SIAM J. OPTIMIZATION
Vol. 1, No. 2, pp. 222-251, May 1991

EXPERIMENTS WITH CONJUGATE GRADIENT ALGORITHMS FOR HOMOTOPY CURVE TRACKING*

KASHMIRA M. IRANI[†], MANOHAR P. KAMAT[‡], CALVIN J. RIBBENS[†], HOMER F. WALKER[§], AND LAYNE T. WATSON[†]

Abstract. There are algorithms for finding zeros or fixed points of nonlinear systems of equations that are globally convergent for almost all starting points, i.e., with probability one. The essence of all such algorithms is the construction of an appropriate homotopy map and then tracking some smooth curve in the zero set of this homotopy map. HOMPACK is a mathematical software package implementing globally convergent homotopy algorithms with three different techniques for tracking a homotopy zero curve, and has separate routines for dense and sparse Jacobian matrices. The HOMPACK algorithms for sparse Jacobian matrices use a preconditioned conjugate gradient algorithm for the computation of the kernel of the homotopy Jacobian matrix, a required linear algebra step for homotopy curve tracking. Here variants of the conjugate gradient algorithm are implemented in the context of homotopy curve tracking and compared with Craig's preconditioned conjugate gradient method used in HOMPACK. The test problems used include actual large scale, sparse structural mechanics problems.

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, and significant progress on both aspects has been made recently [37], [52].

* Received by the editors December 12, 1989; accepted for publication (in revised form) October 15, 1990.

† Department of Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, Virginia 24061. The work of these authors was supported in part by Department of Energy grant DE-FG05-88ER25068, National Aeronautics and Space Administration grant NAG-1-1079, National Aeronautics and Space Administration contract NAS1-18471-24, National Science Foundation grant CTS-8913198, and Air Force Office of Scientific Research grant 89-0497.

‡ School of Aerospace Engineering, Georgia Institute of Technology, Atlanta, Georgia 30332.

§ 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, Army grant DAAL03-88-K, and National Science Foundation grant DMS-8800995.

Homotopies are a traditional part of topology, and only recently have begun to be used for practical numerical computation. The (globally convergent probability-one) 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, which is frequently of interest, the resulting homotopy zero curves must be dealt with as they are, bifurcations, ill-conditioning, etc. The homotopy zero curves for artificial-parameter generic homotopies obey strict smoothness conditions, which 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. Not just any random perturbation will suffice to create a globally convergent probability-one (generic) homotopy, e.g., the perturbation implied by discretization is generally not sufficient to produce a probability-one homotopy map.

If the objective is to solve a "parameter-free" system of equations, $F(x) = 0$, then extra attention can be 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 globally convergent probability-one homotopies is to solve fixed-point and zero-finding problems with homotopies whose zero curves do not have bifurcations and other singular and ill-conditioned behavior. The mathematical software package HOMPACK, used here for comparative purposes, is designed for globally convergent probability-one homotopies.

The theory and algorithms for functions $F(x)$ with small dense Jacobian matrices $DF(x)$ are well developed, which is not the case for large sparse $DF(x)$, the topic of this paper. Solving large sparse nonlinear systems of equations via homotopy methods involves sparse rectangular linear systems of equations and iterative methods for the solution of such sparse systems. Preconditioning techniques are used to make the iterative methods more efficient.

Section 2 discusses the zero-finding problem and the normal flow homotopy algorithm. Section 3 introduces iterative methods for solving invertible linear systems. Section 4 discusses the linear algebra details of homotopy curve tracking and various algorithmic possibilities for that. Section 5 presents the numerical results of the implementation of the various algorithms on several test problems. Some general conclusions from these results are drawn in §6.

2. Globally convergent homotopy algorithms. The philosophy of globally convergent probability-one homotopy algorithms is to create homotopies whose zero curves are well behaved with well-conditioned Jacobian matrices and that reach a solution for almost all choices of a parameter. These homotopies are used to solve fixed-point and zero-finding problems.

Let B be the closed unit ball in n-dimensional real Euclidean space E^n, and let $f : B \to B$ be a C^2 map. The fixed-point problem is to solve $x = f(x)$. Define $\rho_a : [0,1) \times B \to E^n$ by

$$\rho_a(\lambda, x) = \lambda(x - f(x)) + (1 - \lambda)(x - a). \tag{1}$$

The fundamental result [10] is that for almost all a in the interior of B, there is a zero curve $\gamma \subset [0,1) \times B$ of ρ_a, along which the Jacobian matrix $D\rho_a(\lambda, x)$ has rank n, emanating from $(0, a)$, and reaching a point $(1, \bar{x})$, where $\bar{x}$ is a fixed point of f. Thus with probability one, picking a starting point $a \in \text{int } B$ and following γ leads to a fixed point $\bar{x}$ of f. An important distinction between standard continuation and modern probability-one homotopy algorithms is that for the latter λ is not necessarily monotonically increasing along γ. Indeed, part of the power of probability-one homotopy algorithms derives from the lack of a monotonicity requirement for λ.

The zero-finding problem

$$F(x) = 0, \tag{2}$$

where $F : E^n \to E^n$ is a C^2 map, is more complicated. Suppose that 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 $\lambda = 1$. γ does not intersect itself and is disjoint from any other zeros of ρ_a. The globally convergent homotopy algorithm is to pick $a \in E^m$ (which uniquely determines x_0), and then track the homotopy zero curve γ starting at $(0, x_0)$ until the point $(1, \bar{x})$ is reached.

There are many different algorithms for tracking the zero curve γ; the mathematical software package HOMPACK [52], [53] supports three such algorithms: ordinary differential equation-based, normal flow, and augmented Jacobian matrix. Small dense and large sparse Jacobian matrices require substantially different algorithms. Large nonlinear systems of equations with sparse symmetric Jacobian matrices occur in many engineering disciplines (the symmetry in the problems of interest here is due to the fact that the Jacobian matrix is actually the Hessian of a potential energy function). In this paper, we consider only the zero finding problem $F(x) = 0$, the normal flow curve tracking algorithm, and large sparse symmetric Jacobian matrices $DF(x)$ stored in a packed skyline data structure.

Consider the homotopy map

$$\rho_a(x, \lambda) = \lambda F(x) + (1 - \lambda)(x - a). \tag{3}$$

The matrix $D_x\rho_a(x, \lambda) = \lambda DF(x) + (1 - \lambda)I$ is symmetric and sparse with a "skyline" structure. Such matrices are typically stored in 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

$$\frac{d}{ds}\rho_a(x(s),\lambda(s)) = [D_x\rho_a(x(s),\lambda(s)), D_\lambda\rho_a(x(s),\lambda(s))]\begin{pmatrix} dx/ds \\ d\lambda/ds \end{pmatrix} = 0, \tag{4}$$

$$\left\| \left(\frac{dx}{ds}, \frac{d\lambda}{ds} \right) \right\|_2 = 1, \tag{5}$$

$$x(0) = a, \quad \lambda(0) = 0. \tag{6}$$

Since the Jacobian matrix has rank n along γ, the derivative $(dx/ds, d\lambda/ds)$ is uniquely determined by (4), (5) and continuity, and the initial value problem (4)–(6) can be solved for $x(s), \lambda(s)$. From (4) it can be seen that the unit tangent $(dx/ds, d\lambda/ds)$ to γ is in the 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 $P^{(1)} = (x(s_1),\lambda(s_1))$, $P^{(2)} = (x(s_2),\lambda(s_2))$ 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

$$Z^{(0)} = p(s_2 + h), \tag{7}$$

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

$$\begin{aligned} p(s_1) &= (x(s_1),\lambda(s_1)), \quad & p'(s_1) &= (dx/ds(s_1), d\lambda/ds(s_1)), \\ p(s_2) &= (x(s_2),\lambda(s_2)), \quad & p'(s_2) &= (dx/ds(s_2), d\lambda/ds(s_2)), \end{aligned}$$

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

$$Z^{(k+1)} = Z^{(k)} - \left[D\rho_a(Z^{(k)})\right]^+ \rho_a(Z^{(k)}), \qquad k = 0,1,\cdots, \tag{8}$$

where $\left[D\rho_a(Z^{(k)})\right]^+$ 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 (8) 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

$$[D\rho_a]\Delta Z = -\rho_a. \tag{9}$$

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 (9), the optimal step size estimation, and the endgame to obtain the solution at $\lambda = 1$ are in [52], [53].

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 $[D\rho_a]y = 0$. This can be elegantly and efficiently done for small dense matrices [47], [51], 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 (9) where the leading $n \times n$ submatrix $D_x\rho_a$ of $D\rho_a$ is symmetric and sparse. This paper considers various iterative methods for solving such linear systems of equations.

3. Iterative methods for invertible linear systems.

Nonsquare systems of the form (9), involved in the tangent vector and normal flow iteration calculations, are converted to equivalent square linear systems of the form

$$Ay = \begin{pmatrix} B & f \\ c^t & d \end{pmatrix} y = b, \tag{10}$$

where the $n \times n$ matrix B is bordered by the vectors f and c to form a larger system of dimension $(n+1) \times (n+1)$. In the present context $B = D_x\rho_a(x,\lambda)$ is symmetric and sparse, but A is not necessarily symmetric.

