SIAM J. SCI. COMPUT.
Vol. 18, No. 1, pp. 56–68, January 1997

004

A CYCLIC REDUCTION APPROACH TO THE NUMERICAL SOLUTION OF BOUNDARY VALUE ODEs*

PIERLUIGI AMODIO† AND MARCIN PAPRZYCKI‡

Abstract. A parallel algorithm for solving linear systems that arise from the discretization of boundary value ODEs is described. It is a modification of a cyclic reduction algorithm that takes advantage of the almost block diagonal structure of the linear system. A stable modification of the original algorithm is also proposed. Numerical results on a distributed memory parallel computer with 32 processors are presented and discussed.

Key words. boundary value problems, ABD systems, parallel solvers

AMS subject classifications. 65L10, 15A23, 65Y05

PII. S1064827595283033

1. Introduction. Almost block diagonal (ABD) linear systems arise in various mathematical problems: in Chebyshev spectral decomposition on rectangular domains, in orthogonal spline collocation for elliptic problems, and, most importantly, in various discretizations of boundary value ordinary differential equations (BVP ODEs).

There are a variety of methods for high-performance solution of such systems. A level 3 BLAS-based [4] modification of the alternate row and column elimination algorithm proposed in [17] and implemented in [10] provides good performance on high-end workstations [8] as well as on vector computers [16]. For large block sizes an efficiency of 50% has been reported on a shared memory parallel computer with a limited number of processors [11]. Tearing-type methods can be applied to systems with a large number of relatively small blocks [3, 6, 15, 18, 19, 20]. Of them, the methods described in [6, 18] are suitable primarily for shared memory computers. The algorithms introduced in [3, 15, 19, 20] are suitable also for distributed memory computers. Of the latter methods, only the two algorithms introduced by S. Wright in [19, 20] are capable of solving linear systems arising from discretization of BVP ODEs with *nonseparated* boundary conditions (which are characterized by an additional corner block; see (2)).

The aim of this paper is to introduce a new cyclic reduction-based algorithm for parallel solution of such ABD systems. In section 2 the proposed algorithm is introduced. Its parallel implementation is discussed in section 3. The stabilized version of the algorithm is presented in section 4. Section 5 discusses the arithmetic complexity of the proposed algorithm. Section 6 contains the results of numerical experiments.

2. The cyclic reduction approach. Consider the BVP

$$\begin{aligned} \boldsymbol{y}' &= M(t)\boldsymbol{y} + \boldsymbol{q}(t), \qquad t \in [a,b], \quad \boldsymbol{y}, \boldsymbol{q} \in \mathbb{R}^n, M \in R^{n\times n}, \\ B_a\boldsymbol{y}(a) &+ B_b\boldsymbol{y}(b) = \boldsymbol{d}, \qquad \boldsymbol{d} \in \mathbb{R}^n. \end{aligned} \tag{1}$$

When discretized on a grid

$$a = t_0 < t_1 < \cdots < t_m = b,$$

*Received by the editors June 5, 1995; accepted for publication (in revised form) April 23, 1996. http://www.siam.org/journals/sisc/18-1/28722.html

†Dipartimento di Matematica, Università di Bari, Via E. Orabona 4, I-70125 Bari, Italy (labor@alphamath.dm.uniba.it).

‡Department of Mathematics and Computer Science, University of Texas of the Permian Basin, Odessa, TX 79762 (paprzycki_m@utpb.edu).

the following linear system arises:

$$\text{(2)}\qquad \begin{pmatrix} S_1 & R_1 & & & \\ & S_2 & R_2 & & \\ & & \ddots & \ddots & \\ & & & S_m & R_m \\ B_a & & & & B_b \end{pmatrix} \begin{pmatrix} \boldsymbol{y}_0 \\ \boldsymbol{y}_1 \\ \vdots \\ \boldsymbol{y}_m \end{pmatrix} = \begin{pmatrix} \boldsymbol{f}_1 \\ \boldsymbol{f}_2 \\ \vdots \\ \boldsymbol{f}_m \\ \boldsymbol{d} \end{pmatrix}.$$

Note that the same general system arises from using multiple shooting, finite differences as well as collocation methods, and the only difference will be in the particular block sizes. For simplicity of notation, we will assume that a finite difference scheme was used to discretize the BVP. Let

$$S_i \quad \begin{pmatrix} C_{1,i} & B_{1,i} \\ D_{1,i} & A_{1,i} \end{pmatrix}, \qquad R_i = \begin{pmatrix} A_{2,i} & D_{2,i} \\ B_{2,i} & C_{2,i} \end{pmatrix},$$

where $C_{1,i}$ and $A_{2,i}$ are $q \times q$, $A_{1,i}$ and $C_{2,i}$ are $(n-q) \times (n-q)$, $D_{1,i}^T$, $B_{1,i}$ $B_{2,i}^T$, and $D_{2,i}$ are $q \times (n-q)$, q is the number of initial conditions when the boundary conditions are separated or any nonnegative integer less than or equal to n otherwise (usually $q \quad n/2$ can be used; see also section 4). Moreover, let

$$B_a \quad \begin{pmatrix} A_{2,0} & D_{2,0} \\ B_{2,m+1} & C_{2,m+1} \end{pmatrix}, \qquad B_b = \begin{pmatrix} C_{1,0} & B_{1,0} \\ D_{1,m+1} & A_{1,m+1} \end{pmatrix},$$

where the blocks $A_{2,0}$, $D_{2,0}$, $B_{1,m+1}$ ..., have the same dimensions as the corresponding $A_{2,i}$, $D_{2,i}$, $B_{1,i}$ .... Permuting the appropriate equations in (2) we obtain the following matrix:

$$\text{(3)}\qquad \begin{pmatrix} A_{2,0} & D_{2,0} & & & & & C_{1,0} & B_{1,0} \\ D_{1,1} & A_{1,1} & B_{2,1} & C_{2,1} & & & & \\ C_{1,1} & B_{1,1} & A_{2,1} & D_{2,1} & & & & \\ & & D_{1,2} & A_{1,2} & B_{2,2} & C_{2,2} & & \\ & & C_{1,2} & B_{1,2} & A_{2,2} & D_{2,2} & & \\ & & & & & \ddots & & \\ & & & & & D_{1,m} & A_{1,m} & B_{2,m} & C_{2,m} \\ & & & & & C_{1,m} & B_{1,m} & A_{2,m} & D_{2,m} \\ B_{2,m+1} & C_{2,m+1} & & & & & D_{1,m+1} & A_{1,m+1} \end{pmatrix}$$

(observe that blocks $C_{1,0}$, $B_{1,0}$, $B_{2,m+1}$, and $C_{2,m+1}$ are null blocks in the case of separated boundary conditions) which can be expressed in a block tridiagonal form as

$$\text{(4)}\qquad \begin{pmatrix} A_0 & B_0 & & C_0 \\ C_1 & A_1 & \ddots & \\ & \ddots & \ddots & B_{m-1} \\ B_m & & C_m & A_m \end{pmatrix},$$

where

$$C_i = \begin{pmatrix} C_{1,i} & B_{1,i} \\ O & O \end{pmatrix}, \quad A_i = \begin{pmatrix} A_{2,i} & D_{2,i} \\ D_{1,i+1} & A_{1,i+1} \end{pmatrix}, \quad B_i = \begin{pmatrix} O & O \\ B_{2,i+1} & C_{2,i+1} \end{pmatrix}.$$

We may now apply the odd even cyclic reduction to matrix (4). We only require that the first and the last row of (4) be treated as even rows (the first row is considered row 0). For example, the first step of reduction applied to the permuted matrix (assuming m is even)

$$\left(\begin{array}{ccc|ccc} A_1 & & & C_1 & B_1 & \\ & \ddots & & & \ddots & \ddots \\ & & A_{m-1} & & & C_{m-1} \quad B_{m-1} \\ \hline B_0 & & & A_0 & & C_0 \\ C_2 & \ddots & & & A_2 & \\ & \ddots & B_{m-2} & & & \ddots \\ & & C_m & B_m & & A_m \end{array}\right),$$

gives the block tridiagonal matrix

$$\left(\begin{array}{cccc} \tilde{A}_0 & \tilde{B}_0 & & C_0 \\ \tilde{C}_2 & \tilde{A}_2 & \tilde{B}_2 & \\ & \tilde{C}_4 & \tilde{A}_4 & \ddots \\ & & \ddots & \ddots \quad \tilde{B}_{m-2} \\ B_m & & \tilde{C}_m & \tilde{A}_m \end{array}\right), \tag{5}$$

where

$$\begin{aligned} \tilde{A}_0 &= A_0 - B_0 A_1^{-1} C_1, \qquad \tilde{A}_m = A_m - C_m A_{m-1}^{-1} B_{m-1}, \\ \tilde{A}_{2i} &= A_{2i} - B_{2i} A_{2i+1}^{-1} C_{2i+1} - C_{2i} A_{2i-1}^{-1} B_{2i-1} \qquad \text{for } i = 1, \ldots, m/2 - 1, \\ \tilde{C}_{2i} &= -C_{2i} A_{2i-1}^{-1} C_{2i-1} \qquad \text{for } i = 1, \ldots, m/2, \\ \tilde{B}_{2i} &= -B_{2i} A_{2i+1}^{-1} B_{2i+1} \qquad \text{for } i = 0, \ldots, m/2 - 1. \end{aligned} \tag{6}$$

Note that during this step, instead of explicitly inverting blocks A_i (for odd i), we factorize these blocks by means of the LU decomposition with partial pivoting and then apply the linear system solution step to the columns of blocks B_i and C_i. Because B_i and C_i have zeroes in the first q and in the last $n - q$ rows, respectively, $B_{2i} A_{2i+1}^{-1}$ and $C_{2i} A_{2i-1}^{-1}$ will have zeroes in the same locations. Thus in matrix (5) blocks B_m and C_0 as well as the first q rows of $\tilde{A}_0$ and the last $n - q$ of $\tilde{A}_m$ remain unchanged. (This means that the blocks containing the boundary conditions are unchanged.) Moreover, blocks $\tilde{C}_{2i}$ and $\tilde{B}_{2i}$ maintain the same sparsity structure (the last $n - q$ rows of $\tilde{C}_{2i}$ and the first q of B_{2i} are zeros) as the initial C_{2i} and B_{2i}.

The same approach may be repeated and applied to (5) and after $\log_2 m$ steps, a $2n \times 2n$ matrix (that is a 2×2 block matrix if expressed by means of the blocks A_i, B_i, and C_i or a 4×4 block matrix if expressed by means of $A_{j,i}$, $B_{j,i}$, $C_{j,i}$, and $D_{j,i}$) is obtained and factorized using Gaussian elimination with partial pivoting. Note that the first q and the last $n - q$ rows of this final matrix still contain the boundary conditions of the initial differential problem (1). (The elements are just those of the first and the last block row of (3).)

The parallel implementation of this algorithm follows that of the cyclic reduction algorithm presented in [1]. Assuming $m = 2^r$, the number of processors $p = 2^s$, and $r \gg s$, the first $r - s$ steps of reduction do not require communication among the

processors. Only the last s steps require transmissions of matrices in the factorization step and of vectors in the linear system solution step (see the next section). It should be noted that in the case of a hypercube topology these transmissions will be between neighbor processors only (see [2]).

