Comparison of Lanczos with Conjugate Gradient Using Element Preconditioning

B. NOUR-OMID*, B. N. PARLETT†, AND A. RAEFSKY‡

Abstract. We consider $\mathbf{Ax} = \mathbf{b}$ where $\mathbf{A}$ arises from two different applications; (i) from structural analysis, (ii) from incompressible fluid flow. $\mathbf{A}$ will never be assembled but the operation $\mathbf{v} \rightarrow \mathbf{Av}$ will be carried out using the element-by-element representation. This representation bars the use of triangular factorization.

Two different types of preconditioning are examined and each is used in a comparison of the conjugate gradient method and the Lanczos algorithm using partial reorthogonalization. It appears that these element-by-element preconditioners are more powerful than diagonal scaling. Moreover, the cost of maintaining semi-orthogonality is well worth paying; the number of iterations is greatly reduced and consequently the extra burden of occasional orthogonalizations never dominated the cost of the solution algorithm in our examples. The crucial difference is that CG failed to converge on one class of examples. It is this lack of robustness that deters engineers from accepting CG into general use.

* Mechanics and Materials Engineering Laboratory, Lockheed Palo Alto Research Laboratory, 3251 Hanover Street, Palo Alto, CA 94304. This author was supported by the independent research program of LMSC.

† Department of Mathematics, and the Computer Science Division of the Department of Electrical Engineering and Computer Science, University of California at Berkeley, Berkeley, CA 94720. This author was supported by Office of Naval Research under contract number N00014-85-K-0180.

‡ Seismological Laboratory, California Institute of Technology, Pasadena, CA 91125, currently a visiting scholar at Applied Mechanics Division, Department of Mechanical Engineering, Stanford University, Stanford, CA 94305.

1. Introduction. Nonlinear transient finite element problems are generally solved by applying a step-by-step time integration procedure to the governing equations of motion. The result is a system of nonlinear algebraic equations. The solution to this system is obtained using a Newton type iteration. At the heart of this iteration is a set of linear equations

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

whose solution, $\mathbf{x}$, is the Newton correction. $\mathbf{A}$ is symmetric, positive definite and sparse. Typically, there are fewer than 50 nonzero terms in each row of $\mathbf{A}$. The number of equations, n, in this system depends on the complexity of the structure (i.e. the domain) and also on the amount of detail desired in describing the solution. For problems with small bandwidth methods based on triangular factorization of $\mathbf{A}$ are the fastest solvers. When the bandwidth of $\mathbf{A}$ is large (e.g. large complex two and three dimensional problems), the solution of the linear system may be a formidable task and alternative procedures must be considered.

The Lanczos algorithm [1] was first introduced in 1950 as a method for computing eigenvalues and the corresponding eigenvectors of a matrix. In 1952 Hestenes and Stiefel [2] introduced the method of conjugate gradients (CG) for solving linear systems of equations. In the same year, Lanczos showed that his algorithm then called the method of minimized iteration [3], can also be used to obtain the solution of a linear system of equations. In fact, these methods are closely related in the sense that they compute the same approximate solution at each step in exact arithmetic. A fact known to Lanczos and Hestenes.

In 1960, the CG method was first used to solve system of linear equations arising in structural mechanics [4]. In this paper, Lively showed that CG was not effective for solving the ill-conditioned systems that often arise in structural analysis. The popularity of the CG method vanished when it was found that for certain problems it required well over n steps to converge to the correct solution. CG was then abandoned and the more effective direct methods were adopted by structural analysts as their method of choice.

The advantages of Lanczos and CG methods was recognized when attention was focused on linear systems with large sparse matrix coefficients, see [5]. An important property of these methods is that, at each step, the solution to (1) can be obtained with out an explicit knowledge of the matrix. Only a means of computing the matrix vector product $\mathbf{Av}$ for a given vector $\mathbf{v}$ is required. This is an elegant way of exploiting the sparsity structure of $\mathbf{A}$.

The incorporation of preconditioning alleviated the difficulties associated with the slow convergence of CG method. Instead of solving (1) one solves

$$\mathbf{AM}^{-1}\mathbf{y} = \mathbf{b} \tag{2}$$

where $\mathbf{x} = \mathbf{M}^{-1}\mathbf{y}$. $\mathbf{M}$ is referred to as the preconditioning matrix. The advantages of preconditioning can also be realized by solving $\mathbf{M}^{-1}\mathbf{Ax} = \mathbf{M}^{-1}\mathbf{b}$.

Here, we focus on systems which arise from the application of the finite element method to engineering problems. Typically

$$\mathbf{A} = \sum_e \mathbf{N}_e\, \mathbf{a}_e\, \mathbf{N}_e^T \tag{3}$$

where $\mathbf{N}_e$ are long, thin Boolean connectivity matrices and $\mathbf{a}_e$ denotes the small stiffness matrix for element e. We can take advantage of this structure of $\mathbf{A}$ when using either CG or Lanczos methods to solve (1). In [6] and [7], it is pointed out that the matrix vector product

$$\mathbf{Au} = \sum_e (\mathbf{N}_e \mathbf{a}_e \mathbf{N}_e^T \mathbf{u}) \tag{4}$$

can be computed without ever assembling $\mathbf{A}$. The evaluation of $\mathbf{Au}$ using (4) requires more

arithmetic operations than that using an assembled A (assuming some compact structure where no zero entries of A are stored). The number of arithmetic operations might increase as much as three fold. However, the use of parallel and vector computers produces only a modest increase in the elapsed time and in certain cases might even reduce it.

