Parallel Computing 1 (1984) 223–235
North-Holland

On some parallel banded system solvers

Jack J. DONGARRA *

Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439, U.S.A.

Ahmed H. SAMEH **

Department of Computer Science, University of Illinois, Urbana Champaign, Urbana, IL 61801, U.S.A.

Received March 1984

Abstract. This paper describes algorithms for solving narrow banded systems and the Helmholtz difference equations that are suitable for multiprocessing systems. The organization of the algorithms highlight the large grain parallelism inherent in the problems.

Keywords. Banded systems, parallelism, multiprocessor, partitioning.

1. Introduction

We consider algorithms for solving narrow-banded diagonally dominant linear systems which are suitable for multiprocessors. We describe a direct solver similar to that in [12] for tridiagonal systems, and in [9] for solving a banded system on a linearly connected set of processors. We will also provide and analyze a parallel implementation of the partitioning algorithm and the matrix decomposition which we refer to as a hybrid solver (direct and iterative) which is superior to the direct solver especially for strongly diagonally dominant systems. When the interconnection network is not sufficiently powerful, a bottleneck develops in one of the stages of the direct solver in which the cost of the computation is proportional to the number of processors. This inefficiency may be alleviated by using an iterative scheme in this stage that takes full advantage of the parallelism offered even by a linear array of p processors.

A similar approach is also used to handle the positive-definite system that arises from the standard five-point finite-difference discretization of the Helmholtz equation. This problem arises frequently in situations where fast solvers are of primary importance. In this paper we consider the matrix decomposition solver that has been described in several papers, e.g. [1,2,6,10,11].

2. A partitioning algorithm for banded systems

Let the linear system under consideration be denoted by

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

where A is a banded diagonally dominant matrix of order n. We assume that the number of superdiagonals $m \ll n$ is equal to the number of subdiagonals. On a sequential machine such a system would be solved via Gaussian elimination without pivoting at a cost of $O(m^2 n)$ arithmetic operations. We describe here an algorithm for solving this system on a multiprocessor of p processing units. Each unit may be a sequential machine, a vector machine, or an

* Work supported in part by the Applied Mathematical Sciences Research Program (KC-04-02) of the Office of Energy Research of the U.S. Department of Energy under contract W-31-109-Eng-38.

** Work supported in part by the National Science Foundation under grant US NSF MCS 81-17010.

array of processors. In this paper, however, we consider only p sequential processing units.

Let the system (1) be partitioned into the block-tridiagonal form shown below

$$\begin{pmatrix} A_1 & B_1 & & & & \\ C_2 & A_2 & B_2 & & & \\ & \cdot & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & & C_{p-1} & A_{p-1} & B_{p-1} \\ & & & & C_p & A_p \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_{n-1} \\ f_n \end{pmatrix} \tag{2}$$

where A_i, $1 \leqslant i \leqslant p-1$, is a banded matrix of order $q = \lceil n/p \rceil$ and bandwidth $2m+1$ (same as A),

$$B_i = \begin{pmatrix} 0 & 0 \\ \hat{B}_i & 0 \end{pmatrix}, \quad 1 \leqslant i \leqslant p-1 \tag{3a}$$

and

$$C_{i+1} = \begin{pmatrix} 0 & \hat{C}_{i+1} \\ 0 & 0 \end{pmatrix}, \tag{3b}$$

in which $\hat{B}_i$ and $\hat{C}_{i+1}$ are lower and upper triangular matrices, respectively, each of order m. The algorithm consists of four stages.

2.1. Stage 1

Obtain the LU-factorization

$$A_i = L_i U_i, \quad 1 \leqslant i \leqslant p \tag{4}$$

using Gaussian elimination without pivoting, one processor per factorization. Here L_i is unit lower triangular and U_i is a non-singular upper triangular matrix. Note that each A_i is also diagonally dominant.

The cost of this stage is $O(m^2 n/p)$ arithmetic operations, no inter-processor communication is required.

2.2. Stage 2

If we premultiply both sides of (2) by

$$\operatorname{diag}\left(A_1^{-1}, A_2^{-1}, \ldots, A_p^{-1}\right)$$

we obtain a system of the form

$$\begin{pmatrix} I_q & E_1 & & & & \\ F_2 & I_q & E_2 & & & \\ & \cdot & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & & F_{p-1} & I_q & E_{p-1} \\ & & & & F_p & I_q \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} g_1 \\ g_2 \\ \vdots \\ g_{n-1} \\ g_n \end{pmatrix} \tag{5}$$

where

$$E_i = (\hat{E}_i, 0), \qquad F_i = (0, \hat{F}_i),$$

in which $\hat{E}_i$ and $\hat{F}_i$ are matrices of m columns given by

$$\hat{E}_i = A_i^{-1} \begin{pmatrix} 0 \\ \hat{B}_i \end{pmatrix} \quad \text{and} \quad \hat{F}_i = A_i^{-1} \begin{pmatrix} \hat{C}_i \\ 0 \end{pmatrix}$$

and will in general be full. In other words $\hat{E}_i$, $\hat{F}_i$, and g_i are obtained by solving the linear systems

$$L_jU_j\left[\hat{F}_j,\ \hat{E}_j,\ g_j\right]=\left[\begin{pmatrix}0\\ \hat{B}_i\end{pmatrix},\begin{pmatrix}\hat{C}_i\\ 0\end{pmatrix},\ f_i\right]\quad \text{for } 1\leqslant i\leqslant p,$$

here $\hat{C}_1=0$ and $\hat{B}_p=0$. Each processor $2\leqslant k\leqslant p-1$ handles $2m+1$ linear systems of the form $L_kU_kv=r$, while processors 1 and p each handles $m+1$ linear systems of the same form.