Iterative methods are used for solving these linear systems. (If B has only a couple nonpositive eigenvalues, direct methods are a viable alternative; this issue is addressed later.) Iterative methods compute a sequence of approximate solutions $\{x_i\}$ which converge to the exact solution x by some algorithm of the form

$$x_{i+1} = F_i(x_0, x_1, \cdots, x_i),$$

where x_0 is an arbitrary initial guess and F_i may be linear or nonlinear.

Iterative methods require the coefficient matrix A in the algorithm, generally only to compute matrix-vector products. Since matrix-vector computations are quite inexpensive for sparse problems, iterative methods have low computational cost per iteration. Iterative methods are also attractive because they have low storage requirements, due to the fact that at each iteration, only a small number of vectors of length $N = n+1$ need to be computed and stored to calculate the next iterate x_{i+1}, and A itself can be generated or stored compactly. Thus iterative methods are sometimes more attractive than direct methods for solving large sparse linear systems of equations.

Iterative methods such as the successive over-relaxation (SOR) algorithm [43] and the alternating direction implicit (ADI) algorithm [57] require the estimation of scalar parameters. The conjugate gradient procedure [24] is an efficient algorithm for solving symmetric positive definite systems which requires no such estimates. For many years, the only iterative methods known to converge for general nonsymmetric problems were the conjugate gradient method applied to the normal equations [24] and Lanczos' biconjugate gradient algorithm [29]. Other early conjugate gradient-like methods for nonsymmetric problems which avoided the use of the normal equations were the generalized conjugate gradient method of Concus and Golub [11]–[12] and Widlund [56], and Orthomin by Vinsome [44]. These methods apply only to matrices with positive definite symmetric part, although with preconditioning they can in principle be used to solve more general problems [18]. Other conjugate gradient-like methods for more general problems were proposed by Axelsson [1]–[3]; Eisenstat, Elman, and Schultz [17]; Jea [25]; Saad [39]; Young and Jea [58]–[59]; and Saad and Schultz [41]. Preconditioning techniques that have been effective for symmetric, positive definite systems include the incomplete LU factorization [30], [31], the modified incomplete LU factorization [15], [22], and the SSOR preconditioning [57]. Most of these extend naturally to nonsymmetric problems. A lot of work has also been done comparing these various iterative methods and the preconditioning techniques [9], [14], [18], [41]. Unfortunately very little of this existing theory is directly applicable to the sparse linear systems arising from homotopy curve tracking, because they are nonsquare, generally indefinite, and lack special structure typical of PDE problems.

The rate of convergence of conjugate gradient-type methods depends on the symmetry, inertia, spectrum, and condition number of the coefficient matrix. There are efficient conjugate gradient algorithms for solving linear systems with symmetric positive definite coefficient matrices, whereas no comparable theory exists for general systems with nonsymmetric or indefinite A. This paper compares the relative performance of conjugate gradient-type algorithms for solving nonsymmetric or indefinite linear systems of the form $Ax = b$ arising from globally convergent homotopy algorithms, in terms of execution time, storage requirements, and the number of iterations required to converge.

Let Q be an $N \times N$ nonsingular matrix. The solution to $Ax = b$ can also be obtained by solving the system

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

The use of such an auxiliary matrix is known as *preconditioning*. The goal of preconditioning is to decrease the computational effort required to solve linear systems of equations by increasing the rate of convergence of an iterative method. For preconditioning to be effective, the faster convergence must outweigh the costs of applying the preconditioning, so that the total cost of solving the linear system is lower. The preconditioned coefficient matrix $\tilde{A}$ is usually not explicitly computed or stored. The main reason for this is that although A is sparse, $\tilde{A}$ may not be. The extra work of preconditioning, then, occurs in the preconditioned matrix-vector products involving Q^{-1}. The main storage cost for preconditioning is usually for Q, which typically is stored, so that one extra array is required to handle the preconditioning operation.

As mentioned above, one iterative method known to converge for general nonsymmetric problems is the conjugate gradient method applied to the normal equations.

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 similar 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 conjugate gradient algorithm. Once a solution vector z is obtained, the vector y from the original system can be computed as $y = A^tz$. The drawback of this technique is that, while the coefficient matrix is symmetric and positive definite, the convergence rate depends on $\text{cond}(AA^t) = (\text{cond}(A))^2$ rather than on $\text{cond}(A)$; see [18] for a precise statement.

An implementation of the conjugate gradient algorithm in which y is computed directly, without reference to z, any approximations of z, or AA^t is due to Craig [13] and is described in [19] and [23]. (Of course, the convergence rate still depends on $\text{cond}(AA^T) = (\text{cond}(A))^2$ in general.) Craig's preconditioned algorithm is:

```
**choose** $y_0, Q$;
**set** $r_0 = b - Ay_0$;
**set** $\tilde{r}_0 = Q^{-1}r_0$;
**set** $p_0 = A^tQ^{-t}\tilde{r}_0$;
**for** $i = 0$ **step** 1 until convergence **do**
**begin**
    $a_i = \frac{(\tilde{r}_i, \tilde{r}_i)}{(p_i, p_i)}$;
    $y_{i+1} = y_i + a_i p_i$;
    $\tilde{r}_{i+1} = \tilde{r}_i - a_i Q^{-1}Ap_i$;
    $b_i = \frac{(\tilde{r}_{i+1}, \tilde{r}_{i+1})}{(\tilde{r}_i, \tilde{r}_i)}$;
    $p_{i+1} = A^tQ^{-t}\tilde{r}_{i+1} + b_i p_i$;
**end**
```

Here (x, y) denotes the inner product of x and y. For this algorithm, a minimum of $5(n+1)$ storage locations is required (in addition to that for A). The vectors y, $\tilde{r}$, and p all require their own locations; $Q^{-t}\tilde{r}$ can share with Ap; $Q^{-1}Ap$ can share with $A^tQ^{-t}\tilde{r}$. The computational cost per iteration of this algorithm is:

(a) two preconditioning solves ($Q^{-1}v$ and $Q^{-t}v$);

(b) two matrix-vector products (Av and A^tv);

(c) $5(n+1)$ multiplications (the inner products (p,p) and $(\tilde{r},\tilde{r})$, ap, bp, and $aQ^{-1}Ap$).

3.1. Alternatives for solving (10). There are three main approaches to solving (10):

(1) In the block factorization approach to the problem, a block elimination algorithm is used instead of working with the whole matrix A directly. Such an algorithm would take advantage of the special properties of the submatrix B.

(2) The general approach works directly with the whole matrix A without taking any special advantage of the fact that the submatrix B contained in A is symmetric.

(3) The splitting approaches lie somewhere between (1) and (2). Here A is split into the sum of a symmetric matrix M and a low rank correction L. These methods also take advantage of the fact that the leading submatrix B is symmetric and can use conjugate gradient algorithms requiring a symmetric coefficient matrix.

APPROACH 1 (block factorization methods). The linear system (10) can also be written as

$$Ay = \begin{pmatrix} B & f \\ c^t & d \end{pmatrix} \begin{pmatrix} y' \\ y'' \end{pmatrix} = \begin{pmatrix} b' \\ b'' \end{pmatrix}.$$

A block-elimination algorithm [5] would be:

```
**factor** $B$;
**solve** $Bv = f$;
**solve** $Bw = b'$;
**compute** $y'' = (b'' - c^t w)/(d - c^t v)$;
**compute** $y' = w - y''v$.
```

With such block factorization methods, the work consists mainly of one factorization of B (assuming that is possible) and two backsolves with the factors of B. Observe that block elimination will frequently fail in the homotopy context, because even though rank $A = n + 1$ and rank $(B \quad f) =$ rank $D\rho_a = n$, it may very well happen that $B = D_x\rho_a$ is singular (rank $B = n - 1$). Singular B can be handled by deflation techniques [5]–[8], resulting in a direct algorithm very similar to other direct algorithms discussed below under matrix splittings. If the deflated systems were solved iteratively, this would constitute yet another iterative algorithm with no apparent advantage over the other iterative algorithms considered here. Deflation and block elimination will not be considered further.

APPROACH 2 (general methods). These algorithms work on the nonsymmetric A directly. If y_0 is an initial approximation of y, and r_0 the corresponding residual vector $r_0 = b - Ay_0$, then the Krylov subspace methods consist of finding an approximate solution belonging to the affine subspace $y_0 + K_j$, where K_j is the Krylov subspace generated by $r_0, Ar_0, \cdots A^{j-1}r_0$. There are several such methods besides Craig's method known as Orthomin(k) [44], Orthodir and Orthores [59], the Incomplete Orthogonalization Method [40], the GCR method [18], and the GMRES method [42]. Typical of these methods is the preconditioned Orthomin(k) algorithm, given by:

```
choose $y_0$;
set $r_0 = b - Ay_0$;
set $\tilde{r_0} = Q^{-1}r_0$;
set $p_0 = r_0$;
for $i = 0$ step 1 until convergence do
begin
    $a_i = \frac{(\tilde{r}_i, Q^{-1}Ap_i)}{(Q^{-1}Ap_i, Q^{-1}Ap_i)}$;
    $y_{i+1} = y_i + a_i p_i$;
    $\tilde{r}_{i+1} = \tilde{r}_i - a_i Q^{-1}Ap_i$;
    $b_j^{(i)} = -\frac{(Q^{-1}A\tilde{r}_{i+1}, Q^{-1}Ap_j)}{(Q^{-1}Ap_j, Q^{-1}Ap_j)}$,    $j = \max\{0, i-k+1\}, \cdots i$;
    $p_{i+1} = \tilde{r}_{i+1} + \sum_{j=(i-k+1)_+}^{i} b_j^i p_j$;
end
```

where $(i-k+1)_+ \equiv \max\{0, i-k+1\}$. As for the storage costs, Ap_j is overwritten by $Q^{-1}Ap_j$ and $A\tilde{r}$ by $Q^{-1}A\tilde{r}$. Thus storage is required for y, $\tilde{r}$, $\{p_j\}_{(i-k+1)_+}^{i}$, $\{Q^{-1}Ap_j\}_{(i-k+1)_+}^{i}$, and $Q^{-1}A\tilde{r}$.

