UNIVERSITY OF CAMBRIDGE

Numerical Analysis Reports

Runge–Kutta methods for quadratic ordinary differential equations

Arieh Iserles, Geetha Ramaswami and Mark Sofroniou

DAMTP 1996/NA06
March 1996

Department of Applied Mathematics and Theoretical Physics
Silver Street
Cambridge
England CB3 9EW

Runge–Kutta methods for quadratic ordinary differential equations

Arieh Iserles[1], Geetha Ramaswami[2] and Mark Sofroniou[3]

Abstract Many systems of ordinary differential equations are quadratic: the derivative can be expressed as a quadratic function of the dependent variable. We demonstrate that this feature can be exploited in the numerical solution by Runge–Kutta methods, since the quadratic structure serves to decrease the number of order conditions. We discuss issues related to construction design and implementation and present a number of new methods of Runge–Kutta and Runge–Kutta–Nyström type that display superior behaviour when implemented with quadratic equations.

[1] Department of Applied Mathematics and Theoretical Physics, University of Cambridge.
[2] Departamento de Matemática Aplicada y Computación, Universidad de Valladolid.
[3] Wolfram Research Inc., 100 Trade Center Drive, Champaign, IL 61820, USA.

1 Quadratic ODEs

Many ODE systems of interest share the feature that $\boldsymbol{y}'$ is a quadratic function of the elements of $\boldsymbol{y}$. Specific examples are the *Volterra–Lotka* equations, *Belausov–Zhabotinskĭ* equations [8], the *Lorenz* equations [8], the generalized *Riccati* equation, the *Hénon–Heiles* Hamiltonian [21], *Toda lattice* equations with an exponential interaction after the Flaschka transformation [22], the *oreganator* equations [8], *Kermac and Mckendrick* model in epidemeology [8] and certain discretized PDEs: the *Burgers* equation and the *Fisher* equation. Although the above ODE systems exhibit disparate types of qualitative behaviour, their structure can be exploited in computation. This is the theme of the present paper.

Let us consider the *Quadratic Ordinary Differential Equation (QODE)*

$$y_k' = \alpha_k + \sum_{i=1}^{d} \beta_{k,i} y_i + \sum_{i=1}^{d}\sum_{j=1}^{d} \gamma_{k,i,j} y_i y_j, \qquad k = 1, 2, \ldots, d, \quad t \geq 0, \tag{1.1}$$

given in tandem with initial conditions at $t = 0$.

We are concerned with the solution of the QODE by ν-stage *Runge–Kutta* (RK) methods which, in the case of a general ordinary differential system

$$\boldsymbol{y}' = \boldsymbol{f}(t, \boldsymbol{y}), \quad t \geq 0, \qquad \boldsymbol{y}(0) = \boldsymbol{y}_0,$$

can be written in the form

$$\begin{aligned} \boldsymbol{k}_\ell &= \boldsymbol{f}(t_n + h\, c_\ell, \boldsymbol{\varphi}_\ell) \\ \boldsymbol{\varphi}_\ell &= \boldsymbol{y}_n + h \sum_{j=1}^{\nu} a_{\ell,j} \boldsymbol{k}_j \end{aligned} \qquad \ell = 1, \ldots, \nu,$$

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h \sum_{\ell=1}^{\nu} b_\ell \boldsymbol{k}_\ell. \tag{1.2}$$

Here $h = t_{n+1} - t_n$, $\boldsymbol{y}_n$ approximates the exact solution $\boldsymbol{y}(t_n)$ and the coefficients $(A, \boldsymbol{b}, \boldsymbol{c})$ characterize the underlying method. They are frequently displayed in the *Butcher tableau* (4.1). We refer to [5, 8, 10] for Runge–Kutta terminology and theory. An application of the method (1.2) to the QODE (1.1) will be henceforth termed a *Quadratic Runge–Kutta* (QRK) method.

Lawson [14] was the first to note that some of the elements in the higher derivatives vanish for QODEs. He showed that the order conditions for such methods can be represented as bifurcated (i.e., binary) rooted trees. He also derived a 5-stage 5th order explicit RK method for QODEs in his paper. In the present paper we undertake a more systematic study of this subject matter.

2 Binary tree enumeration

In this work we will primarily follow the formalism of Butcher who represents elementary differentials and order conditions for RK methods in terms of *rooted trees* [5]. Specifically, a method (1.2) is of order p if its coefficients satisfy a set of conditions for each rooted tree of order $\leq p$. In the case of RKQ, order conditions can be expressed in terms of binary trees. In this section we examine some issues related to the enumeration and in the next section we discuss the computation of binary trees.

It is standard to use generating functions to derive formulae for the purpose of tree enumeration [12, Section 2.3.4.4]. For the binary case, the following relation holds (see [1] and the references therein),

$$G(z) = 1 + \frac{z}{2}\left[G(z)^2 + G(z^2)\right],$$

Table 1: The number of rooted trees and binary rooted trees.

Order	1	2	3	4	5	6	7	8	9	10
Rooted Trees	1	1	2	4	9	20	48	115	286	719
Binary Trees	1	1	2	3	6	11	23	46	98	207

which implies that certain terms only come into consideration for odd order. Expanding the generating function as

$$G(z) = 1 + zg_1 + z^2 g_2 + \ldots$$

and equating powers of z yields the coefficients recursively. However, an explicit termwise recurrence for the coefficients can be derived, which is more efficient for the purpose of evaluation.

For the standard case, the following recurrence may be used to determine the coefficients (see exercises 1 and 2 in [12, Section 2.3.4.4]),

$$\begin{aligned} a_1 &= 1, \\ a_n &= \frac{1}{n-1} \sum_{i=1}^{(n-1)} \sum_{j=1}^{[(n-1)/i]} i a_i a_{n-ij} \end{aligned}$$

where $[q]$ denotes rounding down the real number q to the nearest integer. For the binary case, a recurrence for tree enumeration is given by:

$$\begin{aligned} g_1 &= 1 \\ g_n &= \delta g_n + g_{n-1} + \sum_{j=1}^{[(n-2)/2]} g_j\, g_{n-j-1} \end{aligned}$$

where

$$\delta g_n = \begin{cases} \frac{1}{2} g_{(n-1)/2} \left(1 + g_{(n-1)/2}\right), & n \text{ odd}, \\ 0, & n \text{ even}. \end{cases}$$

Using the above formulae, the reduction in the number of trees in the binary case is illustrated in Table 1.

This leads to a significant decrease in the number of order conditions which need to be considered for RKQ methods. As a consequence it is frequently possible to choose coefficients so that the order of RKQ is greater than the order of the corresponding RK method. At the very least, this extra freedom can be used to derive methods with smaller principal truncation error.

3 Automating the derivation of Runge–Kutta schemes

It was shown in [19] how order conditions for Runge–Kutta methods may be derived in a systematic fashion in a computer algebra system using Butcher's formalism. There are several advantages to this approach. The manipulation of order conditions, becomes intractable by hand as the order and number of stages of a Runge–Kutta method increase. Indeed, many simplifying assumptions are often made in order to reduce the complexity of this task. The problem with simplifying assumptions is that they place restrictions on the class of methods that we pursue. These restrictions can be circumvented at certain orders by appropriate development and application of modern software tools. We will give several complete categorisations of methods as illustration. Another advantage of the automation of a formalism is obviously the elimination of human error.[4]

There has been significant progress in recent years in algorithms for the symbolic solution of systems of nonlinear algebraic equations (see [11] for an introductory summary). The most significant work in the area of symbolic solutions is undoubtedly the seminal contribution of Buchberger and his algorithm for the computation of Gröbner bases [3]. Most modern computer algebra systems possess some form of implementation of this algorithm. The drawback with the computation of Gröbner bases is the doubly exponential running time as the Bezöut bound of the system grows. Furthermore, the time required to obtain solutions is also highly dependent upon the elimination order. In such cases not all is lost. We may still turn to the traditional approach of utilising numerical optimization routines and translate our algebraic constraints to the target language of our numerical software library. Once solutions have been found, we may use the arbitrary precision capabilities of computer algebra to fix the digits using a local Newton-type search to any desired tolerance. The methods with inexact coefficients in Section 6 were polished so that they are suitable for use in double precision computations.

However, it is worth persisting with analytic technology as far as possible; as we shall see in Section 6, the high degree of local nonlinearity of order conditions means that local optimization routines will often miss solutions even with a fairly fine grid of initial values.

We note in passing that homotopy methods have recently received some attention in the context of numerical solution of nonlinear algebraic problems. These methods show much promise in overcoming limitations arising from the Bezöut bound of the system. However, to date, no general purpose system has an implementation of such a scheme.

3.1 Binary tree computation

The first step in automating the derivation of Runge–Kutta methods is to compute the trees themselves. Perhaps the most natural construction is recursive. It has long been established, though not formally documented, that rooted trees can be constructed by noting the correspondence with the number of unrestricted partitions of an integer [4]. Our primary concern here is the efficient computation of binary trees, a process which is surprisingly difficult to trace in the literature considering the abundance of such objects in computer science. Consider a number paired with its multiplicity as $\{i, j\}$ then we may write the partitions of $n = 5$ as:

$$\{\{5,1\}\}\ \{\{4,1\},\{1,1\}\},\ \{\{3,1\},\{2,1\}\},\ \{\{3,1\},\{1,2\}\},\ \{\{2,2\},\{1,1\}\},\ \{\{2,1\},\{1,3\}\},\ \{\{1,5\}\}.$$

The advantage of this notation is that it avoids the repeated listing of equal parts and enables a running time independent of n. There is a second advantage with respect to binary tree generation. It becomes an easy matter to generate all the partitions by proceeding iteratively from one partition to the next. See [17] for a more detailed description of such an algorithm.

For binary rooted trees we need to consider *restricted* partitions. The key to the construction is to make use of the fact that the number of restricted partitions of n into parts at most k in magnitude is equal to the number of partitions into parts of size at most k. This is most commonly demonstrated by *transposition* or reflection using a *Ferrers* (or *Young*) diagram (see Figure 1) [23]. Thus we can obtain the desired result by transposition of a quantity that is straighforward to generate, as described above.

Assume we have all given sets of binary rooted trees up to order 5 and we seek the binary trees for order 6. The first step is to compute all the partitions of 5 with magnitude at most 2:[5]

$$\{\{\{2,2\},\{1,1\}\},\{\{2,1\},\{1,3\}\},\{\{1,5\}\}\}.$$

[4] The results of any large computer software program should be independently confirmed and modern computer algebra systems are no exception.

