Chapter 5

Ordinary differential equations

Differential equations are a powerful tool in mathematical modelling and computational science. In most practical situations the equations are not solvable in simple closed form, and one needs to adopt numerical approximations. In this chapter we consider some basic aspects of the numerical solution of ordinary differential equations. Our focus is mainly on initial value problems, but at the end of the chapter we briefly discuss a class of two-point boundary value problems.

5.1 Example and motivation

Example 5.1.1. (Biology) *Consider a population of y animals in an environment where a maximum of B individuals can coexist. We assume that the initial population y_0 satisfies $y_0 < B$ and that for small populations $y \ll B$ the reproduction rate of the animals is approximately equal to some constant C. For larger populations the rate should slow down so that the population never exceeds B. The simplest model for such a situation is the logistic equation*

$$y'(t) = Cy(t)\left(1 - \frac{y(t)}{B}\right), \ t > 0, \quad y(0) = y_0, \tag{5.1}$$

where $y' = dy/dt$. By solving this equation we can find the evolution of the population in time.

This equation can be generalised to model two interacting populations in competition for the same resources. For instance, suppose that a predator population y_2 survives by hunting a prey population y_1. Then one can model the evolution of the two populations using the following system of differential equations (sometimes called a Lotka-Volterra predator-prey model)

$$\begin{cases} y_1'(t) = C_1 y_1(t)\left[1 - b_1 y_1(t) - d_2 y_2(t)\right], \\ y_2'(t) = -C_2 y_2(t)\left[1 - b_2 y_2(t) - d_1 y_1(t)\right]. \end{cases} \tag{5.2}$$

Here C_1 and C_2 are the growth factors of the two populations, d_1 and d_2 accounts for interaction between the population (preying, contamination etc.), and b_1 and b_2 are related to the amount of food/resources available for each population.

5.2 Initial value problems

We will consider initial value problems (IVP) of the following form: given a function $f : \mathbb{R}_+ \times \mathbb{R} \to \mathbb{R}$, continuous with respect to both arguments, and initial data $y_0 \in \mathbb{R}$, find $y : \mathbb{R}_+ \to \mathbb{R}$ satisfying

$$\begin{cases} y'(t) = f(t, y(t)) & \text{for } t > 0, \\ y(0) = y_0. \end{cases} \tag{5.3}$$

This is often called the *Cauchy problem* for the differential equation $y'(t) = f(t, y(t))$. In §5.7 we will consider the extension to systems of equations such as (5.2), but for now we restrict our attention to the one-dimensional case.

Example 5.2.1.

- *An example of a linear IVP is given by:*

$$\begin{cases} y'(t) = 3y(t) - 3t & \textit{for } t > 0, \\ y(0) = 1. \end{cases} \tag{5.4}$$

  Here $f(t, v) = 3v - 3t$ and the exact solution is $y(t) = 2/3e^{3t} + t + 1/3$ for $t > 0$.

- *An example of a nonlinear IVP is given by:*

$$\begin{cases} y'(t) = \sqrt[3]{y(t)} & \textit{for } t > 0, \\ y(0) = 0, \end{cases} \tag{5.5}$$

  with $f(t, v) = \sqrt[3]{v}$. This problem admits the following three solutions : $y(t) = 0$, $y(t) = \sqrt{8t^3/27}$, $y(t) = -\sqrt{8t^3/27}$.

- *Another example of a nonlinear IVP is:*

$$\begin{cases} y'(t) = 1 + y^2(t) & \textit{for } t > 0, \\ y(0) = 0. \end{cases} \tag{5.6}$$

  This problem has exact solution $y(t) = \tan(t)$ in the interval $0 < t < \frac{\pi}{2}$. But note that $y(t) \to \infty$ as $t \to \frac{\pi}{2}$, so the solution is only defined locally rather than on the whole of $\mathbb{R}_+$.

We recall the following result on existence and uniqueness for problem (5.3), which we quote without proof from Süli and Mayers.

Theorem 5.2.2 (Picard's Theorem)**.** *Suppose that $f : D \to \mathbb{R}$ is continuous on the rectangle $D = [0, t_{\max}] \times [y_0 - c, y_0 + c]$, for some $t_{\max} > 0$ and $c > 0$, and that f satisfies a Lipschitz condition in D with respect to y, i.e. $\exists L > 0$ such that*

$$|f(t, v) - f(t, w)| \leq L|v - w|, \qquad \forall t \in [0, t_{\max}] \text{ and } \forall v, w \in [y_0 - c, y_0 + c]. \tag{5.7}$$

Suppose also that

$$K := \max_{t \in [0, t_{\max}]} |f(t, y_0)| \leq \frac{cL}{(\mathrm{e}^{Lt_{\max}} - 1)}. \tag{5.8}$$

Then there exists a unique continuously differentiable function $y : [0, t_{\max}] \to [y_0 - c, y_0 + c]$ such that $y(0) = y_0$ and $y'(t) = f(t, y)$ for $t \in [0, t_{\max}]$.

Remark 5.2.3. *This version of Picard's Theorem might look more complicated than other versions you have seen before. The extra condition (5.8) allows us to prove existence and uniqueness of the solution on the* specified *interval $[0, t_{\max}]$ rather than just on* some *interval $[0, \delta]$ as in more standard versions.*

In particular we note that if $f(t, y)$ is differentiable with respect to y throughout D with uniformly bounded partial derivative $\partial f/\partial y$, then the Lipschitz condition (5.7) is satisfied in D with $L = \max_{(t,y) \in D} |\partial f/\partial y(t, y)|$, by the mean value theorem.

In example (5.4), $|f(t, v) - f(t, w)| = |(3v - 3t) - (3w - 3t)| = 3|v - w|$ for any $t, v, w \in \mathbb{R}$. Hence (5.7) is satisfied with $L = 3$ and $c > 0$ arbitrary. Given any $t_{\max} > 0$, we have $K = \max\{3, 3t_{\max}\} = 3 \max\{1, t_{\max} - 1\}$, which means we can satisfy (5.8) by taking $c > \max\{1, t_{\max} - 1\}(\mathrm{e}^{3t_{\max}} - 1)$. Thus the IVP (5.3) has a global solution, i.e. $y(t)$ is uniquely defined for $t \in [0, \infty)$.

For example (5.5), the lack of unique solvability does not contradict Theorem 5.2.2 since the theorem doesn't apply in this case. This is because $f(t, y)$ is not Lipschitz with respect to y in any neighbourhood of $(0, 0)$. Indeed, taking $w = 0$ in (5.7) gives

$$|f(t, v) - f(t, w)| = |f(t, v)| = v^{\frac{1}{3}},$$

which tends to zero slower than Lv as $v \to 0$, for any constant $L > 0$. In other words, we cannot find $L > 0$ such that (5.7) is satisfied, no matter how small $t_{\max}$ and c are.

Finally, for example (5.6) we observe that

$$|f(t, v) - f(t, w)| = |v^2 - w^2| = |(v + w)||(v - w)|, \qquad \forall t, v, w \in \mathbb{R}.$$

Hence (5.7) holds with $L = 2c$. Also, $|f(t, 0)| = 1$. So for Theorem 5.2.2 to apply we need $2c^2 \geq (\mathrm{e}^{2ct_{\max}} - 1)$. This can be satisfied for $t_{\max}$ sufficiently small by choosing c appropriately, giving local existence and uniqueness. But for large enough $t_{\max}$ (e.g. $t_{\max} \geq 1$ is sufficient) this inequality cannot be satisfied for any $c > 0$, and hence we cannot use Theorem 5.2.2

to prove global existence and uniqueness. This is consistent with our observation that the exact solution blows up at finite t. (But notice that the theory breaks down before the exact solution actually blows up at $t = \pi/2$.)

5.3 Numerical discretization

To obtain an approximate solution to the IVP (5.3) on a bounded interval $[0, t_{\max}]$ we will adopt the so-called "finite difference" approach. We start by choosing an integer $N \geq 1$ and discretizing the interval $[0, t_{\max}]$ into N subintervals $[t_n, t_{n+1}]$, where the mesh points (not necessarily evenly spaced) satisfy

$$0 = t_0 < t_1 < \ldots < t_N = t_{\max},$$

and the associated mesh spacings are defined by

$$h_n = t_{n+1} - t_n, \qquad n = 0, 1, \ldots, N-1.$$

Our discrete approximation of the continuously differentiable function $y(t)$ will consist of a set of $N+1$ real numbers $\{u_n\}_{n=0}^{N}$, where for each n the number u_n represents an approximation of the value of $y(t)$ at the mesh point $t = t_n$.

To derive a model satisfied by $\{u_n\}_{n=0}^{N}$, we have to replace the differential equation $y'(t) = f(t, y(t))$ by a suitable approximation. Recall that the derivative $y'(t)$ of a differentiable function y is defined as

$$y'(t) = \lim_{h \to 0} \frac{y(t+h) - y(t)}{h}.$$

Hence, one way to approximate the derivative $y'(t)$ at $t = t_n$ is by a difference quotient $(y(t_n + h) - y(t_n))/h$ for some finite $h > 0$ (this is the origin of the term "finite difference" above). A natural choice is to take $h = h_n$, the local mesh spacing, because then we can replace the unknown values $y(t_n + h_n)$ and $y(t_n)$ by their discrete approximations u_{n+1} and u_n. Finally, replacing $f(t, y(t))$ by $f(t_n, u_n)$ leads to our first numerical method for solving (5.3), the **explicit** (or **forward**) **Euler method**:

$$\begin{cases} u_0 = y_0, \\ u_{n+1} = u_n + h_n f(t_n, u_n), \qquad n = 0, 1, \ldots, N-1. \end{cases} \tag{5.9}$$

The method is called "explicit" because the right-hand side of (5.9) is independent of u_{n+1}, meaning that u_{n+1} can be computed from u_n in a straightforward way. As we shall see later, the price to pay for this convenient feature is that the method may be unstable unless certain conditions are met.

Instead of replacing $f(t, y(t))$ by $f(t_n, u_n)$ we could instead replace it by $f(t_{n+1}, u_{n+1})$, and this leads to a different numerical method, called the **implicit** (or **backward**) **Euler method**:

$$\begin{cases} u_0 = y_0, \\ u_{n+1} = u_n + h_n f(t_{n+1}, u_{n+1}), \qquad n = 0, 1, \ldots, N-1. \end{cases} \tag{5.10}$$

This method is called "implicit" because the right-hand side of (5.10) depends on u_{n+1}, so in general a nonlinear equation has to be solved at each time step to find u_{n+1} from u_n. This can be done using one of the methods described in §3, e.g. Newton's method, or another suitable fixed point iteration. (Question: can you suggest a sensible initial guess for the iteration?) While this increases the computational cost of the method compared to explicit Euler, we shall see that it brings certain advantages in terms of stability.

