Chapter 3

Ordinary Differential Equations

"Chemistry is, well technically, chemistry is the study of matter. But I prefer to see it as the study of change."

— Walter H. White, *Breaking bad*

In this chapter we deal with the numerical treatment of Ordinary Differential Equations (ODEs). ODEs describe mathematically the change of one or several state variables. Such equations arise commonly in the study of chemical kinetics or process dynamics (e.g. batch reactor kinetics). However, in the fewest instances of practical interest analytical solutions are known and this motivates the present chapter.

3.1 Problem statement and examples

Consider the *scalar first order Initial Value Problem* (IVP)

$$\begin{cases} y'(x) = f(x, y(x)) & \text{scalar first order ODE} \\ y(x_0) = y_0 & \text{initial value.} \end{cases} \tag{3.1}$$

The goal is to find the solution $y(x)$, function of the independent variable x, satisfying the differential equation and the initial value from the starting point x_0 to some end point $\bar{x}$. We call $[x_0, \bar{x}]$ the *integration interval*. The function $f(x, y(x))$, commonly referred to as the *right-hand side of the ODE*, depends on the independent variable x and the solution $y(x)$ itself.

It is important to note that here we are not just looking for a number solving an equation (like in the previous chapter), but rather for an entire function! Moreover, Eq. (3.1) is called first order ODE because the highest derivative of $y(x)$ appearing in the equation is the first. Formally we can write down the solution as a so-called integral equation

$$y(x) = y_0 + \int_{x_0}^{x} f(s, y(s))\mathrm{d}s \quad , \quad x_0 \le x \le \bar{x}. \tag{3.2}$$

This may already suggest a way to obtain approximate solutions: numerical quadrature! But again note that the function to integrate f depends on the solution $y(x)$ itself.

Here and in the following we denote the independent variable generically by x, although, in many cases x is the time (i.e. t). However, x may also be some displacement or a (radial) coordinate, etc.

Let's begin with a few (probably) familiar examples.

Example 3.1. A very simple example is the following ODE

$$y'(x) = y(x) \tag{3.3}$$

with initial value $y(x_0) = y_0$. What the differential equation actually tells us is that at every point in the x-y plane the slope of the solution $y(x)$ is given by the value of $y(x)$ itself, e.g. the slope of $y(x)$ at the point $(x, y) = (1, 2)$ is simply 2. This can be visualized with the so-called *slope field*, which is shown for Eq. (3.3) in the left panel of Figure 3.1. The slope field shows the general trend of the solutions of the ODE. A particular solution is then found by choosing an initial value and simply following the slope field (see the blue and green lines in Figure 3.1). Actually, many numerical methods we shall see have a straightforward graphical representation in the slope field. ▲

Example 3.2. As a second example consider the ODE

$$y'(x) = ay(x) - by^2(x) = (a - by(x))y(x) \tag{3.4}$$

with some initial value $y(x_0) = y_0$. The slope field of this example is shown in the right panel of Figure 3.1 with $a = 2$ and $b = 1$. You can see that something special happens for $y = 0$ and $y = a/b$: for these values the slope vanishes, i.e. the solution remains constant. These points are so-called *stationary points*. At these points the derivative of the solution is zero and such steady states are quite common in practice. ▲

Generally, one has to deal with *systems of first order ODEs*

$$\begin{aligned} y_1'(x) &= \quad f_1(x, y_1(x), ..., y_n(x)) \\ &\vdots \\ y_n'(x) &= \quad f_n(x, y_1(x), ..., y_n(x)), \end{aligned}$$

where $y_i(x)$, $i = 1, ..., n$, are functions of the independent variable $x \in [x_0, \bar{x}]$. To state an IVP, we need to specify an integration interval $[x_0, \bar{x}]$ and n initial values

$$\begin{aligned} y_1(x_0) &= \quad y_{1,0} \\ &\vdots \\ y_n(x_0) &= \quad y_{n,0}. \end{aligned} \tag{3.5}$$

The double index notation is rather unaesthetic, but will be avoided with the following definition. Let's define

$$\mathbf{y}(x) = \begin{pmatrix} y_1(x) \\ \vdots \\ y_n(x) \end{pmatrix}, \quad \mathbf{F}(x, \mathbf{y}(x)) = \begin{pmatrix} f_1(x, y_1(x), ..., y_n(x)) \\ \vdots \\ f_n(x, y_1(x), ..., y_n(x)) \end{pmatrix} \text{ and } \mathbf{y}_0 = \begin{pmatrix} y_{1,0} \\ \vdots \\ y_{n,0} \end{pmatrix}, \tag{3.6}$$

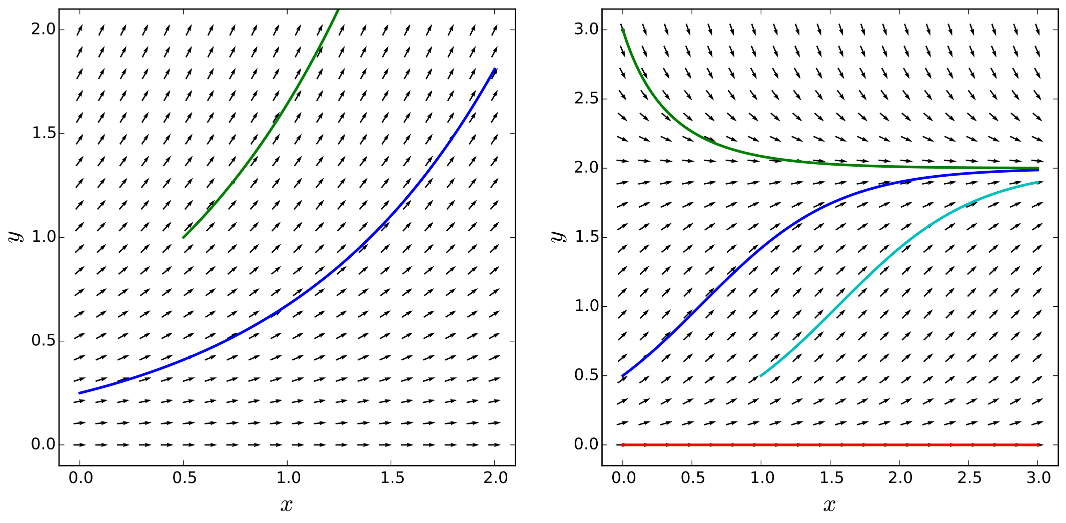

Figure 3.1: ODE slope field and solutions to a few IVPs: left Eq. (3.3) and right Eq. (3.4).

which give us the opportunity to rewrite the first order system IVP in the compact form

$$\begin{cases} \mathbf{y}'(x) = \mathbf{f}(x, \mathbf{y}(x)) & \text{system of first order ODEs} \\ \mathbf{y}(x_0) = \mathbf{y}_0 & \text{initial values.} \end{cases} \tag{3.7}$$

Let's start with a probably familiar example of batch reactor kinetics.

Example 3.3. We consider the batch reactor kinetics for the network of two elementary reactions

$$\begin{aligned} A + B &\to C & r_{R1} &= k_1 c_A c_B \\ C + B &\to D & r_{R1} &= k_2 c_C c_B, \end{aligned} \tag{3.8}$$

where the c_j, $j = A, B, C, D$, are the respective chemical species concentrations, and k_1 and k_2 the respective rate constants. The time evolution of the concentrations are then given by the following system of four ODEs

$$\begin{aligned} \frac{\mathrm{d}c_A}{\mathrm{d}t} &= -k_1 c_A c_B \\ \frac{\mathrm{d}c_B}{\mathrm{d}t} &= -k_1 c_A c_B - k_2 c_C c_B \\ \frac{\mathrm{d}c_C}{\mathrm{d}t} &= +k_1 c_A c_B - k_2 c_C c_B \\ \frac{\mathrm{d}c_D}{\mathrm{d}t} &= +k_2 c_C c_B, \end{aligned} \tag{3.9}$$

with some initial concentrations $c_{j,0}$, $j = A, B, C, D$, respectively. Note that here the independent variable is the time t. By defining

$$\mathbf{y}(t) = \begin{pmatrix} c_A(t) \\ c_B(t) \\ c_C(t) \\ c_D(t) \end{pmatrix}, \ \mathbf{F}(t, \mathbf{y}(t)) = \begin{pmatrix} -k_1 c_A c_B \\ -k_1 c_A c_B - k_2 c_C c_B \\ +k_1 c_A c_B - k_2 c_C c_B \\ +k_2 c_C c_B \end{pmatrix} \text{ and } \mathbf{y}_0 = \begin{pmatrix} c_{A,0} \\ c_{B,0} \\ c_{C,0} \\ c_{D,0} \end{pmatrix}, \tag{3.10}$$

we can write this IVP in form of Eq. (3.7) (just with the independent variable t instead of x). ▲

Example 3.4. As another example consider the linear system of ODEs

$$\mathbf{y}'(x) = A\mathbf{y}(x), \tag{3.11}$$

where $A \in \mathbb{R}^{n\times n}$ is a real square matrix and the initial value $\mathbf{y}(x_0) = \mathbf{y}_0 \in \mathbb{R}^n$ a real vector. The solution to this IVP can be formally written as

$$\mathbf{y}(x) = e^{Ax}\mathbf{y}_0, \tag{3.12}$$

where we introduced the matrix exponential

$$e^A = \sum_{k=0}^{\infty} \frac{A^k}{k!}. \tag{3.13}$$

Let's verify that this is indeed the solution to the IVP:

$$\begin{aligned}
y'(x) &= \frac{\mathrm{d}}{\mathrm{d}x}\left(\sum_{k=0}^{\infty} \frac{A^k x^k}{k!}\mathbf{y}_0\right) \\
&= \frac{\mathrm{d}}{\mathrm{d}x}\left(I + Ax + \frac{1}{2}A^2x^2 + \frac{1}{6}A^3x^3 + ...\right)\mathbf{y}_0 \\
&= \left(A + A^2x + \frac{1}{2}A^3x^2 + ...\right)\mathbf{y}_0 \\
&= A\left(I + Ax + \frac{1}{2}A^2x^2 + \frac{1}{6}A^3x^3 + ...\right)\mathbf{y}_0 \\
&= A\left(\sum_{k=0}^{\infty} \frac{A^k x^k}{k!}\right)\mathbf{y}_0 \\
&= A\mathbf{y}(x).
\end{aligned}$$

▲

So far we have seen only first order ODEs. A n-th order scalar ODE is given by

$$y^{(n)}(x) = f(x, y(x), y'(x), ..., y^{(n-1)}(x)), \tag{3.14}$$

where f is a (in general) nonlinear function of the independent variable x, the solution $y(x)$ and it's derivatives $y'(x)$, $y''(x)$, ..., $y^{(n-1)}(x)$. To obtain a complete IVP, we have to specify n initial values

$$y(x_0) = y_0, \quad y'(x_0) = y_0', \quad ..., \quad y^{(n-1)}(x_0) = y_0^{(n-1)}, \tag{3.15}$$

that specify the value of the solution $y(x)$ and it's $(n-1)$ first derivatives at x_0.

Example 3.5. A classical example of a second order ODE is given by the equation of motion in physics

$$m\ddot{\mathbf{x}} = \mathbf{F}, \tag{3.16}$$

i.e. Newton's second law, linking the acceleration of a body $\ddot{\mathbf{x}}$ to the net force $\mathbf{F}$ acting on it. Here each overdot represents one time derivative. To form a complete IVP, one has to specify the initial position $\mathbf{x}(t_0) = \mathbf{x}_0$ and velocity $\dot{\mathbf{x}}(t_0) = \mathbf{v}_0$ of the body, where t_0 is the initial time. ▲

All the numerical methods we shall see in this chapter are for first order ODEs. The reason for this is that a n-th order ODE can simply be rewritten as a system of n first order ODEs by doing the following relabeling

$$\begin{aligned}
y_0(x) &= y(x) && (3.17)\\
y_0'(x) &= y_1(x) && (3.18)\\
y_1'(x) &= y_2(x) && (3.19)\\
&\vdots && (3.20)\\
y_{n-2}'(x) &= y_{n-1}(x) && (3.21)\\
y_{n-1}'(x) &= f(x, y_0, y_1, ..., y_{n-1}). && (3.22)
\end{aligned}$$

So we end up with a system of first order ODEs

$$\mathbf{y}'(x) = \mathbf{f}(x, \mathbf{y}), \tag{3.23}$$

where

$$\mathbf{y}(x) = \begin{pmatrix} y_0(x) \\ \vdots \\ y_{n-2}(x) \\ y_{n-1}(x) \end{pmatrix} \quad \text{and} \quad \mathbf{f}(x, \mathbf{y}) = \begin{pmatrix} y_1(x) \\ \vdots \\ y_{n-1}(x) \\ f(x, y_0, ..., y_{n-1}) \end{pmatrix}. \tag{3.24}$$

Please note the notational difference between the right-hand side of the n-th order ODE f and the one from the equivalent first-order system $\mathbf{f}$!

Let's illustrate this order reduction with a couple of examples.

Example 3.6. Consider the second order ODE

$$y'' = f(x, y, y'), \quad x \in [x_0, \bar{x}]$$

for some function f and with initial values

$$y(x_0) = 1, \; y'(x_0) = 2.$$

By relabeling as

$$y_0(x) = y(x), \; y_1(x) = y'(x),$$

we obtain the equivalent first order system Eq. (3.23) with

$$\mathbf{y}(x) = \begin{pmatrix} y_0(x) \\ y_1(x) \end{pmatrix}, \; \mathbf{f}(x, \mathbf{y}) = \begin{pmatrix} y_1(x) \\ f(x, y_0(x), y_1(x)) \end{pmatrix}$$

and initial values

$$\mathbf{y}(x_0) = \begin{pmatrix} 1 \\ 2 \end{pmatrix}.$$

▲

Example 3.7. Consider the third order ODE

$$y''' = -2y'' + y' + y^2 - e^x, \quad x \in [x_0, \bar{x}]$$

with initial values

$$y(x_0) = 1,\ y'(x_0) = 0,\ y''(x_0) = 0.$$

By relabeling as

$$y_0(x) = y(x),\ y_1(x) = y'(x),\ y_2(x) = y''(x),$$

we obtain the equivalent first order system Eq. (3.23) with

$$\mathbf{y}(x) = \begin{pmatrix} y_0(x) \\ y_1(x) \\ y_2(x) \end{pmatrix},\ \mathbf{f}(x, \mathbf{y}) = \begin{pmatrix} y_1(x) \\ y_2(x) \\ -2y_2 + y_1 + y_0^2 - e^x \end{pmatrix}$$

and initial values

$$\mathbf{y}(x_0) = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}.$$

▲

Solving IVP analytically is generally difficult or even impossible. This motivates the use of numerical methods.

3.2 The Euler methods

In the following we consider the scalar first order IVP

$$\begin{cases} y'(x) = f(x, y(x)) & \text{scalar first order ODE} \\ y(x_0) = y_0 & \text{initial value.} \end{cases} \tag{3.25}$$

The goal is to find an approximation to the solution $y(x)$ for $x_0 \le x \le \bar{x}$.

The idea is to discretize the integration interval $[x_0, \bar{x}]$ in N equal subintervals

$$x_j = x_0 + jh,\ j = 0, 1, ..., N, \tag{3.26}$$

where the so-called *step size* is

$$h = \frac{\bar{x} - x_0}{N}, \tag{3.27}$$

and approximate the solution by "following" the slope field. Since $f(x_0, y_0)$ gives the slope at x_0, we follow this direction until x_1, i.e. we connect the line segment

$$\frac{y_1 - y_0}{x_1 - x_0} = f(x_0, y_0), \tag{3.28}$$

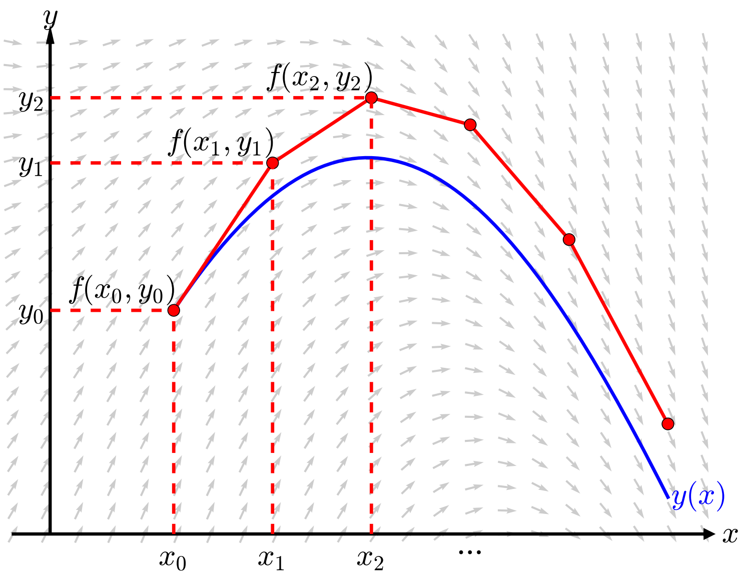

Figure 3.2: Illustration of the explicit Euler method.

which immediately gives the *explicit* expression

$$y_1 = y_0 + \underbrace{(x_1 - x_0)}_{h} f(x_0, y_0) = y_0 + hf(x_0, y_0). \tag{3.29}$$

This procedure is then repeated from x_1 to x_2, etc., and we obtain the so-called *Euler method*

$$y_{j+1} = y_j + hf(x_j, y_j)\ ,\ j = 0, 1, ..., N-1. \tag{3.30}$$

This is illustrated in Figure 3.2. The Euler method is commonly also referred to by *explicit Euler* and *Euler forward* method.

A different method is obtained by tracing a line segment backward from x_1

$$\frac{y_1 - y_0}{x_1 - x_0} = f(x_1, y_1), \tag{3.31}$$

which immediately gives the *implicit*[1] expression

$$y_1 = y_0 + \underbrace{(x_1 - x_0)}_{h} f(x_1, y_1) = y_0 + hf(x_1, y_1). \tag{3.32}$$

The repetition of this procedure gives the so-called *implicit Euler method*

$$y_{j+1} = y_j + hf(x_{j+1}, y_{j+1})\ ,\ j = 0, 1, ..., N-1. \tag{3.33}$$

This is illustrated in Figure 3.3 and is also known as the *Euler backward method* in the literature.

We stress that to obtain y_{j+1} you have to solve the (possibly nonlinear) Eq. (3.33). In order to do this, you could apply the methods we have seen in Chapter 2!

[1]y_1 appears on both sides of Eq. (3.32) and is therefore only implicitly defined!

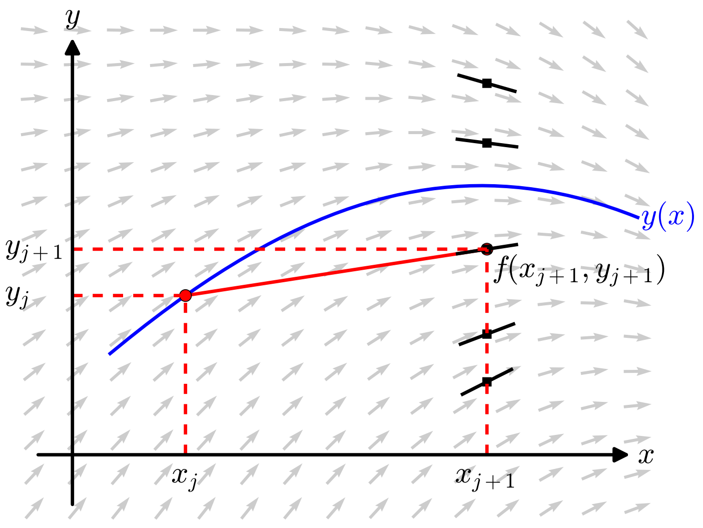

Figure 3.3: Illustration of the implicit Euler method. Also shown are a few trial values for y_{j+1} at which the slopes do not trace back to y_j.

Example 3.8. Let's illustrate the Euler methods by the following IVP

$$\begin{cases} y'(x) = -y(x) + 2\cos(x) \\ y(0) = 1 \end{cases}$$

for $x \in [0,4]$. One easily verifies that the exact solution is given by

$$y(x) = \sin(x) + \cos(x).$$

By applying the Euler methods we get:

- explicit Euler method (EE):

$$\begin{aligned} y_{j+1} &= y_j + hf(x_j, y_j) \\ &= y_j + h(-y_j + 2\cos(x_j)) \\ &= (1-h)y_j + 2h\cos(x_j) \end{aligned}$$

- implicit Euler method (IE):

$$\begin{aligned} y_{j+1} &= y_j + hf(x_{j+1}, y_{j+1}) \\ &= y_j + h(-y_{j+1} + 2\cos(x_{j+1})) \end{aligned}$$

  The latter equation has to be solved for y_{j+1}, which in this simple (linear) example gives

$$y_{j+1} = \frac{1}{1+h}\left(y_j + 2h\cos(x_{j+1})\right).$$

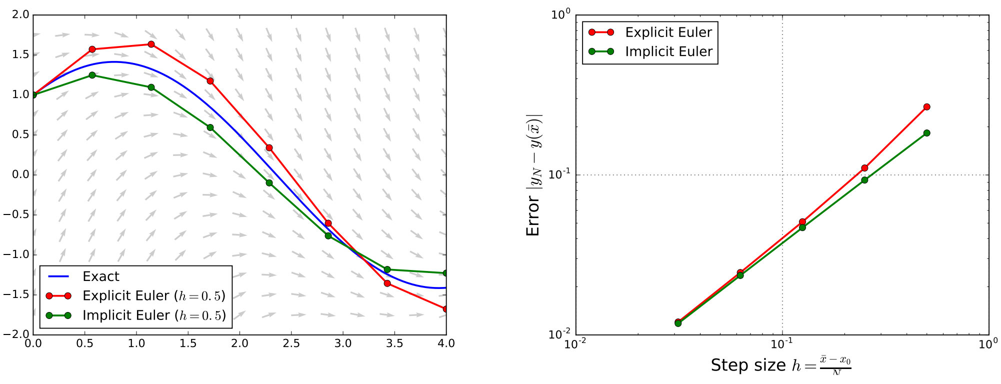

Figure 3.4: Euler methods applied to the example IVP.

N	h	EE $\lvert y_N - y(\bar{x})\rvert$	IE $\lvert y_N - y(\bar{x})\rvert$
8	2^{-1}	2.666E-01	1.830E-01
16	2^{-2}	1.105E-01	9.295E-02
32	2^{-3}	5.097E-02	4.688E-02
64	2^{-4}	2.453E-02	2.354E-02
128	2^{-5}	1.204E-02	1.180E-02

Table 3.1: Errors for the explicit and implicit Euler methods with decreasing step size.

In the left panel of Figure 3.4 is shown the result of the above methods (with $h = 1/2$, i.e. $N = 8$) together with the exact solution. Obviously, the Euler methods are only an approximation of the exact solution! ▲

Let's now apply the above methods for various step sizes h and compare the obtained results with the exact solution, that is we study the influence of h (or N) on the accuracy of the approximation. To this end we compute the absolute difference between the approximate solution at the last step and the exact solution $|y_N - y(\bar{x})|$, the so-called (absolute) error. The result is shown in Table 3.1. From the table we see that the absolute error decreases with smaller step sizes (or accordingly larger number of subintervals N). A result which you have surely anticipated! The table tells us even more: the error decreases proportionally to the step size h (or inversely proportional to the number of subintervals N). This is further illustrated graphically in the right panel of Figure 3.4 (note the double logarithmic axes!).

The Euler methods can easily be applied to systems of first order ODE. Just replace the scalar y_j's and $f(x_j, y_j)$ by corresponding vectors: $\mathbf{y}_j$ and $\mathbf{f}(x_j, \mathbf{y}_j)$. Done!

In Eq. (3.27) we have chosen a constant step size h. However, the step size can easily be varied at each step, i.e. $h = h_j$. For instance, the step size could be varied according the magnitude of the right-hand side or, better, on the basis of an estimate of the approximation error.

3.3 Error estimation and convergence

When solving IVP approximately, it is crucial to have an estimate of the committed approximation error. For simplicity we limit the presentation to the scalar IVP Eq. (3.25) with a constant step size.

To analyze the accuracy of so-called *one-step*[2] methods we consider the general expression

$$y_{j+1} = y_j + h\Phi(x_j, y_j, y_{j+1}, h) \tag{3.34}$$

and call Φ the *increment function*, h the *step size* and y_j is the numerical approximation to the exact solution $y(x_j)$ at x_j. For the two Euler methods the increment functions read

$$\begin{aligned} \Phi(x_j, y_j, y_{j+1}, h) &= f(x_j, y_j) && \text{explicit Euler} \\ \Phi(x_j, y_j, y_{j+1}, h) &= f(x_{j+1}, y_{j+1}) && \text{implicit Euler.} \end{aligned}$$

We discretize the interval $I = [x_0, \bar{x}]$ (on which we seek the approximate solution) uniformly in N subintervals

$$x_j = x_0 + hj \quad , \quad j = 0, 1, ..., N, \tag{3.35}$$

where the constant step size is given by

$$h = \frac{\bar{x} - x_0}{N}. \tag{3.36}$$

We then define the *local truncation error* (LTE) at the j-th step as

$$e_j = y(x_j) - y(x_{j-1}) - h\Phi(x_{j-1}, y(x_{j-1}), y(x_j), h). \tag{3.37}$$

In words: the LTE is the error after one step of a method executed with the exact solution.

However, in practice it is crucial to have an estimate of the error after a certain number of steps, e.g. at the end of the integration interval. For this we define the *global truncation error* (GTE) at the j-th step as

$$E_j = y(x_j) - y_j. \tag{3.38}$$

In words: the GTE is the *cumulated* error after j steps.

The LTE of a method is usually computed with simple Taylor expansions under the assumption that the function $f(x, y)$ and the solution $y(x)$ are sufficiently often

[2] One-step methods because they depend only on the current y_j and the next step y_{j+1}.

continuously differentiable. Let's compute the LTE of the explicit Euler method:

$$\begin{aligned}
e_j &= y(x_j) - y(x_{j-1}) - hf(x_{j-1}, y(x_{j-1})) \\
&= y(x_{j-1}+h) - y(x_{j-1}) - hf(x_{j-1}, y(x_{j-1})) && (x_j = x_{j-1}+h) \\
&= y(x_{j-1}) + h\underbrace{y'(x_{j-1})}_{\overset{ODE}{=} f(x_{j-1},y(x_{j-1}))} + \frac{h^2}{2}y''(x_{j-1}) + \dots - y(x_{j-1}) - hf(x_{j-1}, y(x_{j-1})) && \text{(Taylor expansion)} \\
&= y(x_{j-1}) + hf(x_{j-1}, y(x_{j-1})) + \frac{h^2}{2}y''(x_{j-1}) + \dots - y(x_{j-1}) - hf(x_{j-1}, y(x_{j-1})) \\
&= \frac{h^2}{2}y''(x_{j-1}) + \dots \\
&= O(h^2)
\end{aligned}$$

This tells us that the LTE of the Euler method is proportional to h^2. Try as an exercise to compute the LTE for the implicit Euler method. You should also get $|e_j| = O(h^2)$.

We define that a method has *order of accuracy* p if

$$|e_j| = Ch^{p+1} = O(h^{p+1}), \tag{3.39}$$

where $C > 0$ is a constant (independent of h!). Note especially the subtlety between p and $p+1$ in this definition. This will become clear in the following. Therefore, the Euler methods have order of accuracy $p = 1$. We have already observed this experimentally at the end of the previous section.

So we have seen that the LTE can be easily estimated with Taylor expansions. Let's now try to relate the LTE to the more practically meaningful GTE at the end of the integration, i.e. E_N. This will be slightly technical, but don't be afraid. Also note that nobody will ask you to do these steps in an exam. We do this only for explicit methods[3] which then take the form

$$y_{j+1} = y_j + h\Phi(x_j, y_j, h). \tag{3.40}$$

Let the method have order of accuracy p, that is we have for the LTE

$$|e_j| = |y(x_{j+1}) - y(x_j) - h\Phi(x_j, y(x_j), h)| = Ch^{p+1}. \tag{3.41}$$

Moreover, we have to assume a so-called *Lipschitz condition* for the increment function Φ

$$|\Phi(x, y, h) - \Phi(x, y^*, h)| \le L|y - y^*| \quad \forall x \in [x_0, \bar{x}] \text{ and } \forall y, y^* \in I[x_0, \bar{x}]. \tag{3.42}$$

Now let's state the obvious

$$y(x_N) = y(x_{N-1}) + h\frac{y(x_N) - y(x_{N-1})}{h} \tag{3.43}$$

and the final integration step (to reach $\bar{x} = x_N$!)

$$y_N = y_{N-1} + h\Phi(x_{N-1}, y_{N-1}). \tag{3.44}$$

[3] Doing it for implicit methods is slightly more cumbersome and in the end one draws the same conclusion.

By subtracting Eq. (3.43) from Eq. (3.44) we get

$$\begin{aligned}
y(x_N) - y_N &= y(x_{N-1}) + h\frac{y(x_N) - y(x_{N-1})}{h} - y_{N-1} - h\Phi(x_{N-1}, y_{N-1}) \\
E_N &= y(x_{N-1}) - y_{N-1} + h\left(\frac{y(x_N) - y(x_{N-1})}{h} - \Phi(x_{N-1}, y_{N-1})\right) \\
E_N &= E_{N-1} + h\left(\frac{y(x_N) - y(x_{N-1})}{h} - \Phi(x_{N-1}, y_{N-1})\right),
\end{aligned}$$

where we have used the definition of the GTE in the second and third equality. Now we simply add zero to the right-hand side

$$\begin{aligned}
E_N = E_{N-1} \quad &+ h\left(\frac{y(x_N) - y(x_{N-1})}{h} - \Phi(x_{N-1}, y(x_{N-1}))\right) \\
&+ h\Big(\Phi(x_{N-1}, y(x_{N-1})) - \Phi(x_{N-1}, y_{N-1})\Big),
\end{aligned}$$

that is we simply added and subtracted $h\Phi(x_{N-1}, y(x_{N-1}))$. By taking the absolute value of the last equation and applying the triangle inequality we get

$$\begin{aligned}
|E_N| \le |E_{N-1}| \quad &+ h\overbrace{\left|\frac{y(x_N) - y(x_{N-1})}{h} - \Phi(x_{N-1}, y(x_{N-1}))\right|}^{\overset{Eq.\ (3.41)}{\le} \quad Ch^p} \\
&+ h\underbrace{\Big|\Phi(x_{N-1}, y(x_{N-1})) - \Phi(x_{N-1}, y_{N-1})\Big|}_{\overset{Eq.\ (3.42)}{\le} \quad L|y(x_{N-1}) - y_{N-1}| = L|E_{N-1}|}.
\end{aligned}$$

By making use of the order of accuracy and Lipschitz condition we obtain

$$|E_N| \le (1 + Lh)|E_{N-1}| + Ch^{p+1}$$

which says that we can estimate the GTE at step N by a relation for the GTE at step $N - 1$. By repeatedly using this relation we get

$$\begin{aligned}
|E_N| &\le (1 + Lh)\underbrace{|E_{N-1}|}_{\le(1+Lh)|E_{N-2}+Ch^{p+1}} + Ch^{p+1} \\
&\le (1 + Lh)^2\underbrace{|E_{N-2}|}_{\le(1+Lh)|E_{N-3}+Ch^{p+1}} + Ch^{p+1}\Big(1 + (1 + Lh)\Big) \\
&\le (1 + Lh)^3\underbrace{|E_{N-3}|}_{\le(1+Lh)|E_{N-4}+Ch^{p+1}} + Ch^{p+1}\Big(1 + (1 + Lh) + (1 + Lh)^2\Big) \\
&\vdots \\
&\le (1 + Lh)^N\underbrace{|E_0|}_{=|y(x_0)-y_0|=0!} + Ch^{p+1}\sum_{i=0}^{N-1}(1 + Lh)^i = Ch^{p+1}\sum_{i=0}^{N-1}(1 + Lh)^i
\end{aligned}$$

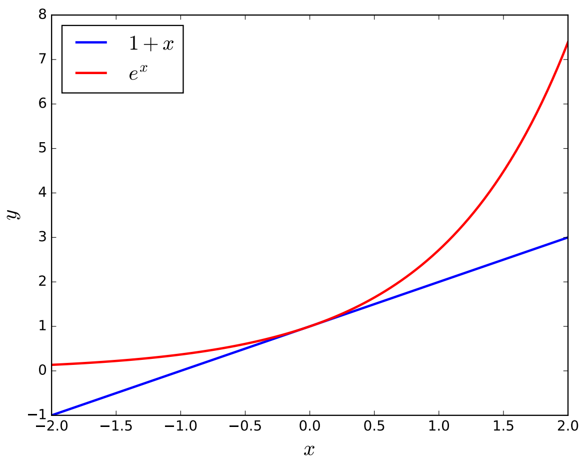

Figure 3.5: $1+x \leq e^x$

For the last term we can use the geometric series formula[4] and one further trick to get

$$\sum_{i=0}^{N-1}(1+Lh)^i = \frac{(1+Lh)^N - 1}{1+Lh-1} = \frac{\overbrace{(1+Lh)^N}^{\leq e^{NhL}} - 1}{Lh} \leq \frac{e^{NhL}-1}{Lh}.$$

To convince yourself of the last inequality just use the series definition of the exponential function, or, have a look at Figure 3.5. Now we can finally come to a conclusion

$$|E_N| \leq Ch^{p+1}\frac{e^{N\overbrace{h}^{\frac{\bar{x}-x_0}{N}}L}-1}{Lh} = Ch^p\frac{e^{L(\bar{x}-x_0)}-1}{L} = \tilde{C}h^p = O(h^p).$$

This tells us that if the LTE of a method is $O(h^{p+1})$, then the GTE is $O(h^p)$, that is the definition of the order of accuracy of a method Eq. (3.39) makes sense!

3.4 Runge-Kutta methods

As we have seen, the Euler methods are first-order accurate, i.e. $|E_N| = O(h)$. So to make the error a hundred times smaller we have to decrease the step size h by the same factor. This means that we have to make hundred times more integration steps, or in other words, a hundred times more work! A natural question then comes up: can we do better? Of course the answer is positive and these higher-order methods are the subject of this section. Again to simplify the presentation, we limit ourselves to the scalar IVP Eq. (3.25) with a constant step size.

[4]Geometric series formula $\sum\limits_{i=0}^{N} q^i = \frac{q^{N+1}-1}{q-1}$.

The Euler methods evaluated the slopes either at the present step x_j or at the new step x_{j+1}. A first improvement could be to evaluate the slope somewhere between x_j and x_{j+1}. A first try would be right in the middle, which would result in a method of the form

$$y_{j+1} = y_j + hf\left(x_j + h/2, \tilde{y}_{j+1/2}\right), \tag{3.45}$$

where $\tilde{y}_{j+1/2}$ is an (yet unknown!) approximation of the solution at $x_j + h/2$. One simple way to approximate $\tilde{y}_{j+1/2}$ could be through half an explicit Euler step

$$\tilde{y}_{j+1/2} = y_j + \frac{h}{2} f(x_j, y_j). \tag{3.46}$$

The method is illustrated in the left panel of Figure 3.6. From the figure one already gets the impression that the new method gives a better approximation than the explicit Euler method.

To show that the new method is indeed better, let's compute the local truncation error

$$e_{j+1} = y(x_{j+1}) - \left(y(x_j) + hf\left(x_j + \frac{h}{2}, y(x_j) + \frac{h}{2} f(x_j, y(x_j))\right)\right) \tag{3.47}$$

The computations proceed as for the Euler methods with heavy usage of Taylor series expansions. On the one hand we have

$$\begin{aligned} y(x_{j+1}) &= y(x_j) + hy'(x_j) + \frac{h^2}{2} y''(x_j) + O(h^3) \\ &= y(x_j) + hf(x_j, y(x_j)) \\ &\quad + \frac{h^2}{2}\left(\frac{\partial f}{\partial x}(x_j, y(x_j)) + \frac{\partial f}{\partial y}(x_j, y(x_j)) \cdot f(x_j, y(x_j))\right) + O(h^3), \end{aligned} \tag{3.48}$$

where we made use of the ODE and its first derivative

$$\begin{aligned} y' &= f(x, y) \\ y'' &= \frac{\partial f}{\partial x}(x, y) + \frac{\partial f}{\partial y}(x, y) \cdot f(x, y). \end{aligned} \tag{3.49}$$

On the other hand we have

$$\begin{aligned} f\left(x_j + \frac{h}{2}, y(x_j) + \frac{h}{2} f(x_j, y(x_j))\right) &= f(x_j, y(x_j)) \\ &\quad + \frac{h}{2}\frac{\partial f}{\partial x}(x_j, y(x_j)) \\ &\quad + \frac{h}{2}\frac{\partial f}{\partial y}(x_j, y(x_j)) \cdot f(x_j, y(x_j)) + O(h^2). \end{aligned} \tag{3.50}$$

Plugging Eq. (3.48) and Eq. (3.50) into Eq. (3.47)

$$\begin{aligned} e_{j+1} = &\left[y(x_j) + hf(x_j, y(x_j)) + \frac{h^2}{2}\left(\frac{\partial f}{\partial x}(x_j, y(x_j)) + \frac{\partial f}{\partial y}(x_j, y(x_j)) \cdot f(x_j, y(x_j))\right) + O(h^3)\right] \\ &- \left[y(x_j) + h\left(f(x_j, y(x_j)) + \frac{h}{2}\frac{\partial f}{\partial x}(x_j, y(x_j)) + \frac{h}{2}\frac{\partial f}{\partial y}(x_j, y(x_j)) \cdot f(x_j, y(x_j)) + O(h^2)\right)\right] \\ = &\ O(h^3). \end{aligned}$$

Therefore the new method has order of accuracy $p = 2$! So it is indeed better than the Euler method. Note that we silently assumed that the function $f(x, y)$ and the solution $y(x)$ are sufficiently often continuously differentiable.

Now, by how much do you have to decrease the step size to get an error that is a hundred times smaller? Ten times! Hence the amount of computational work you have to do is also reduced and this explains the interest in higher-order methods.

Approximate solutions of IVP can be constructed by following a path in the ODE's slope field. In this respect, we introduce a very intuitive notation for the new method above:

$$\begin{aligned} k_1 &= f(x_j, y_j) \\ k_2 &= f(x_j + h/2, y_j + h/2k_1) \\ y_{j+1} &= y_j + hk_2. \end{aligned} \tag{3.51}$$

The k_1 and k_2 are approximations of the slopes and the next integration step y_{j+1} is obtained by combining them. This method is known in the literature under the name of *Runge's method.*

Another method is given by

$$\begin{aligned} k_1 &= f(x_j, y_j) \\ k_2 &= f(x_j + h, y_j + hk_1) \\ y_{j+1} &= y_j + \frac{h}{2}(k_1 + k_2), \end{aligned} \tag{3.52}$$

which is known as *Heun's method.* It takes a full explicit Euler step, computes an approximation of the slope at x_{j+1} and then computes y_{j+1} by taking the average slope (see the right panel of Figure 3.6). This method has order of accuracy $p = 2$ (try to show it!).

Both Runge's and Heun's method are explicit one-step methods. When we constructed Runge's, we took half an explicit Euler step Eq. (3.46). Instead, one could take half an implicit Euler step to get

$$\begin{aligned} k_1 &= f\left(x_j + \frac{h}{2}, y_j + \frac{h}{2}k_1\right) \\ y_{j+1} &= y_j + hk_1, \end{aligned} \tag{3.53}$$

which is known as the *implicit midpoint method.* Implicit because the midpoint slope k_1 is only implicitly defined! Yet another implicit method is given by

$$\begin{aligned} k_1 &= f(x_j, y_j) \\ k_2 &= f\left(x_j + h, y_j + \frac{h}{2}(k_1 + k_2)\right) \\ y_{j+1} &= y_j + \frac{h}{2}(k_1 + k_2), \end{aligned} \tag{3.54}$$

which is known as *implicit trapezoidal method.* Both implicit methods are second order accurate. They are illustrated in Figure 3.7.

So far we have seen a few explicit and implicit examples of higher-order methods. These are representatives of a large family of one-step methods known as *Runge-Kutta methods*. They have the following form

$$y_{j+1} = y_j + h \sum_{i=1}^{s} b_i k_i, \tag{3.55}$$

where

$$k_i = f\left(x_j + c_i h, y_j + h \sum_{l=1}^{s} a_{il} k_l \right). \tag{3.56}$$

Here s is the number of *stages*, c_i the *nodes*, b_i the *weights*, a_{il} the *Runge-Kutta matrix* and k_i the *slopes*.

It is convenient write Runge-Kutta methods with the so-called *Butcher tableau*:

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array} = \begin{array}{c|c} \mathbf{c} & A \\ \hline & \mathbf{b}^T \end{array} \tag{3.57}$$

It is customary to not write the zeros at and above the diagonal in the Runge-Kutta matrix. A few Runge-Kutta methods written with a Butcher tableau are given below

- explicit Euler

$$\begin{array}{c|c} 0 & \\ \hline & 1 \end{array}$$

- implicit Euler

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

- Runge's method

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

- Heuns's method

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

- implicit midpoint method

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

- implicit trapezoidal method

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

- *"The" Runge-Kutta method* (order of accuracy $p = 4$)
  (Also known as *classical Runge-Kutta method* and simply *RK4*)

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

The last example is probably the most well-known Runge-Kutta method and is illustrated in Figure 3.8. It is explicit and fourth-order accurate.

Explicit Runge-Kutta methods are characterized by a lower triangular Runge-Kutta matrix

$$A = \begin{pmatrix} 0 & & & \\ a_{21} & 0 & & \\ \vdots & & \ddots & \\ a_{s1} & \cdots & & 0 \end{pmatrix}.$$

As you probably have guessed, there is a tight connection between Runge-Kutta methods and quadrature. To see this, let's look at the LTE of a Runge-Kutta method

$$\begin{aligned} e_{j+1} &= y(x_{j+1}) - \left(y(x_j) + h \sum_{i=1}^{s} b_i k_i \right) \\ &= y(x_{j+1}) - y(x_j) - h \sum_{i=1}^{s} b_i k_i \\ &= \int_{x_j}^{x_{j+1}} y'(x) \mathrm{d}x - h \sum_{i=1}^{s} b_i k_i \\ &= \int_{x_j}^{x_{j+1}} f(x, y(x)) \mathrm{d}x - h \sum_{i=1}^{s} b_i k_i. \end{aligned}$$

Note that the k_i are evaluation of f! Therefore a Runge-Kutta method has the form of a quadrature rule with nodes $x_j + c_i h$ and weights b_i.

The implicit midpoint/trapezoidal method have the midpoint/trapezoidal quadrature rule as their basis. The Runge-Kutta method has the Simpson quadrature rule as basis.

Example 3.9. Let's illustrate the seen Runge-Kutta methods by the same example IVP as for the Euler methods

$$\begin{cases} y'(x) = -y(x) + 2\cos(x) \\ y(0) = 1 \end{cases}$$

for $x \in [0, 4]$. One again easily verifies that the exact solution is given by

$$y(x) = \sin(x) + \cos(x).$$

By applying the seen Runge-Kutta methods we get:

- Runge's method
$$\begin{aligned} k_1 &= f(x_j, y_j) \\ &= -y_j + 2\cos(x_j) \\ k_2 &= f(x_j + h/2, y_j + h/2k_1) \\ &= -\left(y_j + \frac{h}{2}k_1\right) + 2\cos\left(x_j + \frac{h}{2}\right) \\ y_{j+1} &= y_j + hk_2. \end{aligned}$$

- Heun's method:
$$\begin{aligned} k_1 &= f(x_j, y_j) \\ &= -y_j + 2\cos(x_j) \\ k_2 &= f(x_j + h, y_j + hk_1) \\ &= -(y_j + hk_1) + 2\cos(x_j + h) \\ y_{j+1} &= y_j + \frac{h}{2}(k_1 + k_2) \end{aligned}$$

- Implicit midpoint method:
$$\begin{aligned} k_1 &= f\left(x_j + \frac{h}{2}, y_j + \frac{h}{2}k_1\right) \\ &= -\left(y_j + \frac{h}{2}k_1\right) + 2\cos\left(x_j + \frac{h}{2}\right) \\ y_{j+1} &= y_j + hk_1 \end{aligned}$$

  Note that k_1 is only implicitly defined and we have to solve for it, which in this simple (linear) example just gives
$$k_1 = \frac{1}{1 + h/2}\left(-y_j + 2\cos\left(x_j + \frac{h}{2}\right)\right)$$

- Implicit trapezoidal method:
$$\begin{aligned} k_1 &= f(x_j, y_j) \\ &= -y_j + 2\cos(x_j) \\ k_2 &= f\left(x_j + h, y_j + \frac{h}{2}(k_1 + k_2)\right) \\ &= -\left(y_j + \frac{h}{2}(k_1 + k_2)\right) + 2\cos(x_j + h) \\ y_{j+1} &= y_j + \frac{h}{2}(k_1 + k_2) \end{aligned}$$

Here k_2 is only implicitly defined and solving for it gives

$$k_2 = \frac{1}{1+\frac{h}{2}}\left(-y_j - \frac{h}{2}k_1 + 2\cos\left(x_j + h\right)\right)$$

- (The/Classical) Runge-Method (RK4):

$$\begin{aligned}
k_1 &= f(x_j, y_j) \\
&= -y_j + 2\cos(x_j) \\
k_2 &= f(x_j + h/2, y_j + h/2k_1) \\
&= -(y_j + h/2k_1) + 2\cos(x_j + h/2) \\
k_3 &= f(x_j + h/2, y_j + h/2k_2) \\
&= -(y_j + h/2k_2) + 2\cos(x_j + h/2) \\
k_4 &= f(x_j + h, y_j + hk_3) \\
&= -(y_j + hk_3) + 2\cos(x_j + h) \\
y_{j+1} &= y_j + \frac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right)
\end{aligned}$$

In Figure 3.9 and Figure 3.10 are shown the results of the above methods (with $h = 1$, i.e. $N = 4$). Moreover, in the left panel of Figure 3.10 is shown the final GTE, that is at $x = \bar{x} = 4$, for all methods. As you can see, Runge's, Heun's and both implicit methods are second order accurate, i.e. $|E_N| = O(h^2)$. However, the classical Runge-Kutta method is fourth-order accurate, i.e. $|E_N| = O(h^4)$. ▲

Runge-Kutta methods can be applied to systems of first order ODEs in a straight-forward manner: simply replace the scalar y_j and $f(x_j, y_j)$ by corresponding vectors $\mathbf{y}_j$ and $\mathbf{f}(x_j, \mathbf{y}_j)$.

The step size can also easily be varied, i.e. $h = h_j$. In practice, the step size is changed to reach a certain prescribed accuracy[5]. However, we will not discuss the issue of adaptive step sizes in this lecture and we refer to [2, 1, 3, 4] and references therein.

[5]In MATLAB's `ode45` the desired accuracy can be set with the `RelTol` and `AbsTol` property values.

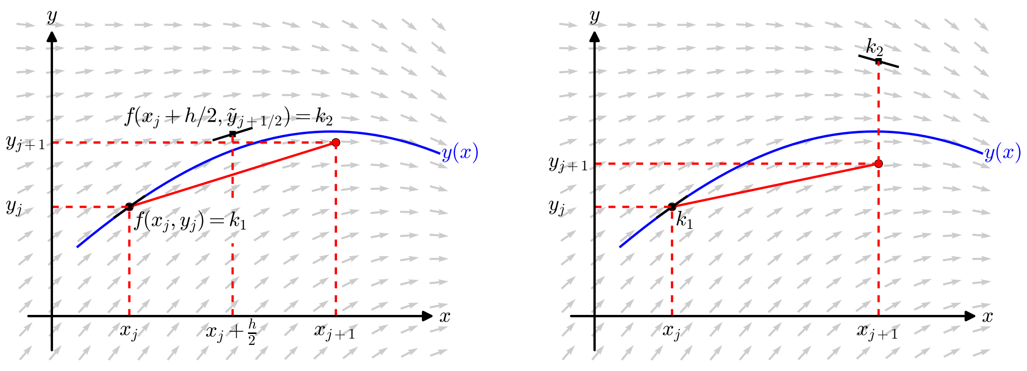

Figure 3.6: Runge's method (left panel) and Heun's method (right panel).

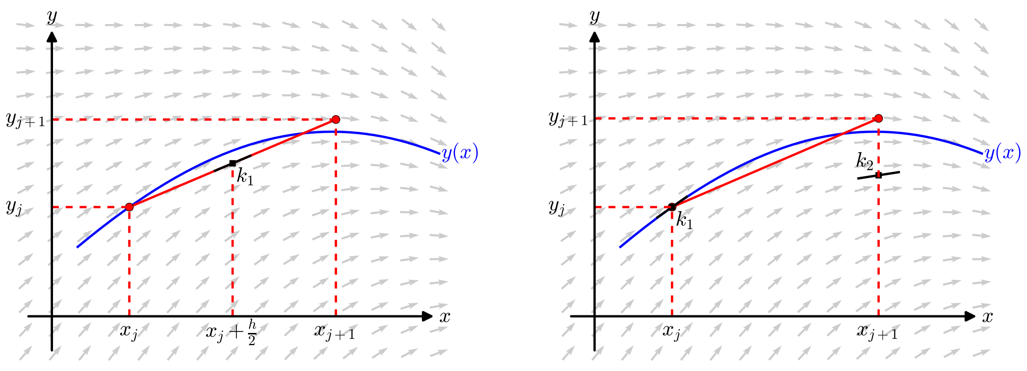

Figure 3.7: Implicit midpoint method (left panel) and implicit trapezoidal method (right panel).

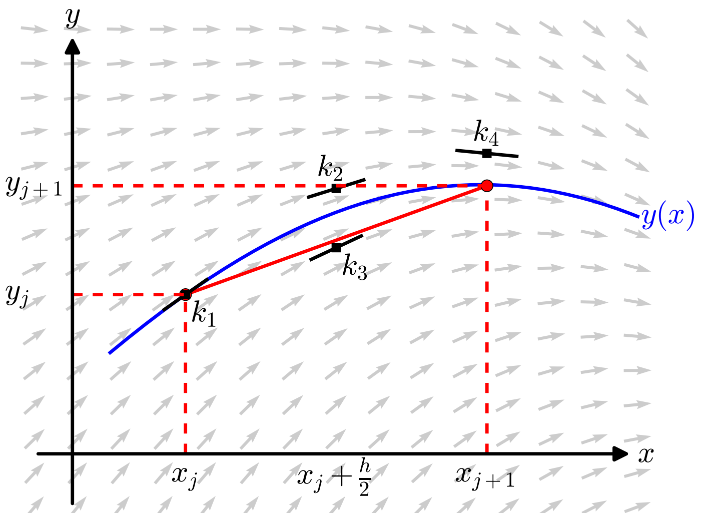

Figure 3.8: "The"/classical Runge-Kutta method.

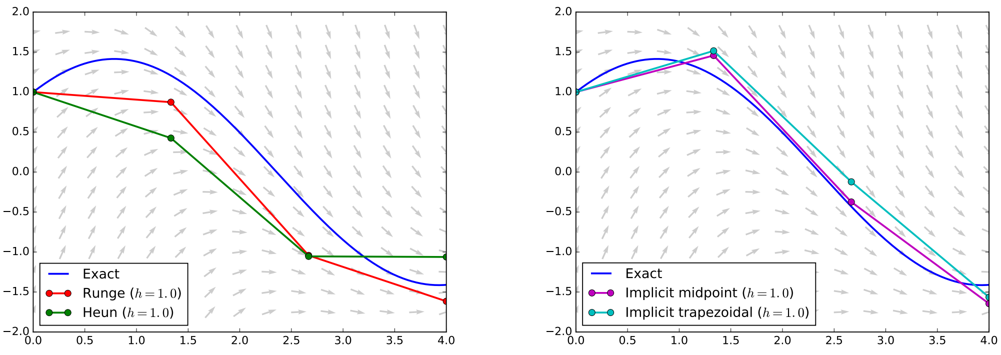

Figure 3.9: Runge's and Heun's method (left panel), and implicit trapezoidal and midpoint method (right panel) applied to the IVP in Example 3.9.

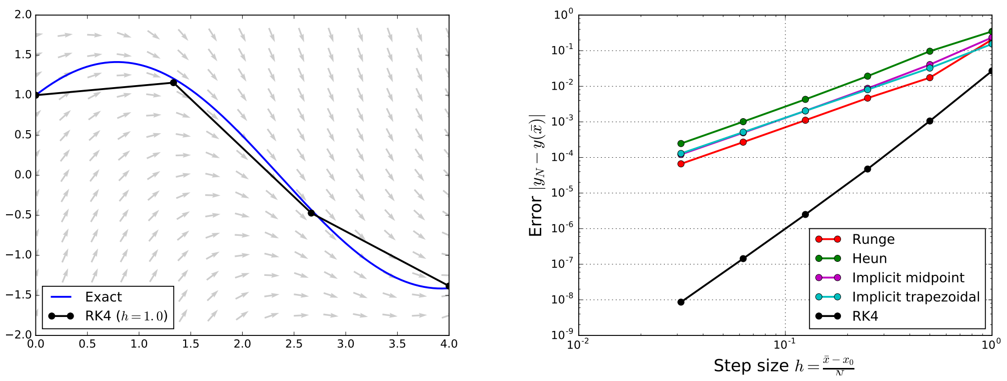

Figure 3.10: Classical Runge-Kutta method applied to the IVP in Example 3.9 (left panel) and errors for all seen Runge-Kutta methods in Example 3.9 (right panel).

3.5 Multistep methods

So far we have seen only one-step methods which compute an approximation of y_{j+1} at x_{j+1} solely from an approximation y_j at x_j (maybe with the help of intermediate stages). In contrast, so-called *multistep methods* use the information available from previous steps x_{j-1}, y_{j-1}, x_{j-2}, y_{j-2}, ... In doing so, the variation of the step size becomes more involved and therefore we shall keep it constant.

For ease of presentation, we limit the discussion to the scalar IVP Eq. (3.25). We start from the integral equation equivalent to the ODE (see Eq. (3.2)) between x_j and x_{j+1}

$$y(x_{j+1}) = y(x_j) + \int_{x_j}^{x_{j+1}} f(x, y(x))\mathrm{d}x. \tag{3.58}$$

The idea is to approximate the integral using interpolation. Suppose we have some approximations $y_{j-1} \approx y(x_{j-1})$ and $y_j \approx y(x_j)$. We can then approximate the right-hand side of the ODE $f(x, y)$ by the linear interpolation polynomial[6]

$$\begin{aligned} p_1(x) &= f(x_{j-1}, y_{j-1})L_0(x) + f(x_{j-1}, y_{j-1})L_1(x) \\ &= f(x_{j-1}, y_{j-1})\frac{x - x_j}{x_{j-1} - x_j} + f(x_j, y_j)\frac{x - x_{j-1}}{x_j - x_{j-1}} \\ &= -f_{j-1}\frac{x - x_j}{h} + f_j\frac{x - x_{j-1}}{h}, \end{aligned}$$

where we introduced the notation $f(x_j, y_j) = f_j$ and used the constant step size $h = x_j - x_{j-1}$. This obviously represents an approximation of f over $[x_{j-1}, x_j]$. By extrapolating $p_1(x)$ over the interval $[x_j, x_{j+1}]$ we can try to approximate the integral equation Eq. (3.58) by

$$\begin{aligned} y_{j+1} &= y_j + \int_{x_j}^{x_{j+1}} p_1(x)\mathrm{d}x \\ &= y_j - \frac{f_{j-1}}{h}\int_{x_j}^{x_{j+1}} (x - x_j)\mathrm{d}x + \frac{f_j}{h}\int_{x_j}^{x_{j+1}} (x - x_{j-1})\mathrm{d}x \\ &= y_j - \frac{f_{j-1}}{h}\left(\frac{x^2}{2} - x_j x\right)\Bigg|_{x_j}^{x_{j+1}} + \frac{f_j}{h}\left(\frac{x^2}{2} - x_{j-1}x\right)\Bigg|_{x_j}^{x_{j+1}} \\ &= y_j - \frac{f_{j-1}}{h}\underbrace{\left(\frac{1}{2}\left(x_{j+1}^2 - x_j^2\right) - x_j\left(x_{j+1} - x_j\right)\right)}_{=\cdots=h^2/2} \\ &\quad + \frac{f_j}{h}\underbrace{\left(\frac{1}{2}\left(x_{j+1}^2 - x_j^2\right) - x_{j-1}\left(x_{j+1} - x_j\right)\right)}_{=\cdots=3h^2/2} \\ &= y_j - \frac{f_{j-1}}{h}\frac{h^2}{2} + \frac{f_j}{h}\frac{3h^2}{2} \end{aligned}$$

[6]See Chapter 1.

So we get an explicit update formula

$$y_{j+1} = y_j + h\left(\frac{3}{2}f_j - \frac{1}{2}f_{j-1}\right), \tag{3.59}$$

which is known in the literature as the *two-step Adams-Bashforth method* (AB2). This method makes use of the known y_{j-1} and y_j to compute y_{j+1}, hence two-step method! See Figure 3.11 for an illustration.

However, to apply the method Eq. (3.59) to an IVP, one needs also an approximation to y_1. This can easily be produced by any[7] one-step method of the previous section.

Let's now have a look at the method's accuracy properties. Similar error and accuracy definitions as for one-step methods hold for multistep methods. For instance, let's compute the LTE of the above method (again tacitly assuming that the function $f(x,y)$ and the solution $y(x)$ are sufficiently often continuously differentiable):

$$\begin{aligned}
e_{j+1} &= y(x_{j+1}) - \left(y(x_j) + h\left(\frac{3}{2}f(x_j, y(x_j)) - \frac{1}{2}f(x_{j-1}, y(x_{j-1}))\right)\right)\\
&\overset{ODE}{=} y(\underbrace{x_{j+1}}_{x_j+h}) - y(x_j) - h\frac{3}{2}y'(x_j) + h\frac{1}{2}y'(\underbrace{x_{j-1}}_{x_j-h})\\
&\overset{Taylor}{=} y(x_j) + hy'(x_j) + \frac{h^2}{2}y''(x_j) + \frac{h^3}{6}y'''(x_j) + \cdots\\
&- y(x_j)\\
&- \frac{3h}{2}y'(x_j)\\
&+ \frac{h}{2}\left(y'(x_j) - hy''(x_j) + \frac{h^2}{2}y'''(x_j) + \cdots\right)\\
&= \frac{5}{12}h^3y'''(x_j) + \cdots\\
&= O(h^3).
\end{aligned}$$

Like for one-step methods one can show that if the LTE is $O(h^{p+1})$, then the GTE is $O(h^p)$. So this motivates the definition, that we call a multistep method p-th order accurate if the LTE is $O(h^{p+1})$.

So the two-step Adam-Bashforth methods is second order accurate. Although this method is computationally as expensive as the first order accurate explicit Euler method (one new evaluation of f per step since you already have $f_{j-1} = f(x_{j-1}, y_{j-1})$), it has higher accuracy. Thereby it is computationally more efficient. This explains the interest in multistep methods. These methods gain computational efficiency by using information from previous integration steps (one-step methods use only the last one!).

Implicit multistep can also be easily constructed. This leads to *Adams-Moulton* and *Backward Difference Formula (BDF)* methods. However, we shall not treat them in this lecture and we refer to [2, 1, 3, 4] and references therein.

[7] Actually, you want to choose a one-step method that has at least the same order of accuracy than the multistep method you want to apply.

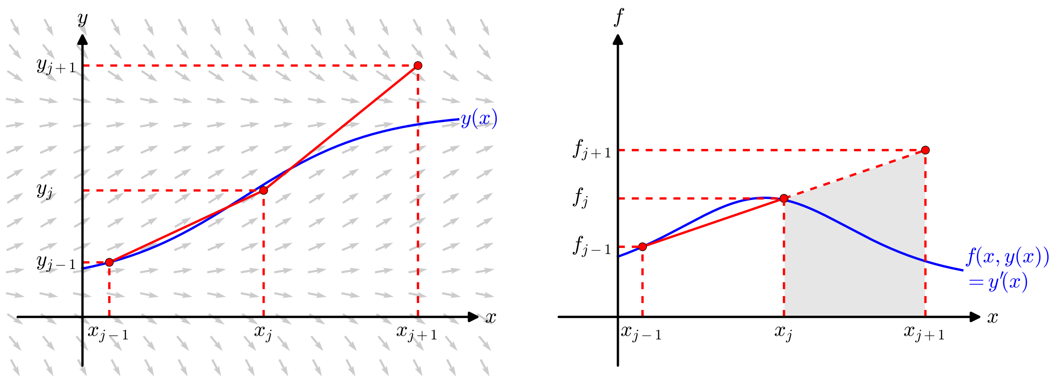

Figure 3.11: Two-step Adam-Bashforth method.

Example 3.10. Let's apply the above multistep method to the now familiar example IVP:

$$\begin{cases} y'(x) = -y(x) + 2\cos(x) \\ y(0) = 1 \end{cases}$$

for $x \in [0, 4]$. Applying the two-step Adam-Bashforth method (AB2) gives

$$\begin{aligned} y_{j+1} &= y_j + h\left(\frac{3}{2}f_j - \frac{1}{2}f_{j-1}\right) \\ &= y_j + h\left(\frac{3}{2}(-y_j + \cos(x_j)) - \frac{1}{2}(-y_{j-1} + \cos(x_{j-1}))\right) \\ &= \left(1 - \frac{3h}{2}\right) y_j + \frac{h}{2} y_{j-1} + h\left(\cos(x_j) - \cos(x_{j-1})\right). \end{aligned}$$

To compute the needed y_1 Runge's method was used. The left panel of Figure 3.12 shows the approximate solution with $h = 1/2$ ($N = 8$).

The right panel of the same figure shows the error as a function of the step size. As you can see, the method is indeed second order accurate. ▲

Solving first order systems with multistep methods is easy: just replace the scalar y_j and $f(x_j, y_j)$ by corresponding vectors $\mathbf{y}_j$ and $\mathbf{f}(x_j, \mathbf{y}_j)$.

3.6 Review questions

Here a few review questions[8] for the present chapter:

(a)

[8]FAQs at exams...

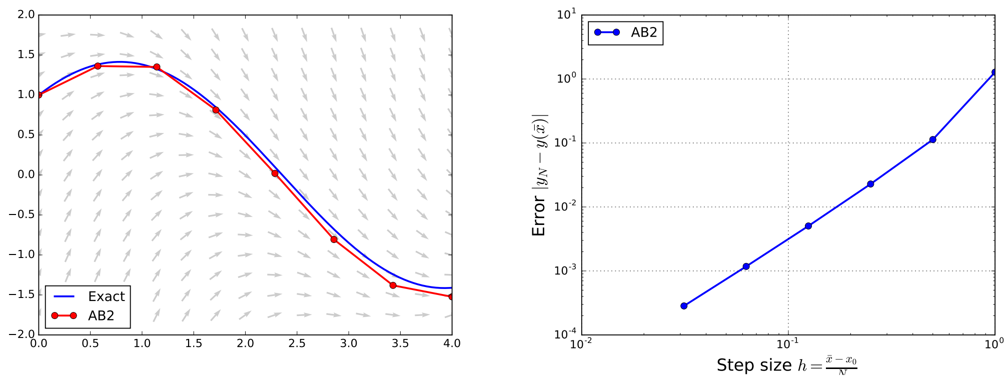

Figure 3.12: Two-step Adam-Bashforth method applied to the example IVP.

3.7 Bibliography

[1] E. Hairer and G. Wanner. *Solving ordinary differential equations. II*, volume 14 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, second edition, 1996. ISBN 3-540-60452-9. doi: 10.1007/978-3-642-05221-7. URL `http://dx.doi.org/10.1007/978-3-642-05221-7`. Stiff and differential-algebraic problems.

[2] E. Hairer, S. P. Nørsett, and G. Wanner. *Solving ordinary differential equations. I*, volume 8 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, second edition, 1993. ISBN 3-540-56670-8. Nonstiff problems.

[3] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical recipes in FORTRAN. The art of scientific computing*. 1992.

[4] Hans Rudolf Schwarz and Norbert Köckler. Numerische mathematik. 2011. doi: 10.1007/978-3-8348-8166-3. URL `http://dx.doi.org/10.1007/978-3-8348-8166-3`.