Parallel Factorizations for Tridiagonal Matrices

P. Amodio; L. Brugnano; T. Politi

SIAM Journal on Numerical Analysis, Vol. 30, No. 3. (Jun., 1993), pp. 813-823.

Stable URL:
http://links.jstor.org/sici?sici=0036-1429%28199306%2930%3A3%3C813%3APFFTM%3E2.0.CO%3B2-E

SIAM Journal on Numerical Analysis is currently published by Society for Industrial and Applied Mathematics.

SIAM J. NUMER. ANAL.
Vol. 30, No. 3, pp. 813–823, June 1993

012

PARALLEL FACTORIZATIONS FOR TRIDIAGONAL MATRICES*

P. AMODIO†, L. BRUGNANO†, AND T. POLITI†

Abstract. The authors analyze the problem of solving tridiagonal linear systems on parallel computers. A wide class of efficient parallel solvers is derived by considering different parallel factorizations of partitioned matrices. These solvers have a minimum requirement of data transmission. In fact, communication is only needed for solving a "reduced system," whose dimension depends on the number of parallel processors used. Moreover, for a given partitioned tridiagonal matrix, the reduced system (which is again tridiagonal) is the same, and represents the only sequential part of the corresponding parallel solver.

Three examples are discussed in more detail; one of them derives a very efficient parallel method based on the cyclic reduction algorithm.

Key words. tridiagonal linear systems, parallel tridiagonal solvers

AMS subject classifications. 65F05, 15A23

1. Introduction.

In many fields of application it is necessary to solve tridiagonal linear systems. The best scalar algorithm for their solution derives from the LU factorization of the coefficient matrix. However, this algorithm is quite inefficient on a parallel or vector computer.

In the last several years many algorithms have been proposed to solve tridiagonal linear systems on parallel computers [6], [11], [14], [15], [16]. In this paper we shall consider parallel direct methods. A coarse classification of such methods is partition methods [1], [2], [7], [10], [17], domain decomposition methods [13], and cyclic reduction [3], [5], [8], [9].

We propose a unifying approach to the problem of efficiently solving tridiagonal linear systems on parallel computers when the size of the problem is much greater than the number p of parallel processing units. The approach is based on a more detailed study of factorizations of partitioned matrices. This allows us to derive most of the known algorithms, as well as new ones, in the class of partition methods.

In §§3, 4, and 5, three factorizations are discussed in more detail; they are obtained as simple extensions of widely known scalar factorizations. In §6 the computational cost and the expected speedup of the three algorithms is discussed. In §7 some numerical tests are reported.

2. Factorizations of partitioned matrices.

Let us consider the tridiagonal matrix

$$A = \begin{pmatrix} a_1 & c_1 & & & \\ b_1 & a_2 & c_2 & & \\ & \cdot & \cdot & \cdot & \\ & & \cdot & \cdot & c_{n-1} \\ & & & b_{n-1} & a_n \end{pmatrix} \tag{1}$$

* Received by the editors February 25, 1991; accepted for publication (in revised form) May 26, 1992. Work supported by the Consiglio Nazionale delle Ricerche contract 89.1791.01 and "Progetto Finalizzato Calcolo Parallelo"-sottoprogetto 1 and by the European Community through the P.C.A.-ESPRIT project.

† Dipartimento di Matematica, Università di Bari, Campus Universitario, 70125 Bari, Italy (00110570@vm.csata.it).

where we assume, for simplicity, that $n = kp - 1$. We consider the following partition of A:

$$(2)\quad A = \begin{pmatrix} A^{(1)} & c_{k-1}^{(1)}\mathbf{e}_{k-1} & & & & & \\ b_{k-1}^{(1)}\mathbf{e}_{k-1}^T & a_k^{(1)} & c_0^{(2)}\mathbf{e}_1^T & & & & \\ & b_0^{(2)}\mathbf{e}_1 & A^{(2)} & c_{k-1}^{(2)}\mathbf{e}_{k-1} & & & \\ & & b_{k-1}^{(2)}\mathbf{e}_{k-1}^T & a_k^{(2)} & & & \\ & & & & \ddots & & \\ & & & & & a_k^{(p-1)} & c_0^{(p)}\mathbf{e}_1^T \\ & & & & & b_0^{(p)}\mathbf{e}_1 & A^{(p)} \end{pmatrix},$$

where $\mathbf{e}_1$ and $\mathbf{e}_{k-1}$ are respectively the first and the last unit vectors of R^{k-1},

$$(3)\quad A^{(i)} = \begin{pmatrix} a_1^{(i)} & c_1^{(i)} & & \\ b_1^{(i)} & a_2^{(i)} & \ddots & \\ & \ddots & \ddots & c_{k-2}^{(i)} \\ & & b_{k-2}^{(i)} & a_{k-1}^{(i)} \end{pmatrix},$$

and

$$a_r^{(i)} = a_{(i-1)k+r}, \qquad b_r^{(i)} = b_{(i-1)k+r}, \qquad c_r^{(i)} = c_{(i-1)k+r}.$$

