5.3 The Explicit Euler Method

The construction of numerical methods for initial value problems as well as basic properties of such methods shall first be explained for the simplest method: The explicit Euler method. Be aware that this method is not the most efficient one from the computational point of view. In later sections, when a basic understanding has been achieved, computationally efficient methods will be presented.

5.3.1 Derivation of the Explicit Euler Method

A general principle to derive numerical methods is to "discretize" constuctions like derivatives, integrals, etc.
Given an initial value problem

$$\dot{y}(t) = f(t, y(t)), \quad y(t_0) = y_0$$

the operation which can not be evaluated numerically obviously is the limit $h \to 0$, that defines the derivative

$$\dot{y}(t) = \lim_{h \to 0} \frac{y(t+h) - y(t)}{h}.$$

However, for any positive (small) h, the finite difference

$$\frac{y(t+h) - y(t)}{h}$$

can easily be evaluated. By definition, it is an approximation of the derivative $\dot{y}(t)$. Let us therefore approximate the differential equation $\dot{y}(t) = f(t, y(t))$ by the difference equation

$$\frac{u(t+h) - u(t)}{h} = f(t, u(t)).$$

Given u at time t, one can compute u at the later time $t + h$, by solving the difference equation

$$u(t+h) = u(t) + hf(t, u(t)).$$

This is exactly one step of the *explicit Euler method* Introducing the notation $t_{n+1} = t_n + h$ and $u_n = u(t_n)$ it reads

$$u_{n+1} = u_n + hf(t_n, u_n), \quad u_0 = y_0. \tag{5.2}$$

We shall see in Sect. 5.5 that u_n really is a first order approximation to the exact solution $y(t_n)$

$$\|u_n - y(t_n)\| = \mathcal{O}(h) \quad h \to 0.$$

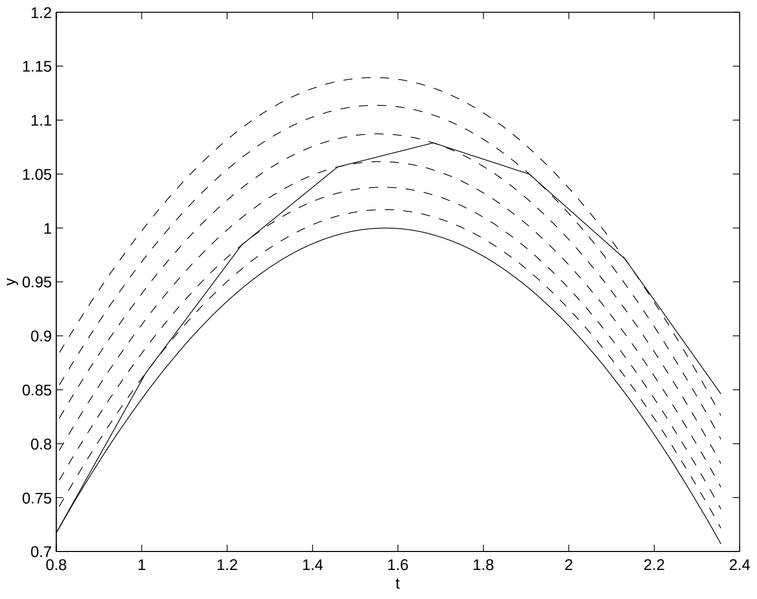

Figure 5.1: Explicit Euler Method

5.3.2 Graphical Illustration of the Explicit Euler Method

Given the solution $y(t_n)$ at some time t_n, the differential equation $\dot{y} = f(t, y)$ tells us “in which direction to continue”. At time t_n the explicit Euler method computes this direction $f(t_n, u_n)$ and follows it for a small time step $t_n \to t_n + h$. This is expressed in the formula (5.2) and illustrated in Fig. 5.1. Obviously each step introduces an error and ends up on a different trajectory. A natural question, that will be answered below is: How do these errors accumulate?

