5. Ordinary Differential Equations

Indispensable for many technical applications!

5.1. Introduction

Differential Equations

- One of the most important fields of application of numerical methods are **differential equations**, i.e. equations that specify a relation between functions and their derivatives.
- In **ordinary differential equations (ODE)**, which we will discuss in the following, only one independent variable appears (typically time); simple applications are for example
  - the oscillation of a pendulum

    $$\ddot{y}(t) = -y(t)$$

    with the solution

    $$y(t) = c_1 \cdot \sin(t) + c_2 \cdot \cos(t)\,;$$

  - the exponential growth

    $$\dot{y}(t) = y(t)$$

    with the solution

    $$y(t) = c \cdot e^t\,.$$

- Here, the contrary of “ordinary” is “partial”: In **partial differential equations (PDE)**, multiple independent variables appear (multiple space coordinates or time and space); simple examples of applications are
  - the **Poisson equation** in 2 D, which describes for example the deformation of a membrane is fixed at its boundary under an external load:

$$\Delta u(x,y) \;:=\; u_{xx}(x,y) + u_{yy}(x,y) \;=\; f(x,y) \quad \text{on } [0,1]^2$$

    (here, giving the solutions explicitly becomes a lot harder!);
  - the **heat equation** in 1 D, which describes for example the heat distribution in a metal wire when the temperature at the endpoints is given:

$$u_t(x,t) \;=\; u_{xx}(x,t) \quad \text{in } [0,1]^2$$

    (this is an unsteady equation (time dependency!)).

Scenarios with Differential Equations

- A differential equation itself usually does not specify the solution uniquely (think about the constants in solutions for ordinary differential equations). Additional constraints have to be given (partly circumscribed above in prose, but not yet expressed in formulas):
  - membrane fixed *at the boundary*,
  - given temperature *at the endpoints* of the beam,
  - *initial position* of a pendulum,
  - population number *at the beginning*.
- Such constraints appear as **initial conditions** (for instance the population strength at start) or as **boundary conditions** (after all, a space shuttle should start and touch down at well defined locations).
- Then, the function u is wanted that fulfills the differential equation *and* these conditions. Furthermore, often there is not only a single differential equation but a whole system of differential equations.
- Accordingly, we talk of **initial value problems (IVP)** or **boundary value problems (BVP)**.
- In this chapter, we will deal exclusively with initial value problems of ODE, specifically with

$$\dot{y}(t) \;=\; f(t, y(t))\,, \qquad y(t_0) \;=\; y_0\,.$$

  Here, we need *one* initial condition, because in this case it is an ODE of *first order* (only first derivative).

Analytical Solvability

- In simple cases, ordinary differential equations can be solved analytically:
  - For the example $\dot{y}(t) = y(t)$ above, the solution is obvious.
  - Sometimes, techniques such as the **separation of variables** can help:

$$\begin{aligned}
\dot{y}(t) &= t \cdot y(t) \\
\frac{1}{y(t)} \cdot \frac{\mathrm{d}y}{\mathrm{d}t} &= t \\
\frac{1}{y} \cdot \mathrm{d}y &= t \cdot \mathrm{d}t \\
\int_{y_0}^{y} \frac{1}{\eta} \cdot \mathrm{d}\eta &= \int_{t_0}^{t} \tau \cdot \mathrm{d}\tau \\
\ln(y) - \ln(y_0) &= \frac{t^2}{2} - \frac{t_0^2}{2} \\
y(t) &= y_0 \cdot e^{t^2/2} \cdot e^{-t_0^2/2}.
\end{aligned}$$

  - Obviously, the function $y(t)$ solves the differential equation and fulfills $y(t_0) = y_0$!
  - Caution! This is a negligent approach: Is it allowed to divide by $y(t)$, is the logarithm of $y(t)$ defined at all, etc.?

- In many cases, it is at least possible to say something about the solvability. The **Lipschitz condition** (a property of f and y)

$$\|f(t, y_1) - f(t, y_2)\| \leq L \cdot \|y_1 - y_2\|$$

guarantees existence and uniqueness of the solution for the initial value problem