[5] The order of the partition is one lower than the order of the trees we seek, since we will be grafting the result onto a new root

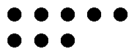


Figure 1 A Ferrers diagram illustrating a partition of eight into parts of size at most two 2+2+2+1+1 (vertical) and the one-to-one correspondence with the partition into two parts 5 + 3 (horizontal)

The next step is to compute the transposition:

$$\{\{\{3,1\},\{2,1\}\},\{\{4,1\},\{1,1\}\},\{\{5,1\}\}\}$$

Finally we construct all possible combinations of trees of appropriate orders according to their multiplicity. Care is needed in the construction to avoid duplicates using the results of Section 2. We will not go into further detail here and the interested reader is referred to the Mathematica package `QuadraticRungeKutta.m` which implements Butcher's formalism for the quadratic case and is available from any of the authors.

4 Explicit Runge–Kutta

4.1 Quadratic order conditions

For 5th order, four rooted trees have disappeared as a result of removing all but binary trees. This might have seem a moderate saving, but the proportion of binary trees decreases fast with order. Moreover – and this will be our next theme – even this moderate decrease is ample to derive superior RK methods. Similar reductions can be observed for RK-Nyström methods and for partitioned RK methods.

For order $p \le 5$ the order conditions are displayed in Table 2. The trees depicted are rooted trees with the root being the bottom vertex. The order conditions should be understood as

$$\sum_{j=1}^{s} b_j \Phi_j(t) = \frac{1}{\gamma(t)},$$

where the underlying RK method is

$$\begin{array}{c|cccc} c_1 & a_{1,1} & a_{1,2} & \cdots & a_{1,s} \\ c_2 & a_{2,1} & a_{2,2} & \cdots & a_{2,s} \\ \vdots & \vdots & \vdots & & \vdots \\ c_s & a_{s,1} & a_{s,2} & \cdots & a_{s,s} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array} \tag{4.1}$$

We will assume that the row-sum conditions

$$c_i = \sum_{j=1}^{s} a_{i,j}, \quad i = 1, 2, \ldots, s \tag{4.2}$$

hold. These appear naturally in the derivation of high order methods. If, in addition, the condition

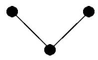
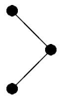
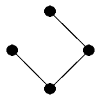
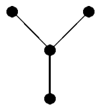
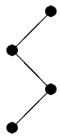
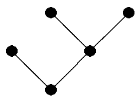
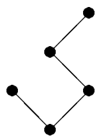
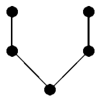
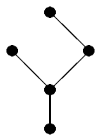
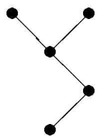
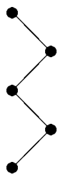
$$a_{i,j} = 0, \quad 1 \le i \le j \le s \tag{4.3}$$

holds, then the method is said to be *explicit* and the intermediate stages $\boldsymbol{k}_i$ in the Runge–Kutta method can be computed recursively.

Henceforth, whenever referring to the order of an ODE method *restricted to equations of the form* (1.1), we designate it as Q-order. Needless to say,

$$Q\text{-order} \le \text{order}.$$

Table 2: Order conditions for RKQ of Q-order ≤ 5.

order	t	graph	$\Phi_j(t)$	$\gamma(t)$
1	τ		1	1
2	$[\tau]$		$\sum_k a_{j,k}$	2
3	$[\tau^2]$		$\sum_{k,\ell} a_{j,k} a_{j,\ell}$	3
	$[[\tau]]$		$\sum_{k,\ell} a_{j,k} a_{k,\ell}$	6
4	$[\tau[\tau]]$		$\sum_{k,\ell,m} a_{j,k} a_{k,\ell} a_{j,m}$	8
	$[[\tau^2]]$		$\sum_{k,\ell,m} a_{j,k} a_{k,\ell} a_{k,m}$	12
	$[[[\tau]]]$		$\sum_{k,\ell.m} a_{j,k} a_{k,\ell} a_{\ell,m}$	24
5	$[\tau[\tau^2]]$		$\sum_{k,\ell,m,i} a_{j,k} a_{k,\ell} a_{k,m} a_{j,i}$	15
	$[\tau[[\tau]]]$		$\sum_{k,\ell,m,i} a_{j,k} a_{k,\ell} a_{\ell,m} a_{j,i}$	30
	$[[\tau]^2]$		$\sum_{k,\ell,m,i} a_{j,k} a_{k,\ell} a_{j,m} a_{m,i}$	20
	$[[\tau[\tau]]]$		$\sum_{k,\ell,m,i} a_{j,k} a_{k,\ell} a_{\ell,m} a_{k,i}$	40
	$[[[\tau^2]]]$		$\sum_{k,\ell,m,i} a_{j,k} a_{k,\ell} a_{\ell,m} a_{\ell,i}$	60
	$[[[[\tau]]]]$		$\sum_{k,\ell,m,i} a_{j,k} a_{k,\ell} a_{\ell,m} a_{m,i}$	120

4.2 Complete characterisation of 4-stage, 4th Q-order explicit RKQ

For explicit RK, the matrix A is now strictly lower triangular and $c_1 = 0$, which follows from (4.2),(4.3). The fact that the order condition corresponding to the tree $[\tau^3]$ is absent from the Q-order conditions at order 4 means that the standard constraint $c_4 = 1$ is no longer valid. As a consequence, the proof of necessity of the column simplifying assumption, $D(1)$, in the standard 4-stage 4th order case no longer holds [5, Lemma 311C].

The following algorithm determines a three-parameter family of solutions. We can solve the 4th Q-order conditions as a *linear* system by choosing the free parameters as c_2, c_3, c_4.

1. Determine b_1, b_2 and b_3 from the quadrature conditions corresponding to τ, $[\tau]$, $[\tau^2]$.
2. $a_{4,2}$ from the condition for $[[\tau]]$.
3. b_4 from the condition for $[[[\tau]]]$.
4. $a_{3,2}$ from the condition for $[[\tau^2]]$.
5. $a_{4,3}$ from the condition for $[\tau[\tau]]$.

The solution thus obtained, subject to some trivial exceptions, is given by

$$\begin{aligned}
b_1 &= \frac{6c_2 + 4c_3 + 4c_4 - 6c_2c_3 - 8c_2c_4 - 6c_3c_4 + 12c_2c_3c_4 - 3}{12c_2c_3c_4}, \\
b_2 &= \frac{3 - 2c_2 - 4c_3 + c_4\left(2c_2 + 6c_3 - 4\right)}{12c_2\left(c_3 - c_2\right)\left(c_4 - c_2\right)}, \\
b_3 &= \frac{\left(2c_2 - 1\right)\left(3 - 4c_4\right)}{12c_3\left(c_2 - c_3\right)\left(c_3 - c_4\right)}, \\
b_4 &= \frac{3 - 4c_3 + c_2\left(6c_3 + 2c_4 - 6\right)}{12c_4\left(c_2 - c_4\right)\left(c_3 - c_4\right)}, \\
a_{3,2} &= \frac{c_3\left(c_2 - c_3\right)}{2c_2\left(2c_2 - 1\right)}, \\
a_{4,2} &= \frac{c_4\left(c_2 - c_4\right)\left(-3c_2 + 5c_3 - 4{c_3}^2 - 2c_4 + 4c_2c_4\right)}{2c_2\left(c_2 - c_3\right)\left(3 - 4c_3 + c_2\left(6c_3 + 2c_4 - 6\right)\right)}, \\
a_{4,3} &= \frac{c_4\left(2c_2 - 1\right)\left(c_2 - c_4\right)\left(c_3 - c_4\right)}{c_3\left(c_2 - c_3\right)\left(3 - 4c_3 + c_2\left(6c_3 + 2c_4 - 6\right)\right)}.
\end{aligned}$$

The remaining coefficients a_{i1}, $i = 2, 3, 4$ are determined by the row-sum conditions (4.2).

4.3 Column Simplifying Assumptions

We now turn our attention to Butcher's column simplifying assumptions, $D(1)$. For an s-stage scheme $D(1)$ is given by:

$$\sum_i b_i a_{i,j} = b_j\left(1 - c_j\right), \qquad j = 1, \ldots, s.$$

We note in passing that if the Runge–Kutta method is of order 2 and the row-simplifying assumptions (4.2) hold then $D(1)$ is inferred for $j = 1$. These assumptions often aid in the derivation of high order schemes. As already remarked, they lead to a loss of generality for the 4th Q-order case. It is natural to investigate the role of the column simplifying assumptions at higher order. It turns out that the assumption $D(1)$ plays an important simplifying role in the derivation of 5 stage 5th order methods. In order to prove this and other statements, we will make use of the following result [5, Lemma 311A].

Lemma 4.1 *Let U, V be 3×3 matrices such that*

$$U\,V = \begin{bmatrix} w_{11} & w_{12} & 0 \\ w_{21} & w_{22} & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

where $w_{11}w_{22} - w_{21}w_{12} \neq 0$. Then either the third row of U or the third column of V is the zero vector.

A proof can now be constructed along the lines of familiar arguments of nonexistence of 5-stage 5th order methods in the standard case [5, Section 311]. Some subtlety is required in order to avoid non-binary trees; specifically bushy trees $[\tau^{(p-1)}]$ with order $p \geq 4$.

