INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING, VOL. 37, 605–621 (1994)

A COMPARISON OF LANCZOS AND OPTIMIZATION METHODS IN THE PARTIAL SOLUTION OF SPARSE SYMMETRIC EIGENPROBLEMS

GIUSEPPE GAMBOLATI AND MARIO PUTTI

Dipartimento di Metodi e Modelli Matematici per le Scienze Applicate-Università degli Studi di Padova, Via Belzoni 7, 35131 Padova, Italy

SUMMARY

In the present paper, we analyse the computational performance of the Lanczos method and a recent optimization technique for the calculation of the p ($p \leqslant 40$) leftmost eigenpairs of generalized symmetric eigenproblems arising from the finite element integration of elliptic PDEs. The accelerated conjugate gradient method is used to minimize successive Rayleigh quotients defined in deflated subspaces of decreasing size. The pointwise Lanczos scheme is employed in combination with both the Cholesky factorization of the stiffness matrix and the preconditioned conjugate gradient method for evaluating the recursive Lanczos vectors. The three algorithms are applied to five sample problems of varying size up to almost 5000. The numerical results show that the Lanczos approach with Cholesky triangularization is generally faster (up to a factor of 5) for small to moderately large matrices, while the optimization method is superior for large problems in terms of both storage requirement and CPU time. In the large case, the Lanczos–Cholesky scheme may be very expensive to run even on modern quite powerful computers.

1. INTRODUCTION

The Lanczos method[1–5] is attracting much attention for solving large sparse generalized eigenproblems that occur in structural mechanics or hydrodynamics:

$$A\mathbf{v} = \lambda B\mathbf{v} \tag{1}$$

The matrix pencil A, B is symmetric and positive-definite with a high degree of sparsity; A is the stiffness matrix, B is the capacity (or mass) matrix and λ, $\mathbf{v}$ indicates an eigenpair. The characteristic values λ are ordered sequentially so that $\lambda_N \leqslant \lambda_{N-1} \leqslant \cdots \leqslant \lambda_1$, N being the dimension of the eigenproblem.

Recently, the accelerated Conjugate Gradient (CG) method has also proved to be an attractive tool for the partial solution[6–8] of equation (1). Its latest version[8] is simple and interesting at the same time in that it combines a sequential deflation procedure with the CG minimization of several Rayleigh quotients defined in subspaces of decreasing size $N - j$, $j = 0, 1, 2, \ldots$, and, unlike other commonly used techniques, does not require the assessment of any acceleration parameter, which is usually difficult to estimate in engineering applications.

The objective of the present paper is to analyse and compare the numerical performance of the pointwise Lanczos method and the Deflation-Accelerated Conjugate Gradient (DACG) technique for the evaluation of the p ($p \leqslant 40$) leftmost eigenpairs of equation (1) in a number of practical eigenproblems of increasing size N of up to almost 5000. The examples arise from the Finite Element (FE) integration of a diffusion-type partial differential equation in 2-D or 3-D

0029–5981/94/040605–17$13.50

Received 17 January 1992
Revised 24 August 1993

spaces, and are irregularly structured with a high degree of sparsity. The pointwise Lanczos scheme, which may be viewed as a block Lanczos technique with unit block size, with partial B-reorthogonalization[9] has been selected since the block Lanczos version with small block size (1, 2 or 3) is simpler and more convenient.[10] For block generalization versions, the reader is referred to References 11–13. For the Lanczos spectral transformation which involves the triangular factorization of several shifted matrices $A - \sigma B$ for various shift parameters σ, see Reference 14.

We have implemented the Lanczos approach in two different computer algorithms. One, called LANCZOS1, relies on the direct triangular factorization of the matrix A. Efficiency is improved by a preliminary reordering of rows and columns of A meant to achieve a reduction of the fill-in of the triangular factor L. The other, called LANCZOS2, constructs the Lanczos vectors using the preconditioned CG method with the same preconditioning scheme as for the solution of large sparse systems of linear equations.[15–22] The preconditioning matrix used in LANCZOS2 is the same as the one adopted in the DACG scheme. LANCZOS2 should be particularly effective for large problems where LANCZOS1 may become inefficient, or even fail to work, because of insufficient available core storage for the factorization of A.

The comparison of the DACG and the two Lanczos techniques is made basically in terms of CPU times required to converge with a prescribed accuracy level on a modern medium-size mainframe such as the IBM 9370/30 computer available in our department. However, storage requirements, when turning into a critical factor (large problems), will also be considered. The number of smallest eigenpairs to be determined is 1, 5, 10, 20 and 40. The matrix pencil will be taken with an irregular (unstructured) pattern of non-zero coefficients. It will be shown that both approaches have a range of advantageous applicability that depends on the size of the desired partial eigenspace as well as the size of the eigenproblem to be solved.

An analysis of the convergence of the DACG and Lanczos procedures will also be performed. Typically, Lanczos provides more accurate results than the DACG method. The final accuracy in Lanczos is, however, beyond our control, and may indeed be higher than is prescribed in practical applications. For problems with closely clustered eigenvalues, results show that the DACG iteration may be conveniently stopped when a stationary mode shape has been approached with a sufficient accuracy although it is not the theoretically leftmost one at the current search level. This may imply the need for a local reordering of the eigenpairs at the end of the calculation. On the other hand, a similar outcome exists for the Lanczos method where convergence for eigenpairs does not ensure the completeness of the leftmost part of the eigenspectrum. We will also indicate the increase in computational load per DACG iteration as the deflation level increases and the cost for solving the tridiagonal eigenproblem in the Lanczos approach. Finally, a set of concluding remarks emphasizing the most salient features and most convenient range of application of the DACG and Lanczos methods are included.

2. DACG AND LANCZOS METHODS

The DACG method, used in the present work, is described by Gambolati *et al.*,[8] while a theoretical analysis of its asymptotic convergence properties may be found in a recent contribution.[23] A few remarks should be mentioned. First, the traditional CG procedure is accelerated by the inexpensive preconditioner supplied by the incomplete Cholesky factorization of A.[17,18] Second, the Rayleigh quotients

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

are minimized by the preconditioned CG in subspaces of decreasing size $N-j, j=0, 1, 2, \ldots$, B-orthogonal to the j leftmost eigenvectors previously found. Third, convergence is controlled by the relative residual

$$r_r = \frac{|A\mathbf{x}_k - R(\mathbf{x}_k)B\mathbf{x}_k|}{|A\mathbf{x}_k|} \tag{2}$$

where $\mathbf{x}_k$ is the kth iterate approximating the $(N-j)$th eigenvector at the current j deflation step and $|\cdot|$ is the Euclidean norm. Fourth, convergence of the DACG algorithm to a single eigenpair is usually non-monotonic and the initial non-asymptotic behaviour may persist for several iterations. It may happen that the method initially converges to a stationary mode shape that is not the smallest one at that specific deflation level. If at this point the test on the tolerance prescribed for quantity (2) is satisfied, the iteration may complete without the asymptotic behaviour being achieved. In this case, an eigenpair close to the leftmost one is found. The probability to converge during the initial stage is very much related to the exit value TOL prescribed for r_r. By properly reducing TOL, we can always drive the DACG procedure to terminate the iteration during the asymptotic stage, thus locating the correct minimal stationary point. Selection of a proper (not too small) TOL may, however, accelerate the overall calculation and yield a significant saving of CPU time, although a final reordering of the eigenpairs may prove necessary in this case.

