus, with probability one, picking an $a \in E^m$ (which uniquely determines x_0), and following γ from $(0,x_0)$ to $(1,\bar{x})$, leads to a zero $\bar{x}$ of F.

There are many different algorithms for tracking the zero curve γ. HOMPACK supports three such algorithms: ordinary differential equation-based, "normal flow," and "augmented Jacobian matrix." We consider only the normal flow algorithm in this paper. A brief description of the normal flow algorithm is now given.

Consider the homotopy map

$$\rho_a(x,\lambda) = \lambda F(x) + (1-\lambda)(x-a).$$

The matrix $D_x\rho_a(x,\lambda) = \lambda DF(x) + (1-\lambda)I$ is symmetric and sparse. (For the problems of interest, $D_x\rho_a(x,\lambda)$ has a "skyline" structure, and is conveniently stored in a packed skyline format, in which the upper triangle is stored in a one-dimensional indexed array. An auxiliary array of diagonal indices is also required.) Assuming that $F(x)$ is C^2, a is such that the Jacobian matrix $D\rho_a(x,\lambda)$ has full rank along γ, and γ is bounded, the zero curve γ is C^1 and can be parameterized by arc length s. Thus $x = x(s), \lambda = \lambda(s)$ along γ, and

$$\rho_a(x(s),\lambda(s)) = 0$$

identically in s.

The zero curve γ given by $(x(s),\lambda(s))$ is the trajectory of the initial value problem

$$\frac{d}{ds}\rho_a\Big(x(s),\lambda(s)\Big) = \Big[D_x\rho_a(x(s),\lambda(s)),\ D_\lambda\rho_a(x(s),\lambda(s))\Big]\begin{pmatrix} \frac{dx}{ds} \\ \frac{d\lambda}{ds} \end{pmatrix} = 0, \tag{1}$$

$$\left\|\left(\frac{dx}{ds},\frac{d\lambda}{ds}\right)\right\|_2 = 1, \tag{2}$$

$$x(0) = a, \qquad \lambda(0) = 0. \tag{3}$$

Since the Jacobian matrix has rank n along γ, the derivative $(dx/ds, d\lambda/ds)$ is uniquely determined by (1), (2), and continuity, and the initial value problem (1–3) can be solved for $x(s),\lambda(s)$. From (1) it can be seen that the unit tangent $(dx/ds, d\lambda/ds)$ to γ is in the one-dimensional kernel of $D\rho_a$.

The normal flow curve tracking algorithm has four phases: prediction, correction, step size estimation, and computation of the solution at $\lambda = 1$. For the prediction phase, assume that two points $(x(s_1),\lambda(s_1))$ and $(x(s_2),\lambda(s_2))$ on γ with corresponding tangent vectors $(dx/ds(s_1),\ d\lambda/ds(s_1))$, $(dx/ds(s_2),\ d\lambda/ds(s_2))$ have been found, and h is an estimate of the optimal step (in arc length) to take along γ. The prediction of the next point on γ is

$$Z^{(0)} = p(s_2 + h),$$

where $p(s)$ is the Hermite cubic interpolating $(x(s),\lambda(s))$ at s_1 and s_2. Precisely,

$$p(s_1) = (x(s_1),\lambda(s_1)), \qquad p'(s_1) = \left(\frac{dx}{ds}(s_1),\frac{d\lambda}{ds}(s_1)\right),$$

$$p(s_2) = (x(s_2),\lambda(s_2)), \qquad p'(s_2) = \left(\frac{dx}{ds}(s_2),\frac{d\lambda}{ds}(s_2)\right),$$

and each component of $p(s)$ is a polynomial in s of degree less than or equal to 3.

Starting at the predicted point $Z^{(0)}$, the corrector iteration is

$$Z^{(k+1)} = Z^{(k)} - \Big[D\rho_a\big(Z^{(k)}\big)\Big]^+ \rho_a\big(Z^{(k)}\big), \qquad k = 0,1,\cdots, \tag{4}$$

where $[D\rho_a(Z^{(k)})]^+$ is the Moore–Penrose pseudoinverse of the $n \times (n+1)$ Jacobian matrix $D\rho_a$. Small perturbations of a produce small changes in the trajectory γ, and the family of trajectories γ for varying a is known as the "Davidenko flow." Geometrically, the iterates given by (4) return to the zero curve along the flow normal to the Davidenko flow, hence the name "normal flow algorithm."

A corrector step ΔZ is the unique minimum norm solution of the equation

$$[D\rho_a]\Delta Z = -\rho_a. \tag{5}$$

Fortunately, ΔZ can be calculated at the same time as the kernel of $[D\rho_a]$, and with just a little more work. The numerical linear algebra details for solving (5), the optimal step size estimation, and the endgame to obtain the solution at $\lambda = 1$ are described by Watson [35], [34].

The calculation of the implicitly defined derivative $(dx/ds, d\lambda/ds)$ is done by computing the one-dimensional kernel of $D\rho_a$, i.e., by solving the $n \times (n+1)$ linear system

$$[D\rho_a]y = 0. \tag{6}$$

This can be elegantly and efficiently done for small dense matrices, but the large sparse Jacobian matrix presents special difficulties. The difficulty now is that the first n columns of the Jacobian matrix $D\rho_a(x, \lambda)$ involving $DF(x)$ are definitely special, and any attempt to treat all $n+1$ columns uniformly would be disastrous from the point of view of storage allocation. Hence, what is required is a good algorithm for solving nonsquare linear systems of equations (5) and (6), where the leading $n \times n$ submatrix of $D\rho_a$ is symmetric and sparse. This paper considers various iterative methods for solving such linear systems of equations.

3. Numerical linear algebra algorithms. As discussed in §2 for the normal flow curve-tracking algorithm, computing both the corrector step ΔZ and the tangent vector $(dx/ds, d\lambda/ds)$ requires the solution of a rectangular linear system involving the $n \times (n+1)$ matrix $[D\rho_a]$. This section describes various algorithms for the solution of such linear systems. In particular, we first consider various ways to construct an $(n+1) \times (n+1)$ invertible system $Ay = b$ whose solution yields a solution to the rectangular systems (5) and (6). We then describe two general approaches to solving systems involving this augmented matrix A, and within these approaches we consider various iterative solution methods. Finally, several preconditioners are suggested for improving the convergence of the iterative methods.