Theorem 4.1 *For 5-stage Q-order 5 methods the condition $D(1)$ holds.*

Proof Denote μ_j, β_j by

$$\begin{aligned} \mu_j &= b_j\,(1 - c_j)\,, \qquad j = 2, \ldots, 5, \\ \beta_j &= \sum_j b_i a_{i,j}\,, \qquad j = 2, \ldots, 5. \end{aligned}$$

Furthermore, define λ by the relation:

$$\lambda_i = \sum_{j,k} a_{i,j} a_{j,k} c_j - \frac{1}{2} \sum_j a_{i,j} c_j^2, \qquad i = 3, \ldots, 5.$$

Next define

$$U = \begin{bmatrix} a_{3,2}c_2 & \sum_{i=2}^{3} a_{4,i}c_i & \sum_{i=2}^{4} a_{5,i}c_i \\ a_{3,2}c_2^2 & \sum_{i=2}^{3} a_{4,i}c_i^2 & \sum_{i=2}^{4} a_{5,i}c_i^2 \\ \lambda_3 & \lambda_4 & \lambda_5 \end{bmatrix}, \qquad V = \begin{bmatrix} b_3 & b_3c_3 & \mu_3 - \beta_3 \\ b_4 & b_4c_4 & \mu_4 - \beta_4 \\ b_5 & b_5c_5 & \mu_5 - \beta_5 \end{bmatrix}.$$

Multiplying $U\,V$ and inserting the five Q-order conditions yields

$$W = UV = \begin{bmatrix} \frac{1}{6} & \frac{1}{8} & 0 \\ \frac{1}{12} & \frac{1}{15} & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Hence the conditions of Lemma 4.1 are satisfied. Now $\lambda_3 \neq 0$ since from the order condition for $[[[[\tau]]]]$ we have $a_{3,2}c_2 \neq 0$. Therefore it cannot be the case that the last row of U vanishes. Consequently we must have $\mu_i = \beta_i,\ i = 3, 4, 5$. Finally, consider the relation:

$$\Phi([\tau]) - \Phi([\tau^2]) = \Phi([[\tau]])\,,$$

which holds if the method is of Q-order 3. Inserting the relations $\mu_i = \beta_i,\ i = 3, 4, 5$ together with the fact that $c_2 \neq 0$ implies $\mu_2 = \beta_2$. □

The interpretation of this result is that several nonlinear algebraic constraints can be replaced by simpler linear conditions $D(1)$ *without* loss of generality. We also have the following consequence,

Corollary 4.1 *Any 5 stage Q-order 5 Runge–Kutta method is of standard order 4.*

Proof We need only satisfy one additional order condition corresponding to $[\tau^3]$ to obtain standard order 4. The order conditions corresponding to $[[\tau^2]]$ and $[\tau^2]$ hold since the method is of Q-order 5. The result follows by insertion of the Q-order conditions into $D(1)$. □

Thus we can safely use any 5-stage 5th order RKQ method and take comfort in the fact that its order will be at least 4 for other types of differential systems. In other words, a scheme of Q-order 5 should be a method of choice among all 5-stage explicit RK methods, because at no extra expense it produces superior solutions of quadratic systems. More generally, $D(1)$ enables the removal of all *one-leg trees*, $[[t]]$ where t is any tree, from consideration. The only proviso here is that we adopt conditions arising from non-binary trees, specifically $[\tau^3]$ for the 5 stage 5th order case. Examined from another point of view, $D(1)$ implies that certain order constraints corresponding to *non-binary* trees are sometimes automatically satisfied by a method of appropriately high Q-order. We are now in a position to derive the complete solution of a method which overcomes an infamous order barrier, by enforcing the weaker condition of Q-order 5.

4.4 Complete characterisation of 5-stage, 5th Q-order explicit RKQ

There are thirteen order conditions for Q-order 5. Theorem 4.1 illustrates that without loss of generality imposing $D(1)$ we can remove the four conditions arising from the slender trees $[[\tau]]$, $[[[\tau]]]$, $[[[[\tau]]]]$ and the tree $[[[\tau^2]]]$. Furthermore adopt $[\tau^3]$ and $[\tau^2[\tau]]$ in preference to $[\tau[\tau]]$ and $[[\tau[\tau]]]$.[6] We are then left with the nine order conditions:

$$\begin{aligned}
\tau: &\quad b_1+b_2+b_3+b_4+b_5=1,\\
[\tau]: &\quad b_2c_2+b_3c_3+b_4c_4+b_5c_5=\frac{1}{2},\\
[\tau^2]: &\quad b_2{c_2}^2+b_3{c_3}^2+b_4{c_4}^2+b_5{c_5}^2=\frac{1}{3},\\
[\tau^3]: &\quad b_2{c_2}^3+b_3{c_3}^3+b_4{c_4}^3+b_5{c_5}^3=\frac{1}{4},\\
[\tau[\tau]]: &\quad a_{3,2}b_3c_2c_3+b_4\left(a_{4,2}c_2+a_{4,3}c_3\right)c_4+b_5\left(a_{5,2}c_2+a_{5,3}c_3+a_{5,4}c_4\right)c_5=\frac{1}{8},\\
[\tau^2[\tau]]: &\quad a_{3,2}b_3c_2{c_3}^2+b_4\left(a_{4,2}c_2+a_{4,3}c_3\right){c_4}^2+b_5\left(a_{5,2}c_2+a_{5,3}c_3+a_{5,4}c_4\right){c_5}^2=\frac{1}{10},\\
[[\tau]^2]: &\quad {a_{3,2}}^2b_3{c_2}^2+b_4(a_{4,2}c_2+a_{4,3}c_3)^2+b_5(a_{5,2}c_2+a_{5,3}c_3+a_{5,4}c_4)^2=\frac{1}{20},\\
[\tau[\tau^2]]: &\quad a_{3,2}b_3{c_2}^2c_3+b_4\left(a_{4,2}{c_2}^2+a_{4,3}{c_3}^2\right)c_4+b_5\left(a_{5,2}{c_2}^2+a_{5,3}{c_3}^2+a_{5,4}{c_4}^2\right)c_5=\frac{1}{15},\\
[\tau[[\tau]]]: &\quad a_{3,2}a_{4,3}b_4c_2c_4+b_5\left(a_{3,2}a_{5,3}c_2+a_{5,4}\left(a_{4,2}c_2+a_{4,3}c_3\right)\right)c_5=\frac{1}{30}.
\end{aligned}$$

The following algorithm enables the determination of a two-parameter family of solutions. Consider c_2, c_3, c_4 as free parameters.

1. Determine $a_{3,2}$, $a_{4,3}$, $a_{5,4}$ and c_5 from $D(1)$.
2. Determine b_1, b_2, b_3, b_4 from the quadrature conditions ($[[\tau^2]]$ reduces to $[\tau^3]$ under $D(1)$ by Corollary 4.1).
3. Determine $a_{4,2}$, $a_{5,2}$, $a_{5,3}$ from the (linear) constraints corresponding to $[\tau[\tau]]$, $[\tau^2[\tau]]$ and $[\tau[\tau^2]]$.
4. Determine b_5 from the (linear) constraint arising from $[\tau[[\tau]]]$.

[6] This simplifies the derivation somewhat but is by no means necessary.

5. Determine one of c_2, c_3, c_4 via the remaining condition corresponding to $[[\tau]^2]$.

The solution thus obtained is given by

$$
\begin{aligned}
a_{3,2} &= \frac{c_3\,(c_2-c_3)}{c_2\,(5c_2-2)},\\
a_{4,2} &= \frac{c_4\,(c_2-c_4)\left(-3c_2+5c_3-5c_3^2-2c_4+5c_2c_4\right)}{c_2\,(c_2-c_3)\,(6-15c_2-10c_3+20c_2c_3+5c_2c_4)},\\
a_{4,3} &= \frac{c_4\,(5c_2-2)\,(c_2-c_4)\,(c_3-c_4)}{c_3\,(c_2-c_3)\,(6-15c_2-10c_3+20c_2c_3+5c_2c_4)},\\
a_{5,2} &= \frac{(c_2-1)\,k_1}{c_2\,(c_2-c_3)\,(c_2-c_4)\,k_2},\\
a_{5,3} &= \frac{(1-c_2)\,(2-5c_2)\,(1-c_3)\left(-3+c_3+7c_4-5c_4^2\right)}{c_3\,(c_2-c_3)\,(c_3-c_4)\,k_2},\\
a_{5,4} &= \frac{(1-c_2)\,(1-c_3)\,(1-c_4)\,(6-15c_2-10c_3+20c_2c_3+5c_2c_4)}{c_4\,(c_2-c_4)\,(c_3-c_4)\,k_2},\\
b_1 &= \frac{15c_2+10c_3-20c_2c_3+10c_4-25c_2c_4-20c_3c_4+60c_2c_3c_4-6}{120c_2c_3c_4},\\
b_2 &= \frac{6-5c_2-10c_3-10c_4+5c_2c_4+20c_3c_4}{120c_2\,(1-c_2)\,(c_2-c_3)\,(c_2-c_4)},\\
b_3 &= \frac{(5c_2-2)\,(3-5c_4)}{120c_3\,(c_2-c_3)\,(1-c_3)\,(c_3-c_4)},\\
b_4 &= \frac{6-15c_2-10c_3+20c_2c_3+5c_2c_4}{120c_4\,(c_2-c_4)\,(1-c_4)\,(c_3-c_4)},\\
b_5 &= \frac{24-25c_2-30c_3+40c_2c_3-30c_4+35c_2c_4+40c_3c_4-60c_2c_3c_4}{120\,(1-c_2)\,(1-c_3)\,(1-c_4)},\\
c_5 &= 1,\\
k_1 &= 6-7c_2-3c_2^2-16c_3+15c_2c_3+10c_3^2-5c_2c_3^2-14c_4+13c_2c_4+5c_2^2c_4+\\
&\quad 36c_3c_4-15c_2c_3c_4-25c_3^2c_4+10c_4^2-10c_2c_4^2-25c_3c_4^2+5c_2c_3c_4^2+20c_3^2c_4^2,\\
k_2 &= 24-25c_2-30c_3+40c_2c_3-30c_4 35c_2c_4+40c_3c_4-60c_2c_3c_4.
\end{aligned}
$$

One of c_2, c_3, c_4 is determined in terms of the other two from the quadratic constraint

$$
\begin{aligned}
0 &= 234c_2-135c_2^2+210c_3-465c_2c_3-45c_2^2c_3-150c_3^2+150c_2c_3^2+\\
&\quad 300c_2^2c_3^2+162c_4-465c_2c_4+150c_2^2c_4-480c_3c_4+900c_2c_3c_4+\\
&\quad 455c_2^2c_3c_4+350c_3^2c_4-300c_2c_3^2c_4-700c_2^2c_3^2c_4-90c_4^2+255c_2c_4^2-\\
&\quad 75c_2^2c_4^2+270c_3c_4^2-555c_2c_3c_4^2-100c_2^2c_3c_4^2-200c_3^2c_4^2+300c_2c_3^2c_4^2-72\,.
\end{aligned}
$$

The remaining coefficients $a_{i,1}, i=2,\ldots,5$ are determined by the row-sum conditions (4.2).

The algorithm described above is not unique and there are other ways to arrive at the same solution set. We now give an example of a method that obeys all the above conditions:

$$
\begin{array}{c|ccccc}
0 & 0 & 0 & 0 & 0 & 0\\
\frac{1}{5} & \frac{1}{5} & 0 & 0 & 0 & 0\\
\frac{1}{2} & -\frac{1}{4} & \frac{3}{4} & 0 & 0 & 0\\
\frac{4}{5} & \frac{11}{10} & -\frac{3}{2} & \frac{6}{5} & 0 & 0\\
1 & -\frac{3}{2} & \frac{55}{18} & -\frac{10}{9} & \frac{5}{9} & 0\\
\hline
 & \frac{1}{12} & \frac{25}{108} & \frac{10}{27} & \frac{25}{108} & \frac{1}{12}
\end{array}
\tag{4.4}
$$

Of particular note here is that we have $b_i > 0$, $i = 1, \ldots, 5$ which is not possible in the standard case because of the order condition corresponding to $[\tau^4]$ [5, 330d]. It is common practice to make the assumption $b_2 = 0$ in conjunction with the adoption of row simplifying assumptions of order 2. It should now be clear why these assumptions leads to loss of generality in the quadratic case.