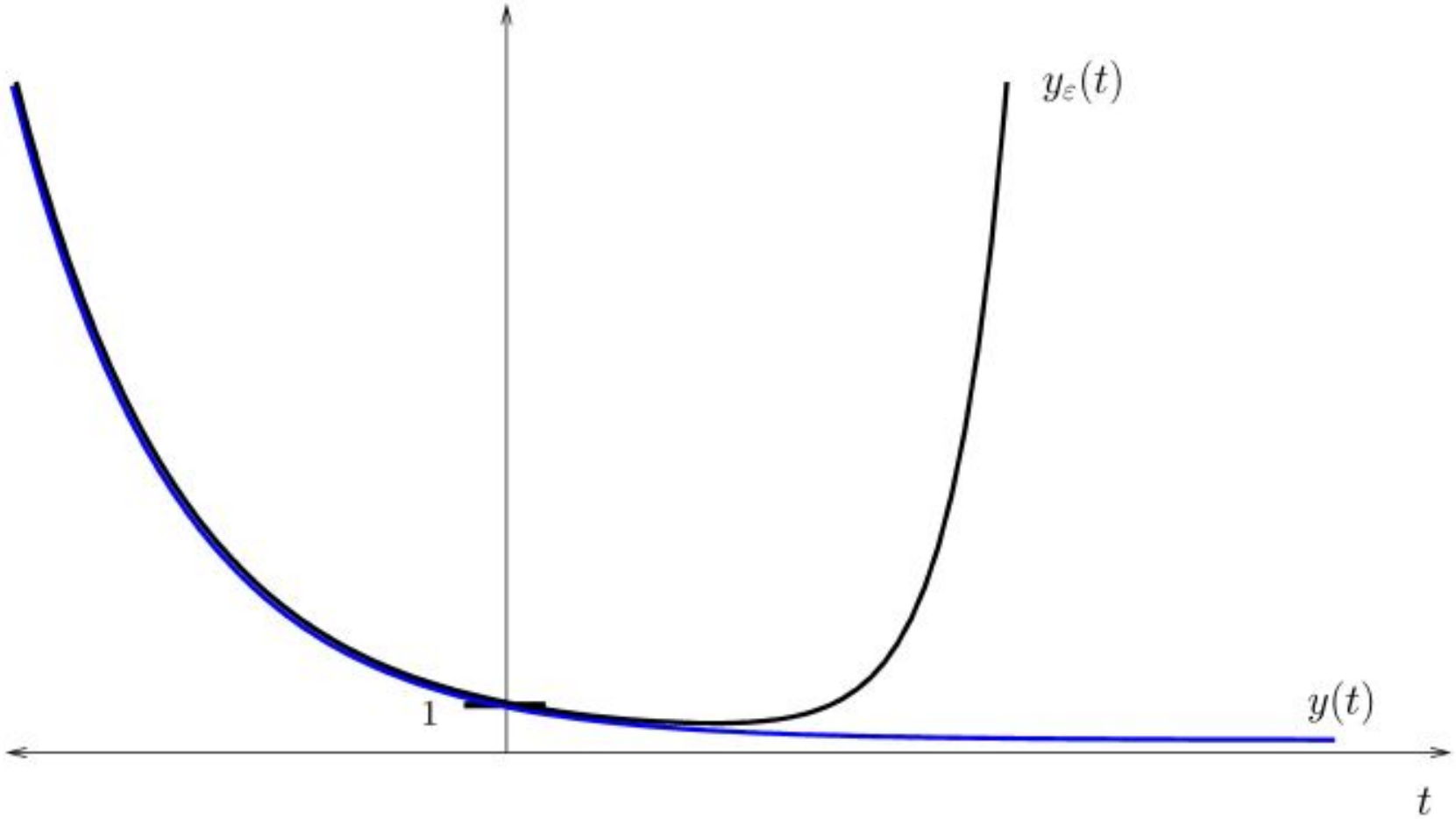
$$\dot{y}(t) = f(t, y(t)), \qquad y(a) = y_a, \qquad t \in [a, b].$$

Condition

- But now to the numerics of initial value problems of ordinary differential equations. Of course, ill-conditioned problems are – as always – the most challenging, so we will keep our hands off in the following. Nevertheless, consider this small example of an ill-conditioned initial value problem as a warning:
  - equation: $\ddot{y}(t) - N\dot{y}(t) - (N+1)y(t) = 0\,, \quad t \geq 0$
  - initial conditions (two because of the second derivative):

$$y(0) = 1, \quad \dot{y}(0) = -1$$

- Solution: $y(t) = e^{-t}$
- Now: perturbed initial condition $y_\varepsilon(0) = 1 + \varepsilon$, everything else as before
- New solution: $y_\varepsilon(t) = (1 + \frac{N+1}{N+2}\varepsilon)e^{-t} + \frac{\varepsilon}{N+2}e^{(N+1)t}$
- As you can see: $y(t)$ and $y_\varepsilon(t)$ have a totally different nature; especially, $y(t)$ converges to zero in case of $t \to \infty$, whereas $y_\varepsilon(t)$ for $N + 1 > 0$ grows infinitely, in fact for arbitrary (i.e. especially even the smallest) $\varepsilon > 0$!
- Smallest perturbations of the input data (here one of the initial conditions) might have a disastrous effect on the solution of the initial value problem – a clear case of very bad condition!