3.1. Defining an invertible system. Let $(\bar{x}, \bar{\lambda})$ be a point on the zero curve γ, and $\bar{y}$ the unit tangent vector to γ at $(\bar{x}, \bar{\lambda})$ in the direction of increasing arc length s. Then the matrix

$$A = \begin{pmatrix} D_x\rho_a(x, \lambda) & D_\lambda\rho_a(x, \lambda) \\ c^t & d \end{pmatrix},$$

where $(c^t \quad d)$ is any vector outside a set of measure zero (a hyperplane), is invertible at $(\bar{x}, \bar{\lambda})$ and in a neighborhood of $(\bar{x}, \bar{\lambda})$. Thus the kernel of $D\rho_a$ for (x, λ) near $(\bar{x}, \bar{\lambda})$ can be found by solving the linear system of equations

$$Ay = \alpha e_{n+1} = b, \tag{7}$$

where $(c^t \quad d)\bar{y} = \alpha$, and e_{n+1} is the $(n+1)$st standard basis vector. Similarly, the corrector step can be found by solving

$$A\Delta Z = \begin{bmatrix} -\rho_a \\ 0 \end{bmatrix}. \tag{8}$$

The coefficient matrix A in the linear systems of equations (7) and (8) has a very special structure, which can be exploited in several ways. Recall that the leading $n \times n$ submatrix of A is $D_x\rho_a$, which is symmetric and sparse, but possibly indefinite. We shall attempt to exploit this property below. The choice of the last row $(c^t \quad d)$ is considered first.

From an implementation point of view, the easiest choice for $(c^t \quad d)$ is probably the kth standard basis vector e_k^t, where the index k is defined by $|\bar{y}_k| = \max_i |\bar{y}_i|$. This is the choice made in HOMPACK. It is not difficult to show that with this choice for $(c^t \quad d)$, A is invertible, though clearly not symmetric.

A second choice for the last row of A considered here is $(c^t \quad d) = \bar{y}^t$. This choice would seem to be best from a conditioning standpoint, since $\bar{y}$ is orthogonal to the rows of $D\rho_a(\bar{x}, \bar{\lambda})$, and hence one expects $\bar{y}$ to be nearly orthogonal to the rows of $D\rho_a(x, \lambda)$ for (x, λ) near $(\bar{x}, \bar{\lambda})$. It is clear that choosing $(c^t \quad d) = \bar{y}^t$ also makes A nonsymmetric, and in addition introduces a dense row into the otherwise sparse matrix A (the last column $D_\lambda\rho_a$ is also dense).

Since symmetry is advantageous for some algorithms, A can be made symmetric and invertible by a third choice $c = D_\lambda\rho_a$. The scalar d must still be chosen so that rank $A = n + 1$. It is enough to consider two cases:

1. Suppose rank $D_x\rho_a = n - 1$. Then $D_\lambda\rho_a$ is not a linear combination of the columns of $D_x\rho_a$, because rank $[D_x\rho_a \; D_\lambda\rho_a] = n$ by the homotopy theory. Thus $c^t = \left(D_\lambda\rho_a\right)^t$ is not a linear combination of the rows of the symmetric matrix $D_x\rho_a$, and we have

$$\text{row rank} \begin{bmatrix} D_x\rho_a \\ (D_\lambda\rho_a)^t \end{bmatrix} = n.$$

   Finally, $(c^t \quad d)^t$ is not a linear combination of the first n columns of A, for *any* choice of d, so the column rank of A is $n + 1$.
2. Now suppose that rank $D_x\rho_a = n$. Then

$$\text{rank} \begin{bmatrix} D_x\rho_a \\ \left(D_\lambda\rho_a\right)^t \end{bmatrix} = n,$$

   and it suffices to choose d to make the last column of A independent from the first n columns. $D_\lambda\rho_a$ is a unique linear combination of the columns of $D_x\rho_a$, and any choice of d other than this combination of the components of $\left(D_\lambda\rho_a\right)^t$ will make the $(n+1)$st column independent. Let $\bar{A}$ denote A at $(\bar{x}, \bar{\lambda})$. Since $\dim[\ker(\bar{A})] \leq 1$, $\bar{A}y = 0$ implies $y = \alpha\bar{y}$, and thus with $\bar{y}^t = \left(\hat{y}^t, \bar{y}_{n+1}\right)$, $\left(D_\lambda\rho_a(\bar{x}, \bar{\lambda})\right)^t \hat{y} + d\,\bar{y}_{n+1} = 0$. Choosing any $\beta \neq 0$ and solving $\left(D_\lambda\rho_a(\bar{x}, \bar{\lambda})\right)^t \hat{y} + d\,\bar{y}_{n+1} = \beta$ for d ($\bar{y}_{n+1} \neq 0$ since rank $D_x\rho_a(\bar{x}, \bar{\lambda}) = n$) gives a d such that rank $A = n + 1$ for (x, λ) near $(\bar{x}, \bar{\lambda})$.

Observe also that if $D_x\rho_a$ is positive definite, choosing $d > 0$ sufficiently large guarantees that

$$A = \begin{pmatrix} D_x\rho_a & D_\lambda\rho_a \\ \left(D_\lambda\rho_a\right)^t & d \end{pmatrix}$$

is also positive definite.

Proof. Since A is symmetric, by Sylvester's theorem A is positive definite if and only if all its leading principal minors are positive. Since $D_x\rho_a$ is positive definite, the first n leading principal minors are positive, and it suffices to show $\det A > 0$. Expanding $\det A$ along the last column,

$$\det A = d \cdot \det D_x\rho_a + \text{terms not involving } d > 0$$

for $d > 0$ sufficiently large. □

3.2. Splitting vs. direct approaches. Given a choice for the last row $(c^t \quad d)$, we now consider two general approaches to solving systems of the form (7) and (8). The first approach deals with the entire matrix A directly. As we have seen, depending on the choice of last row, it may be that A is nonsymmetric. This immediately eliminates several iterative solvers, at least for these cases. However, we do consider a few versions of this approach where possible in the experiments reported in § 4.