In 1983, Hughes, Levit and Winget [8] proposed a time integration algorithm for the solution of heat conduction equations that uses an element-by-element (E×E) splitting. In the same year, Ortiz, Pinsky and Taylor [9] proposed a novel extension of the E×E procedure to the solution of dynamic equations. These E×E time integration algorithms are unconditionally stable, but they lack accuracy which limits their use. Hughes, Levit and Winget [10] reformulate the E×E procedure as an iterative solver to achieve the accuracy and stability of standard finite element algorithms. In [11] Nour-Omid and Parlett addressed the problem of preconditioning (1). The idea is to use the methods for solving differential equations presented in [8,9,10] and employ them as preconditioners. The resulting preconditioners use the element representation of A in (4), and requires no globally assembled matrix. They are defined as the product of positive definite element matrices obtained by applying a diagonal shift to the positive semi-definite element stiffness matrices. Winget and Hughes [12] further developed the ideas of element preconditioners and constructed a variation that replaces the terms on the diagonals of the element matrices with the corresponding ones in the assembled matrix. This modification also results in positive definite element matrices. A product form similar to that in [11] is then constructed using the Choleski factorization of these modified element matrices. It is worth noting that these preconditioners, though computed element-by-element, are an approximate Choleski factorization of A, see [13]. Our primary interest in element-by-element preconditioning is in keeping storage requirements down in the analysis of regular structures, but the advent of vector and parallel computing may make this approach a fast one as well, especially in three dimensions.

In section 2 we briefly describe the Lanczos algorithm and the conjugate gradient method and their inter-relation. We then turn to the problem of orthogonality loss that affects both methods. As a remedy, we consider the approach of partial reorthogonalization proposed by Simon [14]. The element preconditioners used in this study are described in section 4. The results of numerical tests on two characteristically different problems are presented in section 5.

2. Lanczos Algorithm. When used as a method for solving linear systems, the Lanczos process starts from a given initial approximation to the solution, $\mathbf{x}_0$. Associated with $\mathbf{x}_0$, define the residual vector $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$. Unless a good estimate to the desired solution is available, the best choice for $\mathbf{x}_0$ is the zero vector. Then $\mathbf{r}_0 = \mathbf{b}$. Normalizing $\mathbf{r}_0$ gives the first Lanczos vector, $\mathbf{q}_1$. Implicitly, at the end of the first step the algorithm obtains a Galerkin approximation to the solution of (1) from the one dimensional space with $\mathbf{q}_1$ as the base vector. Associated with this approximation is a new residual vector. The normalization of this residual results in the second Lanczos vector, $\mathbf{q}_2$. Repeating the Galerkin process but using the two dimensional space, $span(\mathbf{q}_1, \mathbf{q}_2)$, followed by the normalization of the residual one obtains $\mathbf{q}_3$.

At a typical step, j, the Lanczos algorithm computes a residual vector associated with a best approximation, $\mathbf{x}_k$, (in a Galerkin sense) to $\mathbf{x}$ from the j dimensional space, $span[\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_j]$. This space is often referred to as the space of trial vectors. The next Lanczos vector, $\mathbf{q}_{j+1}$, is the normalized residual $\mathbf{b} - \mathbf{A}\mathbf{x}_k$. This process is repeated until the norm of the current residual is small compared to the that of the starting residual. The Galerkin method chooses the approximate solution $\mathbf{x}_k$ by forcing the associated residual to be orthogonal to the space of trial (Lanczos) vectors. The Lanczos method computes neither $\mathbf{x}_k$ nor the associated residual. Instead it computes, at step j, a j-vector $\mathbf{s}$ that contains the weighting parameters for constructing the Galerkin solution.

Alternatively, Lanczos may be described as the Gram-Schmit orthogonalization process applied to the Krylov space, $[\mathbf{r}_0, \mathbf{A}\mathbf{M}^{-1}\mathbf{r}_0, (\mathbf{A}\mathbf{M}^{-1})^2\mathbf{r}_0, \cdots, (\mathbf{A}\mathbf{M}^{-1})^{j-1}\mathbf{r}_0]$, associated with equation (2). The

orthogonalization is performed with respect to the $\mathbf{M}^{-1}$ inner product. The result of this orthogonalization is the set of Lanczos vectors $[\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_j]$. $\mathbf{q}_{j+1}$ is obtained by orthonormalizing $(\mathbf{AM}^{-1})^j\mathbf{r}_0$ against the computed Lanczos vectors. The same vector $\mathbf{q}_{j+1}$ is obtained if $\mathbf{AM}^{-1}\mathbf{q}_j$ is used instead of $(\mathbf{AM}^{-1})^j\mathbf{r}_0$. It turns out that the components of $\mathbf{AM}^{-1}\mathbf{q}_j$ along the first $j-2$ Lanczos vectors $\mathbf{q}_1$ are zero and orthogonalization needs to be performed only against $\mathbf{q}_j$ and $\mathbf{q}_{j-1}$. The results is a vector $\mathbf{r}_j$ in the same direction as the residual due to the Galerkin approximation described above.

