Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

P.J. van der Houwen, B.P. Sommeijer

Variable step iteration of high-order Runge-Kutta methods on parallel computers

---

Department of Numerical Mathematics Report NM-R8817 November

The Centre for Mathematics and Computer Science is a research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a nonprofit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O.).

Variable Step Iteration of High-Order Runge-Kutta Methods on Parallel Computers

P.J. van der Houwen & B.P. Sommeijer
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

This paper investigates iterated Runge-Kutta methods of high order designed in such a way that the righthand side evaluations can be computed in parallel. Using stepsize control based on embedded formulas a highly efficient code is developed. On parallel computers, the 8th-order mode of this code is more efficient than the DOPRI8 implementation of the formulas of Prince and Dormand. The 10th-order mode is about twice as cheap for comparable accuracies.

1980 Mathematics Subject Classification: 65M10, 65M20
Key Words & Phrases: numerical analysis, Runge-Kutta methods, parallelism.

1. INTRODUCTION

In predictor-corrector (PC) methods for solving the initial-value problem for the system of ordinary differential equations (ODEs)

$$\frac{dy(t)}{dt} = f(y(t)),$$

implicit Runge-Kutta (RK) methods are seldom used as corrector equation, because RK correctors are much more expensive than linear multistep (LM) correctors. This is due to the increased number of implicit relations to be solved when using RK correctors. Although RK correctors of order p usually possess smaller error constants than LM correctors of order p, an accuracy-computational effort graph will be in favour of PC methods based on LM methods. However, on parallel computers things are different. It is well known that PC iteration, being a form of function iteration (or Jacobi iteration), allows a high degree of parallelism, because, by partitioning the system of equations in subsystems of equal computational complexity, we can assign to each processor such a subsystem for which the iteration steps can then be performed in parallel. The problem is of course the partitioning in subsystems of equal computational complexity. In the case of iterating s-stage RK methods, there is a rather natural partitioning based on the s subsystems corresponding to the s stages of the RK method. In this way, the computation time involved in applying RK correctors can be reduced a great deal on parallel computers. We shall call these 'parallel, iterated' RK methods *PIRK methods*. Such methods have been studied in [9], [10], [11] and [12].

If the predictor is itself an (explicit) RK method, then the PIRK method also belongs to the class of explicit RK methods. In Iserles and Nørsett [9] it was proved that explicit RK methods of order p necessarily require at least p *effective* stages, and in Nørsett and Simonsen [12] the question was

posed whether it is always possible to find explicit RK methods of order p using not more than p effective stages, assuming that sufficiently many processors are available (an explicit RK method is said to have p *effective* stages if the computation time required for evaluating all righthand sides in one step is p times the computation time required by one righthand side evaluation). This question motivated us to look in the class of PIRK methods for 'optimal' RK methods (we shall call an explicit RK method *optimal* if the number of effective stages equals its order). We will show that PIRK methods generated by any (not necessarily implicit) s-stage RK corrector of order p do not require more than p effective stages provided that s processors are available. The next question then is how many processors are at least needed to implement an optimal explicit RK method? In [12] an example of a 5th-order, 6-stage RK method of Butcher is mentioned which can be implemented on two processors requiring only 5 effective stages. It is well known that, within the class of RK methods, those of Gauss-Legendre type require a minimal number of stages to obtain a certain order; therefore, in this paper, we concentrate on PIRK methods based on Gauss-Legendre correctors possessing order p=2s for s stages. Hence, for an 'optimal' implementation of these methods, we need only s processors. Furthermore, they allow an extremely simple implementation, the stability regions can directly be derived from known results for Padé polynomials, and we obtain automatically a sequence of embedded methods of varying order which can be used for stepsize control. PIRK codes of order 8 and 10 using automatic stepsize control are compared with the code DOPRI8 of Hairer, Nørsett and Wanner [5] which is a variable step implementation of the 8th-order explicit RK formula with 7th-order embedded formula of Prince and Dormand [13]. All codes use the same stepsize strategy. By a number of experiments, the performance of PIRK codes is demonstrated. Both codes are considerably cheaper than DOPRI8 for comparable accuracies. In the Appendix to this paper, we provide a FORTRAN implementation of the PIRK methods. This implementation has the facility that the user can introduce arbitrary RK correctors by means of their Butcher arrays.

