SPECTRAL ANALYSIS OF LARGE FINITE ELEMENT PROBLEMS BY OPTIMIZATION METHODS

LUCA BERGAMASCHI, GIUSEPPE GAMBOLATI AND GIORGIO PINI
DIPARTIMENTO DI METODI E MODELLI MATEMATICI
PER LE SCIENZE APPLICATE
UNIVERSITY OF PADUA, ITALY

Key words. eigenpairs, conjugate gradient, sparse matrices, Rayleigh quotient, rate of convergence, Hessian condition number

AMS(MOS) subject classifications. 65F15 - 65F50

Abstract. Recently an efficient method for the solution of the partial symmetric eigenproblem (DACG, Deflated-Accelerated Conjugate Gradient) has been developed, based on the conjugate gradient (CG) minimization of successive Rayleigh quotients over deflated subspaces of decreasing size. In the present paper four different choices of the coefficient β_k required at each DACG iteration for the computation of the new search direction $\mathbf{p}_k$, 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 β_k leads to a very cost effective algorithm in terms of CPU time in all the sample problems presented in the paper. Various preconditioners are also analyzed. It is found that DACG using the optimal β_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 (FE) 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.

Introduction. The numerical computation of the p leftmost eigenpairs of the generalized eigenvalue problem $A\mathbf{x} = \lambda B\mathbf{x}$, 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 [2], lightwave technology [22], and the spectral superposition approach for the solution of large sets of differential equations [6]. There are several techniques for solving the generalized eigenproblem: subspace iteration ([3], [16]), the Lanczos method ([12] and [14]), and optimization methods by gradient and conjugate gradient (CG) schemes.

A new optimization method, Deflation-Accelerated Conjugate Gradient (DACG), which performs a preconditioned CG minimization of the Rayleigh quotient over subspaces of decreasing size, has recently been developed [10]. A vector and parallel version of this algorithm can be found in [18].

The theoretical asymptotic convergence rate of DACG has been studied by Bergamaschi et al. [4] who showed it to be related to the spectral condition number of the Hessian of the Rayleigh quotient: $\mathbf{x}^T A\mathbf{x}/\mathbf{x}^T B\mathbf{x}$, restricted over the subspace B-orthogonal to the eigenvectors already computed and evaluated at the desired eigenvector.

In the present paper we first analyze the particular choice of coefficient β_k in the recurrence equation defining the new search direction $\mathbf{p}_k$:

$$\mathbf{p}_k = K^{-1}\mathbf{g}_k + \beta_k \mathbf{p}_{k-1},$$

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

$$\beta_k = -\frac{\mathbf{p}_{k-1}^T M K^{-1} \mathbf{g}_k}{\mathbf{p}_{k-1}^T M \mathbf{p}_{k-1}} \tag{1}$$

where M is an appropriate matrix, a sequence of mutually M-conjugate directions $\mathbf{p}_k$ 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 which 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 [17] and [8] 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 [4].

DACG, preconditioned with $(LL^T)^{-1}$, and using four different choices for β_k is applied to calculate the 40 leftmost eigenpairs of five finite element eigenproblems of large size [9]. The DACG convergence profiles and total CPU times are analyzed and discussed. Finally the behaviour of the Cholesky-preconditioned algorithm, with the best β_k value, is compared to the pointwise Lanczos method [9] for evaluating 5, 10, 20 and 40 eigenpairs of the five test problems.

Gradient methods for the evaluation of the p leftmost eigenpairs. We will analyze the DACG method in the form developed in [10], [8], and [4], to compute the p smallest eigenpairs of the eigenproblem:

$$A\mathbf{x} = \lambda B\mathbf{x} \tag{2}$$

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

$$\lambda_N \leq \lambda_{N-1} \leq \ldots \leq \lambda_1$$

$$\mathbf{u}_N, \mathbf{u}_{N-1}, \ldots, \mathbf{u}_1$$

The eigenpairs are found sequentially, starting from the leftmost one $\lambda_N, \mathbf{u}_N$, by means of a CG optimization of the restricted Rayleigh quotient

$$q(\mathbf{x}) = \frac{\mathbf{x}^T A \mathbf{x}}{\mathbf{x}^T B \mathbf{x}} \tag{3}$$

onto subspaces of decreasing size, which are B-orthogonal to the previously computed eigenvectors.

The 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 $\mathbf{x}_0$ such that $U_j^T B\mathbf{x}_0 = 0$, i.e. take $\mathbf{x}_0$ to be B-orthogonal to the subset $U_j = [\mathbf{u}_N, \ldots, \mathbf{u}_{N-j+1}]$ of the j leftmost eigenvectors previously computed. Set $k = 0$ (iteration index) and $\mathbf{p}_{-1}$ as an arbitrary vector;
2. let $m_k = \mathbf{x}_k^T B\mathbf{x}_k$ and

$$\mathbf{g}_k = \frac{2}{m_k}[A\mathbf{x}_k - q(\mathbf{x}_k)B\mathbf{x}_k] \tag{4}$$

   be the gradient of the Rayleigh quotient (3) assessed at the current iterate $\mathbf{x}_k$.
3. If $k = 0$ set $\beta_k = 0$, otherwise calculate β_k by [17], [8]:

$$\beta_k := \beta_k^{(1)} = -\frac{\mathbf{p}_{k-1}^T A K^{-1}\mathbf{g}_k}{\mathbf{p}_{k-1}^T A\mathbf{p}_{k-1}} \tag{5}$$

   or by [4] (MDACG procedure):

$$\beta_k := \beta_k^{(2)} = -\frac{\mathbf{p}_{k-1}^T (A-\gamma_j B) K^{-1}\mathbf{g}_k}{\mathbf{p}_{k-1}^T (A-\gamma_j B)\mathbf{p}_{k-1}}, \qquad \gamma_j = \lambda_{N-j+1} \tag{6}$$

   or by [20]:

