Applied Numerical Mathematics 13 (1993) 221–240
North-Holland

APNUM 439

Explicit, high-order Runge–Kutta–Nyström methods for parallel computers

B.P. Sommeijer
Centre for Mathematics and Computer Science, P.O. Box 4079, 1009 AB Amsterdam, Netherlands

Received 22 September 1992
Accepted 24 September 1992

Abstract

Sommeijer, B.P., Explicit, high-order Runge–Kutta–Nyström methods for parallel computers, Applied Numerical Mathematics 13 (1993) 221–240.

The paper describes the construction of explicit Runge–Kutta–Nyström (RKN) methods of arbitrarily high order. The order is borrowed from an underlying implicit RKN method. For the approximate solution of this method, an iteration scheme is defined. Prescribing a fixed number of iterations, the resulting scheme is an explicit RKN method. The iteration scheme is defined in such a way that many of the right-hand side evaluations can be done concurrently. As a result, explicit RKN schemes of order p are obtained which require, on a parallel computer, approximately $p/2$ right-hand side evaluations per step. Both in fixed- and variable-step mode, the schemes are compared with existing (sequential) high-order RKN methods from the literature and are shown to demonstrate superior behaviour.

Keywords. Explicit Runge–Kutta–Nyström methods; predictor–corrector methods; parallelism; error control.

1. Introduction

Consider the initial-value problem for the system of special second-order, ordinary differential equations (ODEs)

$$y''(t)=f(t, y(t)), \qquad y(t_0)=y_0, \qquad y'(t_0)=y_0', \tag{1.1}$$

where $y:\mathbb{R}\to\mathbb{R}^N$, $f:\mathbb{R}\times\mathbb{R}^N\to\mathbb{R}^N$, and $t_0\leqslant t\leqslant t_{\text{end}}$. In this paper, the ODEs are assumed to be *nonstiff*. Problems of this kind are encountered in e.g. celestial mechanics. Quite often, the solution of (1.1) is required with high precision; for that purpose integration methods of high order are most efficient.

A possible approach is to convert (1.1) into a (double dimensioned) system of first-order ODEs and to apply a (high-order) Runge–Kutta (RK) method. However, since the right-hand side function f in (1.1) does not depend on y', the use of a direct method is usually more efficient. An example of such a direct method is the (one-step) Runge–Kutta–Nyström (RKN) method.

Correspondence to: B.P. Sommeyer, Centre for Mathematics, P.O. Box 4079, 1009 AB Amsterdam, Netherlands.

In the literature, several high-order RKN methods have been proposed. Methods up to order 8 can be found in [1,2,6,11] and in [13] a method of order 10 is presented. A common challenge in all these papers is to reduce, for a given order, the required number of f-evaluations per step. To illustrate that the use of an RKN method for the special problem (1.1) is cheaper than an RK method for the corresponding first-order ODE, we mention that the RKN method of order 10 constructed in [13] requires eleven f-evaluations per step, whereas all known RK methods of the same order need (at least) seventeen f-evaluations. Within the class of RKN methods with minimal number of f-evaluations, we are not aware of methods of order higher than 10.

The present paper is concerned with the construction of explicit RKN methods of *arbitrarily* high order for which the *effective* number of f-evaluations (or, *effective stages*) per step is approximately equal to half its order. Here, we have to define what we mean by effective stages: Let T be the CPU time (on a sequential computer) needed to perform one evaluation of the right-hand side function f. Then an explicit RKN method is said to require M effective stages if the total CPU time to complete all f-evaluations equals $M \cdot T$.

For the methods presented in this paper, the low number of effective stages, relative to the order, is obtained by exploiting the fact that many of the stages can be performed concurrently. Hence, on a parallel computer these RKN methods are expected to be more efficient. In [15], similar type of methods have been constructed for nonstiff *first-order* ODEs. In that paper, the effective number of stages equals the order. Hence, for second-order ODEs, the number of stages can be reduced by a factor 2 by exploiting the special form of (1.1).

In Section 2, we discuss general RKN methods and define the construction of the low-stage, high-order, parallel RKN methods. Furthermore, order results and stability characteristics are presented. Numerical comparisons with existing (sequential) high-order RKN methods are given in Section 3. Here we measure the efficiency of the various methods in terms of the number of effective stages for a fixed stepsize. However, a "realistic" implementation of an integration method requires the facility to change the stepsize, and possibly the order. Section 4 is devoted to this subject. In changing the stepsize we need some error control mechanism. Common practice nowadays in explicit RK- and RKN-based codes is to calculate a reference solution (in addition to the approximation to proceed the integration) by means of *embedding techniques*. The idea of embedding is that both approximations share as many f-values as possible to minimize the total number of required stages. Nevertheless, the number of stages in the "combined" scheme is significantly larger than in the case where the additional reference solution was not required. Following this approach, Fehlberg [6] constructed a set of RKN pairs of orders $p(p+1)$ (i.e., order p for the step continuation and order $p+1$ for the reference method), for p up to 8. Other embedded RKN methods have been derived by Dormand and Prince [5] (order 7(6), requiring 9 stages), Fehlberg, Filippi and Gräf [8] (order 10(11), requiring 17 stages), Filippi and Gräf [9] (order 11(12), requiring 20 stages), and, by the same authors [10], methods of order $p+1(p)$, where p runs from 7 to 10 and the corresponding number of stages runs from 9 to 17).

In Section 4 we discuss how the parallel RKN methods can be equipped with an embedded reference solution, without additional costs, resulting in an RKN pair of order $p(p-2)$, with approximately $p/2$ effective stages. Based on a simple stepsize strategy, we show the performance of these embedded parallel RKN methods and we make comparisons with the sequential codes form the literature. In Section 5 we briefly discuss some possibilities for further improvement of the efficiency of the parallel code and some conclusions are formulated in Section 6.

2. Runge–Kutta–Nyström methods

For the numerical integration of (1.1) the general RKN method is defined by

$$y_{n+1} = y_n + hy_n' + h^2 \sum_{i=1}^{s} b_i f(t_n + c_i h, Y_i), \tag{2.1a}$$

$$y_{n+1}' = y_n' + h \sum_{i=1}^{s} d_i f(t_n + c_i h, Y_i), \tag{2.1b}$$

$$Y_i = y_n + c_i h y_n' + h^2 \sum_{j=1}^{s} a_{ij} f(t_n + c_j h, Y_j), \quad i = 1, \dots, s, \tag{2.1c}$$

where $y_n \approx y(t_0 + nh)$, h is the (constant) stepsize, b_i, d_i, and a_{ij} are real coefficients, and s is called the number of stages. The method is conveniently represented by means of its so-called Butcher array

$$\begin{array}{c|c} c & A \\ \hline & \boldsymbol{b}^{\mathrm{T}} \\ & \boldsymbol{d}^{\mathrm{T}} \end{array} \tag{2.2}$$

where $A = (a_{ij})$, $\boldsymbol{b} = (b_1, \dots, b_s)^{\mathrm{T}}$, $\boldsymbol{c} = (c_1, \dots, c_s)^{\mathrm{T}}$, and $\boldsymbol{d} = (d_1, \dots, d_s)^{\mathrm{T}}$.

Definition 2.1. Let the locally exact solution y satisfy $y(t_n) = y_n$ and $y'(t_n) = y_n'$. If

$$y_{n+1} - y(t_{n+1}) = \mathrm{O}(h^{p_1+1}), \qquad y_{n+1}' - y'(t_{n+1}) = \mathrm{O}(h^{p_2+1}),$$

then the RKN method (2.1) is said to be of order $p := \min\{p_1, p_2\}$.