The algorithm can then be rewritten as the three term relation

$$\mathbf{r}_j = \beta_{j+1}\,\mathbf{q}_{j+1} = \mathbf{A}\,\mathbf{M}^{-1}\,\mathbf{q}_j - \mathbf{q}_j\,\alpha_j - \mathbf{q}_{j-1}\,\beta_j \tag{5}$$

where $\alpha_j = \mathbf{q}_j^T\,\mathbf{M}^{-1}\,\mathbf{K}\,\mathbf{M}^{-1}\,\mathbf{q}_j$ and $\mathbf{r}_j$ is normalized with respect to the inverse of the preconditioner to obtain $\mathbf{q}_{j+1}$ with normalizing factor $\beta_{j+1} = (\mathbf{r}_j^T\mathbf{M}^{-1}\mathbf{r}_j)^{1/2}$.

The j-th step of the Lanczos algorithm involves the calculation of α_j, β_{j+1}, and $\mathbf{q}_{j+1}$, in that order. In addition to the storage needs of A and M, the algorithm requires storage for 5 vectors of length n; one for each of the vectors, $\mathbf{q}_{j-1}$, $\mathbf{q}_j$, $\mathbf{r}_j$, $\mathbf{p}_j = \mathbf{M}^{-1}\mathbf{q}_j$ and $\mathbf{p}_{j-1}$. The total cost for one step of the algorithm involves one solve with the preconditioner, M, as the coefficient matrix, a multiplication of A by a vector, two inner products and four products of a scalar by a vector. A summary of the Lanczos algorithm is presented in Table 1.

After m Lanczos steps all the quantities obtained from equation (5) can be arranged in a global matrix form

$$\begin{bmatrix} & \\ \mathbf{A}\,\mathbf{M}^{-1} & \\ & \end{bmatrix}\begin{bmatrix} \\ \mathbf{Q}_m \\ \\ \end{bmatrix} - \begin{bmatrix} \\ \mathbf{Q}_m \\ \\ \end{bmatrix}\begin{bmatrix} \mathbf{T}_m \\ \\ \end{bmatrix} = \begin{bmatrix} & | \\ & | \\ 0 & |\mathbf{r}_m \\ & | \\ & | \end{bmatrix} = \mathbf{r}_m\,\mathbf{e}_m^T \tag{6}$$

Here $\mathbf{e}_m^T = (0,0,\cdots,0,1)$, $\mathbf{Q}_m$ is an $n\times m$ matrix with columns $\mathbf{q}_i$, $i = 1,2,\cdots,m$, and $\mathbf{T}_m$ is the tridiagonal matrix

$$\mathbf{T}_m = \begin{bmatrix} \alpha_1 & \beta_2 & & & & \\ \beta_2 & \alpha_2 & \beta_3 & & & \\ & \beta_3 & . & . & & \\ & & . & . & . & \\ & & & . & . & \beta_m \\ & & & & \beta_m & \alpha_m \end{bmatrix} \tag{7}$$

The orthogonality property of the Lanczos vectors, $\mathbf{Q}_m^T\,\mathbf{M}^{-1}\,\mathbf{Q}_m = \mathbf{I}_m$, where $\mathbf{I}_m$ is the $m\times m$ identity matrix, can be used in equation (6) to obtain

$$\mathbf{Q}_m^T\,\mathbf{M}^{-1}\,\mathbf{A}\,\mathbf{M}^{-1}\,\mathbf{Q}_m = \mathbf{T}_m \tag{8}$$

A Galerkin approximation to y in (2) can be constructed by taking a linear combination of the Lanczos vectors. Accordingly,

$$\mathbf{y}_m = \mathbf{Q}_m\mathbf{s} \tag{9}$$

where s satisfies the tridiagonal system of equations

$$\mathbf{T}_m\,\mathbf{s} = \mathbf{Q}^T\,\mathbf{M}^{-1}\,\mathbf{r}_0 = \beta_1\mathbf{e}_1 \tag{10}$$

Given an approximate solution vector $\mathbf{x}_0$:

(1) Set
- (a) $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$,
- (b) $\mathbf{q}_0 = 0$,
- (c) Solve $\mathbf{M}\bar{\mathbf{p}}_1 = \mathbf{r}_0$.
- (d) $\beta_1 = (\mathbf{p}_1^T\mathbf{r}_0)^{1/2}$,
- (e) $\mathbf{q}_1 = \frac{1}{\beta_1}\mathbf{r}_0$
- (f) $\mathbf{p}_1 = \frac{1}{\beta_1}\bar{\mathbf{p}}_1$