Instead of using (one-step explicit) RK predictors one may use LM predictors which reduces the number of effective stages. This approach was followed by Lie [11], who reports on results obtained by the fourth-order, two-stage Gauss-Legendre corrector and a third-order Hermite extrapolation predictor. With this PC pair, one iteration suffices to obtain a fourth-order PIRK scheme. We shall shortly discuss the use of multistep predictors, in particular for RK correctors of general (nonquadrature) type. Various predictor methods are compared showing that the efficiency of PIRK methods using multistep predictors is higher, but the price to be paid for the increased efficiency is more storage and a less easy implementation.

2. Optimal RK Methods

Our starting point is the s-stage, implicit, one-step RK method of the form

$$y_{n+1} = y_n + hb^T r_{n+1}, \tag{2.1a}$$

where $\mathbf{r}_{n+1}$ is implicitly defined by

(2.1b) $\mathbf{r}_{n+1} := f(y_n\mathbf{e} + hA\mathbf{r}_{n+1})$.

Here, h is the integration step, $\mathbf{e}$ is a column vector of dimension s with unit entries, $\mathbf{b}$ is an s-dimensional vector and A is an s-by-s matrix. Furthermore, we use the convention that for any given vector $\mathbf{v}=(v_j)$, $f(\mathbf{v})$ denotes the vector with entries $f(v_j)$. By iterating the equation for $\mathbf{r}_{n+1}$ m times by simple function iteration and using the mth iterate as an approximation to $\mathbf{r}_{n+1}$, we obtain the method

(2.2) $\mathbf{r}^{(j)} = f(y_n\mathbf{e} + hA\mathbf{r}^{(j-1)}), \quad j = 1, \ldots, m; \quad y_{n+1} = y_n + h\mathbf{b}^T\mathbf{r}^{(m)}$.

Since the s components of the vectors $\mathbf{r}^{(j)}$ can be computed in parallel, provided that s processors are available, we obtain a method which requires per integration step the computational time needed for computing the initial approximation $\mathbf{r}^{(0)}$ and m righthand side evaluations. In the following, we always assume that we have s processors at our disposal and we shall speak about computational effort per step when we mean the computational time required per step if s processors are available. If the computational effort per step equals the computation time for performing M righthand side evaluations, then we shall say that the method requires M *effective stages*.

We shall call the method providing $\mathbf{r}^{(0)}$ the *predictor method* and (2.1) the *corrector method* and the resulting parallel, iterated RK method will be briefly called *PIRK method*. It should be observed that in the present case of RK correctors, the predictor and corrector methods do not directly generate approximations to y_{n+1} as is the case in PC methods based on LM methods. However, at any stage of the iteration process we can compute the current approximation to y_{n+1} by means of the formula

(2.3) $y^{(j)} := y_n + h\mathbf{b}^T\mathbf{r}^{(j)}, \quad j = 0, 1, \ldots$.

Let $\mathbf{r}^{(0)}$ be an approximation to $\mathbf{r}_{n+1}$ of order q, i.e.,

(2.4) $\mathbf{r}^{(0)} = \mathbf{r}_{n+1} + O(h^q)$.

Predictor methods satisfying (2.4) will be called predictor methods of order q.

