Parallel Solution of Almost Block Diagonal Systems on a Hypercube

P. Amodio*

Dipartimento di Matematica
Università di Bari
I-70125 Bari, Italy

and

M. Paprzycki†

Department of Mathematics and Computer Science
University of Texas of the Permian Basin
Odessa, Texas 79762

Submitted by André Ran

ABSTRACT

A new tearing-type approach toward the solution of Almost Block Diagonal Systems on distributed memory parallel computers is presented. Its arithmetical complexity is examined and compared with other existing approaches.

1. INTRODUCTION

Recent years have witnessed rapid growth of knowledge about the high-performance direct solution to almost block diagonal (ABD) linear systems (see Fig. 1). Such systems arise in a variety of mathematical problems: discretization of boundary value ordinary differential equations (BVP ODEs), Chebyshev spectral decomposition on rectangular domains, orthogonal spline collocation for elliptic problems and others.

Two parameters influence the parallel solution methods for ABD systems: the size of each block and the number of internal blocks m. For large blocks and a limited number of processors it is possible to use BLAS kernels [6, 7, 12] and introduce parallelism inside each block. On a shared

*E-mail: 00110570@vm.csata.it.
†E-mail: paprzycki_m@utpb.edu.

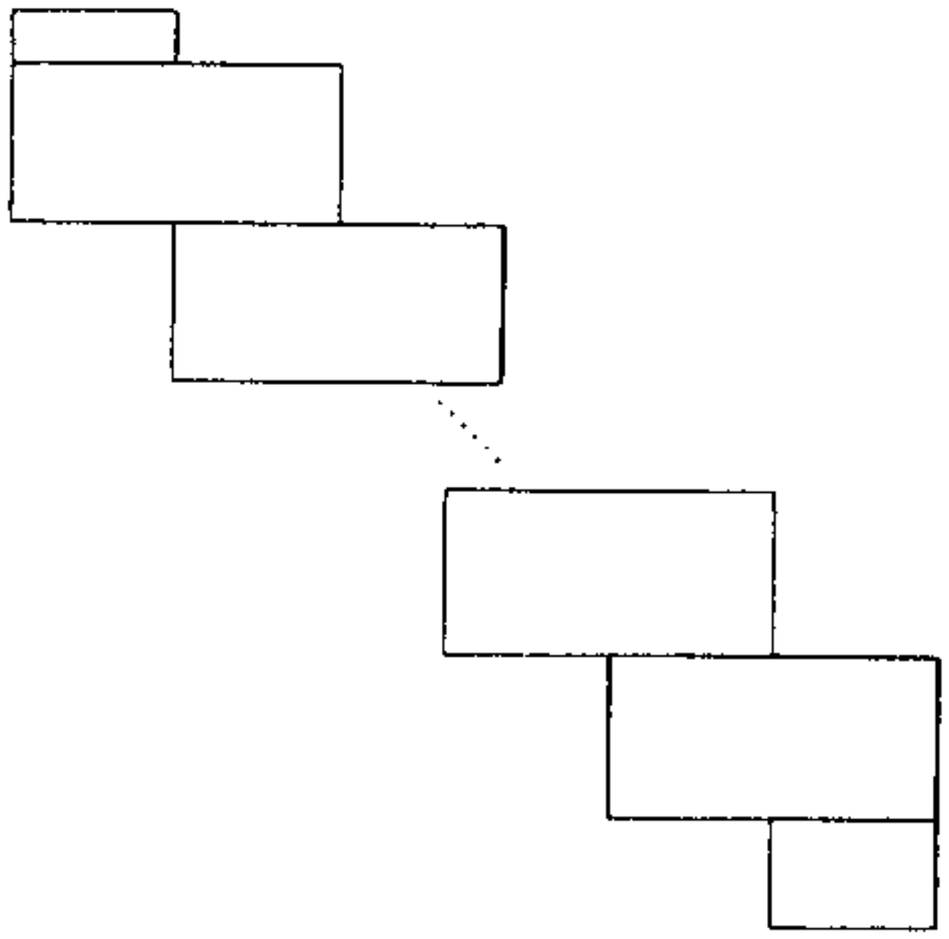

FIG. 1. Structure of an ABD matrix.

memory parallel computer (8-processor Cray Y-MP) a speedup of about 4 was reported for blocksizes 400×800 [10]. This approach will not be successful for larger number of processors, as there would not be enough work inside each block to keep them busy. This is even more true for the distributed memory computers where partitioning the blocks among the processors would lead to a very large number of data transmissions. For a large number of small to medium size blocks, tearing-type methods may be applied. Examples of such methods, applied to the linear systems originating from the discretization of BVP ODEs, are discussed in [4, 13–15]. Some of these algorithms [4, 14] are suitable primarily for shared memory computers. The algorithms introduced in [13, 15] are suitable also for distributed memory computers. S. Wright in [15] derives a stable algorithm assuming that the matrix is block bidiagonal and considering the top and bottom blocks (introduced by the boundary conditions of the BVP ODE; see Fig. 1) only at the last step of the reduction. Paprzycki and Gladwell in [13] try to take advantage of the structure of the coefficient matrix and develop a method similar to that proposed in [8, 9] for banded systems. In this article we propose a slightly different variant of this last algorithm, which allows us to save half of the fill-in vectors and to reduce the number of arithmetical operations.

Section 2 introduces the new tearing-type algorithm. Section 3 discusses the details of the parallel implementation of the proposed algorithm on a hypercube. Section 4 introduces the arithmetic complexity and memory requirements of the new algorithm and compares it with the original algorithm from [13].

2. THE PARALLEL ABD SOLVER FOR DISTRIBUTED MEMORY COMPUTERS

For the purposes of presentation we represent the ABD matrix M (Fig. 1) as:

$$M = \begin{pmatrix} A_{2,0} & D_{2,0} & & & & & \\ D_{1,1} & A_{1,1} & B_{1,1} & C_{1,1} & & & \\ C_{2,1} & B_{2,1} & A_{2,1} & D_{2,1} & & & \\ & & D_{1,2} & A_{1,2} & B_{1,2} & C_{1,2} & \\ & & C_{2,2} & B_{2,2} & A_{2,2} & D_{2,2} & \\ & & & & \ddots & & \\ & & & D_{1,m} & A_{1,m} & B_{1,m} & C_{1,m} \\ & & & C_{2,m} & B_{2,m} & A_{2,m} & D_{2,m} \\ & & & & & D_{1,m+1} & A_{1,m+1} \end{pmatrix}. \tag{1}$$

Without loss of generality we also assume that the ABD system originates from a finite difference discretization of a system of n BVP ODEs with q initial and $n - q$ final separated boundary conditions. Now, for each i the blocks $A_{2,i}$ and $C_{2,i}$ are of size $q \times q$, the blocks $A_{1,i}$ and $C_{1,i}$ are of size $(n - q) \times (n - q)$, and $B_{1,i}$, $D_{1,i}$, $B_{2,i}^T$, and $D_{2,i}^T$, are of size $(n - q) \times q$. Moreover, we treat

$$\begin{pmatrix} D_{1,i} & A_{1,i} & B_{1,i} & C_{1,i} \\ C_{2,i} & B_{2,i} & A_{2,i} & D_{2,i} \end{pmatrix}$$

as one block of the matrix M (see Fig. 1). We thus say that M has m blocks (we do not count here the top and the bottom half-blocks originated by the boundary conditions).

In [13] Paprzycki and Gladwell suggest a decomposition of the original matrix M (1) in a block tridiagonal form and a parallel factorization, which is suitable on shared memory as well as distributed memory parallel computers. This factorization also corresponds to that in [11] or to the LUD factorization in [1]. The parallel phase of the algorithm generates four fill-in block vectors with a total number of $2n^2m$ nonnull elements, which are obtained by means of the solution of six block triangular systems of size $(m + 2)/p$, where p is the number of processors available. As a result of this phase a reduced system is created, which is then solved sequentially on one processor. The reduced system has $p - 2$ internal blocks and the same ABD block structure as M (1).