If the blocks $A^{(i)}$ are nonsingular, we can factor A as

$$(4)\quad A = F\,T,$$

where, with I_{k-1} the identity matrix of order $k-1$,

$$(5)\quad F = \begin{pmatrix} A^{(1)} & \mathbf{0} & & & & & & \\ \mathbf{v}^{(1)^T} & 1 & \mathbf{w}^{(2)^T} & 0 & & & & \\ & \mathbf{0} & A^{(2)} & \mathbf{0} & & & & \\ & 0 & \mathbf{v}^{(2)^T} & 1 & \mathbf{w}^{(3)^T} & 0 & & \\ & & & \mathbf{0} & A^{(3)} & \mathbf{0} & & \\ & & & 0 & \mathbf{v}^{(3)^T} & 1 & & \\ & & & & & & \ddots & \\ & & & & & & 1 & \mathbf{w}^{(p)^T} \\ & & & & & & \mathbf{0} & A^{(p)} \end{pmatrix},$$

$$T = \begin{pmatrix} I_{k-1} & \mathbf{y}^{(1)} & & & & & & \\ \mathbf{0}^T & \alpha^{(1)} & \mathbf{0}^T & \gamma^{(2)} & & & & \\ & \mathbf{z}^{(2)} & I_{k-1} & \mathbf{y}^{(2)} & & & & \\ & \beta^{(2)} & \mathbf{0}^T & \alpha^{(2)} & \mathbf{0}^T & \gamma^{(3)} & & \\ & & & \mathbf{z}^{(3)} & I_{k-1} & \mathbf{y}^{(3)} & & \\ & & & \beta^{(3)} & \mathbf{0}^T & \alpha^{(3)} & & \\ & & & & & & \ddots & \\ & & & & & & \alpha^{(p-1)} & \mathbf{0}^T \\ & & & & & & \mathbf{z}^{(p)} & I_{k-1} \end{pmatrix}.$$

The unknown elements of F and T are obtained from (4) by direct identification:

$$\mathbf{w}^{(i)} = c_0^{(i)}\mathbf{e}_1, \qquad \mathbf{v}^{(i)} = b_{k-1}^{(i)}\mathbf{e}_{k-1},$$

$$\mathbf{z}^{(i)} = b_0^{(i)}(A^{(i)})^{-1}\mathbf{e}_1, \qquad \mathbf{y}^{(i)} = c_{k-1}^{(i)}(A^{(i)})^{-1}\mathbf{e}_{k-1},$$

(6)

$$\alpha^{(i)} = a_k^{(i)} - b_{k-1}^{(i)}c_{k-1}^{(i)}\mathbf{e}_{k-1}^T(A^{(i)})^{-1}\mathbf{e}_{k-1} - b_0^{(i+1)}c_0^{(i+1)}\mathbf{e}_1^T(A^{(i+1)})^{-1}\mathbf{e}_1,$$

$$\beta^{(i)} = -b_0^{(i)}b_{k-1}^{(i)}\mathbf{e}_{k-1}^T(A^{(i)})^{-1}\mathbf{e}_1, \qquad \gamma^{(i)} = -c_0^{(i)}c_{k-1}^{(i)}\mathbf{e}_1^T(A^{(i)})^{-1}\mathbf{e}_{k-1}.$$

This formalism is essentially the same used by Johnsson [7] to derive his algorithm. However, from (4) and (5) it is possible to derive many partition methods by observing that the matrices $A^{(i)}$ may be factored in different ways. This will allow us to derive efficient parallel algorithms for the solution of the tridiagonal linear system

$$A\mathbf{x} = \mathbf{b}. \tag{7}$$

Using the factorization (4) the solution of (7) is obtained (see (5)) by first solving in parallel the p linear subsystems with the matrices $A^{(i)}$ and then updating the corresponding right-hand sides by means of the vectors $\mathbf{v}^{(i)^T}$ and $\mathbf{w}^{(i)^T}$. Then we must solve the linear system with the matrix T. This implies the solution of the *reduced system* with the *reduced matrix*

$$T_p = \begin{pmatrix} \alpha^{(1)} & \gamma^{(2)} & & \\ \beta^{(2)} & \alpha^{(2)} & \ddots & \\ & \ddots & \ddots & \gamma^{(p-1)} \\ & & \beta^{(p-1)} & \alpha^{(p-1)} \end{pmatrix}_{(p-1)\times(p-1)} \tag{8}$$

(which is obvious from (5) and represents the only sequential part of the algorithm). Then the right-hand sides are updated in parallel by means of the vectors $\mathbf{z}^{(i)}$ and $\mathbf{y}^{(i)}$. Some authors (for instance [2]) also examine the problem of solving the reduced system in parallel. We assume $n \gg p$ (the number of parallel processing units); therefore, we will neglect this aspect.

By choosing different factorizations for the blocks $A^{(i)}$, different parallel tridiagonal solvers can be obtained. However, for each solver the reduced matrix is the same (this is, easily derived from (4) and (5)). This fact is very important, because the stability of a parallel algorithm depends on:

1. the stability of the parallel factorization of the $A^{(i)}$;
2. the algorithm for solving the reduced system.

