IN-61
198084
20 P

NASA Contractor Report 191540

ICASE Report No. 93-71

ICASE

20 Years of Excellence

USING PARALLEL BANDED LINEAR SYSTEM SOLVERS IN GENERALIZED EIGENVALUE PROBLEMS

Hong Zhang
William F. Moss

NASA Contract No. NAS1-19480
September 1993

Institute for Computer Applications in Science and Engineering
NASA Langley Research Center
Hampton, Virginia 23681-0001

Operated by the Universities Space Research Association

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-0001

USING PARALLEL BANDED LINEAR SYSTEM SOLVERS IN GENERALIZED EIGENVALUE PROBLEMS

Hong Zhang[1]
Institute for Computer Applications in Science and Engineering
NASA Langley Research Center
Hampton, VA 23681

and

William F. Moss
Department of Mathematical Sciences
Clemson University
Clemson, SC 29634-1907

ABSTRACT

Subspace iteration is a reliable and cost effective method for solving positive definite banded symmetric generalized eigenproblems, especially in the case of large scale problems. This paper discusses an algorithm that makes use of two parallel banded solvers in subspace iteration. A shift is introduced to decompose the banded linear systems into relatively independent subsystems and to accelerate the iterations. With this shift, an eigenproblem is mapped efficiently into the memories of a multiprocessor and a high speed-up is obtained for parallel implementations. An optimal shift is a shift that balances total computation and communication costs. Under certain conditions, we show how to estimate an optimal shift analytically using the decay rate for the inverse of a banded matrix, and how to improve this estimate. Computational results on iPSC/2 and iPSC/860 multiprocessors are presented.

[1]This research was supported by the National Aeronautics and Space Administration under NASA Contract No. NAS1-19480 while the author was in residence at the Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, VA 23681. Research was also supported in part by the National Science Foundation under contract CCR-9103296. The author is on leave from the Department of Mathematical Sciences, Clemson University, Clemson, SC 29634-1907.

1 Introduction

Eigenvalue problems arise in many areas of physics and engineering such as the analysis of electron orbits in atoms and the stability of structures. Due to the large number of applications of such problems, there is a constant demand for algorithms for computing eigenvalues. The development of efficient algorithms has received considerable attention in the literature [1] [2] [23] [29]. With the increased use of advanced computers, parallel algorithms are also becoming available [11] [17] [19] [20] [22] [24] [28]. Most of these algorithms begin by reduction of the problem to a standard form. This is particularly true for generalized eigenproblems [5].

This paper presents two parallel banded linear solvers and their application for generalized positive definite eigenvalue problems, in which, only a few of the smallest eigenvalues and corresponding eigenvectors are needed to moderate accuracy. This type of problem arises in structural analysis and other engineering fields [2] [15]. Among all solution methods, two families are most popular: subspace iteration [2] [4] [25] [26] and the Lanczos method [12] [16]. The Lanczos method has been shown to be superior to subspace iteration on sequential and vector machines [15] [21]. The comparison of the two families on a parallel computer is relatively unknown. Though the Lanczos method is strongly favored by mathematicians, subspace iteration is more often used in engineering, particularly in structural analysis. This may be because subspace iteration is conceptually simple and closely associated with substructure, based on which, one can often construct an approximate subspace from experience. Many software codes are still using subspace iteration for computing a few dominant eigenvalues and corresponding eigenvectors. The major effort in subspace iteration is devoted to solving banded linear systems. This paper discusses an algorithm that makes use of two parallel banded linear solvers in subspace iteration. The main feature of this algorithm is the introduction of a shift that decomposes the banded linear system into relatively independent subsystems and accelerates the subspace iteration. We shall show when this shift is applicable, how to estimate this shift analytically using the decay rate for the inverse of a banded matrix, and how to improve this estimate. The comparison of subspace iteration with the Lanczos method is beyond the consideration of this paper.

This paper is organized as follows. In Section 2 two parallel tridiagonal solvers [27] are extended to banded solvers. The second of these makes use of the decay of the inverse of banded matrices. Theoretical and numerical results are presented for the comparison of efficiencies. In Section 3, the parallel subspace iteration algorithm for the generalized eigenvalue problem is described. A shift is introduced and analyzed, and computational results are presented. Section 4 concludes by pointing out advantages and limitations of our algorithm.

2 Parallel banded solvers

The Parallel Partition LU (PPT) algorithm and the Parallel Diagonal Dominant (PDD) algorithm were proposed for solving tridiagonal linear systems on multicomputers in [27]. Here we extend them to banded linear systems. The PPT algorithm is similar to an algorithm introduced by Lawrie and Sameh in [18]. The PDD algorithm is a variant of PPT which uses the fact that the entries in the inverse of a banded matrix decay away from the main diagonal.

Let us consider a system of order n

$$Ax = d \tag{1}$$

where A is a nonsingular banded matrix with lower band width m_l and upper band width m_u; specifically, $a_{ij} = 0$ if $i - j > m_l$ or $j - i > m_u$. Let p denote the number of processors used, and for convenience assume that $n = pn_s$ and that the half band width $m := \max(m_l, m_u) < n_s/2$. Following [18] we partition A as a block $p \times p$ matrix with blocks of order n_s:

$$A = \begin{bmatrix} A_1 & C_1 & & & \\ B_2 & A_2 & C_2 & & \\ & B_3 & A_3 & \ddots & \\ & & \ddots & \ddots & C_{p-1} \\ & & & B_p & A_p \end{bmatrix}.$$

Let $\tilde{A} = diag\,[A_1, A_2, \ldots, A_p]$ and write $A = \tilde{A} + \Delta A$. The main assumption of the PPT and PDD algorithms is that both A and $\tilde{A}$ are nonsingular. The $n_s \times n_s$ submatrices B_i and C_i have the form

$$B_i = \begin{bmatrix} 0 & R_i \\ 0 & 0 \end{bmatrix}, \qquad C_i = \begin{bmatrix} 0 & 0 \\ L_i & 0 \end{bmatrix}$$

where R_i is $m_l \times m_l$ upper triangular and L_i is $m_u \times m_u$ lower triangular.

