21

A General Algorithm for Local Error Control in the RK*r*GL*m* Method

Justin S. C. Prentice
Department of Applied Mathematics, University of Johannesburg, Johannesburg, South Africa

1. Introduction

Simulation of physical systems often requires the solution of a system of ordinary differential equations, in the form of an initial-value problem. Usually, a Runge-Kutta method is used to solve such a system numerically. Recently, we examined how the computational efficiency of a Runge-Kutta method could be improved through the mechanism of the RK*r*GL*m* algorithm, in the context of global error control via reintegration (Prentice, 2009). The RK*r*GL*m* method for solving the *d*-dimensional system

$$\frac{d\bar{y}}{dx} = \bar{f}\left(x, \bar{y}\right) \quad \bar{y}\left(x_0\right) = \bar{y}_0 \quad a \le x \le b \tag{1.1}$$

is based on an explicit Runge-Kutta method of order *r* (RK*r*), and *m*-point Gauss-Legendre quadrature (GL*m*). The method has a global error of order $r+1$, which is the same order as the local order of the underlying RK*r* method, provided that r and m are chosen such that $r+1 \le 2m$ (Prentice, 2008). Of course, any method designed for solving IVPs must facilitate local error control. In this paper we describe an effective algorithm for controlling the local relative error in RK*r*GL*m*.

2. Terminology and relevant concepts

In this section we describe terminology and concepts relevant to the paper, including a brief description of the RK*r*GL*m* method. Note that, throughout this paper, *overbar*, as in $\bar{v}$, indicates an $d \times 1$ vector, and *caret*, as in $\widehat{M}$, denotes an $d \times d$ matrix.

2.1 Explicit Runge-Kutta methods

We denote an explicit RK method for solving (1.1) by

$$\bar{w}_{i+1} = \bar{w}_i + h_i \bar{F}\left(x_i, \bar{y}_i\right) \tag{2.1}$$

where $h_i \equiv x_{i+1} - x_i$ is a stepsize, $\bar{w}_i$ denotes the numerical approximation to $\bar{y}(x_i)$, and $\bar{F}\left(x, \bar{y}\right)$ is a function associated with the particular RK method (indeed, $\bar{F}\left(x, \bar{y}\right)$ could be regarded as the function that defines the method).

2.2 Local and global errors

We define the *global error* in any numerical solution at x_i by

$$\bar{\Delta}_i \equiv \bar{w}_i - \bar{y}_i \tag{2.2}$$

and, specifically, the *RK local error* at x_i by

$$\bar{\varepsilon}_i \equiv \left[\bar{y}_{i-1} + h_{i-1}\bar{F}\left(x_{i-1}, \bar{y}_{i-1}\right)\right] - \bar{y}_i \tag{2.3}$$

In the above, $\bar{y}_{i-1}$ and $\bar{y}_i$ are the true solutions at x_{i-1} and x_i, respectively. Note that the true value $\bar{y}_{i-1}$ is used in the bracketed term in (2.3).

Note also that for the derivative $\bar{y}' = \bar{f}\left(x, \bar{y}\right)$ we have

$$\bar{f}\left(x_i, \bar{w}_i\right) = f\left(x_i, \bar{y}_i + \bar{\Delta}_i\right) = \bar{f}\left(x_i, \bar{y}_i\right) + \widehat{f_y}\left(x_i, \bar{\vartheta}\right)_i \bar{\Delta}_i \tag{2.4}$$

In the above we use the symbol $\bar{\vartheta}_i$ in $\widehat{f_y}\left(x_i, \bar{\vartheta}_i\right)\bar{\Delta}_i$ simply to denote an appropriate set of constants such that $\widehat{f_y}\left(x_i, \bar{\vartheta}_i\right)\bar{\Delta}_i$ is the residual term in the first-order Taylor expansion of $\bar{f}\left(x_i, \bar{y}_i + \bar{\Delta}_i\right)$. Furthermore, $\widehat{f_y}$ is the Jacobian

$$\widehat{f_y} = \begin{bmatrix} \dfrac{\partial f_1}{\partial y_1} & \cdots & \dfrac{\partial f_1}{\partial y_d} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial f_d}{\partial y_1} & \cdots & \dfrac{\partial f_d}{\partial y_d} \end{bmatrix} \tag{2.5}$$

where $\{f_1, f_2, \ldots, f_d\}$ are the components of $\bar{f}$, and d is the dimension of the system (1.1). Clearly, a global error of $\bar{\Delta}_i$ in $\bar{w}_i$ implies an error of $O\left(\bar{\Delta}\right)_i$ in the derivative $\bar{f}\left(x_i, \bar{w}_i\right)$.

2.3 Gauss-Legendre quadrature

Gauss-Legendre quadrature on $[u, v]$ with *m* nodes is given by (Kincaid & Cheney, 2002)

$$\int_u^v \bar{f}\left(x, \bar{y}\right) dx = h \sum_{i=1}^{m} C_i \bar{f}\left(x_i, \bar{y}_i\right) + O\left(h^{2m+1}\right) \tag{2.6}$$

where the nodes x_i are the roots of the Legendre polynomial of degree m 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 average node separation *h* on $[u, v]$ is defined by

$$h \equiv \frac{v-u}{m+1}. \tag{2.7}$$

The nodes on $[-1, 1]$, denoted $\tilde{x}_i$, are mapped to corresponding nodes x_i on $[u, v]$ via

$$x_i = \frac{1}{2}\left[(v-u)\tilde{x}_i + u + v\right], \tag{2.8}$$

and the weights C_i are constants on any interval of integration. We have referred to the interval $[-1,1]$ above because the nodes $\tilde{x}_i$ on this interval are extensively tabulated.

2.4 The RK*r*GL*m* algorithm

We briefly describe the general RK*r*GL*m* algorithm on the interval $[a,b]$, with reference to Figure 1.

