

## 4 Runge-Kutta methods

The Euler method, as well as the improved and modified Euler methods are all examples on *explicit Runge-Kutta methods* (ERK). Such schemes are given by

$$\begin{aligned}
 k_1 &= f(t_n, y_n), \\
 k_2 &= f(t_n + c_2h, y_n + ha_{21}k_1), \\
 k_3 &= f(t_n + c_3h, y_n + h(a_{31}k_1 + a_{32}k_2)), \\
 &\vdots \\
 k_s &= f(t_n + c_sh, y_n + h \sum_{j=1}^{s-1} a_{sj}k_j), \\
 y_{n+1} &= y_n + h \sum_{i=1}^s b_i k_i,
 \end{aligned} \tag{7}$$

where  $c_i$ ,  $a_{ij}$  and  $b_i$  are coefficients defining the method. We always require  $c_i = \sum_{j=1}^s a_{ij}$ . Here,  $s$  is the number of *stages*, or the number of function evaluations needed for each step. The vectors  $k_i$  are called stage derivatives. The improved Euler method is then a two-stage RK-method, written as

$$\begin{aligned}
 k_1 &= f(t_n, y_n), \\
 k_2 &= f(t_n + h, y_n + hk_1), \\
 y_{n+1} &= y_n + \frac{h}{2}(k_1 + k_2).
 \end{aligned}$$

Also implicit methods, like the trapezoidal rule,

$$y_{n+1} = y_n + \frac{h}{2}(f(t_n, y_n) + f(t_n + h, y_{n+1}))$$

can be written in a similar form,

$$\begin{aligned}
 k_1 &= f(t_n, y_n), \\
 k_2 &= f(t_n + h, y_n + \frac{h}{2}(k_1 + k_2)), \\
 y_{n+1} &= y_n + \frac{h}{2}(k_1 + k_2).
 \end{aligned}$$

But, contrary to what is the case for explicit methods, a nonlinear system of equations has to be solved to find  $k_2$ .

**Definition 4.1.** *An  $s$ -stage Runge-Kutta method is given by*

$$\begin{aligned}
 k_i &= f(t_n + c_ih, y_n + h \sum_{j=1}^s a_{ij}k_j), \quad i = 1, 2, \dots, s, \\
 y_{n+1} &= y_n + h \sum_{i=1}^s b_i k_i.
 \end{aligned}$$

The method is defined by its coefficients, which is given in a Butcher tableau

$$\begin{array}{c|cccc}
 c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\
 c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\
 \vdots & \vdots & & & \vdots \\
 c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\
 \hline
 & b_1 & b_2 & \cdots & b_s
 \end{array}
 , \quad \text{where } c_i = \sum_{j=1}^s a_{ij}, \quad i = 1, \dots, s.$$

The method is explicit if  $a_{ij} = 0$  whenever  $j \geq i$ , otherwise implicit.

**Example 4.2.** The Butcher-tableaux for the methods presented so far are

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

*Euler*                      *improved Euler*                      *modified Euler*                      *trapezoidal rule*

When the method is explicit, the zeros on and above the diagonal is usually ignored. We conclude this section by presenting the maybe most popular among the RK-methods over times, *The 4th order Runge-Kutta method* (Kutta – 1901):

$$\begin{array}{l}
 k_1 = f(t_n, y_n) \\
 k_2 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1) \\
 k_3 = f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2) \\
 k_4 = f(t_n + h, y_n + hk_3) \\
 y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4)
 \end{array}
 \quad \text{or} \quad
 \begin{array}{c|cccc}
 0 & & & & \\
 \frac{1}{2} & \frac{1}{2} & & & \\
 \frac{1}{2} & 0 & \frac{1}{2} & & \\
 1 & 0 & 0 & 1 & \\
 \hline
 & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6}
 \end{array}
 . \quad (8)$$

#### 4.1 Order conditions for Runge-Kutta methods.

The following theorem were proved in Exercise 2, task 2:

**Theorem 4.3.** *Let*

$$y' = f(t, y), \quad y(t_0) = y_0, \quad t_0 \leq t \leq t_{end}$$

*be solved by a one-step method*

$$y_{n+1} = y_n + h\Phi(t_n, y_n; h), \quad (9)$$

*with stepsize  $h = (t_{end} - t_0)/N$  step. If*

1. *the increment function  $\Phi$  is Lipschitz in  $y$ , and*
2. *the local truncation error  $d_{n+1} = \mathcal{O}(h^{p+1})$ ,*