Concerning the Lanczos method, we use the following recursive equation to build the Lanczos vector sequence[2] $\mathbf{q}_j$:

$$\beta_{j+1}\mathbf{q}_{j+1} = \mathbf{r}_j = A^{-1}B\mathbf{q}_j - \alpha_j\mathbf{q}_j - \beta_j\mathbf{q}_{j-1}, \quad j = 1, 2, \ldots \tag{3}$$

where

$$\begin{aligned}
&\beta_1 = 0,\\
&\mathbf{r}_0 = \text{arbitrary},\\
&\mathbf{q}_1 = \mathbf{r}_0/\sqrt{\mathbf{r}_0^{\mathrm{T}}B\mathbf{r}_0},\\
&\alpha_j = \mathbf{q}_j^{\mathrm{T}}BA^{-1}B\mathbf{q}_j,\\
&\beta_{j+1} = \mathbf{q}_{j+1}^{\mathrm{T}}B\mathbf{r}_j = \sqrt{\mathbf{r}_j^{\mathrm{T}}B\mathbf{r}_j}.
\end{aligned}$$

If the starting vector $\mathbf{r}_0$ is B-orthogonal to s eigenvectors of (1), in exact arithmetic the sequence (3) leads to $\mathbf{q}_{j+1} = 0$ with $j = N - s$. The same result holds for arbitrary $\mathbf{r}_0$ if the matrix pencil A, B has only $N - s$ distinct eigenvalues. If we find $\mathbf{q}_{j+1} = 0$, the Lanczos process breaks down and therefore it cannot be used in exact arithmetic to detect repeated eigenvalues. In practical calculations, however, the effects of round-off ensure that no difficulties arise in correctly determining eigenvalues that are equal in the first significant decimal digits. The same result may hold for numerically multiple eigenvalues, if any, due simultaneously to round-off and B-reorthogonalization performed on vectors $\mathbf{q}_j$ and described below.

The most expensive operation of the Lanczos process is the matrix-vector product

$$\mathbf{y}_j = A^{-1}B\mathbf{q}_j \tag{4}$$

One way (LANCZOS1) to perform this product is to factorize A as $A = LL^{\mathrm{T}}$, L being lower triangular and to compute $\mathbf{y}_j$ as $\mathbf{y}_j = (LL^{\mathrm{T}})^{-1}B\mathbf{q}_j$. Once we possess L, the product between $(LL^{\mathrm{T}})^{-1}$ and a vector is a relatively cheap operation (its actual cost depends on the fill-in of L). It is known that factorization may produce a large amount of new non-zero coefficients in L,

especially for matrices with a large bandwidth and an irregular sparsity structure. Depending on the size of the original eigenproblem, the fill-in of L can be so pronounced as to make the triangularization infeasible on the computer in use. Moreover, if the number of L-coefficients is much larger than that of A (say one order of magnitude or more), the cost to perform $(LL^{\mathrm{T}})^{-1} B\mathbf{q}_j$ becomes significant, and hence it pays to plan a reduction of the fill-in of L by preliminary reordering the rows and columns of A. This operation, termed Optimal Preliminary Reordering (OPR), is actually accomplished in the LANCZOS1 procedure with a suitable routine taken from the IBM LS Math Scientific Library.[24] The OPR algorithm is intended to reduce the fill-in of L and to preserve the symmetry of the reordered matrix.

In view of the difficulty connected with the generation of inevitably large triangular factors (despite OPR), an alternative way (LANCZOS2) to perform product (4) is to solve the linear system

$$A\mathbf{y}_j = B\mathbf{q}_j$$

by the CG method preconditioned with the same preconditioner used in the DACG technique. LANCZOS2 allows for the in-core treatment and solution of very large eigenproblems without any restriction on the bandwidth and pattern of the matrix pencil A, B.

Another difficulty with the Lanczos method in actual computations is a progressive loss of B-orthogonality of the currently generated vector $\mathbf{q}_{j+1}$ with respect to earlier vectors due to round-off errors and cancellations. To monitor the loss of B-orthogonality and remediate it, we follow the approach suggested by Simon[9] and adopted by other authors as well.[25-28] When the loss of B-orthogonality has occurred for vector $\mathbf{q}_{j+1}$ against vector $\mathbf{q}_i$ ($i \leqslant j$), i.e. when

$$\mathbf{q}_{j+1}^{\mathrm{T}} B\mathbf{q}_i > \sqrt{\varepsilon}$$

where ε is the unit round-off error of the computer in use, $\mathbf{q}_{j+1}$ is B-reorthogonalized against all previous vectors and the same is done for $\mathbf{q}_{j+2}$.

As is well known, with Lanczos coefficients α_j and β_{j+1}, $j = 1, 2, \ldots, m$, we form a symmetric tridiagonal matrix T_m of dimension m. The eigenpair $\mu^{(m)}$ and $\mathbf{z}^{(m)}$ of T_m is computed by available standard routines and those of (1) are evaluated by the relationships

$$\lambda_{N-i+1} = \frac{1}{\mu_i^{(m)}}, \quad \mathbf{v}_{N-i+1} = Q_m \mathbf{z}_i^{(m)}, \quad i = 1, 2, \ldots \tag{5}$$

where $Q_m = [\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_m]$.

It has been recognized by several authors[26, 29] that (5) converges to the extreme (smallest) eigenpair of (1) already for values of $m \ll N$. In this respect, two observations are worth mentioning. First, as m increases, r_{r} becomes small for the eigenpairs correctly found with its final magnitude essentially related to machine accuracy. Second, r_{r} decreases with m in a much more irregular manner than it does in the DACG procedure with the iteration number k. For some higher eigenvectors, the test on the tolerance TOL may be prematurely satisfied in the subspace Q_m, and hence these eigenvectors may have an incorrect position within the Lanczos partial eigenspectrum, where lower eigenpairs may still be missing. It should be stressed that with the native Lanczos method it is difficult to control the magnitude of r_{r} against m, as can be done in a classical iterative scheme. Hence, depending on machine accuracy, as m increases, r_{r}, for some eigenpairs, may become much smaller than is actually required. In the light of the above remarks, and to make a meaningful comparison with the deflation-CG approach, any eigenpair satisfying the acceptability criterion (i.e. a prescribed TOL) should be accepted, even if the corresponding Lanczos result is much more accurate.

3. SAMPLE PROBLEMS AND NUMERICAL RESULTS

The problems used to experiment with the DACG and the Lanczos methods are typical engineering problems with an irregular sparsity structure and arise from the FE integration of 2-D or 3-D diffusion-type PDEs. The pattern of the non-zero coefficients of the matrix pencil A, B is shown in Figures 1–5 for the various examples. The dimensions are N = 222, 441, 812, 1952 and 4560 with degrees of sparsity 96·9, 98·5, 99·2, 99·6 and 99·7 per cent respectively. The example with $N = 4560$ is related to a three-dimensional finite element problem with tetrahedral elements.[30]

The distributions of the 40 leftmost eigenvalues are shown in Figure 6. Note that for $N = 222$ and $N = 1952$, the characteristic values occur in well-separated small clusters, while in the other examples the eigenspectrum is more uniform with its density increasing to the right. The lengths

