European Society of Computational Methods in Sciences and Engineering (ESCMSE)

Journal of Numerical Analysis, Industrial and Applied Mathematics (JNAIAM)
vol. 1, no. 1, 2006, pp. 5-12
ISSN 1790–8140

Numerical Solution of General Bordered ABD Linear Systems by Cyclic Reduction

P. Amodio[*], I. Gladwell[†] and G. Romanazzi[*]

[*] *Dipartimento di Matematica, Università di Bari, I-70125 Bari, Italy*
[†] *Department of Mathematics, Southern Methodist University, Dallas, TX 75275, USA*

Received 2 December, 2005; accepted in revised form 13 January, 2006

Abstract: We generalize the cyclic reduction algorithm to the solution of Bordered ABD linear systems with blocks of different sizes and with different overlaps. This kind of system often arises from the numerical approximation of BVPs with nonseparated boundary conditions. In particular, in our experiments we refer to spline collocation techniques with monomial and B-spline basis functions.

Keywords: ABD systems, boundary value problem solvers.

Mathematics Subject Classification: 65L10, 15A23

1 Introduction

We analyze the solution of the BABD linear system

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

with the coefficient matrix A having the structure shown in Figure 1.

This is the most general BABD matrix structure: we recognize boundary blocks B_a and B_b on the first row, and block rows V_i defined such that there are overlapped columns only between successive blocks. For this reason, each block V_i may be represented as

$$V_i = \begin{pmatrix} S_{i-1} & T_i & R_i \end{pmatrix}, \tag{2}$$

where the columns of S_{i-1} and R_i overlap columns of V_{i-1} and V_{i+1}, respectively. Also V_1 and V_N have the structure in (2): the first columns of V_1 are overlapped with those of B_a while the last columns of V_N are overlapped with those of B_b, see Figure 1. We set $n_i \times (m_{i-1} + k_i + m_i)$, the size of each block V_i, where m_i is the number of overlapped columns between the blocks V_i and V_{i+1}, and n_0 as the number of rows of the boundary blocks. Then,

$$\sum_{i=0}^{N} n_i = \sum_{i=0}^{N} m_i + \sum_{i=1}^{N} k_i \qquad \text{(A is square)}$$

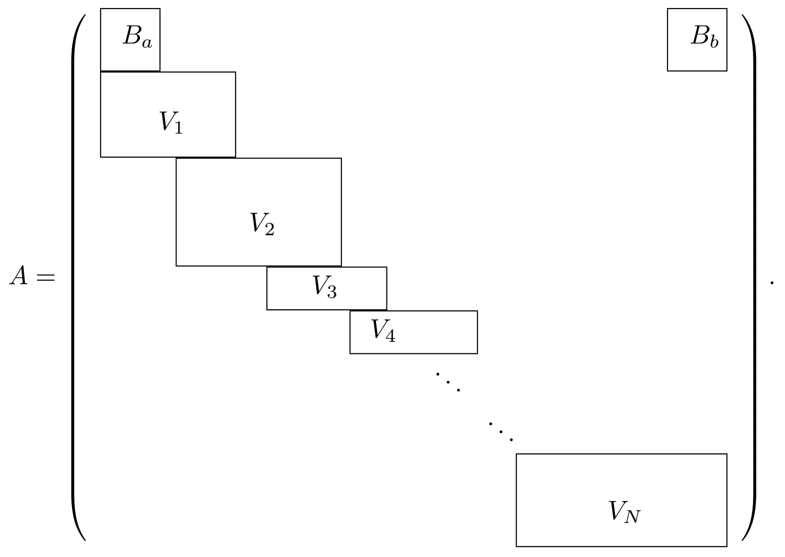

Figure 1: Structure of a general BABD matrix

and, for nonsingularity

$$\begin{aligned} &n_i \geq k_i, && i = 1, \dots N, \\ &n_i + n_{i+1} \geq k_i + m_{i+1} + k_{i+1}, && i = 1, \dots, N-1. \end{aligned} \qquad (3)$$

We suppose that N is much larger than n_i, m_i and k_i.

