3 Runge-Kutta Methods

In contrast to the multistep methods of the previous section, Runge-Kutta methods are single-step methods — however, with multiple *stages* per step. They are motivated by the dependence of the Taylor methods on the specific IVP. These new methods do not require derivatives of the right-hand side function $\boldsymbol{f}$ in the code, and are therefore general-purpose initial value problem solvers. Runge-Kutta methods are among the most popular ODE solvers. They were first studied by Carle Runge and Martin Kutta around 1900. Modern developments are mostly due to John Butcher in the 1960s.

3.1 Second-Order Runge-Kutta Methods

As always we consider the general first-order ODE system

$$\boldsymbol{y}'(t) = \boldsymbol{f}(t, \boldsymbol{y}(t)). \tag{42}$$

Since we want to construct a second-order method, we start with the Taylor expansion

$$\boldsymbol{y}(t+h) = \boldsymbol{y}(t) + h\boldsymbol{y}'(t) + \frac{h^2}{2}\boldsymbol{y}''(t) + \mathcal{O}(h^3).$$

The first derivative can be replaced by the right-hand side of the differential equation (42), and the second derivative is obtained by differentiating (42), i.e.,

$$\begin{aligned} \boldsymbol{y}''(t) &= \boldsymbol{f}_t(t, \boldsymbol{y}) + \boldsymbol{f}_{\boldsymbol{y}}(t, \boldsymbol{y})\boldsymbol{y}'(t) \\ &= \boldsymbol{f}_t(t, \boldsymbol{y}) + \boldsymbol{f}_{\boldsymbol{y}}(t, \boldsymbol{y})\boldsymbol{f}(t, \boldsymbol{y}), \end{aligned}$$

with Jacobian $\boldsymbol{f}_{\boldsymbol{y}}$. We will from now on neglect the dependence of $\boldsymbol{y}$ on t when it appears as an argument to $\boldsymbol{f}$. Therefore, the Taylor expansion becomes

$$\begin{aligned} \boldsymbol{y}(t+h) &= \boldsymbol{y}(t) + h\boldsymbol{f}(t, \boldsymbol{y}) + \frac{h^2}{2}\left[\boldsymbol{f}_t(t, \boldsymbol{y}) + \boldsymbol{f}_{\boldsymbol{y}}(t, \boldsymbol{y})\boldsymbol{f}(t, \boldsymbol{y})\right] + \mathcal{O}(h^3) \\ &= \boldsymbol{y}(t) + \frac{h}{2}\boldsymbol{f}(t, \boldsymbol{y}) + \frac{h}{2}\left[\boldsymbol{f}(t, \boldsymbol{y}) + h\boldsymbol{f}_t(t, \boldsymbol{y}) + h\boldsymbol{f}_{\boldsymbol{y}}(t, \boldsymbol{y})\boldsymbol{f}(t, \boldsymbol{y})\right] + \mathcal{O}(h^3) \end{aligned} \tag{43}$$

Recalling the multivariate Taylor expansion

$$\boldsymbol{f}(t+h, \boldsymbol{y}+\boldsymbol{k}) = \boldsymbol{f}(t, \boldsymbol{y}) + h\boldsymbol{f}_t(t, \boldsymbol{y}) + \boldsymbol{f}_{\boldsymbol{y}}(t, \boldsymbol{y})\boldsymbol{k} + \ldots$$

we see that the expression in brackets in (43) can be interpreted as

$$\boldsymbol{f}(t+h, \boldsymbol{y}+h\boldsymbol{f}(t, \boldsymbol{y})) = \boldsymbol{f}(t, \boldsymbol{y}) + h\boldsymbol{f}_t(t, \boldsymbol{y}) + h\boldsymbol{f}_{\boldsymbol{y}}(t, \boldsymbol{y})\boldsymbol{f}(t, \boldsymbol{y}) + \mathcal{O}(h^2).$$

Therefore, we get

$$\boldsymbol{y}(t+h) = \boldsymbol{y}(t) + \frac{h}{2}\boldsymbol{f}(t, \boldsymbol{y}) + \frac{h}{2}\boldsymbol{f}(t+h, \boldsymbol{y}+h\boldsymbol{f}(t, \boldsymbol{y})) + \mathcal{O}(h^3)$$

or the numerical method

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\left(\frac{1}{2}\boldsymbol{k}_1 + \frac{1}{2}\boldsymbol{k}_2\right), \tag{44}$$

with

$$\begin{aligned}\boldsymbol{k}_1 &= \boldsymbol{f}(t_n, \boldsymbol{y}_n),\\ \boldsymbol{k}_2 &= \boldsymbol{f}(t_n + h, \boldsymbol{y}_n + h\boldsymbol{k}_1).\end{aligned}$$

This is the *classical second-order Runge-Kutta method*. It is also known as *Heun's method* or the *improved Euler method*.

