Available online at http://scik.org

J. Math. Comput. Sci. 2 (2012), No. 4, 880-888

ISSN: 1927-5307

THE RELATIVE EFFICIENCY OF QUENCHING AND REINTEGRATION FOR GLOBAL ERROR CONTROL IN RUNGE-KUTTA METHODS

J.S.C. PRENTICE*

Department of Applied Mathematics, University of Johannesburg, South Africa

Abstract. We compare the efficiency of two algorithms for controlling global error in Runge-Kutta methods, i.e. reintegration and RKQ. We find that RKQ (which uses a quenching strategy) is, on average, more efficient than reintegration, by at least 16% or 30%, depending on the form of RKQ. Some numerical examples indicate that larger gains in efficiency are possible. The Runge-Kutta methods we consider are explicit, implicit (Radau, Lobatto, Gauss), embedded and Nyström.

Keywords: RKQ, RKrvQz, Quenching, Reintegration, Efficiency, Global error, Error control, Runge-Kutta.

2000 AMS Subject Classification: 65L05, 65L06, 65Y20

1. Introduction

We have recently reported on the quenching-based RKrvQz algorithm [1, 2, 3] for controlling the global error in Runge-Kutta (RK) methods in a stepwise manner (i.e. as the integration proceeds). The only other way of controlling the RK global error is through the use of a reintegration algorithm, wherein the RK method has to be reapplied with an appropriate stepsize, after the global error in its original application has been

*Corresponding author

E-mail address: jprentice@uj.ac.za (J.S.C. Prentice)

Received March 4, 2012

estimated. We consider a stepwise control of global error to be a better approach and so, in this paper, we compare RKQ and reintegration in terms of computational efficiency. If both are effective, the more efficient algorithm might be regarded as the preferred choice.

2. Relevant Concepts

Here, we briefly describe local error control via local extrapolation, reintegration and the RKQ algorithm. Our discussion will be brief; the reader is referred to the literature for more detail. This paper is a much more detailed discussion of ideas first studied in [4].

2.1 Local error control

Local extrapolation is the process whereby solutions using two RK methods, RKr and RKv, with $r < v$, are obtained at a node using the RKv solution at the previous node as input for both methods. The local error in the RKr solution is then estimated by means of the difference between these two solutions. If this estimate exceeds the desired tolerance, a new, smaller stepsize is determined and the local extrapolation process is repeated using this new stepsize [5]. If, however, the tolerance is satisfied, we simply move on to the next node. We will refer to such local error control via local extrapolation as LE(r, v).

2.2 Reintegration

Reintegration involves estimating the global error in the RKr solution, after it has been computed over the entire interval of integration. This is achieved using a method RKz of much higher order $(r < v \ll z)$; the difference between the two solutions gives the global error. If this error exceeds the desired tolerance, a new, smaller stepsize is determined. Usually, LE(r, v) has been implemented first, which results in a particular distribution of nodes. It is desirable to preserve this distribution so that, if a new stepsize is needed, we simply insert an integer number of nodes between the existing nodes. This yields a new node distribution with maximum stepsize smaller than that of the original. A new solution RKr is then obtained at these new nodes. This is the source of the term 'reintegration'. It is clear that is an *a posteriori* form of error control – the refinement occurs only after LE(r, v) has been applied over the entire interval of integration.

2.3 RKrvQz

In this algorithm, we apply either LE(r, v) or LE(r, z) to control local error. Knowledge of the RKz solution enables us to determine the global error that is propagated from the previous node and, hence, the total global error at the node of interest. If this global error exceeds the desired tolerance, we *quench* the RKr and RKv solutions – this simply involves replacing them with the RKz solution, which is assumed to be much more accurate. We then proceed to the next node. This algorithm provides *in situ* control of the global error – the global error is kept within the desired tolerance as the RK computation proceeds [1, 2, 3]. We will refer to the usage of LE(r, z) as *Mode I*, and the usage of LE(r, v) as *Mode II*. Using RKQ in Mode I provides a more accurate estimate of local error, whereas Mode II is more efficient (Mode I is the version of RKQ considered in [4]).