The cost at this stage is $O(m^2n/p)$ arithmetic operations, no inter-processor communications are needed.

2.3. Stage 3

Let $\hat{E}_i$ and $\hat{F}_i$ be partitioned, in turn, as follows

$$\hat{F}_i=\begin{pmatrix}P_i\\ M_i\\ Q_i\end{pmatrix}\quad\text{and}\quad \hat{E}_i=\begin{pmatrix}S_i\\ N_i\\ T_i\end{pmatrix},$$

where P_i, Q_i, S_i, and $T_i\in R^{m\times m}$. Also, let g_i and x_i be conformally partitioned:

$$g_i=\begin{pmatrix}h_{2i-2}\\ w_i\\ h_{2i-1}\end{pmatrix}\quad\text{and}\quad x_i=\begin{pmatrix}y_{2i-2}\\ z_i\\ y_{2i-1}\end{pmatrix}.$$

As an illustration we show the system (5) for $p=3$,

$$\begin{pmatrix}
I_m & & & S_1 & & & & & \\
 & I_v & & N_1 & & & & & \\
 & & I_m & T_1 & & & & & \\
 & & P_2 & I_m & & S_2 & & & \\
 & & M_2 & & I_v & N_2 & & & \\
 & & Q_2 & & & I_m & T_2 & & \\
 & & & & & P_3 & I_m & & \\
 & & & & & M_3 & & I_r & \\
 & & & & & Q_3 & & & I_m
\end{pmatrix}
\begin{pmatrix}y_0\\ z_1\\ y_1\\ y_2\\ z_2\\ y_3\\ y_4\\ z_3\\ y_5\end{pmatrix}
=
\begin{pmatrix}h_0\\ w_1\\ h_1\\ h_2\\ w_2\\ h_3\\ h_4\\ w_3\\ h_5\end{pmatrix}.$$

Observe that the unknown vectors y_1, y_2, y_3, and y_4 (each of order m) are disjoint from the rest of the unknowns. In other words, the m equations above and the m equations below each of the $p-1$ partitioning lines form an independent system of order $2m(p-1)$, which we shall refer to as the "reduced system" $Ky=h$, which is of the form

$$\begin{pmatrix}
I_m & T_1 & 0 & 0 & & & & & & \\
P_2 & I_m & 0 & S_2 & & & & & & \\
Q_2 & 0 & I_m & T_2 & 0 & 0 & & & & \\
 & & P_3 & I_m & 0 & S_3 & & & & \\
 & & Q_3 & 0 & I_m & T_3 & & & & \\
 & & & & P_4 & I_m & & & & \\
 & & & & & & \ddots & & & \\
 & & & & & & I_m & T_{p-2} & 0 & 0 \\
 & & & & & & P_{p-1} & I_m & 0 & S_{p-1} \\
 & & & & & & Q_{p-1} & 0 & I_m & T_{p-1} \\
 & & & & & & & & P_p & I_m
\end{pmatrix}
\begin{pmatrix}y_1\\ y_2\\ y_3\\ y_4\\ y_5\\ y_6\\ \cdot\\ y_{2p-5}\\ y_{2p-4}\\ y_{2p-3}\\ y_{2p-2}\end{pmatrix}
=
\begin{pmatrix}h_1\\ h_2\\ h_3\\ h_4\\ h_5\\ h_6\\ \cdot\\ h_{2p-5}\\ h_{2p-4}\\ h_{2p-3}\\ h_{2p-2}\end{pmatrix}.$$

(6)

The cost of the algorithm to be used for solving (6) depends on the interconnection network. Processor 1 contains T_1 and h_1, processor j, $2 \leqslant j \leqslant p-1$, contains P_j, Q_j, S_j, T_j, and h_{2j-2}, h_{2j-1}, and processor p contains P_p, and h_{2p-2}. Hence, if the processors are linearly connected we can only solve (6) sequentially at the cost of $O(m^3p)$ steps, where a step is the cost of an arithmetic operation or the cost of transmitting a floating-point number from one processor to either of its immediate neighbors. We should add here that since A is diagonally dominant it can be shown that (6) is also diagonally dominant and hence can be solved via Gaussian elimination without pivoting.