3. The parallel algorithm. The following algorithm represents the parallel implementation on p processors of the cyclic reduction algorithm applied to the solution of the linear system $M\boldsymbol{x} = \boldsymbol{b}$, where M is almost block diagonal and has $m = 2^r$ internal blocks (corresponding to the BVP (1) being discretized on m subintervals). We assume that

$$\boldsymbol{x}^T = \left(\begin{array}{ccccccc} \boldsymbol{x}_{2,0}^T & \boldsymbol{x}_{1,1}^T & \boldsymbol{x}_{2,1}^T & \dots & \boldsymbol{x}_{1,m}^T & \boldsymbol{x}_{2,m}^T & \boldsymbol{x}_{1,m+1}^T \end{array} \right),$$
$$\boldsymbol{b}^T = \left(\begin{array}{ccccccc} \boldsymbol{b}_{2,0}^T & \boldsymbol{b}_{1,1}^T & \boldsymbol{b}_{2,1}^T & \dots & \boldsymbol{b}_{1,m}^T & \boldsymbol{b}_{2,m}^T & \boldsymbol{b}_{1,m+1}^T \end{array} \right).$$

Algorithms are presented in a pseudoprogramming language, where all the block operations should be substituted by calls to appropriate BLAS subroutines [4]. We distinguish two phases: coefficient matrix factorization and linear system solution. The instructions inside the "forall" cycles are executed in parallel on different processors. The matrices obtained in the successive steps of the reduction will be represented by the superindex in parentheses (the original matrix has index 0).

FACTORIZATION

$s = 1$
for $j = 1 : \log_2 m$
  if $j > \log_2(m/p)$
    forall $k = s : 1 : s * 2 : p - s + 1$
      processor k sends to processor $k - s$ the block

$$\left(\begin{array}{cccc} D_{1,(k-1)*m/p+1}^{(j-1)} & A_{1,(k-1)*m/p+1}^{(j-1)} & B_{2,(k-1)*m/p-1}^{(j-1)} & C_{2,(k-1)*m/p+1}^{(j-1)} \\ C_{1,k*s*m/p}^{(j-1)} & B_{1,k*s*m/p}^{(j-1)} & A_{2,k*s*m/p}^{(j-1)} & D_{2,k*s*m/p}^{(j-1)} \end{array} \right)$$

    end
    $s = s * 2$
  end
  $l = 2^{j-1}$
  forall $k = 1 : s : p - s + 1$
    for $i = (k-1) * m/p + l : l * 2 : k * s * m/p - l$

$$P_i \left(\begin{array}{cc} L_{2,i} & \\ \hat{D}_{1,i+1} & L_{1,i+1} \end{array} \right) \left(\begin{array}{cc} U_{2,i} & \hat{D}_{2,i} \\ & U_{1,i+1} \end{array} \right) = \left(\begin{array}{cc} A_{2,i}^{(j-1)} & D_{2,i}^{(j-1)} \\ D_{1,i-1}^{(j-1)} & A_{1,i+1}^{(j-1)} \end{array} \right)$$

$$\left(\begin{array}{cc} V_{2,i} & W_{2,i} \end{array} \right) = \left(\begin{array}{cc} B_{2,i-l+1}^{(j-1)} & C_{2,i-l+1}^{(j-1)} \end{array} \right) \left(\begin{array}{cc} A_{2,i}^{(j-1)} & D_{2,i}^{(j-1)} \\ D_{1,i+1}^{(j-1)} & A_{1,i+1}^{(j-1)} \end{array} \right)^{-1}$$

$$\left(\begin{array}{cc} W_{1,i+1} & V_{1,i+1} \end{array} \right) = \left(\begin{array}{cc} C_{1,i+l}^{(j-1)} & B_{1,i+l}^{(j-1)} \end{array} \right) \left(\begin{array}{cc} A_{2,i}^{(j-1)} & D_{2,i}^{(j-1)} \\ D_{1,i+1}^{(j-1)} & A_{1,i+1}^{(j-1)} \end{array} \right)^{-1}$$

$$\left(\begin{array}{cc} D_{1,i-l+1}^{(j)} & A_{1,i-l+1}^{(j)} \end{array} \right) = \left(\begin{array}{cc} D_{1,i-l-1}^{(j-1)} & A_{1,i-l+1}^{(j-1)} \end{array} \right) - V_{2,i} \left(\begin{array}{cc} C_{1,i}^{(j-1)} & B_{1,i}^{(j-1)} \end{array} \right)$$

$$\left(\begin{array}{cc} A_{2,i+l}^{(j)} & D_{2,i+l}^{(j)} \end{array} \right) \quad \left(\begin{array}{cc} A_{2,i+l}^{(j-1)} & D_{2,i-l}^{(j-1)} \end{array} \right) - V_{1,i+1} \left(\begin{array}{cc} B_{2,i+1}^{(j-1)} & C_{2,i+1}^{(j-1)} \end{array} \right)$$

$$\begin{pmatrix} B^{(j)}_{2,i-l+1} & C^{(j)}_{2,i-l+1} \end{pmatrix} = -W_{2,i} \begin{pmatrix} B^{(j-1)}_{2,i+1} & C^{(j-1)}_{2,i+1} \end{pmatrix}$$

$$\begin{pmatrix} C^{(j)}_{1,i+l} & B^{(j)}_{1,i+l} \end{pmatrix} = -W_{1,i+1} \begin{pmatrix} C^{(j-1)}_{1,i} & B^{(j-1)}_{1,i} \end{pmatrix}$$

end

end

end

LU factorization with partial pivoting of the matrix

$$\begin{pmatrix} A^{(0)}_{2,0} & D^{(0)}_{2,0} & C^{(0)}_{1,0} & B^{(0)}_{1,0} \\ D^{(k)}_{1,1} & A^{(k)}_{1,1} & B^{(k)}_{2,1} & C^{(k)}_{2,1} \\ C^{(k)}_{1,m} & B^{(k)}_{1,m} & A^{(k)}_{2,m} & D^{(k)}_{2,m} \\ B^{(0)}_{2,m+1} & C^{(0)}_{2,m+1} & D^{(0)}_{1,m+1} & A^{(0)}_{1,m-1} \end{pmatrix}$$

