RUNGE-KUTTA METHODS

Dr JSC Prentice

2018

Contents

Chapter 1

Numerical methods for initial-value problems

The basic idea behind solving an initial-value problem (IVP) such as

$$y' = f(x, y) \qquad y(x_0) = y_0 \tag{1.1}$$

numerically is to discretize the interval into nodes, usually equispaced; to replace the ordinary differential equation (ODE) with a difference equation; and to solve this difference equation at each node. Naturally, we require that the difference equation is a good approximation to the ODE. The quality of a such a numerical method is determined by approximation error, stability and computational efficiency, with accuracy and stability always taking precedence over efficiency. One of the most widely-used methods is the *Runge-Kutta* method, which is the subject of these notes.

Chapter 2

The Taylor method

Consider the Taylor method of order z for solving an IVP:

$$y_0 = \alpha$$
$$y\left(x_{i+1}\right) \approx y\left(x_i\right) + h\left[y'\left(x_i, y\left(x_i\right)\right) + \frac{h}{2}y''\left(x_i, y\left(x_i\right)\right) + \ldots + \frac{h^{z-1}}{z!}y^{(z)}\left(x_i, y\left(x_i\right)\right)\right]$$

Here, y_0 is the initial value of $y(x)$, the interval of integration has been discretized with the nodes denoted by x_i, and the node separation is h. The quantity in the square brackets is simply a truncated Taylor series, the residual term of which is $O(h^{z+1})$, which is the local error of the method.

Note that

$$y' = f$$
$$y'' = \frac{df}{dx} = f_x + f f_y.$$

Furthermore, y''' will require evaluation of f_{xx}, f_{xy} and f_{yy}, and, in general, higher derivatives of y will require evaluation of higher partial derivatives of f. This implies considerable computational complexity, particularly for high-order implementations of Taylor's method.

Chapter 3

Runge-Kutta methods

Runge-Kutta (RK) methods were developed in the late 1800s and early 1900s by Runge, Heun and Kutta. They came into their own in the 1960s after significant work by Butcher, and since then have grown into probably the most widely-used numerical methods for solving IVPs. In this section, we will provide a general definition of RK methods, and give a few examples of types of RK methods.

3.1 General definition

The most general definition of a Runge-Kutta method is

$$\begin{aligned} k_p &= f\left(x_i + c_p h, w_i + h\sum_{q=1}^{m} a_{pq}k_q\right) \qquad p = 1, 2, ..., m \\ w_{i+1} &= w_i + h\sum_{p=1}^{m} b_p k_p \end{aligned} \tag{3.1}$$

Such a method is said to have m stages (the k's). We note that if $a_{pq} = 0$ for all $p \leqslant q$, then the method is said to be *explicit*; otherwise, it is known as an *implicit* RK method. The number of stages is related to the order of the method. The symbol w is used here and throughout to indicate the approximate numerical solution, whereas the symbol y will denote the true solution.

As an example, consider the case with $m = 2$:

$$\begin{aligned} k_1 &= f\left(x_i + c_1 h, w_i + h a_{11}k_1 + h a_{12}k_2\right) \\ k_2 &= f\left(x_i + c_2 h, w_i + h a_{21}k_1 + h a_{22}k_2\right) \\ w_{i+1} &= w_i + h\left(b_1 k_1 + b_2 k_2\right). \end{aligned}$$

If $a_{11} = a_{12} = a_{22} = 0$, we have

$$\begin{aligned} k_1 &= f(x_i + c_1 h, w_i) \\ k_2 &= f(x_i + c_2 h, w_i + h a_{21} k_1) \\ w_{i+1} &= w_i + h(b_1 k_1 + b_2 k_2). \end{aligned}$$

which is an explicit method. Very often in explicit methods we also have $c_1 = 0$, so that $k_1 = f(x_i, w_i)$.

The difference between the two methods is clear: in the explicit method, we can evaluate the stages sequentially, whereas for the implicit method the stages are determined by solving a nonlinear system. Explicit methods are particularly easy to implement computationally, whereas implicit methods possess desirable stability properties that are absent in explicit methods.

It is clear that RK methods involve only evaluations of $f(x, y)$, and not of partial derivatives of f.

3.2 Order of Runge-Kutta methods

We say that an RK method is of order z if its global error is $O(h^z)$. An RK method of order z has a local error of order $z + 1$. The concepts of local and global error will be defined later in the section on error propagation in RK methods. There is a relationship between the number of stages in a method and its order; for current purposes it is sufficient to present the so-called Butcher barriers regarding the order of explicit RK methods:

Stages	2	3	4	$5 \leqslant m \leqslant 7$	$8 \leqslant m \leqslant 9$	$10 \leqslant m$
Best global error	$O(h^2)$	$O(h^3)$	$O(h^4)$	$O(h^{m-1})$	$O(h^{m-2})$	$O(h^{m-3})$

Furthermore, it is known that for any positive integer z, an explicit RK method exists with order z and m stages, where

$$m = \begin{cases} \frac{3z^2-10z+24}{8}, & z \text{ even} \\ \frac{3z^2-4z+9}{8}, & z \text{ odd} \end{cases}$$

Do not misunderstand the information presented here - the table shows the best order that can be attained for a given number of stages, whereas the formula

gives the number of stages in a method of given order. Additionally, the number of stages given by the formula is not optimal; there may exist a method of the given order with fewer stages, but certainly a method of order z and m stages (as given by the formula) is guaranteed to exist. For example, the table suggests that a method of order six must have at least seven stages (although an order six method with exactly seven stages may not actually exist), whereas the formula guarantees the existence of a method of order six with exactly nine stages. In fact, there does exist an order six method with eight stages. As regards implicit methods, we often find that the order is higher than the number of stages.

More detail regarding order will be given in the later section on the order conditions.

3.3 Butcher tableaux

Any RK method can be represented in a tabular format, known as a Butcher tableau. The general method has the tableau

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1m} \\ c_2 & a_{21} & a_{22} & & \vdots \\ \vdots & \vdots & & \ddots & \vdots \\ c_m & a_{m,1} & a_{m,2} & \cdots & a_{m,m} \\ \hline & b_1 & b_2 & \cdots & b_m \end{array} \tag{3.2}$$

The left column contains the c's for each stage, the right part contains the a's for each stage, and the b's are shown in the last line. Note that sometimes when indicating an explicit method that has $c_1 = 0$, the stage k_1 is not explicitly shown in the tableau, since it always has the Euler form. Some well-known RK methods are shown in the following tableaux:

$$\begin{array}{c|cc} 1 & 1 & \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array} \tag{3.3}$$

$$\begin{array}{c|cc} \frac{1}{2} & \frac{1}{2} & \\ \frac{3}{4} & 0 & \frac{3}{4} \\ \hline & \frac{2}{9} & \frac{3}{9} \quad \frac{4}{9} \end{array}$$

The first of these is a two-stage method (2nd order), and the second a three-stage method (3rd order). Both are explicit with $c_1 = 0$. An example of an implicit method is the Kuntzmann-Butcher method

$$\begin{array}{c|ccc} \frac{1}{2}-\frac{\sqrt{15}}{10} & \frac{5}{36} & \frac{8}{36}-\frac{\sqrt{15}}{15} & \frac{5}{36}-\frac{\sqrt{15}}{30} \\ \frac{1}{2} & \frac{5}{36}+\frac{\sqrt{15}}{24} & \frac{8}{36} & \frac{5}{36}-\frac{\sqrt{15}}{24} \\ \frac{1}{2}+\frac{\sqrt{15}}{10} & \frac{5}{36}+\frac{\sqrt{15}}{30} & \frac{8}{36}+\frac{\sqrt{15}}{15} & \frac{5}{36} \\ \hline & \frac{5}{18} & \frac{8}{18} & \frac{5}{18} \end{array}$$

which has three stages.

3.4 Embedded methods

An interesting class of RK methods are the embedded methods. One of the most famous is the Runge-Kutta-Fehlberg method, with tableau

$$\begin{array}{c|cccccc} \frac{1}{4} & \frac{1}{4} & & & & & \\ \frac{3}{8} & \frac{3}{32} & \frac{9}{32} & & & & \\ \frac{12}{13} & \frac{1932}{2197} & -\frac{7200}{2197} & \frac{7296}{2197} & & & \\ 1 & \frac{439}{216} & -8 & \frac{3680}{513} & -\frac{845}{4104} & & \\ \frac{1}{2} & -\frac{8}{27} & 2 & -\frac{3544}{2565} & \frac{1859}{4104} & -\frac{11}{40} & \\ \hline & \frac{25}{216} & 0 & \frac{1408}{2565} & \frac{2197}{4104} & -\frac{1}{5} & 0 \\ & \frac{16}{135} & 0 & \frac{6656}{12825} & \frac{28561}{56430} & -\frac{9}{50} & \frac{2}{55} \end{array} \tag{3.4}$$

The first row of b's is a fourth-order method, while the second row of b's is a fifth-order method. In other words, a fifth-order method is embedded in the stages that are used to give the fourth-order method. This simply means that two solutions can be computed without the need to evaluate any additional stages, which has positive implications regarding computational efficiency. We will see later that certain error control algorithms require the evaluation of two methods of different order, and so embedded methods are potentially very useful.

Although the RKF45 method above is explicit, embedded methods can also be implicit.

3.5 Reflected methods

A reflected RK method allows w_i to be determined, given (x_{i+1}, w_{i+1}). In other words, a reflected method integrates in the negative x-direction - 'backwards', so to speak.

Given any RK method (3.2), the associated reflected method has the tableau

$$\begin{array}{c|cccc} c_1 - \sum_{i=1}^{m} b_i & a_{11} - b_1 & a_{12} - b_2 & \cdots & a_{1m} - b_m \\ c_2 - \sum_{i=1}^{m} b_i & a_{21} - b_1 & a_{22} - b_2 & & \vdots \\ \vdots & \vdots & & \ddots & \vdots \\ c_m - \sum_{i=1}^{m} b_i & a_{m,1} - b_1 & a_{m,2} - b_2 & \cdots & a_{m,m} - b_m \\ \hline & -b_1 & -b_2 & \cdots & -b_m \end{array} \tag{3.5}$$

For such a method we have

$$\begin{aligned} w_i &= w_{i+1} + h \sum_{p=1}^{m} (-b_p)\, k_p \\ &= w_{i+1} - h \sum_{p=1}^{m} b_p k_p \end{aligned}$$

This can be rearranged to give

$$w_{i+1} = w_i + h \sum_{p=1}^{m} b_p k_p$$

which has the tableau

$$\begin{array}{c|cccc} -c_1 + \sum_{i=1}^{m} b_i & -a_{11} + b_1 & -a_{12} + b_2 & \cdots & -a_{1m} + b_m \\ -c_2 + \sum_{i=1}^{m} b_i & -a_{21} + b_1 & -a_{22} + b_2 & & \vdots \\ \vdots & \vdots & & \ddots & \vdots \\ -c_m + \sum_{i=1}^{m} b_i & -a_{m,1} + b_1 & -a_{m,2} + b_2 & \cdots & -a_{m,m} + b_m \\ \hline & b_1 & b_2 & \cdots & b_m \end{array} \tag{3.6}$$

which is just the reflected method with all signs reversed. Clearly, this method is, in general, not the same as the original method (3.2). To regain the original

method, we must apply the reflection algorithm (3.5) to the reflected method itself. The method (3.6) is the positive x-direction version of (3.5).

As an example, consider the method (3.3), which we present here with all coefficients present:

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1 & 0 \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

The reflected form is

$$\begin{array}{c|cc} -1 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{1}{2} & -\frac{1}{2} \\ \hline & -\frac{1}{2} & -\frac{1}{2} \end{array}$$

and the 'forward' form is

$$\begin{array}{c|cc} 1 & \frac{1}{2} & \frac{1}{2} \\ 0 & -\frac{1}{2} & \frac{1}{2} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

This is an implicit method; the question of whether or not it is a second-order method will be discussed in a later section.

Chapter 4

Construction of Runge-Kutta methods using Taylor series

Constructing an RK method requires determining the coefficients $\{a, b, c\}$ in (3.2). This is done by expanding the RK solution in a Taylor series, and then demanding equivalence with the Taylor method up to some prescribed order. Equating coefficients of the various terms in the two series yields a set of nonlinear equations - known as the *order conditions* - that must be solved to yield the coefficients.