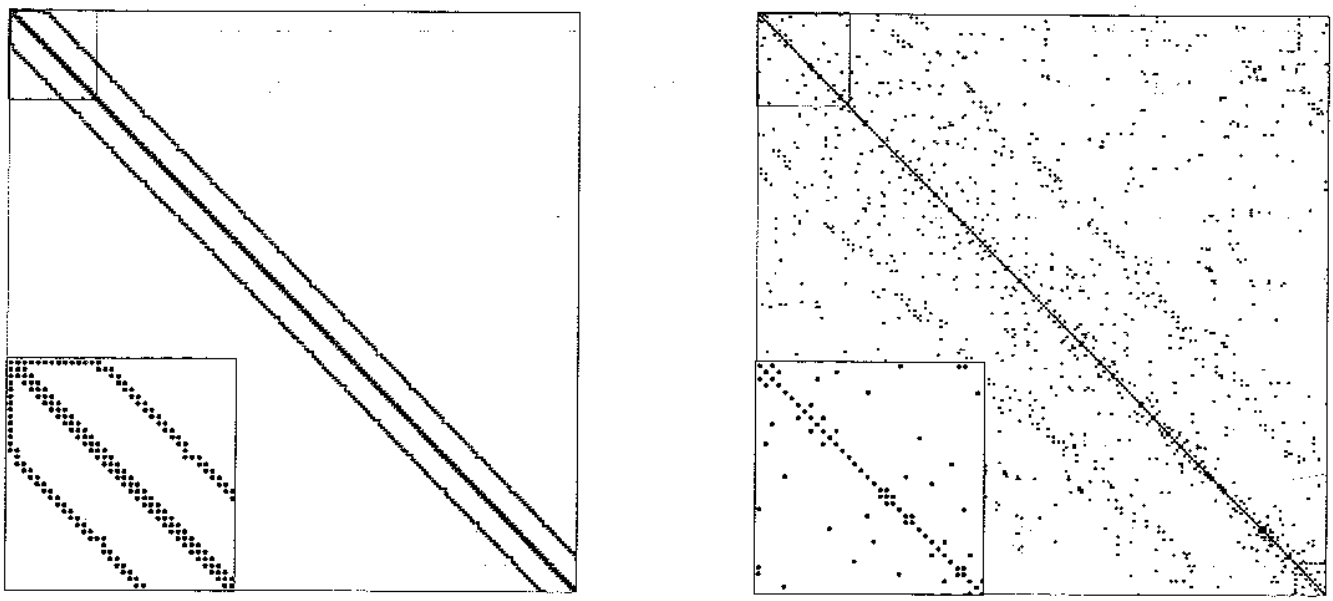

Figure 1. Sparsity pattern for matrix pencil $(A, B)_{222}$: before (left) and after (right) OPR

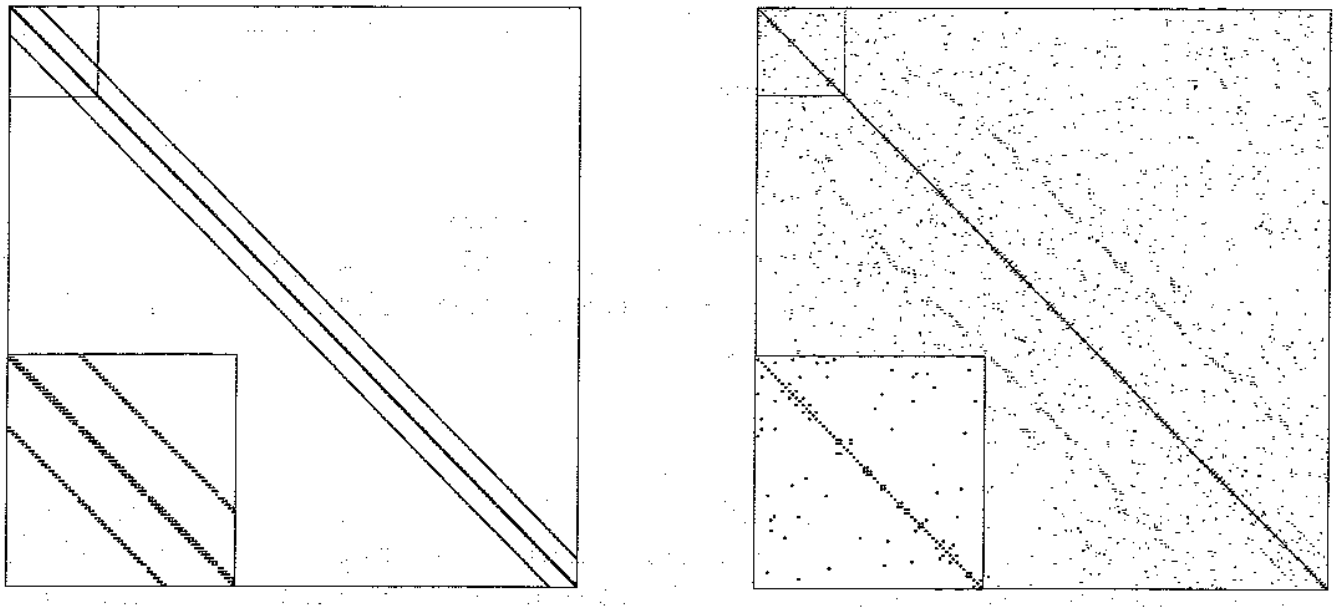

Figure 2. Sparsity pattern for matrix pencil $(A, B)_{441}$: before (left) and after (right) OPR

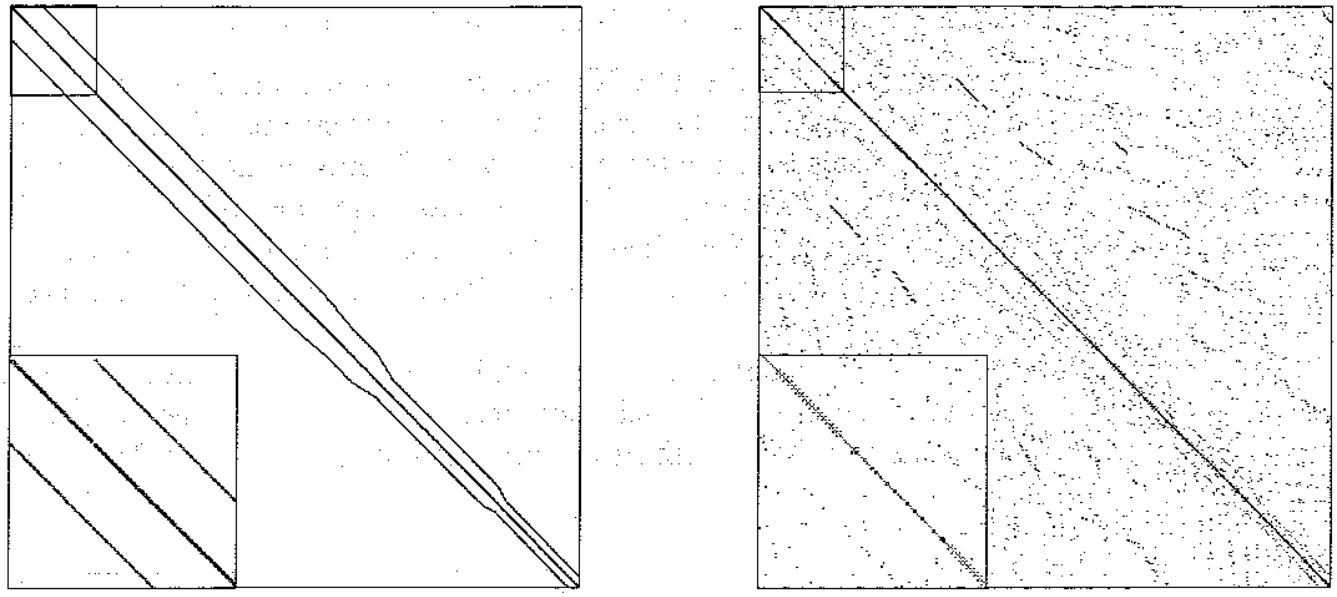

Figure 3. Sparsity pattern for matrix pencil $(A, B)_{812}$: before (left) and after (right) OPR