5.3.3 Two Alternatives to Derive Euler’s Method

In the first derivation, the derivative was discretized using a finite difference quotient. This is not the only way to construct numerical methods for initial value problems. An alternative view on Euler’s method is based on the reformulation of the problem as an integal equation

$$y(t) = y_0 + \int_{t_0}^{t} f(\tau, y(\tau)) \, \mathrm{d}\tau$$

Then any of the quadrature rules of Chapter 1.4 can be applied to approximate the integral. Choosing the rectangular rule,

$$\int_{t_n}^{t_n+1} f(\tau, y(\tau))\, \mathrm{d}\tau \approx h f(t_n, y(t_n)),$$

we find again Euler's method

$$u_{n+1} = u_n + h f(t_n, u_n).$$

Clearly other quadrature rules will lead to other methods.
Finally a constuction principle based on Taylor expansion shall be explained. To this end, one assumes that the solution of the initial value problem (5.1) can be expanded in a Taylor series

$$y(t_n + h) = y(t_n) + h\dot{y}(t_n) + \mathcal{O}(h^2).$$

Ignoring the second order term and using the differential equation to express the derivative $\dot{y}(t_n)$ leads also to Euler's method

$$u_{n+1} = u_n + h f(t_n, u_n).$$

5.3.4 Testing Euler's Method

From the three derivations it is clear, that Euler's method does not compute the exact solution of an initial value problem. All one can ask for is a reasonably good approximation. The following experiment illustrates the quality of the approximation. Consider the differential equation

$$\dot{y} = -100y.$$

The exact solution is $y(t) = y_0 e^{-100t}$. With a positive initial value y_0 it is positive and rapidly decreasing as $t \to \infty$. The explicit Euler method applied to this differential equation reads

$$u_{n+1} = (1 - 100h)u_n.$$

With a step size $h = 0.1$ the numerical "approximation" $u_{n+1} = -9u_n$

$$u_n = (-9)^n u_0$$

oscillates with an exponentially growing amplitude. It does not approximate the true solution at all. Reducing the step size to $h = 0.001$ however, yields $u_{n+1} = 0.9u_n$ and the numerical solution

$$u_n = (0.9)^n u_0$$

is smoothly decaying.
Another test example is the initial value problem

$$\dot{y} = \lambda(y - \sin(t)) + \cos t, \quad y(\pi/4) = 1/\sqrt{2},$$

where λ is a parameter. First we set $\lambda = -0.2$ and compare the results for Euler's method with two different step sizes $h = \pi/10$ and $h = \pi/20$, see Fig. 5.2. Obviously, the errors decrease with the step size. Setting now $\lambda =$

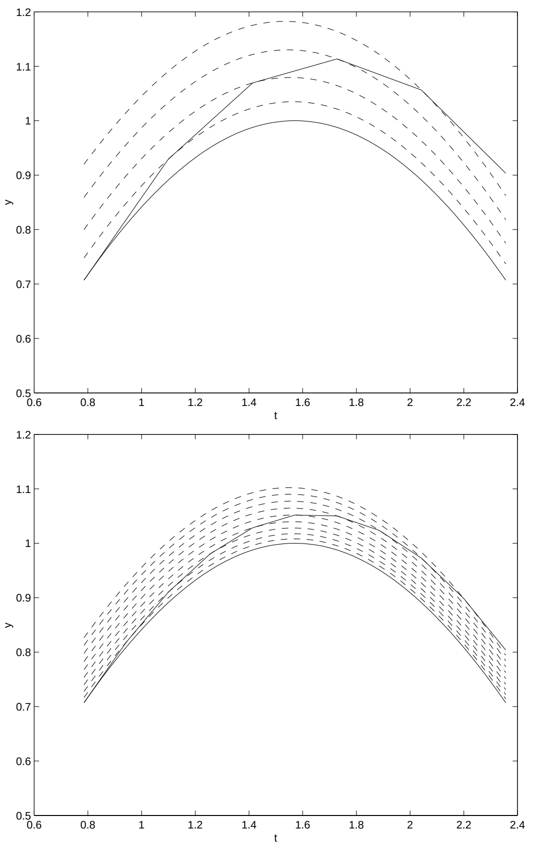

Figure 5.2: Explicit Euler Method, $\lambda = -0.2$, $h = \pi/10$ (left), $h = \pi/20$ (right)

-10 the numerical results for $h = \pi/10$ oscillate arround the true solution and the errors grow rapidly in every single step. For the reduced step size $h = \pi/20$ however, Euler's method gives a quite good approximation to the true solution, see Fig. 5.3.

