Robust Parallel Lanczos Methods for Clustered Eigenvalues*

M. Szularz[1], J. Weston[1], and M. Clint[2]

[1] School of Information & Software Engineering, University of Ulster, Coleraine BT52 1SA, Northern Ireland

[2] Department of Computer Science, The Queen's University of Belfast, Belfast BT7 1NN, Northern Ireland

Abstract. In this paper two recently proposed single-vector Lanczos methods based on a simple restarting strategy are analysed and their suitability for the computation of closely clustered eigenvalues is evaluated. Both algorithms adopt an approach which yields a fixed k-step restarting scheme in which one eigenpair at a time is computed using a deflation technique in which each Lanczos vector generated is orthogonalized against all previously converged eigenvectors. In the first algorithm each newly generated Lanczos vector is also orthogonalised with respect to all of its predecessors; in the second, a selective orthogonalisation strategy permits re-orthogonalization between the Lanczos vectors to be almost completely eliminated. 'Reverse communication' implementations of the algorithms on an MPP Connection Machine CM-200 with 8K processors are discussed. Advantages of the algorithms include the ease with which they cope with genuinely multiple eigenvalues, their guaranteed convergence and their fixed storage requirements.

Key words : Lanczos, restart, deflation, orthogonalization, MPP.

1 The Lanczos Algorithm

Essentially, the Lanczos algorithm [2], [5] generates a sequence of tridiagonal matrices T_j, each with values $(\alpha_1, \ldots, \alpha_j)$ on its main main diagonal and values $(\beta_1, \ldots, \beta_{j-1})$ on its main sub- and super-diagonals, together with a sequence of orthonormal matrices of Lanczos vectors $Q_j = [q_1, \ldots, q_j]$, where $j \leq n$, such that $Q_j^t A Q = T_j$. It can be shown that Q_j is an orthonormal basis for $\mathcal{K}(A, q_1, j)$, the Krylov subspace of order j generated by A and q_1. Let $X^T A X = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$, and where $X = [x_1, \ldots, x_n]$ is orthonormal, be the spectral decomposition of A. Similarly, let $S_j^T T_j S_j = \mathrm{diag}(\theta_1(T_j), \ldots, \theta_j(T_j))$ where $\theta_1 > \theta_2 > \cdots > \theta_j$, and $S_j = (s_{pq})$, be the spectral decomposition of T_j. Then $y_i \in \Re^n$ in $Q_j S_j = Y_j = [y_1, \ldots, y_j]$ is known as the i-th Ritz vector of A for the subspace $range(Q_j)$, and $\theta_i(T_j)$

* This work was supported by the Engineering and Physical Sciences Research Council under grants GR/J41857 and GR/J41864 and was carried out using the facilities of the University of Edinburgh Parallel Computing Centre

is known as the corresponding Ritz value. It can be shown that, for surprisingly small j, the Ritz pair $(\theta_i,\ y_i)$ closely approximates the i-th eigenpair of A, $(\lambda_i,\ x_i)$, provided that the ratio $\mid \lambda_i \mid / \parallel A \parallel_2$ is reasonably close to unity.

1.1 Estimating the Largest Eigenvalue

Let $q_1 \in \Re^n$ be such that $\parallel q_1 \parallel_2 = 1$. Then the Lanczos algorithm for the computation of the largest eigenvalue *only* may be expressed as follows:

Algorithm 1 function$[(\theta_1, y_1)]$ = Lanczos(A, q_1, k)
$\beta_0 \leftarrow 1,\ q_0 \leftarrow 0$
for $j = 1, 2, \ldots, k$
$\alpha_j \leftarrow q_j^T A q_j$
$q_{j+1} \leftarrow (A - \alpha_j I) q_j - \beta_{j-1} q_{j-1}$
$\beta_j \leftarrow \parallel q_{j+1} \parallel_2$
if $\beta_j = 0$ **then** STOP (eigenvalues of T_j are the j largest eigenvalues of A)
$q_{j+1} \leftarrow q_{j+1} / \beta_j$
(A) orthonormalize q_{j+1} against $q_1, \ldots, q_j$
end_for
compute (θ_1, y_1).

