

## Chapter 3

# Numerical Solution of Ordinary Differential Equations

### 3.1 Introduction

We will here discuss the numerical solution of systems of ordinary differential equations (ODEs). There is a simple code enclosed, which can be used for some experiments.

#### 3.1.1 Scalar ODEs.

A scalar ODE is an equation of the form

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

where  $y'(t) = \frac{dy}{dt}$ . The *initial condition*  $y(t_0) = y_0$  is required for a unique solution.

**NB!** It is common to use the term *initial value problem (IVP)* for an ODE for which the initial value  $y(t_0) = y_0$  is given, and we only are interested in the solution for  $t > t_0$ . In this note, only initial value problems are considered.

**Example 3.1.1.** Consider the solution of the ODE

$$y'(t) = -2ty(t),$$

which can be written in the form  $y'(t) = f(t, y(t))$  with

$$f(t, y) = -2ty.$$

The general solution is the function

$$y(t) = Ce^{-t^2},$$

where  $C$  is a constant that depends on the initial condition  $y(t_0)$ , which we can find by using the techniques from Mathematics 1: We write first  $y' = dy/dt$  and then multiply formally with  $dt$  to obtain

$$dy = -2ty \, dt.$$

If  $y$  is equal to 0, we see that the solution of the equation is the constant zero function. Else we can divide by  $y$  and obtain

$$\frac{1}{y} dy = -2t \, dt.$$

Now we compute indefinite integrals of both sides, which yields

$$\log|y| = -t^2 + K.$$

Taking the exponential on both sides, we get

$$|y| = e^K e^{-t^2},$$

or

$$y = \pm e^K e^{-t^2}.$$

Setting  $C = \pm e^K$ , we finally obtain the solution

$$y(t) = C e^{-t^2}$$

for some  $C \in \mathbb{R}$  (the case  $C = 0$  corresponds to the constant zero solution we have discussed previously). In order to find the constant  $C$ , we have to use the initial condition for  $y(t_0)$ . For instance, we obtain for  $t_0 = 0$  and  $y(0) = 1$  the solution

$$y(t) = e^{-t^2}.$$

### 3.1.2 Systems of ODEs.

A system of  $m$  ODEs is given by

$$\begin{aligned} y_1' &= f_1(t, y_1, y_2, \dots, y_m), & y_1(t_0) &= y_{1,0} \\ y_2' &= f_2(t, y_1, y_2, \dots, y_m), & y_2(t_0) &= y_{2,0} \\ &\vdots & &\vdots \\ y_m' &= f_m(t, y_1, y_2, \dots, y_m), & y_m(t_0) &= y_{m,0} \end{aligned}$$

or, more compactly, by

$$\mathbf{y}'(t) = \mathbf{f}(t, \mathbf{y}(t)), \quad \mathbf{y}(t_0) = \mathbf{y}_0$$

where we use boldface to denote vectors in  $\mathbb{R}^m$ .

$$\mathbf{y}(t) = \begin{pmatrix} y_1(t) \\ y_2(t) \\ \vdots \\ y_m(t) \end{pmatrix}, \quad \mathbf{f}(t, \mathbf{y}) = \begin{pmatrix} f_1(t, y_1, y_2, \dots, y_m) \\ f_2(t, y_1, y_2, \dots, y_m) \\ \vdots \\ f_m(t, y_1, y_2, \dots, y_m) \end{pmatrix}, \quad \mathbf{y}_0 = \begin{pmatrix} y_{1,0} \\ y_{2,0} \\ \vdots \\ y_{m,0} \end{pmatrix},$$

**Example 3.1.2.** [The Lotka-Volterra equation](#) is a system of two ODEs describing the interaction between predators and prey over time. The system is given as

$$\begin{aligned} y_1'(t) &= \alpha y_1(t) - \beta y_1(t) y_2(t), \\ y_2'(t) &= \delta y_1(t) y_2(t) - \gamma y_2(t). \end{aligned}$$

Here  $t$  denotes time,  $y_1(t)$  describes the population of the prey species, and  $y_2(t)$  the population of predators. The positive parameters  $\alpha$ ,  $\beta$ ,  $\delta$ , and  $\gamma$  depend on the populations to be modelled. We can write this system in the form  $\mathbf{y}'(t) = \mathbf{f}(t, \mathbf{y}(t))$  with

$$\mathbf{f}(t, \mathbf{y}) = \begin{pmatrix} \alpha y_1 - \beta y_1 y_2 \\ \delta y_1 y_2 - \gamma y_2 \end{pmatrix}.$$

Note that the parameter  $t$  here actually does not appear on the right hand side.

### 3.1.3 Autonomous ODEs.

An ODE is called *autonomous* if  $\mathbf{f}$  is not a function of  $t$ , but only of  $\mathbf{y}$ . The Lotka-Volterra equation is an example of an autonomous ODE. A nonautonomous system can be made autonomous by a simple trick: Just add the equation

$$y'_{m+1} = 1, \quad y_{m+1}(t_0) = t_0,$$

and replace  $t$  with  $y_{m+1}$ . Thus, the ODE from Example 1, with initial values  $y(0) = 1$  can be written as

$$\begin{aligned} y'_1 &= -2y_2y_1, & y_1(0) &= 1, \\ y'_2 &= 1, & y_2(0) &= 0. \end{aligned}$$

### 3.1.4 Higher order ODEs.

An initial value problem for an ODE of order  $m$  is given by

$$u^{(m)} = f(t, u, u', \dots, u^{(m-1)}), \quad u(t_0) = u_0, \quad u'(t_0) = u'_0, \quad \dots, \quad u^{(m-1)}(t_0) = u_0^{(m-1)}.$$

Here  $u^{(1)} = u'$  and  $u^{(m+1)} = \frac{du^{(m)}}{dt}$  for  $m > 0$ .

Such equations can be written as a system of first order ODEs by the following trick:

Let

$$y_1(t) = u(t), \quad y_2(t) = u'(t), \quad y_3(t) = u^{(2)}(t), \quad \dots, \quad y_m(t) = u^{(m-1)}(t)$$

such that

$$\begin{aligned} y'_1 &= y_2, & y_1(t_0) &= u_0, \\ y'_2 &= y_3, & y_2(t_0) &= u'_0, \\ &\vdots & &\vdots \\ y'_{m-1} &= y_m, & y_{m-1}(t_0) &= u_0^{(m-2)}, \\ y'_m &= f(t, y_1, y_2, \dots, y_{m-1}, y_m), & y_m(t_0) &= u_0^{(m-1)}. \end{aligned}$$

This is a normal system of first order ODEs. The solution  $u$  of the original  $m$ -th order ODE is just the first component  $y_1$  of the solution of this system.

**Example 3.1.3.** Van der Pol's equation is a second order differential equation, given by

$$u'' = \mu(1 - u^2)u' - u, \quad u(0) = u_0, \quad u'(0) = u'_0,$$

where  $\mu > 0$  is some constant. Common choices for initial values are  $u_0 = 2$  and  $u'_0 = 0$ .

It can be rewritten as a system of first order ODEs:

$$\begin{aligned} y'_1 &= y_2, & y_1(0) &= u_0, \\ y'_2 &= \mu(1 - y_1^2)y_2 - y_1, & y_2(0) &= u'_0. \end{aligned}$$

## 3.2 Numerical methods for solving ODEs

In this note, we will discuss some techniques for the numerical solution of ordinary differential equations. For simplicity or presentation, we will develop and discuss these methods mostly based on scalar ODEs. The same methods, however, are equally applicable for systems of equations.

All the methods that we will discuss are so-called *one-step methods*. Given the ODE and the initial values  $(t_0, y_0)$ , we choose some step size  $h$  and let  $t_1 = t_0 + h$ . Based on this information, we calculate an approximation  $y_1$  to  $y(t_1)$ . Then, we repeat this process starting from  $(t_1, y_1)$  in order to calculate an approximation  $y_2$  of  $y(t_2)$ , where  $t_2 = t_1 + h$ . This process is repeated until some final point, here called  $t_{\text{end}}$  is reached.

In one-step methods, the approximation  $y_{k+1}$  of  $y(t_{k+1})$  does not depend on the values of  $y_{k-1}, y_{k-2}, \dots, y_0$ . The main alternative to this type of methods are *multi-step methods*, where the approximation  $y_{k+1}$  of  $y(t_{k+1})$  takes into account those values as well.

It should be emphasized that this strategy only will find approximations to the exact solution in some discrete points  $t_n, n = 0, 1, \dots$

### 3.2.1 Euler's method

Let us start with the simplest example, [Euler's method](#), known from Mathematics 1.

Given a (scalar) IVP

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

choose some step size  $h$ . The trick is as follows:

Perform a Taylor expansion (*Preliminaries*, section 4) of the exact (but unknown) solution  $y(t_0 + h)$  around  $t_0$ :

$$y(t_0 + h) = y(t_0) + hy'(t_0) + \frac{1}{2}h^2y''(t_0) + \dots$$

Assume the step size  $h$  to be small, such that the solution is dominated by the first two terms. In that case, these can be used as the numerical approximation in the next step, as the ODE implies that

$$y'(t_0) = f(t_0, y(t_0)) = f(t_0, y_0).$$

From this we obtain the approximation

$$y(t_0 + h) \approx y(t_0) + hy'(t_0) = y_0 + hf(t_0, y_0),$$

giving

$$y_1 = y_0 + hf(t_0, y_0).$$

This extends trivially to a system of equations, and to a general  $n$ . In practice, we would like to compute the solution over a finite interval  $t_0, t_{\text{end}}$ , using a finite number of steps, say  $N$ , in which case the stepsize is  $h = (t_{\text{end}} - t_0)/N$ . The complete algorithm (for a system of ODEs) becomes:

**Algorithm 2** Euler's method

---

**Input:** Right hand side  $\mathbf{f}$ , interval  $(t_0, t_{\text{end}})$ , initial value  $\mathbf{y}_0$  and number of steps  $N$ .  
 $h = (t_{\text{end}} - t_0)/N$   
**for**  $n = 0, 1, 2, \dots, N - 1$  **do**  
     $\mathbf{y}_{n+1} = \mathbf{y}_n + h\mathbf{f}(t_n, \mathbf{y}_n)$   
     $t_{n+1} = t_n + h$   
**end for**  
**Output:**  $(t_n, \mathbf{y}_n)_{n=0}^N$

---

**Exercise 3.2.1.** Test the implementation of Euler's method on the problem

$$y'(t) = -2ty(t), \quad y(0) = 1, \quad 0 \leq t \leq 1,$$

which has the analytic solution

$$y(t) = e^{-t^2}.$$

Try with different step sizes, for instance  $h = 0.1$ ,  $h = 0.05$  and  $h = 0.01$ . In each case, compare the numerical solution with the exact one.

**Exercise 3.2.2.** Solve the Lotka-Volterra equation

$$\begin{aligned} y_1'(t) &= \alpha y_1(t) - \beta y_1(t)y_2(t), & y_1(0) &= y_{1,0}, \\ y_2'(t) &= \delta y_1(t)y_2(t) - \gamma y_2(t), & y_2(0) &= y_{2,0}. \end{aligned}$$

In this exercise, use the parameters and initial values

$$\alpha = 2, \quad \beta = 1, \quad \delta = 0.5, \quad \gamma = 1, \quad y_{1,0} = 2, \quad y_{2,0} = 0.5.$$

Solve the equation over the interval  $[0, 20]$ , and use  $h = 0.02$ . Try also other step sizes, e.g.  $h = 0.1$  and  $h = 0.002$ . **NB!** In this case, the exact solution is not known. What is known is that the solutions are periodic and positive. Is this the case for the numerical solutions as well? Check for different values of  $h$ .

### 3.2.2 Heun's method

We will now discuss a first, improved, alternative to Euler's method:

Assume that we want to solve an

$$\mathbf{y}'(t) = \mathbf{f}(t, \mathbf{y}(t)).$$

Its exact solution can be written in integral form as

$$\mathbf{y}(t_n + h) = \mathbf{y}(t_n) + \int_{t_n}^{t_n+h} \mathbf{y}'(t) dt = \mathbf{y}(t_n) + \int_{t_n}^{t_n+h} \mathbf{f}(t, \mathbf{y}(t)) dt.$$

We now replace the integral on the right hand side by a numerical approximation using the trapezoidal rule:

$$\mathbf{y}(t_n + h) \approx \mathbf{y}(t_n) + \frac{h}{2} (\mathbf{f}(t_n, \mathbf{y}(t_n)) + \mathbf{f}(t_{n+1}, \mathbf{y}(t_{n+1}))).$$

Then we replace  $\mathbf{y}(t_n)$  and  $\mathbf{y}(t_{n+1})$  by the approximations  $\mathbf{y}_n$  and  $\mathbf{y}_{n+1}$ . The resulting method is the *trapezoidal rule for ODEs*, given by

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

This is an example of an *implicit* method. The numerical approximation  $\mathbf{y}_{n+1}$  appears on both sides of this equation and is therefore only implicitly given: If  $t_n, \mathbf{y}_n$  is known, a nonlinear equation has to be solved in order to find  $\mathbf{y}_{n+1}$ , and this has to be done for each step. There are important applications where this actually makes sense, which we will discuss in a later lecture in the context of *stiff ODEs*.

However, for the moment we want to avoid this additional complication and thus replace the  $\mathbf{y}_{n+1}$  on the right hand side by some approximation. One natural possibility here is to apply one step of Euler's method. This results in [Heun's method](#):

$$\begin{aligned} \mathbf{u}_{n+1} &= \mathbf{y}_n + h\mathbf{f}(t_n, \mathbf{y}_n), \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + \frac{h}{2}(\mathbf{f}(t_n, \mathbf{y}_n) + \mathbf{f}(t_{n+1}, \mathbf{u}_{n+1})). \end{aligned}$$

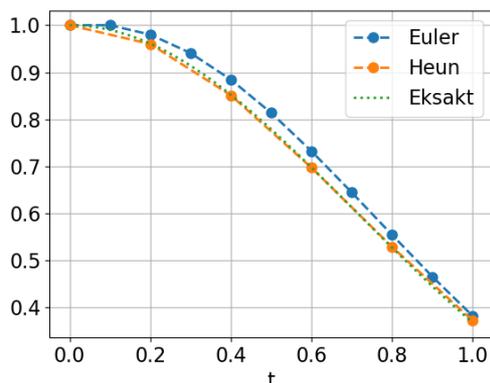
The method is commonly written in the form

$$\begin{aligned} \mathbf{k}_1 &= \mathbf{f}(t_n, \mathbf{y}_n), \\ \mathbf{k}_2 &= \mathbf{f}(t_n + h, \mathbf{y}_n + h\mathbf{k}_1), \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + \frac{h}{2}(\mathbf{k}_1 + \mathbf{k}_2). \end{aligned}$$

**Example 3.2.1.** Let us compare the numerical solution from Euler's and Heun's methods on the scalar test problem

$$y' = -2ty, \quad y(0) = 1$$

with the exact solution  $y(t) = e^{-t^2}$  on the interval  $[0, 1]$ . Use  $h = 0.1$  for Euler's method and  $h = 0.2$  for Heun's method. The result is:



and it is clear that Heun's method is significantly more accurate than Euler's method.

The main computational work in these methods lies in the evaluations of the function  $\mathbf{f}$ , especially in the case of large systems of ODEs. Euler's method requires one function evaluation in each iteration, whereas Heun's method requires two function evaluations: one for the computation of  $\mathbf{k}_1$  and a second one for the computation of  $\mathbf{k}_2$ . This means

that a single iteration of Heun's method is roughly double as computationally expensive as one iteration of Euler's method. If we want to have a fair comparison between the two methods, we should thus use half as many iterations for Heun's method as for Euler's method, which in turn means that the step length  $h$  for Heun's method should be double that of Euler's method. In this example, using Euler's method with  $h = 0.1$  and Heun's method with  $h = 0.2$  both leads to 10 function evaluations for the solution on the interval  $[0, 1]$ , and thus to a comparable total amount of computational work.

Doing so, we observe that Heun's method is significantly more accurate than Euler's method, even when the number of function evaluations are the same. Further, we notice that the error in Heun's method is reduced by a factor of approximately  $1/4$  whenever the step size is reduced by a factor  $1/2$ , indicating that the error satisfies  $|y(t_{\text{end}}) - y_N| \approx Ch^2$ , and the method is of order 2. In the theory part below, we will prove that this certainly is the case.

**Exercise 3.2.3.** Solve the Lotka-Volterra equation by Euler's and Heun's methods, again using twice as many steps for Euler's method as for Heun's method.

- 1) Use  $h = 0.01$  for Euler's method and  $h = 0.02$  for Heun's method.
- 2) Use  $h = 0.1$  for Euler's method and  $h = 0.2$  for Heun's method.

**Exercise 3.2.4.** Solve Van der Pol's equation by use of Heun's method and with Euler's method, and compare. Experiment with different choices of the step size  $h$ . Use  $\mu = 2$ , and solve the equation on the interval  $[0, 20]$ . Experiment with different values of  $\mu$ .

**Exercise 3.2.5.** Implement the classical Runge–Kutta method (also known as RK4) and verify numerically that the order of the method is 4. The method is given by

$$\begin{aligned} \mathbf{k}_1 &= \mathbf{f}(t_n, \mathbf{y}_n) \\ \mathbf{k}_2 &= \mathbf{f}\left(t_n + \frac{h}{2}, \mathbf{y}_n + \frac{h}{2}\mathbf{k}_1\right) \\ \mathbf{k}_3 &= \mathbf{f}\left(t_n + \frac{h}{2}, \mathbf{y}_n + \frac{h}{2}\mathbf{k}_2\right) \\ \mathbf{k}_4 &= \mathbf{f}(t_n + h, \mathbf{y}_n + h\mathbf{k}_3) \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + \frac{h}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4). \end{aligned}$$

### 3.2.3 Runge–Kutta methods

An  $s$ -stage Runge-Kutta method is given by

$$\begin{aligned} \mathbf{k}_i &= \mathbf{f}\left(t_n + c_i h, \mathbf{y}_n + h \sum_{j=1}^s a_{ij} \mathbf{k}_j\right), \quad i = 1, 2, \dots, s, \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + h \sum_{i=1}^s b_i \mathbf{k}_i. \end{aligned}$$

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

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\ \vdots & \vdots & & & \vdots \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array}$$



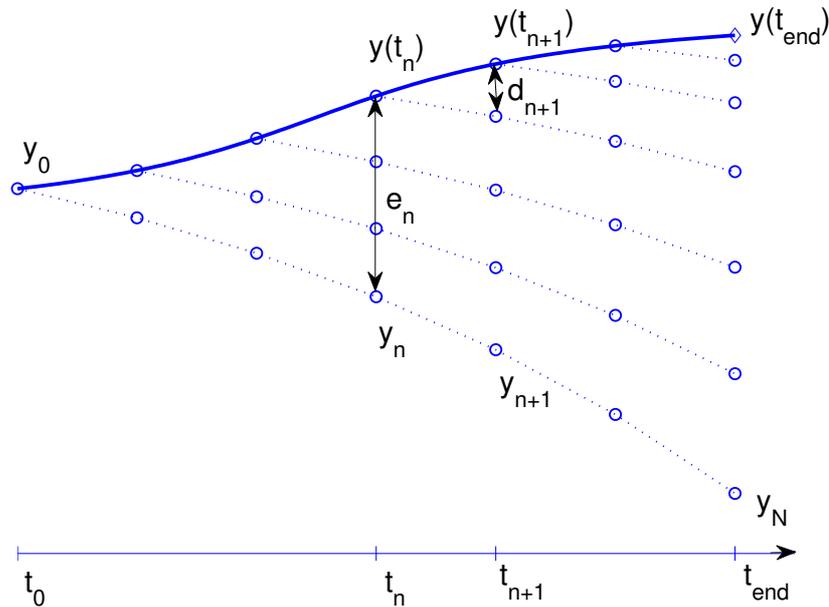


Figure 3.1: Lady Windermere’s fan, named after a comedy play by Oscar Wilde. The figure describes the transport and the accumulation of the local truncation errors  $d_{n+1}$  into the global error at the end point,  $e_N = y(t_{\text{end}}) - y_N$ .

choose the number of steps  $N$ , let the step size be  $h = (t_{\text{end}} - t_0)/N$ , such that  $t_{\text{end}} = t_N$ . What can we say about the error  $e_N = y(t_{\text{end}}) - y_N$ ?

The error analysis will be done on a scalar equation, but it also holds for systems of equations, as described in the end of the section.

**Local and global errors.** In this discussion we have to consider two kinds of errors:

- *Local truncation error*  $d_{n+1}$ : This is the error made in one single step, starting from  $(t_n, y(t_n))$ .
- *Global error*  $e_n$ : This is the difference between the exact and the numerical solution after  $n$  steps,  $e_n = y(t_n) - y_n$ .

**Theorem 3.3.3.** Consider the scalar ODE

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

to be solved over the interval  $[t_0, t_{\text{end}}]$  by Euler’s method, using stepsizes  $h = (t_{\text{end}} - t_0)/N$ . Assume that  $f$ ,  $f_y$  and  $f_t$  are bounded in the neighbourhood of the solution. Then there is a constant  $C > 0$ , independent of  $h$  such that the global error is

$$|e_N| \leq Ch,$$

thus the method is convergent of order 1.

*Proof.* The proof is divided in two parts. First, find an expression for the local truncation error. Then, prove how these errors propagates and adds up to find a bound for the global error.

The Taylor expansion of the exact solution of the ODE, starting from  $(t_n, y(t_n))$  is given by

$$y(t_n + h) = y(t_n) + hy'(t_n) + \frac{1}{2}y''(\xi), \quad t_n < \xi < t_n + h.$$

and one step with Euler's method, starting from  $(t_n, y(t_n))$  is

$$y_{n+1} = y(t_n) + hf(t_n, y(t_n)).$$

where  $y'(t_n) = f(t_n, y(t_n))$ . Thus the local truncation error is

$$d_{n+1} = y(t_n + h) - (y(t_n) + hy'(t_n)) = \frac{1}{2}h^2y''(\xi), \quad \xi \in (t_n, t_n + h). \quad (3.1)$$

The next step is to see how the global error  $e_n$  propagates over one step. We have

$$y(t_n + h) = y(t_n) + hf(t_n, y(t_n)) + d_{n+1}, \quad \text{Equation (3.1)}$$

$$y_{n+1} = y_n + hf(t_n, y_n), \quad \text{Euler's method.}$$

Subtract the second equation from the first, use that  $e_n = y(t_n) - y_n$  and the mean value theorem for functions  $\square$ . This yields the expression

$$e_{n+1} = e_n + h(f(t_n, y(t_n)) - f(t_n, y_n)) + d_{n+1} = e_n + hf_y(t_n, \eta)e_n + d_{n+1},$$

where  $f_y = \frac{\partial f}{\partial y}$ , and  $\eta$  is some value between  $y_n$  and  $y(t_n)$ . We now take the absolute value on each side, and apply the triangle inequality:

$$|e_{n+1}| = |e_n + hf_y(t_n, \eta)e_n + d_{n+1}| \leq |e_n| + h|f_y(t_n, \eta)||e_n| + |d_{n+1}|.$$

From the assumptions we know that  $f_y$  is bounded, e.g. there is a constant  $L > 0$  such that  $|f_y| < L$ . Similar  $y'' = (f(t, y))' = f_t + f_y f'$  is also bounded, there is a  $D > 0$  such that  $|y''(t)| < 2D$ .

From this the following inequality is obtained:

$$|e_{n+1}| \leq (1 + hL)|e_n| + Dh^2.$$

Since  $y_0 = y(t_0)$  the initial error  $e_0 = 0$ . The inequality above therefore results in the following bounds for the global errors:

$$\begin{aligned} |e_1| &\leq Dh^2 \\ |e_2| &\leq (1 + hL)|e_1| + Dh^2 \leq ((1 + hL) + 1)Dh^2 \\ |e_3| &\leq (1 + hL)|e_2| + Dh^2 \leq ((1 + hL)^2 + (1 + hL) + 1)Dh^2 \\ &\vdots \\ |e_N| &\leq (1 + hL)|e_{N+1}| + Dh^2 \leq \sum_{n=0}^{N-1} (1 + hL)^n Dh^2 \end{aligned}$$

Using the formula for the sum of the first terms of a geometric series  $\square$  and the series of

<sup>1</sup>The mean value theorem: There is a  $\nu$  between  $y$  and  $z$  such that  $g(y) - g(z) = g'(\nu)(y - z)$ . In our case, use  $g(y) = f(t, y)$  for a fixed  $t$ .

<sup>2</sup>Summation formula:  $\sum_{n=0}^{N-1} r^n = \frac{r^N - 1}{r - 1}$  for  $r \in \mathbb{R}$ .

the exponential function<sup>3</sup> we obtain the following:

$$\sum_{n=0}^{N-1} (1 + hL)^n = \frac{(1 + hL)^N - 1}{(1 + hL) - 1} < \frac{(e^{hL})^N - 1}{hL} = \frac{e^{hLN} - 1}{hL} = \frac{e^{L(t_{\text{end}} - t_0)} - 1}{hL},$$

where the last equality holds because  $(t_{\text{end}} - t_0) = hN$ . Finally, we plug this into the inequality for  $|e_N|$  above, and we see that we have proved the the following upper bound for the global:

$$|y(t_{\text{end}}) - y_N| = |e_N| \leq \frac{e^{L(t_{\text{end}} - t_0)} - 1}{L} Dh = Ch,$$

where the constant  $C = \frac{e^{L(t_{\text{end}} - t_0)} - 1}{L} D$  depends on the length of the integration interval  $t_{\text{end}} - t_0$ , of certain properties of the equation ( $L$  and  $D$ ), but *not* on the step size  $h$ .  $\square$

The numerical solution converges to the exact solution since

$$\lim_{N \rightarrow \infty} |e_N| = 0.$$

If the step size is reduced by a factor of 0.5, so will the error. This is in agreement with the previous numerical result.

**Remark 3.3.4.** Following the proof of the error estimate for Euler's method, we see that a local truncation error of size  $h^2$  leads to a final, global error of size  $h$ . Intuitively, this is because we need to take roughly  $1/h$  steps in order to reach the point  $t_{\text{end}}$ , although a precise proof is quite a bit more complicated than that, as we have seen. However, what this also should mean is that a method with a truncation order of size  $h^{p+1}$  should lead to a global error of  $h^p$ . The results of the next section show that, under certain conditions, this is indeed the case.

### 3.3.3 A general convergence result

A one-step method applied to a system of ODEs  $\mathbf{y}'(t) = \mathbf{f}(t, \mathbf{y}(t))$  can be written in the generic form

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\Phi(t_n, \mathbf{y}_n; h),$$

where the *increment function*  $\Phi$  typically depends on the function  $\mathbf{f}$  and some parameters defining the method.

For example, for Euler's method we have

$$\Phi(t_n, \mathbf{y}_n; h) = \mathbf{f}(t_n, \mathbf{y}_n),$$

whereas for Heun's method

$$\Phi(t_n, \mathbf{y}_n; h) = \frac{1}{2} \left( \mathbf{f}(t_n, \mathbf{y}_n) + \mathbf{f}(t_n + h, \mathbf{y}_n + h\mathbf{f}(t_n, \mathbf{y}_n)) \right).$$

All Runge–Kutta methods can in principle be written in this form as well (though in general for a rather complicated increment function  $\Phi$ ).

---

<sup>3</sup>The exponential:  $e^x = 1 + x + \frac{1}{2}x^2 + \dots = 1 + x + \sum_{n=2}^{\infty} \frac{x^n}{n!}$ . Thus  $1 + x < e^x$  whenever  $x > 0$ .

**Definition 3.3.5.** Order of a method. A method is of order  $p > 0$  if there is a constant  $C > 0$  such that

$$\|\mathbf{e}_N\| = \|\mathbf{y}(t_{\text{end}}) - \mathbf{y}_N\| \leq Ch^p,$$

where  $N$  is the number of steps taken to reach  $t_{\text{end}}$  using the step size  $h = (t_{\text{end}} - t_0)/N$ .

The local truncation error  $\mathbf{d}_{n+1}$  of such a method is

$$\mathbf{d}_{n+1} = \mathbf{y}(t_{n+1}) - (\mathbf{y}(t_n) + h\Phi(t_n, \mathbf{y}(t_n); h))$$

Replace the absolute values in the above proof with norms (*Preliminaries*, section 1), and the argument above can be used to prove the following:

**Theorem 3.3.6.** *Convergence of one-step methods.* Assume that there exist positive constants  $M$  and  $D$  such that the increment function satisfies

$$\|\Phi(t, \mathbf{y}; h) - \Phi(t, \mathbf{z}; h)\| \leq M\|\mathbf{y} - \mathbf{z}\| \quad (3.2)$$

and the local truncation error satisfies

$$\|\mathbf{y}(t+h) - (\mathbf{y}(t) + h\Phi(t, \mathbf{y}(t), h))\| \leq Dh^{p+1}$$

for all  $t$ ,  $\mathbf{y}$  and  $\mathbf{z}$  in a neighbourhood of the solution. In that case, the global error satisfies

$$\|\mathbf{e}_N\| = \|\mathbf{y}(t_{\text{end}}) - \mathbf{y}_N\| \leq Ch^p, \quad \text{with } C = \frac{e^{M(t_{\text{end}}-t_0)} - 1}{M} D.$$

In the case where  $\Phi$  is a Runge–Kutta method, one can show that (3.2) is always satisfied for some constant  $M$  provided that the function  $\mathbf{f}$  satisfies a Lipschitz condition.

**Example 3.3.7.** Let us consider Eulers method,  $\mathbf{y}_{n+1} = \mathbf{y}_n + h\mathbf{f}(t_n, \mathbf{y}_n)$ . So in this case  $\Phi(t, \mathbf{y}) = \mathbf{f}(t, \mathbf{y})$  and (3.2) is exactly the same as the Lipschitz condition for  $\mathbf{f}$ .

### 3.3.4 Order conditions for Runge–Kutta methods.

It can be proved that a Runge–Kutta method is of order  $p$  with  $p \leq 4$  if all the conditions up to and including  $p$  in the table below are satisfied:

$p$	conditions
1	$\sum_i b_i = 1$
2	$\sum_i b_i c_i = 1/2$
3	$\sum_i b_i c_i^2 = 1/3$ $\sum_{i,j} b_i a_{ij} c_j = 1/6$
4	$\sum_i b_i c_i^3 = 1/4$ $\sum_{i,j} b_i c_i a_{ij} c_j = 1/8$ $\sum_{i,j} b_i a_{ij} c_j^2 = 1/12$ $\sum_{i,j,k} b_i a_{ij} a_{jk} c_k = 1/24$

Here sums are taken over all the indices from 1 to  $s$ , with  $s$  being the number of stages of the method. It is possible to derive similar conditions also for higher orders  $p$ , but the conditions rapidly become more and more complex.

Recall also, that we have always assumed that Runge–Kutta methods satisfy the conditions  $c_i = \sum_j a_{ij}$  for all  $i = 1, \dots, s$ .

**Example 3.3.8.** Apply the conditions to Heun's method, for which  $s = 2$  and the Butcher tableau is

$$\begin{array}{c|cc} c_1 & a_{11} & a_{12} & 0 & 0 & 0 \\ c_2 & a_{21} & a_{22} & 1 & 1 & 0 \\ \hline & b_1 & b_2 & \frac{1}{2} & \frac{1}{2} & \end{array} .$$

The order conditions are:

$$p = 1 \qquad b_1 + b_2 = \frac{1}{2} + \frac{1}{2} = 1 \qquad \text{OK}$$

$$p = 2 \qquad b_1 c_1 + b_2 c_2 = \frac{1}{2} \cdot 0 + \frac{1}{2} \cdot 1 = \frac{1}{2} \qquad \text{OK}$$

$$p = 3 \qquad b_1 c_1^2 + b_2 c_2^2 = \frac{1}{2} \cdot 0^2 + \frac{1}{2} \cdot 1^2 = \frac{1}{2} \neq \frac{1}{3} \qquad \text{Not satisfied}$$

$$b_1(a_{11}c_1 + a_{12}c_2) + b_2(a_{21}c_1 + a_{22}c_2) = \frac{1}{2}(0 \cdot 0 + 0 \cdot 1) + \frac{1}{2}(1 \cdot 0 + 0 \cdot 1)$$

$$= 0 \neq \frac{1}{6} \qquad \text{Not satisfied}$$

The method is of order 2.

**Convergence properties of Heun's method (optional).** This is included mostly to demonstrate that convergence analysis is not a trivial task.

To prove convergence and to find the order of a method two things are needed:

- the local truncation error, expressed as a power series in the step size  $h$
- the condition  $\|\Phi(t, \mathbf{y}; h) - \Phi(t, \mathbf{z}; h)\| \leq M\|\mathbf{y} - \mathbf{z}\|$

The local truncation error is found by comparing Taylor series expansions of the exact and the numerical solutions starting from the same point. In practice, this is not trivial. For simplicity, we will here do this only for a scalar equation  $y'(t) = f(t, y(t))$ . The result is valid for systems as well.

In the following, we will use the notation

$$f_t = \frac{\partial f}{\partial t}, \quad f_y = \frac{\partial f}{\partial y}, \quad f_{tt} = \frac{\partial^2 f}{\partial t^2}, \quad f_{ty} = \frac{\partial^2 f}{\partial t \partial y} \quad \text{etc.}$$

Further, we will suppress the arguments of the function  $f$  and its derivatives. So  $f$  is to be understood as  $f(t, y(t))$  although it is not explicitly written.

The Taylor expansion of the exact solution  $y(t + h)$  is given by

$$y(t + h) = y(t) + hy'(t) + \frac{h^2}{2}y''(t) + \frac{h^3}{6}y'''(t) + \dots$$

Higher derivatives of  $y(t)$  can be expressed in terms of the function  $f$  by using the chain rule and the product rule for differentiation:

$$\begin{aligned}y'(t) &= f, \\y''(t) &= f_t + f_y y' = f_t + f_y f, \\y'''(t) &= f_{tt} + f_{ty} y' + f_{yt} f + f_{yy} y' f + f_y f_t + f_y f_y y' = f_{tt} + 2f_{ty} f + f_{yy} f^2 + f_y f_t + (f_y)^2 f.\end{aligned}$$

We then find the series of the exact and the numerical solution around  $x_0, y_0$  (any other point will do equally well). From the discussion above, the series for the exact solution becomes

$$y(t_0 + h) = y_0 + hf + \frac{h^2}{2}(f_t + f_y f) + \frac{h^3}{6}(f_{tt} + 2f_{ty} f + f_{yy} f^2 + f_y f_t + (f_y)^2 f) + \dots,$$

where  $f$  and all its derivatives are evaluated in  $(t_0, y_0)$ . For the numerical solution we get

$$\begin{aligned}k_1 &= f(t_0, y_0) = f, \\k_2 &= f(t_0 + h, y_0 + hk_1) \\&= f + hf_t + f_y hk_1 + \frac{1}{2}f_{tt}h^2 + f_{ty}h k_1 + \frac{1}{2}f_{yy}h^2 k_1^2 + \dots \\&= f + h(f_t + f_y f) + \frac{h^2}{2}(f_{tt} + 2f_{ty} f + f_{yy} f^2) + \dots, \\y_1 &= y_0 + \frac{h}{2}(k_1 + k_2) = y_0 + \frac{h}{2}\left(f + f + h(f_t + f_y f) + \frac{h^2}{2}(f_{tt} + 2f_{ty} f + f_{yy} f^2) + \dots\right) \\&= y_0 + hf + \frac{h^2}{2}(f_t + f_y f) + \frac{h^3}{4}(f_{tt} + 2f_{ty} f + f_{yy} f^2) + \dots,\end{aligned}$$

and the local truncation error will be

$$d_1 = y(t_0 + h) - y_1 = \frac{h^3}{12}(-f_{tt} - 2f_{ty} f - f_{yy} f^2 + 2f_y f_t + 2(f_y)^2 f) + \dots$$

The first nonzero term in the local truncation error series is called *the principal error term*. For  $h$  sufficiently small this is the term dominating the error, and this fact will be used later.

Although the series has been developed around the initial point, series around  $t_n, y(t_n)$  will give similar results, and it is possible to conclude that, given sufficient differentiability of  $f$ , there is a constant  $D$  such that

$$|d_n| \leq Dh^3.$$

Further, we have to prove the condition on the increment function  $\Phi(t, y)$ . For  $f$  differentiable, there is for all  $y, z$  some  $\xi$  between  $t$  and  $y$  such that  $f(t, y) - f(t, z) = f_y(t, \xi)(y - z)$ . Let  $L$  be a constant such that  $|f_y| \leq L$ , and for all  $t, y, z$  of interest we get

$$|f(t, y) - f(t, z)| \leq L|y - z|.$$

The increment function for Heun's method is given by

$$\Phi(t, y) = \frac{1}{2}(f(t, y) + f(t + h, y + hf(t, y))).$$

By repeated use of the condition above and the triangle inequality for absolute values we get

$$\begin{aligned}
|\Phi(t, y) - \Phi(t, z)| &= \frac{1}{2} |f(t, y) + f(t+h, y+hf(t, y)) - f(t, z) - hf(t+h, z+hf(t, z))| \\
&\leq \frac{1}{2} (|f(t, y) - f(t, z)| + |f(t+h, y+hf(t, y)) - f(t+h, z+hf(t, z))|) \\
&\leq \frac{1}{2} (L|y-z| + L|y+hf(t, y) - z - hf(t, z)|) \\
&\leq \frac{1}{2} (2L|y-z| + hL^2|y-z|) \\
&= \left(L + \frac{h}{2}L^2\right)|y-z|.
\end{aligned}$$

Assuming that the step size  $h$  is bounded above by some  $H$ , we can conclude that

$$|\Phi(t, y) - \Phi(t, z)| \leq M|y-z|, \quad M = L + \frac{H}{2}L^2.$$

In conclusion: Heun's method is convergent of order 2.

### 3.4 Error estimation and step size selection

To control the global error  $y(t_n) - y_n$  is notoriously difficult, and far beyond what will be discussed in this course. To control the local error in each step and adjust the step size accordingly is rather straightforward, as we will see.

#### 3.4.1 Error estimation

Given two methods, one of order  $p$  and the other of order  $p+1$  or higher. Assume we have reached a point  $(t_n, \mathbf{y}_n)$ . One step forward with each of these methods can be written as

$$\begin{aligned}
\mathbf{y}_{n+1} &= \mathbf{y}_n + h\Phi(t_n, \mathbf{y}_n; h), & \text{order } p, \\
\widehat{\mathbf{y}}_{n+1} &= \mathbf{y}_n + h\widehat{\Phi}(t_n, \mathbf{y}_n; h), & \text{order } p+1 \text{ or more.}
\end{aligned}$$

Let  $\mathbf{y}(t_{n+1}; t_n, \mathbf{y}_n)$  be the exact solution of the ODE through  $(t_n, \mathbf{y}_n)$ . We would like to find an estimate for *the local error*  $\mathbf{l}_{n+1}$ , that is, the error in one step starting from  $(t_n, \mathbf{y}_n)$ ,

$$\mathbf{l}_{n+1} = \mathbf{y}(t_{n+1}; t_n, \mathbf{y}_n) - \mathbf{y}_{n+1}.$$

As we already have seen, the local error is found by finding the power series in  $h$  of the exact and the numerical solution. The local error is of order  $p$  if the lowest order terms in the series where the exact and the numerical solution differ is of order  $p+1$ . So the local errors of the two methods are

$$\begin{aligned}
\mathbf{y}(t_{n+1}; t_n, \mathbf{y}_n) - \mathbf{y}_{n+1} &= \Psi(t_n, \mathbf{y}_n)h^{p+1} + \dots, \\
\mathbf{y}(t_{n+1}; t_n, \mathbf{y}_n) - \widehat{\mathbf{y}}_{n+1} &= \dots,
\end{aligned}$$

where  $\Psi(t_n, \mathbf{y}_n)$  is a term consisting of method parameters and differentials of  $\mathbf{f}$  and ... contains all the terms of the series of order  $p+2$  or higher. Taking the difference gives

$$\widehat{\mathbf{y}}_{n+1} - \mathbf{y}_{n+1} = \Psi(t_n, \mathbf{y}_n)h^{p+1} + \dots$$

Assume now that  $h$  is small, such that the *principal error term*  $\Psi(t_n, \mathbf{y}_n)h^{p+1}$  dominates the error series. Then a reasonable approximation to the unknown local error  $\mathbf{l}_{n+1}$  is the *local error estimate*  $\mathbf{le}_{n+1}$ :

$$\mathbf{le}_{n+1} = \widehat{\mathbf{y}}_{n+1} - \mathbf{y}_{n+1} \approx \mathbf{y}(t_{n+1}; t_n, \mathbf{y}_n) - \mathbf{y}_{n+1}.$$

**Example:** Apply Euler's method of order 1 and Heun's method of order 2 with  $h = 0.1$  to the equation

$$y' = -2ty, \quad y(0) = 1.$$

Use this to find an approximation to the error after one step.