Remark 1. The $\boldsymbol{k}_1$ and $\boldsymbol{k}_2$ are known as *stages* of the Runge-Kutta method. They correspond to different estimates for the *slope* of the solution. Note that $\boldsymbol{y}_n + h\boldsymbol{k}_1$ corresponds to an Euler step with stepsize h starting from $(t_n, \boldsymbol{y}_n)$. Therefore, $\boldsymbol{k}_2$ corresponds to the slope of the solution one would get by taking one Euler step with stepsize h starting from $(t_n, \boldsymbol{y}_n)$. The numerical method (44) now consists of a single step with the average of the slopes $\boldsymbol{k}_1$ and $\boldsymbol{k}_2$.

2. The notation used here differs slightly from that used in the Iserles book. There the stages are defined differently. I find the interpretation in terms of slopes more intuitive.

3. We also saw earlier that the classical second-order Runge-Kutta method can be interpreted as a predictor-corrector method where Euler's method is used as the predictor for the (implicit) trapezoidal rule.

We obtain general explicit second-order Runge-Kutta methods by assuming

$$\boldsymbol{y}(t+h) = \boldsymbol{y}(t) + h\left[b_1\tilde{\boldsymbol{k}}_1 + b_2\tilde{\boldsymbol{k}}_2\right] + \mathcal{O}(h^3) \tag{45}$$

with

$$\begin{aligned}\tilde{\boldsymbol{k}}_1 &= \boldsymbol{f}(t, \boldsymbol{y})\\ \tilde{\boldsymbol{k}}_2 &= \boldsymbol{f}(t + c_2h, \boldsymbol{y} + ha_{21}\tilde{\boldsymbol{k}}_1).\end{aligned}$$

Clearly, this is a generalization of the classical Runge-Kutta method since the choice $b_1 = b_2 = \frac{1}{2}$ and $c_2 = a_{21} = 1$ yields that case.

It is customary to arrange the coefficients a_{ij}, b_i, and c_i in a so-called *Runge-Kutta* or *Butcher tableaux* as follows:

$$\begin{array}{c|c} \boldsymbol{c} & A \\ \hline & \boldsymbol{b}^T. \end{array}$$

Accordingly, the Butcher tableaux for the classical second-order Runge-Kutta method is

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

Explicit Runge-Kutta methods are characterized by a strictly lower triangular matrix A, i.e., $a_{ij} = 0$ if $j \geq i$. Moreover, the coefficients c_i and a_{ij} are connected by the condition

$$c_i = \sum_{j=1}^{\nu} a_{ij}, \qquad i = 1, 2, \ldots, \nu.$$

This says that c_i is the row sum of the i-th row of the matrix A. This condition is required to have a method of order one, i.e., for consistency. We limit our discussion to such methods now.

Thus, for an explicit second-order method we necessarily have $a_{11} = a_{12} = a_{22} = c_1 = 0$. We can now study what other combinations of b_1, b_2, c_2 and a_{21} in (45) give us a second-order method. The bivariate Taylor expansion yields

$$\begin{aligned} \boldsymbol{f}(t + c_2 h, \boldsymbol{y} + h a_{21} \tilde{\boldsymbol{k}}_1) &= \boldsymbol{f}(t, \boldsymbol{y}) + c_2 h \boldsymbol{f}_t(t, \boldsymbol{y}) + h a_{21} \boldsymbol{f}_{\boldsymbol{y}}(t, \boldsymbol{y}) \tilde{\boldsymbol{k}}_1 + \mathcal{O}(h^2) \\ &= \boldsymbol{f}(t, \boldsymbol{y}) + c_2 h \boldsymbol{f}_t(t, \boldsymbol{y}) + h a_{21} \boldsymbol{f}_{\boldsymbol{y}}(t, \boldsymbol{y}) \boldsymbol{f}(t, \boldsymbol{y}) + \mathcal{O}(h^2). \end{aligned}$$

