Recent advances in the parallel solution in time of ODEs

Pierluigi Amodio[*] and Luigi Brugnano[†]

[*]*Dipartimento di Matematica, Università di Bari, Via Orabona 4, 70125 Bari (Italy),*
amodio@dm.uniba.it
[†]*Dipartimento di Matematica "U. Dini", Università di Firenze, Viale Morgagni 67/A, 50134 Firenze (Italy),*
luigi.brugnano@unifi.it

Abstract. The parallel solution of initial value problems for ordinary differential equations (ODE-IVPs) has been a very active field of investigation in the past years. In general, the possibility of using parallel computing in this setting concerns different aspects of the numerical solution of ODEs, usually depending on the parallel platform to be used and/or on the complexity of the problem to be solved. In this paper we review possible extensions of a parallel method proposed in the mid-nineties by the authors. Moreover, we analyze its connections with subsequent approaches to the parallel solution of ODE-IVPs, in particular the "Parareal" algorithm.

Keywords: Ordinary differential equations, initial value problems, stiff problems, parallel computing, parallelism across the steps
PACS: 65L05, 65L06, 65L80, 65H10

INTRODUCTION

In this paper we will concern with the parallel solution of linear initial value ordinary differential equation problems

$$\begin{aligned} &y' = Ay + b(t), \qquad t \in [t_0,\, t_f], \quad y \in \mathbb{R}^n, \\ &y(t_0) = y_0. \end{aligned} \tag{1}$$

In the past years this problem has been studied in detail and several approaches have been considered. See the monograph [6] and the special issue [7], both edited by K. Burrage, for more details.

Among the possible choices, we consider *parallelism across the steps*, that is, we perform several integration steps concurrently. This is possible when we are able to subdivide the integration interval a priori in sub-intervals that can be themselves discretized with the same number of points. In this way, by assigning each sub-interval to a different processor, each one will perform the same amount of work.

Here we suppose that the matrix A is large and sparse. Such kind of IVPs arises from the discretization of partial differential equations, for example linear or quasi-linear parabolic problems. These problems are really up-to-date since they require the solution of large systems (with an high computational cost) and are relatively simple to solve (it is possible to use constant stepsize).

The idea of using linear algebra algorithms to solve in parallel problem (1) was originally proposed in the papers [2] and [3] supposing that the matrix A was large and dense. A similar approach was derived a few years later in [9] and [10], which is known as the *parareal algorithm*. It was aimed to solve problems arising from the numerical solution of PDEs. The latter papers originated a very large literature on this subject: see, for example, [8] and [11], and the references therein. Finally, in [4] the approach in [2] and [3] has been updated in order to solve problems with matrix A large and sparse. In this paper we summarize this extension, pointing out some remarks and future developments.

THE PARALLEL ALGORITHM

Let us suppose the integration interval in (1) is subdivided in q sub-intervals by the points $t_0 < t_1 < \cdots < t_q \equiv t_f$. Then, the original problem (1) is equivalent to the following q sub-problems

$$\begin{aligned} &y' = Ay + b(t), \qquad t \in [t_{i-1},\, t_i], \\ &y(t_{i-1}) \text{ known}, \qquad\qquad\qquad i = 1, \dots, q, \end{aligned} \tag{2}$$

where the initial value $y(t_{i-1})$ of the ith problem is the solution in the last point of the previous sub-interval ($y(t_0)$ is the only known quantity).

Let $M_i\mathbf{y}_i = \mathbf{v}_i y_{i-1} + \mathbf{g}_i$ be the numerical approximation to the ith problem in (2) obtained by means of suitable discrete methods (e.g., linear multistep methods), where the matrices M_i and $\mathbf{v}_i$ and the vector $\mathbf{g}_i$ do depend on the chosen method, $\mathbf{y}_i$ contains the numerical solution in the ith sub-interval (apart from the initial condition $y_{i-1} \approx y(t_{i-1})$ which is supposed as given from the previous sub-interval). Then, a parallel solution of (2) may arise from the concurrent solution of the q initial value problems, i.e., from the parallel solution of the following linear system, (here $V_i = [O \,|\, \mathbf{v}_i]$, with O the zero matrix of appropriate size)

$$\begin{pmatrix} M_1 & & & \\ -V_2 & M_2 & & \\ & \ddots & \ddots & \\ & & -V_q & M_q \end{pmatrix} \begin{pmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \vdots \\ \mathbf{y}_q \end{pmatrix} = \begin{pmatrix} \mathbf{g}_1 + \mathbf{v}_1 y_0 \\ \mathbf{g}_2 \\ \vdots \\ \mathbf{g}_q \end{pmatrix},$$

corresponding to the matrix form of the discrete problem. This can be achieved through the following *parallel factorization* [1, 5], of the coefficient matrix,