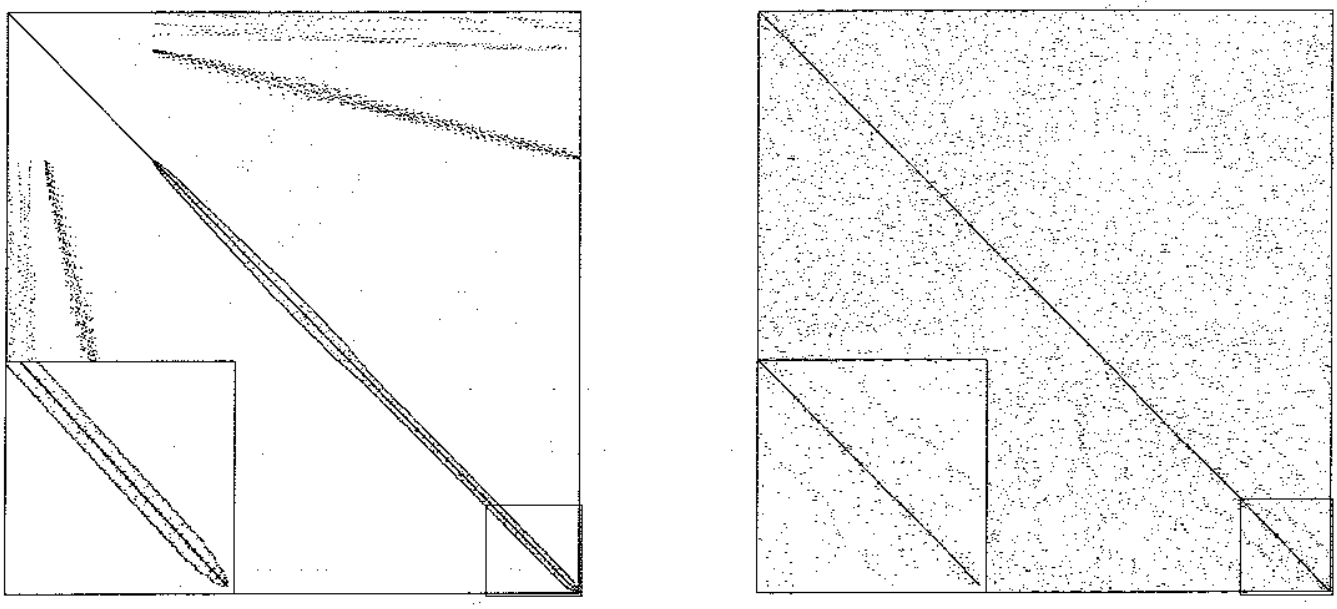

Figure 4. Sparsity pattern for matrix pencil $(A, B)_{1952}$: before (left) and after (right) OPR

of the partial eigenintervals are quite different for the five sample problems: $\lambda_{N-40}/\lambda_N = 240{\cdot}9$ ($N = 222$), 205·8 ($N = 441$), 20636·4 ($N = 812$), 40·1 ($N = 1952$) and 14.8 ($N = 4560$). The large (partial) condition number occurring for $N = 812$ is a probable indication that matrix A_{812} is ill-conditioned, as will be discussed in Section 3.4.

3.1. DACG results

We start by giving some results concerning the DACG convergence. Table I provides the number of iterations at each deflation step needed to meet the exit criteria TOL $= 6 \times 10^{-3}$ and TOL $= 10^{-3}$ (columns a and b, respectively) up to the 40th leftmost eigenpair. Note that meeting the more restrictive criterion requires a much larger number of CG iterations. One reason for this

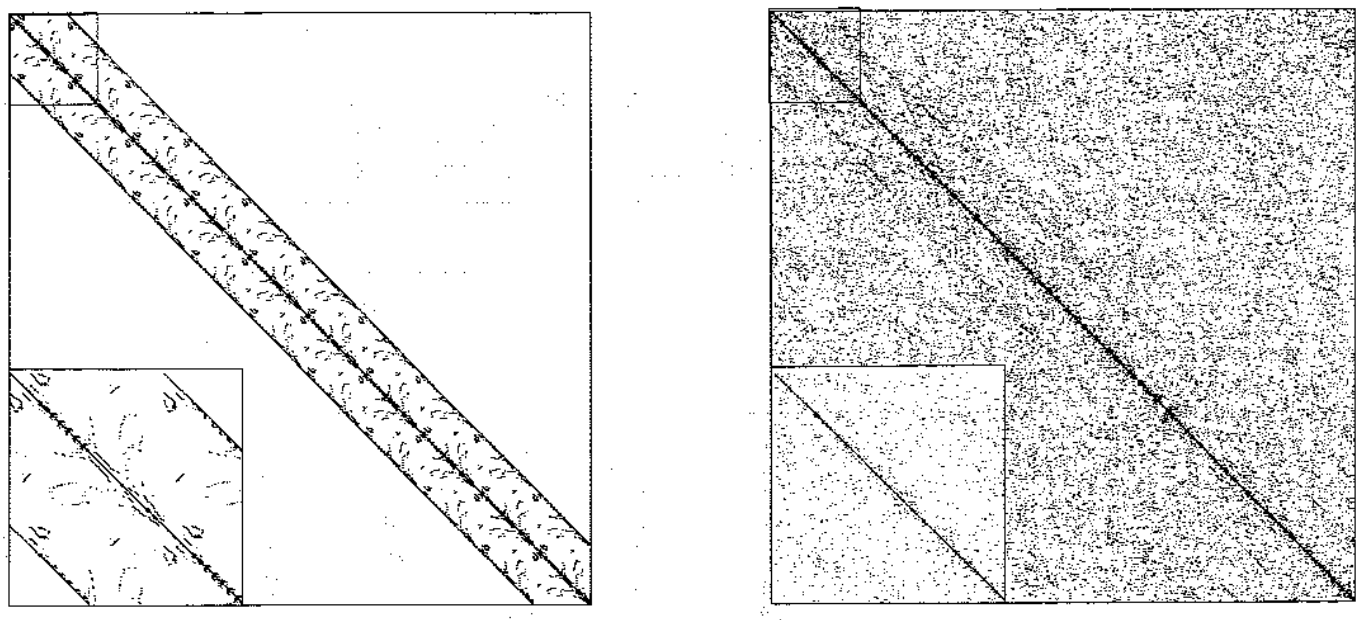

Figure 5. Sparsity pattern for matrix pencil $(A, B)_{4560}$: before (left) and after (right) OPR

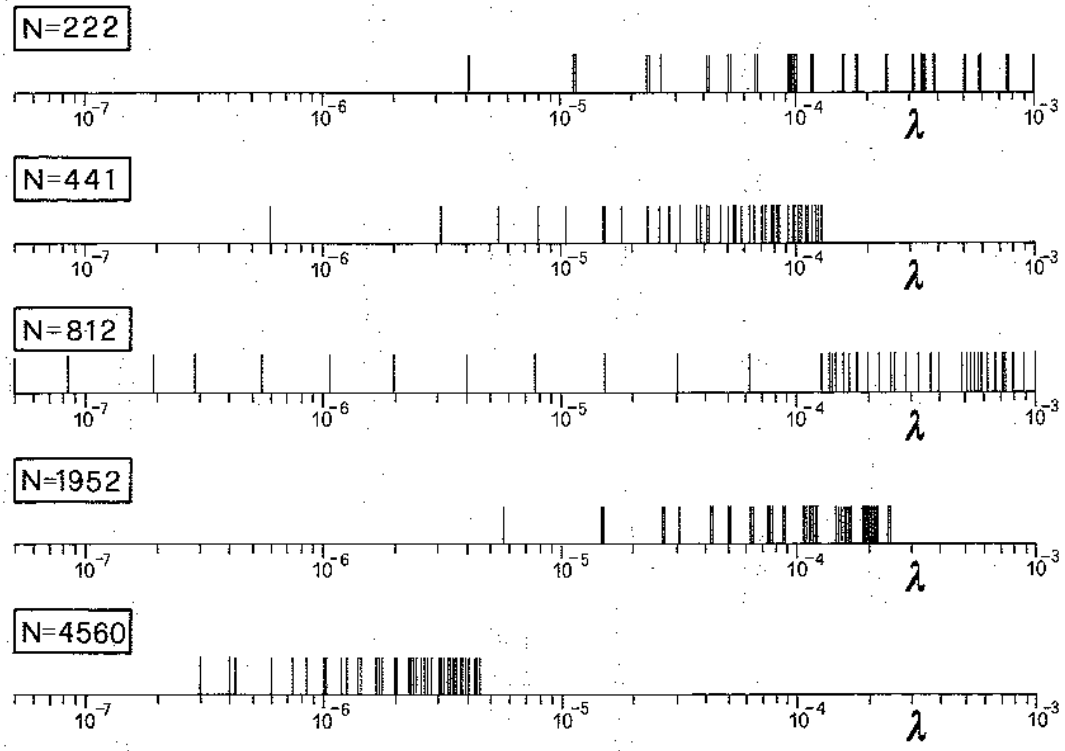

Figure 6. Distribution of the 40 leftmost eigenvalues for the sample matrices

behaviour may be the fact that, in the latter case, the DACG scheme achieves the asymptotic convergence (thus locating the theoretically correct minimal eigenpair), while with the less severe tolerance the iteration is sometimes completed at a characteristic value that is not the optimal one, although very close to it. If, for our practical purposes, a not too strict acceptability test may be employed, the aforementioned behaviour can lead to a significant saving of the overall CPU time. This is indeed the case for TOL $= 6 \times 10^{-3}$ in the examples with $N = 222$ and $N = 1952$ as is also revealed by Table II, which yields the deflation levels at which the DACG procedure has converged to a higher eigenpair than the minimal one. Note that for TOL $= 10^{-3}$ instead, almost all the eigenpairs are evaluated in the correct progressive sequence. Also note (Figure 6) that these