2.1. Parallel, explicit RKN methods

In this subsection, we describe the construction of the parallel, explicit RKN method. For that purpose, it is convenient to introduce a succinct notation for the general RKN method (2.1).

Introducing $\boldsymbol{Y} := (Y_1^{\mathrm{T}}, \dots, Y_s^{\mathrm{T}})^{\mathrm{T}}$, $\boldsymbol{e} := (1, \dots, 1)^{\mathrm{T}}$, and

$$\boldsymbol{F}(t_n\boldsymbol{e} + \boldsymbol{c}h, \boldsymbol{Y}) := \left(f^{\mathrm{T}}(t_n + c_1 h, Y_1), \dots, f^{\mathrm{T}}(t_n + c_s h, Y_s)\right)^{\mathrm{T}},$$

the RKN method (2.1) can be written as

$$y_{n+1} = y_n + hy_n' + h^2(\boldsymbol{b}^{\mathrm{T}} \otimes I)\boldsymbol{F}(t_n\boldsymbol{e} + \boldsymbol{c}h, \boldsymbol{Y}), \tag{2.3a}$$

$$y_{n+1}' = y_n' + h(\boldsymbol{d}^{\mathrm{T}} \otimes I)\boldsymbol{F}(t_n\boldsymbol{e} + \boldsymbol{c}h, \boldsymbol{Y}), \tag{2.3b}$$

$$\boldsymbol{Y} = \boldsymbol{e} \otimes y_n + h\boldsymbol{c} \otimes y_n' + h^2(A \otimes I)\boldsymbol{F}(t_n\boldsymbol{e} + \boldsymbol{c}h, \boldsymbol{Y}), \tag{2.3c}$$

where I is the $N \times N$ identity matrix, and $\otimes$ denotes the Kronecker product. It should be observed that, for a general full A-matrix, this method is implicit, which would require the solution of a system to find Y. Since we are aiming at nonstiff problems, we will restrict our considerations to explicit methods, which correspond to a strictly lower triangular A-matrix. Consequently, we only have to do f-evaluations rather than solving a system of equations. To arrive at such an explicit RKN method, we propose the following algorithm:

Step 1. Select a fully implicit RKN method of the form (2.3) as the starting point (henceforth called the *corrector*).

Step 2. Apply, say m, functional iterations to the equation (2.3c).

Step 3. Substitute the mth iterate into (2.3a) and (2.3b) to obtain the approximation and its derivative at the next step point.

Thus we have defined the method

$$Y^{(j)} = e \otimes y_n + hc \otimes y_n' + h^2(A \otimes I)F\left(t_n e + ch,\, Y^{(j-1)}\right), \quad j = 1, \ldots, m, \tag{2.4a}$$

$$\begin{aligned} y_{n+1} &= y_n + hy_n' + h^2(b^{\mathrm{T}} \otimes I)F\left(t_n e + ch,\, Y^{(m)}\right), \\ y_{n+1}' &= y_n' + h(d^{\mathrm{T}} \otimes I)F\left(t_n e + ch,\, Y^{(m)}\right). \end{aligned} \tag{2.4b}$$

To start the iteration (2.4a) we need, of course, an initial approximation $Y^{(0)}$, which will be referred to as the *predictor*. In this section we shall consider the "trivial" (one-step) predictor

$$Y^{(0)} = e \otimes y_n + hc \otimes y_n'. \tag{2.4c}$$

Although this predictor is of modest accuracy, it has the advantage that we remain in the class of *one-step* methods; multistep predictors of higher accuracy will be discussed in Section 5. Now, the method (2.4a)–(2.4c) is indeed an explicit RKN method, defined by the Butcher array

$$\begin{array}{c|cccccl} c & O & & & & & \}s \text{ stages } (j=0) \\ c & A & O & & & & \}s \text{ stages } (j=1) \\ c & O & A & O & & & \}s \text{ stages } (j=2) \\ \cdot & \cdot & & \cdot & \cdot & & \quad\cdot \\ \cdot & \cdot & & & \cdot\ \cdot & & \quad\cdot \\ \cdot & \cdot & & & \cdot & \cdot & \quad\cdot \\ c & O & \cdots & & A & O & \}s \text{ stages } (j=m) \\ \hline & \mathbf{0}^{\mathrm{T}} & \cdots & & \mathbf{0}^{\mathrm{T}} & b^{\mathrm{T}} & \\ & \mathbf{0}^{\mathrm{T}} & \cdots & & \mathbf{0}^{\mathrm{T}} & d^{\mathrm{T}} & \end{array} \tag{2.5}$$

where O and $\mathbf{0}^{\mathrm{T}}$ denote a matrix and a vector respectively (both of dimension s) with zero entries. We observe that the number of stages in (2.5) equals $(m+1)\cdot s$. However, in calculating $Y^{(j)}$, the iterate $Y^{(j-1)}$ is a *given* vector and hence the f-evaluations of its s components (i.e., the components of $F(t_n e + ch,\, Y^{(j-1)})$) can be computed in parallel, provided that we have a computer with s processors. Consequently, the *effective* computational time needed for one iteration of (2.4a) is approximately equal to the time needed to perform one f-evaluation on a sequential computer. Taking the final f-evaluation needed in (2.4b) into account, we see that the scheme (2.4) requires $m+1$ effective stages. In the sequel, these Parallel-Iterated RKN schemes will be referred to as PIRKN methods.

Let us first consider the order of accuracy of PIRKN methods. The next theorem is formulated for predictors that are more general than (2.4c) so that we can use the result also in the case of higher-order predictors (to be discussed in Section 5).

Theorem 2.2. *Let* $\{A, c, b^{\mathrm{T}}, d^{\mathrm{T}}\}$ *define a (not necessarily implicit) RKN method of order p and let the predictor be of order q, i.e.,* $Y - Y^{(0)} = \mathrm{O}(h^q)$. *Furthermore, let f be Lipschitz continuous. Then the PIRKN method* (2.4a)–(2.4b) *is of order* $p^* := \min\{p, 2m+q\}$.

Proof. Let y satisfy $y(t_n) = y_n$ and $y'(t_n) = y'_n$. Let (u_{n+1}, u'_{n+1}) denote the locally exact solution of the corrector. Then

$$\begin{aligned} y(t_{n+1}) - y_{n+1} &= y(t_{n+1}) - u_{n+1} + u_{n+1} - y_{n+1} \\ &= \mathrm{O}(h^{p_1+1}) + h^2(b^{\mathrm{T}} \otimes I)\left[F(t_n e + ch, Y) - F\left(t_n e + ch, Y^{(m)}\right)\right], \\ y'(t_{n+1}) - y'_{n+1} &= y'(t_{n+1}) - u'_{n+1} + u'_{n+1} - y'_{n+1} \\ &= \mathrm{O}(h^{p_2+1}) + h(d^{\mathrm{T}} \otimes I)\left[F(t_n e + ch, Y) - F\left(t_n e + ch, Y^{(m)}\right)\right], \end{aligned}$$

where $\min\{p_1, p_2\} = p$. Since

$$\begin{aligned} Y - Y^{(j)} &= h^2(A \otimes I)\left[F(t_n e + ch, Y) - F\left(t_n e + ch, Y^{(j-1)}\right)\right] \\ &= \left[Y - Y^{(j-1)}\right] \cdot \mathrm{O}(h^2), \end{aligned}$$

we see that we gain two orders in each iteration. Hence, using

$$Y - Y^{(0)} = \mathrm{O}(h^q),$$

the result after m iterations is

$$F(t_n e + ch, Y) - F\left(t_n e + ch, Y^{(m)}\right) = \mathrm{O}(h^{2m+q}),$$

resulting in

$$\begin{aligned} y(t_{n+1}) - y_{n+1} &= \mathrm{O}(h^{p_1+1}) + \mathrm{O}(h^{2m+q+2}), \\ y'(t_{n+1}) - y'_{n+1} &= \mathrm{O}(h^{p_2+1}) + \mathrm{O}(h^{2m+q+1}). \end{aligned}$$

By Definition 2.1, the PIRKN method is seen to be of order $\min\{p, 2m+q\}$. □

Remark 2.3. From this theorem we see (i) that we cannot exceed the order of the underlying corrector, and (ii) that after $m = \lceil (p-q)/2 \rceil$ iterations this maximum order is reached; this m-value will be denoted by m_p.