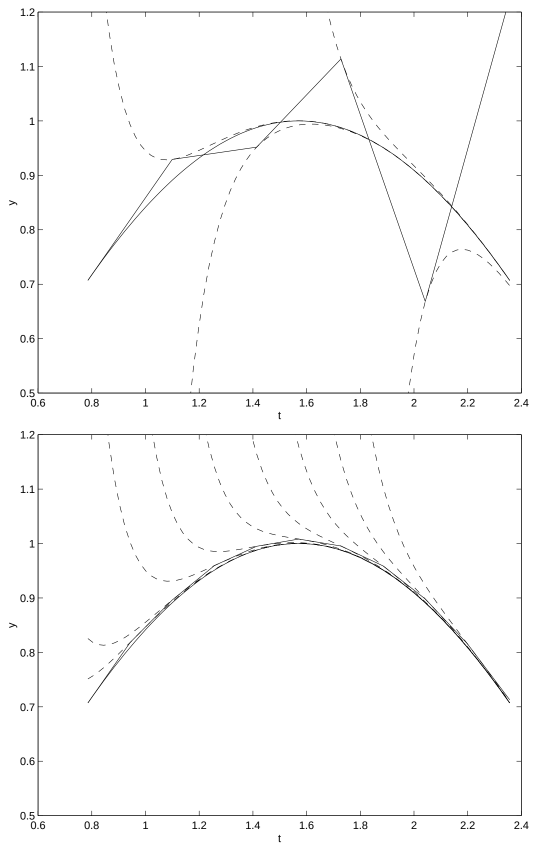

Figure 5.3: Explicit Euler Method, $\lambda = -10$, $h = \pi/10$ (left), $h = \pi/20$ (right)

5.4 Stability Analysis

The previous section showed that in order to obtain reasonable approximations the step size in Euler's method has to be choosen small enough — how small depends on the differential equation. The goal of the present section is to quantify the condition on the step size. To this end consider the test equation

$$\dot{y} = \lambda y, \tag{5.3}$$

where λ now is a complex parameter. The solution

$$y(t) = y_0 e^{\lambda t}$$

remains bounded

$$|y(t)| = |y_0| \cdot |e^{(\alpha+\mathrm{i}\beta)t}| = |y_0| \cdot |e^{\alpha t}|$$

if $\alpha = \mathrm{Re}\lambda$ is non positive. In this case it is reasonable to ask that the numerical solution remains bounded too. For the explicit Euler method,

$$u_{n+1} = (1 + h\lambda)u_n$$

this demand requires that the *amplification factor* is bounded by one

$$|1 + h\lambda| \leq 1. \tag{5.4}$$

The explicit Euler method is called *stable* for the test equation (5.3) if the step size h satisfies the condition (5.4). In the case of real and negative λ, this means $h \leq -2/\lambda$, cf. the experiments in the previous section.
The set

$$\mathcal{S} = \{h\lambda \in \mathbb{C} : |1 + h\lambda| \leq 1\}$$

is called the *stability region* of the Euler method. It is a disc of radius 1 centered at $(-1, 0)$, see Fig. 5.4.

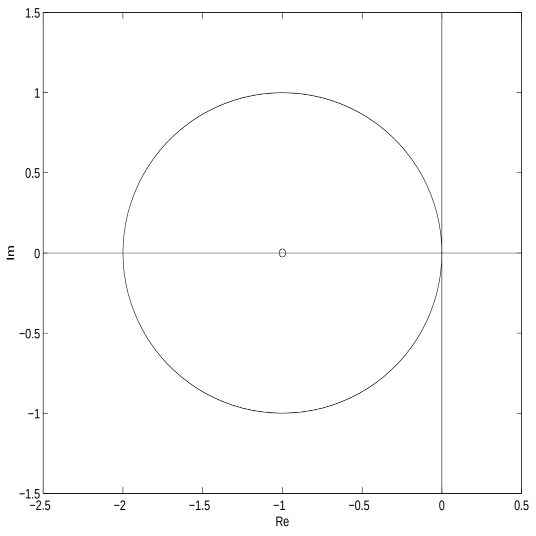

Figure 5.4: Explicit Euler Method, stability region

5.5 Local, Global Errors and Convergence

So far qualitative properties of the approximation (boundedness, monotonicity) have been studied. In this section the actual errors will be analyzed. To this end we have to destinguish local and global errors.