Another important method is obtained by replacing $f(t, y(t))$ by the *average* of $f(t_n, u_n)$ and $f(t_{n+1}, u_{n+1})$, which leads to the **Crank-Nicolson** or **trapezium rule method**:

$$\begin{cases} u_0 = y_0, \\ u_{n+1} = u_n + \frac{h_n}{2}\big(f(t_n, u_n) + f(t_{n+1}, u_{n+1})\big), \qquad n = 0, 1, \ldots, N-1. \end{cases} \tag{5.11}$$

Again the method is implicit, so has good stability properties. As we shall see, it also has better approximation properties than either forward or backward Euler.

Example 5.3.1. *Consider the IVP*

$$\begin{cases} y'(t) = -ty^2(t), \quad t > 0, \\ y(0) = 2, \end{cases} \tag{5.12}$$

the exact solution which is given by $y(t) = 2/(1+t^2)$.

We will solve this equation numerically on the interval $[0, 4]$ using both the forward and backward Euler methods. In both cases we use a constant time step $h = 0.2$, so the unknowns are approximations to the solution $y(t)$ at times $t = 0.2, 0.4, 0.6, \ldots$.

forward Euler

The function **feuler** *evaluates the forward Euler iteration at each time step.*

```
>> f=@(t,y)-t.*y.^2;
>> tmax=4; N=20; y0=2;
>> [t_fe, u_fe] = feuler(f,[0,tmax],y0,N)
```

The variable **t_fe** *contains the sequence of time levels t_n and the variable* **y_fe** *contains the associated approximations u_n computed by the method.*

backward Euler

The function **beuler** *solves the nonlinear equation obtained at each step of the backward Euler method using Newton's method. For this reason we must also supply the derivative $\frac{\partial f}{\partial y}$.*

```
>> dfdy=@(t,y)-2*t.*y;
>> [t_be, u_be] = beuler(f,dfdy,[0,tmax],y0,N)
```

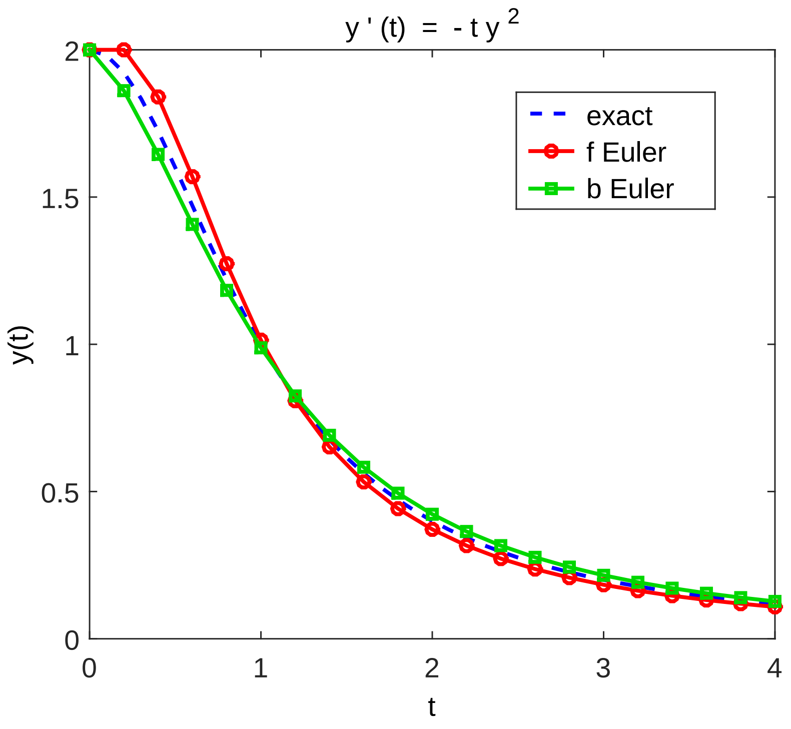

Figure 5.1: *Comparison between the exact solution and the approximate solutions obtained using the forward and backward Euler methods.*

Figure 5.1 shows the exact solution along with the two approximate solutions. This figure was obtained using the commands

```
>> t=[0:0.05:4]; y_ex=2./(1+t.^2);
>> plot(t,y_ex,'b',t_fe,u_fe,'ro-',t_be, u_be','go-')
>> legend('exact','f Euler','b Euler')
>> title('y^\prime(t) =- ty^2')
```

5.4 Discretizations from quadrature

One way of systematically deriving approximation schemes for (5.3) is to use numerical quadrature. First one integrates the equation $y'(t) = f(t, y(t))$ from t_n to t_{n+1} to obtain

$$y(t_{n+1}) - y(t_n) = \int_{t_n}^{t_{n+1}} f(t, y(t))dt. \tag{5.13}$$

Then one applies a numerical quadrature rule to approximate the integral on the right-hand side of (5.13) in terms of values of $f(t, y(t))$ at a discrete set of points in the interval $[t_n, t_{n+1}]$.

Replacing these values by their discrete counterparts leads to a numerical discretization of the differential equation.

A general class of quadrature rules for (5.13) is obtained by replacing the integrand $f(t, y(t))$ on $[t_n, t_{n+1}]$ by an appropriate polynomial approximation, which is then integrated exactly. The simplest polynomial approximation is where we replace $f(t, y(t))$ by a constant $f(t_*, y(t_*))$ for some point $t_* \in [t_n, t_{n+1}]$. The resulting integral is trivial to evaluate, and leads to the approximation

$$y(t_{n+1}) - y(t_n) \approx h_n f(t_*, y(t_*)).$$

Different choices of t_* lead to different methods. In particular, the choice $t^* = t_n$ leads to the forward Euler method (5.9), once we replace $y(t_n)$ and $y(t_{n+1})$ by u_n and u_{n+1}. Similarly, the choice $t^* = t_{n+1}$ leads to the backward Euler method (5.10).

More accurate approximations of the integral $\int_{t_n}^{t_{n+1}} f(t, y(t))dt$ can be obtained using higher-order polynomial approximations of the integrand. The *trapezoidal rule* uses the linear approximation

$$f(t, y(t)) \approx \left(f(t_n, y(t_n)) \frac{t_{n+1} - t}{t_{n+1} - t_n} + f(t_{n+1}, y(t_{n+1})) \frac{t - t_n}{t_{n+1} - t_n} \right),$$

which can be integrated exactly (exercise: check this!) to give the approximation

$$y(t_{n+1}) - y(t_n) \approx \frac{h_n}{2}\big(f(t_n, y(t_n)) + f(t_{n+1}, y(t_{n+1}))\big).$$

Replacing $y(t_n)$ and $y(t_{n+1})$ by u_n and u_{n+1} then leads to the Crank-Nicolson method (5.11).

5.5 One-step methods

The methods introduced above express u_{n+1} in terms of u_n, and as such are called *one-step methods*. There exists a large theory of more general *multi-step methods*, which express u_{n+1} in terms of u_{n+1-j}, $j = 0, 1, \ldots, k$ for some $k \geq 2$, but we shall not study these in this course.

We shall consider one-step methods of the general form

$$\begin{cases} u_0 = y_0, \\ u_{n+1} = u_n + h_n \Psi(t_n, u_n, u_{n+1}; h_n), \end{cases} \tag{5.14}$$

where n ranges from 0 to $N - 1$ and where $\Psi(\cdot, \cdot, \cdot; \cdot)$ is a continuous function of its arguments.

Example 5.5.1. *For forward Euler,*

$$\Psi(t_n, u_n, u_{n+1}; h_n) = f(t_n, u_n).$$

For backward Euler,

$$\Psi(t_n, u_n, u_{n+1}; h_n) = f(t_{n+1}, u_{n+1}) = f(t_n + h_n, u_{n+1}).$$

For Crank-Nicolson,

$$\Psi(t_n, u_n, u_{n+1}; h_n) = \frac{1}{2}\big(f(t_n, u_n) + f(t_{n+1}, u_{n+1})\big) = \frac{1}{2}\big(f(t_n, u_n) + f(t_n + h_n, u_{n+1})\big).$$

5.5.1 Truncation error

For a one-step method of the form (5.14) we define the *truncation error* T_n by

$$\boxed{T_n = \frac{y(t_{n+1}) - y(t_n)}{h_n} - \Psi(t_n, y(t_n), y(t_{n+1}); h_n), \quad n = 0, \ldots, N-1.}$$

Note that the truncation error involves the *exact* values $y(t_n)$ and $y(t_{n+1})$, not their numerical approximations. It is a measure of the extent to which the numerical method approximates the differential operator $y'(t) - f(t, y(t))$. Precisely, rewriting the definition of T_n as

$$T_n = \frac{1}{h_n}\Big(y(t_{n+1}) - \big(y(t_n) + h_n\Psi(t_n, y(t_n), y(t_{n+1}); h_n)\big)\Big)$$

reveals that T_n is $1/h_n$ times the residual obtained by plugging the exact solution into the recurrence relation (5.14).

For simplicity, let us now suppose that we are working on a uniform mesh, with $t_n = nh$, $n = 0, \ldots, N$, where $h = t_{\max}/N$, so that $h_n = h$ for each n.

Definition 5.5.2. *We say the method is* uniformly consistent *if*

$$T(h) = \max_{n=0,\ldots,N-1} |T_n|$$

tends to zero as $h \to 0$ (equivalently, as $N \to \infty$). To quantify consistency, we say the method is of consistent of order $p > 0$ if $T(h) = O(h^p)$ as $h \to 0$ (equivalently, as $N \to \infty$).

Example 5.5.3. *For the forward Euler method with constant step size h we have*

$$T_n = \frac{y(t_{n+1}) - y(t_n)}{h} - f(t_n, y(t_n)).$$

Assume that y is twice differentiable with y'' bounded on $[0, t_{\max}]$. Then by Taylor's theorem

$$y(t_{n+1}) = y(t_n) + hy'(t_n) + \frac{h^2}{2}y''(\xi_n),$$

for some $\xi_n \in [t_n, t_{n+1}]$. Since $y'(t_n) = f(t_n, y(t_n))$ (from the differential equation) we see that

$$|T_n| = \left| y'(t_n) - f(t_n, y(t_n)) + \frac{h}{2} y''(\xi_n) \right| = \left| \frac{h}{2} y''(\xi_n) \right| \leq \frac{h}{2} \sup_{\xi \in [0, t_{\max}]} |y''(\xi)|,$$

so the method is consistent of order one (i.e. first order).

Example 5.5.4. *The backward Euler method is also consistent of order one (i.e. first-order) under suitable assumptions on $y(t)$. For this, note that now*

$$T_n = \frac{y(t_{n+1}) - y(t_n)}{h} - f(t_{n+1}, y(t_{n+1})).$$

We expand $y(t_{n+1})$ as in the previous example. But now we also have to expand

$$f(t_{n+1}, y(t_{n+1})) = y'(t_{n+1}) = y'(t_n + h) = y'(t_n) + hy''(\eta_n),$$

for some $\eta_n \in [t_n, t_{n+1}]$, so that, provided y is twice differentiable with y'' bounded,

$$|T_n| = h \left| \frac{1}{2} y''(\xi_n) - y''(\eta_n) \right| \leq \frac{3h}{2} \sup_{\xi \in [0, t_{\max}]} |y''(\xi)|.$$

Example 5.5.5. *The Crank-Nicolson method (5.11) is consistent of order two (i.e. second order) with $T(h) = O(h^2)$ under suitable assumptions on y. The truncation error is now*

$$T_n = \frac{y(t_{n+1}) - y(t_n)}{h} - \frac{1}{2} \left(f(t_n, y(t_n)) + f(t_{n+1}, y(t_{n+1})) \right),$$

and to prove $T(h) = O(h^2)$ we proceed as for backward Euler, except that we expand to higher order than before. Specifically, combining the expansions

$$y(t_{n+1}) = y(t_n) + hy'(t_n) + \frac{h^2}{2} y''(t_n) + \frac{h^3}{6} y'''(\xi_n)$$

and

$$f(t_{n+1}, y(t_{n+1})) = y'(t_{n+1}) = y'(t_n + h) = y'(t_n) + hy''(t_n) + \frac{h^2}{2} y'''(\eta_n)$$

shows that, assuming y is three times differentiable with y''' bounded,

$$|T_n| = h^2 \left| \frac{1}{6} y'''(\xi_n) - \frac{1}{4} y'''(\eta_n) \right| \leq \frac{5h^2}{12} \sup_{\xi \in [0, t_{\max}]} |y'''(\xi)|.$$

The truncation error is a measure of how well the numerical method approximates the differential equation on a local level. To investigate the global behaviour of the numerical solution u_n we consider the concepts of *stability* and *convergence*.

5.5.2 Zero stability

The concept of stability concerns the behaviour of the numerical method in the presence of small perturbations (for example those caused by the rounding errors inherent in finite precision arithmetic). The first type of stability we consider relates to the behaviour of the numerical method on a fixed time interval $[0, t_{\max}]$ as the mesh size h tends to zero, and is called *zero-stability.* Given a one-step method

$$\begin{cases} u_0 = y_0, \\ u_{n+1} = u_n + h\Psi(t_n, u_n, u_{n+1}; h), \end{cases} \tag{5.15}$$

we introduce the perturbed version

$$\begin{cases} v_0 = y_0 + \delta_0, \\ v_{n+1} = v_n + h\Psi(t_n, v_n, v_{n+1}; h) + h\delta_{n+1}, \end{cases} \tag{5.16}$$

where δ_n, $n = 0, 1, \ldots, N$, are some small perturbations. We can then ask, for perturbations of a given size, how the difference between the solutions u_n and v_n behaves in the limit $h \to 0$.

Definition 5.5.6. *The numerical method* (5.15) *for the approximation of* (5.3) *is said to be* zero-stable *if there exist $h_* > 0$ and $C \geq 0$ such that if $h < h_*$ then the solution of the perturbed problem* (5.16) *satisfies*

$$\max_{n=0,\ldots,N} |v_n - u_n| \leq C \max_{n=0,\ldots,N} |\delta_n|. \tag{5.17}$$

It is essential that the constant C in (5.17) be independent of h for all $h < h_*$. However it may depend on other quantities, such as the final time $t_{\max}$ or on the function $f(t, y)$ defining the problem. See for instance Theorem 5.5.7 below.

Note that one consequence of zero-stability is that it ensures that the computed solution is not overly sensitive to perturbations of the data, which is necessary to control the accumulation of errors introduced by working in finite precision arithmetic when the method is implemented on a computer. A method that is not zero-stable is not suitable for practical computation, because the numerical solution can become increasingly polluted by rounding errors in the evaluation of y_0 and $f(t_n, u_n)$.

The following theorem provides sufficient conditions for a one-step method to be zero-stable, namely that the increment function Ψ satisfies a uniform Lipschitz condition.

Theorem 5.5.7. *Suppose that there exist* $\widetilde{h} > 0$, $L_1 \geq 0$ *and* $L_2 \geq 0$ *such that*

$$|\Psi(t, u, v; h) - \Psi(t, u', v'; h)| \leq L_1|u - u'| + L_2|v - v'| \tag{5.18}$$

for all $0 < h < \widetilde{h}$, *all* $t \in [0, t_{max}]$, *and all* u, u', v, $v' \in \mathbb{R}$. *Then, for any* $\rho \in (0, 1)$, *the one-step method* (5.15) *is zero-stable with constants*

$$h_* = \min\left(\widetilde{h}, \frac{\rho}{L_2}\right), \qquad C = \left(1 + \frac{1}{L_1 + L_2}\right) \exp\left(\frac{t_{\max}(L_1 + L_2)}{1 - \rho}\right).$$

Proof. By the definitions of the numerical methods (5.15) and (5.16), the difference $w_n = v_n - u_n$ satisfies $w_0 = \delta_0$ and, for each $n = 0, \ldots, N - 1$, the recurrence relation

$$w_{n+1} = w_n + h\big(\Psi(t_n, v_n, v_{n+1}; h) - \Psi(t_n, u_n, u_{n+1}; h)\big) + h\delta_{n+1}.$$

Applying the triangle inequality and the Lipschitz condition (5.18) shows that, for $0 < h < \widetilde{h}$,

$$\begin{aligned} |w_{n+1}| &\leq |w_n| + h\big|\Psi(t_n, v_n, v_{n+1}; h) - \Psi(t_n, u_n, u_{n+1}; h)\big| + h|\delta_{n+1}| \\ &\leq |w_n| + hL_1|w_n| + hL_2|w_{n+1}| + h|\delta_{n+1}|, \end{aligned} \tag{5.19}$$

Hence

$$(1 - hL_2)\,|w_{n+1}| \leq (1 + hL_1)\,|w_n| + h\delta. \tag{5.20}$$

Now, let $\rho \in (0, 1)$ and assume that $0 < h < \rho/L_2$. Then (5.20) is equivalent to

$$|w_{n+1}| \leq \underbrace{\frac{(1 + hL_1)}{(1 - hL_2)}}_{=a}|w_n| + \underbrace{\frac{h}{(1 - hL_2)}}_{=b}|\delta_{n+1}|. \tag{5.21}$$

Note that $a \geq 1$. We then obtain by an induction argument *(exercise!)* that

$$|w_n| \leq a^n|\delta_0| + b\sum_{j=1}^{n} a^{n-j}|\delta_j| \quad \forall n = 0, \ldots, N. \tag{5.22}$$

Let $\delta = \max_{n=0,\ldots,N}|\delta_n|$, so that

$$|w_n| \leq \left(a^n + b\frac{a^n - 1}{a - 1}\right)\delta \leq \left(1 + \frac{b}{a - 1}\right)a^n\delta,$$

where we use the geometric sum $\sum_{j=1}^{n} a^{n-j} = \sum_{k=0}^{n-1} a^k = \frac{a^n-1}{a-1}$ and use the simple inequalities $a^n - 1 \leq a^n$ and $a \geq 1$ in the second inequality above. Observe that $\frac{b}{a-1} = \frac{1}{L_1+L_2}$. Also, using the inequality $(1 + x)^n \leq \mathrm{e}^{nx}$ for all $x \geq 0$, we get

$$\begin{aligned} a^n &= \left(\frac{(1 + hL_1)}{(1 - hL_2)}\right)^n = \left(1 + \frac{h(L_1 + L_2)}{1 - hL_2}\right)^n \\ &\leq \exp\left(\frac{nh(L_1 + L_2)}{1 - hL_2}\right) \leq \exp\left(\frac{t_{\max}(L_1 + L_2)}{1 - \rho}\right) \qquad \forall n = 0, \ldots, N, \end{aligned}$$

since $nh \le Nh = t_{\max}$ and since $1 - hL_2 > 1 - \rho > 0$. Therefore we have shown that for all $h < h_*$, we have

$$\max_{n=0,\ldots,N} |w_n| \le C \max_{n=0,\ldots,N} |\delta_n|$$

with constant $C = \left(1 + \frac{1}{L_1+L_2}\right) \exp\left(\frac{t_{\max}(L_1+L_2)}{1-\rho}\right)$. Noting that C is independent of n and of h, we see that the one-step method is zero-stable. □

Example 5.5.8. *If f is uniformly Lipschitz on $\mathbb{R}$, i.e.*

$$|f(t,v) - f(t,w)| \le L_f |v - w|, \qquad \forall t \in [0, t_{\max}] \text{ and } \forall v, w \in \mathbb{R},$$

then the forward Euler, backward Euler and Crank-Nicolson methods are all zero-stable, since they satisfy the uniform Lipschitz condition (5.18) of Theorem 5.5.7 with $(L_1, L_2) = (L_f, 0)$ (FE), $(L_1, L_2) = (0, L_f)$ (BE), and $(L_1, L_2) = (L_f/2, L_f/2)$ (CN).

We end this section by remarking that while the constant C in (5.17) must be independent of h (equivalently, of N), it may depend on the length $t_{\max}$ of the time interval on which we are solving (5.3). In particular it may happen that $C \to \infty$ as $t_{\max} \to \infty$, as is the case in the proof of Theorem 5.5.7.

5.5.3 Convergence

Not only do we want our numerical solution u_n to be stable under perturbations of the data, we want it to be a good approximation to the exact solution y of the original IVP (5.3). Define the *global error* e_n to be the difference between the exact solution and the numerical approximation, i.e.

$$e_n = y(t_n) - u_n.$$

Definition 5.5.9. *We say that the method is* uniformly convergent *if and only if*

$$E(h) = \max_{n=0,\ldots,N} |e_n|$$

tends to zero as $h \to 0$ (equivalently, as $N \to \infty$). We say the method has convergence order *$p > 0$ if and only if $E(h) = O(h^p)$ as $h \to 0$ (equivalently, as $N \to \infty$).*

The following theorem is a part of the celebrated *Lax-Richtmyer equivalence principle*, which can be summarized as "stability plus consistency is equivalent to convergence" (we analyse here only the forward implication that consistency and stability imply convergence). The theorem provides an explicit link between the maximum error $E(h)$ and the maximum truncation error $T(h)$. We can then deduce the order of convergence of the method by analysing the order of the truncation error, as we saw for instance in Example 5.5.3.

Theorem 5.5.10. *Let f satisfy the assumptions of Theorem 5.2.2 (Picard's theorem). Suppose that the one-step method (5.15) is zero-stable for the IVP, in the sense of Definition 5.5.6, and uniformly consistent, in the sense of Definition 5.5.2. Then the method is uniformly convergent, and*

$$E(h) = O(T(h)) \quad \text{as } h \to 0. \tag{5.23}$$

If the method is consistent of order $p > 0$, namely $T(h) = O(h^p)$ as $h \to 0$, then also $E(h) = O(h^p)$ as $h \to 0$ and the method has convergence order p.

Proof. The key to the proof is to note that the exact solution $v_n = y(t_n)$ satisfies the perturbed equations (5.16) with $\delta_0 = 0$ and $\delta_{n+1} = T_n$, $n = 0, 1, \ldots, N-1$. Thus, with h_* and C denoting the constants from Definition 5.5.6, if $h < h_*$ then zero-stability (with $\delta = T(h)$) gives that $|e_n| = |v_n - u_n| \le CT(h)$, $n = 0, 1, \ldots, N$, which proves the general result (5.23). □

Remark 5.5.11. *In particular, revisiting the proof of Theorem 5.5.7, we see that under those hypotheses, we get $\delta_0 = 0$ and $\delta_{n+1} = T_n$ for all $n = 0, \ldots, N-1$. Starting from (5.22), we can then obtain the improved inequality*

$$E(h) \le \frac{1}{L_1 + L_2}\left[\exp\left(\frac{t_{\max}(L_1 + L_2)}{1 - \rho}\right) - 1\right] T(h) \tag{5.24}$$

for all $h < h_*$ (exercise!).

Example 5.5.12. *Combining Examples 5.5.3 and 5.5.8, we have that if f is uniformly Lipschitz then the forward and backward Euler methods are first-order convergent, and the Crank-Nicolson method is second-order convergent.*

5.5.4 Absolute stability

So far we have considered the approximation of the IVP (5.3) on bounded time intervals $[0, t_{\max}]$. Often we wish to integrate the IVP on a very long time interval, typically to recover some asymptotic behaviour of the solution. In this case we want a method that gives accurate approximation over long time intervals, using a time step h that remains relatively large, so as to keep the computational cost at an acceptable level.

An indication that this might not always be straightforward is provided by our convergence analysis from the previous section (specifically, Theorem 5.5.10). This showed that, for methods satisfying the Lipschitz assumption of Theorem 5.5.7, for h sufficiently small the global error satisfies (5.24). Thus, if the method is uniformly consistent (i.e. $T(h) \to 0$ as $h \to 0$), then the numerical solution converges uniformly to the exact solution as $h \to 0$, on the fixed interval $[0, t_{\max}]$. However, because of the presence of the factor $\exp\left(\frac{t_{\max}(L_1+L_2)}{1-hL_2}\right)$, if we fix $h > 0$ and send $t_{\max} = Nh \to \infty$ then the right-hand side of (5.24) increases exponentially.

For some problems and some methods this bound is descriptive of the behaviour of the errors, which become exponentially large as the time window $[0, t_{\max}]$ is increased. In such scenarios, it becomes either necessary to consider alternative methods or to use very small timesteps h in order to bring the errors to a desired tolerance, thus leading to increasingly expensive computations as $t_{\max}$ increases.

However, for other problems and some methods, it can happen that the error bound in (5.24) is not sharp (i.e. overly pessimistic) and that the numerical solution is perfectly well behaved. To correctly capture long-time behaviour, we need to consider methods which enjoy an additional stability property beyond zero-stability, namely *absolute stability*. This concept is defined with reference to the following model problem:

$$\begin{cases} y'(t) = \lambda y(t), \quad t \in (0, \infty), \\ y(0) = 1, \end{cases} \tag{5.25}$$

where we allow λ to be complex, i.e. $\lambda \in \mathbb{C}$. The exact solution is given by $y(t) = e^{\lambda t}$. If $\mathrm{Re}(\lambda) < 0$ then the solution decays exponentially as t increases, and we have that $y(t) \to 0$ as $t \to \infty$. The concepts of region of absolute stability and A-stable method characterize the behaviour of numerical methods with regards to this property.

Definition 5.5.13 (Region of absolute stability and A-stable method)**.** *The* region of absolute stability *of a one-step method* (5.15) *for the IVP* (5.25) *is the set of values λh in the complex plane for which the numerical solutions satisfy*

$$u_n \to 0 \quad \textit{as } n \to \infty. \tag{5.26}$$

The method is said to be A-stable *if the region of absolute stability contains the complex left half-plane, i.e. the set of all points $z = \lambda h$ with* $\mathrm{Re}(z) < 0$*.*

Note that since we are holding h fixed in (5.26), the limit $n \to \infty$ corresponds to the limit $t_n \to \infty$. Thus (5.26) relates to the large time behaviour of the numerical solution. Also, note that since in all cases $h > 0$ is real, we have $\mathrm{Re}(\lambda h) < 0$ if and only if $\mathrm{Re}(\lambda) < 0$. Therefore, a method is A-stable if and only if (5.26) holds for all λ in the complex left half-plane, for any $h > 0$.

For methods that are not A-stable, the asymptotic behaviour at large times then typically depends on the choice of h relative to λ. We often find in practice that a large value of λ requires a significant reduction in h (and thus a large increase in the computation cost) in order to recover (5.26).

The simple nature of the problem (5.25) means that it is often fairly straightforward to investigate the absolute stability of a one-step method by computing the numerical solution by hand. For example, for the forward Euler method we find the numerical solution

$$u_0 = 1, \quad u_{n+1} = u_n(1 + \lambda h) = (1 + \lambda h)^{n+1}, \quad n \geq 0, \tag{5.27}$$

and, consequently, $\lim_{n\to\infty} u_n = 0$ if and only if

$$|1 + h\lambda| < 1 \tag{5.28}$$

Thus the region of absolute stability of the Forward Euler method is the interior of the disc of radius 1 centred on -1 in the complex plane. Thus the Forward Euler method is not A-stable. For a general $\lambda \in \mathbb{C}$ with $\mathrm{Re}(\lambda) < 0$, we see that to obtain the correct asymptotic behaviour we must take h sufficiently small, in particular

$$h < \frac{2|\mathrm{Re}(\lambda)|}{|\lambda|^2}. \qquad \text{(exercise!)}$$

For instance, in the case where $\lambda < 0$ is real, this simplifies to $h < 2/|\lambda|$. If on the other hand h is not chosen small enough, then it is possible to have $|1 + h\lambda| > 1$ in which case the numerical solutions will grow unboundedly as $n \to \infty$.

By contrast, the backward Euler method and the Crank-Nicolson method are A-stable methods. Indeed, if we consider the backward Euler method applied to (5.25) we get $u_{n+1} = u_n + \lambda h u_{n+1}$ and therefore

$$u_{n+1} = \left(\frac{1}{1 - \lambda h}\right)^{n+1},$$

which tends to zero as $n \to \infty$ for every $h > 0$ for any λ in the complex left-half plane, since $|1 - \lambda h| > 1$ as a result of the fact that $1 - \mathrm{Re}(h\lambda) > 1$ whenever $\mathrm{Re}(\lambda) < 0$. Similarly, for the Crank-Nicolson method we find that

$$u_{n+1} = \left[\left(1 + \frac{h\lambda}{2}\right) \Big/ \left(1 - \frac{h\lambda}{2}\right)\right]^{n+1},$$

which again tends to zero as $n \to \infty$ for every $h > 0$ as a result of the fact *(exercise!)* that

$$\frac{|1 + \frac{h\lambda}{2}|}{|1 - \frac{h\lambda}{2}|} < 1 \quad \forall \lambda \in \mathbb{C} \quad \text{s.t. } \mathrm{Re}(\lambda) < 0$$

Example 5.5.14. *We solve the model problem (5.25) with $\lambda = -2$ in the interval $[0, 10]$ using the forward and backward Euler methods with $h = 0.9$ and $h = 1.1$. In the case $h = 0.9$ we use the following Matlab commands:*

```
>> f=@(t,y)-2*y; tmax=10.8; N=12; y0=1;
% We take tmax=10.8 so that we have a whole number of subintervals.
>> [t_fe, u_fe] = feuler(f,[0,tmax],y0,N)
>> dfdy=@(t,y)-2;
>> [t_be, u_be] = beuler(f,dfdy,[0,tmax],y0,N)
>> t=[0:0.1:10]; y_ex=exp(-2*t);
>> plot(t,y_ex,'b',t_fe,u_fe,'ro-',t_be, u_be','go-')
>> xlim([0,10])
>> legend('exact','f Euler','b Euler')
>> title('y^\prime =-2y(t)')
```

Figure 5.2 shows the solutions obtained for $h = 0.9$ (left) and $h = 1.1$ (right) as well as the exact solution e^{-2t}. Note that, in accordance with the absolute stability theory, the forward Euler solution decays with increasing t for the case $h = 0.9 < 1$ but grows with increasing t for the case $h = 1.1 > 1$, and the backward Euler solution decays for both cases.

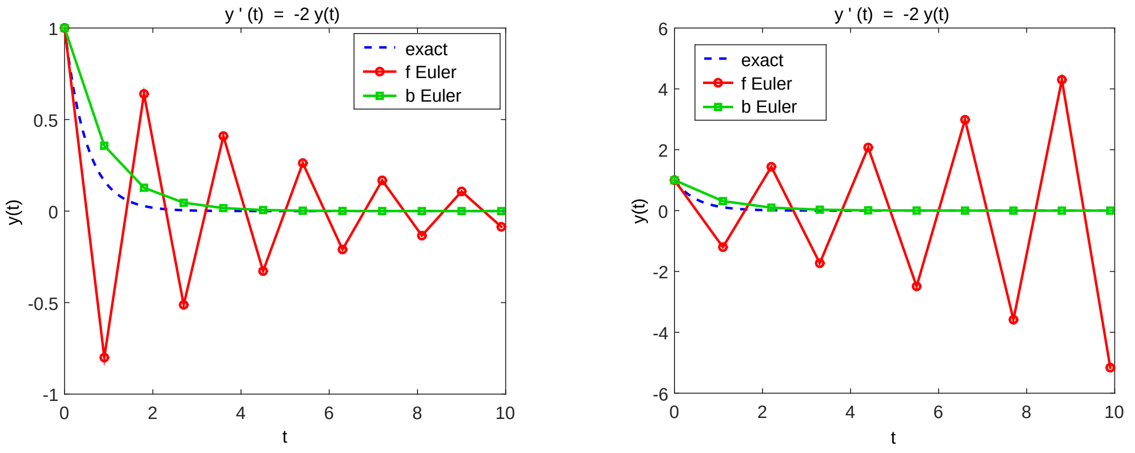

Figure 5.2: *Comparison between the solutions of (5.25) with $\lambda = -2$ obtained using the forward and backward Euler methods with $h = 0.9$ (left) and $h = 1.1$ (right).*

Remark 5.5.15. *The analysis above can be generalised to the case where λ is a negative function of t in (5.25). But if one wishes to use a uniform time step h then $|\lambda|$ must be replaced by $\max_{t\in[0,\infty)} |\lambda(t)|$ in the stability conditions. If one uses a variable time step h_n, taking into account the local behaviour of λ on each interval $[t_n, t_{n+1}]$, then the stability condition can be relaxed. One approach consists of an adaptive forward Euler method of the following form: Let $u_0 = y_0$ and $h_0 = 2\tilde{h}/|\lambda(t_0)|$ for some $\tilde{h} < 1$; and for $n = 0, 1, \ldots$ set*

$$\begin{aligned} t_{n+1} &= t_n + h_n, \\ u_{n+1} &= u_n + h_n\lambda(t_n)u_n, \\ h_{n+1} &= 2\tilde{h}/|\lambda(t_{n+1})|. \end{aligned} \tag{5.29}$$

The requirement that $\tilde{h} < 1$ ensures that the method is absolutely stable, in the sense that the asymptotic decay is correctly captured by the numerical solution.

At first glance, the concept of absolute stability appears to relate only to the very special IVP (5.25), for which one can easily write down an analytic solution. However a further analysis of absolute stability shows that it is relevant to much more general initial value problems.

To explore this idea, let us return to the general initial value problem (5.3), and the pair of numerical solutions u_n and v_n generated by the methods (5.15) and (5.16). Recall that (5.16) is supposed to represent a perturbed version of (5.15) in which small errors (e.g. due to rounding) are introduced at each step of the method. When studying zero-stability we considered the behaviour of the difference $u_n - v_n$ for fixed $t_{\max}$ as $h \to 0$. We now consider the behaviour of this difference for fixed h and $t_{\max} \to \infty$, and how this relates to absolute stability.

For simplicity we shall restrict our attention to the Forward Euler method. In this case, we recall from the proof of Theorem 5.5.7 that the difference $w_n = v_n - u_n$ satisfies the recurrence relation

$$w_{n+1} = w_n + h\big(f(t_n, v_n; h) - f(t_n, u_n; h)\big) + hT_n. \tag{5.30}$$

To obtain a sharper bound we will make a stronger assumption about the function $f(t, y)$, namely that it is differentiable with respect to its second argument and that there exist positive constants $\alpha \geq \beta > 0$ such that

$$-\alpha \leq \frac{\partial f}{\partial y}(t, y) \leq -\beta, \quad \forall t, \forall y.$$

By the mean value theorem one then has that

$$w_{n+1} = (1 + h\lambda_n)w_n + hT_n, \tag{5.31}$$

where $\lambda_n = \frac{\partial f}{\partial y}(t_n, z_n)$ for some z_n between u_n and v_n. Then, taking absolute values and applying the triangle inequality, we obtain the bound

$$|w_{n+1}| \leq |1 + h\lambda_n||w_n| + h|T_n|. \tag{5.32}$$

We now assume that $h < 2/\alpha$ which implies that

$$|1 + h\lambda_n| \leq \underbrace{\max\left(|1 - h\alpha|, |1 - h\beta|\right)}_{=c} \tag{5.33}$$

where we note that $c < 1$ owing to the assumption on $h < 2/\alpha$. Since here we have $w_0 = 0$, we then find by induction that, for all $n \geq 0$,

$$|w_{n+1}| \leq h\sum_{j=0}^{n} c^{n-j}|T_j| \leq \frac{h}{1-c}\max_{j=0,\ldots,n}|T_j|. \tag{5.34}$$

Crucially, the constant in the second inequality above is independent of n, and thus gives a meaningful bound on the error even in the large time limit. Notice that the assumption $h < 2/\alpha$ is used here to ensure that $h\lambda_n$ falls within the region of absolute stability of the Forward Euler method.

Notice that although the constant $\frac{h}{1-c}$ is independent of n, it is apparently still dependent on h. However, this dependence disappears for h small enough, since if $h < 2/(\alpha + \beta)$, then $c = 1 - h\beta$ so that $\frac{h}{1-c} = \frac{1}{\beta}$. In this case, we get, for all $n \geq 0$,

$$|w_{n+1}| \leq \frac{1}{\beta}\max_{j=0,\ldots,n}|T_j|. \tag{5.35}$$

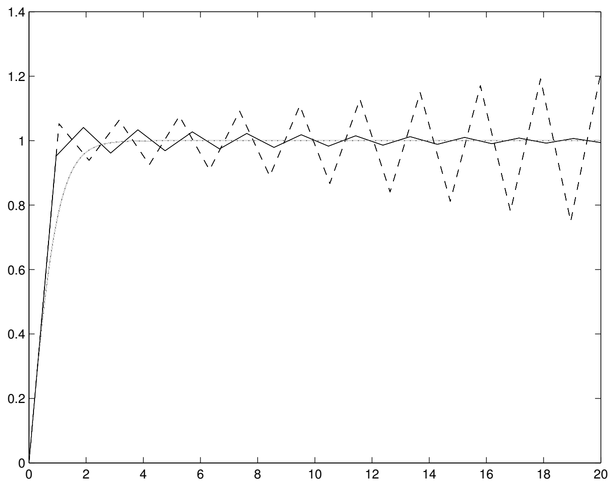

Figure 5.3: Numerical solutions for the problem (5.36) obtained using the forward Euler method and $h = 20/19$ (dashed line) and $h = 20/21$ (full line), along with the exact solution (dotted line).

Example 5.5.16. *Consider the IVP*

$$\begin{cases} y'(t) = \arctan(3y) - 4y + t, \quad t > 0, \\ y(0) = 1. \end{cases}$$

The derivative $f_y = \partial f/\partial y = 3/(1+9y^2) - 4$ is negative, with $-4 < f_y < -1$ for all $t > 0$ and $y \in \mathbb{R}$. Hence $\alpha = 4$ and $\beta = 1$ and so the forward Euler method is stable to perturbations provided $h < 2/4 = 1/2$.

Example 5.5.17. *Consider the IVP*

$$\begin{cases} y' = 1 - y^2, \quad t > 0, \\ y(0) = 0, \end{cases} \tag{5.36}$$

the exact solution to which is $y(t) = (\mathrm{e}^{2t} - 1)/(\mathrm{e}^{2t} + 1) = \tanh t$. Note that $f_y = \partial f/\partial y = -2y$, which is unbounded for $y \in \mathbb{R}$. However, since the exact solution y satisfies $y(t) \in (0, 1)$ for $t > 0$, we have that $f_y(t, y(t)) \in (-2, 0)$ for all $t > 0$. Because of this we might try taking $\alpha = 2$, which would suggest that for the forward Euler method to be stable to perturbations we need to take the step size $h < 2/2 = 1$. In Figure 5.3 we show the exact solution (dotted line), along with the approximate solutions obtained using $h = 20/19 < 1$ (dashed line) and $h = 20/21 > 1$ (continuous line). One can clearly see the error in the case $h < 1$ decaying with increasing t, whereas the error in the case $h > 1$ appears to be blowing up. These results suggest that in this particular case our proposed value of the critical step size is accurate, despite the fact that (i) it was computed only by considering the behaviour of f_y on the solution trajectory; and (ii) on this trajectory $\beta = 0$.

Example 5.5.18. (Example 5.3.1 cont'd) *We return to the IVP (5.12) considered in Example 5.3.1. There we used a step size of $h = 0.2$ and the forward and backward Euler* *solutions on the interval $[0,4]$ plotted in Figure 5.1 appeared to be roughly equivalent in their accuracy. However, when the step size is increased the quality of the forward Euler approximation rapidly deteriorates due to the lack of absolute stability. Figure 5.4 shows the solutions obtained using the forward and backward Euler methods with $h = 0.8$.*

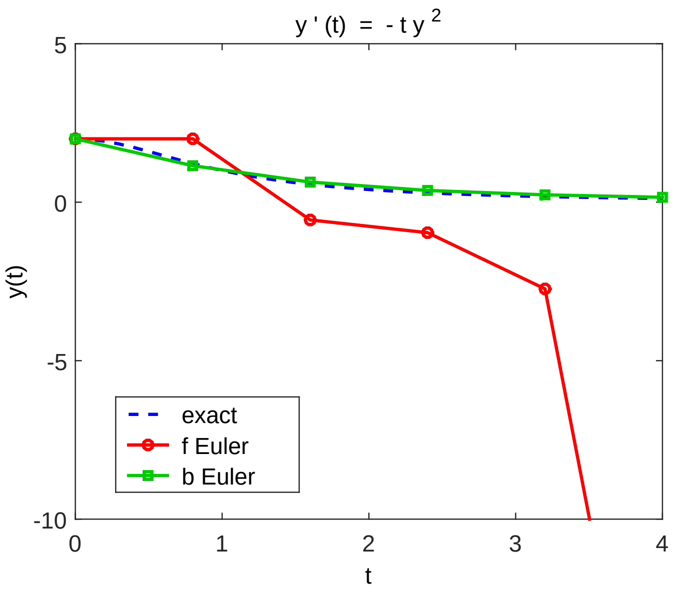

Figure 5.4: *Comparison between the solutions of* (5.12) *obtained using the forward and backward Euler methods with $h = 0.8$.*

5.6 Runge-Kutta methods

Despite its rather delicate stability properties, the forward Euler method is popular because it is an explicit method which is simple and cheap to implement. However, we noted in Example 5.5.12 that it is only first order convergent. It is possible to derive more accurate (albeit slightly more expensive) explicit one-step methods by evaluating $f(\cdot,\cdot)$ not just at (t_n, u_n) but also at other points intermediate between (t_n, u_n) and (t_{n+1}, u_{n+1}). Such methods are called *Runge-Kutta* methods.

We recall from §5.4 that the methods considered so far can all be derived from the integral formula

$$y(t_{n+1}) - y(t_n) = \int_{t_n}^{t_{n+1}} f(t, y(t))dt. \tag{5.37}$$

In particular, using the trapezoidal rule to approximate the integral we obtained the Crank-Nicolson method

$$u_{n+1} - u_n = \frac{h_n}{2}\left[f(t_n, u_n) + f(t_{n+1}, u_{n+1})\right], \tag{5.38}$$

an implicit scheme which was shown to be $\mathcal{A}$-stable and second order accurate. It is possible to modify the scheme to render it explicit, but still conserve the second order accuracy (albeit at the expense of losing $\mathcal{A}$-stability). Indeed, by replacing the unknown u_{n+1} in the right-hand side of (5.38) by the result of a forward Euler step, we obtain so-called **improved Euler method**, or **Heun's method**:

$$u_{n+1} - u_n = \frac{h_n}{2}\left[f(t_n, u_n) + f(t_{n+1}, u_n + h_n f(t_n, u_n))\right]. \tag{5.39}$$

Heun's method may also be written in the form

$$u^n \to \begin{cases} p_1 = f(t_n, u_n) \\ p_2 = f(t_{n+1}, u_n + h_n p_1) \\ u_{n+1} = u_n + \dfrac{h_n}{2}(p_1 + p_2). \end{cases}$$

A different method can be obtained by returning to (5.37) and using the midpoint rule rather than the trapezoidal rule, i.e. using a constant approximation to the integrand with $t_* = t_{n+1/2} = t_n + h_n/2 = (t_n + t_{n+1})/2$ in the notation of §5.4. This gives

$$u_{n+1} - u_n = h_n\, f(t_{n+\frac{1}{2}}, u_{n+\frac{1}{2}}), \tag{5.40}$$

where $u_{n+1/2}$ represents some approximation of the solution $y(t_{n+1/2})$. Calculating $u_{n+1/2}$ using half a forward Euler step, i.e. setting

$$u_{n+\frac{1}{2}} = u_n + \frac{h_n}{2} f(t_n, u_n),$$

gives the **modified Euler method**:

$$u_{n+1} - u_n = h_n\, f(t_{n+\frac{1}{2}}, u_n + \frac{h_n}{2} f(t_n, u_n)), \tag{5.41}$$

which can also be written in the form

$$u^n \to \begin{cases} p_1 = f(t_n, u_n) \\ p_2 = f(t_n + \dfrac{h_n}{2}, u_n + \dfrac{h_n}{2} p_1) \\ u_{n+1} = u_n + h_n p_2. \end{cases}$$

Heun's method and the modified Euler method are both consistent of order 2, provided that y and f are sufficiently smooth. For instance, consider Heun's method, where we shall assume that y is three times differentiable with y''' bounded, and that f is 2 twice continuously differentiable in both arguments with, then we have (using the shorthand notation $y_n = y(t_n)$ and $f_n = f(t_n, y_n)$, etc)

$$\frac{y_{n+1} - y_n}{h} = y'_n + \frac{h}{2} y''_n + \frac{h^2}{6} y'''(z_n)$$

for some $z_n \in (t_n, t_{n+1})$. We also have by Taylor's theorem

$$f(t_n + h, y_n + hf_n) = f_n + h\frac{\partial f}{\partial t}|_{(t_n,y_n)} + hf_n\frac{\partial f}{\partial y}|_{(t_n,y_n)} + R_2,$$

where the remainder term R_2 satisfies $R_2 = O(h^2)$ as $h \to 0$. Then, by differentiating the equation $y'(t) = f(t, y(t))$ we get

$$y_n'' = \frac{\partial f}{\partial t}|_{(t_n,y_n)} + f_n\frac{\partial f}{\partial y}|_{(t_n,y_n)}$$

And therefore the truncation error satisfies after some simplification *(exercise!)*

$$T_n = \frac{h^2}{6}y'''(z_n) - \frac{1}{2}R_2 = O(h^2) \quad \text{as } h \to 0.$$

This shows that Heun's method is consistent of order two.

Example 5.6.1. *Consider the IVP*

$$\begin{cases} y'(t) = (-0.1 + \cos t)\, y(t), & t > 0, \\ y(0) = 1, \end{cases} \tag{5.42}$$

which has exact solution $y(t) = \mathrm{e}^{-0.1t+\sin t}$.

We can solve this problem using the forward Euler method and Heun's method on the interval $[0, 12]$*, with* $h = 0.4$*, using the following Matlab code:*

```
>> f=@(t,y)(cos(t)-0.1)*y;
>> tmax=12; N=30; y0=1;
>> [t_fe, u_fe] = feuler(f,[0,tmax],y0,N)
>> [t_heun, u_heun] = heun(f,[0,tmax],y0,N)
```

In Figure 5.5 we present the solutions obtained using the two methods, along with the exact solution. It is clear that for this fixed step size the approximation produced by Heun's method is much more accurate than that of the forward Euler method. Nevertheless if the step size is reduced in the latter method it will still converge (albeit slowly) to the exact solution, as can be seen in Figure 5.6 where the approximations obtained for $h = 0.4, 0.2, 0.1, 0.05$ *are reported.*

```
>> t=linspace(0,tmax,100);
>> y_ex = exp(-0.1*t+sin(t));
>> plot(t,y_ex,'b--'); hold on
>> N=30;
>> for i=1:4
     [t_fe, u_fe] = feuler(f,[0,tmax],y0,N);
     plot(t_fe,u_fe)
     N=2*N;
   end
```

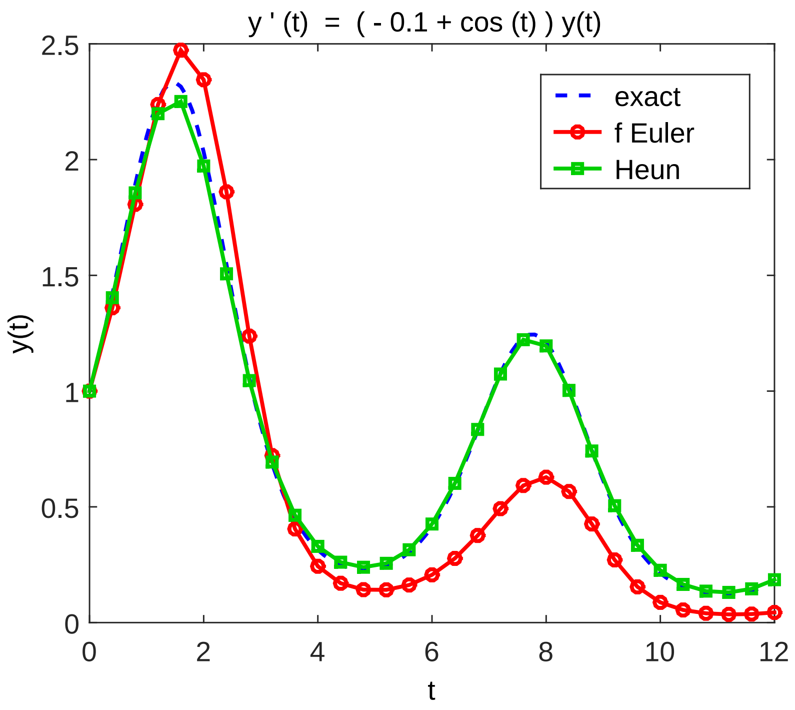

Figure 5.5: *Comparison between the solutions of* (5.42) *obtained using the forward Euler method and Heun's method, with* $h = 0.4$.

Finally we study the convergence order of the two methods. To do this we solve the problem using a number of different time steps and compute the errors in the resulting approximations, evaluated at time $t = 6$*, when compared to the exact solution.*

```
>> N=15; tmax=6;
>> y6 = exp(-0.1*tmax+sin(tmax));
>> for i=1:5
     [t_fe, u_fe] = feuler(f,[0,tmax],y0,N);
     err_fe(i) = abs(y6 - u_fe(end));
     [t_heun, u_heun] = heun(f,[0,tmax],y0,N);
     err_heun(i) = abs(y6 - u_heun(end));
     N=2*N;
   end
>> h=[0.4, 0.2, 0.1, 0.05, 0.025];
>> loglog(h,err_fe,'b',h,err_heun,'r')
```

In Figure 5.7 the errors of the two methods are plotted against h *on a logarithmic scale. First order convergence is observed for the explicit Euler method and second order convergence for Heun's method, as predicted by the theory.*

One can construct higher order Runge-Kutta methods by increasing the number of interior

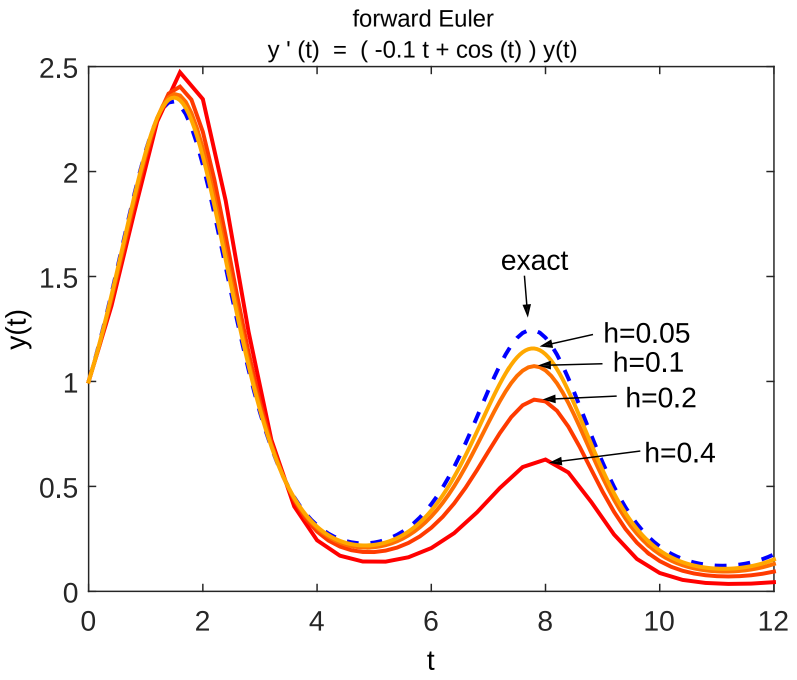

Figure 5.6: *Solutions of* (5.42) *obtained using forward Euler and different timesteps.*

