Parallel ODE solvers based on block BVMs *

Pierluigi Amodio [a] and Luigi Brugnano [b]

[a] *Dipartimento di Matematica, Università di Bari, Via Orabona 4, I-70125 Bari, Italy*
E-mail: na.pamodio@na-net.ornl.gov
[b] *Dipartimento di Matematica, Università di Firenze, Viale Morgagni 67/A, I-50134 Firenze, Italy*
E-mail: na.brugnano@na-net.ornl-gov

In this paper we deal with Boundary Value Methods (BVMs), which are methods recently introduced for the numerical approximation of initial value problems for ODEs. Such methods, based on linear multistep formulae (LMF), overcome the stability limitations due to the well-known Dahlquist barriers, and have been the subject of much research in the last years. This has led to the definition of a new stability framework, which generalizes the one stated by Dahlquist for LMF. Moreover, several aspects have been investigated, including the efficient stepsize control [17,25,26] and the application of the methods for approximating different kinds of problems such as BVPs, PDEs and DAEs [7,23,41]. Furthermore, a block version of such methods, recently proposed for approximating Hamiltonian problems [24], is able to provide an efficient parallel solver for ODE systems [3].

1. Introduction

The numerical approximation of ODE problems has been, and continues to be, a very active field of investigation (see, for example, [29]). Several tracks are currently followed, ranging from the derivation of new methods to the parallel implementation of the existing ones.

In the last few years the search for parallel ODE solvers has led to the definition of a new class of methods called Boundary Value Methods (BVMs). The basic idea on which such methods rely is that of approximating a continuous initial value problem (IVP) by means of a discrete boundary value one (BVP). This idea is established for the approximation of discrete problems (it goes back to Miller [42] and Olver [43]), but is rather unusual for the approximation of continuous ones, even if earlier references exist [15,30,31].

The fact of having a discrete boundary value problem to be solved implies that the solution cannot, in general, be obtained in a "step by step" fashion. Conversely, it must be obtained globally, thus giving rise to methods which do have a natural parallelism in time. As a consequence, the parallel solution of the obtained discrete problem provides, as a by-product, a parallel ODE solver. This was, in fact, the first approach to the implementation of the methods [8–10,13,18,21,40].

* Work supported by CNR, contract n. 96.00243.CT01.

Since the very beginning of research in this area, it was evident that the derivation of the methods would have been difficult, without having a new stability framework. Such a framework should obviously be more general with respect to the one stated by Dahlquist, since the nature of the discrete problem is different. Consequently, when developing BVMs the first goal has been deriving a new stability theory (preliminary results are in [6,11,19,20,36,37], the linear stability theory is stated in [22], and related results are in [34,35]). This has allowed the derivation of whole families of methods [1,6,16,17,27].

Some of the methods have been found to be also able to approximate both continuous BVPs [23] and Hamiltonian problems [16,24,44]. In particular, the study of the methods for the latter problems has led to define a block version of BVMs [24], from which an efficient parallel implementation of the methods has recently been derived [3].

In this paper we shall give a survey on BVMs (a complete treatment can be found in the book [28], to be published soon). In particular, section 2 is devoted to a review of the new stability definitions. In section 3 the main families of BVMs are recalled. Section 4 is devoted to the block version of the methods and, finally, in section 5 their parallel implementation is presented.

2. Boundary value methods

Consider the approximation of an initial value ODE problem,

$$y' = f(t, y), \quad t \in [t_0, T], \qquad y(t_0) = \eta, \tag{1}$$

obtained by using a k-step LMF,

$$\sum_{i=0}^{k} \alpha_i y_{n+i} = h \sum_{i=0}^{k} \beta_i f_{n+i}, \tag{2}$$

where, as usual, y_n denotes the discrete approximation of the solution $y(t)$ at $t = t_n \equiv t_0 + nh$, $h = (T - t_0)/N$, and $f_n = f(t_n, y_n)$. Since (2) is a kth order difference equation, then k independent conditions need to be imposed to obtain the discrete solution. It is customary to assign such conditions by fixing the first k values, $y_0, \ldots, y_{k-1}$, of the discrete solution. That is, the continuous IVP (1) is approximated by means of a discrete IVP. Even if this approach is straightforward, it suffers from heavy theoretical limitations, summarized by the two Dahlquist barriers.

An alternative approach consists in fixing the first $k_1 \leqslant k$ values of the discrete solution, $y_0, \ldots, y_{k_1-1}$, and the last $k_2 = k - k_1$ ones, $y_{N-k_2+1}, \ldots, y_N$. In this way, the discrete problem becomes

$$\sum_{i=-k_1}^{k_2} \alpha_{i+k_1} y_{n+i} = h \sum_{i=-k_1}^{k_2} \beta_{i+k_1} f_{n+i}, \quad n = k_1, \ldots, N - k_2, \tag{3}$$

$$y_0, \ldots, y_{k_1-1}, y_{N-k_2+1}, \ldots, y_N \quad \text{fixed.}$$

That is, the continuous IVP (1) is approximated by means of a discrete BVP which defines a BVM with (k_1, k_2)-*boundary conditions*. Observe that, by choosing $k_1 = k$ and $k_2 = 0$, problem (3) becomes a discrete IVP, so that BVMs contain as a proper subclass the usual initial value methods for ODEs based on LMF.

In order to fully exploit all the potentiality of this new approach, we need to generalize the known stability notions. Before that, we need to report some preliminary definitions [22].

Definition 1. Let $p(z)$ be a polynomial of degree k, and let $|z_1| \leqslant \cdots \leqslant |z_k|$ be its zeros. The polynomial $p(z)$ is said to be an

- $S_{k_1k_2}$-*polynomial* if $|z_{k_1}| < 1 < |z_{k_1+1}|$;
- $N_{k_1k_2}$-*polynomial* if $|z_{k_1}| \leqslant 1 < |z_{k_1+1}|$, with simple zeros of unit modulus.

For $k_1 = k$ and $k_2 = 0$, the above definitions reduce to the usual ones for Schur polynomials and von Neumann polynomials, respectively.

Denoting by $\rho(z)$ and $\sigma(z)$ the two polynomials associated with the LMF (2), and by $\pi(z, q) = \rho(z) - q\sigma(z)$ the corresponding stability polynomial, the following definitions are then stated [22].

Definition 2. A BVM with (k_1, k_2)-boundary conditions is

- $0_{k_1k_2}$-*stable* if the corresponding polynomial $\rho(z)$ is an $N_{k_1k_2}$-polynomial;
- (k_1, k_2)-*absolutely stable* for a given $q \in \mathbb{C}$ if the polynomial $\pi(z, q)$ is an $S_{k_1k_2}$-polynomial. The region

  $$D_{k_1k_2} = \left\{q \in \mathbb{C}\colon\ \pi(z, q) \text{ is an } S_{k_1k_2}\text{-polynomial}\right\}$$

  is said to be the *region of* (k_1, k_2)-*absolute stability*;
- $A_{k_1k_2}$-*stable* if $\mathbb{C}^- \subseteq D_{k_1k_2}$.

As in the previous case, the above definitions reduce to the usual stability notions when $k_1 = k$ and $k_2 = 0$, even though now there are no more order barriers for stable methods. In fact, there exist stable methods of arbitrary high order, as we shall see in section 3.

