BLOCK RUNGE-KUTTA METHODS

P.J. van der Houwen
Amsterdam, The Netherlands

1. Introduction

The work surveyed in this paper is joint work with B.P. Sommeijer and W. Couzy from the Centre for Mathematics and Computer Science in Amsterdam. Full details may be found in the references [7]-[11].
Our starting point is the conventional s-stage RK method for the initial-value problem

$$(1.1)\quad \frac{dy(t)}{dt} = f(y(t)),\ y(t_0) = y_0,$$

defined by

$$(1.2)\quad \begin{array}{|ccccc} a_{11} & . & . & . & a_{1s} \\ : & : & : & : & : \\ a_{s1} & . & . & . & a_{ss} \\ \hline b_1 & . & . & . & b_s \end{array}, \qquad \begin{aligned} Y^{(i)} &= y_n + h\sum_{j=1}^{s} a_{ij}\, f(Y^{(j)}),\quad i = 1, \dots, s; \\ y_{n+1} &= y_n + h\sum_{j=1}^{s} b_j\, f(Y^{(j)}),\quad n = 0, 1, \dots . \end{aligned}$$

The general structure of the Runge-Kutta-type methods considered in this paper is a direct generalization of this conventional method. We introduce k-dimensional block vectors Y_n, the components of which are numerical approximations to the exact solution values:

$$Y_{n+1} := (y_{n,1}, y_{n,2}, \dots, y_{n,k})^T,$$

where $y_{n,j}$ denotes a numerical approximation to the exact solution value $y(t_n+c_jh)$. We shall assume that $c_k=1$, while the other values of c_j are allowed to be any real number. Thus, the last component of the block vector Y_{n+1} always provides an approximation to $y(t_{n+1})$. The vector $c:=(c_j)$ will be called the *block point vector*. For scalar ODEs, we now define the *s-stage block RK (BRK) method*

$$(1.3)\quad \begin{array}{c|cccccc} A_1 & A_{11} & . & . & . & . & A_{1s} \\ : & : & : & : & : & : & : \\ A_s & A_{s1} & . & . & . & . & A_{ss} \\ \hline B_0 & B_1 & . & . & . & . & B_s \end{array}, \qquad \begin{aligned} Y^{(i)} &= A_iY_n + h\sum_{j=1}^{s} A_{ij}\, f(Y^{(j)}),\quad i = 1, \dots, s; \\ Y_{n+1} &= B_0Y_n + h\sum_{j=1}^{s} B_j\, f(Y^{(j)}),\quad n = 0, 1, \dots, \\ y_{n+1} &:= (e_k)^TY_{n+1}, \end{aligned}$$

where the matrices A_i, A_{ij}, B_0 and B_j respectively are r-by-k, r-by-r, k-by-k and k-by-r matrices, e_k denotes the kth unit vector, and where we use the convention that for any given vector $v=(v_j)$, $f(v)$ denotes the vector with entries $f(v_j)$. The method (1.3) can be considered as the block analogue of (1.2). It is straightforwardly extended to systems of ODEs and therefore also to nonautonomous equations. In order to start the method, one needs the initial vector Y_0, which requires as many starting values as there are distinct values c_j $(j=1,\dots,k)$. If $c=e$, e being the vector with unit entries, then only one starting value is needed. In fact, in this case, the BRK method reduces to a one-step RK method providing k approximations to $y(t_{n+1})$. We define the order of BRK methods by the order of y_{n+1}.
The method is *explicit* if the matrices A_{ij} vanish for $j\geq i$, *diagonally-implicit* if the matrices A_{ij} vanish for $j>i$ and if the matrices A_{ii} are diagonal, and *implicit* otherwise. In this survey, we restrict our considerations to explicit and diagonally implicit methods. For such methods, the r components of the

block vectors $Y^{(i)}$ can be computed in parallel. Hence, if r processors are available, then the required computational time per integration step is at most the time needed for computing s block components sequentially. We define the *optimal number of processors* as the number of processors for which the number of (sequential) block component evaluations per step is minimal. In this connection, we remark that often less than r processors are needed for implementing the BRK method.

In the explicit and diagonally-implicit case, the representation (1.3) is very convenient for implementing the method on a computer, because the actual code is a direct translation of the scheme (1.3) and the instructions for the computer in order to exploit the built-in parallelism of the method are obvious. Conversely, by representing a given method in BRK format, it is readily seen whether the method is suitable for use on a parallel computer or not.

Below we present examples of methods from the literature which have been constructed for use on parallel computers. We shall use the BRK notation defined above and we give the order p and the number of processors needed for implementation.