This means that both the parallel factorization of the $A^{(i)}$ and the factorization of T_p (to solve the reduced system) must be stable for the corresponding parallel algorithm to be stable. Consequently, the two problems can be examined separately.

Let us now consider the reduced matrix (8). If the factorization (4) exists, then the following results hold true.

THEOREM 2.1. *The reduced matrix* (8) *preserves the property of diagonal dominance of* A.

Proof. If the factorization (4) exists, then the blocks $A^{(i)}$ (see (2) and (3)) are nonsingular and factorizable as

$$P^{(i)}A^{(i)} = L^{(i)}D^{(i)}U^{(i)},$$

where $P^{(i)}$ is a permutation matrix, $L^{(i)}$ is a lower triangular matrix with unit main diagonal, $U^{(i)}$ is an upper triangular matrix with unit main diagonal, and $D^{(i)}$ is a diagonal matrix. Using the absolute value of matrices, it follows that

$$|A^{(i)}| \leq {P^{(i)}}^T |L^{(i)}| \ |D^{(i)}| \ |U^{(i)}| = \hat{A}^{(i)},$$

$$(\hat{A}^{(i)})^{-1} = |U^{(i)}|^{-1}|D^{(i)}|^{-1}|L^{(i)}|^{-1}P^{(i)} \geq |(A^{(i)})^{-1}|.$$

If A is diagonally dominant, it follows that

$$\hat{A}^{(i)}\mathbf{e} \geq |A^{(i)}|\mathbf{e} > |b_0^{(i)}|\mathbf{e}_1 + |c_{k-1}^{(i)}|\mathbf{e}_{k-1},$$

where $\mathbf{e} = (1,\ldots,1)^T$. The reduced matrix (8) will be diagonally dominant if (see (6))

$$\begin{aligned}\eta^{(i)} := |a_k^{(i)}| - |b_{k-1}^{(i)}| \ \mathbf{e}_{k-1}^T \ |(A^{(i)})^{-1}| \ (|b_0^{(i)}|\mathbf{e}_1 + |c_{k-1}^{(i)}|\mathbf{e}_{k-1}) \\ - |c_0^{(i+1)}| \ \mathbf{e}_1^T \ |(A^{(i+1)})^{-1}| \ (|b_0^{(i+1)}|\mathbf{e}_1 + |c_{k-1}^{(i+1)}|\mathbf{e}_{k-1}) > 0.\end{aligned}$$

Indeed, we have

$$\begin{aligned}\eta^{(i)} > |a_k^{(i)}| - |b_{k-1}^{(i)}|\mathbf{e}_{k-1}^T(\hat{A}^{(i)})^{-1}\hat{A}^{(i)}\mathbf{e} \\ - |c_0^{(i+1)}|\mathbf{e}_1^T(\hat{A}^{(i+1)})^{-1}\hat{A}^{(i+1)}\mathbf{e} = |a_k^{(i)}| - |b_{k-1}^{(i)}| - |c_0^{(i+1)}| > 0.\end{aligned}$$

Similarly, if the matrix is weakly diagonally dominant. □

THEOREM 2.2. *If A is irreducible, then the reduced matrix* (8) *is irreducible.*

Proof. By considering the adjoint matrix of $A^{(i)}$, it is simple to show that if A is irreducible, then

$$(A^{(i)})^{-1}_{1,k-1} \neq 0, \qquad (A^{(i)})^{-1}_{k-1,1} \neq 0.$$

The thesis follows from (6). □

Moreover (see [7]), symmetry and positive definiteness are also preserved.

To underline the local properties of the partition (2), we consider the $(n+p-1)\times n$ matrix R_p recursively defined as follows:

$$R_1 = I_{k-1}, \quad R_i = \begin{pmatrix} I_{k-1} & & \\ & 1 & \\ & 1 & \\ & & R_{i-1} \end{pmatrix}, \quad i = 2,\ldots,p.$$

By simple calculations one verifies that

$$A = R_p^T \cdot M \cdot R_p,$$

where M is block diagonal:

$$M = \begin{pmatrix} M^{(1)} & & \\ & \ddots & \\ & & M^{(p)} \end{pmatrix},$$

and

$$M^{(1)} = \begin{pmatrix} A^{(1)} & c_{k-1}^{(1)}\mathbf{e}_{k-1} \\ b_{k-1}^{(1)}\mathbf{e}_{k-1}^T & a_k^{(1)} \end{pmatrix},$$

$$M^{(i)} = \begin{pmatrix} 0 & c_0^{(i)}\mathbf{e}_1^T & 0 \\ b_0^{(i)}\mathbf{e}_1 & A^{(i)} & c_{k-1}^{(i)}\mathbf{e}_{k-1} \\ 0 & b_{k-1}^{(i)}\mathbf{e}_{k-1}^T & a_k^{(i)} \end{pmatrix}, \quad i = 2, \ldots, p-1, \tag{9}$$

$$M^{(p)} = \begin{pmatrix} 0 & c_0^{(p)}\mathbf{e}_1^T \\ b_0^{(p)}\mathbf{e}_1 & A^{(p)} \end{pmatrix}.$$