Suppose that A and $\mathbf{b}^T$ are such that the corrector (2.1) is of order p and let the predictor method be of order q. Then, it was shown in Jackson and Nørsett [10] that the (global) order of y_{n+1} as defined by (2.2) equals $p^* := \min\{p, q+m\}$. By using the simple predictor method $\mathbf{r}^{(0)} := f(y_n)\mathbf{e} = \mathbf{r}_{n+1} + O(h)$, i.e., q=1, we immediately have as a corollary of this result the theorem:

Theorem 2.1. *Let* $\{A,\mathbf{b}^T\}$ *define a not necessarily implicit* s*-stage* RK *method of order* p. *Then the PIRK method defined by*

$$
\begin{aligned}
\mathbf{r}^{(0)} &= f(y_n)\mathbf{e}, \\
\mathbf{r}^{(j)} &= f(y_n\mathbf{e} + hA\mathbf{r}^{(j-1)}),\ j = 1, \ldots, m, \\
y_{n+1} &= y_n + h\mathbf{b}^T\mathbf{r}^{(m)}
\end{aligned}
\tag{2.5}
$$

represents an (m+1)s*-stage explicit RK method of order* $p^*:=\min\{p,m+1\}$ *requiring* m+1 *effective stages.* []

Method (2.5) can also be represented by its Butcher array. Defing the s-dimensional vector $\mathbf{0}$ and the s-by-s matrix O both with zero entries, we obtain

$$
\begin{array}{c|cccccc}
O & & & & & & \\
A & O & & & & & \\
O & A & O & & & & \\
\vdots & & \ddots & \ddots & & & \\
O & \cdots & & O & A & O \\
\hline
\mathbf{0}^T & \cdots & & \mathbf{0}^T & \mathbf{0}^T & \mathbf{b}^T
\end{array}
\,.
$$

Setting m=p-1, it follows from this theorem that the question posed by Nørsett and Simonsen [12] can be answered positively: any pth-order RK method $\{A,\mathbf{b}^T\}$ generates an explicit RK method of the form (2.5) of order p requiring only p effective stages. Such explicit RK methods will be called *optimal RK methods*. Of course, within the class (2.5) the number of processors needed for the implementation is dictated by the number of stages s of the generating corrector. For example, the 10th-order, 17-stage RK method of Hairer [4] generates an explicit RK method of the form (2.5) which is also of order 10 if we set m=9 and which is optimal in the above sense. However, the implementation of this method requires 17 processors. This suggests the problem of constructing RK methods of order p which are optimal and require a minimal number of processors. The 5th-order, 6-stage RK method of Butcher mentioned in [12] is an example of an optimal RK method requiring a minimal number of processors: this method can be implemented on two processors requiring only 5 effective stages. From the theory for RK methods based on high-order quadrature methods we can immediately deduce a lower bound for the number of processors needed to implement optimal RK methods of the form (2.5):

Theorem 2.2. *RK methods of the form* (2.5) *are optimal if* m=p-1. *For* p *even the minimal number of required processors equals* p/2 *and the generating RK corrector is the* p*th-order Gauss-Legendre method; for* p *odd the minimal number of processors is* (p+1)/2 *and the generating RK corrector is the* p*th-order Radau method.* []

Thus, optimal RK methods requiring less than [(p+1)/2] processors cannot be found among the methods of the form (2.5) (here, [.] denotes the integer part function). Since (2.5) allows an extremely simple implementation and provides automatically a sequence of embedded formulas which can be used for error estimation (see Section 5) and order variation, we have not looked for methods requiring less than [(p+1)/2] processors.

In order to appreciate Theorem 2.2, we make a comparison with explicit RK methods deviced for one-processor computers (sequential methods). In Table 2.1 the minimal number of stages s_{min} (and therefore the minimal number of righthand side evaluations) needed to generate such methods of order p are listed. In addition, we list the number of stages S for which these RK methods have actually been constructed (cf. Butcher [1]), and the numbers of effective stages S_{eff} and S_{pr} processors needed by the optimal RK methods of Theorem 2.2.

