Spectral Analysis of Large Finite Element Problems by Optimization Methods

Recently an efficient method for the solution of the partial symmetric eigenproblem (DACG, deflated-accelerated conjugate gradient) was developed, based on the conjugate gradient (CG) minimization of successive Rayleigh quotients over deflated subspaces of decreasing size. In this article four different choices of the coefficient $\beta_k$ required at each DACG iteration for the computation of the new search direction $p_k$, are discussed. The "optimal" choice is the one that yields the same asymptotic convergence rate as the CG scheme applied to the solution of linear systems. Numerical results point out that the optimal $\beta_k$ leads to a very cost effective algorithm in terms of CPU time in all the sample problems presented. Various preconditioners are also analyzed. It is found that DACG using the optimal $\beta_k$ and $(LL^T)^{-1}$ as a preconditioner, $L$ being the incomplete Cholesky factor of $A$, proves a very promising method for the partial eigensolution. It appears to be superior to the Lanczos method in the evaluation of the 40 leftmost eigenpairs of five finite element problems, and particularly for the largest problem, with size equal to 4560, for which the speed gain turns out to fall between 2.5 and 6.0, depending on the eigenpair level. © 1994 John Wiley & Sons, Inc.

INTRODUCTION

The numerical computation of the $p$ leftmost eigenpairs of the generalized eigenvalue problem $Ax = \lambda Bx$, where $A$ and $B$ are large, sparse, symmetric positive definite matrices, is a problem of major importance in many scientific and engineering applications making use of finite difference (FD) or finite element (FE) models.

Typical applications are in vibrational analysis of mechanical structures (Bathe 1982), lightwave technology (Zoboli and Bassi, 1992), and the spectral superposition approach for the solution of large sets of differential equations (Gambolati, 1993a). There are several techniques for solving the generalized eigenproblem: subspace iteration (Bathe and Wilson, 1973; Parlett, 1980), the Lanczos method (Lanczos, 1950; Paige, 1972), and optimization methods by gradient and conjugate gradient (CG) schemes.

Recently a new optimization method was developed, the deflation-accelerated conjugate gradient (DACG), that performs a preconditioned CG minimization of the Rayleigh quotient over subspaces of decreasing size (Gambolati, Sartoretto, and Florian, 1992). A vector and parallel version of this algorithm can be found in Pini and Sartoretto (1992).

The theoretical asymptotic convergence rate of DACG was studied by Bergamaschi Gambolati, and Pini (1994) who showed it to be related to the spectral condition number of the Hessian
of the Rayleigh quotient: $x^TAx / x^TBx$, restricted over the subspace $B$-orthogonal to the eigenvectors already computed and evaluated at the desired eigenvector.

Here we first analyze the particular choice of coefficient $\beta_k$ in the recurrence equation defining the new search direction $p_k$:

$$p_k = K^{-1}g_k + \beta_k p_{k-1},$$

where $K^{-1}$ is the CG preconditioning matrix and $g_k$ is the gradient of the Rayleigh quotient at the iteration $k$. Using the following general definition for $\beta_k$

$$\beta_k = -\frac{p_{k-1}^T M K^{-1} g_k}{p_{k-1}^T M p_{k-1}}$$

(1)

where $M$ is an appropriate matrix, a sequence of mutually $M$-conjugate directions $p_k$ is constructed. It is shown that several selections for $M$ are possible. However, they are not all equivalent in terms of computational cost. In particular, the choice of $M$ that yields the same asymptotic convergence as the CG method in solving linear systems, appears to be the most convenient one. The numerical performance of DACG with $M$ defined as in Perdon and Gambolati (1986) and Gambolati and Bergamaschi (1992) is discussed using three well known preconditioners: $K^{-1} = A^{-1}$, $K^{-1} = (LL^T)^{-1}$, $L$ being the incomplete Cholesky factor of $A$, and $K^{-1} = D^{-1}$, where $D$ is the diagonal matrix whose entries are the diagonal coefficients of $A$. The numerical asymptotic convergence rate of the Cholesky-preconditioned algorithm is compared to that of DACG, which makes use of the inverse of $A$ as a preconditioner (Bergamaschi et al., 1994).

DACG Procedure

Assume that the $j$ leftmost eigenpairs of (2) are known. Then the $N-j$ eigenpair is obtained by the following procedure.

1. Start with an initial eigenvector guess $x_0$ such that $U_j^T B x_0 = 0$; that is, take $x_0$ to be $B$-orthogonal to the subset $U_j = [u_N, \ldots, u_{N-j+1}]$ of the $j$ leftmost eigenvectors previously computed. Set $k = 0$ (iteration index) and $p_{-1}$ as an arbitrary vector.
2. Let $m_k = x_k^T B x_k$ and

$$g_k = \frac{2}{m_k} [A x_k - q(x_k) B x_k]$$

(4)

be the gradient of the Rayleigh quotient (3) assessed at the current iterate $x_k$.
3. If $k = 0$ set $\beta_k = 0$, otherwise calculate $\beta_k$ by (Perdon and Gambolati, 1986; Gambolati and Bergamaschi, 1992):

$$\beta_k := \beta_k^{(1)} = -\frac{p_{k-1}^T A K^{-1} g_k}{p_{k-1}^T A p_{k-1}}$$

(5)

or by (Bergamaschi et al., 1994) (MDACG procedure):