$$\beta_k := \beta_k^{(3)} = \frac{\mathbf{g}_k^T K^{-1}\mathbf{g}_k}{\mathbf{g}_{k-1}^T K^{-1}\mathbf{g}_{k-1}} \tag{7}$$

   or by [19], [10]:

$$\beta_k := \beta_k^{(4)} = \frac{\mathbf{g}_k^T K^{-1}(\mathbf{g}_k - \mathbf{g}_{k-1})}{\mathbf{g}_{k-1}^T K^{-1}\mathbf{g}_{k-1}} \tag{8}$$

   where $K^{-1} = (LL^T)^{-1}$ and L is the pointwise incomplete Cholesky factor of A ([13], [11]). Equation (7) is different from eq. (1) while eq. (8) can be written under the form (1), as will be shown later.
4. Calculate:

$$\tilde{\mathbf{p}}_k = K^{-1}\mathbf{g}_k + \beta_k\mathbf{p}_{k-1} \tag{9}$$

   and evaluate $\mathbf{p}_k$ by B-orthogonalizing $\tilde{\mathbf{p}}_k$ against the eigenvectors previously computed using a Gram-Schmidt process:

$$\mathbf{p}_k = \tilde{\mathbf{p}}_k - \sum_{i=0}^{j-1}(\tilde{\mathbf{p}}_k^T B\mathbf{u}_{N-i})\mathbf{u}_{N-i} \tag{10}$$

5. Set:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\mathbf{p}_k \tag{11}$$

   where α_k is chosen in order to minimize $q(\mathbf{x}_k + \alpha_k\mathbf{p}_k)$ (see [15], [17]).

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

$$er_{j,k+1} = \frac{|q(\mathbf{x}_{k+1}) - q(\mathbf{x}_k)|}{q(\mathbf{x}_k)} < TOL1 \tag{12}$$

or

$$rr_{j,k+1} = \frac{\|A\mathbf{x}_{k+1} - q(\mathbf{x}_{k+1})B\mathbf{x}_{k+1}\|_2}{\|A\mathbf{x}_{k+1}\|_2} < TOL2; \tag{13}$$

if $er_{j,k+1} < TOL1$ or $rr_{j,k+1} < TOL2$, we set

$$\lambda_{N-j} = q(\mathbf{x}_{k+1})$$

$$\mathbf{u}_{N-j} = \mathbf{x}_{k+1}/\sqrt{m_{k+1}}$$

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

Note that the leftmost eigenpair $\lambda_N, \mathbf{u}_N$ can also be computed by this general procedure by taking U_0 as the null space and $\gamma_0 = 0$ in (6).

Selecting the coefficient β_k. A different coefficient β_k may produce a different asymptotic rate of convergence. Let us now analyze the asymptotic behaviour of the preconditioned CG algorithm with β_k as defined by eqs. (5), (6), (7) and (8).

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

$$\mu_k = \mu_{k-1} = \mu, \qquad m_k = m_{k-1} = m, \qquad \mathbf{g}_k = \frac{2}{m}(A - \mu B)\mathbf{x}_k \tag{14}$$

and use the following equation for α_k (see [20]):

$$\alpha_k = -\frac{\mathbf{p}_k^T(A - \mu B)\mathbf{x}_k}{\mathbf{p}_k^T(A - \mu B)\mathbf{p}_k} = -\frac{m}{2}\frac{\mathbf{p}_k^T\mathbf{g}_k}{\mathbf{p}_k^T(A - \mu B)\mathbf{p}_k} \tag{15}$$

Remark: this choice of α_k is asymptotically equivalent to that indicated at step 5. of the previous section. Actually, the coefficient α_k is chosen such that $q(\mathbf{x}_{k+1})$ is minimized. This leads to the following equation (see [15], [17], and [7]):

$$a\alpha_k^2 + b\alpha_k + c = 0 \tag{16}$$

where

$$a = \mathbf{p}_k^T A\mathbf{p}_k \; \mathbf{p}_k^T B\mathbf{x}_k - \mathbf{p}_k^T A\mathbf{x}_k \; \mathbf{p}_k^T B\mathbf{p}_k$$

$$b = \mathbf{x}_k^T B\mathbf{x}_k \; \mathbf{p}_k^T A\mathbf{p}_k - \mathbf{x}_k^T A\mathbf{x}_k \; \mathbf{p}_k^T B\mathbf{p}_k$$

$$c = \mathbf{x}_k^T B\mathbf{x}_k \ \mathbf{p}_k^T A\mathbf{x}_k - \mathbf{x}_k^T A\mathbf{x}_k \ \mathbf{p}_k^T B\mathbf{x}_k$$

The first order approximation of the larger root of the equation is [20]:

$$\alpha_k = -\frac{c}{b} = \frac{\mathbf{x}_k^T B\mathbf{x}_k \ \mathbf{p}_k^T A\mathbf{x}_k - \mathbf{x}_k^T A\mathbf{x}_k \ \mathbf{p}_k^T B\mathbf{x}_k}{\mathbf{x}_k^T A\mathbf{x}_k \ \mathbf{p}_k^T B\mathbf{p}_k - \mathbf{x}_k^T B\mathbf{x}_k \ \mathbf{p}_k^T A\mathbf{p}_k}$$

Dividing numerator and denominator by $\mathbf{x}_k^T B\mathbf{x}_k$ and recalling that

$$\frac{\mathbf{x}_k^T A\mathbf{x}_k}{\mathbf{x}_k^T B\mathbf{x}_k} = \mu$$

yield eq. (15).