Table 2.1. Comparison of sequential RK methods and optimal RK methods of the form (2.5)

	p	=	≤4	5	6	7	8	9	10
Sequential RK	s_{min}	=	p	6	7	9	11	≥12	≥13
	S	=	p	6	7	9	11	-	17
Optimal RK	S_{eff}	=	p	5	6	7	8	9	10
	S_{pr}	=		3	3	4	4	5	5

Finally, we remark that if the RK corrector is based on quadrature (or collocation) methods, then the initial approximation $\mathbf{r}^{(0)}$ can be interpreted as the derivative $f(\mathbf{Y}^{(0)})$, where $\mathbf{Y}^{(0)}$ is an approximation to $y(t_n\mathbf{e}+h\mathbf{A}\mathbf{e})$. Suppose that the components of $\mathbf{Y}^{(0)}$ are computed (in parallel) by using an explicit (q-1)-stage RK method of order q-1 with stepsizes hAe. The resulting PIRK method is still an explicit RK method itself and it is optimal if m=p-q corrections are performed.

3. Multistep Predictor Methods

Evidently, we can save computing time by using multistep predictor methods. As observed above, such predictors should provide approximations to the derivative values $f(y(t_n\mathbf{e}+h\mathbf{A}\mathbf{e}))$ in the case where the generating RK method $\{\mathbf{A},\mathbf{b}^T\}$ is derived from quadrature formulas. Any set of linear multistep methods providing approximations to the components of $y(t_n\mathbf{e}+h\mathbf{A}\mathbf{e})$ serves this purpose.

In this paper we shortly discuss the case of arbitrary RK correctors where we cannot give an easy interpretation for the initial approximation $\mathbf{r}^{(0)}$. In such cases, it is possible to construct multistep predictor methods by performing the auxiliary vector recursion:

(3.1a) $\mathbf{f}_{n+1} := f(y_n\mathbf{e} + h\delta(E)E^{-k+1}\mathbf{f}_n)$.

The predictor method is now simply defined by

(3.1b) $\mathbf{r}^{(0)} := \mathbf{f}_{n+1}$.

Here $\delta(\zeta)$ is a polynomial of degree k-1 whose coefficients are matrices of appropriate dimension (cf. [7]). The method defined by (2.2) and (3.1) defines a k-step PC method requiring m+1 righthand side evaluations per step. For m=0, this method fits into the class of methods investigated in [7].

By Taylor expansion of $\mathbf{f}_{n+1}$ (or $\mathbf{Y}^{(0)}$), conditions in terms of A and $\delta(\zeta)$ can be derived for achieving that $\mathbf{r}_{n+1}-\mathbf{f}_{n+1}=O(h^q)$ is satisfied. For instance we have

Theorem 3.1. *Let the corrector defined by* $\{A,\mathbf{b}^T\}$ *be of order* p, *then the* k*-step PC method*

$$
\begin{aligned}
&\mathbf{f}_{n+1} = f(y_n\mathbf{e} + h\delta(E)E^{-k+1}\mathbf{f}_n),\\
(3.2)\quad &\mathbf{r}^{(0)} = \mathbf{f}_{n+1},\quad \mathbf{r}^{(j)} = f(y_n\mathbf{e} + hA\mathbf{r}^{(j-1)}),\quad j = 1, \dots, m,\\
&y_{n+1} = y_n + h\mathbf{b}^T\mathbf{r}^{(m)}
\end{aligned}
$$

is of order $p^*:=\min\{p,q+m\}$, *where*

$$
\begin{aligned}
&q = 2 \quad \textit{if} && A\mathbf{e} - \delta(1)\mathbf{e} = \mathbf{0}.\\
&q = 3 \quad \textit{if, in addition,} && A^2\mathbf{e} - \delta^2(1)\mathbf{e} + k\delta(1)\mathbf{e} - \delta'(1)\mathbf{e} = \mathbf{0},\\
& && \tfrac{1}{2}A^2\mathbf{e} - \tfrac{1}{2}\delta^2(1)\mathbf{e} + k\delta(1)\mathbf{e} - \delta'(1)\mathbf{e} = \mathbf{0}. \ \square
\end{aligned}
$$