4.1 The explicit second-order method

We illustrate the process by means of the explicit second-order method, with $w_0 = y_0$,

$$\left.\begin{aligned} k_1 &= hf(x_0, y_0) \\ k_2 &= hf(x_0 + c_2 h, y_0 + a_{21} k_1) \\ w_1 &= y_0 + b_1 k_1 + b_2 k_2 \end{aligned}\right\} \tag{4.1}$$

We note that the term k_1 represents the Euler approximation. The term $b_2 k_2$ may thus be regarded as an attempt to improve this approximation. The four unknowns $b_1,\ b_2,\ c_2$ and a_{21} are obtained by requiring that (4.1) agrees with a

Taylor expansion of second order:

$$\begin{aligned} w_1 \approx y(x_1) &= y(x_0 + h) \\ &= y(x_0) + hy'(x_0) + \frac{1}{2!}h^2 y''(x_0) + \dots \\ &= y(x_0) + hf(x_0, y_0) + \frac{1}{2}h^2 \left[f_x + f_y f\right]_{(x_0,y_0)} + \dots \end{aligned} \tag{4.2}$$

where we have used

$$y'' = \frac{df(x,y)}{dx} = \frac{\partial f}{\partial x} + \frac{\partial f}{\partial y}\frac{dy}{dx} = f_x + f_y f.$$

Equation (4.1) is also expanded to second order:

$$\begin{aligned} w_1 &= y_0 + b_1 hf(x_0, y_0) + b_2 hf(x_0 + c_2 h, y_0 + a_{21} k_1) \\ &= y_0 + b_1 hf(x_0, y_0) + b_2 h\left[f(x_0, y_0 + a_{21}k_1) + c_2 h f_x(x_0, y_0 + a_{21}k_1) + \dots\right] \\ &= y_0 + b_1 hf(x_0, y_0) \\ &\quad + b_2 h\left[\{f(x_0, y_0) + a_{21}k_1 f_y(x_0, y_0) + \dots\} + c_2 h\left\{f_x(x_0, y_0) + \dots\right\}\right] \\ &= y_0 + (b_1 + b_2)\, hf(x_0, y_0) + h^2\left[a_{21}b_2 f_y f + c_2 b_2 f_x\right]_{(x_0,y_0)} + \dots \end{aligned} \tag{4.3}$$

Equating (4.2) and (4.3), term for term, yields the order conditions

$$b_1 + b_2 = 1 \qquad c_2 b_2 = \tfrac{1}{2} \qquad a_{21} b_2 = \tfrac{1}{2}.$$

The last two of these order conditions give

$$c_2 = a_{21}.$$

Hence, there are effectively only two order conditions

$$b_1 + b_2 = 1 \qquad c_2 b_2 = \tfrac{1}{2}$$

and the RK method has the tableau

c_2	c_2	
	$1 - \frac{1}{2c_2}$	$\frac{1}{2c_2}$

which represents a one-parameter family of methods. Choosing $c_2 = 1$ yields (3.3). Of course, any value of c_2 is allowed, but we generally choose the values of any free parameters so that the b's are all positive.

4.2 The reflected second-order method

The forward version of the reflected second-order method studied earlier can be written as

$$\begin{aligned} k_1 &= f\left(x_{i+1}, w_{i+1}\right) \\ k_2 &= f\left(x_{i+1} - h, w_{i+1} - hk_1\right) \\ w_{i+1} &= w_i + \frac{h}{2}\left(k_1 + k_2\right) \end{aligned}$$

as shown in the Supplementary Notes.

Using the expression for w_{i+1}, we have

$$\begin{aligned} k_1 &= f\left(x_i + h, w_i + h\left(\frac{k_1}{2} + \frac{k_2}{2}\right)\right) \\ k_2 &= f\left(x_i, w_i + h\left(-\frac{k_1}{2} + \frac{k_2}{2}\right)\right) \end{aligned}$$

which gives the implicit tableau

$$\begin{array}{c|cc} 1 & \frac{1}{2} & \frac{1}{2} \\ 0 & -\frac{1}{2} & \frac{1}{2} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

4.3 The classical explicit fourth-order method

The classical explicit fourth-order method has the tableau

$$\begin{array}{c|cccc} c_2 & c_2 & & & \\ c_3 & 0 & c_3 & & \\ c_4 & 0 & 0 & c_4 & \\ \hline & b_1 & b_2 & b_3 & b_4 \end{array} \tag{4.4}$$

Note the very specific structure of the tableau. The derivation of the order conditions, using Taylor series equivalence, is shown in the Supplementary Notes. It is clear that the derivation is complicated, and one appreciates that for a general RK method, deriving the order conditions through Taylor series equivalence, particularly for high order methods, is destined to be both complex and cumbersome.

Chapter 5

Construction of Runge-Kutta methods using rooted trees

Determining the order conditions by direct comparison of Taylor series is destined to be an arduous task, particularly for methods of high order. One expects, however, that the clearly defined structure of the RK methods should result in a similarly structured Taylor expansion, so that some underlying pattern should exist. This pattern has been identified, and has led to the use of *rooted trees* as a means of determining the order conditions, without explicitly forming the relevant Taylor series.

5.1 Rooted trees for Runge-Kutta methods

A rooted tree of order z is a graph with z vertices connected in a certain way. In such a graph, the lines joining pairs of vertices are known as *edges*. To draw such a rooted tree for an RK method, we follow a simple set of rules:

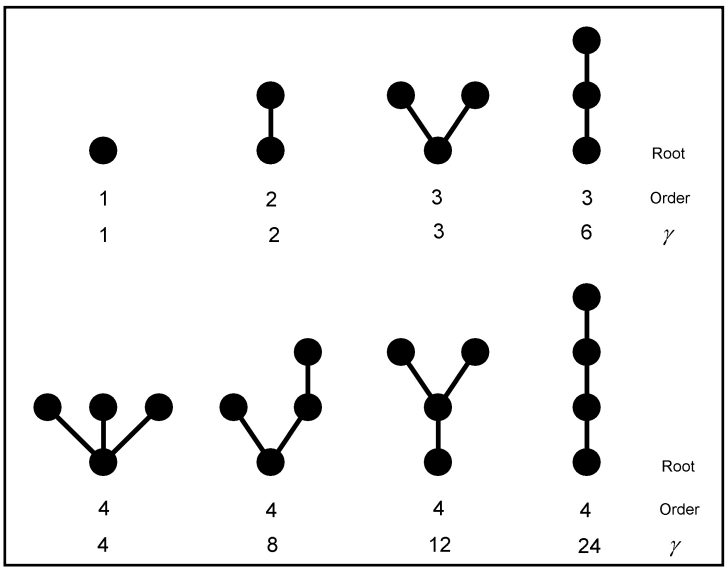
1. One vertex is placed at the lowest point in the graph. This vertex is known as the *root*.

2. The remaining $z - 1$ vertices are drawn above the root.

3. These vertices are connected to each other and to the root by means of edges, such that no edges cross, only one edge joins any pair of vertices, there are no loops (closed portions) in the graph, and all edges are drawn in a 'generally upward' direction (not horizontal or downwards).

4. The resulting graph is fully connected (any vertex can be reached from any other vertex by tracing a path along the edges).

5. All possible combinations of such fully connected graphs must be drawn. The higher the value of z, the more such graphs exist. All possible rooted trees up to order four are shown in the figure below. The 'generally upward' character of these graphs is evident.



6. The next task is to label the vertices. Label the root i. Then label all other vertices $\{j, k, l, \ldots\}$. Each vertex must have a unique label.

7. Now label each edge a_{pq}, where p and q are the labels of the vertices at each end of the edge, with p being the vertex closest to the root. Note that $p, q \in \{i, j, k, l, \ldots\}$.

8. Form the product

$$\Phi_a \equiv b_i \prod_{\text{all edges}} a_{pq}.$$

9. Now identify those vertices, other than the root, that only have one edge attached. These vertices are called *leaves*. If a leaf is at the end of the edge a_{pq}, replace a_{pq} in the above product with c_p, to form

$$\Phi_c \equiv b_i \prod_{\substack{\text{all 'internal'} \\ \text{edges}}} a_{pq} \prod_{\text{all leaves}} c_p.$$

These products will form one side of the order conditions.

10. We now assign an integer to each vertex. Count the number of vertices that can be reached from a particular vertex v, by tracing upwards from v. The integer assigned, n_v, is this number plus one. The 'generally upward' depiction of the trees is vitally important for this step. Obviously, the integer assigned to the root vertex will be z.

11. Determine
$$\gamma = \prod_{\text{vertices}} n_v.$$

12. The order conditions are now given by
$$\sum_{\{i,j,k,l,\ldots\}} \Phi_a = \frac{1}{\gamma} \tag{5.1}$$
$$\sum_{\{i,j,k,l,\ldots\}} \Phi_c = \frac{1}{\gamma} \tag{5.2}$$
where each element of $\{i, j, k, l, \ldots\}$ has the values $\{1, 2, \ldots, m\}$, where m is the number of stages in the method. Note that each graph will yield two order conditions.

5.2 The explicit second-order method

For this method we must find the order conditions for rooted trees of order one and two only. Hence,
$$\sum b_i = 1$$
$$\sum b_i a_{ij} = \frac{1}{2}$$
$$\sum b_i c_i = \frac{1}{2}$$
with $i = \{1, 2\}$, $j = \{1, 2\}$. These give
$$b_1 + b_2 = 1$$
$$b_1 (a_{11} + a_{12}) + b_2 (a_{21} + a_{22}) = \frac{1}{2}$$
$$b_1 c_1 + b_2 c_2 = \frac{1}{2}.$$

However, form the structure imposed on the method, we have $a_{11} = a_{12} = a_{22} = c_1 = 0$, and so

$$\begin{aligned} b_1 + b_2 &= 1 \\ b_2 a_{21} &= \frac{1}{2} \\ b_2 c_2 &= \frac{1}{2} \end{aligned}$$

exactly as obtained by equivalence of Taylor series.

5.3 The explicit third-order method

The explicit third-order method

$$\begin{array}{c|ccc} c_2 & a_{21} & & \\ c_3 & a_{31} & a_{32} & \\ \hline & b_1 & b_2 & b_3 \end{array}$$

has the order conditions

$$\begin{array}{lll} \sum b_i = 1 & \sum b_i a_{ij} = \frac{1}{2} & \sum b_i c_i = \frac{1}{2} \\ \sum b_i a_{ij} a_{ik} = \frac{1}{3} & \sum b_i c_i^2 = \frac{1}{3} & \\ \sum b_i a_{ij} a_{jk} = \frac{1}{6} & \sum b_i a_{ij} c_j = \frac{1}{6} & \end{array}$$

with $i = \{1, 2, 3\}$, $j = \{1, 2, 3\}$, $k = \{1, 2, 3\}$. These arise from the rooted trees of orders one, two and three.

The explicit third-order method has $a_{11} = a_{12} = a_{13} = a_{22} = a_{23} = a_{33} = c_1 = 0$, and so

$$\begin{aligned} b_1 + b_2 + b_3 &= 1 \\ b_2 c_2 + b_3 c_3 &= \frac{1}{2} \\ b_2 c_2^2 + b_3 c_3^2 &= \frac{1}{3} \end{aligned}$$

$$
\begin{aligned}
b_2 a_{21} + b_3 (a_{31} + a_{32}) &= \frac{1}{2} \\
b_2 a_{21} a_{21} + b_3 \underbrace{(a_{31} a_{31} + a_{31} a_{32} + a_{32} a_{31} + a_{32} a_{32})}_{(a_{31}+a_{32})^2} &= \frac{1}{3} \\
b_3 a_{32} c_2 &= \frac{1}{6} \\
b_3 a_{32} a_{21} &= \frac{1}{6}
\end{aligned}
$$

From the last two of these conditions, we have

$$c_2 = a_{21}.$$

From the second and the fourth equations, we have

$$c_3 = a_{31} + a_{32}$$

and so the third and fifth equations are consistent.

All this gives the order conditions for the explicit third-order method as

$$
\begin{aligned}
b_1 + b_2 + b_3 &= 1 \qquad & b_2 c_2 + b_3 c_3 &= \frac{1}{2} \\
b_2 c_2^2 + b_3 c_3^2 &= \frac{1}{3} & & \\
b_3 a_{32} c_2 &= \frac{1}{6} & &
\end{aligned}
$$

$$
\begin{aligned}
c_2 &= a_{21} \\
c_3 &= a_{31} + a_{32}.
\end{aligned}
$$

Note that the c's are the sums of the a's on the corresponding rows of the tableau. This is, in fact, a general result and is summarized by

$$c_p = \sum_{q=1}^{m} a_{pq} \quad p = 1, 2, \ldots, m. \tag{5.3}$$

These relationships are entirely consistent with the order conditions derived from the rooted trees. Usually, we give the order conditions as (5.2) and (5.3). We usually omit (5.1) because they are implied by (5.2) and (5.3). The conditions (5.3) also hold for implicit methods.

5.4 The explicit fourth-order method

The explicit fourth-order method

$$\begin{array}{c|cccc} c_2 & a_{21} & & & \\ c_3 & a_{31} & a_{32} & & \\ c_4 & a_{41} & a_{42} & a_{43} & \\ \hline & b_1 & b_2 & b_3 & b_4 \end{array}$$

has the order conditions (from the rooted trees of orders one to four)

$$\begin{array}{ll} b_1 + b_2 + b_3 + b_4 = 1 & b_3 a_{32} c_2 + b_4 a_{42} c_2 + b_4 a_{43} c_3 = \frac{1}{6} \\ b_2 c_2 + b_3 c_3 + b_4 c_4 = \frac{1}{2} & b_3 a_{32} c_2 c_3 + b_4 a_{42} c_2 c_4 + b_4 a_{43} c_3 c_4 = \frac{1}{8} \\ b_2 c_2^2 + b_3 c_3^2 + b_4 c_4^2 = \frac{1}{3} & b_3 a_{32} c_2^2 + b_4 a_{42} c_2^2 + b_4 a_{43} c_3^2 = \frac{1}{12} \\ b_2 c_2^3 + b_3 c_3^3 + b_4 c_4^3 = \frac{1}{4} & b_4 a_{32} a_{43} c_2 = \frac{1}{24} \end{array}$$

together with, from (5.3),

$$\sum_{q=1}^{4} a_{pq} = c_p \qquad p = 2, 3, 4 \tag{5.4}$$

which gives

$$\begin{aligned} c_2 &= a_{21} \\ c_3 &= a_{31} + a_{32} \\ c_4 &= a_{41} + a_{42} + a_{43}. \end{aligned}$$

The order conditions for the explicit fourth-order method are given in detail in the Supplementary Notes.

