A Practical Parallel Algorithm for Solving Band Symmetric Positive Definite Systems of Linear Equations

ILAN BAR-ON
Courant Institute of Mathematical Sciences

We give a practical parallel algorithm for solving band symmetric positive definite systems of linear equations in $O(m * \log n)$ time using $nm/\log n$ processors. Here n denotes the system size and m its bandwidth. Hence, the algorithm is efficient. For tridiagonal systems, the algorithm runs in $O(\log n)$ time using $n/\log n$ processors. Furthermore, an improved version runs in $O(\log m \log n)$ time using $nm^2/(\log m \log n)$ processors.

Categories and Subject Descriptors: G.1.0 [**Numerical Analysis**]: General—*parallel algorithms*; G.1.3 [**Numerical Analysis**]: Numerical Linear Algebra—*linear systems, matrix inversion*; G.4 [**Mathematics of Computing**]: Mathematical Software—*algorithm analysis, efficiency*

General Terms: Algorithms, Measurement, Performance

Additional Key Words and Phrases: Band Systems of linear equations, Gaussian elimination, parallel algorithms

1. INTRODUCTION

Sparse symmetric positive definite systems of linear equations arise in many important applications such as the finite element method, see [22]. These systems are very large and sparse since they result from the discretization of continuous problems. Furthermore, they can be brought to an almost narrow banded structure, see [6], and hence the importance of an efficient parallel algorithm.

In this paper, we give a practical parallel algorithm for solving band symmetric positive definite systems of linear equations that runs in $O(m \log n)$ time using $nm/\log n$ processors. Here, n denotes the system size and m its bandwidth. Our algorithm improves many suggested partitioning algorithms such as those of Lawrie and Sameh [12], Dongarra and Sameh [3], and Meier [13] by a factor of $\text{sqrt}(n/m)/\log n$. Our algorithm is parallel, oriented through all stages, whereas the above algorithms are parallel in their first stages only. For other parallel-oriented algorithms, see Bar-On and Vishkin [1], and Galil [5]. Moreover, an improved version of the algorithm presented in Section 3 runs in $O(\log n \log m)$

This research was supported in part by the Tenesys Design System Limited, Haifa, Israel.
The author's current address is: P.O. Box 07735, Ahuza, Haifa, Israel.

time using $nm^2/(\log n \log m)$ processors; for as good as the currently known lower time bound algorithms for direct methods, see Csanky [2], and Eberly [4]. The present algorithm competes with the odd–even reduction algorithm first formulated by Hockney [8]; see also [9, 16, 18] and [10]. Furthermore, it improves the running time of the Pan and Reif algorithm [15] by a factor of $\log m$.

In our model of parallel computation, all processors have access to a common memory. Simultaneous reading from the same location is allowed, but simultaneous writing is not. This model is sometimes called the concurrent-read exclusive-write parallel random access machine, see [21].

2. PARALLEL SOLUTION OF BAND SYMMETRIC POSITIVE DEFINITE SYSTEMS OF LINEAR EQUATIONS

Let $Ax = b$ be a system of linear equations where A_{nn} is a band symmetric positive definite (s.p.d.) matrix, that is,

$$A = \begin{pmatrix} a_{1,1} & \cdots & a_{1,m+1} & & & & \\ a_{m,1} & & \ddots & a_{m,2m} & & \varnothing & \\ & \ddots & & & \ddots & & \ddots \\ & \varnothing & & a_{n-m,n-2m} & \cdots & & a_{n-m,n} \\ & & & & a_{n,n-m} & \cdots & a_{n,n} \end{pmatrix}.$$

Using block structure notation we get:

$$A = \begin{pmatrix} C_1 & L_1 & & & \\ U_1 & C_2 & L_2 & & \varnothing \\ & \ddots & \ddots & \ddots & \\ & \varnothing & U_{q-2} & C_{q-1} & L_{q-1} \\ & & & U_{q-1} & C_q \end{pmatrix} \qquad \begin{aligned} U_i &= L_i^t \\ q &= n/m \end{aligned}$$

where we assume for simplicity that m divides n. Here, A_i, L_i, U_i are square submatrices of order m, and L_i, U_i are lower and upper triangular, respectively.