2.2. *Choice of the corrector*

To arrive at a high-order PIRKN method we need, according to Theorem 2.2, a corrector of high order. Correctors having this property are easily obtained from high-order Runge–Kutta methods for first-order ODEs in the following way [12]: Let

$$\begin{array}{c|c} c_{\mathrm{RK}} & A_{\mathrm{RK}} \\ \hline & b_{\mathrm{RK}}^{\mathrm{T}} \end{array} \tag{2.6a}$$

denote an s-stage RK method of order p. Then

$$\begin{array}{c|c} c = c_{\text{RK}} & A = (A_{\text{RK}})^2 \\ \hline & b^{\mathrm{T}} = b_{\text{RK}}^{\mathrm{T}} A_{\text{RK}} \\ & d^{\mathrm{T}} = b_{\text{RK}}^{\mathrm{T}} \end{array} \tag{2.6b}$$

defines an s-stage RKN method of the same order. Well-known RK methods that can serve for this purpose are those of Gauss–Legendre and Radau type, which are easily constructed for arbitrary values of s (cf. [3]). They possess a high order, relative to the number of stages, i.e., $p = 2s$ and $p = 2s - 1$, respectively. Owing to this property, the (high-order) RKN correctors considered in this paper will be based on Gauss–Legendre and Radau type RK methods. Observe that, in the present context of PIRKN methods, this low number of stages with respect to the order means that we need a low number of processors. Although this property is not of decisive importance, it is certainly not an unpleasant feature.

2.3. *Optimality*

Combining the results of the Sections 2.1 and 2.2, we see that, if the PIRKN method (2.4a)–(2.4b) uses (2.4c) as the predictor (i.e., $q = 2$) and an s-stage Gauss–Legendre method as the corrector (i.e., $p = 2s$), then after m_p iterations, an explicit RKN method is obtained of order $2s$ requiring only s effective stages. This is an optimal result, since it can be shown that the order of an arbitrary s-stage, explicit RKN method cannot exceed $2s$. In fact, this optimal order can only be achieved for $s = 1$; for $s \geqslant 2$, one should change to a (fully) implicit RKN method to obtain this optimal order $2s$.

2.4. *Linear stability of PIRKN methods*

The linear stability of PIRKN methods is investigated by applying these schemes to the test equation $y'' = \lambda y$, where λ is supposed to run through the (negative) eigenvalues of the Jacobian matrix $\partial f/\partial y$. Defining $v_n := (y_n, hy_n')^{\mathrm{T}}$ and $z := h^2\lambda$, it is easily verified that application of (2.4) to the test equation yields the recursion

$$v_{n+1} = M_m(z)v_n, \quad M_m(z) := \begin{pmatrix} 1 + zb^{\mathrm{T}}W_m(z)e & 1 + zb^{\mathrm{T}}W_m(z)c \\ zd^{\mathrm{T}}W_m(z)e & 1 + zd^{\mathrm{T}}W_m(z)c \end{pmatrix}, \tag{2.7}$$

where $W_m(z) = I + zA + (zA)^2 + \cdots + (zA)^m$. The matrix $M_m(z)$, which determines the stability of the method, will be called the *amplification matrix*.

Definition 2.4. The interval $[-\beta_m, 0]$ is called the *stability interval* if, in this interval, $\rho(M_m(z)) \leqslant 1$, where $\rho(\cdot)$ denotes the spectral radius; β_m is called the *stability boundary*.

The explicit PIRKN methods have, of course, a finite stability boundary, which implies that the integration step has to satisfy the condition

$$h^2 \leqslant \frac{\beta_m}{\rho(\partial f/\partial y)}. \tag{2.8}$$

Thus, if we have a stiff problem—in which case $\rho(\partial f/\partial y)$ is large—then this condition is a severe restriction on the stepsize and we should change to an (implicit) A-stable RKN method.

A straightforward way to (numerically) calculate the stability boundaries β_m would be to run along the negative z-axis and to check up to what point the spectral radius of $M_m(z)$ is bounded by 1. However, for z-values in the neighbourhood of the origin, the spectral radius is close to 1 and, due to rounding errors, it is hard to decide whether the stability interval is empty or not. Therefore, we shall describe an alternative approach which allows for an analytical calculation of the entries of M and uses a numerical verification only in the final part of the calculations. This approach will be demonstrated for $m = 1$; for larger m-values only the results will be given since the techniques are analogous.

We will confine our considerations to the correctors discussed in the preceding section, i.e., the Gauss–Legendre and Radau-based RKN methods and we shall use some properties of the generating RK methods.

Using the relations between the RKN corrector and the underlying RK method as defined in the Butcher arrays (2.6a) and (2.6b), the amplification matrix $M_m(z)$ (cf. (2.7)) can be written as

$$M_m(z) := \begin{pmatrix} 1 + z\boldsymbol{b}_{\mathrm{RK}}^{\mathrm{T}} A_{\mathrm{RK}}[I + zA_{\mathrm{RK}}^2 + \cdots + z^m A_{\mathrm{RK}}^{2m}]\boldsymbol{e} & 1 + z\boldsymbol{b}_{\mathrm{RK}}^{\mathrm{T}} A_{\mathrm{RK}}[I + zA_{\mathrm{RK}}^2 + \cdots + z^m A_{\mathrm{RK}}^{2m}]\boldsymbol{c}_{\mathrm{RK}} \\ z\boldsymbol{b}_{\mathrm{RK}}^{\mathrm{T}}[I + zA_{\mathrm{RK}}^2 + \cdots + z^m A_{\mathrm{RK}}^{2m}]\boldsymbol{e} & 1 + z\boldsymbol{b}_{\mathrm{RK}}^{\mathrm{T}}[I + zA_{\mathrm{RK}}^2 + \cdots + z^m A_{\mathrm{RK}}^{2m}]\boldsymbol{c}_{\mathrm{RK}} \end{pmatrix} \tag{2.9}$$

Now, we use the following well-known result [4]

$$\text{an RK method (with } A_{\mathrm{RK}}\boldsymbol{e} = \boldsymbol{c}_{\mathrm{RK}}) \text{ of order } p \text{ satisfies } \boldsymbol{b}_{\mathrm{RK}}^{\mathrm{T}}(A_{\mathrm{RK}})^{j-1}\boldsymbol{e} = \frac{1}{j!}, \quad j = 1, \dots, p. \tag{2.10}$$

Substitution of these relations into (2.9) drastically simplifies the matrix $M_m(z)$. Let us work this out for $m = 1$. Here, we have to distinguish three different situations for the order p of the generating RK method:

(i) $p = 3$, e.g., the two-point Radau IIA method;
(ii) $p = 4$, e.g., the two-point Gauss–Legendre method;
(iii) $p \geqslant 5$, e.g., higher-point Radau IIA or Gauss–Legendre methods.