plot of $y(t)$ and $y_\varepsilon(t)$ for $N = 2$ and $\varepsilon = 0.01$

5.2. Approximation of Initial Value Problems with Finite Differences

- In the following, we study the general initial value problem of first order (only first derivative) just mentioned above

$$\dot{y}(t) \;=\; f(t, y(t))\,, \qquad y(a) \;=\; y_a\,, \qquad t \in [a, b]$$

  and assume that it is solvable uniquely.
- If f does not depend on its second argument y, this is a simple integration problem!
- As always, the starting point is the *discretization*, i.e. here: substitute derivatives or *differential coefficients* by *difference quotients* or **finite differences**, respectively, for example

$$\frac{y(t+\delta t) - y(t)}{\delta t} \quad \text{or} \quad \frac{y(t) - y(t-\delta t)}{\delta t} \quad \text{or} \quad \frac{y(t+\delta t) - y(t-\delta t)}{2 \cdot \delta t}$$

  instead of $\dot{y}(t)$ or, for initial value problems of second order,

$$\frac{\frac{y(t+\delta t)-y(t)}{\delta t} - \frac{y(t)-y(t-\delta t)}{\delta t}}{\delta t} = \frac{y(t+\delta t) - 2 \cdot y(t) + y(t-\delta t)}{(\delta t)^2}$$

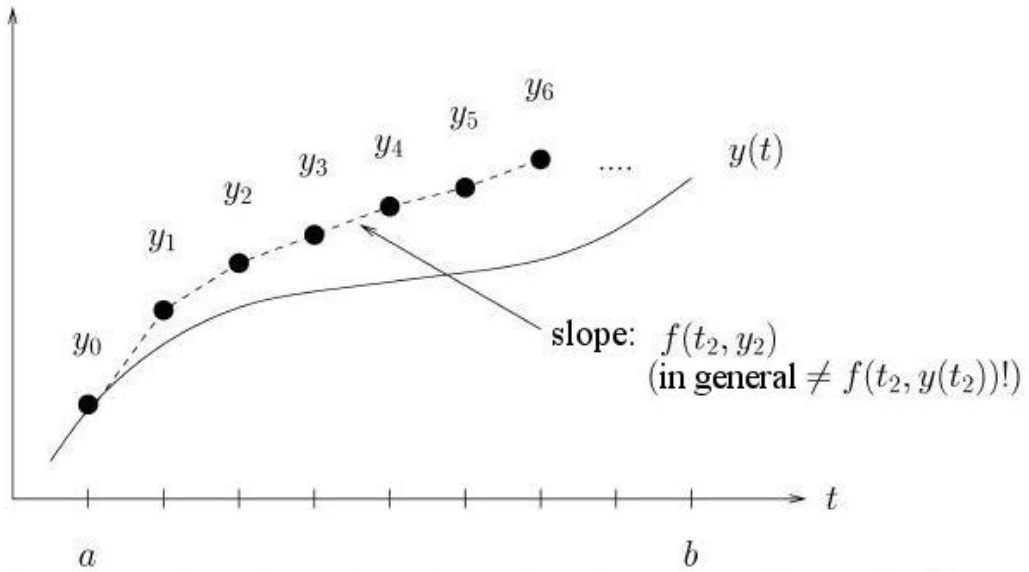
  instead of $\ddot{y}(t)$ and so on.

- The first of the approximations for $\dot{y}(t)$ above leads to

$$\begin{aligned} y(a+\delta t) &\approx y(a) + \delta t \cdot f(a, y(a))\,, \qquad \text{i.e.} \\ y_{k+1} &:= y_k + \delta t \cdot f(t_k, y_k)\,, \\ t_k &= a + k\delta t\,,\ k = 0, 1, \ldots, N\,,\ a + N \cdot \delta t = b \end{aligned}$$

  as the simplest method to produce discrete approximations y_k for $y(t_k)$.

- At the point t_k, the approximation y_k that we have already calculated is used to determine an approximation for the slope (derivative) of y with the help of f, and the slope is used for an estimation of y in the next point in time t_{k+1}. This method is called **Euler's method**.



The Method of Heun

- In addition to Euler's method just introduced, there are a number of other methods for initial value problems of ODE, for example the method of **Heun**:

$$y_{k+1} := y_k + \frac{\delta t}{2}\left(f(t_k, y_k) + f(t_{k+1}, y_k + \delta t f(t_k, y_k))\right) .$$

  - The basic principle stays unchanged: Take the approximation y_k in t_k that was already computed, determine an approximation for the slope $\dot{y}$ and, out of it, determine an approximation for the value of the solution y in the next point in time t_{k+1} by multiplication with the step width δt.
  - The new thing here is the way the slope is estimated. Euler's method simply uses $f(t_k, y_k)$. Heun's method tries to approximate the slope more precisely in the total interval $[t_k, t_{k+1}]$ by using the average of two estimators for $\dot{y}$ in t_k and in t_{k+1}. The problem of not yet having determined y_{k+1} is avoided by using Euler's estimation as the second argument of f!
  - As you can easily see, the single time step has become more costly (two function evaluations of f, more basic arithmetic operations than with the simple Eulers method). Of course, we hope that we will get something better in return – more about that in the next section.