Let $k > s$, then from (14), (11) and (15) we have

$$\begin{aligned} \mathbf{g}_k^T \mathbf{p}_{k-1} &= \frac{2}{m}[(A-\mu B)\mathbf{x}_k]^T \mathbf{p}_{k-1} = \frac{2}{m}(\mathbf{x}_{k-1}^T + \alpha_{k-1}\mathbf{p}_{k-1}^T)(A-\mu B)\mathbf{p}_{k-1} = \\ &= \frac{2}{m}[\mathbf{x}_{k-1}^T(A-\mu B)\mathbf{p}_{k-1} + \alpha_{k-1}\mathbf{p}_{k-1}^T(A-\mu B)\mathbf{p}_{k-1}] = 0 \end{aligned} \tag{17}$$

Then, for every $k > s+1$, using (9), (11), and (15) we obtain the following expression for $\beta_k^{(3)}$:

$$\begin{aligned} \beta_k^{(3)} &= \frac{\mathbf{g}_k^T K^{-1}\mathbf{g}_k}{\mathbf{g}_{k-1}^T K^{-1}\mathbf{g}_{k-1}} = \frac{\mathbf{g}_k^T K^{-1}(2/m)(A-\mu B)\mathbf{x}_k}{\mathbf{g}_{k-1}^T K^{-1}(K\tilde{\mathbf{p}}_{k-1} - \beta_{k-1}K\mathbf{p}_{k-2})} \\ &= \frac{\mathbf{g}_k^T K^{-1}(2/m)(A-\mu B)(\mathbf{x}_{k-1} + \alpha_{k-1}\mathbf{p}_{k-1})}{\mathbf{g}_{k-1}^T(\tilde{\mathbf{p}}_{k-1} - \beta_{k-1}\mathbf{p}_{k-2})} = \\ &= \frac{\mathbf{g}_k^T K^{-1}\mathbf{g}_{k-1} + (2/m)\alpha_{k-1}\mathbf{g}_k^T K^{-1}(A-\mu B)\mathbf{p}_{k-1}}{\mathbf{g}_{k-1}^T\tilde{\mathbf{p}}_{k-1} - \beta_{k-1}\mathbf{g}_{k-1}^T\mathbf{p}_{k-2}} \end{aligned} \tag{18}$$

First observe that $\mathbf{g}_{k-1}^T\mathbf{p}_{k-2} = 0$ by (17). Then we note that

$$\mathbf{g}_{k-1}^T\tilde{\mathbf{p}}_{k-1} = \mathbf{g}_{k-1}^T\mathbf{p}_{k-1} + \mathbf{g}_{k-1}^T\sum_{i=0}^{j-1}(\tilde{\mathbf{p}}_k^T B\mathbf{u}_{N-i})\mathbf{u}_{N-i} = \mathbf{g}_{k-1}^T\mathbf{p}_{k-1}$$

since every iterate $\mathbf{x}_k$ is B-orthogonal to U_j and consequently, for every $i \le j-1$,

$$\mathbf{g}_{k-1}^T\mathbf{u}_{N-i} = (2/m)\ \mathbf{x}_{k-1}^T(A-\mu B)\mathbf{u}_{N-i} = (\lambda_{N-i}-\mu)(2/m)\ \mathbf{x}_{k-1}^T B\mathbf{u}_{N-i} = 0.$$

We now rewrite $\beta_k^{(3)}$ using the above results and eq. (15):

$$\beta_k^{(3)} = \frac{\mathbf{g}_k^T K^{-1}\mathbf{g}_{k-1} + (2/m)\alpha_{k-1}\mathbf{g}_k^T K^{-1}(A-\mu B)\mathbf{p}_{k-1}}{-(2/m)\alpha_{k-1}\mathbf{p}_{k-1}^T(A-\mu B)\mathbf{p}_{k-1}}. \tag{19}$$

Taking now $\beta_k^{(4)}$ as in (8), and observing that eq. (8) differs from eq. (7), and consequently from (19), only by the term $\mathbf{g}_k^T K^{-1}\mathbf{g}_{k-1}$ at the numerator, we can write:

$$\begin{aligned}\beta_k^{(4)} &= \frac{(2/m)\alpha_{k-1}\mathbf{g}_k^T K^{-1}(A-\mu B)\mathbf{p}_{k-1}}{-(2/m)\alpha_{k-1}\mathbf{p}_{k-1}^T(A-\mu B)\mathbf{p}_{k-1}} = \\ &= -\frac{\mathbf{g}_k^T K^{-1}(A-\mu B)\mathbf{p}_{k-1}}{\mathbf{p}_{k-1}^T(A-\mu B)\mathbf{p}_{k-1}} = -\frac{\mathbf{p}_{k-1}^T(A-\mu B)K^{-1}\mathbf{g}_k}{\mathbf{p}_{k-1}^T(A-\mu B)\mathbf{p}_{k-1}}\end{aligned} \tag{20}$$

If we look at $\beta_k^{(4)}$ given by eq. (20) and $\beta_k^{(1)}$ we may recognize that $\beta_k^{(2)}$ is somewhat intermediate between $\beta_k^{(1)}$ and $\beta_k^{(4)}$ since $0 < \gamma_j < \mu$.

Convergence of the DACG method. The selection of $\beta_k = \beta_k^{(4)}$, eq. (20), implies that the $\mathbf{p}$ directions are mutually $(A-\mu B)$-orthogonal. Therefore solving the eigenvalue problem is in this case asymptotically equivalent to solving the linear system:

$$(A-\mu B)\mathbf{x} = 0$$

It has been proved in [1] that the asymptotic convergence rate of the CG method in the solution of linear systems can be approximated by:

$$\phi = \frac{2}{\sqrt{\xi}} \tag{21}$$

ξ being the condition number of the preconditioned iteration matrix. Hence the DACG convergence rate ρ_j, defined as:

$$\rho_j = \lim_{k\to\infty}\rho_{j,k} = -\lim_{k\to\infty}\ln\left(\frac{er_{j,k+1}}{er_{j,k}}\right) \tag{22}$$