The second general approach is to attack (7) and (8) indirectly as follows. Split A into the sum of a symmetric matrix M and a low rank modification L:

$$A = M + L, \tag{9}$$

where

$$M = \begin{pmatrix} D_x\rho_a & c \\ c^t & d \end{pmatrix}, \tag{10}$$

$$L = ue_{n+1}^t, \qquad u = \begin{pmatrix} D_\lambda\rho_a - c \\ 0 \end{pmatrix}. \tag{11}$$

Observe that for almost all choices of $(c^t \quad d)$ the symmetric part M is also invertible. Then using the Sherman–Morrison formula, the solution y to the original system $Ay = b$ can be obtained from

$$y = \left[I - \frac{M^{-1}ue_{n+1}^t}{(M^{-1}u)^t e_{n+1} + 1} \right] M^{-1}b. \tag{12}$$

Equation (12) requires the solution of two linear systems involving the sparse (except possibly for c), symmetric, invertible matrix M. The scheme (9)–(12) was proposed by Kamat, Watson, and Junkins [20], and further investigated by Chan and Saad [6].

A third general approach, not considered here, would be a block-elimination approach that depends on solving systems involving the $n \times n$ matrix $D_x\rho_a$. Observe that block elimination will frequently fail in the homotopy context, because even though rank $A = n + 1$ and rank $(D_x\rho_a \quad D_\lambda\rho_a) =$ rank $D\rho_a = n$, it may very well happen that $D_x\rho_a$ is singular (i.e., rank $= n - 1$). Block-elimination strategies are considered by Chan [4], [3] and Chan and Resasco [5].

3.3. Iterative solution methods. In § 4 we report results from experiments using two iterative solution methods: Craig's method and SYMMLQ. Craig's method is nearly equivalent to the method of conjugate gradients (CG) [17] applied to the normal equations. (Technically, a different norm is minimized over a different Krylov space than if CG were applied directly to the normal equations.) It has long been known that one way to apply CG to nonsymmetric problems is to solve the normal

equations instead of the original system. In particular, given any nonsingular matrix A, the system of linear equations $Ay = b$ can be solved by considering the linear system (normal equations)

$$A^tAy = A^tb,$$

or the related system

$$AA^tz = b, \qquad y = A^tz.$$

Since the coefficient matrix for the latter system is both symmetric and positive definite, the system can be solved by the CG algorithm. Once a solution vector z is obtained, the vector y from the original system can be computed as $y = A^tz$. A major disadvantage of this technique is that the convergence rate depends on $\text{cond}(AA^t) = (\text{cond}(A))^2$ rather than $\text{cond}(A)$. An implementation of the CG algorithm in which y is computed directly, without reference to z or AA^t, is due to Craig [9] and is described in [13] and [16]. Despite the efficiency of the implementation, the convergence rate still depends on $\text{cond}(AA^T) = (\text{cond}(A))^2$ in general. The cost per iteration of Craig's method is dominated by two matrix-vector products, one involving A and one involving A^t. Preconditioning (see § 3.4) increases the work per iteration substantially.

The second solution method considered here is the SYMMLQ algorithm described in Paige and Saunders [23]. SYMMLQ solves symmetric indefinite systems. It is based on a variant of the Lanczos procedure for tridiagonalizing a symmetric matrix. In [23] it is shown that for symmetric positive definite systems, SYMMLQ is mathematically equivalent to CG. However, unlike CG, which can break down when A is indefinite, SYMMLQ is well defined and numerically stable in this case. Like CG, the cost of one iteration of unpreconditioned SYMMLQ is primarily in a single matrix-vector multiplication.

There are many other CG-like methods (i.e., Krylov subspace methods) for solving nonsymmetric or indefinite problems, but most are not satisfactory in our context. The generalized conjugate gradient method of Concus and Golub [8] and Widlund [37] applies only to matrices with positive definite symmetric part (i.e., the matrix must be positive definite, but not necessarily symmetric), although with preconditioning it can sometimes be used to solve more general problems. The generalized conjugate residual method [12] and ORTHOMIN(k) [29] also may break down if the coefficient matrix is not positive definite. More general systems $Ax = b$, where A is not positive definite, can sometimes be solved by ORTHOMIN(k) if a nonsingular matrix Z is known such that ZA is positive definite. ORTHOMIN(k) is then applied to the transformed system $ZAx = Zb$. A related method known as ORTHODIR [38] does not break down in case A is indefinite, but it is observed to have stability problems [26]. ORTHORES is another method with similar properties. GMRES(k) [28], [31], like ORTHOMIN(k), is guaranteed to converge when the coefficient matrix is positive definite. However, for an indefinite coefficient matrix, GMRES(k), while it does not break down, may fail because the residual norms at each step, although nonincreasing, do not converge to zero. Other Krylov subspace methods are studied by Axelsson [1]; Dennis and Turner [10]; Eisenstat, Elman, and Schultz [11]; Jea [19]; Saad [25]; and Saad and Schultz [27].

Finally, there are other efficient iterative methods for solving sparse linear systems based on matrix splittings (see Hageman and Young [15]). Typical of these is the SSOR method, defined in terms of the splitting $A = D - L - U$, where D is the

diagonal of A, L is the strict lower triangle of A, and U is the strict upper triangle of A. The method requires computations involving D^{-1}. In the homotopy context, D^{-1} frequently does not exist, and a diagonal matrix Σ such that $\left[\text{diag}\,(A+\Sigma)\right]^{-1}$ does exist may not be of low rank. Consequently SSOR and methods based on similar splittings are of limited utility in the present context.