The Method of Runge and Kutta

- The method of **Runge** and **Kutta** takes another step in this direction:

$$y_{k+1} := y_k + \frac{\delta t}{6}(T_1 + 2T_2 + 2T_3 + T_4)$$

with

$$\begin{aligned} T_1 &:= f(t_k, y_k)\,, \\ T_2 &:= f\left(t_k + \frac{\delta t}{2}, y_k + \frac{\delta t}{2} T_1\right)\,, \\ T_3 &:= f\left(t_k + \frac{\delta t}{2}, y_k + \frac{\delta t}{2} T_2\right)\,, \\ T_4 &:= f(t_{k+1}, y_k + \delta t T_3)\,. \end{aligned}$$

  - This method also follows our basic principle

$$y_{k+1} := y_k + \delta_t \cdot approximation\_of\_the\_slope\,,$$

    however, the calculation of the approximate value of $\dot{y}$ now is yet another bit more complicated.
  - Starting with the simple Euler approximation $f(t_k, y_k)$, four adequate approximate values are determined by skillful nesting, which then – weighted appropriately – are used for the approximation.

  - What is the appeal of this rule that is obviously even more complicated? Its appeal is, of course, the higher accuracy of the discrete approximation for $y(t)$ provided by it, as we will see in the following!

- Note the analogy between the methods of Euler, Heun, and Runge-Kutta on the one hand and their equivalents in numerical quadrature (rectangle rule, trapezoidal rule, and Kepler's rule) on the other hand.

5.3. Consistency and Convergence

- As we have already mentioned a couple of times, complicated rules of discretization provide more accurate approximations. To quantify this, we have to improve our understanding of the concept of *accuracy* of a method for discretization of ordinary differential equations. Two things have to be separated carefully:
  - first, the error which results *locally* at every point t_k, even when no approximations are used, just because we insert the difference quotients used by the algorithm with the exact $y(t)$ instead of the derivatives $\dot{y}(t)$ of the exact solution $y(t)$;
  - second, the error that accumulates totally *globally* during the calculation from a to b, i.e. over the total time interval considered.
- Accordingly, we distinguish between two different types of discretization errors,
  - the **local discretization error** (which is the error that arises newly after every time step, even if the difference quotient was generated with the exact $y(t)$), as well as
  - the **global discretization error** (which is the maximum of how far off base the calculations over the whole time interval are at the end).

The Local Discretization Error

- The **local discretization error** is the maximum error which arises solely from the local transition from differential to difference quotients – even if the exact solution $y(t)$ is always used.
- Therefore, the local discretization error of Euler's method adds up to

$$l(\delta t) := \max_{a \le t \le b-\delta t} \left\{ \left| \frac{y(t+\delta t) - y(t)}{\delta t} - f(t, y(t)) \right| \right\} .$$

- When generating the maximum, we assume the availability of the local solution in every point and consider the local errors arising due to "differences instead of derivatives".
- If $l(\delta t) \longrightarrow 0$ for $\delta t \longrightarrow 0$, then the discretization scheme is called **consistent**.
- Consistency obviously is the minimum that has to be required. A non-consistent discretization isn't of any use: If the approximation is not even feasible locally in every time step (for example when the denominator rather contains $(\delta t)^2$ or $2\delta t$ instead of δt) and therefore increasing computational effort does not lead to increasingly better results, it cannot be expected that the given initial value problem is solved reasonably.