Let V and E be block $p \times (p-1)$ matrices of the form

$$V = \begin{bmatrix} \tilde{C}_1 & & & \\ \tilde{B}_2 & \tilde{C}_2 & & \\ & \tilde{B}_3 & \ddots & \\ & & \ddots & \tilde{C}_{p-1} \\ & & & \tilde{B}_p \end{bmatrix}, \qquad E = \begin{bmatrix} F & & & \\ G & F & & \\ & G & \ddots & \\ & & \ddots & F \\ & & & G \end{bmatrix}$$

with $n_s \times (m_l + m_u)$ blocks

$$\tilde{C}_i = \begin{bmatrix} 0 & 0 \\ 0 & L_i \end{bmatrix}, \quad \tilde{B}_i = \begin{bmatrix} R_i & 0 \\ 0 & 0 \end{bmatrix}, \quad F = \begin{bmatrix} 0 & 0 \\ I_{m_l} & 0 \end{bmatrix}, \quad G = \begin{bmatrix} 0 & I_{m_u} \\ 0 & 0 \end{bmatrix}$$

where I_k denotes the identity matrix of order k. Set $n_r := (m_l + m_u)(p-1)$. Then V and E are $n \times n_r$ matrices and $\Delta A = VE^T$. Next, choose any $n \times (n - n_r)$ matrix H so that $P = [E|H]$ is an $n \times n$ permutation matrix. Then $E^TE = I_{n_r}$, $H^TH = I_{n-n_r}$, $E^TH = 0$, and $H^TE = 0$, and it follows that

$$P^T\tilde{A}^{-1}AP = \begin{bmatrix} Z & 0 \\ H^T\tilde{A}^{-1}V & I_{n-n_r} \end{bmatrix}$$

where $Z = I_{n_r} + E^T\tilde{A}^{-1}V$. Consequently, if A and $\tilde{A}$ are nonsingular, so is Z.

Now using the Sherman-Morrison-Woodbury formula [13], the solution of equation (1) can be expressed as

$$\begin{aligned} x &= A^{-1}d = (\tilde{A} + VE^T)^{-1}d \\ &= \tilde{A}^{-1}d - \tilde{A}^{-1}V(I_{n_r} + E^T\tilde{A}^{-1}V)^{-1}E^T\tilde{A}^{-1}d. \end{aligned}$$

Using this representation equation (1) can be solved in the following steps:

1. Solve $\tilde{A}(\tilde{x}, Y) = (d, V)$
2. Form $Z = I_{n_r} + E^TY$ and $h = E^T\tilde{x}$, then solve
$$Zy = h.$$
3. Calculate $x = \tilde{x} - Yy$.

The matrix Z in Step 2 is called the *reduced* matrix, and has the form

$$Z = \begin{bmatrix} I_{m_l} & Z_{12} & & & & & \\ Z_{21} & I_{m_u} & & Z_{24} & & & \\ Z_{31} & & I_{m_l} & Z_{34} & & & \\ & & Z_{43} & I_{m_u} & & & \\ & & Z_{53} & & \ddots & & Z_{2p-4,2p-2} \\ & & & & & I_{m_l} & Z_{2p-3,2p-2} \\ & & & & & Z_{2p-2,2p-3} & I_{m_u} \end{bmatrix}.$$

The matrix Z can be reformulated as a tridiagonal block matrix by the permutation of the $(2i-1)$-th and the $2i$-th column blocks. Therefore the reduced system can be treated as a banded system with the upper and lower band widths of $m_l + m_u - 1$. We reiterate that the above matrix partitioning scheme is essentially the same as that proposed in [18].

We now outline the PPT algorithm.

Parallel Partition LU (PPT) Algorithm:

- Input: A, d
- Output: x
- In parallel, do on all processors P_i, $i = 1, \ldots, p$:
  1. Solve $A_i(\tilde{x}_i, Y_i) = (d_i, V_i)$, where $V = [V_1, \ldots, V_p]^T$, $Y = [Y_1, \ldots, Y_p]^T$, with V_i and Y_i $n_s \times n_r$ blocks, and $d = [d_1, \ldots, d_p]^T$, $\tilde{x} = [\tilde{x}_1, \ldots, \tilde{x}_p]^T$ with d_i and $\tilde{x}_i$ n_s-vectors. Stop if A_i is algorithmically singular.
  2. Following an all-to-all communication, form the reduced system $Zy = h$.

3. Solve the reduced system.
4. Calculate the $i-$th block of the solution x: $x_i = \tilde{x}_i - Y_i y$.

The total number of data transferred is $(m_l + m_u)(m_l + m_u + 1)(p-1)$ (see [27]). The number of flops is roughly $(8nm^2 + 5nm)/p + 16(p-1)m^3$ with $m = \max(m_l, m_u)$, the half band width.

As indicated in [27], the PPT algorithm performs well for narrow banded systems when the number of processors is small (say $p < 16$). However, the efficiency of the algorithm decreases quickly as either the number of processors p or the half band width m increases, since Steps 2 and 3 of the algorithm are a bottleneck in both computation and data communication. This is true for all known variants of the matrix tearing technique used here [9] [10]. Under certain conditions this bottleneck can be removed. As was already mentioned above, the entries of the inverse of a nonsingular, banded matrix decay away from the main diagonal (see [6]). When the decay in the inverse of $\tilde{A}$ is sufficiently rapid, the reduced matrix Z can be approximated by a block diagonal matrix $\tilde{Z}$ with blocks of order $m_l + m_u$. The PDD algorithm is the PPT algorithm with Z replaced by $\tilde{Z}$. When the PDD algorithm can be used, it is much more efficient than the PPT algorithm.

We now analyze the conditions under which the PDD algorithm is applicable. For simplicity, we assume that the half band width $m = m_l = m_u$. We will use the notation $A(i_1 : i_2, j_1 : j_2)$ to denote the $(i_2 - i_1 + 1) \times (j_2 - j_1 + 1)$ submatrix of A with upper left corner at entry (i_1, j_1) and lower right corner at entry (i_2, j_2). Also, let $\| \;\; \|$ denote the ∞-norm.

