World Academy of Science, Engineering and Technology
International Journal of Mathematical and Computational Sciences
Vol:4, No:1, 2010

The RK1GL2X3 method for initial value problems in ordinary differential equations

J.S.C. Prentice

***Abstract*—**The RK1GL2X3 method is a numerical method for solving initial value problems in ordinary differential equations, and is based on the RK1GL2 method which, in turn, is a particular case of the general RKrGLm method. The RK1GL2X3 method is a fourth-order method, even though its underlying Runge-Kutta method RK1 is the first-order Euler method, and hence, RK1GL2X3 is considerably more efficient than RK1. This enhancement is achieved through an implementation involving triple-nested two-point Gauss-Legendre quadrature.

***Keywords*—**RK1GL2X3, RK1GL2, RKrGLm, Runge-Kutta, Gauss-Legendre, initial value problem, local error, global error.

I. Introduction

In the simulation and modelling of physical systems it is often necessary to solve initial value problems in ordinary or partial differential equations. Numerical methods typically used for these problems are Runge-Kutta methods. The RKrGLm method [1],[2] is a numerical method, based on an Runge-Kutta method of order r (RKr) and m-point Gauss-Legendre quadrature (GLm), that is of higher order $(r+1)$ than its underlying RK method, and requires less computational effort. As such, it can be used to solve initial value problems more efficiently than an ordinary RK method. In this paper we consider using an RKGL method in place of the RKr method in RKrGLm, as a means of enhancing the order of the RKGL method itself. We use the simple first-order Euler method (RK1) and 2-point GL quadrature, and obtain methods of orders two, three and four via the RKGL mechanism. All of these new methods are more efficient than the underlying RK1 method.

II. Terminology and relevant concepts

A. Euler's method

Euler's method for solving

$$y' = f(x,y) \qquad y(x_0) = y_0 \qquad a \leqslant x \leqslant b \tag{1}$$

is given by

$$w_{i+1} = w_i + h_i f(x_i, w_i) \tag{2}$$

where $h_i \equiv x_{i+1} - x_i$ is a stepsize, and w_i denotes the numerical approximation to $y(x_i)$. We denote this method RK1.

Justin Prentice is with the Department of Applied Mathematics, University of Johannesburg, South Africa, email: jprentice@uj.ac.za

B. Two-point Gauss-Legendre quadrature

GL2 quadrature on $[u,v]$ is [3]

$$\int_u^v f(x, y(x))\,dx = h\sum_{i=1}^{2} C_i f(x_i, y(x_i)) + O\left(h^5\right) \tag{3}$$

where the nodes x_i are the roots of the 2nd degree Legendre polynomial on $[u,v]$. Here, h is the average separation of the nodes on $[u,v]$, a notation we will adopt from now on, and the C_i are appropriate weights. The roots of the 2nd degree Legendre polynomial on $[-1,1]$ are

$$\begin{aligned} \widetilde{x}_1 &= -0.5773502692 \\ \widetilde{x}_2 &= 0.5773502692 \end{aligned} \tag{4}$$

and are mapped to corresponding nodes x_i on $[u,v]$ via

$$x_i = \frac{1}{2}\left[(v-u)\,\widetilde{x}_i + u + v\right]. \tag{5}$$

Also, the average node separation on $[-1,1]$ is $2/3$, and so h on $[u,v]$ is given by

$$h = \frac{2}{3}\left(\frac{v-u}{2}\right), \tag{6}$$

while the weights

$$C_1 = \frac{3}{2}, \quad C_2 = \frac{3}{2} \tag{7}$$

are constants on any interval of integration.

C. The RK1GL2 algorithm

We briefly describe the RK1GL2 algorithm on the interval $[a,b]$, with reference to Figure 1.

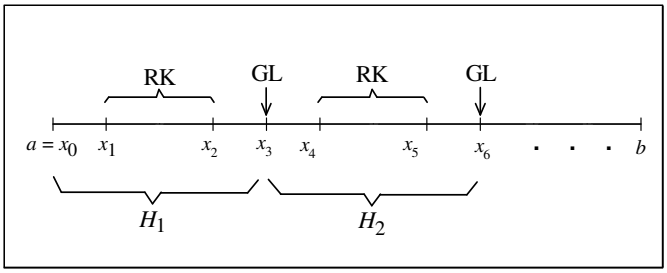

Fig. 1 RK1GL2 algorithm for the first two subintervals H_1 and H_2 on $[a,b]$.