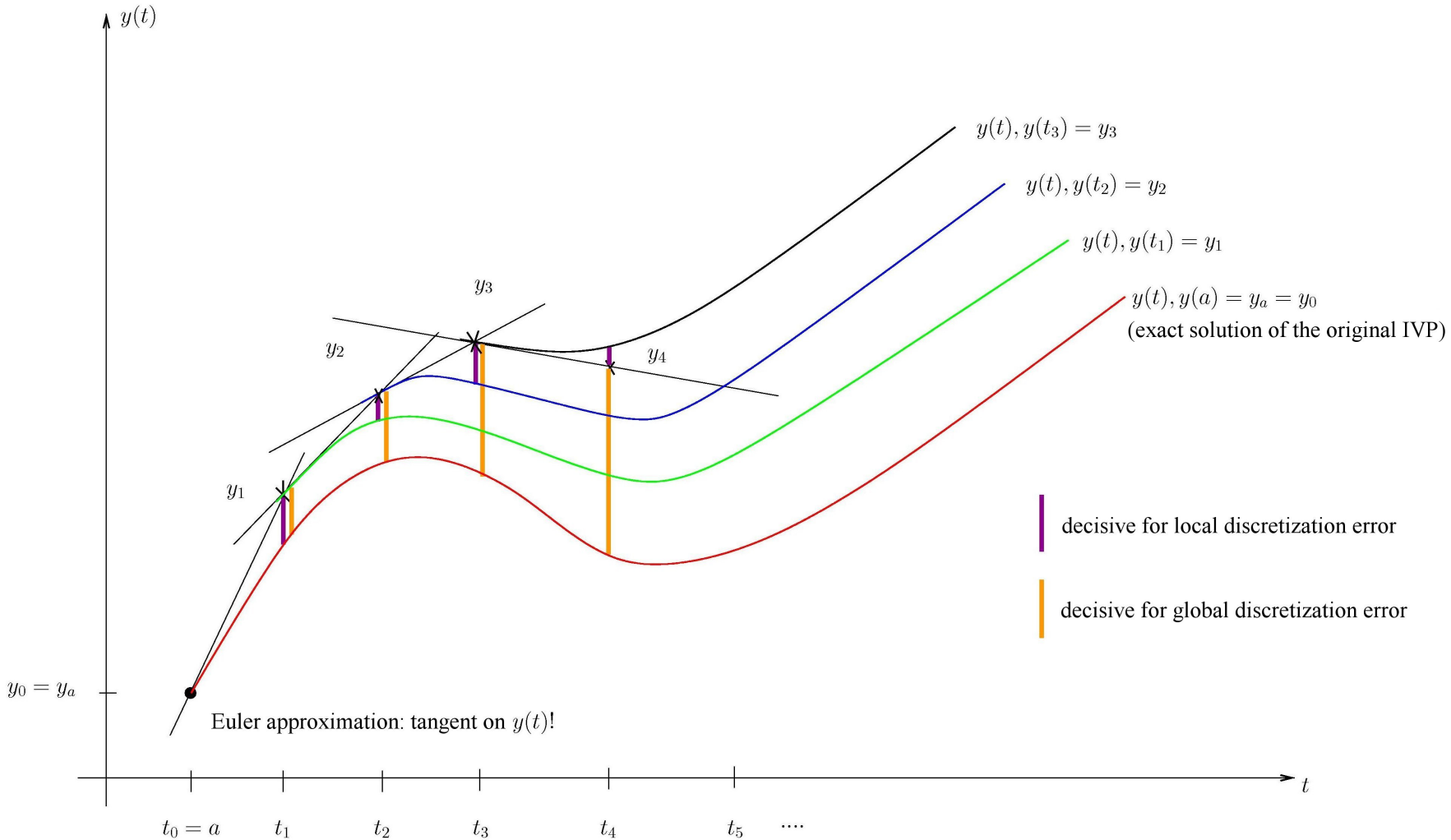
The Global Discretization Error

- The **global discretization error** is the maximum error between the computed approximations y_k and the corresponding values $y(t_k)$ of the exact solution $y(t)$ at the discrete points in time t_k:

$$e(\delta t) := \max_{k=0,\ldots,N} \{|y_k - y(t_k)|\} .$$

- Of course, this is the really interesting value – the global discretization error indicates how good the approximation provided by our method is in the end!
- If $e(\delta t) \to 0$ for $\delta t \to 0$, then the discretization scheme is called **convergent**. Investing more and more computational effort (increasingly smaller steps in time δt) will then lead to increasingly better approximations for the exact solution (infinitesimal error).
- Consistency is the weaker of those two terms, of rather technical nature, and often easy to prove. Convergence, in contrast, is the stronger term (convergence implies consistency, but not the other way round!), of fundamental practical importance, and often not that trivial to show.

Local and Global Discretization Error in Comparison



Consistency and Convergence of the Methods of Euler, Heun, and Runge-Kutta

- All three methods introduced so far are consistent and convergent:
  - **Euler**: method of *first order*, i.e.

$$l(\delta t) = O(\delta t)\,, \qquad e(\delta t) = O(\delta t)\,;$$

  - **Heun**: method of *second order*, i.e.

$$l(\delta t) = O((\delta t)^2)\,, \qquad e(\delta t) = O((\delta t)^2)\,;$$

  - **Runge-Kutta**: method of *fourth order*, i.e.

$$l(\delta t) = O((\delta t)^4)\,, \qquad e(\delta t) = O((\delta t)^4)\,.$$

- Here, the discrepancy in quality becomes apparent: The higher the order of a method, the more effective is an increase in effort. For example, when halfing the step width δt, the error of Euler or Heun or Runge-Kutta is reduced asymptotically by a factor 2, 4, or 16, respectively.
- The expensive methods are (at least asymptotically) the more efficient ones.