Therefore, the general second-order Runge-Kutta assumption (45) becomes

$$\begin{aligned} \boldsymbol{y}(t + h) &= \boldsymbol{y}(t) + h \left[b_1 \boldsymbol{f}(t, \boldsymbol{y}) + b_2 \left\{ \boldsymbol{f}(t, \boldsymbol{y}) + c_2 h \boldsymbol{f}_t(t, \boldsymbol{y}) + h a_{21} \boldsymbol{f}_{\boldsymbol{y}}(t, \boldsymbol{y}) \boldsymbol{f}(t, \boldsymbol{y}) \right\} \right] + \mathcal{O}(h^3) \\ &= \boldsymbol{y}(t) + (b_1 + b_2) h \boldsymbol{f}(t, \boldsymbol{y}) + b_2 h^2 \left[c_2 \boldsymbol{f}_t(t, \boldsymbol{y}) + a_{21} \boldsymbol{f}_{\boldsymbol{y}}(t, \boldsymbol{y}) \boldsymbol{f}(t, \boldsymbol{y}) \right] + \mathcal{O}(h^3). \end{aligned}$$

In order for this to match the general Taylor expansion (43) we want

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

Thus, we have a system of three nonlinear equations for our four unknowns. One popular solution is the choice $b_1 = 0$, $b_2 = 1$, and $c_2 = a_{21} = \frac{1}{2}$. This leads to the *modified Euler method* (sometimes also referred to as the *midpoint rule*, see the discussion in Section 3.3 below)

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h \boldsymbol{k}_2$$

with

$$\begin{aligned} \boldsymbol{k}_1 &= \boldsymbol{f}(t_n, \boldsymbol{y}_n) \\ \boldsymbol{k}_2 &= \boldsymbol{f}(t_n + \frac{h}{2}, \boldsymbol{y}_n + \frac{h}{2} \boldsymbol{k}_1). \end{aligned}$$

Its Butcher tableaux is of the form

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

Remark The choice $b_1 = 1$, $b_2 = 0$ leads to Euler's method. However, since now $c_2 b_2 \neq \frac{1}{2}$ and $a_{21} b_2 \neq \frac{1}{2}$ this method does not have second-order accuracy.

General explicit Runge-Kutta methods are of the form

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\sum_{j=1}^{\nu} b_j \boldsymbol{k}_j$$

with

$$\begin{aligned}
\boldsymbol{k}_1 &= \boldsymbol{f}(t_n, \boldsymbol{y}_n) \\
\boldsymbol{k}_2 &= \boldsymbol{f}(t_n + c_2 h, \boldsymbol{y}_n + a_{21} h \boldsymbol{k}_1) \\
&\vdots \\
\boldsymbol{k}_\nu &= \boldsymbol{f}(t_n + c_\nu h, \boldsymbol{y}_n + h\sum_{j=1}^{\nu-1} a_{\nu,j} \boldsymbol{k}_j).
\end{aligned}$$

Determination of the coefficients is rather complicated. We now describe (without derivation) the most famous Runge-Kutta method.

3.2 Fourth-Order Runge-Kutta Methods

The classical method is given by

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\left[\frac{\boldsymbol{k}_1}{6} + \frac{\boldsymbol{k}_2}{3} + \frac{\boldsymbol{k}_3}{3} + \frac{\boldsymbol{k}_4}{6}\right] \tag{46}$$

with

$$\begin{aligned}
\boldsymbol{k}_1 &= \boldsymbol{f}(t_n, \boldsymbol{y}_n) \\
\boldsymbol{k}_2 &= \boldsymbol{f}\left(t_n + \frac{h}{2}, \boldsymbol{y}_n + \frac{h}{2}\boldsymbol{k}_1\right) \\
\boldsymbol{k}_3 &= \boldsymbol{f}\left(t_n + \frac{h}{2}, \boldsymbol{y}_n + \frac{h}{2}\boldsymbol{k}_2\right) \\
\boldsymbol{k}_4 &= \boldsymbol{f}\left(t_n + h, \boldsymbol{y}_n + h\boldsymbol{k}_3\right).
\end{aligned}$$