Subdivide $[a,b]$ into N subintervals H_j, where $j = 1,2,\ldots,N$. At the RK nodes we use RK1:

$$w_{i+1} = w_i + h_i f(x_i, w_i) \tag{8}$$

where $i = 3(j-1), 3(j-1)+1$. At the GL nodes we use 2-point GL quadrature:

$$w_{3j} = w_{3(j-1)} + h \sum_{i=3(j-1)+1}^{3(j-1)+2} C_i f(x_i, w_i) \tag{9}$$

The GL component is motivated by

$$\int_{x_{3(j-1)}}^{x_{3j}} f(x, y(x))\, dx = y(x_{3j}) - y\left(x_{3(j-1)}\right) \approx h \sum_{i=3(j-1)+1}^{3(j-1)+2} C_i f(x_i, y(x_i)) \tag{10}$$

$$\Rightarrow y(x_{3j}) \approx y\left(x_{3(j-1)}\right) + h \sum_{i=3(j-1)+1}^{3(j-1)+2} C_i f(x_i, y(x_i)). \tag{11}$$

The global error in RK1GL2 at each node has the form

$$Ah^2 + Bh^4 = O\left(h^2\right) \tag{12}$$

where A and B are constants independent of h (but dependent on x). The term Ah^2 is due to the local RK error, and the term Bh^4 is due to the accumulation of the $O\left(h^5\right)$ approximation error in GL2 quadrature (see (3)). The local error at each RK node is also $O\left(h^2\right)$, and the local error at each GL node is $O\left(h^3\right)$ [1],[2]. The reason that the order of the local RK error is preserved in the global error of RK1GL2 is due to the multiplication by h in GL2, which effectively prevents the RK error from accumulating. This "quenching" effect is discussed in detail in [2].

III. The RK1GL2X3 method

Here, we describe the construction of RK1GL2X3. All discussions in this section refer to Figure 2. Note that the interval $[x_0, x_{15}]$ in this figure is consistent with the interval H_1 in Figure 1.

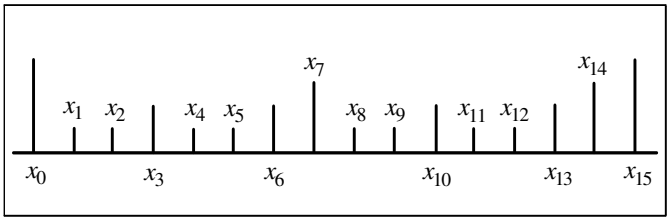

Fig. 2 Nodes relevant to RK1GL2, RK1GL2X2 and RK1GL2X3.

A. RK1GL2

Consider the nodes $\{x_0, x_7, x_{14}, x_{15}\}$ - ignore all the other nodes for the purposes of the immediate discussion - and assume that these nodes are spaced according to the scheme appropriate for GL2 quadrature (the apparent equispacing in the figure is not realistic). RK1GL2 is implemented on $[x_0, x_{15}]$ as follows: given the initial value at x_0, RK1 is used to find approximate solutions w_7 and w_{14}; these are then used in GL2 to find w_{15}. The global error at each of these nodes is $O\left(h^2\right)$ and the local error at x_{15} is $O\left(h^3\right)$, where h is the average separation of $\{x_0, x_7, x_{14}, x_{15}\}$. Note that we need three calls of the function $f(x, y)$ to implement RK1GL2, and since we obtained three solutions (w_7, w_{14}, w_{15}) on the interval, we have an average of one function call per solution.