*then the method is of order  $p$ , that is, the global error at  $t_{end}$  satisfies*

$$e_{Nstep} = y(t_{end}) - y_{Nstep} = \mathcal{O}(h^p).$$

A RK method is a one-step method with increment function  $\Phi(t_n, y_n; h) = \sum_{i=1}^s b_i k_i$ . It is possible to show that  $\Phi$  is Lipschitz in  $y$  whenever  $f$  is Lipschitz and  $h \leq h_{max}$ , where  $h_{max}$  is some predefined maximal stepsize. What remains is the order of the local truncation error. To find it, we take the Taylor-expansions of the exact and the numerical solutions and compare. The local truncation error is  $\mathcal{O}(h^{p+1})$  if the two series matches for all terms corresponding to  $h^q$  with  $q \leq p$ . In principle, this is trivial. In practise, it becomes extremely tedious (give it a try). Fortunately, it is possible to express the two series very elegant by the use of *B-series* and *rooted trees*. Here, we present how this is done, but not why it works. A complete description can be found in the note *the B-series tutorial*.

## B-series and rooted trees






We assume that the equation is rewritten in autonomous form



$$y(t)' = f(y(t)), \quad y(t_0) = y_0. \quad (10)$$

The Taylor expansion of the exact solution of (10) is given by

$$y(t_0 + h) = y(t_0) + hy'(t_0) + \frac{h^2}{2}y''(t_0) + \dots + \frac{h^p}{p!}y^{(p)}(t_0) + \dots. \quad (11)$$

From the ODE (10) and repeated use of the chain rule, we get  $y' = f$ ,  $y'' = f_y f$ ,  $y''' = f_{yy} f f + f_y f_y f$ , etc. Each higher derivative of  $y$  is split into several terms, denoted as *elementary differentials*. These can be represented by rooted trees. A node  $\bullet$  represents  $f$ . A branch out from a bullet represent the derivative of  $f$  with respect to  $y$ . As the chain rule apply, this will always means that we multiply by  $y' = f$ , represented by a new node on the end of the branch. We get the following table:

Elementary differentials	corresponding trees
$y' = f$	
$y'' = f_y f$	
$y''' = f_{yy} f f + f_y f_y f$	
$y^{iv} = f_{yyy} f f f + f_{yy} f_y f f + f_y f_{yy} f f$	
$+ f_{yy} f f_y f + f_y f_{yy} f f + f_y f_y f_y f$	

The elementary differentials corresponding to the trees  and  are equal, thus

$$y^{iv} = f_{yyy} f f f + 3f_{yy} f_y f f + f_y f_{yy} f f + f_y f_y f_y f.$$

And we can go on like that. For each tree  $\tau$  with  $p$  nodes we construct a set of total  $p$  new trees with  $p+1$  nodes by adding one new node to an existing node in  $\tau$ . This procedure might produce the same tree several times, and the total number of ways to construct a distinct tree is denoted by  $\alpha(\tau)$ . Let  $T$  be the set of all possible, distinct, rooted trees constructed this way, and let  $\tau \in T$ . A tree with  $p$  nodes corresponds to one of the terms in  $y^{(p)}$ , thus we call this *the order* of the tree and denote it  $|\tau|$ . The elementary differentials corresponding to a tree is denoted  $F(\tau)(y)$ .

**Example 4.4.**

For  $\tau = \begin{array}{c} \bullet \\ \diagup \quad \diagdown \\ \bullet \quad \bullet \end{array}$  we have  $|\tau| = 4$ ,  $F(\tau)(y) = f_y f_{yy} f f$ ,  $\alpha(\tau) = 1$ .

For  $\tau = \begin{array}{c} \bullet \\ \diagup \quad \diagdown \\ \bullet \quad \bullet \\ \diagup \quad \diagdown \\ \bullet \quad \bullet \end{array}$  we have  $|\tau| = 4$ ,  $F(\tau)(y) = f_{yy} f_y f f$ ,  $\alpha(\tau) = 3$ .

Here,  $f$  and its differentials are evaluated in  $y$ .

Putting this together: If  $y(t)$  is the solution of (10), then

$$y^{(p)}(t_n) = \sum_{\substack{\tau \in T \\ |\tau| = p}} \alpha(\tau) F(\tau)(y(t_n)).$$

Insert this into (11), and we can write the exact solution as a B-series:

$$y(t_n + h) = y(t_n) + \sum_{\tau \in T} \frac{h^{|\tau|}}{|\tau|!} \alpha(\tau) F(\tau)(y(t_n)). \quad (12)$$

The numerical solution after one step can also be written as a B-series, but with some different coefficients

$$y_{n+1} = y_n + \sum_{\tau \in T} \frac{h^{|\tau|}}{|\tau|!} \gamma(\tau) \varphi(\tau) \alpha(\tau) F(\tau)(y_n). \quad (13)$$

where  $\gamma(\tau)$  is an integer, and  $\varphi(\tau)$  depends on the method coefficients, given in the Butcher tableau in Definition 4.1. Both can be found quite easily by the following procedure: Take a tree  $\tau$ . Label the root with  $i$ , and all other non-terminal nodes by  $j, k, l, \dots$ . The root correspond to  $b_i$ . A branch between a lower node  $j$  and an upper node  $k$  correspond to  $a_{jk}$ . A terminal node, connected to a node with label  $k$  corresponds to  $c_k$ .  $\phi(\tau)$  is found by multiplying all these coefficients, and then take the sum over all the indices from 1 to  $s$ .