If the factorization (4) exists we can state (with some obvious differences for $M^{(1)}$ and $M^{(p)}$) that

$$M^{(i)} = \begin{pmatrix} 1 & \mathbf{w}^{(i)^T} & 0 \\ \mathbf{0} & A^{(i)} & \mathbf{0} \\ 0 & \mathbf{v}^{(i)^T} & 1 \end{pmatrix} \begin{pmatrix} \alpha_1^{(i)} & \mathbf{0}^T & \gamma^{(i)} \\ \mathbf{z}^{(i)} & I_{k-1} & \mathbf{y}^{(i)} \\ \beta^{(i)} & \mathbf{0}^T & \alpha_2^{(i)} \end{pmatrix}. \tag{10}$$

The vectors $\mathbf{v}^{(i)}$, $\mathbf{w}^{(i)}$, $\mathbf{z}^{(i)}$, $\mathbf{y}^{(i)}$, and the scalars $\beta^{(i)}$ and $\gamma^{(i)}$ are the same as in (5). Moreover, we have (see (6) and (8))

$$\alpha^{(i)} = \alpha_2^{(i)} + \alpha_1^{(i+1)}, \qquad i = 1, \ldots, p-1,$$
$$\alpha_1^{(i)} = -b_0^{(i)}c_0^{(i)}\mathbf{e}_1^T(A^{(i)})^{-1}\mathbf{e}_1, \qquad \alpha_2^{(i)} = a_k^{(i)} - b_{k-1}^{(i)}c_{k-1}^{(i)}\mathbf{e}_{k-1}^T(A^{(i)})^{-1}\mathbf{e}_{k-1}.$$

Therefore, we can "handle" the blocks $M^{(i)}$, instead of the matrix (2). It follows that a parallel factorization of A can be characterized by the factorizations of the blocks $A^{(i)}$ in (10). Nevertheless, in the following sections we shall consider decompositions of $M^{(i)}$ which are different from (10), but with the constraint that the elements of the reduced matrix must remain unchanged.

3. Local *LU* factorization. In several works the *LU* factorization has been used to factor the blocks $A^{(i)}$; for instance, Johnsson's algorithm [7] is equivalent to the following factorization of $M^{(i)}$:

$$M^{(i)} = \begin{pmatrix} \alpha_1^{(i)} & \mathbf{w}^{(i)^T} & \gamma^{(i)} \\ \mathbf{0} & L^{(i)}U^{(i)} & \mathbf{0} \\ \beta^{(i)} & \mathbf{v}^{(i)^T} & \alpha_2^{(i)} \end{pmatrix} \begin{pmatrix} 1 & \mathbf{0}^T & 0 \\ \mathbf{z}^{(i)} & I_{k-1} & \mathbf{y}^{(i)} \\ 0 & \mathbf{0}^T & 1 \end{pmatrix}.$$

If the *LU* factorizations (without pivoting) of the blocks $A^{(i)}$ exist, then the cost of this algorithm is about $18n$ scalar (arithmetic) operations when $L^{(i)}$ has unit main diagonal. The cost is reduced by n operations when the matrix $U^{(i)}$ has unit main diagonal. However, we shall consider the following alternative factorization for the blocks $M^{(i)}$:

$$M^{(i)} = \begin{pmatrix} \alpha_1^{(i)} & \mathbf{w}^{(i)^T} & \gamma^{(i)} \\ \mathbf{0} & L^{(i)} & \mathbf{0} \\ \beta^{(i)} & \mathbf{v}^{(i)^T} & \alpha_2^{(i)} \end{pmatrix} \begin{pmatrix} 1 & \mathbf{0}^T & 0 \\ \mathbf{z}^{(i)} & U^{(i)} & \mathbf{y}^{(i)} \\ 0 & \mathbf{0}^T & 1 \end{pmatrix}. \tag{11}$$

The matrices $L^{(i)}$, $U^{(i)}$, the vectors $\mathbf{v}^{(i)}$, $\mathbf{y}^{(i)}$, and the scalar $\alpha_2^{(i)}$ in (11) are defined such that

$$\begin{pmatrix} L^{(i)} & \mathbf{0} \\ \mathbf{v}^{(i)^T} & \alpha_2^{(i)} \end{pmatrix} \begin{pmatrix} U^{(i)} & \mathbf{y}^{(i)} \\ \mathbf{0}^T & 1 \end{pmatrix}$$

is a LU factorization of the submatrix (see (9))

$$\begin{pmatrix} A^{(i)} & c_{k-1}^{(i)}\mathbf{e}_{k-1} \\ b_{k-1}^{(i)}\mathbf{e}_{k-1}^T & a_k^{(i)} \end{pmatrix}.$$