LINEAR SYSTEM SOLUTION

```
s = 1
for j = 1 : log2 m
   if j > log2(m/p)
      forall k = s + 1 : s * 2 : p - s + 1
         processor k sends to processor k - s the vector
            ( b_{1,(k-1)*m/p+1} )
            ( b_{2,k*s*m/p}     )
      end
      s = s * 2
   end
   l = 2^(j-1)
   forall k = 1 : s : p - s + 1
      for i = (k-1) * m/p + l : l * 2 : k * s * m/p - l
         b_{1,i-l+1} = V_{2,i} b_{2,i} - W_{2,i} b_{1,i+1}
         b_{2,i+l}   = b_{2,i+l} - W_{1,i+1} b_{2,i} - V_{1,i+1} b_{1,i+1}
      end
   end
end
```

$$\begin{pmatrix} x_{2,0} \\ x_{1,1} \\ x_{2,m} \\ x_{1,m-1} \end{pmatrix} = \begin{pmatrix} A^{(0)}_{2,0} & D^{(0)}_{2,0} & C^{(0)}_{1,0} & B^{(0)}_{1,0} \\ D^{(k)}_{1,1} & A^{(k)}_{1,1} & B^{(k)}_{2,1} & C^{(k)}_{2,1} \\ C^{(k)}_{1,m} & B^{(k)}_{1,m} & A^{(k)}_{2,m} & D^{(k)}_{2,m} \\ B^{(0)}_{2,m+1} & C^{(0)}_{2,m+1} & D^{(0)}_{1,m+1} & A^{(0)}_{1,m+1} \end{pmatrix}^{-1} \begin{pmatrix} b_{2,0} \\ b_{1,1} \\ b_{2,m} \\ b_{1,m+1} \end{pmatrix}$$

```
for j = log2 m : -1 : 1
   l = 2^(j-1)
   forall k = 1 : s : p - s + 1
      for i = (k-1) * m/p + l : l * 2 : k * s * m/p - l
```

$$\begin{pmatrix} \boldsymbol{b}_{2,i} \\ \boldsymbol{b}_{1,i+1} \end{pmatrix} = \begin{pmatrix} \boldsymbol{b}_{2,i} \\ \boldsymbol{b}_{1,i+1} \end{pmatrix} - \begin{pmatrix} C_{1,i}^{(j-1)} \boldsymbol{x}_{2,i-l} + B_{1,i}^{(j-1)} \boldsymbol{x}_{1,i-l+1} \\ B_{2,i+1}^{(j-1)} \boldsymbol{x}_{2,i+l} + C_{2,i+1}^{(j-1)} \boldsymbol{x}_{1,i+l+1} \end{pmatrix}$$

$$\begin{pmatrix} \boldsymbol{x}_{2,i} \\ \boldsymbol{x}_{1,i+1} \end{pmatrix} = \begin{pmatrix} A_{2,i}^{(j-1)} & D_{2,i}^{(j-1)} \\ D_{1,i+1}^{(j-1)} & A_{1,i+1}^{(j-1)} \end{pmatrix}^{-1} \begin{pmatrix} \boldsymbol{b}_{2,i} \\ \boldsymbol{b}_{1,i+1} \end{pmatrix}$$

end

end

if $j > \log_2(m/p)$

forall k 1 : s : p $s + 1$

processor k sends to processor $k + s$ the vectors

$$\begin{pmatrix} \boldsymbol{x}_{2,k*s*m/p-l} \\ \boldsymbol{x}_{1,k*s*m/p-l+1} \end{pmatrix} \text{ and } \begin{pmatrix} \boldsymbol{x}_{2,k*s*m/p} \\ \boldsymbol{x}_{1,k*s*m/p+1} \end{pmatrix}$$

end

s $s/2$

end

end

The number of processors that effectively work at each step j of the algorithm is p/s and varies from p (for j 1, ..., $\log_2(p/m)$) to 1 (j $\log_2 m$).

There are three instances requiring data communication. One of them in the last $\log_2 p$ step of the factorization process ($j > \log_2(m/p)$) requires matrix transmissions while the others are performed during the linear system solution phase and require vector transmissions. Observe that it is possible to avoid the second instance of vector transmissions by performing bidirectional communications in the first two instances and always using all the processors available (see also |2, 3|). In this case some processors perform the same operations on the same data at the same time (hence the parallel arithmetic complexity does not increase) and in the last step of factorization all processors factorize the same 4×4 block matrix.

4. The stabilized algorithm. The stability analysis of cyclic reduction has been the object of several studies from the moment that this algorithm was presented for the first time at the end of the 1960s. A complete stability analysis was first introduced in |7|, where a modified version of the cyclic reduction algorithm (no matrix inversion is performed) is applied to the solution of the Poisson equation. A few years later Heller |12| derived more general stability conditions. In our case, stability problems are possible since the blocks

$$\begin{pmatrix} A_{2,i} & D_{2,i} \\ D_{1,i+1} & A_{1,i+1} \end{pmatrix} \tag{7}$$

may be ill conditioned or even singular. Moreover, even if for a given BVP we may choose a grid such that any block (7) of the original matrix (3) is nonsingular, it is quite difficult to prove that blocks obtained in the reduction process remain nonsingular. To overcome this problem, we can modify the previous algorithm slightly and thereby ensure stability.

