

FIG. 2.1. *Partitioning of a banded matrix. Each point represents a (block) entry of the matrix.*

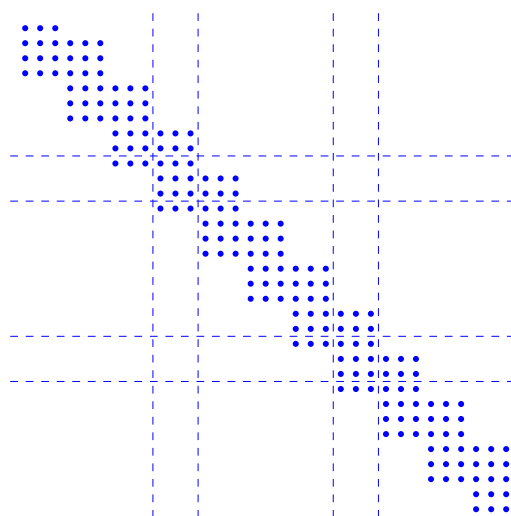


FIG. 2.2. *Partitioning of an ABD matrix. Each point represents an entry of the matrix.*

the number of lower and upper off (block) diagonals (see Figure 2.1), respectively. In case of ABD matrices, $a^{(i)}$ is a block of size equal to the number of rows in each block row of the coefficient matrix (see Figure 2.2). Since row and column permutations inside each block do not destroy the sparsity structure of the coefficient matrix, in ABD matrices we may permute the elements inside $a^{(i)}$ to improve stability properties. Blocks $A^{(i)}$ have the same sparsity structure as the original matrix, and are locally handled by using any suitable sequential algorithm.

In order to keep track of any parallel algorithm, we consider the following factorization [2, 9]

$$A = FTG, \quad (2.3)$$

- *LU factorization*, by setting in (2.8) $N^{(i)} = L^{(i)}$ and $S^{(i)} = U^{(i)}$, where $L^{(i)}U^{(i)}$ is the *LU* factorization of the matrix $A^{(i)}$. In this case, the (block) vectors $\mathbf{y}^{(i)}$ and $\mathbf{v}^{(i)}$ maintain the same sparsity structure as that of $\mathbf{c}_1^{(i)}$ and $\mathbf{b}_1^{(i)}$, respectively, while the vectors $\mathbf{z}^{(i)}$ and $\mathbf{w}^{(i)}$ are non-null fill-in (block) vectors, obtained by solving two triangular systems.
- *LUD factorization* (which derives from the Gauss-Jordan elimination algorithm), by setting in (2.8) $S^{(i)} = D^{(i)}$, a diagonal matrix, and

$$\begin{pmatrix} I & \mathbf{w}^{(i)T} & \\ \mathbf{o} & N^{(i)} & \mathbf{o} \\ & \mathbf{v}^{(i)T} & I \end{pmatrix} = \begin{pmatrix} I & \mathbf{o}^T & \\ \mathbf{o} & L^{(i)} & \mathbf{o} \\ & \mathbf{v}^{(i)T} & I \end{pmatrix} \begin{pmatrix} I & \mathbf{w}^{(i)T} & \\ \mathbf{o} & U^{(i)} & \mathbf{o} \\ & \mathbf{o}^T & I \end{pmatrix},$$

where $L^{(i)}$ and $U^{(i)}$ are lower and upper triangular matrices, respectively, with unit diagonal. Therefore, $\mathbf{v}^{(i)}$ and $\mathbf{w}^{(i)}$ maintain the same sparsity structure as that of $\mathbf{b}_1^{(i)}$ and $\mathbf{c}_0^{(i)}$, respectively, while $\mathbf{z}^{(i)}$ and $\mathbf{y}^{(i)}$ are non-null fill-in (block) vectors.