B. RK1GL2X2

Now consider the nodes $\{x_0, x_3, x_6, x_7, x_{10}, x_{13}, x_{14}, x_{15}\}$. We implement RK1GL2 at $\{x_0, x_3, x_6, x_7\}$ and $\{x_7, x_{10}, x_{13}, x_{14}\}$. This gives solutions at x_7 and x_{14} that are globally $O\left(h^2\right)$ and locally $O\left(h^3\right)$. We then implement RK1GL2 on $\{x_0, x_7, x_{14}, x_{15}\}$ which gives solutions at $\{x_7, x_{14}, x_{15}\}$ that are globally $O\left(h^3\right)$ (recall that RKGL preserves the local order in the global error). Clearly, we have used RK1GL2, instead of RK1, to find solutions at x_7 and x_{14}, and then we have used RK1GL2 again to find w_{15} - hence, the 'X2' notation because of the nesting of GL2 on $[x_0, x_7]$ and $[x_7, x_{14}]$ within the GL2 applied on $[x_0, x_{15}]$ (with this notation, RK1GL2 could have been written as RK1GL2X1). The solution w_{15} thus obtained is globally $O\left(h^3\right)$ and locally $O\left(h^4\right)$, where h is the average separation of $\{x_0, x_3, x_6, x_7, x_{10}, x_{13}, x_{14}, x_{15}\}$. Although we have obtained solutions at seven nodes on the interval, it is only the three third-order solutions w_7, w_{14} and w_{15} that interest us. We need seven calls of $f(x, y)$ in this application, giving an average of $7/3$ function calls per solution on the interval.

C. RK1GL2X3

Now consider all the nodes in the figure. We apply RK1GL2 on $[x_0, x_3]$ and $[x_3, x_5]$ to get solutions w_3 and w_5 that have global order two and local order three. Then we apply RK1GL2 at $\{x_0, x_3, x_6, x_7\}$ to get w_7 which has global order three and local order four. A similar process on $[x_7, x_{14}]$ yields w_{14} which also has global order three and local order four. Finally, we apply RK1GL2 at $\{x_0, x_7, x_{14}, x_{15}\}$; the solutions of interest - w_7, w_{14} and w_{15} - are globally $O\left(h^4\right)$, where h is the average separation of all the nodes in the figure. Of course, the suffix 'X3' in the name of this method refers to the "RK1GL2 within RK1GL2 within RK1GL2" nesting, which we referred to as *triple-nesting* in the abstract. Also, we need 15 calls of $f(x, y)$ in the application of RK1GL2X3, giving an average of 5 function calls per solution on the interval.

D. Description in terms of one-step methods

An alternative description of RK1GL2X3 in terms of one-step methods can be given. If we were to apply RK1GL2 on $[x_0, x_{15}]$, we would need to use RK1 to obtain w_7 and w_{14}. Instead of doing this, we use RK1GL2 on $[x_0, x_7]$ and $[x_7, x_{14}]$ to get w_7 and w_{14}. This, of course, requires the introduction of the nodes x_3, x_6, x_{10} and x_{13}. It is as if we treat RK1GL2 as a one-step method (that replaces RK1), where the step is $[x_0, x_7]$ (and $[x_7, x_{14}]$). The same reasoning now applies to the subintervals $[x_0, x_3]$, $[x_3, x_6]$, $[x_7, x_{10}]$ and $[x_{10}, x_{13}]$. On each of these we introduce appropriate nodes

and apply RK1GL2 to obtain w_3, w_6, w_{10} and w_{13}, rather than RK1. Once again, we are treating RK1GL2 as a one-step method with step $[x_0, x_3]$ etc. Of course, despite obtaining solutions at 15 nodes on the interval, we only sample those at $\{x_7, x_{14}, x_{15}\}$, since they are the ones of highest order.

IV. Numerical example

As an example, we solve the test problem

$$y' = \frac{y}{4}\left(1 - \frac{y}{20}\right) \tag{13}$$

on $[0, 5]$ with $y(0) = 1$, using RK1, RK1GL2, RK1GL2X2 and RK1GL2X3. This equation has solution

$$y(x) = \frac{20}{1 + 19e^{-x/4}} \tag{14}$$

and is one of the test problems used by Hull *et al* [4]. From our previous discussion, we know that the number of function calls per solution increases in going from RK1GL2 to RK1GL2X3. However, the enhancement in the order of the methods is expected to offset the increase in function calls, so that we would expect the methods to become more efficient, particularly for small h (which corresponds to a large number of function calls). In Figure 3 we show efficiency curves for the methods applied to the test problem, where the expected behaviour is obvious. Clearly, RK1GL2X3 is more efficient than the others, due to its higher order. The slopes of the curves correspond to the orders of the methods, and we have confirmed that RK1GL2, RK1GL2X2 and RK1GL2X3 are of global order two, three and four, respectively.