Method of Miranker and Liniger [16]. Explicit, two-processor method which seems to be the first method constructed for parallel solution of initial-value problems:

$$\text{(1.4)}\quad \left.\begin{array}{cccc|cccc}
1 & 0 & 0 & 0 & & & & \\
0 & 1 & 0 & 0 & & & & \\
0 & 0 & 1 & 0 & & & & \\
0 & 0 & 0 & 1 & & & & \\
\hline
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & -1/3 & 4/3 & 8/3 & -5/3 \\
0 & 0 & 0 & 1 & 1/24 & -5/24 & 9/24 & 19/24
\end{array}\right. ,\quad c = (-1, 0, 2, 1)^T,\ p=4,\ k=r=4,\ s=1.$$

Method of Worland [21]. Explicit two-processor method based on the PECE mode of a predictor of Shampine & Watts [19] and the Clippinger-Dimsdale corrector [3]:

$$\text{(1.5)}\quad \left.\begin{array}{cccc|cccccccc}
1 & 0 & 0 & 0 & & & & & & & & \\
0 & 1 & 0 & 0 & & & & & & & & \\
0 & 0 & 1 & 0 & & & & & & & & \\
0 & 0 & 0 & 1 & & & & & & & & \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & & & & \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & & & & \\
0 & 1/3 & 1/3 & 1/3 & 0 & 1/4 & -1/3 & 13/12 & & & & \\
0 & 1/3 & 1/3 & 1/3 & 0 & 29/24 & -3 & 79/24 & & & & \\
\hline
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 5/24 & 0 & 0 & 1/3 & -1/24 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1/6 & 0 & 0 & 2/3 & 1/6
\end{array}\right. ,\quad c = (-1/2, 0, 1/2, 1)^T,\ p=k=r=4,\ s=2.$$

Multi-block method of Chu and Hamilton [2]. Explicit two-processor method based on the PECE mode of a PC pair using full block methods:

$$
(1.6)\qquad
\begin{array}{cc|cccc}
1 & 0 & & & & \\
0 & 1 & & & & \\
5 & -4 & 1 & 2 & & \\
28 & -27 & 6 & 9 & & \\
\hline
0 & 1 & -1/48 & 13/48 & 13/48 & -1/48 \\
0 & 1 & 0 & 1/6 & 2/3 & 1/6
\end{array}
,\quad c = (1/2,\ 1)^T,\ p=4,\ k=r=s=2.
$$

BRK method not derived from PC pairs [7]. Explicit two-processor method requiring two starting values:

$$
(1.7)\qquad
\begin{array}{ccc|ccc}
1 & 0 & 0 & & & \\
0 & 0 & 1 & & & \\
0 & 1-a & a & & & \\
\hline
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & \frac{5}{6(a-1)} & 0 \\
0 & 0 & 1 & -\frac{7}{3}+a & \frac{7}{3}-a & 1
\end{array}
,\quad a = 4/3 \pm \sqrt{13/12},\ c = (0,\ 1,\ 1)^T,\ p=3,\ k=r=3,\ s=1.
$$

DIRK method of Iserles and Nørsett [13]. L-stable, diagonally-implicit two-processor method:

$$
(1.8)\qquad
\begin{array}{c|cccc}
1 & 1/2 & 0 & & \\
1 & 0 & 2/3 & & \\
1 & -5/2 & 5/2 & 1/2 & 0 \\
1 & -5/3 & 4/3 & 0 & 2/3 \\
\hline
1 & -1 & 3/2 & -1 & 3/2
\end{array}
,\quad c = e,\ p=4,\ k=1,\ r=s=2.
$$

2. Construction of high-order methods

The order of the above-mentioned methods do not exceed $p=4$. In this section we outline the construction of higher-order methods. We distinguish methods requiring only one starting value (RK methods) and methods requiring several starting values (multistep methods).

2.1. RK methods

Let O_{ij} be the i-by-j null matrix, let A be a given r-by-r matrix, let B and D be r-by-r diagonal matrices, let b be a given vector of dimension r, and let $\mathbf{0}$ and e respectively denote the null and unit vector whose dimension should be clear from the context in which it is used. Setting $k=s$, we define the matrices

(2.1a) $A_j := (0 \;\dots\; 0\; e),\; i = 1, \dots, s;\;\; B_0 := (0 \;\dots\; 0\; e);\;\; O := O_{rr};$

$B_1 := (b \dots 0\,0\,0\,0)^T,\;\; B_2 := (0\; b\; 0 \dots 0\; 0)^T, \dots,\;\; B_s := (0 \dots 0\,0\,0\; b)^T,$

and we consider BRK methods generated by the Butcher-type array

(2.1b)
$$\begin{array}{c|cccccc}
A_1 & B \\
A_2 & C & D \\
A_3 & O & A\text{-}D & D \\
A_4 & O & O & A\text{-}D & D \\
\vdots & \vdots & & & \ddots & \ddots \\
A_s & O & \cdot & \cdot & O & A\text{-}D & D \\
\hline
B_0 & B_1 & \cdot & \cdot & B_{s-2} & B_{s-1} & B_s
\end{array}, \quad c = e,$$

with y_{n+1} defined by (1.3). Evidently, this tableau generates a particular family of DIRK methods providing k approximations to the true solution $y(t_{n+1})$. If B=D=O, then (2.1) reduces to an explicit RK (ERK) method. It is readily verified that (2.1) originates from iterating the r-stage RK method

(2.2) $\quad Y = y_n e + hAf(Y),\;\; y_{n+1} = y_n + hb^T f(Y).$

(for details of the iteration process we refer to [8] and [10]). The method (2.2) will be called the generating *corrector formula*. Usually, this corrector formula is an implicit RK method, but the considerations below also apply to the case where (2.2) is an ERK method.