Table I. Number of iterations k required by the DACG scheme to meet two acceptability criteria (TOL $= 6 \times 10^{-3}$, a or TOL $= 10^{-3}$, b) in the evaluation of the 40 leftmost eigenpairs for the various sample problems. Initial vectors are the same as used by Sartoretto *et al.*[7]

	N									
	222		441		812		1952		4560	
j	a	b	a	b	a	b	a	b	a	b
1	12	14	21	21	49	55	39	57	24	30
2	11	13	34	42	23	31	39	47	24	36
3	12	14	16	19	25	30	45	54	17	25
4	28	33	25	30	17	22	49	70	15	24
5	8	49	21	24	16	21	72	90	17	26
6	7	10	13	77	15	19	28	36	23	27
7	11	16	33	32	15	20	22	36	39	81
8	22	27	25	29	14	20	27	45	20	26
9	13	23	29	37	13	18	34	46	24	34
10	18	22	35	45	12	16	42	58	39	44
11	13	14	51	63	11	15	118	192	20	41
12	20	22	9	19	9	18	30	45	19	25
13	63	103	24	28	23	33	90	106	116	188
14	79	226	35	49	15	167	26	29	16	36
15	71	98	99	161	122	57	19	30	23	29
16	75	129	23	31	42	150	16	26	18	33
17	13	15	31	40	83	34	25	35	25	31
18	10	14	32	46	64	110	31	49	83	103
19	25	25	28	330	47	23	69	83	51	79
20	12	27	66	44	40	76	59	80	40	59
21	30	35	31	41	33	20	65	83	50	64
22	14	22	33	63	99	293	21	31	43	52
23	22	21	34	41	24	26	35	49	31	39
24	16	17	39	53	16	20	42	60	30	35
25	23	24	97	96	14	20	59	84	44	49
26	70	93	87	170	29	39	136	205	125	142
27	12	122	12	25	72	50	23	32	45	68
28	68	218	108	158	136	500	66	200	23	43
29	10	14	24	32	108	41	13	21	45	71
30	9	19	61	71	23	117	21	27	71	115
31	10	15	105	199	111	92	67	101	39	109
32	16	25	17	23	36	66	85	147	87	127
33	30	21	25	237	47	118	31	45	25	37
34	28	33	37	43	70	41	81	141	83	149
35	20	21	41	57	44	78	151	249	39	53
36	17	25	59	95	115	63	112	151	45	67
37	13	15	32	154	86	111	21	153	27	39
38	20	23	34	48	20	25	31	29	49	489
39	67	95	36	58	34	37	58	75	89	77
40	50	149	26	42	111	28	95	135	25	103
Σ	1068	1901	1588	2873	1883	2720	2093	3232	1668	2905
Total time (s)	171	294	489	877	1537	2137	2953	4562	7831	13584

Table II. Deflation levels at which the DACG scheme converges to the theoretically smallest eigenpair of another level (shown in parentheses)

N	TOL	Deflation levels
222	6×10^{-3}	5(6)–6(5); 14(15)–15(14); 18(19)–19(18); 26(28)–27(26)–28(29)–29(31)–30(27)–31(30)
	10^{-3}	29(30)–30(29)
441	6×10^{-3}	19(20)–20(19)
	10^{-3}	Correct sequence
812	6×10^{-3}	Correct sequence
	10^{-3}	Correct sequence
1952	6×10^{-3}	4(5)–5(4); 14(15)–15(14); 16(17)–17(16) 27(28)–28(27); 37(38)–38(37)
	10^{-3}	14(15)–15(14); 27(28)–28(27)
4560	6×10^{-3}	38(39)–39(38)
	10^{-3}	Correct sequence

Table III. CPU times (s) per CG iteration vs. the eigenpair level j required by the DACG scheme for the various sample problems

	N				
j	222	441	812	1952	4560
1	0·10	0·19	0·34	0·87	3·3
5	0·13	0·23	0·44	1·0	3·7
10	0·14	0·24	0·58	1·1	4·1
20	0·16	0·31	0·73	1·4	4·5
40	0·20	0·43	1·1	1·8	6·2

two problems practically (at eight significant decimal digits) have multiple eigenvalues. It should be mentioned that the number of CG iterations, which is highly variable from one mode shape to the next (Table I), is controlled by the initial as well as the asymptotic convergence. As was emphasized earlier, the initial convergence to a higher eigenpair may lead to the termination of the DACG process. On the other hand, asymptotic convergence is controlled by the spectral condition number of the Hessian of the preconditioned Rayleigh quotients, and is a function of the relative separation between the theoretically smallest eigenvalue and the subsequent one.[23] As can be inferred from Figure 6, relative separation, $(\lambda_{N-j}-\lambda_{N-j+1})/\lambda_{N-j}$, behaves quite irregularly with j and so does the asymptotic rate of convergence.

It is worth observing that convergence is not the exclusive guide for assessing the computational efficiency of the DACG procedure since the cost per iteration grows with the deflation level, as is shown in Table III. This is due to the increasing expenditure for preserving B-orthogonalization between the current eigenvector and the earlier ones. Roughly, the cost per iteration doubles after 40 deflation steps (Table III). The only exception is the $N = 812$ problem where the cost increases by a factor 3. In this case additional operations for B-orthogonality had to be performed because of ill-conditioning of A.

3.2. *Lanczos results*

Now let us consider the Lanczos method, which is implemented in the LANCZOS1 and LANCZOS2 algorithms. The CPU times needed in LANCZOS1 to reorder and triangularize matrices A are supplied in Table IV, which also gives the number of non-zero coefficients of the lower part of A (including the main diagonal) and that of L with and without OPR. Inspection of Table IV reveals the following: (1) OPR is usually quite effective in reducing the fill-in of L, and hence the cost of the matrix–vector product $(LL^{\mathrm{T}})^{-1}\ B\mathbf{q}_j$; (2) the extra cost to reorder and factorize A is low compared to the cost of the complete Lanczos process (see Table IX), except in the case of the large example ($N = 4560$) for which the construction of L is expensive. This is in agreement with the three-dimensional nature of the $N = 4560$ problem. The average number of CG iterations required in LANCZOS2 to perform the product between A^{-1} and $B\mathbf{q}_j$ is provided in Table V.

When using the Lanczos technique, a few considerations must be kept in mind. First, the Lanczos approach is not iterative in the classical sense. Second, convergence to the leftmost eigenpairs may occur in a very irregular manner. Third, the magnitude of r_r for each eigenpair is actually beyond our control and may become much smaller than is practically required (e.g. much smaller than 10^{-3}).