GRADIENT METHODS FOR EVALUATION OF THE $p$ LEFTMOST EIGENPAIRS

We analyze the DACG method in the form developed in Gambolati et al. (1992), Gambolati and Bergamaschi (1992), and Bergamaschi et al. (1994), to compute the $p$ smallest eigenpairs of the eigenproblem:

$$A x = \lambda B x$$

(2)

$A$, $B$ being two sparse, symmetric, positive definite, $N \times N$ matrices. The real positive eigenvalues and corresponding eigenvectors are denoted by

$$\lambda_N \leq \lambda_{N-1} \leq \cdots \leq \lambda_1 \leq u_N, u_{N-1}, \ldots, u_1.$$

The eigenpairs are found sequentially, starting from the leftmost one $\lambda_N$, $u_N$, by means of a CG optimization of the restricted Rayleigh quotient

$$q(x) = \frac{x^T A x}{x^T B x}$$

(3)

onto subspaces of decreasing size, which are $B$-orthogonal to the previously computed eigenvectors.
\[ \beta_k := \beta_k^{(0)} = -\frac{p_k^T(A - \gamma_k B)K^{-1}g_k}{f_k^T(A - \gamma_k B)p_k}, \]
\[ \gamma_k = \lambda_{N-j+1} \]  
(6)

or by (Ruhe, 1977):
\[ \beta_k := \beta_k^{(3)} = -\frac{g_k^T K^{-1}g_k}{g_k^T K^{-1}g_k} \]  
(7)

or by (Polak, 1971; Gambolati et al., 1992):
\[ \beta_k := \beta_k^{(4)} = \frac{g_k^T K^{-1}(g_k - g_{k-1})}{g_k^T K^{-1}g_{k-1}} \]  
(8)

where \( K^{-1} = (LL^T)^{-1} \) and \( L \) is the pointwise incomplete Cholesky factor of \( A \) (Meijerink and van der Vorst, 1977; Kershaw, 1978).

Equation (7) is different from Eq. (1) but Eq. (8) can be written under the form (1), as will be shown later.

4. Calculate:
\[ \hat{p}_k = K^{-1}g_k + \beta_k p_{k-1} \]  
(9)

and evaluate \( p_k \) by B-orthogonalizing \( \hat{p}_k \) against the eigenvectors previously computed using a Gram–Schmidt process:
\[ p_k = \hat{p}_k - \sum_{j=0}^{N-j} (\hat{p}_k^T B u_{N-j}) u_{N-j}. \]  
(10)

5. Set:
\[ x_{k+1} = x_k + \alpha_k p_k \]  
(11)

where \( \alpha_k \) is chosen in order to minimize \( q(x_k + \alpha_k p_k) \) (see Papadrakakis, 1984; Perdon and Gambolati, 1986).

6. Increase the iteration counter and go back to step 2. The iteration is completed whenever \( k \) is larger than the allowed maximum number of iterations IMAX or

\[ er_{j,k+1} = \frac{|q(x_{k+1}) - q(x_k)|}{q(x_k)} < TOL1 \]  
(12)

or

\[ rr_{j,k+1} = \frac{\|Ax_{k+1} - q(x_{k+1})Bx_{k+1}\|_2}{\|Ax_{k+1}\|_2} < TOL2; \]  
(13)

if \( er_{j,k+1} < TOL1 \) or \( rr_{j,k+1} < TOL2 \), we set
\[ \lambda_{N-j} = q(x_{k+1}) \]
\[ u_{N-j} = \frac{x_{k+1}}{\sqrt{m_{k+1}}} \]

which are the \( N-j \) eigenvalue and corresponding B-normalized eigenvector, respectively, of Eq. (2).

Note that the leftmost eigenpair \( \lambda_N, u_N \) can also be computed by this general procedure by taking \( U_N \) as the null space and \( \gamma_0 = 0 \) in (6).

**Selecting the Coefficient \( \beta_k \)**

A different coefficient \( \beta_k \) may produce a different asymptotic rate of convergence. Let us now analyze the asymptotic behavior of the preconditioned CG algorithm with \( \beta_k \) as defined by Eqs. (5)–(8).

Set \( \mu_k = q(x_k), \mu = \lambda_{N-j} \). We assume that the following approximations hold for \( k \geq s \) and \( s \) "sufficiently" large.

\[ \mu_k = \mu_{k-1} = \mu, \quad m_k = m_{k-1} = m, \]
\[ g_k = \frac{2}{m} (A - \mu B)x_k \]  
(14)

and use the following equation for \( \alpha_k \) (see Ruhe, 1977):
\[ \alpha_k = -\frac{p_k^T(A - \mu B)x_k}{p_k^T(A - \mu B)p_k} = \frac{m}{2} \frac{p_k^T g_k}{p_k^T(A - \mu B)p_k}. \]  
(15)

**Remark:** This choice of \( \alpha_k \) is asymptotically equivalent to that indicated at step 5. of the previous section. Actually, the coefficient \( \alpha_k \) is chosen such that \( q(x_{k+1}) \) is minimized. This leads to the following equation (see Papadrakakis, 1984; Perdon and Gambolati, 1986; Gambolati, 1993b):
\[ a\alpha_k^2 + b\alpha_k + c = 0 \]  
(16)

where
\[ a = p_k^T Ap_k p_k^T B x_k - p_k^T A x_k p_k^T B p_k \]
\[ b = x_k^T B x_k p_k^T A p_k - x_k^T A x_k p_k^T B p_k \]
\[ c = x_k^T B x_k p_k^T A x_k - x_k^T A x_k p_k^T B x_k. \]

The first order approximation of the larger root of the equation is (Ruhe, 1977):
\[ \alpha_k = -\frac{c}{b} \frac{x_k^T B x_k p_k^T A x_k - x_k^T A x_k p_k^T B x_k}{x_k^T A x_k p_k^T B p_k - x_k^T B x_k p_k^T A p_k}. \]
Dividing numerator and denominator by $\mathbf{x}^T \mathbf{B} \mathbf{x}_k$ and recalling that
\[
\frac{\mathbf{x}^T \mathbf{A} \mathbf{x}_k}{\mathbf{x}^T \mathbf{B} \mathbf{x}_k} = \mu
\]
yields Eq. (15).

Let $k > s$, then from (14), (11), and (15) we have
\[
\mathbf{g}^T_k \mathbf{p}_{k-1} = \frac{2}{m} [(\mathbf{A} - \mu \mathbf{B}) \mathbf{x}_k] \mathbf{p}_{k-1} = \frac{2}{m} (\mathbf{x}^T_{k-1} + \alpha_{k-1} \mathbf{p}^T_{k-1})(\mathbf{A} - \mu \mathbf{B}) \mathbf{p}_{k-1}
\]
\[
= \frac{2}{m} \left[ \mathbf{x}^T_{k-1} (\mathbf{A} - \mu \mathbf{B}) \mathbf{p}_{k-1} + \alpha_{k-1} \mathbf{p}^T_{k-1} (\mathbf{A} - \mu \mathbf{B}) \mathbf{p}_{k-1} \right] = 0.
\]
(17)

Then, for every $k > s + 1$, using (9), (11), and (15) we obtain the following expression for $\beta_k^{(3)}$:
\[
\beta_k^{(3)} = \frac{\mathbf{g}^T_k K^{-1} \mathbf{g}_k}{\mathbf{g}^T_{k-1} K^{-1} \mathbf{g}_{k-1}} = \frac{\mathbf{g}^T_k K^{-1} (2/m)(\mathbf{A} - \mu \mathbf{B}) \mathbf{x}_k}{\mathbf{g}^T_{k-1} K^{-1} (\mathbf{p}_{k-1} - \beta_{k-1} \mathbf{p}_{k-2})} = \frac{\mathbf{g}^T_k K^{-1} (2/m)(\mathbf{A} - \mu \mathbf{B}) (\mathbf{x}_{k-1} + \alpha_{k-1} \mathbf{p}_{k-1})}{\mathbf{g}^T_{k-1} (\mathbf{p}_{k-1} - \beta_{k-1} \mathbf{p}_{k-2})}
\]
\[
= \frac{\mathbf{g}^T_k K^{-1} \mathbf{g}_{k-1} + (2/m) \alpha_{k-1} \mathbf{g}^T_k K^{-1} (\mathbf{A} - \mu \mathbf{B}) \mathbf{p}_{k-1}}{\mathbf{g}^T_{k-1} \mathbf{p}_{k-1} - \beta_{k-1} \mathbf{g}^T_{k-1} \mathbf{p}_{k-2}}.
\]
(18)

First observe that $\mathbf{g}^T_{k-1} \mathbf{p}_{k-2} = 0$ by (17). Then we note that
\[
\mathbf{g}^T_{k-1} \mathbf{p}_{k-1} = \mathbf{g}^T_{k-1} \mathbf{p}_{k-1} + \mathbf{g}^T_{k-1} \sum_{i=0}^{k-2} (\mathbf{p}^T_{i} \mathbf{B} \mathbf{u}_{N-i}) \mathbf{u}_{N-i}
\]
\[
= \mathbf{g}^T_{k-1} \mathbf{p}_{k-1}
\]
because every iterate $\mathbf{x}_i$ is $\mathbf{B}$-orthogonal to $\mathbf{U}_j$ and consequently, for every $i \leq j - 1$,
\[
\mathbf{g}^T_{i-1} \mathbf{u}_{N-i} = (2/m) \mathbf{x}^T_{i-1} (\mathbf{A} - \mu \mathbf{B}) \mathbf{u}_{N-i}
\]
\[
= (\lambda_{N-i} - \mu)(2/m) \mathbf{x}^T_{i-1} \mathbf{B} \mathbf{u}_{N-i} = 0.
\]

We now rewrite $\beta_k^{(3)}$ using the above results and Eq. (15):
\[
\beta_k^{(3)} = \frac{\mathbf{g}^T_k K^{-1} \mathbf{g}_k + (2/m) \alpha_{k-1} \mathbf{g}^T_k K^{-1} (\mathbf{A} - \mu \mathbf{B}) \mathbf{p}_{k-1}}{(2/m) \alpha_{k-1} \mathbf{p}^T_k (\mathbf{A} - \mu \mathbf{B}) \mathbf{p}_{k-1}}.
\]
(19)

Taking now $\beta_k^{(6)}$ as in (8), and observing that Eq. (8) differs from Eq. (7), and consequently from (19), only by the term $\mathbf{g}^T_k K^{-1} \mathbf{g}_{k-1}$ at the numerator, we can write:
\[
\beta_k^{(6)} = \frac{(2/m) \alpha_{k-1} \mathbf{g}^T_k K^{-1} (\mathbf{A} - \mu \mathbf{B}) \mathbf{p}_{k-1}}{(2/m) \alpha_{k-1} \mathbf{p}^T_k (\mathbf{A} - \mu \mathbf{B}) \mathbf{p}_{k-1}}.
\]
\[
= - \frac{\mathbf{g}^T_k K^{-1} (\mathbf{A} - \mu \mathbf{B}) \mathbf{p}_{k-1}}{\mathbf{p}^T_k (\mathbf{A} - \mu \mathbf{B}) \mathbf{p}_{k-1}}
\]
\[
= - \frac{\mathbf{g}^T_k (\mathbf{A} - \mu \mathbf{B}) \mathbf{p}_{k-1}}{\mathbf{p}^T_k (\mathbf{A} - \mu \mathbf{B}) \mathbf{p}_{k-1}}.
\]
(20)

If we look at $\beta_k^{(6)}$ given by Eq. (20) and $\beta_k^{(1)}$ we may recognize that $\beta_k^{(6)}$ is somewhat intermediate between $\beta_k^{(1)}$ and $\beta_k^{(6)}$ because $0 < \gamma_j < \mu$.

**Convergence of DACG Method**

The selection of $\beta_k = \beta_k^{(6)}$, Eq. (20), implies that the $\mathbf{p}$ directions are mutually $(\mathbf{A} - \mu \mathbf{B})$-orthogonal. Therefore solving the eigenvalue problem is in this case asymptotically equivalent to solving the linear system:
\[
(\mathbf{A} - \mu \mathbf{B}) \mathbf{x} = 0.
\]

It was proved by Axelsson (1976) that the asymptotic convergence rate of the CG method in the solution of linear systems can be approximated by:
\[
\phi = \frac{2}{\sqrt{\xi}}
\]
(21)

$\xi$ being the condition number of the preconditioned iteration matrix. Hence the DACG convergence rate $\rho_j$, defined as:
\[
\rho_j = \lim_{k \to \infty} \mu_{j,k} = - \lim_{k \to \infty} \ln \left( \frac{\lambda_{j,k}}{\lambda_{j,k+1}} \right)
\]
(22)

with $\beta_k = \beta_k^{(6)}$, is expected to be inversely proportional to $\sqrt{\xi}$, with a proportionality factor of 4 instead of 2 (because the convergence toward the eigenvalue is two times as fast as the convergence toward the eigenvector, see Ruhe (1977)), where $\xi$ is here equal to the condition number of $K^{-1}(\mathbf{A} - \mu \mathbf{B})$.

In Bergamaschi et al. (1994) a theoretical analysis of convergence with $\beta_k = \beta_k^{(6)}$ was performed with $K^{-1} = A^{-1}$, and it was shown that $\rho_j$ is inversely (linearly) proportional to the spectral condition number $\xi_j$ of the Hessian of the re-
stricted Rayleigh quotient, calculated at the current eigenvector \( \mathbf{u}_{N-j} \):

\[
\rho_j = \phi_j = 2 \ln \frac{1 + 1/\xi_j}{1 - 1/\xi_j} \approx \frac{4}{\xi_j}
\]  

where

\[
\xi_j = \frac{\lambda_{N-j-1}}{\lambda_{N-j}} - \lambda_{N-j}.
\]

Furthermore, Bergamaschi et al. (1994) give an estimate of the number \( k_j \) of iterations required to reduce the relative error by a factor \( h \):

\[
k_j = \tilde{k} + \xi_j \ln \frac{h}{4}.
\]

where \( \tilde{k} \) is the number of "initial" iterations performed before the asymptotic convergence is achieved. By distinction, and in view of Eq. (21), we might expect that the DACG asymptotic convergence rate with \( \beta_k = \beta_k^{(N)} \) is dependent on the square root of \( \xi_j \). In this case the approximate number \( k_j \) of iterations required to reduce the error by the factor \( h \) is:

\[
k_j = \tilde{k} + \sqrt{\xi_j} \ln \frac{h}{4}.
\]  

Note that if \( K^{-1} \neq A^{-1} \) no analytic expression can be given for \( \xi_j \) because the eigenvalues distribution of \( K^{-1} \) is, in general, unknown.

**NUMERICAL RESULTS**

The DACG procedure (with different choices of the coefficient \( \beta_0 \)) has been applied to five sample problems arising from the FE integration of 2-D and 3-D equations of elliptic type with size \( N = 222, 441, 812, 1952, \) and \( 4560 \) (Gambolati and Putti, 1994). The distribution of the 40 leftmost eigenvalues is shown in Fig. 1. Computations were performed on an IBM 9370 computer in double precision arithmetic.

Table 1 provides the experimental asymptotic convergence rates \( \rho_j \) of DACG with \( \beta_k = \beta_k^{(N)} \) and \( K^{-1} = (LL^T)^{-1} \), and makes a comparison between \( \rho_j \) and the theoretical rate \( \phi_j \), computed by Eq.

<table>
<thead>
<tr>
<th>( N = 222 )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( N = 441 )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( N = 812 )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( N = 1952 )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( N = 4560 )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

**FIGURE 1** Distribution of the 40 leftmost eigenvalues for the five sample problems.
Table 1. Experimental Convergence Rates and Comparison between \( \phi_j \) and \( \rho_j \) for Preconditioners \( A^{-1} \) and \( (LL^T)^{-1} \)

<table>
<thead>
<tr>
<th>( N )</th>
<th>( \phi_j )</th>
<th>( \rho_j )</th>
<th>( \phi_j )</th>
<th>( \rho_j )</th>
<th>( \phi_j )</th>
<th>( \rho_j )</th>
<th>( \phi_j )</th>
<th>( \rho_j )</th>
<th>( \phi_j )</th>
<th>( \rho_j )</th>
</tr>
</thead>
<tbody>
<tr>
<td>222</td>
<td>0.057</td>
<td>0.074</td>
<td>0.057</td>
<td>0.074</td>
<td>0.057</td>
<td>0.074</td>
<td>0.057</td>
<td>0.074</td>
<td>0.057</td>
<td>0.074</td>
</tr>
<tr>
<td>441</td>
<td>0.057</td>
<td>0.074</td>
<td>0.057</td>
<td>0.074</td>
<td>0.057</td>
<td>0.074</td>
<td>0.057</td>
<td>0.074</td>
<td>0.057</td>
<td>0.074</td>
</tr>
<tr>
<td>812</td>
<td>0.057</td>
<td>0.074</td>
<td>0.057</td>
<td>0.074</td>
<td>0.057</td>
<td>0.074</td>
<td>0.057</td>
<td>0.074</td>
<td>0.057</td>
<td>0.074</td>
</tr>
<tr>
<td>1952</td>
<td>0.057</td>
<td>0.074</td>
<td>0.057</td>
<td>0.074</td>
<td>0.057</td>
<td>0.074</td>
<td>0.057</td>
<td>0.074</td>
<td>0.057</td>
<td>0.074</td>
</tr>
<tr>
<td>4560</td>
<td>0.057</td>
<td>0.074</td>
<td>0.057</td>
<td>0.074</td>
<td>0.057</td>
<td>0.074</td>
<td>0.057</td>
<td>0.074</td>
<td>0.057</td>
<td>0.074</td>
</tr>
</tbody>
</table>

Comparison between theoretical asymptotic convergence rate \( \phi_j \), Eq. (23), and the numerical convergence rate \( \rho_j \) of DACG with \( K^{-1} = A^{-1} \), and with \( K^{-1} = (LL^T)^{-1} \), respectively, for some of the leftmost eigenpairs of the five sample problems.

Table 2. Average Time/DACG Iteration

<table>
<thead>
<tr>
<th>Problem</th>
<th>( T_{ORT} )</th>
<th>( T_{A^{-1}} )</th>
<th>( T_{(LL^T)^{-1}} )</th>
<th>( T_{D^{-1}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( N = 222 )</td>
<td>0.076</td>
<td>0.076</td>
<td>0.076</td>
<td>0.076</td>
</tr>
<tr>
<td>( N = 441 )</td>
<td>0.076</td>
<td>0.076</td>
<td>0.076</td>
<td>0.076</td>
</tr>
<tr>
<td>( N = 812 )</td>
<td>0.076</td>
<td>0.076</td>
<td>0.076</td>
<td>0.076</td>
</tr>
<tr>
<td>( N = 1952 )</td>
<td>0.076</td>
<td>0.076</td>
<td>0.076</td>
<td>0.076</td>
</tr>
<tr>
<td>( N = 4560 )</td>
<td>0.076</td>
<td>0.076</td>
<td>0.076</td>
<td>0.076</td>
</tr>
</tbody>
</table>

(-)DACG does not converge within IMAX = 500.

Comparison of the average time (s) per iteration \( T_{\text{iter}} = T_{\text{ORT}} + T_{\text{iter}} \) for DACG with \( \beta_k = \beta_1 \) using the two (previous) preconditioners and the new one \( K^{-1} = D^{-1} \) where \( D = \text{diag}(a_{11}, a_{22}, \ldots, a_{NN}) \), in the evaluation of the theoretical pre-conditioner as it is in the CG solution of linear systems (in which case the spectral condition number of the iteration matrix is equal to 1 and one iteration suffices to converge to the exact solution) and thus it may happen that the preconditioner \( (LL^T)^{-1} \) may occasionally lead to a faster asymptotic convergence than \( A^{-1} \). Table 2 provides the average time per DACG iteration with \( \beta_k = \beta_1 \) using the two (previous) preconditioners and the new one \( K^{-1} = D^{-1} \) where \( D = \text{diag}(a_{11}, a_{22}, \ldots, a_{NN}) \), in the evaluation of the
Table 3. Overall CPU Time for the three different preconditioners

<table>
<thead>
<tr>
<th>$K^{-1}$</th>
<th>$N = 222$</th>
<th></th>
<th>$N = 441$</th>
<th></th>
<th>$N = 812$</th>
<th></th>
<th>$N = 1952$</th>
<th></th>
<th>$N = 4560$</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$A^{-1}$</td>
<td>1059 1088</td>
<td>4989 3020</td>
<td>7233 1789</td>
<td>24156 1839</td>
<td>162130 3657</td>
<td>602360 1933</td>
<td>162130 3657</td>
<td>602360 1933</td>
<td>162130 3657</td>
<td>602360 1933</td>
</tr>
<tr>
<td>$(LL^T)^{-1}$</td>
<td>294 1901</td>
<td>877 2873</td>
<td>2137 2720</td>
<td>4562 3232</td>
<td>13584 2905</td>
<td>13584 2905</td>
<td>13584 2905</td>
<td>13584 2905</td>
<td>13584 2905</td>
<td>13584 2905</td>
</tr>
<tr>
<td>$D^{-1}$</td>
<td>398 3409</td>
<td>1098 4303</td>
<td>— — 4303</td>
<td>— — 4303</td>
<td>— — 4303</td>
<td>— — 4303</td>
<td>— — 4303</td>
<td>— — 4303</td>
<td>— — 4303</td>
<td>— — 4303</td>
</tr>
</tbody>
</table>

(—)DACG does not converge within IMAX = 500.

Comparison of the overall CPU time (s) for DACG with $\beta_k = \beta_k^{(1)}$ and three different preconditioners in the calculation of the 40 leftmost eigenpairs (TOL2 = 10^{-3}).

40 leftmost eigenpairs. Using $D^{-1}$ or $(LL^T)^{-1}$ yields a significant reduction of time per iteration. Note that the calculation of $K^{-1}g_k$ [Eq. 9], when $K^{-1} = A^{-1}$, is performed by iteratively solving the linear system

$$Ay = g_k$$ (27)

by the CG method preconditioned with $(LL^T)^{-1}$, and this accounts for the relatively large time per iteration required by DACG preconditioned with $A^{-1}$. The computation of the product between $A^{-1}$ and $g_k$ is by any method much more expensive than the computation of $(LL^T)^{-1}g_k$. In the $N = 441$ and $N = 4560$ problems, we observe (Table 3) a reduction of the total number of iterations with the preconditioner $(LL^T)^{-1}$ consistent with the asymptotic convergence rates of Table 1.

From a careful inspection of Table 2 we can see that the time per iteration of the incomplete preconditioner is very close to that of the diagonal one although the computation of $D^{-1}g_k$ [Eq. (9)] is from three to four times less expensive than the computation of $(LL^T)^{-1}g_k$. This can be understood by observing that most of the time $T_j$ in a single iteration at level $j$ is used for the $B$-orthogonalization:

$$T_j = T_{K^{-1}} + j \times T_{ort},$$

where $T_{ort}$ is the time needed to perform a $B$-orthogonalization against a single eigenvector, and $T_{K^{-1}}$ the time needed for computing $K^{-1}g_k$ in Eq. (9). Therefore the average time per iteration is:

$$T_k = \frac{1}{40} \sum_{j=0}^{39} T_j = T_{K^{-1}} + 19.5T_{ort} = T_{K^{-1}} + T_{ORT}$$

where $T_{ORT}$ is the average $B$-orthogonalization time in a single iteration. These average times are also given in Table 2.

Table 4 gives the overall number of iterations and CPU times of DACG implemented with $K^{-1} = (LL^T)^{-1}$ as the preconditioner and $\beta_k^{(1)}$, $i = 1, 2, 3, 4$, and the ratio of DACG CPU times when $\beta_k^{(1)}$ and $\beta_k^{(4)}$ are used. Figures 2 and 3 show the different convergence profiles for the relative residual of a few selected eigenpairs $\lambda_{x_k-j, u_k-j}$, for $j = 0, 10, 20, 30$ for the test problem with $N = 441$. Careful inspection of Fig. 2, Fig. 3 and Table 4 reveals the following.

1. The slowest algorithm is DACG with $\beta_k = \beta_k^{(1)}$ (Table 4 and Fig. 2).
2. The choice $\beta_k = \beta_k^{(2)}$ (MDACG) provides a good asymptotic convergence but it may be more time consuming than the other choices because each iteration needs an extra cost due to the computation of $A - \gamma_j B$.
3. In the $N = 1952$ problem, MDACG requires the minimal number of iterations. Furthermore, in the $N = 812$ problem, the other three algorithms perform a complete $B$-orthogonalization of the current $x_k$-vector against the previously computed eigenvectors, thus increasing the time per iteration (by distinction, MDACG requires only a selective $B$-orthogonalization). Therefore MDACG, in the $N = 812$ problem, is superior to the other DACG algorithms in terms of computing time.

3. The choice $\beta_k = \beta_k^{(3)}$ appears to be the best in most of the eigenvalue levels from the point of view of asymptotic rate of convergence (i.e., the slope of the profiles in Figs. 2 and 3). However, in a few cases (such as, for instance, $j = 10$ for the $N = 441$ problem) $\beta_k^{(3)}$ provides initial convergence that leads to a high number of iterations as is also shown by the horizontal rr profile of Fig. 3.
Table 4. Overall Iterations and CPU Times for different choices of $\beta_k$

<table>
<thead>
<tr>
<th>N = 222</th>
<th>N = 441</th>
<th>N = 812</th>
<th>N = 1952</th>
<th>N = 4560</th>
</tr>
</thead>
<tbody>
<tr>
<td># Iter.</td>
<td>Time</td>
<td># Iter.</td>
<td>Time</td>
<td># Iter.</td>
</tr>
<tr>
<td>1)</td>
<td>1901</td>
<td>2873</td>
<td>2720</td>
<td>3232</td>
</tr>
<tr>
<td>2)</td>
<td>820</td>
<td>1119</td>
<td>1058</td>
<td>1656</td>
</tr>
<tr>
<td>3)</td>
<td>1185</td>
<td>1344</td>
<td>1022</td>
<td>1903</td>
</tr>
<tr>
<td>4)</td>
<td>701</td>
<td>1039</td>
<td>1078</td>
<td>1834</td>
</tr>
<tr>
<td>1/4)</td>
<td>2.71</td>
<td>2.76</td>
<td>2.52</td>
<td>1.76</td>
</tr>
</tbody>
</table>

DACG performance vs. parameter $\beta_k$ in terms of total number of iterations and CPU time (s) in the calculation of the 40 leftmost eigenpairs ($TOL_2 = 10^{-3}$).

4. DACG with $\beta_k = \beta_k^{(4)}$ is the fastest scheme in terms of the total CPU time in the $N = 222$ and $N = 441$ problems although the asymptotic convergence rate appears to be almost the same as that of DACG with $\beta_k^{(3)}$. DACG with $\beta_k^{(6)}$ implements an automatic "restart" (see Polak, 1971) that is useful when a slow initial convergence occurs because it prevents the residual from remaining constant for a large number of initial iterations.