Case (i). For a third-order RK method we need the values of $\boldsymbol{b}_{\mathrm{RK}}^{\mathrm{T}} A_{\mathrm{RK}}^2 \boldsymbol{c}_{\mathrm{RK}}$ and $\boldsymbol{b}_{\mathrm{RK}}^{\mathrm{T}} A_{\mathrm{RK}}^3 \boldsymbol{c}_{\mathrm{RK}}$ since they are not covered by property (2.10). For the Radau IIA method, these values are easily found to be $1/36$ and $-1/108$, respectively. Hence, we find

$$M_1(z) := \begin{pmatrix} 1 + \frac{1}{2}z + \frac{1}{36}z^2 & 1 + \frac{1}{6}z - \frac{1}{108}z^2 \\ z + \frac{1}{6}z^2 & 1 + \frac{1}{2}z + \frac{1}{36}z^2 \end{pmatrix},$$

the eigenvalues of which are the zeros of $\zeta^2 - T(z)\zeta + D(z)$, with $T(z) := \mathrm{Trace}(M_1(z))$ and $D(z) := \mathrm{Det}(M_1(z))$. According to the Hurwitz criteria, these zeros are on the unit disc if

$$\begin{aligned} R_1(z) &:= 1 - D(z) \geqslant 0, \\ R_2(z) &:= D(z) + 1 - T(z) \geqslant 0, \\ R_3(z) &:= D(z) + 1 + T(z) \geqslant 0. \end{aligned} \tag{2.11}$$

Calculating the zeros of the polynomials $R_j(z)$, it is easily deduced that $R_j(z) \geqslant 0$, $j = 1, 2, 3$, if $z \in [-4.94067\ldots, 0]$, yielding $\beta_1 \approx 4.94$.

Case (ii). In this case the only value not yet defined by (2.10) is $\boldsymbol{b}_{\text{RK}}^{\text{T}} A_{\text{RK}}^3 \boldsymbol{c}_{\text{RK}}$; for the two-point Gauss–Legendre method, we find the value 1/144. Hence, in this case we have the amplification matrix

$$M_1(z) := \begin{pmatrix} 1 + \frac{1}{2}z + \frac{1}{24}z^2 & 1 + \frac{1}{6}z + \frac{1}{144}z^2 \\ z + \frac{1}{6}z^2 & 1 + \frac{1}{2}z + \frac{1}{24}z^2 \end{pmatrix}.$$

We find that the polynomials $R_j(z)$, $j = 1, 2, 3$, as defined in (2.11) are nonnegative if $z \in [-12, 0]$.

Case (iii). For $p \geqslant 5$, all terms in the amplification matrix are determined by (2.10) and hence this matrix is independent of the particular choice of the corrector. We have stability if $z \in [-7.06782\ldots, 0]$.

Summarizing the above results, we have:

Theorem 2.5. *For* $m = 1$, *the PIRKN method* (2.4a)–(2.4c) *based on*

(i) *the two-point Radau IIA method is of order* 3 *and has* $\beta_1 = 4.94$,
(ii) *the two-point Gauss–Legendre method is of order* 4 *and has* $\beta_1 = 12$,
(iii) *the s-point Radau IIA or Gauss–Legendre methods with* $s \geqslant 3$ *is of order* 4 *and has* $\beta_1 = 7.06$.

For larger values of m, the same techniques can be used. We only need to calculate values for $\boldsymbol{b}_{\text{RK}}^{\text{T}} A_{\text{RK}}^k \boldsymbol{c}_{\text{RK}}$ in the case that $k \geqslant p$ and to calculate the zeros of the "Hurwitz polynomials" $R_j(z)$. For the Radau IIA and Gauss–Legendre methods of various orders we have performed these calculations to find the stability boundaries; the results are given in the Tables 1 and 2, respectively. In these tables we also give results for m-values that are larger than necessary to obtain the maximal order (i.e., $m > m_p$). This is done to see the effect on the stability boundary if we continue the iteration beyond m_p. The results corresponding to m_p are indicated in bold face in Tables 1 and 2. Furthermore, each stability boundary is followed (in parentheses) by the order p^* of the corresponding PIRKN method.

Table 1
Stability boundaries β_m (and the orders p^*) of the PIRKN methods based on Gauss–Legendre RK methods

	$s=2$	$s=3$	$s=4$	$s=5$	$s=6$	$s=7$
	$p=4$	$p=6$	$p=8$	$p=10$	$p=12$	$p=14$
	$m_p=1$	$m_p=2$	$m_p=3$	$m_p=4$	$m_p=5$	$m_p=6$
$m=1$	**12.00** (4)	7.06 (4)	7.06 (4)	7.06 (4)	7.06 (4)	7.06 (4)
$m=2$	12.00 (4)	**0.00** (6)	0.00 (6)	0.00 (6)	0.00 (6)	0.00 (6)
$m=3$	0.00 (4)	9.81 (6)	**9.51** (8)	9.51 (8)	9.51 (8)	9.51 (8)
$m=4$	12.00 (4)	0.00 (6)	0.00 (8)	**0.00** (10)	0.00 (10)	0.00 (10)
$m=5$	12.00 (4)	9.75 (6)	0.00 (8)	9.86 (10)	**9.86** (12)	9.86 (12)
$m=6$	0.00 (4)	0.00 (6)	9.86 (8)	0.00 (10)	0.00 (12)	**0.00** (14)

Table 2
Stability boundaries β_m (and the orders p^*) of the PIRKN methods based on Radau IIA RK methods

	$s=2$	$s=3$	$s=4$	$s=5$	$s=6$	$s=7$
	$p=3$	$p=5$	$p=7$	$p=9$	$p=11$	$p=13$
	$m_p=1$	$m_p=2$	$m_p=3$	$m_p=4$	$m_p=5$	$m_p=6$
$m=1$	**4.94** (3)	7.06 (4)	7.06 (4)	7.06 (4)	7.06 (4)	7.06 (4)
$m=2$	4.99 (3)	**2.19** (5)	0.00 (6)	0.00 (6)	0.00 (6)	0.00 (6)
$m=3$	3.52 (3)	10.46 (5)	**9.50** (7)	9.51 (8)	9.51 (8)	9.51 (8)
$m=4$	5.03 (3)	4.76 (5)	18.21 (7)	**0.21** (9)	0.00 (10)	0.00 (10)
$m=5$	5.44 (3)	11.70 (5)	5.40 (7)	26.35 (9)	**9.86** (11)	9.86 (12)
$m=6$	4.90 (3)	7.81 (5)	18.57 (7)	5.80 (9)	34.68 (11)	**0.00** (13)

From these tables we conclude that the stability boundaries show a rather irregular behaviour. Moreover, we see that several PIRKN methods possess a zero stability boundary. Although the degree of instability is usually rather mild in these cases, these methods are too dangerous to be used, especially in long-range integrations. Unfortunately, for the Gauss–Legendre-based PIRKN methods with odd s-values, these zero stability boundaries occur (also) on the diagonal in Table 1, which corresponds to the "optimal" situation, i.e., $m=m_p$. Therefore, in such cases we have to choose an other combination of m and s, and possibly of the generating RK method. For example, if we apply $m=4$ iterations to an RKN corrector based on the five-point Gauss–Legendre method, then we arrive at a PIRKN method of order 10; however, this method has an empty stability interval. A possible remedy to recover stability is to apply one additional iteration to the same corrector, yielding a stable ($\beta=9.86$) PIRKN method, still of order 10, with $m=s=5$. Another possibility is to change to a Radau-based corrector. The combinations ($m=4$, $s=5$) and ($m=5$, $s=6$) both yield a method with a nonempty stability interval of orders 9 and 11, respectively.