This implies that the matrix $L^{(i)}$ is lower bidiagonal, the matrix $U^{(i)}$ is upper bidiagonal and only the last component of the vectors $\mathbf{y}^{(i)}$ and $\mathbf{v}^{(i)}$ is nonzero. Moreover, from (11) and (9) we obtain

$$L^{(i)}\mathbf{z}^{(i)} = b_0^{(i)}\mathbf{e}_1, \qquad U^{(i)^T}\mathbf{w}^{(i)} = c_0^{(i)}\mathbf{e}_1,$$
$$\alpha_1^{(i)} = -\mathbf{w}^{(i)^T}\mathbf{z}^{(i)}, \quad \beta^{(i)} = -\mathbf{v}^{(i)^T}\mathbf{z}^{(i)}, \quad \gamma^{(i)} = -\mathbf{w}^{(i)^T}\mathbf{y}^{(i)}.$$

We observe that $\mathbf{z}^{(i)}$ and $\mathbf{w}^{(i)}$ are full vectors: we shall call them *fill-in* vectors.

From widely known results about the LU factorization (see, for example, [4]) it follows that the parallel solver corresponding to (11) is stable if A is diagonally dominant or weakly diagonally dominant and irreducible, or symmetric and positive definite.

4. Local LUD factorization. An alternative factorization for the block $M^{(i)}$ which preserves the banded structure of the matrix $A^{(i)}$ is obtained by diagonalizing $A^{(i)}$ by means of Gauss transformations. In this case, the matrix $A^{(i)}$ is factored in the form $L^{(i)}U^{(i)}D^{(i)}$ where $L^{(i)}$ and $U^{(i)}$ are, respectively, lower and upper bidiagonal with unit main diagonal and $D^{(i)}$ is diagonal. Then, the local LUD factorization is given by

$$M^{(i)} = \begin{pmatrix} 1 & \mathbf{0}^T & 0 \\ \mathbf{0} & L^{(i)} & \mathbf{0} \\ 0 & \mathbf{v}^{(i)^T} & 1 \end{pmatrix} \begin{pmatrix} 1 & \mathbf{w}^{(i)^T} & 0 \\ \mathbf{0} & U^{(i)} & \mathbf{0} \\ 0 & \mathbf{0}^T & 1 \end{pmatrix} \begin{pmatrix} \alpha_1^{(i)} & \mathbf{0}^T & \gamma^{(i)} \\ \mathbf{z}^{(i)} & D^{(i)} & \mathbf{y}^{(i)} \\ \beta^{(i)} & \mathbf{0}^T & \alpha_2^{(i)} \end{pmatrix}, \tag{12}$$

where

$$L^{(i)}U^{(i)}\mathbf{z}^{(i)} = b_0^{(i)}\mathbf{e}_1, \qquad D^{(i)}\mathbf{w}^{(i)} = c_0^{(i)}\mathbf{e}_1,$$
$$U^{(i)}\mathbf{y}^{(i)} = c_{k-1}^{(i)}\mathbf{e}_{k-1}, \qquad D^{(i)}\mathbf{v}^{(i)} = b_{k-1}^{(i)}\mathbf{e}_{k-1},$$
$$\alpha_1^{(i)} = -\mathbf{w}^{(i)^T}\mathbf{z}^{(i)}, \qquad \alpha_2^{(i)} = a_k^{(i)} - \mathbf{v}^{(i)^T}\mathbf{y}^{(i)},$$
$$\gamma^{(i)} = -\mathbf{w}^{(i)^T}\mathbf{y}^{(i)}, \qquad \beta^{(i)} = -\mathbf{v}^{(i)^T}\mathbf{z}^{(i)}.$$

Therefore, both $\mathbf{v}^{(i)}$ and $\mathbf{w}^{(i)}$ have only one nonzero entry, while $\mathbf{z}^{(i)}$ and $\mathbf{y}^{(i)}$ are fill-in vectors. The stability considerations of the factorization are similar to those of the local LU factorization.

The parallel solver corresponding to (12) is an improved version of Wang's algorithm [12], [18] (compare the parallel operation count reported in §6 with [18]).

5. Local cyclic reduction (CR) factorization. Cyclic reduction is an interesting algorithm for the solution of tridiagonal linear systems on vector and parallel computers [3], [5], [8], [13]. However, its parallel implementation needs a synchronization among the processors at each step of the reduction (see [8]). Here we propose a way to overcome this problem.

We apply the cyclic reduction locally to each block $A^{(i)}$ in (9). Even if this factorization is more expensive than the LU or LUD factorizations, there are no fill-in vectors in the corresponding factorization of the block $M^{(i)}$. This implies, as we shall see later, that the resulting parallel algorithm is no more expensive than the ones previously examined.