Solvers and packages for solving BABD (and ABD) linear systems have been considered in [1, 7]. In [2, 3] we have already considered the cyclic reduction algorithm applied to a BABD matrix with a simplified structure. In fact, in these papers we suppose that blocks S_i and R_i are square of dimension m and blocks T_i are null blocks. This means that the coefficient matrix is block lower bidiagonal with an additional block in the right-upper corner. In [4] we also show that the code BABDCR developed by two of the authors outperforms the existing codes RSCALE and COLROW and may be used in place of COLROW when boundary value ODE problems with nonseparated boundary conditions are considered.

On the other hand, BABD matrices with a more general structure often arise from the numerical approximation of BVPs for ODEs and PDEs with nonseparated boundary conditions. Matrices with nonnull blocks T_i arise, for example, from the numerical solution of linear two-point BVPs of order m

$$u^{(m)}(x) - \sum_{s=1}^{m} c_s(x) u^{(s-1)}(x) = f(x) \qquad \text{for } x \in [a, b]$$

with m linear boundary conditions

$$B_a z(u(a)) + B_b z(u(b)) = d$$

using collocation at k Gaussian points in N subintervals with B-spline or monomial spline basis. For example, when monomial splines are used, each block V_i is of size $(k+m) \times (k+2m)$ and boundary blocks B_a and B_b are square of dimension m. Moreover, m is the number of overlapped columns with the previous and successive blocks (k is the number of non-overlapped columns); the last m columns have the structure

$$R_i = \begin{pmatrix} 0 \\ I \end{pmatrix},$$

where I is a $m \times m$ identity matrix (the last m rows represent the continuity conditions while the remaining k rows in each block V_i arise from the collocation equation in the gaussian points). When B-splines are used, then the block V_i is $k \times (k+m)$ (where $k \geq m$) with m overlap columns (R_i and S_i have m columns).

The solution of such systems may be computed using COLROW (the size of blocks V_i is constant) or ABDPACK which was developed to solve the linear systems arising in spline collocation with monomial bases. All these packages are based on the Varah algorithm [8] which was originally proposed to solve ABD systems, that is, for systems arising from the numerical approximation of BVPs with separated boundary conditions. To apply them to a system arising from nonseparated boundary conditions, these codes require us to increase the size of each block V_i of the original system.

2 Cyclic reduction algorithm

Let us rewrite the coefficient matrix in Figure 1 as

$$A = \begin{pmatrix} B_a & & & & & & & & & B_b \\ S_0 & T_1 & R_1 & & & & & & & \\ & & S_1 & T_2 & R_2 & & & & & \\ & & & & S_2 & T_3 & R_3 & & & \\ & & & & & & & \ddots & & \\ & & & & & & & S_{N-1} & T_N & R_N \end{pmatrix}. \tag{4}$$

In accordance with the structure of (4) we define the right hand side $f = \begin{pmatrix} f_0^T & f_1^T & \dots & f_N^T \end{pmatrix}^T$ where each f_i is of length n_i, and the solution vector $x = \begin{pmatrix} z_0^T & w_1^T & z_1^T & \dots & w_{N-1}^T & z_N^T \end{pmatrix}^T$, where z_i and w_i are of length m_i and k_i, respectively.

We solve system (1) using a block cyclic reduction algorithm, that is, a recursive approach that reduces the original linear system to subsystems with a smaller number of unknowns. The only condition is that the first and the last unknowns z_0 and z_N are always among the unknowns of the successive reduced systems, and also the first row containing the boundary blocks is unchanged in the reduction process (therefore, boundary blocks B_a and B_b are maintained in the first row of each reduced system).

Following [2, 3], the first step of reduction would be to combine two successive block rows V_i and V_{i+1}, for $i = 1, 3, 5, \dots$, in order to obtain a new block row V'_{i+1} involving at least the unknowns z_{i-1} and z_{i+1}. In [2, 3] we suggest eliminating the overlapped columns between V_i and V_{i+1} by considering the partial pivoting LU factorization of the block $\begin{pmatrix} R_i \\ S_i \end{pmatrix}$. This approach is not appropriate here since the blocks T_i becomes larger in the successive reduced systems and hence the computational cost is large.