with $\beta_k = \beta_k^{(4)}$, is expected to be inversely proportional to $\sqrt{\xi}$, with a proportionality factor of 4 instead of 2 (since the convergence toward the eigenvalue is two times as fast as the convergence towards the eigenvector, see [20]), where ξ is here equal to the condition number of $K^{-1}(A-\mu B)$.
In [4] a theoretical analysis of convergence with $\beta_k = \beta_k^{(1)}$, is performed with $K^{-1} = A^{-1}$, and it is shown that ρ_j is inversely (linearly) proportional to the spectral condition number ξ_j of the Hessian of the restricted Rayleigh quotient, calculated at the current eigenvector $\mathbf{u}_{N-j}$:

$$\rho_j \approx \phi_j = 2\ln\frac{1+1/\xi_j}{1-1/\xi_j} \approx \frac{4}{\xi_j} \tag{23}$$

where

$$\xi_j = \frac{\lambda_{N-j-1}}{\lambda_{N-j-1}-\lambda_{N-j}}. \tag{24}$$

Furthermore, [4] gives an estimate of the number k_j of iterations required to reduce the relative error by a factor h:

$$k_j \approx \tilde{k} + \xi_j \frac{\ln h}{4} \tag{25}$$

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 $\beta_k = \beta_k^{(4)}$ is dependent on the square root of ξ_j. In this case the approximate number k_j of iterations required to reduce the error by the factor h is:

$$k_j \approx \tilde{k} + \sqrt{\xi_j}\ \frac{\ln h}{4} \tag{26}$$

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

Numerical results. The DACG procedure (with different choices of the coefficent β_k) 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 [9]. The distribution of the 40 leftmost eigenvalues is shown in Fig. 1. Computations have been performed on an IBM 9370 computer in double precision arithmetic.

Table 1 provides the experimental asymptotic convergence rates ρ_j of DACG with $\beta_k = \beta_k^{(1)}$ and $K^{-1} = (LL^T)^{-1}$, and makes a comparison between ρ_j and the theoretical rate ϕ_j, computed by eq. (23) for the ideal preconditioner A^{-1}. The results of Table 1 are quite interesting and show that the DACG asymptotic behaviour with the preconditioners A^{-1} and $(LL^T)^{-1}$ are very close, except for the first few eigenpairs. Note that $K^{-1} = (LL^T)^{-1}$ yields a higher convergence rate than $K^{-1} = A^{-1}$ for a significant number of eigenpairs, particularly for the $N = 441$ and the $N = 4560$ problems.

As pointed out in [4], A^{-1} is not the absolutely 'best' theoretical preconditioner as 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 $\beta_k = \beta_k^{(1)}$ using the two (previous) preconditioners and the new one $K^{-1} = D^{-1}$ where $D = diag\{a_{11}, a_{22}, \ldots, a_{NN}\}$, in the evaluation of the 40 leftmost eigenpairs. Using D^{-1} or $(LL^T)^{-1}$, yields a significant reduction of time per iteration. Note that the calculation of $K^{-1}\mathbf{g}_k$ (eq. 9), when $K^{-1} = A^{-1}$, is performed by iteratively solving the linear system

$$A\mathbf{y} = \mathbf{g}_k \tag{27}$$

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 $\mathbf{g}_k$ is by any method much more expensive than the

computation of $(LL^T)^{-1}\mathbf{g}_k$. 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 $D^{-1}\mathbf{g}_k$ (eq. (9)) is from 3 to 4 times less expensive than the computation of $(LL^T)^{-1}\mathbf{g}_k$. 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 $K^{-1}\mathbf{g}_k$ in eq. (9). Therefore the average time per iteration is:

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

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 $\beta_k^{(i)}, i = 1,2,3,4$, 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 $\lambda_{N-j}, \mathbf{u}_{N-j}$, for $j = 0, 10, 20, 30$ for the test problem with $N = 441$. Careful inspection of Fig. 2, Fig. 3 and Table 4 reveals that:

1. The slowest algorithm is DACG with $\beta_k = \beta_k^{(1)}$ (Table 4 and Fig. 3).
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 since each iteration needs an extra cost due to the computation of $A - \gamma_j B$. 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 $\mathbf{x}_k$-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 which leads to a high number of iterations as is also shown by the horizontal rr profile of Fig. 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 $\beta_k^{(4)}$ implements an authomatic 'restart' (see [19]) which is useful when a slow initial convergence occurs since 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 $\beta_k = \beta_k^{(1)}$ 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 (TOL1= 10^{-13}). Experience suggests that the average number $\tilde{k}$ of initial iterations is 25. Our 'initial' error is therefore er_{25}, which we want to reduce by a factor $h = er_{25}/\mathrm{TOL1}$ to achieve the final accuracy TOL1. This needs according to eq. (26) a number of iterations k_j given by:

$$k_j = 25 + \sqrt{\xi_j}\,\frac{\ln h}{4} = 25 + \sqrt{\frac{\lambda_{N-j-1}}{\lambda_{N-j-1} - \lambda_{N-j}}}\,\frac{\ln h}{4} \tag{28}$$

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

$$\phi_j = 4/\sqrt{\xi_j} \tag{29}$$

and the convergence rate ρ_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 6 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 λ_{N-j} currently sought and the next higher one. Also note in Table 5 that the numerical ρ_j is a quite good approximation to the theoretical ϕ_j.

Comparison of DACG and Lanczos methods. In [9] 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 [5], [21] and [14]. Consistent with the numerical results of the previous section we will compare DACG with $K^{-1} = (LL^T)^{-1}$ and $\beta_k = \beta_k^{(4)}$ with the LANCZOS2 procedure developed in [9]. 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

$$\mathbf{y}_j = A^{-1}B\mathbf{q}_j,$$

by iteratively solving the linear system:

$$A\mathbf{y}_j = B\mathbf{q}_j. \tag{30}$$

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 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$= 6 * 10^{-3}$) 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.

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 β_k in the evaluation of the 40 leftmost eigenpairs of generalized sparse eigenproblems.

The asymptotic convergence rate of the Cholesky-preconditioned DACG with $\beta_k = \beta_k^{(1)}$ 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 since the cost per iteration turns out to be smaller by a factor ranging between 5 and 10.