It follows that the Ritz pair (θ_1, y_1) returned by Algorithm 1 is an approximation to the eigenpair $(\lambda_1,\ x_1)$ of A. The accuracy of this approximation may be estimated before y_1 is computed on the basis of the error bounds $\mid \beta_j \parallel s_{j1} \mid$ [2], since

$$\mid \beta_j \parallel s_{j1} \mid = \parallel A y_1 - \theta_1 y_1 \parallel_2 \tag{1}$$

Observe that the orthogonality of the Lanczos vectors has been guaranteed throughout by the inclusion of a complete re-orthogonalization of the Lanczos vectors in each iteration (**Step A**).

A major problem associated with the above algorithm is that k must be chosen to be sufficiently large to guarantee the required accuracy in the solution. Moreover, the value of k required is not known in advance. However, two explicit restarting schemes for the computation of the $p \ll n$ largest eigenvalues of a symmetric matrix A have been developed [8], each of which incorporates a *fixed* k-step variant of Algorithm 1 where k is assumed to be *small*. Brief outlines of these schemes are now given.

2 Lanczos with Explicit Restart: EXPRES1

Let approximated quantities be decorated with the $\hat{}$ symbol. Thus, at the j-th Lanczos step, the Ritz pair (θ_i, y_i) is synonymous with $(\hat{\lambda}_i, \hat{x}_i)$. Suppose that the approximated eigenpairs $(\hat{\lambda}_1, \hat{x}_1), \ldots, (\hat{\lambda}_i, \hat{x}_i)$, where $i < p$, are given. Let $\hat{\mathcal{X}}_i = \text{span}\{\hat{x}_1, \ldots, \hat{x}_i\}$ and let $\hat{\mathcal{X}}_i^{\perp}$ be its orthogonal complement in $\Re^n$. Observe that, if the Lanczos vectors $q_1, \ldots q_k$ are constrained to stay in the subspace $\hat{\mathcal{X}}_i^{\perp}$,

the Lanczos algorithm will converge to the Ritz values of A in the subspace $\hat{\mathcal{X}}_i^{\perp}$, viz, the desired approximations $\hat{\lambda}_{i+1}, \hat{\lambda}_{i+2}, \ldots$. This observation provides the basis for the two fixed k-step restarting schemes described below. The first may be expressed as the following algorithm:

Algorithm 2 function$[(\hat{\lambda}_1, \hat{x}_1), \ldots, (\hat{\lambda}_p, \hat{x}_p)]$ = EXPRES1(A, p, k, tol)

$\hat{\mathcal{X}}_0 = 0$

(1) for $i = 1 : p$

(1) choose $q_1 \notin \hat{\mathcal{X}}_{i-1}$

(2) $y_1 \leftarrow q_1$; $\theta_1 \leftarrow y_1^t A y_1$

(3) while $(\frac{\|Ay_1 - \theta_1 y_1\|_2}{|\theta_1|} > tol)$

(1) $(\theta_1, y_1) \longleftarrow$ Lanczos(A, q_1, k)

(2) $q_1 \leftarrow y_1$

end_while

(4) $(\hat{\lambda}_i, \hat{x}_i) \leftarrow (\theta_1, y_1)$

(5) if $i < p$ **then** $\hat{\mathcal{X}}_i \leftarrow \hat{\mathcal{X}}_{i-1} \oplus \text{span}\{\hat{x}_i\}$

end_for

tol is the user supplied tolerance (normally set to **u**, the relative machine accuracy). It is assumed that Algorithm 1, as used in **Step (1.3.1)** above, is modified to include a mechanism for projecting each newly generated Lanczos vector (including q_1) into $\hat{\mathcal{X}}_i^{\perp} \neq 0$. This can obviously be achieved by the explicit orthogonalization of each q_j against $\hat{x}_1, \ldots, \hat{x}_i$. Algorithm 2 as described above is henceforth referred to as EXPRES1. Observe that, since k is completely independent of p, it can always be chosen to be small.

