22 Initial Value Problems for Ordinary Differential Equations

Many problems in applied mathematics can be formulated as initial value problems for ordinary differential equations or systems of such equations. These lecture notes describe a few numerical methods for their solution.

The initial value problem for an ordinary differential equation entails finding a function \( u(t) \) that satisfies the differential equation

\[
\frac{du(t)}{dt} = f(t, u(t)), \quad t \geq t_0,
\]

and the initial conditions

\[
u(t_0) = u_0.
\]

We may be interested in the solution \( u(t) \) in a finite interval \([t_0, T]\) or in the semi-infinite interval \( t \geq t_0 \). In most of this lecture \( f \) and \( u \) are real-valued functions. A generalization to vector-valued functions is important in applications and will be discussed at the end of the lecture.

Numerical methods for the solution of the initial value problem (1)-(2) generate a sequence of values of the independent variable, \( t_1, t_2, \ldots \), with \( t_0 < t_1 < t_2, \ldots \), and a corresponding sequence of values of the dependent variable, \( u_1, u_2, \ldots \), such that \( u_j \) approximates the function \( u(t) \) at \( t = t_j \), \( j = 1, 2, \ldots \). The difference \( h_j = t_{j+1} - t_j \) is referred to as the step length. Numerical methods differ in the step length, the accuracy in the computed approximations \( u_j \) of \( u(t_j) \), and the computational effort required to determine \( u_j \), when \( u_0, u_1, \ldots, u_{j-1} \) already have been computed. In our discussion on properties of numerical methods, we use a fixed step length \( h \).

Then

\[
t_j = t_0 + hj, \quad j = 0, 1, 2, \ldots.
\]

We set \( t_0 = 0 \) in most of the examples.

It follows from the Fundamental Theorem of Calculus that the computation of the solution of the initial value problem (1)-(2) is equivalent to evaluating the integral,

\[
u(t_1) = u(t_0) + \int_{t_0}^{t_1} f(s, u(s))ds.
\]

Application of the mean-value theorem to the right-hand side yields

\[
\int_{t_0}^{t_1} f(s, u(s))ds = hf(\hat{t}, u(\hat{t})), \quad h = t_1 - t_0,
\]

for some \( \hat{t} \) in the interval \([t_0, t_1]\). Substituting the right-hand side above into the right-hand side of (4) gives

\[
u(t_1) = u(t_0) + hf(\hat{t}, u(\hat{t})).
\]

Unfortunately, this formula cannot be applied in a straightforward way to determine \( u(t_1) \), because neither \( \hat{t} \) nor \( f(\hat{t}, u(\hat{t})) \) are explicitly known. Nevertheless, the relation (5) is suggestive for the development of numerical methods.
22.1 Euler’s method

When the interval \([t_0, t_1]\) is small, \(\hat{t}\) in (5) is close to \(t_0\). For smooth functions \(t \to f(t, u(t))\), the function values \(f(\hat{t}, u(\hat{t}))\) and \(f(t_0, u(t_0))\) also are close. This suggests that we may replace \(f(\hat{t}, u(\hat{t}))\) by \(f(t_0, u(t_0))\) in (5). We obtain

\[
    u_1 = u_0 + hf(t_0, u(t_0)).
\]

The approximation \(u_1\) of \(u(t_1)\) easily can be computed by evaluating the right-hand side.

In order to compute an approximation \(u_2\) of \(u(t_2)\), we consider the initial value problem

\[
    \frac{du(t)}{dt} = f(t, u(t)), \quad t \geq t_1,
\]

with initial value \(u(t_1) = u_1\). Repeating the above argument yields, analogously to (6), the formula

\[
    u_2 = u_1 + hf(t_1, u_1).
\]

Generally, we obtain

\[
    u_{j+1} = u_j + hf(t_j, u_j), \quad j = 0, 1, 2, \ldots, \quad t_j = t_0 + jh,
\]

with \(u_0\) the same as in (2).