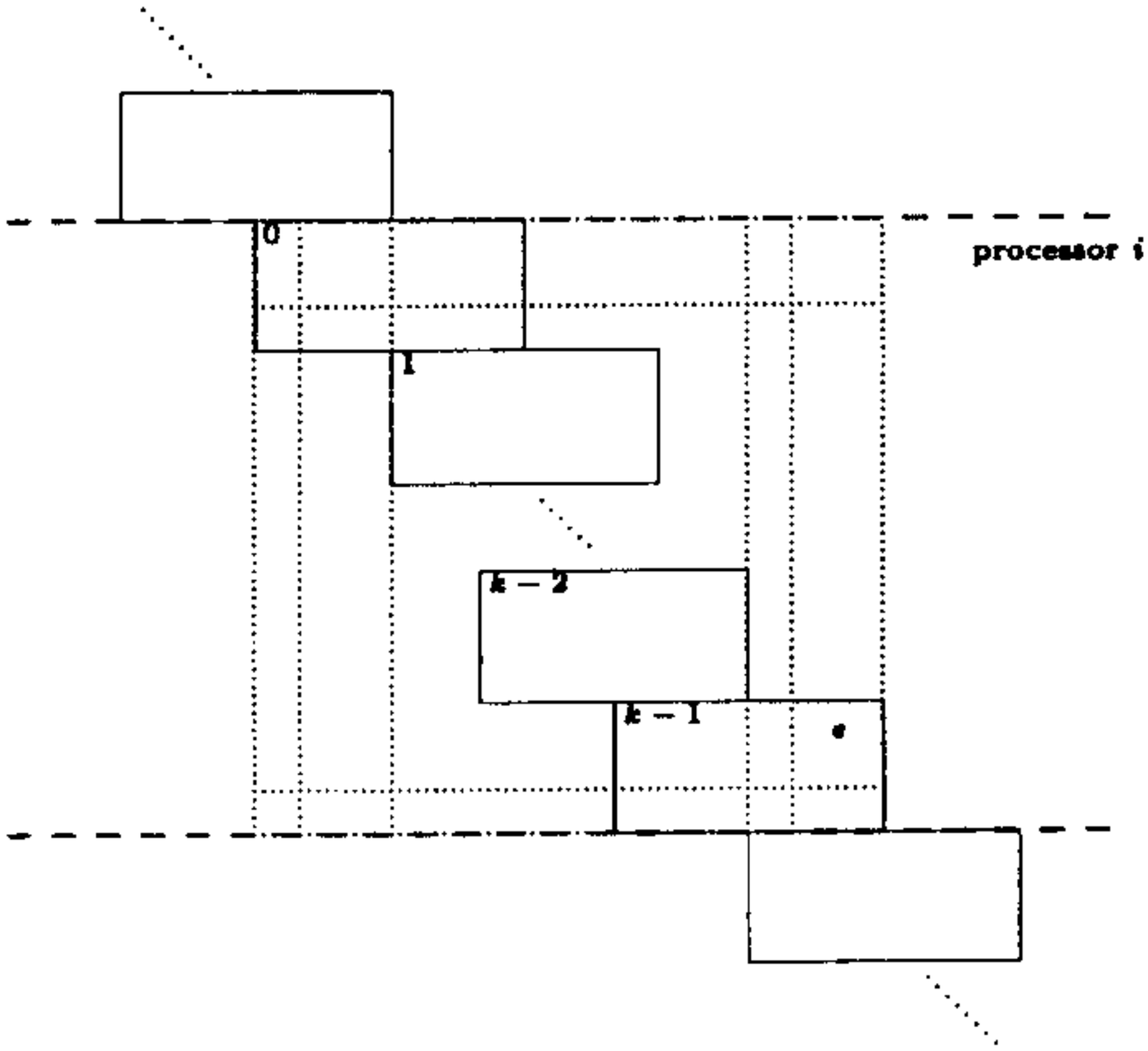

FIG. 2. Division of the coefficient matrix among the processors. The dotted lines represent the division into blocks inside each processor.

This approach can be improved first, by considering a different partitioning of M among the processors and, second, by introducing a modified algorithm for the solution of the reduced system.

2.1. Parallel Factorization

Assume for simplicity that $m = kp - 2$, and associate each processor i $(i = 2, \ldots, p-1)$ with the following k block rows of M (see also Fig. 2):

$$\begin{pmatrix} D_{1,0}^{(i)} & A_{1,0}^{(i)} & \mathbf{r}_0^{(i)^T} & & \\ \mathbf{s}_0^{(i)} & \mathbf{s}_1^{(i)} & M^{(i)} & \mathbf{r}_1^{(i)} & \mathbf{r}_2^{(i)} \\ & & \mathbf{s}_2^{(i)^T} & A_{2,k-1}^{(i)} & D_{2,k-1}^{(i)} \end{pmatrix}, \tag{2}$$

where $M^{(i)}$ is an ABD matrix with $k-2$ blocks, $E_{l,j}^{(i)} = E_{l,(i-1)k+j}$ for $E = A, B, C, D$, and

$$\mathbf{r}_0^{(i)^T} = B_{1,0}^{(i)} \mathbf{e}_1^T + C_{1,0}^{(i)} \mathbf{e}_2^T,$$

$$\mathbf{s}_2^{(i)^T} = C_{2,k-1}^{(i)} \mathbf{e}_{k-2}^T + B_{2,k-1}^{(i)} \mathbf{e}_{k-1}^T,$$
$$\mathbf{s}_0^{(i)} = \mathbf{e}_1 C_{2,0}^{(i)}, \qquad \mathbf{s}_1^{(i)} = \mathbf{e}_1 B_{2,0}^{(i)},$$
$$\mathbf{r}_1^{(i)} = \mathbf{e}_{k-1} B_{1,k-1}, \qquad \mathbf{r}_2^{(i)} = \mathbf{e}_{k-1} C_{1,k-1}.$$

In the previous formulas $\mathbf{e}_i$, for $i = 1, 2, k-2, k-1$, denotes a block unit vector, that is, a block vector with only one nonnull block equal to an identity matrix; $\mathbf{e}_1$ is of size $(k-1)n \times q$ and consists of an identity matrix of size q followed by a $((k-1)n-q) \times q$ block of zeroes; $\mathbf{e}_2$ is of size $(k-1)n \times (n-q)$ and consists of a $q \times (n-q)$ block of zeroes followed by an identity matrix of size $n-q$ and a $(k-2)n \times (n-q)$ block of zeroes; $\mathbf{e}_{k-2}$ is of size $(k-1)n \times q$ and consists of a $(k-2)n \times q$ block of zeroes followed by an identity matrix of size q and a $(n-q) \times q$ block of zeroes; and, $\mathbf{e}_{k-1}$ is of size $(k-1)n \times (n-q)$ and consists of a $((k-2)n+q) \times (n-q)$ block of zeroes followed by an identity matrix of size $n-q$.

Moreover, associate processors 1 and p with the $k-1$ block rows of M,

$$\begin{pmatrix} M^{(1)} & \mathbf{r}_1^{(1)} & \mathbf{r}_2^{(1)} \\ \mathbf{s}_2^{(1)} & A_{2,k-1}^{(1)} & D_{2,k-1}^{(1)} \end{pmatrix} \tag{3}$$

and

$$\begin{pmatrix} D_{1,0}^{(p)} & A_{1,0}^{(p)} & \mathbf{r}_0^{(p)} \\ \mathbf{s}_0^{(p)} & \mathbf{s}_1^{(p)} & M^{(p)} \end{pmatrix}, \tag{4}$$

where each block is defined as previously.