Examples 3.1. The most simple example is the case where k=1 and $\delta(\zeta)=0$, so that $\mathbf{r}^{(0)}=f(y_n)\mathbf{e}$ and q=1. This case has already been considered in the preceding section. Next we choose k=1 and $\delta(\zeta)=A$. It is readily verified that the order conditions for the predictor are satisfied for q=2. The algorithm (3.2) assumes the one-step form

$$
\begin{aligned}
&\mathbf{f}_{n+1} = f(y_n\mathbf{e} + hA\mathbf{f}_n),\\
(3.3)\quad &\mathbf{r}^{(0)} = \mathbf{f}_{n+1},\quad \mathbf{r}^{(j)} = f(y_n\mathbf{e} + hA\mathbf{r}^{(j-1)}),\quad j = 1, \dots, m,\\
&y_{n+1} = y_n + h\mathbf{b}^T\mathbf{r}^{(m)}.
\end{aligned}
$$

If the RK corrector has order p, then by performing m=p-2 corrections this method is also of order p and requires p-1 righthand side evaluations per step. Formally, the method no longer belongs to the

class of one-step RK methods. However, in actual applications, the method is selfstarting if we take $\mathbf{f}_0=f(y_0)\mathbf{e}$.

Finally, we choose k=2 and $\delta(\zeta)=2A\zeta-A$ which satisfy the order conditions for q=3. The algorithm (3.2) assumes the two-step form

$$
\begin{aligned}
\mathbf{f}_{n+1} &= f(y_n\mathbf{e} + 2hA\mathbf{f}_n - hA\mathbf{f}_{n-1}),\\
\mathbf{r}^{(0)} &= \mathbf{f}_{n+1},\quad \mathbf{r}^{(j)} = f(y_n\mathbf{e} + hA\mathbf{r}^{(j-1)}),\quad j = 1, \dots, m,\\
y_{n+1} &= y_n + h\mathbf{b}^T\mathbf{r}^{(m)}.
\end{aligned}
\tag{3.4}
$$

If the RK corrector has order p, then by performing m=p-3 corrections this method is also of order p and requires p-2 righthand side evaluations per step. □

4. STABILITY

We consider linear stability with respect to the test equation

$$y'(t) = \lambda y(t). \tag{4.1}$$

It is easily verified that application of (2.5) yields the recursion

$$y_{n+1} = [1 + z\mathbf{b}^T\mathbf{e} + z^2\mathbf{b}^TA\mathbf{e} + z^3\mathbf{b}^TA^2\mathbf{e} + \dots + z^{m+1}\mathbf{b}^TA^m\mathbf{e}]y_n, \tag{4.2}$$

where we have written $z=\lambda h$. The stability polynomial is given by

$$P_{m+1}(z) = 1 + z\mathbf{b}^T\mathbf{e} + z^2\mathbf{b}^TA\mathbf{e} + z^3\mathbf{b}^TA^2\mathbf{e} + \dots + z^{m+1}\mathbf{b}^TA^m\mathbf{e}. \tag{4.3}$$

In the particular case where we choose m=p-1, p being the order of the corrector, we obtain a stability polynomial of degree p. According to Theorem 2.1, this PIRK method is of order p so that the stability polynomial is consistent of order p, i.e., it approximates exp(z) with pth-order accuracy. Thus, we have proved

Theorem 4.1. *Let the corrector be of order* p. *If* m=p-1, *then the method* (2.5) *becomes an (explicit) RK method with stability polynomial*

$$P_p(z) = 1 + z + \frac{1}{2!}z^2 + \frac{1}{3!}z^3 + \dots + \frac{1}{p!}z^p. \quad \square$$

Using a result on Padé polynomials (cf. [6, p.236]), we have as a corollary of this theorem:

Corollary 4.1. *The method of Theorem* 4.1 *possesses the real stability boundary*

$$\beta_{real} \approx 0.368\,(p+1)\,\sqrt[2(p+1)]{19(p+1)} \quad . \quad [] \tag{4.4}$$

In Table 4.1 we list the approximate values of the real and imaginary stability boundaries of the first 10 Padé polynomials.