2.1. The additional methods

In order to get the discrete problem (3), the k values

$$y_0, y_1, \ldots, y_{k_1-1}, y_{N-k_2+1}, \ldots, y_N,$$

are needed. Of such values, only the initial condition y_0 is provided by the continuous problem and, at first glance, it may seem difficult to obtain approximations for the

others, especially those at the final points. This problem, however, is easily overcome by treating the $k-1$ additional values as unknowns. This is done by introducing a set of $k-1$ *additional equations* independent of those provided by the *main formula* (3). Such equations are conveniently derived by a set of k_1-1 *additional initial methods*,

$$\sum_{i=0}^{r} \alpha_i^{(j)} y_i = h \sum_{i=0}^{r} \beta_i^{(j)} f_i, \quad j = 1, \ldots, k_1 - 1, \tag{4}$$

and k_2 *final* ones,

$$\sum_{i=0}^{r} \alpha_{r-i}^{(j)} y_{N-i} = h \sum_{i=0}^{r} \beta_{r-i}^{(j)} f_{N-i}, \quad j = N - k_2 + 1, \ldots, N, \tag{5}$$

having the same order of the main formula (3). In this way, the whole composite method (3)–(5) has the same order of the main formula (3). For this reason, each BVM will be coupled with the most appropriate set of additional methods and can be regarded as a *composite* scheme.

3. Families of BVMs

In this section, we present the most important families of BVMs. Each family of methods contains $0_{k_1 k_2}$-stable and $A_{k_1 k_2}$-stable methods of arbitrary high order. This, in turn, confirms the absence of order barriers for stable BVMs. We first analyze the Generalized Backward Differentiation Formulae (GBDF). Then we consider the Generalized Adams Methods (GAMs) and the Extended Trapezoidal Rules of second kind (ETR_2s). Finally, we examine the Symmetric Schemes.

3.1. Generalized BDF (GBDF)

For all $k \geqslant 1$, these methods have the form [22],

$$\sum_{i=-\nu}^{k-\nu} \alpha_{i+\nu} y_{n+i} = h f_n, \quad n = \nu, \ldots, N - k + \nu, \tag{6}$$

where

$$\nu = \begin{cases} (k+2)/2, & \text{for even } k, \\ (k+1)/2, & \text{for odd } k. \end{cases}$$

The coefficients $\{\alpha_i\}$ are uniquely determined by imposing that formula (6) has the highest possible order, k. Such a formula must be used with $(\nu, k-\nu)$-boundary conditions and is conveniently coupled with the following set of additional initial equations,

$$\sum_{i=0}^{k} \alpha_i^{(j)} y_i = h f_j, \quad j = 1, \ldots, \nu - 1,$$

and additional final ones,

$$\sum_{i=0}^{k} \alpha_{k-i}^{(j)} y_{N-i} = h f_j, \quad j = N-k+\nu+1, \ldots, N.$$

The coefficients of the additional formulae are uniquely determined in order to have the truncation error of the same order $O(h^{k+1})$ of the main formula (6).

It is well known that BDF, obtained from (6) by setting $\nu = k$, are not 0-stable for $k \geqslant 7$, whereas for any $k \geqslant 1$ GBDF are $0_{\nu,k-\nu}$-stable, $A_{\nu,k-\nu}$-stable, and have order of convergence k.

As examples, we mention that for $k = 1$ and $k = 2$ one obtains the usual BDF of order 1 and 2, respectively. For $k = 3$ one obtains the following third order method,

$$\frac{1}{6}(2y_{n+1} + 3y_n - 6y_{n-1} + y_{n-2}) = h f_n, \quad n = 2, \ldots, N-1,$$

which is coupled with the following additional initial equation,

$$\frac{1}{6}(-y_3 + 6y_2 - 3y_1 - 2y_0) = h f_1,$$

and the following final one,

$$\frac{1}{6}(11y_N - 18y_{N-1} + 9y_{N-2} - 2y_{N-3}) = h f_N.$$

3.2. Generalized Adams methods (GAMs)

The methods in this family are of the form [27]

$$y_n - y_{n-1} = h \sum_{i=-\nu}^{k-\nu} \beta_{i+\nu} f_{n+i}, \quad n = \nu, \ldots, N-k+\nu, \tag{7}$$

where

$$\nu = \begin{cases} (k+1)/2, & \text{for odd } k, \\ k/2, & \text{for even } k, \end{cases} \tag{8}$$

and the coefficients $\{\beta_i\}$ are uniquely determined by imposing that the method has order $k+1$. For each $k \geqslant 1$, they must be used with $(\nu, k-\nu)$-boundary conditions and are both $0_{\nu,k-\nu}$-stable and $A_{\nu,k-\nu}$-stable. They are conveniently used with the following set of additional initial equations,

$$y_j - y_{j-1} = h \sum_{i=0}^{k} \beta_i^{(j)} f_i, \quad j = 1, \ldots, \nu-1,$$

and additional final ones,

$$y_j - y_{j-1} = h\sum_{i=0}^{k} \beta_{k-i}^{(j)} f_{N-i}, \quad j = N-k+\nu+1, \ldots, N.$$

The coefficients of the additional methods are uniquely determined by imposing that each formula has the same order $k+1$ of the main method (7).

There is a difference in the stability properties between the GAMs with odd k and those having k even. For this reason, the formulae obtained in correspondence of the odd values of k are also called *Extended Trapezoidal Rules (ETRs)* [6], since the formula obtained for $k = 1$ is the trapezoidal rule. Such formulae belong to the class of *symmetric schemes* that we shall consider later.

Observe that when in (7) one considers $\nu = k$, the Adams–Moulton methods are obtained, which have bounded stability regions for all $k \geqslant 2$.

As examples, when $k = 2$ one obtains the third order GAM,

$$y_n - y_{n-1} = \frac{h}{12}(-f_{n+1} + 8f_n + 5f_{n-1}), \quad n = 1, \ldots, N-1,$$

which can be used with the following additional final equation,

$$y_N - y_{N-1} = \frac{h}{12}(5f_N + 8f_{N-1} - f_{N-2}).$$

When $k = 3$ one obtains the fourth order ETR,

$$y_n - y_{n-1} = \frac{h}{24}(-f_{n+1} + 13f_n + 13f_{n-1} - f_{n-2}), \quad n = 2, \ldots, N-1,$$

which can be used with the following additional initial equation,

$$y_1 - y_0 = \frac{h}{24}(f_3 - 5f_2 + 19f_1 + 9f_0),$$

and the following final one,

$$y_N - y_{N-1} = \frac{h}{24}(9f_N + 19f_{N-1} - 5f_{N-2} + f_{N-3}).$$

3.3. Extended Trapezoidal Rules of second kind (ETR$_2$s)

Let us consider the methods having the following general form,

$$\sum_{i=-\nu}^{k-\nu} \alpha_{i+\nu} y_{n+i} = h\big(\beta f_n + (1-\beta) f_{n-1}\big), \quad n = \nu, \ldots, N-k+\nu, \tag{9}$$

where ν is chosen according to (8), and the coefficients $\{\alpha_i\}$ and β are uniquely determined by imposing a $O(h^{k+2})$ truncation error. The formulae obtained for k even are called *unsymmetric ETR$_2$s* [17], while those obtained for k odd are called *ETR$_2$s* [16]. The name of such formulae derives from the fact that, for $k = 1$ one obtains the

trapezoidal rule. Moreover, for k odd one obtains $\beta = 1 - \beta = 1/2$, so that the right hand side in formula (9) equals that of the trapezoidal rule. In such a case, the method belongs to the class of *symmetric schemes* that we shall consider in the next section.