Another way to derive Euler’s method is to expand the solution of the initial value problem (1)-(2) at \(t_1 = t_0 + h\) into a Taylor series around \(t_0\), and truncate the expansion after the linear term in \(h\). We obtain

\[
    u(t_1) = u(t_0 + h) = u(t_0) + hu'(t_0) + \frac{h^2}{2}u''(t_0) + \ldots \approx u(t_0) + hu'(t_0) = u(t_0) + hf(t_0, u(t_0)).
\]

Since the computed values are approximations, we replace \(u(t_1)\) by \(u_1\) and obtain (6). Thus, \(u(t)\) is approximated by the linear function \(u_0 + tu'(0)\) in the interval \(t_0 \leq t \leq t_1\). We conclude that Euler’s method approximates \(u(t)\) by a linear function on each subinterval \([t_j, t_{j+1}]\). It follows that Euler’s method provides a piecewise linear approximation of \(u(t)\) for \(t \geq t_0\).

Example 22.1: Consider the initial value model problem

\[
    u'(t) = \lambda u(t), \quad t \geq t_0, \quad u(t_0) = u_0,
\]

with solution

\[
    u(t) = \exp(\lambda(t - t_0))u_0.
\]

We are primarily interested in the model problem with \(\lambda < 0\) and \(u_0 > 0\). The solution \(u(t)\) then decreases as \(t\) increases.

Two steps of length \(h\) with Euler’s method applied to the solution of the initial value problem (1)-(2) give the values

\[
    u_1 = u_0 + h\lambda u_0, \quad u_2 = u_1 + h\lambda u_1,
\]

where \(u_j\) is the computed approximation of \(u(t_j)\). Figure 1 shows the solution \(u(t)\), \(0 \leq t \leq 2\), (continuous blue graph) and the points \((t_j, u_j), j = 0, 1, 2\), marked by o,*,+\), respectively, for \(\lambda = -1\), \(t_j = hj, h = 0.4\), and \(u_0 = 1\). The dashed red line depicts the linear approximation of \(u(t)\) used in the first step of Euler’s method, and the dash-dotted black line shows the linear approximation employed by Euler’s method in the second step. □
Important properties of numerical solution methods for initial value problems can be studied by applying the methods to the model problem (8). We first consider convergence. This is the property that the computed approximations \( u_j \) approach \( u(t_j) \) for all indices \( j \) of interest as the step length \( h \) is reduced to zero. We are also interested in how quickly the \( u_j \) approach \( u(t_j) \) as \( h \) decreases to zero.

This lecture will use the notation \( \mathcal{O}(h^\gamma) \), where \( \gamma \) is a positive scalar independent of \( h \), for a quantity that is bounded by an expression of the form \( c|h|^{\gamma} \) as \( |h| \to 0 \). Here \( c \) also is a positive constant independent of \( h \). Hence,

\[
\lim_{|h|\to 0} \frac{\mathcal{O}(h^\gamma)}{|h|^\gamma} \leq c.
\]

Example 22.2: The Taylor expansion of \( \exp(h) \) at the origin is given by

\[
\exp(h) = 1 + h + \frac{h^2}{2!} + \frac{h^3}{3!} + \ldots,
\]

and it follows that

\[
\exp(h) = 1 + h + \mathcal{O}(h^2).
\]

Similarly, for any constant \( \lambda \) independent of \( h \), we obtain

\[
\exp(h\lambda) = 1 + h\lambda + \frac{h^2\lambda^2}{2!} + \frac{h^3\lambda^3}{3!} + \ldots = 1 + h\lambda + \mathcal{O}(h^2).
\]
Example 22.3: Equation (12) yields that
\[ 1 + h\lambda = \exp(h\lambda) + O(h^2) = \exp(h\lambda)(1 + O(h^2)) = \exp(h\lambda)\exp(O(h^2)) = \exp(h\lambda + O(h^2)). \] (13)
We will use this expression below. □

Let us consider the error in the approximation \( u_1 \) of \( u(t_1) \) determined by the first step of Euler’s method with \( t_1 = h \). We obtain from (9) and (10) that
\[ u_1 - u(t_1) = u_0 + h\lambda u_0 - \exp(h\lambda)u_0. \] (14)
Substituting the expression (12) into the right-hand side of (14) gives
\[ u_1 - u(t_1) = u_0 + h\lambda u_0 - (1 + h\lambda + O(h^2))u_0 = O(h^2). \] (15)
This error is referred to as the local error; it is the error introduced in the computed approximation of the solution when taking one step of length \( h \), starting from an error-free function value.

We have shown that Euler’s method has local error \( O(h^2) \) when applied to the model problem (8). This local error is also obtained when applying Euler’s method to the differential equation (1) for any sufficiently smooth function \( f(t, u) \). The formulas for showing the latter are more complicated. It therefore is advantageous to investigate the local error by studying the model problem.

When reducing \( h \), the point \( t_1 \) where we measure the difference \( u_1 - u(t_1) \) decreases to zero. Often one is interested in the computed approximation furnished by a numerical method at a fixed point \( T \) when reducing \( h \). Let \( t_0 = 0 \) and choose the step length \( h \) so that \( T \) is an integer multiple, \( n \), of \( h \), i.e.,
\[ T = t_n = nh. \] (16)
Thus, \( n = T/h \) increases when \( h \) decreases. The error \( u_n - u(T) \) is referred to as the global error in \( u_n \). It can be determined as follows. First note that
\[ u_n = u_{n-1} + h\lambda u_{n-1} = (1 + h\lambda)u_{n-1} = (1 + h\lambda)^2u_{n-2} = \ldots = (1 + h\lambda)^n u_0. \]
Substituting the expression (13) into the right-hand side, and using the fact that \( T = nh \) is constant, we obtain
\[ u_n = (\exp(h\lambda + O(h^2)))^n u_0 = (\exp(\lambda nh + O(nh^2)))u_0 = (\exp(\lambda T + O(h)))u_0 = (\exp(\lambda T)(1 + O(h)))u_0 = (\exp(\lambda T) + O(h))u_0 = \exp(\lambda T)u_0 + O(h) = u(T) + O(h), \]
where the last equality follows from (9). Thus, the global error for Euler’s method is
\[ u_n - u(T) = O(h). \] (17)
This shows that when we half the step length \( h \), we can expect the error in the computed approximation of \( u(T) \) for any fixed \( T > 0 \) to be halved. The formula (17) also holds for more general initial value problems (1)-(2) than the model problem (8). Euler’s method is said to be a first order method, because the global error depends linearly on \( h \).

It holds generally that when the local error is \( O(h^{k+1}) \) for some positive integer \( k \), then the global error is \( O(h^k) \). We have shown this relation for Euler’s method. A heuristic argument for the validity of this
relation is that we determine an approximation \( u_n \) of \( u(T) \), with \( T = nh \), by carrying out \( n \) integration steps. Each step introduces a local error \( \mathcal{O}(h^{k+1}) \). Adding \( n \) of these local errors yields the global error \( n\mathcal{O}(h^{k+1}) = \mathcal{O}(h^k) \) in \( u_n \).

The convergence result (17) tells us that we obtain more accurate approximations of the solution to the initial value (8) by reducing \( h \), at least in the absence of round-off errors. However, our analysis does not tell us how small \( h \) has to be in order for Euler’s method to furnish meaningful results. We will see that for some problems, the step length for Euler’s method has to be “tiny” even when the solution does not change much. This is related to the stability of Euler’s method.

Example 22.4: We consider the solution of the model problem (8) with \( \lambda = -25 \) and \( u_0 = 1 \) by Euler’s method over the interval \( 0 \leq t \leq 1 \). The solution is depicted by the blue graphs of Figures 2(a) and 2(b). However, Euler’s method with step size \( h = 0.1 \) does not determine a useful approximation of this solution. The red graph of Figure 2(b) shows the computed “solution”. It oscillates wildly.

This behavior of Euler’s method also can be observed if we integrate the differential equation over an interval where the solution does not change much. For instance, Figure 3 shows the exact solution (in blue) of the initial value problem (8) with \( \lambda = -25 \), \( t_0 = 0.2 \) and \( u_0 = \exp(-10) \approx 6.7 \cdot 10^{-3} \). The exact solution does not change much in absolute terms as \( t > t_0 \) increases. One therefore might believe that one can use a large step size for Euler’s method and still obtain an approximate solution with a small absolute error. This, unfortunately, is not the case, as is illustrated by Figure 3. The red oscillating graph displays the solution determined by Euler’s method, while the blue graph depicts the exact solution. □