It is preferable to consider the partial pivoting LU factorization:

$$P_i \begin{pmatrix} T_i & R_i & \\ & S_i & T_{i+1} \end{pmatrix} = \begin{pmatrix} L_i \\ G_i \end{pmatrix} U_i = \begin{pmatrix} I & \\ F_i & I \end{pmatrix} \begin{pmatrix} L_i U_i \\ O \end{pmatrix}. \tag{5}$$

where L_i and U_i are square matrices and $F_i = G_i L_i^{-1}$. Now, from (5) we have

$$\begin{pmatrix} I & \\ -F_i & I \end{pmatrix} P_i \begin{pmatrix} S_{i-1} & T_i & R_i & & \\ & & S_i & T_{i+1} & R_{i+1} \end{pmatrix} = \begin{pmatrix} \widetilde{S}_{i-1} & L_i U_i & \widetilde{R}_{i+1} \\ S'_{i-1} & & R'_{i+1} \end{pmatrix} \tag{6}$$

$$\begin{pmatrix} I & \\ -F_i & I \end{pmatrix} P_i \begin{pmatrix} f_i \\ f_{i+1} \end{pmatrix} = \begin{pmatrix} \widetilde{f}_i \\ f'_{i+1} \end{pmatrix}. \tag{7}$$

From the second row of (6) and (7) we have a new equation involving z_{i-1} and z_{i+1}, that is, a reduced system with $\lceil N/2 \rceil + 1$ block rows and unknowns.

If for each index i we have $m_i = m$ and $k_i = k$ and therefore $n_0 = m$, $n_i = m + k$ for $i \geq 1$, the computational cost of this first step is

$$\tfrac{14}{3}m^3 + 16m^2k + 16mk^2 + \tfrac{16}{3}k^3 - \tfrac{3}{2}m^2 - 4mk - 2k^2 - \tfrac{1}{6}m - \tfrac{1}{3}k \tag{8}$$

for each of the $N/2$ reductions. This case appears when collocation for monomial spline is considered. On the contrary, for B-spline we must set $m_i = m$ and $k_i = k \equiv k - m$.

In (8) we have considered any power of m and k because these values may be quite different.

The reduction process may be optimized by taking into account that the matrix in (5) is not full. Moreover, the unknowns w_i are only multiplied by T_i which is contained in block V_i. For this reason we may consider a condensation step (see [5]) in order to eliminate, locally in each V_i, the dependence on w_i. This step may be also viewed as the first reduction step that eliminates the odd unknowns w_i of the solution vector x. We observe that blocks T_i of size $m_i \times k_i$, have full rank k_i because they are not overlapped by consecutive V_i blocks. So, we may determine the factorization:

$$\widetilde{P}_i T_i = \begin{pmatrix} \widetilde{L}_i \\ \widetilde{G}_i \end{pmatrix} \widetilde{U}_i = \begin{pmatrix} I & \\ \widetilde{F}_i & I \end{pmatrix} \begin{pmatrix} \widetilde{L}_i \widetilde{U}_i \\ O \end{pmatrix} \tag{9}$$

where $\widetilde{L}_i$ and $\widetilde{U}_i$ are square matrices and $\widetilde{F}_i = \widetilde{G}_i \widetilde{L}_i^{-1}$.

Multiplying V_i on the left by $\widetilde{P}_i$ and the inverse of the lower triangular matrix in the last term of (9) we obtain

$$\begin{pmatrix} I & \\ -\widetilde{F}_i & I \end{pmatrix} \widetilde{P}_i \begin{pmatrix} S_{i-1} & T_i & R_i \end{pmatrix} = \begin{pmatrix} \widetilde{S}_{i-1} & \widetilde{L}_i \widetilde{U}_i & \widetilde{R}_i \\ \widehat{S}_{i-1} & & \widehat{R}_i \end{pmatrix}. \tag{10}$$