$$\begin{pmatrix} M_1 & & & \\ -V_2 & M_2 & & \\ & \ddots & \ddots & \\ & & -V_q & M_q \end{pmatrix} = \begin{pmatrix} M_1 & & & \\ & M_2 & & \\ & & \ddots & \\ & & & M_q \end{pmatrix} \begin{pmatrix} I & & & \\ -W_2 & I & & \\ & \ddots & \ddots & \\ & & -W_q & I \end{pmatrix},$$

where $W_i = [O \,|\, \mathbf{w}_i]$ and $\mathbf{w}_i = M_i^{-1}\mathbf{v}_i$. Consequently, at first we solve, in parallel, the systems

$$\begin{aligned} M_1\mathbf{z}_1 &= \mathbf{g}_1 + \mathbf{v}_1 y_0, \\ M_i\mathbf{z}_i &= \mathbf{g}_i, \qquad i = 2, \dots, q, \end{aligned} \tag{3}$$

and, then, recursively update the local solutions: $\mathbf{y}_1 = \mathbf{z}_1$ and

$$\mathbf{y}_i = \mathbf{z}_i + W_i\mathbf{y}_{i-1} \equiv \mathbf{z}_i + \mathbf{w}_i y_{i-1}, \qquad i = 2, \dots, q. \tag{4}$$

We now emphasize two remarks in the solution of (4): first of all, only the first point y_{i-1} in each sub-interval $[t_{i-1},\, t_i]$, $i = 2, \dots, q$, has to be computed sequentially by means of the iteration (let w_i be the block entry of $\mathbf{w}_i$ corresponding to y_i)

$$\begin{aligned} y_1 &= z_1 \\ y_i &= z_i + w_i y_{i-1}, \qquad i = 2, \dots, q, \end{aligned} \tag{5}$$

while the remaining part of each vector $\mathbf{y}_i$, $i = 2, \dots, q$ can be computed in parallel. The computation of (5) consists in the solution of the *reduced system* [1, 5, 2, 3, 4]

$$\begin{pmatrix} I & & & \\ -w_2 & I & & \\ & \ddots & \ddots & \\ & & -w_q & I \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_q \end{pmatrix} = \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_q \end{pmatrix}. \tag{6}$$

We mention that the iterative solution of (6), by means of a suitable splitting procedure, $\underline{\text{exactly}}$ defines the *parareal algorithm.*

The second remark is that, even if A is large and sparse, in general W_i turns out to be dense. Therefore, its computation may be too expensive. Consequently, in such a case it is preferable to solve in parallel the systems (see (3))

$$M_i\mathbf{y}_i = \mathbf{z}_i + \mathbf{v}_i y_{i-1}, \qquad i = 2, \dots, q, \tag{7}$$

instead of using the updates (4). Moreover, we also mention that it is useless to compute the last vector $\mathbf{z}_q$ in (3), because it is not acutally used in (7), as it is shown in the following three-phases algorithm, which summarizes the basic steps of the whole approach for solving problem (1):

Phase 1. parallel solution of the $q-1$ ODE-IVPs:

$$\begin{aligned} z' &= Az + b(t), \qquad t \in [t_0,\, t_1], \\ z(t_0) &= y_0, \end{aligned} \tag{8}$$

$$\begin{aligned} z' &= Az + b(t), \qquad t \in [t_{i-1},\, t_i], \\ z(t_{i-1}) &= 0, \qquad\qquad\qquad i = 2, \dots, q-1. \end{aligned} \tag{9}$$

Each sub-problem is solved by means of a suitable discrete method, e.g., a linear multistep method. Apart from (8) that gives the numerical solution of (1) in the first subinterval $[t_0,\ t_1]$ (the initial condition is exact), in this phase we are only interested in the solution, $z_i \approx z(t_i)$, at the end point of each interval $[t_{i-1},\ t_i]$, for $i = 2,\dots,q-1$. Observe that, at this moment, we are not interested at all to the last sub-interval $[t_{q-1},\ t_q]$.

Phase 2. Sequential computation of the $q-2$ approximations $x_i \approx \mathrm{e}^{(t_i-t_{i-1})A}y_{i-1}$ and, then, of the update $y_i = x_i + z_i$, for $i = 2,\dots,q-1$. We observe that the approximation of the above exponentials, corresponds to the computation of an approximate solution to the following recursively defined problems:

$$\begin{aligned} x_1 &= y_1, \\ x' &= Ax, \qquad t \in [t_{i-1},\ t_i], \\ x(t_{i-1}) &= x_{i-1} + z_{i-1}, \qquad\qquad i = 2,\dots,q-1. \\ x_i &:= x(t_i), \end{aligned}$$

This aspect will be addressed in some detail in the next section. We only observe that, in this phase, we essentially obtain a numerical approximation to the solution of the reduced system (6).

Phase 3. Parallel solution of the $q-1$ ODE-IVPs:

$$\begin{aligned} &y' = Ay + b(t), \qquad t \in [t_{i-1},\ t_i], \\ &y(t_{i-1}) = y_{i-1}, \qquad\qquad i = 2,\dots,q. \end{aligned}$$

Obviously, this can be done by means of the same discrete method used in the first phase.