Clearly, since the i-th eigenvalue of A is sought in the subspace $\hat{\mathcal{X}}_{i-1}^{\perp}$, Algorithm 2 is theoretically ideal for coping with closely clustered and genuinely multiple eigenvalues of A. This paper provides numerical evidence that this is indeed the case in practice. Further, it is demonstrated that, in many cases where closely clustered and genuinely multiple eigenvalues occur, Algorithm 2 is significantly more efficient than Sorensen's state-of-the-art routine when implemented in a massively parallel SIMD environment.

3 Lanczos with Explicit Restart: EXPRES2

A finely tuned version of Algorithm 1 has also been developed for use with Algorithm 2 in which the *complete* re-orthogonalization of the Lanczos vectors (**Step A**) is replaced by a *selective* re-orthogonalization strategy [5], [6]. Thus, as soon as the error bound associated with a Ritz vector satisfies

$$| \beta_j || s_{ji} | \leq \sqrt{\mathbf{u}} \, \| A \|_2 \tag{2}$$

the Lanczos process is immediately restarted with the current value of y_1, even when $j < k$. Further, if y_1 triggered the restart, all subsequent q_{j+1} computed are not only projected into $\hat{\mathcal{X}}_i^{\perp} \neq 0$ but are also orthogonalized against the recently computed, 'converging' Ritz vector y_1. This orthogonalization strategy *purges* all

unwanted, 'converging' Ritz vectors, $y_i : i \neq 1$, from the system and considerably reduces the computational overhead associated with the orthogonalisation process. An outline of this version of Algorithm 1 is given below:

Algorithm 3 function$[(\theta_1, y_1)]$ = Lanczos$(A, q_1, k, converging)$
$\beta_0 \leftarrow 1$, $q_0 \leftarrow 0$
(1) for $j = 1, 2, \ldots, k$
(1) if $i \neq 0$ **then** orthogonalize q_j against $\hat{x}_1, \ldots \hat{x}_i$
(2) $\alpha_j \leftarrow q_j^T A q_j$
(3) $q_{j+1} \leftarrow (A - \alpha_j I) q_j - \beta_{j-1} q_{j-1}$
(4) $\beta_j \leftarrow \| q_{j+1} \|_2$
(5) if $\beta_j = 0$ **then** STOP (eigenvalues of T_j are the j largest eigenvalues of A)
(6) $q_{j+1} \leftarrow q_{j+1} / \beta_j$
(7) if $converging = false$ **then**
(1) compute $s_{j1}, \ldots, s_{jj}$
(2) if $\min\{| \beta_j \| s_{j1} |, \ldots, | \beta_j \| s_{jj} |\} = | \beta_j \| s_{jr} | \leq \sqrt{\mathbf{u}} \| A \|_2$ **then**
(1) compute (θ_1, y_1)
(2) if $r = 1$ **then** $converging \leftarrow true$
(3) exit
else
(3) orthogonalize q_{j+1} against q_1
end_if
end_for
(2) compute (θ_1, y_1).

The variant of Algorithm 2 which incorporates the finely tuned version of Algorithm 1 above is referred to as EXPRES2. Note also that this variant requires $converging \leftarrow false$ to be added to **Step (1.2)** in Algorithm 2 and **Step (1.3.1)** to be replaced by $(\theta_1, y_1) \longleftarrow$ Lanczos$(A, q_1, k, converging)$. Further, it can be established formally that each of the restart schemes, EXPRES1 and EXPRES2, guarantees convergence to the p required eigenvalues of A.

