ELSEVIER

Journal of Computational and Applied Mathematics 66 (1996) 53–71

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

Parallel predictor–corrector methods[1]

P.J. van der Houwen*, B.P. Sommeijer, J.J.B. de Swart

CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

Received 4 May 1994; revised 17 March 1995

Abstract

In this paper we construct predictor–corrector methods using block Runge–Kutta methods as correctors. Like conventional Runge–Kutta methods, these correctors compute stage values at non-uniformly distributed, intermediate points. The predictor–corrector nature of the methods make them suitable for implementation on parallel computers. Comparisons of an 8th-order, 5-processor predictor–corrector method using Radau II points with the celebrated 8(7) Runge–Kutta method of Prince and Dormand show speed-up factors from 1.9 until 2.9.

Keywords: Numerical analysis; Predictor–corrector iteration; Runge–Kutta methods; Parallelism

1. Introduction

We shall consider predictor–corrector methods (PC methods) for solving the (non-stiff) initial value problem (IVP)

$$y'(t) = f(y(t)), \quad y(t_0) = y_0, \; y, f \in \mathbb{R}^d \tag{1.1}$$

on parallel computers. On one-processor computers, PC methods based on linear multistep (LM) methods of Adams-type are most widely used. However, the use of multi-processor computers enables us to apply PC methods with a much more powerful corrector than in the conventional one-processor PC methods. The general characteristic of these correctors is that they relate whole *blocks* of solution values with each other, rather than single solution values (as in classical LM methods). This has already been observed and tried out in a number of papers. For example, in [1,2], correctors have been constructed where the solution values in each block are equidistant (like LM methods), and in [5–9], blocks with non-equidistant solution values have been considered

* Corresponding author. E-mail: senna@cwi.nl.

[1] The research reported in this paper was partly supported by STW (Netherlands Foundation for the Technical Sciences).

SSDI 0377-0427(95)00158-1

(like Runge–Kutta methods). The block structure of both families of correctors makes it possible to implement the PC method efficiently on a parallel computer system. Moreover, parallel computers also allow us to use local Richardson extrapolation for automatic stepsize control without additional *sequential* costs, because the "reference" solution used in the error estimate can be computed in parallel.

In applying PC methods, we may fix the number of iterations in advance ($PE(CE)^m$ mode with m usually 1 or 2), or we may iterate until the iterated values satisfy accuracy requirements. The first strategy was followed by Birta and Abou-Rabia [1] and by Chu and Hamilton [2]. A disadvantage of this approach is that the stability regions usually are extremely small, unless the corrector is tuned to the particular $PE(CE)^m$ mode employed. For example, the real stability boundaries of the "best" PECE methods constructed by Birta and Abou-Rabia (methods using the "null-weight predictor") range from $\beta_{\text{real}} = 0.576$ for blocksize 2 until $\beta_{\text{real}} = 0.078$ for blocksize 10. By using a number of free parameters in the corrector for improving stability, Chu and Hamilton [2] succeeded in increasing the real stability boundary substantially. However, for a given blocksize, the order is of course reduced. Both [1,2] restrict the stability considerations to the real axis.

In the second strategy, where the corrector is iterated until the iterated values satisfy our accuracy requirements, we are not only faced with the stability region of the corrector, but also with its *convergence* region. The cross-section of these regions may be interpreted as the stability region of the PC method. So far, the investigations have mainly been concerned with Runge–Kutta (RK) correctors. As it happens, the classical RK correctors of Gauss–Legendre or Radau type, have a high order of accuracy (superconvergence), they are unconditionally stable, and they possess very large convergence boundaries. Hence, RK-based PC methods are both highly accurate and highly stable. Moreover, by their one-step nature, RK correctors allow an easy and highly efficient stepsize strategy (provided that the predictor formula is also of one-step type). In [5] experiments are reported showing that the sequential costs of one-step PC methods based on the Gauss–Legendre corrector of order 10 are about half the sequential costs of the DOPRI8 code. The DOPRI8 code of Hairer et al. [3] is based on the 13-stage, 8th-order embedded RK method of Prince and Dormand [10], and is generally considered as one of the most efficient sequential codes (cf. [3, p. 378]). We remark that by sacrificing the one-step nature of the predictor–corrector pair, the efficiency of parallel RK-based PC methods can be improved drastically, of course at the cost of a less easy implementation and stepsize strategy. A few first experiments were reported in [8].

The LM-based and RK-based PC methods discussed above are very special examples of methods that fit into the large class of *general linear methods* introduced by Butcher in 1966. In this paper, we shall try to find more efficient predictor–corrector pairs than constructed so far by looking in this class of general linear methods. In particular, we shall combine the multistep nature of the Birta and Abou-Rabia and Chu and Hamilton methods with the non-equidistant-solution-values property of RK methods. In fact, the methods of this paper belong to the family of block Runge–Kutta (BRK) correctors studied in [6], where a first few results for BRK-based PC methods can be found. Here, we shall pursue these investigations. In particular, we pay attention to the stability of the PC method, because the weak point of most block methods is their small stability region.

In Section 2, we specify a family of two-stage BRK correctors and we discuss the order of accuracy and their stability. Section 3 analyses PC iteration of these BRK correctors and defines the convergence factors associated with the iteration process. The main results of this paper can be

found in Section 4 where a number of BRK correctors that combine high order of accuracy, fast convergence and sufficiently large stability boundaries are constructed. Finally, in Section 5, PC methods based on BRK pairs are compared with DOPRI8, showing that speed-up factors derived from sequential function calls range from 1.9 until 2.9. When implemented on a parallel computer system, these speed-up factors will decrease. However, in earlier experiments with iterated RK methods on the Alliant FX/4, we found that in this type of methods, the loss due to synchronization of the processors is about 10% (this is confirmed by experiments of Lie [8]). Since the new methods are of the same type as iterated RK methods, we may expect that the speed-up factors based on sequential function calls will be reduced by only 10% when run on an Alliant.

2. Block Runge–Kutta methods

For the definition and analysis of the block Runge–Kutta (BRK) methods, it is convenient to introduce some notations. Firstly, we shall frequently use the componentwise notation for functions of vectors. For example, v^2 is understood to be the vector whose entries are the squares of the entries of v. Furthermore, e denotes the vector with unit entries, e_i the ith unit vector whose entries vanish except for the ith entry which equals 1, I_{dd} is the d-by-d identity matrix, O_{mn} is the m-by-n zero matrix, and E_{mn} is the m-by-n matrix whose entries are zero except for its nth column which equals e. The dimension of e and e_i may change, but will always be clear from the context.

Our starting point is a method of the form

$$Y = (A \otimes I_{dd})Y_{n-1} + h(B \otimes I_{dd})F(Y_{n-1}) + h(C \otimes I_{dd})F(Y), \tag{2.1a}$$

$$Y_n = (A^* \otimes I_{dd})Y_{n-1} + h(B^* \otimes I_{dd})F(Y_{n-1}) + h(C^* \otimes I_{dd})F(Y), \tag{2.1b}$$

$$y_n = y_{n-1} + h(b_s^{\mathrm{T}} \otimes I_{dd})F(Y_{n-1}) + h(c_s^{\mathrm{T}} \otimes I_{dd})F(Y_n), \quad n = 1, \ldots, N. \tag{2.1c}$$

Here, the s-by-s matrices A, B, C, A^*, B^*, C^* and the s-dimensional vectors b_s and c_s contain the method parameters, h denotes the stepsize $t_n - t_{n-1}$, and $\otimes$ denotes the Kronecker product. Furthermore, Y and Y_n represent numerical approximations to the exact solution vectors $y(et_{n-1} + ah)$, where a denotes the abscissa vector, and where for any vector $V = (V_i)$, $F(V)$ contains the derivative values $(f(V_i))$. It is assumed that the components of a are distinct.