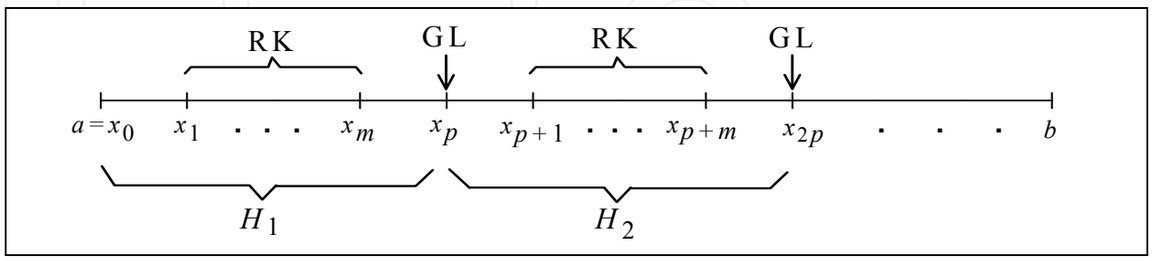

Fig. 1. Schematic depiction of the RK*r*GL*m* algorithm.

Subdivide $[a,b]$ into *N* subintervals H_j. At the RK nodes on H_j we use RK*r*:

$$\bar{w}_{i+1} = \bar{w}_i + h_i \bar{F}(x_i, \bar{w}_i) \qquad i \in \{(j-1)p, \ldots, (j-1)p+m-1\}. \tag{2.9}$$

At the GL nodes we use *m*-point GL quadrature:

$$\bar{w}_{(\mu+1)p} = \bar{w}_{\mu p} + h\sum_{i=1}^{m} C_i \bar{f}\left(x_{i+\mu p}, \bar{w}_{i+\mu p}\right). \tag{2.10}$$

where $\mu = 0,1,2,\ldots$. Note that $p \equiv m+1$.
The GL component is motivated by

$$\begin{aligned} &\int_{x_{\mu p}}^{x_{(\mu+1)p}} \bar{f}(x,y)\,dx = \bar{y}_{(\mu+1)p} - \bar{y}_{\mu p} \approx h\sum_{i=1}^{m} C_i \bar{f}\left(x_{i+\mu p}, \bar{y}_{i+\mu p}\right) \\ &\Rightarrow \bar{y}_{(\mu+1)p} \approx \bar{y}_{\mu p} + h\sum_{i=1}^{m} C_i \bar{f}\left(x_{i+\mu p}, \bar{y}_{i+\mu p}\right). \end{aligned} \tag{2.11}$$

The RK*r*GL*m* algorithm has been shown to be consistent, convergent and zero-stable (Prentice, 2008).

2.5 Local error at the GL nodes

The local error at the GL nodes is defined in a similar way to that for an RK method:

$$\begin{aligned} &\int_{x_{\mu p}}^{x_{\mu p+m+1}=x_{(\mu+1)p}} \bar{f}(x,\bar{y})\,dx = \underbrace{\bar{y}_{\mu p+m+1}}_{(\mu+1)p} - \bar{y}_{\mu p} = h\sum_{i=1}^{m} C_i \bar{f}\left(x_{i+\mu p}, \bar{y}_{i+\mu p}\right) + O\left(h^{2m+1}\right) \\ &\Rightarrow \bar{\varepsilon}_{(\mu+1)p} \equiv \left[\bar{y}_{\mu p} + h\sum_{i=1}^{m} C_i \bar{f}\left(x_{i+\mu p}, \bar{y}_{i+\mu p}\right)\right] - \bar{y}_{(\mu+1)p} = O\left(h^{2m+1}\right). \end{aligned} \tag{2.12}$$

We remind the reader that in RK*r*GL*m* we choose r and m such that $r+1 \le 2m$, which ensures that RK*r*GL*m* has a global error of $O\left(h^{r+1}\right)$ (Prentice, 2008).

2.6 Implementation of RK*r*GL*m*

There are a few points regarding the implementation of RK*r*GL*m* that need to be discussed:

a. If we merely sample the solutions at the GL nodes, treating the computations at the RK nodes as if they were the stages of an ordinary RK method, then RK*r*GL*m* would be reduced to an inefficient one-step method. This is not the intention behind the development of RK*r*GL*m*; rather, RK*r*GL*m* represents an attempt to improve the efficiency of any RK*r* method, simply by replacing the computation at every (*m* + 1)th node by a quadrature formula which does not require evaluation of any of the stages in the underlying RK*r* method.

b. Of course, it is clear from the above that on H_1 the RK nodes are required to be consistent with the nodes necessary for GL quadrature. If, however, the RK nodes are located differently (as would be required by a local error control mechanism, for example) then it is a simple matter to construct a Hermite interpolating polynomial of degree $2m+1$ (which has an error of order $2m+2$) using the solutions at the nodes $\{x_0,\ldots,x_m\}$. Then, assuming x_0 maps to -1 and x_m maps to the largest Legendre polynomial root x on $[-1,1]$, the position of the other nodes $\{x_1^*,\ldots,x_{m-1}^*\}$ suitable for GL quadrature may be determined, and the Hermite polynomial may be used to find approximate solutions of order $r+1$ at these nodes, thus facilitating the GL component of RK*r*GL*m*. A similar procedure is carried out on the next subinterval H_1, and so on. Indeed, we will see that the Hermite polynomial described here will play an important role in our error control process, and is described in more detail in the next subsection.

c. If the underlying RK*r* method possesses a continuous extension it would not be necessary to construct the Hermite polynomial described above. However, there is no guarantee that a continuous extension of appropriate order (at least $2m+1$) will be available, and it is generally true that determining a continuous extension for a RK method requires additional stages in the RK method, which would most likely compromise the gain in efficiency offered by RK*r*GL*m*. Note that the construction of the Hermite polynomial only requires one additional evaluation of $f(x,y)$, at x_m.