First we recall some notions concerning the cyclic reduction factorization. If we consider an odd-even permutation matrix P_1 of dimension $k-1$, it follows that

$$A^{(i)} = P_1^T \begin{pmatrix} C_1 & T_1 \\ S_1 & B_1 \end{pmatrix} P_1,$$

(we neglect the upper index to simplify the notation). C_1 and B_1 are diagonal matrices containing the odd and even diagonal elements of $A^{(i)}$, respectively; S_1 and T_1 are bidiagonal matrices with the off-diagonal elements on the even and odd rows of $A^{(i)}$. If C_1^{-1} exists, then we define

$$A^{(i)} = P_1^T L_1 D_1 U_1 P_1 = P_1^T \begin{pmatrix} I & \\ S_1 C_1^{-1} & I \end{pmatrix} \begin{pmatrix} I & \\ & A_1 \end{pmatrix} \begin{pmatrix} C_1 & T_1 \\ & I \end{pmatrix} P_1, \tag{13}$$

where $A_1 = B_1 - S_1\, C_1^{-1}\, T_1$ is again tridiagonal (of dimension $\lfloor (k-1)/2 \rfloor$). We can repeat the same operations for D_1 by considering the matrix

$$P_2 = \begin{pmatrix} I & \\ & Q_2 \end{pmatrix},$$

where Q_2 is the odd-even permutation matrix of order $\lfloor (k-1)/2 \rfloor$. The process continues until D_r ($r = \lceil \log_2 k \rceil$) is the identity matrix of order $k-1$.

Then, we extend the factorization (13) to the matrix $M^{(i)}$ by means of the following matrices:

$$\hat{P}_1 = \begin{pmatrix} 1 & & \\ & P_1 & \\ & & 1 \end{pmatrix},$$

$$\hat{L}_1 = \begin{pmatrix} 1 & \tilde{\mathbf{w}}_1^T & \mathbf{0}^T & 0 \\ \mathbf{0} & I & O & \mathbf{0} \\ \mathbf{0} & S_1 C_1^{-1} & I & \mathbf{0} \\ 0 & \tilde{\mathbf{v}}_1^T & \mathbf{0}^T & 1 \end{pmatrix}, \qquad \hat{U}_1 = \begin{pmatrix} 1 & \mathbf{0}^T & \mathbf{0}^T & 0 \\ \tilde{\mathbf{z}}_1 & C_1 & T_1 & \tilde{\mathbf{y}}_1 \\ \mathbf{0} & O & I & \mathbf{0} \\ 0 & \mathbf{0}^T & \mathbf{0}^T & 1 \end{pmatrix},$$

and

$$\hat{D}_1 = \begin{pmatrix} \hat{\alpha}_1 & \mathbf{0}^T & \hat{\mathbf{w}}_1^T & \hat{\gamma} \\ \mathbf{0} & I & O & \mathbf{0} \\ \hat{\mathbf{z}}_1 & O & A_1 & \hat{\mathbf{y}}_1 \\ \hat{\beta} & \mathbf{0}^T & \hat{\mathbf{v}}_1^T & \hat{\alpha}_2 \end{pmatrix}. \tag{14}$$

Defining the vector $\mathbf{w}_1^T = \begin{pmatrix} \tilde{\mathbf{w}}_1^T & \hat{\mathbf{w}}_1^T \end{pmatrix}$, and similarly the vectors $\mathbf{v}_1$, $\mathbf{z}_1$, and $\mathbf{y}_1$, we obtain

$$U_1^T \mathbf{w}_1 = c_0^{(i)} \mathbf{e}_1, \qquad U_1^T \mathbf{v}_1 = b_{k-1}^{(i)} P_1^T \mathbf{e}_{k-1},$$

$$L_1 \mathbf{z}_1 = b_0^{(i)} \mathbf{e}_1, \qquad L_1 \mathbf{y}_1 = c_{k-1}^{(i)} P_1^T \mathbf{e}_{k-1},$$

$$\hat{\alpha}_1 = -\tilde{\mathbf{w}}_1^T \tilde{\mathbf{z}}_1, \qquad \hat{\alpha}_2 = a_k^{(i)} - \tilde{\mathbf{v}}_1^T \tilde{\mathbf{y}}_1,$$

$$\hat{\beta} = -\tilde{\mathbf{v}}_1^T \tilde{\mathbf{z}}_1, \qquad \hat{\gamma} = -\tilde{\mathbf{w}}_1^T \tilde{\mathbf{y}}_1.$$

It results that all the vectors $\tilde{\mathbf{w}}_1$, $\tilde{\mathbf{z}}_1$, $\tilde{\mathbf{v}}_1$, $\tilde{\mathbf{y}}_1$, $\hat{\mathbf{w}}_1$, $\hat{\mathbf{z}}_1$, $\hat{\mathbf{v}}_1$, $\hat{\mathbf{y}}_1$ have at most one nonzero element, that is, there are no fill-in vectors.

We observe that $\hat{\beta} = 0$ and $\hat{\gamma} = 0$; moreover if one removes the blocks on the second row and second column of $\hat{D}_1$, the resulting matrix is still tridiagonal.