Analogously, we perform the same operations on the right hand side f_i thus obtaining the vectors $\widetilde{f}_i$ and $\widehat{f}_i$ for the right side of (10). The row with the boundary blocks and the second row of (10) give the linear system

$$\begin{pmatrix} B_a & & & & B_b \\ \widehat{S}_0 & \widehat{R}_1 & & & \\ & \widehat{S}_1 & \widehat{R}_2 & & \\ & & \ddots & \ddots & \\ & & & \widehat{S}_{N-1} & \widehat{R}_N \end{pmatrix} \begin{pmatrix} z_0 \\ z_1 \\ z_2 \\ \vdots \\ z_N \end{pmatrix} = \begin{pmatrix} f_0 \\ \widehat{f}_1 \\ \widehat{f}_2 \\ \vdots \\ \widehat{f}_N \end{pmatrix}. \tag{11}$$

The system with matrix (11) has dimension equal to $\displaystyle\sum_{i=0}^{N} m_i = n_0 + \sum_{i=1}^{N} (n_i - k_i)$ and no longer depends on the unknowns w_i. These unknowns will be computed in the last step of the cyclic reduction back-substitution phase (when all the z_i have been computed) by using the first row of (10)

$$\tilde{L}_i \tilde{U}_i w_i = \tilde{f}_i - \tilde{S}_{i-1} z_{i-1} - \tilde{R}_i z_i.$$

Factorization (10) does not require additional memory since F_i may be saved together with L_i and U_i in place of T_i. Therefore, this condensation should be considered as a (completely parallelizable) initial step to be applied to ABD or BABD matrices in order to simplify their structure before factorization.

Returning to the solution of (1), system (11) may be further on reduced by considering the cyclic reduction algorithm in [2, 3] (even if blocks $\widehat{S}'_i$ and $\widehat{R}'_i$ are not square). At the first step, we consider the LU factorization of the $(n_i - k_i + n_{i+1} - k_{i+1}) \times m_i$ matrix (of rank m_i)

$$P_i \begin{pmatrix} \widehat{R}_i \\ \widehat{S}_i \end{pmatrix} = \begin{pmatrix} L_i \\ G_i \end{pmatrix} U_i = \begin{pmatrix} I & \\ F_i & I \end{pmatrix} \begin{pmatrix} L_i U_i \\ O \end{pmatrix}$$

that, applied to whole block rows i and $i+1$ in (11), gives

$$\begin{pmatrix} I & \\ -F_i & I \end{pmatrix} P_i \begin{pmatrix} \widehat{S}_{i-1} & \widehat{R}_i & \\ & \widehat{S}_i & \widehat{R}_{i+1} \end{pmatrix} = \begin{pmatrix} \bar{S}_{i-1} & L_i U_i & \bar{R}_i \\ S'_{i-1} & & R'_i \end{pmatrix}. \tag{12}$$

whose second row is independent of z_i. The first row may be used to compute z_i from z_{i-1} and z_{i+1}.

This last factorization requires additional memory for the fill-in blocks F_i. After this step we obtain the same reduced matrix as by the previous approach, but with a number of operations (in case of constant m and k)

$$4m^2k + 2mk^2 + \tfrac{2}{3}k^3 - mk - \tfrac{1}{2}k^2 - \tfrac{1}{6}k \tag{13}$$

for each of the N blocks V_i, and

$$\tfrac{14}{3}m^3 - \tfrac{3}{2}m^2 - \tfrac{1}{6}m \tag{14}$$

for each of the $N/2$ reductions. Hence, we save $(4m^2k + 6mk^2 + 2k^3 - mk - \frac{1}{2}k^2)N$ operations.

Iterating this last step on the successively reduced systems we obtain, after $\lceil \log_2 N \rceil$ steps, a 2×2 block full linear system

$$\begin{pmatrix} B_a & B_b \\ S_0^* & R_N^* \end{pmatrix} \begin{pmatrix} z_0 \\ z_N \end{pmatrix} = \begin{pmatrix} f_0^* \\ f_N^* \end{pmatrix}. \tag{15}$$

The factorization and the solution of (15) gives the first and the last unknown of x. Successively, a back substitution phase allows us to compute, in reverse order, all the other unknowns.