Euler's method:

$$y_1 = 1.0 - 0.1 \cdot 2 \cdot 0 \cdot 1.0 = 1.0.$$

Heun's method

$$\begin{aligned} k_1 &= -2 \cdot 0.0 \cdot 1.0 = 0.0, \\ k_2 &= -2 \cdot 0.1 \cdot (1 + 0.0) = -0.2, \\ \widehat{y}_1 &= 1.0 + \frac{0.1}{2} \cdot (0.0 - 0.2) = 0.99. \end{aligned}$$

The error estimate and the local error are respectively

$$le_1 = \widehat{y}_1 - y_1 = -10^{-2}, \quad l_1 = y(0.1) - y_1 = e^{-0.1^2} - 1.0 = -0.995 \cdot 10^{-2}.$$

So in this case the error estimate is a quite decent approximation to the actual local error.

### 3.4.2 Stepsize control

The next step is to control the local error, that is, choose the step size so that  $\|\mathbf{le}_{n+1}\| \leq \text{Tol}$  for some given tolerance Tol, and for some chosen norm  $\|\cdot\|$ .

Essentially:

Given  $t_n, \mathbf{y}_n$  and a step size  $h_n$ .

- Perform one step with the method of choice, and find the error estimate  $\mathbf{le}_{n+1}$ .
- if  $\|\mathbf{le}_{n+1}\| < \text{Tol}$ 
  - Accept the solution  $t_{n+1}, \mathbf{y}_{n+1}$ .
  - If possible, increase the step size for the next step.
- else
  - Repeat the step from  $(t_n, \mathbf{y}_n)$  with a reduced step size  $h_n$ .

In both cases, the step size will change. But how?

From the discussion above, we have that

$$\|\mathbf{le}_{n+1}\| \approx Dh_n^{p+1}.$$

where  $\mathbf{le}_{n+1}$  is the error estimate we can compute,  $D$  is some unknown quantity, which we assume almost constant from one step to the next. We aim for a step size  $h_{\text{new}}$  such that

$$\text{Tol} \approx Dh_{\text{new}}^{p+1}.$$

From these two approximations we get:

$$\frac{\text{Tol}}{\|\mathbf{le}_{n+1}\|} \approx \left(\frac{h_{\text{new}}}{h_n}\right)^{p+1} \quad \Rightarrow \quad h_{\text{new}} \approx \left(\frac{\text{Tol}}{\|\mathbf{le}_{n+1}\|}\right)^{\frac{1}{p+1}} h_n.$$

To avoid too many rejected steps, it is wise to be a bit conservative when choosing the new step size, so the following is used in practice:

$$h_{\text{new}} = P \cdot \left(\frac{\text{Tol}}{\|\mathbf{le}_{n+1}\|}\right)^{\frac{1}{p+1}} h_n.$$

where the *pessimist factor*  $P < 1$  is some constant, normally chosen between 0.5 and 0.95.

### 3.4.3 Implementation

We have all the bits and pieces for constructing an adaptive ODE solver based on Euler's and Heun's methods. There are still some practical aspects to consider:

- The combination of the two methods can be written as

$$\begin{aligned} \mathbf{k}_1 &= \mathbf{f}(t_n, \mathbf{y}_n), \\ \mathbf{k}_2 &= \mathbf{f}(t_n + h, \mathbf{y}_n + h\mathbf{k}_1), \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + h\mathbf{k}_1, && \text{Euler} \\ \widehat{\mathbf{y}}_{n+1} &= \mathbf{y}_n + \frac{h}{2}(\mathbf{k}_1 + \mathbf{k}_2), && \text{Heun} \\ \mathbf{le}_{n+1} &= \|\widehat{\mathbf{y}}_{n+1} - \mathbf{y}_{n+1}\| = \frac{h}{2}\|\mathbf{k}_2 - \mathbf{k}_1\|. \end{aligned}$$

- Even if the error estimate is derived for the lower order method, in this case Euler's method, it is common to advance the solution with the higher order method, since the additional accuracy is for free. This is usually referred to as *local extrapolation*.
- Adjust the last step to be able to terminate the solutions exactly in  $t_{\text{end}}$ .
- To avoid infinite loops, add some stopping criteria. In the code below, there is a maximum number of allowed steps (rejected or accepted).

## 3.5 A summary of some terms and definitions

There have been quite a few definitions and different error terms in this note. So let us list some of them (not exclusive):

**Definitions:**

$\mathbf{y}' = \mathbf{f}(t, \mathbf{y})$	the ODE
$\mathbf{y}(t; t^*, \mathbf{y}^*)$ ,	the exact solution of the ODE through $(t^*, \mathbf{y}^*)$
$\mathbf{y}(t) = \mathbf{y}(t; t_0, \mathbf{y}_0)$ ,	the exact solution of $\mathbf{y}' = \mathbf{f}(t, \mathbf{y})$ , $\mathbf{y}(t_0) = \mathbf{y}_0$
$\mathbf{y}_{n+1} = \mathbf{y}_n + h\Phi(t_n, \mathbf{y}_n; h)$ ,	one step of the method

Let  $\Phi$  represent a method of order  $p$  and  $\hat{\Phi}$  a method of order  $p + 1$ .

**Error concepts:**

$\mathbf{d}_{n+1} = \mathbf{y}(t_n + h; t_n, \mathbf{y}(t_n)) - \left( \mathbf{y}(t_n) + h\Phi(t_n, \mathbf{y}(t_n); h) \right)$ ,	the local truncation error
$\mathbf{l}_{n+1} = \mathbf{y}(t_n + h; t_n, \mathbf{y}_n) - \left( \mathbf{y}_n + h\Phi(t_n, \mathbf{y}_n; h) \right)$ ,	the local error
$\mathbf{le}_{n+1} = h \left( \hat{\Phi}(t_n, \mathbf{y}_n; h) - \Phi(t_n, \mathbf{y}_n; h) \right)$ ,	the local error estimate, $\mathbf{le}_{n+1} \approx \mathbf{l}_{n+1}$
$\mathbf{e}_n = \mathbf{y}(t_n) - \mathbf{y}_n$	the global error