The asymptotic behaviour of DACG is sensitive to β_k. DACG with the optimal $\beta_k = \beta_k^{(4)}$ has an asymptotic convergence rate which 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, since 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 results in a saving of CPU time ranging from 4% ($N = 222$ problem) to 60 % (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 (≤ 5, on 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 (TOL2 $\leq 10^{-5}$).

Acknowledgements. This work has been supported in part by the Italian GNIM-CNR and Fondi MURST.

REFERENCES

[1] O. Axelsson, *A class of iterative methods for finite element equations*, Comp. Methods App. Mech. Eng., **9** (1976), pp. 123–137.

[2] K. J. Bathe, *Finite Element Procedures in Engineering Analysis*, Prentice-Hall, Englewood Cliffs, 1982.

[3] K. J. Bathe and E. Wilson, *Solution methods for eigenvalue problems in structural dynamics*, Int. J. Numer. Methods Eng., **6** (1973), pp. 213–226.

[4] L. Bergamaschi, G. Gambolati, and G. Pini, *Asymptotic convergence of conjugate gradient methods for the partial symmetric eigenproblem*, Math. Comp. (1994). submitted.

[5] J. Cullum and R. A. Willoughby, *Lanczos and the computation in specified intervals of the spectrum of large, sparse, real symmetric matrices*, in Symposium on Sparse Matrix Computation, Knoxville, Tenn., 1978.

[6] G. Gambolati, *On time integration of groundwater flow equations by spectral methods*, Water Resour. Res., **29** (1993), pp. 1257–1267.

[7] ——, *Solution of large scale eigenvalue problems*, in Solving Large Scale Problems in Mechanics: The Development and Application of Computational Solution Methods, M. Papadrakakis, ed., New York, 1993, John Wiley, pp. 125–156.

[8] G. Gambolati and L. Bergamaschi, *Partial eigensolution for transient groundwater flow equations*, in ICCMWR IX. Vol. 1: Numerical Methods in Water Resources, T. R. Russell et al. eds., Southampton, 1992, Computational Mechanics and Elsevier Applied Sciences, pp. 175–192.

[9] G. Gambolati and M. Putti, *A comparison of Lanczos and optimization methods in the partial solution of sparse symmetric eigenproblems*, Int. J. Numer. Methods Eng., **37** (1994), pp. 605–621.

[10] G. Gambolati, F. Sartoretto, and P. Florian, *An orthogonal accelerated deflation technique for large symmetric eigenproblems*, Comp. Methods App. Mech. Eng., **94** (1992), pp. 13–23.

[11] D. S. Kershaw, *The incomplete Cholesky-conjugate gradient method for the iterative solution of systems of linear equations*, J. Comp. Phys., **26** (1978), pp. 43–65.

[12] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, J. Res. Nat. Bur. Standard, **45** (1950), pp. 255–282.

[13] 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.

[14] C. C. Paige, *Computational variants of the Lanczos method for the eigenproblem*, J. Inst. Math. Applics., **10** (1972), pp. 373–381.

[15] M. Papadrakakis, *Solution of the partial eigenproblem by iterative methods*, Int. J. Numer. Methods Eng., **20** (1984), pp. 2283–2301.

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

[17] A. M. Perdon and G. Gambolati, *Extreme eigenvalues of large sparse matrices by Rayleigh quotient and modified conjugate gradients*, Comp. Methods App. Mech. Eng., **56** (1986), pp. 251–264.

[18] G. Pini and F. Sartoretto, *Vector and parallel codes for large sparse eigenproblems*, Supercomputer, **50** (1992), pp. 29–39.

[19] E. Polak, *Computational Methods in Optimization: A Unified Approach*, Academic Press, New York, 1971.

[20] A. Ruhe, *Computation of eigenvalues and eigenvectors*, in Sparse Matrix Techniques, V. A. Barker, ed., Berlin, 1977, Springer-Verlag, pp. 130–184.

[21] H. D. Simon, *The Lanczos algorithm with partial reorthogonalization*, Math. Comp., **42** (1984), pp. 115–142.

[22] M. Zoboli and P. Bassi, *The finite element method for anisotropic optical waveguides*, in Anisotropic and Nonlinear Optical Waveguides, C. Someda and G. Stegeman, eds., Amsterdam, 1992, Elsevier, pp. 77–116.