In the following subsections, we construct matrices A, B, C and D and a vector b such that the components $y_{n,i}$ of Y_{n+1} provided by the BRK method (2.1) have orders p-s+i, i=1,...s. Thus, we shall construct a pth-order DIRK method with embedded formulas of orders p-1, ..., p-s+1. We start with explicit methods, i.e., B=D=O.

2.1.1. ERK methods

We summarize the results we obtained for parallel ERK methods (cf. [8]). For that purpose, we need the notion of the number of *sequential* stages (or, as it was termed by Iserles and Nørsett [13], the number of *effective* stages).

Definition 2.1. An ERK method is said to require s *sequential stages* if the computation time required for evaluating all right-hand sides in one step is s times the computation time required by one right-hand side evaluation, assuming that sufficiently many processors are available. □

Thus, if B=D=O, then (2.1) is an ERK method requiring s sequential stages. In the paper of Iserles and Nørsett, the following theorem was proved:

Theorem 2.1. ERK methods of order p necessarily require at least p sequential stages. □

This assertion led Nørsett and Simonsen [17] to pose the problem whether it is always possible to find ERK methods of order p using not more than p sequential stages, assuming that sufficiently many processors are available. Such methods were called optimal in [17].

Definition 2.2. An ERK method is called *optimal* if its number of sequential stages equals its order. □

For p≤4, the above problem can (of course) be answered positively and for p=5 Nørsett and Simonsen mention a 6-stage, 5th-order method of Butcher possessing 5 sequential stages. For higher-order ERK methods, the following theorem [8] solves the problem posed by Nørsett and Simonsen:

Theorem 2.2. If B=D=O, C=A, and if the corrector formula (2.2) is of order s, then the method defined by (2.1) is optimal and the components $y_{n,i}$ of Y_{n+1} have orders i, i=1,...,s. □

For any even p there exist RK methods of order p requiring p/2 stages (Gauss-Legendre methods) and for any odd p there exist RK methods of order p requiring (p+1)/2 stages (Radau methods). Using these methods as generating corrector formula (2.2), we have in (2.1) r=[(p+1)/2], where [.] denotes the integer part function, and we are led to the corollary:

Corollary 2.1. There exist optimal ERK methods of any order p requiring [(p+1)/2] processors. □

In order to demonstrate that the use of parallel computers may save computing time, we compare the 'parallel, iterated' RK (PIRK) methods of this section with the so-called 8(7) method of Prince and Dormand [18]. According to [6] this method is nowadays generally considered as one of the most efficient methods with automatic stepsize control for TOL-values approximately in the range 10^{-7} to 10^{-13}. We compare the DOPRI8 code, as given by Hairer, Nørsett and Wanner [6], with the PIRK method based on the Gauss-Legendre correctors of orders 8 and 10. To let the comparison of the DOPRI8 code and the PIRK codes not be influenced by a different stepsize strategy, we equiped the PIRK codes with the same strategy. These codes are respectively denoted by PIRK8 and PIRK10.
As test problem we take the equation of motion (cf. Problem B5 from [12]):

$$\begin{array}{lll} y_1' = y_2y_3, & y_1(0) = 0, & \\ y_2' = -y_1y_3, & y_2(0) = 1, & 0 \le t \le T. \\ y_3' = -.51y_1y_2, & y_3(0) = 1, & \end{array} \tag{2.3}$$

In Table 2.1, we have listed the values Δ\N, where Δ denotes the number of correct decimal digits at the endpoint (i.e., we write the maximum norm of the error at t=T in the form $10^{-\Delta}$) and where N denotes the total number of sequential right-hand side evaluations performed during the integration process. For tolerances TOL running from 10^{-5} up to 10^{-12} we list the values of N which were found for a number of values of Δ.

Table 2.1. Values of N for Problem B5 from [12] at T=20.

Method	Δ=6	Δ=7	Δ=8	Δ=9	Δ=10	Δ=11	Δ=12
DOPRI8	415	576	728	898	1133	1422	1817
PIRK8	294	381	534	728	961	1172	1746
PIRK10	252	297	357	426	580	730	920

2.1.2. DIRK methods

Our main results for parallel DIRK methods of the form (2.1) with D≠O obtained in [10] are summarized below.