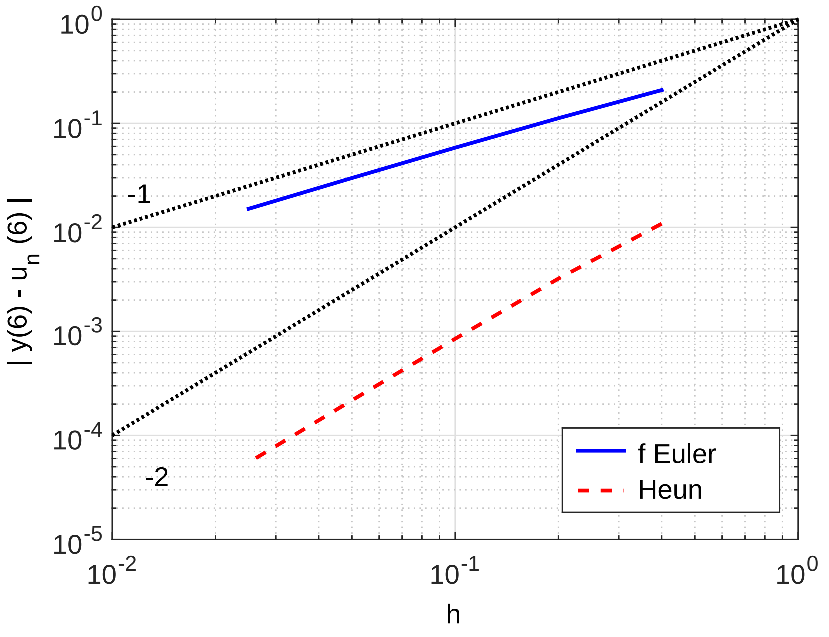

Figure 5.7: *Errors on a logarithmic scale plotted against the timestep h for the forward Euler method and Heun's method approximating $y(6)$ for the problem* (5.42).

stages of the method. The following scheme is a classical explicit Runge-Kutta method of order 4, often known simply as RK4.

$$u_n \to \begin{cases} p_1 = f(t_n, u_n) \\ p_2 = f(t_n + \frac{h_n}{2}, u_n + \frac{h_n}{2} p_1) \\ p_3 = f(t_n + \frac{h_n}{2}, u_n + \frac{h_n}{2} p_2) \\ p_4 = f(t_{n+1}, u_n + h_n p_3) \\ u_{n+1} = u_n + \frac{h_n}{6}\big(p_1 + 2p_2 + 2p_3 + p_4\big). \end{cases}$$

One can also consider implicit Runge-Kutta methods which have improved stability properties. However, we will not study these methods here.

Remark 5.6.2. *In Matlab several algorithms are already implemented for the solution of initial value problems (see for example* `ode45`*,* `ode23`*,* `ode23s`*,* `ode15s`*). In particular,* `ode45` *uses a fourth order method similar to the fourth order Runge-Kutta method RK4 presented above, combined with a fifth order method to compute an estimate of the error and adapt the local time step in order to satisfy a prescribed tolerance on the estimated error.*

5.7 Systems of ordinary differential equations

Much of the above analysis extends in a natural way to higher-dimensional systems of ODEs, where the IVP (5.3) is replaced by the multi-dimensional analogue (cf. (5.2))

$$\begin{cases} \boldsymbol{y}'(t) = \boldsymbol{f}(t, \boldsymbol{y}(t)), \qquad t > 0, \\ \boldsymbol{y}(0) = \boldsymbol{y}_0. \end{cases}$$

Here $\boldsymbol{y} : [0,\infty) \to \mathbb{R}^p$ for some $p \in \mathbb{N}$ and $\boldsymbol{f} : [0,\infty) \times \mathbb{R}^p \to \mathbb{R}^p$.

For instance, the forward Euler method becomes

$$\begin{cases} \boldsymbol{u}_0 = \boldsymbol{y}_0, \\ \boldsymbol{u}_{n+1} = \boldsymbol{u}_n + h\boldsymbol{f}(t_n, \boldsymbol{u}_n), \qquad n = 0, 1, 2, \dots, \end{cases} \tag{5.43}$$

the backward Euler method takes the form

$$\begin{cases} \boldsymbol{u}_0 = \boldsymbol{y}_0, \\ \boldsymbol{u}_{n+1} = \boldsymbol{u}_n + h\boldsymbol{f}(t_{n+1}, \boldsymbol{u}_{n+1}), \qquad n = 0, 1, 2, \dots, \end{cases} \tag{5.44}$$

and the Crank-Nicolson method is

$$\begin{cases} \boldsymbol{u}_0 = \boldsymbol{y}_0, \\ \boldsymbol{u}_{n+1} = \boldsymbol{u}_n + \frac{h}{2}\left(\boldsymbol{f}(t_n, \boldsymbol{u}_n) + \boldsymbol{f}(t_{n+1}, \boldsymbol{u}_{n+1})\right), \qquad n = 0, 1, 2, \dots. \end{cases} \tag{5.45}$$

Concerning stability, one has to consider the eigenvalues of the Jacobian matrix of partial derivatives

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}(t,\boldsymbol{y}) = \begin{bmatrix} \frac{\partial f_1}{\partial y_1}(t,\boldsymbol{y}) & \dots & \frac{\partial f_1}{\partial y_n}(t,\boldsymbol{y}) \\ \vdots & & \\ \frac{\partial f_n}{\partial y_1}(t,\boldsymbol{y}) & \dots & \frac{\partial f_n}{\partial y_n}(t,\boldsymbol{y}) \end{bmatrix}.$$

Suppose that the eigenvalues $\lambda_j(t,\boldsymbol{y})$, $j = 1,\dots,p$, of $\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}(t,\boldsymbol{y})$ are all distinct, real and negative. Then the backward Euler method and the Crank-Nicolson method are both unconditionally stable to perturbations, whereas the forward Euler method is stable under the condition that

$$h < \frac{2}{\max_{t,y} \max_{j=1,\dots,p} |\lambda_j(t,\boldsymbol{y})|} = \frac{2}{\max_{t,y} \rho(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}(t,\boldsymbol{y}))}. \tag{5.46}$$

Regarding implementation, explicit methods (such as forward Euler) just require evaluations of the function $\boldsymbol{f}$ to compute each increment. But implicit methods (such as backward Euler or Crank-Nicolson) require the solution of a system of nonlinear equations, using for example a fixed point method such as Newton's method. (In the special case where the function $\boldsymbol{f}$ is a linear function of $\boldsymbol{y}$, i.e. $\boldsymbol{f}(t,\boldsymbol{y}) = A(t)\boldsymbol{y} + \boldsymbol{b}(t)$ for some $A \in \mathbb{R}^{p\times p}$ and $\boldsymbol{b}(t) \in \mathbb{R}^p$, one has to solve a linear system of equations, using for example one of the iterative methods considered in §4.) This makes implicit methods considerably more expensive than explicit methods in general, and so in practical applications a great deal of thought goes into deciding how to trade off computational cost against stability, especially when the size of the systems involved is large.

5.8 Applications

We conclude our study of initial value problems by revisiting the examples introduced at the beginning of the chapter.

Example 5.8.1. **(Example 5.1.1 continued)** *Let us first consider the scalar equation (5.1). Let the initial population consist of* 40 *rabbits with a reproductive factor of* $C = 0.08$ *(where the time unit is one month) and a maximum population of* $B = 70$ *rabbits. We solve the equation using Heun's method with* $h = 1$ *(month) over a period of three years:*

```
>> f=@(t,y)0.08*y.*(1-(y/70)); tmax=36; N=36; y0=40;
>> [t,y] = heun(f,[0,tmax],y0,N); plot(t,y)
```

The resulting solution is shown in Figure 5.8.

To compute the solution using the built-in Matlab routine `ode45` *(which, as discussed above, uses an adaptive fourth order method similar to the fourth order Runge-Kutta method presented above), one could use the commands*

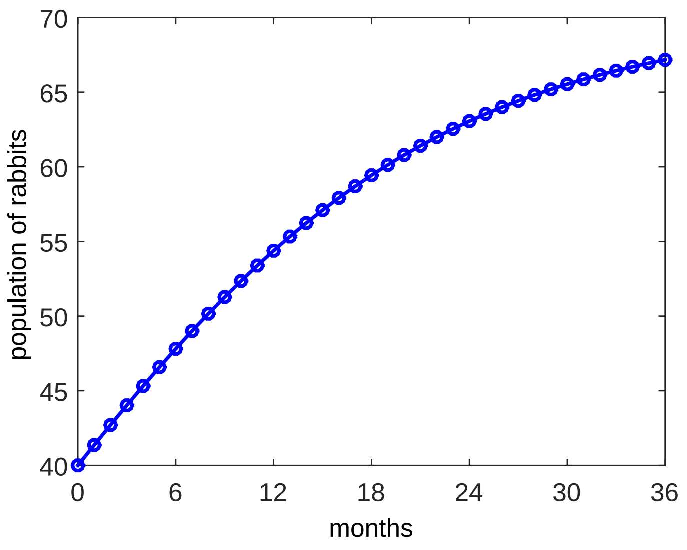

Figure 5.8: *Solution of the scalar Lotka-Volterra equation (5.1).*

```
>> options=odeset('RelTol',1e-4);
>> [t,y] = ode45(f,[0,tmax], y0, options);
```

The first command sets the options for the solver. Here we use a relative tolerance of 10^{-4} *(type* ***help odeset*** *to see all the options). The second command calls the routine approximating the solution of the IVP defined by the function* ***f****, over the interval* ***[0, tmax]*** *and with initial condition* ***y0****.*

Let us now turn to the system (5.2). Once again we consider an initial population of $y_1(0) = 40$ *rabbits but now we also introduce an initial population* $y_2(0)$ *of 20 foxes and consider the Lotka-Volterra system*

$$\begin{cases} y_1'(t) = 0.08\, y_1(t) - 0.004\, y_1(t) y_2(t), \\ y_2'(t) = -0.06\, y_2(t) + 0.002\, y_1(t) y_2(t). \end{cases} \tag{5.47}$$

If the following vectors are introduced

$$\boldsymbol{y}(t) = \begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix}, \qquad \boldsymbol{f}(t, \boldsymbol{y}) = \begin{bmatrix} 0.08\, y_1(t) - 0.004\, y_1(t) y_2(t) \\ -0.06\, y_2(t) + 0.002\, y_1(t) y_2(t) \end{bmatrix},$$

then the system (5.47) may be written in the form:

$$\boldsymbol{y}'(t) = \boldsymbol{f}(t, \boldsymbol{y}), \quad t > 0, \qquad \boldsymbol{y}(0) = [y_1(0), y_2(0)]^T. \tag{5.48}$$

The forward Euler method

$$\frac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^n}{h_n} = \boldsymbol{f}(t_n, \boldsymbol{u}^n)$$

would give the numerical scheme

$$\begin{cases} \dfrac{u_1^{n+1} - u_1^n}{h_n} = 0.08\, u_1^n - 0.004\, u_1^n u_2^n, \quad n \geq 0 \\ \dfrac{u_2^{n+1} - u_2^n}{h_n} = -0.06\, u_2^n + 0.002\, u_1^n u_2^n, \quad n \geq 0 \\ u_1^0 = y_1(0), \quad u_2^0 = y_2(0), \end{cases}$$

which can be easily implemented in Matlab. Alternatively, one can use a built-in Matlab solver such as `ode45`*:*

```
>>  y0=[40;20]; tmax=120;
>>  f=@(t,y)[0.08*y(1) - 0.004*y(1)*y(2);-0.06*y(2) + 0.002*y(1)*y(2)];
>>  options=odeset('RelTol',1e-4);
>>  [t,y] = ode45(f,[0,tmax], y0, options);
>>  plot(t,y(:,1),'b', t,y(:,2),'r')
```

The first column of `y` *contains the solution y_1 and the second y_2.*

In Figure 5.9 the evolution of the two populations over a time interval of 10 years (120 months) is plotted. Observe that the timestep used by the `ode45` *solver is not uniform across the whole time interval.*

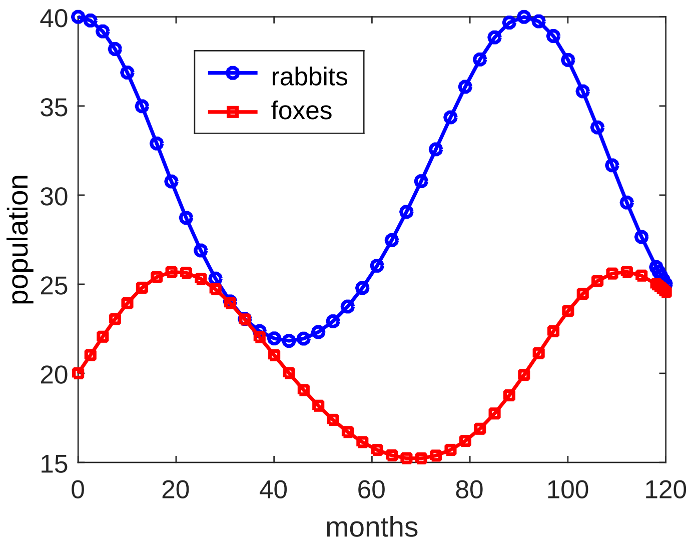

Figure 5.9: *Evolution of the populations of rabbits and foxes during a 10 year period, computed using the Matlab* `ode45` *solver.*

5.9 Boundary value problems

So far we have considered the numerical solution of initial value problems, where initial conditions are prescribed at one end of the interval on which we solve the differential equation. In this section we turn our attention briefly to the study of *boundary value problems* (BVPs), in which boundary conditions are prescribed at both ends of the interval. The study of BVPs has a huge literature and there are many powerful numerical methods available. In this course we shall just consider the application of *finite difference* approximations, as we did in the study of initial value problems. More advanced techniques such as the finite element method will not be considered here.

We shall restrict our attention mostly to one-dimensional second-order linear BVPs of the form

$$\begin{cases} L(y)(x) := -y''(x) + r(x)y(x) = f(x), \qquad x \in (0, X), \\ y(0) = y_0, \quad y(X) = y_X. \end{cases} \tag{5.49}$$

Here $X > 0$ is the length of the interval on which the problem is posed, y_0 and y_X are the prescribed boundary values, and r and f are given continuous functions on $[0, X]$ with

$$r(x) \geq 0, \quad \forall x \in [0, X]. \tag{5.50}$$

The problem (5.49) can be shown to be well-posed. We now consider how to solve it numerically.

5.9.1 Finite difference approximation

To obtain a numerical approximation to the solution of (5.49) we begin, as in the case of initial value problems, by discretizing the interval $[0, X]$ using a uniform mesh with mesh points $x_n = nh$, $n = 0, 1, \ldots, N$, where $h = X/N$ for some $N \in \mathbb{N}$. Letting u_n, $n = 0, 1, \ldots, N$, denote our approximation to $y(x_n)$, we replace the continuous system (5.49) by the discrete approximation

$$\begin{cases} L_N(u_n) := -D^2 u_n + r_n u_n = f_n, \qquad n = 1, \ldots, N-1, \\ u_0 = y_0, \quad u_N = y_X, \end{cases} \tag{5.51}$$

where $r_n = r(x_n)$, $f_n = f(x_n)$, and the operator D^2 is defined, for any set of numbers v_n, $n = 0, 1, \ldots, N$, by

$$D^2 v_n = \frac{v_{n-1} - 2v_n + v_{n+1}}{h^2}, \qquad n = 1, \ldots, N-1.$$

Provided that u is four times differentiable on $[0, X]$ with u'''' bounded (see Exercise sheet)

$$\max_{n=1,\ldots,N-1} \left|u''(x_n) - D^2 u(x_n)\right| = O(h^2), \qquad \text{as } h \to 0. \tag{5.52}$$

The discrete system (5.51) can be written more compactly as a linear system

$$A\boldsymbol{u} = \boldsymbol{b} \tag{5.53}$$

of size $(N-1)$-by-$(N-1)$, where $\boldsymbol{u} = (u_1, \ldots, u_{N-1})^T$,

$$A = \frac{1}{h^2}\begin{pmatrix} 2 & -1 & 0 & \cdots & \cdots & 0 \\ -1 & 2 & -1 & & & \vdots \\ 0 & -1 & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & & -1 & 0 \\ \vdots & & & -1 & 2 & -1 \\ 0 & \cdots & \cdots & 0 & -1 & 2 \end{pmatrix} + \begin{pmatrix} r_1 & 0 & \cdots & & \cdots & 0 \\ 0 & r_2 & 0 & & & \vdots \\ \vdots & 0 & \ddots & \ddots & & \\ & & \ddots & & 0 & \vdots \\ \vdots & & & 0 & r_{N-2} & 0 \\ 0 & \cdots & & \cdots & 0 & r_{N-1} \end{pmatrix}, \tag{5.54}$$

and

$$\boldsymbol{b} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_{N-2} \\ f_{N-1} \end{pmatrix} + \frac{1}{h^2}\begin{pmatrix} u_0 \\ 0 \\ \vdots \\ 0 \\ u_N \end{pmatrix}. \tag{5.55}$$

Note that the matrix A describes the action of the discrete operator L_N and the boundary conditions are incorporated into the first and last elements of the vector $\boldsymbol{b}$.

Note that A is symmetric and diagonally dominant by rows, but not strictly diagonally dominant unless $r_n > 0$ for all n. That A is invertible follows from the fact that it is positive definite, which can be checked by noting that, for any $\mathbf{0} \neq \boldsymbol{x} = (x_1, \ldots, x_{N-1})^T \in \mathbb{R}^{N-1}$,

$$\boldsymbol{x}^T A \boldsymbol{x} \geq x_1^2 + x_{N-1}^2 + \sum_{n=2}^{N-1} (x_n - x_{n-1})^2 > 0. \tag{5.56}$$

Hence the numerical solution u_n is uniquely defined.

To determine the accuracy of the numerical approximation we introduce the truncation error

$$T_n = L_N(y(x_n)) - f_n,$$

and the global error

$$e_n = y(x_n) - u_n,$$

which, by the definition of the numerical solution, satisfy the relation

$$L_N(e_n) = T_n. \tag{5.57}$$

From (5.52) and the fact that $y(t)$ satisfies the BVP (5.49) we know that

$$T(h) = \max_{n=1,\ldots,N-1} |T_n| = O(h^2), \qquad \text{as } h \to 0. \tag{5.58}$$

Hence if we can prove a stability bound on the operator L_N (i.e. a bound on $|e_n|$ given a bound on $|T_n|$), we can derive an estimate of the global error e_n. There are different ways to achieve this; here we shall prove stability with respect to the supremum norm using the *discrete maximum principle* and the concept of a so-called *companion function*.

Theorem 5.9.1 (Discrete maximum principle)**.** *Let a_n, b_n, c_n, $n = 1, \dots, N-1$ be positive real numbers such that $b_n \geq a_n + c_n$, and let U_n, $n = 0, \dots, N$, be real numbers such that*

$$-a_n U_{n-1} + b_n U_n - c_n U_{n+1} \leq 0, \qquad n = 1, \dots, N-1.$$

Then

$$U_n \leq \max\{U_0, U_N, 0\}, \qquad n = 1, \dots, N-1.$$

Proof. See exercise sheet. □

To use this result to prove stability, we introduce the quadratic "companion function"

$$\varphi(x) = \frac{x(X-x)}{2},$$

which satisfies the BVP

$$\varphi''(x) = -1, \quad x \in (0, X), \qquad \text{with } \varphi(0) = \varphi(X) = 0.$$

Furthermore, we note that $\varphi(x) \geq 0$ for all $x \in [0, X]$, with $\varphi(x)$ attaining its maximum value of $X^2/8$ at the midpoint $x = X/2$. The discrete samples $\varphi_n = \varphi(x_n)$ satisfy

$$D^2 \varphi_n = -1, \quad n = 1, \dots, N-1, \qquad \text{with } \varphi_0 = \varphi_N = 0,$$

and $0 \leq \varphi_n \leq X^2/8$ for $n = 0, \dots, N$.

The point of the companion function is that it allows us to bound the solution error in terms of the truncation error. Indeed, where $T(h) = \max_{n=1,\dots,N-1} |T_n|$, it holds that

$$|e_n| \leq T(h)\varphi_n, \qquad n = 0, \dots, N. \tag{5.59}$$

To prove this, we first note that

$$L_N(\varphi_n) = 1 + r_n \varphi_n \geq 1, \qquad n = 1, \dots, N-1, \tag{5.60}$$

which means that, by (5.57),

$$L_N(e_n - T(h)\varphi_n) = T_n - T(h) L_N(\varphi_n) \leq T_n - T(h) \leq 0, \qquad n = 1, \dots, N-1. \tag{5.61}$$

Then, recalling the definition of L_N and its matrix representation A, application of the discrete maximum principle (Theorem 5.9.1) with $a_n = c_n = 1/h^2$, $b_n = 2/h^2 + r_n$ and $U_n = e_n - T(h)\varphi_n$, tells us that $e_n - T(h)\varphi_n \leq 0$ for all $n = 0, \dots, N$. (Note that $e_0 - T(h)\varphi_0 =$

$e_N - T(h)\varphi_N = 0$.) The same argument, with e_n replaced by $-e_n$, shows that $-e_n - T(h)\varphi_n \leq 0$ for all $n = 0, \ldots, N$, and together these two inequalities imply (5.59).

Finally, recalling that $0 \leq \varphi_n \leq X^2/8$ for $n = 0, \ldots, N$, (5.59) implies the stability bound

$$E(h) = \max_{n=0,\ldots,N} |e_n| \leq \frac{X^2 T(h)}{8}, \tag{5.62}$$

and combining this with the consistency result (5.58), we deduce that the numerical method is second order convergent, with

$$E(h) = \max_{n=0,\ldots,N} |e_n| = O(h^2), \qquad \text{as } h \to 0. \tag{5.63}$$