Figure 7 shows the convergence of the first 6 or 7 eigenpairs for the Lanczos method vs. the number of Lanczos vectors m, for the problem with $N = 222$ (Figure 7(a)), 1952 (Figure 7(b)) and 4560 (Figure 7(c)). In all the examples the initial vector $\mathbf{q}_1$ is set equal to $A^{-1}[1,\ldots,1]^{\mathrm{T}}$ and is B-normalized. Figures 7(a) and 7(b) reveal that r_r may display non-monotonic convergence and, more importantly, that the number m_1 of eigenpairs that satisfy the prescribed tolerance

Table IV. CPU times (s) required to perform the optimal preliminary reordering (OPR) and the triangular factorization (TF) of the reordered matrix A. The number of non-zero coefficients of A and L is also given

	CPU times (s)		Number of coefficients		
N	OPR	TF	Lower part of A	L (without OPR)	L (with OPR)
222	0·75	0·65	872	2994	2849
441	1·7	2·4	1681	9246	5628
812	4·6	3·3	3135	27 988	12 087
1952	26·3	15·4	7744	75 234	38 984
4560	3298·3	2222·9	34 295	1 860 646	834 529

Table V. Average number of preconditioned CG iterations required to perform the product $A^{-1}\,B\mathbf{q}_j$ of equations (3) and (4)

N	Iterations
222	25
441	41
812	53
1952	90
4560	75

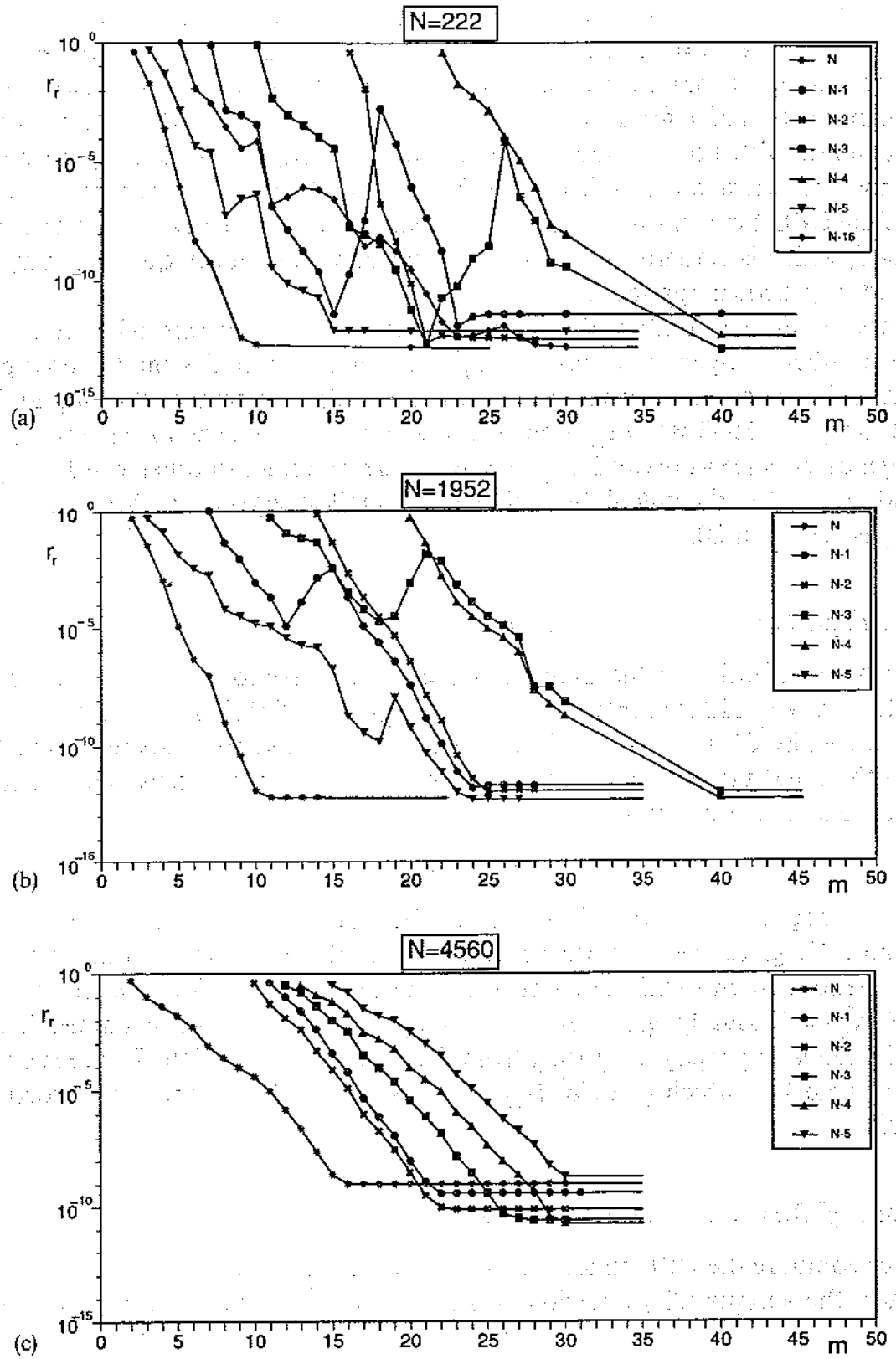

Figure 7. Relative residuals vs. Lanczos subspace dimension in the computation of the leftmost eigenpairs by the Lanczos method for three sample problems

for a given size m of the Lanczos subspace are not necessarily the m_1 smallest eigenpairs of equation (1). As may be seen from Figure 7(a), $\mathbf{v}_{N-5}$ and $\mathbf{v}_{N-16}$ are determined before $\mathbf{v}_{N-1}$, $\mathbf{v}_{N-2}$, $\mathbf{v}_{N-3}$ and $\mathbf{v}_{N-4}$, while $\mathbf{v}_{N-3}$ is determined before $\mathbf{v}_{N-2}$. Hence, we may experience some difficulties in recognizing whether the eigenpairs that satisfy the test are actually the leftmost ones. Also note in Figure 7(a) and 7(b) that an eigenpair determined with good accuracy is not necessarily stable,

and the same eigenpair may be affected by a much larger error in a higher subspace Q_m (in Figure 7(a), $r_r(\mathbf{v}_{N-1})$ is 3×10^{-12} at $m = 15$ and 2×10^{-3} at $m = 18$). A major practical consequence of the previous result is that there may be the need for repeating the Lanczos eigenvalue calculation for increasingly higher values of m for as long as r_r and the leftmost part of the estimated eigenspectrum have stabilized with no additions of intermediate eigenpairs. This may require, for the Lanczos method, an extra cost, which is problem-dependent and is not considered in the comparison of the CPU times shown later. Figure 7 emphasizes the high accuracy attained by the Lanczos method, which is actually comparable to that of a direct technique, and may be higher than is required in practical applications.

Let us denote by m_1 the number of eigenpairs that satisfy the acceptability test, and by m_2 ($m_2 \leqslant m_1$) the number of eigenpairs that overlap without discontinuity the leftmost part of the eigenspectrum of (1). Experience shows that m_2 can be significantly smaller than m_1 (and indeed is in the $N = 222$ and $N = 1952$ sample problems). As a rule of thumb, we can say that to assess the 40 leftmost eigenpairs twice as many Lanczos vectors are required. In other words, $2m_2$ Lanczos vectors should suffice for the calculation of the m_2 smallest eigenpairs. We may need more, however, if m_2 is less than 20.