Since the bulk of the computational effort required by these methods goes into the solution of the s-1 systems of equations, we define:

Definition 2.3. A DIRK method is said to require s *sequential stages* if the computation time required for solving all systems of equations in one step is s times the computation time required by solving one system of equations, assuming that sufficiently many processors are available. □

Thus, if both B and D do not vanish, then (2.1) is a DIRK method requiring s sequential stages. I B=O and D≠O, then s-1 sequential stages are required. The following theorem determines the order of the approximations $y_{n,i}$:

Theorem 2.3. If the corrector formula (2.2) is of order p*, then the components $y_{n,i}$ of Y_{n+1} defined by the method (2.1) have orders p_i, i=1, ... , s which are given by:

Type IA.1:	$B=O, C=A-D$	$\Rightarrow p_i=\min\{p^*,i\}$
Type IB.1:	$B=D, C=A-D$	$\Rightarrow p_i=\min\{p^*,i\}$
Type IB.2:	$B=D=\operatorname{diag}(Ae), C=A-D$	$\Rightarrow p_i=\min\{p^*,i+1\}$
Type IC.1:	$B=O$	$\Rightarrow p_i=\min\{p^*,i-1\}, i\geq 2$
Type IC.2:	$B=O, D=\operatorname{diag}(Ae-Ce))$	$\Rightarrow p_i=\min\{p^*,i\}$
Type IC.3:	$B=O, D=\operatorname{diag}(Ae-Ce)), DAe=A^2e$	$\Rightarrow p_i=\min\{p^*,i+1\}$. □

Since $y_{n+1}:=y_{n,s}$, it follows from this theorem that Type IC.3 methods are the most efficient ones because order p^* requires only p^*-1 sequential stages. Unfortunately, in general Type IC.3 methods are not A-stable. The following corollary of Theorem 2.3 can be proved[10]:

Corollary 2.2. The order of A-stable Type IC methods cannot exceed their number of sequential stages plus 1, unless $b^TD^{-1}[I-AD^{-1}]^{s+2}Ce=0$. □

Thus, in order to construct a DIRK method of Type IC of order 4, at least 3 sequential stages are required, whereas the L-stable, fourth-order DIRK method (1.8) of Iserles and Nørsett requires only 2 sequential stages. On the other hand, (2.1) also generates formulas of orders 3, 2 and 1. However, more important is the possibility to generate embedded methods of orders higher than $p=4$ possessing quit favourable stability properties. First we consider the case where D has constant diagonal elements. This case allows a theoretical analysis. Analogous to an analysis by Wolfbrandt [20] of SDIRK methods (that is, RK methods with constant diagonal in the Butcher tableau), stating that for $1\leq p\leq 6$ and $p=8$ the stability function of SDIRK methods of order p and requiring p stages is L-stable, and for $p=7$ and $9\leq p\leq 15$ it does not (see [1, p. 248] for a summary of Wolfbrandt's result), we arrive at the theorem:

Theorem 2.4. Let the corrector formula (2.2) be of order $p^*=s$, then there exist values of d such that the Type IB.1 methods are L-stable for $1\leq s\leq 6$ and $s=8$. □

Within the class of methods with $D=dI$, it is possible to construct (at the cost of an additional sequential stage) still higher-order L-stable approximations. This can be achieved by choosing the corrector formula (2.2) such that $b=A^Te_r$ (so-called *stiffly accurate* corrector formula) and we have to define y_{n+1} by $y_{n+1}:=(e_r)^TY^{(s)}$. Let us call these methods a Type II method. First we state the analogue of Theorem 2.3 for Type II methods:

Theorem 2.5. If the corrector formula (2.2) is of order p^* and satisfies $b=A^Te_r$, then the order of $y_{n,i}$ is given by Theorem 2.3 and $y_{n+1}:=(e_r)^TY^{(s)}$ is also stiffly accurate with order p given by:

Type IIA.1:	$B=O, C=A-D$	$\Rightarrow p=\min\{p^*,s-1\}$
Type IIB.1:	$B=D, C=A-D$	$\Rightarrow p=\min\{p^*,s-1\}$
Type IIB.2:	$B=D=\operatorname{diag}(Ae), C=A-D$	$\Rightarrow p=\min\{p^*,s\}$
Type IIC.1:	$B=O$	$\Rightarrow p=\min\{p^*,s-2\}$
Type IIC.2:	$B=O, D=\operatorname{diag}(Ae-Ce))$	$\Rightarrow p=\min\{p^*,s-1\}$
Type IIC.3:	$B=O, D=\operatorname{diag}(Ae-Ce)), DAe=A^2e$	$\Rightarrow p=\min\{p^*,s\}$. □

The main stability results obtained for the Type II methods are given by

Theorem 2.6. If the corrector formula (2.2) is stiffly accurate ($b=A^Te_r$) and has order $p^*=s-1$, then there exist values of d such that:

(a) Type IIA.1 methods are L-stable for $1\leq s\leq 7$ and $s=9$.
(b) Type IIB.1 methods are L-stable for $1\leq s\leq 9$ and $s=11$. □