4 Implementation on the Connection Machine CM-200

The algorithms have been implemented using a *Reverse Communication* strategy [7], [3] in which the user is responsible for supplying the code for all matrix-vector products of the form Aq_j. Such an approach enables the user to take advantage of the architecture of the target machine and of any available highly optimised code when implementing these computationally expensive operations. Thus, all vectors of length n are declared and stored as distributed CM arrays, whereas all other quantities are confined to the 'front-end' machine. It follows that all massive *saxpy* type operations involving the n-vectors are *fine-grain* parallel operations and, consequently, they are performed on the CM itself using the highly optimised matrix-vector product routines which are available in the *Connection*

Machine Scientific Support Library. These operations include, for example, the dense matrix-vector products in the reorthogonalization steps **(1.1)** and **(1.7.3)** of Algorithm 3 and they occur also in the computation of the Ritz vectors in steps **(2)** and **(1.7.2.1)** of this algorithm. In contrast, the remaining computations, viz. those involving the matrices T_j, are computationally inexpensive and are performed on the 'front-end' machine, thereby enabling heavy use to be made of functions from the BLAS library.

5 Numerical Experience

The performances of the restarting algorithms, EXPRES1 and EXPRES2, have been compared with the performance of the appropriate driver program calling the symmetric Arnoldi (Lanczos) routine 'SSAUPD' from the ARPACK library [4]. Single precision CM-200 versions of the three methods, SSAUPD, EXPRES1 and EXPRES2, have been constructed and numerical experiments using a variety of matrices with closely clustered and/or multiple eigenvalues taken from real world applications have been conducted. Test results for four sparse matrices

	PLAT1919 $n = 1,919$ ($nz = 17159$) $\lambda_1 = 2.9216371$ $\lambda_{64} = 1.3470327$			NOS7 $n = 729$ ($nz = 2,673$) $\lambda_1 = 9.86403 \times 10^6$ $\lambda_{64} = 9.62953 \times 10^2$		
p	SSAUPD $k = \max(16, 2p)$	EXPRES1 $k = 32$	EXPRES2 $k = 32$	SSAUPD $k = \max(16, 2p)$	EXPRES1 $k = 32$	EXPRES2 $k = 32$
1	1.5 (18%)	1.1 (63%)	0.4 (93%)	0.7 (21%)	0.4 (41%)	0.4 (53%)
2	3.3 (22%)	2.2 (62%)	0.9 (92%)	1.4 (18%)	8.1 (42%)	2.5 (83%)
4	3.9 (18%)	3.7 (59%)	2.2 (90%)	4.2 (11%)	24.0 (33%)	3.2 (82%)
8	8.9 (11%)	8.1 (55%)	5.1 (88%)	5.1 (8%)	30.3 (40%)	4.8 (79%)
16	28.1 (5%)	21.9 (50%)	11.9 (81%)	8.5 (4%)	78.9 (34%)	24.1 (65%)
32	123.3 (2%)	53.2 (43%)	29.0 (71%)	18.9 (3%)	107.5 (31%)	48.3 (56%)
64	568.8 (< 1%)	162.4 (33%)	90.7 (62%)	192.8 (1%)	208.2 (24%)	85.0 (79%)
128				1816.4 (≈ 0%)	585.9 (15%)	274.5 (28%)
	61u	**11u**	**14u**	$11\sqrt{\mathbf{u}}$	$4\sqrt{\mathbf{u}}$	$17\sqrt{\mathbf{u}}$

Table 1. Time (in seconds) for matrices PLAT1919 and NOS7 from the PLATZ and LANPRO collections, respectively (figures in brackets show the percentage of total time taken for the computation of matrix-vector products).

selected from the Harwell-Boeing Sparse Matrix Collection [1] where p ranges from 1 to 128 are presented in Tables 1 and 2. In Table 1 the matrix PLAT1919 provides a well known difficult sparse symmetric eigenproblem whose eigenvalues occur in pairs (except for an isolated singleton at zero); the condition number of matrix NOS7 in the same table has the value 1.8×10^9. The matrix BCSSTK19