Next we examine some classes of embedded methods, with intrinsic error estimation. These are used by modern codes for step control purposes [8, 10]. To dampen our enthusiasm slightly we present some negative results.

4.5 Embedded 4(3) RKQ methods of 4 stages

Theorem 4.2 *No 4(3) RKQ method with 4 stages exists.*

Here are two ways to demonstrate this. The first is a computer-based proof that can be resolved in about a second by using Mathematica version 3.0 on a typical personal computer. The second proof is more traditional.

Proof 1 The embedded method must possess distinct coefficients $\hat{b}_i \neq b_i$ for some $i \in \{1, 2, 3, 4\}$. For this to be the case the following system must be singular

$$\begin{bmatrix} c_2 & c_3 & c_4 \\ c_2^2 & c_3^2 & c_4^2 \\ 0 & a_{3,2}c_2 & a_{4,2}c_2 + a_{4,3}c_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \\ \frac{1}{3} \\ \frac{1}{6} \end{bmatrix}$$

which is equivalent to requiring:

$$(c_2 - c_3)\, c_3\, (a_{4,2}c_2 + a_{4,3}c_3) + a_{3,2}c_2c_4\, (c_4 - c_2) = 0 \tag{4.5}$$

since we know from the order condition for $[[[\tau]]]$ that $c_2 \neq 0$. If we add the constraint (4.5) to the seven Q-order 4 conditions , we obtain an inconsistent system (the Gröbner basis is $\{1\}$). □

Proof 2 Assume $\hat{b}_i \neq b_i$ for some $i \in \{1, 2, 3, 4\}$, so that the higher order method and the embedded method do not coincide. Consider the following system derived from the Q-order conditions:

$$\begin{bmatrix} b_2 & b_3 & b_4 \\ \sum_{i=3}^{4} b_i a_{i,2} & b_4 a_{4,3} & 0 \\ b_2 - \hat{b}_2 & b_3 - \hat{b}_3 & b_4 - \hat{b}_4 \end{bmatrix} \begin{bmatrix} c_2 & c_2^2 & -\dfrac{c_2}{2} \\ c_3 & c_3^2 & a_{3,2}c_j - \dfrac{c_3}{2} \\ c_4 & c_4^2 & \sum_{j=2}^{3} a_{4,j}c_j - \dfrac{c_3}{2} \end{bmatrix} = \begin{bmatrix} \dfrac{1}{2} & \dfrac{1}{3} & 0 \\ \dfrac{1}{6} & \dfrac{1}{12} & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Now the conditions of Lemma 4.1 are satisfied. It cannot be the case that the last column equals zero since this would have implied $c_2 = 0$ and hence $a_{2,1} = 0$ and the method reduces to one with fewer stages. Therefore we must have $\hat{b}_i = b_i$, $i = 2, 3, 4$ and, together with consistency, this contradicts the requirement that $\hat{b}_i$, b_i are distinct. □

A consequence of this proof is that no 4(3) explicit RK pair with 4-stages exists in the standard case.

4.6 A class of embedded schemes

A natural idea, which follows from embedded pairs of Fehlberg, Sarafyan and Dormand and Prince, is to choose a p-stage RK method of Q-order p that embeds a $(p-1)$-stage method of Q-order $(p-1)$. We now demonstrate that this is impossible.

Theorem 4.3 *No embedded $p(p-1)$ RKQ methods with p and $p-1$ stages exist for $p \geq 3$.*

Proof 1 Let $[t]_n$ denote the tree t rooted n times. Now form

$$\Phi([\tau^2]_{p-2}) - c_2\Phi([\tau]_{p-2}) = \Pi c_3(c_3 - c_2) = \frac{1}{p!}(2 - pc_2). \tag{4.6}$$

where $\Pi \neq 0$ denotes a product of terms, the precise form of which is not important. Now consider the order condition corresponding to $[\tau]_{p-1}$, namely:

$$\Pi a_{3,2}c_2 = \frac{1}{p!}. \tag{4.7}$$

Substituting (4.7) into (4.6) we can solve for an explicit value of $a_{3,2}$, namely

$$a_{3,2} = \frac{c_3(c_3 - c_2)}{c_2(2 - pc_2)}. \tag{4.8}$$

Comparison of the expression (4.8) for order p and $p-1$ affirms at once that they are contradictory, unless $c_2 \notin \{0, 2/(p-1), 2/p\}$ and either $c_3 = c_2$ or $c_3 = 0$. These values are forbidden; the former implies the method is reducible (to a method with fewer stages) whereas the latter violates invertibility of the equations corresponding to $[\tau]$ and $[\tau^2]$. We deduce that no such embedding is possible. $\square$

In fact the previous result holds for all explicit RK methods, but of course there is little point in seeking an embedded explicit RK method with $p = s \geq 5$ in the standard case.

5 SDIRK with 4 stages

In the present section we prove that it is impossible to have a Q-order 5, 4-stage SDIRK method. Hence, no improvement takes place *vis-á-vis* classical SDIRK methods.

We consider the method

$$\begin{array}{c|cccc} c_1 & c_1 & 0 & 0 & 0 \\ c_2 & c_2 - c_1 & c_1 & 0 & 0 \\ c_3 & c_3 - c_1 - a_{3,2} & a_{3,2} & c_1 & 0 \\ c_4 & c_4 - c_1 - a_{4,2} - a_{4,3} & a_{4,2} & a_{4,3} & c_1 \\ \hline & b_1 & b_2 & b_3 & b_4 \end{array} \tag{5.1}$$

and the Q-order 5 conditions (modulo lower order) are

$$b_1 + b_2 + b_3 + b_4 = 1, \tag{5.2}$$
$$b_1\, c_1 + b_2\, c_2 + b_3\, c_3 + b_4\, c_4 = \tfrac{1}{2}, \tag{5.3}$$
$$b_1\, c_1^2 + b_2\, c_2^2 + b_3\, c_3^2 + b_4\, c_4^2 = \tfrac{1}{3}, \tag{5.4}$$
$$b_3a_{3,2}(c_2 - c_1) + b_4(a_{4,2}(c_2 - c_1) + a_{4,3}(c_3 - c_1)) = \tfrac{1}{6} - c_1 + c_1^2, \tag{5.5}$$
$$b_3c_3a_{3,2}(c_2 - c_1) + b_4c_4(a_{4,2}(c_2 - c_1) + a_{4,3}(c_3 - c_1)) = \tfrac{1}{8} - \tfrac{2}{3}c_1 + \tfrac{1}{2}c_1^2, \tag{5.6}$$
$$b_3a_{3,2}(c_2^2 - c_1^2) + b_4(a_{4,2}(c_2^2 - c_1^2) + a_{4,3}(c_3^2 - c_1^2)) = \tfrac{1}{12} - \tfrac{1}{3}c_1 - \tfrac{1}{2}c_1^2 + c_1^3, \tag{5.7}$$
$$\begin{aligned} &b_3a_{3,2}c_1(c_2 - c_1) + b_4c_1(a_{4,2}(c_2 - c_1) + a_{4,3}(c_3 - c_1)) + \tfrac{1}{3}b_4a_{4,3}a_{3,2}(c_2 - c_1) \\ &\quad = \tfrac{1}{72} - \tfrac{1}{2}c_1^2 + \tfrac{2}{3}c_1^3, \end{aligned} \tag{5.8}$$
$$b_3c_3(c_2^2 - c_1^2)a_{3,2} + b_4c_4(a_{4,2}(c_2^2 - c_1^2) + a_{4,3}(c_3^2 - c_1^2)) \tag{5.9}$$

$$= \tfrac{1}{15} - c_1(b_1c_1^3 + b_2c_2^3 + b_3c_3^3 + b_4c_4^3) - \tfrac{1}{3}c_1^2 + \tfrac{1}{2}c_1^3,$$

$$b_3c_3a_{3,2}c_1(c_2 - c_1) + b_4c_4c_1(a_{4,2}(c_2 - c_1) + a_{4,3}(c_3 - c_1)) + \tfrac{1}{3}b_4c_4a_{4,3}a_{3,2}(c_2 - c_1) = \tfrac{1}{30} - c_1^2 + c_1^3, \tag{5.10}$$

$$b_1c_1^4 + b_2(2c_2c_1 - c_1^2)^2 + b_3(2c_3c_1 - c_1^2 + (c_2 - c_1)a_{3,2})^2 + b_4(2c_4c_1 - c_1^2 + a_{4,2}(c_2 - c_1) + a_{4,3}(c_3 - c_2))^2 = \tfrac{1}{20} \tag{5.11}$$

$$b_3a_{3,2}(c_2 - c_1)c_1(c_1 + 2c_2 + c_3) + b_4a_{4,2}(c_2 - c_1)c_1(c_1 + 2c_2 + c_4) + b_4a_{4,3}(c_3 - c_1)c_1(c_1 + 2c_3 + c_4) + b_4a_{4,3}c_3a_{3,2}(c_2 - c_1) = \tfrac{1}{40} - \tfrac{2}{3}c_1^2 + c_1^4, \tag{5.12}$$

$$b_3a_{3,2}(c_2 - c_1)c_1(3c_1 + 2c_2) + b_4a_{4,2}(c_2 - c_1)c_1(3c_1 + 2c_2) + b_4a_{4,3}(c_3 - c_1)c_1(3c_1 + 2c_3) + b_4a_{4,3}a_{3,2}(c_2^2 - c_1^2) = \tfrac{1}{60} - \tfrac{1}{3}c_1^2 - c_1^3 + 2c_1^4, \tag{5.13}$$

$$b_3a_{3,2}(c_2 - c_1)c_1^2 + b_4a_{4,2}(c_2 - c_1)c_1^2 + b_4a_{4,3}(c_3 - c_1)c_1^2 + \tfrac{2}{3}b_4a_{4,3}a_{3,2}(c_2 - c_1)c_1 = \tfrac{1}{720} - \tfrac{1}{3}c_1^3 + \tfrac{1}{2}c_1^4. \tag{5.14}$$

Our purpose is to prove that (5.2)–(5.14) possesses no solution. To that end we define

$$D_1 := b_3a_{3,2}(c_2 - c_1), \quad D_2 := b_4a_{4,2}(c_2 - c_1), \quad D_3 := b_4a_{4,3}(c_3 - c_1)$$

and $E := b_4a_{4,3}a_{3,2}(c_2 - c_1)$.