Its Butcher tableaux is of the form

$$\begin{array}{c|cccc}
0 & 0 & 0 & 0 & 0 \\
\frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\
\frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\
1 & 0 & 0 & 1 & 0 \\
\hline
 & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6}.
\end{array}$$

The local truncation error for this method is $\mathcal{O}(h^5)$. It is also important to note that the classical fourth-order Runge-Kutta method requires four evaluations of the function $\boldsymbol{f}$ per time step.

Remark We saw earlier that in each time step of the second-order Runge-Kutta method we need to perform two evaluations of $\boldsymbol{f}$, and for a fourth-order method there are four evaluations. More generally, one can observe the situation described in Table 4.

evaluations of $\boldsymbol{f}$ per time step	2	3	4	5	6	7	8	9	10	11
maximum order achievable	2	3	4	4	5	6	6	7	7	8

Table 4: Efficiency of Runge-Kutta methods.

These data imply that higher-order (> 4) Runge-Kutta methods are relatively inefficient. Precise data for higher-order methods does not seem to be known. However, certain higher-order methods may still be appropriate if we want to construct a Runge-Kutta method which adaptively chooses the step size for the time step in order to keep the local truncation error small (see Section 5).

3.3 Connection to Numerical Integration Rules

We now illustrate the connection of Runge-Kutta methods to numerical integration rules.

As before, we consider the IVP

$$\begin{aligned} \boldsymbol{y}'(t) &= \boldsymbol{f}(t, \boldsymbol{y}(t)) \\ \boldsymbol{y}(t_0) &= \boldsymbol{y}_0 \end{aligned}$$

and integrate both sides of the differential equation from t to $t+h$ to obtain

$$\boldsymbol{y}(t+h) - \boldsymbol{y}(t) = \int_t^{t+h} \boldsymbol{f}(\tau, \boldsymbol{y}(\tau)) d\tau. \tag{47}$$

Therefore, the solution to our IVP can be obtained by solving the integral equation (47). Of course, we can use numerical integration to do this:

1. Using the left endpoint method

$$\int_a^b \boldsymbol{f}(\boldsymbol{x}) d\boldsymbol{x} \approx \underbrace{\frac{b-a}{n}}_{=h} \sum_{i=0}^{n-1} \boldsymbol{f}(\boldsymbol{x}_i)$$

   on a single interval, i.e., with $n = 1$, and $a = t$, $b = t + h$ we get

$$\begin{aligned} \int_t^{t+h} \boldsymbol{f}(\tau, \boldsymbol{y}(\tau)) d\tau &\approx \frac{t+h-t}{1} \boldsymbol{f}(\tau_0, \boldsymbol{y}(\tau_0)) \\ &= h \boldsymbol{f}(t, \boldsymbol{y}(t)) \end{aligned}$$

   since $\tau_0 = t$, the left endpoint of the interval. Thus, as we saw earlier, (47) is equivalent to Euler's method.

2. Using the trapezoidal rule

$$\int_a^b \boldsymbol{f}(\boldsymbol{x}) d\boldsymbol{x} \approx \frac{b-a}{2} \left[\boldsymbol{f}(a) + \boldsymbol{f}(b)\right]$$

   with $a = t$ and $b = t + h$ gives us

$$\int_t^{t+h} \boldsymbol{f}(\tau, \boldsymbol{y}(\tau)) d\tau \approx \frac{h}{2} \left[\boldsymbol{f}(t, \boldsymbol{y}(t)) + \boldsymbol{f}(t+h, \boldsymbol{y}(t+h))\right].$$

The corresponding IVP solver is therefore

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \frac{h}{2}\boldsymbol{f}(t_n, \boldsymbol{y}_n) + \frac{h}{2}\boldsymbol{f}(t_{n+1}, \boldsymbol{y}_{n+1}).$$

Note that this is *not* equal to the classical second-order Runge-Kutta method since we have a $\boldsymbol{y}_{n+1}$ term on the right-hand side. This means that we have an *implicit* method. In order to make the method explicit we can use Euler's method to replace $\boldsymbol{y}_{n+1}$ on the right-hand side by