3.3 Lanczos with partial B-orthogonalization

Partial B-orthogonalization in the Lanczos process is effective for all our problems, except in LANCZOS2 for the $N = 812$ example, and leads to a saving in CPU time of up to almost 50 and 15 per cent in LANCZOS1 and LANCZOS2, respectively, compared to the full B-reorthogonalization. B-reorthogonalization occurs every three to six Lanczos steps, depending on the problem, with the ε value set equal to

$$\varepsilon = |\mathbf{q}_1^T B \mathbf{q}_2| \tag{6}$$

where, following a suggestion by van Kats and van der Vorst,[31] $\mathbf{q}_2$ is always B-reorthogonalized against $\mathbf{q}_1$. Table VI gives the ε values obtained from (6) on our IBM computer in double-precision arithmetic with and without B-reorthogonalization of $\mathbf{q}_2$. Note that in the latter case the values of $|\mathbf{q}_1^T B \mathbf{q}_2|$ usually grow, in particular for the $N = 4560$ example. The loss of B-orthogonality tends to propagate quickly from the very beginning of the Lanczos process as is emphasized in Table VII, which gives $|\mathbf{q}_1^T B \mathbf{q}_j|$, $j = 2, 3, \ldots$, in the absence of B-reorthogonalization ($N = 222$).

3.4. Comparison of DACG and Lanczos methods

Finally, let us compare the CPU times required by the DACG, LANCZOS1 and LANCZOS2 methods to meet the acceptability criterion (TOL $= 6 \times 10^{-3}$). A few considerations should be

Table VI. Values of $|\mathbf{q}_1^T B \mathbf{q}_2|$ with (a) and without (b) B-reorthogonalization of $\mathbf{q}_2$ against $\mathbf{q}_1$ in double-precision arithmetic on the IBM machine 9370/30

	N				
$\mathbf{q}_1^T B \mathbf{q}_2$	222	441	812	1952	4560
a	$0{\cdot}21 \times 10^{-16}$	$-0{\cdot}30 \times 10^{-15}$	$-0{\cdot}12 \times 10^{-14}$	$-0{\cdot}94 \times 10^{-15}$	$-0{\cdot}14 \times 10^{-22}$
b	$0{\cdot}15 \times 10^{-14}$	$0{\cdot}58 \times 10^{-13}$	$0{\cdot}77 \times 10^{-15}$	$-0{\cdot}29 \times 10^{-14}$	$-0{\cdot}43 \times 10^{-9}$

Table VII. Loss of B-orthogonality in the Lanczos method without B-orthogonalization in double-precision arithmetic on the IBM 9370/30 computer (problem with $N = 222$)

j	$\mathbf{q}_1^T B\mathbf{q}_j$
2	$0{\cdot}15 \times 10^{-14}$
3	$0{\cdot}11 \times 10^{-13}$
4	$0{\cdot}58 \times 10^{-12}$
5	$0{\cdot}88 \times 10^{-10}$
6	$0{\cdot}40 \times 10^{-7}$
7	$0{\cdot}10 \times 10^{-5}$
8	$0{\cdot}11 \times 10^{-3}$
9	$0{\cdot}47 \times 10^{-1}$
10	$0{\cdot}62 \times 10^{0}$

Table VIII. CPU times (s) vs. dimension m of matrix T_m required to execute the routine IMTQL2 in double-precision arithmetic on the IBM computer 9370/30

m	Time (s)
5	0·01
10	0·03
20	0·17
40	1·2
80	18·0

mentioned first. For the sample problem with $N = 812$, partial B-reorthogonalization fails to work in the LANCZOS2 algorithm for $m > 40$ (i.e. it is effective only up to $m = 40$) and full B-reorthogonalization proves necessary to calculate correctly 40 eigenpairs. A similar difficulty is experienced with the DACG method as well, where we obtain good estimates up to the 20th smallest eigenpair, the subsequent ones being increasingly incorrect. To preserve the prescribed accuracy up to the end of the calculation, not only the search directions $\mathbf{p}_k$ but also the current iterate $\tilde{\mathbf{x}}_k$ (see Reference 8) have to be B-orthogonalized against previous eigenvectors with an extra cost for the DACG procedure. It is likely that the anomalous behaviour of the DACG and LANCZOS2 algorithms in the $N = 812$ example is accounted for by the relative ill-conditioning of A (in fact, the spectral condition number ξ of A is $\xi(A_{812}) = 4 \times 10^7$ while $10^3 < \xi(A) < 10^6$ for the remaining problems).

The CPU times, which will be shown below, are to be viewed primarily as an indication of the basic computational cost required by each algorithm on a scalar machine. No programming attempt has been made to optimize the corresponding codes. The times are comprehensive of the input–output time, time for the eigensolution of the tridiagonal matrix T_m by the routine IMTQL2 from the Argonne National Laboratory (in the LANCZOS1 and LANCZOS2 codes) and the time for reordering and factorizing matrix A (in the LANCZOS1 code). The time to execute IMTQL2 against m (Table VIII) is a small fraction of the overall time required by the Lanczos method.

Table IX. CPU times (s) on the IBM 9370/30 computer required to evaluate the 40 leftmost eigenpairs at the accuracy level $r_t = 6 \times 10^{-3}$ by the DACG scheme (a), LANCZOS1 (b) and LANCZOS2 (c) algorithms with partial B-reorthogonalization

Number of eigenpairs	N														
	222			441			812			1952			4560		
	a	b	c	a	b	c	a	b	c	a	b	c	a	b	c
1	1·2	3·5	5	3·9	7·2	19	16·6	16	57	34	58	158	79	5870	584
5	8.4	8	24	27	11	50	56	19	90	254	95	762	422	6030	1976
10	18	14	45	67	16	86	91	25	158	519	123	1036	967	6190	2998
20	73	23	60	172	34	168	407	61	342	1071	210	1741	2855	6500	5093
40	171	40	100	489	97	334	1537	140	657	2953	650	3732	7831	7400	10968

The LANCZOS1 code could not be run on our IBM 9370/30 computer for the $N = 4560$ problem since the number of L non-zero elements exceeded the available core memory (16 MB). This example had to be run on the IBM RISC/6000 and the corresponding time was scaled up by one order of magnitude (actually by a factor 12) to make it consistent with the other IBM 9370/30 times. Even with the much larger core provision offered by the RISC/6000 (32 MB), only the version of LANCZOS1 with OPR could be kept in memory for the $N = 4560$ sample test.

Table IX shows the CPU times required to compute 1, 5, 10, 20 and 40 leftmost eigenpairs of (1) with TOL $= 6 \times 10^{-3}$ by the DACG, LANCZOS1 and LANCZOS2 methods. Careful inspection of this table points out that no algorithm is definitely superior to the others. With the significant exception of the large example, LANCZOS1 is better and sometimes much better than DACG (up to a factor 5 for the $N = 222$, 441 and 1952 cases and a factor 10 for $N = 812$ problem. However, we should not forget that in the $N = 812$ case DACG has to perform extra operations). Observe that, if only the smallest eigenpair, or very few ones, are needed, DACG is superior to LANCZOS1. The latter is generally less expensive than LANCZOS2, except for the $N = 4560$ example, in which case DACG is faster than both LANCZOS1 and LANCZOS2.

It may therefore be concluded that LANCZOS1 is the most efficient method in the calculation of the p ($1 < p \leqslant 40$) leftmost eigenpairs for small-to-medium-size problems whenever the triangular factorization does not generate a too large L-factor and can be done in core. For large matrices the DACG procedure exhibits attractive advantages as it is both faster and less demanding than LANCZOS1 and LANCZOS2 in terms of computer storage. LANCZOS2 may also be superior to LANCZOS1 for large problems. Finally, the DACG method is also to be recommended when only the minimal or very few eigenpairs are sought.

4. CONCLUSIONS

The deflation-accelerated CG and the pointwise Lanczos methods have been tried for the computation of the 40 leftmost eigenpairs of generalized sparse symmetric eigenproblems. For a not too severe relative residual tolerance (TOL $= 5 \times 10^{-3}$), the DACG procedure may converge quite fast. Asymptotic convergence is influenced by the relative separation between adjacent eigenvalues and may behave quite irregularly with the deflation level. The iteration may be completed, however, during the initial convergence phase, thus locating a shape mode, which is not the minimal one at the current deflation step. Hence, a final reordering of the eigenpairs may be required at the end of the DACG process.