	BCSSTK19			BCSSTK25		
	$n = 817$ $(nz = 3835)$ $\lambda_1 = 1.92216 \times 10^{15}$ $\lambda_{64} = 6.12629 \times 10^{14}$			$n = 15,439$ $(nz = 133840)$ $\lambda_1 = 1.06002 \times 10^{16}$ $\lambda_{32} = 4.03627 \times 10^{13}$		
p	SSAUPD $k = \max(16, 2p)$	EXPRES1 $k = 32$	EXPRES2 $k = 32$	SSAUPD $k = 32$	EXPRES1 $k = 32$	EXPRES2 $k = 32$
1	1.8 (32)	0.4 (32)	0.2 (34)	18.9 (32)	3.1 (32)	3.0 (34)
2	1.7 (32)	1.1 (96)	0.6 (100)	18.8 (32)	6.2 (64)	11.6 (132)
4	4.1 (58)	2.2 (192)	1.2 (200)	18.8 (32)	12.5 (128)	23.4 (264)
8	6.6 (73)	4.3 (352)	2.7 (432)	42.8 (49)	28.3 (288)	38.2 (432)
16	15.6 (99)	10.0 (736)	6.4 (896)	40.9 (40)	54.4 (544)	71.5 (800)
32	64.2 (148)	28.8 (1696)	21.2 (2,368)		121.5 (1,152)	163.0 (1,760)
64	185.9 (192)	90.9 (3,968)	51.1 (4,480)			
	32u	**14u**	**12u**	**29u**	**269u**	**1,205u**

Table 2. Time (in seconds) for matrices BCSSTK19 and BCSSTK25 from the BCSSTRUC3 collection (figures in brackets show the number of matrix-vector products).

in Table 2 is very poorly conditioned and its computed eigenvalues occurred in clusters. The computed eigenvalues of the matrix BCSSTK25 also occur in clusters. Further, storage restrictions prevented the SSAUPD routine from obtaining partial eigensolutions for this matrix for those cases where $p = 32$ and 64. Table 3 presents the results for a tridiagonal matrix A, all of whose diagonal and off-diagonal elements are set to 2 and 1, respectively. This matrix (henceforth referred to as the (1,2,1) matrix) arises in a variety of discretization problems in numerical mathematics and is known to be fairly pathological and slow to converge. In the header of each Table nz denotes the number of nonzero elements in the upper triangle of the matrix. Also in the case of the SSAUPD routine k

	(1,2,1) Matrix		
	$n = 10,000$ $(nz = 19,999)$ $\lambda_1 = 0.39999959 \times 10^1$, $\lambda_{16} = 0.39999380 \times 10^1$		
p	SSAUPD $k = 32$	EXPRES1 $k = 32$	EXPRES2 $k = 32$
1	32.8 (64)	1.4 (96)	0.7 (100)
2	427.9 (1310)	23.2 (1,538)	9.0 (1,190)
4	474.8 (1,242)	82.0 (5,248)	30.3 (3,882)
8	1114 (2,094)	227.2 (13,632)	85.6 (10,226)
16	2735 (2,763)	584.3 (31,360)	220.8 (23,554)
	706u	**1006u**	**1006u**

Table 3. Time (in seconds) for the (1,2,1) matrix(figures in brackets show the number of matrix-vector products).

denotes the number of Lanczos steps after which an 'implicit' restart is made; the chosen mode is 'exact shifts' [4], [7]. The final row of each Table gives the accuracy of the computed solution as $c\mathbf{u}$ (or $c\sqrt{\mathbf{u}}$), where c is a natural number. This accuracy has been computed 'independently' as

$$\max\left(\frac{\| A\hat{x}_i - \hat{\lambda}_i\hat{x}_i \|_2}{| \hat{\lambda}_i |}\right) \; ; \; i = 1, \ldots, \max(p)$$

In all cases the initial value of q_1 is chosen to be $\frac{1}{\sqrt{n}} \times [1, 1, \ldots, 1]$ and the products Aq_j are computed using the Connection Machine Scientific Support Library routine 'sparse_matvec_mult'. With the exception of the (1,2,1) matrix, the requested tolerance has been set in all cases to $\mathbf{u}$.