We regard A as a tridiagonal matrix with elements being submatrices of order m, so that, by row i, we mean the $m \times n$ submatrix:

$$(0\ U_{i-1}\ C_i\ L_i\ 0).$$

We ignore the vector b in the computation that follows. Suffice to say that all operations applied to the rows of A should be applied to the rows of b.

2.1 The Parallel Elimination Procedure

Let $p = n/4m$ be the number of processors. We assign processor i, $i = 1, \ldots, p$ to the block structured submatrix,

$$T_i = \begin{pmatrix} Ur_i - 1 & Cr_i & Lr_i & & & \\ & Ur_i & Cr_i + 1 & Lr_i + 1 & & \\ & & Ur_i + 1 & Cr_i + 2 & Lr_i + 2 & \\ & & & Ur_i + 2 & Cr_i + 3 & Lr_i + 3 \end{pmatrix} \qquad \begin{aligned} i &= 1, \ldots, p \\ r_i &= 4(i-1) + 1 \end{aligned}$$

$$= (0\ R_i T_i^d Q_i\ 0)$$

Here, T_i^d is a $4m \times 4m$ s.p.d. submatrix, R_i, Q_i submatrices of order $4m \times m$, and we assume for simplicity that $p = 2^t$, t some positive integer.

Step 0. Each processor diagonalizes its corresponding T_i^d principal submatrix by Gaussian elimination (see Figure 1):

$$T_i^0 = \begin{pmatrix} 0 & E_i^0 & V_i^0 & 0 & 0 & 0 & F_i^0 & 0 \\ & * & 0 & * & 0 & 0 & * & \\ & * & 0 & 0 & * & 0 & * & \\ 0 & G_i^0 & 0 & 0 & 0 & W_i^0 & H_i^0 & 0 \end{pmatrix} \qquad i = 1, \ldots, p$$

where E_i^0, F_i^0, G_i^0, H_i^0 and "$*$" denote some matrices of order $m \times m$.

Step $k = 1 \ldots t = \log p$. We define

$$T_i^k = \begin{vmatrix} T_{2i-1}^{k-1} \\ T_{2i}^{k-1} \end{vmatrix} \qquad i = 1, \ldots, p/2^k.$$

$$T_i^k = \begin{pmatrix} E_{2i-1}^{k-1} & V_{2i-1}^{k-1} & & 0 \ldots 0 & F_{2i-1}^{k-1} & & \\ & & * & & & & \\ G_{2i-1}^{k-1} & 0 \ldots 0 & & W_{2i-1}^{k-1} & H_{2i-1}^{k-1} & & \\ & & & E_{2i}^{k-1} & V_{2i}^{k-1} & 0 \ldots 0 & F_{2i}^{k-1} \\ & & & & & * & \\ & & & G_{2i}^{k-1} & 0 \ldots 0 & W_{2i}^{k-1} & H_{2i}^{k-1} \end{pmatrix}.$$

We eliminate E_{2i}^{k-1}, H_{2i-1}^{k-1} in parallel, as follows. We subtract $E_{2i}^{k-1} * (W_{2i-1}^{k-1})^{-1}$ times the last row of T_{2i-1}^{k-1} from the first row of T_{2i}^{k-1}, and at the same time we subtract $H_{2i-1}^{k-1} * (V_{2i}^{k-1})^{-1}$ times the first row of T_{2i}^{k-1} from the last row of T_{2i-1}^{k-1}. By *parallel* we mean that the processors of the middle rows perform the above operations, respectively,

$$p_{2i-1}^{k-1} = 2^k(i-1) + 2^{k-1}, \qquad p_{2i}^{k-1} = 2^k(i-1) + 2^{k-1} + 1$$

using the elements values of step $(k-1)$. Clearly, some elements should be copied before the computation starts. However, the complexity of this operation is negligible as compared to the other operations, so we ignore it from now on. As a result, we get the following submatrices:

$$T_i^k = \begin{pmatrix} * & V_{2i-1}^{k-1} & 0 \ldots 0 & F_{2i-1}^{k-1} & & & \\ * & 0 \ldots 0 & R_{2i-1}^{k-1} & 0 & 0 \ldots 0 & * & 0 \\ * & 0 \ldots 0 & 0 & S_{2i}^{k-1} & 0 \ldots 0 & * & 0 \\ & & G_{2i}^{k-1} & 0 \ldots 0 & W_{2i}^{k-1} & * & 0 \end{pmatrix}.$$