(2) for $j = 1,2,\cdots$ repeat;
- (a) $\bar{\mathbf{r}}_j = \mathbf{A}\mathbf{p}_j$
- (b) $\hat{\mathbf{r}}_j = \bar{\mathbf{r}}_j - \mathbf{q}_{j-1}\beta_j$
- (c) $\alpha_j = \mathbf{q}_j^T\mathbf{M}^{-1}\hat{\mathbf{r}}_j = \mathbf{p}_j^T\hat{\mathbf{r}}_j$
- (d) $\mathbf{r}_j = \hat{\mathbf{r}}_j - \mathbf{q}_j\alpha_j$
- (e) Solve $\mathbf{M}\bar{\mathbf{p}}_j = \mathbf{r}_j$
- (f) $\beta_{j+1} = (\mathbf{r}_j^T\mathbf{M}^{-1}\mathbf{r}_j)^{1/2} = (\bar{\mathbf{p}}_j^T\mathbf{r}_j)^{1/2}$
- (g) if residual norm is small then terminate the loop.
- (h) $\mathbf{q}_{j+1} = \frac{1}{\beta_{j+1}}\mathbf{r}_j$
- (i) $\mathbf{p}_{j+1} = \frac{1}{\beta_{j+1}}\bar{\mathbf{p}}_j$

(3) Solution $\mathbf{x} = \mathbf{x}_0 + \mathbf{M}^{-1}\mathbf{Q}_j\mathbf{s}$

Table 1. The Lanczos algorithm.

The last equality is obtained using the fact that the starting vector is $\mathbf{r}_0 = \beta_1\mathbf{q}_1$. $\mathbf{e}_1$ is the first column of the identity matrix. Equation (10) is a weak form of (2) and is obtained by first substituting the approximation to y from (9) into equation (2) to obtain the residual

$$\mathbf{g}_m = \mathbf{A}\mathbf{M}^{-1}\mathbf{Q}_m\mathbf{s} - \mathbf{b} \tag{11}$$

Orthogonalizing $\mathbf{g}_m$ against $\mathbf{Q}_m$ with respect to the $\mathbf{M}^{-1}$ inner product results in equation (10). $\mathbf{g}_m$ is simply related to $\mathbf{r}_m$ through

$$\mathbf{g}_m = \mathbf{r}_m\sigma_m \tag{12}$$

where σ_m is the bottom element of s. The norm of this residual, $\rho_m = \|\mathbf{g}_m\| = \beta_{m+1}|\sigma_m|$, can be used to monitor the convergence. Once ρ_m is sufficiently small the Lanczos algorithm is terminated and the solution is constructed using (9).

The conjugate gradient method can be derived from the Lanczos algorithm by implicitly computing the triangular factorization of $\mathbf{T}_m = \mathbf{L}_m\mathbf{D}_m\mathbf{L}_m^T$. Define $\bar{\mathbf{Z}}_m = \mathbf{M}^{-1}\mathbf{Q}_m\mathbf{L}_m^{-T}$. The columns of $\bar{\mathbf{Z}}_m$ are orthogonal with respect to A. To show this consider

$$\begin{aligned}\bar{\mathbf{Z}}_m^T\mathbf{A}\bar{\mathbf{Z}}_m &= \mathbf{L}_m^{-1}\mathbf{Q}_m\mathbf{M}^{-1}\mathbf{A}\mathbf{M}^{-1}\mathbf{Q}_m\mathbf{L}_m^{-T}\\ &= \mathbf{L}_m^{-1}\mathbf{T}_m\mathbf{L}_m^{-T}\\ &= \mathbf{D}_m\end{aligned}$$

Given an approximate solution $\mathbf{x}_0$ then:

(1) Set
- (a) $\mathbf{g}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$
- (b) $\mathbf{z}_0 = \mathbf{g}_0$
- (c) Solve $\mathbf{M}\mathbf{d}_0 = \mathbf{g}_0$
- (d) $\rho_0 = \mathbf{g}_0^T\mathbf{d}_0$

(2) for $k = 0,1, \cdots$ repeat;
- (a) $\mathbf{u}_k = \mathbf{A}\mathbf{z}_k$
- (b) $\gamma_k = \dfrac{\rho_k}{\mathbf{z}_k^T\mathbf{u}_k}$
- (c) $\mathbf{x}_{k+1} = \mathbf{x}_k + \gamma_k\mathbf{z}_k$
- (d) $\mathbf{g}_{k+1} = \mathbf{g}_k - \gamma_k\mathbf{u}_k$
- (e) Solve $\mathbf{M}\mathbf{d}_{k+1} = \mathbf{g}_{k+1}$.
- (f) $\rho_{k+1} = \mathbf{g}_{k+1}^T\mathbf{d}_{k+1}$
- (g) if $\rho_{k+1} \le tol\cdot\rho_0$ then terminate the loop.
- (h) $\omega_k = \dfrac{\rho_{k+1}}{\rho_k}$
- (i) $\mathbf{z}_{k+1} = \mathbf{d}_{k+1} + \omega_k\mathbf{z}_k$