We illustrate the performance of the methods by integrating a test problem proposed by Kaps [1981]:

$$\text{(2.4)}\qquad \frac{dy_1}{dt}=-(2+\frac{1}{\varepsilon})y_1+\frac{1}{\varepsilon}y_2,\ \frac{dy_2}{dt}=y_1-y_2(1+y_2),\ y_1(0)=y_2(0)=1,\ 0\leq t\leq 1,$$

with exact solution $y_1=\exp(-2t)$ and $y_2=\exp(-t)$ for all values of the parameter ε. We tested several correctors and all types of methods which are L-stable. In Table 2.2 the values of Δ are listed (cf. Table 2.1). Notice that the Type II methods require a stiffly accurate corrector (such as the Radau II formulas) and that L-stable, seventh-order methods are only possible within the family of Type IIB methods. In all experiments we observe the phenomenon of order reduction (if p is the order of the method, then, on halving the step size, the value of Δ should increase by .3p if no order reduction is exhibited). Roughly speaking, we see a reduction by one order.

Table 2.2. Values of Δ for problem (2.4) at $t=1$ with $\varepsilon=10^{-2}$.

Type	Corrector	Order	h=1/4	h=1/8	h=1/16	h=1/32	h=1/64	Seq. Stages	Proc.
IB.1	Radau IIA	3	3.9	4.7	5.4	5.8	6.4	3/h	2
	Gauss-Legendre	4	3.0	3.7	4.4	5.2	6.0	4/h	2
	Explicit RK	4	3.1	3.8	4.5	5.3	6.1	4/h	4
	Radau IIA	5	3.5	4.3	5.2	6.2	7.4	5/h	3
	Gauss-Legendre	6	3.2	4.1	5.0	6.2	7.5	6/h	3
IIA.1	Radau IIA	3	3.6	4.3	4.9	5.6	6.2	3/h	2
	Radau IIA	5	3.8	4.5	5.3	6.3	7.5	5/h	3
IIB.1	Radau IIA	3	4.2	4.6	5.2	5.9	6.7	4/h	2
	Radau IIA	5	6.2	5.6	6.1	6.9	8.0	6/h	3
	Radau IIA	7	4.3	5.2	6.2	7.7	9.4	8/h	4

We also considered Type II methods with matrix D possessing *distinct* diagonal entries. From Theorem 2.5 and by means of computer calculations found:

Corollary 2.3. Let the corrector formula (2.2) be defined by the r-stage Radau IIA formula, and let $s=2r-1$, then the Type IIC.3 methods have order $p=2r-1$, they require p-1 sequential stages, and they are $A(\alpha_p)$-stable where $\alpha_3=\pi/2$, $\alpha_5 \approx \pi/2 - 3 10^{-3}$, $\alpha_7 \approx \pi/2 - 4 10^{-2}$. □

Thus, by using distinct diagonal values we can save a sequential stage.

We conclude this section with a comparison of the various DIRK methods available in the literature and those discussed in this paper. The methods designed for parallel computation are indicated by PDIRK. Effectively, all methods in this survey are SDIRK methods. The order range of the embedded formulas are listed in the column headed with p_{emb}.

Method	p	Stages	Seq. st.	Processors	Stability	p_{emb}	Reference/Specification
SDIRK	p=3	p-1	p-1	1	A-stable		Nørsett [1974]
SDIRK	p=4	p-1	p-1	1	A-stable		Crouziex [1976], Alexander [1977]
SDIRK	p=3	p	p	1	S-stable	<p	Cash [1979], Cash & Liem [1980]
SDIRK	p=4	p+1	p+1	1	S-stable	<p	Cash [1979], Cash & Liem [1980]
PDIRK	p=4	p	p-2	2	L-stable	p-1	Iserles & Nørsett [1988]
PDIRK	p≤6	ps	p	[(p+1)/2]	L-stable	<p	Type IIA.1, D=dI
PDIRK	p=8	ps	p	[(p+1)/2]	L-stable	<p	Type IIA.1, D=dI
PDIRK	p≤8	(p+1)s	p+1	[(p+1)/2]	L-stable	<p	Type IIB.1, D=dI
PDIRK	p=10	(p+1)s	p+1	[(p+1)/2]	L-stable	<p	Type IIB.1, D=dI
PDIRK	p=3	ps	p	[(p+1)/2]	L(α)-stable	<p	Type IIB.2, α=87.47
PDIRK	p=5	ps	p	[(p+1)/2]	L(α)-stable	<p	Type IIB.2, α=89.12
PDIRK	p=3	(p-1)s	p-1	[(p+1)/2]	A-stable	p-1	Type IIC.3
PDIRK	p=5	(p-1)s	p-1	[(p+1)/2]	A(α)-stable	[2,p-1]	Type IIC.3, α=89.997
PDIRK	p=7	(p-1)s	p-1	[(p+1)/2]	A(α)-stable	[2,p-1]	Type IIC.3, α=89.959

2.2. Multistep methods

We consider the special two-stage method

$$\begin{array}{c|cc} I & O & O \\ A & B & C \\ \hline A & B & C \end{array}\,, \quad c \neq e: \qquad Y_{n+1} = AY_n + hBf(Y_n) + hCf(Y_{n+1}),$$

where A, B and C are arbitrary k-by-k matrices (here, r=k). Unlike the RK methods of the preceding sections, not just one component of Y_n plays a role in this scheme, and therefore it belongs to the class of multistep methods.

