Numerical Solutions of Ordinary Differential Equations (III): Runge-Kutta

Runge-Kutta methods: Introduction

While *multistep methods* achieve higher orders than *one-step methods* by incresing the number of steps, **Runge-Kutta (RK) methods** do this by approximating the Taylor series of the solution up to a higher order while mantaining the structure of one-step methods.

There are *explicit* and *implicit* RK methods, and we are going to start deriving a **two-stage explicit RK method**.

Runge-Kutta methods: Deriving the Methods

In this section, we will derive the **2-stage explicit RKs** (there are infinite). Assuming that the method can be written as:

$$\begin{cases} K_1 = f(t_n, u_n) \\ K_2 = f(t_n + c_2 h, u_n + h a_{21} f(t_n, u_n)) \\ u(t+h) = u(t) + h(b_1 K_1 + b_2 K_2) \end{cases}$$

We need to compute the $2D$ Taylor series of K_1 and K_2 and substitute them in the formula for $u(t+h)$, making it as close as possible to the Taylor series of $y(t+h)$ (in this case the highest order we can achieve is order 2).

For a general function $g(t)$, we have the following $2D$ Taylor Series around (x, y):

$$g(x + h_x, y + h_y) = g(x, y) + h_x \frac{\partial g}{\partial x} + h_y \frac{\partial g}{\partial y} + O(h_x^2 + h_y^2)$$

Therefore, for K_2, we have:

$$K_2 = f(t_n + c_2 h, u_n + h a_{21} f(t_n, u_n)) = f(t_n, u_n) + (c_2 h) \frac{\partial f(t_n, u_n)}{\partial t} + (a_{21} h f(t_n, u_n)) \frac{\partial f(t_n, u_n)}{\partial y} + O(h^2)$$

If we define $f_n = f(t_n, u_n)$, $f_{n,t} = \frac{\partial f(t_n, u_n)}{\partial t}$ and $f_{n,y} = \frac{\partial f(t_n, u_n)}{\partial y}$, this simplifies to:

$$K_2 = f(t_n + c_2 h, u_n + h a_{21} f) = f_n + h(c_2 f_{n,t} + a_{21} f_n f_{n,y}) + O(h^2)$$

and

$$K_1 = f_n$$

Then:

$$u(t+h) = u(t) + h\left[b_1 f_n + b_2 f_n + b_2 h(c_2 f_{n,t} + a_{21} f_n f_{n,y}) + O(h^2)\right] =$$

$$= u(t) + h(b_1 + b_2) f_n + b_2 h^2 (c_2 f_{n,t} + a_{21} f_n f_{n,y}) + O(h^3)$$

But if we compute the Taylor series of $y(t)$, and taking into account that $y'(t) = f$ (by definition of the problem), we obtain:

$$y(t+h) = y(t) + h y'(t) + \frac{h^2}{2} y''(t) + O(h^3) = y(t) + h f_n + \frac{h^2}{2} f'(t) + O(h^3) = y(t) + h f_n + \frac{h^2}{2}(f_{n,t} + f_n f_{n,y}) + O(h^3)$$

As $y(t+h) = u(t+h)$, we have:

$$y(t) + h f_n + \frac{1}{2}(f_{n,t} + f_n f_{n,y}) h^2 + O(h^3) = u(t) + (b_1 + b_2) f_n h + b_2 (c_2 f_{n,t} + a_{21} f_n f_{n,y}) h^2 + O(h^3)$$

From this expression, we find that:

$$\begin{cases} b_1 + b_2 = 1 \\ \frac{1}{2}(f_{n,t} + f_n f_{n,y}) = b_2(c_2 f_{n,t} + a_{21} f_n f_{n,y}) \end{cases} \begin{cases} b_2 c_2 = \frac{1}{2} \\ b_2 a_{21} = \frac{1}{2} \rightarrow c_2 = a_{21} \end{cases}$$

Finally, we can construct an RK two stage method by choosing any combination of b_1, b_2, a_{21} and c_2 that fulfills the following conditions:

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

The most common are:

- **Heun's method**: $c_2 = a_{21} = 1$, $b_1 = b_2 = \frac{1}{2}$.
- **Modified Euler method**: $c_2 = a_{21} = \frac{1}{2}$, $b_1 = 0$ and $b_2 = 1$
- **Ralston method**: $c_2 = a_{21} = \frac{2}{3}$, $b_1 = \frac{1}{4}$ and $b_2 = \frac{3}{4}$.

General Form of Runge Kutta Methods

The general form of RK methods is:

> Any RK method can be written as:
>
> $$u_{n+1} = u_n + hF(t_n, u_n, h; f),\ n \geq 0$$
>
> where F is the increment function defined as:
>
> $$F(t_n, u_n, t_n; f) = \sum_{i=1}^{s} b_i K_i$$
>
> $$K_i = f\left(t_n + c_i h, u_n + h \sum_{j=1}^{s} a_{ij} K_i\right),\ i = 1, 2, 3, \ldots, s$$
>
> where s denotes the number of stages of the method.