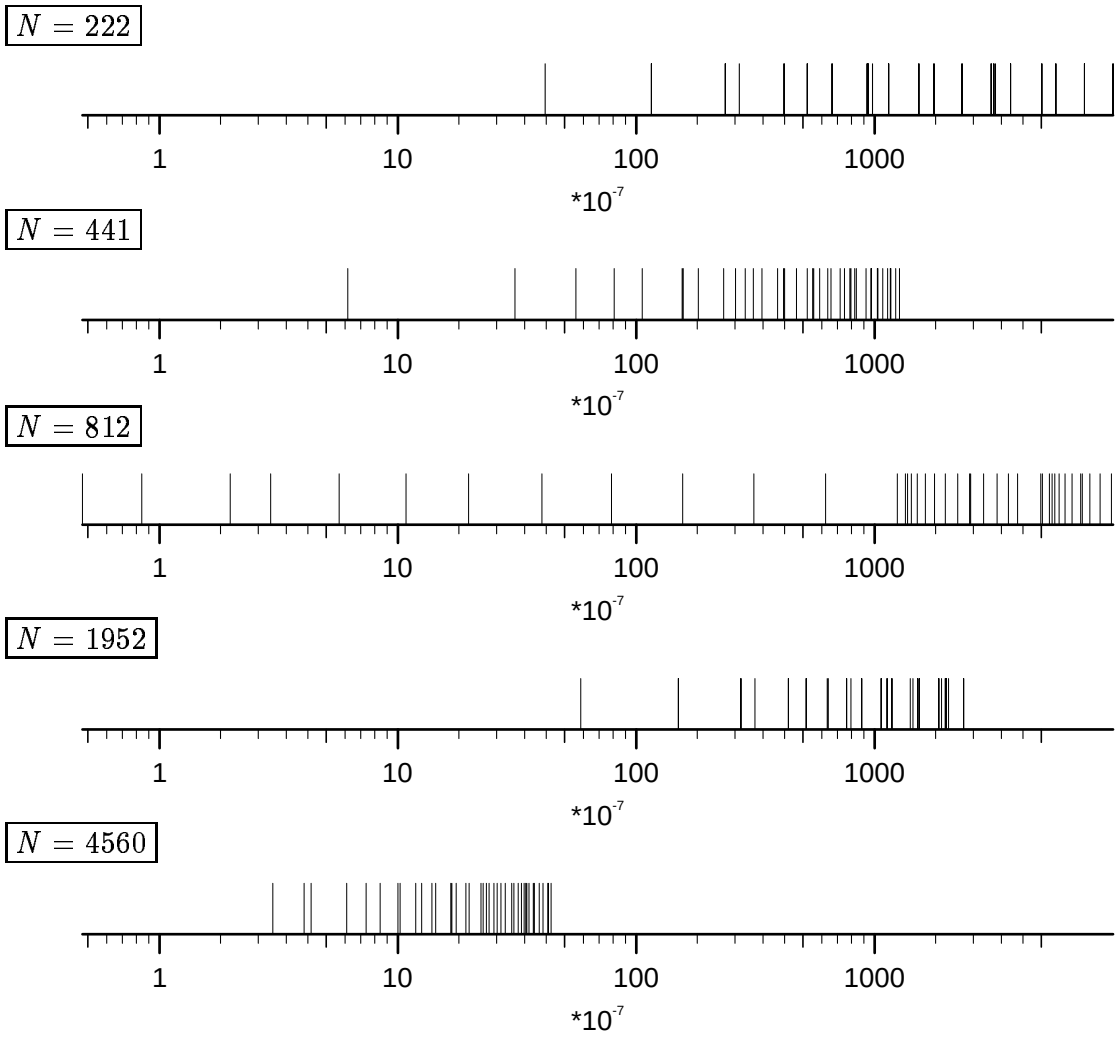

FIG. 1. *Distribution of the 40 leftmost eigenvalues for the five sample problems.*

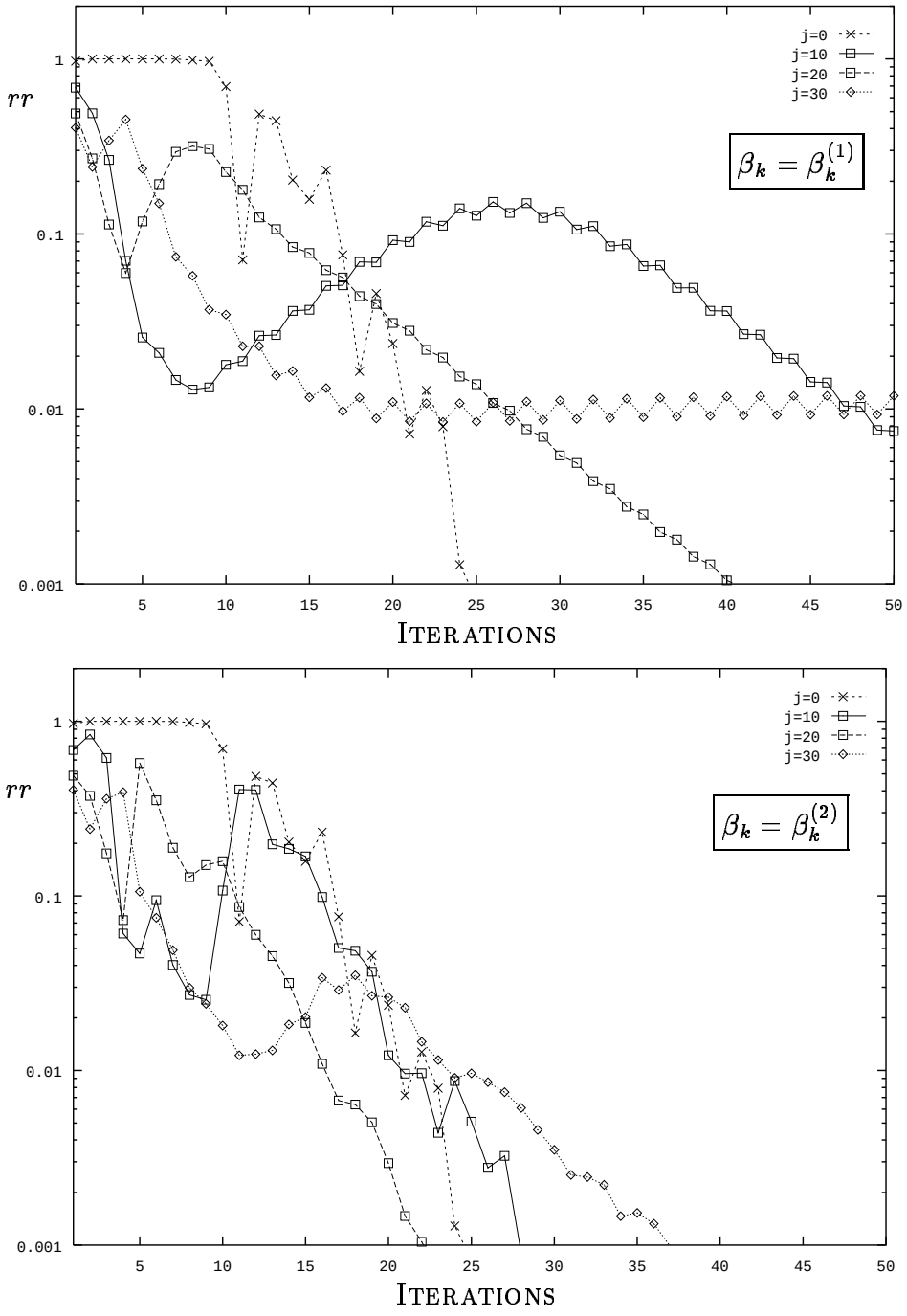

FIG. 2. *Convergence profile of relative residual of DACG with* $\beta_k = \beta^{(1)}$ *and* $\beta_k = \beta^{(2)}$, *for the evaluation of* λ_{N-j}, $\mathbf{v}_{N-j}$, *j=0, 10, 20, 30 for the eigenproblem with* $N = 441$. *The initial guess vector is* $\mathbf{x}_0 = [1, \ldots, 1]$ *and* $K^{-1} = (LL^T)^{-1}$.