It is important to note that, if RKz is not reliable as a global error estimator, then neither reintegration nor RKQ will be successful as global error controllers. It is implicitly assumed in this work that RKz is reliable.

3. Efficiency Analysis

We will study efficiency in terms of the number of RK stage evaluations that are carried out. We assume that RKr requires s_r stage evaluations, and similarly for RKv and RKz. We assume that LE(r, v) or LE(r, z) results in a node distribution of $N + 1$ nodes, the first of which is the initial node.

3.1 Reintegration

The total number of stage evaluations required by the reintegration algorithm is

$$T_{\text{reint}} = N\left((p+2)\,s_r + s_v + s_z\right). \tag{1}$$

Here, p is the number of nodes that are inserted between each pair of adjacent nodes, due to a stepsize adjustment. The minimum value of p is one, so we can write

$$T_{\text{reint}} \geqslant N\left(3s_r + s_v + s_z\right). \tag{2}$$

This is the lower bound we will use in our analysis.

3.2 RKrvQz

The total number of stage evaluations required by the RKQ algorithm in Mode I is

$$T^I_{RKQ} = N\left(2s_r + s_v + s_z\right), \tag{3}$$

and, in Mode II,

$$T^{II}_{RKQ} = N\left(s_r + s_v + s_z\right). \tag{4}$$

3.3 Comparison

It is immediately clear that we would expect reintegration to be less efficient than RKQ, simply because of the extra stage evaluations that it needs. To obtain a quantitative indication of the relative efficiency of the two methods, we will determine

$$R_I \equiv \frac{2s_r + s_v + s_z}{3s_r + s_v + s_z} \tag{5}$$

and

$$R_{II} \equiv \frac{s_r + s_v + s_z}{3s_r + s_v + s_z} \tag{6}$$

for a variety of RK methods.

4. Comparison of Methods

We consider explicit RK methods, explicit embedded RK methods, implicit RK methods (Radau, Lobatto, Gauss) and Nyström methods. Notation, order, number of stages and type are indicated in Table 1. The reader is referred to [5], [6] and [7] for a description of these (and other) RK methods.

Method	Type	Order	Stages
RK2	Explicit	2	2
RK4	Explicit	4	4
RK5	Explicit	5	6
RK8	Explicit	8	13
RK10	Explicit	10	17
RK$(3,4)$	Explicit, embedded	3 and 4	5
RK$(4,5)$	Explicit, embedded	4 and 5	6
RKN4	Nyström	4	3
RKN5	Nyström	5	4
RKN10	Nyström	10	17
Method	**Type**	**Order**	**Stages**
RKL2	Implicit, Lobatto III	2	2
RKL4	Implicit, Lobatto IIIA	4	3
RKL8	Implicit, Lobatto IIIB	8	5
RKR3	Implicit, Radau IIA	3	2
RKR5	Implicit, Radau IA	5	3
RKG4	Implicit, Gauss	4	2
RKG6	Implicit, Gauss	6	3
RKG8	Implicit, Gauss	8	4
RKG10	Implicit, Gauss	10	5

Table 1. Methods used in the efficiency comparison.

4.1 Explicit RK methods

Here, we consider $r = 3, 4$ $(s_r = 3, 4)$, $v = 4, 5$ $(s_v = 4, 6)$ and $z = 8, 10$ $(s_z = 13, 17)$. These correspond to the methods RK3, RK4, RK5, RK8 and RK10. Values of T^I_{RKQ}, T^{II}_{RKQ}, T_{reint}, R_I and R_{II} are shown in Table 1, where T_{reint} is the lower bound in (2). The value in parentheses in the T_{reint} column is the value of T_{reint} assuming no reintegration is necessary (i.e. LE fortuitously resulted in an acceptably small global error). T^I_{RKQ}, T^{II}_{RKQ} and T_{reint} are in units of N. These also hold for the other tables in the paper.

s_r	s_v	s_z	T^I_{RKQ}	T^{II}_{RKQ}	T_{reint}	R_I	R_{II}
3	4	13	23	20	26 (20)	0.88	0.77
3	6	13	25	22	28 (22)	0.89	0.79
4	6	13	27	23	31 (23)	0.87	0.74
4	6	17	31	27	35 (27)	0.88	0.77

Table 2. $T^I_{RKQ}, T^{II}_{RKQ}, T_{\text{reint}}$, R_I and R_{II} for the indicated values of s_r, s_v and s_z, for explicit methods.