We now subtract, in parallel, $G_{2i}^{k-1} * (R_{2i-1}^{k-1})^{-1}$ times the last row of T_{2i-1}^{k-1} from the last row of T_{2i}^{k-1}, and $F_{2i-1}^{k-1} * (S_{2i}^{k-1})^{-1}$ times the first row of T_{2i}^{k-1} from the first row of T_{2i-1}^{k-1}, using the same processors, respectively, we get (see Figure 2):

$$T_i^k = \begin{pmatrix} E_i^k & V_i^k & & 0 \ldots 0 & F_i^k \\ & & * & & \\ G_i^k & 0 \ldots 0 & & W_i^k & H_i^k \end{pmatrix}$$

of the same form as the submatrices of the previous step. Clearly, in each step the number of submatrices is halved, so after $\log p$ steps there remains one submatrix of the above form.

Correctness. Clearly, V_i^0, W_i^0 are positive definite, and hence V_{2i}^{k-1}, W_{2i-1}^{k-1}, since they are, respectively, the same.

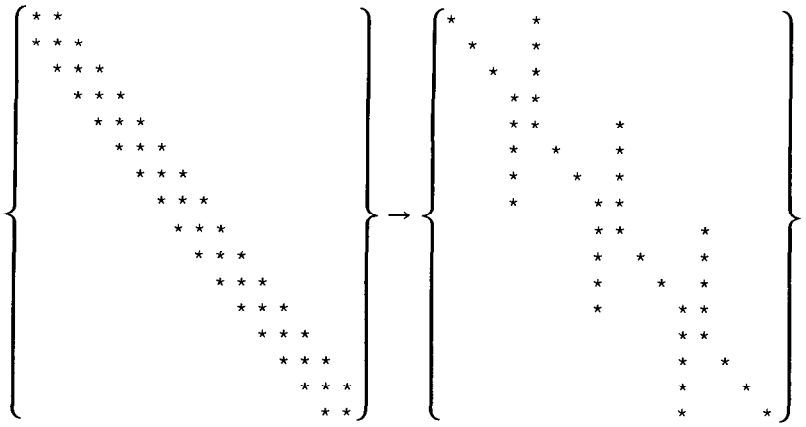

Fig. 1. Elimination step 0, with $n = 16$, $m = 1$.

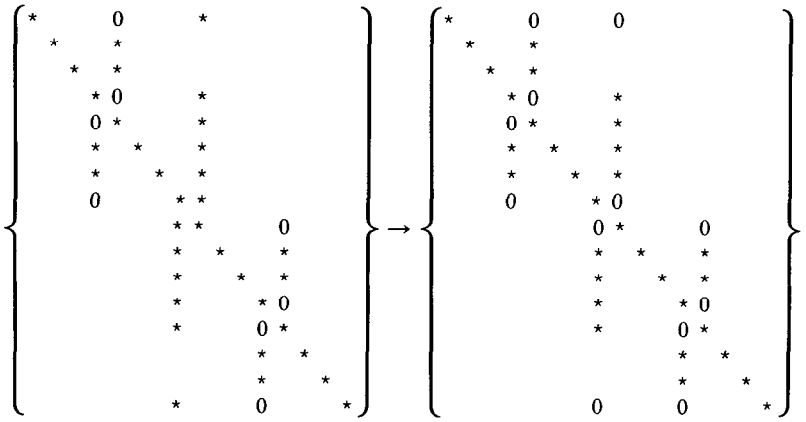

Fig. 2. Elimination steps 1, 2.