For $i = 1, \ldots, p-2$ the off-diagonal blocks $Z_{2i+1,2i-1}$ and $Z_{2i,2i+2}$ are obtained from

$$Z_{2i,2i+2} = A_{i+1}^{-1}(1 : m, n_s - m + 1 : n_s)\, L_{i+1}$$

and

$$Z_{2i+1,2i-1} = A_{i+1}^{-1}(n_s - m + 1 : n_s, 1 : m)\, R_{i+1},$$

and hence

$$\|Z_{2i,2i+2}\| \le \|A_{i+1}^{-1}(1 : m, n_s - m + 1 : n_s)\| \|L_{i+1}\|$$

and

$$\|Z_{2i+1,2i-1}\| \le \|A_{i+1}^{-1}(n_s - m + 1 : n_s, 1 : m)\| \|R_{i+1}\|.$$

Using Proposition 2.3 of [6], it can be shown that there are constants C and q, $C > 0$ and $0 < q < 1$, so that

$$\max_{1 \le i \le p-2} \left\{ \|A_{i+1}^{-1}(1 : m, n_s - m + 1 : n_s)\|, \|A_{i+1}^{-1}(n_s - m + 1 : n_s, 1 : m)\| \right\} \le C q^{dis/m}$$

where $dis = n_s - 2m + 1$ is the shortest distance from the main diagonal to an entry of $A_{i+1}^{-1}(1 : m, n_s - m + 1 : n_s)$ or $A_{i+1}^{-1}(n_s - m + 1 : n_s, 1 : m)$. If A is positive definite, dis can be replaced by $2dis$. When $n_s/m \gg 1$, these off-diagonal blocks can be neglected and the reduced matrix Z can be replaced by the block diagonal approximation $\tilde{Z}$. Our test for applicability of the PDD algorithm is as follows. Let u denote machine precision (unit roundoff), and let c denote a small positive constant. We use PDD if

$$\delta(A) := \frac{\max_{1\leq i\leq p-2}\{\|Z_{2i,2i+2}\|, \|Z_{2i+1,2i-1}\|\}}{\|Z\|} < cu. \tag{2}$$

This means that PDD is used only if the perturbation in Z caused by zeroing the off-diagonal blocks is comparable to the roundoff error introduced by solving the reduced system by a numerically stable method. Equation (2) will be referred as *test (2)* in the following sections.

We now outline the PDD algorithm.

Parallel Diagonal Dominant (PDD) Algorithm:

- Input: m-banded A, d
- Output: x_{dd} (an approximate solution of $Ax = d$)
- In parallel, do on all processors P_i, $i = 1, \dots, p$:
  1. Solve $A_i(\tilde{x}_i, Y_i) = (d_i, V_i)$.
  2. Send $Y_i(1 : m, 2m(i-2)+1 : 2m(i-2)+m)$ and $\tilde{x}_i(1 : m)$ to the processor P_{i-1} $(i \neq 1)$. Form the i-th block of matrix $\tilde{Z}$ $(i \neq p)$:
  $$\begin{bmatrix} I_m & Z_{2i-1,2i} \\ Z_{2i,2i-1} & I_m \end{bmatrix}.$$
  3. Solve
  $$\begin{bmatrix} I_m & Z_{2i-1,2i} \\ Z_{2i,2i-1} & I_m \end{bmatrix} y\,(2m\,(i-1)+1 : 2mi) = h\,(2m\,(i-1)\,m+1 : 2mi)$$
  on P_i $(i \neq p)$, then send $y\,(m\,(2i-1)+1 : 2mi)$ to the processor P_{i+1}.
  4. Calculate the i-th block of x_{dd}: $(x_{dd})_i = \tilde{x}_i - Y_i y$.

Note that the system at Step 3 can be treated as a banded system with half band width m if a permutation of column blocks is used. The total number of flops is roughly $(8nm^2 + 5nm)/p + 4m^3$, and the number of data transferred is $m^2 + 2m$.

Our **Hybrid Algorithm** is simply: use PDD if applicable; otherwise use PPT.

The PPT and the PDD algorithms have been implemented on a 64-node NCUBE-1 computer for tridiagonal linear systems [27]. We present in Tables 1 and 2 the numerical results of the two algorithms for banded matrices on a 16-node Intel iPSC/2 computer. The testing banded matrices are of the form $A = [a_{ij}]$, with

$$a_{ij} = \begin{cases} 1 & \text{if } 0 < |i-j| \leq m \\ 2m+s & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

Table 1:

m	n	p	Execution Time (seconds)				Ratio		
			PPT	SPT	PDD	SDD	SPT/PPT	SDD/PDD	PPT/PDD
10	640	1							
		2	2.624	4.811	2.598	4.768	1.83	1.84	1.01
(s=69)		4	2.224	6.429	1.887	6.144	2.89	3.26	1.18
		8	1.907	7.434	0.922	6.700	3.90	7.27	2.07
10	3200	1							
		2	13.186	24.231	13.160	24.288	1.84	1.85	1.00
(s=-9)		4	10.092	32.117	9.741	31.800	3.18	3.26	1.04
		8	5.801	36.010	4.816	35.160	6.21	7.30	1.20
		16	4.643	38.109	2.370	36.390	8.21	15.35	1.96
20	3200	1							
		2	43.678	78.893	43.514	78.979	1.81	1.82	1.00
(s=12)		4	34.774	106.706	32.629	104.976	3.07	3.22	1.07
		8	22.408	121.335	16.106	116.594	5.41	7.24	1.39
		16	22.319		7.771	118.908		15.30	2.87

where s is chosen so that the off-diagonal blocks in the reduced matrix Z can be neglected. In our numerical experiments, we have used Gauss elimination with partial pivoting to solve the reduced system and a value $c = 10$ for the test in equation (2). In our experiments, the accuracy of the PPT and the PDD algorithms is comparable to working precision.

Table 1 shows the execution time for different algorithms and the relative speedups. SPT and SDD are the sequential algorithms for PPT and PDD. They are executed on a single processor. We use them to measure the relative speedup of the PPT and PDD algorithms. In our tables, n is the order of the test matrix and p is the number of processors being used for all the algorithms except SPT and SDD. For SPT and SDD, p is the number of blocks in the partitioning scheme.

As already indicated, the solution of the reduced system $Zy = h$ is a computation and communication bottleneck of the PPT algorithm. Table 2 contains, for each algorithm, columns for the percentage of time spent on communication (comm. %) and for the time spent on computation (comp. %) at Step 2 and 3. It is clear from both theoretical and experimental results that the PDD algorithm is much more efficient when it is applicable.