- *cyclic reduction* algorithm [2, 9] (see also [1, 12, 15, 16]), which is one of the best known parallel algorithms but that, in its original form, requires a synchronization at each step of reduction. In fact, the idea of this algorithm is to perform several reductions that, at each step, halve the size of the system. On the other hand, to obtain a factorization in the form (2.8), it is possible to consider cyclic reduction as a sequential algorithm to be applied locally,

$$M^{(i)} = (\hat{P}_1^{(i)} \hat{L}_1^{(i)} \hat{P}_2^{(i)} \hat{L}_2^{(i)} \dots) \hat{D}^{(i)} (\dots \hat{U}_2^{(i)} \hat{P}_2^{(i)T} \hat{U}_1^{(i)} \hat{P}_1^{(i)T}),$$

where $\hat{P}_i^{(i)}$ are suitable permutation matrices that maintain the first and last row in the reduced matrix. The computational cost, which is much higher if the algorithm is applied to A on a sequential computer, becomes comparable to the previous local factorizations since this algorithm does not compute fill-in block vectors. As a consequence, the corresponding parallel factorization algorithm turns out to be one of the most effective.

- *Alternate row and column elimination* [25] which is an algorithm suitable for ABD matrices. In fact, for such matrices alternate row and column permutations always guarantee stability without fill-in. This feature extends to the parallel algorithm, by taking into account that row permutations between the first block row of $A^{(i)}$ and the block containing $\mathbf{c}_0^{(i)}$ (see (2.7)), still make the parallel algorithm stable without introducing fill-in. Such parallel factorization is defined by setting $N^{(i)} = P^{(i)}L^{(i)}$ and $S^{(i)} = U^{(i)}Q^{(i)}$, where $P^{(i)}$ and $Q^{(i)}$ are permutation matrices and $L^{(i)}$ and $U^{(i)}$, after a suitable reordering of the rows and of the columns, are 2×2 block triangular matrices (see [10] for full details). Finally, the (block) vectors $\mathbf{y}^{(i)}$ and $\mathbf{z}^{(i)}$ maintain the same sparsity structure as that of $\mathbf{b}_1^{(i)}$ and $\mathbf{c}_0^{(i)}$, respectively, whereas $\mathbf{w}^{(i)}$ and $\mathbf{v}^{(i)}$ are fill-in (block) vectors.

For what concerns the solution of the systems associated to the previous parallel factorizations, there is much parallelism inside. The solution of the systems with the matrices F and G may proceed in parallel on the different processors. Conversely, the solution of the system with the matrix T requires a sequential part, consisting in the solution of a *reduced system* with the (block) tridiagonal *reduced matrix*

$$T_p = \begin{pmatrix} \alpha^{(1)} & \gamma^{(2)} & & & \\ \beta^{(2)} & \alpha^{(2)} & \ddots & & \\ & \ddots & \ddots & \gamma^{(p-1)} & \\ & & \beta^{(p-1)} & \alpha^{(p-1)} & \end{pmatrix}. \tag{2.9}$$

We observe that the (block) size of T_p only depends on p and is independent of n .

For a matrix A with singular or ill-conditioned sub-blocks $A^{(i)}$, the local factorizations may be unstable or even undefined. Consequently, it is necessary to slightly modify the factorization (2.8), in order to obtain stable parallel algorithms. The basic idea is that factorization (2.8) may produce more than two entries in the *reduced system*. In other words, the factorization of $A^{(i)}$ is stopped when the considered sub-block is ill-conditioned (or

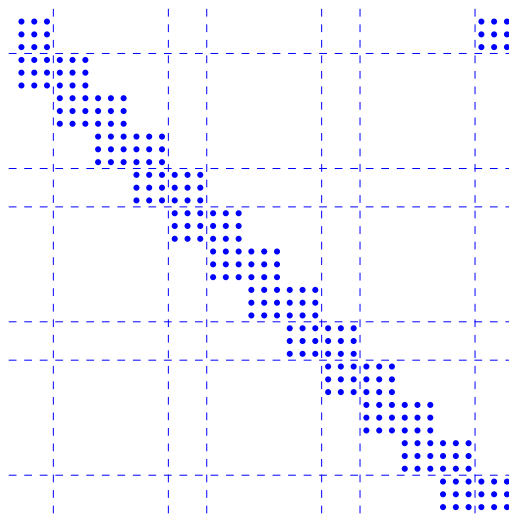


FIG. 2.3. Partitioning of a BABD matrix. Each point represents an entry of the matrix.