Table 2. The Conjugate Gradient Algorithm

The columns of $\bar{\mathbf{Z}}_m$ are said to be conjugate and the orthogonality condition of $\bar{\mathbf{Z}}_m$ with respect to A is referred to as the conjugacy condition. In the CG method the triangular factors of $\mathbf{T}_m$ are obtained by updating those of $\mathbf{T}_{m-1}$ [15]. The result is the algorithm in Table 2. In this table, $\mathbf{z}_{j-1}$ is a scalar multiple of the j-th column of $\bar{\mathbf{Z}}_m$. It is important to note that $\mathbf{T}_m$ is often indefinite when A is a symmetric indefinite matrix. In this case the conjugate gradient algorithm is not reliable since the triangular factorization of $\mathbf{T}_m$ may be numerically unstable.

The CG algorithm generates a sequence of approximations, $\mathbf{x}_k$, to the solution $\mathbf{x}$ with a corresponding residual vector $\mathbf{g}_k$. The termination criterion can be chosen based on these quantities. In addition to storage demands for A and M the algorithm requires storage for 4 vectors.

3. Loss of Orthogonality. In finite precision, each computation introduces a small error and therefore the computed quantities will differ from their exact counterparts. Our objective here is to state the effect of roundoff error on the Lanczos process. For this purpose we denote by ε the smallest number in the computer such that $1 + \varepsilon > 1$. It is known as the unit roundoff error.

Although the tridiagonal relation, Eq. (6), is preserved to within roundoff, the $\mathbf{M}^{-1}$ orthogonality property of the Lanczos vectors completely breaks down after a certain number of steps depending on ε and the distribution of the eigenvalues of $\mathbf{M}^{-1}\mathbf{A}$ [16]. The Lanczos vectors not only lose their orthogonality, but may even become linearly dependent. This problem also effects the conjugate

gradient method in the form of loss of conjugacy.

The loss of orthogonality can be viewed as the subsequent amplification of the errors introduced after each computation. We let $\mathbf{Q}_m$ denote the computed Lanczos vectors and define the following matrix

$$\mathbf{H}_m = \mathbf{Q}_m^T \mathbf{M}^{-1} \mathbf{Q}_m \tag{13}$$

In exact arithmetic $\mathbf{H}_m$ is the identity matrix. The off-diagonals of $\mathbf{H}_m$ will depend on ε, the unit roundoff error. Simon [14] found a recurrence relation that can be used to estimate the elements of a column of $\mathbf{H}_m$ from the elements of $\mathbf{T}_m$ and the elements in the previous columns of $\mathbf{H}_m$. This recursion can be stated in vector form

$$\beta_{j+1}\mathbf{h}_{j+1} \approx \mathbf{T}_{j-1}\mathbf{h}_j - \alpha_j \mathbf{h}_j - \beta_j \mathbf{h}_{j-1} \tag{14}$$

where $\mathbf{h}_{j-1}$, $\mathbf{h}_j$ and $\mathbf{h}_{j+1}$ are vectors of length $j-1$ containing the top $j-1$ elements of the $j-1$, j, and $j+1$-th columns of $(\mathbf{H}_m - \mathbf{I}_m)$. Here, the bottom element of $\mathbf{h}_{j-1}$ is ε. The orthogonality state can be monitored by updating $\mathbf{h}_{j+1}$ in the course of the Lanczos algorithm.

A number of preventive measures can be taken to maintain a certain level of orthogonality. Lanczos was aware of the effects of roundoff on the algorithm when he presented his work. He proposed that the newly computed vector, $\mathbf{q}_{j+1}$, be explicitly orthogonalized against all the preceding vectors at the end of each step. We will refer to this technique as "full reorthogonalization" method. It enforces orthogonality to within roundoff (i.e. $|\mathbf{q}_i^T \mathbf{M}^{-1} \mathbf{q}_j| < n\varepsilon, \quad i \neq j$). In [14] Simon showed that the computed tridiagonal remains accurate to within roundoff if the more relaxed orthogonality condition

$$|\mathbf{q}_i^T \mathbf{M}^{-1} \mathbf{q}_j| < \sqrt{n\varepsilon}, \quad i \neq j \tag{15}$$

is enforced. We refer to this as the semi-orthogonality condition, and to procedures that adopt the weaker condition as selective orthogonalization methods. Simon proposed to update $\mathbf{h}_{j+1}$ using (15) and monitor the magnitude of its elements. Whenever any component of $\mathbf{h}_{j+1}$ is greater than $\sqrt{\varepsilon}$ then semi-orthogonality may be lost between $\mathbf{q}_{j+1}$ and some columns of $\mathbf{Q}_j$. At this step the appropriate Lanczos vectors are brought in from secondary store and $\mathbf{q}_{j+1}$ is orthogonalized against each of them. This operations must be carried out in two successive steps to avoid propagation of the errors.