Each processor performs the following factorization (it is assumed that an appropriately modified factorization is applied to the blocks stored by the first and the last processor):

$$\begin{pmatrix} \delta_1^{(i)} & \alpha_1^{(i)} & \mathbf{w}^{(i)^T} & \beta_1^{(i)} & \gamma_1^{(i)} \\ & & I & & \\ \gamma_2^{(i)} & \beta_2^{(i)} & \mathbf{v}^{(i)^T} & \alpha_2^{(i)} & \delta_2^{(i)} \end{pmatrix} \begin{pmatrix} I & & & & \\ & I & & & \\ \mathbf{s}_0^{(i)} & \mathbf{s}_1^{(i)} & L^{(i)}U^{(i)} & \mathbf{r}_1^{(i)} & \mathbf{r}_2^{(i)} \\ & & & I & \\ & & & & I \end{pmatrix}, \tag{5}$$

where

$$\mathbf{v}^{(i)^T} = \mathbf{s}_2^{(i)^T} M^{(i)^{-1}}, \qquad \mathbf{w}^{(i)^T} = \mathbf{r}_0^{(i)^T} M^{(i)^{-1}} \tag{6}$$

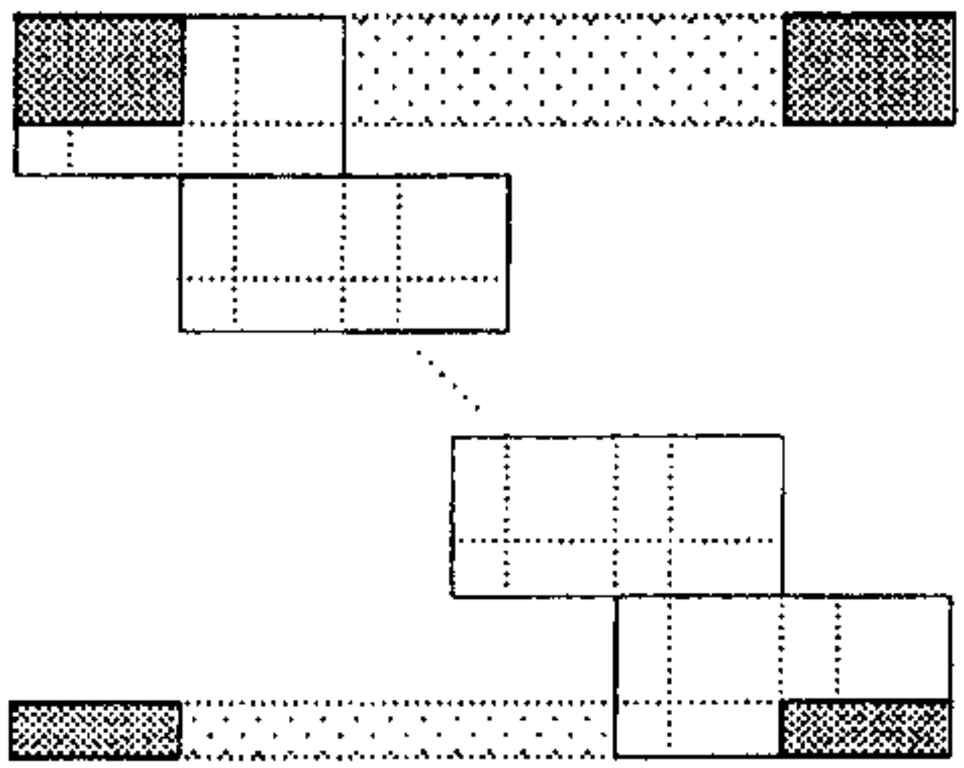

FIG. 3. Fill-in generated by each processor. The dotted lines represent the block partitioning of M. The shaded areas represent the fill-in. The double shaded blocks constitute the reduced matrix.

are the fill-in block vectors, and

$$\begin{aligned}
\left(\delta_1^{(i)}\alpha_1^{(i)}\right) &= \left(D_{1,0}^{(i)}A_{1,0}^{(i)}\right) - \mathbf{w}^{(i)^T}\left(\mathbf{s}_0^{(i)}\mathbf{s}_1^{(i)}\right),\\
\left(\alpha_2^{(i)}\delta_2^{(i)}\right) &= \left(A_{2,k-1}^{(i)}D_{2,k-1}^{(i)}\right) - \mathbf{v}^{(i)^T}\left(\mathbf{r}_1^{(i)}\mathbf{r}_2^{(i)}\right),\\
\left(\beta_1^{(i)}\gamma_1^{(i)}\right) &= -\mathbf{w}^{(i)^T}\left(\mathbf{r}_1^{(i)}\mathbf{r}_2^{(i)}\right),\\
\left(\gamma_2^{(i)}\beta_2^{(i)}\right) &= -\mathbf{v}^{(i)^T}\left(\mathbf{s}_0^{(i)}\mathbf{s}_1^{(i)}\right).
\end{aligned} \tag{7}$$

This decomposition requires only three block triangular system solutions and involves no communication among the processors. It results in fill-in as represented in Fig. 3 that is half-size (a total number of n^2m nonnull elements) of the fill-in resulting from other known algorithms [4, 13, 15]. The reduced matrix R maintains the same ABD structure and dimensions of internal blocks as the original matrix M (and the same structure and dimensions as the reduced system in [13]), that is,

$$R = \begin{pmatrix}
\alpha_2^{(1)} & \delta_2^{(1)} & & & & & & \\
\delta_1^{(2)} & \alpha_1^{(2)} & \beta_1^{(2)} & \gamma_1^{(2)} & & & & \\
\gamma_2^{(2)} & \beta_2^{(2)} & \alpha_2^{(2)} & \delta_2^{(2)} & & & & \\
 & & & \ddots & & & & \\
 & & & \delta_1^{(p-1)} & \alpha_1^{(p-1)} & \beta_1^{(p-1)} & \gamma_1^{(p-1)} & \\
 & & & \gamma_2^{(p-1)} & \beta_2^{(p-1)} & \alpha_2^{(p-1)} & \delta_2^{(p-1)} & \\
 & & & & & \delta_1^{(p)} & \alpha_1^{(p)}
\end{pmatrix}. \tag{8}$$

2.2. Parallel Solution of the Reduced System

On distributed memory parallel computers it is important to select an algorithm that minimizes the number of data transmissions rather than the number of operations. Let us analyze the solution of the reduced ABD system (8) with $p-2$ internal blocks resulting from the factorization presented in Section 2.1. We assume that $p = 2^r$ and use a recursive approach based on the algorithm proposed in the previous section applied to ABD matrices with $p-2$, $p/2-2, \ldots,$ internal blocks.

To minimize data transmission, instead of sending all the elements of the reduced system to one processor as it was performed in [13], or using a reduced number of processors at each step of the algorithm (similarly to typical cyclic reduction algorithms), we use a method similar to that introduced in [2], which utilizes all available processors in every step and performs the same operations on each. This approach implemented on a hypercube requires communication only among the neighbor processors.