Suppose w.l.g. S_{2i}^{k-1} becomes singular. In T_i^k, the row of S_{2i}^{k-1} contains zero in all columns corresponding to rows of T_i^k beside the diagonal element itself. Let P denote the nonsingular matrix corresponding to the operations done on the rows of A, then PAP^t is positive definite. Since no operation is applied to rows of T_i^k and to rows outside it, the corresponding column's operations applied to the row of S_{2i}^{k-1} subtract multiples of S_{2i}^{k-1}. Hence, the diagonal element becomes: $S_{2i}^{k-1}X$, where X is some matrix of order m. Clearly, S_{2i}^{k-1} can not be singular.

2.2 The Parallel Substitution Procedure

By the end of the elimination algorithm we get a new system, $T_1^t x = c$, where

$$T_1^t = \begin{pmatrix} V_1^1 & & 0 \ldots 0 & & \\ 0 \ldots 0 & R_1^1 & 0 & 0 \ldots 0 & \\ 0 \ldots 0 & 0 & S_2^1 & 0 \ldots 0 & \\ & 0 \ldots 0 & & & W_2^1 \end{pmatrix} \qquad 1 = t - 1.$$

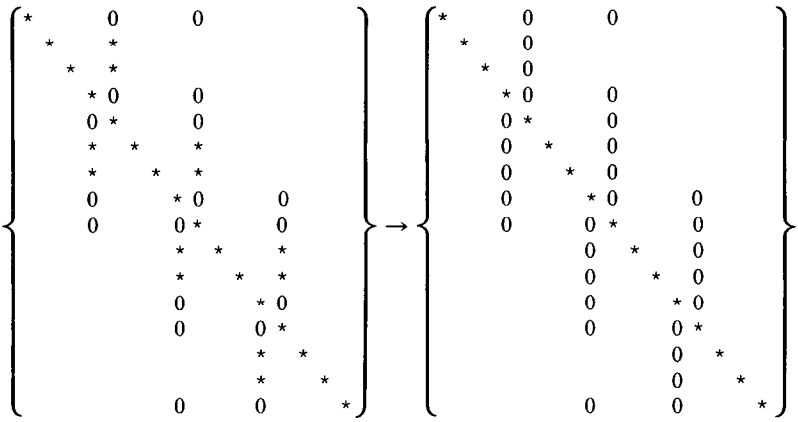

Fig. 3. Substitution steps 1, 2.

We can therefore obtain the following solutions:

$$x_1 = (V_1^1)^{-1} * c_1 \qquad x_{q/2} = (R_1^1)^{-1} * c_{q/2}$$
$$x_{q/2+1} = (S_2^1)^{-1} * c_{q/2+1} \qquad x_q = (W_2^1)^{-1} * c_q$$

where x_i, c_i denote vectors of length m, and $q = n/m$. Clearly, processor 1 finds the solution for x_1, processor $p/2$ the solution for $q/2$, and so on. We now proceed to substitute these solutions to get new ones.

Step k, $k = 1, \ldots, t = \log p$. For $i = 1, \ldots, 2^k - 1$, we substitute

$$x_{iq/2^k+1} \quad \text{in rows} \quad (2i-1)q/2^{k+1}, \quad (2i-1)q/2^{k+1} + 1$$
$$x_{iq/2^k} \quad \text{in rows} \quad (2i+1)q/2^{k+1}, \quad (2i+1)q/2^{k+1} + 1.$$

We then obtain the solutions (see Figure 3):

$$x_{q/2^{k+1}} = (R_1^{1-k})^{-1} * c_{q/2^{k+1}}.$$

Again, the substitution in the above rows and the computation of the corresponding solutions are done by the respective processors. Clearly, by the end of step t, we get the complete solution x.

Complexity of the Algorithm. The complexity of the elimination procedure is dominated by the time to invert a $m \times m$ matrix. We invert a positive definite $m \times m$ matrix in time $O(m)$ using m^2 processors (see, [11, 16] and [23]). We invert a general nonsingular matrix by Gaussian elimination with partial pivoting [7]. Here, we take as pivot the maximum element (in absolute value) among at most m elements, and, using m^2 processors, it takes $O(1)$ time, (see, [17, 20]. Hence, the total complexity remains $O(m)$ time using m^2 processors.

Let $P = nm/4 = (n/4m) * m^2$ be the number of processors. We now assume that each of our $n/4m$ original processors consists of m^2 processors capable of multiplying and inverting a matrix of order m in $O(m)$ time. Clearly, each step of the elimination procedure takes $O(m)$ time for a total of $O(m \log n)$ time. In the substitution procedure all inverses are already known, and we essentially compute matrix vector multiplications of order m, each taking $O(\log m)$ time

with m^2 processors. Since there are at most three nonzero entries in each row, and $\log p \leq O(\log n)$ steps, the substitution procedure takes $O(\log m \log n)$ time. The total complexity of the algorithm is therefore $O(m \log n)$ time using $P = nm/4$ processors. However, we can improve the efficiency as follows.

We divide the rows of A into submatrices of $m * \log n$ consecutive rows each, that is,

$$T_i = \begin{pmatrix} Ur_i - 1 & Cr_i & Lr_i \\ & \searrow & \searrow \\ & Us_i - 1 & Cs_i \quad Ls_i \end{pmatrix} \qquad \begin{aligned} r_i &= (i-1) * \log n + 1 \\ s_i &= i * \log n \end{aligned}$$

$$= (0 * T_i^d * 0),$$

and we assign m^2 processors to each such submatrix, for a total of $nm/\log n$. We then diagonalize the principal submatrix, T_i^d, arriving at the submatrices:

$$T_i^0 = \begin{pmatrix} 0 \ldots 0 & E_i^0 & V_i^0 & & 0 \ldots 0 & F_i^0 & 0 \ldots 0 \\ & | & & & & | & \\ 0 & * & & * & & * & 0 \\ & | & & & & | & \\ 0 \ldots 0 & G_i^0 & 0 \ldots 0 & & W_i^0 & H_i^0 & 0 \ldots 0 \end{pmatrix},$$

applying a method such as Gaussian elimination. Clearly, V_i^0, W_i^0 are positive definite, and we proceed as before, but for a final step, where we compute the above inbetween solutions. The complexity is therefore the same, but for an additional $O(m \log n + \log n \log m)$ time. Hence,

$$O(nm^2/p) \text{ time}, \qquad \text{for } p \leq nm/\log n \text{ processors},$$

and the algorithm is efficient, that is,

$$E_p = (T_1/T_p)/p = O((nm^2/(nm^2/p))/p) = O(1).$$

3. AN $O(\log m \log n)$ EFFICIENT PARALLEL ALGORITHM

We give, respectively, efficient $O(\log^2 n)$, $O(\log n \log m)$ time-parallel algorithms for inverting a symmetric positive definite matrix of order n, and solving a band s.p.d. system of linear equations.

Inverting a Symmetric Positive Definite Matrix. Let A_{nn} be a symmetric positive definite matrix, that is,

$$A = \begin{pmatrix} C & Q \\ Q^t & D \end{pmatrix} \qquad C, D, Q \text{ are of order } (n/2) \times (n/2),$$

where we assume for simplicity that $n = 2^s$ for some integer $s > 0$. Since A is s.p.d., so are C, D ([19]). Let us eliminate Q, Q^t by applying the following nonsingular matrix P to A, that is,

$$A' = PA = \begin{pmatrix} I & -QD^{-1} \\ -Q^tC^{-1} & I \end{pmatrix} * \begin{pmatrix} C & Q \\ Q^t & D \end{pmatrix} = \begin{pmatrix} C' & 0 \\ 0 & D' \end{pmatrix},$$

where $C' = C - QD^{-1}Q^t$ and $D' = D - Q^tC^{-1}Q$ are again s.p.d. Clearly, the inverse of A is given by

$$A^{-1} = A'^{-1}P = \begin{pmatrix} C'^{-1} & 0 \\ 0 & D'^{-1} \end{pmatrix} * \begin{pmatrix} I & -QD^{-1} \\ -Q^tC^{-1} & I \end{pmatrix}.$$

Given C^{-1}, D^{-1}, we compute D'^{-1} (similarly, C'^{-1}) as follows:

$$\begin{aligned} D'^{-1} &= (D - Q^tC^{-1}Q)^{-1} = (I - D^{-1}Q^tC^{-1}Q)^{-1}D^{-1} \\ &= (I - X)^{-1}D^{-1} \\ X &= D^{-1}Q^tC^{-1}Q = D^{-1/2}(D^{-1/2}Q^tC^{-1}QD^{-1/2})D^{1/2} \\ &= G^{-1}(R^tC^{-1}R)G \qquad G = D^{1/2},\ R = QD^{-1/2}. \end{aligned}$$

Hence, X is similar to a nonnegative symmetric matrix, and therefore all its eigenvalues are nonnegative. Furthermore, the largest eigenvalue of X is less than unity, otherwise:

$$Xz = \lambda z \quad \lambda \geq 1 \qquad \text{for some vector} \quad z\langle\rangle 0$$

$$\langle D'z, z\rangle = \langle D(I - X)z, z\rangle = (1 - \lambda)\langle Dz, z\rangle \leq 0$$

and D' is not positive definite, a contradiction. Therefore, $X^k \to 0$ as $k \to \infty$, and the following series converges (see [14]),

$$(I - X)^{-1} = I + X + X^2 + X^3 + \ldots = \sum_{k=0}^{\infty} X^k = S_M + R_M$$

$$S_M = \sum_{k=0}^{M-1} X^k \qquad\qquad R_M = \sum_{k=M}^{\infty} X^k$$

Furthermore, let $N = R^tC^{-1}R$, then,

$$\| N \| = q < 1,\ \| X^k \| = \| G^{-1}N^kG \| \leq K(G) * q^k,$$

where $\| * \|$ denotes the spectral norm, and $K(G)$ the condition number of G, that is, $\| G \| * \| G^{-1} \|$. Then,

$$\| (I - X)^{-1} - S_M \| = \| R_M \| \leq \sum_{i=M}^{\infty} \| X \|^k \leq K(G) \frac{q^M}{1 - q}.$$

Hence, S_M is a good approximation for M large enough. For example, if $q \leq \frac{1}{2}$ and $c = K(G)$, then

$$\| (I - X)^{-1} - S_M \| \leq 2c * (\tfrac{1}{2})^M,$$

and $M = 2t$ is a good choice when c is not too large, t being the machine precision. Moreover,

$$S_M = \prod_{j=0}^{\log M/2} (I + X^{2**j}), \qquad \prod \text{ denotes the product of the factors}$$

and S_M is computed in $O(\log M * \log n) = O(\log n)$ time.

Applying the algorithm recursively, we finally compute the inverse of s.p.d. submatrices of order less than or equal to $\log n$, which we do directly by Cholesky decomposition [11]. We then proceed as above.

Let $P = n^3/\log^2 n$ be the number of processors, then the last step takes $O((n/2)^3/(P/2)) = O(\log^2 n/4)$ time. The step before the last takes $O((n/4)^3/(P/2^2)) = O(\log^2 n/4^2)$, and so on. In general, the step before the last takes $O(\max\{(n/2^{i+1})^3/(P/2^{i+1}), \log n/2^{i+1}\}) \leq O(\max\{\log^2 n/4^{i+1}, \log n\})$, and clearly the first step of inverting the above submatrices of order $\log n$ takes $O(\log n)$ time. Hence, the total complexity for inverting an s.p.d. matrix of order n is

$$O(n^3/p) \text{ time}, \qquad \text{for } p \leq n^3/\log^2 n \text{ processors},$$

and the algorithm is efficient.

Consider the submatrix

$$\begin{pmatrix} 0 & * & 0 \ldots 0 & W_{2i-1}^{k-1} & H_{2i-1}^{k-1} & & 0 \ldots 0 & \\ 0 \ldots 0 & & & E_{2i}^{k-1} & V_{2i}^{k-1} & 0 \ldots 0 & * & 0 \end{pmatrix}$$

of Section 2.1, which we denote for simplicity by

$$\begin{pmatrix} * & 0 \ldots 0 & W & H & & 0 \ldots 0 & \\ 0 \ldots 0 & & E & V & 0 \ldots 0 & * & \end{pmatrix}.$$

Eliminating H, E, we get the submatrix

$$\begin{pmatrix} 0 * 0 \ldots 0 R 0 0 \ldots 0 * 0 \\ 0 * 0 \ldots 0 0 S 0 \ldots 0 * 0 \end{pmatrix}$$

where R, S denote the corresponding submatrices, as above. We now apply to the columns of A the same operations done to its rows, excluding the last one corresponding to the elimination of H, E. We then get the submatrix

$$\begin{pmatrix} 0 & * & \ldots & * & R' & 0 & 0 & \ldots & 0 & * & 0 \\ 0 & * & 0 & \ldots & 0 & 0 & S' & * & \ldots & * & 0 \end{pmatrix}.$$

Applying the last column operation, we get the principal submatrix

$$\begin{pmatrix} R' & 0 \\ 0 & S' \end{pmatrix} \begin{pmatrix} I & -W^{-1}E^t \\ -V^{-1}H^t & I \end{pmatrix} = \begin{pmatrix} R' & -R'W^{-1}E^t \\ -S'V^{-1}H^t & S' \end{pmatrix},$$

which is symmetric and positive definite—since it is the same principal submatrix as that of PAP^t, where P denotes the elementary operations done on the rows of A. From the preceding discussion of s.p.d. matrices, we see that the eigenvalues of

$$S'^{-1}(-S'V^{-1}H^t)R'^{-1}(-R'W^{-1}E^t)$$

are all real nonnegative numbers less than 1. Hence,

$$\lambda(EW^{-1}HV^{-1}) = \lambda(V^{-1}H^tW^{-1}E^t) < 1,$$

where $\lambda(*)$ denotes the largest eigenvalue in absolute value.

$$S^{-1} = (V - EW^{-1}H)^{-1} = V^{-1}(I - (EW^{-1}HV^{-1}))^{-1}$$
$$= V^{-1}(I - X)^{-1} \qquad \lambda(X) < 1,$$

and the inverse of S can be computed as before.

Let $P = nm^2/(\log m \log n)$ be the number of processors, then the elimination procedure takes

$$\sum_{k=0}^{\log n/4m} O(\max\{(m^3/(P * 2^k/4(n/m))), \log m\}) = O(\log m \log n) \text{ time},$$

and similarly the substitution procedure takes

$$\sum_{k=1}^{\log n/4m} O(\max\{(m^2/(P/2^{k+1})), \log m\}) = O(\log m \log n) \text{ time}.$$

Therefore, the complexity of the algorithm for band s.p.d. systems is

$$O(nm^2/p) \text{ time} \qquad \text{for} \quad p \leq nm^2/(\log m \log n) \text{ processors},$$

and it is efficient.

4. CONCLUSION

We have presented a practical parallel algorithm for solving band symmetric positive definite systems of linear equations faster than the sequential algorithm by a factor of $nm/\log n$. Moreover, we have shown that with enough processors its running time is the best currently known lower time bound for direct methods.

Our algorithm competes with the previously known odd–even reduction algorithm, which has the same complexity. However, there are some differences that deserve attention: In the odd–even algorithm the diagonal submatrices are modified repeatedly, whereas in our algorithm the off-diagonal ones are. Whether this has any numerical significance requires more theoretical and experimental analysis. Furthermore, the odd–even algorithm has only one variance corresponding to the maximum number of processors $p = n/m$, whereas our algorithm can be adjusted to any number of processors.

We have made some comparison tests with the sequential Gaussian elimination algorithm. For relatively small matrices $n \leq 500$ and randomly chosen data, the results were as good as the sequential method. However, a systematic analysis of the algorithm's numerical properties is beyond the scope of this paper.

REFERENCES

1. BAR-ON, I., AND VISHKIN, U. Optimal parallel generation of a computation tree form. *ACM Trans. Program. Lang. Syst. 7* (Apr. 1985), 348–357.
2. CSANKY, L. Fast parallel matrix inversion algorithms. *SIAM J. Comput. 5*, 4 (Dec. 1976), 618–623.
3. DONGARRA, J. J., AND SAMEH, A. H. On some parallel banded systems solvers. *Parallel Comput. 1* (1984), 223–235.
4. EBERLY, W. Very fast parallel matrix and polynomial arithmetic. In *Proceedings of the 25th FOCS Conference* (1984), 21–30.
5. GALIL, Z. Optimal parallel algorithms for string matching. In *Proceedings of the 16th Annual Symposium on the Theory of Computing* (Apr. 1984), 240–248.
6. GEORGE, J. A., AND LIU, J. W. *Computer Solution of Large Sparse Positive Definite Systems.* Prentice-Hall, Englewood Cliffs, N.J., 1981.
7. GOLUB, G. H., AND VAN LOAN, C. F. *Matrix Computations.* Johns Hopkins University Press, Baltimore, Md., 1983.

8. HOCKNEY, R. W. A fast direct solution of Poisson's equation using Fourier analysis. *J. ACM 12* (1965), 95–113.
9. HELLER, D. A survey of parallel algorithms in numerical linear algebra. *SIAM Rev. 20*, 4 (Oct. 1978), 740–777.
10. HOSHINO, T., KAMIMURA, T., IIDA, T., AND SHIRAKAWA, T. Parallelized ADI scheme using GECR (Gauss-Elimination-Cyclic-Reduction) method and implementation of Navier–Stokes equation in the PAX computer. In *Proceedings of the 26th FOCS Conference* (1985), 426–433.
11. KUMAR, S. P., AND KOWALICK, J. S. Parallel factorization of a positive definite matrix on an MIMD computer. *Proceedings of the 1984 ICPP* (1984), 410–416.
12. LAWRIE, H. D., SAMEH, H. A. Complexity of a parallel banded system solver. *ACM Trans. Math. Softw. 10*, 2 (June 1984), 184–195.
13. MEIER, U. A parallel partition method for solving banded systems of linear equations. *Parallel Comput.* 2 (1985), 33–43.
14. ORTEGA, J. M. *Numerical Analysis, A Second Course.* Academic Press, New York, 1972.
15. PAN, V., AND REIF, J. Efficient parallel solution of linear systems. In *Proceedings of the 26th FOCS Conference*, (1985), 143–152.
16. SCHENDEL, U. *Introduction to Numerical Methods for Parallel Computers.* Ellis Horwood Limited, 1984.
17. SHILOACH, Y., AND VISHKIN, U. Finding the maximum merging and sorting in parallel computation model. *J. Algorithms 2*, 1 (1981), 88–102.
18. STONE, H. An efficient parallel algorithm for the solutions of a tridiagonal linear system of equations. *J. ACM 20*, 1 (Jan. 1973), 27–38.
19. STRANG, G. *Linear Algebra and Its Applications.* Academic Press, New York, 1980.
20. VALIANT, L. G. Parallelism in comparison problems. *SIAM J. Comput. 4*, 3 (Sept. 1975), 348–355.
21. VISHKIN, U. Synchronous parallel computation—a survey. TR-71, Dept. of Computer Science, Courant Institute, New York, Univ., 1983.
22. WEAVER, H., JR., AND JOHNSTON, P. R. *Finite Elements for Structural Analysis.* Prentice-Hall, Englewood Cliffs, N.J., 1984.
23. WING, O., AND HUANG, J. W. A computational model of parallel solution of linear equations. *IEEE Trans. Comput. C-29*, 7 (1980), 632–638.

Received December 1986; revised July 1987; accepted September 1987