Numerical Ordinary Differential Equations

Introductory outline of methods

1. First-order initial value problems

A first-order initial value problem is a differential equation of the form

$$\boxed{y'(x) = f(x, y(x))} \quad \text{with initial condition} \quad \boxed{y(x_0) = y_0} \tag{1.1}$$

The function f often is called the r.h.s.-function of the equation and often is subject to constraints like continuity (C^0) or continuous differentiability (C^1, C^2, C^3 ...).

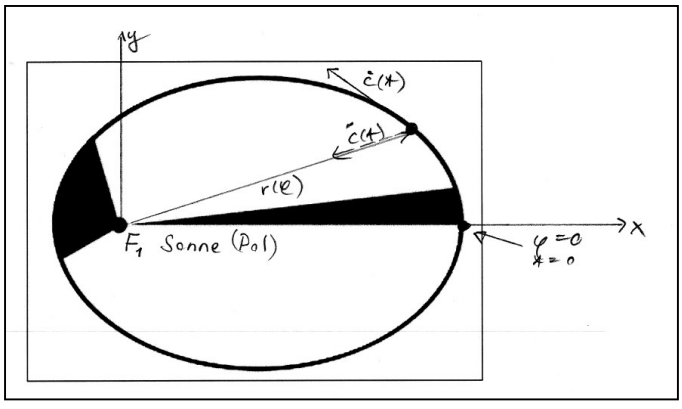
Example 1.1: Angle function of Keplerian Earth orbit from Keplers 2nd law

From Kepler's second law it follows (cf. appendix) that the angle-time function $\varphi(t)$ [1] fulfills the first-order differential equation

$$\boxed{\varphi'(t) = 2\frac{ab\pi}{T}\frac{\left(1-\varepsilon\cos(\varphi(t))\right)^2}{p^2} \qquad \varphi(0) = 0} \tag{1.2}$$

Here a and b denote the elliptic semi-axes ($a > b$), T the orbital time period, $\varepsilon = \frac{\sqrt{a^2-b^2}}{a}$ the numerical eccentrity (in case of an elliptic orbit $0 < \varepsilon < 1$) and $p = a(1-\varepsilon^2)$ the (elliptic) parameter. The angle φ denotes the polar angle (s. Figure 1.1a below).

Figure 1.1a: Elliptic orbit with gravitational center in focus F_1 (center of polar coordinate system). The polar equation of the ellipse is $r(\varphi) = \frac{p}{1-\varepsilon\cos(\varphi)}$ and the cartesian equations follow as $x = r(\varphi)\cos(\varphi),\ y = r(\varphi)\sin(\varphi)$.



A "simple" formula[2] for $\varphi(t)$ thus would yield formulas for the cartesian coordinates of the orbit and the velocity.

[1] This angle is named eccentric anomaly in celestial mechanics

[2] There is no elementary formula in closed form for $\varphi(t)$. A rather advanced representation is the following. For the circumcircle center angle $\tilde{\varphi}$ there is an infinite Fourier series

$\tilde{\varphi}(2\pi\frac{t}{T}) = 2\pi\frac{t}{T} + \sum_{n=1}^{\infty}\frac{2}{n}J_n(n\varepsilon)\sin(n\cdot 2\pi\frac{t}{T})$ with J_n denoting the Bessel functions of the first kind:

Abbreviating the constant $c := 2\frac{ab\pi}{Tp^2}$ (1.2) has the shorter form

$$\boxed{\varphi' = c\left(1-\varepsilon\cos(\varphi)\right)^2 \qquad \varphi(0)=0} \qquad \textbf{(1.2')}$$

with right-hand side function $f(t,\varphi) = c\left(1-\varepsilon\cos(\varphi)\right)^2$ only depending on φ.

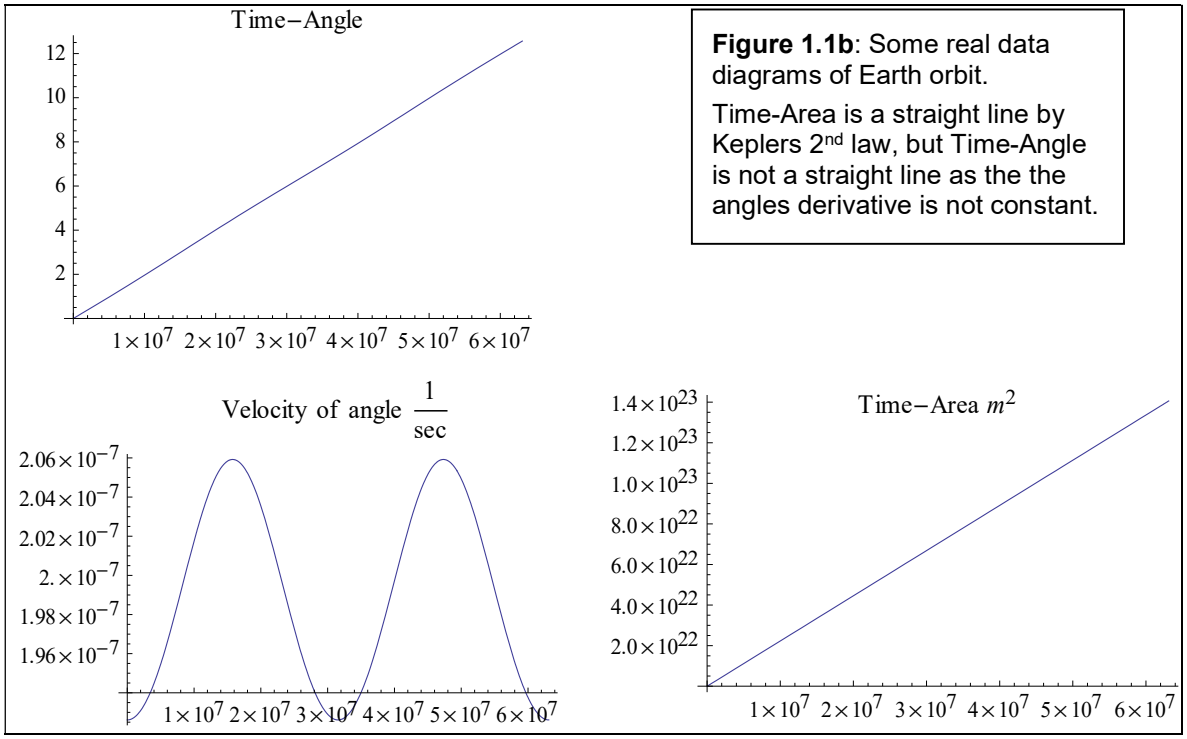


Figure 1.1b: Some real data diagrams of Earth orbit.
Time-Area is a straight line by Keplers 2nd law, but Time-Angle is not a straight line as the the angles derivative is not constant.

$J_n(x) = \frac{1}{\pi}\int_0^{\pi}\cos\left(n\tau - x\sin(\tau)\right)d\tau$ (cf. H. Heuser, Gewöhnliche Differentialgleichungen, 5th edition, Teubner).

From $\tilde{\varphi}$ one can deduce $r(t) = \sqrt{\left(\varepsilon a + a\cos(\tilde{\varphi}(t))\right)^2 + \left(b\sin(\tilde{\varphi}(t))\right)^2}$ and from this the time-dependent formulas $\varphi(t) = \arccos\left(\frac{1}{\varepsilon}\left(1-\frac{p}{r(t)}\right)\right)$, $\left(0 \le t \le \frac{T}{2}\right)$ and

$\varphi(t) = \arccos\left(\frac{1}{\varepsilon}\left(1-\frac{p}{r(t)}\right)\right)+\pi$, $\left(\frac{T}{2} \le t \le T\right)$, respectively.

1.1 Explicit methods

If the r.h.s.-function f in (1.1) fulfills smoothness conditions then solutions for (1.1) have smoothness properties, too. The Taylor expansion and the multi-variate chain rule for differentiation play a key role in the development of explicit methods. More precisely:

Property 1.1: Differentiability order for solutions and Taylor expansions

If the r.h.s. function f in (1.1) has continuous partial derivatives up to order $p \geq 1$ in an open neighbourhood U[1] of (x_0, y_0) then a solution[2] y of (1.1)

(a) is continuously differentiable with respect to x in a neighbourhood of x_0 up to order p+1, and

(b) for x, $x+h$ in a neighbourhood of x_0 has Taylor expansions of the form

$$y(x+h) =$$

$$y(x) + \frac{y'(x)}{1!}h + \frac{y''(x)}{2!}h^2 + \frac{y'''(x)}{3!}h^3 + \frac{y^{(4)}(x)}{4!}h^4 + \cdots + \frac{y^{(p)}(x)}{p!}h^p + \frac{y^{(p+1)}(\xi)}{(p+1)!}h^{p+1} =$$

$$\begin{aligned}
&y(x) + \frac{f(x,y(x))}{1!}h + \\
&\frac{1}{2!}\left(\frac{\partial f(x,y)}{\partial x}1 + \frac{\partial f(x,y)}{\partial y}f(x,y(x))\right)h^2 + \\
&\frac{1}{3!}\left(\frac{\partial^2 f(x,y)}{\partial x^2}1 + 2\frac{\partial^2 f(x,y)}{\partial x \partial y}f(x,y(x)) + \frac{\partial^2 f(x,y)}{\partial y^2}f(x,y(x))^2 + \left(\frac{\partial f(x,y)}{\partial y}\right)^2 f(x,y(x)) + \right. \\
&\qquad \left. \frac{\partial f(x,y)}{\partial x}\frac{\partial f(x,y)}{\partial y}\right)h^3 + \\
&\frac{1}{4!}\underbrace{\left(y^{(4)}(x)\right)}_{\substack{\text{Expressions in } \frac{\partial^{|\alpha|} f(x,y(x))}{\partial \{x,y\}^{|\alpha|}} \\ \text{with } |\alpha| \leq 4-1}} h^4 + \cdots + \frac{1}{p!}\underbrace{\left(y^{(p)}(x)\right)}_{\substack{\text{Expressions in } \frac{\partial^{|\alpha|} f(x,y(x))}{\partial \{x,y\}^{|\alpha|}} \\ \text{with } |\alpha| \leq p-1}} h^p + \frac{1}{(n+1)!}\underbrace{\left(y^{(p+1)}(\xi)\right)}_{\substack{\text{Expressions in } \frac{\partial^{|\alpha|} f(\xi,y(\xi))}{\partial \{x,y\}^{|\alpha|}} \\ \text{with } |\alpha| \leq p}} h^{p+1}
\end{aligned} \qquad \textbf{(1.3)}$$

(Proof: By the chain rule of multi-variate calculus

$$y''(x) = \nabla f(x,y) \cdot (1, y'(x))^{tr} = \frac{\partial f(x,y)}{\partial x}1 + \frac{\partial f(x,y)}{\partial y}y'(x) \qquad (**)$$

---

[1] A typical situation is a rectangular neighbourhood "around" (x_0, y_0):
$U = \{x_0 - h_1 < x < x_0 + h_1 \wedge y_0 - h_2 < y < y_0 + h_2\}$.

[2] To be more precise: If f is continuous and has a continuous partial derivative $\frac{\partial f(x,y)}{\partial y}$ in U, then there is a unique (local) solution for (1.1) in a neighbourhood of (x_0, y_0). This is the Picard-Lindelöf theorem.

If f is continuous in U then there is a local solution for (1.1) in a neighbourhood of (x_0, y_0). This is the Peano theorem. For the proofs cf. H. Heuser, Gewöhnliche Differentialgleichungen, 5th edition, Teubner, or Schaum's Outline of Numerical Analysis, 1st ed., Problem 19.7 (Picard-Lindelöf theorem).

This already proves the property (a) for p = 1 since the r.h.s $\frac{\partial f(x,y)}{\partial x}1+\frac{\partial f(x,y)}{\partial y}y'(x)$ (**) is C^{p-1}, which is at least C^0. But by (**) again $y''(x)$ is C^{p-1} and thus y must be C^{p+1}.

The proof of (b) is similar: Iteratively, applying the chain rule in combination with the product rule to the equation (**) gives the expressions in (1.3), e.g.:

$$y'''(x)=\left(\frac{\partial^2 f(x,y)}{\partial x^2}1+2\frac{\partial^2 f(x,y)}{\partial x\partial y}f(x,y(x))+\frac{\partial^2 f(x,y)}{\partial y^2}f(x,y(x))^2+\left(\frac{\partial f(x,y)}{\partial y}\right)^2 f(x,y(x))+\frac{\partial f(x,y)}{\partial x}\frac{\partial f(x,y)}{\partial y}\right)$$

■)

1.1.1 Explicit Euler method

From Property 1.1b the explicit Euler method can be deduced when setting the order p = 1:

$$y(x+h)=y(x)+y'(x)h+\frac{y^{(2)}(\xi)}{(2)!}h^2$$

$$y(x+h)=y(x)+f(x,y(x))h+\frac{1}{2!}\left(\frac{\partial f(\xi,y(\xi))}{\partial x}1+\frac{\partial f(\xi,y(\xi))}{\partial y}f(\xi,y(\xi))\right)h^2 \qquad \textbf{(1.3A)}$$

Applying (1.3A) iteratively for $x=x_0+kh \quad (k=0,...,n)$ with a fixed step-size h yields the following scheme for computing a discrete sequence of y-values:

$$\left.\begin{array}{l} y_0=y(x_0) \\ y(x_0+h)\approx y_1=y_0+f(x_0,y_0)h=y(x_0)+y'(x_0)h \\ y(x_1+h)\approx y_2=y_1+f(x_1,y_1)h\approx y(x_1)+y'(x_1)h \\ \vdots \\ y(x_{n-1}+h)\approx y_n=y_{n-1}+f(x_{n-1},y_{n-1})h\approx y(x_{n-1})+y'(x_{n-1})h \\ y(x_n+h)\approx y_{n+1}=y_n+f(x_n,y_n)h\approx y(x_n)+y'(x_n)h \end{array}\right\} y_{k+1}=y_k+f(x_k,y_k)h \qquad \textbf{(1.3A')}$$

This scheme is called explicit Euler method. The size of h is called step size. It has not necessarily to be constant and thus may have an index k, too.

Example 1.2: Cont. Ex. 1.1: Angle function of Keplerian Earth orbit from Keplers 2nd law

According to (1.2') the time differential equation for the angle function is $\varphi'=c(1-\varepsilon\cos(\varphi))^2$ with initial value $\varphi(0)=0$. The right-hand side function is $f(t,\varphi)=c(1-\varepsilon\cos(\varphi))^2$ only depending on φ. The scheme (1.3A') takes the following form:

$$\varphi_0 = 0 \quad t_0 = 0$$
$$\varphi_1 = \varphi_0 + f(t_0, \varphi_0)h = 0 + c(1-\varepsilon)^2 h$$
$$\varphi_2 = \varphi_1 + f(t_1, \varphi_1)h = c(1-\varepsilon)^2 h + c\left(1 - \varepsilon\cos(c(1-\varepsilon)^2 h)\right)^2 h$$
$$\vdots$$
$$\varphi_{k+1} = \varphi_k + f(t_k, \varphi_k)h = \varphi_k + c(1 - \varepsilon\cos(\varphi_k))^2 h$$

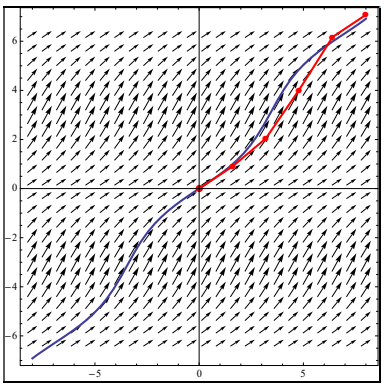


Figure 1.2a: $c = 1, \varepsilon = 0.25, h = 1.6, n = 4$

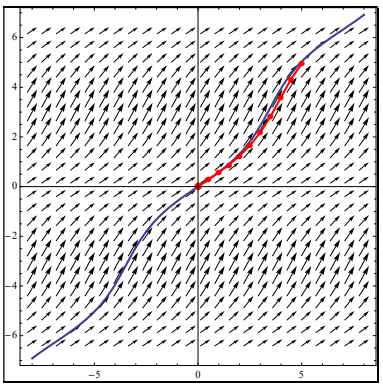


Figure 1.2b: $c = 1, \varepsilon = 0.25, h = 0.5, n = 9$

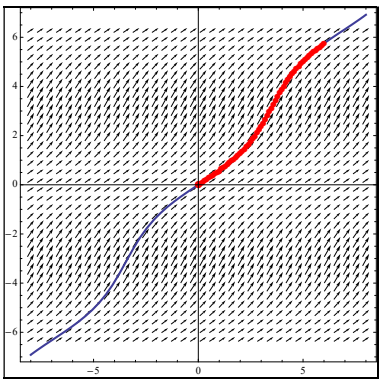


Figure 1.2c: $c = 1, \varepsilon = 0.25, h = 0.1, n = 59$

Obviously, there are errors between the approximation values y_k and the "true" solution values $y(x_k)$. The explicit Euler method is rather inaccurate since it assumes that the derivative remains constant per step. In spite of that the method is convergent if the r.h.s-function f is continuous in a neighbourhood of the initial point. Convergence means that for $x = x_k = x_{k-1} + h_{k-1} \quad (k \in N)$ in an appropriate neighbourhood of x_0 the maximum difference

$$\max_{0 \le i \le k} |y_i - y(x_i)| \to 0 \quad (h = \max_i h_i \to 0) \tag{1.4}$$

This quantitiy is called global error[1]. (Fixing $x = x_k$ means that $k \to \infty$ if $\max_i h_i \to 0$.)

The explicit Euler method is a special case of the Taylor method: It is an iterative application of Property 1.1b when setting the order p to 1. Setting p to higher values yields higher-order Taylor methods.

---

[1] The local error or single-step error of a p-th order local Taylor method is defined as $\tau_h(x_n) := \frac{y(x_n + h) - y(x_n)}{h} - \left(\frac{y'(x_n)}{1!} + \frac{y''(x_n)}{2!} h^1 + \cdots + \frac{y^{(p)}(x_n)}{p!} h^{p-1} \right)$ or multiplied by the step-size h as $h\tau_h(x_n) := y(x_n + h) - y(x_n) - \left(\frac{y'(x_n)}{1!} h + \frac{y''(x_n)}{2!} h^2 + \cdots + \frac{y^{(p)}(x_n)}{p!} h^p \right)$. Assuming that the initial value of the current step is exactly correct, i.e. $y_n = y(x_n)$, the local error $h\tau_h(x_n)$ is equal to $y(x_n + h) - \underbrace{\left(y_n + \frac{y'(x_n)}{1!} h + \frac{y''(x_n)}{2!} h^2 + \cdots + \frac{y^{(p)}(x_n)}{p!} h^p \right)}_{y_{n+1}} = y(x_n + h) - y_{n+1}$ which is intuitively the meaning of a local error.

1.1.2 Higher-order local Taylor methods

From Property 1.1b the explicit Euler method was deduced from the Taylor method by setting the order p equal to 1. This was a truncation of the Taylor expansion after the first derivative. Using Property 1.1b again but truncating not before the p th derivative yields a generic scheme of order p:

$$y(x_k + h_k) \approx y(x_k) + \frac{y'(x_k)}{1!}h_k + \frac{y''(x_k)}{2!}h_k{}^2 + \frac{y'''(x_k)}{3!}h_k{}^3 + \frac{y^{(4)}(x_k)}{4!}h_k{}^4 + \cdots + \frac{y^{(p)}(x_k)}{p!}h_k{}^p$$

which in short form is written as

$$\boxed{\begin{aligned} y_0 &= y(x_0) \\ y_{k+1} &= y_k + \frac{y_k{}'}{1!}h_k + \frac{y_k{}''}{2!}h_k{}^2 + \frac{y_k{}'''}{3!}h_k{}^3 + \frac{y_k{}^{(4)}}{4!}h_k{}^4 + \cdots + \frac{y_k^{(p)}}{p!}h_k{}^p \end{aligned}} \qquad \textbf{(1.5)}$$

The disadvantage of this method are the costs of the computations of higher-order derivatives, as:

$$y''(x) = \left(\frac{\partial f(x,y)}{\partial x}1 + \frac{\partial f(x,y)}{\partial y}f(x,y(x))\right) \Leftrightarrow y_k{}'' = \left(\frac{\partial f(x_k,y_k)}{\partial x}1 + \frac{\partial f(x_k,y_k)}{\partial y}f(x_k,y_k)\right)$$

$$y'''(x) = \left(\frac{\partial^2 f(x,y)}{\partial x^2}1 + 2\frac{\partial^2 f(x,y)}{\partial x \partial y}f(x,y(x)) + \frac{\partial^2 f(x,y)}{\partial y^2}f(x,y(x))^2 + \left(\frac{\partial f(x,y)}{\partial y}\right)^2 f(x,y(x)) + \frac{\partial f(x,y)}{\partial x}\frac{\partial f(x,y)}{\partial y}\right) \Leftrightarrow$$

$$y_k{}''' = \left(\frac{\partial^2 f(x_k,y_k)}{\partial x^2}1 + 2\frac{\partial^2 f(x_k,y_k)}{\partial x \partial y}f(x_k,y_k) + \frac{\partial^2 f(x_k,y_k)}{\partial y^2}f(x_k,y_k)^2 + \left(\frac{\partial f(x_k,y_k)}{\partial y}\right)^2 f(x_k,y_k) + \frac{\partial f(x_k,y_k)}{\partial x}\frac{\partial f(x_k,y_k)}{\partial y}\right)$$

a.s.o.…

Example 1.3: Cont. Ex. 1.1: Angle function of Keplerian Earth orbit from Keplers 2nd law

According to (1.2') the time differential equation for the angle function is $\varphi' = c(1-\varepsilon\cos(\varphi))^2$ with initial value $\varphi(0) = 0$. The right-hand side function is $f(t,\varphi) = c(1-\varepsilon\cos(\varphi))^2$ only depending on φ. In order to implement the scheme (1.5) we need the partial derivatives of f.

$$\frac{\partial}{\partial t}f = 0, \quad \frac{\partial}{\partial \varphi}f = 2c(1-\varepsilon\cos(\varphi))\varepsilon\sin(\varphi) = 2c\varepsilon\sin(\varphi) - c\varepsilon^2\sin(2\varphi)$$

From this it is already obvious that any partial derivative containing the operation $\frac{\partial}{\partial t}$ is 0 and only derivatives with respect to φ have to be considered:

$$\frac{\partial^2}{\partial\varphi^2} f = 2c\varepsilon\cos(\varphi) - c\varepsilon^2 2\cos(2\varphi)$$

$$\frac{\partial^3}{\partial\varphi^3} f = -2c\varepsilon\sin(\varphi) + c\varepsilon^2 2^2\sin(2\varphi)$$

$$\frac{\partial^4}{\partial\varphi^4} f = -2c\varepsilon\cos(\varphi) + c\varepsilon^2 2^3\cos(2\varphi)$$

$$\vdots$$

For order p = 2 the scheme in (1.5) with constant step size $h_k = h$ takes the following form:

$$\varphi_0 = 0$$

$$\varphi_{k+1} = \varphi_k + c\left(1-\varepsilon\cos(\varphi_k)\right)^2 h + \frac{1}{2!}\left(2c\varepsilon\sin(\varphi_k) - c\varepsilon^2\sin(2\varphi_k)\right)c\left(1-\varepsilon\cos(\varphi_k)\right)^2 h^2$$

The next figures show a number of steps with constant step-size. They should be compared with Figure 1.2ab.

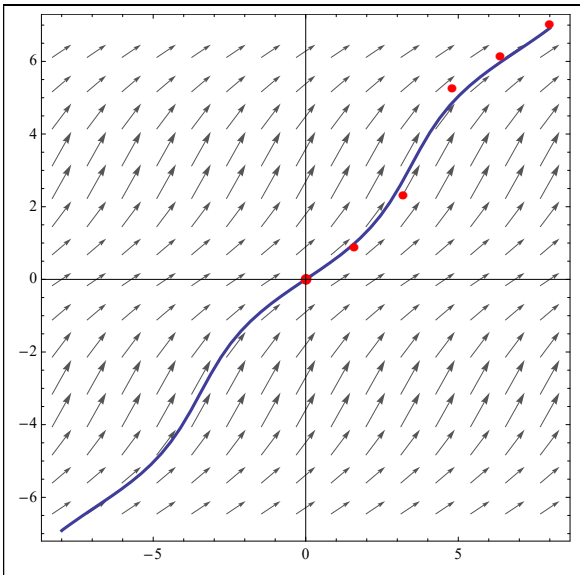


Figure 1.3a: Local Taylor method of order 2 with $c = 1, \varepsilon = 0.25, h = 1.6, n = 4$

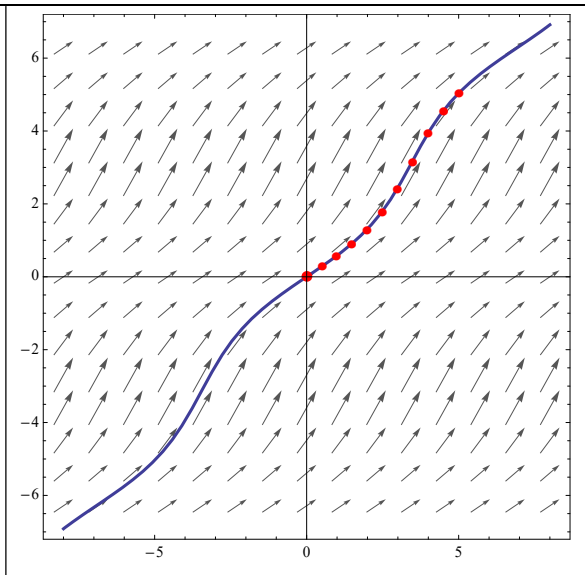


Figure 1.3b: Local Taylor method of order 2 with $c = 1, \varepsilon = 0.25, h = 0.5, n = 9$

For the proof of the convergence in formula (1.4) we need a special notion of continuity called Lipschitz-continuity.

Property 1.2: Lipschitz continuity of derivatives up to order p-1.

If the r.h.s. function f in (1.1) has continuous partial derivatives up to order $p \geq 1$ in a compact convex set K then there is a constant $L \geq 0$ such that

$$\left|\frac{\partial^{\alpha_1+\alpha_2}}{\partial x^{\alpha_1}\partial y^{\alpha_2}} f(x,y) - \frac{\partial^{\alpha_1+\alpha_2}}{\partial x^{\alpha_1}\partial y^{\alpha_2}} f(x,\tilde{y})\right| \leq L\left|y-\tilde{y}\right| \quad (x,y,\tilde{y}\in K) \qquad \textbf{(1.6)}$$

if $\alpha_1 + \alpha_2 \leq p-1$.

The constant $L \geq 0$ is called Lipschitz constant. The set K often is a rectangular neighbourhood around a central point.

(Proof: If $x, y, \tilde{y} \in K$ and $y \neq \tilde{y}$ then by the mean-value theorem from calculus the ratio $\dfrac{\left| \dfrac{\partial^{\alpha_1+\alpha_2}}{\partial x^{\alpha_1} \partial y^{\alpha_2}} f(x,y) - \dfrac{\partial^{\alpha_1+\alpha_2}}{\partial x^{\alpha_1} \partial y^{\alpha_2}} f(x,\tilde{y}) \right|}{|y-\tilde{y}|}$ is equal to a partial derivative $\left| \dfrac{\partial}{\partial y} \left(\dfrac{\partial^{\alpha_1+\alpha_2}}{\partial x^{\alpha_1} \partial y^{\alpha_2}} f(x,\xi) \right) \right|$ with $\xi_x \in (y, \tilde{y})$. But by continuity this partial derivative of order $\leq p$ is bounded by $\max\limits_K \left| \dfrac{\partial}{\partial y} \left(\dfrac{\partial^{\alpha_1+\alpha_2}}{\partial x^{\alpha_1} \partial y^{\alpha_2}} f(x,y) \right) \right|$ on K

■)

Theorem 1.1: Convergence rate of global error for local Taylor methods

Assuming that the r.h.s. function f in (1.1) has continuous partial derivatives up to order $p \geq 1$ in a rectangular compact set R around the central point (x_0, y_0) and that (1.1) has a solution y in R.

Then the local Taylor method (1.5) of order p is convergent with order p:

$$\max_{0 \leq i \leq k} |y_i - y(x_i)| \leq B \frac{h^p}{(p+1)!} \frac{e^{M(x_k - x_0)} - 1}{M} = O(h^p) \quad (h \to 0) \tag{1.7}$$

with constants B, M and $h = \max\limits_{0 \leq i \leq k} h_i$.

(Proof: Cf. Problem 19.12 in Schaum's Outline of Numerical Analysis, 2nd ed. (or Problem 19.18 in 1st ed., respectively). The key elements in the proof are the Taylor expansions according to Property 1.1 and the Lipschitz continuity due to Property 1.2 ■)

1.1.3 Explicit Runge-Kutta methods

As mentioned above the local Taylor methods have rather high costs arising from computions of higher-order derivatives. On the other hand, as an advantage the order of convergence rate is the same as the order of the highest derivative (cf. Theorem 1.1 or (1.7)).

The explicit Runge-Kutta methods are devised so as to keep a high-order convergence rate while avoiding the computations of high-order derivatives. Instead, these methods use additional (intermediate) evaluations of the r.h.s. function f. The next subsection gives the idea behind the methods:

1.1.3.1 Explicit Runge-Kutta methods of order 2 (Heun, Midpoint)

We are to find coefficients a, b and m, n such that the (so-called 2-stage Runge-Kutta) formulas

$$\left.\begin{aligned} k_1 &= h\, f(x,y) \\ k_2 &= h\, f(x + mh, y + mk_1) \end{aligned}\right\} \; 2 \; stages$$

$$y(x+h) \approx y(x) + ak_1 + bk_2 \tag{1.8}$$

duplicate the Taylor series of the solution y through the (second-order) term in h^2. The last formula of (1.8), though not a polynomial expression, is then near the Taylor polynomial of order 2.

Introducing the usual abbreviations for f and its partial derivatives we get from (1.5):

$$y(x+h) \approx y(x) + f\,h + \frac{1}{2!}\underbrace{\left(f_x + f_y f\right)}_{F_1}h^2 + \frac{1}{3!}\left(\underbrace{f_{xx} + 2f_{xy}f + f_{yy}f^2}_{F_2} + \underbrace{f_y f_x + f_y^2 f}_{f_y F_1}\right)h^3 \quad \textbf{(1.8a)}$$

Multi-variate Taylor expansions for the k-values produce:

$$\begin{aligned}
k_1 &= hf \\
k_2 &= h\left(f + f_x mh + f_y mk_1 + \frac{1}{2!}f_{xx}m^2h^2 + f_{xy}(mh)(mk_1) + \frac{1}{2!}f_{yy}m^2k_1^2 + \cdots\right) \\
&= h\left(f + f_x mh + f_y mhf + \frac{1}{2!}f_{xx}m^2h^2 + f_{xy}(mh)(mhf) + \frac{1}{2!}f_{yy}m^2(hf)^2 + \cdots\right) \qquad \textbf{(1.8b)} \\
&= h\left(f + m\underbrace{(f_x + f_y f)}_{F_1}h + \frac{1}{2!}m^2\underbrace{\left(f_{xx} + 2f_{xy}f + f_{yy}f^2\right)}_{F_2}h^2 + \cdots\right)
\end{aligned}$$

Combining the expressions in (1.8b) as suggested in (1.8) yields

$$y(x+h) \approx y(x) + (a+b)hf + (bm)F_1h^2 .$$

Comparing coefficients with (1.8a) up to order 2 then gives a redundant system of equations[1]:

$$a+b=1\,,\; bm=\frac{1}{2}. \qquad \textbf{(1.8c)}$$

A (possible) solution is $a=b=\frac{1}{2}$, $m=1$ and then

$$y(x+h) \approx y(x) + \frac{1}{2}hf(x,y) + \frac{1}{2}hf\left(x+1h, y+1hf(x,y)\right).$$

This is called Heun's method. Its recursive scheme in the discrete sense of (1.5) is:

$$\begin{aligned}
y_0 &= y(x_0) \\
y_{k+1} &= y_k + \frac{1}{2}h_k\left(f(x_k,y_k) + f(x_{k+1}, y_k + h_k f(x_k,y_k))\right)
\end{aligned} \qquad \textbf{(1.9a)}$$

with $x_{k+1} = x_k + h_k$ and step-size $h_k > 0 \quad (k = 0,1,...)$.

Another (possible) solution is $a=0$, $b=1$, $m=\frac{1}{2}$ and then

$$y(x+h) \approx y(x) + hf\left(x+\frac{1}{2}h, y+\frac{1}{2}hf(x,y)\right).$$

---

[1] It is not possible to match the whole cubic h^3-coefficient of (1.8a) with 2 stages: The best thing to achieve would be to match the F_2-part with the additional condition $\frac{1}{2!}bm^2 = \frac{1}{3!}$. This would determine $b=3/4, m=2/3$ uniquely. But in any case the part $\frac{1}{3!}\left(f_y f_x + 2f_y^2 f\right)$ remains unmatched.

The latter is called explicit midpoint method. Its recursive scheme is:

$$y_0 = y(x_0)$$
$$y_{k+1} = y_k + h_k\left(f(x_k + \frac{1}{2}h_k, y_k + \frac{1}{2}h_k f(x_k, y_k)) \right) \qquad \textbf{(1.9b)}$$

with $x_{k+1} = x_k + h_k$ and step-size $h_k > 0 \quad (k = 0,1,...)$.

Due to redundancy in the system (1.8c) an infinite number of solutions is possible. Yet another solution often used is $a = \frac{1}{4}, b = \frac{3}{4}, m = \frac{2}{3}$ (cf. footnote to 1.8c for the optimality of this). Its recursive scheme is

$$y_0 = y(x_0)$$
$$y_{k+1} = y_k + \frac{1}{4}h_k f(x_k, y_k) + \frac{3}{4}h_k\left(f(x_k + \frac{2}{3}h_k, y_k + \frac{2}{3}h_k f(x_k, y_k)) \right) \qquad \textbf{(1.9c)}$$

with $x_{k+1} = x_k + h_k$ and step-size $h_k > 0 \quad (k = 0,1,...)$.

The convergence rate of the Runge-Kutta methods of order 2 is 2 if the conditons of Theorem 1.1 are met. This is not a surprise, since the Runge-Kutta methods duplicate the local Taylor polynomial up to order 2 they duplicate the convergence rate, too.

1.1.3.2 The classical Runge-Kutta method of order 4

A classical and often used Runge-Kutta method generalizes the ideas of the previous section to order 4 in a straightforward way.

Theorem 1.2: Common Runge-Kutta method of order 4

Assuming that the r.h.s. function f in (1.1) has continuous partial derivatives up to order p = 4 in a rectangular compact set R around the central point (x_0, y_0) and that (1.1) has a solution y in R.

The 4-stage Runge-Kutta formulas

$$k_1 = h\, f(x, y)$$
$$k_2 = h\, f(x + mh, y + mk_1)$$
$$k_3 = h\, f(x + nh, y + nk_2)$$
$$k_4 = h\, f(x + ph, y + pk_3) \qquad \textbf{(1.10a)}$$

$$y(x + h) \approx y(x) + ak_1 + bk_2 + ck_3 + dk_4$$

duplicate the Taylor polynomial in (1.5) up to order p = 4 if the following (redundant) system of 8 equations is fulfilled:

$$\begin{array}{rclrcl} a+b+c+d & = & 1 & cmn+dnp & = & \frac{1}{6} \\ bm+cn+dp & = & \frac{1}{2} & cmn^2+dnp^2 & = & \frac{1}{8} \\ bm^2+cn^2+dp^2 & = & \frac{1}{3} & cm^2n+dn^2p & = & \frac{1}{12} \\ bm^3+cn^3+dp^3 & = & \frac{1}{4} & dmnp & = & \frac{1}{24} \end{array} \qquad \textbf{(1.10b)}$$

The rate of convergence then is of order p=4, i.e. $o(h^4) \quad (h \to 0)$.

(Proof: The proof is along the same lines as in Subsection 1.1.3.1 for the second-order Runge-Kutta methods, cf. (1.8). For all the details cf. Schaum's outline of Numerical Analysis, 2nd ed., Problem 19.6 ■)

Example 1.4: Classical Runge-Kutta method of order 4

The classical solution to (1.9b) is $m=n=\frac{1}{2}, \quad p=1, \quad a=d=\frac{1}{6}, \quad b=c=\frac{1}{3}$ leading to the Runge-Kutta formulas

$$\begin{aligned} k_1 &= h\,f(x,y) \\ k_2 &= h\,f(x+\tfrac{1}{2}h, y+\tfrac{1}{2}k_1) \\ k_3 &= h\,f(x+\tfrac{1}{2}h, y+\tfrac{1}{2}k_2) \\ k_4 &= h\,f(x+h, y+k_3) \end{aligned} \qquad \textbf{(1.10c)}$$

$$y(x+h) \approx y(x) + \frac{1}{6}\left(k_1+2k_2+2k_3+k_4\right)$$

It is remarkable that for a (continuous) r.h.s-function $f = f(x)$ not depending on y the classical Runge-Kutta formulas (1.9c) turn into an integral approximation formula (Simpson's rule):

$$\int_{x_0}^{x_0+h} \underbrace{f(x)}_{y'(x)} dx = y(x+h) - y(x) \approx \frac{1}{6}h\left(f(x) + 4f\left(x_0+\frac{h}{2}\right) + f(x_0+h)\right) \qquad \textbf{(1.10d)}$$

Example 1.5: Cont. Ex. 1.1: Angle function of Keplerian Earth orbit from Keplers 2nd law

According to (1.2') the time differential equation for the angle function is $\varphi' = c\left(1-\varepsilon\cos(\varphi)\right)^2$ with initial value $\varphi(0)=0$. The right-hand side function is $f(t,\varphi) = c\left(1-\varepsilon\cos(\varphi)\right)^2$ only depending on φ.

The classical Runge-Kutta 4-stage formulas turn into the following recursion:

$$\left.\begin{aligned} k_1 &= h_k c\left(1-\varepsilon\cos(\varphi_k)\right)^2 \\ k_2 &= h_k c\left(1-\varepsilon\cos(\varphi_k+\frac{1}{2}k_1)\right)^2 \\ k_3 &= h_k c\left(1-\varepsilon\cos(\varphi_k+\frac{1}{2}k_2)\right)^2 \\ k_4 &= h_k c\left(1-\varepsilon\cos(\varphi_k+k_3)\right)^2 \end{aligned}\right\} \quad 4\ stages$$

$$\varphi_0 = \varphi(t_0) = \varphi(0) = 0$$

$$\varphi_{j+1} = \varphi_j + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right) \quad (j = 0,1,\ldots)$$

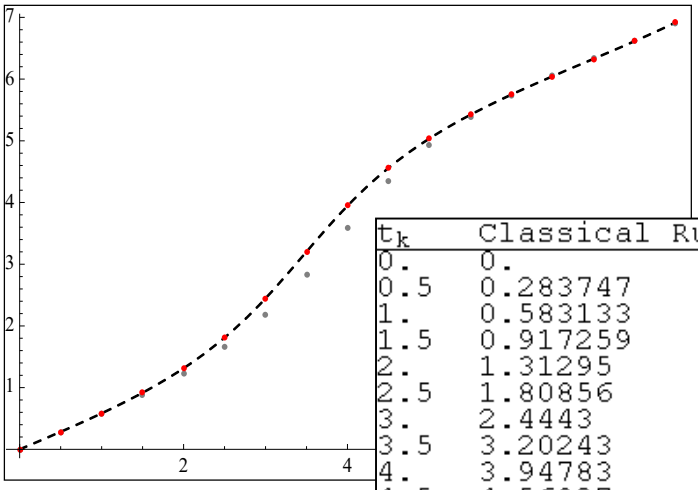


Figure 1.4: Fixed step size h = 0.5. The figure compares the classical Runge-Kutta method (red points) with the explicit Euler method (gray points) and a high-precision reference solution (dashed curve) for $c = 1, \varepsilon = 0.25$.

t_k	Classical Runge-Kutta	ExplicitEuler
0.	0.	0.
0.5	0.283747	0.28125
1.	0.583133	0.569915
1.5	0.917259	0.881581
2.	1.31295	1.23524
2.5	1.80856	1.6563
3.	2.4443	2.17788
3.5	3.20243	2.83067
4.	3.94783	3.597
4.5	4.56027	4.34673
5.	5.03737	4.94012
5.5	5.42126	5.38527
6.	5.74846	5.7416
6.5	6.04428	6.05022

Tables 1.1ab: Numerical comparisons of globar errors for different step sizes: (h = 0.5, h = 0.1)

t_k	Classical Runge-Kutta	ExplicitEuler
0.	-3.48978×10^{-19}	-1.72795×10^{-18}
0.1	0.0562698	0.05625
0.2	0.112658	0.112559
0.3	0.169286	0.169047
0.4	0.226274	0.225833
0.5	0.283748	0.283039
0.6	0.341837	0.340791
0.7	0.400675	0.399218
0.8	0.460404	0.458456
0.9	0.521171	0.518645
1.	0.583136	0.579934
1.1	0.646465	0.642483
1.2	0.711341	0.706458
1.3	0.777956	0.772041
1.4	0.846521	0.839425
1.5	0.917263	0.908819
1.6	0.990428	0.980446

1.1.3.3 A general framework for Runge-Kutta methods (Butcher tableaus)

In this subsection the method and computations of the previous section will be extended. The generalization of formula (1.9a) is the *s*-stage Runge-Kutta procedure:

$$\left.\begin{aligned}
k_1 &= f(x + c_1 h, \underbrace{y + h\sum_{j=1}^{s} a_{1,j} k_j}_{g_1}) \\
k_2 &= f(x + c_2 h, \underbrace{y + h\sum_{j=1}^{s} a_{2,j} k_j}_{g_2}) \\
k_3 &= f(x + c_3 h, \underbrace{y + h\sum_{j=1}^{s} a_{3,j} k_j}_{g_3}) \\
&\vdots \\
k_s &= f(x + c_s h, \underbrace{y + h\sum_{j=1}^{s} a_{s,j} k_j}_{g_s})
\end{aligned}\right\} \; s \text{ stages}$$

$$y(x+h) \approx y(x) + h\left(\sum_{j=1}^{s} b_j k_j\right) \qquad \textbf{(1.11a)}$$

A step from x_n to $x_{n+1} = x_n + h_n \quad (n = 0,1,...)$ in the general Runge-Kutta method then is

$$\left.\begin{aligned}
k_1 &= f(x_n + c_1 h_n, y_n + h_n\sum_{j=1}^{s} a_{1,j} k_j) \\
k_2 &= f(x_n + c_2 h_n, \underbrace{y_n + h_n\sum_{j=1}^{s} a_{2,j} k_j}_{g_2}) \\
k_3 &= f(x_n + c_3 h_n, \underbrace{y_n + h_n\sum_{j=1}^{s} a_{3,j} k_j}_{g_3}) \\
&\vdots \\
k_s &= f(x_n + c_s h_n, \underbrace{y_n + h_n\sum_{j=1}^{s} a_{s,j} k_j}_{g_s})
\end{aligned}\right\} \; s \text{ stages}$$

$$y_{n+1} = y_n + h_n\sum_{j=1}^{s} b_j k_j \qquad \textbf{(1.11b)}$$

The numbers $a_{i,j}, b_j$ and $c_j \quad (i,j = 1,2,...,s)$ - usually called *a*-numbers, *b*-numbers or *c*-numbers, respectively - are laid out in a Butcher tableau:

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \dots & a_{1s} \\ c_2 & a_{21} & a_{22} & \dots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \dots & a_{ss} \end{array} = -$$

For explicit Runge-Kutta methods it is required that $c_1 = 0, a_{1j} = 0$ and that the matrix of a-numbers is strictly lower triangular: $a_{ij} = 0 \quad (j \geq i)$. The Butcher tableau then specializes to

$$\begin{array}{c|cccc} 0 & 0 & 0 & \cdots & 0 \\ c_2 & a_{2,1} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s,1} & a_{s,2} & \cdots & a_{s,s} \end{array} \qquad \textbf{(1.11c)}$$

Moreover, it is common to require the row-sum-property: $c_i = \sum_{j=1}^{s} a_{ij} = \sum_{j=1}^{i-1} a_{ij} \quad (i = 2,...,s)$ **(1.11d)**

Property 1.3 : Consistency condition for explicit Runge-Kutta methods

The local error of a step is defined as $\tau_h(x_n) := \dfrac{y(x_n + h) - y(x_n)}{h} - \left(\sum_{j=1}^{s} b_j k_j \right).$ **(1.12a)**

Assuming that the r.h.s. function f in (1.1) has continuous partial derivatives in a rectangular compact set R around the central point (x_0, y_0) and that (1.1) has a solution y in R, then

$$\underbrace{\max}_{n} |\tau_h(x_n)| \to 0 \quad (h = \underbrace{\max}_{n} h_n \to 0) \qquad \textbf{(1.12b)}$$

provided that $\sum_{j=1}^{s} b_j = 1$.

(Proof: The proof is along the same lines as the formulas (1.8ab). By a first-order Taylor expansion:

$$y(x_n + h) = y(x_n) + f(x_n, y(x_n))\, h + o(h) \Rightarrow \frac{y(x_n + h) - y(x_n)}{h} - \underbrace{f(x_n, y(x_n))}_{y'(x_n)} = o(1) \qquad (*)$$

On the other hand, first-order expansions for the k-values produce (cf. 1.8b):

$$k_1 = f, \quad k_j = f + o(1) \quad (j = 2,...,s)$$

Combining the expressions as suggested in (1.11bc) yields

$$\frac{y(x_n+h)-y(x_n)}{h}-\left(\sum_{j=1}^{s} b_j k_j\right)=\frac{y(x_n+h)-y(x_n)}{h}-\left(\sum_{j=1}^{s} b_j\left(f+o(1)\right)\right)=$$

$$\frac{y(x_n+h)-y(x_n)}{h}-\left(\sum_{j=1}^{s} b_j\right) f-\left(\sum_{j=1}^{s} b_j\right) o(1)$$

Setting $\left(\sum_{j=1}^{s} b_j\right)$ to 1 yields (1.12) from (∗)

■)

The property (1.12) of the local error is named consistency condition.

Example 1.6: Butcher tableaus for some explicit Runge-Kutta methods

$\begin{array}{c\|cccc} 0 & & & & \\ 1/2 & 1/2 & & & \\ 1/2 & 0 & 1/2 & & \\ 1 & 0 & 0 & 1 & \\ \hline & 1/6 & 1/3 & 1/3 & 1/6 \end{array}$	$\begin{array}{c\|c} 0 & \\ \hline & 1 \end{array}$ $\begin{array}{c\|cc} 0 & & \\ 1/2 & 1/2 & \\ \hline & 0 & 1 \end{array}$	$\begin{array}{c\|cc} 0 & & \\ 2/3 & 2/3 & \\ \hline & 1/4 & 3/4 \end{array}$
Classical Runge-Kutta of order 4 (cf. 1.10c)	Explicit Euler method (cf. 1.3A') Explicit Midpoint method (1.9b)	Yet another method (cf. 1.9c)

Example 1.7: FSAL-Schemes

A interesting class of explicit Runge–Kutta methods, used in modern implementations, are those for which the coefficients have a special structure known as First Same As Last (FSAL):

$a_{sj}=b_j \quad (j=1,\ldots,s-1)$ and $b_s=0$.

For consistent FSAL schemes (cf. Property 1.3) the Butcher tableau has the form:

$$\begin{array}{c|ccc} 0 & 0 & 0 & \cdots \\ c_2 & a_{2,1} & 0 & \cdots \\ \vdots & \vdots & \vdots & \ddots \\ c_{s-1} & a_{s-1,1} & a_{s-1,2} & \cdots \\ \hline 1 & b_1 & b_2 & \cdots \quad b_{s-1} \end{array}$$

with $\sum_{j=1}^{s-1} b_j = 1$.

An advantage of FSAL methods is that the function value k_s at the end of one step is the same as the first function value k_1 at the next integration step. The most important examples for FSAL-schemes are the embedded Dormand-Prince adaptive Runge-Kutta method of orders 5 and 4, respectively, also known as RKDP5(4), or the Bogacki-Shampine 5(4) method, also known as RKBS5(4).

1.1.4 Adaptive explicit methods

Standard implementations of numerical solvers for differential equations support features for step-size (h_k) adaption. This means that there is a need of some sort of generic quantitative criteria and norms allowing the computation (i.e. adaption) of an "optimal" step size. Most common implementations use estimates of the local error (1.12a) to control the step size.

1.1.4.1 Embedded pairs of explicit Runge-Kutta methods

An embedded pair of explicit Runge-Kutta methods consists of two Runge-Kutta schemes of different orders p and $\hat{p}$ that share the same coefficient matrix and hence function values. A commonly used notation is $p(\hat{p})$ typically with $\hat{p} = p-1$ or $\hat{p} = p+1$. This constitutes an efficient means of obtaining local error estimates for adaptive step-size control. The combined Butcher tableaus of an embedded pair has the form (cf 1.11.c):

$$\begin{array}{c|cccccc} 0 & 0 & 0 & \cdots & 0 & 0 \\ c_2 & a_{2,1} & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 & \vdots \\ c_{s-1} & a_{s-1,1} & a_{s-1,2} & \cdots & 0 & 0 \\ c_s & a_{s,1} & a_{s,2} & \cdots & a_{s,s-1} & 0 \\ \hline & b_1 & b_2 & \cdots & b_{s-1} & b_s \\ & \hat{b}_1 & \hat{b}_2 & \cdots & \hat{b}_{s-1} & \hat{b}_s \end{array} \qquad \textbf{(1.13a)}$$

From this scheme two solutions are given according to (1.11.b):

$$\begin{aligned} y_{n+1} &= y_n + h_n \sum_{j=1}^{s} b_j k_j \\ \hat{y}_{n+1} &= y_n + h_n \sum_{j=1}^{s} \hat{b}_j k_j \end{aligned} \qquad \textbf{(1.13b)}$$

In modern codes the solution normally is evolved with the higher order formula so that $\hat{p} = p-1$ (so-called local extrapolation mode).

From the difference in (1.13b) we get a measure for the local error (in the current step):

$$e_n = h_n \sum_{j=1}^{s} \left(b_j - \hat{b}_j\right) k_j \quad \Rightarrow \quad \|e_n\| = \left\| h_n \sum_{j=1}^{s} \left(b_j - \hat{b}_j\right) k_j \right\| \qquad \textbf{(1.13c)}$$

The double bar indicates that a norm function is used. This yields a scalar measure and will be useful for the case of systems of differential equations[1] when $y = \vec{y}(x)$ is a vector solution function (then, of course, $k = \vec{k}$ and $e_n = \vec{e}_n$ are vectors, too).

[1] An m-dimensional system of differential equations generalizes (1.1) to
$\vec{y}'(x) = \vec{f}(x, \vec{y}(x)) = \left(f_1(x, \vec{y}(x)), f_1(x, \vec{y}(x)), \cdots, f_m(x, \vec{y}(x))\right)$ with initial vector condition $\vec{y}'(x_0) = \vec{y}_0$.

The classical step-size control uses the formula[1]:

$$h_{n+1} = h_n \left(\frac{\|e_n\|}{\varepsilon} \right)^{-\frac{1}{\tilde{p}}} = h_n \left(\frac{\varepsilon}{\|e_n\|} \right)^{\frac{1}{\tilde{p}}} \qquad \tilde{p} := \min\{p, \hat{p}\} + 1 \tag{1.13d}$$

The tolerance parameter ε commonly considers absolute and relative tolerances (accuracy and precision):

$$\varepsilon = \varepsilon_a + \varepsilon_r |y_n| \tag{1.13e}$$

Given accuracy and precision goals, ag and pg, respectively, the absolute and relative tolerances are computed as $\varepsilon_a = 10^{-ag}$ and $\varepsilon_r = 10^{-pg}$, respectively. Typical settings for these goals are 8, 16, 24, 32 a.s.o. Accuracy of a number is defined as the effective number of digits to the right of the decimal point. Precision of a number is defined as the effective number of digits[2].

In modern implementations non-classical extensions of the step-size formula (1.13d) are widely used. The next formula is known as scaled proportional integral step control.

$$h_{n+1} = s_1 h_n \left(\frac{s_2 \varepsilon}{\|e_n\|} \right)^{c_1 \frac{1}{\tilde{p}}} \left(\frac{\|e_{n-1}\|}{\|e_n\|} \right)^{c_2 \frac{1}{\tilde{p}}} \qquad \tilde{p} := \min\{p, \hat{p}\} + 1 \tag{1.13f}$$

---

[1] In the case of an m- dimensional system of differential equations the norm in (1.13d) generalizes to a scaled vector norm: $h_{n+1} = h_n \left(\left\| \left(\underbrace{\frac{e_n^{(1)}}{\varepsilon_a + \varepsilon_r |y_n^{(1)}|}}_{w_1}, \underbrace{\frac{e_n^{(2)}}{\varepsilon_a + \varepsilon_r |y_n^{(2)}|}}_{w_2}, \cdots, \underbrace{\frac{e_n^{(m)}}{\varepsilon_a + \varepsilon_r |y_n^{(m)}|}}_{w_m} \right) \right\|^{-\frac{1}{\tilde{p}}}$ with the super-scripts in round brackets indicating vector components. The double bar denotes a scaled q-norm ($q \geq 1$ with typical cases $q=1$ or $q=2$) or the infinity norm, i.e.

$$\|(w_1, w_2, ..., w_m)\|_q = \left(\frac{1}{m} \sum_{i=1}^{m} |w_i|^q \right)^{\frac{1}{q}} \quad \text{or} \quad \|(w_1, w_2, ..., w_m)\|_\infty = \max_{1 \leq i \leq m} |w_i| .$$

[2] With (absolute) uncertainty $dx = \varepsilon_a$ the accuracy of a number x is equal to $\log(\frac{1}{dx}) = -\log(dx)$ and thus the absolute uncertainty ε_a is equal to $10^{-accuracy(x)}$. The precision of a number $x \neq 0$ is equal to $\log(\frac{x}{dx}) = -\log(\frac{dx}{x})$ and thus the relative uncertainty $\frac{dx}{x} = \varepsilon_r$ is equal to $10^{-precision(x)}$.

A typical value for the precision is a 53-Bit representation: $precision(1.0) = \frac{53}{ld(10)} = 15.95458977...$ (machine precision).

The numbers c_1 and c_2 are step-size control parameters and s_1,s_2 are step-size safety parameters. For the first integration step it is taken that $\|e_n\| = \|e_{n-1}\|$.

Typical settings are $\{c_1, c_2\} = \{1, 0\}$ together with $\{s_1, s_2\} = \{0.85, 0.9\}$ or $\{c_1, c_2\} = \{0.3, 0.4\}$ together with $\{s_1, s_2\} = \{0.85, 0.9\}$.

Another additional feature is step-size ratio bounding: By two ratio bounds $\{r_1, r_2\}$ the ratio of step-sizes is bounded as $r_1 \le \left|\frac{h_{n+1}}{h_n}\right| \le r_2$. A typical setting is $\{0.125, 4\}$.

Example 1.8: Heun-Euler 2(1) method

The combined Butcher tableau for Heun-Euler is (cf. 1.9a):

$\begin{array}{c\|cc} 0 & & \\ 1 & 1 & \\ \hline & 1/2 & 1/2 \\ & 1 & 0 \end{array}$	Heun-method (single step) $y_0 = y(x_0)$ $y_{k+1} \approx y_k + \frac{1}{2}h_k\left(\underbrace{f(x_k, y_k)}_{\propto k_1} + \underbrace{f(x_k + h_k, y_k + h_k f(x_k, y_k))}_{\propto k_2}\right)$ Euler (single step) $\hat{y}_{k+1} = y_k + f(x_k, y_k)h_k$ where $x_{k+1} = x_k + h_k$ and step-size $h_k > 0$

So by (1.13c) we get a simple local error estimate in the form of $e_k = y_{k+1} - \hat{y}_{k+1}$ = $-\frac{1}{2}h_k k_1 + \frac{1}{2}h_k k_2 = \frac{1}{2}h_k(k_2 - k_1)$ and the classical step-size control formula (1.13d) equals $h_{k+1} = h_k \sqrt[2]{\left(\frac{\varepsilon}{|e_k|}\right)}$.

The most basic use of (1.13d) is as follows:

- If the scaled error norm expression $\left(\frac{\|e_n\|}{\varepsilon}\right)$ is > 1 then the step is rejected, i.e. repeated with unaltered values of x_k and y_k but with a new decreased step-size $h_{k+1} = h_k\left(\frac{\|e_k\|}{\varepsilon}\right)^{-\frac{1}{\tilde{p}}} < h_k$.
- If $\left(\frac{\|e_n\|}{\varepsilon}\right)$ is $\le$ 1 then $x_k \mapsto x_{k+1} = x_k + h_k$ and $y_k \mapsto y_{k+1}$. The next step is processed with $h_{k+1} = h_k\left(\frac{\|e_k\|}{\varepsilon}\right)^{-\frac{1}{\tilde{p}}} \ge h_k$.

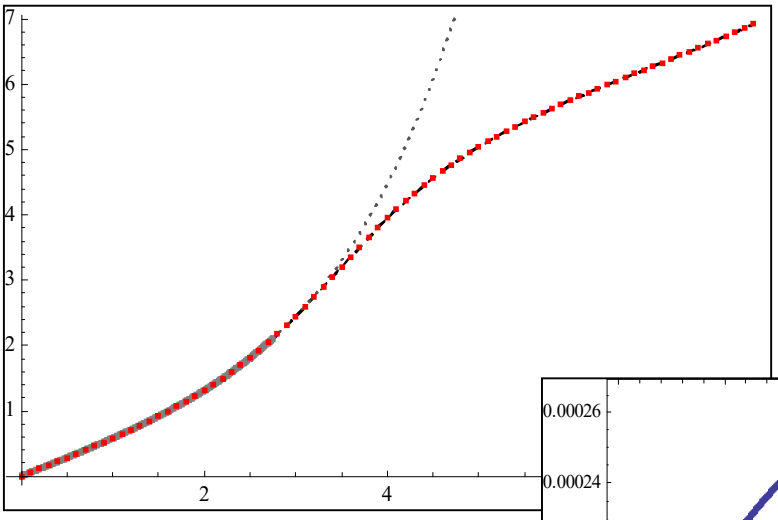


Figure 1.5a: Comparison of embedded adaptive Heun-Euler 2(1) method (gray) with non-adaptive Heun method (red points) with fixed step-size h = 0.1.

The embedded method has been run with 169'832 steps and step-size safety factors {1,1}, step-size control parameters {1,0} and step-size ratio bounds between 0 and ∞.

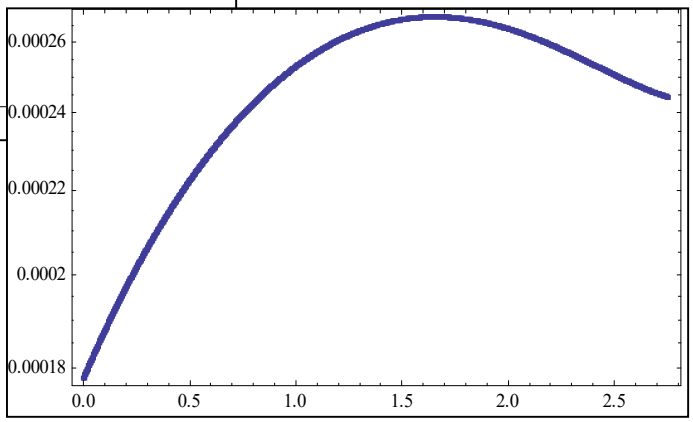


Figure 1.5b: Variable step-size plot…

Tables 1.2ab: Numerical comparison of the methods in Example 1.8. The adaptive embedded method Heun-Euler 2(1) is not always more accurate (cf. left table). Its costs are very high due to the large number (169832) of small steps. The accuracy goal is 4, the precision goal is 4, too.

t_k	Heun-Euler 2(1)	Heun	Reference
2.41244	1.71258	1.71262	1.71319
2.41269	1.71285	1.7129	1.71346
2.41294	1.71312	1.71317	1.71373
2.41319	1.71339	1.71344	1.714
2.41345	1.71367	1.71371	1.71427
2.4137	1.71394	1.71398	1.71454
2.41395	1.71421	1.71425	1.71482
2.41421	1.71448	1.71452	1.71509
2.41446	1.71475	1.71479	1.71536
2.41471	1.71502	1.71507	1.71563
2.41497	1.7153	1.71534	1.7159
2.41522	1.71557	1.71561	1.71618
2.41547	1.71584	1.71588	1.71645
2.41573	1.71611	1.71615	1.71672
2.41598	1.71638	1.71642	1.71699
2.41623	1.71665	1.7167	1.71726
2.41648	1.71692	1.71697	1.71753
2.41674	1.7172	1.71724	1.71781
2.41699	1.71747	1.71751	1.71808
2.41724	1.71774	1.71778	1.71835
2.4175	1.71801	1.71805	1.71862

t_k	Heun-Euler 2(1)	Heun	Reference
1.09801	0.645193	0.645298	0.645173
1.09827	0.645358	0.645462	0.645337
1.09853	0.645522	0.645627	0.645502
1.09878	0.645687	0.645791	0.645667
1.09904	0.645851	0.645956	0.645831
1.0993	0.646016	0.646121	0.645996
1.09956	0.646181	0.646285	0.646161
1.09981	0.646345	0.64645	0.646325
1.10007	0.64651	0.646615	0.64649
1.10033	0.646674	0.646779	0.646655
1.10058	0.646839	0.646944	0.646819
1.10084	0.647004	0.647109	0.646984
1.1011	0.647169	0.647273	0.647149
1.10135	0.647333	0.647438	0.647314
1.10161	0.647498	0.647603	0.647478
1.10187	0.647663	0.647768	0.647643
1.10213	0.647828	0.647932	0.647808
1.10238	0.647992	0.648097	0.647973
1.10264	0.648157	0.648262	0.648138
1.1029	0.648322	0.648427	0.648303
1.10315	0.648487	0.648592	0.648468

In the next subsections some widely used embedded pairs are presented.

1.1.4.1.1 Bogacki-Shampine-Ralston 3(2) [1]

The Bogacki-Shampine-Ralston methods are implemented in Matlab as `ode23`, TI-85 and RKSuite. Because of the relatively low orders they are not the first choice for high accuracy or precision. Their prime advantage is speed.

The combined Butcher tableaus (cf. 1.13a) are:

0				
1/2	1/2			
3/4	0	3/4		
1	2/9	1/3	4/9	
	2/9	1/3	4/9	0
	7/24	1/4	1/3	1/8

0				
1/2	1/2			
3/4	0	3/4		
1	2/9	1/3	4/9	
	2/9	1/3	4/9	0
	-5/72	1/12	1/9	-1/8

The first row of b-numbers is of order 3 (due to Ralston) and the second (embedded) method is of order 2.

Obviously it is a 4-stage FSAL-scheme (cf. Example 1.7) and uses only three function evaluations per step.

1.1.4.1.2 Sofroniou-Spaletta 3(2) [2]

The Sofroniou-Spaletta-method is implemented in *Mathematica* (`NDSolve`). By the relatively low orders it is not the first choice for high accuracy or precision. Its prime advantage is speed.

The combined Butcher tableau (cf. 1.13a) is:

0				
$\frac{1}{2}$	$\frac{1}{2}$			
1	-1	2		
1	$\frac{1}{6}$	$\frac{2}{3}$	$\frac{1}{6}$	
	$\frac{1}{6}$	$\frac{2}{3}$	$\frac{1}{6}$	0
	$\frac{1}{72}\left(\sqrt{82}-10\right)$	$\frac{1}{36}\left(10-\sqrt{82}\right)$	$\frac{1}{144}\left(28-\sqrt{82}\right)$	$\frac{1}{48}\left(\sqrt{82}-16\right)$

The first row of b-numbers is of order 3 and the second (embedded) method is of order 2.

---

[1] Bogacki, Przemyslaw; Shampine, Lawrence F. (1989), "A 3(2) pair of Runge–Kutta formulas", *Applied Mathematics Letters* **2** (4): 321–325

Shampine, Lawrence F.; Reichelt, Mark W. (1997), "The Matlab ODE Suite", *SIAM Journal on Scientific Computing* **18** (1): 1–22

[2] Sofroniou, M. and G. Spaletta. "Construction of Explicit Runge–Kutta Pairs with Stiffness Detection." *Mathematical and Computer Modelling*, special issue on The Numerical Analysis of Ordinary Differential Equations, 40, no. 11–12 (2004): 1157–1169

Obviously it is a 4-stage FSAL-scheme (cf. Example 1.7) and uses only three function evaluations per step. It has the additional property that $c_{s-1} = c_s = 1$. For this the method is suitable for stiffness detection.

Example 1.9: Comparison of 3(2) methods

The accuracy and precision goal is set to 4.

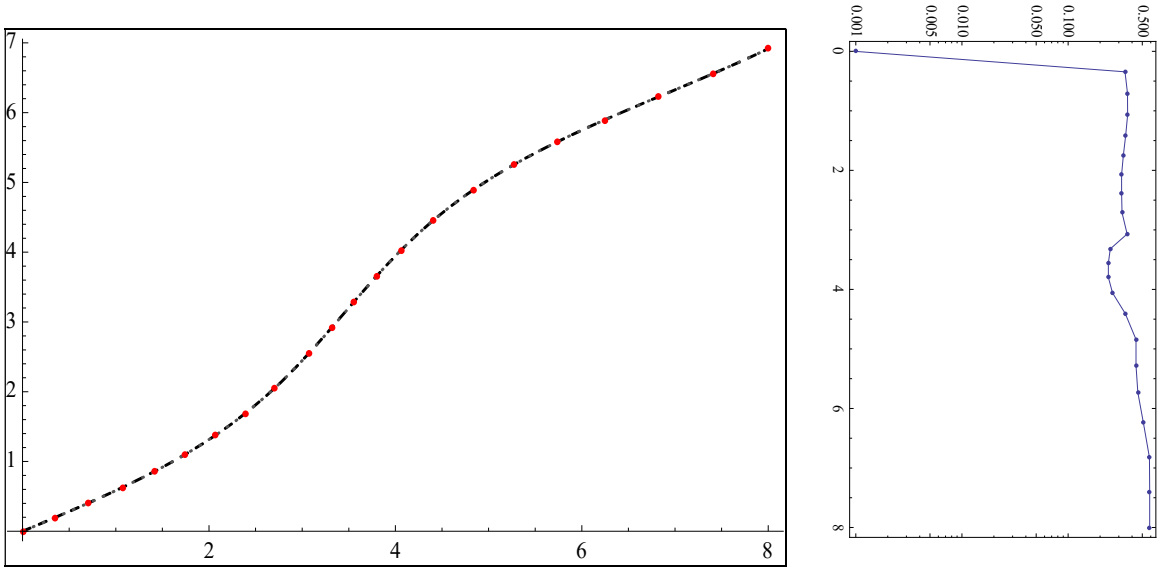


Figure 1.6a: Bogacki-Shampine-Ralston 3(2) with step-size log-plot (rotated)

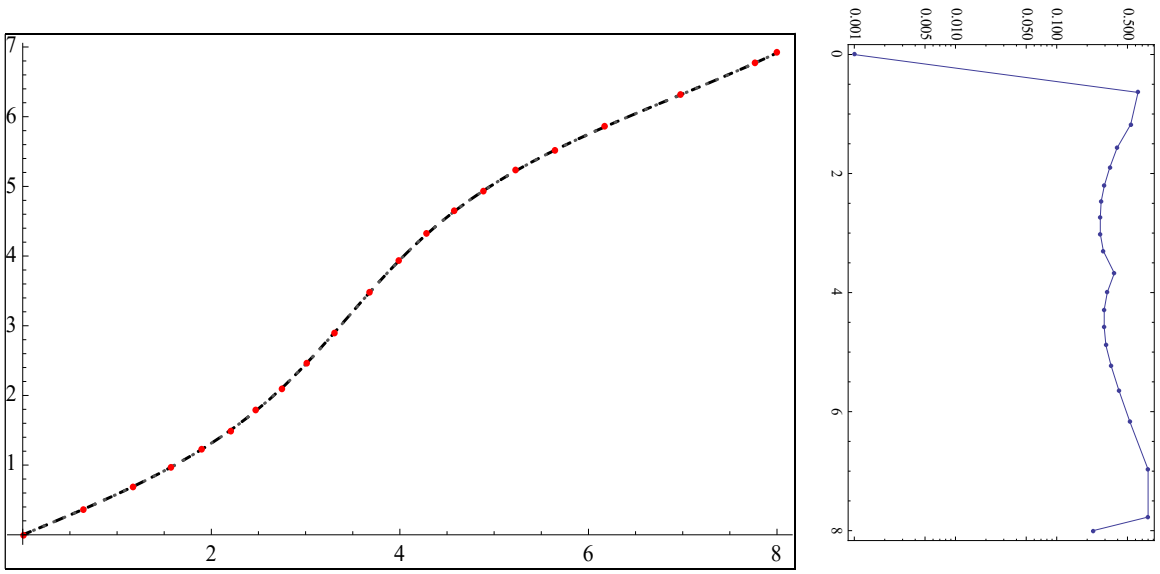


Figure 1.6b: Sofroniou-Spaletta 3(2) with step-size log-plot (rotated)

t_k	Bogacki-Shampine-Ralston 3(2)	Reference
0.	1.51937×10^{-20}	0.
0.001	0.0005625	0.0005625
0.344349	0.194508	0.194508
0.706688	0.40464	0.404633
1.0701	0.627373	0.627351
1.41839	0.859346	0.859248
1.74965	1.10501	1.10484
2.06869	1.37407	1.37413
2.38772	1.68616	1.68683
2.70969	2.05661	2.05699
3.06934	2.544	2.54567
3.31883	2.92013	2.92203
3.55802	3.29291	3.29325
3.79721	3.65761	3.65674
4.05906	4.02811	4.02601
4.40418	4.45458	4.45466
4.84056	4.89767	4.89765
5.27694	5.25916	5.25908
5.73068	5.57777	5.57776
6.24242	5.89481	5.89472
6.826	6.22937	6.22924
7.413	6.56194	6.56181
8.	6.9158	6.91568

Table 1.3a: 22 steps with adaptive step-size

t_k	Sofroniou-Spaletta 3(2)	Reference
0.	9.95264×10^{-21}	0.
0.001	0.0005625	0.0005625
0.63687	0.363458	0.363436
1.17523	0.6952	0.695109
1.57025	0.968519	0.968182
1.90386	1.23061	1.23039
2.19969	1.49656	1.4966
2.47425	1.78007	1.78054
2.74397	2.10001	2.10001
3.01369	2.46396	2.46499
3.30319	2.89604	2.89794
3.67422	3.47228	3.47235
3.99103	3.93565	3.93338
4.28567	4.3165	4.31573
4.5803	4.64531	4.64521
4.88659	4.93915	4.9391
5.23133	5.22427	5.22422
5.64254	5.51918	5.51934
6.17127	5.85234	5.85245
6.97127	6.31066	6.31098
7.77127	6.77322	6.77344
8.	6.91541	6.91568

Table 1.3b: 21 steps with adaptive step-size

Method	Steps	Costs	Error
BSR32	{22, 35}	173	0.0000177355
SS32	{21, 36}	173	0.0000390222

Method	Steps	Costs	Error
BSR32	{426, 338}	2294	2.02487×10^{-9}
SS32	{374, 337}	2135	5.30919×10^{-9}

Method	Steps	Costs	Error
BSR32	{195814, 155801}	1054847	1.2843×10^{-16}
SS32	{171083, 149301}	961154	1.2843×10^{-16}

Tables 1.4abc: Comparisons for accuracy and precision goals 4,8 and 16. The second number in the step-column is the number of rejected steps. The costs-column indicates the number of function evaluations and the error-column indicates the final (global) minimum of the relative and absolute error.

1.1.4.1.3 Bogacki-Shampine 5(4) [1]

The Bogacki-Shampine 5(4)-method is fast and relatively accurate. It is the default explicit Runge-Kutta-method of order 5 with embedded order 4 in *Mathematica* (`NDSolve`) . The combined Butcher tableau (cf. 1.13a) is given below.

The first row of b-numbers is of order 5 and the second (embedded) method is of order 4.

This is a 8-stage FSAL-scheme which has the additional property that $c_{s-1} = c_s = 1$. For this the method is suitable for stiffness detection.

[1] Bogacki, P. and L. F. Shampine. "An Efficient Runge–Kutta (4, 5) Pair." *Report 89–20, Math. Dept. Southern Methodist University*, Dallas, Texas, 1989.

0								
$\frac{1}{6}$	$\frac{1}{6}$							
$\frac{2}{9}$	$\frac{2}{27}$	$\frac{4}{27}$						
$\frac{3}{7}$	$\frac{183}{1372}$	$-\frac{162}{343}$	$\frac{1053}{1372}$					
$\frac{2}{3}$	$\frac{68}{297}$	$-\frac{4}{11}$	$\frac{42}{143}$	$\frac{1960}{3861}$				
$\frac{3}{4}$	$\frac{597}{22528}$	$\frac{81}{352}$	$\frac{63099}{585728}$	$\frac{58653}{366080}$	$\frac{4617}{20480}$			
1	$\frac{174197}{959244}$	$-\frac{30942}{79937}$	$\frac{8152137}{19744439}$	$\frac{666106}{1039181}$	$-\frac{29421}{29068}$	$\frac{482048}{414219}$		
1	$\frac{587}{8064}$	0	$\frac{4440339}{15491840}$	$\frac{24353}{124800}$	$\frac{387}{44800}$	$\frac{2152}{5985}$	$\frac{7267}{94080}$	
	$\frac{587}{8064}$	0	$\frac{4440339}{15491840}$	$\frac{24353}{124800}$	$\frac{387}{44800}$	$\frac{2152}{5985}$	$\frac{7267}{94080}$	0
	$\frac{3817}{1959552}$	0	$-\frac{140181}{15491840}$	$\frac{4224731}{272937600}$	$-\frac{8557}{403200}$	$\frac{57928}{4363065}$	$\frac{23930231}{4366535040}$	$-\frac{3293}{556956}$

1.1.4.1.4 Dormand-Prince 5(4) [1]

The Dormand-Prince 5(4)-method is fast and relatively accurate, too. It is the default explicit Runge-Kutta-method of order 5 with embedded order 4 in Matlab (`ode45`) . The combined Butcher tableau (cf. 1.13a) is:

0							
$\frac{1}{5}$	$\frac{1}{5}$						
$\frac{3}{10}$	$\frac{3}{40}$	$\frac{9}{40}$					
$\frac{4}{5}$	$\frac{44}{45}$	$-\frac{56}{15}$	$\frac{32}{9}$				
$\frac{8}{9}$	$\frac{19372}{6561}$	$-\frac{25360}{2187}$	$\frac{64448}{6561}$	$-\frac{212}{729}$			
1	$\frac{9017}{3168}$	$-\frac{355}{33}$	$\frac{46732}{5247}$	$\frac{49}{176}$	$-\frac{5103}{18656}$		
1	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	
	$\frac{35}{384}$	0	$\frac{500}{1113}$	$\frac{125}{192}$	$-\frac{2187}{6784}$	$\frac{11}{84}$	0
	$\frac{71}{57600}$	0	$-\frac{71}{16695}$	$\frac{71}{1920}$	$-\frac{17253}{339200}$	$\frac{22}{525}$	$-\frac{1}{40}$

The first row of b-numbers is of order 5 and the second (embedded) method is of order 4.

As before, this is a FSAL-scheme (7 stages) which has the additional property that $c_{s-1} = c_s = 1$. So again, this is suitable for stiffness detection.

[1] Dormand, J. R. and P. J. Prince. "A Family of Embedded Runge–Kutta Formulae." *J. Comp. Appl. Math.* 6 (1980): 19–26

Example 1.10: Comparison of 5(4) methods (higher accuracy and precision)

As in the previous example (1.9) the outputs of different 5(4)-methods are compared with respect to costs, step-sizes and error. But the accuracy and precision goals now are set to 8.

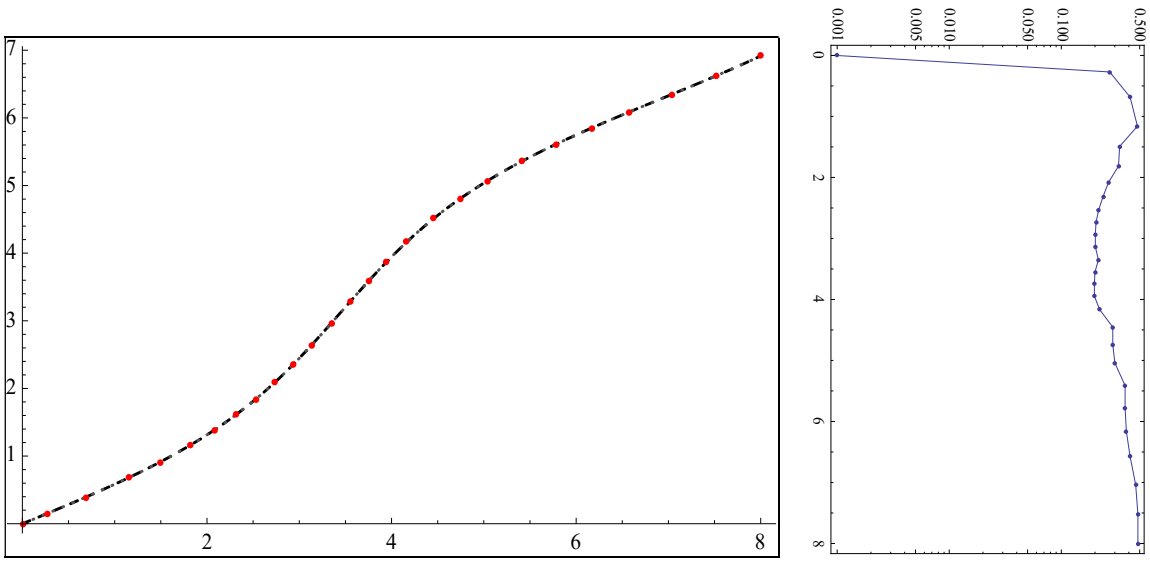


Figure 1.7a: Dormand-Prince 5(4) with step-size log-plot (rotated)

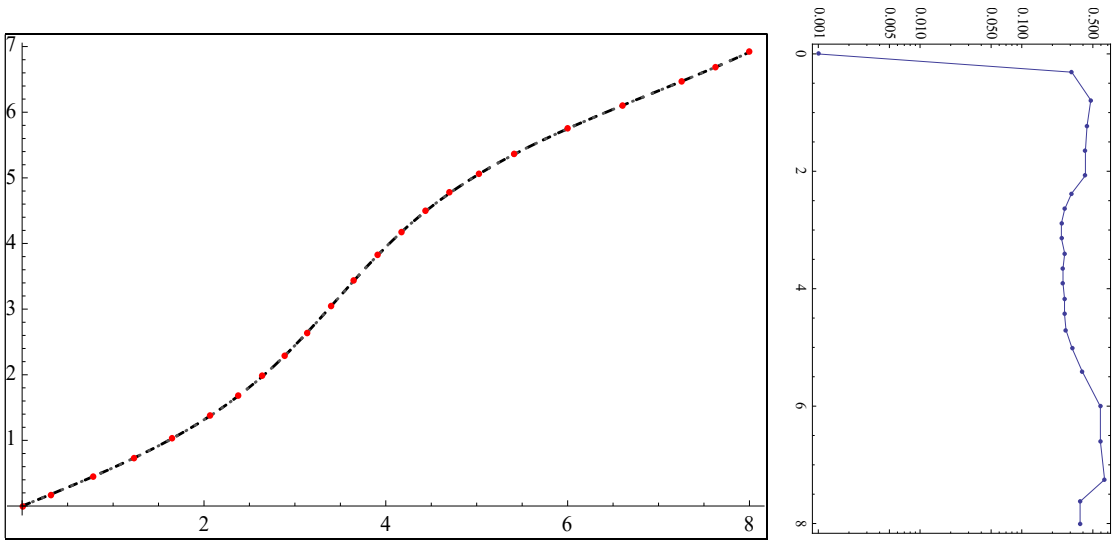


Figure 1.7b: Bogacki-Shampine 5(4) with step-size log-plot (rotated)

t_k	Dormand-Prince 5(4)	Reference
0	$1.259961509 \times 10^{-20}$	0
0.001000000000	0.0005625000198	0.0005625000198
0.2712656148	0.1529828749	0.1529828487
0.6820045761	0.3900256332	0.3900219380
1.159847445	0.6850937167	0.6850856338
1.491310525	0.9110238579	0.9108547784
1.815182659	1.157357470	1.157212157
2.078859641	1.383346335	1.383363876
2.314459499	1.610141936	1.610596021
2.529996757	1.842478439	1.843092763
2.734019604	2.087280019	2.087446743
2.935327347	2.353735029	2.354174109
3.136635090	2.643214077	2.645109856
3.350964798	2.970106834	2.971675528
3.550530365	3.281409084	3.281629854
3.747917985	3.584184644	3.583604660
3.945305606	3.871913037	3.869764701
4.165066261	4.166998028	4.165279893
4.452788705	4.509003023	4.509001927
4.740511149	4.804371522	4.804390752
5.041167423	5.072084236	5.072106524
5.412131596	5.358912969	5.359050326
5.783095770	5.611763854	5.611860598
6.163982629	5.848057139	5.848088217
6.574808839	6.087136848	6.087154495
7.035753712	6.347278421	6.347273302
7.517876856	6.622693833	6.622673953
8.000000000	6.915679697	6.915679756

Table 1.5a: 27 steps with adaptive step-size

t_k	Bogacki-Shampine 5(4)	Reference
0	$1.259961509 \times 10^{-20}$	0
0.001000000000	0.0005625000198	0.0005625000198
0.3114536671	0.1757925955	0.1757925937
0.7885025102	0.4534865990	0.4534639697
1.226695364	0.7289452829	0.7289451212
1.649281086	1.027455611	1.027229834
2.071866808	1.377000287	1.377012682
2.380097951	1.678203236	1.678770652
2.644418397	1.976658679	1.977068109
2.889618660	2.291093426	2.291239768
3.134818923	2.640513443	2.642401759
3.399036312	3.044942213	3.046183650
3.653303577	3.440389824	3.440372009
3.907570841	3.818422680	3.816478442
4.170976584	4.174529993	4.172856311
4.434382328	4.488594312	4.488594112
4.706739431	4.771865773	4.771877870
5.019819319	5.054246549	5.054273413
5.414460706	5.360592745	5.360731538
6.006986598	5.752876048	5.752877709
6.599512491	6.101198251	6.101211188
7.251453714	6.469291768	6.469289562
7.625726857	6.686151414	6.686116372
8.000000000	6.915679743	6.915679756

Table 1.5b: 23 steps with adaptive step-size

Method	Steps	Costs	Error
DP5432	{27, 23}	302	8.51259×10^{-9}
BS54	{23, 31}	380	1.9442×10^{-9}

Method	Steps	Costs	Error
DP5432	{930, 842}	10 634	1.2843×10^{-16}
BS54	{775, 624}	9795	2.56859×10^{-16}

Tables 1.6ab: Comparisons for accuracy and precision goals 8 and 16. The second number in the step-column is the number of rejected steps. The costs-column indicates the number of function evaluations and the error-column indicates the final (global) minimum of the relative and absolute error.