In Table 3 we specify the selection of PIRKN methods that will be used in the numerical tests. Here, the principal variable is the number of iterations m. For $m=1,\dots,5$, we select the PIRKN method of highest possible order, provided that it possesses a nonempty stability interval. In case of non-uniqueness, we choose the method with smallest s (i.e., the smallest number of processors needed).

We see that the Gauss–Legendre-based methods (with even s-value) have a reasonable stability boundary, whereas the Radau-based methods possess a small β, especially method IV.

Table 3
Selection of PIRKN methods

Method	Number of iterations m	Type of generating RK method	Number of stages (= processors)	Order of PIRKN method	Stability boundary
I	1	Gauss–Legendre	2	4	12.00
II	2	Radau IIA	3	5	2.19
III	3	Gauss–Legendre	4	8	9.51
IV	4	Radau IIA	5	9	0.21
V	5	Gauss–Legendre	6	12	9.86

For methods II and IV we have also calculated the stability boundaries in the case that the predictor (2.4c) is replaced by the even simpler variant

$$Y^{(0)} = e \otimes y_n. \tag{2.4c$'$}$$

Notice that this predictor yields $q = 1$ (cf. Theorem 2.2), but this does not increase the value of m_p since the Radau methods are of odd order. Unfortunately, this predictor did not improve the stability (as a matter of fact, the resulting stability intervals turned out to be empty). Also for the methods based on other (m, s) combinations, this predictor did not give rise to larger stability boundaries. Consequently, we will restrict our considerations to the predictor (2.4c).

3. Numerical experiments

In this section, we will test the methods specified in Table 3 and compare their behaviour to several sequential explicit RKN methods from the literature.

In all experiments described here, we shall use a fixed stepsize. An implementation using variable steps will be discussed in the next section.

The low-order PIRKN methods I and II will be compared with the very first RKN method, constructed by Nyström in 1925 [20] and defined by

$$\begin{array}{c|ccc}
0 & 0 & & \\
1/2 & 1/8 & & \\
1 & 0 & 1/2 & \\
\hline
 & 1/6 & 1/3 & 0 \\
 & 1/6 & 2/3 & 1/6
\end{array}$$

This method is of order 4 and its stability boundary is 6.69. This method will be referred to as N4.

Most papers dealing with high-order RKN methods that we found in the literature provide an embedded pair, allowing for stepsize variation. This property increases the number of stages that is needed to obtain a certain order; therefore, it would be unfair to use them as a reference method in this section where error control is not applied. Within the class of methods without embedding, we selected the eighth-order, eight-stage RKN method constructed by Hairer [11], and the eleven-stage method of order 10 given by the same author in [13]. We are not aware of a method of still higher order using the fewest number of stages possible. Furthermore, we will use the eighth-order, nine-stage method constructed by Beentjes and Gerritsen [2] as a reference method. Although this method uses one extra f-evaluation to reach order 8 (which is used to construct an seventh-order embedded reference solution), it has been chosen because of its small error constants and at the same time large stability boundary; it possesses $\beta = 26.6$. These three reference methods will be denoted by H8, H10, and BG8, respectively.

3.1. *Specification of test problems*

To test the performance of the various methods, they will be applied to a selection of test problems found in the literature.

Our first test example is the two-body gravitational problem, which is Problem Class D from the test set of Hull et al. [18] (see also [10])

$$y_1'' = -\frac{y_1}{r^3}, \qquad y_2'' = -\frac{y_2}{r^3}, \quad \text{with } r = \sqrt{y_1^2 + y_2^2}, \quad 0 \leqslant t \leqslant 20. \tag{3.1}$$

The initial conditions are given by $y(0) = (1 - \varepsilon, 0)^T$ and $y'(0) = (0, \sqrt{(1+\varepsilon)/(1-\varepsilon)})^T$, where ε denotes the eccentricity of the orbit. In the tests we set $\varepsilon = 0.9$. The exact solution can be found in [18].

The second example is a well-known test problem in the RKN literature (cf. e.g. [6,7,9,10]), defined by

$$y_1'' = -4t^2 y_1 - \frac{2y_2}{r}, \qquad y_2'' = -4t^2 y_2 + \frac{2y_1}{r}, \quad \text{with } r = \sqrt{y_1^2 + y_2^2}, \quad \sqrt{\tfrac{1}{2}\pi} \leqslant t \leqslant 10, \tag{3.2}$$

where the initial conditions are taken from the exact solution given by $y(t) = (\cos(t^2), \sin(t^2))^T$.

The last two examples are scalar problems and are taken from [9]. They are defined by

$$y'' = 2y^3, \quad 1 \leqslant t \leqslant 100, \qquad y(1) = 1, \qquad y'(1) = -1, \tag{3.3}$$

yielding the exact solution $y(t) = 1/t$, and by

$$y'' + 25y = 100\cos(5t), \quad 0 \leqslant t \leqslant 10, \qquad y(0) = 1, \qquad y'(0) = 5, \tag{3.4}$$

for which the exact solution is given by $y(t) = \cos(5t) + \sin(5t) + 10t\sin(5t)$.

3.2. Test results

In this subsection we will present the results obtained by applying the PIRKN methods I–V and the reference methods to the set of test problems. In the tables of results, we give the value of D, denoting the number of correct decimal digits at the endpoint, i.e., we write the maximum norm of the global error (for the y-component) in the form 10^{-D}. Furthermore, we measure the computational effort of a particular scheme by the number M, denoting the total number of *effective* f-evaluations required over the whole interval of integration. To allow for an easy comparison of the various methods, we choose a number of fixed M-values and

Table 4
D-values for problem (3.1) obtained for several values of M

Method	p	$M = 3200$	$M = 6400$	$M = 12800$	$M = 25600$	$M = 51200$
N4	4	0.1	1.1	2.4	3.8	5.1
PIRKN I	4	0.9	2.3	3.7	5.0	6.2
PIRKN II	5	1.0	2.4	3.8	5.3	6.8
H8	8	0.1	2.0	4.6	7.4	10.6
BG8	8	0.6	3.1	6.5	8.2	10.5
H10	10	0.3	1.8	4.9	8.4	11.7
PIRKN III	8	3.1	5.5	8.1	10.7	13.2
PIRKN IV	9	2.8	5.3	7.7	10.4	13.1
PIRKN V	12	3.7	7.4	11.1	15.5	19.1

Table 5
D-values for problem (3.2) obtained for several values of M

Method	p	$M = 400$	$M = 800$	$M = 1600$	$M = 3200$	$M = 6400$
N4	4	0.6	1.8	3.0	4.2	5.4
PIRKN I	4	1.1	2.4	3.5	4.7	5.9
PIRKN II	5	1.7	3.3	4.9	6.5	8.0
H8	8	0.3	2.6	5.2	7.6	10.0
BG8	8	0.9	3.1	5.6	8.0	10.4
H10	10	−0.3	2.2	5.4	8.5	11.5
PIRKN III	8	2.7	5.1	7.6	9.9	12.3
PIRKN IV	9	3.4	6.4	9.4	12.2	15.1
PIRKN V	12	4.1	7.6	11.2	14.9	18.5

adapted the stepsize of each method to arrive (approximately) at those pre-selected M-values. The results for the test problems (3.1)–(3.4) can be found in Tables 4–7, respectively.

For the first problem, we may draw the following conclusions:

(i) Especially the PIRKN methods nicely show their theoretical order behaviour; in this connection we remark that—for a method of order p—the value of D should increase by $\log_{10}(2^p) \approx 0.3p$ on halving the stepsize.