However, Orthomin(k) is guaranteed to converge only for positive definite coefficient matrices A (equivalently, A with positive definite symmetric part $(A + A^t)/2$). (Some authors define *positive definite* only for symmetric matrices, while others say A is positive definite if $x^tAx > 0$ for all $x \neq 0$ in E^n, whether A is symmetric or not. This latter meaning is used here.) More general systems $Ax = b$, where A is not positive definite, can be solved by applying Orthomin(k) to the transformed system $ZAx = Zb$, where Z is nonsingular and ZA is positive definite. The matrix Z must be known and used explicitly in the iteration, a major obstacle to the general applicability of Orthomin(k).

The GCR method may also break down if the coefficient matrix is not positive definite. Although Orthodir does not break down in this case, it is observed to have stability problems [40]. GMRES, on the other hand, although equivalent to GCR for positive definite coefficient matrices, can be used to solve systems for which the coefficient matrix is not positive definite, and requires half as much storage as GCR. However, GMRES requires storage of the order of the number of iterations performed for convergence. Hence, the algorithm is used iteratively, i.e., it is restarted every k steps, where k is a fixed parameter, leading to the following GMRES(k) algorithm [42]:

```
choose $y_0$, tol;
set $r_0 = b - Ay_0$;
while $\|r_0\| >$ tol do
```

```
begin
    set $v_1 = r_0/\|r_0\|$;
    for $j = 1$ step 1 until $k$ do
    begin
        for $i = 1$ step 1 until $j$ do $h_{i,j} = (Av_j, v_i)$;
        $\tilde{v}_{j+1} = Av_j - \sum_{i=1}^{j} h_{i,j}v_i$;
        $h_{j+1,j} = \|\tilde{v}_{j+1}\|$;
        $v_{j+1} = \tilde{v}_{j+1} \,/\, h_{j+1,j}$
    end
    Solve $\min_x \big\| \|r_0\|e_1 - \bar{H}_k x\big\|$ for $x_k$ where $\bar{H}_k$ is described in [42];
    set $y_0 = y_0 + V_k x_k$;        set $r_0 = b - Ay_0$
end
```

In practice the algorithm calculates $\|r_j\|$ ($\|r_j\|$ can be calculated without forming y_j or $r_j = b - Ay_j$ explicitly) at each iteration of the j loop, and breaks the j loop if $\|r_j\| <$ tol [42], [45]. Also, it is important in practice that the classical Gram–Schmidt process in the inner j loop be replaced by the modified Gram–Schmidt process to ensure stability. GMRES(k), 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.

APPROACH 3 (coefficient matrix splittings). There are several ways of splitting the coefficient matrix A in (10) as the sum of a symmetric matrix M and a low rank matrix L. The choice (c^t, d) as the last row of M gives the splitting

$$M = \begin{pmatrix} B & c \\ c^t & d \end{pmatrix}, \qquad L = ue_{n+1}^t, \qquad u = \begin{pmatrix} f - c \\ 0 \end{pmatrix}, \tag{11}$$

where e_{n+1} is a vector with 1 in the $(n+1)$st component and zeros elsewhere. There are many reasonable choices for (c^t, d), discussed later (recall that (c^t, d) can be almost any, in the sense of Lebesgue measure, vector for which (10) produces a solution to the true problem (9) or $[D\rho_a]y = 0$). The linear system $Ay = b$ is then solved by applying iterative techniques to two linear systems with coefficient matrix M followed by the Sherman–Morrison formula; the algorithmic details of this are in the next section. Another possibility would be to compute a symmetric indefinite factorization of M, and not use iterative methods at all. However, this destroys the skyline data structure containing M, and a tacit assumption here is that the skyline data structures must be preserved. If it were acceptable to destroy the skyline data structure, this direct approach would likely be the most efficient of all for skyline sparsity patterns, but would not generalize to arbitrary sparsity patterns (which the iterative methods will).

Another way of splitting up the coefficient matrix A is

$$A = D - A_L - A_U, \tag{12}$$

where D is the diagonal of A, A_L is the strict lower triangle of $-A$, and A_U is the strict upper triangle of $-A$. The symmetric successive over-relaxation (SSOR) iterative method [57] is the following two stage algorithm:

$$(D-\omega A_L)x_{i+1/2}=[(1-\omega)D+\omega A_U]x_i+\omega b,$$
$$(D-\omega A_U)x_{i+1}=[(1-\omega)D+\omega A_L]x_{i+1/2}+\omega b,$$

where ω is a real scalar parameter between 0 and 2. With

$$Q=\frac{1}{\omega(2-\omega)}(D-\omega A_L)D^{-1}(D-\omega A_U),$$

this method can be formulated as a one step algorithm

$$Qx_{i+1}=(Q-A)x_i+b.$$

In the homotopy context, D^{-1} frequently does not exist, and a diagonal matrix Σ such that $\left[\mathrm{diag}\ (A+\Sigma)\right]^{-1}$ does exist may not be of low rank (meaning that the solution for A cannot be easily recovered from the solution for $A+\Sigma$). Consequently, SSOR and methods based on similar splittings are of limited utility in the homotopy context; in fact, SSOR failed for all the test problems in §5. A few experiments were also tried with SSOR ($\omega=1$) as a preconditioner, but it was not competitive, and is not considered further here.

3.2. Some preconditioning techniques. This section considers some preconditioning techniques to be used in conjunction with the algorithms just described. Preconditioning matrices constructed from approximate factorizations of the coefficient matrix are considered first. A lower triangular matrix L and an upper triangular matrix U that are in some sense approximations of the factors in the LU factorization of A, but that are also sparse, are constructed. The preconditioning matrix is the product $Q=LU$. The heuristic used to insure that the preconditioning is inexpensive to implement is to force the factors to be sparse by allowing nonzeros only within a specified set of locations.

(i) *The incomplete* LU *factorization* (ILU). Let Z be a set of indices contained in $\{(i,j)\mid 1\le i,j\le N,\ i\ne j\}$, typically where A is known to be zero. The incomplete LU factorization is given by $Q=LU$, where L and U are lower triangular and unit upper triangular matrices, respectively, that satisfy

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

The incomplete LU factorization algorithm is:

```
for $i=1$ step 1 until $N$ do
    for $j=1$ step 1 until $N$ do
        if $((i,j)\notin Z)$ then
            begin
                $s_{ij}=A_{ij}-\sum_{t=1}^{\min\{i,j\}-1} L_{it}U_{tj}$;
                if $(i\ge j)$ then $L_{ij}=s_{ij}$ else $U_{ij}=s_{ij}/L_{ii}$;
            end
```

It can happen that L_{ii} is zero in this algorithm. In this case L_{ii} is set to a small positive number, so that $Q_{ii} \neq A_{ii}$.

(ii) *The modified incomplete* LU *factorization* (MILU). Let Z be the set of indices that determine the zero structure, and assume that $(i,i) \notin Z$, $1 \leq i \leq N$. The modified incomplete LU factorization is given by $Q = LU$, where L and U are lower triangular and unit upper triangular matrices, respectively, that satisfy

$$\begin{cases} L_{ij} = U_{ij} = 0, & (i,j) \in Z, \\ Q_{ij} = A_{ij}, & (i,j) \notin Z, i \neq j, \\ \sum_{j=1}^{N} (Q_{ij} - A_{ij}) = \alpha, & 1 \leq i \leq N, \end{cases}$$

where α is a scalar. The modified incomplete LU factorization algorithm is:

```
for $i = 1$ step 1 until $N$ do
    begin
        $L_{ii} = \alpha$;
        for $j = 1$ step 1 until $N$ do
            begin
                $s_{ij} = A_{ij} - \sum_{t=1}^{\min\{i,j\}-1} L_{it} U_{tj}$;
                if $((i,j) \notin Z)$ then
                    begin
                        if $(i > j)$ then $L_{ij} = s_{ij}$ ;
                        if $(i = j)$ then $L_{ii} = L_{ii} + s_{ii}$ ;
                        if $(i < j)$ then $\tilde{U}_{ij} = s_{ij}$ ;
                    end
                else $L_{ii} = L_{ii} + s_{ij}$;
            end
        for $j = i + 1$ step 1 until $n$ do
            $U_{ij} = \tilde{U}_{ij} / L_{ii}$;
    end
```

Since LU factorizations preserve a skyline sparsity structure, the MILU factorization is the same as the ILU factorization for $\alpha = 0$. The motivation for the MILU factorization is to control the elements of Q where it does not match A, at least in an average sense. In the homotopy context here with skyline A and $\alpha > 0$, Q can be construed as an approximation to A that is closer to (or more) positive definite than A.

4. Algorithms for computing $\ker[D\rho_a]$. As discussed in §2 for the normal flow algorithm, a corrector step ΔZ is the unique minimum norm solution of (9), which uses the solution of the rectangular linear system $[D\rho_a]y = 0$. This section describes various algorithms for the solution of such linear systems.