If m and k are constant, the computational cost of the algorithm, that we call GBABDCR, is

$$(\tfrac{14}{3}m^3 + 4m^2k + 2mk^2 + \tfrac{2}{3}k^3 - \tfrac{3}{2}m^2 - mk - \tfrac{1}{2}k^2 - \tfrac{1}{6}m - \tfrac{1}{6}k)N. \tag{16}$$

If $k = 0$ (all columns overlap), from (16) the cost is $\left(\frac{14}{3}m^3 - \frac{3}{2}m^2 - \frac{1}{6}m\right)N$ as in the BABDCR algorithm in [3]. The additional memory requirement (fill-in) is m^2N since the factorization of the T_i blocks does not require fill-in.

3 Numerical comparisons

First, we compare the two different cyclic reduction based approaches proposed above. We aim to prove, rather than simply theoretically (GBABDCR requires less computational cost and storage), that condensation may reduce the execution time. We call GBABDCR-0 the non optimized algorithm that, at the first reduction step, uses the factorization (5). Supposing that m and k are constant, then recall that the computational cost of GBABDCR-0 is

$$\left(\tfrac{14}{3}m^3 + 8m^2k + 8mk^2 + \tfrac{8}{3}k^3 - \tfrac{3}{2}m^2 - 2mk - k^2 - \tfrac{1}{6}m - \tfrac{1}{6}k\right)N \tag{17}$$

which is slight larger than that in (16).

Second, we compare these codes with COLROW [6] which is one of the fastest available codes for linear systems with ABD coefficient matrix

$$A = \begin{pmatrix} B_{top} & & & & & & \\ S_0 & T_1 & R_1 & & & & \\ & & S_1 & T_2 & R_2 & & \\ & & & & \ddots & & \\ & & & & S_{N-1} & T_N & R_N \\ & & & & & & B_{bot} \end{pmatrix} \tag{18}$$

where blocks S_i, T_i and R_i have constant size (m is the overlap between different rows and k is the number of non overlapped columns). COLROW uses a modified alternate row and column elimination which does not create fill-in. For the matrix (18) it has computational cost

$$\tfrac{2}{3}(m+k)^3 + (m+k)^2 m_{bot} + m^2 m_{top} + m m_{top} m_{bot},$$

where m_{top} and m_{bot} are the number of rows of B_{top} and B_{bot}, respectively. To apply COLROW to a BABD matrix, we rearrange the system as follows:

$$\widetilde{A} = \begin{pmatrix} -I & I & & & & & & & & & \\ S_0 & 0 & T_1 & R_1 & & & & & & & \\ & -I & 0 & 0 & I & & & & & & \\ & & & S_1 & 0 & T_2 & R_2 & & & & \\ & & & & -I & 0 & 0 & I & & & \\ & & & & & & \ddots & & & & \\ & & & & & & S_{N-1} & 0 & T_N & R_N & \\ & & & & & & & -I & 0 & 0 & I \\ & & & & & & & & & B_b & B_a \end{pmatrix} \tag{19}$$

$$\widetilde{x} = \begin{pmatrix} z_0 & y_0 & w_1 & z_1 & y_1 & w_2 & \dots & z_{N-1} & y_{N-1} & w_N & z_N & y_N \end{pmatrix}$$

$$\widetilde{f} = \begin{pmatrix} 0 & f_0 & 0 & f_1 & 0 & f_2 & \dots & 0 & f_N \end{pmatrix}$$

where S_i, T_i, R_i, f_i, z_i, w_i are the same blocks and vectors as above and vectors y_i are new unknowns $y_N = \ldots = y_1 = y_0 = z_0$. Then, we can apply COLROW to the ABD linear system (19) and compare its results to those of BABDCRG and BABDCR-0 applied to the original BABD linear system. In this case, the cost of COLROW is $\left(\frac{49}{3}m^3 + \frac{20}{3}m^2k + 4mk^2 + \frac{2}{3}k^3\right)N$.

We used a AlphaServer DS20E with a 667 MHz EV67 processor and a Compaq Fortran 90 (formerly Digital Fortran 90) compiler. For each test, the elements of the BABD coefficient matrix A were generated randomly. The right hand side f was set so that the solution $x = (1, \ldots, 1)^T$.