It is a simple matter to determine the order conditions for the classic fourth-order method, using the conditions above. Moreover, we see from (4.4) that the structure of the method reflects the conditions (5.4).

5.5 The reflected second-order method

We now consider the order of the method

$$\begin{array}{c|cc} 1 & \frac{1}{2} & \frac{1}{2} \\ 0 & -\frac{1}{2} & \frac{1}{2} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array} \tag{5.5}$$

which is the 'forward' form of the reflection of the explicit second-order method. The first- and second-order conditions for the implicit method

$$\begin{array}{c|cc} c_1 & a_{11} & a_{12} \\ c_2 & a_{21} & a_{22} \\ \hline & b_1 & b_2 \end{array}$$

are

$$\begin{aligned} b_1 + b_2 &= 1 \\ b_1 c_1 + b_2 c_2 &= \frac{1}{2} \\ c_1 &= a_{11} + a_{12} \\ c_2 &= a_{21} + a_{22} \end{aligned}$$

and clearly the coefficients in (5.5) satisfy these conditions. Hence, (5.5) is at least second order. The relevant third-order conditions are

$$\begin{aligned} b_1 c_1^2 + b_2 c_2^2 &= \frac{1}{3} \\ b_1 a_{11} c_1 + b_1 a_{12} c_2 + b_2 a_{21} c_1 + b_2 a_{22} c_2 &= \frac{1}{6} \end{aligned}$$

but these are not satisfied by the coefficients in (5.5). Hence, (5.5) is a second-order implicit method. We have presented this example because it should not be assumed that an implicit method with m stages is automatically an mth-order method. Some implicit methods have order greater than their number of stages, although that is not the case in this particular example.

Chapter 6

Consistency, convergence and stability

Consistency refers to the ability of the method to faithfully reproduce the original differential equation, in the limit $h \to 0$. Convergence refers to the fact that the numerical solution tends to the exact solution, as $h \to 0$. A *stable* method is one for which the numerical solution does not diverge 'dramatically' from the exact solution under iteration.

6.1 Consistency

We demonstrate consistency for the general RK method (3.1), which we reproduce here for convenience.

$$k_p = f\left(x_i + c_p h, w_i + h\sum_{q=1}^{m} a_{pq}k_q\right) \qquad p = 1, 2, ..., m$$
$$w_{i+1} = w_i + h\sum_{p=1}^{m} b_p k_p$$

Manipulating the last line of this definition gives

$$\frac{w_{i+1} - w_i}{h} = \sum_{p=1}^{m} b_p k_p. \tag{6.1}$$

Taking the limit $h \to 0$ gives

$$\lim_{h\to 0} k_p = f(x_i, y_i)$$
$$\lim_{h\to 0} \frac{w_{i+1} - w_i}{h} = \lim_{h\to 0} \frac{y_{i+1} - y_i + O(h^z)}{h} = y'(x_i)$$

where we have assumed that a global error of $O(h^z)$ exists in w_i and w_{i+1}. Hence, from (6.1) we have

$$y'(x_i) = \sum_{p=1}^{m} b_p f(x_i, y_i) = f(x_i, y_i) \sum_{p=1}^{m} b_p = f(x_i, y_i)$$

and so the original differential equation is obtained.

6.2 Convergence

An RK method of order z has a global error $O(h^z)$. Obviously, this error must tend to zero as $h \to 0$, implying convergence. It is worth noting that all RK methods have a global error $O(h^z)$ for z at least equal to one, and so all RK methods are convergent. We will study the global error in more detail in a later section.

6.3 Stability

Say the method has an error of the form

$$\Delta_i = [p(h)]^i . \tag{6.2}$$

where i indicates the iteration count, and $p(h)$ is a polynomial in h. The presence of such a term indicates potential instability. If

$$|p(h)|^i \to \infty$$

under iteration ($i \to \infty$), then, no matter how small h is made, the error grows without bound. Nevertheless, we often find that there exists a critical value of h, such that for stepsizes on one side of this critical value, $|p(h)|^i \to \infty$ as $i \to \infty$, but for stepsizes on the other side of the critical value, $|p(h)|^i \to 0$ as $i \to \infty$, which corresponds to stable behaviour. In a later section, we will consider RK methods applied to stiff differential equations, and instability/stability in the sense described here will be demonstrated. Suffice it to say at this juncture, an unstable method cannot be convergent.

Chapter 7

Implementation for systems

To solve the d-dimensional system

$$\mathbf{y}' = \mathbf{f}(x, \mathbf{y}) \qquad \mathbf{y}(x_0) = \mathbf{y}_0 \qquad x_0 \leqslant x \leqslant L \tag{7.1}$$

using an RK method, we consider the definition of an RK method in vector form

$$\mathbf{k}_p = \mathbf{f}\left(x_i + c_p h, \mathbf{w}_i + h \sum_{q=1}^{m} a_{pq}\mathbf{k}_q\right) \qquad p = 1, 2, ..., m$$
$$\mathbf{w}_{i+1} = \mathbf{w}_i + h \sum_{p=1}^{m} b_p \mathbf{k}_p$$

wherein each stage $\mathbf{k}_p$ is a $d \times 1$ vector. Each stage vector contains d entries, and so there are md entries that must be computed at each step. However, each stage equation is a vector equation with d components, and so there are md equations for the md stage vector entries. For an explicit method, the stages are computed sequentially; for an implicit method, a nonlinear solver such as Newton's method would have to be used.

7.1 A two-dimensional system

As an example, consider the solution of

$$\mathbf{y}' \equiv \begin{bmatrix} y_1' \\ y_2' \end{bmatrix} = \begin{bmatrix} f_1(x, y_1, y_2) \\ f_2(x, y_1, y_2) \end{bmatrix} \equiv \mathbf{f}(x, \mathbf{y})$$

using the two-stage RK method

$$\begin{array}{c|cc} c_1 & a_{11} & a_{12} \\ c_2 & a_{21} & a_{22} \\ \hline & b_1 & b_2 \end{array}$$

We have

$$\begin{aligned} \mathbf{k}_1 &= \mathbf{f}\left(x_i + c_1 h, \mathbf{w}_i + h a_{11}\mathbf{k}_1 + h a_{12}\mathbf{k}_2\right) \\ \mathbf{k}_2 &= \mathbf{f}\left(x_i + c_2 h, \mathbf{w}_i + h a_{21}\mathbf{k}_1 + h a_{22}\mathbf{k}_2\right) \end{aligned}$$

or, more explicitly,

$$\begin{aligned} \mathbf{k}_1 = \begin{bmatrix} k_{11} \\ k_{12} \end{bmatrix} &= \begin{bmatrix} f_1\left(x_i + c_1 h, w_{i1} + h a_{11} k_{11} + h a_{12} k_{21}, w_{i2} + h a_{11} k_{12} + h a_{12} k_{22}\right) \\ f_2\left(x_i + c_1 h, w_{i1} + h a_{11} k_{11} + h a_{12} k_{21}, w_{i2} + h a_{11} k_{12} + h a_{12} k_{22}\right) \end{bmatrix} \\ \mathbf{k}_2 = \begin{bmatrix} k_{21} \\ k_{22} \end{bmatrix} &= \begin{bmatrix} f_1\left(x_i + c_2 h, w_{i1} + h a_{21} k_{11} + h a_{22} k_{21}, w_{i2} + h a_{21} k_{12} + h a_{22} k_{22}\right) \\ f_2\left(x_i + c_2 h, w_{i1} + h a_{21} k_{11} + h a_{22} k_{21}, w_{i2} + h a_{21} k_{12} + h a_{22} k_{22}\right) \end{bmatrix} \end{aligned}$$

which gives four nonlinear equations for the unknowns k_{11}, k_{12}, k_{21} and k_{22}. Using Newton's method to solve the system

$$\begin{bmatrix} g_1 \\ g_2 \\ g_3 \\ g_4 \end{bmatrix} \equiv \begin{bmatrix} k_{11} - f_1\left(x_i + c_1 h, w_{i1} + h a_{11} k_{11} + h a_{12} k_{21}, w_{i2} + h a_{11} k_{12} + h a_{12} k_{22}\right) \\ k_{12} - f_2\left(x_i + c_1 h, w_{i1} + h a_{11} k_{11} + h a_{12} k_{21}, w_{i2} + h a_{11} k_{12} + h a_{12} k_{22}\right) \\ k_{21} - f_1\left(x_i + c_2 h, w_{i1} + h a_{21} k_{11} + h a_{22} k_{21}, w_{i2} + h a_{21} k_{12} + h a_{22} k_{22}\right) \\ k_{22} - f_2\left(x_i + c_2 h, w_{i1} + h a_{21} k_{11} + h a_{22} k_{21}, w_{i2} + h a_{21} k_{12} + h a_{22} k_{22}\right) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

involves the Jacobian

$$\begin{bmatrix} \frac{\partial g_1}{\partial k_{11}} & \frac{\partial g_1}{\partial k_{12}} & \frac{\partial g_1}{\partial k_{21}} & \frac{\partial g_1}{\partial k_{22}} \\ \frac{\partial g_2}{\partial k_{11}} & \frac{\partial g_2}{\partial k_{12}} & \frac{\partial g_2}{\partial k_{21}} & \frac{\partial g_2}{\partial k_{22}} \\ \frac{\partial g_3}{\partial k_{11}} & \frac{\partial g_3}{\partial k_{12}} & \frac{\partial g_3}{\partial k_{21}} & \frac{\partial g_3}{\partial k_{22}} \\ \frac{\partial g_4}{\partial k_{11}} & \frac{\partial g_4}{\partial k_{12}} & \frac{\partial g_4}{\partial k_{21}} & \frac{\partial g_4}{\partial k_{22}} \end{bmatrix}$$

and, of course, Newton's method would have to used at each step.

In the explicit case, we have $c_1 = a_{11} = a_{12} = a_{22} = 0$, which gives

$$\begin{aligned} \mathbf{k}_1 = \begin{bmatrix} k_{11} \\ k_{12} \end{bmatrix} &= \begin{bmatrix} f_1\left(x_i, w_{i1}, w_{i2}\right) \\ f_2\left(x_i, w_{i1}, w_{i2}\right) \end{bmatrix} \\ \mathbf{k}_2 = \begin{bmatrix} k_{21} \\ k_{22} \end{bmatrix} &= \begin{bmatrix} f_1\left(x_i + c_2 h, w_{i1} + h a_{21} k_{11}, w_{i2} + h a_{21} k_{12}\right) \\ f_2\left(x_i + c_2 h, w_{i1} + h a_{21} k_{11}, w_{i2} + h a_{21} k_{12}\right) \end{bmatrix} \end{aligned}$$

and so the stages can be computed sequentially.

Chapter 8

Error propagation in Runge-Kutta methods

We now define local and global errors in RK methods formally, and study the propagation of local error in the implementation of an explicit RK method. We will also find the relationship between local and global error.

In this section, boldface type, as in $\mathbf{v}$, indicates a $d \times 1$ vector, and boldface type with caret, as in $\widehat{\mathbf{M}}$, denotes a $d \times d$ matrix. Also, we will denote an explicit RK method for solving the d-dimensional system (7.1) by

$$\mathbf{w}_{i+1} = \mathbf{w}_i + h\mathbf{F}\left(x_i, \mathbf{w}_i\right)$$

where $\mathbf{w}_i$ denotes the numerical approximation to $\mathbf{y}\left(x_i\right)$ and $\mathbf{F}\left(x, \mathbf{y}\right)$ is a function associated with the particular RK method. Indeed, $\mathbf{F}$ is simply the linear combination of the stages

$$\sum_{p=1}^{m} b_p \mathbf{k}_p.$$

8.1 Local and global errors

We define the global error in a numerical solution at x_{i+1} by

$$\boldsymbol{\Delta}_{i+1} \equiv \mathbf{w}_{i+1} - \mathbf{y}_{i+1},$$

and the local error at x_{i+1} by

$$\boldsymbol{\varepsilon}_{i+1} \equiv \left[\mathbf{y}_i + h\mathbf{F}\left(x_i, \mathbf{y}_i\right)\right] - \mathbf{y}_{i+1}. \tag{8.1}$$

In the above, $\mathbf{y}_i$ denotes the true solution $\mathbf{y}(x_i)$, and similarly for $\mathbf{y}_{i+1}$. Note the use of the exact value $\mathbf{y}_i$ in the bracketed term in (8.1).