The first step of the factorization phase consists of $p/2$ bidirectional communications between processors $i-1$ and i, for $i = 2, 4, 6, \ldots, p$. Processor i sends to processor $i-1$ blocks with superindex (i) whereas processor $i-1$ sends to the processor i blocks with the superindex $(i-1)$ in order to have the same ABD matrix on both processors

$$\begin{pmatrix} \delta_1^{(i-1)} & \alpha_1^{(i-1)} & \beta_1^{(i-1)} & \gamma_1^{(i-1)} & & \\ \gamma_2^{(i-1)} & \beta_2^{(i-1)} & \alpha_2^{(i-1)} & \delta_2^{(i-1)} & & \\ & & \delta_1^{(i)} & \alpha_1^{(i)} & \beta_1^{(i)} & \gamma_1^{(i)} \\ & & \gamma_2^{(i)} & \beta_2^{(i)} & \alpha_2^{(i)} & \delta_2^{(i)} \end{pmatrix} \quad \text{if } i = 4, 6, 8, p-2, \quad (9)$$

$$\begin{pmatrix} \alpha_2^{(1)} & \delta_2^{(1)} & & \\ \delta_1^{(2)} & \alpha_1^{(2)} & \beta_1^{(2)} & \gamma_1^{(2)} \\ \gamma_2^{(2)} & \beta_2^{(2)} & \alpha_2^{(2)} & \delta_2^{(2)} \end{pmatrix} \quad \text{if } i = 2, \quad (10)$$

and

$$\begin{pmatrix} \delta_1^{(p-1)} & \alpha_1^{(p-1)} & \beta_1^{(p-1)} & \gamma_1^{(p-1)} \\ \gamma_2^{(p-1)} & \beta_2^{(p-1)} & \alpha_2^{(p-1)} & \delta_2^{(p-1)} \\ & & \delta_1^{(p)} & \alpha_1^{(p)} \end{pmatrix} \quad \text{if } i = p. \quad (11)$$

In accordance with the notation used in (2)–(4), we may now define $D_{1,0}^{(i-1:i)} = \delta_1^{(i-1)}$, $A_{1,0}^{(i-1:i)} = \alpha_1^{(i-1)}$, $A_{2,k-1}^{(i-1:i)} = \alpha_2^{(i)}$, $D_{2,k-1}^{(i-1:i)} = \delta_2^{(i)}$,

$$M^{(i-1:i)} = \begin{pmatrix} \alpha_2^{(i-1)} & \delta_2^{(i-1)} \\ \delta_1^{(i)} & \alpha_1^{(i)} \end{pmatrix},$$

$$\mathbf{s}_0^{(i-1:i)} = \begin{pmatrix} \gamma_2^{(i-1)} \\ O \end{pmatrix}, \quad \mathbf{s}_1^{(i-1:i)} = \begin{pmatrix} \beta_2^{(i-1)} \\ O \end{pmatrix}, \quad \mathbf{s}_2^{(i-1:i)^T} = \begin{pmatrix} \gamma_2^{(i)} & \beta_2^{(i)} \end{pmatrix},$$

$$\mathbf{r}_0^{(i-1:i)^T} = \begin{pmatrix} \beta_1^{(i-1)} & \gamma_1^{(i-1)} \end{pmatrix}, \quad \mathbf{r}_1^{(i-1:i)} = \begin{pmatrix} O \\ \beta_1^{(i)} \end{pmatrix}, \quad \mathbf{r}_2^{(i-1:i)} = \begin{pmatrix} O \\ \gamma_1^{(i)} \end{pmatrix},$$

where the superindex $(i-1:i)$ means that the elements are stored in processors $i-1$ and i. Then each processor i applies factorization (5) to its corresponding block (with one obvious difference for the factorization applied by processors 1, 2, $p-1$ and p).

Similarly to Section 2.1 where the corner blocks in factorization (5) constituted the reduced system (8), the corner blocks of factorizations of the matrices (9)–(11) create a new reduced system with $p/2-2$ blocks.

In the second step, each processor i exchanges its corner blocks with processor $i-2$, for $i = 3, 4, 7, 8, 11, 12, \ldots, p-1, p$. Then each processor i decomposes a 4×6 block matrix of the form (9) if $i > 5$ and $i < p-3$, and a 3×4 block matrix of the form (10) or (11) otherwise. After $r-1$ steps of this process each processor decomposes a 3×4 block matrix of the form (10) or (11), and finally, after r steps, each processor contains the same 2×2 block matrix $M^{(1:n)}$ that gives the first block components of the solution. Overall, in the proposed factorization algorithm for the reduced system after j steps of reduction, processors from $i+1$ to $i+2^j$, for $i = 0, 2^j, 2^{j+1}, \ldots$, contain the same components of the reduced system and perform the same operations. This choice reduces the total amount of data communications.

The solution of the reduced system needs the same transmissions as the factorization of the reduced system (where appropriate parts of the right-hand side are updated). Each processor communicates with all its neighbors in the hypercube structure. After r steps, when the 2×2 linear system with the coefficient matrix $M^{(1:n)}$ is solved, the first two components of the reduced system (corresponding to the central rows of R in (8)) are obtained. From this point on, each processor may calculate its components of the reduced system and its blocks of the solution without any other communication (see Section 3 for further details).

3. THE PARALLEL ALGORITHM FOR A HYPERCUBE

The following presents the algorithms for all steps of solving the linear system $M\mathbf{x} = \mathbf{b}$, where M is the ABD matrix in (1) with $m = kp-2$ internal blocks on a hypercube with $p = 2^r$ processors. It is also assumed that

$$\mathbf{x}^T = (\mathbf{x}_{2,0} \quad \mathbf{x}_{1,1} \quad \mathbf{x}_{2,1} \quad \dots \quad \mathbf{x}_{1,m} \quad \mathbf{x}_{2,m} \quad \mathbf{x}_{1,m+1})$$
$$\mathbf{b}^T = (\mathbf{b}_{2,0} \quad \mathbf{b}_{1,1} \quad \mathbf{b}_{2,1} \quad \dots \quad \mathbf{b}_{1,m} \quad \mathbf{b}_{2,m} \quad \mathbf{b}_{1,m+1}).$$

Algorithms are presented in a pseudo-programming language, where all the block operations should be substituted by calls to BLAS subroutines. We distinguish the parallel factorization phase from the linear system solution phase. We also present separately algorithms for matrix M and for the reduced system.

3.1. Algorithms for Matrix M

In the parallel factorization phase the coefficient matrix is partitioned as suggested in (2). For simplicity, the superindex (i) is neglected for all elements (obviously for $i = 1$ and $i = p$ certain operations are not performed). The instructions inside the "forall" cycle are executed in parallel on different processors.

forall $i = 1 : p$

for $j = 1 : k-2$ (suppose $\widetilde{A}_{2,0} = A_{2,0}$ and $\widetilde{D}_{2,0} = D_{2,0}$)

$$\begin{pmatrix} L_{2,j-1} & \\ \widehat{D}_{1,j} & \widetilde{A}_{1,j} \\ \widehat{C}_{2,j} & \widetilde{B}_{2,j} \end{pmatrix} \begin{pmatrix} U_{2,j-1} & \widehat{D}_{2,j-1} \\ & I \end{pmatrix} Q_j = \begin{pmatrix} \widetilde{A}_{2,j-1} & \widetilde{D}_{2,j-1} \\ D_{1,j} & A_{1,j} \\ C_{2,j} & B_{2,j} \end{pmatrix}$$

$$P_j \begin{pmatrix} L_{1,j} & \\ \widehat{B}_{2,j} & I \end{pmatrix} \begin{pmatrix} U_{1,j} & \widehat{B}_{1,j} & \widehat{C}_{1,j} \\ & \widetilde{A}_{2,j} & \widetilde{D}_{2,j} \end{pmatrix} = \begin{pmatrix} \widetilde{A}_{1,j} & B_{1,j} & C_{1,j} \\ \widetilde{B}_{2,j} & A_{2,j} & D_{2,j} \end{pmatrix}$$

end

$$\begin{pmatrix} L_{2,k-2} & \\ \widehat{D}_{1,k-1} & L_{1,k-1} \end{pmatrix} \begin{pmatrix} U_{2,k-2} & \widehat{D}_{2,k-2} \\ & U_{1,k-1} \end{pmatrix} Q_{k-1} = \begin{pmatrix} \widetilde{A}_{2,k-2} & \widetilde{D}_{2,k-2} \\ D_{1,k-1} & A_{1,k-1} \end{pmatrix}$$

$$\begin{pmatrix} \widetilde{V}_{2,k-2} & \widetilde{V}_{1,k-1} \end{pmatrix} \begin{pmatrix} U_{2,k-2} & \widehat{D}_{2,k-2} \\ & U_{1,k-1} \end{pmatrix} Q_{k-1} = (C_{2,k-1} \quad B_{2,k-1})$$