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

$$A = \begin{pmatrix} D_x\rho_a(x,\lambda) & D_\lambda\rho_a(x,\lambda) \\ c^t & d \end{pmatrix}, \tag{13}$$

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 $D\rho_a$ can be found by solving the linear system of equations

$$Ay = \alpha e_{n+1} = b, \tag{14}$$

where $(c^t \quad d)\bar{y} = \alpha$.

The coefficient matrix A in the linear system of equations (14) has a very special structure which can be exploited in several ways. Note that the leading $n \times n$ submatrix of A is $D_x\rho_a$, which is symmetric and sparse, but possibly indefinite. Since symmetry is advantageous for some algorithms, A can be made symmetric and invertible by choosing $c = D_\lambda\rho_a$. If 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 $c^t = (D_\lambda\rho_a)^t$ is not a linear combination of the rows of the symmetric matrix $D_x\rho_a$, and the

$$\text{row rank} \begin{bmatrix} D_x\rho_a \\ (D_\lambda\rho_a)^t \end{bmatrix} = n. \tag{15}$$

Finally,

$$\begin{pmatrix} D_\lambda\rho_a \\ * \end{pmatrix}$$

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

$$\text{rank} \begin{bmatrix} D_x\rho_a \\ (D_\lambda\rho_a)^t \end{bmatrix} = n, \tag{16}$$

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 $(D_\lambda\rho_a)^t$ will make the $(n+1)$st column independent. Let $\bar{A}$ denote A at $(\bar{x}, \bar{\lambda})$. Since $\dim[\ker(\bar{A})] \le 1$, $\bar{A}y = 0$ implies $y = \alpha\bar{y}$, and thus with $\bar{y}^t = (\hat{y}^t, \bar{y}_{n+1})$, $(D_\lambda\rho_a(\bar{x},\bar{\lambda}))^t\hat{y} + d\,\bar{y}_{n+1} = 0$. Choosing any $\beta \ne 0$ and solving $(D_\lambda\rho_a(\bar{x},\bar{\lambda}))^t\hat{y} + d\,\bar{y}_{n+1} = \beta$ for d ($\bar{y}_{n+1} \ne 0$ 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

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

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. □

Another approach is to attack (14) indirectly as follows. Write

$$A = M + L, \tag{17}$$

where

$$M = \begin{pmatrix} D_x\rho_a(\bar{x},\bar{\lambda}) & c \\ c^t & d \end{pmatrix}, \tag{18}$$

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

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

$$y = \left[I - \frac{M^{-1}ue_{n+1}^t}{(M^{-1}u)^t e_{n+1} + 1} \right] M^{-1}b, \tag{19}$$

which requires the solution of two linear systems $Mz = u$ and $Mz = b$ with the sparse, symmetric, invertible matrix M. The scheme (17)–(19) was proposed in [27], and further investigated by Chan and Saad [9]. First, the HOMPACK approach to the solution of these linear systems will be discussed.

Let $|\bar{y}_k| = \max_i |\bar{y}_i|$ define the index k. In HOMPACK, $(c^t \quad d) = e_k^t$, where e_k is a vector with 1 in the kth component and zeros elsewhere. Hence (13) becomes

$$A = \begin{pmatrix} D\rho_a(x,\lambda) \\ e_k^t \end{pmatrix}.$$

The kernel of $D\rho_a$ can be found by solving the linear system of equations

$$Ay = \bar{y}_k e_{n+1} = b.$$

Again, splitting the coefficient matrix as

$$A = M + L$$

gives a symmetric

$$M = \begin{pmatrix} D_x\rho_a(\bar{x},\bar{\lambda}) & * \\ & e_k^t \end{pmatrix},$$

$$L = ue_{n+1}^t, \qquad u = \begin{pmatrix} D_\lambda\rho_a(\bar{x},\bar{\lambda}) \\ 0 \end{pmatrix} - e_k(1-\delta_{k,n+1}).$$

(The Kronecker $\delta_{k,n+1}$ takes care of the special case $k = n+1$.) Then the Sherman–Morrison formula (19) is used for the solution y of the linear system (14). Craig's preconditioned algorithm is used for solving the systems $Mz = u$ and $Mz = b$. The preconditioning matrix Q is taken as a positive definite approximation of M (the *Gill–Murray preconditioner*, described in detail in [21] and [52]). The details are intricate, but essentially Q is computed as a Cholesky factorization of $M + \Sigma$, where Σ is a positive semidefinite diagonal matrix chosen to guarantee that Q is well conditioned. This algorithm is not especially well matched to the skyline data structure.

If M had only one or two negative eigenvalues, then after several rank one updates making M positive definite, a direct Cholesky factorization could be obtained, and then the solution to (14) recovered after several more applications of (19). This direct algorithm for solving (14) would be effective for such M, but since only one or two negative eigenvalues for M cannot be assumed in general (the M for a large shallow dome problem has many negative eigenvalues along the unloading portions of the equilibrium curve), a direct rank one update/Cholesky scheme would not be suitable for HOMPACK.

There are several other schemes which could be used instead of the one in HOMPACK for finding the kernel of $D\rho_a$, for example,

(i) using different last rows for the augmented coefficient matrix A of (13), i.e., other vectors $(c^t \quad d)$ instead of e_k^t;

(ii) using other preconditioners on M;

(iii) using other algorithms for the solution of the linear systems $Mz = u$ and $Mz = b$, e.g., Orthomin(k), SSOR, etc., instead of Craig's algorithm;

(iv) doing (i), (ii), or (iii) on the nonsymmetric A directly instead of on the symmetric M in the splitting $A = M + L$.

Combining the preconditioning techniques with the algorithms for solving linear systems with different last rows for A produces a large number of possible methods. The next section focuses on a subset of these possible methods and compares their efficiency.

5. Numerical experiments.

Of the various algorithmic possibilities mentioned in the previous section, those considered further are given short names in the list below. Some possibilities do not make sense or are impractical in the homotopy context, and thus are not considered. Of the almost all mathematically valid choices for the last row $(c^t \quad d)$ of A, only e_k^t (the easiest to implement), $c = D_\lambda\rho_a(\bar{x},\bar{\lambda})$ (the easiest symmetrization of A), and the tangent vector $\bar{y}^t$ at the previous point on the zero curve (the optimal choice for conditioning, since it is orthogonal to the top n rows of A at $(\bar{x},\bar{\lambda})$) are used.

SC	– $A = M + L$ splitting, Craig's method with M, no preconditioning;
SCGM	– $A = M+L$ splitting, Craig's method with M, Gill–Murray preconditioning from HOMPACK;
SCILU	– $A = M + L$ splitting, Craig's method with M, incomplete LU preconditioning;
SCMILU	– $A = M + L$ splitting, Craig's method with M, modified incomplete LU preconditioning;
C	– no splitting, Craig's method with A, no preconditioning;
CGM	– no splitting, Craig's method with A, Gill–Murray preconditioning from HOMPACK;
CILU	– no splitting, Craig's method with A, incomplete LU preconditioning;
CMILU	– no splitting, Craig's method with A, modified incomplete LU preconditioning.
SR	– $A = M + L$ splitting, GMRES(2) with M, no preconditioning;
SRGM	– $A = M + L$ splitting, GMRES(2) with M, Gill–Murray preconditioning from HOMPACK;
SRILU	– $A = M+L$ splitting, GMRES(2) with M, incomplete LU preconditioning;
SRMILU	– $A = M + L$ splitting, GMRES(2) with M, modified incomplete LU preconditioning;
R	– no splitting, GMRES(2) with A, no preconditioning;
RGM	– no splitting, GMRES(2) with A, Gill–Murray preconditioning from HOMPACK;
RILU	– no splitting, GMRES(2) with A, incomplete LU preconditioning;
RMILU	– no splitting, GMRES(2) with A, modified incomplete LU preconditioning.

The test problems are now described in detail, beginning with the shallow arch structural response problem.

5.1. Shallow arch. The equations of equilibrium of the arch are obtained from the principle of the stationary value of the total potential energy, according to which, of all the kinematically admissible displacement fields, the one that makes the total potential energy of a structure stationary also satisfies its equations of equilibrium. The total potential energy π of a structure is given by the sum of its strain energy and the potential of external loads.

The shallow arch of Fig. 1 is discretized by an assemblage of straight p-q frame elements such as those described in [26]. A frame element is a structural component that is initially straight and undergoes axial, bending, and torsional deformation resulting from finite displacements and rotations of its ends (nodes) p and q. The displacements of the end q relative to the end p are

$$\begin{pmatrix} \delta u \\ \delta v \\ \delta w \end{pmatrix} = [T]_p \begin{pmatrix} X_q - X_p \\ Y_q - Y_p \\ Z_q - Z_p \end{pmatrix} - \begin{pmatrix} L \\ 0 \\ 0 \end{pmatrix} + [T]_p \begin{pmatrix} U_q - U_p \\ V_q - V_p \\ W_q - W_p \end{pmatrix},$$

where L is the initial rigid body length, and U_i, V_i, W_i ($i = p$ or q) denote the global displacements of the nodes. The matrix $[T]_p$ can be shown to be [26] $[T]_p = [T_1(\phi_x, \phi_y, \phi_z)]\,[T_1(\theta_{xp}, \theta_{yp}, \theta_{zp})]$ with

$$[T_1(\alpha_x, \alpha_y, \alpha_z)] = \begin{pmatrix} c_y c_z & c_y s_z & -s_y \\ -c_x s_z + s_x s_y c_z & c_x c_z + s_x s_y s_z & s_x c_y \\ s_x s_z + c_x s_y c_z & -s_x c_z + c_x s_y s_z & c_x c_y \end{pmatrix},$$

$c_i = \cos\alpha_i$ and $s_i = \sin\alpha_i$ for $i = x$, y, and z. Angles ϕ_x, ϕ_y, and ϕ_z are the initial orientation angles and angles θ_{xp}, θ_{yp}, and θ_{zp} are the rigid body rotations of the end p. In the equation for $[T]_p$, Euler angle transformations are implied with the order of the rotations being α_z, α_y, and α_x.