3 A Parallel Algorithm for the Banded Generalized Eigenvalue Problem

In this section, we consider generalized symmetric eigenproblems of the form

$$Ax = \lambda Bx, \tag{3}$$

where A and B are symmetric, m-banded matrices and B is positive definite. We wish to approximate the q, $q \ll n$, smallest eigenvalues $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_q$ and corresponding eigenvectors $x_1, \ldots, x_q$. This type of problem frequently arises in vibration mode analysis

Table 2: Percentage of Execution Time in Solving Reduced System $Zy = h$

m	n	p	PPT		PDD	
			comm(%)	comp(%)	comm(%)	comp(%)
10	640	2	0.6	2.6	0.2	1.5
		4	1.2	17.2	0.2	2.3
		8	3.2	54.0	0.7	5.1
	3200	2	0.1	0.5	0.0	0.3
		4	0.3	3.8	0.0	0.4
		8	1.1	17.8	0.1	0.9
		16	2.8	50.0	0.1	1.8
20	3200	2	0.1	0.8	0.0	0.5
		4	0.3	6.9	0.0	0.7
		8	1.0	29.1	0.1	1.5
		16	13.5	77.5	0.1	3.0

[2] [3], [15], where A and B are, respectively, the stiffness matrix and mass matrix. The eigenvalues λ_i are the squares of the the free vibration frequencies and the eigenvectors x_i are the corresponding mode shape vectors.

Many sequential algorithms for solving the symmetric eigenvalue problem have been developed including those in [1]. In recent years parallel algorithms for advanced computers have appeared such as those found in [11] [15] [17] [19] [20] [22] [24]. Most of these algorithms require reduction to tridiagonal form for an equivalent standard problem, or computation of all eigenpairs. The algorithm proposed in this paper is a parallel subspace iteration algorithm for finding the q smallest eigenvalues and corresponding eigenvectors that takes advantage of the assumption that A and B are banded. Subspace iteration [2] [4] [23] [25] [26] is a reliable and cost effective method for solving the eigenvalue problem considered here to moderate accuracy. It has been used extensively in a number of general purpose finite element analysis programs [2].

Using a shift s, a symmetric matrix pencil (A, B), with B positive definite, can be transformed to a positive definite matrix pencil $(A - sB, B)$. This shift leaves eigenspaces unchanged. For the moment let us assume that the pencil (A, B) is positive definite.

The basic subspace iteration algorithm then consists of the following steps:

- Establish a starting subspace of dimension $\tilde{q}$ spanned by the columns of $S^{(0)}$ where $\tilde{q} > q$ and q is the number of eigenpairs to be calculated.
- For $k = 0, 1, \ldots,$ until the first q eigenvalues satisfy a stopping criterion.
  1. apply the Rayleigh-Ritz (RR) procedure to extract the "best" eigenvalue and eigenvector approximations from $S^{(k)}$.
  2. improve $S^{(k)}$ by an inverse iteration

$$AS^{(k+1)} = BS^{(k)}.$$

The starting subspace can be generated as discussed in [2] or can be generated randomly. The user can input the value of $\tilde{q}$; the default value is $\tilde{q} = \min\{2q, q + 8\}$ based on the numerical experiments found in [2]. The details of the RR procedure are described in the RR algorithm :

RR Algorithm:

- Input: S
- Output: eigenpair approximations for $Ax = \lambda Bx$ from S
    1. Orthonormalize the columns of S with respect to B to obtain Q written over S, i.e. $Q^T BQ = I$.
    2. Form the Rayleigh Quotient: $H_A = Q^T AQ$.
    3. Find eigenpairs of H_A: (θ_j, ϕ_j), $j = 1, \ldots, \tilde{q}$.
    4. Form the "best" eigenpair approximations from S: (θ_j, x_j), with $x_j = Q\phi_j$, $j = 1, \ldots, \tilde{q}$.

Let

$$r_j^{(k)} = \frac{\theta_j^{(k+1)} - \theta_j^{(k)}}{\theta_j^{(k)} - \theta_j^{(k-1)}}, \quad j = 1, \ldots, q. \tag{4}$$

Since $\lim_{k\to\infty} r_j^{(k)} = (\lambda_j/\lambda_{\tilde{q}+1})^2$, the efficiency of the subspace iteration method depends on $\lambda_q/\lambda_{\tilde{q}+1}$, the ratio of the largest desired eigenvalue λ_q and the eigenvalue $\lambda_{\tilde{q}+1}$ for the pencil (A, B). If this ratio is small, as in inverse iteration with a good shift, only a few iterations will be needed. In this case, the fact that the method is simple, does not require reduction to tridiagonal form, and economizes on data movement from memory, make it an ideal algorithm, especially for high-performance computers. We note that the most operation intensive parts of the algorithm are the first two steps of the RR procedure and the solution of the linear systems. The computation cost for the eigenpairs of H_A can be ignored since $q \ll n$. The RR procedure can be implemented efficiently on a variety of architectures using computationally primitive BLAS [1] [7]. The linear systems involved are banded and positive definite.

For our parallel subspace iteration algorithm, we divide the banded pencil (A, B) into p blocks, with n_s rows per block,

$$(A, B) = \begin{bmatrix} A_1, & B_1 \\ A_2, & B_2 \\ \vdots & \vdots \\ A_p, & B_p \end{bmatrix}, \tag{5}$$

and assign one block to each processor. Our algorithm is outlined as follows.

Parallel Subspace Iteration Algorithm:

- Input: (A, B), tol

- Output: (λ_i, x_i), $i = 1, \ldots, q$, first q eigenpairs of (A, B)
- In parallel, do on all processors P_j, $j = 1, \ldots, p$:

  1. Calculate concurrently a shift s, then shift (A, B) to (A_s, B) with $A_s = A - sB$.
  2. Generate $S_j^{(0)}$, the $j-$th block of a starting matrix $S^{(0)}$, where $S_j^{(0)}$ is an $n_s \times \tilde{q}$ matrix.
  3. For $k = 0, 1, \ldots,$
     - Apply the RR procedure concurrently.
     - Solve $A_s S^{(k+1)} = BS^{(k)}$ by the PPT or PDD algorithm until the computed eigenvalues of (A_s, B) satisfy

$$\frac{|\theta_i^{(k)} - \theta_i^{(k-1)}|}{|\theta_i^{(k)} + s|} < tol, i = 1, \ldots, q.$$

  4. Set $\lambda_i := \theta_i^{(k)} + s$, $x_i := x_i^{(k)}$, $i = 1, \ldots, q$.

The selection of the shift s at Step 1 is critical to the reliability and efficiency of the algorithm. There are, however, several competing requirements placed on the shift s. First, for the convergence and stability of the algorithm, the shift s should satisfy

$$\max_{1 \le j \le q} |\lambda_j - s| < \max_{j' > \tilde{q}} |\lambda'_j - s|,$$

and s should not be an eigenvalue or too close to an eigenvalue of (A, B) [2]. Second, in order to reduce the number of iterations, s should be close to λ_j, $j = 1, \ldots, q$, since the rate of convergence of $\theta_j^{(k)}$ is $(\lambda_j - s)^2/(\lambda_{\tilde{q}+1} - s)^2$. Finally, to obtain the greatest efficiency the shift s should be chosen, whenever possible, so that the PDD algorithm can be used to solve the systems $A_s S^{(i+1)} = BS^{(i)}$, because the PDD algorithm, among all parallel banded solvers, is one of the most efficient parallel algorithms. If it is not possible to find a shift s so that A_s satisfies test (2), then the PPT algorithm must be used to solve this system. Based on these requirements, we use the following process to select a shift s.