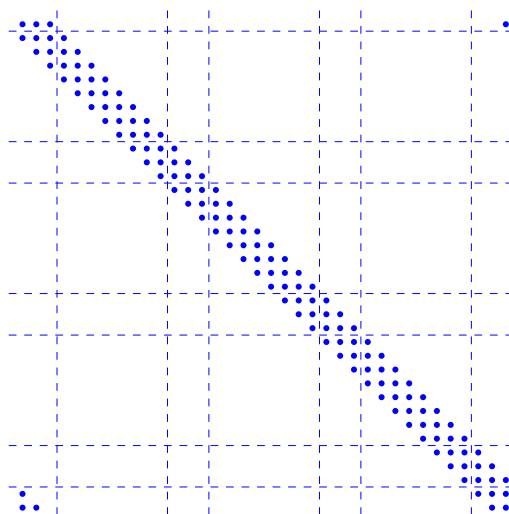


FIG. 2.4. Partitioning of a matrix with a circulant-like structure. Each point represents a (block) entry of the matrix.

where b is the smallest rectangular block containing all the corner elements.

A factorization similar to that in (2.3)–(2.6) (the obvious differences are related to the first and last (block) rows) produces a corresponding *reduced system* with the *reduced matrix*

$$T_p = \begin{pmatrix} \alpha^{(0)} & \gamma^{(1)} & & \beta^{(0)} \\ \beta^{(1)} & \alpha^{(1)} & \gamma^{(2)} & \\ & \beta^{(2)} & \alpha^{(2)} & \ddots \\ & & \ddots & \ddots & \gamma^{(p)} \\ & & & \beta^{(p)} & \alpha^{(p)} \end{pmatrix}. \quad (2.11)$$

We observe that, for the very important classes of BABD and circulant-like matrices (the latter, after a suitable row permutation, see Figure 2.5), both the matrix (2.10) and the *reduced matrix* (2.11) have the form of a lower block bidiagonal matrix (i. e., $c_j^{(i)} = 0$ and $\gamma^{(i)} = 0$ for all i and j) with an additional right-upper

Obviously, this problem may be solved in a sequential fashion, by means of the iteration (see (3.6)-(3.7)):

$$y_{N0} = y_0, \quad M_i \mathbf{y}_i = \mathbf{g}_i + \mathbf{v}_i y_{N,i-1}, \quad i = 1, \dots, p.$$

Nevertheless, by following arguments similar to those in the previous section, we consider the factorization:

$$M = \begin{pmatrix} I_m & & & & & \\ & M_1 & & & & \\ & & M_2 & & & \\ & & & \ddots & & \\ & & & & M_p & \end{pmatrix} \begin{pmatrix} I_m & & & & & \\ -\mathbf{w}_1 & I_{mN} & & & & \\ & -W_2 & I_{mN} & & & \\ & & & \ddots & & \\ & & & & -W_p & I_{mN} \end{pmatrix},$$

where (see (3.9))

$$W_i = [O \mid \mathbf{w}_i] \in \mathbb{R}^{mN \times mN}, \quad \mathbf{w}_i = M_i^{-1} \mathbf{v}_i \in \mathbb{R}^{mN \times m}. \tag{3.10}$$

Consequently, at first we solve, in parallel, the systems

$$M_i \mathbf{z}_i = \mathbf{g}_i, \quad \mathbf{z}_i = (z_{1i}, \dots, z_{Ni})^T, \quad i = 1, \dots, p, \tag{3.11}$$

and, then, (see (3.10) and (3.6)) recursively update the local solutions,

$$\mathbf{y}_1 = \mathbf{z}_1 + \mathbf{w}_1 y_{01}, \tag{3.12}$$

$$\mathbf{y}_i = \mathbf{z}_i + W_i \mathbf{y}_{i-1} \equiv \mathbf{z}_i + \mathbf{w}_i y_{0i}, \quad i = 2, \dots, p.$$