Definition 94 *Given an initial value problem $\dot{y} = f(t, y)$ with $y(0) = y_0$ and numerical approximations $u_n \approx y(t_n)$. The difference*

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

is called the global error*. The difference*

$$\hat{e}_{n+1} = u_{n+1} - \hat{y}(t_{n+1})$$

is called the local error*, where $\hat{y}$ is the solution to $\dot{\hat{y}} = f(t, \hat{y})$ with initial condition $\hat{y}(t_n) = u_n$.*

Note that the local error is an error in the numerical approximation, that is introcuced in one single time step; at time t_n the values u_n and $\hat{y}(t_n)$ are identical. The global error however, is an error at time t_n that has accumulated during n steps of integration. That is the error that one naturally observes when performing numerical calculation. In order to estimate the global error, we first analyze the local error and then study how local errors accumulate during many integration steps.
Local errors can be analyzed by Taylor expansion. We demonstrate this for the explicit Euler method

$$u_{n+1} = u_n + hf(t_n, u_n).$$

Inserting the initial condition for $\hat{y}$ yields

$$u_{n+1} = \hat{y}(t_n) + hf(t_n, \hat{y}(t_n)). \tag{5.5}$$

Taylor expansion of $\hat{y}$ reads

$$\hat{y}(t_{n+1}) = \hat{y}(t_n) + h\dot{\hat{y}}(t_n) + \frac{h^2}{2}\ddot{\hat{y}}(t_n) + \ldots$$

Using the differential equation for $\hat{y}$ we find

$$\hat{y}(t_{n+1}) = \hat{y}(t_n) + hf(t_n, \hat{y}(t_n)) + \frac{h^2}{2}\ddot{\hat{y}}(t_n) + \ldots \tag{5.6}$$

Subtracting (5.6) from (5.5) gives the local error for the explicit Euler method
:

$$\hat{e}_{n+1} = -\frac{h^2}{2}\ddot{\hat{y}}(t_n) + \ldots$$

The accumulation of all local errors during the time stepping procedure determines the global error which is observed after many iteration steps. To investigate the global error, we subtract the Taylor expansion of the true solution

$$\begin{aligned} y(t_{n+1}) &= y(t_n) + h\dot{y}(t_n) + \frac{h^2}{2}\ddot{y}(t_n) + \dots \\ &= y(t_n) + hf(t_n, y(t_n)) + \frac{h^2}{2}\ddot{y}(t_n) + \dots \end{aligned}$$

from the explicit Euler method

$$u_{n+1} = u_n + hf(t_n, u_n).$$

This gives the global error recursion

$$e_{n+1} = e_n + hf(t_n, y(t_n) + e_n) - hf(t_n, y(t_n)) - \varepsilon_{n+1}, \qquad (5.7)$$

where $\varepsilon_{n+1} = \frac{h^2}{2}\ddot{y}(t_n) + \dots$ is called the *global error increment.* Somewhat inexact, ε is often called "local error". Note that $\hat{e}_{n+1} \neq \varepsilon_{n+1}$, however both terms are of the same order $\|\hat{e}_{n+1}\| = \mathcal{O}(h^2) = \|\varepsilon_{n+1}\|$. Taking norms and using Lipschitz–continuity of f yields

$$\|e_{n+1}\| \leq \|e_n\| + hL_f\|e_n\| + \|\varepsilon_{n+1}\|.$$

To get an explicit bound for $\|e_n\|$, we apply the following result.