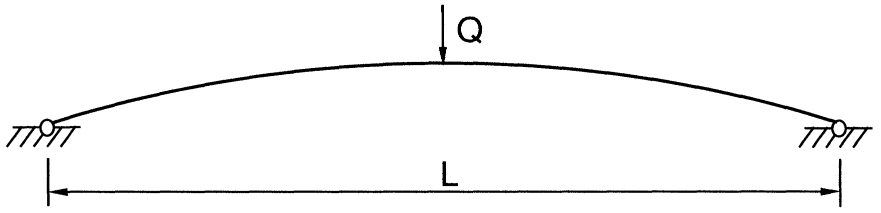

FIG. 1. *Shallow arch.*

Similarly, with the restriction of small relative rotations within the element, the rotations ψ_x, ψ_y, ψ_z of the end q relative to the end p are

$$\begin{pmatrix} \psi_x \\ \psi_y \\ \psi_z \end{pmatrix} = [T]_p \begin{pmatrix} \theta_{xq} - \theta_{xp} \\ \theta_{yq} - \theta_{yp} \\ \theta_{zq} - \theta_{zp} \end{pmatrix}.$$

With the relative generalized displacements $(\delta u, \delta v, \delta w)$ and (ψ_x, ψ_y, ψ_z) known, the usual deformation patterns of the reference axis of the beam element in the corotational coordinate system are assumed to be

$$\begin{aligned} u(\xi) &= \xi\frac{\delta u}{L}, & v(\xi) &= \frac{1}{L}(3\xi^2 - 2\xi^3)(\delta v - z_s\psi_x) + (\xi^3 - \xi^2)\psi_z, \\ \beta &= \xi\psi_x, & w(\xi) &= \frac{1}{L}(3\xi^2 - 2\xi^3)(\delta w + y_s\psi_x) - (\xi^3 - \xi^2)\psi_y, \end{aligned}$$

where $\xi = x/L$ and y_s and z_s are the coordinates of the shear center of the cross section of the beam. The strain at any point (y, z) on the cross-section of the frame element can be shown to be

$$\begin{aligned} \epsilon = \frac{\delta u}{L} &- \eta\left[\frac{6}{L}(1 - 2\xi)(\delta v - z_s\psi_x) + 2(3\xi - 1)\psi_z\right] \\ &- \zeta\left[\frac{6}{L}(1 - 2\xi)(\delta w + y_s\psi_x) - 2(3\xi - 1)\psi_y\right], \end{aligned}$$

with $\eta = y/L$ and $\zeta = z/L$. In these equations it is implicitly assumed that the lateral displacements and twists are referenced to a longitudinal axis through the shear center, while the axial displacements and rotations are referenced to the centroidal axis.

The total potential energy of such a discretized model of the arch can be expressed as

$$\pi = \sum_{e=1}^{m} U^e - q^t Q,$$

where U^e is the strain energy of the eth element, $e = 1, \cdots, m$, q is the vector of nodal displacement degrees of freedom of the entire model and Q is the vector of externally applied loads. The strain energy U^e of the eth frame element is given by

$$U^e = \frac{E}{2}\int_V \epsilon^2 dv = \frac{E}{2}\int_0^{L_e}\int_{A_e} \epsilon^2 dA\, dx,$$

where ϵ is the strain of a point (x, y, z) of the beam, which was derived above. Substituting for ϵ and doing the integration gives

$$U^e = U_{\mathrm{p-q}} = \frac{E}{2L_e}\left\{A_e(\delta u)^2 + \frac{12}{L_e^2}I_z\left[(\delta v)^2 + \frac{1}{3}L_e^2\psi_z^2 - L_e\delta v\,\psi_z\right]\right.$$
$$\left. + \frac{12}{L_e^2}I_y\left[(\delta w)^2 + \frac{1}{3}L_e^2\psi_y^2 + L_e\delta w\,\psi_y\right]\right\},$$

where A_e is the cross-sectional area, and I_y and I_z are the cross-sectional moments of inertia about the y and z axes, respectively. It is evident that the potential energy π of the model is a highly nonlinear function of the nodal displacements. The equations of equilibrium of the model are obtained by setting the variation $\delta\pi$ to zero, or equivalently by

$$\nabla\pi = 0.$$

Closed form analytical expressions for $\nabla\pi$ can be obtained with some difficulty, but obtaining the Jacobian matrix of $\nabla\pi$ analytically seems out of the question. Hence the Jacobian matrix of the equilibrium equations is obtained by finite difference approximations.

By symmetry only half the arch need be modelled, and the results here are for the arch parameters used in [28], with a full arch load of 3000 lbs. This is just below the limit point. To go through the limit point and along the unloading portion of the equilibrium curve apparently requires very accurate Jacobian matrices and numerical linear algebra, and none of these iterative linear equation solvers used in HOMPACK were able to go past the limit point without tweaking the HOMPACK step size control parameters.

5.2. Shallow dome. The shallow dome of Fig. 2 is built up from space truss elements with three global displacement degrees of freedom (u_1, u_2, u_3) at each of the two nodes. For an element of original length L between its two nodes p and q, the change in length δL is given by

$$\delta L = \left[\sum_{i=1}^{3}(x_{qi} + u_{qi} - x_{pi} - u_{pi})^2\right]^{1/2} - \left[\sum_{i=1}^{3}(x_{qi} - x_{pi})^2\right]^{1/2},$$

where x_{ij}, u_{ij}, $i = p, q$; $j = 1, 2, 3$ are the global coordinates and displacements of the two nodes. This can be simplified to

$$\delta L = L\left[1 + \sum_{i=1}^{3}\left(\frac{2(\Delta x_i \Delta u_i)}{L^2} + \frac{(\Delta u_i)^2}{L^2}\right)\right]^{1/2} - L,$$

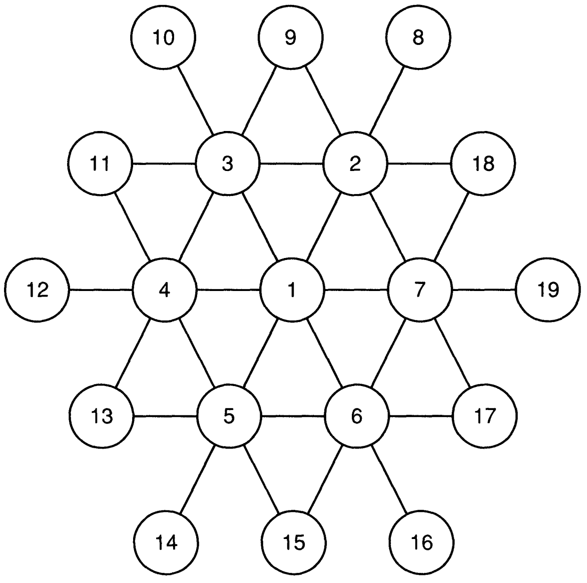

FIG. 2. *Triangulation for* 21 *degree of freedom shallow dome.*

where Δ is the difference operator for the q and p values. Accordingly, the axial strain in the eth element is

$$\epsilon^e = \frac{\delta L}{L} = \left[1 + \sum_{i=1}^{3}\left(\frac{2(\Delta x_i \Delta u_i)}{L^2} + \frac{(\Delta u_i)^2}{L^2}\right)\right]^{1/2} - 1.$$

The strain energy of the eth element in a purely linearly elastic response is given by

$$U_s^e = \frac{E}{2}\int_V (\epsilon^e)^2 dV = \frac{EA^eL^e}{2}(\epsilon^e)^2,$$

where E and A are the Young's modulus and cross-sectional area, respectively, of the eth element.

The total potential energy of the dome is then given by

$$\pi = \sum_{e=1}^{m} U_s^e - U^T Q,$$

where U_i, $i = 1, \cdots, 6$ are the six components u_{qk}, u_{pk}, $k = 1, 2, 3$, and Q is the generalized force vector. The equations of equilibrium of the model are then obtained by setting

$$\nabla\pi = \sum_{e=1}^{m} EA^eL^e\nabla\epsilon^e - Q = 0.$$

Both the gradient of π as well as its Hessian can be evaluated explicitly without resorting to finite differencing operations as in the case of the frame element used to model the shallow arch.

The effect of modelling the shallow dome with truss elements in concentric rings is that changing the number of truss elements changes the model and its behavior. Thus the dome problems with different degrees of freedom reported in the tables are qualitatively different, with different buckling loads and bifurcation points. The results reported here are for shallow domes with base radius 720 and sphere radius 3060, and a point load at the very top.