The above example illustrates that it may be necessary to apply Euler’s method with a small step size even when the exact solution of the initial value problem does not change much. The following discussion shows why this is the case. Let \( \lambda < 0 \) in the model problem (8). Then the exact solution (9) decreases in magnitude as \( t \) increases. Euler’s method is said to be stable if the computed approximations \( u_j \) of \( u(t_j) \)

Figure 2: The solution of the initial value problem (8) with \( \lambda = -25 \) and \( u_0 = 1 \) is depicted by the blue graphs in figures (a) and (b). The red graph of figure (b) shows the “solution” determined by Euler’s method with step length \( h = 0.1 \). The function values \( u_j \), \( 1 \leq j \leq 10 \), computed by Euler’s method are marked by ∗.
Figure 3: The solution of the initial value problem (8) with \( \lambda = -25 \), \( t_0 = 0.2 \), and \( u_0 = \exp(-10) \), is depicted by the blue graph. The red graph shows the “solution” determined by Euler’s method with step length \( h = 0.1 \). The function values \( u_j \), \( 1 \leq j \leq 8 \), computed by Euler’s method are marked by *.

also decrease in magnitude as \( j \) increases. Thus, we would like that

\[
|u_{j+1}| < |u_j|, \quad j = 0, 1, 2, \ldots.
\]

Since \( u_{j+1} = (1 + h\lambda)u_j \) and \( u_j \neq 0 \), this is equivalent to

\[
|1 + h\lambda| < 1.
\]

This inequality holds if and only if

\[-2 < h\lambda < 0.\]

Since \( h > 0 \) and \( \lambda < 0 \), the upper bound is satisfied. The lower bound is equivalent to

\[
h < -\frac{2}{\lambda}
\]

and therefore is satisfied when \( h \) is sufficiently small.

Example 22.4 (cont’d): With \( \lambda = -25 \) the inequality (20) is satisfied for all \( h < 0.08 \). The step length used in the computations was 0.1, which gave rise to the observed instability. □

Remark 1: Stability analysis of integration methods often is carried out in the complex plane, i.e., \( \lambda \) is allowed to be a complex variable. The condition (19) then is satisfied by \( h\lambda \) in the open disk with center \(-1\) and radius 1 in the complex plane. In this course, we restrict the stability analysis to real \( \lambda \). □

22.2 The backward Euler method

This integration method is obtained by setting \( \hat{t} \) to \( t_1 \) in (5). The recursion formula for the backward Euler method is given by

\[
u_{j+1} = u_j + hf(t_{j+1}, u_{j+1}), \quad j = 0, 1, 2, \ldots, \quad t_j = t_0 + jh,
\]
with \( u_0 \) as in (2). Integration of one time-step with this method typically is more demanding integration of one time-step with Euler’s method, because the desired value, \( u_{j+1} \), is present both in the left-hand side and the right-hand side. We will discuss below how to handle this difficulty.

Figure 4: The solution of the initial value problem (8) with \( \lambda = -1 \) and \( u_0 = 1 \) (blue continuous graph), and the approximate solution determined by two steps of length \( h = 0.4 \) with the backward Euler method. The initial value is marked by o. After one step, the backward Euler method determines the value marked by *, and after two steps the value marked by +.

Figure 4 displays the first two integration steps with the backward Euler method applied to the model problem (8) with \( \lambda = -1 \) and \( u_0 = 1 \). The figure is analogous to Figure 1 for Euler’s method. Thus, the backward Euler method provides a piecewise linear approximation of the solution \( u(t) \) of the model problem. The slope of the line segment between \( t_j \) and \( t_{j+1} \) is \( \exp(-t_{j+1}) \); for Euler’s method the slope is \( \exp(-t_j) \).

Our interest in the backward Euler method stems from the fact that the stability requirement (18) does not impose a restriction on the step size \( h \). This can be seen as follows. Application of the backward Euler method to the model problem (8) with \( \lambda < 0 \) yields the recursion formula

\[
u_{j+1} = u_j + h\lambda u_{j+1}, \quad j = 0, 1, 2, \ldots ,
\]

which also can be expressed as

\[
u_{j+1} = \frac{1}{1 - h\lambda} u_j, \quad j = 0, 1, 2, \ldots .
\]  

Substituting this formula into (18) shows that the stability requirement is equivalent to

\[
\frac{1}{|1 - h\lambda|} < 1.
\]
Since $\lambda < 0$, this inequality is satisfied for all $h > 0$.

We turn to the local error for the backward Euler method. When applied to the model problem (8) the method computes the approximation $u_{j+1}$, given by (22), of $u(t_{j+1})$. Let $t_0 = 0$. The local error at $t_1 = h$ is given by

$$u_1 - u(t_1) = \frac{1}{1 - h\lambda}u_0 - \exp(h\lambda)u_0.$$  \hfill (23)

An expansion of $\exp(h\lambda)$ in terms of powers of $h$ is provided by (12). The first term in the right-hand side can be expressed in terms of powers of $h$ as follows. For any $|s| < 1$, we have

$$\frac{1}{1 - s} = 1 + s + s^2 + s^3 + \ldots.$$  \hfill (24)

The right-hand side is known as a geometric series. The series converges because $|s| < 1$. Therefore the right-hand side is meaningful. The equality (24) is discussed in Exercise 22.1. Letting $s = h\lambda$ yields

$$\frac{1}{1 - h\lambda} = 1 + h\lambda + h^2\lambda^2 + \mathcal{O}(h^3).$$

Substituting this expansion and (12) into (23) gives

$$u_1 = u(t_1) = 1 + h\lambda + h^2\lambda^2 + \mathcal{O}(h^3) - (1 + h\lambda + \frac{h^2\lambda^2}{2!} + \mathcal{O}(h^3)) = \mathcal{O}(h^2).$$

Thus, the local error for the backward Euler method is $\mathcal{O}(h^2)$, similarly as for Euler’s method. This suggests that the global error for the backward Euler method is $\mathcal{O}(h)$, i.e., that the backward Euler method is a first order method. The fact that this indeed is the case is the subject of Exercise 22.2.

We finally comment on how to apply the backward Euler method to the integration of a general initial value problem (1)-(2). The first step yields the following equation for $u_1$,

$$u_1 = u_0 + hf(t_1, u_1).$$

In many applications, $f$ is a nonlinear function of $u_1$. We therefore have to determine $u_1$ by one of the methods for solving nonlinear equations discussed in Lecture 15. For instance, define the (nonlinear) function

$$g(u) = u - u_0 - hf(t_1, u).$$  \hfill (25)

We would like to determine a zero close to $u_0$. It is convenient to use a zero-finders that does not require the derivative $g'(u)$, because the latter is cumbersome to evaluate. We may use $u_0$ as initial iterate, or, better, determine the initial iterate by one step with Euler’s method. If the zero-finder fails to determine $u_1$, then we reduce $h$ and seek to determine a solution of (25) for the smaller value of $h$.

Let $f(t, u)$ be a nonlinear function of $u$. This is the case in most problems of interest in applications. Methods for which the approximations $u_j$ of $u(t_j)$ have to be determined by solving an equation are referred to as implicit. The backward Euler method is the simplest implicit method. Methods which determine $u_j$ without solving an equation are said to be explicit. Euler’s method is the simplest explicit method.

### 22.3 Some other integration methods

Our interest in the Euler and backward Euler methods stems from that concepts, such as convergence and stability, easily can be illustrated. However, the integration methods provided in MATLAB and Octave do
not include these integration methods. The reason for this is that the Euler and backward Euler methods are only of first order. Therefore, these methods generally require a quite small step size $h$ in order to be able to determine an accurate approximation of the solution of initial value problems. There are several ways to derive higher order methods.

Example 22.5: Figures 1 and 4 suggest that the average of the approximations determined by the Euler and backward Euler methods might be a more accurate approximation of the desired solution than those determined by the Euler and backward Euler methods. Thus, given $u_j$, we define $u_{j+1}$ according to

$$u_{j+1} = \frac{1}{2} (u_j + hf(t_j, u_j)) + \frac{1}{2} (u_j + hf(t_{j+1}, u_{j+1})) = u_j + \frac{h}{2} (f(t_j, u_j) + f(t_{j+1}, u_{j+1})).$$

(26)

This method is known as the trapezoidal method. The local error can be shown to be $O(h^3)$; see Exercise 22.3. The method therefore is of second order. It is implicit.

Example 22.6: Combining the backward Euler and the Euler methods in a different manner than in Example 22.5 gives an explicit method. Replace the argument $u_{j+1}$ of $f$ in the trapezoidal method (26) by an approximation determined by Euler’s method. Thus, we define

$$u_{j+1}^{\text{Euler}} = u_j + hf(t_j, u_j),$$

$$u_{j+1}^{\text{Heun}} = u_j + \frac{h}{2} (f(t_j, u_j) + f(t_{j+1}, u_{j+1}^{\text{Euler}})).$$

This method is known as Heun’s method. It requires two evaluations of $f$ in each time step. This is the simplest one of a family of Runge-Kutta methods, which are commonly used as general purpose numerical integration methods. The MATLAB method ode45 belongs to this family.

There are many approaches to deriving integration methods for initial value problems. Integration methods differ in their order, their stability requirements, and in the computational work needed for integrating one step. Some implicit methods use a fixed number of iterations with a simple zero-finder in order to reduce the computational effort. This makes the implicit method explicit, even though it may be cumbersome to write up the recursion formula. This approach typically reduces the computational effort significantly per time step, compared with a truly implicit method, but it also affects the stability requirements. The choice of method should depend on the problem at hand.

Available software for integration methods adjusts the step length during the integration with the aim to compute an approximate solution of the initial value problem with desired accuracy with as little computational effort as possible. The step length $h$ is often determined by comparing the local error for two related integration methods. The methods in this comparison are chosen so that integration with both methods requires only little more computational work than integrating with one of the methods. For instance, the MATLAB integration method ode45 combines Runge-Kutta methods of orders 4 and 5 to determine an estimate of the local error in each integration step. Several of the function values can be used by both methods. If the computed error estimate is deemed too large, then the step size is reduced; if the estimate is much smaller than the desired accuracy, then the step size can be increased.

In Example 22.6, we evaluated $f$ twice for each step. This is an attractive approach to increasing the accuracy, when $f$ is fairly inexpensive to compute. For initial value problems for which $f$ is expensive to evaluate, we may consider to use information from previous time steps to increase the accuracy. This is possible with multistep methods. There are both explicit and implicit multistep methods.

Example 22.7: Consider the explicit multistep method

$$u_{j+1} = u_j + h(\alpha u_j' + \beta u_{j-1}'), \quad j = 1, 2, \ldots ,$$

(27)
where \( u'_\ell = f(t_\ell, u_\ell) \). This method requires two initial values, \( u_0 \) and \( u_1 \). The value \( u_0 \) is available; cf. (2). The other initial value, \( u_1 \), has to be computed, e.g., by an explicit integration method. The coefficients \( \alpha \) and \( \beta \) are determined so that the method integrates polynomials exactly of as high degree as possible. This is equivalent to determining the coefficients so as to maximize the order of the method.

Let \( u = 1 \). Then \( du/dt = f(t, u) \) vanishes, and therefore \( u'_j = u'_{j+1} = 0 \). Formula (27) holds for all coefficients \( \alpha \) and \( \beta \). We turn to the function \( u(t) = t \). Then \( du/dt = f(t, u) \equiv 1 \), and therefore \( u'_j = u'_{j+1} = 1 \). Substituting these values into (27) gives the equation

\[
  u_{j+1} = u_j + h(\alpha + \beta).
\]

We may choose \( j = 0 \) and \( t_0 = 0 \). Then \( u_0 = 0 \) and \( u_1 = 1 \). Substitution into the above formula gives

\[
  h = h(\alpha + \beta),
\]

i.e.,

\[
  \alpha + \beta = 1. \tag{28}
\]

Letting \( u(t) = t^2 \) and using the fact that \( du/dt = f(t, u) = 2t \), we obtain with \( t_0 = 0 \) that

\[
  u_0 = u'_0 = 0, \quad u_1 = t_1^2 = h^2, \quad u'_1 = 2t_1 = 2h.
\]

Substituting into (27) yields

\[
  h^2 = h(\beta 2h),
\]

i.e., \( \beta = 1/2 \). It follows from (28) that \( \alpha = 1/2 \). Hence, the multistep method is given by

\[
  u_{j+1} = u_j + \frac{h}{2} (f(t_j, u_j) + f(t_{j-1}, u_{j-1})), \quad j = 1, 2, \ldots . \tag{29}
\]

This is a member of the family of Adams-Bashforth methods. They were used by Adams to track the motion of planets. \( \square \)

### 22.4 Initial value problems for systems of ODEs

We have so far assumed \( f \) to be real-valued. However, all integration methods discussed allow \( f \) to be vector-valued.

**Example 22.8:** Consider the vector-valued model problem

\[
  u'(t) = Au(t), \quad t \geq 0, \quad u(0) = u_0, \tag{30}
\]

where \( A \in \mathbb{R}^{k \times k} \) and \( u(t), u_0 \in \mathbb{R}^k \). It has the solution

\[
  u(t) = \exp(At)u_0, \tag{31}
\]

which is analogous to (9). The matrix exponential can be defined by the power series

\[
  \exp(A) = I + A + \frac{A^2}{2!} + \frac{A^3}{3!} + \ldots .
\]

similar to (11). Here \( I \) denotes the identity matrix. The power series converges for any square matrix. Differentiating the series term-wise, just like if \( A \) were a scalar, shows that the derivative of the function \( t \to \exp(At) \) is \( A \exp(At) \). Therefore the function (31) indeed solves (30). \( \square \)
Remark 2: The matrix exponential of a square matrix $A = [a_{ij}]$ of small to moderate size can be computed in MATLAB by the command `expm(A)`. Note that this is different from the MATLAB command `exp(A)`, which gives a matrix with entries $\exp(a_{ij})$.

Application of Euler’s method with step length $h$ to the solution of the initial value problem (30) gives a recursion formula of the form

$$u_{j+1} = u_j + hAu_j, \quad j = 0, 1, 2, \ldots .$$

Each iteration requires the evaluation of a matrix-vector product with $A$. The computations are simple, and they would not be much more complicated if the entries of the matrix $A$ would depend on $t$ or $u$, i.e., if $A = A(t, u)$. Other explicit methods are essentially equally easy to apply.

The corresponding recursion formula for the backward Euler method is of the form

$$u_{j+1} = u_j + hAu_{j+1}, \quad j = 0, 1, 2, \ldots ,$$

which can be written as

$$(I - hA)u_{j+1} = u_j, \quad j = 0, 1, 2, \ldots .$$

In order to compute $u_{j+1}$, we have to solve a linear system of equations with the matrix $I - hA$. This can be done, for instance, by computing the LU-factorization of this matrix. If $A$ does not change with $t$ or $u$, and if the step length $h$ is kept fixed, then we can apply the same LU-factorization for all time steps. The factorization has to be recomputed whenever $A$ or $h$ change. Other implicit methods can be applied in roughly the same manner. We conclude that it may be much more expensive to integrate one step with an implicit method than with an explicit one, in particular when $A$ is large. However, some applications of interest in science and engineering give rise to initial value problems for which stability considerations for explicit methods force the step length to be so small that implicit methods, which allow a larger step length, are competitive.

Example 22.9: Consider the initial value problem for the second order ordinary differential equation,

$$u''(t) = f(t, u(t), u'(t)), \quad t \geq t_0, \quad (32)$$

$$u(t_0) = u_0, \quad u'(t_0) = u'_0,$$

where $'$ denotes differentiation with respect to $t$. Initial values both for the function and the first derivative at $t = t_0$ are prescribed.

This initial value problem can be transformed to an initial value problem for a system of two first order ordinary differential equations. The latter can be solved by standard software for systems of first order equations. Let $w(t) = u'(t)$. Then $w'(t) = u''(t)$ and we can write (32) in the form

$$\frac{d}{dt} \begin{bmatrix} u \\ w \end{bmatrix} = \begin{bmatrix} w \\ f(t, u, w) \end{bmatrix}, \quad t \geq t_0, \quad (33)$$

with initial values

$$u(t_0) = u_0, \quad w(t_0) = u'_0.$$

This approach to transform an initial value problem for a second order differential equation to an initial value problem for a system of first order differential equations can be applied to transform initial value problems for differential equations of order higher than 2 as well; each derivative provides one function in the system.
**Exercise 22.1**

There are several ways to verify formula (24). One may consider the right-hand side a Taylor expansion at the origin of the left-hand side. Alternatively, one may multiply equation (24) by $1 - s$ and show equality for the expressions so obtained. Show (24) by one of these or by some other method. □

**Exercise 22.2**

Show that the global error for the backward Euler method applied to the model problem (8) is $O(h)$. Hint: Use the fact that $u_n = (1 - h\lambda)^{-n}u_0$. □

**Exercise 22.3**

Apply the trapezoidal method of Example 22.5 to the model problem (8) and show that the local error is $O(h^3)$. □

**Exercise 22.4**

What is the local error for the multistep method (29)? □

**Exercise 22.5**

Determine the coefficients for the implicit multistep method

$$u_{j+1} = u_j + h(\alpha u_{j+1} + \beta u_j), \quad j = 0, 1, \ldots,$$

where $u'_\ell = f(t_\ell, u_\ell)$, so that the method integrates polynomials of as high degree as possible. This is the simplest member of a family of Adams-Moulton method. Hint: Proceed similarly as in Example 22.7. □

**Exercise 22.6**

Apply the trapezoidal method, Heun’s method, and the method of Example 22.7 to the model problem (8) with $\lambda = -25$. Determine empirically the largest step length $h$ that does not give stability problems. Which one of the methods allows the largest step length? □

**Exercise 22.7**

Apply the MATLAB integration methods ode45 and ode15s to the solution of the model problem (8) with $\lambda = -25$ and $u_0 = 1$ over the interval $0 \leq t \leq 10$. Which method requires the fewest steps? Octave has different integration methods. Let me know if you plan to use Octave. □

**Exercise 22.8**

Consider the van der Pol equation

$$u'' - \epsilon(1 - u^2)u' + u = 0, \quad y = y(t), \quad 0 \leq t \leq 25,$$

with initial values $u(0) = u'(0) = 1/2$. This equation models electrical circuits connected with triod oscillators. Solve the equation for $\epsilon = 1$, $\epsilon = 0.1$, $\epsilon = 0.01$. Compare the performance of a few different methods,
e.g., ode45, ode15s, ode23 in MATLAB, for the default relative accuracy (what is it?) and relative accuracy $1 \cdot 10^{-6}$. Plot the solutions and compare them. (Write a paragraph on your findings and show me a few plots.) □

**Exercise 22.9**

Apply your favorite integration method to determine the solution of the Lotka-Volterra equations, for $t \geq 0$,

\[
\begin{align*}
x' &= \alpha x + \beta xy, \\
y' &= \gamma y + \delta xy,
\end{align*}
\]

where $\alpha > 0$, $\beta < 0$, $\gamma < 0$, and $\delta > 0$, models the population dynamics of two interacting species that have a predator-prey relationship. Here $x = x(t)$ and $y = y(t)$ denotes the number of prey and predators, respectively. We assume that the prey population, in the absence of predators, increases at a rate proportional to $x$. Moreover, we assume that the number of times that the predator kills the prey depends on the probability of the two coming together and is therefore proportional to $xy$. This gives the equation for $x'$. We assume that the number of predators would decrease by natural causes if the prey were removed, contributing to the term $\gamma y$. However, the number of predators increases as a result of encounters with prey, giving the term $\delta xy$ in the equation for $y'$. The initial conditions are $x(0) = x_0$ and $y(0) = y_0$.

First use $\alpha = 0.25$, $\beta = -0.01$, $\gamma = -1$, $\delta = 0.01$, and the initial conditions $x_0 = 100$ and $y_0 = 25$. Then change some coefficients and initial values and explain what happens. What happens if you choose $\alpha = 0.25$, $\beta = -0.01$, $\gamma = -0.25$, $\delta = 0.01$? What can one say about the matrix

\[
A = \begin{bmatrix}
\alpha & \beta \\
\gamma & \delta
\end{bmatrix}
\]

for this choice of parameters? Plot the computed solutions. □