Lemma 95 (discrete Gronwall lemma) *Let $a_{n+1} \leq (1 + h\mu)a_n + b$ with $h > 0$, $\mu > 0$, $b > 0$ and $a_0 = 0$. Then*

$$a_n \leq \frac{b}{h}\frac{e^{t_n\mu} - 1}{\mu}, \quad t_n = nh.$$

For the global error of the Euler method we find the bound

$$\|e_n\| \leq \frac{h \max \|\ddot{y}\|}{2} \frac{e^{t_n L_f} - 1}{L_f}.$$

For any fixed time level $t_n = nh$, the global error decreases linearly with h :

$$\|e_n\| = \mathcal{O}(h) \text{ as } h \to 0.$$

We say that Euler's method is *convergent of order 1.* More precisely:

Definition 96 *The order of convergence of a method is p, if the global error satisfies*

$$\|e_n\| = \mathcal{O}(h^p), \quad h \to 0.$$

Note that the global error increment for Euler's method is of second order $\|\varepsilon_n\| = \mathcal{O}(h^2)$. The accumulation of this increment over $n = \mathcal{O}(h^{-1})$ steps causes the the order of the global error to decrease by one. The same effect was also observed for quadrature errors in Section 1.4, c.f. Example 33. It is also interesting to compare the error accumulation in quadrature formulas, which consists in simply summing up local errors, with the non–linear recursion (5.7).

5.6 Stiffness

The explicit Euler method is always stable for the test equation $\dot{y} = \lambda y$, $\lambda < 0$ when only the step size h is small enough

$$h < -2/\lambda.$$

However, for strongly negative $\lambda \ll -1$, this leads to extremely small step sizes. Small step sizes may be reasonable and hence acceptable if the right hand side of the differential equation is large and the solution has an large gradient. But, a stongly negative λ does not neccessarily imply large gradients (the right hand side depends on $y(t)$ also). Consider the example

$$\dot{y} = \lambda(y - \sin t) + \cos t, \quad \lambda = -50. \tag{5.8}$$

A particular solution is

$$y_p(t) = \sin t.$$

The general solution for the homogenous equation is

$$y_h(t) = ce^{\lambda t},$$

thus the general solution for the nonhomogenous differential equation (5.8) is

$$y(t) = (y_0 - \sin t_0)e^{\lambda(t-t_0)} + \sin t. \tag{5.9}$$

This solution consists of a slowly varying part, $\sin t$ and an exponentially fast decaying *initial layer*

$$(y_0 - \sin t_0)e^{\lambda(t-t_0)}.$$

Generally, differential equations which admit very fast initial layers as well as slow solution components, are called *stiff* problems.
When in (5.9) $y_0 \neq \sin(t_0)$ and $\lambda \ll -1$, then the layer has a large derivative and it is plausible that a numerical method requires small step sizes. However, after only a short time the initial layer has decayed to zero and is no longer visible in the solution (5.9). Then $y(t) \approx \sin t$ and it would be reasonable to use much larger time steps. Unfortunatelly, the explicit Euler method does not allow time steps larger then the stability bound $h < -2/\lambda$. Even if we start the initial value problem for $t_0 = \pi/4$ exactly on the slow component $y_0 = \sin(\pi/4)$ —that means there is no initial layer present— the explicit Euler approximation diverges with $h = 0.3 > -2/\lambda$, see Fig. 5.5. The reason for this effect is as follows. In the first step, the method introduces an local error as $u_1 \neq \sin(t_1)$. Then in the second step the initial layer is activated, the step size is too large, and the error gets amplified.
As it is impossible to avoid local errors, the only way out of this problem is is to construct methods with better stability properties. This leads to implicit methods.

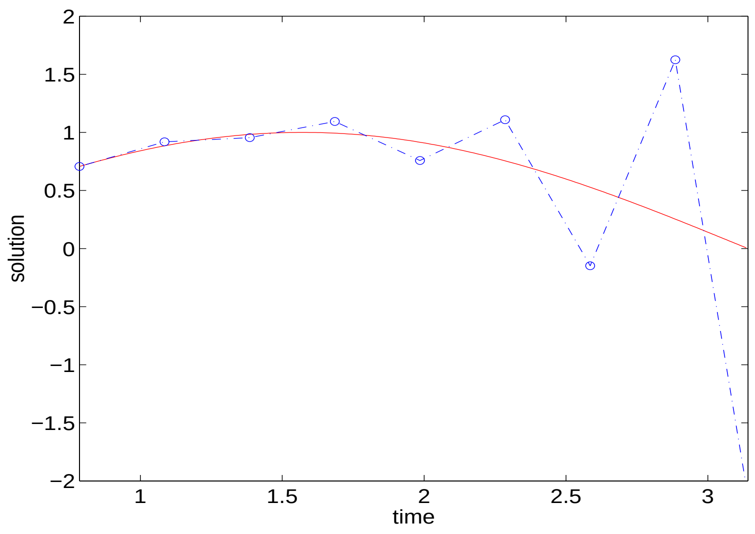

Figure 5.5: Explicit Euler Method

5.7 The Implicit Euler Method

Similar as for the explicit counterpart, there are several ways to derive the implicit Euler method. We begin by discretizing the derivative in $\dot{y}(t) = f(t, y(t))$. For small h it holds that

$$\frac{y(t) - y(t-h)}{h} \approx \dot{y}(t) = \lim_{h \to 0} \frac{y(t) - y(t-h)}{h},$$

thus

$$\frac{y(t) - y(t-h)}{h} \approx f(t, y(t))$$

leading to the scheme

$$u_{n+1} = u_n + hf(t_{n+1}, u_{n+1}). \tag{5.10}$$

This method is known as the *implicit Euler* method. Given $u_n \approx y(t_n)$, a new approximation $u_{n+1} \approx y(t_{n+1})$ is defined by formula (5.10). However, this is an implicit definition for u_{n+1} and one has to solve the nonlinear equation (5.10) to compute u_{n+1}. Clearly, the methods of Ch. 4 can be applied for that task. For example a fixed point iteration applied to (5.10)

$$u_{n+1}^{(j+1)} = u_n + hf(t_{n+1}, u_{n+1}^{(j)}), \quad j = 0, 2, \ldots$$

is easy to compute once an initial guess $u_{n+1}^{(0)}$ is known. However, to find this guess, which may be a rough approximation to u_{n+1}, an explicit Euler step is good enough

$$u_{n+1}^{(0)} = u_n + hf(t_n, u_n).$$

In this context the explicit Euler step is called a *predictor* to the fixed point iteration which is then used as a *corrector* of the approximation. So called *predictor–corrector* algorithms will be discussed in more detail in Sect. 5.8.1. Before analyzing the implicit Euler method let us first give a second explanation. We have seen in Sec. 5.3.3 that numerical methods can also be derived from the integral formulation of the initial value problem

$$y(t) = y_0 + \int_{t_0}^{t} f(\tau, y(\tau))\, d\tau.$$

Approximating the integral by a rectangular rule where the integrand is evaluated at the right end point of the integration intervall (instead of the left one),

$$\int_{t_n}^{t_n+1} f(\tau, y(\tau))\, d\tau \approx h \cdot f(t_{n+1}, y(t_{n+1})),$$

we obtain again Euler's implicit method

$$u_{n+1} = u_n + hf(t_{n+1}, u_{n+1}).$$

5.7.1 Graphical Illustration of the Implicit Euler Scheme

The implicit version of Euler's method uses a numerical gradient in the $n+1^{st}$ step from t_n to t_{n+1} which is equal to the gradient of the true solution in the new approximation u_{n+1}:

$$\frac{u_{n+1} - u_n}{h} = f(t_{n+1}, u_{n+1}).$$

This is illustrated in Fig. 5.6.