8.2 Error propagation in explicit methods

For the sake of generality we will assume an error $\boldsymbol{\Delta}_0$ exists in the initial value, although in most practical cases $\boldsymbol{\Delta}_0 = \mathbf{0}$. We have

$$\begin{aligned} \mathbf{w}_1 &= \mathbf{y}_0 + \boldsymbol{\Delta}_0 + h\mathbf{F}(x_0, \mathbf{y}_0 + \boldsymbol{\Delta}_0) \\ \Rightarrow\ \boldsymbol{\Delta}_1 &= [\,\mathbf{y}_0 + h\mathbf{F}(x_0, \mathbf{y}_0) - \mathbf{y}_1] + \left[\,\widehat{\mathbf{I}} + h\widehat{\mathbf{F}}_y(x_0, \boldsymbol{\xi}_0)\right]\boldsymbol{\Delta}_0 \\ &= \boldsymbol{\varepsilon}_1 + \widehat{\boldsymbol{\alpha}}_0\boldsymbol{\Delta}_0 \end{aligned}$$

where $\widehat{\boldsymbol{\alpha}}_0$ has been implicitly defined. In the above we use the symbol $\boldsymbol{\xi}_0$ in $\widehat{\mathbf{F}}_y(x_0, \boldsymbol{\xi}_0)\,\boldsymbol{\Delta}_0$ simply to denote an appropriate set of constants such that $\widehat{\mathbf{F}}_y(x_0, \boldsymbol{\xi}_0)\,\boldsymbol{\Delta}_0$ is the residual term in the first-order Taylor expansion of $\mathbf{F}(x_0, \mathbf{y}_0 + \boldsymbol{\Delta}_0)$. Moreover, $\widehat{\mathbf{F}}_y$ is the Jacobian

$$\widehat{\mathbf{F}}_y = \begin{bmatrix} \frac{\partial F_1}{\partial y_1} & \cdots & \frac{\partial F_1}{\partial y_d} \\ \vdots & \ddots & \vdots \\ \frac{\partial F_d}{\partial y_1} & \cdots & \frac{\partial F_d}{\partial y_d} \end{bmatrix}$$

where $\{F_1, F_2, \ldots, F_d\}$ are the components of $\mathbf{F}$. The matrix $\widehat{\mathbf{I}}$ is the identity matrix.

For $\boldsymbol{\Delta}_2$ we have

$$\begin{aligned} \mathbf{w}_2 &= \mathbf{w}_1 + h\mathbf{F}(x_1, \mathbf{w}_1) \\ \Rightarrow\ \mathbf{y}_2 + \boldsymbol{\Delta}_2 &= [\,\mathbf{y}_1 + \boldsymbol{\Delta}_1] + h\mathbf{F}(x_1, \mathbf{y}_1 + \boldsymbol{\Delta}_1) \\ &= [\,\mathbf{y}_1 + \boldsymbol{\Delta}_1] + h\mathbf{F}(x_1, \mathbf{y}_1) + h\widehat{\mathbf{F}}_y(x_1, \boldsymbol{\xi}_1)\,\boldsymbol{\Delta}_1 \\ \Rightarrow\ \boldsymbol{\Delta}_2 &= [\,\mathbf{y}_1 + h\mathbf{F}(x_1, \mathbf{y}_1) - \mathbf{y}_2] + \left[\,\widehat{\mathbf{I}} + h\widehat{\mathbf{F}}_y(x_1, \boldsymbol{\xi}_1)\right]\boldsymbol{\Delta}_1 \\ &= \boldsymbol{\varepsilon}_2 + \widehat{\boldsymbol{\alpha}}_1\boldsymbol{\Delta}_1 \\ &= \boldsymbol{\varepsilon}_2 + \widehat{\boldsymbol{\alpha}}_1\boldsymbol{\varepsilon}_1 + \widehat{\boldsymbol{\alpha}}_1\widehat{\boldsymbol{\alpha}}_0\boldsymbol{\Delta}_0. \end{aligned}$$

It is easy to show that

$$\begin{aligned} \boldsymbol{\Delta}_3 &= \boldsymbol{\varepsilon}_3 + \widehat{\boldsymbol{\alpha}}_2\boldsymbol{\varepsilon}_2 + \widehat{\boldsymbol{\alpha}}_2\widehat{\boldsymbol{\alpha}}_1\boldsymbol{\varepsilon}_1 + \widehat{\boldsymbol{\alpha}}_2\widehat{\boldsymbol{\alpha}}_1\widehat{\boldsymbol{\alpha}}_0\boldsymbol{\Delta}_0 \\ \boldsymbol{\Delta}_4 &= \boldsymbol{\varepsilon}_4 + \widehat{\boldsymbol{\alpha}}_3\boldsymbol{\varepsilon}_3 + \widehat{\boldsymbol{\alpha}}_3\widehat{\boldsymbol{\alpha}}_2\boldsymbol{\varepsilon}_2 + \widehat{\boldsymbol{\alpha}}_3\widehat{\boldsymbol{\alpha}}_2\widehat{\boldsymbol{\alpha}}_1\boldsymbol{\varepsilon}_1 + \widehat{\boldsymbol{\alpha}}_3\widehat{\boldsymbol{\alpha}}_2\widehat{\boldsymbol{\alpha}}_1\widehat{\boldsymbol{\alpha}}_0\boldsymbol{\Delta}_0 \end{aligned}$$

and, in general,

$$\begin{aligned}\boldsymbol{\Delta}_n = &\boldsymbol{\varepsilon}_n + \widehat{\boldsymbol{\alpha}}_{n-1}\boldsymbol{\varepsilon}_{n-1} + \ldots + \widehat{\boldsymbol{\alpha}}_{n-1}\widehat{\boldsymbol{\alpha}}_{n-2}\cdots\widehat{\boldsymbol{\alpha}}_2\widehat{\boldsymbol{\alpha}}_1\boldsymbol{\varepsilon}_1 \\ &+ \widehat{\boldsymbol{\alpha}}_{n-1}\widehat{\boldsymbol{\alpha}}_{n-2}\cdots\widehat{\boldsymbol{\alpha}}_2\widehat{\boldsymbol{\alpha}}_1\widehat{\boldsymbol{\alpha}}_0\boldsymbol{\Delta}_0\end{aligned}$$

where

$$\widehat{\boldsymbol{\alpha}}_k = \widehat{\mathbf{I}} + h\widehat{\mathbf{F}}_y\left(x_k, \boldsymbol{\xi}_k\right)$$

in which, for each k, $\boldsymbol{\xi}_k$ is an appropriate set of constants (as explained above).

If $\left\|h\widehat{\mathbf{F}}_y\left(x_k, \boldsymbol{\xi}_k\right)\right\|$ is small then $\widehat{\boldsymbol{\alpha}}_k \approx \widehat{\mathbf{I}}$, and so

$$\boldsymbol{\Delta}_n \approx \boldsymbol{\Delta}_0 + \sum_{i=1}^{n} \boldsymbol{\varepsilon}_i$$

but this is generally not expected to be the case, particularly if $\widehat{\mathbf{F}}_y\left(x_k, \boldsymbol{\xi}_k\right)$ has large norm. Furthermore, if the $\widehat{\boldsymbol{\alpha}}$'s have norm greater than one, then the term in $\boldsymbol{\varepsilon}_1$ (or $\boldsymbol{\Delta}_0$ if $\boldsymbol{\Delta}_0$ is nonzero) could make the most significant contribution to the global error.

If the local errors have the form

$$\boldsymbol{\varepsilon}_i = \boldsymbol{\beta}_i h^{z+1}$$

we have

$$\begin{aligned}\boldsymbol{\Delta}_n &= \left(\boldsymbol{\beta}_n + \widehat{\boldsymbol{\alpha}}_{n-1}\boldsymbol{\beta}_{n-1} + \ldots + \widehat{\boldsymbol{\alpha}}_{n-1}\widehat{\boldsymbol{\alpha}}_{n-2}\cdots\widehat{\boldsymbol{\alpha}}_2\widehat{\boldsymbol{\alpha}}_1\boldsymbol{\beta}_1\right)h^{z+1} \\ &\quad + \widehat{\boldsymbol{\alpha}}_{n-1}\widehat{\boldsymbol{\alpha}}_{n-2}\cdots\widehat{\boldsymbol{\alpha}}_2\widehat{\boldsymbol{\alpha}}_1\widehat{\boldsymbol{\alpha}}_0\boldsymbol{\Delta}_0 \\ &= \sum_{i=1}^{n}\boldsymbol{\theta}_i h^{z+1} + \boldsymbol{\theta}_0\boldsymbol{\Delta}_0\end{aligned}$$

where the $\boldsymbol{\theta}_i$ are appropriate coefficients, comprised of the $\widehat{\boldsymbol{\alpha}}$'s and the $\boldsymbol{\beta}$'s. Hence, the global error $\boldsymbol{\Delta}_n$ is, in a sense, a linear accumulation of these local errors. Furthermore, we have

$$\begin{aligned}\boldsymbol{\Delta}_n &= \sum_{i=1}^{n}\boldsymbol{\theta}_i h^{z+1} + \boldsymbol{\theta}_0\boldsymbol{\Delta}_0 \\ &= \left(\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{\theta}_i\right)(nh)\,h^z + \boldsymbol{\theta}_0\boldsymbol{\Delta}_0 \\ &= \overline{\boldsymbol{\theta}}\,(L - x_0)\,h^z + \boldsymbol{\theta}_0\boldsymbol{\Delta}_0 \qquad (8.2)\end{aligned}$$

where $\overline{\boldsymbol{\theta}}$ is the vector of the average values of the $\boldsymbol{\theta}_i$ on $[x_0, x_n]$, and we have used $nh = L - x_0$. We see that $\boldsymbol{\Delta}_n$ is $O(h^z)$ if $\boldsymbol{\Delta}_0$ is small.

We included the initial error $\boldsymbol{\Delta}_0$ simply to indicate the fact that, if the $\widehat{\boldsymbol{\alpha}}$'s have large magnitude, then this initial error could make a substantial contribution to the global error. In other words, the contribution of this initial error does not necessarily tend to zero as we iterate the method. This initial error could represent roundoff error in the initial-value, although a proper treatment of roundoff requires that we replace each local error $\boldsymbol{\varepsilon}_i$ by $\boldsymbol{\varepsilon}_i + \boldsymbol{\mu}_i$ in the foregoing analysis, where $\boldsymbol{\mu}_i$ is the roundoff error incurred at x_i.

Chapter 9

Error control

The importance of being able to control the approximation error in the implementation of any numerical method cannot be overstated. In this section we will study several techniques for estimating and controlling both local and global errors in RK methods.

9.1 Error estimation: Local extrapolation

Consider two RK methods of order z and $z+1$. Let w_{i+1}^z denote the approximate solution at x_{i+1} obtained with the order z method, and similarly for w_{i+1}^{z+1}. Let the local error at x_{i+1} in the order z method be denoted by $\varepsilon_{i+1}^z = \beta_{i+1}^z h^{z+1}$, and similarly for $\varepsilon_{i+1}^{z+1} = \beta_{i+1}^{z+1} h^{z+2}$. Hence, with $w_i^z, w_i^{z+1} = y_i$, we have

$$\begin{aligned} w_{i+1}^z - w_{i+1}^{z+1} = \varepsilon_{i+1}^z - \varepsilon_{i+1}^{z+1} &= \beta_{i+1}^z h^{z+1} - \beta_{i+1}^{z+1} h^{z+2} \\ &\approx \beta_{i+1}^z h^{z+1} \end{aligned}$$

if h is sufficiently small. This gives

$$\beta_{i+1}^z \approx \frac{w_{i+1}^z - w_{i+1}^{z+1}}{h^{z+1}}. \tag{9.1}$$

9.2 Error estimation: Richardson extrapolation

Here, we use the order z method only. We find approximate solutions at $x_i + \frac{h}{2}$ and x_{i+1}, so that

$$\begin{aligned} w_{i+1}^z(h) - w_{i+1}^z\left(\frac{h}{2}\right) &\approx \beta_{i+1}^z h^{z+1} - 2\beta_{i+1}^z \left(\frac{h}{2}\right)^{z+1} \\ &= \beta_{i+1}^z \left(1 - 2^{-z}\right) h^{z+1} \end{aligned} \tag{9.2}$$

which gives

$$\beta_{i+1}^z \approx \frac{w_{i+1}^z(h) - w_{i+1}^z\left(\frac{h}{2}\right)}{\left(1 - 2^{-z}\right) h^{z+1}}. \tag{9.3}$$

A justification for the factor of $2\beta_{i+1}^z$ in (9.2) is given in the Supplementary Notes.

9.3 Local error control

Once we have estimated the local error, we can perform error control. Assume that we require that the local error at each step must be less than a user-defined tolerance δ. Moreover, assume that, using stepsize h, we find

$$\left|\varepsilon_{i+1}^z\right| = \left|\beta_{i+1}^z h^{z+1}\right| \geqslant \delta.$$

In other words, the magnitude of the local error ε_{i+1}^z exceeds the desired tolerance. We remedy the situation by determining a new stepsize h^* from

$$\left|\beta_{i+1}^z (h^*)^{z+1}\right| = \delta \Rightarrow h^* = \left(\frac{\delta}{\left|\beta_{i+1}^z\right|}\right)^{\frac{1}{z+1}} \tag{9.4}$$

and we repeat the RK computation with this new stepsize. This, of course, gives

$$x_{i+1} = x_i + h^*.$$

This procedure is then carried out on the next step, and so on. Such form of error control is known as *absolute* error control. If the estimated error does not exceed the tolerance, then no stepsize adjustment is necessary, and we proceed to the next step.