3.4. Preconditioners. It is widely known that "preconditioning" can dramatically improve the performance of many iterative methods. For example, the solution to $Ax = b$ can also be obtained by solving the system

$$\tilde{A}x = (Q^{-1}A)x = Q^{-1}b = \tilde{b},$$

where Q is the so-called *preconditioner*. The goal of preconditioning is to decrease the computational effort required to solve systems of linear equations by increasing the rate of convergence of an iterative method. For preconditioning to be effective, faster convergence must outweigh the costs of applying the preconditioner, so that the total cost of solving the linear system is lower. The preconditioned coefficient matrix $\tilde{A}$ usually is not explicitly computed or stored, since although A is sparse, $\tilde{A}$ may not be. The extra work of preconditioning, then, occurs in solving systems involving the matrix Q. The main storage cost for preconditioning is usually for an extra array to hold a factorization of Q. In this paper we consider two preconditioners:

Gill–Murray (GM). The preconditioner is taken as the modified Choleksy factorization GG^t of a symmetric matrix A (see Gill and Murray [14]). In particular, if A is "sufficiently" positive definite, then $GG^t = A$. Otherwise $GG^t = A + \Sigma$, where Σ is diagonal with nonnegative diagonal entries. In the matrix splitting approach described in § 3.2, we apply GM preconditioning to the symmetric matrix M; in the direct approach we apply GM to the entire matrix A (necessitating the choice $c = D_\lambda \rho_a$ to make A symmetric). We apply the GM preconditioner on the left when Craig's method is used (i.e., if A is the original matrix, $(GG^t)^{-1}A$ is the preconditioned matrix). In the case of SYMMLQ, the preconditioned system is $G^{-1}A(G^t)^{-1}$, since SYMMLQ requires symmetry.

ILU. The incomplete LU factorization described in [22] computes a lower triangular matrix L and unit upper triangular matrix U satisfying

$$\begin{cases} L_{ij} = U_{ij} = 0, & (i,j) \in Z, \\ (LU)_{ij} = A_{ij}, & (i,j) \notin Z. \end{cases}$$

Here, Z is the set of indices where A is known to be zero off the diagonal. (This method is often referred to as ILU(0) to indicate that 0 fill-in is allowed.) It is possible that $L_{ii} = 0$ in this algorithm. In this case L_{ii} is set to a small positive number, in which case $(LU)_{ii} \neq A_{ii}$. The ILU factorization is modified slightly so that it may be used with SYMMLQ. In this case we compute an incomplete LDL^T factorization, and apply $LD^{1/2}$ symmetrically as a preconditioner as described above. If during the factorization procedure, an element of the diagonal matrix D is negative, we simply take its absolute value. This is very similar to the GM factorization, except with ILU we do no extra work to ensure that the factorization is well conditioned.

4. Numerical experiments. Of the various algorithmic possibilities mentioned in the previous section, we consider further 22 distinct combinations. Some possibilities do not make sense or are impractical in the homotopy context, and thus are not considered. Ignoring the choice for the last row of A, and also ignoring the question of

TABLE 1
Execution time in seconds for turning point problem.

	CR			CRILU			CRGM
n	e_k	$\bar{y}$	$D_\lambda \rho_a$	e_k	$\bar{y}$	$D_\lambda \rho_a$	$D_\lambda \rho_a$
20	17	19	21	4	7	4	5
60	167	176	186	13	22	13	22
125	1117	1132	1384	38	64	42	85
250	2296	1925	3873	66	110	74	134
500	4741	3899	8352	129	210	148	260
1000	11577	9335	20375	323	493	353	617

TABLE 2
Execution time in seconds for turning point problem.

	CR-S		CRILU-S		CRGM-S	
n	e_k	$\bar{y}$	e_k	$\bar{y}$	e_k	$\bar{y}$
20	28	36	6	6	12	13
60	266	356	20	22	41	50
125	1635	2310	54	65	127	170
250	3026	3767	95	109	228	267
500	6279	7783	189	207	448	501
1000	14150	17768	434	490	1077	1174

splitting, we have six basic methods: unpreconditioned Craig's method (CR), Craig's method with ILU preconditioning (CRILU), Craig's method with GM preconditioning (CRGM), unpreconditioned SYMMLQ (SY), SYMMLQ with ILU preconditioning (SYILU), and SYMMLQ with GM preconditioning (SYGM). Data for all of these methods, with both the splitting and direct approaches, and for various choices of the last row, are given in Tables 1–24. In the splitting cases, the method names have an -S appended (e.g., CR-S, SY-S, ...). We report results from all of these methods on three test problems, which are now briefly described.

Turning point problem. The turning point problem is a relatively simple (and artificial) example derived from the system of equations

$$F(\mathbf{x}) = \big(F_1(\mathbf{x}), F_2(\mathbf{x}), \cdots, F_N(\mathbf{x})\big)^t = 0$$

where

$$F_i(\mathbf{x}) = \tan^{-1}\big(\sin[x_i(i \bmod 100)]\big) - \frac{(x_{i-1} + x_i + x_{i+1})}{20}, \qquad i = 1, \cdots, N,$$

and $x_0 = x_{N+1} = 0$. The zero curve γ tracked from $\lambda = 0$ to $\lambda = 1$ corresponds to $\rho_a(x, \lambda) = (1 - .8\lambda)(x - a) + .8\lambda F(x)$, where a is chosen artificially to produce turning points in γ. The Jacobian matrix $D_x \rho_a$ for the turning point problem is tridiagonal. Tables 1–4 and 13–16 contain the data for this problem.

Shallow arch problem. This is a relatively small but quite difficult problem from structural mechanics. It results from solving the equilibrium equations for a discretization of a shallow arch under an externally applied load. See [21] and [18] for a more complete description. Although this problem is small, it is included in this study because it is a good test of the accuracy of our methods. To go through the limit point and along the unloading portion of the equilibrium curve requires very accurate Jacobian matrices and numerical linear algebra. In fact, the standard iterative linear equation solver used in HOMPACK is unable to go past the limit point without tweaking the HOMPACK step size control parameters. $D_x \rho_a$ for the

TABLE 3
Execution time in seconds for turning point problem.

	SY	SYILU	SYGM
n	$D_\lambda \rho_a$	$D_\lambda \rho_a$	$D_\lambda \rho_a$
20	12	5	5
60	87	17	20
125	405	54	61
250	701	89	104
500	1376	174	199
1000	3270	400	457

TABLE 4
Execution time in seconds for turning point problem.