Shift Selection Strategy:

1. Approximate the smallest eigenvalue λ_1 of (A, B), then shift (A, B) to (A_{s_1}, B), where $A_{s_1} = A - s_1 B$ with

$$s_1 = \lambda_1 - \delta \quad \text{for a } \delta > 0$$

   and δ is made small compared to λ_1. Since the smallest eigenvalue of (A_{s_1}, B) is δ, the pencil (A_{s_1}, B) continues to be positive definite, but has a small extreme eigenvalue. This guarantees the convergence and stability of the iterations, and also minimizes the number of iterations. Find a positive lower bound for the smallest eigenvalue of A.

2. Compute the off-diagonal blocks of the reduced matrix for A_{s_1} and the timing ratio for the PPT and PDD algorithms

$$r_{time} = \frac{\text{execution time of PPT}}{\text{execution time of PDD}}. \tag{6}$$

   The ratio r_{time} is a function of the matrix parameters n, m, p, and τ_{comp}/τ_{comm}, where p is the number of processors used, and τ_{comp} and τ_{comm} are the speeds for floating point computation and data communication. r_{time} can be estimated from the complexities of the PPT and PDD algorithms given in Section 2 or measured by numerical experiments.

3. If test (2) is satisfied by A_{s_1} or $r_{time} \approx 1$, then set $s := s_1$, and stop.

4. If A passes test (2), apply the Shift Refinement algorithm (below) to (A_{s_1}, B), and $\epsilon = s_1$, and set $s_2 = t$, $s = s_1 - s_2$, and stop. Here t denotes the output of the Shift Refinement algorithm.

5. Apply test (2) to B. If B does not pass test (2), then set $s_2 := 0$, $s := s_1$, and stop.

6. Attempt to compute a positive ζ so that $A + \zeta B$ satisfies test (2). If such an ζ is found, apply the Shift Refinement algorithm to (A, B) and $\epsilon = \zeta$. Set $s_2 := s_1 + t$ and $s := s_1 - s_2$; otherwise set $s_2 = 0$ and $s := s_1$.

In the following, we give the details of Steps 1 and 6.

Computation of s_1:

The inverse iteration:

$$Ax^{(k)} = \tau^{(k-1)} Bx^{(k-1)}$$

$$\lambda_1^{(k)} = \frac{x^{(k)T} Ax^{(k)}}{x^{(k)T} Bx^{(k)}} \qquad k = 1, 2, \ldots,$$

where $\tau^{(k)}$ is a normalization constant, is used to obtain an approximation of λ_1. The iteration $\{(\lambda_1^{(k)}, x^{(k)})\}$ converges to the first eigenpair of (A, B) since the matrix pencil (A, B) is positive definite. Few iterations are needed at this step because only a crude approximation of λ_1 is necessary. In our experiments, the linear systems $Ax^{(k)} = \tau^{(k-1)} Bx^{(k-1)}$ are solved by the Hybrid algorithm introduced in Section 2. We note that matrix factorization is needed only at the first iteration. The iterations are terminated when

$$\frac{|\lambda_1^{(k)} - \lambda_1^{(k-1)}|}{\lambda_1^{(k)}} < 10^{-2}, \tag{7}$$

and s_1 is chosen as $s_1 = 0.95\lambda_1^{(k)}$, which implies that $\delta \approx 0.04\lambda_1$. Similarly, we find a positive lower bound for the smallest eigenvalue of A for use in Step 6. The quantity $\delta(A)$ required in Step 4 of the Shift Selection Strategy is a by-product of the inverse iteration in Step 1.

Computation of s_2:

This shift is introduced only when A_{s_1} has failed to pass test (2) and $r_{time} \gg 1$. This is the situation when the PDD algorithm can not be applied to the systems $A_{s_1} S^{(k+1)} = BS^{(k)}$ and the PPT is much more expensive than the PDD algorithm (most parallel banded solvers [9]

[10] [14] [18] have roughly similar computation and communication complexities as those of PPT). As indicated by the theoretical and experimental results of Section 2, r_{time} increases significantly when the number of processors or the band width increases. If shift s_2 can be found, it can be used to remove the bottleneck of the PPT algorithm. However, while this shift reduces execution time for each single iteration, it increases the total number of iterations at the same time. Whether this shift can and should be performed depends on the eigenproblem (3) and the machine architecture. We now describe Step 6.

For a positive definite m-banded matrix M it is shown in [6] that

$$\left|M^{-1}(i,j)\right| \leq 2\|M^{-1}\|_2\gamma^{|i-j|}, \gamma = \gamma(M) := \left(\frac{\sqrt{r}-1}{\sqrt{r}+1}\right)^{2/m}, r = \frac{\lambda_{max}(M)}{\lambda_{min}(M)}$$

where $\lambda_{max}(M)$ and $\lambda_{min}(M)$ denote the largest and smallest eigenvalues of M. If M_i is a diagonal block of M, then it follows that $\gamma(M_i) \leq \gamma(M)$. Applying test (2) to M, we find that

$$\delta(M) \leq C(\gamma(M))^{dis}$$

where C is a constant which depends on M. We will use the heuristic

$$\delta(A+\zeta B) \approx C(\gamma(A+\zeta B))^{dis}$$

and assume that the constant C does not depend on ζ. We determine it by setting $C = \delta(A)/(\gamma(A))^{dis}$.