The basis of the stabilized algorithm is that the proposed cyclic reduction is applied to a matrix that is essentially (apart from the boundary conditions) block bidiagonal. Consider the first step of reduction. Observe that for each $i = 1, 3, 5, \ldots, m-1$,

we apply the reduction process to the $2n \times 3n$ block

$$(8)\quad \begin{pmatrix} D_{1,i} & A_{1,i} & B_{2,i} & C_{2,i} & & \\ C_{1,i} & B_{1,i} & A_{2,i} & D_{2,i} & & \\ & & D_{1,i+1} & A_{1,i+1} & B_{2,i+1} & C_{2,i+1} \\ & & C_{1,i+1} & B_{1,i+1} & A_{2,i+1} & D_{2,i+1} \end{pmatrix} \cdot Q_i \begin{pmatrix} S_i & R_i & \\ & S_{i+1} & R_{i+1} \end{pmatrix}$$

(Q_i is a permutation matrix; see also section 2) in order to obtain the following $n \times 2n$ subblock (see section 3)

$$(9)\quad \begin{pmatrix} \tilde{D}_{1,i} & \tilde{A}_{1,i} & \tilde{B}_{2,i} & \tilde{C}_{2,i} \\ \tilde{C}_{1,i+1} & \tilde{B}_{1,i+1} & \tilde{A}_{2,i+1} & \tilde{D}_{2,i+1} \end{pmatrix}.$$

To stabilize the algorithm, instead of specifying the value of q a priori, we apply the LU factorization with partial pivoting to the block

$$(10)\quad \begin{pmatrix} R_i \\ S_{i+1} \end{pmatrix}.$$

This is always possible because block (10) has full column rank if the original matrix (3) is nonsingular, and hence we can always extract from it a nonsingular $n \times n$ block.

If q_i of the pivot elements ($0 \le q_i \le n$) were chosen among the rows of R_i, we now choose the permutation matrix Q_i in (8) such that the size of $A_{2,i}$ is $q_i \times q_i$ and the size of $A_{1,i+1}$ is $(n - q_i) \times (n \quad q_i)$, and both contain the pivot rows of R_i and S_{i+1}. This means that we must substitute the instructions inside the "for i" loop in the factorization phase of our algorithm in section 3 with the following:

$$P_i \begin{pmatrix} L_{2,i} & \\ \hat{D}_{1,i+1} & L_{1,i+1} \\ \bar{B}_{2,i\ \ l+1} & \bar{C}_{2,i\ \ l+1} \\ \bar{C}_{1,i+l} & \tilde{B}_{1,i+l} \end{pmatrix} \begin{pmatrix} U_{2,i} & \hat{D}_{2,i} \\ & U_{1,i+1} \end{pmatrix} \quad \begin{pmatrix} R_i^{(j\ \ 1)} \\ S_{i+l}^{(j\ \ 1)} \end{pmatrix}$$

$$\begin{pmatrix} V_{2,i} & W_{2,i} \\ W_{1,i} & V_{1,i} \end{pmatrix} \quad \begin{pmatrix} \bar{B}_{2,i-l+1} & \bar{C}_{2,i-l+1} \\ \tilde{C}_{1,i+l} & \tilde{B}_{1,i+l} \end{pmatrix} \begin{pmatrix} L_{2,i} & \\ \hat{D}_{1,i+1} & L_{1,i+1} \end{pmatrix}^{1}.$$

Define permutation matrices $Q_{i,1}$ and $Q_{i,2}$,

$$\begin{pmatrix} D_{1,i\ \ l+1} & A_{1,i-l+1} \\ C_{1,i} & B_{1,i} \\ B_{2,i+1} & C_{2,i+1} \\ A_{2,i+l} & D_{2,i+l} \end{pmatrix} = \begin{pmatrix} Q_{i,1} & \\ & Q_{i,2} \end{pmatrix} \begin{pmatrix} S_i^{(j\ \ 1)} \\ R_{i+l}^{(j-1)} \end{pmatrix}$$

$$S_{i+l}^{(j)} = \begin{pmatrix} D_{1,i-l+1} & A_{1,i\ \ l+1} \\ O & O \end{pmatrix} - \begin{pmatrix} V_{2,i} \\ W_{1,i+1} \end{pmatrix} \begin{pmatrix} C_{1,i} & B_{1,i} \end{pmatrix}$$

$$R_{i+l}^{(j)} = \begin{pmatrix} O & O \\ A_{2,i+l} & D_{2,i+l} \end{pmatrix} - \begin{pmatrix} W_{2,i} \\ V_{1,i+1} \end{pmatrix} \begin{pmatrix} B_{2,i+1} & C_{2,i+1} \end{pmatrix}.$$

Observe that in matrix (3), blocks $A_{1,i}$ and $A_{2,i+1}$, for $i \quad 1, 3, 5, \ldots, m-1$, are not square anymore. Because of the way that q_i is selected we can only derive that

$A_{1,i}$ has $n - q_i$ rows and $A_{2,i+1}$ has q_i rows. This means that also blocks A_i in (4), for $i = 2, 4, 6, \ldots, m$, have size $n \times (n - q_{i+1} - q_{i-1})$ and hence can be nonsquare, but this is not important since the reduction operations (6) can still be performed.