Often, we introduce a so-called 'safety factor' σ, as in

$$h^* = \sigma \left(\frac{\delta}{\left|\beta_{i+1}^z\right|}\right)^{\frac{1}{z+1}}$$

where $\sigma < 1$, so that the new stepsize is slightly smaller than that given by (9.4). This is an attempt to cater for the possibility that β_{i+1}^z may have been underestimated, due to the assumptions made in deriving (9.1) and (9.3). The choice of the value of σ is subjective, although a representative value is 0.8.

Additionally, we generally implement *relative* local error control, in the sense that we demand that

$$\frac{\left|\varepsilon_{i+1}^z\right|}{\left|w_{i+1}^z\right|} \leqslant \delta_R \Rightarrow \left|\varepsilon_{i+1}^z\right| \leqslant \delta_R \left|w_{i+1}^z\right|.$$

However, if $\left|w_{i+1}^z\right| \simeq 0$ we implement absolute error control, by demanding that

$$\left|\varepsilon_{i+1}^z\right| \leqslant \delta_A.$$

To cater for both of these cases, we modify (9.4) to give

$$h^* = \left(\frac{\max\left\{\delta_A, \delta_R \left|w_{i+1}^z\right|\right\}}{\left|\beta_{i+1}^z\right|}\right)^{\frac{1}{z+1}}.$$

Thus, if $\left|w_{i+1}^z\right| \simeq 0$ the error is most likely subjected to the tolerance δ_A, otherwise the tolerance is most likely $\delta_R \left|w_{i+1}^z\right|$. Usually, we use $\delta_A = \delta_R$.

Note that because this error control algorithm is applied on each step, we could find that over the interval of integration we have stepsizes of varying lengths. For this reason, it is appropriate to make the replacement

$$h \longrightarrow h_i$$

where $h_i \equiv x_{i+1} - x_i$ in the definition of RK methods (3.1).

9.3.1 Propagation of the higher-order solution

There is a very important point that must be discussed. Our methods for determining β_{i+1}^z hinged on the requirement $w_i^z, w_i^{z+1} = y_i$. However, we only have the exact solution at the initial point x_0; at all subsequent nodes, the solution is approximate. How do we meet the requirement $w_i^z, w_i^{z+1} = y_i$?

In the case of local extrapolation, the answer is simple: simply use the higher-order solution w_i^{z+1} as input to generate both w_{i+1}^z (with the order z method), and w_{i+1}^{z+1} (with the order $z+1$ method). In other words, we are assuming that w_i^{z+1} is accurate enough, relative to w_i^z, to be regarded as the exact value,

an assumption entirely consistent with the assumption made in deriving (9.1). This means that we determine the higher-order solution at each node, and this solution is used as input for both methods in computing solutions at the next node. The question of whether or not the global error that accumulates in the higher-order solution affects the calculation of β_{i+1}^{z} in (9.1) is addressed in the Supplementary Notes.

For Richardson extrapolation, the requirement $w_i^z, w_i^{z+1} = y_i$ is not actually necessary, also shown in the Supplementary Notes.

9.4 Reintegration

Controlling the global error requires a *reintegration* process. This means that we estimate the maximum global error in the numerical solution on the entire interval of integration (obtained using an order z method with stepsize h), impose the condition

$$G_A \left(h^*\right)^z \leqslant \delta$$

where G_A is the coefficient of the maximum absolute global error, determine a new stepsize h^*, and then repeat the entire calculation using this new stepsize (hence the term 'reintegration'). We assume, for simplicity, that the nodes are equispaced on the interval of integration.

Estimation of the global error is most easily achieved by obtaining a higher-order ($z+1$, for example) solution, using the stepsize h, in addition to the order z solution. Hence,

$$G_A = \frac{\max\limits_i \left|w_i^z - w_i^{z+1}\right|}{h^z}.$$

We have assumed that the global error in the higher-order solution can be neglected, relative to the lower-order solution, in similar fashion to what was done in the local extrapolation procedure described earlier. For relative global error control we have

$$G_R = \frac{\max\limits_i \left|\frac{w_i^z - w_i^{z+1}}{\max\left\{\delta_A, \delta_R \left|w_i^z\right|\right\}}\right|}{h^z}$$

$$h^* = \left(\frac{1}{G_R}\right)^{\frac{1}{z}}.$$

Here, G_R is the coefficient of the maximum relative global error, taking into account the possibility $|w_i^z| \simeq 0$.

Because of the need to obtain three numerical solutions, the reintegration procedure is generally regarded as inefficient, and local error control is usually preferred over global error control.

9.4.1 The possibility of bounded global error via local error control

Of course, controlling the global error may desirable in many applications, and if efficiency concerns are not important, then reintegration must be used. However, it is worth investigating the effect on the global error if local error control, via local extrapolation, is implemented.

Consider the expression obtained previously for the global error at x_n

$$\begin{aligned}\Delta_n &= \varepsilon_n + \widehat{\alpha}_{n-1}\varepsilon_{n-1} + \ldots + \widehat{\alpha}_{n-1}\widehat{\alpha}_{n-2}\cdots\widehat{\alpha}_2\widehat{\alpha}_1\varepsilon_1 \\ &= \varepsilon_n + \widehat{\alpha}_{n-1}\Delta_{n-1}. \end{aligned} \tag{9.5}$$

We consider the scalar case, and we assume $\Delta_0 = 0$. If we have the exact value y_i at each node, then we have

$$\Delta_{i+1} = \varepsilon_{i+1}$$

at each node, so that the global error is equal to the local error. If the local error has been controlled (subject to tolerance δ), we have

$$|\Delta_{i+1}| \leqslant \delta$$

which means that the global error satisfies the tolerance δ. However, as discussed previously, we do not have y_i at each node; rather, in the case of local extrapolation, where we have a higher-order solution available, and we propagate this

higher-order solution, we have, from (9.5),

$$\begin{aligned}
\Delta_{i+1}^{z} &= \varepsilon_{i+1}^{z} + \widehat{\alpha}_{i}^{z}\Delta_{i}^{z+1} \\
&= \varepsilon_{i+1}^{z} + \left(1 + hF_{y}^{z}\left(\eta_{i}\right)\right)\left(\varepsilon_{i}^{z+1} + \widehat{\alpha}_{i-1}^{z+1}\varepsilon_{i-1}^{z+1} + \ldots + \widehat{\alpha}_{i-1}^{z+1}\widehat{\alpha}_{i-2}^{z+1}\cdots\widehat{\alpha}_{1}^{z+1}\varepsilon_{1}^{z+1}\right) \\
&= \varepsilon_{i+1}^{z} + \sum_{j=1}^{i}\varepsilon_{j}^{z+1} + O\left(h^{z+3}\right) \\
&\lesssim \left|\varepsilon_{i+1}^{z}\right| + \sum_{j=1}^{i}\sigma^{z+2}\left|\beta_{j}^{z+1}\right|h^{z+2} \\
&= \sigma^{z+1}\left|\beta_{i+1}^{z}\right|h^{z+1} + \sigma^{z+2}\left|\bar{\beta}^{z+1}\left(x_{i} - x_{0}\right)\right|h^{z+1}
\end{aligned}$$

where η_i is an appropriate constant, h is assumed to be the stepsize determined from (9.4) and we have included the safety factor σ explicitly. In the second last line, we assume that, since $\left|\varepsilon_{i+1}^{z}\right| \leqslant \delta$ and $\left|\varepsilon_{j}^{z+1}\right| \ll \left|\varepsilon_{j}^{z}\right|$ (the fundamental assumption in local extrapolation), we must have$\left|\varepsilon_{j}^{z+1}\right| \leqslant \delta$. In the last line, $\bar{\beta}^{z+1}$ denotes an average value, and we have used the same analysis as in deriving (8.2). Now, assuming $\left|\bar{\beta}^{z+1}\left(x_{i} - x_{0}\right)\right|h^{z+1} < \delta$ (see the Supplementary Notes), we have

$$\begin{aligned}
\left|\Delta_{i+1}^{z}\right| &\approx \sigma^{z+1}\delta + \sigma^{z+2}\left|\bar{\beta}^{z+1}\left(x_{i} - x_{0}\right)\right|h^{z+1} \\
&\leqslant \sigma^{z+1}\delta + \sigma^{z+2}\delta \\
&= \left(\sigma^{z+1} + \sigma^{z+2}\right)\delta.
\end{aligned}$$

It is easily confirmed that for $\sigma = 0.8$, we have$(\sigma^{z+1} + \sigma^{z+2}) < 1$ for $z \geqslant 2$. This means that

$$\left|\Delta_{i+1}^{z}\right| < \delta$$

for these values of σ and z, which suggests that the global error, like the local error, satisfies the user-defined tolerance. In other words, propagation of the higher-order solution in local error control via local extrapolation, has resulted in control of the global error, although the significance of the safety factor in deriving this result should be clear. More importantly, the above result holds only if the various assumptions made here are true; if they are not, then $\left|\Delta_{i+1}^{z}\right|$ is probably greater than δ. For this reason, we say that the global error is *possibly* bounded, but this is *not guaranteed*. We should appreciate that such a bounding of the global error is a beneficial *by-product* of local error control, and is not the designated objective.

9.5 Error control in systems

For a system, error control differs only in the calculation of the stepsize. If there are d components in the numerical solution, then an error coefficient can be determined for each component, as in

$$\beta^{z,}_{i+1,j} \approx \frac{w^{z}_{i+1,j} - w^{z+1}_{i+1,j}}{h^z}$$

where $j = 1, 2, \ldots, d$ is the component index. We then determine h^* from

$$h^* = \min_j \left[\left(\frac{\max\left\{\delta_A, \delta_R \left| w^{z}_{i+1,j} \right|\right\}}{\left| \beta^{z}_{i+1,j} \right|} \right)^{\frac{1}{z+1}} \right].$$

Effectively, we determine a new stepsize for each component of the system, and choose the smallest. The procedure is similar for global error control via reintegration.

Chapter 10

Stiff differential equations

We now study the use the use of RK methods in solving stiff differential equations. The concept of stability, briefly mentioned in a previous section, is particularly important in this context.

10.1 Definition of stiff differential equations

Consider the d-dimensional system

$$\mathbf{y}' = \mathbf{f}(x, \mathbf{y}) \qquad \mathbf{y}(x_0) = \mathbf{y}_0 \qquad x_0 \leqslant x \leqslant L. \tag{10.1}$$

If the Jacobian

$$\widehat{\mathbf{f}}_y = \begin{bmatrix} \frac{\partial f_1}{\partial y_1} & \cdots & \frac{\partial f_1}{\partial y_d} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_d}{\partial y_1} & \cdots & \frac{\partial f_d}{\partial y_d} \end{bmatrix} \tag{10.2}$$

has at least one eigenvalue in the left half of the complex plane, then the system is said to be *stiff*.

It must be appreciated that $\widehat{\mathbf{f}}_y = \widehat{\mathbf{f}}_y(x, \mathbf{y}(x))$, so that the degree of stiffness of a system can vary with x.

10.2 First-order approximation of a system

Let $\boldsymbol{\pi}(x)$ be a solution of (10.1). Define

$$\mathbf{Y} \equiv \mathbf{y} - \boldsymbol{\pi}$$

and consider

$$\begin{aligned}\mathbf{y}' = \mathbf{f}(x, \mathbf{y}) = \mathbf{f}(x, \boldsymbol{\pi} + \mathbf{Y}) &= \mathbf{f}(x, \boldsymbol{\pi}) + \mathbf{Y}\widehat{\mathbf{f}}_y(x, \boldsymbol{\pi}) + \dots \\ &= \boldsymbol{\pi}' + \mathbf{Y}\widehat{\mathbf{f}}_y(x, \boldsymbol{\pi}) + \dots \\ \Rightarrow \mathbf{y}' - \boldsymbol{\pi}' = \mathbf{Y}' &= \mathbf{Y}\widehat{\mathbf{f}}_y(x, \boldsymbol{\pi}) + \dots\end{aligned}$$

Also, $\boldsymbol{\pi}$ is not a function of $\mathbf{y}$, so that

$$\widehat{\mathbf{f}}_y(x, \boldsymbol{\pi}) = \widehat{\mathbf{f}}_Y(x, \boldsymbol{\pi})$$

$$\left(\frac{df}{dy} = \frac{df}{dY}\frac{dY}{dy} = \frac{df}{dY}\right)$$

and we have

$$\mathbf{Y}' = \widehat{\mathbf{f}}_Y \mathbf{Y} \tag{10.3}$$

as a first-order approximation to (10.1).

The solution to (10.3) in scalar form is

$$\begin{aligned}Y(x) = Ke^{\lambda x} \Rightarrow y(x) &= Ke^{\lambda x} + \pi(x) \\ \lambda &= f_Y(x, \pi) \\ K &= e^{-\lambda x_0}(y_0 - \pi_0).\end{aligned}$$

Note that the 'stiffness constant' λ is a constant in the sense that it is independent of y. We see that if $\lambda < 0$ then the term $Ke^{\lambda x}$ is a decreasing function of x, and if the magnitude of λ is large, then this term decreases rapidly towards zero.

10.3 The Dahlquist equation

The Dahlquist problem is

$$\begin{aligned}y' &= \lambda y \\ y(0) &= 1,\ \lambda < 0\end{aligned}$$

and has the solution

$$y(x) = e^{\lambda x}.$$

Clearly, this equation has the same form as (10.3), so that we may suppose that any IVP 'subsumes' the Dahlquist equation. The significance of this is that, if we wish to study the application of an RK method to a stiff problem, then it is sufficient to simply consider the application of that RK method to the Dahlquist equation. This greatly simplifies the analysis of RK methods applied to stiff systems.