Firstly, we multiply (5.5) by c_1 and subtract from (5.8). This results in

$$E = \tfrac{1}{24} - \tfrac{1}{2}c_1 + \tfrac{3}{2}c_1^2 - c_1^3. \tag{5.15}$$

Next, comparing the left-hand sides,

$$2 \times (5.7) + 3c_1 \times (5.5) + c_1 \times (5.15) + c_2E = (5.13),$$

therefore

$$c_2E = \tfrac{1}{60} - \tfrac{5}{24}c_1 + \tfrac{2}{3}c_1^2 - \tfrac{1}{2}c_1^3. \tag{5.16}$$

Likewise,

$$c_1\left(c_1 \times (5.5) + (5.6) + 2 \times (5.7)\right) + c_3E = (5.12)$$

implies

$$c_3E = \tfrac{1}{40} - \tfrac{7}{24}c_1 + \tfrac{5}{6}c_1^2 - \tfrac{1}{2}c_1^3 \tag{5.17}$$

and

$$c_1 \times (5.6) + \tfrac{1}{3}c_4E = (5.10)$$

results in

$$c_4E = \tfrac{1}{30} - \tfrac{3}{8}c_1 + c_1^2 - \frac{1}{2}c_1^3. \tag{5.18}$$

Next we observe that

$$c_1^2 \times (5.5) + \tfrac{2}{3}c_1E = (5.14),$$

hence

$$c_1E = \tfrac{1}{480} - \tfrac{1}{4}c_1^2 + c_1^3 - \tfrac{3}{4}c_1^4.$$

Comparison with (5.15) results in the quartic equation

$$c_1^4 - 2c_1^3 + c_1^2 - \tfrac{1}{6}c_1 + \tfrac{1}{120} = 0, \tag{5.19}$$

which restricts the choice of c_1 to just four values.

Note that (5.19) could have been deduced from the standard SDIRK theory of Nørsett and Crouzeix [8], since it is identical to $L_4^{(1)}(c_1^{-1}) = 0$, the order-5 condition for the 4th restricted Padé approximant. This condition survives upon the passage to quadratic equations, since it originates in order conditions for linear equations. However, our derivation is much more than a simple check, since it also provides a means of determining c_2, c_3, c_4.

Given c_1 that solves (5.19),[7] we use (5.16)–(5.18) for a closed-form expression for $\mathbf{c}$. We also exploit (5.5)–(5.7) to determine D_1, D_2, D_3, as a solution of a linear system – e.g.,

$$D_1 = \frac{\left(\frac{1}{6} - c_1 + c_1^2\right)\left(\frac{1}{2} - c_2\right)}{c_3 - c_2}.$$

Next it follows from the definition of E and D_3 that

$$a_{3,2} = \frac{(c_3 - c_1)E}{(c_2 - c_1)D_3}.$$

Moreover, the definition of D_1 results in

$$b_3 = \frac{D_1}{a_{3,2}(c_2 - c_1)}.$$

The weight b_3 being known, we exploit (5.2)–(5.4) to derive b_1, b_2, b_4 as a solution of a linear system. Finally,

$$a_{4,2} = \frac{D_2}{b_4(c_2 - c_1)}, \qquad a_{4,3} = \frac{D_3}{b_4(c_3 - c_1)}$$

and $a_{k,1}$, $k = 1, 2, 3, 4$, are obtained from the condition $A\mathbf{1} = \boldsymbol{c}$. For each of the four zeros of (5.19) this results in a unique choice of the RK matrix A and the RK weights $\mathbf{b}$. By construction, all the Q-order 5 conditions are automatically satisfied, except for (5.9) and (5.11). Checking on a case-by-case basis verifies that (5.9) is, in fact, satisfied as well. Unfortunately, this is not the case with (5.11), which is violated for all permitted choices of c_1. We deduce that no such Q-order 5 method is possible.

6 Runge–Kutta–Nyström methods for QODEs

So far, we have discussed numerical methods for first order ODEs of the form (1.1). The same argument can be extended to schemes for second-order differential equations

$$\boldsymbol{y}''(t) = \boldsymbol{f}(t, \boldsymbol{y}(t)), \quad t \geq 0, \quad \boldsymbol{y}(t_0) = \boldsymbol{y}_0, \quad \boldsymbol{y}'(t_0) = \boldsymbol{y}_0' \tag{6.1}$$

where the function $\boldsymbol{f}$ is quadratic in its elements. Examples of such systems include the double harmonic oscillator and the Hénon–Heiles Hamiltonian, as well as certain models arising in biology [2]. Such ODEs can be solved by Runge–Kutta–Nyström (RKN) methods.

In general, order conditions for RKN methods can be represented by rooted special Nyström trees (SN-trees) [9]. When the function $\boldsymbol{f}$ in (6.1) is quadratic, some of the elementary differentials in the expansion of higher derivatives of $\boldsymbol{y}$ vanish and this results in a reduction of order conditions. The order conditions for such QODEs can be represented by binary SN-trees as shown in Table 3.

Definition 6.1 *A rooted Special Nyström tree of order q, where q denotes the number of vertices, is a mapping*

$$t : A_q \setminus \{j\} \to A_q$$

where A_q consists of fat and meagre vertices with the root at j satisfying the following:

[7] All these zeros are positive and distinct, as they should be: after all, $L_4^{(1)}$ is an orthogonal polynomial, supported in $[0, \infty)$.

1. *the root of t at j is always fat,*
2. *a meagre vertex has at most one child and such a child must be fat,*
3. *a fat vertex has only meagre children.*

If all vertices of SNR-trees have at most two children then the SNR-tree is binary. Table 3 depicts binary SNR-trees up to order six. RKN methods satisfying order conditions corresponding to binary SN-trees will be henceforth called RKN methods of Q-order, consistently with our terminology for RK methods (in Section 4.1). In the figures in Table 3 black dots represent the fat vertices, the white dots the meagre ones. In the table, order conditions corresponding to each tree should be written as ([9])

$$\sum_i b_i \Phi_i(t) = \frac{1}{\gamma(t)}.$$

Note that the above is true only if $\bar{b}_i = b_i(1-c_1)$, $i = 1, \ldots, s$. $\Phi(t)$ is a polynomial in $a_{i,j}$'s and c_i's and is written according to the following rules:

- To each pair of consecutive fat vertices labeled i, j associate $a_{i,j}$.
- To each terminal meagre vertex labeled i associate c_i.

Thus there are as many summation indices as fat vertices. Table 4 shows the number of rooted SNR-trees and rooted binary SNR-trees. The densities γ are the same as for the standard case. Note that the number of rooted binary SNR-trees is growing at a much smaller rate asymptotically. It is interesting to give the difference "parameters–equations". This measure was given in tabular form for RKN and symplectic RKN methods in [7] and we reproduce the format here. Table 5 gives the difference for explicit RKN and RKNQ methods. A blank entry indicates a deficit. The entries provide an indication that methods of certain class exist. However, this can be misleading since, for example, even though solutions may exist, they may not be real-valued.

Instead of exploiting these conditions to derive more accurate RKNQ methods we look at symplectic methods of this form. The reason is that all the examples of second order QODEs mentioned at the beginning of this section are separable Hamiltonians [18], hence it makes sense to solve them by using symplectic solvers [21].

6.1 Symplectic RKNQ methods

Symplectic methods are numerical methods which, when applied to Hamiltonian systems, preserve the symplecticity of solutions. These methods are advantageous when applied to Hamiltonian systems: we refer the interested reader to [21] for a discussion of this issue. Suppose an s-stage RKN method given by

$$\begin{aligned} \boldsymbol{\xi}_i &= \boldsymbol{y}_n + h\, c_i \boldsymbol{y}'_n + h^2 \sum_{j=1}^{s} a_{i,j} \boldsymbol{f}(\boldsymbol{\xi}_j), \qquad i = 1, \ldots, s, \\ \boldsymbol{y}'_{n+1} &= \boldsymbol{y}'_n + h \sum_{j=1}^{s} b_j \boldsymbol{f}(\boldsymbol{\xi}_j), \\ \boldsymbol{y}_{n+1} &= \boldsymbol{y}_n + h \boldsymbol{y}'_n + h^2 \sum_{j=1}^{s} \bar{b}_j \boldsymbol{f}(\boldsymbol{\xi}_j). \end{aligned} \tag{6.2}$$

The above method is symplectic if (Sanz-Serna, Lasagni and Suris [21])

$$\begin{aligned} \bar{b}_i - b_i + b_i c_i &= 0, \qquad i = 1, 2, \ldots, s, \\ b_i a_{i,j} - b_j a_{j,i} - b_i \bar{b}_j + b_j \bar{b}_i &= 0, \qquad i = 1, \ldots, j-1, \; j = 2, \ldots, s. \end{aligned} \tag{6.3}$$

Table 3: Order conditions for RKNQ of Q-order ≤ 6.

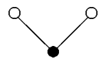
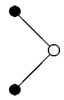
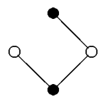
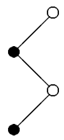
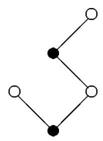
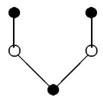
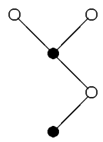
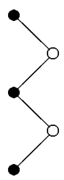
order	t	graph	$\Phi_j(t)$	$\gamma(t)$	$\sigma(t)$
1	t		1	1	1
2	t_1		c_j	2	1
3	t_2		c_j^2	3	2
	t_3		$\sum_k a_{j,k}$	6	1
4	t_4		$\sum_k a_{j,k} c_j$	8	1
	t_5		$\sum_k a_{j,k} c_k$	24	1
5	t_6		$\sum_k c_j a_{j,k} c_k$	30	1
	t_7		$\sum_{k,\ell} a_{j,k} a_{j,\ell}$	20	2
	t_8		$\sum_k a_{j,k} c_k^2$	60	2
	t_9		$\sum_{k,\ell} a_{j,k} a_{k,\ell}$	120	1

Table 3: (contd) Order conditions for RKNQ of Q-order ≤ 6.