2.7 The Hermite interpolating polynomial

If the data $\{x_i, y_i, y_i' : i = 0,\ldots,m\}$ are available, then a polynomial $H_P(x)$, of degree at most $2m+1$, with the interpolatory properties

$$H_P(x_i) = y_i \qquad H_P'(x_i) = y_i' \tag{2.13}$$

for each *i*, may be constructed. If the nodes x_i are distinct, then $H_P(x)$ is unique. This approximating polynomial is known as the Hermite interpolating polynomial (Burden & Faires, 2001) and has an approximation error given by

$$y(x) - H_P(x) = \frac{y^{(2m+2)}\left(\xi(x)\right)}{(2m+2)!}\prod_{i=0}^{m}(x-x_i)^2 \tag{2.14}$$

where $x_0 < \xi(x) < x_m$. If h is the average separation of the nodes on $\left[x_0, x_m\right]$, it is possible to write $x - x_i = \sigma_i h$, where σ_i is a suitable constant, and hence

$$y(x) - H_P(x) = O(h^{2m+2}). \tag{2.15}$$

The algorithm for determining the coefficients of $H_P(x)$ is linear, as in

$$\mathbf{c} = \mathbf{A}^{-1}\mathbf{b} \tag{2.16}$$

where $\mathbf{c}$ is a vector of the coefficients of $H_P(x)$, $\mathbf{A}$ is the relevant interpolation matrix, and $\mathbf{b}$ is a vector containing y_i and y_i'. The details of these terms need not concern us here; rather, if an error $O(\Delta)$ exists in each of y_i and y_i', then an error of $O(\Delta)$ will exist in each component of $\mathbf{c}$. Moreover, since $H_P(x)$ is linear in its coefficients, then an error of $O(\Delta)$ will also exist in any computed value of $H_P(x)$. Consequently, we may write

$$y(x) - H_P(x) = O(h^{2m+2}) + O(\Delta) \tag{2.17}$$

where the $O(\Delta)$ term arises from errors in y_i and y_i'. We have assumed, of course, that the errors in y_i and y_i' are of the same order, which is the situation that we will encounter later.

3. Local error control in RK*r*GL*m*

3.1 The order of the tandem method

The idea behind the use of a *tandem* method is that it must be of sufficiently high order such that, relative to the approximate solution generated by RK*r*GL*m*, the tandem method yields a solution that may be assumed to be essentially exact. This solution is propagated in both RK*r*GL*m* and the tandem method itself, and the difference between the two solutions is taken as an estimate of the local error in RK*r*GL*m*. This amounts to so-called *local extrapolation* and is not dissimilar in spirit to error estimation techniques employed using Runge-Kutta embedded pairs (Hairer et al., 2000; Butcher, 2003). Generally speaking, though, the tandem method is not embedded.

To decide on an appropriate order for the tandem method we consider the local error at the GL nodes

$$\begin{aligned}\bar{\varepsilon}_{(\mu+1)p} &= \bar{y}_{\mu p} + h\sum_{i=1}^{m} C_i \bar{f}\left(x_{i+\mu p}, \bar{y}_{i+\mu p}\right) - \bar{y}_{(\mu+1)p} \\ &= \left(\bar{w}_{\mu p,t} - \bar{\Delta}_{\mu p,t}\right) + h\sum_{i=1}^{m} C_i \bar{f}\left(x_{i+\mu p}, \bar{w}_{i+\mu p,t} - \bar{\Delta}_{i+\mu p,t}\right) - \left(\bar{w}_{(\mu+1)p,t} - \bar{\Delta}_{(\mu+1)p,t}\right)\end{aligned} \tag{3.1}$$

where $\bar{w}_{(\bullet),t}$ is the solution from the tandem method at $x_{(\bullet)}$, and $\bar{\Delta}_{(\bullet),t}$ is the global error in $\bar{w}_{(\bullet),t}$. Expanding the term in the sum in a Taylor series gives

$$\begin{aligned}\bar{\varepsilon}_{(\mu+1)p} &= \bar{w}_{\mu p,t} + h\sum_{i=1}^{m} C_i \bar{f}\left(x_{i+\mu p}, \bar{w}_{i+\mu p,t}\right) - \bar{w}_{(\mu+1)p,t} \\ &\quad - \bar{\Delta}_{\mu p,t} + \bar{\Delta}_{(\mu+1)p,t} + h\sum_{i=1}^{m} C_i \widehat{f_y}(x_{i+\mu p}, \bar{\zeta}_{i+\mu p,t})\bar{\Delta}_{i+\mu p,t}\end{aligned} \tag{3.2}$$

and so

$$\begin{aligned}&\bar{w}_{\mu p,t}+h\sum_{i=1}^{m}C_i\bar{f}\left(x_{i+\mu p},\bar{w}_{i+\mu p,t}\right)-\bar{w}_{(\mu+1)p,t}\\&=\bar{\varepsilon}_{(\mu+1)p}+\left(\bar{\Delta}_{\mu p,t}-\bar{\Delta}_{(\mu+1)p,t}-h\sum_{i=1}^{m}C_i\widehat{f_y}(x_{i+\mu p},\bar{\zeta}_{i+\mu p,t})\bar{\Delta}_{i+\mu p,t}\right)\end{aligned}\tag{3.3}$$

where $\bar{\zeta}_{i+\mu p,t}$ is analogous to $\bar{\vartheta}_i$ in (2.4). The sum on the RHS of (3.3) is of higher order than $\bar{\Delta}_{\mu p,t}-\bar{\Delta}_{(\mu+1)p,t}$, because of the multiplication by h, and since we cannot expect, in general, that $\bar{\Delta}_{\mu p,t}-\bar{\Delta}_{(\mu+1)p,t}=0$, the term in parentheses must be $O\left(h^q\right)$, where q is the global order of the tandem method. Since $\bar{\varepsilon}_{(\mu+1)p}=O\left(h^{2m+1}\right)$ in the RK*r*GL*m* method, we require $q>2m+1$ in order for

$$\bar{w}_{\mu p,t}+h\sum_{i=1}^{m}C_i\bar{f}\left(x_{i+\mu p},\bar{w}_{i+\mu p,t}\right)-\bar{w}_{(\mu+1)p,t}\approx\bar{\varepsilon}_{(\mu+1)p}\tag{3.4}$$

to be a good (and asymptotically $\left(h\to 0\right)$ correct) estimate for the local error in RK*r*GL*m*. The first two terms on the LHS of (3.4) arise from RK*r*GL*m* with the tandem solution as input, while $\bar{w}_{(\mu+1)p,t}$ is the tandem solution at $x_{(\mu+1)p}$.