The latter recursion, however, has still much parallelism. Indeed, if we consider the partitionings (see (3.8), (3.11), and (3.10))

$$\mathbf{y}_i = \begin{pmatrix} \hat{\mathbf{y}}_i \\ y_{Ni} \end{pmatrix}, \quad \mathbf{z}_i = \begin{pmatrix} \hat{\mathbf{z}}_i \\ z_{Ni} \end{pmatrix}, \quad \mathbf{w}_i = \begin{pmatrix} \hat{\mathbf{w}}_i \\ w_{Ni} \end{pmatrix}, \quad w_{Ni} \in \mathbb{R}^{m \times m}, \tag{3.13}$$

then (3.12) is equivalent to solve, at first, the *reduced system*

$$\begin{pmatrix} I_m & & & & \\ -w_{N1} & I_m & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & -w_{N,p-1} & I_m \end{pmatrix} \begin{pmatrix} y_{01} \\ y_{02} \\ \vdots \\ y_{0p} \end{pmatrix} = \begin{pmatrix} y_0 \\ z_{N1} \\ \vdots \\ z_{N,p-1} \end{pmatrix}, \tag{3.14}$$

i. e.,

$$y_{01} = y_0, \quad y_{0,i+1} = z_{Ni} + w_{Ni} y_{0i}, \quad i = 1, \dots, p-1, \tag{3.15}$$

after which performing the p parallel updates

$$\hat{\mathbf{y}}_i = \hat{\mathbf{z}}_i + \hat{\mathbf{w}}_i y_{0i}, \quad i = 1, \dots, p-1, \quad \mathbf{y}_p = \mathbf{z}_p + \mathbf{w}_p y_{0p}. \tag{3.16}$$

We observe that:

- the parallel solution of the p systems in (3.11) is equivalent to compute the approximate solution of the following p ODE-IVPs,

$$z' = Lz + g(t), \quad t \in [\tau_{i-1}, \tau_i], \quad z(\tau_{i-1}) = 0, \quad i = 1, \dots, p, \tag{3.17}$$

in place of the corresponding ones in (3.4);

- the solution of the *reduced system* (3.14)-(3.15) consists in computing the proper initial values $\{y_{0i}\}$ for the previous ODE-IVPs;
- the parallel updates (3.16) update the approximate solutions of the ODE-IVPs (3.17) to those of the corresponding ODE-IVPs in (3.4).

REMARK 1. *Clearly, the solution of the first (parallel) system in (3.11) and the first (parallel) update in (3.12) (see also (3.16)) can be executed together, by solving the linear system (see (3.6))*

$$M_1 \mathbf{y}_1 = \mathbf{g}_1 + \mathbf{v}_1 y_0, \tag{3.18}$$

thus directly providing the final discrete approximation on the first processor; indeed, this is possible, since the initial condition y_0 is given.

We end this section by emphasizing that one obtains an almost perfect parallel speed-up, if p processors are used, provided that the cost for the solution of the *reduced system* (3.14) and of the parallel updates (3.16) is small, with respect to that of (3.11) (see [5, 6] for more details). This is, indeed, the case when the parameter N in (3.3) is large enough and the coarse partition (3.2) can be supposed to be *a priori* given.

4. Connections with the “Parareal” algorithm. We now briefly describe the “Parareal” algorithm introduced in [23, 24], showing the existing connections with the parallel method previously described. This method, originally defined for solving PDE problems, for example linear or quasi-linear parabolic problems, can be directly cast into the ODE setting via the semi-discretization of the space variables; that is, by using the method of lines. In more detail, let us consider the problem

$$\frac{\partial}{\partial t} y = \mathcal{L} y, \quad t \in [t_0, T], \quad y(t_0) = y_0, \tag{4.1}$$

where \mathcal{L} is an operator from a Hilbert space V into V' . Let us consider again the partition (3.2) of the time interval, and consider the problems

$$\frac{\partial}{\partial t} y = \mathcal{L} y, \quad t \in [\tau_{i-1}, \tau_i], \quad y(\tau_{i-1}) = y_{0i}, \quad i = 1, \dots, p. \tag{4.2}$$

Clearly, in order for (4.1) and (4.2) to be equivalent, one must require that

$$y_{0i} = y(\tau_{i-1}), \quad i = 1, \dots, p. \tag{4.3}$$

The initial data (4.3) are then formally related by means of suitable *propagators* \mathcal{F}_i such that

$$y_{0,i+1} = \mathcal{F}_i y_{0i}, \quad i = 1, \dots, p-1. \tag{4.4}$$

The previous relations can be cast in matrix form as (\mathcal{I} now denotes the identity operator)

$$F \mathbf{y} \equiv \begin{pmatrix} \mathcal{I} & & & & \\ -\mathcal{F}_1 & \mathcal{I} & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & -\mathcal{F}_{p-1} & \mathcal{I} \end{pmatrix} \begin{pmatrix} y_{01} \\ y_{02} \\ \vdots \\ y_{0p} \end{pmatrix} = \begin{pmatrix} y_0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \equiv \boldsymbol{\eta}. \tag{4.5}$$