The structure of the matrices $\hat{D}_i$, for $i = 2, \ldots, r-1$, is similar to (14). At the rth step, we have

$$\hat{D}_r = \begin{pmatrix} \alpha_1^{(i)} & \mathbf{0}^T & \gamma^{(i)} \\ \mathbf{0} & I_{k-1} & \mathbf{0} \\ \beta^{(i)} & \mathbf{0}^T & \alpha_2^{(i)} \end{pmatrix},$$

where $\alpha_1^{(i)}$, $\alpha_2^{(i)}$, $\beta^{(i)}$, $\gamma^{(i)}$ are the scalars defined in (10).

The corresponding parallel solver is stable when A is strongly diagonally dominant or weakly diagonally dominant and irreducible (see [3]), or symmetric and positive definite.

We observe that the cyclic reduction algorithm is easily vectorizable [9], [13]: it follows that the local cyclic reduction algorithm can also be efficiently implemented on a parallel computer whose processors have vector facilities. This is not true for the parallel methods previously considered.

6. Computational cost. We compare the three methods presented in the previous sections from the point of view of their computational cost, in terms of number of operations, trasmission of data, and required memory.

The number of scalar operations and data transmission is almost the same for the three methods. In fact, the parallelizable section of the three algorithms accounts for about $17n$ operations; in Tables 6.1–6.3 there are operation counts for each processor (for the first one and the last one it is different, because of the different structure of $M^{(1)}$ and $M^{(p)}$ (see (9))).

TABLE 6.1
Parallel operation counts for the local LU algorithm.

processor	add	multiply	divide	total
1	$3k-3$	$3k-3$	$2k-2$	$8k-8$
$2, \ldots, p-1$	$6k-8$	$8k-8$	$3k-3$	$17k-19$
p	$6k-10$	$8k-14$	$3k-4$	$17k-28$

TABLE 6.2
Parallel operation counts for the local LUD algorithm.

processor	add	multiply	divide	total
1	$4k-5$	$5k-7$	$3k-4$	$12k-16$
$2,\ldots,p-1$	$6k-8$	$8k-8$	$3k-3$	$17k-19$
p	$5k-9$	$6k-9$	$3k-4$	$14k-22$

TABLE 6.3
Parallel operation counts for the local CR *algorithm.*

processor	add	multiply	divide	total
1	$6k-6-3\delta$	$8k-8-4\delta$	$3k-3-\delta$	$17k-17-8\delta$
$2,\ldots,p-1$	$6k-6$	$8k-8$	$3k-3$	$17k-17$
p	$6k-6-3\sigma$	$8k-8-4\sigma$	$3k-3-\sigma$	$17k-17-8\sigma$

$\delta = \lceil \log_2 k \rceil$, $\sigma \in \{1, 2, \ldots, \lfloor \log_2 k \rfloor\}$.

Moreover, on a distributed memory parallel computer with p processors, we have $6(p-1)$ parallel data transmissions on each processor to construct the reduced system. This cost is for a one-dimensional torus interconnection topology among the parallel units. The scalar count of nonparallel operations is $10p - 17$. This cost is for the construction and the solution of the reduced system by means of its LU factorization.

The comparison algorithm, for evaluating the speedup of the problem, is the scalar LU factorization algorithm, whose cost is $8n-7$ operations. It follows that the expected speedup of the problem on p processors is given by ($n = kp - 1$):

$$(15)\quad s(n;p) \approx \frac{8n}{17\frac{n}{p} + 10p + 6p\eta} \approx p\frac{8k}{17k + 10p + 6p\eta} \to p\frac{8}{17}, \quad \text{when } n \to \infty,$$

where η is the ratio between the time to execute one *operation* and the time to send or receive one datum.

Even if the expected speedup is the same for the three algorithms, their memory requirements are different. In fact, the local LU and LUD algorithms require six vectors of length k per processor: three vectors are for the coefficients of the matrix; one vector contains the right-hand side at the beginning, and then, the computed solution; the last two are fill-in vectors. On the contrary, the local cyclic reduction algorithm, which has no fill-in vectors, needs only four vectors per processor. This implies that this solver has a minimum memory requirement, and in this sense it is best of the three algorithms.

7. Numerical tests. The algorithms examined in §§3, 4, and 5, have been implemented in Parallel Fortran [19] with the Express communication library [20] on a MicroWay Multiputer, which has a network of 32 transputers T800-20 each one with a local memory of 1 Mb. The scalar LU algorithm has been implemented in Fortran on a single transputer T800-20 with 16 Mb of memory.

The speedup obtained on 4, 8, 16, and 32 processors is reported in Table 7.1 for the local LU algorithm, in Table 7.2 for the local LUD algorithm, and in Table 7.3 for the local cyclic reduction algorithm.

We have observed that (for the compiler used) the smaller the data area is, the faster the code runs. This fact should be sufficient to explain why some of the speedups reported in Tables 7.1–7.3 are greater than the asymptotic speedup in (15) (a similar

result is also in [6, §4.4]), and why local cyclic reduction (which has no fill-in vectors) is the algorithm with the best performance. Nevertheless, we are not able to explain the measured differences between the performances of the local LU and the local LUD algorithms.

From the above results, and from the considerations made in §6, we can conclude that the local cyclic reduction algorithm is the most efficient parallel tridiagonal solver of these considered.