We use inverse iteration to find $0 < c_0 \leq \lambda_{min}(B)$ and $0 < d_0 \leq \lambda_{min}(A)$, and we use Gershgorin's Theorem to find $c_1 = \max_i \sum_{j=1}^{n} |b_{ij}| \geq \lambda_{max}(B)$ and $d_1 = \max_i \sum_{j=1}^{n} |a_{ij}| \geq \lambda_{max}(A)$. The lower bound d_0 was found in Step 1. In many cases B has special structure (such as diagonal or diagonally dominant) so that c_0 can be found analytically. We approximate $\gamma(A)$, $\gamma(B)$, and $\gamma(A+\zeta B)$ using $r_A = d_1/d_0$, $r_B = c_1/c_0$, and $r_\zeta = (d_1+\zeta c_1)/(d_0+\zeta c_0)$. These γ's and r's are upper bounds for the exact values. If

$$C(\gamma(B))^{dis} = \delta(A)\left(\frac{\gamma(B)}{\gamma(A)}\right)^{dis} \geq cu,$$

then the heuristic has failed and we set $s_2 := 0$ and $s := s_1$. Otherwise, the heuristic can be solved for ζ by setting $\delta(A+\zeta B) = cu$ and solving for ζ:

$$\zeta = \frac{d_1/c_0 - rd_0/c_0}{r - r_B}, r = \left(\frac{1+\alpha}{1-\alpha}\right)^2, \alpha = \left(\frac{cu}{C}\right)^{m/(2dis)}. \tag{8}$$

Finally, if $A+\zeta B$ does not satisfy test (2), the heuristic fails and we set $s_2 := 0$ and $s := s_1$.

Let A and B be partitioned as in equation (5). The following Shift Refinement Algorithm uses local bisection iteration to approximate the optimal shift s_2.

Shift Refinement Algorithm:

- Input: $Kmax$, and (A, B), $\epsilon > 0$ so that $A + \epsilon B$ satisfies test (2)
- Output: $0 < t \leq \epsilon$ so that $A + tB$ satisfies test (2)

Table 3: Execution Time of the Parallel Subspace Iterations on iPSC/2. ($\alpha = 1.0$)

n	p	T_p (seconds)	K	T_1/T_p	n	p	T_p (seconds)	K	T_1/T_p
1200	1	178.867	5		1200	1	297.537	6	
	2	92.849	5	1.93		2	158.383	6	1.88
($m = 5$)	4	48.798	5	3.67	($m = 10$)	4	83.953	6	3.54
	8	26.467	5	6.76		8	44.328	6	6.71
	16	15.473	5	11.56		16	24.907	6	11.95
3600	1	443.994	4		3600	1	763.193	5	
	2	225.086	4	1.97		2	396.942	5	1.92
($m = 5$)	4	116.258	4	3.82	($m = 10$)	4	213.064	5	3.58
	8	60.248	4	7.37		8	108.022	5	7.07
	16	31.807	4	13.96		16	56.136	5	13.60

Table 4: Execution Time of the Parallel Subspace Iterations on iPSC/860. ($\alpha = 0.1$)

m	n	p	T_p (seconds)		m	n	p	T_p (seconds)	
			PDD	PPT				PDD	PPT
10	3200	1	69.745	69.745	15	6400	2	*	*
		4	20.856	23.090			4	55.472	61.219
		8	10.775	14.110			8	27.725	32.583
		16	5.910	10.744			16	14.342	21.978
		32	4.150	11.988			32	7.906	*
		64	10.432**	*			64	4.679	*

* Exceed memory capacity; ** Shift s_2 is used.

```
• In parallel, do on all processors $P_i$, $i = 1, \ldots, p$:
  1. If the corresponding off-diagonal blocks of $Z$ for $A_i$ passed the test (2),
       set $t_i := 0$;
     else
       initialize $t_i^{(0)} := \epsilon$, $t_i^{(1)} := \epsilon/2$; for $k = 1, \ldots, Kmax$,
         compute corresponding off-diagonal blocks of $Z$ for $A_i + t_i^{(k)} B_i$.
         If the test (2) is satisfied, $t_i^{(k+1)} := t_i^{(k)} - \epsilon/2^{k+1}$;
         otherwise set $t_i^{(k+1)} := t_i^{(k)} + \epsilon/2^{k+1}$.
       Set $t_i := min\{t_i^{(k)} : t_i^{(k)} > t_i^{(Kmax+1)}, \quad k = 0, \ldots, Kmax\}$
     endif
  2. Following a global communication, get $t := max\{t_i, \quad i = 1, \ldots, p\}$.
```

In Tables 3-6, we present the numerical results of the Parallel Subspace Iteration algorithm on iPSC/2 and iPSC/860 multiprocessors. The program was written in Intel Fortran using its communication library. Some Linpack [8] and BLAS [7] Fortran source codes were

Table 5: Execution Time of the Parallel Subspace Iterations ($n/p = 85$, $\alpha = 0.1$)

p	PDD with s_2			PPT without s_2		
	T_p (iPSC/2)	T_p (iPSC/860)	K	T_p (iPSC/2)	T_p (iPSC/860)	K
4	96.610	7.391	24	86.101	6.879	19
8	97.173	7.584	24	112.811	8.899	19
16	97.757	7.729	24	166.732	12.541	19
32		7.954	24		20.489	20

used on the nodes. The testing matrices, except in Table 6, are $A = [a_{ij}]$, $B = I_n$, with

$$a_{ij} = \begin{cases} 1 & \text{if } 0 < |i-j| \le m \\ 2m + \alpha i & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

where α roughly equals the separation of eigenvalues. T_p is the execution time in seconds using p processors which includes the time spent on the shift selection, and K is the number of iterations. The number of required eigenpairs is $q = 10$. The error tolerance tol, except in Table 5, is chosen such that the eigenvalues are approximated to about six-digit accuracy. In Tables 3 and 4, excepting the case marked by **, all testing matrices A_{s_1} passed the test (2). Thus, the PDD algorithm is applied directly without any extra iterations. For comparison, in Table 4, the results of using the PPT algorithm are presented. The efficiencies are much higher when the PDD algorithm is applied, especially for large problem sizes. Table 4 shows that if the number of processors is small, the size of a problem may exceed the memory capacity of the multiprocessors. More processors must be employed. However as the number of processors increases, the order of the resulting reduced system may become too large for a single processor. In this situation the PDD algorithm is the only choice for solving the linear systems involved even though it may cause more iterations. For the experiments presented in Tables 3 and 4, the order n of A and B was held constant. In Table 5 experiments are presented in which the order of the local submatrices is held constant at $\frac{n}{p} = 85$ and the order n of A and B is scaled with the number of processors. Here, the error tolerance was also increased to nine-digit accuracy. The half band width is $m = 10$. These results demonstrate the parallel efficiency of the shifting strategy, especially on a large number of processors. For these problems, as the number of processors increases, the efficiency remains unchanged when s_2 is used. Without it, the performance of the algorithm deteriorates quickly. The parallel scalability afforded by the shifting strategy more than offsets the initial overhead in computing s_2 and the additional number of iterations. This seems to suggest that as long as the shift s_2 is relatively small and introduces a moderate number of iterations, using it with the PDD algorithm should be more efficient than the PPT algorithm, especially for large problems on a large number of processors. When the shift s_2 is too large, it can cause a significant increase in the number of iterations. Whether the shift s_2 should be performed is problem and machine dependent, and an a prior criterion is not available. Fortunately, once the shift s_2 is performed, the efficiency of the shift can be assessed after a few iterations.