(ii) Both for low-order and high-order methods, the parallel PIRKN methods are more efficient than the corresponding sequential RKN methods.

(iii) Comparing the best sequential RKN method (i.e., H10) and the best parallel PIRKN method (i.e., method V), we see that the global errors of the PIRKN method are—for the same amount of work—smaller by several orders of magnitude; or, putting it differently, the PIRKN methods can obtain a certain accuracy in about 30% of the number of f-evaluations needed by H10.

(iv) It is remarkable that the BG8 method can easily compensate for the extra stage in comparison with the H8 method, which is of the same order.

The results for the second problem are given in Table 5 and give rise to roughly the same conclusions as formulated for the first example.

Table 6
D-values for problem (3.3) obtained for several values of M

Method	p	$M = 800$	$M = 1600$	$M = 3200$	$M = 6400$	$M = 12800$
N4	4	*	*	*	*	1.8
PIRKN I	4	*	*	*	1.9	3.0
PIRKN II	5	*	*	1.8	3.2	4.7
H8	8	*	*	2.5	4.8	7.2
BG8	8	*	*	2.2	4.6	7.0
H10	10	*	*	3.0	6.1	9.4
PIRKN III	8	*	2.6	4.9	7.3	9.7
PIRKN IV	9	0.2	3.1	5.6	8.3	11.0
PIRKN V	12	2.4	5.3	8.7	12.2	15.7

Table 7
D-values for problem (3.4) obtained for several values of M

Method	p	$M=200$	$M=400$	$M=800$	$M=1600$	$M=3200$
N4	4	−0.2	0.9	2.1	3.3	4.5
PIRKN I	4	0.2	1.4	2.6	3.8	5.0
PIRKN II	5	1.1	2.7	4.3	5.9	7.4
H8	8	0.5	2.6	5.0	7.4	9.8
BG8	8	2.6	3.2	5.5	7.8	10.2
H10	10	0.6	2.7	5.7	8.7	11.7
PIRKN III	8	2.8	5.0	7.3	9.7	12.1
PIRKN IV	9	4.0	6.6	9.5	12.4	15.3
PIRKN V	12	5.9	8.5	11.9	15.5	18.7

For the third test problem (3.3), we observe that an approximation to the local value of $\partial f/\partial y$ is given by $6y^2$, which is positive. Consequently, the methods show an unstable behaviour for particular values of the stepsize; such results are indicated by an "$*$" in Table 6. For h sufficiently small, we see that the behaviour of the methods is dictated by their convergence properties, resulting in an accurate integration. Again, the PIRKN methods are superior to the sequential methods; the number of required f-evaluations of the most accurate members from both families (viz. PIRKN V and H10) differ by a factor which is approximately equal to 3. For this problem, the H8 method is slightly more efficient than the BG8 method.

For the fourth test example (3.4) we draw the same conclusions as formulated for the first problem. The most accurate PIRKN method is more efficient than the best sequential method by a factor 4 (see Table 7).

4. Variable-step implementation

As mentioned in the introduction, a "realistic" implementation of an ODE solver requires the possibility to adapt the stepsize to the local behaviour of the solution. For that purpose, one usually employs the technique of *embedding*, which means that apart from the approximation for y_{n+1}, an additional result (of a different order) is calculated based on the same f-evaluations. A comparison of both results then gives an indication of the local truncation error. Many of such embedded RKN pairs have been constructed by Fehlberg. In [6–8] pairs are presented of the form $p(p+1)$ for p up to 10, where p is the order of the approximation to proceed the integration and $p+1$ is the order of the reference solution. However, similar to embedded pairs of RK methods for first-order ODEs, it is now generally believed that the strategy to use the highest-order approximation for the step continuation is superior. Based on this approach, Dormand and Prince [5] constructed an embedded RKN pair of order 7(6) using nine stages (an implementation of this method can be found in [14, code DOPRIN]). More recently, Filippi and Gräf continued this work and increased the order. In [10], they present pairs of order $p+1(p)$ with p running from 7 until 10, the corresponding number of stages varying between 9 and 17. Based on the test problems (3.1) and (3.2) they make in the same paper a comparison between many embedded pairs. Their conclusion is that (i) indeed the $p+1(p)$ strategy is more efficient than the "old" $p(p+1)$ approach, and (ii) that it is worthwhile to have high-order methods, because of the usually stringent accuracy demands for this type of problems.

The variable-step implementation based on their 11(10) pair (using 17 stages) is reported to be superior, particularly in the high-accuracy range. Therefore, the results of this code (denoted by FG11) will be used as a reference for the variable-step implementation of the PIRKN methods.

The error control that we have implemented is analogous to the one we have used in our code PIRK for first-order ODEs (this code is given in the appendix to [15]). This error control is essentially based on the observation that the order of the iterates $Y^{(j)}$ in (2.4a) increases by 2 in each iteration and that substitution of $Y^{(j)}$ into the final line of the RKN methods (i.e., into (2.3a)), yields a reference solution z_{n+1} of order $\min\{p, 2j+q\}$ (cf. Theorem 2.2).

Since $Y^{(j)}$ and its f-evaluation are anyhow needed to continue the iteration to reach the final iterate $Y^{(m_p)}$, the additional computational effort to construct this "embedded" reference solution is negligible. In principle, any j from the range 0 to $m_p - 1$ can be used for this purpose. In our implementation we set $j = m_p - 1$, and as an estimate of the local error LTE in the step from t_n to t_{n+1}, we take

$$\mathrm{LTE} := \| y_{n+1} - z_{n+1} \|, \tag{4.1}$$

for some norm $\| \cdot \|$. Using the definition of m_p, it is easily verified that, *locally*, LTE behaves as $O(h^{2s})$, both for Gauss–Legendre and Radau-based methods.

As usual, LTE is compared to some (user-)specified tolerance TOL and the step is accepted if $\mathrm{LTE} \leqslant \mathrm{TOL}$, and rejected otherwise. Furthermore, based on the value of LTE, the new stepsize is calculated according to (cf. [14, p. 167])

$$h_{\text{new}} = h_{\text{old}} \cdot \min\left\{\gamma_{\text{up}}, \max\left\{\gamma_{\text{low}}, \alpha(\mathrm{TOL}/\mathrm{LTE})^{1/(2s)}\right\}\right\},$$

where the constants γ_{up} and γ_{low} serve to avoid a too drastic change in the stepsize (viz. the amplification factor to obtain the new stepsize is required to be in the interval $[\gamma_{\text{low}}, \gamma_{\text{up}}]$), and α is introduced as a "safety parameter". In our code, which will be termed PIRKN, these parameters are set to $\gamma_{\text{up}} = 4$, $\gamma_{\text{low}} = \frac{1}{2}$, and $\alpha = 0.9$. It turned out that variation of the γ-values does not have a large influence on the performance of the code; this is what we expect, since a small change in the stepsize will result, for a high-order method, in a significant change in the local truncation error. The influence of α is noticeable, although its choice is not very critical.

For several values of the tolerance parameter TOL we applied the code PIRKN to the test problems. The results are given in the Tables 8–11. In these tables we give the values of D (accuracy) and M (computational effort), as defined in the previous section, as well as the total number of steps and, in parentheses, the number of rejected steps.

The orbit equation (3.1) represents a severe test for the error control mechanism, particularly due to its relatively large ε-value. Since low-accuracy results are not provided in [10], the codes PIRKN and FG11 can only be compared for very high precisions. From Table 8 we see that, in this range, the PIRKN code is much more efficient (roughly by a factor 3). A comparison of PIRKN with DOPRIN reveals that PIRKN is more efficient by a factor 2 in the low-accuracy range, increasing to a factor 10 in the high-precision range.