Table 4.1. Real and imaginary stability boundaries of Padé polynomials

	m=1	m=2	m=3	m=4	m=5	m=6	m=7	m=8	m=9	m=10
True value of β_{real}	2.00	2.00	2.52	2.78	3.22	3.55	3.95	4.31	4.70	5.07
Value according to (4.4)	1.83	2.17	2.53	2.90	3.28	3.65	4.03	4.41	4.78	5.16
True value of β_{imag}	0.00	0.00	1.73	2.82	0.00	0.00	1.76	3.39	0.00	0.00

5. STEPSIZE CONTROL

In this section we will decribe a simple strategy to implement the before mentioned methods with a variable stepsize in order to keep control of the local truncation error. This strategy is the same as the one employed by Hairer, Nørsett & Wanner [5, p.167] in their code DOPRI8, in which they have implemented the 13-stage, 8th-order explicit RK method with embedded method of order 7 of Prince and Dormand.

This strategy is based on the observation that when iterating the equation (2.1b) for $\mathbf{r}_{n+1}$ we obtain approximations $\mathbf{r}^{(j)}$ of successively increasing order, i.e.,

$$\mathbf{r}^{(j)} - \mathbf{r}_{n+1} = O(h^{\min\{p,q+j\}}), \qquad j=1,2,\ldots,m.$$

Thus, apart from our final approximation $y_{n+1}:=y_n+h\mathbf{b}^T\mathbf{r}^{(m)}$, we can easily construct a reference solution (cf. (2.3))

$$y^{(k)} := y_n + h\mathbf{b}^T\mathbf{r}^{(k)}, \tag{5.1}$$

for some k<m. Since $\mathbf{r}^{(k)}$ has already been computed, this does not require additional function evaluations. This reference solution $y^{(k)}$ can be considered as an 'embedded' solution [5].
Now, as an estimate for the local error ε in the step from t_n to $t_{n+1}=t_n+h$, we take

$$\varepsilon := \|y_{n+1} - y^{(k)}\|, \tag{5.2}$$

for some norm ||.||. Usually, one uses reference solutions $y^{(k)}$ such that the orders of y_{n+1} and $y^{(k)}$ differ by 1. Here we follow this approach and choose k=m-1. Furthermore, we restrict our stepsize

strategy to methods in which the number of iterations m is fixed in each step and is given by m=p-q. Hence, $\mathbf{r}^{(m)}$ and $\mathbf{r}^{(m-1)}$ are approximations to $\mathbf{r}_{n+1}$ of orders p and p-1, respectively, and, consequently,

$$\varepsilon = \|y_{n+1} - y^{(m-1)}\| = \|y_n + h\mathbf{b}^T\mathbf{r}^{(m)} - y_n - h\mathbf{b}^T\mathbf{r}^{(m-1)}\| = O(h^p).$$

Then ε is compared with some prescribed tolerance TOL and the step is accepted if $\varepsilon \leq TOL$, and rejected otherwise. Furthermore, the value of ε allows us to make an estimate for the optimal stepsize:

$$h \sqrt[p]{\frac{TOL}{\varepsilon}},$$

which will be taken in the next step (or to recompute the current step in case of rejection). However, to give the code some robustness, we actually implemented (cf. [5, p.167])

$$(5.3) \qquad h_{new} = h \cdot \min\left\{6, \max\left\{\tfrac{1}{3}, \; 0.9 \sqrt[p]{\frac{TOL}{\varepsilon}}\right\}\right\}.$$

The constants 6 and $\frac{1}{3}$ in this expression serve to prevent a too drastic change in the stepsize and the safety factor 0.9 is added to increase the probability that the next step will be accepted.