5.3. Artificial turning point problem. The turning point problem is 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 was chosen artificially to produce turning points in γ. HOMPACK had no difficulty going through numerous turning points using iterative linear equation solvers.

Tables 1–6 show some timing results for these three test problems. An asterisk indicates either that the iterative linear equation solver stalled, γ was lost because of inaccurate tangents from the linear equation solver, or the time was at least an order of magnitude larger than anything else in the table. The times are for tracking the entire zero curve γ and thus represent the solution of many linear systems of varying degrees of difficulty. The experiments were 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. The MILU algorithms used $\alpha = 1$. There is asymmetry in the tables because some possibilities do not make sense. For instance, there is no CGM with e_k because the Gill–Murray preconditioner requires a symmetric matrix, and there are no S* with $D_\lambda \rho_a$ since the choice $c^t = \big(D_\lambda \rho_a\big)^t$ makes A symmetric so there is no need to split off a symmetric matrix M from A.

6. Discussion and conclusions. The convergence rate of conjugate gradient iterative methods for linear systems depends on the spectrum and the condition number of the coefficient matrix, and therefore one would predict $\bar{y}^t$ should be a better choice for the last row of A than e_k^t. Since $\bar{y}$ is orthogonal to the rows of $D\rho_a(\bar{x}, \bar{\lambda})$, a good approximation to the first n rows $D\rho_a(x, \lambda)$ of A, one expects A with $\bar{y}$ to be better conditioned than with e_k. Tables 1, 3, and 5 show that apparently this better conditioning does not compensate for the extra work involved in using $\bar{y}$. Although $\bar{y}$ is sometimes better than e_k, there seems to be no strong evidence that $\bar{y}$ is worth the trouble.

Figure 3 shows the condition numbers of A and $Q^{-1}A$ along γ for the shallow arch problem with $n = 29$ (CGM). The shallow arch problem is indeed a hard problem, but Fig. 3 alone would not suggest that—see the discussion of the shallow arch problem's spectra later. The Jacobian matrix $D_x \rho_a$ becomes indefinite near $\lambda = .88$, at which

TABLE 1
Execution time in seconds for shallow arch problem.

	SC		SCGM		SCILU		SCMILU	
n	e_k^t	$\bar{y}^t$	e_k^t	$\bar{y}^t$	e_k^t	$\bar{y}^t$	e_k^t	$\bar{y}^t$
29	1108	947	468	818	599	470	908	975
47	16904	17593	7322	10105	5674	5957	11538	12390
	SR		SRGM		SRILU		SRMILU	
29	*	*	461	*	559	443	*	*
47	*	*	5314	*	5796	6332	*	*

TABLE 2
Execution time in seconds for shallow arch problem.

	C			CILU			CMILU			CGM
n	e_k^t	$\bar{y}^t$	$D_\lambda \rho_a$	e_k^t	$\bar{y}^t$	$D_\lambda \rho_a$	e_k^t	$\bar{y}^t$	$D_\lambda \rho_a$	$D_\lambda \rho_a$
29	856	884	919	533	458	443	841	845	900	464
47	14205	13591	14606	5794	5807	6776	9943	10968	10135	6921
	R			RILU			RMILU			RGM
29	*	*	*	443	431	506	*	*	*	429
47	*	*	*	5355	5304	5697	*	*	*	5260

TABLE 3
Execution time in seconds for shallow dome problem.

	SC		SCGM		SCILU		SCMILU	
n	e_k^t	$\bar{y}^t$	e_k^t	$\bar{y}^t$	e_k^t	$\bar{y}^t$	e_k^t	$\bar{y}^t$
21	57	86	108	57	21	25	92	141
546	3127	4803	2710	1787	492	630	4892	6687
1050	5615	8553	5107	3177	887	1133	8259	11672
	SR		SRGM		SRILU		SRMILU	
21	*	*	*	*	14	15	*	*
546	*	*	*	*	299	335	*	*
1050	*	*	*	*	559	625	*	*

TABLE 4
Execution time in seconds for shallow dome problem.

	C			CILU			CMILU			CGM
n	e_k^t	$\bar{y}^t$	$D_\lambda \rho_a$	e_k^t	$\bar{y}^t$	$D_\lambda \rho_a$	e_k^t	$\bar{y}^t$	$D_\lambda \rho_a$	$D_\lambda \rho_a$
21	46	47	47	16	16	16	68	79	69	89
546	2495	2545	2573	355	369	365	3037	3585	3094	2233
1050	4504	4691	4690	632	665	651	5536	6327	5570	4313
	R			RILU			RMILU			RGM
21	*	*	*	11	12	11	*	*	*	*
546	*	*	*	230	241	232	*	*	*	*
1050	*	*	*	425	446	430	*	*	*	*

TABLE 5
Execution time in seconds for turning point problem.

	SC		SCGM		SCILU		SCMILU	
n	e_k^t	$\bar{y}^t$	e_k^t	$\bar{y}^t$	e_k^t	$\bar{y}^t$	e_k^t	$\bar{y}^t$
20	28	36	12	13	6	6	39	47
60	266	356	41	50	20	22	163	213
125	1635	2310	127	170	54	65	568	795
250	3026	3767	228	267	95	109	1032	1335
500	6279	7783	448	501	189	207	2130	2656
1000	14150	17768	1077	1174	434	490	4874	6052
	SR		SRGM		SRILU		SRMILU	
20	1301	*	13	18	4	4	120	*
60	*	*	47	*	13	14	*	*
125	*	*	*	*	36	40	*	*
250	*	*	*	*	60	69	*	*
500	*	*	*	*	119	131	*	*
1000	*	*	*	*	274	296	*	*

TABLE 6
Execution time in seconds for turning point problem.

	C			CILU			CMILU			CGM
n	e_k^t	$\bar{y}^t$	$D_\lambda\rho_a$	e_k^t	$\bar{y}^t$	$D_\lambda\rho_a$	e_k^t	$\bar{y}^t$	$D_\lambda\rho_a$	$D_\lambda\rho_a$
20	17	19	21	4	7	4	24	26	27	5
60	167	176	186	13	22	13	109	112	118	22
125	1117	1132	1384	38	64	42	412	421	446	85
250	2296	1925	3873	66	110	74	765	699	726	134
500	4741	3899	8352	129	210	148	1573	1381	1465	260
1000	11577	9335	20375	323	493	353	3605	3031	3308	617
	R			RILU			RMILU			RGM
20	*	*	1263	3	3	3	43	70	75	6
60	*	*	*	9	10	9	*	*	2016	*
125	*	*	*	26	31	28	*	*	*	*
250	*	*	*	45	51	46	*	*	*	*
500	*	*	*	88	98	88	*	*	*	*
1000	*	*	*	199	225	202	*	*	*	*

point the Gill–Murray preconditioner ceases being nearly perfect. This figure is typical for the Gill–Murray preconditioner. (During the course of the experiments it was observed that points far from γ frequently generate much worse conditioned problems. This has important implications for curve tracking strategy, because large steps along γ will be offset by expensive numerical linear algebra to return to γ.)

Tables 2, 4, 6 show that there is no clear winner between e_k, $\bar{y}$, and $D_\lambda\rho_a$, and further that there is little correlation between the algorithm and the best choice for c. One is tempted to pick CGM with $c^t = \left(D_\lambda\rho_a\right)^t$ over SCGM with $c^t = e_k^t$, based on Tables 5, 6, and the fact that CGM only solves one linear system per tangent vector computation, as opposed to two linear systems with M for the splitting algorithms.

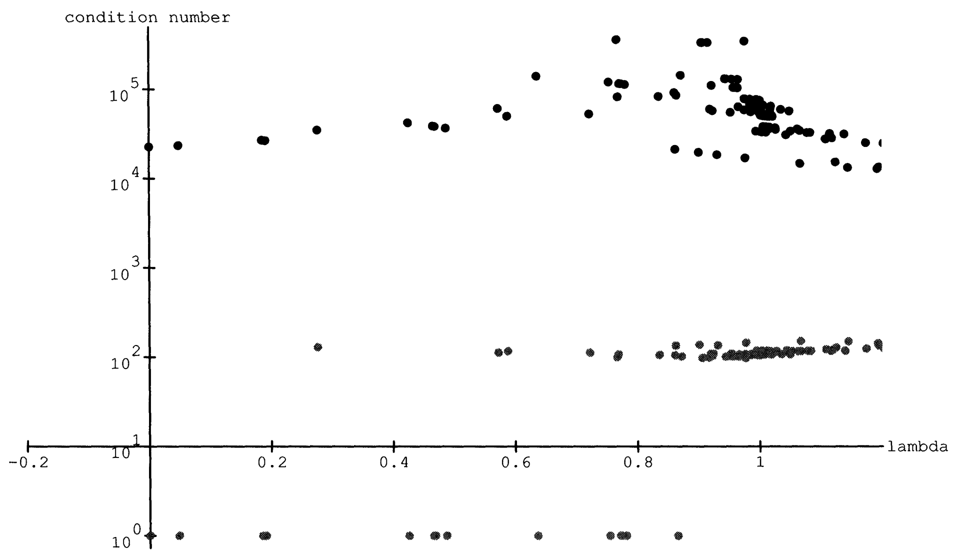

FIG. 3. *Condition numbers for* A (*black dots*) *and* $Q^{-1}A$ (*grey dots*) *against* λ *along* γ; *shallow arch,* $n = 29$, CGM.