A variant of Simon's scheme is to restore orthogonality of $\mathbf{q}_j$ and $\mathbf{q}_{j+1}$ at the same time. In this way no reorthogonalization of $\mathbf{q}_{j+2}$ will be necessary, at the end of the next step. The number of operations for this scheme is the same as that of the scheme above, but vectors are retrieved only once and therefore the I/O overhead is halved.

The disadvantage of reorthogonalization is that additional storage is required to keep the Lanczos vector. If m steps are required to reduce the residual to the desired level then storage for m vectors of length n is needed. The advantage is that reorthogonalization can significantly reduce the number of steps. This is demonstrated by the numerical examples.

4. Element by Element Preconditioning. Theoretical considerations suggest that at the end of each of the first few iterations of both Lanczos and CG methods the residual norm is reduced by a factor $\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}$ where κ is the condition number of $\mathbf{AM}^{-1}$, defined by $\kappa = \|\mathbf{AM}^{-1}\| \, \|\mathbf{MA}^{-1}\|$. Note that when $\kappa = 1$, one iteration is sufficient to solve the equation. This provides us with a guideline for choosing $\mathbf{M}$. For a well chosen $\mathbf{M}$ only a few iterations reduce the residual norm to the desired level.

The idea of using solution algorithms for discretized partial differential equations as a preconditioner is not new. As early as 1963 Wachspress proposed one such preconditioner based on the ADI method [17]. Recently, in [11] we proposed a number of preconditioners, based on the element-by-

element representation of **A** for solving $\mathbf{D}\dot{\mathbf{x}} + \mathbf{A}\mathbf{x} = \mathbf{b}$. Here **D** is a diagonal matrix. The steady state solution of this equation is the same as that of (1).

The first preconditioner uses a Choleski factorization of the element matrices shifted by a diagonal matrix. We denote the diagonally scaled **A** by

$$\bar{\mathbf{A}} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} = \sum_{e=1}^{n_e} \mathbf{N}_e \bar{\mathbf{a}}_e \mathbf{N}_e^T \tag{16}$$

where, n_e is the total number of elements. Then the proposed preconditioner can be constructed in the following manner. First, the Choleski factors $\bar{\mathbf{c}}_e\bar{\mathbf{c}}_e^T = \sigma\mathbf{I} + \bar{\mathbf{a}}_e$ is computed. The shift was applied to eliminate the singularity of $\bar{\mathbf{a}}_e$. A lower triangular matrix, $\bar{\mathbf{C}}$ is formed as the product of the $\bar{\mathbf{c}}_e$'s. $\bar{\mathbf{C}}$ is an approximation to the Choleski factorization of $\bar{\mathbf{A}}$. The resulting preconditioner is given by

$$\mathbf{M} = \mathbf{D}^{1/2}\prod_{e=1}^{n_e}\mathbf{N}_e\bar{\mathbf{c}}_e\mathbf{N}_e^T\prod_{e=n_e}^{1}\mathbf{N}_e\bar{\mathbf{c}}_e^T\mathbf{N}_e^T\mathbf{D}^{1/2} \tag{17}$$

Note that the second product is carried out in the reverse order of the first. Numerical results indicated that a shift $\sigma = 1$ results in a preconditioner that is close to the optimum.

Writting $\bar{\mathbf{a}}_e = \bar{\mathbf{u}}_e + \bar{\mathbf{u}}_e^T + \bar{\mathbf{d}}_e$, where $\bar{\mathbf{d}}_e$ and $\bar{\mathbf{u}}_e$ denote the diagonal and strict upper triangular part of $\bar{\mathbf{a}}_e$, a second preconditioner was constructed as

$$\mathbf{M} = \mathbf{D}^{1/2}\prod_{e=1}^{n_e}\mathbf{N}_e(\mathbf{I} + \bar{\mathbf{d}}_e + \bar{\mathbf{u}}_e^T)\mathbf{N}_e^T\prod_{e=n_e}^{1}\mathbf{N}_e(\mathbf{I} + \bar{\mathbf{d}}_e + \bar{\mathbf{u}}_e)\mathbf{N}_e^T\mathbf{D}^{1/2} \tag{18}$$

A comparison of these two preconditioners on small problems indicated that the Choleski form is more effective.

E×E Choleski: In [12] Winget replaced the diagonal of $\bar{\mathbf{a}}_e$ with the identity matrix to form

$$\mathbf{M} = \mathbf{D}^{1/2}\prod_{e=1}^{n_e}\mathbf{N}_e\bar{\mathbf{c}}_e\mathbf{N}_e^T\prod_{e=n_e}^{1}\mathbf{N}_e\bar{\mathbf{c}}_e^T\mathbf{N}_e^T\mathbf{D}^{1/2} \tag{19}$$

where $\bar{\mathbf{c}}_e\bar{\mathbf{c}}_e^T = \mathbf{I} + \bar{\mathbf{u}}_e + \bar{\mathbf{u}}_e^T$. This avoids the artificial shifting of $\bar{\mathbf{a}}_e$.