10.4 Instability, stability and stability regions

It is easy to verify that applying Euler's method to the Dahlquist problem gives

$$w_i = (1 + h\lambda)^i .$$

If $|1 + h\lambda| > 1$, then $|w_i| \to \infty$ as $i \to \infty$. However, since $\lambda < 0$, we expect that $|w_i|$ should approach zero under iteration. Certainly, if $|w_i| \to \infty$, we have an unstable solution. Hence, we must impose the condition

$$|1 + h\lambda| < 1 \Rightarrow h < \frac{2}{|\lambda|}. \tag{10.4}$$

If the stepsize satisfies this condition, the solution will be stable and will exhibit the expected decreasing behaviour; if h violates this condition the numerical solution will increase without bound, quite the opposite to what is expected. In other words, we see that the stability of the method depends critically on the value of h.

Generally, for an explicit RK method applied to the Dahlquist problem, we have

$$w_i = R\,(h\lambda)^i$$

where R is known as the stability function, and is a polynomial in $h\lambda$. The stability functions for explicit RK methods of order one to four are

$$R\,(h\lambda) = \begin{cases} 1 + h\lambda & \text{1st order} \\ 1 + h\lambda + \frac{(h\lambda)^2}{2} & \text{2nd order} \\ 1 + h\lambda + \frac{(h\lambda)^2}{2} + \frac{(h\lambda)^3}{6} & \text{3rd order} \\ 1 + h\lambda + \frac{(h\lambda)^2}{2} + \frac{(h\lambda)^3}{6} + \frac{(h\lambda)^4}{24} & \text{4th order} \end{cases} \tag{10.5}$$

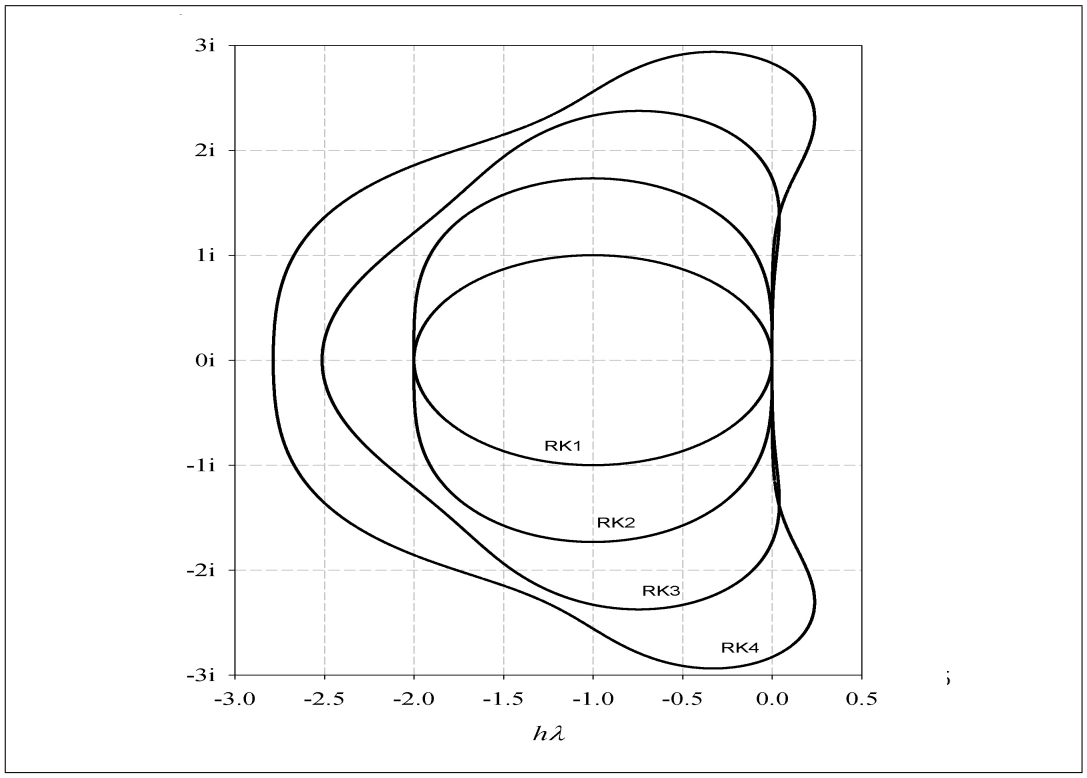
and for stability, we require

$$|R\,(h\lambda)| < 1.$$

This condition defines a region in the complex plane, the boundary of which is the locus of the points that satisfy

$$R\,(h\lambda) = e^{\mathbf{i}\phi}.$$

This equation can be solved for $h\lambda$; each of these roots is parameterized by ϕ. For $\phi \in [0, 2\pi]$, each root races out a segment(s) of the boundary of the stability region, and the union of these segments gives the complete boundary of the stability region. Stability regions corresponding to the stability functions in (10.5) are shown in the figure below.



For stability, we must choose h such that $h\lambda$ lies within the appropriate stability region, for any and all λ that lie in the left half of the complex plane. Recall that λ represents the eigenvalues of the Jacobian (10.2) of the system (10.1), and can certainly be complex. If none of the eigenvalues lie in the left half of the complex plane, then the system is not stiff, or *nonstiff*, and the analysis presented here does not apply. Indeed, for λ in the right half of the complex plane, we expect that the solution will increase exponentially with x.

10.4.1 Interpretation of $R(h\lambda)$

We have

$$w_i = [R(h\lambda)]^i$$
$$y_i = e^{\lambda x_i} = e^{\lambda i h} = \left(e^{\lambda h}\right)^i$$

so that the stability function is an approximation to the exponential function. Indeed, we see in (10.5) that each stability function is a truncated Taylor series for the exponential function $e^{\lambda h}$.

10.4.2 Unstable term in the error expression

Let us derive an error expression consistent with (6.2), one that explicitly shows the unbounded growth of error in an unstable solution. For Euler's method we have

$$\begin{aligned} R(h\lambda) &= 1 + h\lambda \approx e^{\lambda h} \\ \Rightarrow R(h\lambda) - E &= e^{\lambda h} \end{aligned}$$

where E is an error term. For the global error after i iterations

$$\Delta_i = R^i - e^{ih\lambda} = \left(e^{\lambda h} + E\right)^i - e^{ih\lambda}.$$

Now,

$$\begin{aligned} \left(e^{\lambda h} + E\right)^i - e^{ih\lambda} &= e^{ih\lambda}\left(1 + e^{-\lambda h}E\right)^i - e^{ih\lambda} \\ &\approx e^{ih\lambda}\left(e^{-\lambda h}E\right)^i - e^{ih\lambda} \\ &= e^{ih\lambda}\left(\left(e^{-\lambda h}E\right)^i - 1\right) \\ &\approx E^i \end{aligned}$$

where we have assumed that, because $\lambda < 0$, $-\lambda h$ is large positive, so that $e^{-\lambda h}E \gg 1$. For the error E we have

$$\begin{aligned} E &= 1 + h\lambda - e^{\lambda h} \\ &= -\frac{(h\lambda)^2}{2} - \frac{(h\lambda)^3}{6} - \frac{(h\lambda)^4}{24} - \cdots \end{aligned}$$

and so

$$\Delta_i = \left(-\frac{(h\lambda)^2}{2} - \frac{(h\lambda)^3}{6} - \frac{(h\lambda)^4}{24} - \cdots\right)^i.$$

Hence, the global error is polynomial in h and exponential in i, as described in (6.2). Thus, if h is too large, the error will grow without bound.

10.5 Stepsize selection

How do we find a stepsize that guarantees stability? We will describe the process for the explicit fourth-order method; it is similar for other explicit methods.

To begin with, consider

$$R_4(h\lambda) \equiv 1 + h\lambda + \frac{(h\lambda)^2}{2} + \frac{(h\lambda)^3}{6} + \frac{(h\lambda)^4}{24}. \tag{10.6}$$

Now, say we wish to use the fourth-order method with suitable stepsize h to find an approximate solution $\mathbf{w}_1$ to (10.1) at the node $x_1 = x_0 + h$. The procedure for finding such a stepsize is as follows:

1. Find the eigenvalues of the Jacobian (10.2) at $(x_0, \mathbf{w}_0)$, and select only those that lie in the left half of the complex plane. Call this set $\{\lambda_j\}$.

2. For each λ_j, roughly estimate the stepsize as

   $$h_j = \frac{2.8}{|\lambda_j|}.$$

   This amounts to approximating the stability region with a semicircle of radius 2.8.

3. Determine

   $$|R_4(h_j\lambda_j)|.$$

4. For each λ_j for which

   $$|R_4(h_j\lambda_j)| > 1$$

   use a numerical solver, such as the Bisection method (see the Supplementary Notes), to find h_j^* such that

   $$\left|R_4\left(h_j^*\lambda_j\right)\right| = 1.$$

5. From the set $\{h_j\} \cup \left\{h_j^*\right\}$, find $h \equiv \min\left\{\{h_j\} \cup \left\{h_j^*\right\}\right\}$. Use this value for the stepsize to carry out the RK calculation, to generate a stable $\mathbf{w}_1$ at $x_1 = x_0 + h$.

In Step 2, the numerator 2.8 is the extent of the boundary of the stability region for RK4 (see the figure) along the real axis. For our purposes, we will refer to this number as the *stiffness limit* for the given RK method. Stiffness limits for the methods in the figure are given in the table below.

	RK1	RK2	RK3	RK4
Stiffness limit	2	2	2.5	2.8

It is clear that the stepsize selection process described here is both local and predictive. By contrast, stepsize adjustment in local error control is retrodictive. *It is important to ensure that, in the case of local error control via local extrapolation, the higher order method has a larger stability region than the lower order method.* There is no point in using an unstable high order solution to estimate the error in the low order solution; we have already made the point that unstable methods are not convergent, so that unstable solutions are probably very inaccurate - quite the opposite property to what is required in local extrapolation.

10.6 Implicit methods and A-stability

The obvious drawback of using explicit RK methods to solve stiff systems is that, if the system is very stiff (eigenvalues with very large magnitude), then the appropriate stepsizes would be very small. This would mean a very large number of steps/nodes would be required on the interval of integration, with adverse consequences for computational efficiency.

An implicit RK method with tableau

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1m} \\ c_2 & a_{21} & a_{22} & & \vdots \\ \vdots & \vdots & & \ddots & \vdots \\ c_m & a_{m,1} & a_{m,2} & \cdots & a_{m,m} \\ \hline & b_1 & b_2 & \cdots & b_m \end{array}$$

has a stability function given by

$$R(h\lambda) = 1 + h\lambda \mathbf{B}(\widehat{\mathbf{I}}_m - h\lambda\widehat{\mathbf{A}})^{-1}\mathbf{1}_m$$

where

$$\widehat{\mathbf{A}} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & & \vdots \\ \vdots & & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,m} \end{bmatrix} \quad \mathbf{B} = [b_1\ b_2\ \ldots\ b_m] \quad \mathbf{1}_m = \left.\begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}\right\} m \text{ times}$$

In fact, this expression holds for any RK method, including explicit methods.

Consider the method

$$\begin{array}{c|cc} 1 & \frac{1}{2} & \frac{1}{2} \\ 0 & -\frac{1}{2} & \frac{1}{2} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

encountered previously. For this implicit method, we have

$$R(h\lambda) = \frac{1}{\frac{(h\lambda)^2}{2} - h\lambda + 1}.$$

This is an example of a so-called Padé approximation to $e^{h\lambda}$, and it is known to have magnitude less than one on the left half of the complex plane. In other words, this stability function satisfies

$$|R(h\lambda)| < 1$$

for all λ in the left half of the complex plane, irrespective of the value of h. An RK method with such a stability function is said to be *A-stable*. For such methods, no control of the stepsize is necessary, as far as stability with regard to stiff differential equations is concerned. Not all implicit methods are A-stable, but only implicit methods can be A-stable. There are no A-stable explicit methods. If an A-stable method is used to solve a stiff problem, particularly a very stiff problem, it would require fewer nodes, simply because the only restriction on h that could be present would be that relating to local error control. Consequently, such a method could very well be more efficient than an explicit method, even with the extra computational burden of using Netwon's method at each step.

An example of stability in a two-dimensional system is given in the Supplementary Notes.

Chapter 11

Supplementary Notes

11.1 The reflected second-order method

Here we construct the reflected second-order method formally, using the equivalence of the two relevant Taylor series. We have

$$\begin{aligned} y(x_0) = y(x_1 - h) &= y(x_1) - hy'(x_1) + \frac{h^2}{2}y''(x_1) + \dots \\ &= y(x_1) - h[f]_1 + \frac{h^2}{2}[f_x + ff_y]_1 + \dots \end{aligned} \tag{11.1}$$

where $[f]_1$ indicates that f is evaluated at $(x_1, y(x_1))$, and similarly for $[f_x + ff_y]_1$. Here, we have used

$$y' = f \;\Rightarrow\; y'' = f_x + ff_y.$$

Rearranging (11.1) gives

$$y(x_1) = y(x_0) + h[f]_1 - \frac{h^2}{2}[f_x + ff_y]_1 + \dots. \tag{11.2}$$

Assume that forward form of the reflected method has the form

$$\begin{aligned} k_1 &= f(x_1, y(x_1)) \\ k_2 &= f(x_1 + \gamma h, y(x_1) + \theta h k_1) \\ y(x_1) &= y(x_0) + \eta k_1 + \lambda k_2 \end{aligned}$$

where γ, θ, η and λ are parameters to be determined. Hence, we have

$$\begin{aligned} y(x_1) &= y(x_0) + \eta f(x_1, y(x_1)) + \lambda f(x_1 + \gamma h, y(x_1) + \theta h k_1) \\ &= y(x_0) + (\eta + \lambda)[f]_1 + h[\gamma\lambda f_x + \theta\lambda f f_y]_1 + \dots \end{aligned} \tag{11.3}$$

after suitable expansion in a Taylor series.