The implication, then, is that the tandem method must have a global order of at least $2m+2$, which implies $q>r+2$, since we already have $r+1=2m$ in RK*r*GL*m*. We acknowledge that our choice of q differs from conventional wisdom (which would choose $q>r+1$ so that the local order of the tandem method is one greater than the RK local order), but it is clear from (3.3) that the propagation of the tandem solution requires the global order of the tandem method to be greater than the order of $\bar{\varepsilon}_{(\mu+1)p}$. Of course, at the RK nodes the local order is $r+1$, so the tandem method with global order $q>r+2$ is more than suitable at these nodes.

3.2 The error control algorithm

We describe the error control algorithm on the first subinterval $H_1=[x_0(=a),x_p]$ (see Figure 1). The same procedure is then repeated on subsequent subintervals.

Solutions $\bar{w}_{1,r}$ and $\bar{w}_{1,q}$ are obtained at x_1 using RK*r* and RK*q*, respectively. We assume

$$\left|\bar{w}_{1,r}-\bar{y}_1\right|=\bar{L}_1h_0^{r+1}\approx\left|\bar{w}_{1,r}-\bar{w}_{1,q}\right|\tag{3.5}$$

where $h_0\equiv x_1-x_0$ and $\bar{L}_1$ is a vector of local error coefficients (we will discuss the choice of a value for h_0 later). The exponent of $r+1$ indicates the order of the local error in RK*r*. We find the maximum value of

$$\left|\frac{w_{1,r,i}-y_{1,i}}{y_{1,i}}\right|\approx\left|\frac{w_{1,r,i}-w_{1,q,i}}{w_{1,q,i}}\right|\qquad i=1,\ldots,d\tag{3.6}$$

where the index i refers to the components of the indicated vectors (so $w_{1,r,i}$ is the ith component of $\bar{w}_{1,r}$, etc). Call this maximum M_1 and say it occurs for $i = k$. Hence,

$$M_1 = \left| \frac{w_{1,r,k} - w_{1,q,k}}{w_{1,q,k}} \right| \tag{3.7}$$

is the largest relative error in the components of $\bar{w}_{1,r}$. Note that k may vary from node to node, but at any particular node we will always intend for k to denote the maximum value of (3.6). We now demand that

$$M_1 \le \delta_R \quad \Rightarrow \quad \left| w_{1,r,k} - w_{1,q,k} \right| \le \delta_R \left| w_{1,q,k} \right| \tag{3.8}$$

where δ_R is a user-defined *relative* tolerance. If this inequality is violated we find a new stepsize h_0^* such that

$$h_0^* = 0.9 \left(\frac{\delta_R \left| w_{1,q,k} \right|}{L_{1,k}} \right)^{\frac{1}{r+1}} \qquad \left(\Rightarrow \; L_{1,k} \left(h_0^* \right)^{r+1} < \delta_R \left| w_{1,q,k} \right| \right) \tag{3.9}$$

where $L_{1,k}$ is the kth component of $\bar{L}_1$, and then we find new solutions $\bar{w}_{1,r}$ and $\bar{w}_{1,q}$ using h_0^* ($\bar{L}_1$ is determined from (3.5)). The factor 0.9 in (3.9) is a *safety factor* allowing for the fact that $\bar{w}_{1,q}$ is not truly exact. To cater for the possibility that any component of $\bar{w}_{1,q}$ is close to zero we actually demand

$$\left| w_{1,r,k} - w_{1,q,k} \right| \le \max \left\{ \delta_A , \delta_R \left| w_{1,q,k} \right| \right\} \tag{3.10}$$

where δ_A is a user-defined *absolute* tolerance. We then set $h_1 = h_0^*$ and proceed to the node x_2, where the error control process is repeated, and similarly for x_3 up to x_m. The process of recalculating a solution using a new stepsize is known as a step rejection.

In the event that the condition in (3.10) is satisfied, we still calculate a new stepsize h_0^* (which would now be larger than h_0) and set $h_1 = h_0^*$, on the assumption that if h_0^* satisfies (3.10) at x_1, then it will do so at x_2 as well (however, we also place an upper limit on h_0^* of $2h_0$, although the choice of the factor two here is somewhat arbitrary). In the worst-case scenario we would find that h_1 is too large and a new, smaller value h_1^* must be used. The exception occurs when $\left| \bar{w}_{1,r,k} - \bar{w}_{1,q,k} \right| = \bar{0}$. In this case we simply set $h_1 = 2h_0$ and proceed to x_2.

The above is nothing more than well-known local relative error control in an explicit RK method using local extrapolation. It is at the GL node x_p that the algorithm deviates from the norm. A step-by-step description of the procedure at x_p follows:

1. Once error control at $\{x_1, x_2, \ldots, x_m\}$ has been effected (which necessarily defines the positions of $\{x_1, x_2, \ldots, x_m\}$ due to stepsize modifications that may have occurred), the location of x_p must be determined such that the local relative error at x_p is less than $\max \left\{ \delta_A , \delta_R \left| w_{p,q,k} \right| \right\}$, in which k has the meaning discussed earlier.