	SY-S		SYILU-S		SYGM-S	
n	e_k	$\bar{y}$	e_k	$\bar{y}$	e_k	$\bar{y}$
20	20	23	9	9	9	9
60	134	165	31	32	33	34
125	594	738	98	101	105	108
250	1030	1202	164	165	175	175
500	2109	2421	315	320	332	337
1000	4872	5500	736	738	765	787

shallow arch problem has bandwidth 5. Tables 5–8 and 17–20 contain the data for this problem.

Shallow dome problem. This is another realistic problem from structural mechanics, in which the equations of equilibrium for a model of a shallow dome must be solved. See [18] for a more complete description. $D_x \rho_a$ for the shallow dome problem is block diagonal, with dense 21×21 blocks. Tables 9–12 and 21–24 contain the data for this problem.

The times reported in Tables 1–12 are for tracking the entire zero curve γ and thus represent the solution of many linear systems of varying degrees of difficulty. The average, maximum, and minimum number of iterations for each method (Craig's and SYMMLQ) are reported in Tables 13–24. The experiments are done in double precision using a single processor of a Sequent Symmetry S81 multiprocessor. The major headings are the acronyms for the algorithms, and the subheadings denote the choice $(c^t \quad d)$ for the last row of A. There is asymmetry in the tables because some possibilities do not make sense. For instance, there is no CRGM with e_k because the Gill–Murray preconditioner requires a symmetric matrix; and there are no methods based on splitting when $c = D_\lambda \rho_a$, since this choice makes A symmetric, so there is no need to split A into the sum of a symmetric matrix and a low rank modification.

5. Discussion and conclusions. Regarding the choice of last row $(c^t \quad d)$, Tables 1–12 show that there is no clear winner between e_k, $\bar{y}$, and $D_\lambda \rho_a$. Furthermore, there seems to be little correlation between the algorithm and the best choice for c. If anything, a weak conclusion—that all other things being equal, the best choice is e_k—seems to be indicated by the data. Apparently better conditioning (from $\bar{y}$) or symmetry (from $D_\lambda \rho_a$) does not compensate for the extra work involved in these choices compared with e_k.

A comparison of the direct with the splitting approach results in a slight preference for the direct approach. However, the advantage is not a strong one in most cases. In fact, there are cases where the splitting approach is better; and on the shallow arch problem there is virtually no difference. The average number of iterations is

TABLE 5
Execution time in seconds for shallow arch problem.

	CR			CRILU			CRGM
n	e_k	$\bar{y}$	$D_\lambda \rho_a$	e_k	$\bar{y}$	$D_\lambda \rho_a$	$D_\lambda \rho_a$
29	856	884	919	533	458	443	464
47	14205	13591	14606	5794	5807	6776	6921

TABLE 6
Execution time in seconds for shallow arch problem.

	CR-S		CRILU-S		CRGM-S	
n	e_k	$\bar{y}$	e_k	$\bar{y}$	e_k	$\bar{y}$
29	1108	947	599	470	468	818
47	16904	17593	5674	5957	7322	10105

considerably lower for many of the splitting cases than for the corresponding direct case. This helps explain why the direct methods are often not significantly faster, despite the fact that they solve half as many linear systems.

Regarding a comparison between the two basic iterative schemes, the data indicate that unpreconditioned SYMMLQ is faster than unpreconditioned Craig's method, often by a significant amount. The advantage of SYMMLQ seems to be both in fewer iterations and in less work per iteration (one less matrix-vector product). With GM preconditioning, SYMMLQ is still a bit faster than Craig's method for the turning point and shallow dome problems; on the shallow arch problem the two perform roughly the same. When ILU preconditioning is used, Craig's method appears to be superior. It performs slightly better on the turning point and shallow arch problems; and ILU preconditioning combined with SYMMLQ fails completely on the shallow dome problem (SYMMLQ is not converging to a solution of some of the linear systems, and consequently the curve-tracking algorithm does not make progress).

It is tempting to conclude from the data that the best method overall is CRILU. However, it must be pointed out that the ILU factorization fails to exist at turning points and is unstable whenever A is indefinite (as is illustrated by SYILU on the shallow dome problem). We encountered other homotopy curve-tracking runs, on slightly different problems, which failed because the ILU preconditioner failed to exist or generated an overflow, or because of the difficulty caused HOMPACK by inaccurate tangents resulting from ILU. Because of this potential catastrophic failure or instability, it is difficult to seriously consider the ILU preconditioner for use in robust homotopy software. Still, the data do show why the concern of numerical analysts about unstable algorithms is not always shared by others.

The GM preconditioner, meanwhile, is fairly competitive with ILU on the turning point and shallow arch problems. Furthermore, it is more robust in the presence of turning points and when A becomes indefinite. However, the data for the shallow dome problem show that the GM preconditioner may do a very poor job indeed at a few points on the curve. Tables 21–24 indicate that while the average number of iterations is reduced by using the GM preconditioner, the maximum number can actually increase. Thus the net improvement in efficiency is not at all impressive.

The algorithms SSOR and ORTHOMIN(k), discussed earlier, are not shown in the tables because they totally fail at turning points and along unloading portions of equilibrium curves (for reasons stated in § 3). When these methods do work, they can be very efficient (e.g., ORTHOMIN(1) on A with $c = D_\lambda \rho_a$ took 443 (6092) seconds for the shallow arch problem with $n = 29$ (47)), but that is no consolation for homo-

TABLE 7
Execution time in seconds for shallow arch problem.

	SY	SYILU	SYGM
n	$D_\lambda \rho_a$	$D_\lambda \rho_a$	$D_\lambda \rho_a$
29	635	488	463
47	7343	5350	6277

TABLE 8
Execution time in seconds for shallow arch problem.

	SY-S		SYILU-S		SYGM-S	
n	e_k	$\bar{y}$	e_k	$\bar{y}$	e_k	$\bar{y}$
29	615	664	464	499	500	537
47	8992	8362	5593	5683	5760	6506

TABLE 9
Execution time in seconds for shallow dome problem.

	CR			CRILU			CRGM
n	e_k	$\bar{y}$	$D_\lambda \rho_a$	e_k	$\bar{y}$	$D_\lambda \rho_a$	$D_\lambda \rho_a$
21	46	47	47	16	16	16	89
546	2495	2545	2573	355	369	365	2233
1050	4504	4691	4690	632	665	651	4313

TABLE 10
Execution time in seconds for shallow dome problem.