**Example 4.5.**

The tree  $\tau = \begin{array}{c} \bullet \\ \diagup \quad \diagdown \\ \bullet \quad \bullet \\ \diagup \quad \diagdown \\ \bullet \quad \bullet \end{array}$  can be labelled  $\begin{array}{c} \bullet \\ \diagup \quad \diagdown \\ \bullet \quad \bullet \\ \diagup \quad \diagdown \\ \bullet \quad \bullet \\ \diagup \quad \diagdown \\ \bullet \quad \bullet \end{array}$  so that  $\varphi(\tau) = \sum_{i,j,k,l=1}^s b_i a_{ij} a_{jk} c_k^2 a_{il} c_l$ .

A tree  $\tau$  can also be described by its subtrees. Let  $\tau = [\tau_1, \tau_2, \dots, \tau_l]$  be the tree composed by joining the root of the subtrees  $\tau_1, \tau_2, \dots, \tau_l$  to a joint new root. The term  $\gamma(\tau)$  is defined recursively by

- $\gamma(\bullet) = 1$ .
- $\gamma(\tau) = |\tau| \cdot \gamma(\tau_1) \cdots \gamma(\tau_l)$  for  $\tau = [\tau_1, \tau_2, \dots, \tau_l]$ .

**Example 4.6.**

$$\begin{aligned}
 \tau = \bullet &= [\bullet], & \gamma(\tau) &= 2 \cdot 1 = 2 \\
 \tau = \begin{array}{c} \bullet \\ \diagup \quad \diagdown \\ \bullet \end{array} &= [\bullet, \bullet], & \gamma(\tau) &= 3 \cdot 1 \cdot 1 = 3 \\
 \tau = \begin{array}{c} \bullet \\ \diagup \quad \diagdown \\ \bullet \\ \diagup \quad \diagdown \\ \bullet \end{array} &= [\begin{array}{c} \bullet \\ \diagup \quad \diagdown \\ \bullet \end{array}], & \gamma(\tau) &= 4 \cdot 3 = 12 \\
 \tau = \begin{array}{c} \bullet \\ \diagup \quad \diagdown \\ \bullet \\ \diagup \quad \diagdown \\ \bullet \\ \diagup \quad \diagdown \\ \bullet \end{array} &= [\begin{array}{c} \bullet \\ \diagup \quad \diagdown \\ \bullet \end{array}, \bullet], & \gamma(\tau) &= 7 \cdot 12 \cdot 2 = 168
 \end{aligned}$$

By comparing the two series (12) and (13) with  $y(t_n) = y_n$  we can state the following theorem:

**Theorem 4.7.** *A Runge-Kutta method is of order  $p$  if and only if*

$$\varphi(\tau) = \frac{1}{\gamma(\tau)} \quad \forall \tau \in T, \quad |\tau| \leq p.$$

The order conditions up to order 4 are:

$\tau$	$ \tau $	$\varphi(\tau) = 1/\gamma(\tau)$
$\bullet$	1	$\sum b_i = 1$
$\begin{array}{c} \bullet \\ \bullet \end{array}$	2	$\sum b_i c_i = 1/2$
$\begin{array}{c} \bullet \\ \diagup \quad \diagdown \\ \bullet \\ \bullet \end{array}$	3	$\sum b_i c_i^2 = 1/3$
$\begin{array}{c} \bullet \\ \bullet \\ \bullet \end{array}$		$\sum b_i a_{ij} c_j = 1/6$
$\begin{array}{c} \bullet \\ \diagup \quad \diagdown \\ \bullet \\ \bullet \end{array}$	4	$\sum b_i c_i^3 = 1/4$
$\begin{array}{c} \bullet \\ \diagup \quad \diagdown \\ \bullet \\ \bullet \\ \bullet \end{array}$		$\sum b_i c_i a_{ij} c_j = 1/8$
$\begin{array}{c} \bullet \\ \diagup \quad \diagdown \\ \bullet \\ \bullet \\ \bullet \\ \bullet \end{array}$		$\sum b_i a_{ij} c_j^2 = 1/12$
$\begin{array}{c} \bullet \\ \bullet \\ \bullet \\ \bullet \end{array}$		$\sum b_i a_{ij} a_{jk} c_k = 1/24$