2. To this end, we utilize the map (2.8), demanding that $x_0 (= u)$ corresponds to -1 on the interval $[-1,1]$, and x_m corresponds to the largest root $\tilde{x}_m$ of the mth-degree Legendre polynomial in $[-1,1]$. This allows $x_p (= v)$ to be found, where x_p corresponds to 1 on $[-1,1]$, and so new nodes $\{x_1^*, x_2^*, \ldots, x_{m-1}^*\}$ can be determined such that $\{x_1^*, x_2^*, \ldots, x_{m-1}^*, x_m\}$ are consistent with the GL quadrature nodes on $[x_0, x_p]$.
3. We wish to perform GL quadrature, using the nodes $\{x_1^*, x_2^*, \ldots, x_{m-1}^*, x_m\}$, on $[x_0, x_p]$, but we do not have the approximate solutions $\{\bar{w}_{1,q}^*, \ldots, \bar{w}_{m-1,q}^*\}$ at $\{x_1^*, x_2^*, \ldots, x_{m-1}^*\}$.
4. Hence, we construct the Hermite interpolating polynomial $H_P(x)$ on $[x_0, x_m]$ using the original nodes $\{x_1, x_2, \ldots, x_m\}$ and the solutions that have been obtained at these nodes; of course, the derivative of $\bar{y}(x)$ at these nodes is given by $\bar{f}(x, \bar{y})$. Note that a Hermite polynomial must be constructed for each of the s components of the system, so if $d > 1$, $H_P(x)$ is actually a $d \times 1$ vector of Hermite polynomials.
5. We use the qth-order solutions that are available, so that we expect the approximation error in each $H_P(x)$ to be $O(h^q)$, as shown in (2.4) and (2.17).
6. The solutions $\{\bar{w}_{1,q}^*, \ldots, \bar{w}_{m-1,q}^*\}$ at $\{x_1^*, x_2^*, \ldots, x_{m-1}^*\}$ are then obtained from $\{H_P(x_1^*), \ldots, H_P(x_{m-1}^*)\}$.
7. GL quadrature then gives $\bar{w}_p$ with local error $O(h^{2m+1})$, as per (2.12).
8. The tandem method RKq is used to find $\bar{w}_{p,q}$, and $|\bar{w}_p - \bar{w}_{p,q}|$ is then used for error control:
   a. we know that the local error in $\bar{w}_p$ is $O(h^{2m+1})$, where h here is the average node separation on $[x_0, x_p]$;
   b. if the local error is too large then a new average node separation h^* is determined; using h^*, a new position for x_p, denoted x_p^*, is found from $x_p^* = x_0 + ph^*$;
   c. if $x_p^* > x_m$, we redefine the nodes $\{x_1^*, x_2^*, \ldots, x_{m-1}^*, x_m\}$, find qth-order solutions at these new nodes using $H_P(x)$, and then find solutions at x_p^* using GL quadrature and RKq;
   d. if $x_p^* \le x_m$, we reject the GL step since there is now no point in finding a solution at x_p^*.
9. After all this, the node x_p^* or x_m (if $x_p^* \le x_m$) defines the endpoint of the subinterval H_1; the stepsize h is set equal to the largest separation of the nodes on H_1, and the

entire error control procedure is implemented on the next subinterval H_2. Note also that it is the *q*th-order solution at the endpoint of H_1 that is propagated in the RK solution at the next node.

3.3 Initial stepsize

To find a stepsize h_0 to begin the calculation process, we assume that the local error coefficient $L_{1,k}=1$ and then find h_0 from

$$h_0=\left(\max\left\{\delta_A,\delta_R\left|\overline{y}_{0,k}\right|\right\}\right)^{\frac{1}{r+1}} \tag{3.11}$$

Solutions obtained with RK*r* and RK*q* using this stepsize then enable a new, possibly larger, h_0 to be determined, and it is this new h_0 that is used to find the solutions $\overline{w}_{1,r}$ and $\overline{w}_{1,q}$ at the node x_1.

3.4 Final node

We keep track of the nodes that evolve from the stepsize adjustments, until the end of the interval of integration *b* has been exceeded. We then backtrack to the node on $[a,b]$ closest to *b* (call it x_{f-1}), determine the stepsize $h_{f-1}\equiv b-x_{f-1}$, and then find $\overline{w}_{b,r}$ and $\overline{w}_{b,q}$, the numerical solutions at *b* using RK*r* and RK*q*, with h_{f-1}, x_{f-1} and $\overline{w}_{f-1,q}$ as input for both RK*r* and RK*q*. This completes the error control procedure.

4. Comments on embedded RK methods and continuous extensions

Our intention has been to develop an effective local error control algorithm for RK*r*GL*m*, and we believe that the above-mentioned algorithm achieves this objective. Moreover, the algorithm is general in the choice of RK*r* and RK*q*. These two methods could be entirely independent of each other, or they could constitute an embedded pair, as in RK(*r*,*q*). This latter choice would require fewer stage evaluations at each RK node, and so would be more efficient than if RK*r* and RK*q* were independent. Nevertheless, the use of an embedded pair is not necessary for the proper functioning of our error control algorithm.

The option of constructing $H_P(x)$ using the nodes $\{x_m=x_{p-1},x_p,\ldots,x_{2p-1}\}$ for error control at x_{2p} (as opposed to using $\{x_p,\ldots,x_{2p-1}\}$) is worth considering. Such a polynomial, together with the Hermite polynomial constructed on $\{x_0,x_1,\ldots,x_m\}$, forms a piecewise continuous approximation to $\overline{y}(x)$ on $[x_0,x_{2p-1}]$. Of course, this process is repeated at the nodes $\{x_{2p},x_{2p+1},\ldots,x_{3p-1}\}$, and so on. In this way the Hermite polynomials, which must be constructed out of necessity for error control purposes, become a piecewise continuous (and smooth) extension of the approximate discrete solution. Such an extension is not constructed *a posteriori*; rather, it is constructed on each subinterval H_i as the RK*r*GL*m* algorithm proceeds, and so may be used for event trapping.