2.4. Stage 4

Once the y_i's are obtained, with y_1 in processor 1, y_{2j-2} and y_{2j-1} in processor j $(2 \leqslant j \leqslant p-1)$, and y_{2p-2} in processor p, the rest of the components of the solution vector of (5) may be computed as follows. Processor k, $1 \leqslant k \leqslant p$, evaluates

$$z_k = w_k - M_k y_{2k-3} - N_k y_{2k} \tag{7}$$

with processors 1 and p performing the additional tasks

$$y_0 = h_0 - S_1 y_2 \quad \text{and} \quad y_{2p-1} = h_{2p-1} - Q_p y_{2p-3}, \tag{8}$$

respectively (M_1 and N_p are non-existence and are taken to be zero in this equation). The cost of this stage is $O(mn/p)$ steps, with no inter-processor communication.

It can be shown that for a linear array of processors, the speedup of this algorithm over the classical sequential algorithm behaves as shown in Fig. 1 where p_0 and σ_0 are $O(\sqrt{n/m})$. Stage 2 dominates the computation until p_0, then the communication costs impact the performance and Stage 3 has a greater influence.

For a linear array of processors, the bottleneck in this parallel algorithm is the process of solving the reduced system (Stage 3). It is the only stage whose cost increases as p becomes large. This inefficiency may be alleviated by solving the reduced system (6) using an iterative scheme that takes full advantage of the parallelism offered even by a linear array of p processors. Since the reduced system is diagonally dominant, the simplest iterative scheme that is suited for such a linear array of processors is the block-Jacobi algorithm which we outline below.

Let

$$G_i = \begin{pmatrix} I_m & T_i \\ P_{i+1} & I_m \end{pmatrix}, \quad i = 1, 2, \ldots, p-1, \qquad G = \operatorname{diag}(G_1, G_2, \ldots, G_{p-1}), \tag{9}$$

and let H be the block-tridiagonal matrix

$$H = \left[\begin{pmatrix} -Q_j & 0 \\ 0 & 0 \end{pmatrix}, 0, \begin{pmatrix} 0 & 0 \\ 0 & -S_{j+1} \end{pmatrix} \right],$$

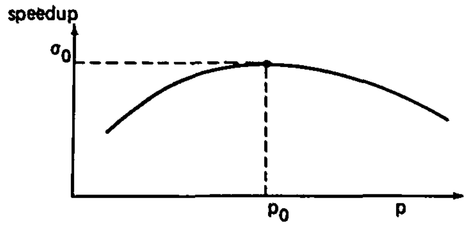

Fig. 1.

i.e. $K = G - H$. Now, the Jacobi iteration is given by

$$Gy^{(k+1)} = Hy^{(k)} + h. \tag{10}$$

If each processor j, $1 \leqslant j \leqslant p-1$, transmits T_j and h_{2j-1} to processor $j+1$, and each processor $2 \leqslant j \leqslant p-1$, transmits S_j to processor $j+1$, at a total cost of $m(2m+1)$ steps, then the matrices G, H, the right-hand side h, and an iterate $y^{(k)} = (y_1^{(k)^{\mathrm{T}}}, \ldots, y_{2p-2}^{(k)^{\mathrm{T}}})^{\mathrm{T}}$ are stored in processors 2, 3, ..., p as shown in Table 1.

Now, the Jacobi scheme may be organized on a linear array of p processors as follows.

(a) Preprocessing.

Each processor $2 \leqslant j \leqslant p$ obtains the LU-factorization of G_{j-1}, and generates the random vectors $y_{2j-3}^{(0)}$ and $y_{2j-2}^{(0)}$

(b) Iteration: $k = 0, 1, 2, \ldots$.

Each processor $j = 2, 3, \ldots, p$ performs the following:

(1) Computes $u_{j+1}^{(k)} = Q_j y_{2j-3}^{(k)}$ and transmit it to processor $j+1$; note that $Q_p \equiv 0$.

(2) Computes $v_{j-1}^{(k)} = S_{j-1} y_{2j-2}^{(k)}$ and transmits it to processor $j-1$; note that $S_1 \equiv 0$.

(3)

$$\begin{pmatrix} \tilde{u}_j^{(k)} \\ \tilde{v}_j^{(k)} \end{pmatrix} = \begin{pmatrix} u_j^{(k)} \\ v_j^{(k)} \end{pmatrix} + \begin{pmatrix} h_{2j-3} \\ h_{2j-2} \end{pmatrix};$$

note that $u_2^{(k)} = v_p^{(k)} = 0$.

(4) Given the factorization of G_{j-1}, solve the linear system

$$G_{j-1} \begin{pmatrix} y_{2j-3}^{(k+1)} \\ y_{2j-2}^{(k+1)} \end{pmatrix} = \begin{pmatrix} \tilde{u}_j^{(k)} \\ \tilde{v}_j^{(k)} \end{pmatrix}.$$

As a result, the cost per iteration is roughly $6m^2$ steps. Thus, if the number of iterations τ necessary to achieve a "reasonable" accuracy is small, the cost of this iterative scheme will be independent of p and appreciably smaller than the cost of the direct solver used in Stage 3 ($\mathrm{O}(m^3 p)$). For strong diagonally systems (1), convergence of the iterative scheme should be fast as the system has been preconditioned by $\operatorname{diag}(A_1^{-1}, \ldots, A_p^{-1})$.