Another conclusion that can be drawn is that using variable stepsizes is much more efficient than integrating with a fixed stepsize (cf. Table 4). For this problem, the solution shows

Table 8
Performance of the parallel code PIRKN and the sequential codes FG11 and DOPRIN when applied to problem (3.1)

Code	TOL	D	M	Total number of steps (rejected ones)
PIRKN	10^{-4}	1.2	306	51 (19)
	10^{-8}	4.7	462	77 (23)
	10^{-12}	8.9	786	131 (33)
	10^{-16}	12.2	1488	248 (59)
	10^{-20}	16.7	2694	449 (67)
	10^{-24}	19.3	4806	801 (0)
FG11		17.0	9000	
		18.0	10900	
		19.0	13300	
DOPRIN	10^{-4}	2.3	737	92 (35)
	10^{-8}	5.7	1545	193 (56)
	10^{-12}	9.7	3777	472 (70)
	10^{-16}	14.7	11945	1493 (12)
	10^{-20}	18.5	44273	5534 (7)

relatively much variation in the neighbourhood of $t = 2\pi$, 4π, and 6π. Since PIRKN is able to reduce its stepsize in the vicinity of these points and takes large steps outside these regions, we see that PIRKN is approximately 10 times faster than the fixed stepsize analogue, i.e., method V.

For the second problem, we selected the best result form [10], viz. the 11(10) pair with the "... more or less arbitrary choice ..." for the parameter $\lambda = 1.4 \cdot 10^{-6}$, which turned out to be optimal for this problem. Unfortunately, only one TOL-value has been tested in [10]. Since this test problem has been used in other papers too, we collected some results from the literature and added these results to Table 9.

For this problem the difference between the variable-step and fixed-step implementation is much less pronounced than for the previous example. In fact, the stepsize variation in PIRKN is very modest and we observe almost the theoretical order behaviour of the underlying method of order 12: the quotient of the numbers of (successful) steps for successive TOL-values is approximately 2.15; the corresponding gain in the D-value is seen to be 4 digits, which reflects the relation $(2.15)^{12} \approx 10^4$.

Making a mutual comparison of the sequential codes, we see that FG11 is indeed the most efficient one, at least in the high-precision range (the question marks in the table mean that the number of rejected steps was not specified). However, when compared to PIRKN, the sequential code FG11 requires a number of f-evaluations which is 2.5 times as large. For DOPRIN we observe that, similar to the first test problem, its relatively low order prevents this code to be competitive in the high-accuracy range (say, $D \geqslant 10$); if modest accuracies are required (say, $D = 4$), then we see that PIRKN is about two times faster.

For the third test example, we found results in [7–9]. These results are listed in Table 10, together with the results of PIRKN and DOPRIN. Again, PIRKN is by far the most efficient code; DOPRIN encounters difficulties both in the low- and in the high-precision range. Also

Table 9
Performance of the parallel code PIRKN, the sequential codes FG11 and DOPRIN, and some additional methods when applied to problem (3.2)

Code	TOL	D	M	Total number of steps (rejected ones)
PIRKN	10^{-4}	3.9	300	50 (4)
	10^{-8}	7.9	588	98 (3)
	10^{-12}	12.0	1242	207 (2)
	10^{-16}	16.0	2658	443 (0)
	10^{-20}	19.9	5736	956 (0)
FG11	10^{-24}	20.7	15614	919 (?)
8(9) pair from [6]	10^{-17}	13.5	15973	1452 (?)
9(10) pair from [7]	10^{-17}	15.1	8793	628 (?)
	10^{-24}	21.4	45291	3235 (?)
11(12) pair from [9]	10^{-24}	20.3	17521	876 (?)
DOPRIN	10^{-4}	3.8	633	79 (4)
	10^{-8}	8.3	2825	353 (51)
	10^{-12}	12.3	9665	1208 (43)
	10^{-16}	16.3	35729	4466 (41)
	10^{-20}	20.3	133337	16667 (54)

for this example, we observe that the variable-step implementation PIRKN is very effective compared with the use of fixed stepsizes. Comparing the results of Tables 6 and 10, we observe a gain factor 15.

Table 10
Performance of the parallel code PIRKN, the sequential code DOPRIN, and some additional methods when applied to problem (3.3)

Code	TOL	D	M	Total number of steps (rejected ones)
PIRKN	10^{-4}	3.1	72	12 (0)
	10^{-8}	5.0	102	17 (0)
	10^{-12}	8.4	168	28 (0)
	10^{-16}	11.7	318	53 (0)
	10^{-20}	15.5	636	106 (0)
9(10) pair from [7]	10^{-24}	17.5	4831	345 (?)
10(11) pair from [8]	10^{-24}	16.2	3146	185 (?)
11(12) pair from [9]	10^{-24}	17.4	2641	132 (?)
DOPRIN	10^{-4}	code could not solve the problem		
	10^{-8}	3.7	273	34 (0)
	10^{-12}	7.5	897	112 (0)
	10^{-16}	11.6	3193	399 (0)
	10^{-20}	15.6	11777	1472 (0)

Table 11
Performance of the parallel code PIRKN, the sequential code DOPRIN, and an additional method when applied to problem (3.4)

Code	TOL	D	M	Total number of steps (rejected ones)
PIRKN	10^{-4}	2.5	168	28 (3)
	10^{-8}	6.6	366	61 (11)
	10^{-12}	10.5	666	111 (6)
	10^{-16}	14.5	1374	229 (4)
	10^{-20}	18.4	2958	493 (8)
11(12) pair from [9]	10^{-20}	15.7	4861	243 (?)
DOPRIN	10^{-4}	2.6	393	49 (9)
	10^{-8}	7.2	1529	191 (43)
	10^{-12}	11.3	4641	580 (38)
	10^{-16}	15.2	16481	2060 (33)
	10^{-20}	19.1	60833	7604 (29)

Results for the last test problem are found only in [9]. Together with the PIRKN and DOPRIN results, they are listed in Table 11.

For this problem, the variable-step implementation is not more efficient than the fixed-step version (cf. Table 7). A mutual comparison of the variable-step codes reveals that PIRKN offers a gain of a factor 3 when compared with the 11(12) pair from Filippi and Gräf and with a factor ranging from 2 to 15 (depending on the accuracy requested) when compared to DOPRIN.

5. Further improvement

In the following subsections, we briefly discuss a number of modifications that might further increase the efficiency of the code PIRKN.

5.1. *The order of the predictor*

In the code PIRKN we used the "trivial" predictor (2.4c). An advantage of this simple choice is that we remain within the class of *one-step* methods; clearly, a disadvantage is its low order, i.e., $q = 2$. Since the number of iterations m_p is defined as $\lceil (p-q)/2 \rceil$, it is obvious that we can save some iterations if we start with a predictor of higher order. The information needed to construct such a prediction could be obtained from approximations calculated in the preceding step. For example, in [17], van der Houwen and Nguyen huu Cong analyse (for first-order ODEs) a block version of the PIRK method [15] with the aim to construct a high-order prediction. The idea is to apply a PIRK method (which is very similar to a PIRKN method) not only with stepsize h, but, in addition (and simultaneously) with stepsizes $h_i = a_i h$, $i = 1, \dots, r-1$. In this way, r approximations are found in each step and this information is used to create a high-order prediction in the next step. Compared with the original PIRK method (i.e., with blocklength $r = 1$), a substantial increase in efficiency is reported in [17]. This idea is of course equally well applicable in the present context of second-order equations.