E×E LU Split: Similarly, by dropping $\bar{\mathbf{d}}_e$ in (18), a new but simpler preconditioner

$$\mathbf{M} = \mathbf{D}^{1/2}\prod_{e=1}^{n_e}\mathbf{N}_e(\mathbf{I} + \bar{\mathbf{u}}_e^T)\mathbf{N}_e^T\prod_{e=n_e}^{1}\mathbf{N}_e(\mathbf{I} + \bar{\mathbf{u}}_e)\mathbf{N}_e^T\mathbf{D}^{1/2} \tag{20}$$

can be constructed that eliminates the need for forming Choleski factorization of element matrices. The main advantage is the reduction in storage since $\mathbf{M}^{-1}\mathbf{v}$ can be evaluated for any **v** using the element representation of **A**. In the next section we compare the two preconditioners defined in (19) and (20) using both Lanczos and CG methods.

5. Numerical Examples. We apply the Lanczos and CG methods to two different example problems. The first example is a cavity driven flow problem (Stokes flow). The incompressibility of the fluid is represented by local volumetric constraints. These constraints are enforced in each finite element using a penalty method. The penalty parameter represents the bulk modulus of the fluid. 400 elements are used to model this problem. See figure 1(a). The condition number of **A** increases with the penalty parameter. We refer to this as material ill-conditioning.

The second example we used is a beam in pure bending. Taking advantage of symmetry, a quarter of the beam is modeled using plane stress elements, see figure 1(b). The beam was analyzed for a range of different thicknesses while keeping the length constant. This way the element aspect ratio (ratio of the largest dimension to the smallest) can be varied. Again, the condition of **A**

increases with the aspect ratio. We refer to this as geometric ill-conditioning. Three different levels of mesh refinement were used to study the effect of problem size on the algorithms ; 4×16, 8×32 and 16×64.

To illustrate the advantages of semi-orthogonality we evaluate the solution for the 4×16 beam problem using the CG method, Lanczos method with full reorthogonalization, and Lanczos with Simon's scheme for maintaining semi-orthogonality. In figure 2 we plot the residual norm against the iteration number for each of the methods. The results for the two implementations of the Lanczos method are indistinguishable. The residual norm in the CG method starts off the same as that in the Lanczos method, but it deviates quickly and takes four times as many steps to converge. The difference in the curves for the CG and Lanczos is due to loss of orthogonality in the CG method.

We use the 20×20 Stokes flow problem to make a direct comparison between three different preconditioners; diagonal scaling, the E×E Choleski defined in (19), and the E×E LU split defined in (20). We solve this problem for a range of different penalty parameters using both Lanczos and CG methods. A summary of the results is given in Table 3. Sample plots of the residual norm against the iteration number are illustrated in Figures 3 and 4. The number of iterations required to obtain the solution using E×E LU split is marginally more than that using E×E Choleski. On average the cost of reorthogonalization for E×E Choleski was slightly less. On the other hand, E×E Choleski requires additional storage to keep the preconditioning matrix. It is interesting to note that the number of reorthogonalizations increases with the penalty parameter (condition number). This indicates that Simon's scheme performs reorthogonalizations when ever it is needed. The number of iterations given in Table 3 are plotted against the penalty parameter; see Figure 5. One can observe form this plot that maintaining orthogonality can results in reductions of factors of two in the number of iterations for this example.

		Lanczos with Partial Reorthogonalization						Conjugate Gradients		
		E×E LU Split		E×E Choleski		Diagonal		E×E LU Split	E×E Choleski	Diagonal
Penalty Param.	κ	No. of Iter.	Reorth. Cost*	No. of Iter.	Reorth. Cost*	No. of Iter.	Reorth. Cost*	No. of Iter.	No. of Iter.	No. of Iter.
10^4	10^6	236	19445	228	17245	393	95424	473	424	792
10^3	10^5	193	11134	184	8900	348	60869	277	261	517
10^2	10^4	117	440	110	768	247	21393	117	110	252
10^1	10^3	40	0	39	0	98	708	40	39	98
1	10^2	26	0	25	0	60	0	26	25	60

* Unit is one dot product and one SAXPY (vector plus a scalar times a vector).

Table 3. Result of tests using 20×20 mesh. $n = 722$.

We obtain a similar set of results for the beam example; see Table 4. Both diagonal and E×E LU split are used to precondition the problem. The solution is evaluated for three different levels of discretization using Lanczos and CG methods. Sample plots for the 8×32 mesh are illustrated in Figure 6 for thickness $T = 1.0$ and in Figure 7 for $T = 0.5$. When $T = 1.0$, CG method required three times as many steps to converge as the Lanczos method. More crucial is the fact that CG failed to converge after 6000 iterations when $T \leq 0.5$, even for E×E preconditioners. On the other hand

Lanczos delivered the solution in less than 300 iterations using ExE preconditioners.

These methods were implemented on a 32 node Caltech Hypercube [18-20] parallel processors. In Figures 11 and 12, we plot the computer time and speed up factors for the first 50 iterations of the beam problem. The results of our studies indicate that both Lanczos and CG method can take advantage of the special architecture of these computers.