It is possible to apply a more sophisticated strategy in which also the number of iterations m may vary from step to step. Similar as described above, we can construct a *sequence* of reference solutions, i.e., after each iteration the 'embedded' solution

$$y^{(j)} := y_n + h\mathbf{b}^T\mathbf{r}^{(j)}$$

is computed. Then, we can use the difference of two successive reference solutions as an estimate for the local error, i.e.,

$$\varepsilon^{(j)} := \| y^{(j)} - y^{(j-1)} \|.$$

If, during the iteration, the tolerance criterium $\varepsilon^{(j)} \leq TOL$ is satisfied for some $j=j_0<m$, then there is no need to proceed the iteration process and we will accept $y^{(j_0)}$ as the numerical solution y_{n+1}. This suggests to try the next step with the value of m defined by $m=j_0$. Since

$$\varepsilon^{(j_0)} = O(h^{p^*}), \qquad p^* = \min\{p+1, q + j_0\},$$

a prediction for the next stepsize can be made according to (5.3), where p is replaced by p* and ε by $\varepsilon^{(j_0)}$.

It may happen that the tolerance condition is *not* satisfied for $j=j_0 \le m$. In such cases, the values of m and h predicted in the preceding step were not reliable. One may then decide to reject the current value of m and to continue the iteration process. This is particularly recommendable if the value of the current p* is less than p. If the continuation of the iteration process does not help to satisfy the tolerance condition $\varepsilon^{(j)} \le TOL$ for $j \le M$, where M is some prescribed upper bound for the number of iterations per step, then the (relatively costly) alternative is rejection of the current value of h, to redefine h according to (5.3) using the most recent information on the error, and to perform the present step once again. In this way a variable order variable stepsize RK method can be constructed.

6. NUMERICAL EXPERIMENTS

We present a few examples illustrating the efficiency of PIRK methods on parallel computers. The methods tested were all applied in $P(EC)^mE$ mode.

6.1. Comparison of various predictor methods

In order to see the effect of the various predictor methods on the efficiency of the PIRK method we performed a few tests by integrating the equation of motion for a rigid body without external forces (cf. Problem B5 from [8]):

$$
\begin{aligned}
&y_1' = y_2 y_3, && y_1(0) = 0, \\
&y_2' = -y_1 y_3, && y_2(0) = 1, \quad 0 \le t \le T. \\
&y_3' = -.51 y_1 y_2, && y_3(0) = 1,
\end{aligned}
\tag{6.1}
$$

In these tests we used the 10th-order Gauss-Legendre corrector and the following predictor methods:

Predictor I:	$r^{(0)} = f(y_n)e$ (cf. (2.5))	q=1	p=min{m+1,10}
Predictor II:	$r^{(0)}$ defined by standard 4th-order RK	q=5	p=min{m+5,10}
Predictor III:	$r^{(0)} = f(y_n e + hAf_n)$ (cf. (3.3))	q=2	p=min{m+2,10}
Predictor IV:	$r^{(0)} = f(y_n e + 2hAf_n - hAf_{n-1})$ (cf. (3.4))	q=3	p=min{m+3,10}

In Table 6.1 we have listed the values D\N, where D 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^{-D}, and where N denotes the total number of effective righthand side evaluations performed during the integration process. Furthermore, we indicated the effective order p_{eff}, that is the order of accuracy which is shown numerically.

Table 6.1. Values D\N for problem (6.1) with T=20.