As a consequence of this modification, we leave the class of one-step methods and, second, this extension needs $r \cdot s$ processors. However, both effects seem to be not too restrictive.

5.2. The rate of convergence

In the aforementioned considerations the number of iterations was merely determined by order conditions, i.e., we used the minimal number to reach to *order* of the corrector. A slightly different approach would be to really obtain the *solution* of the corrector, rather than only its order. Taking this point of view, (2.4a) is now considered as an iteration scheme, where "iteration" has to be understood in the classical sense of the word. Hence, to converge as fast to the corrector solution as possible, a simple analysis based on the model equation $y'' = \lambda y$ indicates that we should minimize the spectral radius of the iteration matrix $h^2\lambda A$. In [19], Nguyen huu Cong studied this subject in more detail and found that RKN correctors based on so-called "direct collocation" (see also [16]) possess a smaller convergence factor (i.e., $\rho(A)$) than the "indirect collocation"-based correctors that we used in the present paper. Although these "direct RKN correctors" are usually not unconditionally stable, their stability boundaries are in many cases sufficiently large for nonstiff problems.

5.3. An inhomogeneous term

Finally, we mention a feature of PIRKN methods that is not present in the sequential embedded RKN pairs, such as FG11 and DOPRIN. From (2.4a) we see that, throughout the erations, the independent variable t is evaluated at the fixed point $t_n + c_i h$ for the ith omponent. This implies that if we have to solve an ODE of the form $y''(t) = f(t, y) + g(t)$, then the evaluation of g has to be done only in the first iteration and can be re-used in all subsequent iterations. Since most traditional RKN methods have c-vectors with many different elements, this option is not applicable for these methods.

6. Conclusions

An algorithm to obtain explicit Runge–Kutta–Nyström (RKN) methods of arbitrarily high order has been described. These methods are obtained by iterating an implicit RKN method (called the corrector) of sufficiently high order. Correctors satisfying this property are easily obtained by transforming a high-order Runge–Kutta (RK) method for first-order ODEs (which in turn are straightforwardly constructed; cf. [4]) to the case of second-order ODEs.

Applying the iteration scheme m times to an s-stage RKN corrector yields an explicit RKN scheme with $(m + 1) \cdot s$ stages. However, on a parallel computer with at least s processors, the scheme allows for the concurrent computation of the s f-evaluations within each iteration. In this way, the *effective* number of f-evaluations is reduced to $m + 1$. Taking $\lceil (p - q)/2 \rceil$ iterations (p and q denoting the order of the corrector and the predictor, respectively) it is proved that the order of the resulting Parallel-Iterated RKN (PIRKN) method equals the order of the corrector. For the Gauss–Legendre-based RKN correctors of order p we obtain a

PIRKN method of the same order requiring $p/2$ effective f-evaluations, which is an optimal result within the class of explicit RKN methods. For Radau IIA-based correctors of order p, the required number of effective f-evaluations equals $(p+1)/2$.

For these PIRKN methods we have studied the stability properties and it turned out that there are many (m, p) combinations that result in an empty stability region. On the basis of these results, we give a selection of suitable (m, p, corrector)-combinations. The resulting PIRKN methods have been compared with existing high-order (sequential) methods from the literature. We found that the PIRKN methods are more efficient by a factor 3 or 4.

A nice feature of the PIRKN methods is that they provide an embedded reference solution without additional f-evaluations. This feature has been exploited to control the local error in a variable step implementation of a PIRKN method of order 12. The resulting code, termed PIRKN, has been compared with high-order (sequential) codes from the literature, which also use the technique of embedding to control the local error. In terms of the required number of f-evaluations, PIRKN is shown to be much more efficient, especially in the high-accuracy range (which is the usual range for many of the problems under consideration).

Finally, in Section 5, some modifications and improvements have been mentioned, which may result in a further speedup of the parallel code over the sequential codes.

References

[1] R.H. Battin, Resolution of Runge–Kutta–Nyström condition equations through eighth order, *AIAA J.* 14 (1976) 1012–1021.

[2] P.A. Beentjes and W.J. Gerritsen, Higher order Runge–Kutta methods for the numerical solution of second order differential equations without first derivative, Report NW 34/76, Centre for Mathematics and Computer Science, Amsterdam (1976).

[3] J.C. Butcher, Implicit Runge–Kutta processes, *Math. Comp.* 18 (1964) 50–64.

[4] J.C. Butcher, *The Numerical Analysis of Ordinary Differential Equations, Runge-Kutta and General Linear Methods* (Wiley, New York, 1987).

[5] J.R. Dormand and P.J. Prince, New Runge–Kutta–Nyström algorithms for numerical simulation in dynamical astronomy, *Celestial Mech.* 18 (1978) 223–232.

[6] E. Fehlberg, Classical eighth- and lower-order Runge–Kutta–Nyström formulas with stepsize control for special second-order differential equations, NASA Tech. Report R-381; summary, *Computing* 10 (1972) 305–315 (in German).

[7] E. Fehlberg, Eine Runge–Kutta–Nyström-Formel 9-ter Ordnung mit Schrittweitenkontrolle für Differentialgleichungen $x'' = f(t, x)$, *Z. Angew. Math. Mech.* 61 (1981) 477–485.

[8] E. Fehlberg, S. Filippi and J. Gräf, Ein Runge–Kutta–Nyström Formelpaar der Ordnung 10(11) für Differentialgleichungen der Form $y'' = f(x, y)$, *Z. Angew. Math. Mech.* 66 (1986) 265–270.

[9] S. Filippi and J. Gräf, Ein Runge–Kutta–Nyström-Formelpaar der Ordnung 11(12) für Differentialgleichungen der Form $y'' = f(x, y)$, *Computing* 34 (1985) 271–282.

[10] S. Filippi and J. Gräf, New Runge–Kutta–Nyström formula-pairs of order 8(7), 9(8), 10(9) and 11(10) for differential equations of the form $y'' = f(x, y)$, *J. Comput. Appl. Math.* 14 (1986) 361–370.

[11] E. Hairer, Méthodes de Nyström pour l'équation différentielle $y'' = f(x, y)$, *Numer. Math.* 27 (1977) 283–300.

[12] E. Hairer, Unconditionally stable methods for second order differential equations, *Numer. Math.* 32 (1979) 373–379.

[13] E. Hairer, A one-step method of order 10 for $y'' = f(x, y)$, *IMA J. Numer. Anal.* 2 (1982) 83–94.

[14] E. Hairer, S.P. Nørsett and G. Wanner, *Solving Ordinary Differential Equations I: Nonstiff Problems*, Springer Series in Computational Mathematics 8 (Springer, Berlin, 1987).

[15] 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.
[16] P.J. van der Houwen, B.P. Sommeijer and Nguyen huu Cong, Stability of collocation-based Runge–Kutta–Nyström methods, *BIT* 31 (1991) 469–481.
[17] P.J. van der Houwen and Nguyen huu Cong, Parallel block predictor–corrector methods of Runge–Kutta type, *Appl. Numer. Math.* 13 (1993) 109–123 (this issue).
[18] T.E. Hull, W.H. Enright, B.M. Fellen and A.E. Sedgwick, Comparing numerical methods for ordinary differential equations, *SIAM J. Numer. Anal.* 9 (1972) 603–637.
[19] Nguyen huu Cong, Note on the performance of direct- and indirect Runge–Kutta–Nyström methods, *J. Comput. Appl. Math.* 45 (1993) 347–355.
[20] E.J. Nyström, Über die numerische Integration von Differentialgleichungen, *Acta Soc. Sci. Fenn.* 50 (13) (1925) 1–54.