For solving (4.5), the authors essentially define the splitting

$$F = (F - G) + G, \quad G = \begin{pmatrix} \mathcal{I} & & & & \\ -\mathcal{G}_1 & \mathcal{I} & & & \\ & \ddots & \ddots & & \\ & & & -\mathcal{G}_{p-1} & \mathcal{I} \end{pmatrix},$$

with *coarse propagators*

$$\mathcal{G}_i \approx \mathcal{F}_i, \quad i = 1, \dots, p,$$

and consider the iterative procedure

$$G\mathbf{y}^{(k+1)} = (G - F)\mathbf{y}^{(k)} + \boldsymbol{\eta}, \quad k = 0, 1, \dots,$$

with an obvious meaning of the upper index. This is equivalent to solve the problems

$$\begin{aligned} y_{01}^{(k+1)} &= y_0, \\ y_{0,i+1}^{(k+1)} &= \mathcal{G}_i y_{0i}^{(k+1)} + (\mathcal{F}_i - \mathcal{G}_i)y_{0i}^{(k)}, \quad i = 1, \dots, p-1, \end{aligned} \quad (4.6)$$

thus providing good parallel features, if we can assume that the coarse operators \mathcal{G}_i are “cheap” enough. The iteration (4.6) defines the “Parareal” algorithm, which is iterated until

$$\|y_{0i}^{(k+1)} - y_{0i}^{(k)}\|, \quad i = 2, \dots, p,$$

are suitably small. In the practice, in case of linear operators, problem (4.1) becomes, via the method of lines, an ODE in the form (3.1), with L a huge and very sparse matrix. Consequently, problems (4.2) become in the form (3.4). Similarly, the propagator \mathcal{F}_i consists in the application of a suitable discrete method for approximating the solution of the corresponding i th problem in (3.4), and the coarse propagator \mathcal{G}_i describes the application of a much cheaper method for solving the same problem. As a consequence, if the discrete problems corresponding to the propagators $\{\mathcal{F}_i\}$ are in the form (3.8), then the discrete version of the recurrence (4.4) becomes exactly (3.15), as well as the discrete counterpart of the matrix form (4.5) becomes (3.14).

We can then conclude that the “Parareal” algorithm in [23, 24] *exactly* coincides with the iterative solution of the *reduced system* (3.14), induced by a suitable splitting.

We observe that the previous iterative procedure may be very appropriate, when the matrix L is large and sparse since, in this case, the computations of the block vectors $\{\mathbf{w}_i\}$ in (3.10), and then of the matrices $\{w_{Ni}\}$ (see (3.13)) would be clearly impractical. Moreover, it can be considerably improved by observing that

$$w_{Ni}y_{0i} \approx e^{(\tau_i - \tau_{i-1})L}y_{0i}.$$

Consequently, by considering a suitable approximation to the matrix exponential, the corresponding parallel algorithm turns out to become semi-iterative and potentially very effective, as recently shown in [8].

REFERENCES

- [1] P. Amodio. Optimized cyclic reduction for the solution of linear tridiagonal systems on parallel computers. *Comput. Math. Appl.* **26** (1993) 45–53.
- [2] P. Amodio and L. Brugnano. Parallel factorizations and parallel solvers for tridiagonal linear systems. *Linear Algebra Appl.* **172** (1992) 347–364.
- [3] P. Amodio and L. Brugnano, The parallel QR factorization algorithm for tridiagonal linear systems. *Parallel Comput.* **21** (1995) 1097–1110.