Assessing the efficiency of the shift s_2:

Assuming the iteration vector $x_q^{(k)}$ has reached its asymptotic rate of convergence, that

Table 6: Execution Time of the Parallel Subspace Iterations (PPT)

n	p	T_p (seconds)	T_1/T_p	K	$max_{1\leq i\leq q}\frac{\lvert\lambda_i^{(K)}-\lambda_i^{(K-1)}\rvert}{\lvert\lambda_i^{(K)}\rvert}$
640	1	156.258		12	2.3×10^{-6}
	2	78.719	1.99	11	2.0×10^{-6}
	4	40.831	3.83	10	5.1×10^{-6}
	8	25.815	6.05	10	2.7×10^{-7}
	16	16.828	9.29	8	1.4×10^{-6}
3600	1	796.953		11	3.4×10^{-6}
	2	422.973	1.88	11	1.5×10^{-5}
	4	163.940	4.86	8	1.2×10^{-5}
	8	75.556	10.55	7	2.3×10^{-5}
	16	43.429	18.35	7	1.6×10^{-6}

is $r_q^{(k)} \approx (\lambda_q/\lambda_{\bar{q}+1})^2$ (see equation (4)), the convergence rates of the two approaches (PDD with shift s_2 or PPT without shift s_2) can be estimated by $r_{dd} = r_q^{(k)}$ and $r_{pt} = ((\lambda_q^{(k)} - s_2)/(\lambda_q^{(k)}/\sqrt{r_q^{(k)}} - s_2))^2$ respectively. Let $tol = 10^{-d}$. The numbers of iterations for these two approaches are roughly $n_{dd} = -d/logr_{dd}$ and $n_{pt} = -d/logr_{pt}$. The timing ratio of the two approaches is

$$R_{time} = \frac{Time(PPT)\times n_{pt}}{Time(PDD)\times n_{dd}} = r_{time}\times\frac{\log r_{dd}}{\log r_{pt}}.$$

If $R_{time} \ll 1$, the shift s_2 should be abandoned and A_{s_1} used on $X^{(k)}$. In this case, the use of shift s_2 may cause a large increase in the number of iterations.

Finally we present interesting numerical results for an eigenvalue problem of a simply supported beam from structural mechanics. The stiffness matrix A is positive definite with half band width 3, while the mass matrix B is a diagonal matrix with a wide range of diagonal entries. The problem comes from the discretization of a differential system by central finite differences. Consequently, the largest eigenvalue of (A, B) is $O(n^2)$. Since

$$\lambda_{max}(A,B) \leq \lambda_{max}(A)/\lambda_{min}(B) \leq d_1/\lambda_{min}(B),$$

$$\zeta \approx d_1 >= \lambda_{max}(A,B)\cdot\lambda_{min}(B) \approx O(n^2) \quad \text{as } n\to\infty$$

(see equation (8)), the shift s_2 would be very large if it is needed. In this example, matrix A fails test (2) and the shift s_2 is too large, therefore the linear solver PPT is used. Table 6 shows the performance of the algorithm. It is interesting to note that when the number of processors increases, the number of iterations K decreases. Although when using the PPT with a large number of processors, the Parallel subspace iterations algorithm generally has low efficiency, here "super linear" speedups are observed. In this example, smaller subsystems are much better conditioned than larger ones. This may explain the decrease in K with increasing p. The answer awaits further investigation.

4 Remarks and Conclusions

The parallel algorithm proposed here needs little more storage than its sequential version. Most other parallel eigensolvers require that each processor have direct access to the entire matrix pencil (A, B), while in our algorithm, the matrix pencil (A, B) and the iterative vectors $S^{(k)}$ are divided evenly into blocks, which are allocated to corresponding processors. Each processor operates only on its own blocks of (A, B) and $S^{(k)}$ most of the time. All processors solve identical subproblems and communicate the same amount of data at each step of the computation. The load is perfectly balanced. The shift s_2 reduces data references between processors and greatly increases the parallel efficiencies in some situations. When the number of processors is large, secondary memory is usually not necessary even for large problems. Data transfer between different levels of memory can be reduced by employing block matrix computations, such as BLAS [1] [7]. When the number of processors or the band width is large, the size of the reduced system $Zy = h$ can become prohibitive for the PPT algorithm and most other banded solvers. In this situation, shift s_2, with its possibly high number of iterations, seems to be the only alternative. The efficiency of the shift s_2 is machine dependent. Our numerical results suggested that, on iPSC/2 or iPSC/860 hypercube multiprocessors, this shift can lead to higher efficiencies when the ratio of largest to smallest eigenvalues of the pencil is $O(n)$, but it is detrimental when this ratio is $O(n^2)$ or greater, which is typically the case with problems derived from differential equations. However, in this case, the numerical experiments of Table 6 indicate that the PPT version of our algorithm exhibits "superlinear" convergence. This may occur because smaller submatrices are much better conditioned than larger ones.

Many acceleration schemes have been applied to the basic sequential subspace iteration method [3] [26], making it efficient for a wide variety of applications. Most of these accelerating schemes can be easily incorporated into our parallel algorithm.

References

[1] E. Anderson et al., *Lapack Users' Guide*, Society for Industrial and Applied Mathematics, 1992.

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

[3] K. J. Bathe and S. Ramaswamy, *An accelerated subspace iteration method*, J. Computer Methods in Applied Mechanics and Engineering, 23 (1980), pp. 313–331.

[4] M. Clint and A. Jennings, *The evaluation of eigenvalues and eigenvectors of a real symmetric matrix by simultaneous iteration*, Comput. J., 13 (1970), pp. 76–80.

[5] C. R. Crawford, *Reduction of a band-symmetric generalized eigenvalue problem*, Comm. ACM, 16 (1973), pp. 41–44.