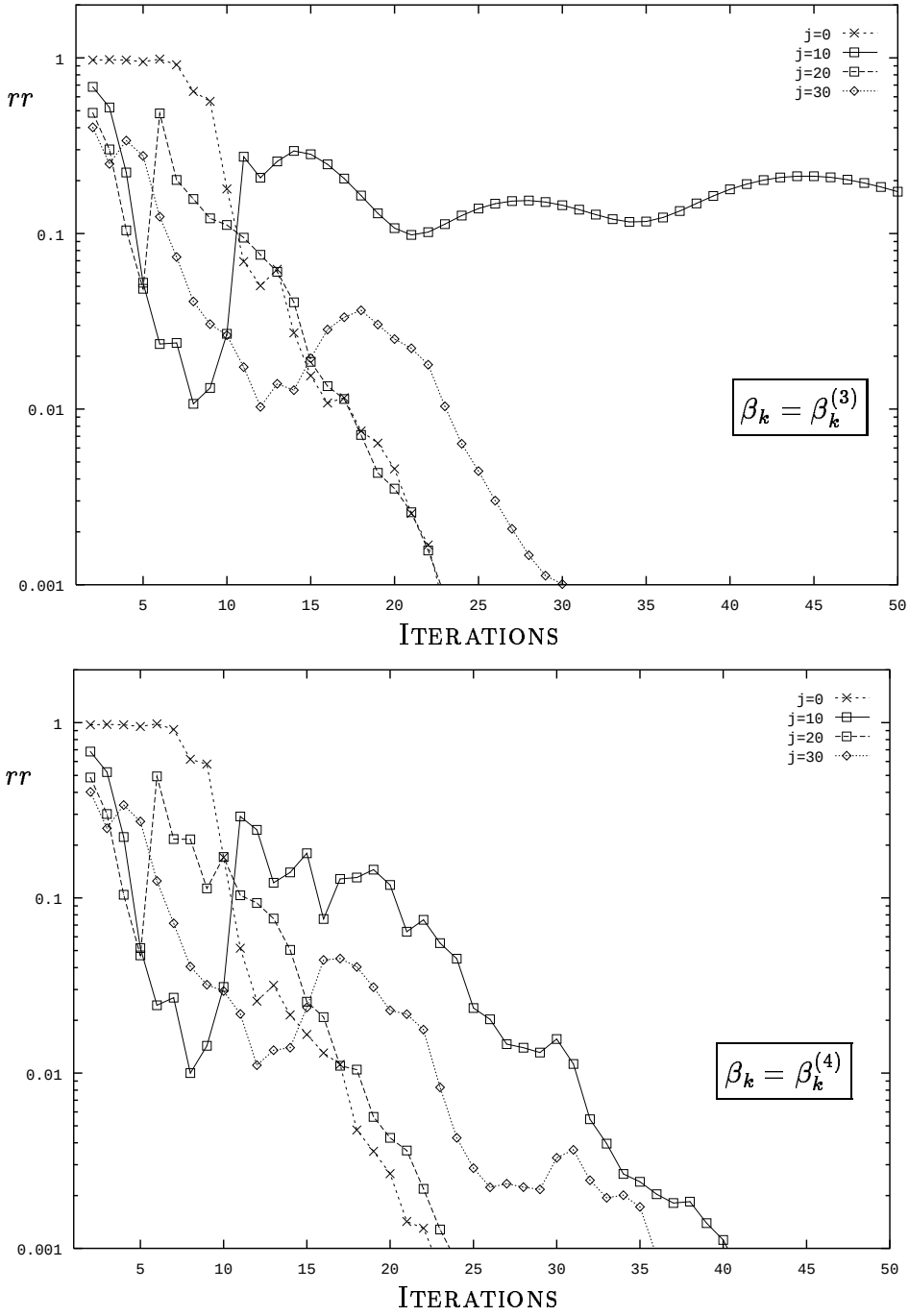

FIG. 3. *The same as Figure 2 with* $\beta_k = \beta^{(3)}$ *and* $\beta_k = \beta^{(4)}$.