	CR-S		CRILU-S		CRGM-S	
n	e_k	$\bar{y}$	e_k	$\bar{y}$	e_k	$\bar{y}$
21	57	86	21	25	108	57
546	3127	4803	492	630	2710	1787
1050	5615	8553	887	1133	5107	3177

TABLE 11
Execution time in seconds for shallow dome problem.

	SY	SYILU	SYGM
n	$D_\lambda \rho_a$	$D_\lambda \rho_a$	$D_\lambda \rho_a$
21	22	∞	29
546	957	∞	693
1050	1743	∞	1276

TABLE 12
Execution time in seconds for shallow dome problem.

	SY-S		SYILU-S		SYGM-S	
n	e_k	$\bar{y}$	e_k	$\bar{y}$	e_k	$\bar{y}$
21	35	44	∞	∞	41	35
546	1420	2027	∞	∞	1052	928
1050	2529	3690	∞	∞	1902	1629

topy curve tracking.

GMRES(k) has a solid theoretical justification and has been used very successfully in a variety of contexts [28], [2], [31], [30]. Nevertheless, GMRES(k) with $k < n$ performed unacceptably on the test problems here, at least without preconditioning. For the shallow arch problem with $n = 29$ and tol $= 10^{-12}$, GMRES(29) on A with $c = D_\lambda \rho_a$ took 591 seconds, comparable to CRGM and CRGM-S. For $k = 1$, 3, 25, GMRES(k) took over a day of CPU time. Relaxing the tolerance to 10^{-6}, GMRES(25) took 18,330 seconds. This is especially noteworthy because the A matrices are (theoretically) symmetric and positive definite. For the turning point problem with $n = 20$, tol $= 10^{-12}$, $c = D_\lambda \rho_a$, the performance degradation from the full GMRES to GMRES(k) was dramatic. With $k = 20$, 19, 18, 15, 10, 8, GMRES(k) took, respectively, 19, 117, 154, 375, 338, 420 seconds. Thus for these problems, without preconditioning, only the full GMRES method is competitive. Consequently GMRES(k) was not included in the tables.

There are some theoretical results concerning the convergence of GMRES(k) given in [28, §3.4]. These results give worst-case bounds on the rate of residual norm reduction which are determined by the distribution of eigenvalues of A. For the shallow arch and turning point problems, the eigenvalues of A were determined numerically along the homotopy curve, and the resulting bounds were often (although not in every case) found to guarantee only hopelessly slow residual norm reduction, indeed, often to guarantee no residual norm reduction at all even when $k = n$.

Actually, it is apparent from the data that a crucial advantage of Craig's method over methods such as GMRES(k) and ORTHOMIN(k) is that it can iterate indefinitely, if necessary long after the solution would have been reached in exact arithmetic, without incurring increasing costs per iteration and without restarting or otherwise losing information from earlier iterations. Furthermore, there are theoretical guarantees that Craig's method will make progress at each iteration, whereas GMRES(k) may fail to make any progress at all if A is indefinite. It is possible that if the number of iterations necessary to meet the stopping tolerance could be kept small through preconditioning, then GMRES(k) would be competitive for $k < n$. A complete study, similar to that done here for Craig's method, of GMRES(k) with preconditioning and polynomial acceleration would be interesting and will be the topic of a future paper.

Tables 13–24 show the average, maximum, and minimum number of iterations per linear system solution along the homotopy zero curve γ for the three problems, using the same algorithms as in Tables 1–12. Such iteration statistics give an intuitive feel for how the algorithms behave and are sometimes very revealing. For example, Tables 13 and 14 show that symmetry does improve the algorithms' efficiency (compare CR and CR-S with last row e_k^t), and that, all other things being equal, achieving symmetric coefficient matrices is worthwhile. (The algorithms based on splitting to achieve symmetry are not uniformly better, because all other things are not equal.) Note that in all cases (except for the shallow dome problem with GM preconditioning) the maximum number of iterations is less than or equal to four times the average, which says that the convergence behavior is fairly consistent. On the other hand the range between the minimum and maximum is as great as 3 to 536, showing that there is a wide variation in the difficulty of the linear systems encountered along γ.

A succinct, albeit oversimplified, summary of the discussion is that ILU preconditioning is the most efficient, but it may completely fail for some cases, while the Gill–Murray preconditioner rarely fails but may be considerably slower on extremely difficult problems.

TABLE 13

Average, maximum, and minimum number of iterations per linear system along homotopy curve for turning point problem.

	CR			CRILU			CRGM
n	e_k	$\bar{y}$	$D_\lambda \rho_a$	e_k	$\bar{y}$	$D_\lambda \rho_a$	$D_\lambda \rho_a$
20	24,29,1	24,28,1	26,31,1	2,2,1	4,5,1	2,3,1	3,9,2
60	70,86,1	69,84,1	74,91,2	2,3,1	4,7,1	2,3,2	3,12,1
125	159,292,1	151,232,1	179,328,3	2,3,1	4,5,1	2,3,2	5,15,2
250	196,404,1	150,246,1	231,407,3	2,3,1	4,5,1	2,3,2	4,15,2
500	216,427,1	165,337,1	268,489,3	2,3,1	4,6,1	2,3,2	5,16,2
1000	224,446,1	164,323,1	285,536,3	2,3,1	4,5,1	3,3,2	5,16,2

TABLE 14

Average, maximum, and minimum number of iterations per linear system along homotopy curve for turning point problem.

	CR-S		CRILU-S		CRGM-S	
n	e_k	$\bar{y}$	e_k	$\bar{y}$	e_k	$\bar{y}$
20	21,28,1	24,29,1	2,2,1	2,2,1	4,6,1	5,7,1
60	60,100,1	69,87,1	2,3,1	2,3,1	4,8,1	5,8,1
125	127,261,1	154,264,1	2,3,1	2,3,1	5,9,1	6,11,1
250	139,302,1	150,246,1	2,2,1	2,3,1	5,11,1	5,10,1
500	149,314,1	164,281,1	2,2,1	2,3,1	5,11,1	5,10,1
1000	151,312,1	162,289,1	2,2,1	2,3,1	5,11,1	5,11,1

TABLE 15

Average, maximum, and minimum number of iterations per linear system along homotopy curve for turning point problem.