Equating coefficients in (11.2) and (11.3) gives

$$\begin{aligned}\eta + \lambda &= h \\ \gamma\lambda &= -\frac{h}{2} \\ \theta\lambda &= -\frac{h}{2}.\end{aligned}$$

Choosing $\eta = \frac{h}{2}$ gives

$$\lambda = \frac{h}{2},\ \gamma = -1,\ \theta = -1.$$

11.2 The classic explicit fourth-order method

The structure of the classical explicit fourth-order RK method is

$$\begin{aligned}k_1 &= hf(x, y) \\ k_2 &= hf\left(x + c_2 h, y + c_2 k_1\right) \\ k_3 &= hf\left(x + c_3 h, y + c_3 k_2\right) \\ k_4 &= hf\left(x + c_4 h, y + c_4 k_3\right) \\ y\left(x + h\right) - y(x) &= b_1 k_1 + b_2 k_2 + b_3 k_3 + b_4 k_4.\end{aligned}$$

We seek values of $c_2, c_3, c_4, b_1, b_2, b_3$ and b_4 such that $b_1 k_1 + b_2 k_2 + b_3 k_3 + b_4 k_4$ agrees with the Taylor expansion of $y\left(x + h\right) - y(x)$ up to the fourth-order term.

If we define

$$\begin{aligned}F_1 &\equiv f_x + f f_y \\ F_2 &\equiv f_{xx} + 2f f_{xy} + f^2 f_{yy} \\ F_3 &\equiv f_{xxx} + 3f f_{xxy} + 3f^2 f_{xyy} + f^3 f_{yyy}\end{aligned}$$

where $f_x \equiv \frac{\partial f}{\partial x}, f_{xy} \equiv \frac{\partial^2 f}{\partial x \partial y}$ and so on, then by differentiating $y' = f(x, y)$ we obtain

$$\begin{aligned}y^{(2)} &= F_1 \\ y^{(3)} &= F_2 + f_y F_1 \\ y^{(4)} &= F_3 + f_y F_2 + 3F_1\left(f_{xy} + f f_{yy}\right) + f_y^2 F_1\end{aligned}$$

so that the Taylor expansion is

$$\begin{aligned} y(x+h) - y(x) = {} & hf + \frac{h^2}{2} F_1 + \frac{h^3}{6} \left(F_2 + f_y F_1\right) \\ & + \frac{h^4}{24} \left(F_3 + f_y F_2 + 3\left(f_{xy} + f f_{yy}\right) F_1\right) \\ & + \frac{h^4}{24} f_y^2 F_1 + O(h^5). \end{aligned} \tag{11.4}$$

Expanding the k's in Taylor series yields

$$\begin{aligned} k_1 = {} & hf \\ k_2 = {} & h\left[f + c_2 h F_1 + \frac{1}{2} c_2^2 h^2 F_2 + \frac{1}{6} c_2^3 h^3 F_3 + \ldots\right] \\ k_3 = {} & h\left[f + c_3 h F_1 + \frac{1}{2} h^2 \left(c_3^2 F_2 + 2 c_2 c_3 f_y F_1\right)\right. \\ & \left. + \frac{1}{6} h^3 \left(c_3^3 F_3 + 3 c_2^2 c_3 f_y F_2 + 6 c_2^2 c_3 \left(f_{xy} + f f_{yy}\right) F_1\right) + \ldots\right] \\ k_4 = {} & h\left[f + c_4 h F_1 + \frac{1}{2} h^2 \left(c_4^2 F_2 + 2 c_3 c_4 f_y F_1\right)\right. \\ & + \frac{1}{6} h^3 \left(c_4^3 F_3 + 3 c_3^2 c_4 f_y F_2 + 6 c_3^2 c_4 \left(f_{xy} + f f_{yy}\right) F_1\right) \\ & \left. + h^3 6 c_2 c_3 c_4 f_y^2 F_1 + \ldots\right] \end{aligned}$$

and so

$$\begin{aligned} y(x+h) - y(x) = {} & b_1 k_1 + b_2 k_2 + b_3 k_3 + b_4 k_4 \\ = {} & \left(b_1 + b_2 + b_3 + b_4\right) hf + \left(b_2 c_2 + b_3 c_3 + b_4 p\right) h^2 F_1 \\ & + \frac{1}{2}\left(b_2 c_2^2 + b_3 c_3^2 + b_4 c_4^2\right) h^3 F_2 + \frac{1}{6}\left(b_2 c_2^3 + b_3 c_3^3 + b_4 c_4^3\right) h^4 F_3 \\ & + \left(b_3 c_2 c_3 + b_4 c_3 c_4\right) h^3 f_y F_1 + \frac{1}{2}\left(b_3 c_2 c_3^2 + b_4 c_3 c_4^2\right) h^4 f_y F_2 \\ & + \left(b_3 c_2^2 c_3 + b_4 c_3^2 c_4\right) h^4 \left(f_{xy} + f f_{yy}\right) F_1 \\ & + b_4 c_2 c_3 c_4 h^4 f_y^2 F_1 + O\left(h^5\right). \end{aligned} \tag{11.5}$$

Comparing (11.4) and (11.5) gives the order conditions

$$\begin{aligned} & b_1 + b_2 + b_3 + b_4 = 1 && b_3 c_2 c_3 + b_4 c_3 c_4 = \tfrac{1}{6} \\ & b_2 c_2 + b_3 c_3 + b_4 c_4 = \tfrac{1}{2} && b_3 c_2 c_3^2 + b_4 c_3 c_4^2 = \tfrac{1}{8} \\ & b_2 c_2^2 + b_3 c_3^2 + b_4 c_4^2 = \tfrac{1}{3} && b_3 c_2^2 c_3 + b_4 c_3^2 c_4 = \tfrac{1}{12} \\ & b_2 c_2^3 + b_3 c_3^3 + b_4 c_4^3 = \tfrac{1}{4} && b_4 c_2 c_3 c_4 = \tfrac{1}{24}. \end{aligned}$$

11.3 Order conditions for the explicit fourth-order method

The complete set of order conditions for the explicit fourth-order method is

$$
\begin{aligned}
&b_1 + b_2 + b_3 + b_4 = 1 && b_3 a_{32} c_2 + b_4 a_{42} c_2 + b_4 a_{43} c_3 = \tfrac{1}{6} \\
&b_2 c_2 + b_3 c_3 + b_4 c_4 = \tfrac{1}{2} && b_3 a_{32} c_2 c_3 + b_4 a_{42} c_2 c_4 + b_4 a_{43} c_3 c_4 = \tfrac{1}{8} \\
&b_2 c_2^2 + b_3 c_3^2 + b_4 c_4^2 = \tfrac{1}{3} && b_3 a_{32} c_2^2 + b_4 a_{42} c_2^2 + b_4 a_{43} c_3^2 = \tfrac{1}{12} \\
&b_2 c_2^3 + b_3 c_3^3 + b_4 c_4^3 = \tfrac{1}{4} && b_4 a_{32} a_{43} c_2 = \tfrac{1}{24}
\end{aligned}
$$

$$
b_2 a_{21} + b_3 a_{31} + b_3 a_{32} + b_4 a_{41} + b_4 a_{42} + b_4 a_{43} = \frac{1}{2}
$$

$$
\begin{aligned}
&b_2 a_{21} a_{21} + b_3 a_{31} a_{31} + b_3 a_{31} a_{32} + b_3 a_{32} a_{31} + b_3 a_{32} a_{32} \\
&+ b_4 a_{41} a_{41} + b_4 a_{41} a_{42} + b_4 a_{41} a_{43} + b_4 a_{42} a_{41} + b_4 a_{42} a_{42} \\
&+ b_4 a_{42} a_{43} + b_4 a_{43} a_{41} + b_4 a_{43} a_{42} + b_4 a_{43} a_{43} = \frac{1}{3}
\end{aligned}
$$

$$
b_3 a_{32} a_{21} + b_4 a_{42} a_{21} + b_4 a_{43} a_{31} + b_4 a_{43} a_{32} = \frac{1}{6}
$$

$$
\begin{aligned}
&b_2 a_{21} a_{21} a_{21} + b_3 a_{31} a_{31} a_{31} + b_3 a_{31} a_{31} a_{32} + b_3 a_{31} a_{32} a_{32} \\
&+ b_3 a_{32} a_{32} a_{32} + b_3 a_{32} a_{32} a_{31} + b_3 a_{32} a_{31} a_{31} + b_3 a_{31} a_{32} a_{31} \\
&+ b_3 a_{32} a_{31} a_{32} + b_4 a_{41} a_{41} a_{41} + b_4 a_{41} a_{41} a_{42} + b_4 a_{41} a_{42} a_{41} \\
&+ b_4 a_{41} a_{42} a_{42} + b_4 a_{41} a_{41} a_{43} + b_4 a_{41} a_{43} a_{41} + b_4 a_{41} a_{43} a_{43} \\
&+ b_4 a_{41} a_{42} a_{43} + b_4 a_{41} a_{43} a_{42} + b_4 a_{42} a_{41} a_{41} + b_4 a_{42} a_{41} a_{42} \\
&+ b_4 a_{42} a_{42} a_{41} + b_4 a_{42} a_{42} a_{42} + b_4 a_{42} a_{41} a_{43} + b_4 a_{42} a_{43} a_{41} \\
&+ b_4 a_{42} a_{43} a_{43} + b_4 a_{42} a_{42} a_{43} + b_4 a_{42} a_{43} a_{42} + b_4 a_{43} a_{41} a_{41} \\
&+ b_4 a_{43} a_{41} a_{42} + b_4 a_{43} a_{42} a_{41} + b_4 a_{43} a_{42} a_{42} + b_4 a_{43} a_{41} a_{43} \\
&+ b_4 a_{43} a_{43} a_{41} + b_4 a_{43} a_{43} a_{43} + b_4 a_{43} a_{42} a_{43} + b_4 a_{43} a_{43} a_{42} = \frac{1}{4}
\end{aligned}
$$

$$
\begin{aligned}
&b_3 a_{32} a_{21} a_{31} + b_3 a_{32} a_{21} a_{32} + b_4 a_{42} a_{21} a_{41} \\
&+ b_4 a_{42} a_{21} a_{42} + b_4 a_{42} a_{21} a_{43} + b_4 a_{41} a_{43} a_{31} \\
&+ b_4 a_{42} a_{43} a_{31} + b_4 a_{43} a_{43} a_{31} + b_4 a_{43} a_{32} a_{41} \\
&+ b_4 a_{43} a_{32} a_{42} + b_4 a_{43} a_{32} a_{43} = \frac{1}{8}
\end{aligned}
$$

$$b_3a_{32}a_{21}a_{21} + b_4a_{42}a_{21}a_{21} + b_4a_{43}a_{31}a_{31}$$
$$+ b_4a_{43}a_{31}a_{32} + b_4a_{43}a_{32}a_{31} + b_4a_{43}a_{32}a_{32} = \frac{1}{12}$$

$$b_4a_{43}a_{32}a_{21} = \frac{1}{24}$$

It is easy to show that the conditions

$$c_2 = a_{21}$$
$$c_3 = a_{31} + a_{32}$$
$$c_4 = a_{41} + a_{42} + a_{43}$$

follow from and are consistent with the order conditions above.

11.4 The factor of $2\beta_{i+1}^z$ in Richardson extrapolation

We note that, since an RK method of order z is equivalent to a Taylor method of order z, we will take the local error in the Taylor method as the local error in the RK method. Hence,

$$\begin{aligned}\beta_{i+1}^z h^{z+1} &= \frac{y^{(z)}\left(\xi_{i+1}\right)}{(z+1)!}h^{z+1} \\ &= \frac{y^{(z)}\left(x_i\right)}{(z+1)!}h^{z+1} + \frac{y^{(z+1)}\left(\zeta_i\right)}{(z+2)!}h^{z+2} \\ &\approx \frac{y^{(z)}\left(x_i\right)}{(z+1)!}h^{z+1}\end{aligned}$$

if h is small, and where ξ_{i+1} and ζ_i are appropriate constants.