[6] S. Demko, W. F. Moss, and P. W. Smith, *Decay rates for inverses of band matrices*, Mathematics of Computation, 43 (1984), pp. 491–499.

[7] J. DEMMEL, J. DONGARRA, J. DUCROZ, A. GREENBAUM, S. HAMMARLING, AND D. SORENSEN, *Prospectus for the development of a linear algebra library for high-performance computers*, Tech. Rep. TM-97, Mathematics and Computer Science Div., Argonne National Laboratory, Argonne, IL, (1987).

[8] J. J. DONGARRA ET AL., *Linpack Users' Guide*, Society for Industrial and Applied Mathematics, 1979.

[9] J. J. DONGARRA AND L. JOHNSSON, *Solving banded systems on a parallel processor*, Parallel Comput., 5 (1987), pp. 219–246.

[10] J. J. DONGARRA AND A. SAMEH, *On some parallel banded system solvers*, Parallel Comput., 1 (1984), pp. 223–235.

[11] J. J. DONGARRA AND D. C. SORENSEN, *A fully parallel algorithm for the symmetric eigenvalue problem*, SIAM J. Sci. Stat. Comput., 8 (1987), pp. 139–154.

[12] G. H. GOLUB AND D. P. O'LEARY, *Some history of the conjugate gradient and Lanczos algorithms: 1948-1976*, Siam Review, 11 (1989), pp. 50–100.

[13] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, The Johns Hopkins University Press, Baltimore, 1983.

[14] C. T. HO AND S. L. JOHNSON, *Optimizing tridiagonal solvers for alternating direction methods on boolean cube multiprocessors*, SIAM J. Sci. Stat. Comput., 11 (1990), pp. 563–592.

[15] M. T. JONES AND M. L. PATRICK, *The use of Lanczos's method to solve the large generalized symmetric definite eigenvalue problem*, Report no. 89-69, Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, VA, (1989).

[16] M. T. JONES AND P. E. PLASSMANN, *Scalable iterative solution of sparse linear systems*, Technical Report MCS-P277-1191, Mathematics and Computer Science Division, Argonne National Laboratory, (1991).

[17] L. KAUFMAN, *An algorithm for the banded symmetric generalized matrix eigenvalue problem*, SIAM J. Matrix Anal. Appl., 14 (1993).

[18] D. H. LAWRIE AND A. H. SAMEH, *The computation and communication complexity of a parallel banded system solver*, ACM Transactions of Mathematical Software, 10 (1984), pp. 185–195.

[19] T. Y. LI, H. ZHANG, AND X. H. SUN, *Parallel homotopy algorithm for symmetric tridiagonal eigenvalue problems*, SIAM J. Sci. Stat. Comput., 12 (1991).

[20] S. S. LO, B. PHILIPPE, AND A. SAMEH, *A multiprocessor algorithm for the symmetric tridiagonal eigenvalue problem*, SIAM J. Sci. Stat. Comput., 8 (1987), pp. 155–165.

[21] B. Nour-Omid, B. N. Parlett, and R. L. Taylor, *Lanczos versus subspace iteration for solution of eigenvalue problems*, International Journal for Numerical Methods in Engineering, 19 (1983), pp. 859–871.

[22] R. D. Pantazis and D. B. Szyld, *A multiprocessor method for the solution of the generalized eigenvalue problem on an interval*, Proceedings of the Fourth SIAM Conference on Parallel Processing for Scientific Computing, (1990), pp. 36–41.

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

[24] C. J. Ribbens and C. Beattie, *Parallel solution of a generalized symmetric eigenvalue problem*, Proceedings of the Fifth SIAM Conference on Parallel Processing for Scientific Computing, (1991), pp. 16–21.

[25] H. Rutishauser, *Simultaneous iteration method for symmetric matrices*, Numer. Math., 16 (1970), pp. 205–223.

[26] G. W. Stewart, *Accelerating the orthogonal iteration for the eigenvalues of a hermitian matrix*, Numer. Math., 13 (1969), pp. 362–376.

[27] X. Sun, H. Zhang, and L. M. Ni, *Efficient tridiagonal solvers on multicomputers*, IEEE Trans. on Compu., 41 (1992), pp. 286–296.

[28] S. Wang and S. Zhao, *An algorithm for $Ax = \lambda Bx$ with symmetric and positive definite a and b*, SIAM J. Matrix Anal. Appl., (1993).

[29] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Clarendon Press, Oxford, England, 1965.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE September 1993	3. REPORT TYPE AND DATES COVERED Contractor Report

4. TITLE AND SUBTITLE USING PARALLEL BANDED LINEAR SYSTEM SOLVERS IN GENERALIZED EIGENVALUE PROBLEMS	5. FUNDING NUMBERS C NAS1-19480 WU 505-90-52-01
6. AUTHOR(S) Hong Zhang William F. Moss	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Institute for Computer Applications in Science and Engineering Mail Stop 132C, NASA Langley Research Center Hampton, VA 23681-0001	8. PERFORMING ORGANIZATION REPORT NUMBER ICASE Report No. 93-71
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration Langley Research Center Hampton, VA 23681-0001	10. SPONSORING/MONITORING AGENCY REPORT NUMBER NASA CR-191540 ICASE Report No. 93-71

11. SUPPLEMENTARY NOTES
Langley Technical Monitor: Michael F. Card
Final Report
Submitted to Journal of Parallel Computing

12a. DISTRIBUTION/AVAILABILITY STATEMENT Unclassified - Unlimited Subject Category 61	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*
Subspace iteration is a reliable and cost effective method for solving positive definite banded symmetric generalized eigenproblems, especially in the case of large scale problems. This paper discusses an algorithm that makes use of two parallel banded solvers in subspace iteration. A shift is introduced to decompose the banded linear systems into relatively independent subsystems and to accelerate the iterations. With this shift, an eigenproblem is mapped efficiently into the memories of a multiprocessor and a high speed-up is obtained for parallel implementations. An optimal shift is a shift that balances total computation and communication costs. Under certain conditions, we show how to estimate an optimal shift analytically using the decay rate for the inverse of a banded matrix, and how to improve this estimate. Computational results on iPSC/2 and iPSC/860 multiprocessors are presented.

14. SUBJECT TERMS generalized eigenvalue; banded matrix; parallel algorithm	15. NUMBER OF PAGES 19
	16. PRICE CODE A03

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT