$$h_{\text{new}} \approx \left(\frac{1}{\text{err}_{\text{sc}}}\right)^{1/(p+1)} h. \tag{1.23}$$

- If $\text{err}_{\text{sc}} > 1$, then the performed step will be rejected. The Richardson method is repeated from (x_0, y_0) with a step length $h_{\text{new}} < h$.
  That means, the work that was spent for performing the step with step length h was wasted. One likes to avoid this situation.

□

Remark 1.40. Issues of the practical implementation. In practical simulations, one uses some modifications of (1.23) for stabilizing the algorithm.

- A safety factor $\alpha \in (0, 1)$ is introduced

  $$h_{\text{new}} = \alpha \left(\frac{1}{\text{err}_{\text{sc}}}\right)^{1/(p+1)} h,$$

  often $\alpha \in [0.8, 0.9]$.
- One likes to avoid large oscillations of the sizes of subsequent steps. For this reason, a factor for the maximal increase $\alpha_{\max}$ of the new step size with respect to the current step size and a factor for the maximal decrease $\alpha_{\min} < \alpha_{\max}$ are used. Then, one obtains

  $$h_{\text{new}} = h \min\left\{\alpha_{\max}, \max\left\{\alpha_{\min}, \alpha\left(\frac{1}{\text{err}_{\text{sc}}}\right)^{1/(p+1)}\right\}\right\}.$$

  If a very large step length is proposed

  $$\alpha \left(\frac{1}{\text{err}_{\text{sc}}}\right)^{1/(p+1)} > \alpha_{\max},$$

  then the factor $\alpha_{\max}$ becomes effective and similarly $\alpha_{\min}$ for the case that a very small step length is proposed.
- Usually, one prescribes a minimal step length $h_{\min}$ and a maximal step length $h_{\max}$ and requires for all step lengths that $h_k \in [h_{\min}, h_{\max}]$.
- In the first step, one has to estimate h. Generally, this estimate has to be corrected. In practice, this correction is done very fast by algorithms for automatic step length control. An algorithm for determining a good initial step length can be found in (Hairer *et al.*, 1993, p. 168).

□

1.3.2 Embedded Runge–Kutta Schemes

Remark 1.41. Motivation, embedded Runge–Kutta schemes. Richardson extrapolation is quite expensive in terms of evaluations of the incremental function. It is possible to construct a step length control that needs less evaluations, with so-called embedded Runge–Kutta schemes.

The idea of embedded Runge–Kutta schemes consists in computing numerical approximations of the solution at the next time with two one-step methods with different order. The methods are chosen in such a way that it is possible to use the evaluations of the incremental function for both of them. That means, one has to construct a Runge–Kutta scheme of the form

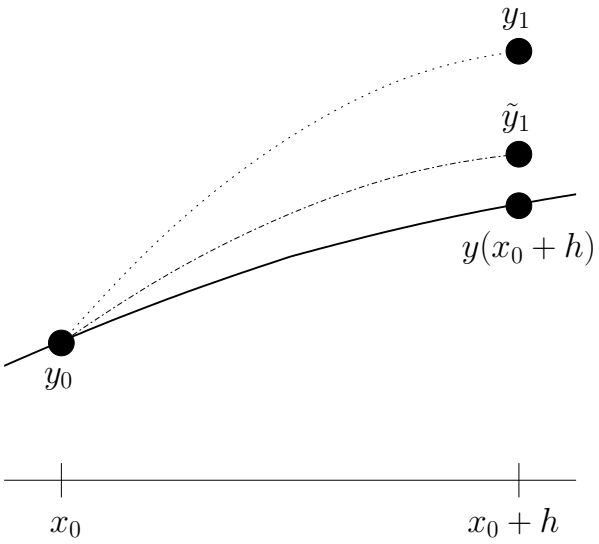

Fig. 1.4 Sketch of embedded Runge–Kutta schemes.

$$\begin{array}{c|ccc} 0 & & & \\ c_2 & a_{21} & & \\ \vdots & & \ddots & \\ c_s & a_{s1} & \cdots & a_{s,s-1} \\ \hline & b_1 & \cdots & b_{s-1} \quad b_s \\ & \tilde{b}_1 & \cdots & \tilde{b}_{s-1} \quad \tilde{b}_s \end{array},$$

which is the short form of

$$\begin{array}{c|ccc} 0 & & & \\ c_2 & a_{21} & & \\ \vdots & & \ddots & \\ c_s & a_{s1} & \cdots & a_{s,s-1} \\ \hline & b_1 & \cdots & b_{s-1} \quad b_s \end{array} \quad \text{and} \quad \begin{array}{c|ccc} 0 & & & \\ c_2 & a_{21} & & \\ \vdots & & \ddots & \\ c_s & a_{s1} & \cdots & a_{s,s-1} \\ \hline & \tilde{b}_1 & \cdots & \tilde{b}_{s-1} \quad \tilde{b}_s \end{array},$$

such that

$$y_1 = y_0 + h \sum_{i=1}^{s} b_i K_i(x, y)$$

is order of p and

$$\tilde{y}_1 = y_0 + h \sum_{i=1}^{s} \tilde{b}_i K_i(x, y)$$

is of order q, see Figure 1.4. In general, it is $q = p - 1$ or $q = p + 1$. □

Example 1.42. Runge–Kutta–Fehlberg 2(3) method. Consider explicit Runge–Kutta schemes with 3 stages

$$\begin{array}{c|cccc} 0 & & & & \\ c_2 & a_{21} & & & \\ c_3 & a_{31} & a_{32} & & \\ \hline & b_1 & b_2 & b_3 & p=2 \\ & \tilde{b}_1 & \tilde{b}_2 & \tilde{b}_3 & q=3 \end{array}.$$

One of the schemes should be of order 2 and the other one of third order. There are 11 parameters to choose. From Theorem 1.26, Theorem 1.27, and Remark 1.28, it follows that 8 equations have to be satisfied

$$\begin{aligned} c_2 &= a_{21}, \\ c_3 &= a_{31} + a_{32}, \\ b_1 + b_2 + b_3 &= 1, \\ b_2 c_2 + b_3 c_3 &= \frac{1}{2}, \\ \tilde{b}_1 + \tilde{b}_2 + \tilde{b}_3 &= 1, \\ \tilde{b}_2 c_2 + \tilde{b}_3 c_3 &= \frac{1}{2}, \\ \tilde{b}_2 c_2^2 + \tilde{b}_3 c_3^2 &= \frac{1}{3}, \\ \tilde{b}_3 a_{32} c_2 &= \frac{1}{6}. \end{aligned}$$

That means, one has to set three parameters. First, one can choose $c_2 = 1$, $b_3 = 0$. Then, it follows from the first equation that $a_{21} = 1$, from the fourth equation that $b_2 = 1/2$, and from the third equation that $b_1 = 1/2$. Now, one chooses $c_3 = 1/2$. From the sixth and seventh equation, it follows that $\tilde{b}_2 = 1/6$ and $\tilde{b}_3 = 4/6$. Then, one gets from the fifth equation $\tilde{b}_1 = 1/6$ and from the eighth equation $a_{32} = 1/4$. Finally, the second equation gives $a_{31} = 1/4$. The resulting methods have the form

$$\begin{array}{c|cccc} 0 & & & & \\ 1 & 1 & & & \\ 1/2 & 1/4 & 1/4 & & \\ \hline & 1/2 & 1/2 & 0 & p=2 \\ & 1/6 & 1/6 & 4/6 & q=3 \end{array}.$$

The method with order $q = 3$ is the Simpson rule. The complete embedded approach is called Runge–Kutta–Fehlberg[7] 2(3) method (RKF 2(3)). □

Remark 1.43. Error estimate, theoretical drawback. By construction, it holds for the embedded scheme that

$$y_1 = y(x_0 + h) + \mathcal{O}\left(h^{p+1}\right), \quad \tilde{y}_1 = y(x_0 + h) + \mathcal{O}\left(h^{q+1}\right).$$