The order condtions for methods of the above special two-stage form are extremely simple:

Theorem 2.7. Let the error vectors C_j be defined by

$$C_j := A(c - e)^j + j[B(c - e)^{j-1} + Cc^{j-1}] - c^j, \; j = 0, 1, \ldots .$$

Then order p is obtained if the error vectors C_j vanish for $j = 0, 1, \ldots, p$. □

In this theorem powers of vectors are meant to be componentwise powers. The above order conditions are sufficient conditions but often they are not all necessary (for a discussion cf. [9]). The block point vector c plays an important role in the order conditions and by using this vector, we can achieve higher order or better stability than is possible within the class of block methods where the abscissas t_n+c_jh are equally spaced.

In the following subsections we shall give examples of, respectively, explicit, diagonally-implicit, and fully implicit methods. Based on these methods, and by means of predictor-corrector (PC) iteration, we can construct high-order explicit and diagonally-implicit BRK methods (see Section 2.2.4).

2.2.1. Explicit BRK methods

Explicit methods arise for C=O reducing the method to one-stage form:

$$\begin{array}{c|c} I & O \\ \hline A & B \end{array}\,.$$

This method needs k starting values and on k-processor computers it requires one sequential righthand side evaluation per step. Therefore, its computational complexity is comparable with that of explicit k-step linear multistep (LM) methods. For example, for k=2 we can construct the one-parameter family of third-order BRK methods (examples of higher-order methods up to order 7 can be found in [9])

$$(2.5) \quad \begin{array}{cc|cc} 1 & 0 & & \\ 0 & 1 & & \\ \hline \frac{c^2(3-c)}{(1-c)^3} & \frac{1-3c}{(1-c)^3} & \frac{c^2}{(1-c)^2} & \frac{c}{(1-c)^2} \\ \frac{5-3c}{(1-c)^3} & \frac{-c^3+3c^2-4}{(1-c)^3} & \frac{2-c}{(1-c)^2} & \frac{(2-c)^2}{(1-c)^2} \end{array}\,, \quad c = (c, 1)^T, \; c \neq 1, \; p=3, k=2, s=1.$$

For $c \leq 1-\sqrt{3}$ and $c \geq 1+\sqrt{3}$ this method is zero-stable and can be used as a method in its own right. For $c=1\pm\sqrt{6}$, this method has zero parasitic roots if h=0. Alternatively, the parameter c may be used for maximizing stability intervals (cf.[11]).

2.2.2. Diagonally-implicit BRK methods

Such methods arise when we set C=diag(d). As before, this method needs k starting values and on k-processor computers it requires the solution of one sequential, implicit relation per step. Therefore, its computational complexity is comparable with that of implicit k-step LM methods. In this case, we can construct for k=2 a three-parameter family of third-order methods. The length of the formulas prevents us from presenting them here (see [9]). In this family there are several strongly A-stable methods, of which one of them is given below:

$$
(2.6)\qquad \begin{array}{cc|cccc}
1 & 0 & 0 & 0 & & \\
0 & 1 & 0 & 0 & & \\
1/2 & 1/2 & -9/4 & -1/4 & 5/2 & 0 \\
1 & 0 & -7/4 & 9/4 & 0 & 5/2 \\
\hline
1/2 & 1/2 & -9/4 & -1/4 & 5/2 & 0 \\
1 & 0 & -7/4 & 9/4 & 0 & 5/2
\end{array}\ , \quad c = (-1,1)^T,\ p=3,\ k=s=2,\ \text{A-stable.}
$$

2.2.3. Fully implicit BRK methods

We constructed fully implicit methods for use in PC-type methods. For k=2, we found the one-parameter family of fourth-order, zero-stable methods

$$
(2.7)\qquad \begin{array}{cc|cccc}
1 & 0 & 0 & 0 & & \\
0 & 1 & 0 & 0 & & \\
0 & 1 & \frac{-c^3}{12(1-c)} & \frac{c(c^2-6c+6)}{12(1-c)} & \frac{c(c^2-6c+6)}{12(1-c)} & \frac{-c^3}{12(1-c)} \\
0 & 1 & \frac{(1-2c)}{12(1-c)(2-c)} & \frac{-6c^2+10c-3}{12c(1-c)} & \frac{3-2c}{12c(1-c)} & \frac{6c^2-14c+7}{12(1-c)(2-c)} \\
\hline
0 & 1 & \frac{-c^3}{(1-c)} & \frac{c(c^2-6c+6)}{12(1-c)} & \frac{c(c^2-6c+6)}{12(1-c)} & \frac{-c^3}{12(1-c)} \\
0 & 1 & \frac{(1-2c)}{12(1-c)(2-c)} & \frac{-6c^2+10c-3}{12c(1-c)} & \frac{3-2c}{12c(1-c)} & \frac{6c^2-14c+7}{12(1-c)(2-c)}
\end{array}\ , \quad c = (c,\ 1),\ p=4,\ k=s=2.
$$

If $c=1-\sqrt{1/5}$, then the method becomes fifth-order accurate.