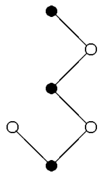
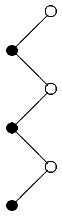
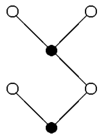
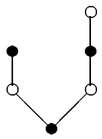
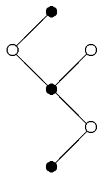
order	t	graph	$\Phi_j(t)$	$\gamma(t)$	$\sigma(t)$
6	t_{10}		$\sum_{k,\ell} c_j a_{j,k} a_{k,\ell}$	144	1
	t_{11}		$\sum_{k,\ell} a_{j,k} a_{k,\ell} c_\ell$	720	1
	t_{12}		$\sum_{k} c_j a_{j,k} c_k^2$	72	2
	t_{13}		$\sum_{k,\ell} a_{j,k} a_{j,\ell} c_\ell$	72	1
	t_{14}		$\sum_{k,\ell} a_{j,k} c_k a_{k,\ell}$	240	1

Table 4: The number of rooted trees for RKN and RKNQ methods.

Order	1	2	3	4	5	6	7	8	9	10
Rooted SNR-Trees	1	1	2	3	6	10	20	36	72	137
Rooted Binary SNR-Trees	1	1	2	2	4	5	9	13	23	35

Table 5: The difference "parameters–equations" for explicit RKN and RKNQ methods.

Method	p/s	1	2	3	4	5	6
RKN	1	1	4	8	13	19	26
	2	0	3	7	12	18	25
	3	-	1	5	10	16	23
	4	-	-	2	7	13	20
	5	-	-	-	1	7	14
	6	-	-	-	-	-	4
RKNQ	1	1	4	8	13	19	26
	2	0	3	7	12	18	25
	3	-	1	5	10	16	23
	4	-	-	3	8	14	21
	5	-	-	-	4	10	17
	6	-	-	-	-	5	12

The converse is true for RKN with non-redundant stages and hence we say that RKN methods satisfying (6.3) are symplectic [21]. Any explicit symplectic RKN without redundant stages can be written in the following formulation [16],

$$
\begin{array}{c|cccccc}
c_1 & 0 & & & & & \\
c_2 & b_1(c_2-c_1) & 0 & & & & \\
c_3 & b_1(c_3-c_1) & b_2(c_3-c_2) & 0 & & & \\
\vdots & \vdots & \vdots & \ddots & \ddots & & \\
c_s & b_1(c_s-c_1) & b_2(c_s-c_2) & \cdots & b_s(c_s-c_{s-1}) & 0 \\
\hline
 & b_1 & b_2 & \cdots & b_{s-1} & b_s
\end{array}
.
$$

The advantage of this form is that implementations require the minimal possible storage of $2\,s$-dimensional vectors.

When a RKN method is symplectic its coefficients are no longer independent, as is evident from the above formulation. Sanz-Serna and Calvo [21] proved that the number of order conditions for symplectic RKN methods corresponds to the number of (unrooted) SN-trees, where an SN tree is an equivalence class of SNR-trees. Table 6 displays the reduction in the number of order conditions for RKNQ and symplectic RKNQ methods and also gives the values for RKN and SRKN for comparison. These numbers are given some formal justification in the next section.

The number of RKNQ order conditions is already less than the number for symplectic RKN methods. There is no difference in the number of order conditions for symplectic RKN and symplectic RKNQ methods through order three. At order four, the condition corresponding to one special Nyström rooted tree is redundant for a symplectic RKNQ method, which leaves us with five order conditions. A considerable reduction in the number of order conditions can be seen to take place at higher orders.

Table 6: The number of order conditions for RKN, RKNQ, SRKN and SRKNQ methods.

Order	1	2	3	4	5	6	7	8	9	10
RKN	1	2	4	7	13	23	43	79	151	288
RKNQ	1	2	4	6	10	15	24	37	60	95
SRKN	1	2	4	6	10	15	25	39	66	109
SRKNQ	1	2	4	5	8	11	16	22	32	45

6.2 Binary Nyström tree enumeration

Calvo and Sanz-Serna showed how to enumerate m_n, the number of special Nyström trees in [7]. The idea is as follows. Consider a special Nyström tree of order n. This gives rise to, say, j_k graphs with k vertices. For $k \geq 2$ each among the j_k graphs with k vertices consists of a meagre vertex (that was a child of the root in the original tree) together with a special Nyström tree of order $k-1$. Hence for $k \geq 2$ the j_k graphs with k vertices can be chosen in

$$\binom{m_{k-1}+j_k-1}{j_k}$$

different ways. There is only one way of choosing the j_1 graphs with one vertex and hence the resulting enumeration formula is given by

$$m_n = \sum_{j_1+2j_2+\ldots+kj_k=n-1} \binom{m_1+j_2-1}{j_2}\binom{m_2+j_3-1}{j_3}\cdots\binom{m_{k-1}+j_k-1}{j_k}. \tag{6.4}$$

This formula can be used to compute m_n recursively. The range of the summation index j_i is $i=0$ to $[(n-1)/i]$. This looks quite complicated, but it is merely restating that the terms range over the partitions of $(n-1)$ – a point which we have remarked upon in greater detail in Section 3.

Insofar as number m_n^B of binary special Nyström trees is concerned, we have an additional restriction. If the tree that remains after deletion of the meagre vertex has order $k=(n-1)$ then it has multiplicity 1 and the last term of the sum (6.4) simplifies to obtain the value 1 or m_k^B. Otherwise exactly two products are involved in the summation. This leads us to

$$m_n^B = \sum_{\substack{j_1+2j_2+\ldots+kj_k=n-1\\ \sum_{i=1}^{n-1} j_i \leq 2}} \binom{m_1^B+j_2-1}{j_2}\binom{m_2^B+j_3-1}{j_3}\cdots\binom{m_{k-1}^B+j_k-1}{j_k}. \tag{6.5}$$

The range of summation is now over the partitions of $(n-1)$ into two parts together with a single term when the index is j_i, $i=(n-1)$. Each binomial expression in equation (6.5) thus attains one of the values

$$\{1,\ m_{j_i}^B,\ \tfrac{1}{2}m_{j_i}^B(1+m_{j_i}^B)\} \quad i=2,\ldots,n-2$$

and the single value 1 for $i=n-1$. Combining these facts, we have arrived at a very efficient means of recursively computing the number of binary special Nyström trees. The resulting quantities at each order through order 10 are displayed in Table 4.

It should be possible to modify the derivation of Calvo and Sanz-Serna in [7] to obtain an expression for the number of binary non-rooted special Nyström trees. These correspond to the number of order conditions for symplectic Q-order Runge–Kutta Nyström methods. We have not investigated this however. Instead, since the number of trees is now so small, it is an easy task to calculate the trees by hand through order 10. The resulting enumeration is given implicitly in Table 6 where the number of order conditions through order 10 is displayed.

6.3 The principal error and order of attainability

In order to compare various methods we use the weighted principal local truncation error under the Euclidean norm,

$$\sqrt{\sum_{t \in NQ} \left(\frac{1}{\sigma(t)} \left(\Phi(t) - \frac{1}{\gamma(t)} \right) \right)^2}.$$

The weight of each order condition is the symmetry, $\sigma(t)$, of the corresponding Nyström binary tree, $t \in NQ$. We will refer to this weighted error as the Q_2-error.

In our search for SRKNQ methods we have also found it useful to consider *symmetric* (also sometimes termed self-adjoint) schemes [8, definition 8.7]. Such methods provide their own inverse upon time reversal. This property has the advantage that, although the number of free parameters is roughly halved, all even order conditions in the local error expansion are automatically satisfied. An s-stage SRKNQ method is symmetric if

$$b_i = b_{s-i}, \quad c_i = 1 - c_{s-i}, \qquad i = 1, \ldots, s.$$

We will only consider irreducible methods, $b_i \neq 0$ for all i, so that the number of stages is minimal. Explicit SRKN schemes are completely characterised by $2\,s$ free parameters. In Table 7 we consider the difference "parameters − equations" for SRKN and SRKNQ methods (the latter for both standard and symmetric schemes). Symmetric methods have $c_{s/2} = \frac{1}{2}$ for odd s and hence we should not count it as a free parameter.

6.4 Specific RKNQ methods

Suppose that we wish to consider a three stage method, then the conditions for Q-order four can be written in terms of b_i and c_i as:

$$\begin{aligned} b_1 + b_2 + b_3 &= 1 \\ b_1\, c_1 + b_2\, c_2 + b_3\, c_3 &= \tfrac{1}{2} \\ b_1\, c_1^2 + b_2\, c_2^2 + b_3\, c_3^2 &= \tfrac{1}{3} \\ b_1\, b_2\, (c_2 - c_1) + b_3\, (b_1\, (c_3 - c_1) + b_2\, (c_3 - c_2)) &= \tfrac{1}{6} \\ b_1\, b_2\, c_2\, (c_2 - c_1) + b_3\, c_3 (b_1\, (c_3 - c_1) + b_2\, (c_3 - c_2)) &= \tfrac{1}{8} \end{aligned} \tag{6.6}$$

The first three equations can be solved for b_i's, which leaves us with three parameters and two remaining equations to solve for. We have three more additional equations to solve for order five. We minimize the three conditions for order five subject to the last two conditions in (6.6) vanishing.

Using an `IMSL` subroutine for minimisation, the best we could obtain is the first method in Table 8, whose Q_2-error is 0.00571508. Using Mathematica it is possible to derive the optimal method in this class in a closed form. The optimal method occurs near $c_3 = 0.76$ and has Q_2-error of 0.0029085. However, the weights b_i of this method are rather large. This is computationally undesirable because it results in large bounds on rounding errors at each step. Instead, the second method in Table 8 possesses relatively simple rational coefficients and is only around 33% in excess of the minimum. The Q_2-error of this method is 0.00389689.

For a method to have order five, one is left with eight equations. If we consider a four stage symplectic RKNQ method, then there are eight equations in eight unknowns. We tried to solve

Table 7: The difference "parameters-equations" for explicit SRKN and SRKNQ methods.