4.2 Explicit embedded RK methods

We consider $r = 3, 4$, $v = 4, 5$ $(s_r = 5, 6)$ and $z = 8, 10$ $(s_z = 13, 17)$. It is understood that the embedded pair consists of RKr and RKv. The methods used here are RK(3,4) and RK(4,5). This simply means that the RKv solution requires no stage evaluations, since it is computed from the same stages as the RKr solution - hence, the zeros in the s_v column of Table 3.

s_r	s_v	s_z	T^I_{RKQ}	T^{II}_{RKQ}	T_{reint}	R_I	R_{II}
5	0	13	23	18	28 (18)	0.82	0.64
6	0	13	25	19	31 (19)	0.80	0.61
6	0	17	29	23	35 (23)	0.83	0.66

Table 3. $T^I_{RKQ}, T^{II}_{RKQ}, T_{\text{reint}}$, R_I and R_{II} for the indicated values of s_r, s_v and s_z, for explicit embedded methods.

4.3 Explicit Nyström methods

We consider $r = 4$ $(s_r = 3)$, $v = 5$ $(s_v = 4)$ and $z = 10$ $(s_z = 17)$. The corresponding methods are RKN4, RKN5 and RKN10. The RK10 method used here is, in fact, the 10th-order component of a $(10, 12)$ embedded pair [8].

s_r	s_v	s_z	T^I_{RKQ}	T^{II}_{RKQ}	T_{reint}	R_I	R_{II}
3	4	17	27	24	30 (24)	0.90	0.80

Table 4. $T^I_{RKQ}, T^{II}_{RKQ}, T_{\text{reint}}$, R_I and R_{II} for the indicated values of s_r, s_v and s_z, for explicit Nyström methods.

4.4 Implicit RK methods

Here, we consider $r = 2, 3, 4$ $(s_r = 2)$, $v = 4, 5, 6$ $(s_v = 3)$ and $z = 8, 10$ $(s_z = 4, 5)$. The corresponding methods are RKL2, RKL4, RKL8, RKR3, RKR5, RKG4, RKG6, RKG8 and RKG10. Implicit methods do not involve explicit stage evaluations; rather, they require the solution of a nonlinear system of a given number of stages. We will assume here that the stage numbers s_r, s_v and s_z give a suitable indication of the computational effort required to solve these nonlinear stage equations, so that the ratios defined in (5) and (6) are still valid indicators of relative efficiciency.

s_r	s_v	s_z	T^I_{RKQ}	T^{II}_{RKQ}	T_{reint}	R_I	R_{II}
2	3	5	12	9	14 (10)	0.86	0.64
2	3	5	12	9	14 (10)	0.86	0.64
2	3	4	11	8	13 (9)	0.85	0.73
2	3	5	12	9	14 (10)	0.86	0.64

Table 5. $T^I_{RKQ}, T^{II}_{RKQ}, T_{\text{reint}}$, R_I and R_{II} for the indicated values of s_r, s_v and s_z, for implicit methods. First row: Radau (the method with 5 stages is a Lobatto method); second row: Lobatto; third and fourth rows: Gauss.

4.5 Discussion

We see that, in all cases, RKQ is more efficient than reintegration requiring, on average, only 86% (for Mode I) and 70% (for Mode II) of the computational effort needed by reintegration. The exception is, of course, the case when reintegration is not actually required (the values in parentheses), but it is reasonable to assume that this is a very rare case (note that this case is obtained from (1) with $p = -1$).