- Of course, we are not satisfied yet. The number of *evaluations* of the function f for different arguments has strongly increased (see the Runge-Kutta formulas: T_2, T_3, and T_4 each require an additional evaluation of f). In numerical practice, f is typically very complicated (often another differential equation has to be solved for a single evaluation of f), such that even a single evaluation of f involves high computational effort.
- Therefore, we will learn about an additional approach in the next section.

5.4. Multistep Methods

- The methods so far all are so-called **one-step methods**: For the computation of y_{k+1} no points in time dating back further than t_k are used (but – as said – new evaluation points instead).
- Not so for **multistep methods**: Here, no additional evaluation points of f are being produced, but rather older function values (that are already computed) are re-used, for example in t_{k-1} for the **Adams-Bashforth method of second order**:

$$y_{k+1} := y_k + \frac{\delta t}{2}\left(3f(t_k, y_k) - f(t_{k-1}, y_{k-1})\right)$$

(the consistency of second order can be shown easily).

- Methods of even higher order can be constructed analogously by falling back on points of time t_{k-i}, $i = 1, 2, ...$, dating back even further.
- The principle used here is a good old 'friend' from quadrature: Replace f by a polynomial p of adequate degree that interpolates f in the (t_i, y_i) considered and then use this p according to

$$y_{k+1} := y_k + \int_{t_k}^{t_{k+1}} \dot{y}(t)\mathrm{d}t = y_k + \int_{t_k}^{t_{k+1}} f(t, y(t))\mathrm{d}t \approx y_k + \int_{t_k}^{t_{k+1}} p(t)\mathrm{d}t\,,$$

to compute y_{k+1} (the polynomial p can be integrated easily).

- At the beginning, i.e. as long as there are not enough "old" values, an adequate one-step method is usually used.
- The multistep methods that are based on this principle of interpolation are called **Adams-Bashforth methods**.

The Concept of Stability

- The methods of Euler, Heun, and Runge-Kutta are consistent and convergent.
- This also holds for the class of multistep methods of Adams-Bashforth type that we have just introduced: they are consistent as well as convergent, too.
- Nevertheless, multistep methods show that consistency and convergence do not always apply at the same time. To be convergent, a consistent method additionally has to be **stable** (in terms of our definition of a numerically stable algorithm from chapter 1).
- Therefore, it is extremely important to verify stability. A detailed discussion would go too far at this point. However, you should remember the following rule of thumb:

$$\text{consistency} \quad + \quad \text{stability} \quad \Rightarrow \quad \text{convergence}\,.$$

- To experience the effects of instability, we consider the **midpoint rule** as discretization rule:

$$y_{k+1} := y_{k-1} + 2\delta t f_k \,.$$

  - The midpoint rule obviously is a two-step method.
  - It's easy to see its consistency, as the difference quotient

$$\frac{y_{k+1} - y_{k-1}}{2\delta t}$$

    really converges to the first derivative of y in t_k and, therefore, to f for $\delta t \to 0$.
  - The order of consistency is 2 (easy to show via Taylor expansion).

An Example of Instability

- Now, we will apply the midpoint rule to the following initial value problem:

$$\dot{y}(t) \;=\; -2y(t) + 1\,, \qquad y(0) \;=\; 1\,, \qquad t \geq 0\,,$$

with the solution

$$y(t) \;=\; \frac{1}{2}(e^{-2t} + 1)\,.$$

- We therefore get the method

$$y_{k+1} \;:=\; y_{k-1} + 2\delta t(-2y_k + 1) \;=\; y_{k-1} - 4\delta t y_k + 2\delta t\,, \qquad y_0 \;=\; 1\,.$$

- With the exact values $y(0)$ and $y(\delta t)$ as starting values, the algorithm delivers the following results:

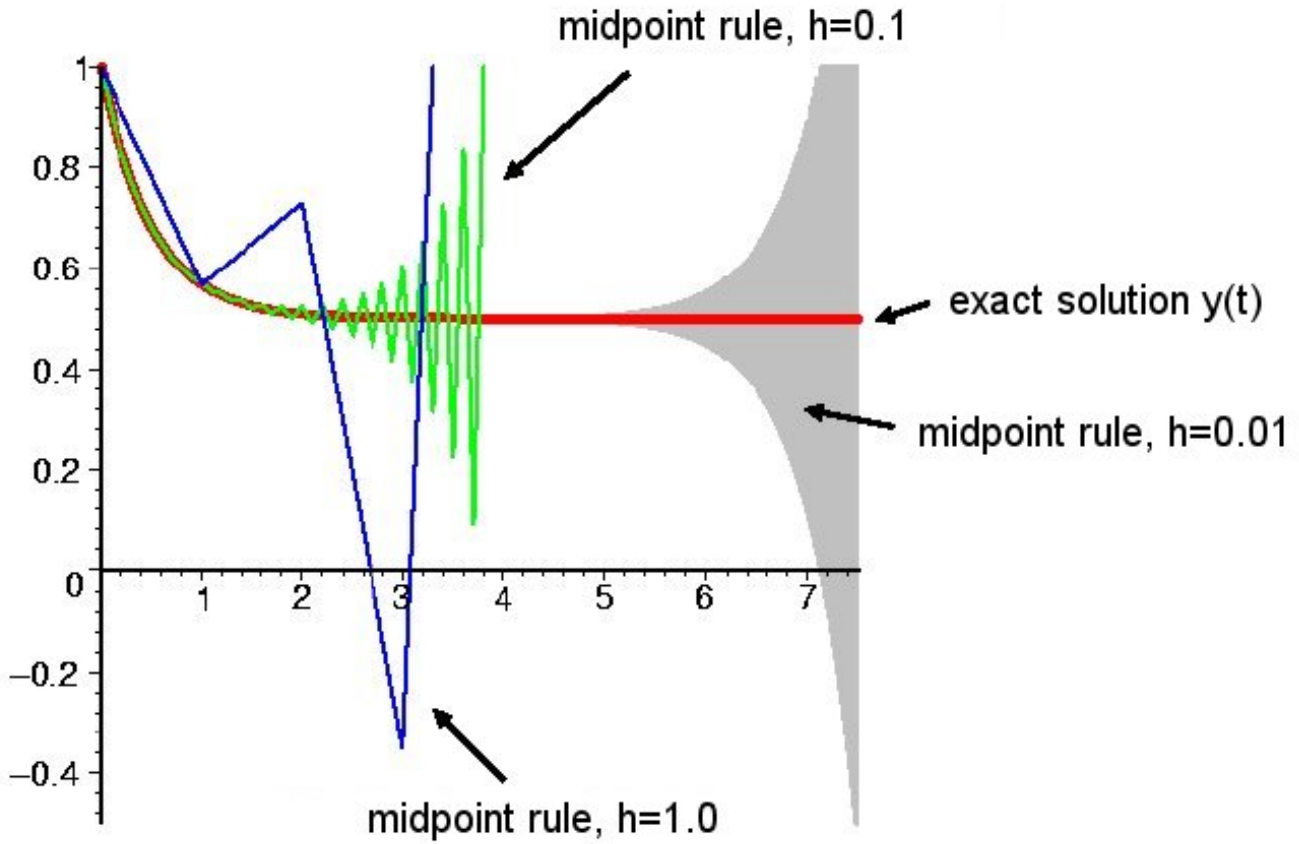
δt	y_9	y_{10}	y_{79}	y_{80}	y_{999}	y_{1000}
1.0	-4945.9	20953.9				
0.1	0.5820	0.5704	-1725.3	2105.7		
0.01	0.9176	0.9094	0.6030	0.6010	-154.6	158.7

- I.e.: For every (arbitrarily small) step width δt the sequence of the calculated y_k oscillates with unbounded absolute values for $k \to \infty$, instead of converging to the exact solution $1/2$. Therefore, we indeed have consistency, but obviously *no convergence* – the midpoint rule is not a stable algorithm!

An Example for Stability and Midpoint Rule

- Now, the whole thing as a chart:

  The IVP has been $\dot{y}(t) = -2y(t) + 1, \quad y(0) = 1,$ **solution** $y(t) = \dfrac{e^{-2t+1}}{2}$



5.5. Stiff Problems, Implicit Methods

- The last phenomenon we will examine is the one of **stiff** differential equations.
- For an introduction, we will consider another simple initial value problem:
  - equation: $\dot{y} = -1000y + 1000\,, \quad t \geq 0$
  - initial condition: $y(0) = y_0 = 2$
  - This initial value problem is well-conditioned: The perturbed initial condition $y(0) = 2 + \varepsilon$, for example, only produces a minimal disturbance of the solution; therefore, we have no excuse considering the problem posed!
  - solution: $y(t) = e^{-1000t} + 1$
  - Discretization method: Euler's method (generally known to be consistent and convergent of first order and numerically stable, therefore, we do not expect anything bad)
- Now, we will program Euler's method for the sample problem from above,

$$y_{k+1} = y_k + \delta t(-1000y_k + 1000) = (1 - 1000\delta t)y_k + 1000\delta t$$

or (complete induction!)

$$y_{k+1} = (1 - 1000\delta t)^{k+1} + 1\,,$$

and, in horror, we notice that inconvenience is impending, if the term in brackets takes total values bigger than 1!