The formulas (2.1a), (2.1b) and (2.1c) are, respectively, called the stage vector equation with s *internal* stages, the output formula with s *external* stages, and the step point formula. The *external* stages are all explicit, while the internal stages can be implicit or explicit. For example, if C has q zero rows and $r := s - q$ rows with non-zero entries, then there are q explicit stages and r fully implicit stages. The quantities Y, Y_n and y_n are, respectively, called the stage vector, the output vector and the step point value.

With respect to parallel implementation, methods of this type have been studied in [6], and were called block Runge–Kutta (BRK) methods, because they can be obtained from conventional RK methods by replacing the scalar RK parameters by matrices and the stage values by blocks of stage values. Like RK methods, the stage values correspond to non-uniformly spaced points at the t-axis. In terms of the array notation used in [6], the method (2.1) can be represented as a two-stage BRK

method with one explicit and one implicit (block) stage:

$$\begin{array}{c|cc} I & O & O \\ A & B & C \\ \hline A^* & B^* & C^* \end{array}. \tag{2.1'}$$

In the determination of the parameter matrices in the stage vector equation (2.1a), the order conditions (see Section 2.1) will play an important role, together with the requirement that the iteration method used for solving the stage vector equation is rapidly converging. In this paper, we will solve the stage vector equation by predictor–corrector (PC) type iteration. The convergence of PC iteration is largely controlled by the "magnitude" of the matrix C, that is, convergence is better as C is smaller in some sense. For example, its spectral radius is often a first indicator of the potential convergence speed (see Section 3). In this connection, we should remark that strictly triangular matrices C lead to a zero spectral radius (in fact, the method is an *explicit* method, so that no iteration process is needed). However, such explicit methods approximate the components of $\boldsymbol{Y}$ by *extrapolation* formulas which are considerably less accurate than the *interpolation* formulas associated with *implicit* methods. Fortunately, it turns out that high accuracy and fast convergence often go together. Hence, if the stage vector $\boldsymbol{Y}$ has sufficient accuracy and stability, then the output formula (2.1b) can be dropped (i.e. $A = A^*$, $B = B^*$, $C = C^*$, so that $\boldsymbol{Y}_n = \boldsymbol{Y}$).

In most of the BRK correctors constructed in this paper, we do not use an output formula. However, we shall show in Section 4.3 that output formulas can be used for stabilizing the corrector.

The step point formula can be used to increase the order at the step points (superconvergence). This can be achieved by setting $\boldsymbol{b}_s = \boldsymbol{0}$ and by identifying the components of $\boldsymbol{c}_s$ and the abscissa vector $\boldsymbol{a}$ with the quadrature weights and quadrature points of Gaussian quadrature formulas. The step point order is then one higher than the order of the output vector $\boldsymbol{Y}_n$. Since in the methods considered in this paper, the order of the output vector will be at most $s + 1$ or $s + 2$, there is, as far as order of accuracy is concerned, no need for basing the abscissa vector on quadrature formulas of the highest possible order (Gauss–Legendre formulas). This leads us to use abscissa vectors with $a_1 \neq 0$ and $a_s = 1$ which often simplifies the implementation (e.g. the Radau II points fit into this group).

Finally, we remark that (2.1) reduces to an RK method by setting $A = A^* = E_{ss}$, $B = B^* = O$, $C = C^*$, $\boldsymbol{b}_s = \boldsymbol{0}$ and $\boldsymbol{c}_s^{\mathrm{T}} = \boldsymbol{e}_s^{\mathrm{T}} C$ with C denoting the RK matrix of the collocation method defined by the abscissa vector $\boldsymbol{a}$. We shall call this collocation method the *RK method associated with the abscissa vector* $\boldsymbol{a}$. By identifying the BRK method (2.1) in the first step with such an RK method, we can avoid the problem of computing starting values, because we only need the initial value $\boldsymbol{y}_0$, and not the whole starting vector $\boldsymbol{Y}_0$.

2.1. Accuracy

Given the abscissa vector $\boldsymbol{a}$, the conditions for pth-order consistency of the stage vector equation (2.1a) are given by (see, e.g. [6])

$$\boldsymbol{Ae} = \boldsymbol{e}, \quad A(\boldsymbol{a} - \boldsymbol{e})^j + jB(\boldsymbol{a} - \boldsymbol{e})^{j-1} + jC\boldsymbol{a}^{j-1} = \boldsymbol{a}^j, \; j = 1, \ldots, p.$$

We may write these order conditions in the form

$$Ae = e, \quad AX_{sp} + BW_{sp} + CV_{sp} = U_{sp},$$
$$U_{sp} := \left(\frac{1}{j}a^j\right), \quad V_{sp} := (a^{j-1}), \quad W_{sp} := ((a-e)^{j-1}), \quad X_{sp} := \left(\frac{1}{j}(a-e)^j\right), \quad j = 1,\dots,p, \tag{2.2a}$$

where the lower indices again refer to the number of rows and columns of the matrix. If (2.2a) is satisfied, then p will be called the *internal stage order*. The vector of principal error constants associated with the stage vector equation is given by

$$E_{p+1} := \frac{1}{(p+1)!}\left\{a^{p+1} - A(a-e)^{p+1} - (p+1)(B\ C)\begin{pmatrix}(a-e)^p\\ a^p\end{pmatrix}\right\}. \tag{2.3a}$$

Note that for $p = s$, $A = E_{ss}$ and $B = O_{ss}$, the stage vector equation reduces to the stage vector equation of the RK method with RK matrix $C = U_{ss}V_{ss}^{-1}$.

For the output formula (2.1b) we proceed as follows. Imposing the localizing assumption, that is, assuming that the components $Y_{n-1,i}$ are on the locally exact solution curve through the point (t_{n-1}, y_{n-1}), we may set $Y_{n-1} = y(et_{n-2} + ah)$ and, by virtue of (2.2a), $Y = y(et_{n-1} + ah) + O(h^{p+1})$. Hence,

$$\begin{aligned} Y_n &= (A^* \otimes I_{dd})Y_{n-1} + h(B^* \otimes I_{dd})F(Y_{n-1}) + h(C^* \otimes I_{dd})F(Y) \\ &= (A^* \otimes I_{dd})y(et_{n-2} + ah) + h(B^* \otimes I_{dd})y'(et_{n-2} + ah) \\ &\quad + h(C^* \otimes I_{dd})y'(et_{n-1} + ah) + O(h^{p+2}). \end{aligned}$$

By Taylor expansion it can be shown that

$$Y_n = y(et_{n-1} + ah) + O(h^{p+2}) + O(h^{p^*+1}),$$

provided that

$$A^*e = e, \quad A^*X_{sp^*} + B^*W_{sp^*} + C^*V_{sp^*} = U_{sp^*}, \tag{2.2b}$$

where the matrices X_{sp^*}, W_{sp^*}, V_{sp^*}, and U_{sp^*} are defined as in (2.2a) with p replaced by p^*. Thus, the output vector Y_n has order $\min\{p+1, p^*\}$. This order will be called the *external stage order*, or briefly *the* stage order.

There are two error vectors associated with the output formula, viz.

$$E^*_{1,p+2} := C^*E_{p+1},$$
$$E^*_{2,p^*+1} := \frac{1}{(p^*+1)!}\left\{a^{p^*+1} - A^*(a-e)^{p^*+1} - (p^*+1)(B^*\ C^*)\begin{pmatrix}(a-e)^{p^*}\\ a^{p^*}\end{pmatrix}\right\}, \tag{2.3b}$$

where E_{p+1} is defined by (2.3a).