$$V_{1,k-1} L_{1,k-1} = \widetilde{V}_{1,k-1}$$

$$(V_{1,k-2} \quad V_{2,k-2}) P_{k-2} \begin{pmatrix} L_{1,k-2} & \\ \widehat{B}_{2,k-2} & L_{2,k-2} \end{pmatrix}$$
$$= (O \quad \widetilde{V}_{2,k-2} - V_{1,k-1} \widehat{D}_{1,k-1})$$

for $j = k-3 : -1 : 1$

$$(V_{1,j} \quad V_{2,j})P_j \begin{pmatrix} L_{1,j} & \\ \widehat{B}_{2,j} & L_{2,j} \end{pmatrix}$$

$$= (O \quad -V_{1,j+1}\widehat{D}_{1,j+1} - V_{2,j+1}\widehat{C}_{2,j+1})$$

end

$V_{2,0}L_{2,0} = -V_{1,1}\widehat{D}_{1,1} - V_{2,1}\widehat{C}_{2,1}$

$$(\widetilde{\mathrm{W}}_{2,0} \quad \widetilde{\mathrm{W}}_{1,1}) \begin{pmatrix} U_{2,0} & \widehat{D}_{2,0} \\ & U_{1,1} \end{pmatrix} Q_1 = (B_{1,0} \quad C_{1,0})$$

for $j = 2 : 2 : k-1$

$$(\widetilde{\mathrm{W}}_{2,j-1} \ \widetilde{\mathrm{W}}_{1,j}) \begin{pmatrix} U_{2,j-1} & \widehat{D}_{2,j-1} \\ & U_{1,j} \end{pmatrix} Q_j = -\widetilde{\mathrm{W}}_{1,j-1}(\widehat{B}_{1,j-1} \quad \widehat{C}_{1,j-1})$$

end

$W_{1,k-1}L_{1,k-1} = \widetilde{\mathrm{W}}_{1,k-1}$

for $j = k-2 : -1 : 1$ (suppose $W_{2,k-1} = O$)

$$(W_{1,j} \ W_{2,j})P_j \begin{pmatrix} L_{1,j} & \\ \widehat{B}_{2,j} & L_{2,j} \end{pmatrix}$$

$$= (O \quad \widetilde{\mathrm{W}}_{2,j} - W_{1,j+1}\widehat{D}_{1,j+1} - W_{2,j+1}\widehat{C}_{2,j+1})$$

end

$W_{2,0}L_{2,0} = \widetilde{\mathrm{W}}_{2,0} - W_{1,1}\widehat{D}_{1,1} - W_{2,1}\widehat{C}_{2,1}$

$(\delta_1 \quad \alpha_1) = (D_{1,0} \quad A_{1,0}) - W_{2,0}(C_{2,0} \quad B_{2,0})$

$(\alpha_2 \quad \delta_2) = (A_{2,k-1} \quad D_{2,k-1}) - V_{1,k-1}(B_{1,k-1} \quad C_{1,k-1})$

$(\gamma_2 \quad \beta_2) = -V_{2,0}(C_{2,0} \quad B_{2,0})$

$(\beta_1 \ \gamma_1) = -W_{1,k-1}(B_{1,k-1} \quad C_{1,k-1})$

end

< factorization of the reduced system—see Section 3.2 >

The following is the algorithm for the solution of the previously factorized ABD linear system. Again we have posed $\mathbf{b}_{l,j}^{(i)} = \mathbf{b}_{l,(i-1)k+j}$ and $\mathbf{x}_{l,j}^{(i)} = \mathbf{x}_{l,(i-1)k+j}$, and neglected the superindex (i). First, appropriate parts of $\mathbf{b}$ are updated, then the reduced system is solved, and finally the solution to the original problem is calculated.

forall $i = 1 : p$

for $j = 1 : k-1$

$\mathbf{b}_{1,0} = \mathbf{b}_{1,0} - W_{2,j-1}\mathbf{b}_{2,j-1} - W_{1,j}\mathbf{b}_{1,j}$

$\mathbf{b}_{2,k-1} = \mathbf{b}_{2,k-1} - V_{2,j-1}\mathbf{b}_{2,j-1} - V_{1,j}\mathbf{b}_{1,j}$

end

end

< solution of the reduced system—see Section 3.2 >

$\Rightarrow \mathbf{x}_{2,-1}, \mathbf{x}_{1,0}, \mathbf{x}_{2,k-1}, \mathbf{x}_{1,k}$

forall $i = 1 : p$

$L_{2,0}\mathbf{x}_{2,0} = \mathbf{b}_{2,0} - C_{2,0}\mathbf{x}_{2,-1} - B_{2,0}\mathbf{x}_{1,0}$

for $j = 1 : k-2$

$$P_j \begin{pmatrix} L_{1,j} & \\ \widehat{B}_{2,j} & L_{2,j} \end{pmatrix} \begin{pmatrix} \mathbf{x}_{1,j} \\ \mathbf{x}_{2,j} \end{pmatrix} = \begin{pmatrix} \mathbf{b}_{1,j} \\ \mathbf{b}_{2,j} \end{pmatrix} - \begin{pmatrix} \widehat{D}_{1,j} \\ \widehat{C}_{2,j} \end{pmatrix} \mathbf{x}_{2,j-1}$$

end

$L_{1,k-1}\mathbf{x}_{1,k-1} = \mathbf{b}_{1,k-1} - B_{1,k-1}\mathbf{x}_{2,k-1} - C_{1,k-1}\mathbf{x}_{1,k}$

$$\begin{pmatrix} U_{2,k-2} & \widehat{D}_{2,k-2} \\ & U_{1,k-1} \end{pmatrix} Q_{k-1} \begin{pmatrix} \mathbf{x}_{2,k-2} \\ \mathbf{x}_{1,k-1} \end{pmatrix} = \begin{pmatrix} \mathbf{x}_{2,k-2} \\ \mathbf{x}_{1,k-1} \end{pmatrix}$$

for $j = k-2 : -1 : 1$

$$\begin{pmatrix} U_{2,j-1} & \widehat{U}_{2,j-1} \\ & U_{1,j} \end{pmatrix} Q_j \begin{pmatrix} \mathbf{x}_{2,j-1} \\ \mathbf{x}_{1,j} \end{pmatrix} = \begin{pmatrix} \mathbf{x}_{2,j-1} \\ \mathbf{x}_{1,j} - \widehat{B}_{1,j}\mathbf{x}_{2,j} - \widehat{C}_{1,j}\mathbf{x}_{1,j+1} \end{pmatrix}$$

end

end

3.2. Algorithms for the Reduced System

As we already observed in Section 2.2, in the factorization phase, and in the solution of the reduced system phase, all processors always have the same workload and, for $j = 1, \ldots, \log_2 p$, at the jth step processors from $i - 2^j + 1$ to i, for $i = 2^j, 2^{j+1}, 2^{j+2}, \ldots$ perform the same operations. In the following algorithm we assume that $r = i - 2^j$, $s = i - 2^{j-1}$, and $t = i$. Each processor l performs an operation if and only if the index l is contained in the range $r + 1 : t$. If $r = 0$ and/or $t = p$, then appropriate operations are not performed.