All BVMs (9) are $0_{\nu,k-\nu}$-stable, $A_{\nu,k-\nu}$-stable formulae, and must be used with $(\nu, k-\nu)$-boundary conditions. The following set of additional initial equations,

$$\sum_{i=0}^{k} \alpha_i^{(j)} y_i = h\big(\beta^{(j)} f_j + \big(1-\beta^{(j)}\big) f_{j-1}\big), \quad j = 1, \ldots, \nu-1,$$

and additional final ones,

$$\sum_{i=0}^{k} \alpha_{k-i}^{(j)} y_{N-i} = h\big(\beta^{(j)} f_j + \big(1-\beta^{(j)}\big) f_{j-1}\big), \quad j = N-k+\nu+1, \ldots, N,$$

are conveniently associated with the main formula (9). The coefficients of each additional method are uniquely determined by imposing the same order, $k+1$, of the main formula.

As an example, when $k = 2$ one obtains the third order unsymmetric ETR_2,

$$\frac{1}{6}(y_{n+1} + 4y_n - 5y_{n-1}) = \frac{h}{3}(2f_n + f_{n-1}), \quad n = 1, \ldots, N-1,$$

which is conveniently used with the following final additional equation,

$$\frac{1}{6}(5y_N - 4y_{N-1} - y_{N-2}) = \frac{h}{3}(f_N + 2f_{N-1}).$$

When $k = 3$ one obtains the fourth order ETR_2,

$$\frac{1}{12}(y_{n+1} + 9y_n - 9y_{n-1} - y_{n-2}) = \frac{h}{2}(f_n + f_{n-1}), \quad n = 2, \ldots, N-1,$$

which is used with the following additional initial equation,

$$\frac{1}{24}(-y_3 + 9y_2 + 9y_2 - 17y_0) = \frac{h}{4}(3f_1 + f_0),$$

and the following final one,

$$\frac{1}{24}(17y_N - 9y_{N-1} - 9y_{N-2} + y_{N-3}) = \frac{h}{4}(f_N + 3f_{N-1}).$$

3.4. Symmetric schemes

We group as *symmetric schemes* BVMs having the following general properties:

- an odd number of steps, $k = 2\nu - 1$, and $(\nu, \nu-1)$-boundary conditions (i.e., they require $\nu - 1$ initial and $\nu - 1$ final additional methods);
- the corresponding polynomials $\rho(z)$ have skew-symmetric coefficients, that is $z^k\rho(z^{-1}) = -\rho(z)$, while the polynomials $\sigma(z)$ have symmetric coefficients, that is $z^k\sigma(z^{-1}) = \sigma(z)$;

- $D_{\nu,\nu-1} \equiv \mathbb{C}^-$ (their boundary loci coincide, in fact, with the imaginary axis);
- for $k = 1$, one obtains the basic trapezoidal rule.

Such schemes are also conveniently used for approximating both continuous BVPs [23] and Hamiltonian problems [16,24,44].

ETRs and ETR$_2$s are symmetric schemes. Another important family of BVMs fitting this class is that of *Top Order Methods (TOMs)* [1]. They are methods of the form

$$\sum_{i=0}^{\nu-1} \alpha_i (y_{n-\nu+i} - y_{n+\nu-1-i}) = h \sum_{i=0}^{\nu-1} \beta_i (f_{n-\nu+i} + f_{n+\nu-1-i}), \qquad (10)$$
$$n = \nu, \ldots, N - \nu + 1,$$

where the coefficients $\{\alpha_i\}$ and $\{\beta_i\}$ are determined in order to have the maximum possible order for a k-step formula, that is $2k$ (and this is the reason for their name). They can be conveniently used with the following initial equations,

$$y_j - y_{j-1} = h \sum_{i=0}^{2k-1} \beta_i^{(j)} f_i, \quad j = 1, \ldots, \nu - 1,$$

and final additional equations,

$$y_j - y_{j-1} = h \sum_{i=0}^{2k-1} \beta_{2k-1-i}^{(j)} f_{N-i}, \quad j = N - \nu + 2, \ldots, N,$$

where the unknown coefficients $\{\beta_i^{(j)}\}$ are uniquely determined by imposing each formula to have the same order $2k$ of the main formula (10).

We observe that such methods were introduced by Dahlquist as examples of unstable initial value methods [32], whereas they are perfectly stable when used as BVMs.

As an example, for $k = 3$ one obtains the sixth order TOM,

$$\frac{1}{60}(y_{n+1} + 27y_n - 27y_{n-1} - y_{n-2}) = \frac{h}{20}(f_{n+1} + 9f_n + 9f_{n-1} + f_{n-2}),$$
$$n = 2, \ldots, N - 1,$$

to be used with the following additional initial and final equations,

$$y_1 - y_0 = \frac{h}{1440}(27f_5 - 173f_4 + 482f_3 - 798f_2 + 1427f_1 + 475f_0),$$

and

$$y_N - y_{N-1} = \frac{h}{1440}(475f_N + 1427f_{N-1} - 798f_{N-2} + 482f_{N-3} - 173f_{N-4} + 27f_{N-5}).$$

4. Block BVMs

We now introduce a block version of BVMs, which will be very effective in order to derive an efficient parallel implementation of the methods. The basic idea on which the block version is based is that of approximating the given continuous IVP by means of a set of discrete BVPs defined on contiguous subintervals, where there is only one point of connection between two consecutive problems (the final point of each BVP is the initial one of the BVP defined in the subsequent interval).

For simplicity, we consider a linear problem of size m,

$$y' = L(t)y + g(t), \quad t \in [t_0, T], \qquad y(t_0) = \eta. \tag{11}$$

Let us then apply to it a BVM (used as a composite method) with (k_1, k_2)-boundary conditions and stepsize $h = (T - t_0)/s$. If $L_i = L(t_i)$, the following discrete problem is obtained,

$$\left(\begin{pmatrix} 1 & \mathbf{0}_s^T \\ \boldsymbol{a} & A \end{pmatrix} \otimes I_m - h \begin{pmatrix} 0 & \mathbf{0}_s^T \\ \boldsymbol{b} & B \end{pmatrix} \otimes I_m \begin{pmatrix} L_0 & & \\ & \ddots & \\ & & L_s \end{pmatrix} \right) \begin{pmatrix} y_0 \\ \boldsymbol{y} \end{pmatrix} = \begin{pmatrix} \eta \\ h\boldsymbol{g} \end{pmatrix},$$

where, for any integer r, $\mathbf{0}_r$ denotes the null vector in $\mathbb{R}^r$ and I_r is the identity matrix of size r. Moreover, the vector

$$\boldsymbol{y} = (y_1, \ldots, y_s)^T$$

contains the discrete approximation, the vector $\boldsymbol{g}$ depends on the inhomogeneity of the continuous problem, and

$$[\boldsymbol{a} \mid A] = \left(\begin{array}{c|cccccccc} \alpha_0^{(1)} & \alpha_1^{(1)} & \dots & \alpha_k^{(1)} & & & & & \\ \vdots & \vdots & & \vdots & & & & & \\ \alpha_0^{(k_1-1)} & \alpha_1^{(k_1-1)} & \dots & \alpha_k^{(k_1-1)} & & & & & \\ \alpha_0 & \alpha_1 & \dots & \alpha_k & & & & & \\ & \alpha_0 & \alpha_1 & \dots & \alpha_k & & & & \\ & & \ddots & \ddots & & \ddots & & & \\ & & & \ddots & \ddots & & \ddots & & \\ & & & & \alpha_0 & \alpha_1 & \dots & \alpha_k & \\ & & & & \alpha_0^{(s-k_2+1)} & \alpha_1^{(s-k_2+1)} & \dots & \alpha_k^{(s-k_2+1)} & \\ & & & & \vdots & \vdots & & \vdots & \\ & & & & \alpha_0^{(s)} & \alpha_1^{(s)} & \dots & \alpha_k^{(s)} & \end{array} \right)_{s \times (s+1)},$$