Finally, we consider the step point formula (2.1c). It is possible to achieve order of consistency $2s$ for this formula, so that the order at the step points becomes $\min\{2s, p+2, p^*+1\}$. However, this may lead to rather large entries in b_s and c_s. Alternatively, we may use a zero b_s vector and identify

c_s^T with the last row vector of the RK matrix associated with a, that is, $c_s^T = e_s^T U_{ss} V_{ss}^{-1}$. If p_{RK} denotes the order of the RK method, then we have order of accuracy $\min\{p_{RK}, p+2, p^*+1\}$ at the step points. This will be called the *step point order*. We summarize the preceding discussion in the following theorem:

Theorem 2.1. *If* (2.2a) *and* (2.2b) *are satisfied, then the BRK method* (2.1) *has stage order* $\min\{p+1, p^*\}$ *and output vector errors given by* (2.3b). *If, in addition,* $b_s = 0$, $c_s^T = e_s^T U_{ss} V_{ss}^{-1}$, *and if the abscissa vector* a *defines an RK method of order* p_{RK}, *then the step point order is given by* $\min\{p_{RK}, p+2, p^*+1\}$.

2.2. Stability

In order to ensure stability for $h = 0$ (zero-stability), we shall require that A^* has $s-1$ eigenvalues inside the unit circle (since (2.2b) prescribes that $A^* e = e$, A^* necessarily has one eigenvalue 1). Such matrices will be referred to as *zero-stable matrices*.

For $h > 0$, stability also depends on the other parameter matrices and on the abscissa vector a. With respect to the scalar test equation $y' = \lambda y$, where λ runs through the spectrum of the Jacobian matrix $\partial f(y_n)/\partial y$, we obtain the recursion

$$Y_n = M(z) Y_{n-1}, \quad M(z) := A^* + zB^* + zC^*(I - zC)^{-1}(A + zB), \quad z := \lambda h. \tag{2.4}$$

Assuming that the stability matrix $M(z)$ has s distinct eigenvalues, we have that (cf. [11])

$$\| Y_n \|_2 \leqslant \| M^n(z) \|_2 \| Y_0 \|_2, \quad \| M^n(z) \|_2 \approx v(z)[\rho(M(z))]^n \quad \text{as } n \to \infty, \tag{2.5}$$

where $v(z)$ is bounded by the condition number of the eigensystem of $M(z)$. This estimate suggests defining the stability region, and the real and imaginary stability intervals according to

$$\begin{aligned} &\mathbb{S} := \{z: \rho(M(z)) < 1\}, \\ &(-\beta_{\text{real}}, 0) := \{z: z \in \mathbb{S}, z < 0\}, \quad (-\beta_{\text{imag}}, \beta_{\text{imag}}) := \{z: z \in \mathbb{S}, \operatorname{Re}(z) = 0, z \neq 0\}, \end{aligned} \tag{2.6}$$

where $\rho(\cdot)$ denotes the spectral radius function. The quantities β_{real} and β_{imag} are respectively called the *real* and the *imaginary stability boundary* of the BRK method. By (2.6) stability conditions of the type $h < \beta/\rho(\partial f/\partial y)$ are implied, so that we should require the method to have sufficiently large stability boundaries, say not less than 1. In addition, we should impose the condition that $v(z)$ is of moderate size, particularly for $z = 0$, because zero-stability implies $\rho(M(0)) = \rho(A^*) = 1$, so that

$$\| Y_n \|_2 \leqslant v(0) \| Y_0 \|_2 \quad \text{as } n \to \infty. \tag{2.5'}$$

If A^* is singular, then estimating an upper bound for $v(0)$ by means of the condition number of A^* is not possible. For example, this happens in the important case where $A^* = E_{ss}$ (in [6] BRK methods of this form were called BRK methods of *Adams type*). However, for such methods, $M^n(0) = [A^*]^n = E_{ss}$, hence $\| M^n(0) \|_2 = \| E_{ss} \|_2 = \sqrt{s}$, so that for $z = 0$, Adams-type BRK methods satisfy (2.5) with $v(0) = \sqrt{s}$.

3. The iteration scheme

We approximate the solution Y of (2.1a) by successive iterates $Y^{(j)}$ satisfying the PC scheme (or fixed-point iteration scheme)

$$Y^{(j)} = (A \otimes I_{dd})Y_{n-1} + h(B \otimes I_{dd})F(Y_{n-1}) + h(C \otimes I_{dd})F(Y^{(j-1)}), \quad j = 1, \ldots, m; \; n \geqslant 1. \tag{3.1}$$

Evidently, if the iterates $Y^{(j)}$ satisfying (3.1) converge to a fixed vector V as $j \to \infty$, then $V = Y$. In actual computation, the number of iterations m is dynamically determined by requiring that the corrector equation is solved within a given tolerance (cf. Section 5). This iteration scheme has a high degree of parallelism, because the *sequential* costs of each iteration on s processors are independent of the number of implicit stages r.

For the predictor formula providing $Y^{(0)}$, we may take the explicit BRK method

$$Y^{(0)} = (A_0 \otimes I_{dd})Y_{n-1} + h(B_0 \otimes I_{dd})F(Y_{n-1}). \tag{3.2}$$

One option defines A_0 and B_0 according to

$$(A_0 \quad B_0) = (U_{s,2s-1} \quad e)\begin{pmatrix} X_{s,2s-1} & e \\ W_{s,2s-1} & 0 \end{pmatrix}^{-1}, \tag{3.3a}$$

where $X_{s,2s-1}$, $W_{s,2s-1}$ and $U_{s,2s-1}$ are defined as in (2.2a). It is easily seen that (3.3a) satisfies (2.2a) for $A = A_0$, $B = B_0$, $C = O$ and $p = 2s - 1$, so that (3.3a) generates predictor values of order $2s - 1$ (provided that the stage order of the BRK method (2.1) is at least $2s - 1$). In fact, (3.3a) is a Hermite integration formula generated by the abscissa vector a. In addition to the high orders of Hermite formulas, the error constants $\|E_{p+1}\|_\infty$ as defined in (2.3a) are extremely small. In Table 1, this is illustrated for the Radau II abscissas. However, in spite of their high orders and relatively small error constants, Hermite predictor formulas have the drawback of extremely large coefficients in the parameter matrices A_0 and B_0, especially for larger values of s. This may cause considerable round-off errors unless sufficiently high arithmetic is used (we remark that to some extent, round-off can be suppressed by using shifted iterates $X^{(j)} := Y^{(j)} - e \otimes y_{n-1}$ in an actual implementation (cf. [4, p. 128])).

An alternative to the Hermite predictor formula is offered by the Adams–Bashforth-type formula defined by

$$A_0 = E_{ss}, \quad B_0 = U_{ss}W_{ss}^{-1}, \tag{3.3b}$$

which satisfies (2.2a) for $A = E_{ss}$, $B = B_0$, $C = O$ and $p = s$. Its error constants associated with the Radau II abscissas can be found in Table 1. From these figures it is clear that if arithmetic allows, we should use the Hermite predictors.

Table 1
Error constants $\|E_{p+1}\|_\infty$ associated with Radau II abscissas for Hermite and Adams–Bashforth predictor formulas

Predictor	$s = 2$	$s = 3$	$s = 4$	$s = 5$
Hermite	0.12	0.0087	0.00034	0.0000081
Adams–Bashforth	0.33	0.13	0.041	0.010

The method {(3.1),(3.2)} will be called a PIBRK method (parallel iterated BRK method). The sequential costs of PIBRK methods depend on the structure of the parameter matrices. Therefore, we postpone a discussion of computational costs until the special cases developed in this paper have been specified.