It follows that

$$|\text{err}| := |\tilde{y}_1 - y_1| = \left| y(x_0 + h) + \mathcal{O}\left(h^{q+1}\right) - y(x_0 + h) + \mathcal{O}\left(h^{p+1}\right) \right| = \left| \mathcal{O}\left(h^{p+1}\right) + \mathcal{O}\left(h^{q+1}\right) \right| \tag{1.24}$$

is an estimate of the main error term of the Runge–Kutta scheme of order $q^* = \min\{p, q\}$. That means, one obtains only an estimate of the error of the lower order method. To obtain information only on the lower order method is the main theoretical drawback of this approach, since one is interested actually in the higher order method and one will continue the computation also from the higher order approximation. □

[7] Erwin Fehlberg (1911 – 1972)

Remark 1.44. Automatic step length control, I Controller. Let h be the step size that was used for computing y_1 of order p and $\tilde{y}_1$ of order q with $p < q$. From (1.24), one has

$$|\text{err}| = |y_1 - \tilde{y}_1| = Ch^{p+1}. \tag{1.25}$$

Given a tolerance tol for the maximal local error.

- One approach consists in controlling the error per step (EPS). Then, one requires that

  $$r_1 = |\text{err}| \leq \text{tol}. \tag{1.26}$$

  If this condition is satisfied, then the current step is accepted. Next, one requires for the new step size that the local error is equal to the tolerance

  $$Ch_{\text{new}}^{p+1} = \text{tol},$$

  with C from (1.25) This requirement gives

  $$h_{\text{new}} = \left(\frac{\text{tol}}{C}\right)^{1/(p+1)} = \left(\frac{\text{tol}}{Ch^{p+1}}\right)^{1/(p+1)} h.$$

  With (1.25) and (1.26), the new step length is computed by

  $$h_{\text{new}} = \left(\frac{\alpha\ \text{tol}}{|\text{err}|}\right)^{1/k} h = \left(\frac{\alpha\ \text{tol}}{r_1}\right)^{1/k} h, \tag{1.27}$$

  where $k = p + 1$ and $\alpha \in (0, 1)$ is again a safety factor.
- Another way is the consideration of the error relative to the current step length, the so-called error per unit step (EPUS),

  $$r_1 = \frac{|\text{err}|}{h} \leq \text{tol}. \tag{1.28}$$

  The satisfaction of a condition of form (1.28) leads to a new step of form (1.27) with $k = p$.
- If (1.26) or (1.28) is not satisfied, then the step is rejected and it will be repeated with a step length smaller than h.
- A generalization of this approach is the so-called I Controller. Replacing in (1.27) $1/k$ by k_I gives

  $$h_{\text{new}} = \left(\frac{\alpha\ \text{tol}}{r_1}\right)^{k_I} h.$$

  For obtaining a useful automatic step length control mechanism, the choice $k_I = 1/k$ or equivalently $kk_I = 1$ is not necessary. The following choices can be found in the literature

  $$\begin{aligned} kk_I \in [0, 2] &\Longleftrightarrow k_I \in [0, 2/k] && \text{stable control,} \\ kk_I \in (1, 2) &\Longleftrightarrow k_I \in (1/k, 2/k) && \text{fast and oscillating control,} \\ kk_I \in (0, 1) &\Longleftrightarrow k_I \in (0, 1/k) && \text{slow and smooth control,} \\ kk_I = 1 &\Longleftrightarrow k_I = 1/k && \text{standard I Controller.} \end{aligned}$$

  There are more sophisticated controllers that are used in practical simulations, see Söderlind (2002) for an overview.

□

Remark 1.45. Methods used in practice. In practice, one uses, e.g.,

- RKF 4(5), $s = 6$, Fehlberg (1964),
- RKF 7(8), $s = 13$, Fehlberg (1969),

- DOPRI 4(5) (or DOPRI 5(4) or DOPRI5), $s = 6$, Dormand[8], Prince[9]: Dormand & Prince (1980),
- DOPRI 7(8), $s = 13$, Prince & Dormand (1981).

The standard routine `ode45` from MATLAB uses DOPRI 4(5). □

Remark 1.46. Fehlberg trick. The Fehlberg trick requires that

$$K_s = f\left(x_k + c_s h, y_k + h\sum_{i=1}^{s-1} a_{si}K_i\right) \overset{!}{=} f\left(x_k + h, \underbrace{y_k + h\sum_{i=1}^{s} b_i K_i}_{y_{k+1}}\right),$$

i.e., the last evaluation of the incremental function of the old step can be used as first value of the incremental function in the new step. The conditions for applying this trick are

$$a_{si} = b_i,\ i = 1, \ldots, s-1, \quad b_s = 0, \quad c_s = 1.$$

It can be applied, e.g., in DOPRI 4(5). This trick works only if $h_{\text{old}} \approx h_{\text{new}}$. □

[8] John R. Dormand

[9] P. J. Prince

Chapter 2
Numerical Methods for Stiff Ordinary Differential Equations

2.1 Stiff Ordinary Differential Equations

Remark 2.1. Stiffness. It was observed in Curtiss & Hirschfelder (1952) that explicit methods failed for the numerical solution of initial value problems for ordinary differential equations that model certain chemical reactions. They introduced the notation stiffness for such chemical reactions where the fast reacting components arrive in a very short time in their equilibrium and the slowly changing components are more or less fixed, i.e., stiff. In 1963, Dahlquist found out that the reason for the failure of explicit Runge–Kutta methods is their bad stability, see Section 2.3. It should be emphasized that the stability properties of the equations themselves are good, it is in fact a problem of the explicit methods.

There is no unique definition of stiffness in the literature. However, essential properties of stiff systems are as follows:

- There exist, for certain initial conditions, solutions that change slowly.
- Solutions in a neighborhood of these smooth solutions converge quickly to them.

A definition of stiffness can be found in (Strehmel & Weiner, 1995, p. 202), (Strehmel *et al.*, 2012, p. 208). This definition involves a certain norm that depends on the equation and it might be complicated to evaluate this norm. If the solution of (1.1) is sought in the interval $[x_0, x_e]$ and if the right-hand side of (1.1) is Lipschitz continuous in the second argument with Lipschitz constant L, then an approximation of this definition is as follows. A system of ordinary differential equations is called stiff if

$$L\,(x_e - x_0) \gg 1. \tag{2.1}$$

The term on the left-hand side corresponds to the term in the exponential of the error bound (1.7) for the global error. Thus, the first factor in the error bound is very large.

Another definition of stiffness will be given in Definition 2.28. □

Example 2.2. Stiff system of ordinary differential equations. Consider the system

$$\begin{aligned} y_1' &= -80.6y_1 + 119.4y_2, \\ y_2' &= 79.6y_1 - 120.4y_2, \end{aligned}$$

in $(0, 1)$. This system is a linear system of ordinary differential equations that can be written in the form

$$\boldsymbol{y}' = \begin{pmatrix} -80.6 & 119.4 \\ 79.6 & -120.4 \end{pmatrix} \boldsymbol{y}.$$

Taking as Lipschitz constant, e.g., the l_1 norm of the system matrix (column sums), one gets $L = 239.8$ and condition (2.1) is satisfied. The general solution of this system is, compare Appendix A.2.3,

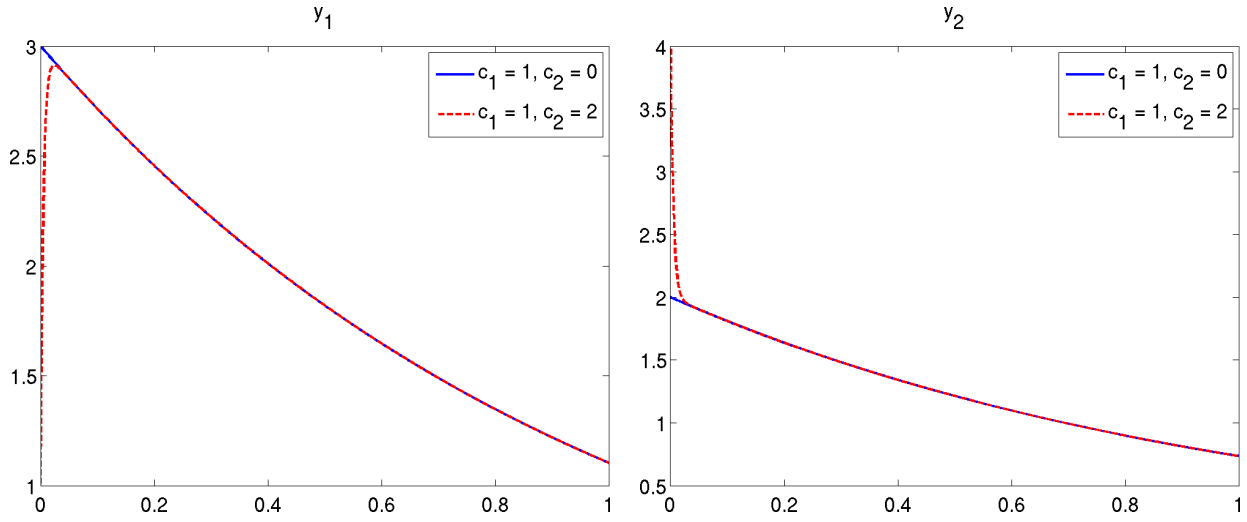

Fig. 2.1 Solutions of Example 2.2, left: first component, right: second component.

$$\boldsymbol{y}(x) = c_1 \begin{pmatrix} 3 \\ 2 \end{pmatrix} e^{-x} + c_2 \begin{pmatrix} -1 \\ 1 \end{pmatrix} e^{-200x}.$$

The first component is the slowly changing one and the second component the quickly (close to $x = 0$) changing one. The constants are determined by the initial condition. If the initial condition is such that $c_2 = 0$, then the solution is smooth for all $x > 0$. Otherwise, if $c_2 \neq 0$, then the solutions changes rapidly for small x while approaching the smooth solution, see Figure 2.1 □

2.2 Implicit Runge–Kutta Schemes

Remark 2.3. Motivation. If the upper triangular part of the matrix of a Runge–Kutta method, see Definition 1.22, is not identical to zero, the Runge–Kutta method is called implicit. That means, there are increments that depend not only on previously computed increments but also on not yet computed increments. Thus, one has to solve a nonlinear problem for computing these increments. Consequently, the implementation of implicit Runge–Kutta methods is much more involved compared with the implementation of explicit Runge–Kutta methods. Generally, performing one step of an implicit method is much more time-consuming than for an explicit method. However, the great advantage of implicit methods is that they can be used for the numerical simulation of stiff systems, see the stability theory in Section 2.3. □

Remark 2.4. Derivation of implicit Runge–Kutta methods. Implicit Runge–Kutta schemes can be derived from the integral representation (1.8) of the initial value problem. One can show that for each implicit Runge–Kutta scheme with the weights b_j and the nodes $x_k + c_j h$ there is a corresponding quadrature rule with the same weights and the same nodes, see the section on Gaussian quadrature in Numerical Mathematics I. □

Example 2.5. Gauss–Legendre quadrature. Consider the interval $[x_k, x_k + h] = [x_k, x_{k+1}]$. Let $c_1, \ldots, c_s$ be the roots of the Legendre polynomial $P_s(t)$ of degree s with the arguments

$$t = \frac{2}{h}(x - x_k) - 1 \quad \Longrightarrow \quad t \in [-1, 1].$$

There are s mutually distinct real roots in $(-1, 1)$. After having computed $c_1, \ldots, c_s$, one can determine the coefficients a_{ij}, b_j such that one obtains a method of order $2s$, see Example 2.8. □