5. Numerical examples

We will use RK5GL3 to demonstrate the error control algorithm. In RK5GL3 we have $r=5,m=3$ so that the tandem method must be an eighth-order RK method, which we

denote RK8. The RK5 method in RK5GL3 is due to Fehlberg (Hairer et al., 2000), as is RK8 (Hairer et al., 2000; Butcher, 2003).
By way of example, we solve

$$y' = \frac{1}{1+x^2} - 2y^2 \tag{5.1}$$

on $[0,5]$ with $y(0) = 0$, and

$$y' = \frac{y}{4}\left(1 - \frac{y}{20}\right) \tag{5.2}$$

on $[0,30]$ with $y(0) = 1$. The first of these has a unimodal solution on the indicated interval, and we will refer to it as IVP1. The second problem is one of the test problems used by Hull et al (Hull et al., 1972), and we will refer to it as IVP2. These problems have solutions

$$\begin{aligned} &\text{IVP1:} \quad y(x) = \frac{x}{1+x^2} \\ &\text{IVP2:} \quad y(x) = \frac{20}{1+19e^{-x/4}} \end{aligned} \tag{5.3}$$

In Table 1 we show the results of implementing our local error control algorithm in solving both test problems. The absolute tolerance δ_A was always 10^{-10}, except for IVP1 with $\delta_R = 10^{-10}$, for which $\delta_A = 10^{-12}$ was used.

IVP1				
δ_R	10^{-4}	10^{-6}	10^{-8}	10^{-10}
RK step rejections	2	2	0	2
GL step rejections	2	5	10	19
nodes	12	20	37	79
RKGL subintervals	4	6	12	25

IVP2				
δ_R	10^{-4}	10^{-6}	10^{-8}	10^{-10}
RK step rejections	2	2	4	5
GL step rejections	2	3	5	9
nodes	10	19	39	87
RKGL subintervals	3	6	11	24

Table 1. Performance data for error control algorithm applied to IVP1 and IVP2.

In this table, *RK step rejections* is the number of times a smaller stepsize had to be determined at the RK nodes; *GL step rejections* is the number of times that $x_4^* \le x_3$, as described in the previous section; *nodes* is the total number of nodes used on the interval of integration, including the initial node x_0; and *RKGL subintervals* is the total number of subintervals H_i used on the interval of integration. It is clear that as δ_R is decreased so the number of nodes and RKGL subintervals increases (consistent with a decreasing stepsize), and so there is

more chance of step rejections. There are not many RK step rejections for either problem. When $\delta_R = 10^{-10}$ the GL step rejections for IVP1 are 19 out a possible 25 (almost 80%), but for IVP2 the GL step rejections number only about 38%). In both cases the GL step rejections arise as a result of relatively large local error coefficients at the GL nodes, which necessarily lead to relatively small values of h, the average node separation, so that the situation $x_4^* \le x_3$ is quite likely to occur.

Figures 2 and 3 show the RK5GL3 local error for IVP1 and IVP2. The curve labelled *tolerance* in each figure is $\delta_R |\overline{y}_i|$, which is the upper limit placed on the local error.

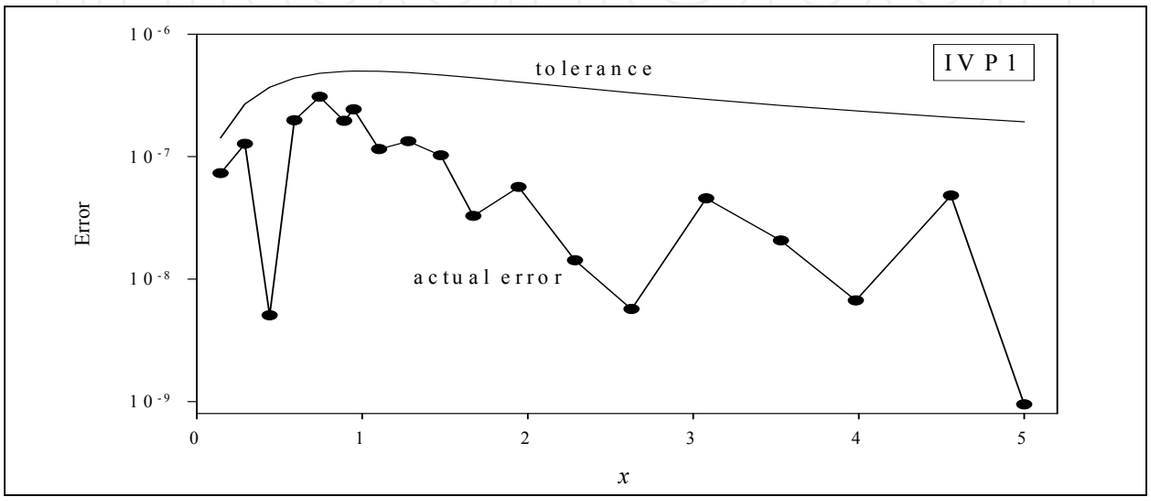

Fig. 2. RKGL local error for IVP1, with $\delta_R = 10^{-6}$.

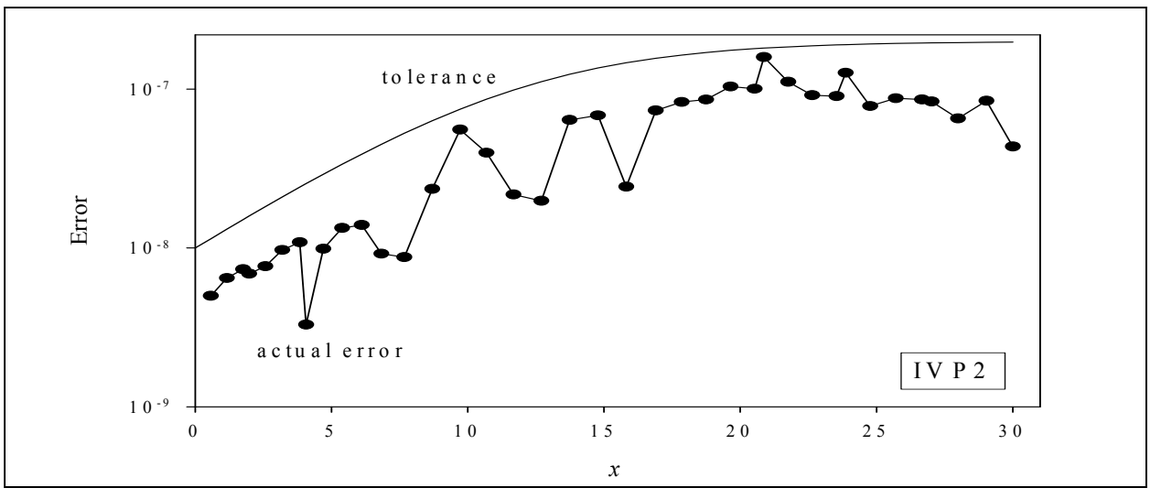

Fig. 3. RKGL local error for IVP2, with $\delta_R = 10^{-8}$.