In Table 1 we show the execution times, in seconds, as functions of k, m and N, varying each parameter in turn and fixing the others. The timings confirm our expectations from theory: GBABDCR is faster than GBABDCR-0 and COLROW, though the speed-ups are lower than theory suggests. Moreover, the gap between GBABDCR and GBABDCR-0 increases for larger values of k/m; conversely, GBABDCR improves over COLROW when m/k increases. The ratios of costs between GBABDCR-0 and GBABDCR and between COLROW and GBABDCR are approximately 1.5 and 2.8, respectively. In Table 2 we show errors for the same tests. Note that the order of the error is the same for all the methods. The case $m = 5, k = 10, N = 2000$ gives a slight larger than predicted error, but this is true for all the methods and probably depends on the chosen random matrix.

Table 1: *Execution times (in seconds) to solve linear systems with varying m, k and N using GBABDCR, GBABDCR-0 and COLROW.*

problem	GBABDCR	GBABDCR-0	COLROW
$m = 5, k = 10, N = 2000$	6.53E-2	9.86E-2	0.148
$m = 10, k = 10, N = 2000$	0.156	0.222	0.439
$m = 20, k = 10, N = 2000$	0.529	0.651	1.581
$m = 10, k = 5, N = 2000$	0.106	0.135	0.317
$m = 10, k = 10, N = 2000$	0.156	0.222	0.439
$m = 10, k = 20, N = 2000$	0.305	0.490	0.733
$m = 10, k = 10, N = 1000$	7.12E-2	0.105	0.208
$m = 10, k = 10, N = 2000$	0.156	0.222	0.439
$m = 10, k = 10, N = 4000$	0.345	0.463	0.876

Table 2: *Computed errors in the solution of linear systems with varying m, k and N using GBABDCR, GBABDCR-0 and COLROW.*

problem	GBABDCR	GBABDCR-0	COLROW
$m = 5, k = 10, N = 2000$	4.12E-07	4.09E-07	9.95E-07
$m = 10, k = 10, N = 2000$	5.56E-10	9.75E-10	3.53E-10
$m = 20, k = 10, N = 2000$	1.32E-10	6.40E-11	8.15E-11
$m = 10, k = 5, N = 2000$	4.72E-11	7.88E-11	3.47E-11
$m = 10, k = 10, N = 2000$	5.56E-10	9.75E-10	3.53E-10
$m = 10, k = 20, N = 2000$	4.13E-11	3.41E-11	3.71E-11
$m = 10, k = 10, N = 1000$	7.08E-11	4.04E-11	4.61E-11
$m = 10, k = 10, N = 2000$	5.56E-10	9.75E-10	3.53E-10
$m = 10, k = 10, N = 4000$	5.56E-10	1.09E-09	3.53E-10

Acknowledgment

This work was partially supported by COFIN-PRIN 2004 (project "Metodi numerici e software matematico per le applicazioni").

References

[1] P. Amodio, J.R. Cash, G. Roussos, R.W. Wright, G. Fairweather, I. Gladwell, G.L. Kraut and M. Paprzycki, Almost block diagonal linear systems: sequential and parallel solution techniques, and applications, *Numer. Linear Algebra Appl.*, **7** (2000), 275-317.

[2] P. Amodio and M. Paprzycki, A cyclic reduction approach to the numerical solution of boundary value ODEs. *SIAM J. Sci. Comp* **18** 1 (1997), 56-68.

[3] P. Amodio and G. Romanazzi, BABDCR: a Fortran 90 package for the solution of Bordered ABD systems, Technical Report n. 40/04, Dipartimento di Matematica, Università di Bari, 2004 (submitted).

[4] P. Amodio, I. Gladwell and G. Romanazzi, An algorithm for the solution of Bordered ABD linear systems arising from BVPs, Technical Report n. 35/05, Dipartimento di Matematica, Università di Bari, 2005 (submitted).

[5] U.M. Ascher, S. Pruess and R.D. Russell, On spline basis selection for solving differential equations, *SIAM J. Numer. Anal.*, **20** 1 (1983), 121-142.

[6] 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** 3 (1983) 358-375.

[7] G. Fairweather and I. Gladwell, Algorithms for Almost Block Diagonal Linear Systems, *SIAM Review* **46** 1 (2004) 49-58.

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