The following is the algorithm for the factorization of the reduced system:

for $j = 1 : \log_2 p$

forall $i = 2^j : 2^j : p$

forall $l = i - 2^{j-1} + 1 : i$

send/receive operations between processors l and $l - 2^{j-1}$ in order to obtain the same matrix on both processors

$$\begin{pmatrix} \delta_1^{(r+1:s)} & \alpha_1^{(r+1:s)} & \beta_1^{(r+1:s)} & \gamma_1^{(r+1:s)} & & & \\ \gamma_2^{(r+1:s)} & \beta_2^{(r+1:s)} & \alpha_2^{(r+1:s)} & \delta_2^{(r+1:s)} & & & \\ & & \delta_1^{(s+1:t)} & \alpha_1^{(s+1:t)} & \beta_1^{(s+1:t)} & \gamma_1^{(s+1:t)} \\ & & \gamma_2^{(s+1:t)} & \beta_2^{(s+1:t)} & \alpha_2^{(s+1:t)} & \delta_2^{(s+1:t)} \end{pmatrix}$$

end

$$p^{(r+1:t)} \begin{pmatrix} l_{1,1}^{(r+1:t)} & \\ l_{2,1}^{(r+1:t)} & l_{2,2}^{(r+1:t)} \end{pmatrix} \begin{pmatrix} u_{1,1}^{(r+1:t)} & u_{1,2}^{(r+1:t)} \\ & u_{2,2}^{(r+1:t)} \end{pmatrix}$$

$$= \begin{pmatrix} \alpha_2^{(r+1:s)} & \delta_2^{(r+1:s)} \\ \delta_1^{(s+1:t)} & \alpha_1^{(s+1:t)} \end{pmatrix}$$

$$\begin{pmatrix} w_1^{(r+1:t)} & w_2^{(r+1:t)} \end{pmatrix} \begin{pmatrix} \alpha_2^{(r+1:s)} & \delta_2^{(r+1:s)} \\ \delta_1^{(s+1:t)} & \alpha_1^{(s+1:t)} \end{pmatrix} = \begin{pmatrix} \beta_1^{(r+1:s)} & \gamma_1^{(r+1:s)} \end{pmatrix}$$

$$\begin{pmatrix} v_1^{(r+1:t)} & v_2^{(r+1:t)} \end{pmatrix} \begin{pmatrix} \alpha_2^{(r+1:s)} & \delta_2^{(r+1:s)} \\ \delta_1^{(s+1:t)} & \alpha_1^{(s+1:t)} \end{pmatrix} = \begin{pmatrix} \gamma_2^{(s+1:t)} & \beta_2^{(s+1:t)} \end{pmatrix}$$

$$\begin{pmatrix} \delta_1^{(r+1:t)} & \alpha_1^{(r+1:t)} \end{pmatrix}$$

$$= \begin{pmatrix} \delta_1^{(r+1:s)} & \alpha_1^{(r+1:s)} \end{pmatrix} - w_1^{(r+1:t)} \begin{pmatrix} \gamma_2^{(r+1:s)} & \beta_2^{(r+1:s)} \end{pmatrix}$$

$$\begin{pmatrix} \alpha_2^{(r+1:t)} & \delta_2^{(r+1:t)} \end{pmatrix}$$

$$= \begin{pmatrix} \alpha_2^{(s+1:t)} & \delta_2^{(s+1:t)} \end{pmatrix} - v_2^{(r+1:t)} \begin{pmatrix} \beta_1^{(s+1:t)} & \gamma_1^{(s+1:t)} \end{pmatrix}$$

$$\begin{pmatrix} \gamma_2^{(r+1:t)} & \beta_2^{(r+1:t)} \end{pmatrix} = -v_1^{(r+1:t)} \begin{pmatrix} \gamma_2^{(r+1:s)} & \beta_2^{(r+1:s)} \end{pmatrix}$$

$$\begin{pmatrix} \beta_1^{(r+1:t)} & \gamma_1^{(r+1:t)} \end{pmatrix} = -w_2^{(r+1:t)} \begin{pmatrix} \beta_1^{(s+1:t)} & \gamma_1^{(s+1:t)} \end{pmatrix}$$

end

end

At the beginning of the reduced system solution phase, each processor i contains two blocks of the right-hand side $\mathbf{b}_{1,0}^{(i)} = \mathbf{b}_{1,(i-1)k}$ and $\mathbf{b}_{2,k-1}^{(i)} = \mathbf{b}_{2,ik-1}$. At the end the same processor obtains the following blocks of the solution:

$$\mathbf{x}_{2,-1}^{(i)} = \mathbf{x}_{2,(i-1)k-1}, \quad \mathbf{x}_{1,0}^{(i)} = \mathbf{x}_{1,(i-1)k}, \quad \mathbf{x}_{2,k-1}^{(i)} = \mathbf{x}_{2,ik-1}, \quad \mathbf{x}_{1,k}^{(i)} = \mathbf{x}_{1,ik}.$$

Again it is supposed that $\mathbf{x}_{2,-1}$, $\mathbf{x}_{1,0}$, $\mathbf{x}_{2,m+1}$, and $\mathbf{x}_{1,m+2}$ are null vectors and that operations involving these blocks are not performed. The solution of the reduced system is obtained as follows:

for $j = 1 : \log_2 p - 1$
  forall $i = 2^j : 2^j : p$
    forall $l = i - 2^{j-1} + 1 : i$
      send/receive operations between processors l and $l - 2^{j-1}$ to obtain the same blocks on both processors
      $\mathbf{b}_{1,rk}$, $\mathbf{b}_{2,sk-1}$, $\mathbf{b}_{1,sk}$, $\mathbf{b}_{2,tk-1}$
    end

$$\mathbf{b}_{1,rk} = \mathbf{b}_{1,rk} - w_1^{(r+1:t)}\mathbf{b}_{2,sk-1} - w_2^{(r+1:t)}\mathbf{b}_{1,sk}$$
$$\mathbf{b}_{2,tk-1} = \mathbf{b}_{2,tk-1} - v_1^{(r+1:t)}\mathbf{b}_{2,sk-1} - v_2^{(r+1:t)}\mathbf{b}_{1,sk}$$

end

end

$$\begin{pmatrix} \alpha_2^{(1:p/2)} & \delta_2^{(1:p/2)} \\ \delta_1^{(p/2+1:p)} & \alpha_1^{(p/2+1:p)} \end{pmatrix} \begin{pmatrix} \mathbf{x}_{2,(p/2)k-1} \\ \mathbf{x}_{1,(p/2)k} \end{pmatrix} = \begin{pmatrix} \mathbf{b}_{2,(p/2)k-1} \\ \mathbf{b}_{1,(p/2)k} \end{pmatrix}$$

for $j = \log_2 p - 1 : -1 : 1$

forall $i = 2^j : 2^j : p$

$$\begin{pmatrix} \alpha_2^{(r+1:s)} & \delta_2^{(r+1:s)} \\ \delta_1^{(s+1:t)} & \alpha_1^{(s+1:t)} \end{pmatrix} \begin{pmatrix} \mathbf{x}_{2,sk-1} \\ \mathbf{x}_{1,sk} \end{pmatrix}$$
$$= \begin{pmatrix} \mathbf{b}_{2,sk-1} - \gamma_2^{(r+1:s)}\mathbf{x}_{2,rk-1} - \beta_2^{(r+1:s)}\mathbf{x}_{1,rk} \\ \mathbf{b}_{1,sk} - \beta_1^{(s+1:t)}\mathbf{x}_{2,tk-1} - \gamma_1^{(s+1:t)}\mathbf{x}_{1,tk} \end{pmatrix}$$

end

end

4. ARITHMETIC COMPLEXITY

In this section we analyze the arithmetic complexity functions associated with the ABD algorithm of the previous section and compare the obtained results with those reported in [13]. For simplicity, we assume that we want to solve a system of n first-order BVPs with q initial and $n-q$ final conditions on a parallel computer with p processors. The discretization leads to the ABD system with m internal blocks (an interval is divided into $m+1$ subintervals). The computational cost thus depends on the following parameters:

- n, number of first-order ODEs;
- q, number of initial conditions;
- m, number of internal blocks;
- p, number of processors.

Finally, assume that $m = kp-2$; that is, in each processor $k = (m+2)/p$ blocks are stored.