We have made extensive use of LINPACK [4].

This Jacobi scheme may be replaced by a C.G.-like algorithm, Orthomin 1, e.g. see [5] or [13] for solving the reduced system $Ky = h$ in (6). The only restriction on this scheme is that $(K + K^{\mathrm{T}})$ must be positive definite. We consider two preconditioners: $M_1 = G$ in (9), and M_2 being the lower block triangular part of (6), i.e.

$$M_2 = \begin{bmatrix} G_1 & & & & \\ U_2 & G_2 & & & \\ & * & * & & \\ & & * & * & \\ & & & U_{p-1} & G_{p-1} \end{bmatrix},$$

Table 1

Processor 2	Processor 3		Processor j		Processor p
G_1	G_2		G_{j-1}		G_{p-1}
	S_2		S_{j-1}		S_{p-1}
Q_2	Q_3	...	Q_j	...	
h_1, h_2	h_3, h_4		h_{2j-3}, h_{2j-2}		h_{2p-3}, h_{2p-2}
$y_1^{(k)}$, $y_2^{(k)}$	$y_3^{(k)}$, $y_4^{(k)}$		$y_{2j-3}^{(k)}$, $y_{2j-2}^{(k)}$		$y_{2p-3}^{(k)}$, $y_{2p-2}^{(k)}$

where G_i is given by (9), and

$$U_i = \begin{pmatrix} Q_i & 0 \\ 0 & 0 \end{pmatrix}.$$

The Orthomin (1) scheme is given by

(a) Initial step.

y_0 is chosen arbitrarily,

$r_0 = h - Ky_0$,

$s_0 = M^{-1} r_0$,

$p_0 = s_0, \quad q_0 = Kp_0, \quad u_0 = Ks_0.$

(b) Iteration: $i = 0, 1, 2, \ldots.$

(1) $\tau_i = q_i^T q_i$,

(2) $\alpha_i = r_i^T q_i / \tau_i$,

(3) $_{i+1} = y_i + \alpha_i p_i$,

(4) $r_{i+1} = r_i - \alpha_i q_i$,

(5) $s_{i+1} = M^{-1} r_{i+1}$,

(6) $u_{i+1} = Ks_{i+1}$,

(7) $\beta_i = -(u_{i+1}^T q_i)/\tau_i$,

(8) $p_{i+1} = s_{i+1} + \beta_i p_i$,

(9) $q_{i+1} = u_{i+1} + \beta_i q_i$.

When $M = M_1$ we refer to the scheme as PBAND2, and as PBAND3 when $M = M_2$. The cost of the arithmetic of one iteration of Orthomin (1) is naturally higher than one iteration of the block-Jacobi scheme PBAND1. While one iteration of the block-Jacobi scheme is practically free of inter-processor communication, an iteration of Orthomin (1) requires solving systems of the form $Ms = r$, and requires that the result of two inner products be made available to all processing units. Hence, Orthomin (1) is to be preferred only on multiprocessors with powerful interconnection networks.

3. The matrix-decomposition algorithm on a multiprocessor

Consider the Helmholtz problem

$$-\nabla^2 u + \alpha^2 u = \varphi(x, y) \tag{11}$$

on the unit square with Dirichlet boundary conditions. Using a uniform mesh of size $h = 1/(n + 1)$, the usual five-point finite difference discretization of (11) yields the linear system

$$\begin{pmatrix} T & -I_n & & & & \\ -I_n & T & -I_n & & & \\ & \cdot & \vdots & \vdots & \cdot & \\ & & & -I_n & T & -I_n \\ & & & & -I_n & T \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n-1} \\ u_n \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_{n-1} \\ f_n \end{pmatrix}, \tag{12}$$

where u_i, $f_i \in \mathbb{R}^n$, and $T_i = [-1, 4 + \alpha^2, -1]$ is a tridiagonal matrix of order n. Assuming that the number of processing units p is such that $q = n/p$ is an integer that is greater than or equal to 4, the system (12) may be partitioned as shown below

$$\begin{pmatrix} N & E & & & & \\ E^{\mathrm{T}} & N & E & & & \\ & \cdot & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & & E^{\mathrm{T}} & N & E \\ & & & & E^{\mathrm{T}} & N \end{pmatrix} \begin{pmatrix} \tilde{u}_1 \\ \tilde{u}_2 \\ \vdots \\ \tilde{u}_{p-1} \\ u_p \end{pmatrix} = \begin{pmatrix} \tilde{f}_1 \\ \tilde{f}_2 \\ \vdots \\ \tilde{f}_{p-1} \\ \tilde{f}_p \end{pmatrix}, \tag{13}$$

where

$$E = \begin{pmatrix} 0 & 0 \\ -I_n & 0 \end{pmatrix}$$

and $N = [-I_n, T, -I_n]$ are of order qn,

$$\tilde{u}_i = \left(u_{(i-1)q+1}^{\mathrm{T}}, \ldots, u_{iq}^{\mathrm{T}}\right)^{\mathrm{T}} \quad \text{and} \quad \tilde{f}_i = \left(f_{(i-1)q+1}^{\mathrm{T}}, \ldots, f_{iq}^{\mathrm{T}}\right)^{\mathrm{T}}, \quad 1 \leqslant i \leqslant p.$$

If the multiprocessor consist of p processing units, the matrix decomposition algorithm (MD-algorithm) may be organized as follows.