The Phenomenon of Stiffness

- Although the exact solution converges extremely fast towards its limit 1 and is therefore absolutely harmless on the biggest part of the interesting domain, the sequence of y_k above does show oscillations and diverges, if $\delta t > 0.002$.
- I.e. we are forced to use an extremely fine step width, although the characteristics of the exact solution $y(t)$ suggest that this – if at all – is necessary only for values of t close to zero, because of the fast negative exponential decay that occurs there.
- The answer to this mystery is exactly the stiffness:
  - The concepts of consistency, convergence, and stability are of *asymptotic* nature: $\delta t \to 0$ and $O(\delta t)$ always imply "for sufficiently small δt", and in our case this "sufficiently small" is "unacceptably small".
  - This phenomenon is called **stiffness**: For stiff problems, the exact solution includes some really unimportant terms (in our case the exponential term), which, nevertheless, enforce an extremely fine resolution and, thus, are responsible for a ludicrous computational effort.
  - In case of stiff problems, we seem to be at our wits' end. What we have considered to be of advantage (consistency, convergence), does not help us here. Therefore, we need other methods.

Implicit Methods

- **Implicit methods** constitute a remedy for stiff problems:
  - **explicit**: everything up to this point; the formula can be explicitly solved for y_{k+1};
  - **implicit**: the unknown y_{k+1} also appears on the right side of the calculation rule, which, therefore, first has to be solved for y_{k+1} (possibly, this can be only done with an iterative method for solving the implicit equation).
- The simplest example of an implicit algorithm is the so called **implicit Euler method** or **backward Euler method**:

$$y_{k+1} = y_k + \delta t f(t_{k+1}, y_{k+1}) .$$

  Note the appearance of the y_{k+1} (that has to be determined) on the right-hand side of the formula!
- For the stiff problem we observed above, the application of the implicit Euler method leads to

$$y_{k+1} = \frac{y_k + 1000\delta t}{1 + 1000\delta t} = \frac{1}{(1 + 1000\delta t)^{k+1}} + 1 .$$

  As you can see, now, $\delta t > 0$ can be chosen arbitrarily. Thus, the convergence is secured without restriction of the step width δt!

Implicit Methods (2)

- Why that? To put it very simple, explicit methods approximate the solution of an initial value problem with polynomials, implicit methods do the same but with rational functions (you can see this, for example, when you look at the bracket terms of the formula for the stiff problem above). However, polynomials cannot approximate e^{-t} for $t \to \infty$, for example, whereas rational functions can do so very well.
- For stiff differential equations, implicit methods are therefore indispensable.
- However, the formula can not always be solved for y_{k+1} as easily as in our example. Usually, we might need a (nonlinear) iterative method such as Newton's one (see chapter 6) to crack the implicit equation. Often, the **predictor-corrector approach** is also helpful: At first, determine an initial approximation for y_{k+1} with an explicit method and, then, put it into the right-hand side of the implicit equation (actually, this is cheating a little, as we basically calculate explicitly twice, but this approach works very well in many cases).
- Basically, it holds that one implicit time step is more expensive than an explicit one. Still, due to no or only few restrictive step width limitations, we often need a lot fewer time steps with implicit methods.

The Victims of Daring the Gap . . .

- **Systems of ODE**: We only have considered one scalar unknown $y(t)$, but most times they appear in a pack as vectors.
- **ODE of higher order**: We have only considered ODE of first or (in some examples) second order. But there definitively are also ODE that contain higher derivatives.
- **Boundary value problems**: We have only considered initial value problems at which every additional constraint (next to the actual ODE) appear as an initial condition. In many scenarios, however, we have to deal with boundary conditions. For example, when the ideal trajectory of a space shuttle is to be determined, then NASA obviously wants to be able to define the starting and landing point!!
- ...

5.6. Applications of ODE in Computer Science

- **Automatic control engineering**: For quite a long time, computers have been used for control and feedback control of systems and processes (process computers). The mathematical description of such control systems is often based upon ODE – consequently, those have to be solved for successful feedback control.
- **Optimal control**: The optimal control of systems is a task that is essential, e.g. in robotics: Robots should move optimally with respect to time or energy and, in a soccer tournament such as the *RoboCup*, maybe even interact as autonomous agents. The modeling of such optimal control problems leads to ODE.
- **Visualization**: After a flow simulation (car in a wind tunnel, tornado prediction, etc.), you would like to visualize the calculated results (i.e. the velocity field). For this purpose, virtual particles are brought into the flow field and their paths are displayed, e.g. to get an impression of the turbulences. The calculation of those paths requires solving ODE.
- **Chip layout**: The description of circuits is based on Kirchhoff's laws, which – for time-variant variables – are no longer simple algebraic equations but ODE. Circuit simulation is therefore an important example of application of ODE.