On balance Table 4 emphasizes the poor performance of DACG with $\beta_k = \beta_k^{(6)}$ and indicates that the remaining DACG are to some extent equivalent in terms of overall computer cost, the $\beta_k = \beta_k^{(4)}$ procedure being perhaps slightly superior.

The convergence properties of DACG with $\beta_k = \beta_k^{(4)}$ have been numerically studied to check the validity of formula (26). The tolerance for the relative error is set to a very low value ($TOL_1 = 10^{-13}$). Experience suggests that the average number $k$ of initial iterations is 25. Our "initial" error is therefore $e_{25}$, that we want to reduce by a factor $h = e_{25}/TOL_1$ to achieve the final accuracy TOL1. This needs according to Eq. (26) a number of iterations $k_j$ given by:

$$k_j = 25 + \sqrt{\xi_j \ln \frac{h}{4} = 25 + \sqrt{\frac{\lambda_{N-j-1}}{\lambda_{N-j} - \lambda_{N-j}} \ln \frac{h}{4}}}$$

Table 5 compares the theoretical number of iterations $k_j$ provided by (28) and the actual one for the $N = 441$ problem, and also gives the theoretical convergence rate

$$\phi_j = 4/\sqrt{\xi_j}$$

and the convergence rate $p_j$, numerically computed by Eq. (22). Table 5 shows that the actual and the expected number of iterations differ by at most 10, with the exception of the six leftmost eigenpairs, thus providing experimental evidence that Eq. (26) is a reliable approximation of the DACG iteration number as a function of the relative separation of the eigenvalue $\lambda_{N-j}$ currently sought and the next higher one. Also note in Table 5 that the numerical $p_j$ is a quite good approximation to the theoretical $\phi_j$.

Comparison of DACG and Lanczos Methods

In Gambolati and Putti (1994) DACG with $\beta_k = \beta_k^{(1)}$ is compared with two variants of the Lanczos
method. For a description of the pointwise Lanczos algorithm also see Cullum and Willoughby (1978), Simon (1984), and Paige (1972). Consistent with the numerical results of the previous section we compare DACG with $K^{-1} = (LL^T)^{-1}$ and $\beta_k = \beta_k^{(i)}$ with the LANCZOS2 procedure developed in Gambolati and Putti (1994). LANCZOS2 is a variant of the classical Lanczos algorithm, especially designed to solve eigenproblems with a pronounced fill in of the triangular factors of matrix $A$. LANCZOS2 performs the Lanczos recursive product

FIGURE 2 Convergence profile of relative residual of DACG with $\beta_k = \beta_k^{(1)}$ and $\beta_k = \beta_k^{(2)}$, for the evaluation of $\lambda_{N-j}, v_{N-j}, j = 0, 10, 20, 30$ for the eigenproblem with $N = 441$. The initial guess vector is $x_0 = [1, \ldots, 1]$ and $K^{-1} = (LL^T)^{-1}$. 
The iterative method used to solve system (30) is the CG scheme, accelerated by $(LL^T)^{-1}$. This technique enables in-core treatment of very large eigenproblems without any restriction on the bandwidth and nonzero pattern of the matrix.
Table 5. Theoretical No. of Iterations and Convergence Rate

<table>
<thead>
<tr>
<th>$j$</th>
<th>$k_j$</th>
<th>NIT</th>
<th>$\phi_j$</th>
<th>$\rho_j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>27</td>
<td>32</td>
<td>3.579</td>
<td>1.253</td>
</tr>
<tr>
<td>1</td>
<td>33</td>
<td>57</td>
<td>2.669</td>
<td>0.705</td>
</tr>
<tr>
<td>2</td>
<td>27</td>
<td>29</td>
<td>2.231</td>
<td>0.742</td>
</tr>
<tr>
<td>3</td>
<td>34</td>
<td>28</td>
<td>1.941</td>
<td>0.849</td>
</tr>
<tr>
<td>4</td>
<td>31</td>
<td>38</td>
<td>2.270</td>
<td>1.065</td>
</tr>
<tr>
<td>5</td>
<td>81</td>
<td>42</td>
<td>0.299</td>
<td>0.686</td>
</tr>
<tr>
<td>6</td>
<td>39</td>
<td>45</td>
<td>1.485</td>
<td>1.068</td>
</tr>
<tr>
<td>7</td>
<td>30</td>
<td>35</td>
<td>1.865</td>
<td>1.064</td>
</tr>
<tr>
<td>8</td>
<td>38</td>
<td>46</td>
<td>1.305</td>
<td>0.818</td>
</tr>
<tr>
<td>9</td>
<td>44</td>
<td>54</td>
<td>1.198</td>
<td>0.811</td>
</tr>
<tr>
<td>10</td>
<td>42</td>
<td>44</td>
<td>1.105</td>
<td>0.986</td>
</tr>
<tr>
<td>11</td>
<td>29</td>
<td>28</td>
<td>1.137</td>
<td>1.115</td>
</tr>
<tr>
<td>12</td>
<td>32</td>
<td>36</td>
<td>1.505</td>
<td>0.997</td>
</tr>
<tr>
<td>13</td>
<td>45</td>
<td>53</td>
<td>0.924</td>
<td>0.627</td>
</tr>
<tr>
<td>14</td>
<td>80</td>
<td>70</td>
<td>0.451</td>
<td>0.497</td>
</tr>
<tr>
<td>15</td>
<td>34</td>
<td>38</td>
<td>1.316</td>
<td>1.020</td>
</tr>
<tr>
<td>16</td>
<td>40</td>
<td>48</td>
<td>1.231</td>
<td>0.822</td>
</tr>
<tr>
<td>17</td>
<td>42</td>
<td>43</td>
<td>0.918</td>
<td>0.837</td>
</tr>
<tr>
<td>18</td>
<td>80</td>
<td>88</td>
<td>0.386</td>
<td>0.347</td>
</tr>
<tr>
<td>19</td>
<td>39</td>
<td>40</td>
<td>0.941</td>
<td>1.043</td>
</tr>
<tr>
<td>20</td>
<td>48</td>
<td>48</td>
<td>0.849</td>
<td>0.808</td>
</tr>
<tr>
<td>29</td>
<td>47</td>
<td>45</td>
<td>0.696</td>
<td>0.775</td>
</tr>
</tbody>
</table>