The values of T_{reint} in the tables were obtained from (1) assuming $p = 1$. It could, of course, transpire that a larger value of p is required. This, in turn, will reduce the value of R_I and R_{II}. For example, if we apply RK34Q8 to the initial-value problem

$$y' = \left(\frac{\ln 1000}{100}\right) y,\ y(0) = 1,\ x \in [1, 100],$$

using a tolerance of 10^{-4} on both local and global error, we find that reintegration requires $p = 5$. This then gives

$$R_I = \frac{23}{35} = 0.66 \text{ and } R_{II} = \frac{20}{35} = 0.57,$$

which is smaller than the 86% and 70% averages obtained above. Applying RK12Q8 (RK1 is Euler's method) to the Lotka-Volterra system

$$u' = u\,(v-1)\,, \quad v' = v(2-u), \ x \in [1, 100]$$

with a tolerance of 10^{-2} results in $p = 19$ and, hence,

$$R_I = \frac{17}{36} = 0.47 \text{ and } R_{II} = \frac{16}{36} = 0.44.$$

An even greater reduction occurs for the Hamiltonian example considered in [9] where, with a tolerance of 10^{-6}, we find $p = 22$, giving

$$R_I = \frac{23}{86} = 0.27 \text{ and } R_{II} = \frac{20}{86} = 0.23,$$

so that, in this case, RKQ requires roughly only one quarter of the computational effort needed by reintegration.

There is an additional contribution to the computational cost for both RKQ and reintegration: that of step rejections incurred in the LE component. These occur when the local error is deemed too large, and a new, smaller stepsize must be used. This leads to an additional term Ms_r that must be added to the RHS of (1),(3) and (4), and where M is the number of step rejections. However, since the LE used in both algorithms is the same, we have assumed that Ms_r will be the same for each, and so may be effectively ignored. Of course, if Ms_r is the dominant term in (1),(3) and (4), then R_I and R_{II} will be close to unity. We should note that since reintegration uses LE(r, v), RKQ in Mode I might return a value of M different to that of reintegration. If this difference does exist, however, we do not expect it to be significant.

5. Conclusion

We have compared the efficiency of two algorithms for controlling global error in Runge-Kutta solutions of initial-value problems. These algorithms are reintegration, based on a reapplication of the Runge-Kutta method using a refined node distribution, and RKQ, based on the concept of quenching. Reintegration is an *a posteriori* procedure, whereas RKQ is *in situ*. We find that RKQ can generally be expected to be more efficient than reintegration, by at least 16% or 30%, on average (depending on the mode of implementation of RKQ). Some numerical examples indicate that the gain in efficiency could be much greater - in one case, RKQ was almost four times more efficient. It is our opinion that *in situ* control of the global error is more desirable than *a posteriori* control and so, since RKQ can also be expected to be more efficient, we believe that RKQ should be regarded as the better of the two algorithms for global error control.

References

[1] J.S.C. Prentice, Stepwise global error control in an explicit Runge-Kutta method using local extrapolation with high-order selective quenching, *Journal of Mathematics Research*, 3, 2 (2011) 126-136.

[2] J.S.C. Prentice, Relative Global Error Control in the RKQ Algorithm for Systems of Ordinary Differential Equations, *Journal of Mathematics Research*, 3, 4 (2011) 59-66.

[3] J.S.C. Prentice, Nyström Methods in the RKQ Algorithm for Initial-value Problems, *arXiv.org* (Cornell University Library), 2011, 7p, [arXiv:1110.6749].

[4] J.S.C. Prentice, Efficiency of the RKQ Algorithm, *Applied Mathematical Sciences,* 6, 52 (2012) 2587-2591.

[5] E. Hairer, S.P. Norsett, and G. Wanner, *Solving Ordinary Differential Equations I: Nonstiff Problems,* Berlin: Springer, 2000.

[6] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems,* Berlin: Springer, 2002.

[7] J.C. Butcher, *Numerical Methods for Ordinary Differential Equations,* Chichester: Wiley, 2003.

[8] J.R. Dormand, M.E.A. El-Mikkawy and J. Prince, High-Order Embedded Runge-Kutta-Nyström Formulae, *IMA Journal of Numerical Analysis*, 7, (1987) 423-430.

[9] J.S.C. Prentice, Global Error Control in the Runge-Kutta Solution of a Hamiltonian System, *arXiv.org* (Cornell University Library), 2011, 12p, [arXiv:1111.6996].