2.2.4. PC-type methods

Let the predictor be of one of the two forms:

$$
(2.8)\qquad \begin{array}{c|cc}
I & O & O \\
D & E & F \\
\hline
D & E & F
\end{array}\ , \qquad\qquad \begin{array}{c|c}
I & O \\
\hline
D & E
\end{array}\ \text{with } F = O,
$$

and let the corrector be of the form

$$
(2.9)\qquad \begin{array}{c|cc}
I & O & O \\
A & B & C \\
\hline
A & B & C
\end{array}\ ,
$$

then we can construct higher-stage methods by PC iteration. In choosing a PC pair, the block point vactors c should be identical. For example, the PC pair {(2.5),(2.7)} generates the 3-stage BRK method

$$(2.10)\qquad \begin{array}{c|ccc} I & O & & \\ D & E & O & \\ A & B & C & O \\ \hline A & B & C & O \end{array},$$

which is fourth-order accurate for all values of c and requires two sequential righthand side evaluations and two starting values. The same {(2.5),(2.7)}pair generates the 4-stage, fifth-order BRK method:

$$(2.11)\qquad \begin{array}{c|cccc} I & O & & & \\ D & E & O & & \\ A & B & C & O & \\ A & B & O & C & O \\ \hline A & B & O & C & O \end{array}, \quad c=1-\sqrt{1/5}\,,$$

which requires 3 sequential righthand side evaluations.

In this way, we can construct high-order methods in a relatively straightforward manner. However, as for most block methods, the stability of the higher-order methods offers a considerable problem (see, e.g., Donelson & Hansen [4]). The highest order method we constructed so far is an explicit, stabilized, eighth-order, 3-stage method of the form (2.10) requiring four starting values and with real stability interval [-0.302, 0]. The matrices A, B, C, D and E in (2.10) are given by:

$$(2.12)\quad A := \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ \frac{5^3\cdot7^3\cdot13\cdot83}{30469\cdot2^{10}} & \frac{3^6\cdot5^3\cdot263}{30469\cdot2^7} & 0 & -\frac{3^6\cdot7^3\cdot827}{30469\cdot2^{10}} \\ \frac{4549}{30469} & \frac{3^3\cdot1039}{30469} & 0 & \frac{-3^3\cdot79}{30469} \end{pmatrix}, \quad C := \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{3^2\cdot5\cdot7\cdot809}{30469\cdot2^5} & \frac{3^3\cdot5^3\cdot7^3\cdot37}{30469\cdot2^{10}} \\ 0 & 0 & \frac{2^9\cdot11}{30469\cdot3\cdot5\cdot7} & \frac{14369}{30469} \end{pmatrix},$$

$$B := \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{3^3\cdot5^4\cdot7^3}{30469\cdot2^9} & \frac{3^6\cdot5^2\cdot7\cdot17\cdot67}{30469\cdot2^{10}} & \frac{3^5\cdot5^2\cdot7^3}{30469\cdot2^5} & \frac{3^5\cdot5\cdot7^3\cdot13}{30469\cdot2^5} \\ \frac{23029}{30469\cdot3\cdot7} & \frac{3^3\cdot13\cdot1709}{30469\cdot5\cdot7} & -\frac{2^8\cdot3^2\cdot31}{30469\cdot5\cdot7} & \frac{3^2\cdot61\cdot337}{30469\cdot5} \end{pmatrix}, \quad \mathbf{c} = (-1,\, 0,\, 5/2,\, 1)^T,$$

$$D := \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ \frac{5975}{224} & \frac{1539}{20} & -\frac{537}{35} & -\frac{2793}{32} \\ \frac{82}{343} & \frac{117}{125} & \frac{63232}{128625} & -\frac{2}{3} \end{pmatrix}, \quad E := \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{225}{32} & \frac{567}{8} & 9 & \frac{2205}{32} \\ \frac{3}{49} & \frac{18}{25} & -\frac{128}{1225} & 1 \end{pmatrix}.$$

2.2.5. Numerical examples

In Table 2.2 we compare the BRK methods of the preceding Section 2.2 with methods from the literature (see also Table 2.1). The methods are grouped according to their k-value which mainly determines the computational complexity.

Table 2.2. Values of Δ for Problem B5 from [12] at T=20.