	SY	SYILU	SYGM
n	$D_\lambda \rho_a$	$D_\lambda \rho_a$	$D_\lambda \rho_a$
20	22,28,2	2,5,1	2,7,2
60	48,70,2	2,7,2	2,9,2
125	77,123,3	3,9,1	3,11,2
250	75,131,3	2,9,1	3,11,2
500	80,146,3	2,8,1	3,11,2
1000	83,156,3	3,8,1	3,11,2

TABLE 16

Average, maximum, and minimum number of iterations per linear system along homotopy curve for turning point problem.

	SY-S		SYILU-S		SYGM-S	
n	e_k	$\bar{y}$	e_k	$\bar{y}$	e_k	$\bar{y}$
20	18,25,0	21,25,0	2,5,0	2,6,0	2,5,0	2,5,0
60	37,75,0	46,61,0	2,6,0	3,7,0	2,5,0	3,7,0
125	56,116,0	71,108,0	3,6,0	3,9,0	3,7,0	3,9,0
250	58,118,0	68,100,0	3,8,0	3,8,0	3,7,0	3,8,0
500	61,122,0	72,107,0	3,8,0	3,7,0	3,7,0	3,8,0
1000	62,127,0	71,106,0	3,9,0	3,8,0	3,8,0	3,8,0

TABLE 17

Average, maximum, and minimum number of iterations per linear system along homotopy curve for shallow arch problem.

	CR			CRILU			CRGM
n	e_k	$\bar{y}$	$D_\lambda \rho_a$	e_k	$\bar{y}$	$D_\lambda \rho_a$	$D_\lambda \rho_a$
29	99,127,51	91,107,38	98,120,52	3,3,2	4,5,2	3,3,2	6,7,2
47	265,360,109	239,305,133	265,355,105	3,3,2	4,4,2	3,3,2	6,7,2

TABLE 18

Average, maximum, and minimum number of iterations per linear system along homotopy curve for shallow arch problem.

	CR-S		CRILU-S		CRGM-S	
n	e_k	$\bar{y}$	e_k	$\bar{y}$	e_k	$\bar{y}$
29	66,109,1	66,101,1	2,3,1	3,3,1	4,10,1	28,40,1
47	190,313,1	194,291,1	2,3,1	3,3,1	5,10,1	37,53,1

TABLE 19

Average, maximum, and minimum number of iterations per linear system along homotopy curve for shallow arch problem.

	SY	SYILU	SYGM
n	$D_\lambda \rho_a$	$D_\lambda \rho_a$	$D_\lambda \rho_a$
29	58,79,37	2,5,2	2,4,2
47	115,152,72	3,5,2	3,5,2

TABLE 20

Average, maximum, and minimum number of iterations per linear system along homotopy curve for shallow arch problem.

	SY-S		SYILU-S		SYGM-S	
n	e_k	$\bar{y}$	e_k	$\bar{y}$	e_k	$\bar{y}$
29	39,78,0	42,74,0	2,7,0	4,7,0	2,5,0	10,12,0
47	91,150,0	82,147,0	2,7,0	4,7,0	2,7,0	12,16,0

TABLE 21

Average, maximum, and minimum number of iterations per linear system along homotopy curve for shallow dome problem.

	CR			CRILU			CRGM
n	e_k	$\bar{y}$	$D_\lambda \rho_a$	e_k	$\bar{y}$	$D_\lambda \rho_a$	$D_\lambda \rho_a$
21	26,36,14	26,36,14	26,36,14	2,3,2	2,3,2	2,3,2	23,113,2
546	58,81,17	57,82,17	58,82,18	2,3,2	2,3,2	2,3,2	23,111,2
1050	58,87,18	59,91,18	58,83,18	2,3,2	2,3,2	2,3,2	23,113,2

TABLE 22

Average, maximum, and minimum number of iterations per linear system along homotopy curve for shallow dome problem.

	CR-S		CRILU-S		CRGM-S	
n	e_k	$\bar{y}$	e_k	$\bar{y}$	e_k	$\bar{y}$
21	17,31,1	24,36,1	2,3,1	2,3,1	17,118,1	7,46,1
546	38,75,1	54,87,1	2,3,1	3,3,1	15,113,1	9,63,1
1050	38,76,1	53,91,1	2,3,1	3,3,1	16,114,1	8,101,1

TABLE 23

Average, maximum, and minimum number of iterations per linear system along homotopy curve for shallow dome problem.

	SY	SYILU	SYGM
n	$D_\lambda \rho_a$	$D_\lambda \rho_a$	$D_\lambda \rho_a$
21	17,32,10	∞	7,23,2
546	34,55,11	∞	6,33,2
1050	34,52,11	∞	7,35,2

TABLE 24

Average, maximum, and minimum number of iterations per linear system along homotopy curve for shallow dome problem.

	SY-S		SYILU-S		SYGM-S	
n	e_k	$\bar{y}$	e_k	$\bar{y}$	e_k	$\bar{y}$
21	14,34,0	19,32,0	∞	∞	5,33,0	4,17,0
546	25,58,0	37,69,0	∞	∞	6,40,0	4,18,0
1050	24,54,0	36,64,0	∞	∞	6,40,0	4,23,0

REFERENCES

[1] O. AXELSSON, *Conjugate gradient type methods for unsymmetric and inconsistent systems of linear equations*, Linear Algebra Appl., 29 (1980), pp. 1–16.

[2] P. N. BROWN AND A. C. HINDMARSH, *Reduced storage matrix methods in stiff ode systems*, J. Appl. Math. Comp., 31 (1989), pp. 40–91.

[3] T. F. CHAN, *Deflated decomposition of solutions of nearly singular systems*, Tech. Report 225, Department of Computer Science, Yale University, New Haven, CT, 1982.

[4] ———, *Deflation techniques and block-elimination algorithms for solving bordered singular systems*, Tech. Report 226, Department of Computer Science, Yale University, New Haven, CT, 1982.