However, from Tables 1 and 2, SCGM with e_k is substantially better than CGM with $D_\lambda\rho_a$. This demonstrates that only counting the number of linear system solves can be dangerously misleading. The results do indicate that for a given choice of c^t, when no preconditioning is used or when MILU preconditioning is used, it is slightly more efficient to use the no-splitting strategy than the splitting. However, with ILU preconditioning the differences between corresponding splitting and no-splitting cases are not at all significant.

Tables 1–6 seem to strongly support an argument for CILU as the best Craig method, even though the ILU factorization fails to exist at turning points, and is unstable whenever A is indefinite. What is not indicated in the tables, though, are all the homotopy curve tracking runs which failed because the ILU preconditioner failed to exist or generated an overflow, or the difficulty caused HOMPACK by inaccurate tangents resulting from the ILU. Because of this potential catastrophic failure or instability, the ILU preconditioner would never be seriously considered for robust homotopy algorithms. Still, the tables do show why numerical analysts' paranoia about unstable algorithms is not shared by engineers.

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 $c^t = (D_\lambda\rho_a)^t$ took 443 (6092) seconds for the shallow arch problem with $n = 29$ (47)), but that is no consolation for homotopy curve tracking.

GMRES(k) has a solid theoretical justification [42], and has been used very successfully in a variety of contexts [4], [42], [45], [46]. Nevertheless, GMRES(k) with $k < n$ performed unacceptably on the test problems here without preconditioning.

For the shallow arch problem with $n = 29$ and tol $= 10^{-12}$, GMRES(29) on A with $c^t = \left(D_\lambda \rho_a\right)^t$ took 591 seconds, somewhat better than C or SC and comparable to CGM and SCGM. 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 symmetric and positive definite up to $\lambda = .88$, and mildly indefinite from there to $\lambda = 1$. For the turning point problem with $n = 20$, tol $= 10^{-12}$, $c^t = \left(D_\lambda \rho_a\right)^t$, 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.

The tables also show results for GMRES(2), implemented with all the same preconditioners and choices of $(c^t \quad d)$ as was Craig's method. $k = 2$ was chosen because a preconditioned GMRES(2) requires exactly the same amount of storage as the preconditioned Craig's method, although, of course, the storage penalty for $k = 5$, say, is not significant. Numerous other runs were made with $k = 1, 3, 5$, or 10, but there was no substantial difference from $k = 2$ on the larger problems. In virtually all cases the asterisks in the tables correspond to a stalled residual norm somewhere along γ. It was noted, though, that many of the linear systems along γ were solved efficiently by GMRES(2). Perhaps the most disappointing failure was that of RGM on the shallow dome problem even for $n = 21$, because the Gill–Murray preconditioner was fairly good there. It is evident from the tables that GMRES(2), without nearly perfect preconditioning (ILU), is unsuitable for use in a general, robust homotopy curve tracking code like HOMPACK.

There are some theoretical results concerning the convergence of GMRES(k) given by Saad and Schultz [42]. 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 $k = n$.

Tables 7–12 show the average, maximum, and minimum number of conjugate gradient iterations per linear system solution along the homotopy zero curve γ for the same algorithms as Tables 1–6. Such iteration statistics give an intuitive feel for how the algorithms behave, and are sometimes very revealing. Tables 7 and 8 show that symmetry does improve the algorithms' efficiency, and that all other things being equal, achieving symmetric coefficient matrices is worthwhile. (The S* algorithms based on symmetry are not uniformly better, because all other things are not equal.) Note that in all cases for Craig's method the maximum number of conjugate gradient iterations is less than or equal to eight times the average, which says that the convergence behavior is fairly consistent. On the other hand the range between the minimum and maximum (for the C* algorithms) is as great as 3 to 536 (C for $n = 1000$ in Table 12), showing that there is a wide variation in the difficulty of the linear systems encountered along γ. The convergence behavior of GMRES(2) is not as consistent as for Craig's method, with the maximum being as much as 70 times the average (SRGM for $n = 47$ in Table 7).

TABLE 7

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

	SC		SCGM		SCILU		SCMILU	
n	e_k^t	$\bar{y}^t$	e_k^t	$\bar{y}^t$	e_k^t	$\bar{y}^t$	e_k^t	$\bar{y}^t$
29	66,109,1	66,101,1	4,10,1	28,40,1	2,3,1	3,3,1	39,63,1	39,58,1
47	190,313,1	194,291,1	5,10,1	37,53,1	2,3,1	3,3,1	64,103,1	65,97,1
	SR		SRGM		SRILU		SRMILU	
29	*	*	4,92,1	*	1,2,1	1,2,1	*	*
47	*	*	2,140,1	*	1,2,1	1,2,1	*	*

TABLE 8

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

	C			CILU		
n	e_k^t	$\bar{y}^t$	$D_\lambda \rho_a$	e_k^t	$\bar{y}^t$	$D_\lambda \rho_a$
29	99,127,51	91,107,38	98,120,52	3,3,2	4,5,2	3,3,2
47	265,360,109	239,305,133	265,355,105	3,3,2	4,4,2	3,3,2
	CMILU			CGM		
29	56,68,30	56,65,35	55,65,30	—	—	6,7,2
47	87,119,48	91,102,53	87,131,48	—	—	6,7,2
	R			RILU		
29	*	*	*	1,1,1	1,2,1	1,1,1
47	*	*	*	1,1,1	1,2,1	1,1,1
	RMILU			RGM		
29	*	*	*	—	—	2,2,1
47	*	*	*	—	—	2,2,1

TABLE 9

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

	SC		SCGM		SCILU		SCMILU	
n	e_k^t	$\bar{y}^t$	e_k^t	$\bar{y}^t$	e_k^t	$\bar{y}^t$	e_k^t	$\bar{y}^t$
21	17,31,1	24,36,1	16,115,1	7,46,1	2,3,1	2,3,1	14,30,1	21,32,1
546	38,75,1	54,87,1	15,113,1	9,63,1	2,3,1	3,2,1	24,45,1	37,71,1
1050	38,76,1	53,91,1	16,114,1	8,101,1	2,3,1	3,3,1	24,47,1	36,61,1
	SR		SRGM		SRILU		SRMILU	
21	*	*	*	*	1,2,1	1,2,1	*	*
546	*	*	*	*	1,2,1	1,2,1	*	*
1050	*	*	*	*	1,2,1	1,2,1	*	*

TABLE 10

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

	C			CILU		
n	e_k^t	$\bar{y}^t$	$D_\lambda\rho_a$	e_k^t	$\bar{y}^t$	$D_\lambda\rho_a$
21	26,36,14	26,36,14	26,36,14	2,3,2	2,3,2	2,3,2
546	58,81,17	57,82,17	58,82,18	2,3,2	2,3,2	2,3,2
1050	58,87,18	59,91,18	58,83,18	2,3,2	2,3,2	2,3,2
	CMILU			CGM		
21	19,25,8	22,28,8	19,24,8	—	—	23,113,2
546	34,47,12	38,52,12	34,45,11	—	—	22,111,2
1050	34,49,12	38,50,12	34,49,13	—	—	23,113,2
	R			RILU		
21	*	*	*	1,1,1	1,1,1	1,1,1
546	*	*	*	1,1,1	1,1,1	1,1,1
1050	*	*	*	1,1,1	1,1,1	1,1,1
	RMILU			RGM		
21	*	*	*	—	—	*
546	*	*	*	—	—	*
1050	*	*	*	—	—	*

TABLE 11

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

	SC		SCGM		SCILU		SCMILU	
n	e_k^t	$\bar{y}^t$	e_k^t	$\bar{y}^t$	e_k^t	$\bar{y}^t$	e_k^t	$\bar{y}^t$
20	21,28,1	24,29,1	4,6,1	5,7,1	2,2,1	2,2,1	17,27,1	19,26,1
60	60,100,1	69,87,1	4,8,1	5,8,1	2,3,1	2,3,1	21,37,1	25,39,1
125	127,261,1	154,264,1	5,9,1	6,11,1	2,3,1	2,3,1	26,51,1	31,51,1
250	139,302,1	150,246,1	5,11,1	5,10,1	2,2,1	2,3,1	27,60,1	30,55,1
500	149,314,1	164,281,1	5,11,1	5,10,1	2,2,1	2,3,1	28,62,1	31,53,1
1000	151,312,1	162,289,1	5,11,1	5,11,1	2,2,1	2,3,1	28,64,1	31,56,1
	SR		SRGM		SRILU		SRMILU	
20	732,4446,1	*	6,58,1	8,75,1	1,1,1	1,1,1	49,315,1	*
60	*	*	6,54,1	*	1,1,1	1,1,1	*	*
125	*	*	*	*	1,1,1	1,1,1	*	*
250	*	*	*	*	1,1,1	1,1,1	*	*
500	*	*	*	*	1,1,1	1,1,1	*	*
1000	*	*	*	*	1,1,1	1,1,1	*	*

The Gill–Murray preconditioner is clearly excellent, as shown by the average number of iterations in Tables 7–12 and Fig. 3. It is more robust than the ILU and MILU preconditioners in the presence of turning points and indefinite $D_x\rho_a$. However, the shallow dome problem (Tables 3, 4, 9, and 10) shows that the Gill–Murray precon-