The Lanczos method with partial B-reorthogonalization requires a subspace of m Lanczos vectors, which is roughly twice the number of wanted eigenpairs (for $m \geqslant 40$), at least in the sample tests analysed in the present paper. If some eigenvalues are very close to each other (i.e. they are multiple up to several significant decimal digits), the eigenpairs satisfying the acceptability test for a given Lanczos vector subspace are not necessarily the leftmost ones and the Lanczos calculation is to be continued for increasing m until the smallest part of the eigenspectrum has stabilized, with no new addition of intermediate characteristic values.

The DACG and the LANCZOS2 schemes proved the least demanding in terms of storage requirement and appear to be well suited to treat large and very large eigenproblems, especially those with an irregular sparsity structure. However, if sufficient computer storage is available, the LANCZOS1 algorithm, which performs the in-core factorization of the stiffness matrix A, is the least expensive in terms of CPU times up to a factor 5 with respect to the DACG approach. For large matrices, however, the last method is significantly superior to both LANCZOS1 and LANCZOS2, as is also for the computation of only the smallest eigenpair (or very few ones). In summary, LANCZOS1 is recommended for small-to-medium-size eigenproblems, while DACG

(and subordinately LANCZOS2) should be used in connection with very large matrices. Finally, if a high accuracy is required (TOL $\leqslant 10^{-3}$), the Lanczos approach (either LANCZOS1 or LANCZOS2 version) should always be used.

ACKNOWLEDGEMENTS

The authors wish to thank the three anonymous reviewers for their constructive criticisms and many helpful suggestions. The present work has been supported in part by the Italian National Research Council (CNR) under the *Progetto Finalizzato 'Sistemi Informatici e Calcolo Parellelo', Sottoprogetto 'Calcolo Scientifico per Grandi Sistemi'*.

REFERENCES

1. C. Lanczos, 'An iteration method for the solution of the eigenvalue problem of linear differential and integral operators', *J. Res. Nat. Bureau Standard*, **45**, 255–282 (1950).
2. C. C. Paige, 'Computational variants of the Lanczos method for the eigenproblem', *J. Inst. Math. Appl.*, **10**, 373–381 (1972).
3. J. Cullum and R. A. Willoughby, 'Lanczos and the computation in specified intervals of the spectrum of large, sparse, real symmetric matrices', *Proc. Symp. on Sparse Matrix Computation*, Knoxville, TN, 1978.
4. J. Cullum and R. A. Willoughby, 'Fast modal analysis of large, sparse but unstructured symmetric matrices', *Proc. 17th IEEE Conf. on Decision and Control*, S. Diego, CA, 1979.
5. B. N. Parlett, *The Symmetric Eigenvalue Problem*, Prentice-Hall, Englewood Cliffs, N.J., 1980.
6. G. Gambolati, G. Pini and F. Sartoretto, 'An improved iterative optimization technique for the leftmost eigenpairs of large symmetric matrices', *J. Comput. Phys.*, **74**, 41–60, (1988).
7. F. Sartoretto, G. Pini and G. Gambolati, 'Accelerated simultaneous iterations for large finite element eigenproblems', *J. Comput. Phys.*, **81**, 53–69 (1989).
8. G. Gambolati, F. Sartoretto and P. Florian, 'An orthogonal accelerated deflation technique for large symmetric eigenproblems', *Comput. Methods Appl. Mech. Eng.*, **94**, 13–23 (1992).
9. H. D. Simon, 'The Lanczos algorithm with partial reorthogonalization', *Math. Comput.*, **42**, 115–142 (1984).
10. B. Nour-Omid, B. N. Parlett and R. L. Taylor, 'Lanczos versus subspace iteration for solution of eigenvalue problems', *Int. j. numer. methods eng.*, **19**, 859–871 (1983).
11. J. Cullum and W. Donath, 'A block Lanczos algorithm for computing the q algebraically largest eigenvalues and corresponding eigenspace of large, sparse, real symmetric matrices', *Proc. IEEE Conf. Decision and Control*, Phoenix, AZ, 1974.
12. G. H. Golub and R. R. Underwood, 'The block Lanczos method for computing eigenvalues', in J. R. Rice (ed.), *Mathematical Software III*, Academic Press, New York, 1977.
13. G. H. Golub and C. F. van Loan, *Matrix Computation*, Johns Hopkins University Press, Baltimore, 1991.
14. T. Ericsson and A. Ruhe, 'The spectral transformation Lanczos method for the numerical solution of large sparse generalized symmetric eigenvalue problems', *Math. Comput.*, **35**, 1251–1268 (1980).
15. P. Concus, G. H. Golub and D. P. O'Leary, 'A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations', in J. Bunch and D. Rose (eds.), *Sparse Matrix Computation*, Academic Press, New York, 1976.
16. O. Axelsson, 'A class of iterative methods for finite element equations', *Comput. Methods Appl. Mech. Eng.*, **9**, 123–137 (1976).
17. 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. Comput.*, **31**, 148–162 (1977).
18. D. S. Kershaw, 'The incomplete Cholesky-conjugate gradient method for the iterative solution of systems of linear equations', *J. Comput. Phys.*, **26**, 43–65 (1978).
19. G. Gambolati, 'Fast solution to finite element flow equations by Newton iteration and modified conjugate gradient method', *Int. j. numer. methods eng.*, **15**, 661–675 (1980).
20. G. Gambolati, 'Perspective on a modified conjugate gradient method for the solution of linear sets of subsurface equations', in S. Y. Wang *et al.* (eds.), *Proc. IV Int. Conf. Finite Elements in Water Resources*, CM Publications, Southampton, 1980 pp. 2.15–2.30.
21. T. Manteuffel, 'An incomplete factorization technique for positive definite linear systems', *Math. Comput.*, **24**, 473–480 (1980).
22. G. Gambolati and A. M. Perdon, 'The conjugate gradients in flow and land subsidence modeling', in J. Bear and Y. Corapcioglu (eds.), *Fundamentals of Transport Phenomena in Porous Media*, NATO-ASI Series, Applied Sciences 82, Martinus Nijhoff, The Hague, 1984, pp. 953–984.
23. G. Gambolati, 'Solution of large scale eigenvalue problems', in M. Papadrakakis, (ed.), *Solving Large Scale Problems in Mechanics: The Development and Application of Computational Solution Methods*, Wiley, New York 1993, pp. 125–156.

24. IBM, *IBM System/3270 and System/370 Subroutine Library—Mathematics: User's Guide*, 2nd edn, 1974.
25. B. Nour-Omid and R. W. Clough, 'Dynamic analysis of structures using Lanczos coordinates', *Earthquake Eng. Struct. Dyn.*, **12**, 565–577 (1984).
26. T. J. R. Hughes, *The Finite Element Method*, Prentice-Hall, Englewood Cliffs, N.J., 1987.
27. A. L. G. Coutinho, L. Landau, E. C. P. Lima and N. F. F. Ebecken, 'The application of Lanczos mode superposition method in dynamic analysis of offshore structures', *Comput. Struct.*, **25**, 615–625 (1987).
28. W. S. Dunbar and A. D. Woodbury, 'Application of the Lanczos algorithm to the solution of the groundwater flow equation', *Water Resources Res.*, **25**, 551–558, 1989.
29. K. J. Bathe, *Finite Element Procedures in Engineering Analysis*, Prentice-Hall, Englewood Cliffs, N.J., 1982.
30. G. Gambolati, A. di Monaco, G. Galeati, F. Uliana, P. Mosca and C. Mascardi, 'New approaches and applications in subsurface flow modeling: 3-D finite element analysis of dewatering for an electro-nuclear plant', in E. Custodio. *et al.* (ed.), *Ground-water Flow and Quality Modelling*, Reidel, Dordrecht, 1988, pp. 717–759.
31. J. M. van Kats and H. A. van der Vorst, '*Numerical Results of the Paige-Style Lanczos Method for the Computation of Extreme Eigenvalues of Large Sparse Matrices*, A.C.C.U., Reeks 18, Utrecht, The Netherlands, 1976.