Moreover, at each step of reduction it is necessary to select a new value of q_i depending on the choice of the pivot elements in the LU factorization of the odd main diagonal blocks of the reduced matrices.

In order to illustrate the stabilization improvement, let us consider the following example. Assume that we want to factorize the following two block rows ("*" in the matrix represents a nonzero element with an unspecified value)

$$\begin{pmatrix} * & * & * & 1 & 1 & 0 & & & \\ * & * & * & 2 & 2 & 1 & & & \\ * & * & * & -1 & 1 & 1 & & & \\ & & & 0 & 5 & -5 & * & * & * \\ & & & 3 & 0 & -1 & * & * & * \\ & & & 2 & -1 & 1 & * & * & * \end{pmatrix}. \tag{11}$$

Observe that if we choose $q = 2$, block

$$\begin{pmatrix} 2 & 2 & 1 \\ 4 & 1 & 1 \\ 0 & 5 & 5 \end{pmatrix}$$

is singular and the reduction cannot be performed. On the other hand, if we apply three steps of LU factorization with partial pivoting to the columns 4, 5, and 6, then we obtain the following factorization:

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1.00 & & \\ 0.00 & 1.00 & \\ 0.75 & 0.15 & 1.00 \\ -0.50 & -0.30 & 0.00 \\ 0.50 & -0.30 & -0.80 \\ 0.25 & 0.25 & 0.60 \end{pmatrix} \begin{pmatrix} 4.0 & 1.0 & 1.0 \\ & 5.0 & 5.0 \\ & & 2.5 \end{pmatrix}.$$

Rows involved in the pivoting process are 3, 4, and 6 (see the first three columns in the above permutation matrix). This means that it is sufficient to exchange rows 5 and 6 of the original matrix (11) (setting $Q_{i,1}$ equal to the 3×3 identity matrix and $Q_{i,2}$ that exchanges rows 2 and 3) and choose $q = 1$ in order to ensure that the reduction process can be performed.

A stability analysis of this new algorithm can be derived from that proposed by S. Wright in [20] for his structured elimination algorithm. In fact it is sufficient to observe that if we apply Wright's algorithm to factorize matrix (2) and set $P = m/2$ (see [20]), then we obtain just the same matrix obtained after one step of the stabilized cyclic reduction algorithm. The choice of $\log_2 P$ levels allows us to perform the same operations. The difference is that row permutation used in the cyclic reduction factorization allows us to consider only operations effectively performed and hence to reduce the number of operations and memory requirement (see the next section). Summarizing, the stability analysis of [20] applied to the case in which the maximum possible number of processors is used shows that, under the set of conditions specified by S. Wright, the stabilized cyclic reduction algorithm is stable when used to solve BVPs.

TABLE 1
Arithmetic complexity of the proposed ABD parallel algorithms; n, m, and p represent the number of first-order ODEs, the number of internal blocks, and the number of processors, respectively.

factorization	$(\frac{14}{3}n^3 - \frac{5}{2}n^2 - \frac{1}{6}n)(m/p + \log_2 p) + \frac{2}{3}n^3 + \frac{1}{2}n^2 - \frac{1}{6}n$
solution	$(6n^2 - n)(m/p + \log_2 p) + 2n^2 - n$

5. Arithmetic complexity. In this section we analyze the arithmetic complexity of the proposed algorithms. We assume that we want to solve a system of n first-order BVPs on a parallel computer with p processors and that the discretization based on finite differences leads to the ABD system with $m > p$ internal blocks. We will also assume that both m and p are powers of 2. The computational cost depends on the following parameters:

- n is the number of first-order ODEs,
- m is the number of internal blocks,
- p is the number of processors.

Observe that the cost of the proposed algorithm is independent of q. The number of operations of one generic step of reduction (from (8) to (9), that is to reduce two $n \times 2n$ blocks to one; see the operations inside the "for i" loop of the factorization phase) is $\frac{14}{3}n^3 \quad \frac{5}{2}n^2 \quad \frac{1}{6}n$. When m is greater than p the number of blocks participating in each reduction step is $(m/p)/2 + (m/p)/4 + \cdots + 1$. This is followed by $\log_2 p$ simple reduction steps. Finally, the factorization of the last 4×4 block system (that corresponds to a $2n \times 2n$ system) requires $\frac{16}{3}n^3 \quad 2n^2 \quad \frac{1}{3}n$ operations. During the solution of the linear system (for one right-hand side) the solution of the 4×4 block system requires $8n^2 \quad 2n$ operations. The total number of operations inside the two "for i" loops is $6n^2 - n$ and these loops are executed $m/p - 1 + \log_2 p$ times. The total number of arithmetical operations is summarized in Table 1.

As already mentioned, data transmissions are required only in the last $\log_2 p$ steps of reduction. In the factorization phase a $n \times 2n$ block is transmitted at every step while in the solution phase a vector of size n is transmitted (using the data-transmission reducing approach suggested in section 3). Hence the total cost of transmission is (assuming that $t(k)$ is the time needed for one transmission of k elements)

$$(t(2n^2) \cdot t(n)) \log_2 p.$$

As a comparison we have chosen the parallel algorithm proposed by S. Wright in [20] which also solves BVP with nonseparated boundary conditions using Gaussian elimination. This algorithm shows good performance when compared with other existing algorithms for the solution of ABD systems (see also [20], Table 3). Following Wright's "two-level algorithm" suggestion, the algorithm requires

$$(\tfrac{23}{3}n^3 \cdot \tfrac{11}{2}n^2)(m/p + p)$$

parallel operations and approximately $t(n^2)p$ data transmissions.