$$[\boldsymbol{b}\mid B]=\left(\begin{array}{c|cccccccc}
\beta_0^{(1)} & \beta_1^{(1)} & \dots & \beta_k^{(1)} & & & & \\
\vdots & \vdots & & \vdots & & & & \\
\beta_0^{(k_1-1)} & \beta_1^{(k_1-1)} & \dots & \beta_k^{(k_1-1)} & & & & \\
\beta_0 & \beta_1 & \dots & \beta_k & & & & \\
 & \beta_0 & \beta_1 & \cdots & \beta_k & & & \\
 & & \ddots & \ddots & & \ddots & & \\
 & & & \ddots & \ddots & & \ddots & \\
 & & & & \beta_0 & \beta_1 & \dots & \beta_k \\
 & & & & \beta_0^{(s-k_2+1)} & \beta_1^{(s-k_2+1)} & \dots & \beta_k^{(s-k_2+1)} \\
 & & & & \vdots & \vdots & & \vdots \\
 & & & & \beta_0^{(s)} & \beta_1^{(s)} & \dots & \beta_k^{(s)}
\end{array}\right)_{s\times(s+1)}.$$

For simplicity, in the above expressions we have assumed that the additional methods, defining the first and last group of rows in the matrices $[\boldsymbol{a}\,|\,A]$ and $[\boldsymbol{b}\,|\,B]$, have the same number of steps k of the main formula (this is, in fact, the case for most of the methods examined in the previous section).

Observe that the block matrix

$$M = A\otimes I_m - hB\otimes I_m \begin{pmatrix} L_1 & & \\ & \ddots & \\ & & L_s \end{pmatrix}$$

needs to be nonsingular, for the discrete solution to be defined.

We observe that, in such form, a BVM essentially becomes a one-step method. This allows to define the block version of the methods [24], which consists in using two different meshes: a *coarse mesh*, containing the $p+1$ points

$$\tau_i = \tau_{i-1} + \widehat{h}_i, \quad i=1,\dots,p, \qquad \tau_0\equiv t_0, \qquad \tau_p\equiv T,$$

and a *fine mesh* which discretizes each subinterval $[\tau_{i-1},\tau_i]$ by using a stepsize $h_i = \widehat{h}_i/s$. The following *internal steps*,

$$t_{ji} = \tau_{i-1} + jh_i, \quad j=1,\dots,s,\ i=1,\dots,p,$$

are then defined, where the rightmost lower index of t_{ji} identifies the ith subinterval. On each subinterval $[\tau_{i-1},\tau_i]$, one then applies the same composite BVM, thus obtaining a *block BVM with s internal steps.*

Let us denote by $L_{ji} = L(t_{ji})$, and introduce the block matrices

$$\begin{aligned}
&\boldsymbol{v}_i = \boldsymbol{a}\otimes I_m - h\boldsymbol{b}\otimes L_{0i},\\
&V_i = [\widehat{O}_{s,s-1}\,|\,\boldsymbol{v}_i],\\
&M_i = A\otimes I_m - h_iB\otimes I_m \begin{pmatrix} L_{1i} & & \\ & \ddots & \\ & & L_{si} \end{pmatrix},
\end{aligned}$$

where, denoting by $O_{s,s-1}$ the $s \times (s-1)$ zero matrix and, for any integer r, with O_r the $r \times r$ zero matrix,

$$\widehat{O}_{s,s-1} = O_{s,s-1} \otimes O_m.$$

Then, the application of a block BVM to problem (11) leads to the following discrete problem,

$$M^{(p)} \boldsymbol{y}^{(p)} = \boldsymbol{g}^{(p)}, \tag{12}$$

where

$$M^{(p)} = \begin{pmatrix} I_m & & & & \\ \boldsymbol{v}_1 & M_1 & & & \\ & V_2 & M_2 & & \\ & & \ddots & \ddots & \\ & & & V_p & M_p \end{pmatrix}, \qquad \boldsymbol{y}^{(p)} = \begin{pmatrix} y_0 \\ \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \\ \vdots \\ \boldsymbol{y}_p \end{pmatrix}, \qquad \boldsymbol{g}^{(p)} = \begin{pmatrix} \eta \\ h_1 \boldsymbol{g}_1 \\ h_2 \boldsymbol{g}_2 \\ \vdots \\ h_p \boldsymbol{g}_p \end{pmatrix},$$

the block vector $\boldsymbol{y}_i$ contains the discrete approximation to the solution at the internal steps belonging to the ith subinterval in the coarse mesh, and $\boldsymbol{g}_i$ is a block vector containing suitable combinations of the inhomogeneity in (11) at the points in the same subinterval.

The above block version of BVMs has relevant implications when approximating continuous Hamiltonian systems by using symmetric schemes [24]. Moreover, it has a twofold effect on the implementation of the methods:

(i) It is possible to implement variable stepsize strategies by simply varying the stepsize in the coarse mesh, while the internal steps remain equally spaced (or fixed in some way) [17,26].

(ii) It is possible to derive an efficient parallel ODE solver [3]. In fact, the parallel implementation of block BVMs is based on the particular structure of the coefficient matrix $M^{(p)}$.

5. The parallel implementation

We now examine the parallel implementation of BVMs which exploit a sort of *parallel factorization* [2,4] of the matrix $M^{(p)}$ in (12). In fact, we factor such a matrix as

$$M^{(p)} = D_1^{(p)} D_2^{(p)}, \tag{13}$$

where

$$D_1^{(p)} = \begin{pmatrix} I_m & & & \\ & M_1 & & \\ & & \ddots & \\ & & & M_p \end{pmatrix}$$

and

$$D_2^{(p)} = \begin{pmatrix} I_m & & & & \\ \boldsymbol{w}_1 & \widehat{I}_s & & & \\ & W_2 & \widehat{I}_s & & \\ & & \ddots & \ddots & \\ & & & W_p & \widehat{I}_s \end{pmatrix}.$$

Hereafter, for any integer r,

$$\widehat{I}_r = I_r \otimes I_m,$$

and, for all allowed i, $W_i = [\widehat{O}_{s,s-1} \mid \boldsymbol{w}_i]$, where $\boldsymbol{w}_i$ is the solution of the linear system

$$M_i \boldsymbol{w}_i = \boldsymbol{v}_i, \quad i = 1, \ldots, p, \tag{14}$$

obtained by means of the LU factorization with partial pivoting algorithm. Of course, each block M_i, which represents the application of the method over the subinterval $[\tau_{i-1}, \tau_i]$ of the coarse mesh, must be nonsingular. Observe that the factorizations of the p blocks M_i, as well as the solution of the linear systems (14), can be performed independently on p processors.

The solution of equation (12) is then obtained by first solving the block diagonal system

$$D_1^{(p)} \boldsymbol{x}^{(p)} = \boldsymbol{g}^{(p)}, \tag{15}$$

and then the block lower bidiagonal one,

$$D_2^{(p)} \boldsymbol{y}^{(p)} = \boldsymbol{x}^{(p)}. \tag{16}$$

For convenience, the auxiliary vector $\boldsymbol{x}^{(p)}$ will be partitioned as $\boldsymbol{y}^{(p)}$, that is,

$$\boldsymbol{x}^{(p)} = \begin{pmatrix} x_0 \\ \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_p \end{pmatrix}, \quad x_0 \in \mathbb{R}^m, \ \boldsymbol{x}_i \in \mathbb{R}^{sm}.$$

Owing to its block diagonal structure, the solution of equation (15) can be obtained in parallel on p processors. This is not the case for equation (16), even if some straight parallelism is still present. To make this evident, we consider a further partition of the block vectors $\boldsymbol{x}_i$, $\boldsymbol{y}_i$ and $\boldsymbol{w}_i$,

$$\boldsymbol{x}_i = \begin{pmatrix} \widehat{\boldsymbol{x}}_i \\ x_{si} \end{pmatrix}, \quad \boldsymbol{y}_i = \begin{pmatrix} \widehat{\boldsymbol{y}}_i \\ y_{si} \end{pmatrix}, \quad \boldsymbol{w}_i = \begin{pmatrix} \widehat{\boldsymbol{w}}_i \\ w_{si} \end{pmatrix}, \qquad i = 1, \ldots, p,$$

where $x_{si}, y_{si} \in \mathbb{R}^m$ and $w_{si} \in \mathbb{R}^{m\times m}$. Moreover, we introduce the following $(ps+1)m \times (ps+1)m$ permutation matrix,

$$Q = \begin{pmatrix} \widehat{\mathbf{0}}_{s-1} & \widehat{I}_{s-1} & \widehat{\mathbf{0}}_{s-1} & \cdots & & & & \\ \widehat{\mathbf{0}}_{s-1} & \widehat{O}_{s-1} & \widehat{\mathbf{0}}_{s-1} & \widehat{I}_{s-1} & \widehat{\mathbf{0}}_{s-1} & \cdots & & \\ \widehat{\mathbf{0}}_{s-1} & \widehat{O}_{s-1} & \widehat{\mathbf{0}}_{s-1} & \widehat{O}_{s-1} & \widehat{\mathbf{0}}_{s-1} & \widehat{I}_{s-1} & \widehat{\mathbf{0}}_{s-1} & \cdots \\ & & \cdots & \cdots & & & & \\ I_m & \widehat{\mathbf{0}}^T_{s-1} & \cdots & & & & & \\ O_m & \widehat{\mathbf{0}}^T_{s-1} & I_m & \widehat{\mathbf{0}}^T_{s-1} & \cdots & & & \\ O_m & \widehat{\mathbf{0}}^T_{s-1} & O_m & \widehat{\mathbf{0}}^T_{s-1} & I_m & \widehat{\mathbf{0}}^T_{s-1} & \cdots & \\ & & \cdots & \cdots & & & & \end{pmatrix}, \tag{17}$$

where $\widehat{\mathbf{0}}_{s-1} = \mathbf{0}_{s-1} \otimes O_m$. The following permuted system,

$$QD_2^{(p)}Q^TQ\boldsymbol{y}^{(p)} = Q\boldsymbol{x}^{(p)},$$

is then considered. It is a 2×2 upper block triangular system, namely,

$$\left(\begin{array}{ccccc|ccccc} \widehat{I}_{s-1} & & & & & \widehat{\boldsymbol{w}}_1 & O_m & & & \\ & \widehat{I}_{s-1} & & & & & \widehat{\boldsymbol{w}}_2 & O_m & & \\ & & \ddots & & & & & \ddots & \ddots & \\ & & & \widehat{I}_{s-1} & & & & & \widehat{\boldsymbol{w}}_p & O_m \\ \hline & & & & & I_m & & & & \\ & & & & & w_{s1} & I_m & & & \\ & & & & & & w_{s2} & I_m & & \\ & & O & & & & & \ddots & \ddots & \\ & & & & & & & & w_{sp} & I_m \end{array}\right) \left(\begin{array}{c} \widehat{\boldsymbol{y}}_1 \\ \widehat{\boldsymbol{y}}_2 \\ \vdots \\ \widehat{\boldsymbol{y}}_p \\ \hline y_0 \\ y_{s1} \\ \vdots \\ y_{sp} \end{array}\right) = \left(\begin{array}{c} \widehat{\boldsymbol{x}}_1 \\ \widehat{\boldsymbol{x}}_2 \\ \vdots \\ \widehat{\boldsymbol{x}}_p \\ \hline x_0 \\ x_{s1} \\ \vdots \\ x_{sp} \end{array}\right). \tag{18}$$

As consequence, the solution is obtained by first solving the *reduced system*

$$\begin{pmatrix} I_m & & & \\ w_{s1} & I_m & & \\ & \ddots & \ddots & \\ & & w_{sp} & I_m \end{pmatrix} \begin{pmatrix} y_0 \\ y_{s1} \\ \vdots \\ y_{sp} \end{pmatrix} = \begin{pmatrix} x_0 \\ x_{s1} \\ \vdots \\ x_{sp} \end{pmatrix}, \tag{19}$$

and then updating in parallel the right hand side in (18). Only the reduced system (19) cannot be directly solved in parallel. However, its solution can still be parallelized, by using block cyclic reduction. In this case, the stability of the block cyclic reduction follows from the stability of the used method. In fact, the $m \times m$ block w_{si} is nothing but the discrete map which brings a given initial condition at τ_{i-1} to the subsequent point τ_i in the coarse mesh.

Finally, we observe that the size of the reduced system only depends on the number p of points in the coarse mesh and on the size m of the continuous problem.

It is then independent of both the number k of steps of the main method defining the BVM, and the number s of internal steps used in its block version.

5.1. The case of two-point BVPs

Symmetric schemes are also conveniently used for approximating continuous BVPs. In this section we describe the modifications to the previous approach, in order to derive a stable parallel solver for approximating two-point BVPs. For simplicity, we shall consider the following linear problem,

$$y(t)' = L(t)y(t) + g(t), \qquad B_0 y(t_0) + B_1 y(T) = \eta, \tag{20}$$

where B_0 and B_1 are $m \times m$ matrices and $\eta \in \mathbb{R}^m$. The discrete problem is still given by equation (12), which now assumes the following form,

$$\begin{pmatrix} B_0 & & & & (\widehat{\mathbf{0}}_{s-1}^T & B_1) \\ \boldsymbol{v}_1 & M_1 & & & & \\ & V_2 & M_2 & & & \\ & & \ddots & \ddots & & \\ & & & V_p & M_p & \end{pmatrix} \begin{pmatrix} y_0 \\ \boldsymbol{y}_1 \\ \boldsymbol{y}_2 \\ \vdots \\ \boldsymbol{y}_p \end{pmatrix} = \begin{pmatrix} \eta \\ h_1\boldsymbol{g}_1 \\ h_2\boldsymbol{g}_2 \\ \vdots \\ h_p\boldsymbol{g}_p \end{pmatrix}. \tag{21}$$

We observe that equations (12) and (21) only differ for the first block row of the corresponding coefficient matrices $M^{(p)}$. Nevertheless, a factorization of the form (13) (with the first block row of the matrix $D_2^{(p)}$ changed accordingly) is now inappropriate for the given problem. In fact, it corresponds to the use of multiple shooting for solving problem (20), and this method may be unstable because now the blocks M_i may be ill conditioned or even singular, as a consequence of *dichotomy* (that is, the presence of both stable and unstable modes) of well conditioned continuous problems [14,38,39].

Consequently, we look for a more appropriate factorization of the matrix $M^{(p)}$. This is done by considering that the $sm \times sm$ blocks M_i can be partitioned as

$$M_i = \left[M_i^{(1)} \mid M_i^{(2)} \right],$$

where $M_i^{(1)}$ has size $sm \times (s-1)m$, and must have full column rank, in order for the discrete problem to be solvable (as we obviously assume). As a consequence, its LU factorization is defined,

$$M_i^{(1)} = P_i L_i \begin{pmatrix} \widehat{U}_i \\ \widehat{\mathbf{0}}_{s-1}^T \end{pmatrix}, \quad i = 1, \ldots, p, \tag{22}$$

where P_i is an $sm \times sm$ permutation matrix, L_i is lower triangular, and $\widehat{U}_i$ is an $(s-1)m \times (s-1)m$ upper triangular matrix. We observe that the p factorizations (22)

are independent of each other, and can be computed in parallel. This is also the case for the solution of the following linear systems, defining the block vectors $\boldsymbol{u}_i$ and $\boldsymbol{z}_i$,

$$P_i L_i \boldsymbol{u}_i = M_i^{(2)}, \quad P_i L_i \boldsymbol{z}_i = \boldsymbol{v}_i, \qquad i = 1, \ldots, p.$$

If we now partition the above block vectors as

$$\boldsymbol{u}_i = \begin{pmatrix} \widehat{\boldsymbol{u}}_i \\ u_{si} \end{pmatrix}, \qquad \boldsymbol{z}_i = \begin{pmatrix} \widehat{\boldsymbol{z}}_i \\ z_{si} \end{pmatrix},$$

where u_{si} and z_{si} are $m \times m$ matrices, then each block M_i can be written as

$$M_i = P_i L_i D_i U_i, \quad D_i = \begin{pmatrix} \widehat{I}_{s-1} & \\ & u_{si} \end{pmatrix}, \; U_i = \begin{pmatrix} \widehat{U}_i & \widehat{\boldsymbol{u}}_i \\ & I_m \end{pmatrix}.$$

As a consequence, the following factorization of the coefficient matrix of equation (21) is obtained,

$$M^{(p)} = L^{(p)} D^{(p)} U^{(p)},$$

where

$$L^{(p)} = \begin{pmatrix} I_m & & & \\ & P_1 L_1 & & \\ & & \ddots & \\ & & & P_p L_p \end{pmatrix}, \qquad U^{(p)} = \begin{pmatrix} I_m & & & \\ & U_1 & & \\ & & \ddots & \\ & & & U_p \end{pmatrix},$$

$$D^{(p)} = \begin{pmatrix} B_0 & & & & & & & & B_1 \\ \widehat{\boldsymbol{z}}_1 & \widehat{I}_{s-1} & & & & & & & \\ z_{s1} & & u_{s1} & & & & & & \\ & & \widehat{\boldsymbol{z}}_2 & \widehat{I}_{s-1} & & & & & \\ & & z_{s2} & & \ddots & & & & \\ & & & & & u_{s(p-1)} & & & \\ & & & & & \widehat{\boldsymbol{z}}_p & \widehat{I}_{s-1} & & \\ & & & & & z_{sp} & & & u_{sp} \end{pmatrix}.$$

The discrete solution is then obtained by solving the following equations,

$$L^{(p)} \boldsymbol{c}^{(p)} = \boldsymbol{g}^{(p)}, \tag{23}$$

$$D^{(p)} \boldsymbol{x}^{(p)} = \boldsymbol{c}^{(p)}, \tag{24}$$

$$U^{(p)} \boldsymbol{y}^{(p)} = \boldsymbol{x}^{(p)}. \tag{25}$$

Because of the block diagonal structure, the solution of equations (23) and (25) can be obtained in parallel on p processors. Concerning equation (24), we consider the following partition of the two auxiliary vectors $\boldsymbol{c}^{(p)}$ and $\boldsymbol{x}^{(p)}$,

$$\boldsymbol{c}^{(p)} = \begin{pmatrix} c_0 \\ \boldsymbol{c}_1 \\ \vdots \\ \boldsymbol{c}_p \end{pmatrix}, \quad \boldsymbol{x}^{(p)} = \begin{pmatrix} x_0 \\ \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_p \end{pmatrix}, \qquad c_0, x_0 \in \mathbb{R}^m, \ \boldsymbol{c}_i, \boldsymbol{x}_i \in \mathbb{R}^{sm},$$

and the further partitioning of the vectors $\boldsymbol{x}_i$ and $\boldsymbol{c}_i$ as

$$\boldsymbol{x}_i = \begin{pmatrix} \widehat{\boldsymbol{x}}_i \\ x_{si} \end{pmatrix}, \quad \boldsymbol{c}_i = \begin{pmatrix} \widehat{\boldsymbol{c}}_i \\ c_{si} \end{pmatrix}, \qquad i = 1, \ldots, p,$$

where $x_{si}, c_{si} \in \mathbb{R}^m$. Then, the use of the permutation matrix (17) allows the transformation of (24) into the following equation,

$$QD^{(p)}Q^TQ\boldsymbol{x}^{(p)} = Q\boldsymbol{c}^{(p)},$$

which is a 2×2 upper block triangular system,

$$\left(\begin{array}{cccc|cccc} \widehat{I}_{s-1} & & & & \widehat{\boldsymbol{z}}_1 & O_m & & \\ & \widehat{I}_{s-1} & & & & \widehat{\boldsymbol{z}}_2 & O_m & \\ & & \ddots & & & & \ddots & \ddots \\ & & & \widehat{I}_{s-1} & & & & \widehat{\boldsymbol{z}}_p \ \ O_m \\ \hline & & & & B_0 & & & B_1 \\ & & & & z_{s1} & u_{s1} & & \\ & & & & & z_{s2} & u_{s2} & \\ & & O & & & & \ddots & \ddots \\ & & & & & & & z_{sp} \ \ u_{sp} \end{array}\right) \left(\begin{array}{c} \widehat{\boldsymbol{x}}_1 \\ \widehat{\boldsymbol{x}}_2 \\ \vdots \\ \widehat{\boldsymbol{x}}_p \\ \hline x_0 \\ x_{s1} \\ \vdots \\ x_{sp} \end{array}\right) = \left(\begin{array}{c} \widehat{\boldsymbol{c}}_1 \\ \widehat{\boldsymbol{c}}_2 \\ \vdots \\ \widehat{\boldsymbol{c}}_p \\ \hline c_0 \\ c_{s1} \\ \vdots \\ c_{sp} \end{array}\right). \quad (26)$$

Its solution is then obtained by first solving the reduced system

$$\begin{pmatrix} B_0 & & & B_1 \\ z_{s1} & u_{s1} & & \\ & \ddots & \ddots & \\ & & z_{sp} & u_{sp} \end{pmatrix} \begin{pmatrix} x_0 \\ x_{s1} \\ \vdots \\ x_{sp} \end{pmatrix} = \begin{pmatrix} c_0 \\ c_{s1} \\ \vdots \\ c_{sp} \end{pmatrix}. \quad (27)$$

After that, the right hand side in (26) is updated in parallel, thus giving the remaining entries of the solution vector. Consequently, as in the case of initial value problems, the solution of the reduced system (27) represents the only sequential section of the solver. Furthermore, we observe that the matrix R_p has a block structure similar to that of the original matrix $M^{(p)}$, so that the above steps can be cyclically repeated, thus giving a cyclic reduction-like approach [12,45].