TABLE 12

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

	C			CILU		
n	e_k^t	$\bar{y}^t$	$D_\lambda \rho_a$	e_k^t	$\bar{y}^t$	$D_\lambda \rho_a$
20	24,29,1	24,28,1	26,31,1	2,2,1	4,5,1	2,3,1
60	70,86,1	69,84,1	74,91,2	2,3,1	4,7,1	2,3,2
125	159,292,1	151,232,1	179,328,3	2,3,1	4,5,1	2,3,2
250	196,404,1	150,246,1	231,407,3	2,3,1	4,5,1	2,3,2
500	216,427,1	165,337,1	268,489,3	2,3,1	4,6,1	2,3,2
1000	224,446,1	164,323,1	285,536,3	2,3,1	4,5,1	3,3,2
	CMILU			CGM		
20	20,23,1	20,23,1	20,26,3	—	—	3,9,2
60	26,36,1	25,34,1	26,36,2	—	—	3,12,1
125	33,49,1	31,48,1	33,53,3	—	—	5,15,2
250	36,61,1	30,45,1	32,48,1	—	—	4,15,2
500	38,74,1	31,53,1	33,53,3	—	—	5,16,2
1000	39,75,1	31,50,1	33,54,3	—	—	5,16,2
	R			RILU		
20	*	*	1905,21000,2	1,1,1	1,2,1	1,1,1
60	*	*	*	1,1,1	1,2,1	1,1,1
125	*	*	*	1,1,1	1,2,1	1,1,1
250	*	*	*	1,1,1	1,2,1	1,1,1
500	*	*	*	1,1,1	1,2,1	1,1,1
1000	*	*	*	1,1,1	1,2,1	1,1,1
	RMILU			RGM		
20	47,110,2	61,200,2	79,226,2	—	—	4,92,1
60	*	*	568,12486,2	—	—	*
125	*	*	*	—	—	*
250	*	*	*	—	—	*
500	*	*	*	—	—	*
1000	*	*	*	—	—	*

ditioner may do a very poor job indeed on strongly indefinite matrices (which occur on the unloading parts of the shallow dome equilibrium curve). While reducing the average number of iterations, the Gill–Murray preconditioner actually increases the maximum number of iterations compared to the unpreconditioned algorithm.

It would be possible to test separately each aspect of the iterative linear system solving algorithms, such as convergence rate, sensitivity to starting point, cost of preconditioning, storage cost, computational complexity per iteration, etc. What ultimately matters, however, is the combined performance of the total algorithm on a wide range of typical realistic problems. Measuring the performance along homotopy zero curves for nontrivial problems is an attempt to measure the overall performance *in situ*.

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 is somewhat slower, especially for very large or strongly indefinite problems. With somewhat imperfect preconditioning, Craig's method is more robust than GMRES(k) for $k \ll n$ for homotopy curve tracking.

Acknowledgments. The authors are indebted to the referees for excellent suggestions, and to Tony Chan, Dianne O'Leary, Philippe Toint, and David Young for comments on this work.

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] ———, *Solution of linear systems of equations: iterative methods*, in Sparse Matrix Techniques, Springer-Verlag, New York, 1976, pp. 1–51.

[3] O. Axelsson, S. Brinkkemper, and V.P. Ilin, *On some versions of incomplete block-matrix factorization iterative methods*, Linear Algebra Appl., 58 (1984), pp. 3–15.

[4] P. N. Brown and A. C. Hindmarsh, *Reduced storage matrix methods in stiff* ODE *systems*, J. Appl. Math. Comp., 31 (1989), pp. 40–91.

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

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

[7] ———, *On the existence and computation of* LU*-factorizations with small pivots*, Tech. Report 227, Department of Computer Science, Yale University, New Haven, CT, 1982.

[8] T. F. Chan and D. C. Resasco, *Generalized deflated block-elimination*, Tech. Report 337, Department of Computer Science, Yale University, New Haven, CT, 1985.

[9] 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.

[10] 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.

[11] 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.

[12] P. Concus, G. H. Golub, and D. P. O'Leary, *A generalised conjugate gradient method for the numerical solution of elliptic partial differential equations*, in Sparse Matrix Computations, J. R. Bunch and D. J. Rose, eds., Academic Press, New York, 1976, pp. 309–352.

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

[14] J. E. Dennis, Jr. and K. Turner, *Generalized conjugate directions*, Linear Algebra Appl., 88/89 (1987), pp. 187–209.

[15] T. Dupont, R. P. Kendall, and K. K. Rachford Jr., *An approximate factorization procedure for solving self-adjoint elliptic difference equations*, SIAM J. Numer. Anal., 5 (1968), pp. 559–573.

[16] S. C. Eisenstat, *Efficient implementation of a class of conjugate methods*, SIAM J. Sci. Statist. Comput., 2 (1981), pp. 1–4.

[17] 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.

[18] H. C. Elman, *Iterative methods for large, sparse, nonsymmetric systems of linear equations*, Ph.D. thesis, Computer Science Department, Yale University, 1982.

[19] D. K. Fadeev and V. N. Fadeeva, *Computational Methods of Linear Algebra*, W. H. Freeman, London, 1963.

[20] R. Fletcher, *Conjugate gradient methods for indefinite systems*, in Numerical Analysis Dundee 1975, G. A. Watson, ed., Springer-Verlag, New York, 1976, pp. 73–89.

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

[22] I. Gustafsson, *A class of first order factorizations*, BIT, 18 (1978), pp. 142–156.

[23] M. R. Hestenes, *The conjugate gradient method for solving linear equations*, in Proc. Sympos. Appl. Math., 6, Numer. Anal., AMS, New York, 1956, pp. 83–102.

[24] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Res. Nat. Bur. Standards, 49 (1952), pp. 409–435.

[25] K. C. Jea, *Generalised conjugate gradient acceleration of iterative methods*, Ph.D. thesis, Mathematics Department, University of Texas, Austin, TX, 1982.

[26] M. P. Kamat, *Nonlinear transient analysis by energy minimization–theoretical basis for the* ACTION *computer code*, NASA Report CR-3287, National Aeronautics and Space Administration, 1980.

[27] 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, France, 1983, pp. 799–808.

[28] 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.

[29] C. Lanczos, *Solution of systems of linear equations by minimized-iterations*, J. Res. Nat. Bur. Standards, 49 (1952), pp. 33–53.

[30] 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.

[31] ———, *Guidelines for the usage of incomplete decompositions in solving sets of linear equations as occur in practical problems*, Tech. Report 550, Koninklijke/Shell Exploratie on Produktie Laboratorium, 1980.

[32] D. P. O'Leary, *Hybrid conjugate gradient algorithms*, Ph.D. thesis, Computer Science Department, Stanford University, Stanford, CA, 1976.

[33] ———, *The block conjugate gradient algorithm and related methods*, Linear Algebra Appl., 29 (1980), pp. 293–322.

[34] J. M. Ortega, *Efficient implementations of certain iterative methods*, SIAM J. Sci. Statist. Comput., 9 (1988), pp. 882–891.

[35] W. C. Rheinboldt, *Numerical analysis of continuation methods for nonlinear structural problems*, Comput. & Structures, 13 (1981), pp. 103–113.

[36] ———, *Numerical Analysis of Parametrized Nonlinear Equations*, Wiley-Interscience, New York, 1986.

[37] 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.

[38] J. K. Reid, *On the method of conjugate gradients for the solution of sparse systems of linear equations*, in Large Sparse Sets of Linear Equations, J. K. Reid, ed., Academic Press, New York, 1971, pp. 231–254.

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

[40] ———, *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.

[41] Y. Saad and M. H. Schultz, *Conjugate gradient-like algorithm for solving nonsymmetric linear systems*, Math. Comp., 44/170 (1985), pp. 417–424.

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

[43] R. S. Varga, *Matrix iterative analysis*, Prentice-Hall, New York, 1962.

[44] 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 American Institute of Mechanical Engineers, 1976, pp. 149–159.

[45] H. F. WALKER, *Implementations of the* GMRES *method*, Comput. Phys. Comm., 53 (1989), pp. 311–320

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

[47] L. T. WATSON, *A globally convergent algorithm for computing fixed points of C^2 maps*, Appl. Math. Comput., 5 (1979), pp. 297–311.

[48] ———, *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.

[49] ———, *Solving finite difference approximations to nonlinear two-point boundary value problems by a homotopy method*, SIAM J. Sci. Statist. Comput., 1 (1980), pp. 467–480.

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

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

[52] 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.

[53] L. T. WATSON AND D. FENNER, *Chow–Yorke algorithm for fixed points or zeros of C^2 maps*, ACM Trans. Math. Software, 6 (1980), pp. 252–260.

[54] L. T. WATSON AND M. R. SCOTT, *Solving spline-collocation approximations to nonlinear two-point boundary-value problems by a homotopy method*, Appl. Math. Comput., 24 (1987), pp. 333–357.

[55] L. T. WATSON AND L. R. SCOTT, *Solving Galerkin approximations to nonlinear two-point boundary value problems by a globally convergent homotopy method*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 768–789.

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

[57] D. M. YOUNG, *Iterative solution of large linear systems*, Academic Press, New York, 1971.

[58] D. M. YOUNG AND K. C. JEA, *Generalised conjugate gradient acceleration of iterative methods. Part 2: the nonsymmetrizable case*, Tech. Report CNA-163, Center for Numerical Analysis, University of Texas, Austin, TX, 1981.

[59] ———, *Generalised conjugate gradient acceleration of nonsymmetrizable iterative methods*, Linear Algebra Appl., 34 (1980), pp. 159–194.