For the convergence analysis of (3.1) we define the iteration error

$$\varepsilon^{(j)} := \boldsymbol{Y}^{(j)} - \boldsymbol{Y}.$$

On substitution in (3.1), we obtain

$$\varepsilon^{(j)} = h(C \otimes I_{dd})[\boldsymbol{F}(\boldsymbol{Y}^{(j-1)}) - \boldsymbol{F}(\boldsymbol{Y})].$$

This relation immediately leads to the estimate

$$\|\varepsilon^{(j)}\| \leqslant hL\|C\|\,\|\varepsilon^{(j-1)}\|,$$

where L denotes a Lipschitz constant on the right-hand side function f. Although this estimate has the advantage of being valid for the general IVP (1.1), it does not provide much information for selecting efficient corrector methods. Therefore, we resort to the familiar approach of approximating the IVP by a linear model. In this way, we obtain detailed information on the iteration process for the class of linear IVPs. Like the linear stability theory, this linear convergence theory turns out to be highly reliable for a large class of *non-linear* problems.

Assuming the right-hand side function f sufficiently smooth, we may write

$$\boldsymbol{F}(\boldsymbol{U} + \delta) - \boldsymbol{F}(\boldsymbol{U}) = J(\boldsymbol{U})\delta + \mathrm{O}(\delta^2),$$

where $J(\boldsymbol{U})$ is an sd-by-sd block-diagonal matrix whose diagonal blocks consist of the Jacobian matrices $\partial \boldsymbol{f}(\boldsymbol{U}_i)/\partial \boldsymbol{y}$, $\boldsymbol{U}_i$ being the components of $\boldsymbol{U}$. On substitution, we straightforwardly derive the error recursion

$$\varepsilon^{(j)} = Z\varepsilon^{(j-1)} + \mathrm{O}(\varepsilon^{(j-1)})^2,$$

where the matrix

$$Z = Z(hJ(\boldsymbol{Y})) := h(C \otimes I)J(\boldsymbol{Y}) \tag{3.4}$$

controls the convergence of the iteration scheme. Assuming that higher-order terms can be neglected, the iteration error of the stage vector satisfies

$$\varepsilon^{(j)} = [Z(hJ(\boldsymbol{Y}))]^j\varepsilon^{(0)} = h^j[(C \otimes I_{dd})J(\boldsymbol{Y})]^j\varepsilon^{(0)}. \tag{3.5}$$

Thus, the iteration matrix C plays a crucial role in the convergence of the PC iteration process.

We shall define the (averaged) *convergence factor* for the scalar test equation $y' = \lambda y$. For this test equation, (3.5) reduces to

$$\varepsilon^{(j)} = z^jC^j\varepsilon^{(0)}, \quad z := \lambda h. \tag{3.6}$$

Hence,

$$\|\varepsilon^{(m)}\|_\infty \leqslant |z|^m\|C^m\|_\infty\|\varepsilon^{(0)}\|_\infty,$$

so that, with respect to the maximum norm, the (averaged) convergence factor over m iterations is given by

$$\alpha(m,z) := |z| \sqrt[m]{\|C^m\|_\infty}. \tag{3.7}$$

The region of convergence in the complex z-plane is given by $\alpha(m,z) < 1$, that is, the open disk

$$\mathbb{C}_m := \{z: |z| < \gamma_m\}, \quad \gamma_m := \frac{1}{\sqrt[m]{\|C^m\|_\infty}}, \tag{3.8}$$

where γ_m may be considered as the *convergence boundary*. From (3.8) we deduce the stepsize condition $h < \gamma_m/\rho(\partial f/\partial y)$. Thus, large convergence boundaries relax the convergence condition and improve convergence at the same time.

In actual computation, one should satisfy both the convergence condition associated with (3.8) and the stability condition associated with (2.6), that is, the spectrum of the matrix $h\partial f/\partial y$ should be contained in the intersection of the stability region $\mathbb{S}$ and the convergence region $\mathbb{C}_m$ (here, we assume that the IVP is itself stable, so that the spectrum of $\partial f/\partial y$ is located in the left half-plane). As a consequence, there is no point in trying to construct correctors whose stability region is much larger than their region of convergence. However, it may be feasible to have correctors whose convergence region is much larger than their region of stability, because, as we just saw, large regions of convergence also improve convergence speed. Notice that strictly lower (or upper) triangular matrices C have zero convergence factors for $m \geqslant s + 2$. However, as already remarked, then the generating BRK corrector (2.1) is explicit and therefore has reduced accuracy.

4. Construction of BRK correctors

In all BRK correctors considered in this paper, the abscissa vector $\boldsymbol{a}$ is identified with the Radau II points. We do not claim that these points are most optimal, but it is likely that results based on Radau II points are indicative for other sets of abscissas.

In the construction of BRK correctors, we have to take into account: (i) the consistency conditions (2.2), (ii) the zero-stability and condition of the matrix A^*, (iii) the stability region, and (iv) the rate of convergence. These aspects will be characterized by the step point order, by the condition number $\kappa_\infty(A^*)$ of A^* (provided A^* is non-singular), the stability boundaries defined by (2.6), the condition number $\kappa_\infty(C)$ and the convergence boundaries defined in (3.8).

4.1. Adams-type correctors without output formulas

We start with the class of methods without output formula, i.e. the output formula coincides with the stage vector equation, so that $A = A^*$, $B = B^*$, and $C = C^*$ (i.e. there are no external stages). Within this class, zero-stability is automatically achieved by choosing the subclass of Adams methods, i.e. $A = E_{ss}$. One option for choosing the remaining matrices B and C is such that a high order of consistency is obtained. In our preliminary analysis of this type of methods we found that in general the convergence factors associated with the matrix C improve as the order of consistency increases. In particular, we observed that the entries in the upper part and in the lower right-hand

corner of C are relatively small. This observation led us to consider BRK correctors of which the matrix C is of the form

$$C = \begin{pmatrix} O & O \\ \underline{C}_1 & \underline{C}_2 \end{pmatrix},$$

where $\underline{C}_1$ and $\underline{C}_2$, respectively, are an r-by-q and an r-by-r matrix with $q + r = s$. Evidently, such correctors have r *implicit* stages and q *explicit* stages. In particular, the matrix $\underline{C}_2$ determines the convergence of the PC iteration process.

We shall define the first q rows of the matrix B completely by consistency conditions. From (2.2a) it follows that the first q stages are consistent of order s if they coincide with the first q rows of the matrix $U_{ss}W_{ss}^{-1}$. In fact, the resulting formulas are *Adams–Bashforth* formulas (cf. the Adams–Bashforth predictor formula (3.3b)). Although these formulas are based on pure *extrapolation*, the extrapolation errors are relatively small, because they correspond to the first (and therefore smaller) components of the abscissa vector $\boldsymbol{a}$.

The class of methods indicated above is defined by

$$A = E_{ss}, \qquad B = (U_{ss} - CV_{ss})W_{ss}^{-1}, \qquad C = \begin{pmatrix} O & O \\ \underline{C}_1 & \underline{C}_2 \end{pmatrix}, \tag{4.1a}$$

$$A^* = A, \qquad B^* = B, \qquad C^* = C, \tag{4.1b}$$

where the r-by-s matrix $\underline{C} := (\underline{C}_1 \ \underline{C}_2)$ is still free. This method is zero-stable (because A is zero-stable). Since (4.1) satisfies (2.2a) for $p = s$ it follows from Theorem 2.1 that the stage order equals s. In the following subsections, a few options for choosing the matrix $\underline{C}$ will be discussed.