Assuming that n is large enough for the n^3 terms to dominate the arithmetic complexity functions and that m is large, the following observations can be made:

1. The number of operations of the new algorithm is sensibly reduced in comparison with Wright's algorithm (approximately from $\frac{23}{3}n^3 m/p$ to $\frac{14}{3}n^3 m/p$).
2. The optimal number of processors for the new algorithm is proportional to m, in which case its arithmetic complexity is $O(n^3 \log_2 m)$.

As far as the storage requirement is concerned, the proposed algorithm does not require additional memory (as the usual cyclic reduction algorithm for block tridiagonal systems) if the factorization step is performed together with the linear system solution step. On the other hand, if those two steps are performed separately, the per-processor memory requirement is

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

whereas the minimum memory requirement to store all the data on a single processor

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

Assuming that the optimal number of processors was used, the total memory requirement for the new algorithm is approximately $3n^2 m \log_2 m$.

In comparison, the per processor memory requirement of Wright's algorithm is $3n^2 m/p$ when the factorization step is performed together with the linear system solution step, and $4mn^2/p$ when the two steps are performed separately.

6. Numerical tests. The stability properties of the presented algorithms were tested on two well-known BVPs used also by Wright (see [19, 13]). The first example is a system of three ODEs.

Problem 1.

$$M(t) \quad \begin{pmatrix} 1 & 19\cos(2t) & 0 & 1 - 19\sin(2t) \\ 0 & & 19 & 0 \\ 1 + 19\sin(2t) & & 0 & 1 + 19\cos(2t) \end{pmatrix},$$

$$q(t) \quad e^t \begin{pmatrix} 1 + 19(\cos(2t) & \sin(2t)) \\ 18 & \\ 1 & 19(\cos(2t) + \sin(2t)) \end{pmatrix},$$

with the following separated boundary conditions on the interval $[0, \pi]$ (Problem 1a):

$$\begin{aligned} y_1(0) &= 1, \\ y_2(\pi) &= e^\pi, \\ y_1(\pi) + 3y_3(\pi) &= 4e^\pi. \end{aligned}$$

The same problem has also been solved with the following nonseparated boundary conditions (see also [14], (Problem 1b)):

$$\begin{aligned} y_1(0) &= 1, \\ y_2(0) + y_2(\pi) &= 1 + e^\pi, \\ y_3(0) + y_3(\pi) &= 1 - e^\pi. \end{aligned}$$

The second example is a singular perturbation problem (see also [13]):

Problem 2.

$$\begin{aligned} &\epsilon y'' + (t^3 - t/2)y' - y = 0, \qquad t \in [-1, 1], \quad \epsilon = 10^{-3}, \\ &y(-1) = 1, \quad y(1) = 2. \end{aligned}$$

TABLE 2
Relative error in the solution computed by the ABDCR, SABDCR, and SLU algorithms for different numbers of internal blocks; all three algorithms generate exactly the same error.

problem	$m = 32$	$m = 128$	$m = 512$
1a	$5.8 \cdot 10^{-5}$	$3.6 \cdot 10^{-6}$	$2.3 \cdot 10^{-7}$
1b	$5.8 \cdot 10^{-5}$	$3.6 \cdot 10^{-6}$	$2.3 \cdot 10^{-7}$
2	$2.5 \cdot 10^{-2}$	$1.9 \cdot 10^{-3}$	$9.0 \cdot 10^{-5}$

TABLE 3
Execution times on random problems for $m = 64$ internal blocks; time in ticks.

algorithm	$n = 5$	$n = 10$	$n = 20$
ABDCR	6173	33532	218385
SABDCR	7966	38327	233710
SLU	8004	41889	257505

The cyclic reduction algorithm (ABDCR) and its stabilized version (SABDCR) have been compared with S. Wright's structured elimination algorithm (SLU). In all the examples we have used the trapezoidal rule with constant stepsize to discretize the differential equation [5]. The *total error* was evaluated using the following formula (where $\boldsymbol{y}$ is the computed solution and $\boldsymbol{y}(t)$ is the exact one) that combines the absolute and the relative error:

$$\max_i \| \boldsymbol{y}_i \quad \boldsymbol{y}(t_i) \| ./ (1 + \quad \boldsymbol{y}(t_i)) \| ,$$

where the operation ./ is a componentwise division and $\boldsymbol{y}_i$ and $\boldsymbol{y}(t_i)$ are vectors of length n.

In all cases (the two cyclic reduction algorithms and S. Wright's code), exactly the same values of total error were observed (see Table 2). In particular, for Problem 1 we obtained exactly the same total error while using separated and non-separated boundary conditions. We have also solved the problems with separated boundary conditions using SOLVEBLOCK [9] as the linear equations solver. The values of total error were exactly the same as for the remaining three algorithms. Hence the obtained error is just that due to the trapezoidal rule (the rate of convergence of the method is always maintained).

The single-processor efficiency of the proposed algorithms was studied on a single transputer T800 20 with 16 Mb of memory. We have used all three algorithms to solve a general system of first order differential equations (1) for random matrices M of sizes 5, 10, and 20. In all cases, nonseparated boundary conditions have been chosen and the right-hand side vector $\boldsymbol{q}(t)$ has been selected in such a way as to have the components of the solution behave as e^t. All codes were implemented using calls to the BLAS kernels and the LINPACK subroutines DGEFA and DGESL (optimized for the Transputer). In Table 3 we summarize the obtained results for $m = 64$ expressed in ticks (1 tick $= 64 \cdot 10^{-6}$ seconds). All values are averages of multiple runs.

As predicted, the ABDCR algorithm outperforms the remaining algorithms. For small n the SABDCR algorithm has essentially the same performance as the SLU algorithm, but for large n its performance becomes superior. Additional experiments indicate that, as predicted theoretically, the execution time of all algorithms increases linearly for increasing values of m.