	Predictor I			Predictor II			Predictor III			Predictor IV	
h^{-1}	m=8	m=9	m=10	m=4	m=5	m=6	m=7	m=8	m=9	m=7	m=8
1	5.6\180	6.5\200	6.9\220	5.3\180	7.0\200	6.8\220	4.8\160	5.5\180	7.5\200	4.6\160	5.7\180
2	8.0\360	9.7\400	9.8\440	7.8\360	10.2\400	9.7\440	7.2\320	8.5\360	9.6\400	7.2\320	8.8\360
4	10.6\720	13.0\800	12.3\880	10.5\720	13.3\800	12.2\880	9.7\640	11.6\720	12.1\800	10.4\640	12.4\720
$p_{eff} \approx 9$		10	10	9	10	10	9	10	10	10	10

Comparing experiments with equal N (notice that this table contains for each h and each predictor an experiment with $N=180h^{-1}$) we conclude that in most experiments the third-order predictor IV and the second-order predictor III yield the most accurate values. However, the price we pay is more storage and a more complicated implementation because of the auxiliary recursion for $\mathbf{f}_n$. The predictors I and II produce comparable accuracies. Therefore, we recommend predictor I in actual computations. The resulting PIRK method is a true one-step RK method of an extremely simple structure, and consequently allowing an easy and straightforward implementation.

6.2. Comparison with the 10th-order methods of Curtis and Hairer

Curtis [2] and Hairer [4] used the test problem (6.1) for testing and comparing their 10th-order RK methods. In Table 6.2 the results of the experiments performed by Curtis and Hairer are reproduced together with results obtained by the PC pairs consisting of the predictors I, II and III, and the 10th-order Gauss-Legendre corrector. Again we see that the simple predictor I can compete with the predictors II and III.

Table 6.2. Values D\N for problem (6.1) with T=60.

Method	p	60/h	D	N
Runge-Kutta	4	12000	9.6	48000
Adams-Moulton-Bashforth	4	6000	8.1	12000
Bulirsch-Stoer: polynomial extrapolation	-	-	8.9	5276
Bulirsch-Stoer: rational extrapolation	-	-	9.6	4860
Runge-Kutta-Curtis	10	240	9.9	4320
Runge-Kutta-Hairer	10	240	10.1	4080
(2.2) with predictor I and m=9	10	156	10.0	1560
(2.2) with predictor I and m=10	10	150	10.0	1650
(2.2) with predictor II and m=5	10	150	10.1	1500
(2.2) with predictor II and m=6	10	156	10.1	1716
(2.2) with predictor III and m=8	10	210	10.0	1891
(2.2) with predictor III and m=9	10	168	10.0	1681

6.3. Comparison with the 8(7) method of Prince and Dormand

The 8(7)-method of Prince and Dormand [13] 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}. In this subsection we compare the DOPRI8 code, as given by Hairer, Nørsett and Wanner [5], with the PIRK method based on predictor I and 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 equipped the PIRK codes with the same strategy (see Section 5). These codes are respectively denoted by PIRK8 and PIRK10.

6.3.1. Fehlberg problem

As a first test problem we take an example from Fehlberg [3]:

$$\begin{aligned} y_1' &= 2\,t\,y_1 \log(\max\{y_2, 10^{-3}\}), \quad y_1(0) = 1, \\ y_2' &= -2\,t\,y_2 \log(\max\{y_1, 10^{-3}\}), \quad y_2(0) = e, \end{aligned} \qquad 0 \le t \le 5, \tag{6.2}$$

with exact solution $y_1(t)=\exp(\sin(t^2))$, $y_2(t)=\exp(\cos(t^2))$. For tolerances TOL running from 10^{-5} up to 10^{-12} we computed the D and corresponding $\log_{10}(N)$ values. Instead of presenting the polygon graph for these values as was done in [5], we preferred to present the D\N lying on this polygon for a number of integer values of D. In Table 6.3 these values are listed.

Table 6.3. Values of N for problem (6.2).

Method	D=5	D=6	D=7	D=8	D=9	D=10	D=11
DOPRI8	595	759	963	1227	1574	1990	2503
PIRK8	379	495	623	786	978	1383	1874
PIRK10	327	388	490	704	884	977	1078

6.3.2. Euler equations

Next, we apply the codes to Euler's equation for a rigid body (cf. (6.1)). The performance of the code is presented in Table 6.4.

Table 6.4. Values of N for problem (6.1).

Method	D=6	D=7	D=8	D=9	D=10	D=11	D=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