- [4] P. Amodio and L. Brugnano. Stable Parallel Solvers for General Tridiagonal Linear Systems. *Z. Angew. Math. Mech.* **76**, suppl. 1 (1996) 115–118.
- [5] P. Amodio and L. Brugnano. Parallel implementation of block boundary value methods for ODEs, *J. Comput. Appl. Math.* **78** (1997) 197–211.
- [6] P. Amodio and L. Brugnano. Parallel ODE solvers based on block BVMs, *Adv. Comput. Math.* **7** (1997) 5–26.
- [7] P. Amodio and L. Brugnano. ParalleloGAM: a Parallel Code for ODEs, *Appl. Numer. Math.* **28** (1998) 95–106.
- [8] P. Amodio and L. Brugnano. Parallel solution in time of ODEs: some achievements and perspectives. *Appl. Numer. Math.* **59** (2009) 424–435.
- [9] P. Amodio, L. Brugnano and T. Politi. Parallel factorizations for tridiagonal matrices. *SIAM J. Numer. Anal.* **30** (1993) 813–823.
- [10] 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**, no. 5 (2000) 275–317.
- [11] P. Amodio, I. Gladwell and G. Romanazzi. Numerical solution of general Bordered ABD linear systems by cyclic reduction. *JNAIAM J. Numer. Anal. Ind. Appl. Math.* **1**, no. 1 (2006) 5–12.
- [12] P. Amodio, N. Mastronardi. A parallel version of the cyclic reduction algorithm on a Hypercube. *Parallel Comput.* **19** (1993) 1273–1281.
- [13] P. Amodio and F. Mazzia. A parallel Gauss-Seidel method for block tridiagonal linear systems. *SIAM J. Sci. Comput.* **16** (1995) 1451–1461.
- [14] P. Amodio and F. Mazzia. Parallel iterative solvers for banded linear systems. *Lecture Notes in Comput. Sci.* **1196**, (Numerical Analysis and its Applications. L. Vulkov, J. Wąsniewski, and P. Yalamov editors, Springer, Berlin), 17–24, 1997.
- [15] P. Amodio and M. Paprzycki. Parallel solution of Almost Block Diagonal systems on a Hypercube. *Linear Algebra Appl.* **241-243** (1996) 85–103.
- [16] P. Amodio and M. Paprzycki. On the parallel solution of Almost Block Diagonal systems. *Control Cybernet.* **25**, no. 3 (1996) 645–656.
- [17] P. Amodio and M. Paprzycki. A cyclic reduction approach to the numerical solution of boundary value ODEs. *SIAM J. Sci. Comput.* **18**, no. 1 (1997) 56–68.
- [18] P. Amodio, M. Paprzycki and T. Politi. A survey of parallel direct methods for block bidiagonal linear systems on distributed memory computers. *Comput. Math. Appl.* **31** (1996) 111–127.
- [19] P. Amodio and G. Romanazzi. BABDCR: a Fortran 90 package for the solution of Bordered ABD systems. *ACM Trans. Math. Software* **32**, no. 4 (2006) 597–608. (Available on the url: <http://www.netlib.org/toms/859>)
- [20] L. Brugnano and D. Trigiante. On the potentiality of sequential and parallel codes based on extended trapezoidal rules (ETRs), *Appl. Numer. Math.* **25** (1997) 169–184.
- [21] L. Brugnano and D. Trigiante. Parallel implementation of block boundary value methods on nonlinear problems: theoretical results, *Appl. Numer. Math.* **78** (1997) 197–211.
- [22] L. Brugnano and D. Trigiante. *Solving Differential Problems by Multistep Initial and Boundary Value Methods*, Gordon and Breach, Amsterdam, 1998.
- [23] J. L. Lions, Y. Maday and G. Turinici. Résolution d’EDP par un schéma en temps “pararéel”, *C. R. Acad. Sci. Paris, Série I* **332** (2001) 661–668.
- [24] Y. Maday and G. Turinici. A parareal in time procedure for the control of partial differential equations, *C. R. Acad. Sci. Paris, Série I* **335** (2002) 387–392.
- [25] J. M. Varah. Alternate row and column elimination for solving certain linear systems. *SIAM J. Numer. Anal.* **13** (1976) 71–75.
- [26] S. J. Wright. A collection of problems for which Gaussian elimination with partial pivoting is unstable. *SIAM J. Sci. Stat. Comput.* **14** (1993) 231–238.

Edited by: Francesca Mazzia

Received: June 27, 2009

Accepted: September 10, 2009