In Figure 2 we have used $\delta_R = 10^{-6}$, and in Figure 3 we have used $\delta_R = 10^{-8}$. It is clear that in both cases the tolerance has been satisfied, and the error control algorithm has been successful. In Figure 4, for interest's sake, we show the stepsize variation as function of node index (#) for these two problems.

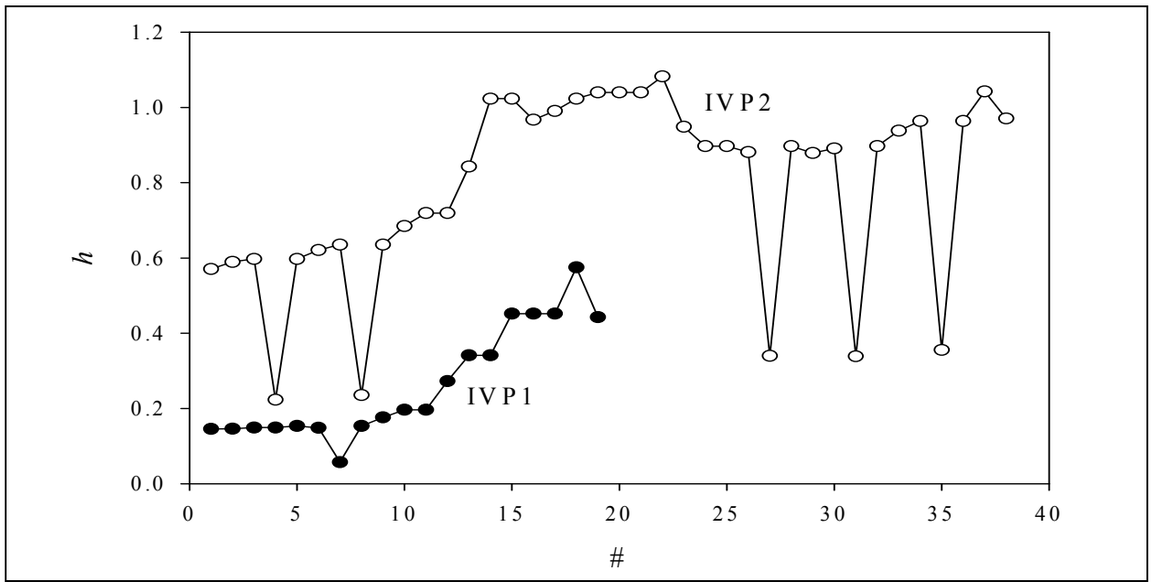

Fig. 4. Stepsize h vs node index (#) for IVP1 and IVP2.

To demonstrate error control in a system, we use RK5GL3 to solve

$$\begin{aligned} y_1' &= y_2 \\ y_2' &= e^{2x}\sin x - 2y_1 + 2y_2 \\ y_1(0) &= -\frac{2}{5},\ \ y_2(0) = -\frac{3}{5} \end{aligned} \tag{5.4}$$

on $[0,3]$. The solution to this system, denoted SYS1, is

$$\begin{aligned} y_1 &= \frac{e^{2x}}{5}(\sin x - 2\cos x) \\ y_2 &= \frac{e^{2x}}{5}(4\sin x + 3\cos x) \end{aligned} \tag{5.5}$$

The performance table for RK5GL3 local error control applied to this problem is shown in Table 2.

SYS1				
δ_R	10^{-4}	10^{-6}	10^{-8}	10^{-10}
RK step rejections	3	5	6	9
GL step rejections	3	4	8	8
nodes	10	25	52	115
RKGL subintervals	3	7	15	31

Table 2. Performance data for error control algorithm applied to SYS1.

The performance is similar to that shown in Table 1. In all calculations reflected in Table 2, we have used $\delta_A = 10^{-12}$. The error in the components y_1 and y_2 of SYS1 is shown in Figures 5 and 6.

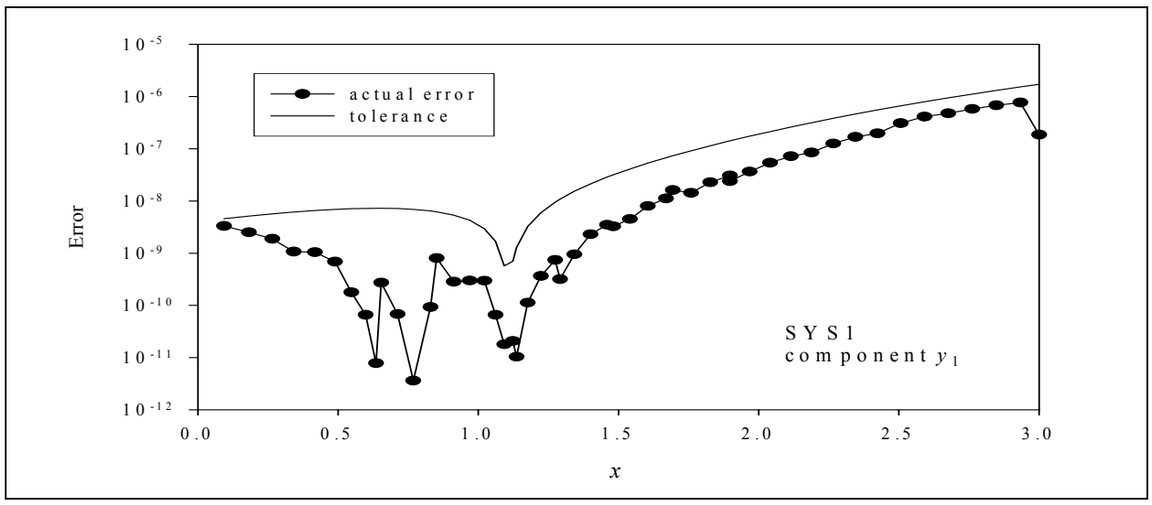

Fig. 5. Error in component y_1 of SYS1.