4.1.1. Adams–Bashforth–Moulton methods

The most simple option defines the implicit stages by imposing consistency conditions of highest possible order, i.e. $\underline{C}$ is defined by the r-by-s matrix occurring in the lower right-hand corner of the s-by-$2s$ matrix

$$U_{s,2s}\begin{pmatrix} W_{s,2s} \\ V_{s,2s} \end{pmatrix}^{-1}. \tag{4.2}$$

The resulting BRK corrector defines the first q components of $\boldsymbol{Y}_n = \boldsymbol{Y}$ by (explicit) Adams–Bashforth-type formulas (of order s) and the last r components of $\boldsymbol{Y}_n$ by implicit Adams-type formulas of highest possible order of consistency (i.e. order $2s$). In this respect, these implicit formulas resemble the conventional Adams–Moulton formulas. Therefore, we shall refer to these correctors as *Adams–Bashforth–Moulton* correctors (ABM correctors). The special methods arising for $q = 0$ and $q = r$ will be called *Adams–Moulton* (AM) correctors and *Adams–Bashforth* (AB) correctors, respectively.

We recall that the stage order of ABM correctors equals s. However, for AM correctors where no explicit stages occur ($q = 0$), the stage order becomes $2s$. For $q > 0$, Theorem 2.1 shows that the step point order can be raised to $s + 1$ by choosing in the step point formula $\boldsymbol{b}_s = \boldsymbol{0}$ and $\boldsymbol{c}_s^{\mathrm{T}} = \boldsymbol{e}_s^{\mathrm{T}} U_{ss} V_{ss}^{-1}$. However, it turns out that for ABM correctors

$$\boldsymbol{e}_s^{\mathrm{T}} B \approx \boldsymbol{0} \quad \text{and} \quad \boldsymbol{e}_s^{\mathrm{T}} C \approx \boldsymbol{e}_s^{\mathrm{T}} U_{ss} V_{ss}^{-1}. \tag{4.3}$$

Hence, in practical applications, we achieve superconvergence at the step points by defining the step point formula simply by $y_n = (e_s^T \otimes I_{dd})Y_n$.

We computed the convergence and stability characteristics for a large number of ABM correctors. In the case of a zero imaginary stability boundary, we have also computed the value of β^*_{imag} defined by the length of the imaginary interval where the spectral radius of the amplification matrix $M(z)$ is bounded by $1 + \varepsilon$, $\varepsilon > 0$. For sufficiently small values of ε, these values can be used as the "effective" imaginary stability boundary in practical computations. For a given value of r, it turns out that the stability boundaries *decrease* and the convergence boundaries *increase* with q. For each r ($2 \leqslant r \leqslant 5$), Table 2 presents the two cases where the stability boundaries β_{real} and β^*_{imag} (with $\varepsilon = 10^{-3}$) are both sufficiently large (say at least ≈ 1) while the convergence boundaries are maximal. A more extensive list including cases with smaller convergence and stability boundaries can be found in the Appendix to the corresponding CWI-report NM-R9408. Notice that a large condition number for $\underline{C}_2$ implies relatively small convergence boundaries in the first few iterations.

Next, we discuss the sequential costs of the PIBRK method based on ABM correctors. Let us consider the following implementation of the PIBRK method:

$$\begin{aligned}
\underline{Y}^{(0)} &= (\underline{A}_0 \otimes I_{dd})Y_{n-1} + h(\underline{B}_0 \otimes I_{dd})F^*_{n-1},\\
\bar{Y}^{(j)} &= (\bar{A} \otimes I_{dd})Y_{n-1} + h(\bar{B} \otimes I_{dd})F^*_{n-1},\\
\underline{Y}^{(j)} &= (\underline{A} \otimes I_{dd})Y_{n-1} + h(\underline{B} \otimes I_{dd})F^*_{n-1} + h(\underline{C}_1 \otimes I_{dd})F(\bar{Y}^{(j-1)}) + h(\underline{C}_2 \otimes I_{dd})F(\underline{Y}^{(j-1)}),\\
Y_n &= \begin{pmatrix} \bar{Y}^{(m)} \\ \underline{Y}^{(m)} \end{pmatrix}, \qquad y_n = (e_s^T \otimes I_{dd})Y_n, \qquad F^*_n = F(Y^{(m-1)}),
\end{aligned} \tag{4.4}$$

where $j = 1, \ldots, m$. Here, upper and lower bars refer to the first q and last r rows of a matrix, and the underlying matrices A, B and C are defined by (4.1) and (4.2). Notice that the first q components of the iterates $Y^{(j)}$ do not depend on j.

It is easily verified that (4.4) does yield the solution to the corrector method as $m \to \infty$ (provided that it converges). Assuming that $1 \leqslant q \leqslant r$, the sequential costs on r processors are $m + 1$

Table 2
Characteristics for selected ABM correctors

$s = q + r$	Order	β_{real}	β_{imag}	β^*_{imag}	$\kappa_\infty(\underline{C}_2)$	γ_2	γ_3	γ_4	γ_{10}	...	γ_∞
3 = 1 + 2	4	3.33	0	3.81	17	2.48	3.08	3.55	5.16	...	6.17
4 = 2 + 2	5	0.94	0	0.91	15	3.82	4.55	5.20	7.79	...	9.06
4 = 1 + 3	5	2.88	0	2.53	81	1.79	2.39	2.96	5.91	...	8.23
5 = 2 + 3	6	1.26	0	1.78	64	2.48	3.26	3.99	7.47	...	10.35
5 = 1 + 4	6	1.88	0	2.90	383	1.68	2.11	2.59	5.45	...	10.68
6 = 2 + 4	7	1.38	0.99	0.99	239	2.06	2.64	3.26	6.60	...	12.28
5 = 0 + 5	6	2.26	0.01	3.16	13 853	1.33	1.93	2.26	4.50	...	10.80
6 = 1 + 5	7	0.92	1.13	1.13	2700	1.56	2.05	2.47	5.12	...	13.05

right-hand side evaluations, i.e. the evaluation of $F(\bar{Y}^{(m)})$ plus the evaluation of the m right-hand side functions $F(\underline{Y}^{(j-1)})$. The evaluation of $F(\bar{Y}^{(m)})$ can be done in parallel with that of $F(\underline{Y}^{(0)})$, but this would require q additional processors. However, if we apply local Richardson extrapolation for stepsize control and if we use additional processors for computing the "reference" solution, then these processors can also be used for evaluating $F(\bar{Y}^{(m)})$. Since the "reference" solution is computed with a double step, it is likely that there is some idle time, in spite of the fact that larger steps will require more iterations to solve the stage vector equation. Hence, in such a case, the total sequential costs per step are just m right-hand side evaluations.

4.1.2. Adams–Bashforth–Radau methods

A second option identifies the matrix $(\underline{C}_1 \ \underline{C}_2)$ in (4.1a) with the last r rows of the Radau IIA matrix $U_{ss}V_{ss}^{-1}$. Then the matrix $\underline{B}$ vanishes, so that the r implicit stages are determined by Radau formulas. The stage order and the step point order are the same as for ABM correctors, i.e. s and $s+1$, respectively. We shall call this corrector an *Adams–Bashforth–Radau corrector* (ABR corrector) because the first q components of Y_n are defined by Adams–Bashforth formulas and the last r components by Radau IIA formulas. For $r=0$ the corrector reduces to the AB corrector and $r=s$ leads to the Radau IIA correctors. The PIBRK method generated by the ABR correctors can be defined according to (4.4), so that the sequential costs are the same.

The analogue of Table 2 is given in Table 3 where we included the case $q=0$ defining the pure Radau IIA corrector. A comparison of these selected methods with the corresponding ABM correctors reveals that ABR correctors have smaller convergence boundaries (particularly for larger m), but possess considerably larger stability boundaries. One may argue that the stability boundaries of the selected ABM correctors are sufficiently large for integrating non-stiff problems, so that the ABM correctors seem to be the more attractive ones. However, if the stage vector equation is not solved to convergence (for example, if the tolerance parameter in the stopping criterion is not sufficiently small), then we are faced with the fact that the stability region of the

Table 3
Characteristics for selected ABR correctors

$s=q+r$	Order	β_{real}	β_{imag}	β^*_{imag}	$\kappa_\infty(\underline{C}_2)$	γ_2	γ_3	γ_4	γ_{10}	...	γ_∞
$2=0+2$	3	∞	∞	∞	7	1.41	1.59	1.86	2.36	...	2.45
$3=1+2$	4	8.30	4.32	4.32	9	2.15	2.48	2.87	3.66	...	4.31
$4=2+2$	5	1.05	0	0.93	10	3.39	3.92	4.49	5.93	...	7.11
$3=0+3$	5	∞	∞	∞	18	1.41	1.82	2.21	3.03	...	3.64
$4=1+3$	5	17.18	0.00	9.02	24	1.81	2.32	2.62	4.08	...	4.94
$5=2+3$	6	1.97	0.02	1.89	27	2.45	3.08	3.47	5.80	...	7.03
$4=0+4$	7	∞	∞	∞	34	1.41	1.82	2.21	4.28	...	5.04
$5=1+4$	6	30.16	0.04	15.74	44	1.65	2.11	2.55	4.84	...	5.99
$6=2+4$	7	3.35	0	2.86	50	2.04	2.61	3.15	5.80	...	7.74
$5=0+5$	9	∞	∞	∞	55	1.41	1.82	2.21	4.44	...	6.29
$6=1+5$	7	47.80	0	24.92	69	1.57	2.02	2.44	4.70	...	6.87
$7=2+5$	8	5.23	0.07	4.57	79	1.84	2.36	2.85	5.40	...	8.39

ABM method is much smaller than that of the ABR method. Section 5 will show that ABR is more efficient than ABM because of its better stability characteristics for small numbers of iterations.

4.2. Adams-type correctors with Radau output formula

By adding an output formula (i.e. introducing external stages), it is possible to improve the stability of the corrector method. We shall illustrate this by using output formulas of Radau-type:

$$A^* = E_{ss}, \qquad B^* = O_{ss}, \qquad C^* = U_{ss}V_{ss}^{-1}. \tag{4.5}$$

If the stage order of Y is p, then Y_n has stage order $\min\{p+1, s\}$ and step point order $s+1$.

The stability matrix $M(z)$ associated with $\{(4.1a), (4.5)\}$ is given by

$$M(z) = E_{ss} + zU_{ss}V_{ss}^{-1}M_{\text{Adams}}(z), \tag{4.6}$$

where $M_{\text{Adams}}(z)$ is the stability matrix of (4.1). The large entries in $M_{\text{Adams}}(z)$ are responsible for the possibly poor stability of (4.1). Since the entries of the Radau matrix $U_{ss}V_{ss}^{-1}$ are rather small, it is likely that the large entries in $M_{\text{Adams}}(z)$ are neutralized, so that the stability region of $M(z)$ is improved. This is confirmed by the Tables 4 and 5.

In comparison with the Adams methods without output formula, the higher stability has to be paid for by the additional evaluation of $F(Y_{n-1})$. This can be concluded from the following implementation of the generated PIBRK method (cf. (4.4)):

$$\begin{aligned}
\underline{Y}^{(0)} &= (\underline{A}_0 \otimes I_{dd})Y_{n-1} + h(\underline{B}_0 \otimes I_{dd})F(Y_{n-1}),\\
\bar{Y}^{(j)} &= (\bar{A} \otimes I_{dd})Y_{n-1} + h(\bar{B} \otimes I_{dd})F(Y_{n-1}),\\
\underline{Y}^{(j)} &= (\underline{A} \otimes I_{dd})Y_{n-1} + h(\underline{B} \otimes I_{dd})F(Y_{n-1}) + h(\underline{C} \otimes I_{dd})F(Y^{(j-1)}),\\
Y_n &= (E_{ss} \otimes I_{dd})Y_{n-1} + h(U_{ss}V_{ss}^{-1} \otimes I_{dd})F(Y^{(m-1)}), \qquad y_n = (e_s^{\mathrm{T}} \otimes I_{dd})Y_n,
\end{aligned} \tag{4.7}$$

Table 4
ABM (+ Radau) correctors

Corrector	$s = q + r$	Order	β_{real}	β_{imag}	β^*_{imag}
ABM	$6 = 4 + 2$	7	0.02	0.02	0.02
ABM + R	$6 = 4 + 2$	7	1.51	0.02	1.40
ABM	$6 = 3 + 3$	7	0.17	0.03	0.18
ABM + R	$6 = 3 + 3$	7	1.98	1.79	1.79
ABM	$7 = 4 + 3$	8	0.01	0.01	0.01
ABM + R	$7 = 4 + 3$	8	1.01	0.01	0.95
ABM	$7 = 3 + 4$	8	0.19	0.22	0.22
ABM + R	$7 = 3 + 4$	8	2.18	1.83	1.83
ABM	$7 = 2 + 5$	8	0.47	0.36	0.36
ABM + R	$7 = 2 + 5$	8	2.31	0.03	2.78

Table 5
ABR (+ Radau) correctors

Corrector	$s = q + r$	Order	β_{real}	β_{imag}	β^*_{imag}
ABR	6 = 4 + 2	7	0.01	0.01	0.02
ABR + R	6 = 4 + 2	7	1.52	0.01	1.41
ABR	6 = 3 + 3	7	0.18	0.04	0.19
ABR + R	6 = 3 + 3	7	1.70	0	1.81
ABR	7 = 4 + 3	8	0.01	0.01	0.01
ABR + R	7 = 4 + 3	8	0.98	0	0.97
ABR	7 = 3 + 4	8	0.27	0.06	0.28
ABR + R	7 = 3 + 4	8	1.90	0.01	2.08
ABR	8 = 3 + 5	9	0.40	0.01	0.43
ABR + R	8 = 3 + 5	9	2.27	0.01	2.54

where $j = 1, \ldots, m$. Note that the evaluation of $F(Y_{n-1})$ cannot be replaced by F^*_{n-1} as in (4.4), because then the effect of the stabilizing output formula is not taken into account. However, in the ABR case (where the mth iteration is identical with the output formula), we may replace the last r components of $F(Y_{n-1})$ by those of F^*_{n-1}, without changing the corrector solution. Thus, with respect to the method (4.4), the additional costs are one right-hand side evaluation in the ABR case and two right-hand side evaluations in the ABM case. As before, if we have q additional processors at our disposal, then the total sequential costs per step can be reduced by one right-hand side evaluation (cf. the discussion of the method (4.4)).

The Tables 4 and 5 illustrate the stabilizing effect of adding a Radau output formula.

4.3. *More general correctors*

In the ABM and ABR correctors of Section 4.1, the first q (explicit) stages have order s, so that the resulting stage order can never exceed s. The stage order can easily be increased by using a number of the zero entries occurring in the matrices A, B and C defined in (4.1). For example, adding to the ABR corrector, the $(s-1)$st column of $\bar{A}$ and the last column of $\underline{B}$ for satisfying additional consistency conditions, we obtain a corrector of order $s+1$. This corrector may be considered as a "minimal" modification of the ABR corrector and will be referred to as the *modified* ABR corrector. A drawback of these modified correctors is the rather large magnitude of the entries in the matrix $\bar{A}$, even in the case of this minimal modification. Using more zero entries for a further increase of the stage order leads to dramatically large entries, so that it does not seem feasible to use this approach for constructing correctors with stage order $\geqslant s+2$.

Since the matrix $\underline{C}$ of the modified ABR correctors is no longer defined by the Radau IIA formulas, the step point formula $y_n = (e_s^{\mathrm{T}} \otimes I_{dd}) Y_n$ does not have superconvergence at the step points. Nevertheless, in practical applications we do observe step point order $s+2$, because, again it turns out that $e_s^{\mathrm{T}} B \approx 0$ and $e_s^{\mathrm{T}} C \approx e_s^{\mathrm{T}} U_{ss} V_{ss}^{-1}$ (cf. the discussion in Subsection 4.1.1). Hence, the modified ABR method can be implemented according to (4.4), so that the sequential costs per step are the same as for the ABM and ABR methods.

Table 6
Characteristics for selected modified ABR correctors

$s = q + r$	Order	β_{real}	β_{imag}	β^*_{imag}	$\kappa_\infty(C_2)$	γ_2	γ_3	γ_4	γ_{10}	...	γ_∞
3 = 1 + 2	5	4.15	0	1.44	12	2.33	2.72	3.12	4.14	...	4.98
4 = 1 + 3	6	7.16	0	2.41	34	1.83	2.37	2.86	4.85	...	5.97
6 = 2 + 4	8	0.99	0.02	1.17	64	2.05	2.63	3.20	6.04	...	8.73
7 = 2 + 5	9	1.42	0.01	1.74	107	1.85	2.38	2.89	5.66	...	9.55

The characteristics of a few modified ABR correctors are summarized in Table 6. A comparison with the corresponding ABR correctors of Table 3 reveals that the modification leads to comparable convergence boundaries and smaller stability boundaries. However, the stage order and (effective) step point order is raised by one. A detailed investigation of this promising family of methods will be subject of future research.

5. Numerical experiments

Our numerical tests were performed using 15-digits arithmetic. The accuracies obtained are given by the number of correct digits Δ, defined by writing the maximum norm of the absolute error at the endpoint in the form $10^{-\Delta}$. The PIBRK method is implemented according to (4.4) with the PC pairs (AB, ABM) and (AB, ABR), where the correctors have orders 7 and 8, and are selected from the Tables 2 and 3. In view of the relatively high corrector orders and the 15-digits arithmetic, we did not use Hermite predictors.

First, we will make a mutual comparison between ABM and ABR correctors, using a constant number of iterations per step. For that purpose, we select the following well-known test problems (cf. [3]), viz. the Fehlberg problem

$$\begin{aligned} y_1' &= 2ty_1 \log(\max\{y_2, 10^{-3}\}), \quad y_1(0) = 1, \\ y_2' &= -2ty_2 \log(\max\{y_1, 10^{-3}\}), \quad y_2(0) = \mathrm{e}, \end{aligned} \qquad 0 \leqslant t \leqslant 5, \tag{5.1}$$

and the Euler problem

$$\begin{aligned} y_1' &= y_2 y_3, & y_1(0) &= 0, \\ y_2' &= -y_1 y_3, & y_2(0) &= 1, \\ y_3' &= -0.51 y_1 y_2, & y_3(0) &= 1, \end{aligned} \qquad 0 \leqslant t \leqslant 20. \tag{5.2}$$

Secondly, in Section 5.2, we add a dynamic iteration strategy to the 8th-order (AB, ABR) and we compare this code with three existing codes from the literature. The paper is concluded with a performance evaluation of these four codes on the Brusselator problem [3, p. 381] which was transformed into an IVP for ODEs of dimension $d = 882$.

5.1. Comparison of ABM and ABR correctors

We applied the PC pairs (AB, ABM) and (AB, ABR) with $s = 2 + 4$. These correctors are both of order 7, require four processors, and are equally expensive. The Tables 7 and 8 present Δ-values for a few values of h and m (overflow is indicated by $*$). From these figures, we may conclude that the efficiency of the two methods is comparable in the case of convergence, but for larger stepsizes (AB, ABR) is more robust than (AB, ABM). This can be explained by the larger stability regions of the (AB, ABR) method.

5.2. Comparisons with DOPRI8, PIRK8 and PIRK10

Since (AB, ABR) pairs are more stable and therefore more robust, we restrict our considerations to this family of PC methods. In particular, we tested the $s = 2 + 5$ method. This parallel, eighth-order (AB, ABR) method was compared with the 8(7) RK pair of Prince and Dormand [10]

Table 7
Δ-values for (5.1) obtained by (4.4) with (AB, ABM) and (AB, ABR) PC pairs

PC pair	$s = q + r$	h^{-1}	$m = 1$	$m = 2$	$m = 3$	$m = 4$	...	$m = \infty$
(AB, ABM)	2 + 4	10	*	*	*	*	...	3.7
(AB, ABR)	2 + 4	10	*	*	2.6	3.7	...	4.2
(AB, ABM)	2 + 4	20	0.5	*	3.0	6.5	...	7.5
(AB, ABR)	2 + 4	20	0.5	4.5	5.9	6.5	...	6.9
(AB, ABM)	2 + 4	40	4.6	*	8.8	9.3	...	9.6
(AB, ABR)	2 + 4	40	4.6	7.2	9.0	9.2	...	9.3
(AB, ABM)	2 + 4	80	7.9	9.2	10.7	10.7	...	10.7
(AB, ABR)	2 + 4	80	7.9	9.4	12.0	11.5	...	11.5

Table 8
Δ-values for (5.2) obtained by (4.4) with (AB, ABM) and (AB, ABR) PC pairs

PC pair	$s = q + r$	h^{-1}	$m = 1$	$m = 2$	$m = 3$	$m = 4$	...	$m = \infty$
(AB, ABM)	2 + 4	1	*	*	*	*	...	4.6
(AB, ABR)	2 + 4	1	*	*	3.2	3.9	...	4.9
(AB, ABM)	2 + 4	2	*	*	2.2	6.4	...	6.5
(AB, ABR)	2 + 4	2	*	4.6	5.4	6.6	...	6.4
(AB, ABM)	2 + 4	4	4.4	*	8.0	8.4	...	8.4
(AB, ABR)	2 + 4	4	4.4	7.7	8.1	8.3	...	8.3
(AB, ABM)	2 + 4	8	7.2	9.1	10.5	10.6	...	10.6
(AB, ABR)	2 + 4	8	7.1	10.2	10.4	10.4	...	10.4

and the parallel PC methods based on Gauss–Legendre correctors of order 8 and 10. For the Dormand–Prince method we took the DOPRI8 implementation of Hairer et al. [3], and for the Gauss–Legendre methods we used the four and five-processor one-step codes PIRK8 and PIRK10 developed in [5].

The (AB, ABR) method was equipped with a dynamic iteration strategy based on the requirement that the step point component of the residue left on substitution of the jth iterate into the stage vector equation should be less than a tolerance parameter TOL, i.e.

$$\|(e_s^{\mathrm{T}} \otimes I_{dd})\boldsymbol{R}^{(j)}\| \leqslant \mathrm{TOL},$$

where

$$\boldsymbol{R}^{(j)} := \boldsymbol{Y}^{(j)} - (E_{ss} \otimes I_{dd})\boldsymbol{Y}_{n-1} - h(B \otimes I_{dd})\boldsymbol{F}(\boldsymbol{Y}_{n-1}) - h(C \otimes I_{dd})\boldsymbol{F}(\boldsymbol{Y}^{(j)}).$$

According to (4.4) we may write

$$\boldsymbol{R}^{(j)} = \boldsymbol{Y}^{(j)} - \boldsymbol{Y}^{(j+1)} - h(B \otimes I_{dd})(\boldsymbol{F}(\boldsymbol{Y}_{n-1}) - \boldsymbol{F}^*_{n-1}).$$

Clearly, the error caused by the iteration process should be smaller than the local truncation error. This is achieved by requiring TOL to be a factor δ less than the local error. In our experiments, we shall estimate the local error at t_{n-1} by $\|(e_s^{\mathrm{T}} \otimes I_{dd})(\boldsymbol{Y}_{n-1} - \boldsymbol{Y}^{(0)}_{n-1})\|$, where $\boldsymbol{Y}^{(0)}_{n-1}$ denotes the prediction in the preceding step. Using the maximum norm and observing that $(e_s^{\mathrm{T}} \otimes I_{dd})(\boldsymbol{F}(\boldsymbol{Y}_{n-1}) - \boldsymbol{F}^*_{n-1})$ vanishes in the case of ABR correctors, we are led to the stopping criterion

$$\|(e_s^{\mathrm{T}} \otimes I_{dd})(\boldsymbol{Y}^{(j)} - \boldsymbol{Y}^{(j+1)})\|_\infty \leqslant \delta \|(e_s^{\mathrm{T}} \otimes I_{dd})(\boldsymbol{Y}_{n-1} - \boldsymbol{Y}^{(0)}_{n-1})\|_\infty. \tag{5.3}$$

Below, the resulting implementation will be referred to as the ABR8 code (we did not yet implement a stepsize strategy, so that the results produced by this code may be improved when this facility is included).

As a third test problem, we take the Brusselator problem (see [3, p. 381]), defined by

$$\begin{aligned} \frac{\partial u}{\partial t} &= 1 + u^2 v - 4.4u + \alpha\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right), \\ \frac{\partial v}{\partial t} &= 3.4u - u^2 v + \alpha\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right), \end{aligned} \qquad 0 \leqslant x \leqslant 1, \quad 0 \leqslant t \leqslant 23.5, \tag{5.4a}$$

supplemented with homogeneous Neumann boundary conditions and the initial conditions

$$\begin{aligned} u(t = 0, x, y) &= 0.5 + y, \\ v(t = 0, x, y) &= 1 + 5x. \end{aligned} \tag{5.4b}$$

Furthermore, $\alpha = 2 \cdot 10^{-3}$ and N (the number of equidistant points in the spatial direction) is set to 21, resulting in an ODE-system of dimension 882.

The Tables 9–11 show results for the various methods (for the first two test problems, the results for DOPRI8, PIRK8, and PIRK10 were taken from [5]). In the ABR8 code, δ is set to 10^{-4}. To facilitate a comparison, the listed numbers of sequential right-hand side evaluations have been obtained by interpolation to arrive at *integer* values of Δ. Furthermore, we list the (averaged) factor

Table 9
Number of sequential right-hand side evaluations for the Fehlberg problem (5.1)

Method	$\Delta = 5$	$\Delta = 6$	$\Delta = 7$	$\Delta = 8$	$\Delta = 9$	$\Delta = 10$	$\Delta = 11$	μ
DOPRI8	595	759	963	1227	1574	1990	2503	2.3
PIRK8	379	495	623	786	978	1383	1874	1.5
PIRK10	327	388	490	704	884	977	1078	1.2
ABR8	240	335	430	532	689	846	1067	1.0

Table 10
Number of sequential right-hand side evaluations for the Euler problem (5.2)

Method	$\Delta = 6$	$\Delta = 7$	$\Delta = 8$	$\Delta = 9$	$\Delta = 10$	$\Delta = 11$	$\Delta = 12$	μ
DOPRI8	415	576	728	898	1133	1422	1817	2.9
PIRK8	294	381	534	728	961	1172	1746	2.3
PIRK10	252	297	357	426	580	730	920	1.5
ABR8	160	192	223	293	379	506	643	1.0

Table 11
Number of sequential right-hand side evaluations for the Brusselator problem (5.4)

Method	$\Delta = 3$	$\Delta = 4$	$\Delta = 5$	$\Delta = 6$	$\Delta = 7$	$\Delta = 8$	$\Delta = 9$	μ
DOPRI8	1594	2376	2908	3570	4532	5985	7856	1.9
PIRK8	1161	1579	1950	2462	3225	4190	5448	1.3
PIRK10	901	1362	1824	2148	2673	3265	3989	1.1
ABR8	591	1169	1747	2237	2624	3288	3953	1.0

by which the existing codes are more expensive than ABR8 (this factor is denoted by μ). These tables clearly show that ABR8 is the most efficient solver. We see that the speed-up factor of ABR8 with respect to the 8th-order code DOPRI8 (to be considered as one of the most efficient sequential codes) ranges from 1.9 to 2.9. For the four-processor PIRK8 code of order 8, and the five-processor 10th-order code PIRK10 this factor is in the range 1.3–2.3 and 1.1–1.5, respectively.

6. Concluding remarks

The search for efficient parallel PC methods reported in this paper has resulted in several fastly converging and sufficiently stable PC pairs. With respect to the fully automatic code DOPRI8, the averaged speed-up factor of the fixed stepsize, five-processor ABR8 code ranges from 1.9 to 2.9. The efficiency of this code can be improved by including a stepsize strategy. If the local error estimate is based on local Richardson extrapolation where the "reference" solution is computed in parallel on an additional set of processors, then these processors can also be used for saving one function call

per step (see the discussion of the scheme (4.4)). In the numerical examples of this paper, this would increase the speed-up factor by about 20%.

Acknowledgements

The authors are grateful to Dr. K.J. in 't Hout for his interest and useful remarks during the preparation of this paper.

References

[1] L.G. Birta and O. Abou-Rabia, Parallel block predictor-corrector methods for ODEs, *IEEE Trans. Comput.* **C-36** (1987) 299–311.
[2] M.T. Chu and H. Hamilton, Parallel solution of ODE's by multi-block methods, *SIAM J. Statist. Comput.* **8** (1987) 342–353.
[3] E. Hairer, S.P. Nørsett and G. Wanner, *Solving Ordinary Differential Equations I. Nonstiff Problems*, Springer Series in Comp. Math., Vol. 8 (Springer, Berlin, 1987).
[4] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II. Stiff and Differential–Algebraic Problems*, Springer Series in Comp. Math., Vol. 14 (Springer, Berlin, 1991).
[5] P.J. van der Houwen and B.P. Sommeijer, Parallel iteration of high-order Runge–Kutta methods with stepsize control, *J. Comput. Appl. Math.* **29** (1990) 111–127.
[6] P.J. van der Houwen and B.P. Sommeijer, Block Runge–Kutta methods on parallel computers, *Z. angew. Math. Mech.* **72** (1992) 3–18.
[7] K.R. Jackson and S.P. Nørsett, The potential for parallelism in Runge–Kutta methods, Part I: RK formulas in standard form, *SIAM J. Numer. Anal.* **32** (1995) 49–82.
[8] I. Lie, Some aspects of parallel Runge–Kutta methods, Report 3/87, Dept. Mathematics, University of Trondheim, 1987.
[9] S.P. Nørsett and H.H. Simonsen, Aspects of parallel Runge–Kutta methods, in: A. Bellen, C.W. Gear and E. Russo, Eds, *Numerical Methods for Ordinary Differential Equations, Proc. L'Aquila 1987*, Lecture Notes in Mathematics, Vol. 1386 (Springer, Berlin, 1989).
[10] P.J. Prince and J.R. Dormand, High order embedded Runge–Kutta formulae, *J. Comput. Appl. Math.* 7 (1981) 67–75.
[11] R.S. Varga, *Matrix Iterative Analysis* (Prentice-Hall, Englewood Cliffs, NJ, 1962).