5.2. Expected speed-up

We now briefly consider the expected speed-up for the parallel algorithms described in the previous sections. The estimate is obtained when linear systems of ODEs are solved, so that only one linear system of equations in the form (12) has to be solved. However, when nonlinear ODE problems are to be approximated, an iterative procedure for solving the obtained discrete problem is then considered (e.g., Newton's method). Such a procedure usually requires the solution of a linear system in the form (12) at each iterate. Disregarding the question of convergence (which will be addressed elsewhere), it is then clear that the following arguments can also be extended to discuss the speed-up in the general case.

Moreover, assuming that a distributed memory parallel computer is used, the following items must be taken into account:

1. The rightmost lower index in the used notation denotes the processor where the given quantity has to be stored (consequently, no data redundancy is needed).
2. Data communication is needed only for constructing and solving the reduced system (19), in the case of IVPs, or (27), in the case of BVPs. As a consequence, the complexity of communications is definitely smaller than arithmetic complexity. It will then be neglected in deriving the simplified model for the expected speed-up.

Concerning the arithmetic complexity, its expression has the same leading term, either when approximating continuous IVPs or continuous BVPs. In particular, the sequential arithmetic complexity for solving (12) can be proved to be approximately given by

$$psk^2m^3 \quad \text{flops},$$

where we count as 1 ***flop*** one of the four basic operations, and we recall that

- p is the number of parallel processors,
- s is the number of internal steps in the block implementation of BVMs,
- k is the number of steps of the main method, and
- m is the size of the continuous problem.

Concerning the parallel complexity, we have a section of approximately

$$psk^2m^3 \quad \text{flops}$$

having a perfect degree of parallelism p. The only section which does not have a perfect degree of parallelism is that devoted to the solution of the reduced system. Assuming that block cyclic reduction is used for its solution (or a block cyclic reduction-like approach, in the case of two-point BVPs), it is possible to derive a parallel complexity of approximately

$$\gamma m^3 \log_2 p \quad \text{flops},$$

where

$$\gamma = \begin{cases} 2, & \text{when solving IVPs,} \\ 20/3, & \text{when solving two-point BVPs.} \end{cases}$$

In this case, it is assumed that $p = 2^r$, for some integer r, as in the case of hypercubes. Consequently, the expected speed-up on p processors turns out to be given by

$$S_p \approx \frac{p}{1 + \gamma \log_2 p/(sk^2)}. \tag{28}$$

It is worth noting that:

1. The speed-up grows with the number of steps (k) of the main formula defining the BVM, and the number of steps (s) in its block implementation. This is due to the fact that such quantities do not affect the size of the reduced system.
2. The quantity at the denominator in the expression of S_p grows as $\log_2 p$. This, in turn, allows the use of many parallel processors, if available.

From the above results, we expect an almost perfect speed-up, provided that the number of internal steps for the block implementation is chosen appropriately. This will be confirmed by the numerical tests reported in the following section.

5.3. Numerical tests

In this section we report a few numerical tests carried out on a distributed memory parallel computer. The used platform is a transputer based machine, with 20 nodes T805 connected through a pipeline. However, the most appropriate architecture would be a hypercube, because in such case communications would be done only between physically contiguous processors. The programming language used is Fortran, with the Express communication library [34].

We shall consider two linear test problems:

1) an initial value problem,

2) a boundary value problem.

Our aim is to show that, at least on linear problems and by using a fixed mesh, a BVM can reach an almost perfect speed-up, with respect to its sequential implementation. A compararison with other solvers is here avoided, since we have not yet an efficient parallel mesh selection strategy, even though in the sequential case results are already avaliable [5,17,26]. The considered methods are ETRs (see section 3.2), used with constant stepsize.

The first problem is Hamiltonian,

$$y' = \begin{pmatrix} O_8 & -I_8 \\ I_8 & O_8 \end{pmatrix} Ly, \qquad y(0) = \eta, \tag{29}$$

where L is a suitable spd matrix of size 16 and η is a given vector. We solve the problem, covering the same integration interval, on $p = 1, 4, 8, 16$ processors, and by

Table 1
Problem (29): measured speed-ups.

	s			
k	10	20	40	p
3	3.65	3.81	3.91	
5	3.76	3.88	3.95	4
7	3.86	3.92	3.96	
3	6.91	7.41	7.72	
5	7.28	7.63	7.83	8
7	7.56	7.76	7.88	
3	13.15	14.46	15.24	
5	14.15	15.07	15.54	16
7	14.85	15.41	15.69	

Table 2
Problem (29): efficiency of the parallel algorithm.

	s			
k	10	20	40	p
3	0.912	0.952	0.978	
5	0.940	0.971	0.987	4
7	0.964	0.981	0.991	
3	0.863	0.926	0.965	
5	0.910	0.953	0.978	8
7	0.945	0.969	0.986	
3	0.822	0.904	0.953	
5	0.885	0.942	0.971	16
7	0.928	0.963	0.980	

using different numbers of internal steps, $s = 10, 20, 40$. Moreover, we consider ETRs with $k = 3, 5, 7$ steps (i.e., having order 4, 6 and 8, respectively). Table 1 contains the measured speed-ups, S_p, while table 2 contains the corresponding values of the efficiency, S_p/p. As predicted by (28), the parallel performances of the algorithm improve as k and s are increased.

Finally, let us consider the following second order, singular perturbation boundary value problem, where $\varepsilon = 10^{-3}$,

$$\begin{aligned} &\varepsilon u'' = u, \\ &\varepsilon y'' = t + \frac{t}{2}u' - \frac{t}{2}y' - \varepsilon\pi^2\cos(\pi t) - \frac{t}{2}\pi\sin(\pi t), \\ &u(-1) = -y(-1) = 1, \qquad u(1) = y(1) = \mathrm{e}^{-2/\sqrt{\varepsilon}}. \end{aligned} \tag{30}$$

The problem is solved after recasting as a first order system. In this case, we fix the number $s = 40$ of internal steps for the considered BVMs. Then, we solve the problem on p processors, by using a uniform mesh with stepsize $h = (ps)^{-1}$, $p =$

Table 3
Problem (30): execution times.

$p \setminus k$	3	5	7	9
1	4,217	6,137	8,482	10,899
2	3,955	5,822	8,571	10,993
4	4,110	5,987	8,007	10,259
8	4,176	6,043	8,116	10,404
16	4,265	6,030	8,151	10,551

Table 4
Problem (30): maximum errors.

$p \setminus k$	3	5	7	9
1	3.6e−2	1.2e−2	7.5e−03	5.5e−03
2	3.8e−3	9.3e−4	1.2e−04	5.0e−04
4	2.4e−4	1.4e−5	3.5e−06	1.6e−06
8	1.1e−5	1.9e−7	3.9e−09	4.5e−10
16	8.3e−7	3.1e−9	2.1e−11	5.6e−13

$1, 2, 4, 8, 16$. This means that when the stepsize is halved, and consequently the size of the discrete problem is doubled, the number of used parallel processors is also doubled. As a consequence, we expect the execution time to remain approximately constant for increasing number of processors, even if the accuracy of the solution improves, due to the use of a smaller stepsize.

In table 3 we report the measured execution times, expressed in units of time (*ticks*), each corresponding to 64 μsec, for ETRs with $k = 3, 5, 7, 9$ steps (i.e., having order 4, 6, 8 and 10, respectively). Finally, in table 4 the maximum absolute errors are reported. As predicted, as the number p of processors increases, the error decreases.

Concerning table 3, it is worth mentioning that the execution times on multiple processors are often smaller than the execution times on 1 processor. This is due to the fact that, as the stepsize is decreased (i.e., p increases), the discrete problem changes, and the LU factorization algorithm requires fewer permutations for pivoting.

6. Conclusions

In this paper we gave a brief survey about a new class of numerical methods for ODEs, called Boundary Value Methods (BVMs). Such methods have very good stability properties, and are conveniently used for approximating several kinds of differential problems.

The principal aim of our discussion has been that of showing that such methods do allow an efficient parallel implementation, leading to a parallel ODE solver with *parallelism across the steps*.

Numerical tests confirm that an almost perfect speed-up is to be expected on linear problems, even when many parallel processors are used.

Acknowledgement

The authors are very indebted to professor Donato Trigiante for the helpful discussions.

References

[1] P. Amodio, A-stable k-step linear multistep formulae of order $2k$ for the solution of stiff ODEs, submitted.
[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, Parallel implementation of block boundary value methods for ODEs, J. Comput. Math. Appl., to appear.
[4] P. Amodio, L. Brugnano and T. Politi, Parallel factorizations for tridiagonal matrices, SIAM J. Numer. Anal. 30 (1993) 813–823.
[5] P. Amodio, W. L. Golik and F. Mazzia, Variable step boundary value methods based on reverse Adams schemes and their grid redistribution, Appl. Numer. Math. 18 (1995) 5–22.
[6] P. Amodio and F. Mazzia, A boundary value approach to the numerical solution of ODEs by multistep methods, J. Differential Equations Appl. 1 (1995) 353–367.
[7] P. Amodio and F. Mazzia, Boundary value methods for the solution of differential-algebraic equations, Numer. Math. 66 (1994) 411–421.
[8] P. Amodio and F. Mazzia, A parallel Gauss–Seidel method for block tridiagonal linear systems, SIAM J. Sci. Comput. 16 (1995) 1451–1461.
[9] P. Amodio and F. Mazzia, Parallel block preconditioning for the solution of boundary value methods, J. Comput. Appl. Math. 69 (1996) 191–206.
[10] P. Amodio and F. Mazzia, Parallel iterative solvers for boundary value methods, Math. Comput. Modelling 23(7) (1996) 29–43.
[11] P. Amodio, F. Mazzia and D. Trigiante, Stability of some boundary value methods for the solution of initial value problems, BIT 33 (1993) 434–451.
[12] P. Amodio and M. Paprzycki, A cyclic reduction approach to the numerical solution of boundary value ODEs, SIAM J. Sci. Comput. 18 (1997), to appear.
[13] P. Amodio and D. Trigiante, A parallel direct method for solving initial value problems, Appl. Numer. Math. 11 (1993) 85–93.
[14] U. M. Ascher, R. M. M. Mattheij and R. D. Russel, *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations* (Prentice-Hall, Englewood Cliffs, NJ, 1988).
[15] A. O. H. Axelsson and J. G. Verwer, Boundary value techniques for initial value problems in ordinary differential equations, Math. Comp. 45 (1985) 153–171.
[16] L. Brugnano, Essentially symplectic boundary value methods for linear Hamiltonian systems, J. Comput. Math., to appear.
[17] L. Brugnano, Boundary value methods for the numerical approximation of ordinary differential equations, in: *Proc. 1st Workshop on Numerical Analysis and Applications*, Rousse, 24–27 June 1996, Lecture Notes in Computer Science, to appear.
[18] L. Brugnano, F. Mazzia and D. Trigiante, Parallel implementation of BVM methods, Appl. Numer. Math. 11 (1993) 115–124.
[19] L. Brugnano and D. Trigiante, Tridiagonal matrices and numerical solution of ordinary differential equations, WSSIA 2 (1993) 71–83.
[20] L. Brugnano and D. Trigiante, Stability properties of some BVM methods, Appl. Numer. Math. 13 (1993) 291–304.

[21] L. Brugnano and D. Trigiante, A parallel preconditioning technique for BVM methods, Appl. Numer. Math. 13 (1993) 277–290.
[22] L. Brugnano and D. Trigiante, Convergence and stability of boundary value methods for ordinary differential equations, J. Comput. Appl. Math. 66 (1996) 97–109.
[23] L. Brugnano and D. Trigiante, High order multistep methods for boundary value problems, Appl. Numer. Math. 18 (1995) 79–94.
[24] L. Brugnano and D. Trigiante, Block boundary value methods for linear Hamiltonian systems, Appl. Math. Comput. 81 (1997) 49–68.
[25] L. Brugnano and D. Trigiante, On the characterization of stiffness for ODEs, Dynamics of Continuous, Discrete and Impulsive Systems 2 (1996) 317–335.
[26] L. Brugnano and D. Trigiante, A new mesh selection strategy for ODEs, Appl. Numer. Math., to appear.
[27] L. Brugnano and D. Trigiante, Boundary value methods: the third way between linear multistep and Runge–Kutta methods, Comput. Math. Appl., to appear.
[28] L. Brugnano and D. Trigiante, *Solving ODEs by Linear Multistep Initial and Boundary Value Methods*, to appear.
[29] K. Burrage, *Parallel and Sequential Methods for Ordinary Differential Equations* (Clarendon Press, Oxford, 1995).
[30] A. Carasso and S. V. Parter, An analysis of boundary value techniques for parabolic equations, Math. Comp. 24 (1970) 315–340.
[31] J. R. Cash, *Stable Recursions* (Academic Press, London, 1979).
[32] G. Dahlquist, Convergence and stability in the numerical integration of ordinary differential equations, Math. Scand. 4 (1956) 33–53.
[33] G. Di Lena, F. Iavernaro and F. Mazzia, On the unique solvability of the systems arising from boundary value methods, Report 9/95, Dipartimento di Matematica dell'Università degli Studi di Bari (1995).
[34] *Express Fortran User's Guide* (Parasoft Corp., Pasadena, 1990).
[35] F. Iavernaro and F. Mazzia, Convergence and stability of multistep methods solving nonlinear initial value problems, SIAM J. Sci. Comput. 18 (1997), to appear.
[36] L. Lopez and D. Trigiante, Boundary value methods and BV-stability in the solution of initial value problems, Appl. Numer. Math. 11 (1993) 225–239.
[37] P. Marzulli and D. Trigiante, Stability and convergence of boundary value methods for solving ODE, J. Differential Equations Appl. 1 (1995) 45–55.
[38] R. M. M. Mattheij, Estimates for the errors in the solutions of linear boundary value problems, due to perturbations, Computing 27 (1981) 299–318.
[39] R. M. M. Mattheij, Decoupling and stability of algorithms for boundary value problems, SIAM Rev. 27 (1985) 1–44.
[40] F. Mazzia, Boundary value methods for initial value problems: parallel implementation, Ann. Numer. Math. 1 (1994) 439–450.
[41] F. Mazzia, Boundary value methods for the numerical solution of boundary value problems in differential-algebraic equations, submitted.
[42] J. C. P. Miller, *Bessel Functions, Part II*, Math. Tables, Vol. X, British Association for the Advancement of Sciences (Cambridge University Press, 1952).
[43] F. W. J. Olver, Numerical solutions of second order linear difference equations, J. Res. Nat. Bur. Standards Math. and Math. Phys. 71B (2 and 3) (1967) 111–129.
[44] D. Trigiante, Multipoint methods for linear Hamiltonian systems, in: *Advances in Nonlinear Dynamics*, Ser. Stability and Control: Theory, Methods and Applications (Gordon & Breach, Reading, UK, 1996).
[45] S. J. Wright, Stable parallel elimination for boundary value ODEs, Numer. Math. 67 (1994) 521–535.