6 Conclusions

The results presented in Tables 1 and 2 show clearly that Sorensen's implicit Arnoldi routine and the two explicit restart routines discussed in this paper are capable of computing very accurate partial eigensolutions of real world large sparse symmetric matrices whose eigenvalue distribution contains closely clustered and/or multiple eigenvalues. Thus, in particular, EXPRES1 and EXPRES2 both computed partial eigensolutions of the matrix PLAT1919 of order up to 64 with accuracy better than $14\mathbf{u}$. Observe that SSAUPD computed the same solutions with accuracy better than $61\mathbf{u}$. However, in the case of the (1,2,1) matrix, it was not possible to obtain partial eigensolutions of similar accuracy using any of the algorithms as is indicated in Table 3. Nevertheless, considering the pathological nature of this matrix, the single precision solutions obtained were good.

Table 3 shows that, for the pathological (1,2,1) matrix, the explicit restart routines developed by Szularz et al [8] perform signicantly better than Sorensen's routine [4]. Further, it seems to be the case that, in general, the explicit restart routine EXPRES2 is much more efficient than Sorensen's routine.

It is clear from Tables 2 and 3 that, for any given example, the restart strategies discussed in this paper require significantly more matrix vector products of the form Aq_j than does the SSAUPD routine. Further, Table 1 suggests that a greater proportion of EXPRES2 consists of these products than is the case for the other two algorithms. Considering the efficiency with which these products can be implemented in the highly parallel Connection Machine environment, it is this property of the algorithm which enables it consistently to outperform the others on the CM-200.

The results show also that, with the exception of the matrix BCSSTK25, EXPRES2 is considerably more efficient than EXPRES1. The reduction in the amount of re-orthogonalization required appears to be the significant factor although another important factor is that, if reorthogonalization can be avoided, a 'purer' Krylov subspace is constructed and, consequently, convergence is faster.

The reason why EXPRES1 is more efficient than EXPRES2 in the case of BCSSTK25 is currently unknown. However, it is our opinion that, for those matrices where orthogonalization is an issue, the use of a solver which incorporates full re-orthogonalization of the Lanczos vectors is to be preferred to one which incorporates only a selective re-orthogonalizaton. The matrix BCSSTK25 may belong to this category.

In conclusion, it has been shown that the two algorithms discussed in this paper (together with Sorensen's routine taken from the ARPACK library) are well suited to the computation of partial eigensolutions of large sparse symmetric matrices containing closely clustered or multiple eigenvalues. It has also been demonstrated that one of the algorithms (EXPRES2) is highly competitive with Sorensen's algorithm for problems of this kind when implemented in a highly parallel SIMD environment.

The algorithms are currently being implemented on the Cray T3D and an analysis of their performances on this highly parallel shared distributed memory machine will form the basis of a future paper.

References

1. Duff, I.S., Grimes, R.G., and Lewis, J.G., (1992), 'User's Guide for the Harwell-Boeing Sparse Matrix Collection' (release I), available online ftp `orion.cerfacs.fr`.
2. Golub, G., and Van Loan, C.F., (1989), 'Matrix Computations', John Hopkins University Press, London.
3. O.A. Marques (1995), 'BLZPACK: Description and User's Guide', CERFACS Report TR/PA/95/30, Toulouse, France, October 1995.
4. Lehoucq, R., Sorensen, D.C., and Vu, P.A., (1994), 'SSAUPD : Fortran subroutines for solving large scale eigenvalue problems', Release 2.1, available from `netlib@ornl.gov` in the `scalapack` directory.
5. Parlett, B.N., and Scott, D.S., (1979), 'The Lanczos algorithm with selective orthogonalization', *Math. Comput.*, **33**, 217-238.
6. Simon, H.D., (1984), 'Analysis of the Symmetric Lanczos Algorithm with Reorthogonalization Methods', *Linear Algebra Appl.*, **61**, 101-131.
7. Sorensen, D.C., (1992), 'Implicit Application of Polynomial Filters in a k-step Arnoldi Method', *SIAM J. Matrix Anal. Appl.*, **13**, 357-385.
8. Szularz, M., Weston, J., Clint, M., and Murphy, K., (1996), 'A Highly Parallel Explicitly Restarted Lanczos Algorithm', in *Applied Parallel Computing Industrial Computation and Optimization*, J. Wasniewski, J. Dongarra, K. Madsen and D. Olesen (Eds.), LNCS 1184, Springer-Verlag, 651-660.