4.1. Arithmetic Operation Count

The factorization $M^{(i)} = L^{(i)}U^{(i)}$ (Section 2.1) requires $(\frac{5}{3}n^3 + qn^2 - q^2n - \frac{3}{2}n^2 + qn - \frac{1}{6}n)(m+2)/p - \frac{8}{3}n^3 - 2qn^2 + 2q^2n + \frac{5}{2}n^2 - 2qn + \frac{1}{6}n$ arithmetical operations, the construction of the fill-in vectors $\mathbf{v}^{(i)}$ and $\mathbf{w}^{(i)}$ in (6) requires $(4n^3 - 3qn^2 + 2q^2n - 2n^2)(m+2)/p - 6n^3 + 6qn^2 - 4q^2n +$

TABLE 1

SIZE OF THE REDUCED SYSTEM ON 16 PROCESSORS AFTER j STEPS OF REDUCTIONS, FOR $j = 1, \ldots, 4$

Block form	Size	$j=1$	$j=2$	$j=3$	$j=4$
3×4 (see (10))	$(n+q) \times 2n$	1:2	1:4	1:8	
4×6 (see (9))	$2n \times 3n$	3:14	5:12		
3×4 (see (11))	$(2n-q) \times 2n$	15:16	13:16	9:16	
2×2	$n \times n$				1:16

TABLE 2

ARITHMETICAL COMPLEXITY OF THE ABD PARALLEL ALGORITHM

Factorization	$\left(\frac{17}{3}n^3 - 2qn^2 + q^2n - \frac{7}{2}n^2 + qn - \frac{1}{6}n\right)(m+2)/p$
Fact. red. system	$\left(\frac{14}{3}n^3 - \frac{5}{2}n^2 - \frac{1}{6}n\right)\log_2 p$
Solution	$(6n^2 - n)(m+2)/p$
Sol. red. system	$(6n^2 - n)\log_2 p$

$3n^2$ arithmetical operations, and the calculation of the elements that will constitute the reduced system requires $2n^3 - n^2$ arithmetical operations.

The number of operations of one generic step of the factorization of the reduced matrix is easily obtained by setting $(m+2)/p = 2$. The last two steps of factorization require a smaller number of operations since the matrices involved are smaller (see Table 1). The $(\log_2 p - 1)$th step requires $\frac{8}{3}n^3 - \frac{3}{2}n^2 - \frac{1}{6}n$ operations, while the final step requires $\frac{2}{3}n^3 - \frac{1}{2}n^2 - \frac{1}{6}n$ operations.

The solution step of the algorithm requires the solution of the linear system with coefficient matrices $L^{(i)}$ and $U^{(i)}$ $((4n^2 - n)(m+2)/p - 6n^2 + n$ operations) and the updating of the right-hand side $(2n^2(m+2)/p$ operations). For $j = 1, \ldots, \log_2 p - 2$, a generic step j of the solution of the reduced system requires $6n^2 - n$ operations, the $(\log_2 p - 1)$th step requires $4n^2 - n$ operations, and the final step $2n^2 - n$ operations. Table 2 summarizes the computational cost of the four phases of the algorithm. The total computational cost is

$$\left(\frac{17}{3}n^3 - 2qn^2 + q^2n + \frac{5}{2}n^2 + qn - \frac{7}{6}n\right)(m+2)/p + \left(\frac{14}{3}n^3 - \frac{7}{2}n^2 - \frac{7}{6}n\right)\log_2 p - \frac{38}{3}n^3 + 4qn^2 - 2q^2n - \frac{9}{2}n^2 - 2qn + \frac{7}{6}n. \tag{12}$$

Observe that the two extreme cases of the arithmetical complexity occur for $q = n - 1$ and $q = n/2$ (if $q < n/2$ the ABD system can be reversed;

observe also that the arithmetical complexity function (12) is asymmetric with respect to q). Assuming now that n is large enough that n^3 term dominates the remaining terms, the arithmetical complexity functions for the two extreme cases are

$$\left(\frac{14}{3}n^3\right)(m+2)/p + \left(\frac{14}{3}n^3\right)\log_2 p - \frac{32}{3}n^3 \qquad \text{for } q = n-1 \qquad (13)$$

and

$$\left(\frac{59}{12}n^3\right)(m+2)/p + \left(\frac{14}{3}n^3\right)\log_2 p - \frac{67}{6}n^3 \qquad \text{for } q = n/2. \qquad (14)$$

As already mentioned, data transmissions are required only in the phases involving the reduced system. In the factorization step a $n \times 2n$ block is transmitted in all but the last step, when a block of size $q \times n$ is transmitted by half of the processors and a block of size $(n-q) \times n$ by the others. In the solution step a vector of size n is transmitted at each step, except for the last where a block of size q or $n-q$ is transmitted.

Hence the total cost of transmission is (assuming that $t(m)$ is the time needed for one transmission of m elements):

$$(\log_2 p - 1)(t(2n^2) + t(n)) + \max(t(nq) + t(q), t(n^2 - nq) + t(n-q)).$$

4.2. Comparisons with Other Algorithms

We compare the arithmetical complexity of the algorithm presented above with the original algorithm by Paprzycki and Gladwell [13] and the best existing sequential algorithm by Varah [16]. The notation remains the same as above and only the highest order terms are included. The arithmetical complexity functions for the two extreme cases of the parallel algorithm proposed in [13] are respectively:

$$\left(\frac{29}{3}n^3\right)(m+2)/p + \left(\frac{17}{3}n^3\right)p - \frac{52}{3}n^3 \qquad \text{for } q = n-1$$

and

$$\left(\frac{119}{12}n^3\right)(m+2)/p + \left(\frac{71}{12}n^3\right)p - \frac{56}{3}n^3 \qquad \text{for } q = n/2.$$

Assuming (as is most often the case in the computational practice) that m is large, p is fixed, and $m \gg p$, the first term in the above functions dominates the remaining two terms. In this case the advantage of the new algorithm is $5n^3(m+2)/p$ arithmetical operations.