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\boldsymbol{f}(t_n, \boldsymbol{y}_n).$$

Then we end up with the method

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \frac{h}{2}\boldsymbol{f}(t_n, \boldsymbol{y}_n) + \frac{h}{2}\boldsymbol{f}(t_{n+1}, \boldsymbol{y}_n + h\boldsymbol{f}(t_n, \boldsymbol{y}_n))$$

or

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\left[\frac{1}{2}\boldsymbol{k}_1 + \frac{1}{2}\boldsymbol{k}_2\right]$$

with

$$\begin{aligned} \boldsymbol{k}_1 &= \boldsymbol{f}(t_n, \boldsymbol{y}_n) \\ \boldsymbol{k}_2 &= \boldsymbol{f}(t_n + h, \boldsymbol{y}_n + h\boldsymbol{k}_1), \end{aligned}$$

i.e., the classical second-order Runge-Kutta method.

3. The midpoint integration rule

$$\int_a^b \boldsymbol{f}(\boldsymbol{x})d\boldsymbol{x} \approx (b-a)\boldsymbol{f}((a+b)/2)$$

with $a = t$, $b = t + 2h$ gives us

$$\int_t^{t+2h} \boldsymbol{f}(\tau, \boldsymbol{y}(\tau))d\tau \quad \approx \quad 2h\boldsymbol{f}(t+h, \boldsymbol{y}(t+h)).$$

Thus, we have the *explicit midpoint rule*

$$\boldsymbol{y}_{n+2} = \boldsymbol{y}_n + 2h\boldsymbol{f}(t_{n+1}, \boldsymbol{y}_{n+1}).$$

This is not a Runge-Kutta method. It is an explicit 2-step method. In the context of PDEs this method reappears as the *leapfrog method*. As mentioned above, sometimes the modified Euler method

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\boldsymbol{f}(t_n + \frac{h}{2}, \boldsymbol{y}_n + \frac{h}{2}\boldsymbol{f}(t_n, \boldsymbol{y}_n)).$$

is called the midpoint rule. This can be explained by applying the midpoint integration rule with $a = t$ and $b = t + h$. Then we have

$$\int_t^{t+h} \boldsymbol{f}(\tau, \boldsymbol{y}(\tau))d\tau \quad \approx \quad h\boldsymbol{f}(t + \frac{h}{2}, \boldsymbol{y}(t + \frac{h}{2})).$$

If we represent $\boldsymbol{y}(t + \frac{h}{2})$ by its Euler approximation $\boldsymbol{y}(t) + \frac{h}{2}\boldsymbol{f}(t, \boldsymbol{y})$, then we arrive at the modified Euler method stated above.

4. Simpson's rule yields the fourth-order Runge-Kutta method in case there is no dependence of $\boldsymbol{f}$ on $\boldsymbol{y}$.

5. Gauss quadrature leads to so-called *Gauss-Runge-Kutta* or *Gauss-Legendre* methods. One such method is the implicit midpoint rule

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\boldsymbol{f}(t_n + \frac{h}{2}, \frac{1}{2}(\boldsymbol{y}_n + \boldsymbol{y}_{n+1}))$$

encountered earlier. The Butcher tableaux for this one-stage order two method is given by

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

Note that the general implicit Runge-Kutta method is of the form

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\sum_{j=1}^{\nu} b_j \boldsymbol{k}_j$$

with

$$\boldsymbol{k}_j \quad = \quad \boldsymbol{f}(t_n + c_j h, \boldsymbol{y}_n + h\sum_{i=1}^{j} a_{j,i}\boldsymbol{k}_i)$$

for all values of $j = 1, \dots, \nu$. Thus, the implicit midpoint rule corresponds to

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + h\boldsymbol{k}_1$$

with

$$\boldsymbol{k}_1 = \boldsymbol{f}(t_n + \frac{h}{2}, \boldsymbol{y}_n + \frac{h}{2}\boldsymbol{k}_1)$$

— obviously an implicit method.

6. More general *implicit Runge-Kutta methods* exist. However, their construction is more difficult, and can sometimes be linked to *collocation methods*. Some details are given at the end of Chapter 3 in the Iserles book.