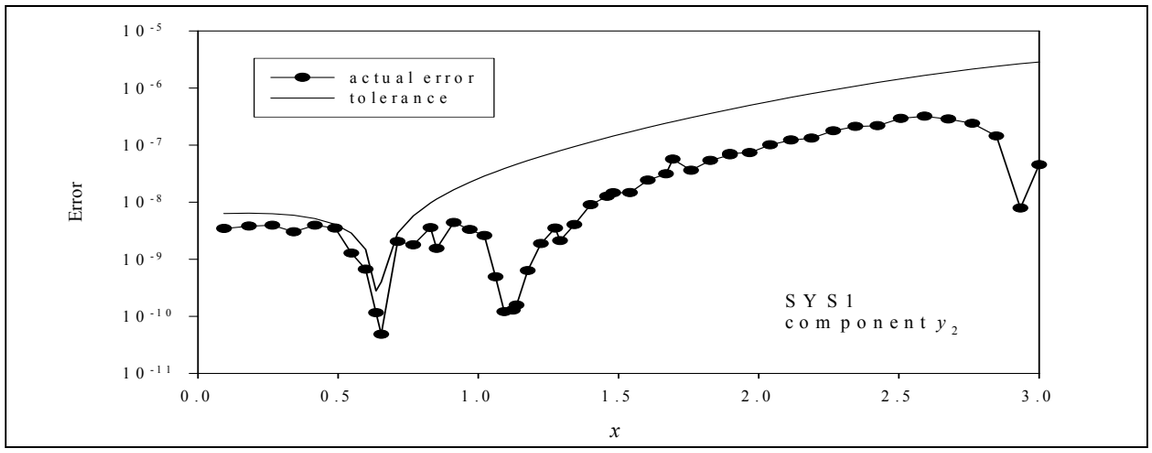

Fig. 6. Error in component y_2 of SYS1.

7. Conclusion and scope for further research

We have developed an effective algorithm for controlling the local relative error in RK*r*GL*m*, with $r+1 \le m$. The algorithm utilizes a tandem RK method of order $r+3$, at least. A few numerical examples have demonstrated the effectiveness of the error control procedure.

7.1 Further research

Although the algorithm is effective, it is somewhat inefficient, as evidenced by the large number of step rejections shown in the tables. Ways to improve efficiency might include :

a. The use of an embedded RK pair, such as DOPRI853 (Dormand & Prince, 1980), to reduce the total number of RK stage evaluations,
b. Using a high order RKGL method as the tandem method, since the RKGL methods were originally designed to improve RK efficiency,
c. Error control per subinterval H_j, rather than per node, which might require reintegration on each subinterval,

d. Optimal stepsize adjustment, so that stepsizes that are *smaller than necessary* are not used. Smaller stepsizes implies more nodes, which implies greater computational effort.

8. References

Burden, R.L. and Faires, J.D., (2001), *Numerical analysis, 7th ed.*, Brooks/Cole, 0-534-38216-9, Pacific Grove.

Butcher, J.C., (2003), *Numerical methods for ordinary differential equations*, Wiley, 0-471-96758-0, Chichester.

Dormand, J.R. and Prince, P.J., A family of embedded Runge-Kutta formulae, *Journal of Computational and Applied Mathematics*, 6 (1980) 19-26, 0377-0427.

Hairer, E., Norsett, S.P. and Wanner, G., (2000), *Solving ordinary differential equations I: Nonstiff problems*, Springer-Verlag, 3-540-56670-8, Berlin.

Hull, T.E., Enright, W.H., Fellen, B.M. and Sedgwick, A.E., Comparing numerical methods for ordinary differential equations, *SIAM Journal of Numerical Analysis*, 9, 4 (1972) 603-637, 0036-1429.

Kincaid, D. and Cheney, W., (2002), *Numerical Analysis: Mathematics of Scientific Computing, 3rd ed.*, Brooks/Cole, 0-534-38905-8, Pacific Grove.

Prentice, J.S.C., The RKGL method for the numerical solution of initial-value problems, *Journal of Computational and Applied Mathematics*, 213, 2 (2008) 477-487, 0377-0427.

Prentice, J.S.C.,Improving the efficiency of Runge-Kutta reintegration by means of the RKGL algorithm, (2009), In: *Advanced Technologies*, Kankesu Jayanthakumaran, (Ed.), 677-698, INTECH, 978-953-307-009-4, Vukovar.

Numerical Simulations - Applications, Examples and Theory
Edited by Prof. Lutz Angermann

ISBN 978-953-307-440-5
Hard cover, 520 pages
Publisher InTech
Published online 30, January, 2011
Published in print edition January, 2011

This book will interest researchers, scientists, engineers and graduate students in many disciplines, who make use of mathematical modeling and computer simulation. Although it represents only a small sample of the research activity on numerical simulations, the book will certainly serve as a valuable tool for researchers interested in getting involved in this multidisciplinary ï¬eld. It will be useful to encourage further experimental and theoretical researches in the above mentioned areas of numerical simulation.

How to reference
In order to correctly reference this scholarly work, feel free to copy and paste the following:

Justin Prentice (2011). A General Algorithm for Local Error Control in the RKrGLm Method, Numerical Simulations - Applications, Examples and Theory, Prof. Lutz Angermann (Ed.), ISBN: 978-953-307-440-5, InTech, Available from: http://www.intechopen.com/books/numerical-simulations-applications-examples-and-theory/a-general-algorithm-for-local-error-control-in-the-rkrglm-method

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen