Contents

Plan

- Derivation of three well-known numerical schemes: forward Euler, backward Euler, trapezoidal rule
- Discussion of explicit and implicit schemes
- How to solve nonlinear algebraic systems
- Higher-order methods: Taylor and explicit Runge-Kutta methods
- Overview of other methods: Implicit RK, Linear Multistep methods

Problem Setting

- We consider first-order systems of ODEs in explicit form

$$u'(t) = f(t, u(t)), \qquad t \in (t_0, t_0 + T], \qquad u(t_0) = u_0 \tag{11}$$

  to determine the unknown function $u : [t_0, t_0 + T] \to \mathbb{R}^d$
- In components this reads:

$$\begin{pmatrix} u_1'(t) \\ \vdots \\ u_d'(t) \end{pmatrix} = \begin{pmatrix} f_1(t, u_1(t), \ldots, u_d(t)) \\ \vdots \\ f_d(t, u_1(t), \ldots, u_d(t)) \end{pmatrix}$$

- The right hand side $f : [t_0, t_0 + T] \times \mathbb{R}^d \to \mathbb{R}^d$ is Lipschitz-continuous

$$\|f(t, u) - f(t, w)\| \leq L(t)\|u - w\|$$

- Thus, the system has a unique solution

Taylor's Theorem

An important tool in the derivation and analysis of numerical methods for ODEs is the following theorem

Theorem 9 (Taylor's Theorem with Lagrangian Remainder)

Let $u : I \to \mathbb{R}$ be $(n+1)$-times continuously differentiable. Then, for $t, t+\Delta t \in I$, it holds

$$u(t+\Delta t) = \sum_{k=0}^{n} \frac{u^{(k)}(t)}{k!}\Delta t^k + \frac{u^{(n+1)}(t+\theta\Delta t)}{(n+1)!}\Delta t^{n+1} \quad \theta \in [0,1].$$

As a consequence of Taylor's theorem we have for $n = 1$

$$u(t+\Delta t) = u(t) + u'(t)\Delta t + \frac{u''(t+\xi)}{2}\Delta t^2, \quad 0 \le \xi \le \Delta t$$

Taken component-wise this holds also for vector-valued u

Explicit Euler Method

- Choose N time steps

$$t_0 < t_1 < t_2 < \ldots < t_{N-1} < t_N = t_0 + T, \qquad \Delta t_i = t_{i+1} - t_i$$

- y_i^N denotes the approximation of $u(t_i)$ *computed with N steps*
- Take Taylor, use ODE and omit error term to obtain the *explicit Euler approximation*

$$\begin{aligned} u(t_{i+1}) &= u(t_i) + \Delta t_i u'(t_i) + \frac{u''(t_i + \xi_i)}{2} \Delta t_i^2 \\ &= u(t_i) + \Delta t_i f'(t_i, u(t_i)) + \frac{u''(t_i + \xi_i)}{2} \Delta t_i^2 \\ \Rightarrow \quad y_{i+1}^N &= y_i^N + \Delta t_i f(t_i, y_i^N) \end{aligned}$$

- Assuming $y_i^N = u(t_i)$ and subtracting we obtain

$$u(t_{i+1}) - y_{i+1}^N = \frac{u''(t_i + \xi_i)}{2} \Delta t_i^2$$

- *The error after one step is $O(\Delta t^2)$, how does it propagate?*

Implicit Euler Method

- Using Taylor's theorem slightly differently gives

$$\begin{aligned} u(t_i) = u(t_{i+1} - \Delta t_i) &= u(t_{i+1}) - \Delta t_i u'(t_{i+1}) + \Delta t_i^2 \frac{u''(t_{i+1} - \xi_i)}{2} \\ &= u(t_{i+1}) - \Delta t_i f(t_{i+1}, u(t_{i+1})) + \Delta t_i^2 \frac{u''(t_{i+1} - \xi_i)}{2} \\ \Leftrightarrow u(t_{i+1}) - \Delta t_i f(t_{i+1}, u(t_{i+1})) &= u(t_i) - \Delta t_i^2 \frac{u''(t_{i+1} - \xi_i)}{2} \end{aligned}$$

- Which yields the *implicit Euler approximation*

$$y_{i+1}^N - \Delta t_i f(t_{i+1}, y_{i+1}^N) = y_i^N \tag{12}$$

- Need to solve a *nonlinear algebraic equation* to obtain y_{i+1}^N which is computationally much more demanding!
- *Is it worth the effort?*

Local Error in Implicit Euler Method

- We can modify the analysis of the explicit scheme
- From the construction of the scheme we obtain

$$u(t_{i+1}) = u(t_i) + \Delta t_i f(t_{i+1}, u(t_{i+1})) - \Delta t_i^2 \frac{u''(t_{i+1} - \xi_i)}{2}$$

$$y_{i+1}^N = y_i^N + \Delta t_i f(t_i, y_{i+1}^N)$$

- Subtracting, taking norms and using L-continuity gives

$$u(t_{i+1}) - y_{i+1}^N = u(t_i) - y_i^N + \Delta t_i [f(t_{i+1}, u(t_{i+1})) - f(t_i, y_{i+1}^N)] - \Delta t_i^2 \frac{u''(t_{i+1} - \xi_i)}{2}$$

$$\| u(t_{i+1}) - y_{i+1}^N \| \leq \| u(t_i) - y_i^N \| + \Delta t_i L(t_{i+1}) \| u(t_{i+1}) - y_{i+1}^N \| + \frac{\Delta t_i^2}{2} \| u''(t_{i+1} - \xi_i) \|$$

$$\| u(t_{i+1}) - y_{i+1}^N \| \leq \frac{1}{1 - \Delta t_i L(t_{i+1})} \left[\| u(t_i) - y_i^N \| + \frac{\Delta t_i^2}{2} \| u''(t_{i+1} - \xi_i) \| \right]$$

- For $\Delta t < 1/L$ the error after one step is also $O(\Delta t^2)$
- This time step restriction is actually superficial

Implicit Trapezoidal Rule

- Another approach follows from integrating the ODE with the trapezoidal rule

$$u(t_{i+1}) - u(t_i) = \int_{t_i}^{t_{i+1}} f(\xi, u(\xi)) d\xi = \frac{\Delta t_i}{2}[f(t_i, u(t_i)) + f(t_{i+1}, u(t_{i+1}))] + O(\Delta t_i^3)$$

- Resulting in the *implicit trapezoidal rule*

$$y_{i+1}^N - y_i^N = \frac{\Delta t_i}{2}[f(t_i, y_i^N) + f(t_{i+1}, y_{i+1}^N)]$$

$$\Leftrightarrow \quad y_{i+1}^N - \frac{\Delta t_i}{2} f(t_{i+1}, y_{i+1}^N) = y_i^N + \frac{\Delta t_i}{2} f(t_i, y_i^N)$$

- which has an error $O(\Delta t_i^3)$ after one step
- The computational effort is the same as for the implicit Euler method but it is more accurate

$\Rightarrow$ How to solve algebraic systems efficiently?
$\Rightarrow$ How to construct methods with high accuracy systematically?

Fixed Point Iteration in Implicit Methods

- In implicit Euler we need to solve

$$y_{i+1}^N - \Delta t_i f(t_{i+1}, y_{i+1}^N) = y_i^N$$

- Consider the following iteration

$$y_{i+1}^{N,k+1} = y_i^N + \Delta t_i f(t_{i+1}, y_{i+1}^{N,k}) = g(y_{i+1}^{N,k})$$

- For $\Delta t_i < 1/L$ we obtain a contraction since

$$\|g(u) - g(w)\| = \|y_i^N + \Delta t_i f(t_{i+1}, u) - y_i^N - \Delta t_i f(t_{i+1}, w)\| \leq \Delta t_i L \|u - w\|$$

- According to the Banach fixed point theorem the iteration converges then to the unique solution
- Consider the linear, autonomous ODE $u'(t) = f(t, u) = Au$, then $L = \|A\|$ which might be large

Newton Iteration in Implicit Methods

- We rewrite the implicit Euler scheme as

$$F(y_{i+1}^N) = y_{i+1}^N - \Delta t_i f(t_{i+1}, y_{i+1}^N) - y_i^N = 0$$

- Newton's method is based on Taylor expansion of f:

$$\begin{aligned} F(y_{i+1}^{N,k+1}) = F(y_{i+1}^{N,k} + \Delta y) &= y_{i+1}^{N,k} + \Delta y - \Delta t_i f(t_{i+1}, y_{i+1}^{N,k} + \Delta y) - y_i^N \\ &\approx y_{i+1}^{N,k} + \Delta y - \Delta t_i f(t_{i+1}, y_{i+1}^{N,k}) - \Delta t_i \nabla f(t_{i+1}, y_{i+1}^{N,k}) \Delta y - y_i^N = 0 \\ \Rightarrow (I - \Delta t_i \nabla f(t_{i+1}, y_{i+1}^{N,k})) \Delta y &= y_i^N - y_{i+1}^{N,k} + \Delta t_i f(t_{i+1}, y_{i+1}^{N,k}) \end{aligned}$$

- For the update Δy a linear system needs to be solved
- Newton's method for computing y_{i+1}^N reads

$$y_{i+1}^{N,k+1} = y_{i+1}^{N,k} + (I - \Delta t_i \nabla f(t_{i+1}, y_{i+1}^{N,k}))^{-1} (y_i^N - y_{i+1}^{N,k} + \Delta t_i f(t_{i+1}, y_{i+1}^{N,k}))$$

- For $u'(t) = f(t,u) = Au$ this would converge in one iteration without a time step restriction

Discussion of Solution Methods

- Fixed point iteration requires a time step reduction $\Delta t_i \leq q/L$ with $q < 1$ to obtain convergence factor q
- It converges from any initial guess (global convergence)
- **But**: implicit methods are typically used to **avoid** time step restrictions
- Newton's method can often handle much larger time steps
- Its convergence is guaranteed if the initial guess is close enough to the solution (local convergence)
- It can require a globalization strategy such as line search
- Combination of both methods is possible

One Step Methods

- All methods discussed so far are called *one step methods* as they are computing an approximation at t_{i+1} from one at t_i
- The *general one step method* has the form

$$y_{i+1}^N = y_i^N + \Delta t_i F(\Delta t_i, t_i, y_i^N, y_{i+1}^N)$$

- All methods discussed so far can be put in this form
- **Question**: How can we *systematically* construct methods where the error after one step is of the form $O(\Delta t^{p+1})$
- p is called the *order* of the method (not to be confused with the order of the ODE)
- Explicit and implicit Euler are first order methods ($p = 1$)

Taylor Method

- Recall the Taylor expansion:

$$u(t+\Delta t) = \sum_{k=0}^{n} \frac{u^{(k)}(t)}{k!}\Delta t^k + \frac{u^{(n+1)}(\xi)}{(n+1)!}\Delta t^{n+1} \quad \xi \in [t, t+\Delta t]$$

- Differentiating the differential equation $k-1$ times yields:

$$u^{(k)}(t) = \frac{d^{k-1}}{dt^{k-1}} f(t, u(t)) =: f^{k-1}(t, u(t)) \qquad (k \geq 1)$$

- The n-step Taylor method reads

$$u(t+\Delta t) = u(t) + \sum_{k=1}^{n} \frac{\Delta t^k}{k!} f^{k-1}(t, u(t)) + \frac{\Delta t^{n+1}}{(n+1)!} u^{(n+1)}(\xi)$$

- Omitting the remainder term yields the numerical method

$$y_{i+1}^N = y_i^N + \sum_{k=1}^{n} \frac{\Delta t^k}{k!} f^{k-1}(t_i, y_i^N)$$

Why the Taylor Method is Impractical

- We need to compute $f^0(t_i, y_i^N), f^1(t_i, y_i^N), f^2(t_i, y_i^N), \ldots$
- Ok, $k = 1$ is easy:

$$f^0(t_i, y_i^N) = f(t_i, y_i^N)$$

- $k = 2$

$$f_r^1(t, u(t)) = \frac{d}{dt} f_r(t, u(t)) = \frac{\partial f_r}{\partial t}(t, u(t)) + \sum_{s=1}^{d} \frac{\partial f_r}{\partial u_s}(t, u(t)) \frac{du_s}{dt}(t)$$

$$f^1(t, u(t)) = f'(t, u(t)) + \nabla_u f(t, u(t)) f(t, u(t))$$

- $(\nabla_u f(t, u(t)))_{r,s} = \frac{\partial f_r}{\partial u_s}(t, u(t))$ is the Jacobian of f
- We need to compute derivatives of the function f in $d + 1$ variables, i.ee. $d(d + 1)$ derivatives
- For $k = 3$, 2nd derivatives of f need to be computed, together with complicated expressions!

Order and a Way Out

Assuming $y_i^N = u(t_i)$ and subtracting yields:

$$\begin{aligned} u(t_i + \Delta t) - y_{i+1}^N &= u(t_i) + \sum_{k=1}^{n} \frac{\Delta t_i^k}{k!} f^{k-1}(t_i, u(t_i)) + \frac{\Delta t_i^{n+1}}{(n+1)!} u^{(n+1)}(\xi) \\ &\quad - y_i^N - \sum_{k=1}^{n} \frac{\Delta t^k}{k!} f^{k-1}(t_i, y_i^N) \\ &= \frac{\Delta t_i^{n+1}}{(n+1)!} u^{(n+1)}(\xi) \end{aligned}$$

Thus, Taylor's method has the order $p = n$

- **Idea:** It suffices to *approximate* $f^k(t, y_i^N)$ in such a way that the error is at least Δt_i^{n+1}
- This leads to the class of (explicit) **Runge**[2]**-Kutta**[3] methods

[2] Carl Runge, dt. Mathematiker, 1856-1927, Prof. in Hannover and Göttingen
[3] Wilhelm Kutta,dt. Mathematiker, 1867-1944, Prof. in Aachen and Stuttgart

Example: A second-order Method

- Consider $n = 2$ and recall our method:

$$u(t + \Delta t) = u(t) + \Delta t f(t, u(t)) + \frac{\Delta t^2}{2} f^1(t, u(t)) + O(\Delta t^3)$$

- If we approximate (assuming $d = 1$!) by using Taylor 2 times:

$$\begin{aligned} f^1(t, u(t)) &= \frac{1}{\Delta t} \left[f(t + \Delta t, u(t + \Delta t)) - f(t, u(t)) \right] + O(\Delta t) \\ &= \frac{1}{\Delta t} \left[f(t + \Delta t, u(t) + \Delta t f(t, u(t)) + O(\Delta t^2))) - f(t, u(t)) \right] + O(\Delta t) \\ &= \frac{1}{\Delta t} \left[f(t + \Delta t, u(t) + \Delta t f(t, u(t))) - f(t, u(t)) \right] + O(\Delta t) \end{aligned}$$

- we get

$$u(t + \Delta t) = u(t) + \frac{\Delta t}{2} f(t, u(t)) + \frac{\Delta t}{2} f(t + \Delta t, u(t) + \Delta t f(t, u(t))) + O(\Delta t^3)$$

- and from that a 2nd order method called *Heun's method*

Explicit Runge-Kutta Methods

- This suggests the following class of methods:

$$y_{i+1}^N = y_i^N + \Delta t_i \left(b_1 k_1 + \ldots + b_s k_s\right) \quad \text{with}$$

$$k_1 = f(t_i, y_i^N), \qquad k_r = f\left(t_i + c_r \Delta t_i, y_i^N + \Delta t_i \sum_{j=1}^{r-1} a_{rj} k_j\right), \quad r > 1.$$

- s is the *number of stages*
- The k_r can be computed recursively (explicit method)
- The coefficients are collected in a *Butcher tableau*:

$$\begin{array}{c|cccc} 0 & 0 & & \cdots & 0 \\ c_2 & a_{21} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \\ c_s & a_{s1} & \cdots & a_{s,s-1} & 0 \\ \hline & b_1 & \cdots & b_{s-1} & b_s \end{array} = \begin{array}{c|c} c & A \\ \hline & b^T \end{array}$$

Systematic Construction of Runge-Kutta Methods

- Basic idea is comparison of coefficients
- Consider $s = 1$
- Then the method reads

$$y_{i+1}^N = y_i^N + \Delta t_i b_1 f(t_i, y_i^N)$$

  and there is only one parameter to be determined
- From Taylor's expansion we know

$$u(t_{i+1}) = u(t_i) + \Delta t_i f(t_i, u(t_i)) + O(\Delta t_i^2)$$

- By comparison of coefficients we obtain $b_1 = 1$
- *Explicit Euler* is the only RK method of order 1

Systematic Construction of Runge-Kutta Methods II

- For $s = 2$ one obtains (use Taylor expansion)

$$y_{i+1}^N = y_i^N + \Delta t_i \left[(b_1 + b_2) f + \Delta t_i b_2 c_2 \frac{\partial f}{\partial t} + \Delta t_i b_2 a_{21} \frac{\partial f}{\partial u} f \right] (t_i, y_i^N) + O(\Delta t_i^3)$$

$$u(t_{i+1}) = u(t_i) + \Delta t_i \left[f + \frac{\Delta t_i}{2} \frac{\partial f}{\partial t} + \frac{\Delta t_i}{2} \frac{\partial f}{\partial u} f \right] (t_i, u(t_i)) + O(\Delta t_i^3)$$

- By comparison of coefficients we obtain the three conditions

$$b_1 + b_2 = 1 \qquad b_2 c_2 = \frac{1}{2} \qquad b_2 a_{21} = \frac{1}{2}$$

  for four unknown coefficients
- Need to solve *under-determined* nonlinear algebraic system
- Two solutions are

1	1	
	$\frac{1}{2}$	$\frac{1}{2}$

(Heun)

$\frac{1}{2}$	$\frac{1}{2}$	
	0	1

(modified Euler)

Remarks About (Explicit) Runge-Kutta Methods

- With $s = 2$ one can achieve at most order 2
- For $s \leq 4$ the order is $p = s$, for larger s one has $p < s$
- This is true for the scalar case $d = 1$, RK methods can be extended for systems $d > 1$ but the order is in general lower than for scalar equations
- The nonlinear conditions become *very complicated* for larger s. So-called Butcher trees allow a systematic representation of the required derivatives
- Alternatively computer algebra systems are used
- For a given maximum achievable order the number of equations is usually smaller than the number of parameters. This allows for optimization of additional properties, e.g. stability.

Some Example Methods

- $s = 3$, 8 Parameters, 6 conditions

$$\begin{array}{c|ccc} 0 & & & \\ \frac{1}{3} & \frac{1}{3} & & \\ \frac{2}{3} & 0 & \frac{2}{3} & \\ \hline & \frac{1}{4} & 0 & \frac{3}{4} \end{array}$$

Heun's 3rd order method

- $s = 4$, 13 Parameters, 11 conditions

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{2}{6} & \frac{2}{6} & \frac{1}{6} \end{array}$$

THE Runge-Kutta method (order 4)

Other Methods

- Implicit Runge-Kutta Methods
  - Diagonally implicit Runge-Kutta Methods: A is lower triangular. Need to solve s nonlinear systems of dimension d
  - Fully implicit Runge-Kutta Methods: A is full. Need to solve on nonlinear system of size $s \cdot d$
  - These methods may have very good stability properties and high order (e.g. $p = 2s$ for Gauß' method)
- Linear Multistep Methods
  - Use several previous values $y_i^N, y_{i-1}^N, \ldots, y_{i-r}^N$ to compute y_{i+1}^N
  - May be very efficient in terms of f evaluations for a given order
  - Explicit and implicit variants
- Galerkin's method
  - Approximate solution u in finite-dimensional function space, e.g. (trigonometric) polynomials
  - Use variational principle to determine the approximation
  - Good stability properties, a-posteriori error estimates
- Error control and choice of time step Δt

Contents

Problem statement

Recall:

Formulation 2 (Initial value problem - IVP)

Find a differentiable function $y(t)$ for $0 \leq t < T < \infty$ such that

$$y'(t) = f(t, y(t)),$$
$$y(0) = y_0.$$

We recall from yesterday, the ODE model problem:

$$y' = ay, \quad y(0) = y_0, \quad , a \in \mathbb{R}. \tag{13}$$

This ODE has the solution:

$$y(t) = \exp(at)y_0.$$

Stiff problems

Definition 10

An IVP is called **stiff** (along a solution $y(t)$) if the eigenvalues $\lambda(t)$ of the Jacobian $f'_y(t, y(t))$ yield the stiffness ratio:

$$\kappa(t) := \frac{\max_{Re\lambda(t)<0} |Re\lambda(t)|}{\min_{Re\lambda(t)<0} |Re\lambda(t)|} \gg 1.$$

Remark 2

Stiff problems arise often and in particular in combination with time-dependent PDEs. Stiff problems require very well designed numerical algorithms (in terms of stability) as we will see in this lecture.

Stiff problems: Examples

- For $y' = ay$, the eigenvalue corresponds to a. This means: $\lambda = a$. For big (negative) a we therefore need to be a bit careful with the design of our numerical schemes.
- Be careful: in some of the literature, stiffness is often defined for DE systems. In order to have a consistent presentation we nonetheless call the first example 'stiff' when $a \gg 1$.
- As example 2 consider

$$u'(t) = Au(t)$$

with $u(0) = (1, 0, -1)^T$ and

$$A = \begin{pmatrix} -21 & 19 & -20 \\ 19 & -21 & 20 \\ 40 & -40 & -40 \end{pmatrix}$$

Stiff problems: Examples (cont'd)

- Our definition tells us that we have to compute $f'_u(t,u)$. What is this here? Well,

$$f(t,u) = Au(t) \quad \Rightarrow \quad f'_u(t,u) = A.$$

- The eigenvalues of $f'_u(t,u) = A$ are $\lambda_1 = -2$ and $\lambda_{2,3} = -40 \pm 40i$.
- Recall that $i := \sqrt{-1}$, the imaginary part of a complex number.
- The negative real parts are

$$Re(\lambda_1) = -2, \quad Re(\lambda_{2,3}) = -40$$

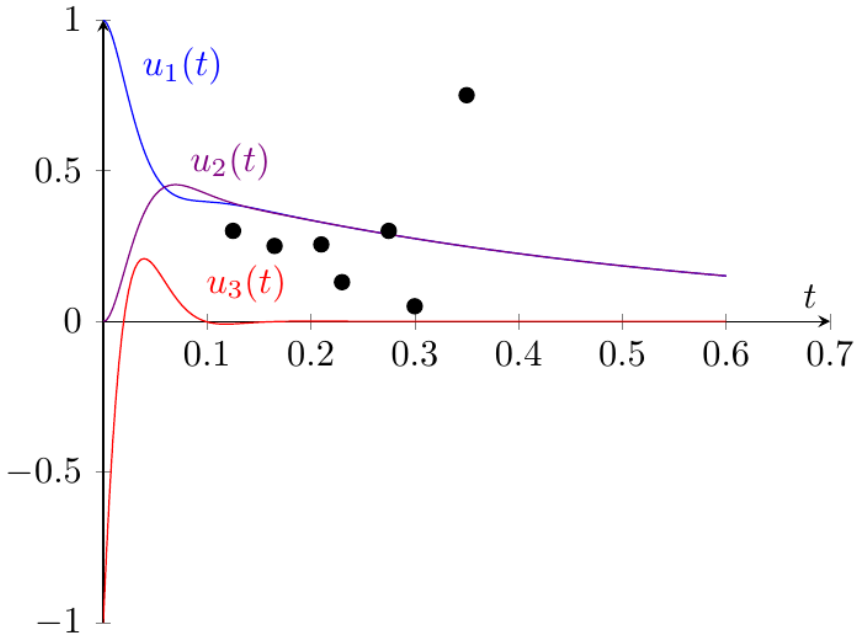
- The stiffness ratio is

$$\kappa(t) = \frac{|-40|}{|-2|} = 20.$$

- Consequently, we consider this as a stiff problem.

Stiff problems: Examples (cont'd)

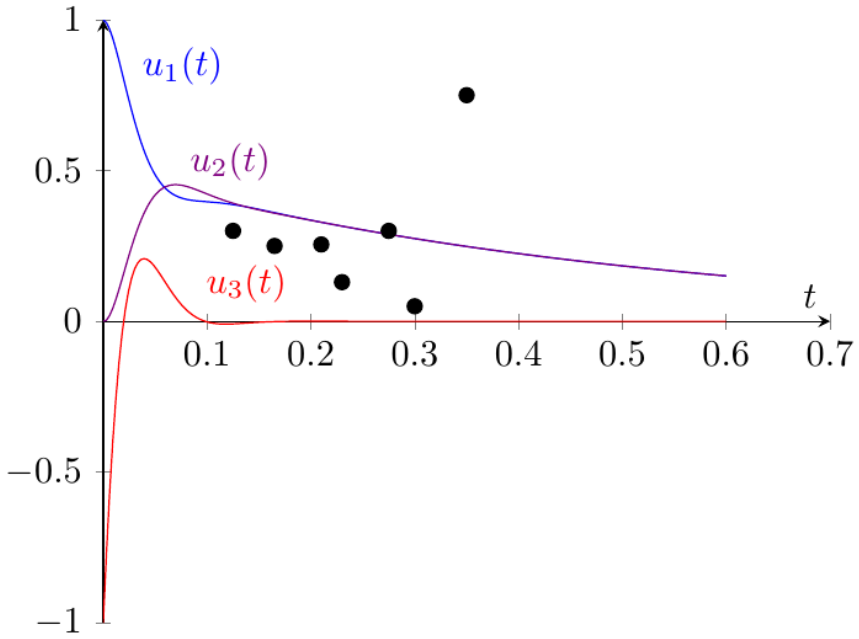
The solution components evolve as follows:



What do we observe?

Stiff problems: Examples (cont'd)

The solution components evolve as follows:



What do we observe?

- We observe that at the beginning we see (physical!) variations
- For $t \geq 0.1$, we observe $u_1 \approx u_2$ and $u_3 \to 0$
- In black dots, an unstable numerical method is used
- Why this occurs and how this can be repaired, we study in the following

Numerical analysis: preliminaries

- In the previous sections, we have constructed algorithms that yield a sequence of discrete solution $\{y_n\}_{n\in\mathbb{N}}$.
- Specifically, in lecture 03, we have seen the first steps how the convergence order can be detected.
- In the numerical analysis our goal is to derive a convergence result of the form

$$\|y_n - y(t_n)\| \leq Ck^\alpha$$

  where α is the order of the scheme.
- This result will tell us that the discrete solution y^n really approximates the exact solution y and if we come closer to the exact solution at which rate we come closer.

Numerical analysis: splitting into stability and consistency

- For the forward Euler method we have:

$$y_n = (1 + k\lambda)y_{n-1} = B_E y_{n-1},$$

  with $B_E := (1 + k\lambda)$.
- Let us write the error at each time point t_n as:

$$e_n := y_n - y(t_n) \quad \text{for } 1 \leq n \leq N.$$

Numerical analysis: splitting into stability and consistency

It holds:

$$
\begin{aligned}
e_n &= y_n - y(t_n), \\
&= B_E y_{n-1} - y(t_n), \\
&= B_E(e_{n-1} + y(t_{n-1})) - y(t_n), \\
&= B_E e_{n-1} + B_E y(t_{n-1}) - y(t_n), \\
&= B_E e_{n-1} + \frac{k(B_E y(t_{n-1}) - y(t_n))}{k}, \\
&= B_E e_{n-1} - k \underbrace{\frac{y(t_n) - B_E y(t_{n-1})}{k}}_{=:\eta_{n-1}}.
\end{aligned}
$$

Numerical analysis: splitting into stability and consistency

Therefore, the error can be split into two parts:

Definition 11 (Error splitting of the model problem)

The error at step n can be decomposed as

$$e_n := \underbrace{B_E e_{n-1}}_{\textit{Stability}} - \underbrace{k\eta_{n-1}}_{\textit{Consistency}} . \tag{14}$$

The first term, namely the stability, provides an idea how the previous error e_{n-1} is propagated from t_{n-1} to t_n. The second term η_{n-1} is the so-called truncation error (or local discretization error), which arises because the exact solution does not satisfy the numerical scheme and represents the consistency of the numerical scheme. Moreover, η_{n-1} yields the speed of convergence of the numerical scheme.

Numerical analysis: stability

We recapitulate (absolute) **stability** and **A-stability**. From the model problem

$$y'(t) = \lambda y(t), \quad y(t_0) = y_0, \ \lambda \in \mathbb{C},$$

we know the solution $y(t) = y_0 \exp(\lambda t)$. For $t \to \infty$ the solution is characterized by the sign of $Re\,\lambda$:

$$\begin{aligned} Re\,\lambda < 0 \quad &\Rightarrow |y(t)| = |y_0| \exp(Re\,\lambda) \to 0, \\ Re\,\lambda = 0 \quad &\Rightarrow |y(t)| = |y_0| \exp(Re\,\lambda) = |y_0|, \\ Re\,\lambda > 0 \quad &\Rightarrow |y(t)| = |y_0| \exp(Re\,\lambda) \to \infty. \end{aligned}$$

For a **good numerical scheme**, the first case is particularly interesting whether such a scheme can produce a bounded discrete solution when the continuous solution is bounded.

Numerical analysis: stability

Definition 12 ((Absolute) stability)

A (one-step) method is absolute stable for $\lambda k \neq 0$ if its application to the model problem produces in the case $Re\,\lambda \leq 0$ a sequence of bounded discrete solutions: $\sup_{n\geq 0} |y_n| < \infty$. To find the stability region, we work with the stability function $R(z)$ where $z = \lambda k$. The region of absolute stability is defined as:

$$SR = \{z = \lambda k \in \mathbb{C} : |R(z)| \leq 1\}.$$

Remark 3

Recall that $R(z) := B_E$.

Remark 4

Nonstability often exhibits non-physical oscillations in the discrete solution. For this reason, it is important to have a feeling for the physics in order to decide whether oscillations are physically-wanted or numerical instabilities. See also Exercise 1 (day 1) and the end of lecture 02.

Numerical analysis: stability

Proposition 5

For the simplest time-stepping schemes forward Euler, backward Euler and the trapezoidal rule, the stability functions $R(z)$ read:

$$R(z) = 1 + z,$$
$$R(z) = \frac{1}{1 - z},$$
$$R(z) = \frac{1 + \frac{1}{2}z}{1 - \frac{1}{2}z}.$$

Numerical analysis: stability

Proof.

We take again the model problem $y' = \lambda y$. Let us discretize this problem with the forward Euler method:

$$\frac{y_n - y_{n-1}}{k} = \lambda y_{n-1} \quad \Rightarrow y_n = (y_{n-1} + \lambda k) y_{n-1} \tag{15}$$

$$= (1 + \lambda k) y_{n-1} = (1 + z) y_{n-1} \tag{16}$$

$$= R(z) y_{n-1}. \tag{17}$$

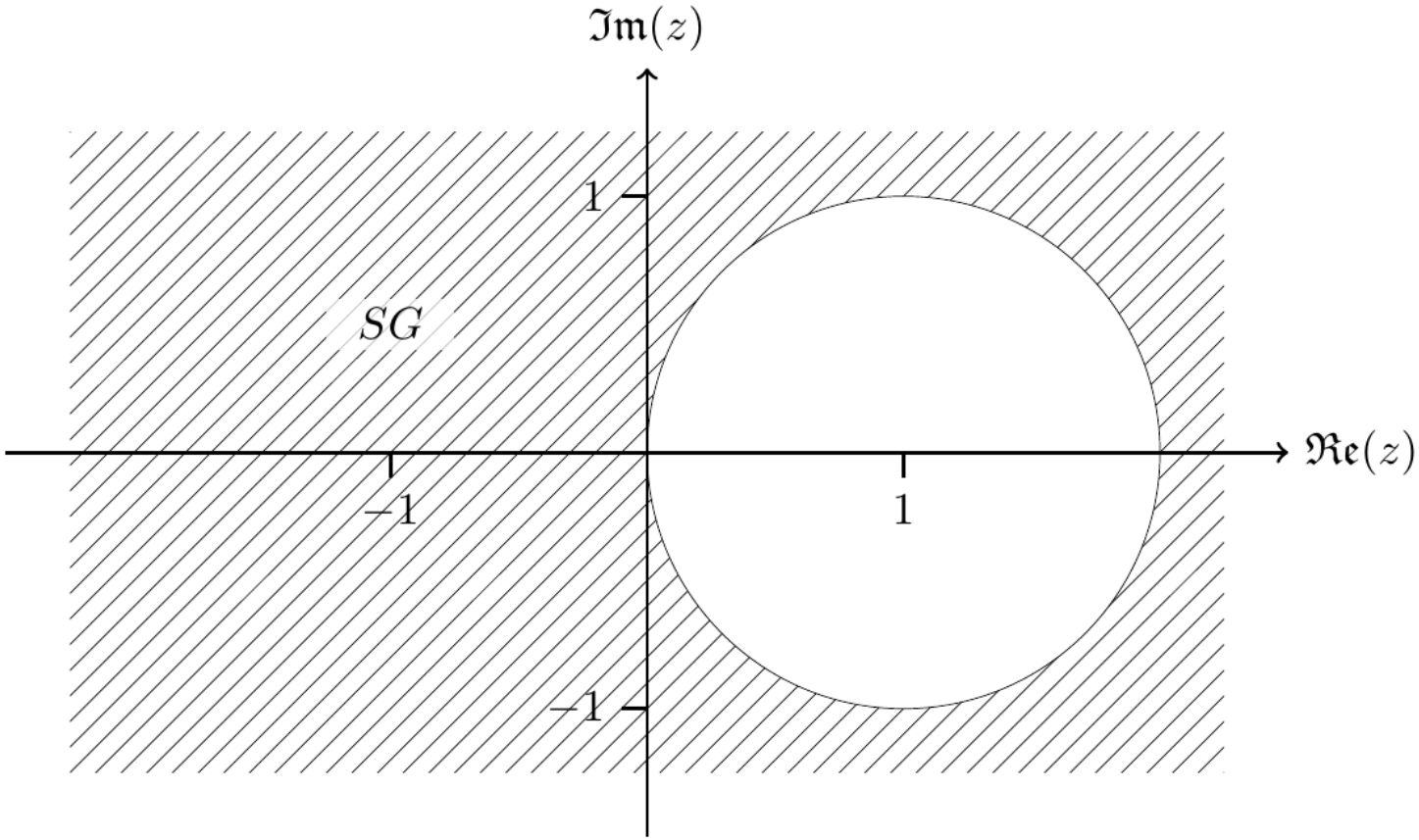
For the implicit Euler method we obtain:

$$\frac{y_n - y_{n-1}}{k} = \lambda y_n \quad \Rightarrow y_n = (y_{n-1} + \lambda k) y_n \quad \Rightarrow y_n = \frac{1}{1 - ak} y_n \tag{18}$$

$$\Rightarrow y_n = \underbrace{\frac{1}{1 - z}}_{=:R(z)} y_n. \tag{19}$$

The procedure for the trapezoidal rule is again the analogous. □

Stability domain backward Euler



Numerical analysis: A-stability

Definition 13 (A-stability)

A difference method is A-stable if its stability region is part of the absolute stability region:

$$\{z \in \mathbb{C} : Re\, z \leq 0\} \subset SR,$$

here Re denotes the real part of the complex number z. A brief introduction to complex numbers can be found in any calculus lecture.

Numerical analysis: A-stability, alternative definition

Definition 14 (A-stability)

Let $\{y_n\}_n$ the sequence of solutions of a difference method for solving the ODE model problem. Then, this method is A-stable if for arbitrary $\lambda \in \mathbb{C}^- = \{\lambda : Re(\lambda) \leq 0\}$ the approximate solutions are bounded (or even contractive) for arbitrary, but fixed, step size k. That is to say:

$$|y_{n+1}| \leq |y_n| < \infty \quad \text{for } n = 1, 2, 3, \ldots$$

Numerical analysis: A-stability

Proposition 6

The explicit Euler scheme cannot be A-stable.

Proof.

For the forward Euler scheme, it is $R(z) = 1 + z$. For $|z| \to \infty$ is holds $R(z) \to \infty$ which is a violation of the definition of A-stability. □

Remark 7

More generally, explicit schemes can never be A-stable.

Proposition 8

The implicit Euler scheme and the trapezoidal rule are *A*-stable.

Numerical analysis: A-stability

We illustrate the previous statements.

1) In Proposition 5 we have seen that for the forward Euler method it holds:

$$y_n = R(z)y_{n-1},$$

where $R(z) = 1 + z$. Thus, according to Definition 13 and 14, we obtain convergence when the sequence $\{y_n\}$ is contracting:

$$|R(z)| \leq |1 + z| \leq 1. \tag{20}$$

- Thus if the value of λ (in $z = \lambda k$) is very big, we must choose a very small time step k in order to achieve $|1 - \lambda k| < 1$.
- Otherwise the sequence $\{y_n\}_n$ will increase and thus diverge (recall that stability is defined with respect to decreasing parts of functions! Thus, the continuous solution is bounded and consequently the numerical approximation should be bounded, too).
- In conclusion, the forward Euler scheme is only conditionally stable, i.e., it is stable provided that (20) is fulfilled.

Numerical analysis: A-stability

2) For the implicit Euler scheme, we see that

- a large λ and large k even both help to stabilize the iteration scheme
- but be careful, the implicit Euler scheme, stabilizes actually too much. Because it computes contracting sequences also for case where the continuous solution would grow.
- Thus, no time step restriction is required.
- Consequently, the implicit Euler scheme is well suited for stiff problems with large parameters/coefficients λ.

Example

We briefly compute the time step restriction for the system from the beginning. We had

$$Re(\lambda_1) = -2, \quad Re(\lambda_{2,3}) = -40.$$

Consequently:

$$|1 + h\lambda| = |1 - 40h| \leq 1.$$

Then, we obtain:

$$1 - 40h \leq 1 \Rightarrow -40h \leq 0 \Rightarrow h = 0.$$

$$-(1 - 40h) \leq 1 \Rightarrow 40h \leq 2 \Rightarrow h = \frac{1}{20}.$$

The first result is useless. The second finding shows that the critical time step size is $h = \frac{1}{20}$. In order to obtain a stable numerical result, we need to work with

$$h < \frac{1}{20}$$

when using the forward Euler scheme.

Stability for higher-order methods: Runge-Kutta

- Consider the Taylor scheme of order R:

$$y_n = y_{n-1} + h\sum_{r=1}^{R} \frac{h^{r-1}}{r!} f^{(r-1)}(t_{n-1}, y_{n-1}) = y_{n-1} + h\sum_{r=1}^{R} \frac{h^{r-1}}{r!} \lambda^r y_{n-1}$$

  where we recall that $f(t, y) = \lambda y$ represents our problem.
- The stability refers to the question whether

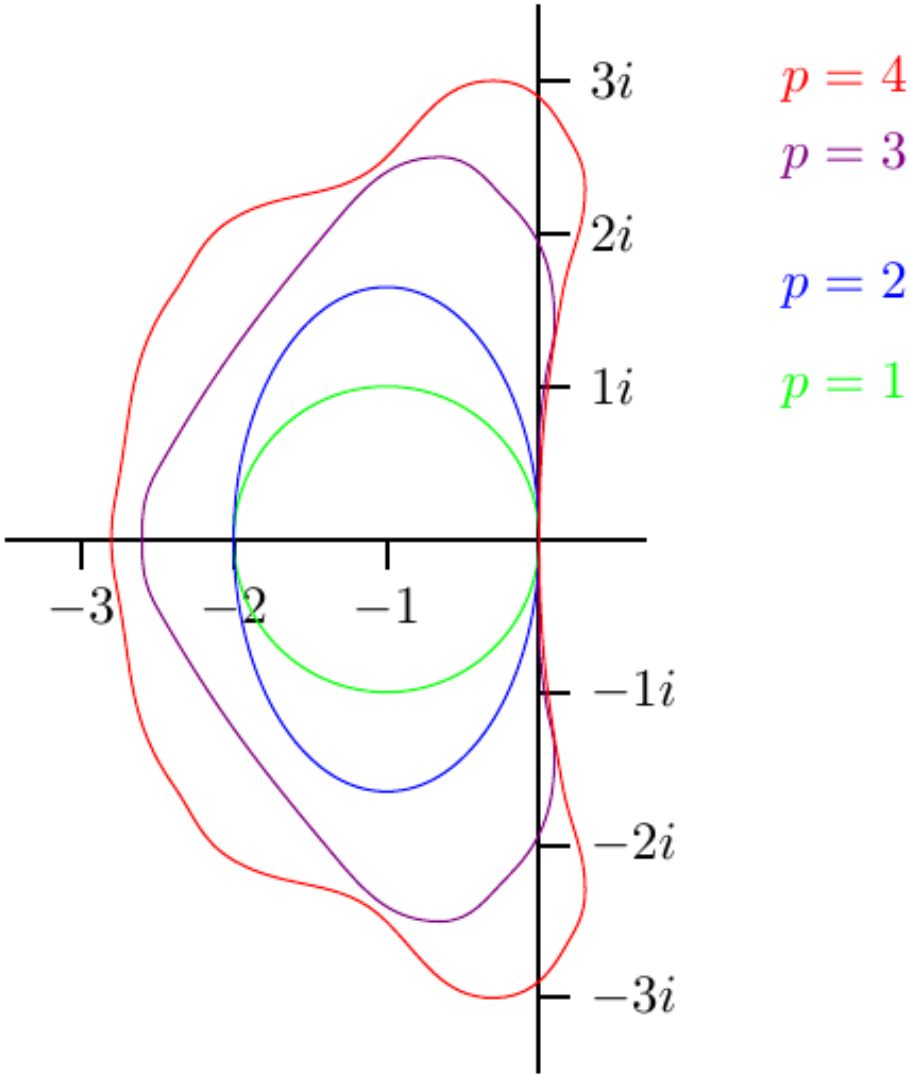
$$|y_n| \leq |y_{n-1}|$$

- Then, the stability factor $\omega(z)$ is given by:

$$\omega(z) = \sum_{r=0}^{R} \frac{z^r}{r!}, \quad z = \lambda h$$

- We then obtain (without any proofs!), the stability regions shown on the next slide.

Stability for higher-order methods: Runge-Kutta



Numerical analysis: consistency, local discretization error

We briefly formally recall Taylor expansion. For a function $f(x)$ we develop at a point $a \neq x$ the Taylor series:

$$T(f(x)) = \sum_{j=0}^{\infty} \frac{f^{(j)}(a)}{j!}(x-a)^j.$$

Let us continue with the forward Euler scheme and let us now specify the truncation error η_{n-1}:

$$y'(t_{n-1}) + \eta_{n-1} = \frac{y(t_n) - y(t_{n-1})}{k}.$$

To this end, we need information about the solution at the old time step t_{n-1} in order to eliminate $y(t_n)$.

Numerical analysis (cont'd)

Thus we use Taylor and develop $y(t^n)$ at the time point t^{n-1}:

$$y(t^n) = y(t^{n-1}) + y'(t^{n-1})k + \frac{1}{2}y''(\tau^{n-1})k^2$$

We obtain the difference quotient of forward Euler by the following manipulation:

$$\frac{y(t^n) - y(t^{n-1})}{k} = y'(t^{n-1}) + \frac{1}{2}y''(\tau^{n-1})k.$$

We observe that the first terms correspond to the forward Euler scheme. The remainder term is

$$\frac{1}{2}y''(\tau^{n-1})k$$

and therefore the truncation error η_{n-1} can be estimated as

$$\|\eta_{n-1}\| \leq \max_{t\in[0,T]} \frac{1}{2}\|y''(t)\|k = O(k).$$

Therefore, the convergence order is k (namely linear convergence speed).

Numerical analysis: convergence

Theorem 15 (Convergence of implicit/explicit Euler)

We have

$$\max_{t_n \in I} |y_n - y(t_n)| \leq c(T, y)k = O(k),$$

where $k := \max_n k_n$.

Remark 9

The following proof does hold for both schemes, except that when we plug-in the stability estimate one should recall that the backward Euler scheme is unconditionally stable and the forward Euler scheme is only stable when the step size k is sufficiently small.

Numerical analysis: convergence (proof)

It holds for $1 \leq n \leq N$:

$$|y_n - y(t_n)| = \|e_n\| = k\|\sum_{k=0}^{n-1} B_E^{n-k}\eta_k\| \leq k\sum_{k=0}^{n-1}\|B_E^{n-k}\eta_k\| \quad \text{(triangle inequality)}$$

$$\leq k\sum_{k=0}^{n-1}\|B_E^{n-k}\|\ \|\eta_k\|$$

$$\leq k\sum_{k=0}^{n-1}\|B_E^{n-k}\|\ Ck \quad \text{(consistency)}$$

$$\leq k\sum_{k=0}^{n-1} 1\ Ck \quad \text{(stability)}$$

$$= kN\ Ck$$

$$= T\ Ck, \quad \text{where we used } k = T/N$$

$$= C(T, y)k$$

$$= O(k)$$

Computational convergence analysis

In order to calculate the convergence order α from numerical results, we make the following derivation.

- Let $P(k) \to P$ for $k \to 0$ be a converging process and assume that

$$P(k) - \tilde{P} = O(k^\alpha).$$

- Here $\tilde{P}$ is either the exact limit P (in case it is known) or some 'good' approximation to it.
- Let us assume that three numerical solutions are known (this is the minimum number of runs if the limit P is not known). That is

$$P_1 := P(k), \quad P_2 := P(k/2), \quad P_3 := P(k/4).$$

- Then, the convergence order can be calculated via the formal approach $P(k) - \tilde{P} = ck^\alpha$ with the following formula.

Computational convergence analysis

Proposition 10 (Computationally-obtained convergence order)

Given three numerically-obtained values P_1, P_2 and P_3, the convergence order can be estimated as:

$$\alpha = \frac{1}{log(2)} log\left(\left|\frac{P_1 - P_2}{P_2 - P_3}\right|\right). \tag{21}$$

The order α is an estimate and heuristic because we assumed a priori a given order, which strictly speaking we have to proof first.

Substantiating our theoretical results: example

We solve our ODE model problem numerically.

Formulation 3

Let $a = g - m$ be $a = 0.25$ (test 1) or $a = -0.25$ (test 2) or $a = -10$ (test 3). The IVP is given by:

$$y' = ay, \quad y(t_0 = 2011) = 2.$$

The end time value is $T = 2014$.

The tasks are:

- Use the forward Euler (FE), backward Euler (BE), and trapezoidal rule (CN) for the numerical approximation.
- Please observe the accuracy in terms of the discretization error.
- Observe for (stiff) equations with a large negative coefficient $a = -10 \ll 1$ the behavior of the three schemes.

Discussion of the results for test 1 $a = 0.25$

In the following, we present our results for the end time value $y(T = 2014)$ for test case 1 ($a = 0.25$) on three mesh levels:

Scheme	#steps	k	y_N	Error (abs.)
FE	8	0.37500	4.0961	0.13786
BE	8	0.37500	4.3959	0.16188
CN	8	0.37500	4.2363	0.0023295
FE	16	0.18750	4.1624	0.071567
BE	16	0.18750	4.3115	0.077538
CN	16	0.18750	4.2346	0.00058168
FE	32	0.093750	4.1975	0.036483
BE	32	0.093750	4.2720	0.037974
CN	32	0.093750	4.2341	0.00014538

Discussion of the results for test 1 $a = 0.25$

- In the second column, i.e., $8, 16, 32$, the number of steps ($=$ number of intervals, i.e., so called mesh cells - speaking in PDE terminology) are given. In the column after, the errors are provided.
- In order to compute numerically the convergence order α with the help of formula (21), we work with $k = k_{max} = 0.375$. Then we identify in the above table that

$$\begin{aligned} P(k_{max}) &= P(0.375) = |y(T) - y_8|, \\ P(k_{max}/2) &= P(0.1875) = |y(T) - y_{16}| \\ P(k_{max}/4) &= P(0.09375) = |y(T) - y_{32}|. \end{aligned}$$

Discussion of the results for test 1 $a = 0.25$

- We monitor that doubling the number of intervals (i.e., halving the step size k) reduces the error in the forward and backward Euler scheme by a factor of 2. This is (almost) linear convergence, which is confirmed by using Formula (21) yielding $\alpha = 1.0921$. The trapezoidal rule is much more accurate (for instance using $N = 8$ the error is 0.2% rather than $13 - 16\%$) and we observe that the error is reduced by a factor of 4. Thus quadratic convergence is detected. Here the 'exact' order on these three mesh levels is $\alpha = 2.0022$.
- A further observation is that the forward Euler scheme is unstable for $N = 16$ and $a = -10$ and has a zig-zag curve, whereas the other two schemes follow the exact solution and the decreasing exp-function. But for sufficiently small step sizes, the forward Euler scheme is also stable which we know from our A-stability calculations. These step sizes can be explicitely determined for this ODE model problem and shown below.

Discussion of the results for test 1 $a = 0.25$

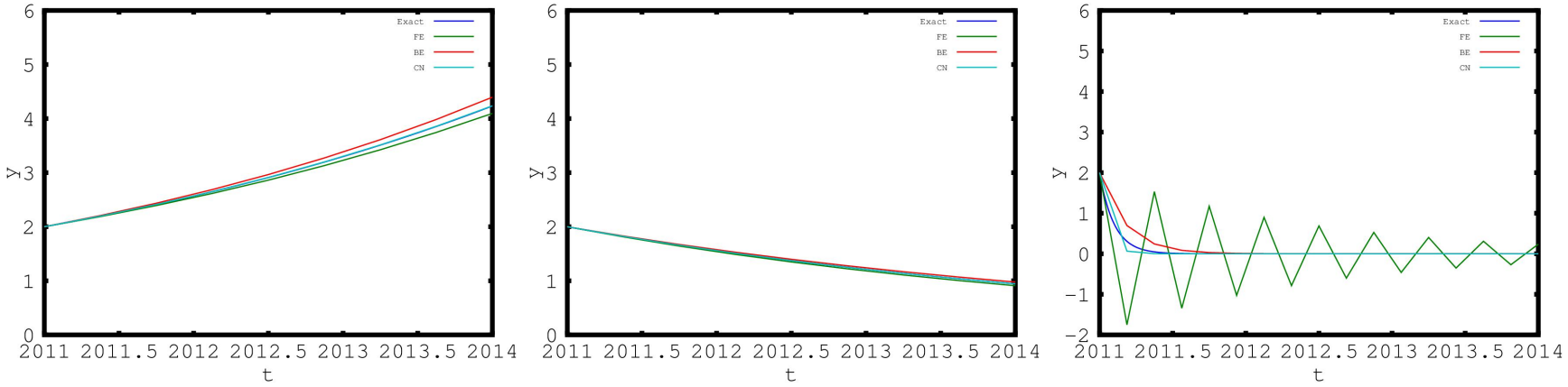


Figure: On the left, the solution to test 1 is shown. In the middle, test 2 is plotted. On the right, the solution of test 3 with $n = 16$ (number of intervals) is shown. Here, $n = 16$ corresponds to a step size $k = 0.18$ which is slightly below the critical step size for convergence. Thus we observe the instable behavior of the forward Euler method, but also see slow convergence towards the continuous solution.

Discussion of the results for test 3 $a = -10$

The convergence interval for the forward Euler scheme reads:

$$|1 + z| \leq 1 \quad \Rightarrow \quad |1 + ak| \leq 1$$

In test 3, we are given $a = -10$, yielding:

$$|1 + z| \leq 1 \quad \Rightarrow \quad |1 - 10k| \leq 1$$

- Thus, we need to choose a k that fulfills the previous relation. In this case, we easily calculate $k < 0.2$.
- This means that for all $k < 0.2$ we should have convergence of the forward Euler method and for $k \geq 0.2$ non-convergence (and in particular no stability!).

Discussion of the results for test 3 $a = -10$

We perform the following additional tests:

- Test 3a: $N = 10$, yielding $k = 0.3$;
- Test 3b: $N = 15$, yielding $k = 0.2$; exactly the boundary of the stability interval;
- Test 3c: $N = 16$, yielding $k = 0.1875$; from before;
- Test 3d: $N = 20$, yielding $k = 0.15$.

Discussion of the results for test 3 $a = -10$

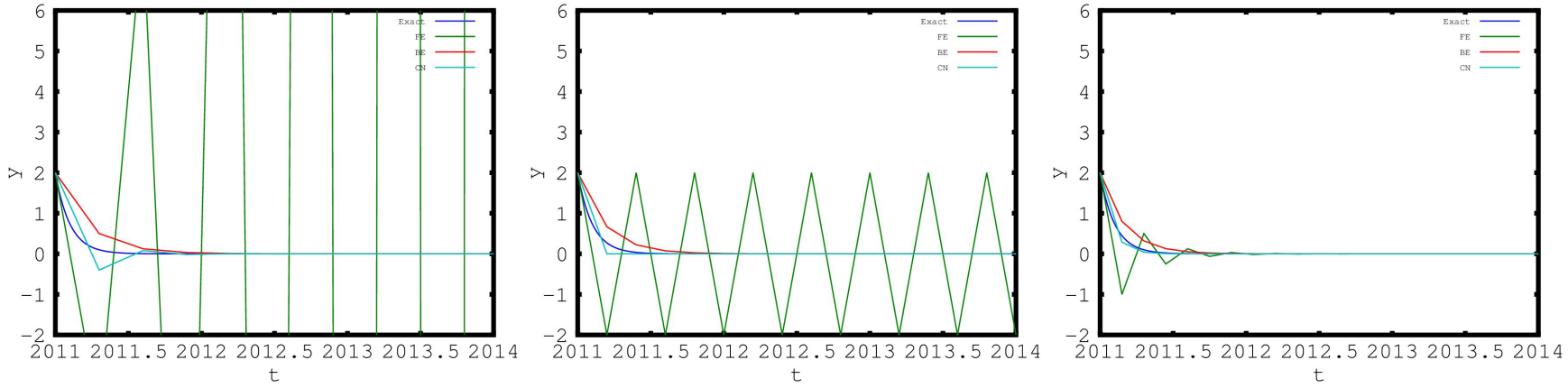


Figure: Tests 3a,3b,3d: Blow-up, constant zig-zag non-convergence, and convergence of the forward Euler method.

Summary of lecture 04

- Numerical analysis of some numerical schemes:
  - forward Euler (first order, explicit, only conditionally stable with time step size, critical for stiff problems),
  - backward Euler (first order, implicit, A-stable),
  - trapezoidal rule (second order, implicit, A-stable)
- Numerical tests demonstrating the theoretical results:
  - Nonstable schemes exhibit oscillations or blow-up of the solution
  - A higher convergence order (trapezoidal rule) has a higher accuracy and converges faster

Exercise 2 Overview

- Recapitulate when explicit Euler produces bounded approximations for the model problem $u' = \lambda u$ and confirm the results in HDNUM
- Learn how an ODE Solver can be implemented in an object-oriented way
- Investigate errors and convergence rates of various explicit and implicit schemes for a linear oscillator problem
- Explore the nonlinear Van der Pol oscillator using explicit and implicit methods.

Task 1

- Consider the linear, scalar model problem

$$u'(t) = \lambda u(t), \quad u(0) = 1, \quad \mathbb{R} \ni \lambda < 0$$

- Derive the explicit Euler scheme
- What is the condition on Δt such that the explicit Euler scheme produces bounded approximations for all $t > 0$
- Confirm your result with the implementation in file `eemodelproblem.cc` provided in the exercise yesterday

Task 2

- We will explain how ODE solvers are implemented in an object-oriented way in HDNUM
- Download the file `linearoscillator.cc` available on the cloud `https://cloud.ifam.uni-hannover.de/index.php/s/Cwe4ZqwLRMixS3J`. It solves the problem

$$u'(t) = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} u(t) \text{ in } (0, 20\pi], \qquad u(0) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

using the methods

#	Scheme	#	Scheme
0	Explicit Euler	4	Implicit Euler
1	Heun 2nd order	5	Implicit midpoint
2	Heun 3rd order	6	Alexander
3	Runge-Kutta 4th order	7	Crouzieux
		8	Gauß 6th order

- and provides errors $e(T)$ and convergence rates for all schemes
- What conclusions can you draw from the tables?

Task 3

- In this exercise we explore the nonlinear Van der Pol oscillator

$$u_0'(t) = -u_1(t) \qquad u_0(0) = 1$$
$$u_1'(t) = 1000 \cdot (u_0(t) - u_1^3(t)) \qquad u_1(0) = 2$$

  which is an example for a stiff ODE system
- Download an updated version of the file `linearoscillator.cc` from the cloud
- Compile and run the following four combinations of methods and timesteps:
  - `RKF45` method is an adaptive embedded Runge Kutta method of 5th order. Run it with tolerances $TOL_1 = 0.2$ and $TOL_2 = 0.001$ using an initial time step $\Delta t = 1/16$
  - The implicit Euler method. Run it with $\Delta t_1 = 1/16$ and $\Delta t_2 = 1/512$
- The output file contains in each line $t_i u_0(t_i) u_1(t_i) \Delta t_i$
- Compare the solutions, especially $u_1(t)$ as well as the time step sizes Δt_i for all four runs. What do you observe?