TECHNISCHE UNIVERSITÄT MÜNCHEN

Institut für Informatik

Informatik V
Chair for
Scientific Computing

Scientific Computing Lab

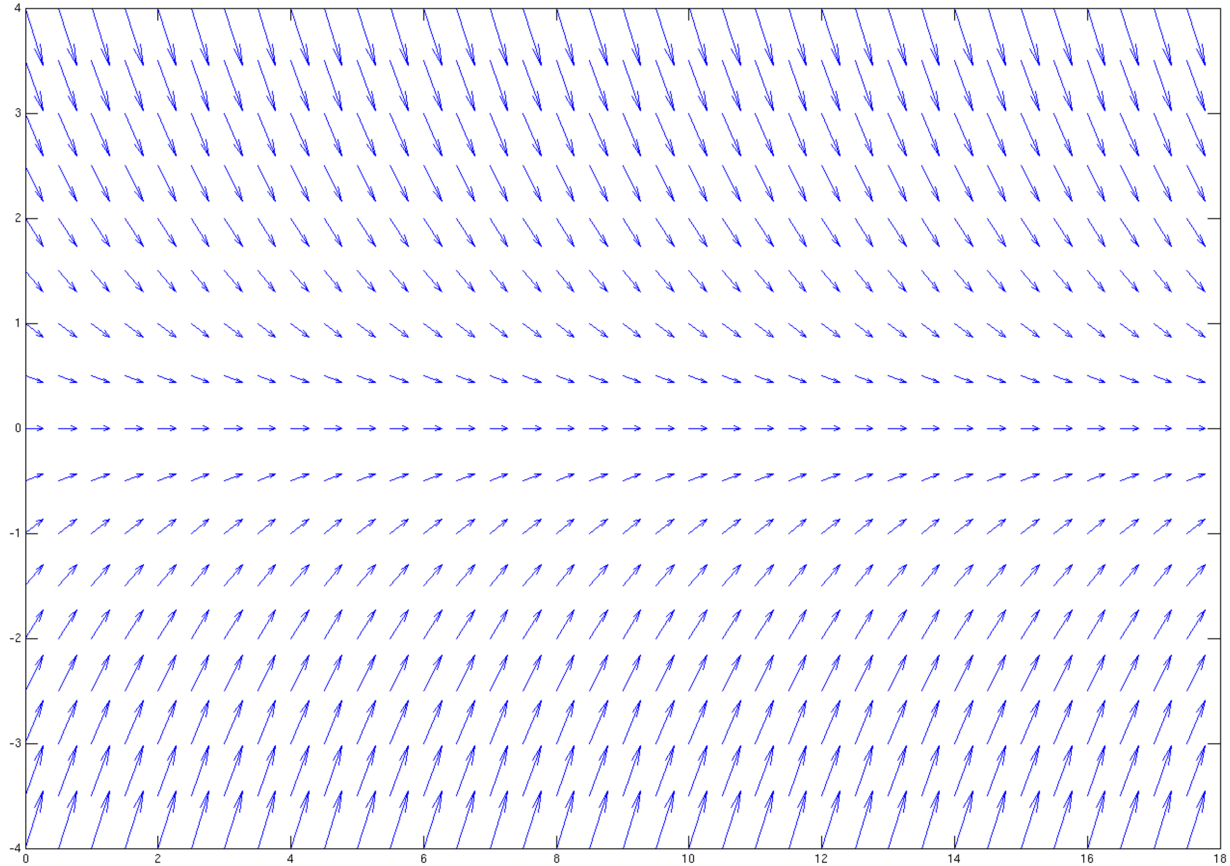
Ordinary Differential Equations

Implicit Discretization

Tobias Neckel

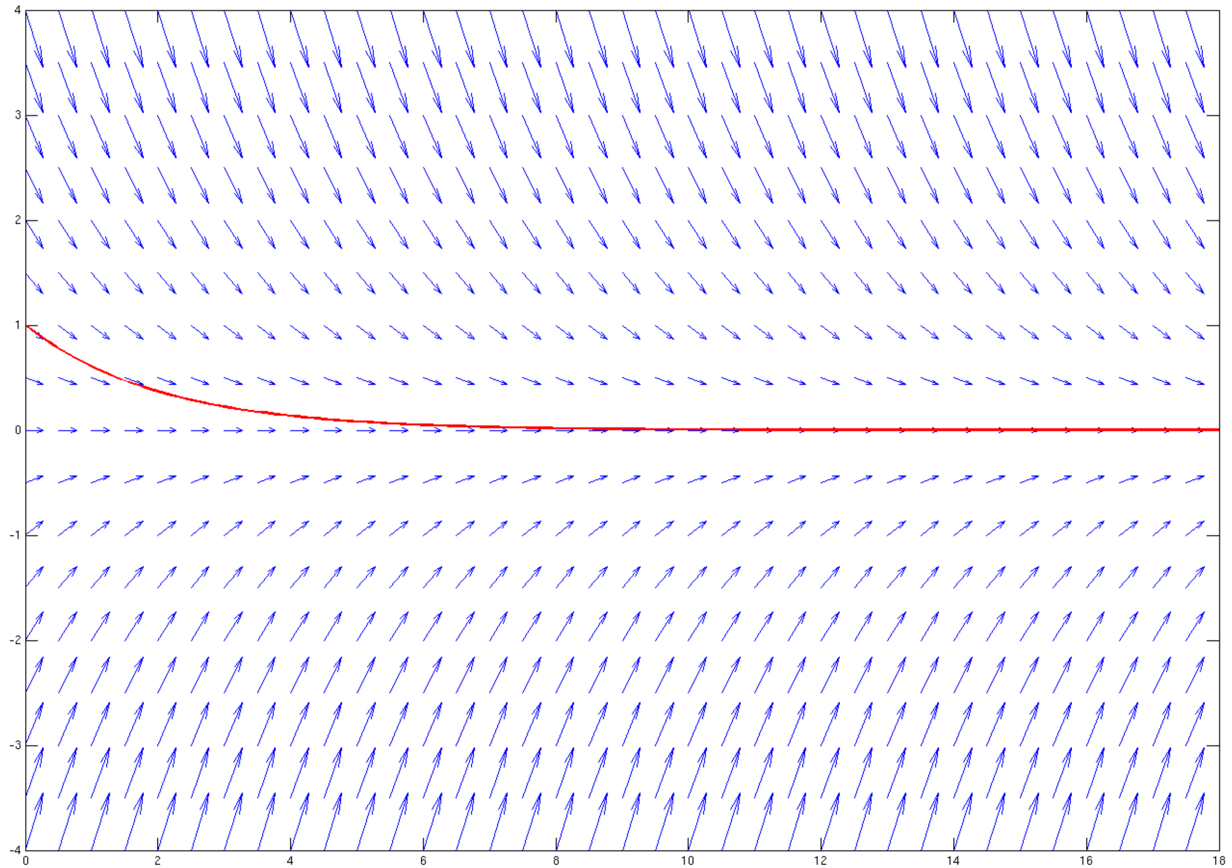
Instability

- Vector field of ODE $\dot{y}(t) = -k \cdot y(t)$



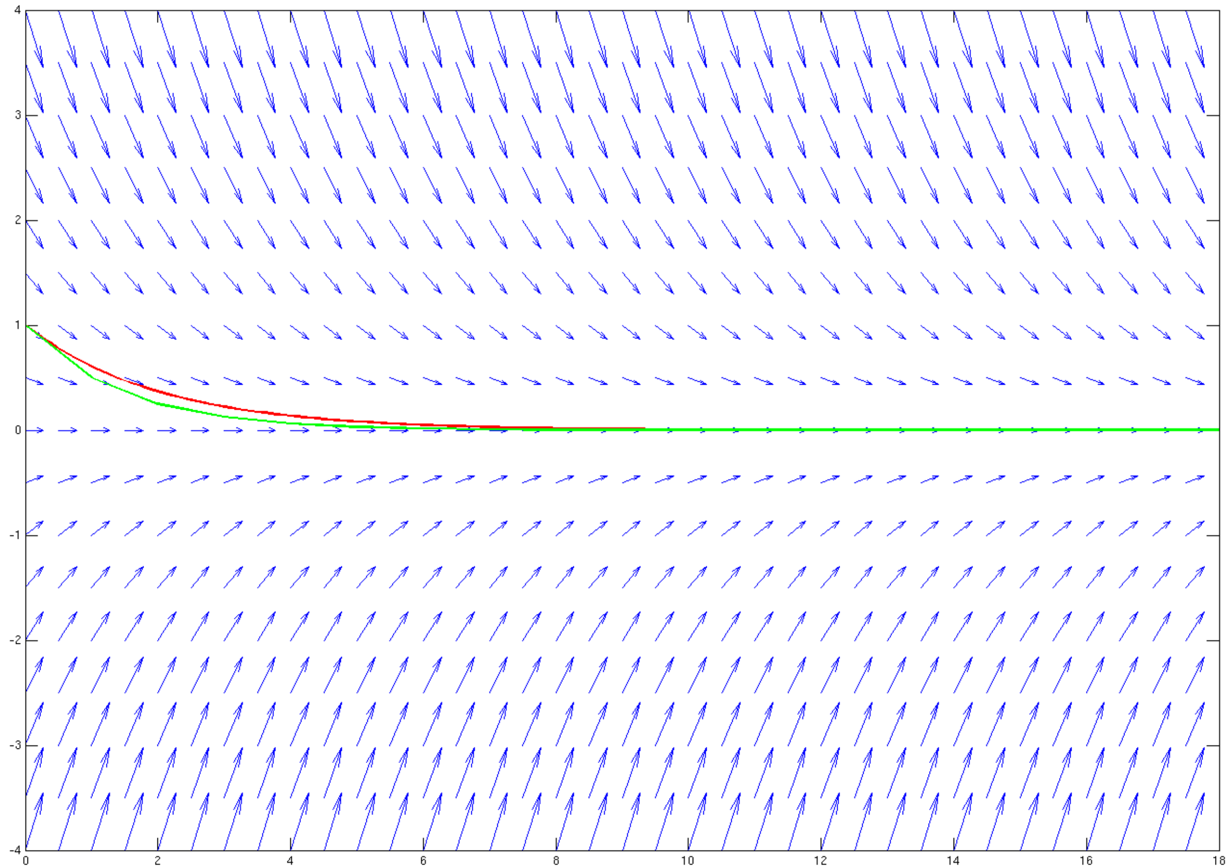
Instability

- Vector field of ODE $\dot{y}(t) = -k \cdot y(t)$



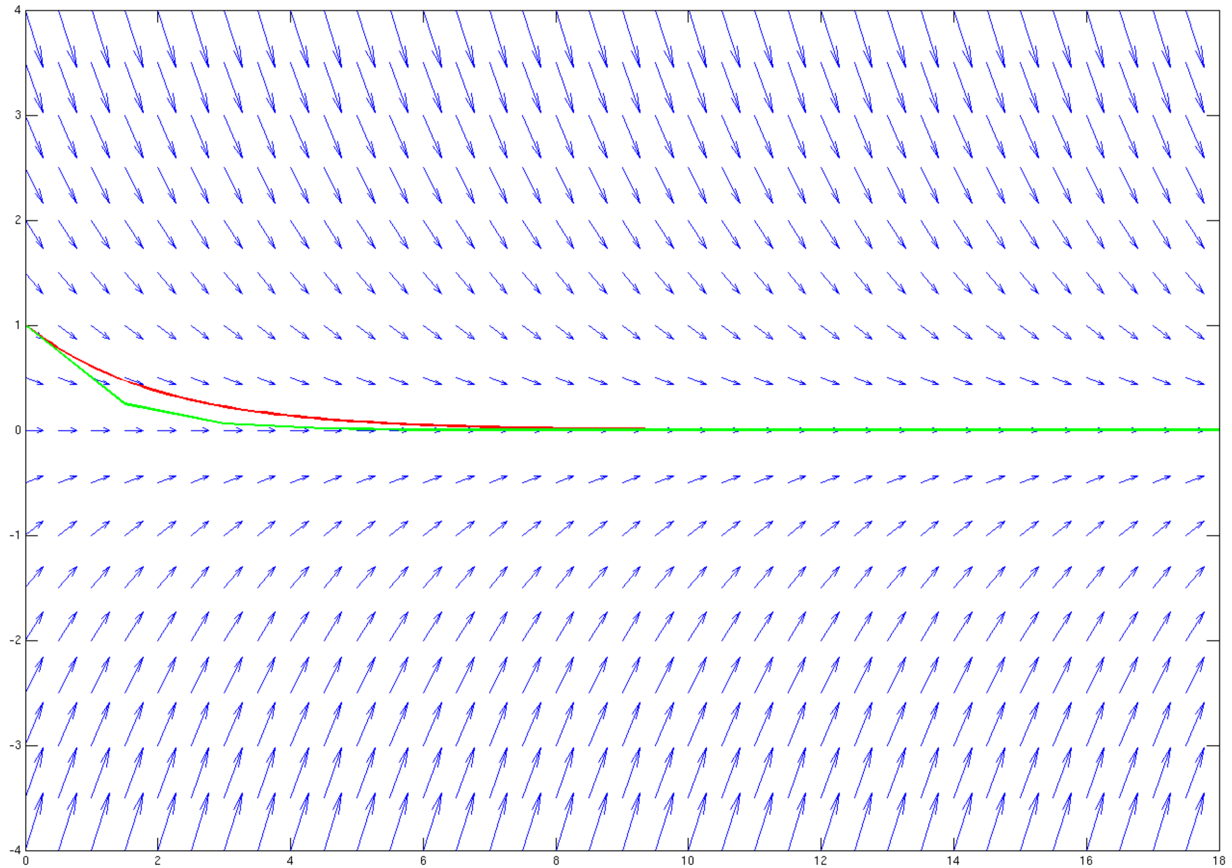
Instability

- Exlicit Euler, time step size: 1



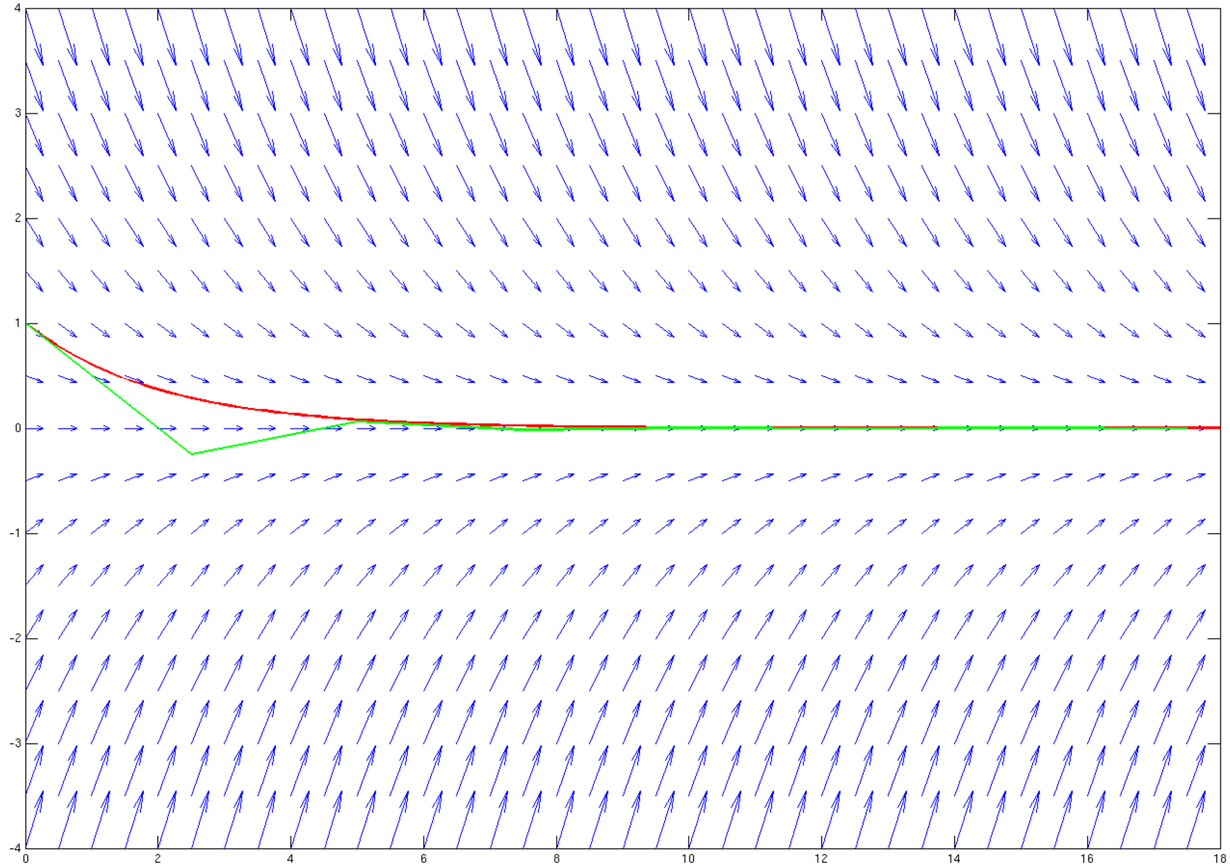
Instability

- Exlicit Euler, time step size: 1.5



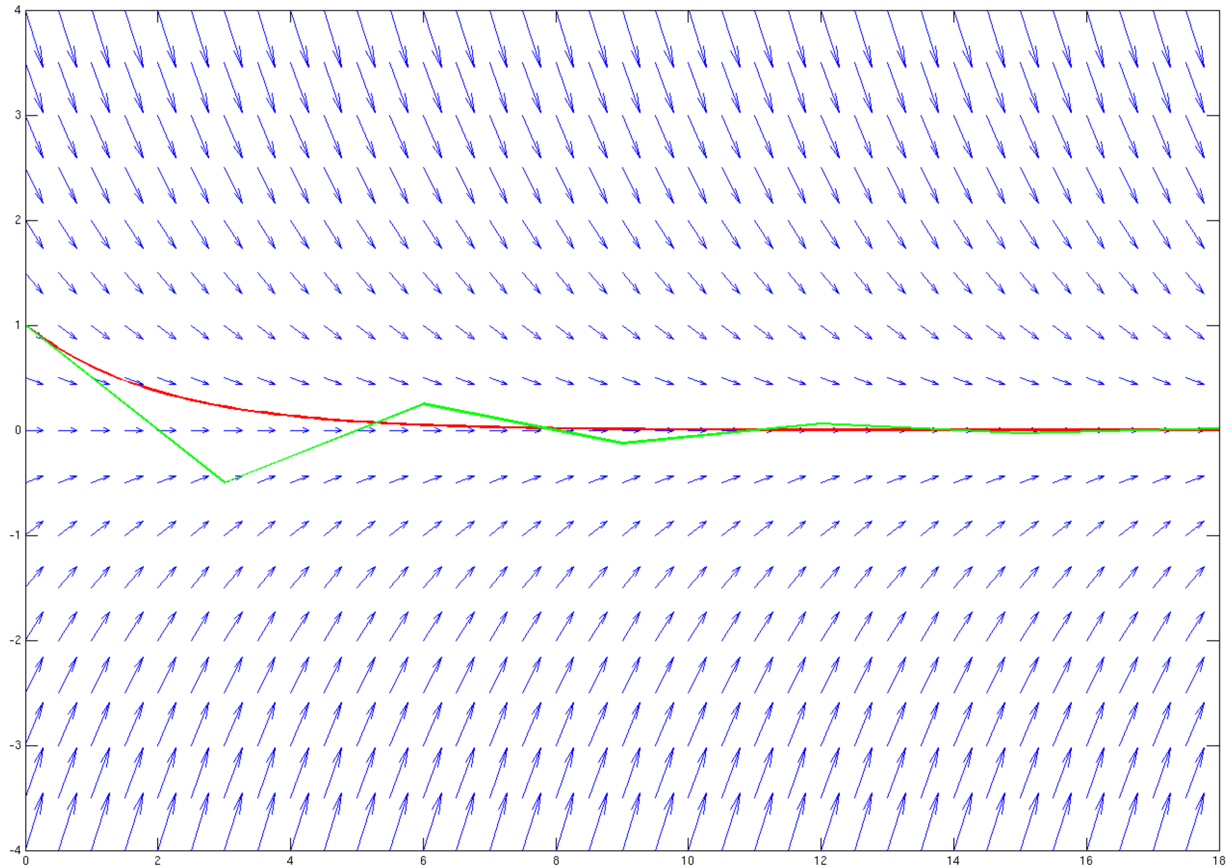
Instability

- Exlicit Euler, time step size: 2.5



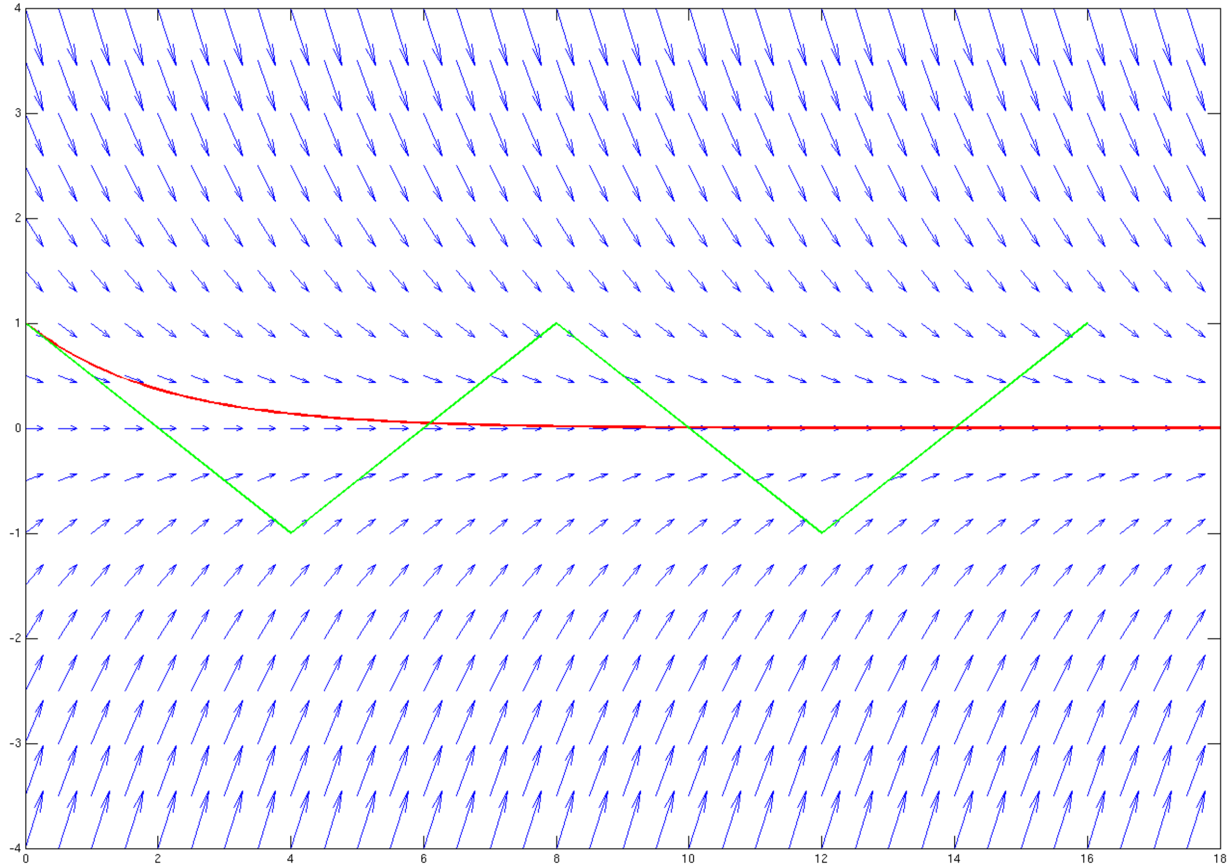
Instability

- Exlicit Euler, timestep-size: 3.0



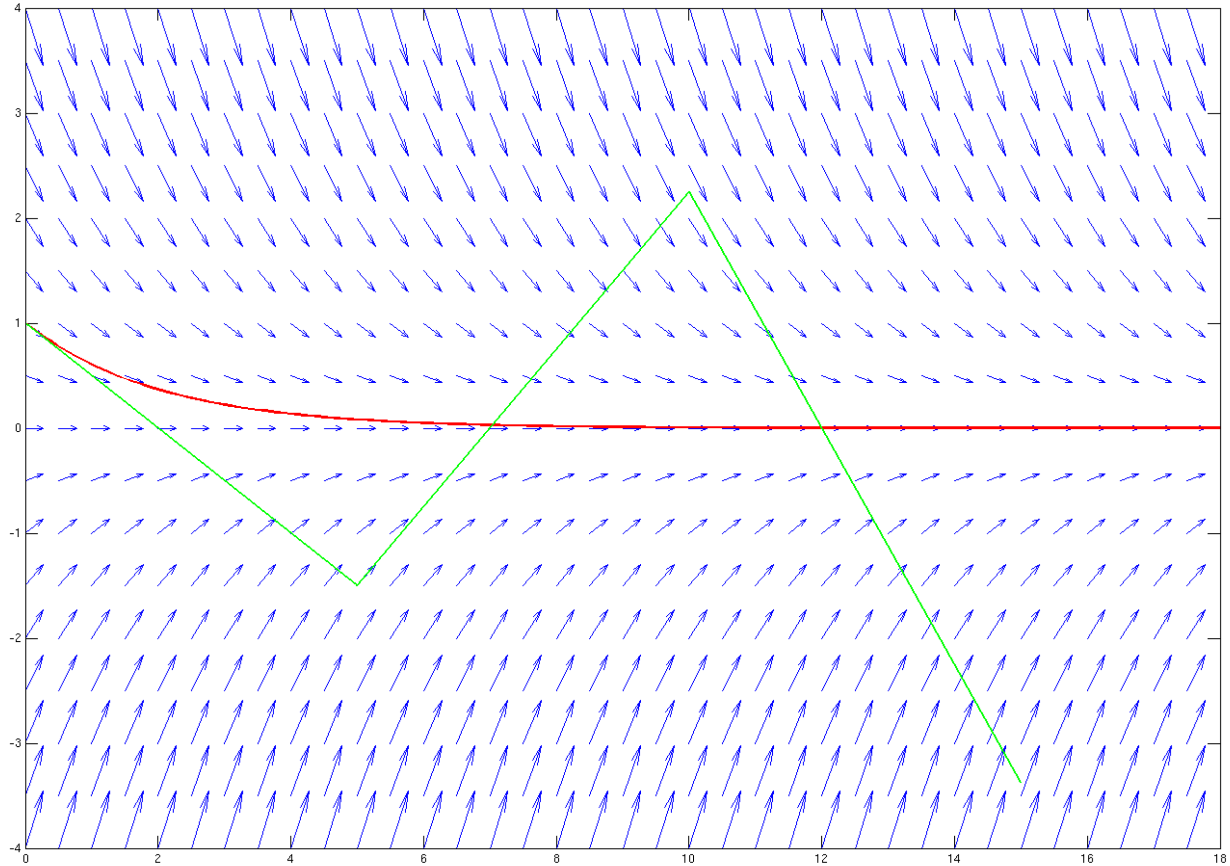
Instability

- Exlicit Euler, time step size: 4.0



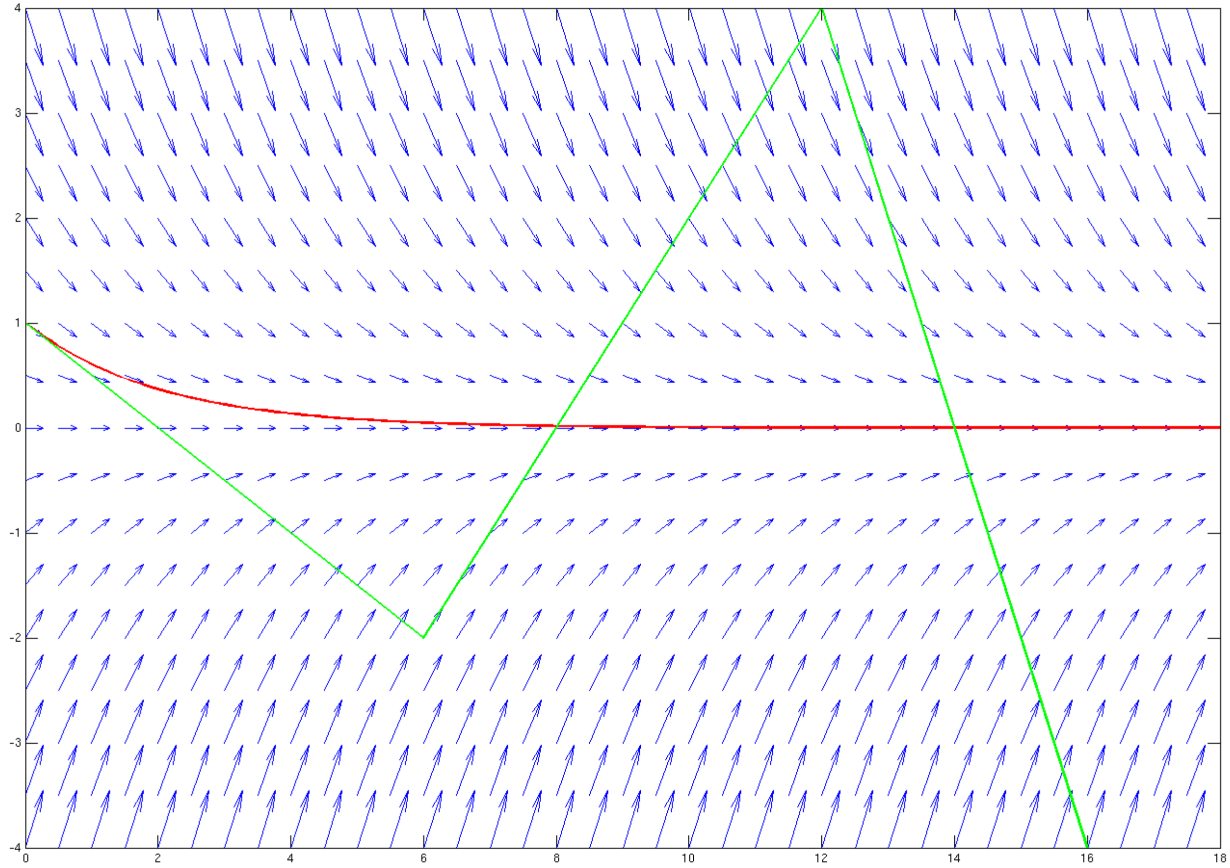
Instability

- Exlicit Euler, time step size: 5.0



Instability

- Exlicit Euler, time step size: 6.0



Instability

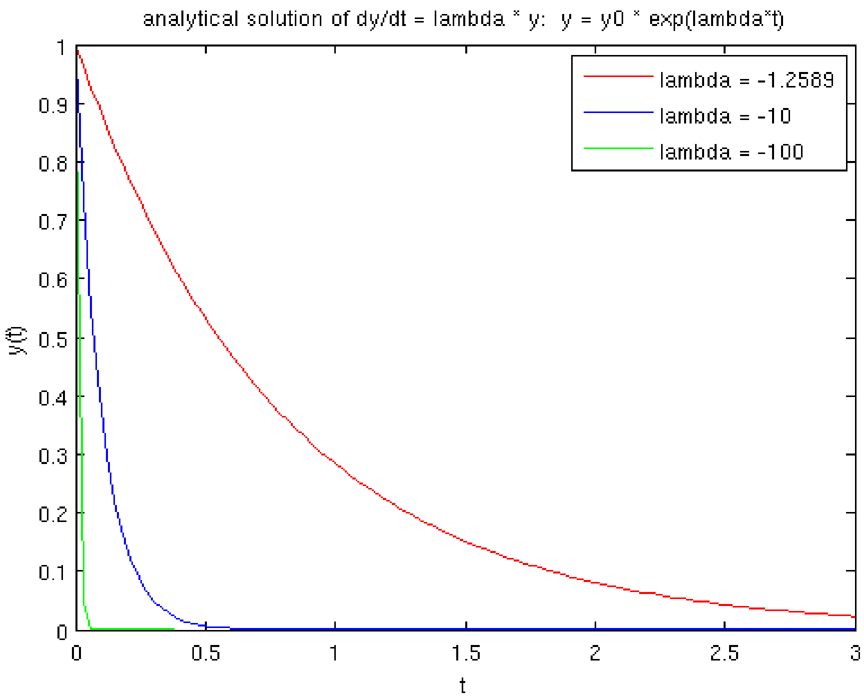
- stiff equations
  - instabilities if dt does not obey restrictions
  - „all explicit SSM with stable dt provide very small local errors”
  - examples: damped mech. systems, parabolic PDE, chem. reaction kinetics
- remedy: (special) implicit methods

Stiff Equations

- Example: Dahlquist's test equation: $\dot{y}(t) = \lambda \cdot y(t)$
  - stable analyt. solution
  - explicit methods:

    $\Delta t \leq |c/\lambda|$ also in steady phase

    (c depending on the method)



transient phase steady phase

Δt limited by stability & accuracy Δt limited by stability only!

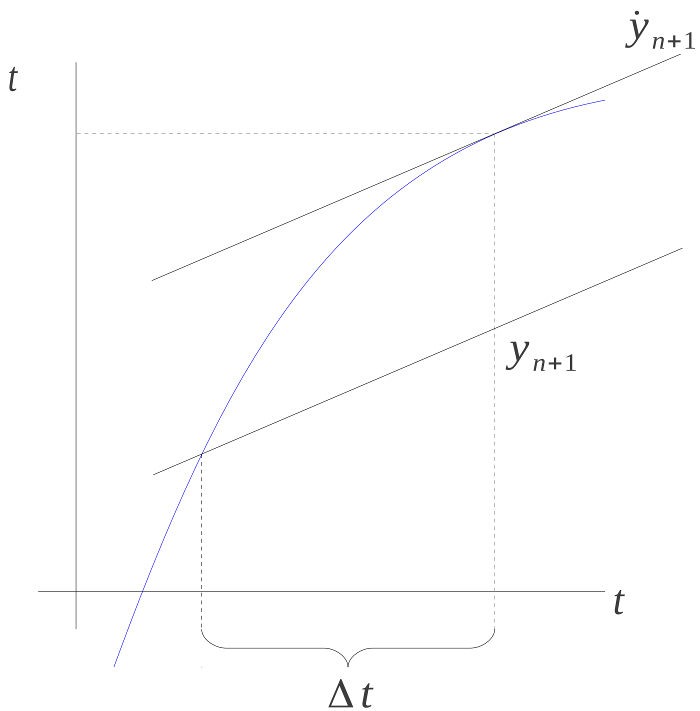
Implicit Methods

- implicit Euler
- 2nd-order Adams Moulton

➢ $$y_{n+1} = F(y_{n+1}, t_n, \Delta t)$$

Implicit Methods

- implicit Euler (1st-order method)

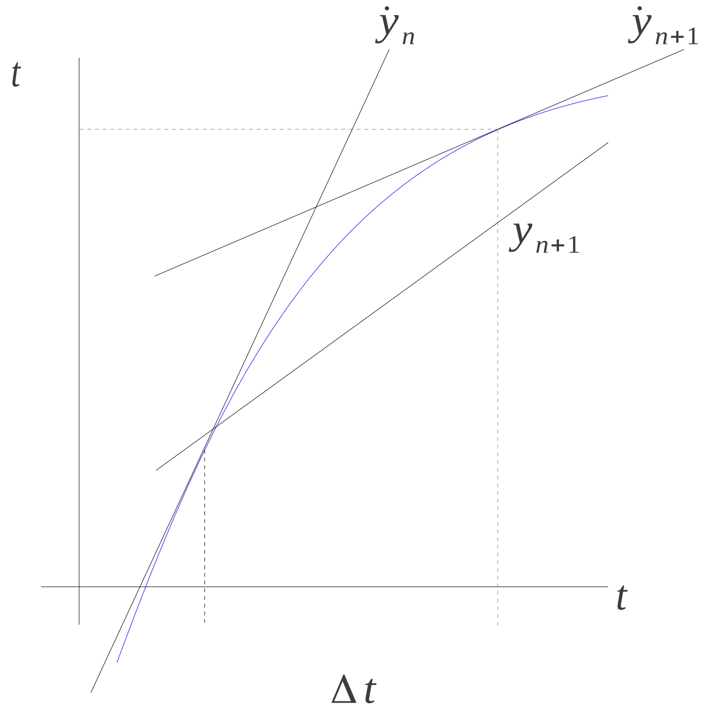


$$y_{n+1} = y_n + \Delta t \cdot \dot{y}_{n+1}$$

$$= y_n + \Delta t \cdot f\left(t_{n+1}, y_{n+1}\right)$$

Implicit Methods

- 2^{nd}-order Adams Moulton (Trapezoidal Rule)



$$y_{n+1} = y_n + \frac{\Delta t}{2} \cdot \left(\dot{y}_n + \dot{y}_{n+1} \right)$$

$$= y_n + \frac{\Delta t}{2} \cdot \left(f\left(t_n, y_n\right) + f\left(t_{n+1}, y_{n+1}\right) \right)$$

Newton's Method

- Implicit method may result in complex expressions:

  ODE

  $$f(t, y(t)) = -\log(y(t))$$

  implicit Euler:

  $$y_{n+1} = y_n + \Delta t \cdot \dot{y}_{n+1}$$

  $$= y_n + \Delta t \cdot f(t_{n+1}, y_{n+1})$$

  $$= y_n - \Delta t \cdot \log(y_{n+1})$$

Newton's Method

- Implicit method may result in complex expressions:

  implicit Euler:

  $$y_{n+1} = y_n - \Delta t \cdot \log(y_{n+1})$$

  needs to be solved:

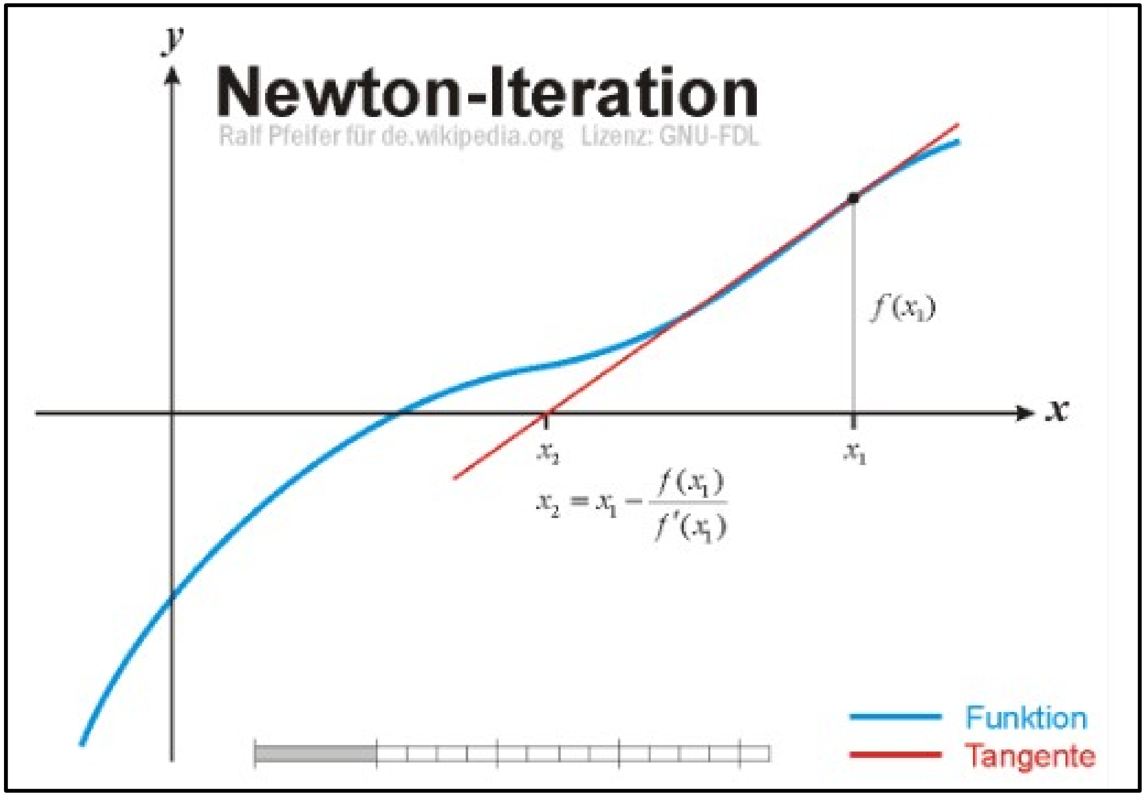
  $$y_{n+1} + \Delta t \cdot \log(y_{n+1}) - y_n = 0$$

  Can (only) be solved numerically!

Newton's Method

- Numerical approach to find roots of functions

$$G(x)=0$$

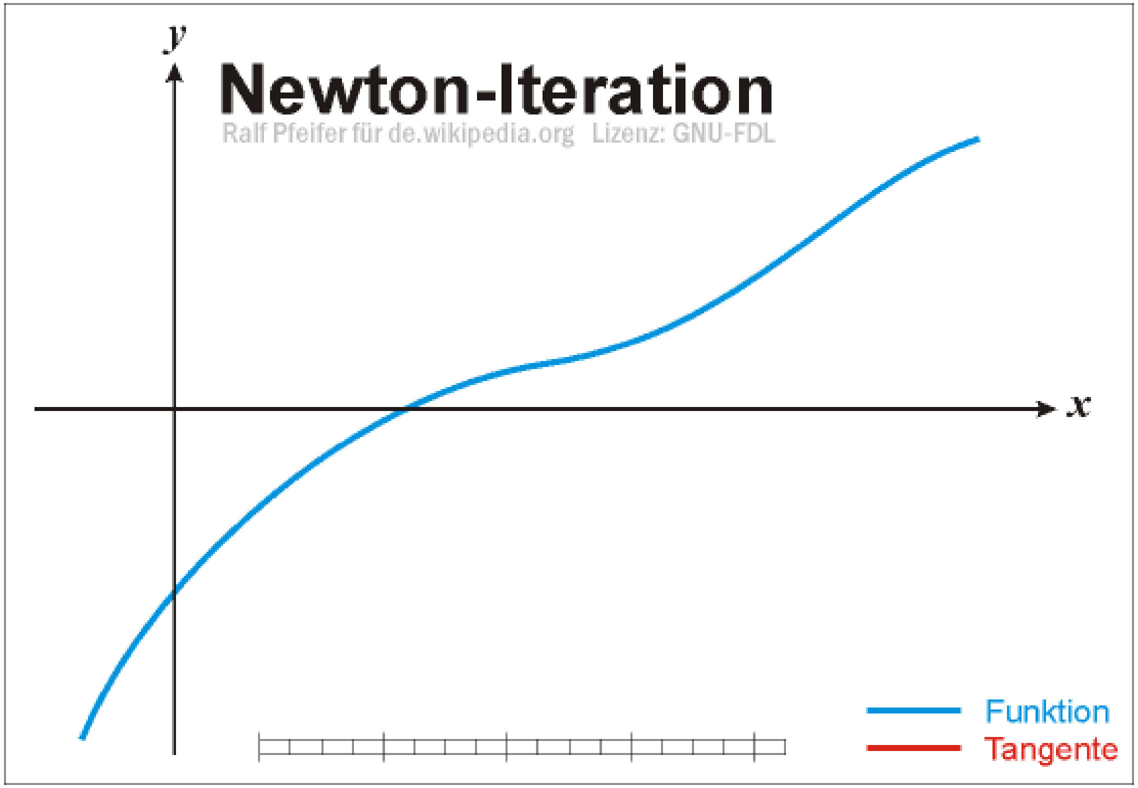


Ralf Pfeifer – http://upload.wikimedia.org/wikipedia/commons/e/e0/NewtonIteration_Ani.gif

Newton's Method

- Numerical approach to find roots of functions

$$G(x) = 0$$



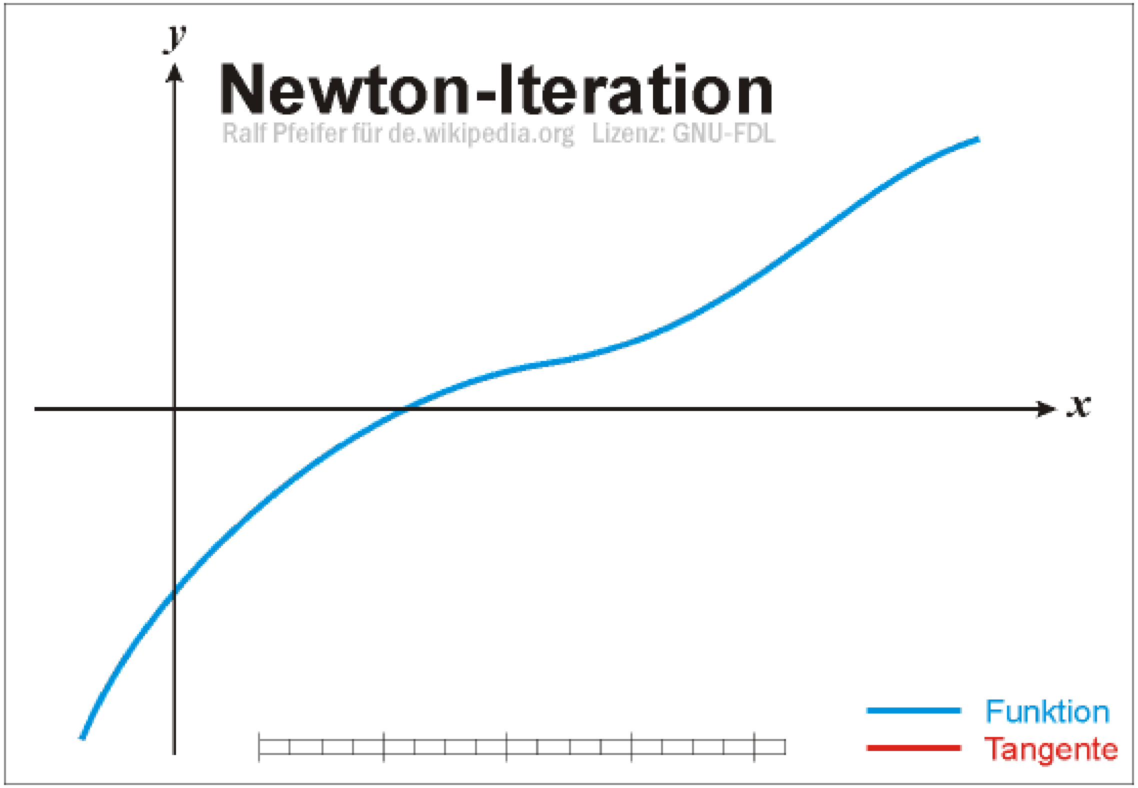
Ralf Pfeifer – http://upload.wikimedia.org/wikipedia/commons/e/e0/NewtonIteration_Ani.gif

Newton's Method

- Numerical approach to find roots of functions

$$\boxed{G(x)=0}$$

$$x_{i+1}=x_i-\frac{G(x_i)}{G'(x_i)}$$



Ralf Pfeifer – http://upload.wikimedia.org/wikipedia/commons/e/e0/NewtonIteration_Ani.gif

Newton's Method

- Numerical approach to find roots of functions
- In our case we get

$$G(y_{n+1})=0$$

- Thus, we need $G'(y_{n+1})$

Newton's Method

- Initial example: $y_{n+1} = y_n - \Delta t \cdot \log(y_{n+1})$

$$G(y_{n+1}) = y_{n+1} + \Delta t \cdot \log(y_{n+1}) - y_n$$

$$G'(y_{n+1}) = 1 + \frac{\Delta t}{y_{n+1}}$$

Newton's Method

- Convergence examples of the Newton iteration (cf. separate file)

Explicit versus Implicit

- explicit:
  - cheap time steps
  - many time steps
- implicit:
  - expensive/impossible time steps
  - less time steps