Now, $w_{i+1}^z\left(h\right)$ is the solution at x_{i+1} obtained with stepsize h. We have

$$\begin{aligned}\Delta_{i+1}\left(h\right) &= \beta_{i+1}^z h^{z+1} + \left(1 + hF_y^z\left(\eta_1\right)\right)\Delta_i \\ &\approx \frac{y^{(z)}\left(x_i\right)}{(z+1)!}h^{z+1} + \Delta_i + hF_y^z\left(\eta_1\right)\Delta_i\end{aligned}$$

where η_1 is an appropriate constant. The solution at $x_i + \frac{h}{2}$, denoted $w_{i+\frac{1}{2}}^z\left(\frac{h}{2}\right)$, has

$$\Delta_{i+\frac{1}{2}}\left(\frac{h}{2}\right) \approx \frac{y^{(z)}\left(x_i\right)}{(z+1)!}\left(\frac{h}{2}\right)^{z+1} + \left(1 + \frac{h}{2}F_y^z\left(\eta_2\right)\right)\Delta_i$$

where η_2 is an appropriate constant. If we use $w^z_{i+\frac{1}{2}}\left(\frac{h}{2}\right)$ to compute a solution at x_{i+1}, which we denote $w^z_{i+1}\left(\frac{h}{2}\right)$, we have

$$\begin{aligned}\Delta_{i+1}\left(\frac{h}{2}\right) \approx \frac{y^{(z)}\left(x_i+\frac{h}{2}\right)}{(z+1)!}\left(\frac{h}{2}\right)^{z+1} &+ \left(1+\frac{h}{2}F^z_y(\eta_3)\right)\frac{y^{(z)}(x_i)}{(z+1)!}\left(\frac{h}{2}\right)^{z+1} \\ &+ \left(1+\frac{h}{2}F^z_y(\eta_3)\right)\left(1+\frac{h}{2}F^z_y(\eta_2)\right)\Delta_i\end{aligned}$$

where η_3 is an appropriate constant. Expanding the first term gives

$$\begin{aligned}\frac{y^{(z)}\left(x_i+\frac{h}{2}\right)}{(z+1)!}\left(\frac{h}{2}\right)^{z+1} &= \frac{y^{(z)}(x_i)}{(z+1)!}\left(\frac{h}{2}\right)^{z+1} + \frac{y^{(z+1)}(x_i)}{(z+1)!}\left(\frac{h}{2}\right)^{z+2} + \dots \\ &\approx \frac{y^{(z)}(x_i)}{(z+1)!}\left(\frac{h}{2}\right)^{z+1}\end{aligned}$$

for small h. Hence,

$$\begin{aligned}\Delta_{i+1}\left(\frac{h}{2}\right) &\approx \frac{y^{(z)}(x_i)}{(z+1)!}\left(\frac{h}{2}\right)^{z+1} + \frac{y^{(z)}(x_i)}{(z+1)!}\left(\frac{h}{2}\right)^{z+1} + \frac{y^{(z)}(x_i)F^z_y(\eta_3)}{(z+1)!}\left(\frac{h}{2}\right)^{z+2} \\ &\quad + \Delta_i + \frac{h}{2}F^z_y(\eta_3)\Delta_i + \frac{h}{2}F^z_y(\eta_2)\Delta_i + \frac{h^2}{4}F^z_y(\eta_3)F^z_y(\eta_2)\Delta_i \\ &\approx 2\beta^z_{i+1}\left(\frac{h}{2}\right)^{z+1} + \Delta_i + \frac{h}{2}F^z_y(\eta_3)\Delta_i + \frac{h}{2}F^z_y(\eta_2)\Delta_i\end{aligned}$$

and so

$$\begin{aligned}\Delta_{i+1}(h) - \Delta_{i+1}\left(\frac{h}{2}\right) &\approx \beta^z_{i+1}h^{z+1} + \Delta_i + hF^z_y(\eta_1)\Delta_i \\ &\quad - 2\beta^z_{i+1}\left(\frac{h}{2}\right)^{z+1} - \Delta_i - \frac{h}{2}F^z_y(\eta_3)\Delta_i - \frac{h}{2}F^z_y(\eta_2)\Delta_i \\ &= \beta^z_{i+1}h^{z+1} - 2\beta^z_{i+1}\left(\frac{h}{2}\right)^{z+1} \\ &\quad + \left(F^z_y(\eta_1) - \frac{F^z_y(\eta_3)}{2} - \frac{F^z_y(\eta_2)}{2}\right)h\Delta_i. \qquad (11.6)\end{aligned}$$

Now, assume

$$\begin{aligned}\eta_2 &= \eta_1 + \rho_2 h \\ \eta_3 &= \eta_1 + \rho_3 h\end{aligned}$$

where ρ_2 and ρ_3 are suitable constants. These give

$$\begin{aligned}F^z_y(\eta_2) &= F^z_y(\eta_1 + \rho_2 h) = F^z_y(\eta_1) + \rho_2 h F^z_{yy}(\eta_1) + \dots \\ F^z_y(\eta_3) &= F^z_y(\eta_1 + \rho_3 h) = F^z_y(\eta_1) + \rho_3 h F^z_{yy}(\eta_1) + \dots\end{aligned}$$

so that

$$F_y^z(\eta_1) - \frac{F_y^z(\eta_3)}{2} - \frac{F_y^z(\eta_2)}{2} = \left(\frac{\rho_2 F_{yy}^z(\eta_1) + \rho_3 F_{yy}^z(\eta_1) + \dots}{2}\right) h$$

which means that the third term in (11.6) is $O(h^{z+2})$ and so is neglected. Hence,

$$\Delta_{i+1}(h) - \Delta_{i+1}\left(\frac{h}{2}\right) \approx \beta_{i+1}^z h^{z+1} - 2\beta_{i+1}^z \left(\frac{h}{2}\right)^{z+1}.$$

Note that in the derivation of this result, we have *not* assumed $w_i^z, w_i^{z+1} = y_i$.

11.5 Effect of higher-order global error in local extrapolation

We assume that w_i^{z+1} is used to generate w_{i+1}^z and w_{i+1}^{z+1}. Let Δ_i^{z+1} denote the global error in w_i^{z+1}. Hence, we have

$$\begin{aligned}
\Delta_{i+1}^z &= \beta_{i+1}^z h^{z+1} + \left(1 + hF_y^z(\eta_1)\right)\Delta_i^{z+1} \\
&= \beta_{i+1}^z h^{z+1} + \Delta_i^{z+1} + hF_y^z(\eta_1)\Delta_i^{z+1} \\
\Delta_{i+1}^{z+1} &= \beta_{i+1}^{z+1} h^{z+2} + \left(1 + hF_y^{z+1}(\eta_2)\right)\Delta_i^{z+1} \\
&= \beta_{i+1}^{z+1} h^{z+2} + \Delta_i^{z+1} + hF_y^{z+1}(\eta_2)\Delta_i^{z+1}
\end{aligned}$$

where η_1 and η_2 are appropriate constants.

Hence,

$$\begin{aligned}
w_{i+1}^z - w_{i+1}^{z+1} &= \beta_{i+1}^z h^{z+1} + \Delta_i^{z+1} + hF_y^z(\eta_1)\Delta_i^{z+1} \\
&\quad - \beta_{i+1}^{z+1} h^{z+2} - \Delta_i^{z+1} - hF_y^{z+1}(\eta_2)\Delta_i^{z+1} \\
&= \beta_{i+1}^z h^{z+1} - \beta_{i+1}^{z+1} h^{z+2} + \left(F_y^z(\eta_1) - F_y^{z+1}(\eta_2)\right) h\Delta_i^{z+1} \\
&\approx \beta_{i+1}^z h^{z+1}
\end{aligned}$$

for small h, because $h\Delta_i^{z+1} = O(h^{z+2})$. We see that the presence of global error in the higher-order solution does not affect the expression for β_{i+1}^z obtained under the assumption $w_i^z, w_i^{z+1} = y_i$. However, if one is not convinced that the $O(h^{z+2})$ term can always be assumed to be negligible in relation to the $O(h^{z+1})$ terms, then one could always use a method of order greater than $z+1$.

11.6 The assumption $\left|\bar{\beta}^{z+1}\left(x_i - x_0\right)\right| h^{z+1} < \delta$

The underlying premise of local extrapolation is the assumption

$$\left|\beta_{i+1}^{z}\right| h^{z+1} \gg \left|\beta_{i+1}^{z+1}\right| h^{z+2}$$

which implies

$$\left|\beta_{i+1}^{z}\right| h^{z+1} = M \left|\beta_{i+1}^{z+1}\right| h^{z+2}, \tag{11.7}$$

where $M \gg 1$ is a large number. Assuming that β_{i+1}^{z+1} is a slowly varying function of x, we can replace β_{i+1}^{z+1} with its average value $\bar{\beta}^{z+1}$, and so we may write

$$\left|\beta_{i+1}^{z}\right| h^{z+1} \geqslant i \left|\bar{\beta}^{z+1}\right| h^{z+2} = \left|\bar{\beta}^{z+1}\left(x_i - x_0\right)\right| h^{z+1},$$

assuming that i is not too large. Hence,

$$\left|\beta_{i+1}^{z}\right| h^{z+1} < \delta \Rightarrow \left|\bar{\beta}^{z+1}\left(x_i - x_0\right)\right| h^{z+1} < \delta.$$

Of course, if i is such that

$$\left|\beta_{i+1}^{z}\right| h^{z+1} < i \left|\bar{\beta}^{z+1}\right| h^{z+2}$$

then our analysis fails, and the global error will not be bounded by δ, even though the local error is bounded.

11.7 The Bisection method for stepsize adjustment in a stiff problem

If

$$\left|R_4\left(h_j \lambda_j\right)\right| > 1$$

then we implement the Bisection method to find the root α_j of

$$\left|R_4\left(\alpha_j h_j \lambda_j\right)\right| - 1 = 0$$

with

$$[\beta, 1]$$

as the starting interval.

To find β, we determine

$$\left| R_4\left(\frac{h_j\lambda_j}{2^n}\right)\right| - 1$$

for successive values of n $(n = 1, 2, \ldots)$, until we find

$$\left| R_4\left(\frac{h_j\lambda_j}{2^n}\right)\right| - 1 < 0.$$

Say this occurs at $n = n_C$. We then take

$$\beta = \frac{1}{2^{n_C}}.$$

Hence,

$$|R_4(\beta h_j\lambda_j)| - 1 < 0$$

and $\alpha_j = 1$ gives

$$|R_4(h_j\lambda_j)| - 1 > 0$$

so that the root certainly lies on $[\beta, 1]$. Once we have found α_j, we define

$$h_j^* = 0.95\alpha_j h_j.$$

where the 0.95 is a safety factor.

11.8 Stability in a stiff two-dimensional system

We investigate stability in the two-dimensional system

$$\mathbf{y}' = \begin{bmatrix} y_1' \\ y_2' \end{bmatrix} = \mathbf{f}(x, \mathbf{y}) = \begin{bmatrix} f_1(x, y_1, y_2) \\ f_2(x, y_1, y_2) \end{bmatrix}$$

with

$$\widehat{\mathbf{f}}_y = \begin{bmatrix} \frac{\partial f_1}{\partial y_1} & \frac{\partial f_1}{\partial y_2} \\ \frac{\partial f_2}{\partial y_1} & \frac{\partial f_2}{\partial y_2} \end{bmatrix} \equiv \begin{bmatrix} A & B \\ C & D \end{bmatrix}.$$

The eigenvalues of $\widehat{\mathbf{f}}_y$ are

$$\lambda_1 = \frac{A+D}{2} + \frac{\sqrt{A^2 - 2AD + D^2 + 4BC}}{2}$$
$$\lambda_2 = \frac{A+D}{2} - \frac{\sqrt{A^2 - 2AD + D^2 + 4BC}}{2}.$$

Now assume that

$$\begin{aligned} A + D &< 0 \\ A^2 - 2AD + D^2 + 4BC &< 0 \end{aligned}$$

so that

$$\begin{aligned} \lambda_1 &= \frac{A+D}{2} + \left(\frac{\sqrt{-\left(A^2 - 2AD + D^2 + 4BC\right)}}{2} \right) \mathbf{i} \\ \lambda_2 &= \frac{A+D}{2} - \left(\frac{\sqrt{-\left(A^2 - 2AD + D^2 + 4BC\right)}}{2} \right) \mathbf{i}. \end{aligned}$$

In this case,

$$|\lambda_1| = |\lambda_2| = \frac{\sqrt{(A+D)^2 - \left(A^2 - 2AD + D^2 + 4BC\right)}}{2} = \sqrt{AD - BC}.$$

Hence, the condition for stability is

$$h < \frac{2}{\sqrt{AD - BC}}$$

if we were to use Euler's method to solve the problem; see (10.4). From the plot of stability regions it seems reasonable to use a numerator of 2.5 if we were going to use RK3 or RK4 to solve the problem, i.e.

$$h < \frac{2.5}{\sqrt{AD - BC}}.$$

Strictly speaking, the numerators used here (our so-called stiffness limits), while reasonable, are not entirely correct. To properly determine the stepsize, we should use the algorithm described earlier which takes into account the complex nature of the eigenvalue(s).

Note that if $\sqrt{AD - BC}$ is small, then the condition on h is not stringent, so that the use of an explicit method would probably be acceptable. Furthermore, the eigenvalues will certainly have small magnitude if the entries in $\widehat{\mathbf{f}}_y$ are small. Since the largest of these entries (in magnitude) serves as a Lipschitz constant for the problem, we infer that stiff systems with small Lipschitz constants will very likely be only slightly or moderately stiff, so that an explicit method is suitable. If, on the other hand, the Lipschitz constant is large, then the system, if it is stiff, is probably very stiff, and an implicit method must be used.