3.1. Stage 1

Each matrix T has the spectral decomposition

$$T = Q\Lambda Q$$

where Q is an orthogonal matrix whose elements are given by

$$e_i^{\mathrm{T}} Q e_j = \sqrt{2/(n+1)}\ \sin(ij\pi/(n+1)), \quad 1 \leqslant i,\ j \leqslant n$$

and $\Lambda = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$, with

$$\lambda_j = (2 + \alpha^2) + 4\sin^2(j\pi/(n+1)), \quad 1 \leqslant j \leqslant n.$$

Now, (13) is reduced to the form

$$\begin{pmatrix} M & E & & & & \\ E^{\mathrm{T}} & M & E & & & \\ & \cdot & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & & E^{\mathrm{T}} & M & E \\ & & & & E^{\mathrm{T}} & M \end{pmatrix} \begin{pmatrix} \tilde{v}_1 \\ \tilde{v}_2 \\ \vdots \\ \tilde{v}_{p-1} \\ \tilde{v}_p \end{pmatrix} = \begin{pmatrix} \tilde{g}_1 \\ \tilde{g}_2 \\ \vdots \\ \tilde{g}_{p-1} \\ \tilde{g}_p \end{pmatrix}, \tag{14}$$

where $M = [-I_n, \Lambda, -I_n]$ is of order qn.

$$\tilde{v}_i = \tilde{Q}\tilde{u}_i, \quad \text{and} \quad \tilde{g}_i = \tilde{Q}\tilde{f}_i$$

in which $\tilde{Q} = \operatorname{diag}(Q, \ldots, Q)$ is of order qn. Here, the ith processing unit performs the q sine transforms

$$g_{(i-1)q+k} = Qf_{(i-1)g+k}, \quad 1 \leqslant k \leqslant q, \tag{15}$$

where $\tilde{g}_i = (g_{(i-1)q+1}^{\mathrm{T}}, \ldots, g_{iq}^{\mathrm{T}})^{\mathrm{T}}$, $1 \leqslant i \leqslant p$.

3.2. Stage 2

Premultiplying both sides of (14) by $\operatorname{diag}(M^{-1}, \ldots, M^{-1})$, we obtain a system of the form

$$\begin{pmatrix} I & F & & & & \\ G & I & F & & & \\ & \cdot & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & & G & I & F \\ & & & & G & I \end{pmatrix} \begin{pmatrix} \tilde{v}_1 \\ \tilde{v}_2 \\ \vdots \\ \tilde{v}_{p-1} \\ \tilde{v}_p \end{pmatrix} = \begin{pmatrix} \tilde{h}_1 \\ \tilde{h}_2 \\ \vdots \\ \tilde{h}_{p-1} \\ \tilde{h}_p \end{pmatrix}, \tag{16}$$

where

$$\tilde{h}_i = \left(h^{\mathrm{T}}_{(i-1)q+1}, \ldots, h^{\mathrm{T}}_{iq}\right)^{\mathrm{T}}, \qquad \tilde{v}_i = \left(v^{\mathrm{T}}_{(i-1)q+1}, \ldots, v^{\mathrm{T}}_{iq}\right)^{\mathrm{T}},$$

$G = M^{-1}E^{\mathrm{T}}$, $F = M^{-1}E$, and $\tilde{h}_i = M^{-1}\tilde{g}_i$, $1 \leqslant i \leqslant p$. From the structure of M, we see that solving a linear system of the form $Mx = r$, where $x = (\xi_1, \ldots, \xi_{qn})^{\mathrm{T}}$ and $r = (\rho_1, \ldots, \rho_{qn})^{\mathrm{T}}$, consists of solving n independent positive-definite tridiagonal systems $T_j x_j = r_j$, $1 \leqslant j \leqslant n$, each of order q.

$$\begin{pmatrix} \lambda_j & -1 & & & \\ -1 & \lambda_j & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & \lambda_j & -1 \\ & & & -1 & \lambda_j \end{pmatrix} \begin{pmatrix} \xi_j \\ \xi_{n+j} \\ \vdots \\ \xi_{(q-2)n+j} \\ \xi_{(q-1)n+j} \end{pmatrix} = \begin{pmatrix} \rho_j \\ \rho_{n+j} \\ \vdots \\ \rho_{(q-2)n+j} \\ \rho_{(q-1)n+j} \end{pmatrix}.$$

Hence, F and G are obtained by solving the positive-definite systems

$$T_j b_j = -e_1, \quad 1 \leqslant j \leqslant n \tag{17a}$$

and

$$T_j c_j = -e_q, \quad 1 \leqslant j \leqslant n. \tag{17b}$$

Observing that each T_j is a symmetric Toeplitz matrix, then if $c_j = (\gamma_1^{(j)}, \ldots, \gamma_q^{(j)})^{\mathrm{T}}$, b_j is given by $b_j = (\gamma_q^{(j)}, \ldots, \gamma_1^{(j)})^{\mathrm{T}}$, e.g. see [8]. Note that G and F are of the form

$$G = \begin{pmatrix} & H_q \\ 0 & \vdots \\ & H_2 \\ & H_1 \end{pmatrix}, \quad \text{and} \quad F = \begin{pmatrix} H_1 & \\ H_2 & \\ \vdots & 0 \\ H_q & \end{pmatrix},$$

respectively, with H_i being a diagonal matrix of order n,

$$H_i = \begin{pmatrix} \gamma_i^{(1)} & & & \\ & \gamma_i^{(2)} & & \\ & & \ddots & \\ & & & \gamma_i^{(n)} \end{pmatrix}.$$

$M\tilde{h}_i = \tilde{g}_i$ may also be solved by considering the n independent systems

$$T_j d_j^{(i)} = s_j^{(i)}, \quad 1 \leqslant j \leqslant n \tag{17c}$$

where

$$s_j^{(i)} = \begin{pmatrix} e_j^{\mathrm{T}} g_{(i-1)q+1} \\ e_j^{\mathrm{T}} g_{(i-1)q+2} \\ \vdots \\ e_j^{\mathrm{T}} g_{iq} \end{pmatrix} \quad \text{and} \quad h_{(i-1)q+k} = \begin{pmatrix} e_k^{\mathrm{T}} d_1^{(i)} \\ e_k^{\mathrm{T}} d_2^{(i)} \\ \vdots \\ e_k^{\mathrm{T}} d_n^{(i)} \end{pmatrix}, \quad 1 \leqslant k \leqslant q.$$

Hence the ith processing unit, $1 \leqslant i \leqslant p$, handles the following tasks:

(1) The factorizations

$$T_k = L_k D_k L_k^{\mathrm{T}}, \quad (i-1)q + 1 \leqslant k \leqslant iq.$$

where L_k is unit lower bidiagonal and D_k is diagonal with positive elements.

(2) Solving

$$L_k D_k L_k^{\mathrm{T}} c_k = -e_q, \quad (i-1)q+1 \leqslant k \leqslant iq,$$

where

$$c_k = \left(\gamma_1^{(k)}, \gamma_2^{(k)}, \ldots, \gamma_q^{(k)}\right).$$

(3) Solving

$$L_j D_j L_j^{\mathrm{T}} d_j^{(i)} = s_j^{(i)}, \quad 1 \leqslant j \leqslant n.$$

For $p = 4$, and $q = 4$, the matrix of coefficients of system (16) is of the form

$$\begin{pmatrix}
I & & & & D & & & & & & & & & & & \\
 & I & & & D & & & & & & & & & & & \\
 & & I & & D & & & & & & & & & & & \\
 & & & I & D & & & & & & & & & & & \\
 & & & D & I & & & & D & & & & & & & \\
 & & & D & & I & & & D & & & & & & & \\
 & & & D & & & I & & D & & & & & & & \\
 & & & D & & & & I & D & & & & & & & \\
 & & & & & & & D & I & & & & D & & & \\
 & & & & & & & D & & I & & & D & & & \\
 & & & & & & & D & & & I & & D & & & \\
 & & & & & & & D & & & & I & D & & & \\
 & & & & & & & & & & & D & I & & & \\
 & & & & & & & & & & & D & & I & & \\
 & & & & & & & & & & & D & & & I & \\
 & & & & & & & & & & & D & & & & I
\end{pmatrix},$$

where D denotes a diagonal matrix.

3.3. Stage 3

Again, as we have seen in the previous section, we extract from (16) an independent system of order $2n(p-1)$,

$$Hw = u, \tag{18}$$

where

$$H = \begin{pmatrix}
\Gamma & \Omega & & & \\
\hat{\Omega} & \Gamma & \Omega & & \\
 & \cdot & \cdot & \cdot & \\
 & & \hat{\Omega} & \Gamma & \Omega \\
 & & & \hat{\Omega} & \Gamma
\end{pmatrix},$$

in which

$$\Gamma = \begin{pmatrix} I_n & H_q \\ H_q & I_n \end{pmatrix}, \quad \Omega = \begin{pmatrix} 0 & 0 \\ 0 & H_1 \end{pmatrix}, \quad \hat{\Omega} = \begin{pmatrix} H_1 & 0 \\ 0 & 0 \end{pmatrix},$$

$$u = \left(h_q^{\mathrm{T}}, h_{q+1}^{\mathrm{T}}; h_{2q}^{\mathrm{T}}, h_{2q+1}^{\mathrm{T}}; \ldots; h_{(p-1)q}^{\mathrm{T}}, h_{(p-1)q+1}^{\mathrm{T}}\right)^{\mathrm{T}},$$

and

$$w = \left(v_q^{\mathrm{T}}, v_{q+1}^{\mathrm{T}}; v_{2q}^{\mathrm{T}}, v_{2q+1}^{\mathrm{T}}; \ldots; v_{(p-1)q}^{\mathrm{T}}, v_{(p-1)q+1}^{\mathrm{T}}\right)^{\mathrm{T}}.$$

In turn, this system consists of n independent pentadiagonal systems each of order $2(p-1)$ and is of the form

$$P_k w_k = u_k, \quad 1 \leqslant k \leqslant n, \tag{19}$$

where

$$P_k = \begin{pmatrix} \Phi & \Psi & & & \\ \hat{\Psi} & \Phi & \Psi & & \\ & \cdot & \cdot & \cdot & \\ & & \cdot & \cdot & \\ & & \hat{\Psi} & \Phi & \Psi \\ & & & \hat{\Psi} & \Phi \end{pmatrix},$$

in which

$$\Phi = \begin{pmatrix} 1 & \gamma_q^{(k)} \\ \gamma_q^{(k)} & 1 \end{pmatrix}, \quad \Psi = \begin{pmatrix} 0 & 0 \\ 0 & \gamma_1^{(k)} \end{pmatrix} \quad \hat{\Psi} = \begin{pmatrix} \gamma_1^{(k)} & 0 \\ 0 & 0 \end{pmatrix},$$

$$u_k = \begin{pmatrix} e_k^{\mathrm{T}} h_q \\ e_k^{\mathrm{T}} h_{q+1} \\ e_k^{\mathrm{T}} h_{2q} \\ e_k^{\mathrm{T}} h_{2q+1} \\ \vdots \\ e_k^{\mathrm{T}} h_{(p-1)q} \\ e_k^{\mathrm{T}} h_{(p-1)q+1} \end{pmatrix}, \quad \text{and} \quad w_k = \begin{pmatrix} e_k^{\mathrm{T}} v_q \\ e_k^{\mathrm{T}} v_{q+1} \\ e_k^{\mathrm{T}} v_{2q} \\ e_k^{\mathrm{T}} v_{2q+1} \\ \vdots \\ e^{j\mathrm{T}} v_{(p-1)q} \\ e_k^{\mathrm{T}} v_{(p-1)q+1} \end{pmatrix}.$$

Note that H and P_k, $1 \leqslant k \leqslant n$, are non-singular, and it can be verified that each system (19) may be solved using Gaussian elimination without pivoting. Processing unit i solves $P_k u_k = u_k$ for $k = (i-1)q + 1, \ldots, iq$.

3.4. Stage 4

Finally, the rest of the components of the solution of (16) are extracted as follows:

(a) Processing unit 1.

$$v_j = h_j - H_j v_{q+1}, \quad 1 \leqslant j \leqslant q-1,$$

(b) Processing unit i.

$$v_{(i-1)q+j} = h_{(i-1)q+j} - \left(H_j v_{iq+1} + H_{q-j+1} v_{(i-1)q}\right), \quad 2 \leqslant j \leqslant q-1,$$

(c) Processing unit p.

$$v_{(p-1)q+j} = h_{(p-1)q+j} - H_{q-j+1} v_{(p-1)q}, \quad 2 \leqslant j \leqslant q.$$

3.5. Stage 5

Once $\bar{v}_1, \ldots, \bar{v}_p$ are computed the solution of (13) is obtained via the sine transforms

$$u_k = Q v_k, \quad k = 1, 2, \ldots, n$$

with the ith processing unit computing $u_{(i-1)q+1}, \ldots, u_{iq}$.

The cost of the arithmetic (alone) of this algorithm is $O(n \log_2 n)$ and is dominated by the sine transforms in Stages 1 and 5. If the arithmetic is overlapped with the inter-processing communication (through the global memory), the total cost remains $O(n \log_2 n)$.

4. Numerical results

The algorithms described above require more arithmetic operations than the well-known sequential algorithms currently in use for solving the problems described above. Hence, they should not be competitive on sequential machines. We have used all of our parallel schemes as sequential solvers and have obtained results of equal numerical accuracy with LINPACK [4], for solving narrow banded systems, and with FISHPACK for solving the Helmholtz equation.

PBAND consumed twice the time as the routines SGBFA and SGBSL from LINPACK on a VAX-780 for solving the system of linear equations (1) with $n = 512$, $m = 5$, and the number of processors $p = 16$. This ratio of time consumed held for three examples: $A_k = [a_{ij}^{(k)}]$, $1 \leqslant k \leqslant 3$, with $a_{ij}^{(k)}$, $i \neq j$, obtained by a random number generator, and the diagonal elements $a_{ii}^{(k)} = a_k$ given by $a_1 = 32$, $a_2 = 5$, and $a_3 = 3$. All A_k are non-singular with A_1 being the only diagonally dominant matrix. In Table 2 we compare the number of iterations required by PBAND1, 2, and 3 to achieve a residual of 2-norm less than 10^{-5}. We see, then, that if the multiprocessor possesses a reasonably powerful interconnection network, the PBAND 3 version of Orthomin (1) can be an effective scheme for solving the system when A is not diagonally dominant but the reduced system has a positive definite symmetric part.

We ran our FORTRAN program on the CRAY X-MP-4 using the multitasking features available on the machine [3]. The CRAY X-MP-4 has four processors which can be used by a single FORTRAN program. In running the band solver we using one, two, three, and four processors to solve the problem. Table 3 shows the results for PBAND3 for a positive-definite random matrix of various orders. For large problems using two processors the speedups are almost perfect. Results with three and four processors show some degradation in preformance, but considering the level of granularity the results are quite impressive.

We also implemented the same program on the Denelcor HEP. The Denecor HEP is a MIMD computer which supports tightly coupled parallel processing. The fully configured computing system offered by Denelcor consists of up to 16 processing elements (PEMs) sharing a large global memory through a crossbar switch. Within a single PEM, parallelism is achieved through pipelining independent serial instructions streams called processes. The principal pipeline that handles the numerical and logical operations consists of synchronous functional

Table 2

Algorithm	$\alpha_1 = 32$	$\alpha_1 = 5$	$\alpha_1 = 3$
PBAND1	2	4	29
PBAND2	1	3	23
PBAND3	1	3	10

Table 3

n	Bandwidth	Time (seconds) 1 processor	Speedup 2 processors	Speedup 3 processors	Speedup 4 processors
512	11	0.024	1.8	2.40	2.86
4096	15	0.256	1.98	2.86	3.75
16384	31	2.12	1.995	2.91	3.87

Table 4

n	Bandwidth	Speedup over running sequentially
512	11	6.
4096	15	8.
16384	31	Too large to run

Table 5

p	1	8	16	32
FISHPACK	12			
Helm		36	37	37

units that have been segmented into an eight-stage pipe. The HEP we used had only a single PEM, and the maximum speedup over the same process running sequentially is between eight and ten. For further details on the HEP architecture see the article [7] by H. Jordan. See also Table 4.

Similarly, we compared HELM with the cyclic reduction scheme of FISHPACK [13] for solving the Poisson equation ($\alpha = 0$) on the unit square with mesh size 1/129, i.e. $n = 128$, for $p = 8$, 16, and 32 processors. Both schemes succeeded in producing residual of 2-norms less than 10^{-5}. Table 5 shows the time in seconds consumed by HELM and FISHPACK running on a VAX11/780 for different values of p.

The results in Table 5 reflect the fact that HELM performs redundant arithmetic operations compared to their sequential scheme in FISHPACK (cyclic reduction). One may also conclude that with an "appropriate" inter-connection of the p processors, HELM may achieve a speedup of $p/3$ over FISHPACK's sequential algorithm. True speedups achieved by these parallel algorithms over the sequential counterparts can only be measured by actual runs on a specific multiprocessor.

Acknowledgements

We would like to thank the people at CRAY Research, especially Chris Hsiung and Tom Hewitt, for helping with the runs on the CRAY X-MP-4 and Ken Hillstrom at Argonne for help with the HEP runs.

References

[1] B. Buzbee, A fast Poisson solver amenable to parallel computation, *IEEE Trans. Comput.* C-22 (1973) 793–796.

[2] B. Buzbee, G. Golub and C. Nielson, On direct methods for solving Poisson's equation, *SIAM J. Numer. Anal.* 7 (1970) 627–656.

[3] S.S. Chen, J.J. Dongarra and C.C. Hsiung, Multiprocessing linear algebra algorithms on CRAY X-MP-2: Experiences with small granularity, *J. Parallel* and Distributed Comput. 1 (1984) 22–31.

[4] J. Dongarra, J. Bunch, C. Moler and G. Stewart, *LINPACK Users' Guide* (SIAM, Philadelphia, 1979).

[5] H. Elman, Iterative methods for non-self-adjoint elliptic problems, *Elliptic Problem Solvers II* (Academic Press, New York, 1984) (to appear).

[6] R. Hockney, A fast direct solution of Poisson's equation using Fourier analysis, *J. ACM* 12 (1965) 95–113.

[7] H.F. Jordan, Experience with pipelined multiple instruction streams, *IEEE Proc.* (1984) 113–123.

[8] T. Kailath, A. Vieira and M. Morf, Inverses of Toeplitz operators, innovations, and orthogonal polynomials, *SIAM Rev.* 20 (1978) 106–119.

[9] D. Lawrie and A. Sameh, The computation and communication complexity of a parallel banded system solver, to appear in *ACM Trans. Math. Software* (1985).

[10] A. Sameh, A fast Poisson solver for multiprocessors, *Elliptic Problems Solvers II* (Academic Press, New York, 1984) (to appear).

[11] A. Sameh, S. Chen and D. Kuck, Parallel Poisson and biharmonic solvers, *Computing* 17 (1976) 219–230.

[12] P. Swarztrauber and R. Sweet, Algorithm 541: Efficient Fortran subprograms for the solution of separable elliptic partial differential equations, *ACM Trans. Math. Software* 5 (1979) 352–364.

[13] P. Vinsome, ORTHOMIN, an iterative method for solving sparse sets of simultaneous linear equations, *Proc. 4th Symposium on Reservoir Simulation,* Society of Petroleum Engineers of the American Institute of Mechanical Engineering (1976) 149–159.