Method	p/s	1	2	3	4	5	6
SRKN	1	1	3	5	7	9	11
	2	0	2	4	6	8	10
	3	-	0	2	4	6	8
	4	-	-	0	2	4	6
	5	-	-	-	-	0	2
	6	-	-	-	-	-	-
SRKNQ	1	1	3	5	7	9	11
	2	0	2	4	6	8	10
	3	-	0	2	4	6	8
	4	-	-	1	3	5	7
	5	-	-	-	0	2	4
	6	-	-	-	-	-	1
SRKNQ (symmetric)	2	1	1	2	3	4	5
	4	-	-	0	1	2	3
	6	-	-	-	-	-	0

Table 8: 3 stage SRKNQ methods of order 4

b_1	0.4321783884695422637469	c_1	0.1569296691827464175187
b_2	-0.1356432230609154725062	c_2	1.0
b_3	0.7034648345913732087593	c_3	0.8071783884695422637469
b_1	11/24	c_1	2/11
b_2	11/12	c_2	8/11
b_3	-3/8	c_3	2/3

Table 9: 5 stage symmetric SRKNQ methods of order 4

b_1	-0.095496309138422121	c_1	0.260690358844585134
b_2	0.366594381355683831	c_2	0.141423461963184371
b_3	0.457803855565476580	c_3	0.5
b_4	0.366594381355683831	c_4	0.858576538036815629
b_5	-0.095496309138422121	c_5	0.739309641155414866
b_1	-0.06435293417353206365	c_1	0.3137678571364566410
b_2	0.3283290714757769551	c_2	0.1343457928582398283
b_3	0.4720477253955102171	c_3	0.5
b_4	0.3283290714757769551	c_4	0.8656542071417601717
b_5	-0.06435293417353206365	c_5	0.6862321428635433590

these using the **IMSL** subroutine **DNEQNF** without success. The full equations proved intractable using analytic solution techniques, although it is relatively straightforward to demonstrate that no 4-stage symmetric scheme of order 5 exists, since the only solutions satisfy $c_1 + c_2 = 1$, which violates solutions of the order conditions corresponding to $[\tau]$ and $[\tau^2]$ in terms of b_1, b_2. The optimal 4-stage symmetric order 4 method has Q_2-error 0.0029176 which is not an improvement over the 3-stage method described above.

Let us instead consider a five stage symmetric symplectic RKNQ method. We can prove (again symbolically) that no such real-valued 6th order methods exist. Using six stages we have two free parameters (see Table 7) which we use to minimise the Q_2-error. Using the **IMSL** routine we obtained the coefficients for the first method in Table 9. The method was rederived and the coefficients refined using Mathematica. The Q_2-error of this method is 0.000173694. The following hybrid symbolic-numeric solution process yields a scheme with Q_2-error roughly half that of the previous method, namely 0.0000914326. First we reduce the order conditions using symmetry. This fixes b_4, b_5, c_4, c_5 in terms of b_2, b_1, c_2, c_1 respectively (of course, $c_3 = \frac{1}{2}$). Then we solve the first and third (linear) equations of (6.6) to obtain b_1, b_2; the second and fifth relations being redundant by symmetry. We then obtain values from subdividing a grid such that $0 \leq c_1, c_2 \leq 1$ and substitute into the fourth equation in (6.6). Mathematica's **NRoots**, which currently uses the Jenkins–Traub method to solve univariate polynomials, can be used to find an appropriate value of b_3 (there are two such solutions since the equation is quadratic). The values obtained by this process guarantee an order 4 scheme. It is then a straightforward task to numerically evaluate a symbolic expression representing the Q_2-error and to select the minimum value.[8]

Table 7 reveals that there are insufficient free parameters to obtain a six stage, sixth order SRKN method. However, it is feasible that SRKNQ methods of this type do exist. Since some of the order conditions become equivalent due to symplecticity, we have eleven equations in twelve unknowns. We restricted our search to symmetric six stage methods, in which case the number of order conditions reduces to six and there are six unknowns. We again used both the **IMSL** subroutine **DNEQNF** and Mathematica to solve the nonlinear algebraic equations numerically. We used 10^7 random values such that $b_3 \in [-1, 1]$ and $c_1, c_2, c_3 \in [0, 1]$ and there are many such schemes, hence we chose the method with the least Q_2 error. The resulting method is presented in Table 10 and has Q_2 error 0.000217384. A Newton search was again used to polish the results

[8] Mathematica's **Compile** was used to perform this numerical evaluation efficiently.

Table 10: A 6th order 6-stage SRKNQ method.

b_1	0.37713739049618554124	c_1	0.11227085341577697644
b_2	0.282587770307696605093	c_2	0.646591806618089019634
b_3	-0.159725160803882146334	c_3	0.863479271151296544992
b_4	-0.159725160803882146334	c_4	0.136520728848703455008
b_5	0.282587770307696605093	c_5	0.353408193381910980366
b_6	0.37713739049618554124	c_6	0.88772914658422302356

through 20 significant digits.

6.5 Computational issues

The functions we are interested in are polynomials and there are several issues that should be considered for efficient and stable numerical computations. Here we discuss some general strategies and note that there may be more optimal techniques for specific problems. One primary concern is the cost of multiplications, which are the dominant computational component. Another is the effect of rounding errors arising from the finite representation of real numbers by granular machine arithmetic.

It is well known that a nested or *Horner*'s form for a polynomial has advantages over the expanded form.[9]

$$\begin{aligned} p(x) &= a_0\, x^n + \cdots + a_{n-1}\, x + a_n \\ &= (\cdots((a_0\, x + a_1)\ x + a_2)\ x + \cdots)\ x + a_n \end{aligned}$$

One advantage of Horner's form is that the work involved in exponentiation is distributed across addition and multiplication, resulting in savings of some basic arithmetic operations. Another advantage is that Horner's form is more stable to evaluate numerically when compared with the expanded form. The reason for this is that each sum and product involve quantities which vary on a more evenly distributed scale. The `Mathematica` version 3.0 package `NumericalMath‘Horner.m` can be used to rearrange polynomials in Horner's form.

When dealing with a system where each component is a polynomial, it makes sound computational sense to extract common multiplications across components and to perform these only once. Some optimizing compilers will try to do this, but in our experience, as expressions become large, it is not possible to rely on this behaviour. Most modern computer algebra systems also possess some form of expression optimization which could be used for this purpose. Such operations are often complementary to Horner's form.

See [13, section 4.6] for a discussion of issues involving polynomial arithmetic.

[9] This form is commonly attributed to W.G. Horner but dates back at least to the days of Newton.

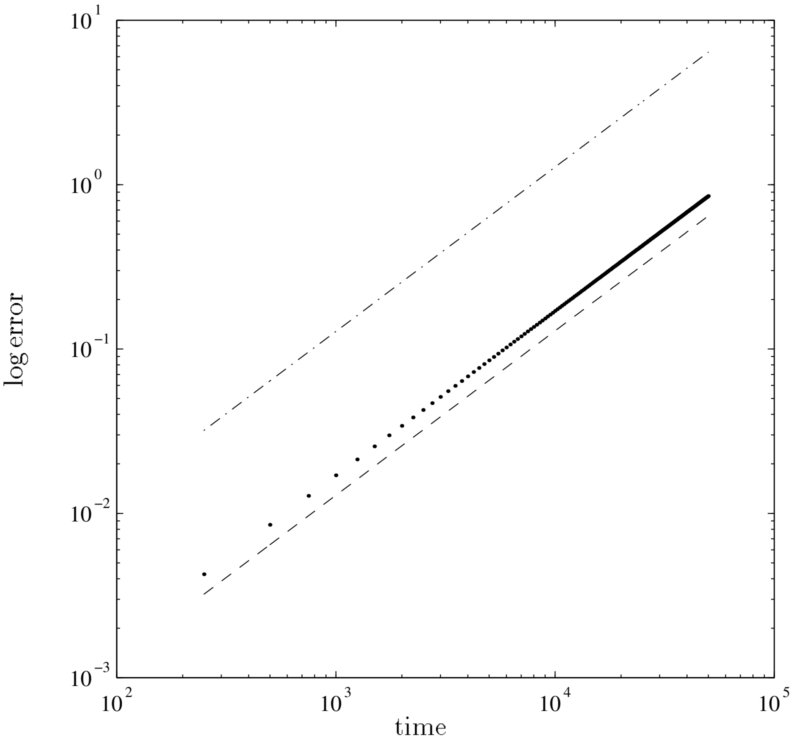


Figure 2 Error as a function of time for the Hamiltonian $H = \frac{1}{2}(p^2 + q^2) - \frac{1}{3}q^3$. Dash-dot, fat dot, dash and slim dot lines represent *OS3, SQ3, Calvo* and *SQ5* respectively, except that the errors of the last two methods are virtually indistinguishable.

6.6 Numerical experiments

We have tested the methods obtained in the last section, namely symplectic RKNQ methods for quadratic separable Hamiltonians, comparing them with conventional general symplectic RKN methods for separable Hamiltonians. Among the latter we have chosen 3-stage SRKN of order four derived by Okunbor and Skeel [16] to compare with the 3-stage SRKNQ method of order four with minimised error coefficients. Other methods used for testing are Calvo's [21] 5-stage SRKN method of order four, being compared with the 5-stage symplectic RKNQ method of order four that was derived in Section 6.1.

More specifically, the following methods have been employed in our numerical experiments.

1. A three-stage fourth order symmetric SRKN method of order four derived by Okunbor and Skeel (1992),

b_1	1.35120719195965763405	c_1	0.67560359597982881702
b_2	−1.70241438391931526809	c_2	0.5
b_3	1.35120719195965763405	c_3	0.32439640402017118297

henceforth labeled *OS3*. This is the only symmetric method of this class with real coefficients and has weighted error 0.119748.

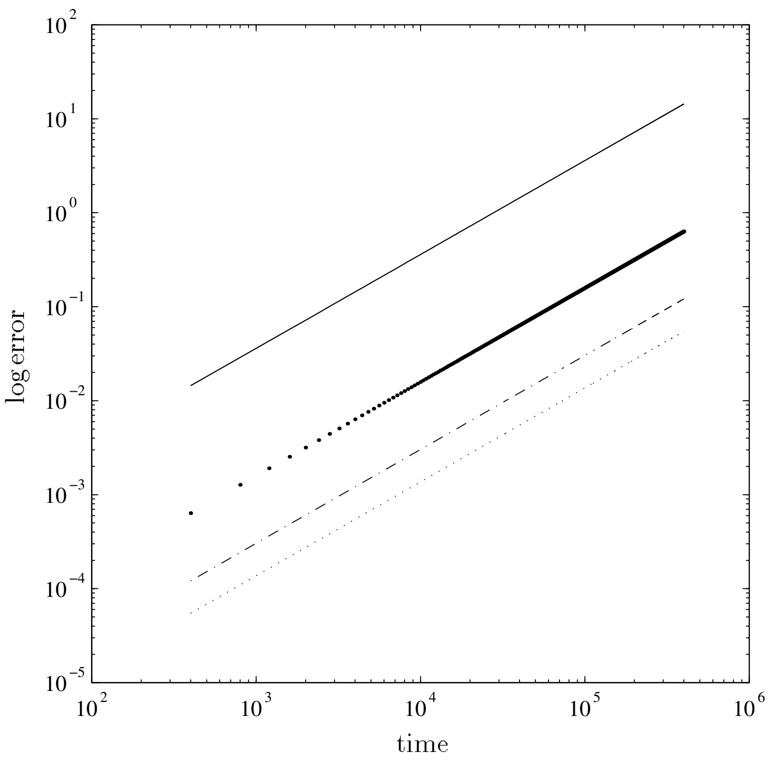


Figure 3 Error as a function of time for the Henon–Heiles Hamiltonian $H = \frac{1}{2}(p_1^2 + p_2^2 + q_1^2 + q_2^2) + q_1^2 q_2 - \frac{1}{3}q_2^3$.