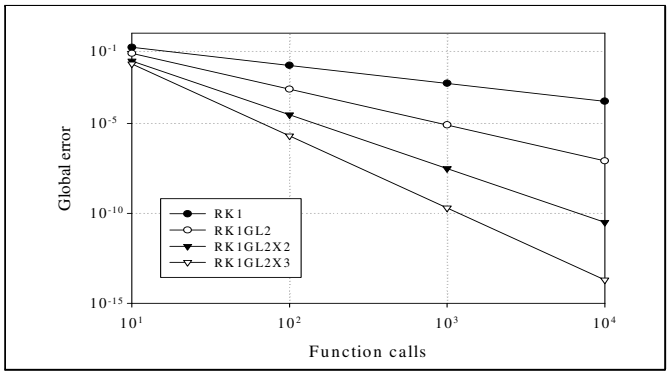

Fig. 3 Efficiciency curves for the indicated methods applied to the test problem.

V. Generalization

In the general case RKrGLm has global error

$$Ah^{r+1} + Bh^{2m} = O\left(h^{\min\{r+1,2m\}}\right). \tag{15}$$

If $r < 2m$ then the general method RKrGLmXn has

$$1 \leqslant n \leqslant 2m - r. \tag{16}$$

The upper limit on n is due to the term Bh^{2m} in (15), which arises from the approximation error in GLm quadrature. The lower limit on n gives RKrGLmX1, which is the default method RKrGLm (in fact, a completely unified notation is possible if we define RK$r \equiv$ RKrGLmX0, which allows us to include $n = 0$ in the above range). In this paper, we have used $r = 1, m = 2$ and so $n \leqslant 3$. We can conceive of other possibilities as well; for example, RK4GL3X2 which has global order six. Indeed, the global error for RKrGLmXn is $O\left(h^{\min\{r+n,2m\}}\right)$.

VI. Applications

The most obvious application of RKrGLmXn, and RK1GL2X3 in particular, is the reduction in computational effort when solving initial value problems. This is particularly attractive when solving a large system of ordinary differential equations, such as arises when a parabolic partial differential equation is suitably discretized. We stress the particular usefulness of RK1GL2X3 because it constitutes an enhancement of the simplest of the Runge-Kutta methods, the Euler method. The first order character of Euler's method makes it unattractive as a solution algorithm, despite its simplicity; the RK1GL2X3 method effectively transforms it into a fourth order method, which is far more appealing.

Improvements in efficiency notwithstanding, there is another useful application of RKrGLmXn. Consider RK1GL2 and RK1GL2X2 at $\{x_0, x_7, x_{14}, x_{15}\}$. Since RK1GL2X2 has higher order than RK1GL2, we can use RK1GL2X3 as an error estimator for the solutions obtained with RK1GL2, in a local extrapolation sense. In other words, this nested implementation allows RK1GL2 to generate its own error estimator, without the need to use an RKGL method based on some other RK method. Of course, this idea can be generalized, with RKrGLmXn_2 serving as error estimator for RKrGLmXn_1, provided that $n_1 < n_2$.

VII. Conclusion

We have described the RK1GL2X3 method for initial value problems. This method is based on Euler's method and a nested implementation of two-point Gauss-Legendre quadrature, following the character of the RK1GL2 method. The method has global order four. We have offered a generalization of the method - RKrGLmXn - noting that the quadrature nesting n is limited above by the values of r and m. Application of the method to a test problem has demonstrated the superior efficiency of RK1GL2X3 relative to RK1, RK1GL2 and RK1GL2X2.

References

[1] J.S.C. Prentice, "The RKGL method for the numerical solution of initial-value problems", *Journal of Computational and Applied Mathematics,* 213, 2 (2008) 477 – 487

[2] J.S.C. Prentice, "General error propagation in the RKrGLm method", *Journal of Computational and Applied Mathematics,* 228, (2009) 344 – 354.

[3] D. Kincaid and W. Cheney, *Numerical Analysis: Mathematics of Scientific Computing, 3rd ed.*, Pacific Grove: Brooks/Cole, 2002, pp492 – 498.

[4] T.E. Hull, W.H. Enright, B.M Fellen, and A.E. Sedgwick, "Comparing numerical methods for ordinary differential equations", *SIAM Journal of Numerical Analysis*, 9, 4 (1972) 603 – 637.