[5] T. F. CHAN AND D. C. RESASCO, *Generalized deflated block-elimination*, Tech. Report 337, Department of Computer Science, Yale University, New Haven, CT, 1985.

[6] T. F. CHAN AND Y. SAAD, *Iterative methods for solving bordered systems with applications to continuation methods*, SIAM J. Sci. Statist. Comput., 6 (1985), pp. 438–451.

[7] S. N. CHOW, J. MALLET-PARET, AND J. A. YORKE, *Finding zeros of maps: Homotopy methods that are constructive with probability one*, Math. Comp., 32 (1978), pp. 887–899.

[8] P. CONCUS AND G. H. GOLUB, *A generalised conjugate gradient method for nonsymmetric systems of linear equations*, in Lecture Notes in Economics and Mathematical Systems, 134, R. Glowinski and J. L. Lions, eds., Springer-Verlag, Berlin, 1976, pp. 56–65.

[9] E. J. CRAIG, *Iteration procedures for simultaneous equations*, Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA, 1954.

[10] J. E. DENNIS, JR. AND K. TURNER, *Generalized conjugate directions*, Linear Algebra Appl., 88/89 (1987), pp. 187–209.

[11] S. C. EISENSTAT, H. C. ELMAN, AND M. H. SCHULTZ, *Variational iterative methods for nonsymmetric systems of linear equations*, SIAM J. Numer. Anal., 5 (1983), pp. 345–357.

[12] H. C. ELMAN, *Iterative methods for large, sparse, nonsymmetric systems of linear equations*, Ph.D. thesis, Yale University, New Haven, CT, 1982.

[13] D. K. FADEEV AND V. N. FADEEVA, *Computational Methods of Linear Algebra*, Freeman, London, 1963.

[14] P. E. GILL AND W. MURRAY, *Newton-type methods for unconstrained and linearly constrained optimization*, Math. Programming, 28 (1974), pp. 311–350.

[15] L. A. HAGEMAN AND D. M. YOUNG, *Applied Iterative Methods*, Academic Press, New York, 1981.

[16] M. R. HESTENES, *The conjugate-gradient method for solving linear equations*, Proc. Sympos. Appl. Math., 6 (1956), pp. 83–102.

[17] M. R. HESTENES AND E. STIEFEL, *Methods of conjugate gradients for solving linear systems*, J. Res. National Bureau of Standards, 49 (1952), pp. 409–435.

[18] K. M. IRANI, M. P. KAMAT, C. J. RIBBENS, H. F. WALKER, AND L. T. WATSON, *Experiments with conjugate gradient algorithms for homotopy curve tracking*, SIAM J. Optimization, 1 (1991), pp. 222–251.

[19] K. C. JEA, *Generalised conjugate gradient acceleration of iterative methods*, Ph.D. thesis, University of Texas at Austin, Austin, TX, 1982.

[20] M. P. KAMAT, L. T. WATSON, AND J. L. JUNKINS, *A robust and efficient hybrid method for finding multiple equilibrium solutions*, in Proc. Third Internat. Symposium on Numerical Methods in Engineering, Paris, 1983, pp. 799–808.

[21] H. H. KWOK, M. P. KAMAT, AND L. T. WATSON, *Location of stable and unstable equilibrium configurations using a model trust region quasi-Newton method and tunnelling*, Comput. & Structures, 21 (1985), pp. 909–916.

[22] J. A. MEIJERINK AND H. A. VAN DER VORST, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric m-matrix*, Math. Comp., 31 (1977), pp. 148–162.

[23] C. C. PAIGE AND M. A. SAUNDERS, *Solution of sparse indefinite systems of linear equations*, SIAM J. Numer. Anal., 12 (1975), pp. 617–629.

[24] W. C. RHEINBOLDT AND J. V. BURKARDT, *Algorithm* 596*: A program for a locally parameterized continuation process*, ACM Trans. Math. Software, 9 (1983), pp. 236–241.

[25] Y. SAAD, *Krylov subspace methods for solving large unsymmetric linear systems*, Math. Comp., 37 (1981), pp. 105–126.

[26] ———, *Practical use of some Krylov subspace methods for solving indefinite and unsymmetric linear systems*, Tech. Report 214, Department of Computer Science, Yale University, New Haven, CT, 1982.

[27] Y. SAAD AND M. H. SCHULTZ, *Conjugate gradient-like algorithm for solving nonsymmetric linear systems*, Math. Comp., 44 (1985), pp. 417–424.

[28] ———, GMRES*: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[29] P. K. W. VINSOME, *Orthomin, an iterative method for solving sparse sets of simultaneous linear equations*, in Proc. Fourth Symposium on Reservoir Simulation, Society of Petroleum Engineers of the AIME, 1976, pp. 149–159.

[30] H. F. WALKER, *Implementation of the* GMRES *method using Householder transformations*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 152–163.

[31] ———, *Implementations of the* GMRES *method*, Comput. Phys. Comm., 53 (1989), pp. 311–320.

[32] L. T. WATSON, *An algorithm that is globally convergent with probability one for a class of nonlinear two-point boundary value problems*, SIAM J. Numer. Anal., 16 (1979), pp. 394–401.

[33] ———, *A globally convergent algorithm for computing fixed points of* C^2 *maps*, Appl. Math. Comp., 5 (1979), pp. 297–311.

[34] ———, *Globally convergent homotopy methods: A tutorial*, Tech. Report 87–13, Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, MI, 1985.

[35] ———, *Numerical linear algebra aspects of globally convergent homotopy methods*, SIAM Rev., 28 (1986), pp. 529–545.

[36] L. T. WATSON, S. C. BILLUPS, AND A. P. MORGAN, HOMPACK*: A suite of codes for globally convergent homotopy algorithms*, ACM Trans. Math. Software, 13 (1987), pp. 281–310.

[37] O. WIDLUND, *A Lanczos method of a class of nonsymmetric systems of linear equations*, SIAM J. Numer. Anal., 15 (1978), pp. 801–812.

[38] D. M. YOUNG AND K. C. JEA, *Generalised conjugate gradient acceleration of nonsymmetrizable iterative methods*, Linear Algebra Appl., 34 (1980), pp. 159–194.