The coefficients $\{a_{ij}\}$, $\{c_i\}$ and $\{b_i\}$ fully characterize a RK method and are usually collected in the so-called Butcher array:

> The **Butcher array** is defined as:
>
> $$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array}$$
>
> or:
>
> $$\begin{array}{c|c} \vec{c} & A \\ \hline & \vec{b}^T \end{array}$$

For a general RK method:

> In a general RK method, we have:
>
> $$c_i = \sum_{j=1}^{s} a_{ij} \text{ for } i = 1, 2, 3, \ldots, s$$

Consistency of Runge Kutta Methods

> An RK method is consistent if:
>
> $$\sum_{i=1}^{s} b_i = 1$$

We say that a RK scheme is a method of order p (≥ 1) with respect to h if $\tau(h) = O(h^p)$ as $h \to 0$.

Convergence of Runge Kutta Methods

Since RK methods are *one-step methods*, consistency implies stability and, in turn, convergence. The convergence order of a RK method is equal to the order of the local truncation error. We can also establish the following theorems:

> **Theorem**: The order of an s-stage explicit RK method cannot be greater than s. Also, if an explicit s-stage Runge–Kutta method has order $p \geq 5$, then $s > p$.
> For orders ranging between 1 and 10, the minimum number of stages $s_{\min}$ required to get a method of corresponding order is:
>
> | **Order** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
> |---|---|---|---|---|---|---|---|---|
> | $s_{\min}$ | 1 | 2 | 3 | 4 | 6 | 7 | 9 | 11 |
>
> For implicit RK, the maximum achievable order using s stages is equal to $2s$.

Stepsize Adaptivity in Runge Kutta Methods

When solving a problem with the RK method, depending on the value of the derivative of the solution at a particular point it may be better to use a smaller or larger step size to obtain the required precision without excessive computational effort. Up to now, we have used fixed step size methods. However, as RK methods are one-step methods, it is possible to increase or decrease the stepsize automatically from one step to the next.

In order to do this, we need to have a way of computing the error of the method in real time, so we know how to adapt our step size. As the formulas for the error of the RK method are too complicated, there are two alternative methods of computing the error:

- Using the same RK method, but with two different stepsizes (typically $2h$ and h).
- Using two RK methods of different order, but with the same number s of stages.

In this case, we will use the second method. The idea is to derive two methods that share the same A and $\vec{c}$ and have different $\vec{b}$, one of them with order p and the other with order $p+1$. The *Butcher Array* of these methods will be:

$$\begin{array}{c|c} \vec{c} & A \\ \hline & \overrightarrow{b_1}^T \end{array} \qquad \begin{array}{c|c} \vec{c} & A \\ \hline & \overrightarrow{b_2}^T \end{array} \qquad \Rightarrow \qquad \begin{array}{c|c} \vec{c} & A \\ \hline & \overrightarrow{b_1}^T \\ & \overrightarrow{b_2}^T \end{array}$$

where the method of order p is identified by the coefficients $\vec{c}$, A and $\overrightarrow{b_1}$, while that of order $p+1$ is identified by $\vec{c}$, A and $\overrightarrow{b_2}$.

Taking the difference between the approximate solutions at t_{n+1} produced by the two methods provides an estimate of the local truncation error for the scheme of lower order. On the other hand, since the coefficients K_i coincide, this difference (which does not require any extra functional evaluations) is given by:

$$\tau_n(h) \approx h \sum_{i=1}^{s} (\overrightarrow{b_1} - \overrightarrow{b_2}) K_i$$

Notice that if you calculate the solution with the order p method the you will get order p convergence, while if you use the order $p+1$ you will get order $p+1$ convergence.

The general expression of RK methods with stepsize adaptivity, where u_n is the order p solution and v_n is the order $p+1$ solution, is:

> **Order p Solution**
>
> $$u_{n+1} = u_n + h_n \sum_{i=1}^{s} (b_1)_i K_i, \ n \geq 0$$
>
> $$K_i = f\left(t_n + c_i h, u_n + h_n \sum_{j=1}^{s} a_{ij} K_i \right), \ i = 1, 2, 3, \ldots, s$$
>
> where s denotes the number of stages of the order p method.

> **Order $p+1$ Solution**
>
> $$v_{n+1} = v_n + h_n \sum_{i=1}^{s+1} (b_2)_i K_i, \ n \geq 0$$
>
> $$K_i = f\left(t_n + c_i h, u_n + h_n \sum_{j=1}^{s} a_{ij} K_i \right), \ i = 1, 2, 3, \ldots, s$$
>
> $$K_{s+1} = f\left(t_n + c_{s+1} h_n, u_{n+1} \right)$$
>
> where s denotes the number of stages of the order p method.

This particular method has a property that makes it even more efficient: **FSAL (first-same-as-last)**. It simplifies calculations by reusing the already obtained approximation u_{n+1} to compute K_{s+1}. The procedure that should be followed to obtain the solution is:

1. Calculate K_i for $i = 1, 2, 3, \ldots, s$ to calculate the approximate solution u_{n+1} given by the order p method.
2. Calculate the order $p+1$ solution v_{n+1}, which uses the order p solution.
3. Calculate the approximation of the error: $(e_n)_{\text{current}} = |v_{n+1} - u_{n+1}|$.
4. If the error is smaller than some tolerance E_{tol} we impose, we accept the step and use it. Otherwise, we reject it and try a new stepsize using:

$$(h_n)_{\text{new}} = (h_n)_{\text{current}} \left| \frac{E_{\text{tol}}}{(e_n)_{\text{current}}} \right|^p$$