Sequential righthand sides N	120	240	480	960	1920	p	k
Two-step Adams-Bashforth method	1.2	1.9	2.5	3.1	3.7	2	2
Two-step Adams pair: PECE	1.2	2.0	2.9	3.8	4.7	3	2
Chu-Hamilton multi-block method (1.6)	*	3.3	4.7	6.0	7.3	4	2
BRK method (2.5): $c=1-\sqrt{6}$	1.6	2.6	3.5	4.4	5.3	3	2
BRK method (2.11)	2.5	3.9	5.5	7.0	8.5	5	2
Four-step Adams-Bashforth method	3.3	3.8	4.8	6.0	7.1	4	4
Four-step Adams pair: PECE	2.5	3.4	4.8	6.2	7.7	5	4
Miranker-Liniger method (1.4)	3.1	5.0	6.3	7.2	8.3	4	4
Shampine-Watts method (1.5)	1.9	3.3	4.6	5.9	7.2	4	4
BRK method (2.12)	2.9	7.4	9.8			8	4

References

[1] **Butcher, J.C. (1987):** The numerical analysis of ordinary differential equations, Runge-Kutta and general linear methods, Wiley, New York.

[2] **Chu, M.T. & Hamilton, H. (1987):** Parallel solution of ODE's by multi-block methods, SIAM J. Sci. Stat. Comput. 3, 342-53.

[3] **Clippinger & Dimsdale (1958):** in: E. Grabbe, S. Ramo & D. Woolridge, Handbook of Automation, Computation and Control, Vol. I: Control fundamentals, Chapt. 14, p. 14-60, Wiley, New York, 1958.

[4] **Donelson, J. & Hansen, E. (1971):** Cyclic composite multistep predictor-corrector formulas, SIAM J. Numer. Anal. 8, 137-57.

[5] **Gear, C.W. (1987):** Parallel methods for ordinary differential equations, Report No. UIUCDCS-R-87-1369, University of Illinois, Urbana Champaign, Urbana.

[6] **Hairer, E., Nørsett, S.P. & Wanner, G. (1987):** Solving ordinary differential equations I. Nonstiff problems, Springer-Verlag, Berlin-Heidelberg-New York-London-Paris-Tokyo.

[7] **Houwen, P.J. van der, Sommeijer, B.P. & Mourik, P.A. van (1988):** Note on explicit parallel multistep Runge-Kutta methods, Report **NM-R8814**, Centre for Mathematics and Computer Science, Amsterdam (to appear in J. Comp. Appl. Math., 1989).

[8] **Houwen, P.J. van der & Sommeijer, B.P. (1988):** Variable step iteration of high-order Runge-Kutta methods on parallel computers, Report **NM-R8817**, Centre for Mathematics and Computer Science, Amsterdam (to appear in J. Comp. Appl. Math.).

[9] **Houwen, P.J. van der & Sommeijer, B.P. (1989):** Block Runge-Kutta methods on parallel computers, Report **NM-R89..**, Centre for Mathematics and Computer Science, Amsterdam (submitted for publication).

[10] **Houwen, P.J. van der, Sommeijer, B.P. & Couzy, W. (1989):** Embedded diagonally-implicit Runge-Kutta algorithms on parallel computers (in preparation).

[11] **Houwen, P.J. van der, Sommeijer, B.P. & Couzy, W. (1989):** Stability of parallel block Runge-Kutta methods (in preparation).

[12] **Hull, T.E., Enright, W.H., Fellen, B.M. & Sedgwick, A.E. (1972):** Comparing numerical methods for ordinary differential equations, SIAM J. Numer. Anal. 9, 603-637.

[13] **Iserles, A. & Nørsett, S.P. (1988):** On the theory of parallel Runge-Kutta methods, Report DAMTP 1988/NA12, University of Cambridge.

[14] **Jackson, K. & Nørsett, S.P. (1988):** Parallel Runge-Kutta methods, to appear.

[15] **Lie, I. (1987):** Some aspects of parallel Runge-Kutta methods, Report No. 3/87, University of Trondheim, Division Numerical Mathematics.

[16] **Miranker, W.L. & Liniger, W. (1967):** Parallel methods for the numerical integration of ordinary differential equations, Math. Comput. 21, 303-20.

[17] **Nørsett, S.P. & Simonsen, H.H. (1989)**: Aspects of parallel Runge-Kutta methods, in: A. Bellen (ed.): Workshop on Numerical Methods for Ordinary Differential Equations, L'Aquila, 1987, Lecture Notes in Mathematics, Springer-Verlag.

[18] **Prince, P.J. & Dormand, J.R. (1981)**: High-order embedded Runge-Kutta formulae, J. Comp. Appl. Math. 7, 67-75.

[19] **Shampine, L.F. & Watts, H.A. (1969)**: Block implicit one-step methods, Math. Comput. **23**, 731-40.

[20] **Wolfbrandt, A. (1977)**: Thesis, Chalmers Institute of Technology, Göteborg.

[21] **Worland, P.B. (1976)**: Parallel methods for the numerical solution of ordinary differential equations, Trans. Comput. C-**25**, 1045-48.