2. A five stage RKN method of order four with minimised error coefficients derived by Calvo (1992) in her thesis and labeled *Calvo* [6]. This method is also FSAL (first stage same as last stage) and hence the number of function evaluations is reduced by one per step. This method is often considered the optimal symplectic RKN method available in the literature and has a weighted error 0.000756197.

b_1	0.0617588581356263250	c_1	0.0
b_2	0.3389780265536433551	c_2	0.2051776615422863869
b_3	0.6147913071755775662	c_3	0.6081989431465009739
b_4	-0.1405480146593733802	c_4	0.4872780668075869657
b_5	0.1250198227945261338	c_5	1.0

3. The three stage explicit symplectic RKNQ method of order four with optimised error coefficients (see Section 6.1) labeled *SQ3*.

4. The five stage explicit symplectic RKNQ method of order four with optimised error coefficients (see Section 6.1) labeled *SQ5*.

As in [21], we have measured the Euclidean norm of the error in both the $\boldsymbol{p}$ and $\boldsymbol{q}$ components. All the figures are drawn on a log-log scale. We denote OS3 by a dash-dot line, SQ3 by large dots, Calvo by a dash line and SQ5 by slim dots.

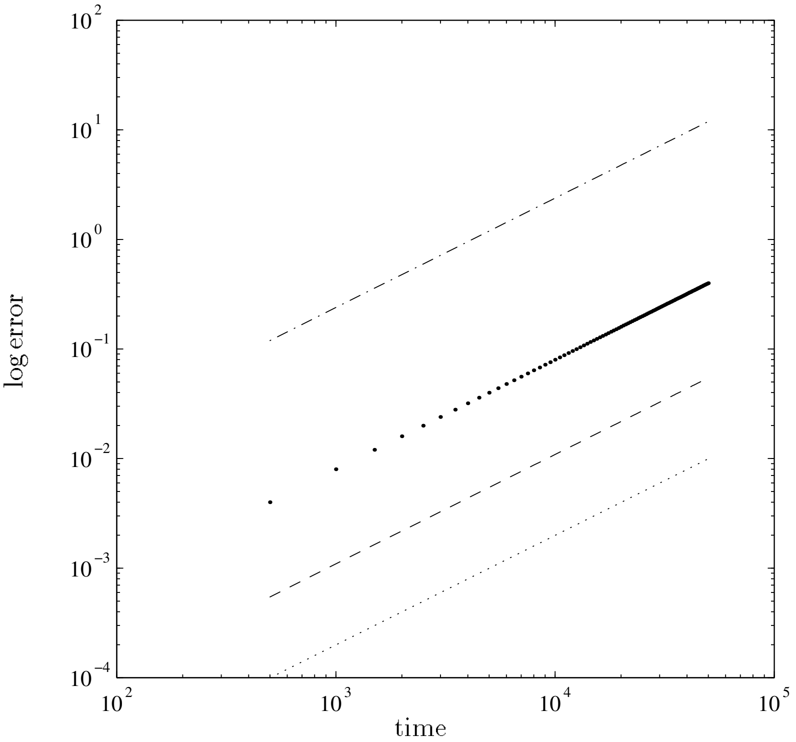


Figure 4 Error as a function of time for the Hamiltonian (6.7).

First we apply the OS3, SQ3 and Calvo methods to the Hamiltonian $H = \frac{1}{2}(p^2+q^2)-\frac{1}{3}q^3$. The initial conditions are $p(0) = \frac{1}{2}$, $q(0) = \frac{1}{5}$, the step-size is $h = \frac{1}{4}$ and the integration has been carried out in the interval $[0, 5 \times 10^4]$. The results have been displayed in Figure 2. While SQ3 performs much better than OS3 for this system, our SQ5 method performs as well as Calvo: their errors coincide to four significant digits. Moreover, the performance of these methods is only marginally better to the (cheaper) SQ3 scheme.

We again compare the performance of the four methods by applying them in Figure 3 to the Henon–Heiles Hamiltonian. The initial value used was $[0.12, 0.12, 0.12, 0.12]$, the step-size $h = \frac{1}{5}$ and the integration has been carried out in the interval $[0, 4 \times 10^5]$. As before, SQ3 performs better than OS3. Moreover, SQ5 produces smaller error than Calvo but the comparison is somewhat unfair since Calvo, being FSAL, requires one less function evaluation per step.

Finally, we address ourselves in Figure 4 to the Hamiltonian

$$H(\boldsymbol{p}, \boldsymbol{q}) = \frac{1}{2}\sum_{i=1}^{3} p_i^2 + \sum_{i=1}^{3} q_i^3 - \sum_{i=1}^{3} q_i, \tag{6.7}$$

with $\boldsymbol{p}(0) = [0.2, 0.4, 0.6]$, $\boldsymbol{q} = [0.1, 0.3, 0.5]$, $h = \frac{1}{10}$ and $t \in [0, 5 \times 10^4]$. The pecking order is clear: again, SQ3 clearly outperforms OS3, while among the five-stage methods SQ5 is definitely better than Calvo by a margin that outweighs the advantage of the latter in cheaper implementation.

7 Concluding remarks

For polynomial vector fields, Mrozek has recently demonstrated a rigourous symbolic technique for error analysis of numerical methods, and in particular of Runge–Kutta schemes [15]. The method is an alternative to interval arithmetic when efficiency and rigour are paramount, but tight bounds are not the main concern. The work outlined here simplifies the task of deriving such apriori estimates and makes it more efficient for certain systems by introducing several new classes of numerical methods.

We have not explored methods for mildly stiff problems. For such problems, explicit methods with enhanced stability characteristics may be of interest. The Mathematica version 3.0 package `NumericalMath'OrderStar'` can be used to assist in the analysis and construction of methods with extended stability domains [20].

It may well be that FSAL is the way to proceed for symplectic methods, since we never reject steps due to local error considerations. Hence we always obtain efficiency benefits of step reuse. The condition for FSAL symplectic RKN schemes is $c_1 = 0$, $c_s = 1$.

It seems, from the experiments of Section 6.6 that the Q_2-error gives a very good indication of the relative performance of methods. Could this be because functions and derivatives vary on a multiplicative scale, devoid of singularities and other types of ill-conditioned growth?

There remain many open problems in this area. For example, is there a 5 stage 5(4) embedded explicit QRK method?

Acknowledgements

The authors are grateful to John Butcher for bringing Lawson's paper [14] to their attention. The second author wishes to thank IBERDROLA for financial support.

References

[1] H. W. Becker (1949), "Genetic Algebra." A.M.M. **56**, 697–699.

[2] E. Beltrami (1987), *Mathematics for Dynamics Modelling,* Academic Press, San Diego.

[3] B. Buchberger (1970), "Ein Algorithmisches Kriterium für die Lösbarkeit eines algebraischen Gleichungssystems." *Æquationes Mathematicæ* **4**, 374–383.

[4] K. Burrage (private communication).

[5] J.C. Butcher (1987). *The Numerical Analysis of Ordinary Differential Equations,* John Wiley, Chichester.

[6] M.P. Calvo (1992), "Canonical Runge–Kutta–Nyström methods." Ph. D. Thesis, Universidad de Valladolid.

[7] M.P. Calvo and J.M. Sanz-Serna (1992), "Order conditions for canonical Runge–Kutta–Nyström methods." BIT, **32**, pp. 131–142.

[8] E. Hairer, S.P. Nørsett and G. Wanner (1993), *Solving Ordinary Differential Equations I: Nonstiff Problems* (2nd ed.), Springer-Verlag, Berlin.

[9] E. Hairer, and G. Wanner (1976), "A theory for Nyström methods", *Numer. Math.* **20**, 383-400.

[10] A. Iserles (1996), *A First Course in the Numerical Analysis of Differential Equations,* Cambridge University Press, Cambridge.

[11] D. Kapur and Y.N. Lakshman (1992), "Elimination methods: an introduction." in *Symbolic and Numerical Computation for Artificial Intelligence* (B. Donald, D. Kapur & J. Mundy, eds), Academic Press, New York.

[12] D.E. Knuth (1973), *Fundamental Algorithms* (2nd ed.), Addison Wesley, Reading, MA.

[13] D.E. Knuth (1981), *Seminumerical Algorithms* (2nd ed.), Addison Wesley, Reading, MA.

[14] J.D. Lawson (1970), "A note on Runge–Kutta processes for quadratic derivative functions", Proceedings of Combinatorics, Graph Theory and Computing, Louisana State Univ.

[15] M. Mrozek (1996), "Rigorous error analysis of numerical algorithms via symbolic computation", *J. Symb. Comp.* (to appear).

[16] D. Okunbor and R.D. Skeel (1992a), "Explicit Canonical methods for Hamiltonian systems", *Math. Comp.* **59**, 439–455.

[17] A. Nijenhuis and H. S. Wilf, (1978). *Combinatorial Algorithms*. 2nd Edition. Academic Press, New York.

[18] G.P. Ramaswami (1996), "Numerical solution of separable Hamiltonians", submitted as a doctoral thesis, University of Cambridge, England.

[19] M. Sofroniou (1994), "Symbolic derivation of Runge–Kutta methods", *J. Symb. Comp.* **18** (3), pp. 265–296.

[20] M. Sofroniou (1996), "Order stars and linear stability theory", *J. Symb. Comp.* (to appear).

[21] J.M. Sanz-Serna and M.P. Calvo (1994), *Numerical Hamiltonian Problems,* Chapman & Hall, London.

[22] M. Toda (1981), *Theory of Nonlinear Lattices*, Springer-Verlag, Berlin

[23] J.H. Van Lint and R.M. Wilson (1992), *A Course in Combinatorics,* Cambridge University Press, Cambridge.