Comparison between the expected number of iterations $k_j$, Eq. (28), and the actual number of iterations NIT required by DACG with $\beta_k = \beta_k^{(4)}$ to achieve the prescribed tolerance TOL1 = $10^{-13}$, and between the theoretical $|\phi_j|$, Eq. (29) and the numerical ($\rho_j$) convergence rate for the $N = 441$ problem. $K^{-1} = (LLT)^{-1}$.

Table 6. DACG and LANCZOS2 in Overall CPU Time

<table>
<thead>
<tr>
<th>$j$</th>
<th>$N = 222$</th>
<th>$N = 441$</th>
<th>$N = 812$</th>
<th>$N = 1952$</th>
<th>$N = 4560$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>5</td>
<td>24</td>
<td>3.0</td>
<td>19</td>
<td>50</td>
<td>2.6</td>
</tr>
<tr>
<td>10</td>
<td>55</td>
<td>3.0</td>
<td>44</td>
<td>86</td>
<td>2.0</td>
</tr>
<tr>
<td>20</td>
<td>60</td>
<td>1.7</td>
<td>128</td>
<td>168</td>
<td>1.3</td>
</tr>
<tr>
<td>40</td>
<td>96</td>
<td>1.0</td>
<td>258</td>
<td>334</td>
<td>1.3</td>
</tr>
</tbody>
</table>

Comparison of CPU times (s) for (a) DACG with $\beta_k = \beta_k^{(4)}$ and (b) LANCZOS2 in the calculation of the $j$ ($j = 40$) leftmost eigenpairs (TOL2 = $6 \times 10^{-3}$). Column (c) provides the ratio of LANCZOS2 and DACG CPU times.

CONCLUSIONS

The DACG performance with three preconditioners $[(LL^T)^{-1}$, $L$ being the incomplete Cholesky factor of $A$, $A^{-1}$, and $D^{-1}$, where $D$ is a diagonal matrix whose entries are the diagonal coefficients of $A$], has been compared. The DACG convergence properties have also been analyzed, using $(LL^T)^{-1}$ as a preconditioner and four different choices of parameter $\beta_k$ in the evaluation of the 40 leftmost eigenpairs of generalized sparse eigenproblems.

The asymptotic convergence rate of the Cholesky-preconditioned DACG with $\beta_k = \beta_k^{(4)}$ has been found to be close to that of DACG preconditioned with $A^{-1}$, with the exception of the first few eigenpairs. Computational efficiency is, however, remarkably higher because the cost per iteration turns out to be smaller by a factor ranging between 5 and 10.

The asymptotic behavior of DACG is sensitive to $\beta_k$. DACG with the optimal $\beta_k = \beta_k^{(4)}$ has an asymptotic convergence rate that is practically equal to that of the CG method used to solve linear systems. The choice $\beta_k = \beta_k^{(4)}$ and $K^{-1} = (LL^T)^{-1}$ leads to the lowest CPU time in most of the eigenproblems.

The cost of a single DACG iteration with the incomplete Cholesky preconditioner is comparable to that of the diagonal one because most of the CPU time is spent to perform the $B$-orthogonalization. Iterations are, however, much less.

Finally, DACG with $K^{-1} = (LL^T)^{-1}$ and the optimal choice of $\beta_k = \beta_k^{(4)}$ has been compared with the well known Lanczos algorithm. DACG appears to be superior in the evaluation of the 40 leftmost eigenpairs of five sample problems and pencil $A$, $B$. Table 6 shows the performance of DACG and LANCZOS2 in terms of overall CPU time for the first 5, 10, 20, and 40 eigenpairs to meet the exit test (TOL2 = $6 \times 10^{-3}$) for the relative residual. Inspection of Table 6 reveals that DACG is faster than LANCZOS2, and particularly so when only a few eigenpairs are sought or the problem is large ($N > 1000$). It may also be noted that DACG for large eigenproblems proves less demanding than LANCZOS2 in terms of computer storage.
results in a saving of CPU time ranging from 4% \((N = 222\) problem\) to 60% \((\text{problem with } N = 4560\) equations\). These values, however, are found to grow significantly if a smaller number of eigenpairs are required. On balance DACG is recommended for large problems \((N > 1000)\) and for the computation of few eigenpairs \((\leq 5, \text{on the condition that a not too strict tolerance is prescribed})\). Alternatively, the Lanczos method should be used for eigenproblems of small size and whenever a high accuracy is required \((\text{TOL} \leq 10^{-5})\).

REFERENCES


Submit your manuscripts at
http://www.hindawi.com