	$N = 222$		$N = 441$		$N = 812$		$N = 1952$		$N = 4560$	
j	ϕ_j	ρ_j	ϕ_j	ρ_j	ϕ_j	ρ_j	ϕ_j	ρ_j	ϕ_j	ρ_j
0	3.047	1.458	4.540	1.269	1.952	0.532	2.834	0.448	1.136	0.672
1	2.247	1.279	1.937	0.910	2.744	0.584	1.959	0.392	0.270	0.308
2	2.247	1.942	1.308	0.883	1.351	0.518	1.959	0.380	1.254	0.656
3	0.502	0.487	0.970	0.968	2.163	0.742	0.502	0.141	0.707	0.504
4	0.502	0.379	1.373	1.152	2.124	0.792	0.502	0.123	0.507	0.388
5	1.408	1.182	0.023	0.023	1.985	0.748	1.112	0.321	0.691	0.537
6	0.821	0.747	0.563	0.536	2.304	0.807	0.630	0.345	0.078	0.078
7	0.821	0.740	0.899	0.847	2.168	0.749	0.630	0.257	0.572	0.405
8	0.852	0.817	0.433	0.445	2.235	0.890	0.744	0.306	0.232	0.327
9	0.852	0.870	0.366	0.374	2.215	1.034	0.744	0.307	0.386	0.488
10	1.149	0.957	0.312	0.319	2.308	1.086	0.051	0.053	0.131	0.151
11	1.149	0.889	0.330	0.339	2.272	0.912	0.641	0.308	0.645	0.566
12	0.044	0.044	0.583	0.595	0.300	0.307	0.165	0.206	0.054	0.054
13	0.044	0.044	0.221	0.224	0.088	0.051	0.165	0.195	0.159	0.171
14	0.156	0.172	0.052	0.053	0.147	0.126	0.400	0.328	0.358	0.397
15	0.156	0.172	0.446	0.454	0.225	0.206	0.677	0.305	0.134	0.218
16	0.577	0.543	0.390	0.399	0.291	0.294	0.677	0.388	0.503	0.573
17	1.018	0.922	0.218	0.222	0.354	0.325	0.230	0.230	0.076	0.076
18	1.018	1.058	0.038	0.041	0.414	0.399	0.230	0.246	0.132	0.134
19	0.535	0.549	0.228	0.237	0.454	0.462	0.196	0.206	0.096	0.103
29	0.621	0.558	0.189	0.199	0.101	0.074	0.783	0.554	0.078	0.078
39	0.105	0.112	0.122	0.146	0.125	0.104	0.215	0.221	0.021	0.021

TABLE 1

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

		$K^{-1} = A^{-1}$		$K^{-1} = (LL^T)^{-1}$		$K^{-1} = D^{-1}$	
problem	T_{ORT}	$T_{A^{-1}}$	T_{it}	$T_{(LL^T)^{-1}}$	T_{it}	$T_{D^{-1}}$	T_{it}
$N = 222$	0.097	0.876	0.973	0.057	0.154	0.020	0.117
$N = 441$	0.225	1.427	1.652	0.080	0.305	0.030	0.255
$N = 812$	0.474	3.375	3.849	0.118	0.592	*	*
$N = 1952$	1.042	12.093	13.135	0.370	1.412	0.122	1.164
$N = 4560$	3.200	41.134	44.334	1.476	4.676	0.334	3.554
* DACG does not converge within IMAX= 500.							

TABLE 2

Comparison of the average time (s) per iteration T_{it} ($T_{it} = T_{ORT} + T_{K^{-1}}$) for DACG with $\beta_k = \beta_k^{(1)}$ and three different preconditioners in the calculation of the 40 leftmost eigenpairs. ($TOL2$=10^{-3})

	$N = 222$		$N = 441$		$N = 812$		$N = 1952$		$N = 4560$	
K^{-1}	Time	# it	Time	# it	Time	# it	Time	# it	Time	# it
A^{-1}	1059	1088	4989	3020	7233	1789	24156	1839	162130	3657
$(LL^T)^{-1}$	294	1901	877	2873	2137	2720	4562	3232	13584	2905
D^{-1}	398	3409	1098	4303	*	*	5524	4745	57632	16216
* DACG does not converge within IMAX= 500										

TABLE 3

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. ($TOL2 = 10^{-3}$)

	$N = 222$		$N = 441$		$N = 812$		$N = 1952$		$N = 4560$	
	# it.	Time	# it.	Time	# it.	Time	# it.	Time	# it.	Time
1)	1901	294	2873	877	2720	2137	3232	4562	2905	13584
2)	820	144	1119	365	1058	693	1656	2440	1328	6806
3)	1185	171	1344	366	1022	739	1903	2423	1290	5746
4)	701	111	1039	303	1078	792	1834	2431	1303	5827
1)/4)	2.71	2.65	2.76	2.89	2.52	2.70	1.76	1.88	2.23	2.33

1) $\beta_k = \beta_k^{(1)} = \frac{\mathbf{p}_{k-1}^T A (LL^T)^{-1} \mathbf{g}_k}{\mathbf{p}_{k-1}^T A \mathbf{p}_{k-1}}$ 2) $\beta_k = \beta_k^{(2)} = \frac{\mathbf{p}_{k-1}^T (A - \gamma_j B)(LL^T)^{-1} \mathbf{g}_k}{\mathbf{p}_{k-1}^T (A - \gamma_j B) \mathbf{p}_{k-1}}$

3) $\beta_k = \beta_k^{(3)} = \frac{\mathbf{g}_k^T (LL^T)^{-1} \mathbf{g}_k}{\mathbf{g}_{k-1}^T (LL^T)^{-1} \mathbf{g}_{k-1}}$ 4) $\beta_k = \beta_k^{(4)} = \frac{\mathbf{g}_k^T (LL^T)^{-1} (\mathbf{g}_k - \mathbf{g}_{k-1})}{\mathbf{g}_{k-1}^T (LL^T)^{-1} \mathbf{g}_{k-1}}$

TABLE 4

DACG performance vs parameter β_k in terms of total number of iterations and CPU time (s) in the calculation of the 40 leftmost eigenpairs ($TOL2$=10^{-3}).

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

TABLE 5

Comparison between the expected number of iterations k_j, eq. (26), 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 (ϕ_j, eq. (29)) and the numerical (ρ_j) convergence rate for the $N = 441$ problem. $K^{-1} = (LL^T)^{-1}$.

	$N = 222$			$N = 441$			$N = 812$			$N = 1952$			$N = 4560$		
j	a	b	c	a	b	c	a	b	c	a	b	c	a	b	c
5	8	24	3.0	19	50	2.6	44	90	2.1	315	762	2.5	319	1976	6.2
10	15	45	3.0	44	86	2.0	78	158	2.0	460	1036	2.3	747	2998	4.0
20	36	60	1.7	128	168	1.3	184	342	1.9	809	1741	2.2	1676	5093	3.0
40	96	100	1.0	258	334	1.3	568	657	1.2	1644	3732	2.3	4208	10968	2.5

TABLE 6

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