Remark. We observe that, when a parallel computing platform with p processors is used, then the number q of sub-problems (2) is conveniently chosen as:

$$q = p + 1. \tag{10}$$

In such a case, in fact, we use the first processor for the sub-interval $[t_0,\ t_1]$ in the first phase, and for the sub-interval $[t_{q-1},\ t_q]$ in the third phase.

SOLUTION OF THE SECOND PHASE

The computation of approximations of the exponential of a matrix has been largely investigated, starting from the sixties. There is a rich literature available for this topic, concerning approaches following quite different strategies depending on the size and the sparsity pattern of the coefficient matrix. See, for example, the review paper [12].

Obviously the problem is not that of estimating the matrix exponential, but only that of computing the product of the exponential times a vector, say $\mathrm{e}^{hA}y$ (here h denotes the witdh of the generic sub-interval in (2). One of the most interesting approach consists in approximating the matrix exponential by means of a Krylov subspace method. In [4] it was suggested to obtain a low-rank approximation of A in the form

$$A \approx U_k T_k U_k^T, \tag{11}$$

where $U_k \in \mathbb{R}^{n\times k}$ is the matrix, with orthogonal columns, made-up with the first k Arnoldi vectors (the first one being fixed as $y/\|y\|_2$), and T_k is upper Hessemberg. However, in [13] it is suggested to use a fixed shift parameter $\gamma > 0$ and to consider the Arnoldi algorithm for obtaining the approximation, analogous to (11),

$$(I-\gamma A)^{-1} \approx \hat{U}_{\hat{k}} \hat{T}_{\hat{k}} \hat{U}_{\hat{k}}^T, \tag{12}$$

and hence

$$\mathrm{e}^{hA}y = \hat{U}_{\hat{k}} \mathrm{e}^{h\widetilde{T}_{\hat{k}}} \hat{U}_{\hat{k}}^T y = \|y\|\ \hat{U}_{\hat{k}} \mathrm{e}^{h\widetilde{T}_{\hat{k}}} e_1$$

being e_1 the first unit vector and $\widetilde{T}_{\hat{k}} = \gamma^{-1}(I - \hat{T}_{\hat{k}}^{-1})$. Clearly, each Arnoldi iterate for getting (12) is more expensive than the corresponding iterate for getting (11). Nevertheless, if γ is appropriately chosen, it turns out that, for a given accuracy requirement, the number of Arnoldi iterates in (12), $\hat{k}$, is much smaller than that in (11) (i.e., k). Consequently, when the dimension n of problem (1) is large enough, the computational cost of the second phase of the algorithm (which is the only sequential one) is negligible with respect to that of the phases 1 and 3, which are fully parallel. Hence, one can expect a maximum speed-up which is close to $p/2$, if p (see (10)) parallel processors are used.

REFERENCES

1. P. Amodio and L. Brugnano, *Parallel factorizations and parallel solvers for tridiagonal linear systems*, Linear Algebra Appl. **172** (1992), pp. 347–364.
2. P. Amodio and L. Brugnano, *Parallel implementation of block boundary value methods for ODEs*, J. Comput. Appl. Math. **78** (1997), pp. 197–211.
3. P. Amodio and L. Brugnano, *Parallel ODE solvers based on block BVMs*, Adv. Comput. Math. **7** (1997), pp. 5–26.
4. P. Amodio and L. Brugnano, *Parallel solution in time of ODEs: some achievements and perspectives*, Appl. Numer. Math. (2008), doi:10.1016/j.apnum.2008.03.024
5. P. Amodio, L. Brugnano and T. Politi, *Parallel factorizations for tridiagonal matrices*, SIAM J. Numer. Anal. **30** (1993), pp. 813–823.
6. K. Burrage, *Parallel and Sequential Methods for Ordinary Differential Equations*, Clarendon Press, Oxford, 1995.
7. K. Burrage (Ed.), *Parallel Methods for ODEs*, Advances in Computational Mathematics **7**, 1-2 (1997).
8. M.J. Gander and S. Vandewalle, *Analysis of the parareal time–parallel time–integration method*, SIAM J. Sci. Comput. **29** (2007), pp. 556–578.
9. J.L. Lions, Y. Maday and G. Turinici, *Résolution d'EDP par un schéma en temps "pararéel"*, C.R. Acad. Sci. Paris, Ser. I **332** (2001), pp. 661–668.
10. Y. Maday and G. Turinici, *A parareal in time procedure for the control of partial differential equations*, C.R. Acad. Sci. Paris, Ser. I **335** (2002), pp. 387–392.
11. Y. Maday, E.M. Rønquist, and G.A. Staff, *The parareal–in–time algorithm: basics, stability and more*, submitted, 2006.
12. C. Moler and C. Van Loan, *Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later*, SIAM Rev. **45** (2003), pp. 3–49.
13. J. van den Eshof and M. Hochbruck, *Preconditioning Lanczos approximations to the matrix exponential*, SIAM J. Sci. Comput. **27** (2006), pp. 1438–1457.