5.7.2 Stability Analysis

The main motivation to search for implicit methods rather than explicit ones is to construct a more stable algorithm. Let us therefore check the stability of the implicit version. Consider the test equation

$$\dot{y} = \lambda y$$

and apply the implicit Euler scheme

$$u_{n+1} = u_n + h\lambda u_{n+1}.$$

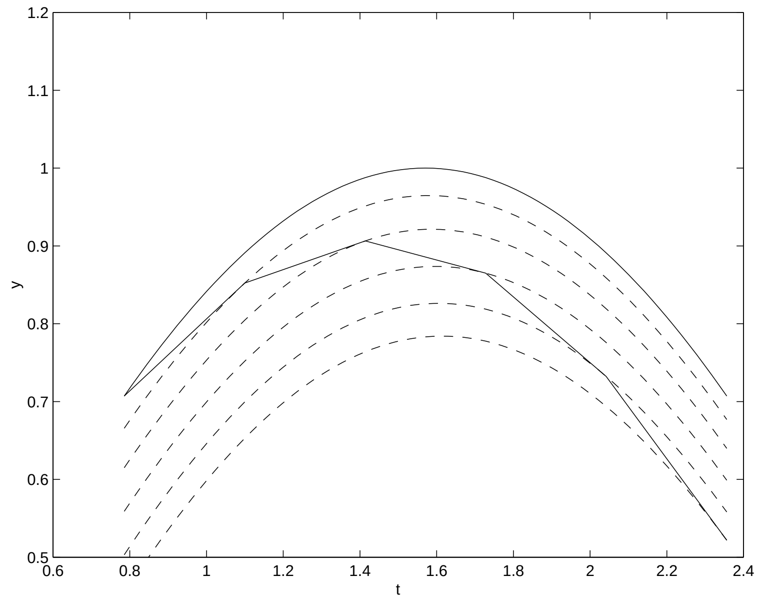

Figure 5.6: Implicit Euler Method

As the test equation is linear, this is easily solved for u_{n+1}

$$u_{n+1} = \frac{1}{1-h\lambda} u_n.$$

The stability condition requires the amplification factor to be bounded

$$\left| \frac{1}{1-h\lambda} \right| \leq 1, \quad \text{Re}\lambda \leq 0.$$

This condition is satisfied for any positive step size h. Hence the implicit Euler method is *unconditionally stable* and the stability region

$$\mathcal{S} = \{h\lambda \in \mathbb{C} : |1 - h\lambda| \geq 1\}$$

includes the entire left half plane.

5.7.3 Testing the Implicit Euler Method

Does the unconditional stability of the implicit method effect practical computations? We return to the initial value problem

$$\dot{y} = \lambda(y - \sin(t)), \quad y(\pi/4) = 1/\sqrt{2}.$$

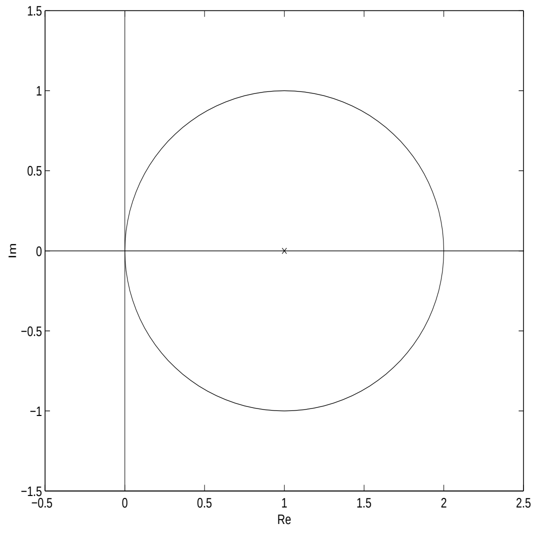

Figure 5.7: Implicit Euler Method, stability region

which could not be approximated by the explicit Euler method in the case $\lambda = -10$ and $h = \pi/10$, see Sect. 5.3.4.
Figs. 5.8 and 5.9 show a stable behaviour of the implicit method independent of the parameters λ and h. Also the errors obviously decrease as $h \to 0$ (with λ fixed). In fact, the unconditionally stable implicit Euler method produces qualitatively correct approximations for all (reasonable) step sizes. Of course the robustness of the method has its price: solving a nonlinear equation in every single step.

5.8 Multistep Methods

5.8.1 Adams Methods

The idea leading to *Adams methods* is quite simple. It is based on transforming the initial value problem

$$\dot{y} = f(t, y) \text{ with } y(t_0) = y_0 \tag{5.11}$$

into its integral form

$$y(t) = y_0 + \int_{t_0}^{t} f(\tau, y(\tau))\, d\tau \tag{5.12}$$