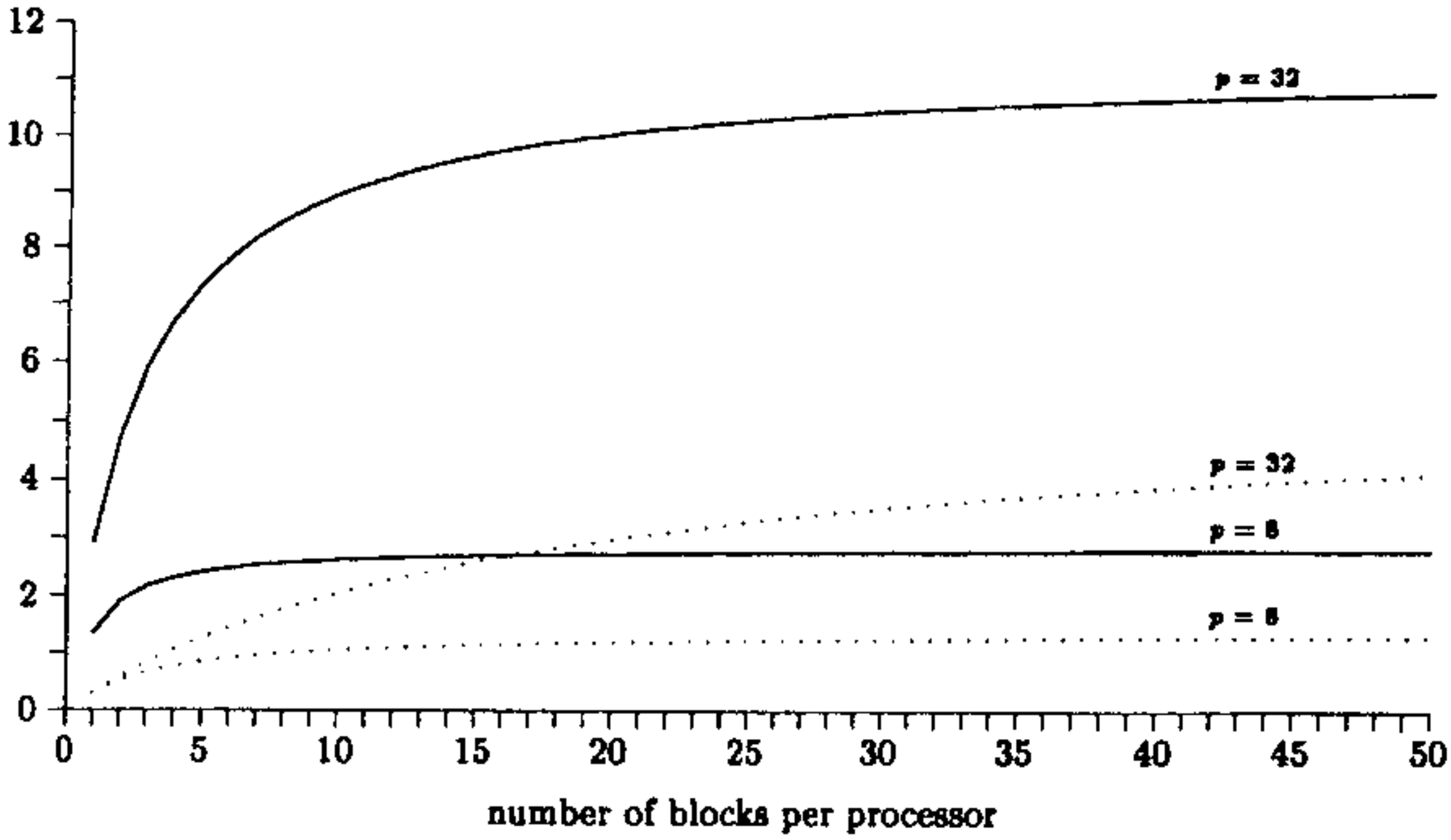

FIG. 4. Theoretical speed-up of the proposed algorithm (solid lines) and of the algorithm in [13] (dotted lines) for $p = 8$ and $p = 32$ and for increasing number of internal blocks m.

Moreover, the algorithm in [13] requires the transmission to a single processor of approximately $2pn^2$ elements in order to set up the reduced system.

The arithmetical complexity of the two extreme cases of the sequential solver are:

$$\left(\frac{5}{3}n^3\right)m + \frac{2}{3}n^3 \qquad \text{for } q = n - 1$$

and

$$\left(\frac{23}{12}n^3\right)m + \frac{2}{3}n^3 \qquad \text{for } q = n/2.$$

Using these functions a theoretical speed-up of the proposed algorithm for $q = n - 1$ has the following form (for the other extreme case only the appropriate constants change):

$$\frac{5m + 2}{14(m + 2)/p + 14\log_2 p - 32}.$$

Figure 4 shows the theoretical speed-up of the proposed algorithm and of the algorithm in [13] for $q = n - 1$, for $p = 8$ and $p = 32$ processors for increasing number of internal blocks stored per processor. A number of observations can be made:

- speed-up does not depend on n (at least when the communication costs are negligible),
- the optimal number of processors (the number of processors for which the maximum speed-up is achieved) is $O(m)$,
- if the optimal number of processors is used the proposed algorithm has a theoretical speed-up $O(m/\log(m))$,
- the increase in m leads to speed-up increase,
- for a fixed p and large m the maximal speed-up is slightly higher than $p/3$, whereas for the original algorithm the maximal theoretical speed-up was approximately $p/6$.

4.3. Memory Requirement

Let us suppose that the factorization step and the linear system solution are performed separately (in this case the memory requirement is larger). Each processor requires to store its portion of the ABD matrix ($2(m+2)n^2/p$ elements) and the fill-in vectors ($((m+2)/p-1)n^2$ elements). Moreover, at each step, each processor must store matrices obtained from the reduction (each matrix has $3n^2$ elements). Hence the per-processor memory requirement is

$$(3(m+2)/p + 3\log_2 p - 1)n^2$$

and the total memory requirement is

$$(3m + 3p\log_2 p - p)n^2.$$

In comparison the per-processor memory requirement of the original algorithm is

$$(4(m+2)/p + 2)n^2$$

and its total memory requirement (including the memory for the reduced system which is treated separately) is

$$(4m + 4p + 5)n^2,$$

whereas the total memory requirement of the sequential algorithm is

$$(2m + 1)n^2.$$

Assuming again that p is fixed, m is large, and $m \gg p$ the per-processor gain from using the new algorithm is approximately $(m+2)n^2/p$ elements. At the same time the total memory requirements of the three algorithms are approximately $2mn^2$ for the sequential algorithm, $3mn^2$ for the new algorithm, and $4mn^2$ for the original parallel algorithm.

5. CONCLUSIONS

A new algorithm for the solution of almost block diagonal systems has been introduced, its implementation on a hypercube has been proposed, and its arithmetical complexity has been presented and compared with that of other algorithms. It was shown that the new algorithm is well suited for distributed memory parallel computers and should outperform other algorithms proposed for the solution of the ABD system. In the near future we plan to implement the proposed algorithm to confirm the theoretical results presented here.

REFERENCES

1 P. Amodio, L. Brugnano, and T. Politi, *Parallel factorizations for tridiagonal matrices, SIAM J. Numer. Anal.* 30:813–823 (1993).

2 P. Amodio and N. Mastronardi, *A parallel version of the cyclic reduction algorithm on a hypercube, Parallel Comput.* 19:1273–1281 (1993).

3 E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen, *LAPACK Users' Guide,* SIAM, Philadelphia, 1993.

4 U. M. Ascher and S. Y. P. Chan, *On parallel methods for boundary value ODEs, Computing* 46:1-17 (1991).

5 J. C. Diaz, G. Fairweather, and P. Keast, *Fortran packages for solving certain almost block diagonal linear systems by modified alternate row and column elimination, ACM Trans. Math. Software* 9:358–375 (1983).

6 J. Dongarra, J. Du Croz, and S. Hammarling, *A set of level 3 basic linear algebra subprograms,* Technical report ANL-MCS-TM57, Argonne National Laboratory, 1988.

7 J. Dongarra, J. Du Croz, S. Hammarling, and R. J. Hanson, *An extended set of FORTRAN basic linear algebra subprograms, ACM Trans. Math. Software* 14:1-17 (1988).

8 J. Dongarra and L. Johnsson, *Solving banded systems on a parallel processor, Parallel Comput.* 5:219–246 (1987).

9 J. Dongarra and A. H. Sameh, *On some parallel banded system solvers, Parallel Comput.* 1:223-235 (1984).

10 I. Gladwell and M. Paprzycki, *Parallel solution of almost block diagonal systems on the CRAY Y-MP using level 3 BLAS, J. Comput. Appl. Math.* 45:181–189 (1993).

11 S. L. Johnsson, *Solving narrow banded system on ensemble architectures, ACM Trans. Math. Software* 11:271–288 (1985).

12 C. L. Lawson, R. J. Hanson, D. R. Kincaid, and F. T. Krogh, *Basic linear algebra subprograms for FORTRAN usage, ACM Trans. Math. Software* 5:306–323 (1979).

13 M. Paprzycki and I. Gladwell, *Solving almost block diagonal systems on parallel computers, Parallel Comput.* 17:133–153 (1991).
14 K. Wright, *Parallel treatment of block-bidiagonal matrices in the solution of ordinary differential boundary value problems, J. Comput. Appl. Math.* 45:191–200 (1993).
15 S. Wright, *Stable parallel algorithms for two-point boundary value problems, SIAM J. Sci. Statist. Comput.* 13:742–764 (1992).
16 J. M. Varah, *Alternate row and column elimination for solving certain linear systems, SIAM J. Numer. Anal.* 13:71–75 (1976).

Received 28 November 1994; final manuscript accepted 26 June 1995