TABLE 7.1
Measured speedup for the local LU algorithm.

n	$p=4$	$p=8$	$p=16$	$p=32$
400	1.50			
800	1.76	2.11		
1600	1.93	2.82		
3200	2.03	3.41	4.05	
6400	2.08	3.79	5.49	
12800	2.11	4.02	6.69	8.01
25600	2.12	4.14	7.50	10.92
51200		4.20	7.99	13.29
128000			8.30	15.38
256000				16.14

TABLE 7.2
Measured speedup for the local LUD algorithm.

n	$p=4$	$p=8$	$p=16$	$p=32$
400	1.56			
800	1.73	2.43		
1600	1.82	2.98		
3200	1.88	3.37	4.64	
6400	1.91	3.63	5.84	
12800	1.92	3.74	6.66	8.89
25600	1.93	3.80	7.17	11.15
51200		3.84	7.44	12.78
128000			7.62	14.16
256000				14.65

TABLE 7.3
Measured speedup for the local CR *algorithm.*

n	$p=4$	$p=8$	$p=16$	$p=32$
400	1.52			
800	1.83	2.14		
1600	2.05	2.92		
3200	2.18	3.58	4.12	
6400	2.25	4.05	5.72	
12800	2.28	4.32	7.07	8.22
25600	2.30	4.48	8.01	11.37
51200		4.56	8.59	14.08
128000			8.98	16.42
256000				17.38

Acknowledgments. We express our thanks to Mrs. Paulene Butts for her help in the preparation of the manuscript. We also thank Prof. J. M. Ortega and the referees for their comments and suggestions.

REFERENCES

[1] L. BRUGNANO, *A parallel solver for tridiagonal linear systems for distributed memory parallel computers*, Parallel Comput., 17 (1991), pp. 1017–1023.

[2] I. N. HAJJ AND S. SKELBOE, *A multilevel parallel solver for block tridiagonal and banded linear systems*, Parallel Comput., 15 (1990), pp. 21–45.

[3] D. HELLER, *Some aspects of the cyclic reduction algorithm for block tridiagonal linear systems*, SIAM J. Numer. Anal., 13 (1976), pp. 484–496.

[4] N. J. HIGHAM, *Bounding the error in Gaussian elimination for tridiagonal systems*, SIAM J. Matrix Anal. Appl., 11 (1990), pp. 521–530.

[5] R. W. HOCKNEY AND C. R. JESSHOPE, *Parallel Computers*, Adam Hilger, Bristol, 1981.

[6] W. HOFFMANN AND K. POTMA, *Implementing linear algebra algorithms on a Meiko computing surface*, Appl. Numer. Math., 8 (1991), pp. 127–148.

[7] S. L. JOHNSSON, *Solving narrow banded systems on ensemble architectures*, ACM Trans. Math. Software, 11 (1985), pp. 271–288.

[8] ———, *Solving tridiagonal systems on ensemble architectures*, SIAM J. Sci. Statist. Comput., 8 (1987), pp. 354–392.

[9] D. KERSHAW, *Solution of single tridiagonal linear systems and vectorization of the ICCG algorithm on the Cray* 1, in Parallel Computations, G. Rodrigue, ed., Academic Press, New York, 1982, pp. 85–99.

[10] A. KRENCHEL, H. J. PLUM, AND K. STÜBEN, *Parallelization and vectorization aspects of the solution of tridiagonal linear systems*, Parallel Comput., 14 (1990), pp. 31–49.

[11] J. J. LAMBIOTTE AND R. VOIGT, *The solution of tridiagonal linear systems on the* CDC STAR-100 *computer*, ACM Trans. Math. Software, 1 (1975), pp. 308–329.

[12] P. H. MICHIELSE AND H. A. VAN DER VORST, *Data transport in Wang's partition methods*, Parallel Comp., 7 (1988), pp. 87–95.

[13] J. M. ORTEGA, *Introduction to Parallel and Vector Solution of Linear Systems*, Plenum Press, New York, 1988.

[14] H. S. STONE, *An efficient parallel algorithm for the solution of a tridiagonal linear system of equations*, J. Assoc. Comput. Mach., 20 (1973), pp. 27–38.

[15] ———, *Parallel tridiagonal solvers*, ACM Trans. Math. Software, 1 (1975), pp. 289–307.

[16] H. A. VAN DER VORST, *Large tridiagonal and block tridiagonal linear systems on vector and parallel computers*, Parallel Comput., 5 (1987), pp. 45–54.

[17] ———, *Analysis of a parallel solution method for tridiagonal linear systems*, Parallel Comput., 5 (1987), pp. 303–311.

[18] H. H. WANG, *A parallel method for tridiagonal equations*, ACM Trans. Math. Software, 7 (1981), pp. 170–183.

[19] *Parallel Fortran User Guide*, 3L Ltd., Livingstone, Scotland, 1988.

[20] *Express Fortran User's Guide*, ParaSoft Corporation, Pasadena, CA, 1990.