			Lanczos with Partial Reorthogonalization				Conjugate Gradients	
			ExE LU Split		Diagonal		ExE LU Split	Diagonal
Problem	Aspect Ratio	κ	No. of Iter.	Reorth. Cost*	No. of Iter.	Reorth. Cost*	No. of Iter.	No. of Iter.
	1	10^4	146	511	378	17352	182	506
16x64	2	2×10^5	199	1208	542	106880	344	1027
$n = 2142$	4	3×10^6	329	3261	896	296617	850	2336
	8	5×10^7	586	47667	1132+	811600+	2714	6000+
	40	3×10^{10}	886	217883	1648+	1317042+	6000+	6000+
	1	2×10^2	41	146	89	2382	41	96
4x16	2	3×10^3	56	196	111	5625	71	236
$n = 151$	4	5×10^4	87	624	141	13951	191	507
	8	7×10^5	114	2312	151	17482	605	1532
	40	5×10^8	150	6461	151	17784	2216	5614

* Unit is one dot product and one SAXPY (vector plus a scalar times a vector).

Table 4. Result of tests using the beam in pure bending. + indicates that CPU time exceeded.

References

[1] C. Lanczos, "An Iteration Method for the Solution of the Eigenvalue Problems of Linear Differential and Integral Operators," *J. Res. Nat. Bur. Standards*, **45** (1950), pp. 255-282.

[2] M. R. Hestenes and E. Stiefel, "Method of Conjugate Gradient for Solving Linear Systems," *J. Res. Nat. Bur. Standards*, **45** (1952), pp. 409-436.

[3] C. Lanczos, "Solution of Systems of Linear Equations by Minimized Iteration," *J. Res. Nat. Bur. Standards*, **49** (1952), pp. 33-53.

[4] R. K. Lively, "The Analysis of Large Structural Systems," *Computer J.*, **3** (1960), pp. 34-39.

[5] J. K. Reid, "On the Method of Conjugate Gradients for the Solution of Large Sparse Systems of Linear Equations" in *Large Sparse Sets of Linear Equations*, Academic Press, New York, 1971, pp. 231-254.

[6] R. L. Fox and E. L. Stanton, "Developments in Structured Analysis by Direct Energy Minimization," *AIAA Journal*, **6** (1968), pp. 1036-1042.

[7] I. Fried, "More on Gradient Iterative Methods in Finite-Element Analysis," *AIAA Journal*, **7** (1969), pp. 565-567.

[8] T. J. R. Hughes, I. Levit and J. Winget, "Implicit, Unconditionally Stable Algorithms for Heat Conduction Analysis," *ASCE, Journal of the Engineering Mechanics Division*, **109** (1983), pp. 576-585.

[9] M. Ortiz, P. M. Pinsky and R. L. Taylor, "Unconditionally Stable Element-by-Element algorithm for Dynamic Problems," *Computer Methods in Applied Mechanics and Engineering*, **36** (1983), pp. 223-239.

[10] T. J. R. Hughes, I. Levit and J. Winget, "An Element-by-Element Solution Algorithms for Problems of Structural and Solid Mechanics," *Computer Methods in Applied Mechanics and Engineering*, **36** (1983), pp. 241-254.

[11] B. Nour-Omid and B. N. Parlett, "Element Preconditioning Using Splitting Techniques," *SIAM J. Sci. Stat. Comput.*, **6** (1985), pp. 761-770.

[12] J. M. Winget and T. J. R. Hughes, "Solution Algorithms for Nonlinear Transient Heat Conduction Analysis Employing Element-by-Element Iterative Strategies, *Computer Methods in Applied Mechanics and Engineering*, **52** (1985), pp. 711-815.

[13] D. S. Kershaw, "The Incomplete Choleski-Conjugate Gradient Method for Solution of System of Linear Equations," *Journal of Computational Physics*, **26** (1978), pp. 43-65.

[14] H. D. Simon, "The Lanczos Algorithm with Partial Reorthogonalization," *Mathematics of Computation*, **42** (1984), pp. 115-142.

[15] G. H. Golub and C. F. Van Loan, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, 1983.

[16] B. N. Parlett, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1980.

[17] E. L. Wachspress, "Extended Application of Alternating Direction Implicit Iteration Model Problem Theory," *J. Soc. Industr. Appl. Math.*, **11** (1963), pp. 994-1016.

[18] G. C. Fox, M. A. Johnson, G. A. Lyzenga, S. W. Otto, and J. K. Salmon, *Solving Problems on Concurrent Processors, Volume I: General Techniques and Regular Problems*, Prentice Hall, Inc., to be published.

[19] G. A. Lyzenga, A. Raefsky, and B. H. Hager, "Finite Elements and the Method of Conjugate Gradient on a Concurrent Processors," *Solving Problems on Concurrent Processors, Volume II: Scientific and Engineering Applications*, edited by J. Fox and G. A. Lyzenga, Prentice Hall, Inc., to be published.

[20] B. Nour-Omid and K. C. Park, "Solving Structural Mechanics Problems on the Caltech Hypercube Machine," *Computer Methods in Applied Mechanics and Engineering*, to appear.