The parallel implementation of the cyclic reduction based solvers has been evaluated on a network of 32 Microway transputers T800 20, each with a local memory

TABLE 4

Speed-ups of the ABDCR parallel algorithm for various number of processors p and $m = 32p$ internal blocks.

problem	$m = 256$ $p = 8$	$m = 512$ $p = 16$	$m = 1024$ $p = 32$
$n = 5$	7.05	12.62	21.04
$n = 10$	7.40	13.99	25.99
$n = 20$	7.44	14.27	27.11

TABLE 5

Speed ups of the SABDCR parallel algorithm for various number of processors p and $m = 32p$ internal blocks.

problem	$m = 256$ $p = 8$	$m = 512$ $p = 16$	$m = 1024$ $p = 32$
$n = 5$	6.80	12.42	21.47
$n = 10$	7.13	13.56	25.36
$n = 20$	7.21	13.88	26.45

of 1 Mb. The parallel algorithms have been implemented in Parallel FORTRAN [21] with the Express communication library [22]. As in the single-processor case, optimized BLAS/LINPACK kernels have been used wherever appropriate.

Tables 4 and 5 represent the speed-ups of the proposed parallel algorithms for $p = 8$, 16, and 32 processors and $m = 32p$ blocks (where the speed-up was calculated as a ratio between the single-processor and parallel execution times of a given algorithm). It can be observed that ABDCR outperforms SABDCR. This becomes more transparent for larger values of n. The performance dependency on n does not follow from the arithmetic complexity function presented in Table 1. It can be attributed to the computation to communication ratio which is approximately a function of n (see section 5). Thus as n increases communication becomes less important.

7. Conclusion. A new cyclic reduction algorithm for the solution of almost block diagonal systems arising from the discretization of BVP ODEs has been introduced. The experiments performed suggest that it has relatively good numerical properties (similar to those of S. Wright's stable algorithm). In the single-processor mode the proposed algorithm outperforms that of Wright for all cases tested. Our current work concentrates on an efficient distributed memory implementation of S. Wright's algorithm that will allow us a fair comparison and an extensive study of the parallel performance of the available algorithms.

Acknowledgments. We would like to express our gratitude to the anonymous referees who helped us substantially improve the paper and to Katarzyna Paprzycka for help with the English.

REFERENCES

[1] P. AMODIO, L. BRUGNANO, AND T. POLITI, *Parallel factorizations for tridiagonal matrices*, SIAM J. Numer. Anal., 30 (1993), pp. 813–823.

[2] P. AMODIO AND N. MASTRONARDI, *A parallel version of the cyclic reduction algorithm on a hypercube*, Parallel Comput., 19 (1993), pp. 1273–1281.

[3] P. AMODIO AND M. PAPRZYCKI, *Parallel solution of almost block diagonal systems on a hypercube*, Linear Algebra Appl., 241–243 (1996), pp. 85-103.

[4] 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, PA, 1993.

[5] U. M. Ascher, R. M. M. Mattheij, and R. D. Russell, *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*, Prentice Hall, Englewood Cliffs, NJ, 1988.

[6] U. M. Ascher and S. Y. P. Chan, *On parallel methods for boundary value ODE's*, Computing, 46 (1991), pp. 1–17.

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

[8] C. Cyphers, M. Paprzycki, and A. Karageorghis, *High Performance Solution of Partial Differential Equations Discretized Using a Chebyshev Spectral Collocation Method*, J. Comput. Appl. Math., 69 (1996), pp. 71–80.

[9] C. De Boor and R. Weiss, *SOLVEBLOCK: A package for solving almost block diagonal linear systems*, ACM Trans. Math. Software, 6 (1980), pp. 80–87.

[10] 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 (1983), pp. 358–375.

[11] 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 (1993), pp. 181–189.

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

[13] H. O. Kreiss, K. Nichols, and D. Brown, *Numerical methods for stiff two-point boundary value problems*, SIAM J. Numer. Anal., 23 (1986), pp. 325–368.

[14] R. M. M. Mattheij, *Decoupling and stability of algorithms for boundary value problems*, SIAM Rev., 27 (1985), pp. 1–44.

[15] M. Paprzycki and I. Gladwell, *Solving almost block diagonal systems on parallel computers*, Parallel Comput., 17 (1991), pp. 133–153.

[16] M. Paprzycki and I. Gladwell, *Using Level 3 BLAS to Solve Almost Block Diagonal Systems*, in Proc. Fifth SIAM Conference on Parallel Processing for Scientific Computing, J. Dongarra, K. Kennedy, P. Messina, D. C. Sorensen, and R. V. Voigt, eds., SIAM, Philadelphia, PA, 1992, pp. 52–62.

[17] J. M. Varah, *Alternate row and column elimination for solving certain linear systems*, SIAM J. Numer. Anal., 13 (1976), pp. 71–75.

[18] K. Wright, *Parallel treatment of block-bidiagonal matrices in the solution of ordinary differential boundary value problems*, J. Comput. Appl. Math., 45 (1993), pp. 191–200.

[19] S. Wright, *Stable parallel algorithms for two-point boundary value problems*, SIAM J. Sci. Statist. Comput., 13 (1992), pp. 742–764.

[20] S. Wright, *Stable parallel elimination for boundary value ODE's*, Numer. Math., 67 (1994), pp. 521–536.

[21] *Parallel FORTRAN User Guide*, 3L Ltd., Livingston, Scotland, 1988.

[22] *Express FORTRAN User's Guide*, ParaSoft Corp., Pasadena, CA, 1990.