TECHNISCHE UNIVERSITÄT DRESDEN

Computergraphik und Visualisierung

CG3 Physically based Graphics

numeric solution of differential equations

Numeric Foundations
Differential Equations

- A differential equation describes a relation between a function in one or several variables and the [partial] derivatives of the function (often a system of equations)
- In case of a single variable, e.g. t one calls this an ordinary differential equation:

$$\{t, f(t), \dot{f}(t), \ddot{f}(t), \dddot{f}(t), \ldots\}$$

$$\ddot{x}(t) = -\omega^2 x(t)$$

- If derivatives of several variables arise one calls this partial differential equation or pde:

$$\{x, y, t, f(x,y,t), \partial_t f, \partial_x f, \partial_y f, \ldots\}$$

$$\frac{\partial^2 u(x,t)}{\partial t^2} = c^2 \frac{\partial^2 u(x,t)}{\partial x^2}$$

Numeric Foundations
Differential Equations

- We look for unknown functions[s] $f(t[,x,y,\ldots])$
- E.g.: $\ddot{f}(t) = -f(t) \quad \Rightarrow f(t) = A\cos t + B\sin t$

  $f(0)=0, \dot{f}(0)=1 \Rightarrow A=0, B=1$
- For a unique solution with concrete values for A and B additional so called boundary conditions are needed
- More examples:
  - ordinary: oscilator, particle-spring system
  - pde: Wave Eq., Maxwell Eg., Schrödinger Eq., Einsteinean Field Eq., Navier-Stokes-Eq.

Numeric Foundations
Ordinary Differential Equations

- Order n of differential equation corresponds to order of highest derivative
- implicit representation:

$\omega^2 x + \ddot{x} = 0$

$$DE\{t, f(t), \dot{f}(t), \ddot{f}(t), \ldots\} = 0$$

- explicit representation:

$\ddot{x} = -\omega^2 x$

$$\frac{\partial^n f}{\partial t^n} = DE\{t, f(t), \dot{f}(t), \ddot{f}(t), \ldots\}$$

- Order reduction: a DE of order n can be transformed into a system of DE of order 1.

$f_0 = x$

$f_1 = \dot{f}_0 = \dot{x} = v$

$\dot{f}_1 = \dot{v} = -\omega^2 x$

$$f_0(t) = f(t) \qquad f_2(t) = \dot{f}_1(t)$$

$$f_1(t) = \dot{f}_0(t) \qquad \vdots$$

$$\dot{f}_{n-1}(t) = \mathrm{DG}\{t, f_0(t), f_1(t), f_2(t), \ldots\}$$

Numeric Foundations Phase Space

$$\vec{y}(t) = \begin{pmatrix} x(t) \\ v(t) \end{pmatrix}$$

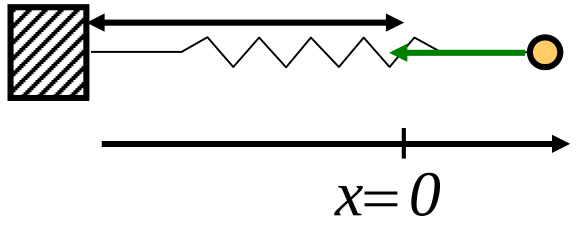
- In the reduction to a system of 1st order DEs the function value are combined with derivatives up to $(n-1)$ to a n-d phase space:

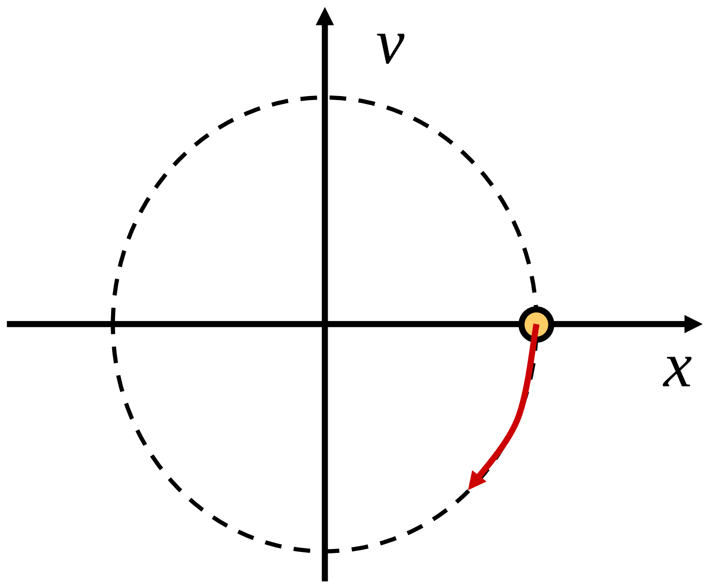
$$\vec{y}(t) = \begin{pmatrix} \underline{\boldsymbol{x}}(t) & \underline{\dot{\boldsymbol{x}}}(t) & \underline{\ddot{\boldsymbol{x}}}(t) & \cdots & \frac{\partial^{n-1}}{\partial t^{n-1}} \underline{\boldsymbol{x}}(t) \end{pmatrix}$$

- A location $\vec{\boldsymbol{y}}_0$ in phase space uniquely defines time evolution
- In physical system of 2nd order, the state is uniquely defined through position $\underline{\boldsymbol{x}}_0$ and velocity $\vec{\boldsymbol{v}}_0$ or momentum $\vec{\boldsymbol{p}}_0 = m\vec{\boldsymbol{v}}_0$
- Careful: instead of $\vec{\boldsymbol{y}}$ often symbol $\boldsymbol{x}$ is used for a phase space location in literature

Numeric Foundations
Phase Space

Harmonic Oscillator





$$\ddot{x}(t) = -\omega^2 x(t),\ \omega = \sqrt{\frac{k}{m}}$$

$$x(t) = A\sin\omega t + B\cos\omega t$$

$$v(t) = \omega\left(A\cos\omega t - B\sin\omega t\right)$$

$$\vec{y}(t) = \begin{pmatrix} x(t) \\ v(t) \end{pmatrix}$$

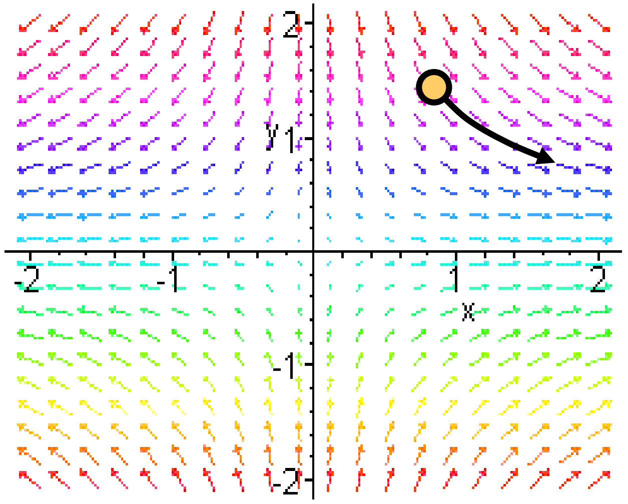
time evolution function

$$\dot{\vec{y}}(t) = \begin{pmatrix} v(t) \\ -\omega^2 x(t) \end{pmatrix} = \vec{f}\left(t, \vec{y}(t)\right)$$

equations of motion

Numeric Foundations
Ordinary Differential Equations

- The time evolution $\vec{f}$ is vector field on the simulation domain $\Omega \subseteq \boldsymbol{R}^n$
- It defines in combination with phase space location $\vec{y}_0 = \vec{y}(t=0)$ an initial value problem
- If $\vec{f}$ does not depend explicitly on t, the DE is called autonomous, otherwise non-autonomous
- In case of autonomous DE, the time evolution correspond to stream-lines of the vector field



Numeric Foundations
Ordinary Differential Equations

TECHNISCHE UNIVERSITÄT DRESDEN

Computergraphik und Visualisierung

- Theorem of Picard / Lindelöf:
  If for given $\vec{\boldsymbol{f}}: [0, a] \times \Omega \to \boldsymbol{R}^n$ the Lipschitz-condition: $\exists L: \forall t \in [0, a] \wedge \forall \vec{\boldsymbol{y}}_1, \vec{\boldsymbol{y}}_2 \in \Omega$:
  $$\left\| \vec{\boldsymbol{f}}(t, \vec{\boldsymbol{y}}_1) - \vec{\boldsymbol{f}}(t, \vec{\boldsymbol{y}}_2) \right\| \leq L \cdot \| \vec{\boldsymbol{y}}_1 - \vec{\boldsymbol{y}}_2 \|$$
  holds, then for each initial value $\vec{\boldsymbol{y}}_0 \in \Omega$ exists a differentiable function $\vec{\boldsymbol{y}}(t)$ which solves the intial value problem $\{\vec{\boldsymbol{y}}_0, \vec{\boldsymbol{f}}\}$.
- There is no similar result an the existence and uniqueness of a solution for pdes.
- In case of collisions the Lipschitz-condition cannot be fulfilled due to instantaneous changes in the velocity.

Numeric Foundations
Analytic Solutions

- Look up solution formulary
- Use algebra program like Maple
- Use research related software like Crack

Analytic Solution often not needed as

- There exist efficient numeric solvers with control of error
- Collisions and other effects make system of DEs too complicated for analytic solutions

Numeric Integration

- Let us consider the initial value problem

$$\vec{y}(0) = \vec{y}_0$$
$$\dot{\vec{y}}(t) = \vec{f}\big(t, \vec{y}(t)\big)$$

- Integration of both sides from 0 to t yields

$$\vec{y}(t) = \vec{y}_0 + \int_0^t \vec{f}\big(\tilde{t}, \vec{y}(\tilde{t})\big) d\tilde{t}$$

- Integral on right side is determined numerically

Numeric Integration

- This is not a typical integration as unknown function $\vec{y}(t)$ also appears on right side:

$$\vec{y}(t) = \int_0^t \vec{f}\left(\tau, \vec{y}(\tau)\right) d\tau + \vec{y}_0$$

- Approximate $\vec{y}(t)$ with polygonal line at discrete times $t_i := i \cdot h$ at $\vec{y}_0, \vec{y}_1, \vec{y}_2, \vec{y}_3, \ldots$, where each phase location $\vec{y}_i$ approximates $\vec{y}(t_i)$.

- discretize one step: $\vec{y}_1 = \vec{f}\left(0, \vec{y}(0)\right) \cdot h + \vec{y}_0$

yields explicit Euler:

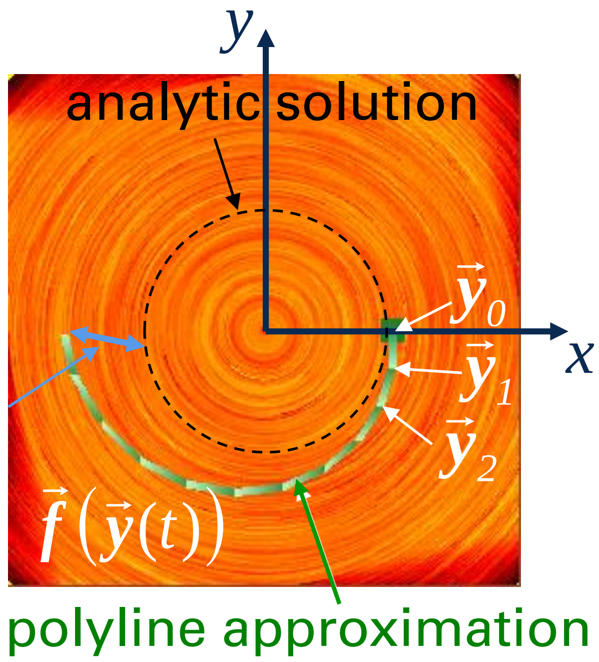
$$\vec{y}_{i+1} = h \cdot \vec{f}\left(t_i, \vec{y}_i\right) + \vec{y}_i$$

Numeric Integration

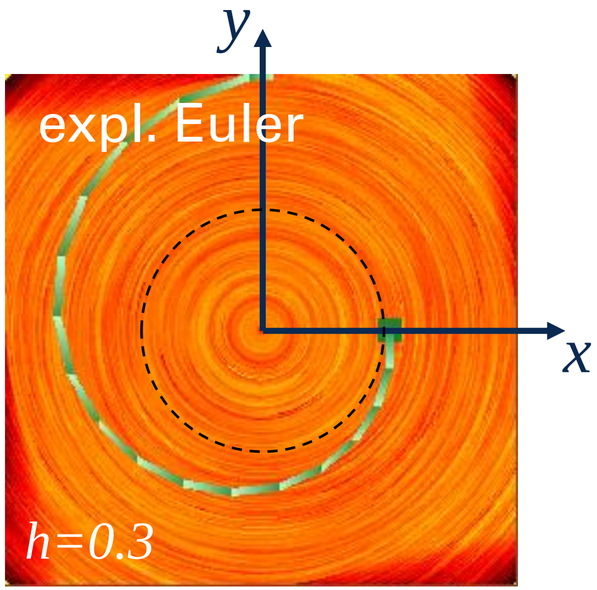
- $\vec{f}(t,\vec{y}(t))$ is a vector field
- In case of an autonomous DE it is static and the numerically integrate time evolutions approximate stream lines of the field
- Each integration step introduces an approximation error with respect to analytic solution
- Decreasing the size of the time step decreases the approximation error

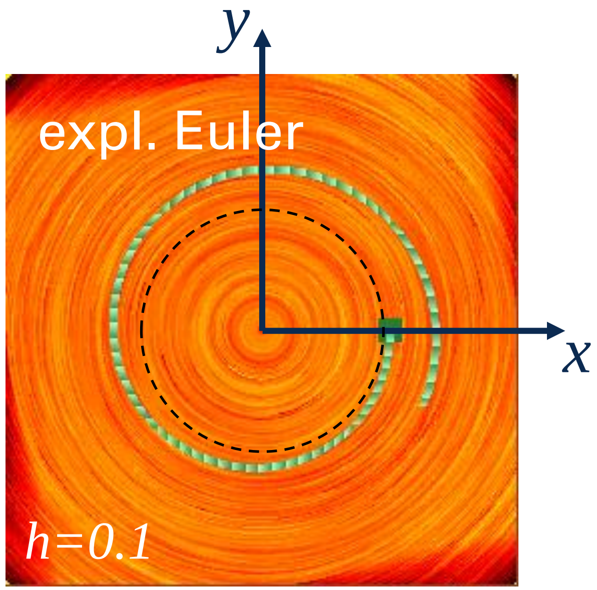
Ex.: harmon. Oscillator

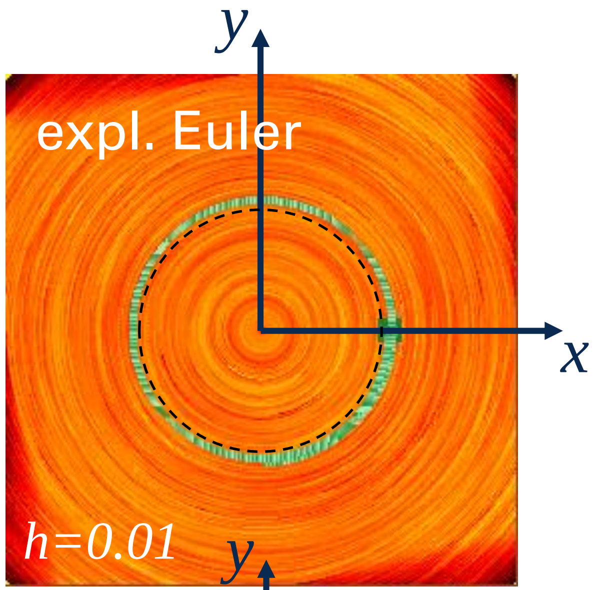
$$\vec{f}\left(\vec{y}(t)\right) = \begin{pmatrix} v(t) \\ -\omega^2 x(t) \end{pmatrix}$$

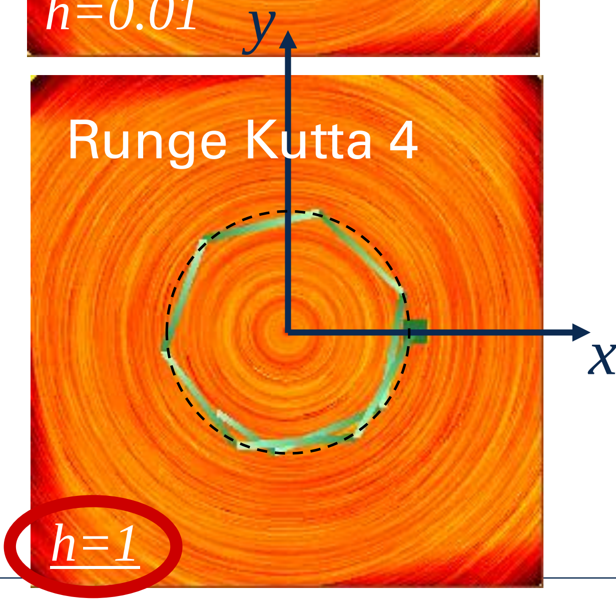


Numeric Integration









Careful: for large time steps as supported by integration methods of high approximation order polyline can deviate significantly from solution between integration points

Numeric Integration
Approximation Order

Approximation Error

- notation for analytic solution of integration:

$$\underline{\vec{y}}_i(s) = \vec{y}_i + \int_{t_i}^{s} \vec{f}\left(\tau, \vec{y}(\tau)\right) d\tau$$

- error per step h:

$$\varepsilon_i = \left\| \underbrace{\vec{y}_{i+1}}_{\text{numeric}} - \underbrace{\underline{\vec{y}}_i\left(t_{i+1}\right)}_{\text{analytic}} \right\|$$

- Ex.: explicit Euler

$$\vec{y}_{i+1} = \vec{y}_i + h \cdot \vec{f}\left(t_i, \vec{y}_i\right)$$

- Taylor series around t_i:

$$\underline{\vec{y}}_i(t_i + h) = \vec{y}_i + h \cdot \vec{f}\left(t_i, \vec{y}_i\right) + O(h^2)$$

- ➡ step error is proportional to h^2

Numeric Integration Approximation Order

- Accumulated error from integration over interval Δt:
  - Number of necessary steps: $n = \Delta t/h$
  - For a per step error proportional to h^2, an upper bound for the accumulated error is given by:

$$\varepsilon_{\Delta t} = \left\|\vec{y}_n - \vec{y}(\Delta t)\right\| \overset{!}{\leq} \frac{(1+Lh)^n - 1}{2L}\left\|\ddot{\vec{y}}\right\|_{[0,\Delta t]} \cdot h \leq \underbrace{\frac{e^{L\Delta t} - 1}{2L}\left\|\ddot{\vec{y}}\right\|_{[0,\Delta t]}}_{\text{constant}} \cdot h$$

Lipschitz constant of $\vec{f}$

- i.e. the accumulated error grows with $O(h)$
- For a step error of h^{k+1} the accumulated error grows with $O(h^k)$. One calls k the approximation order of the method (careful: don‘t mix up approximation order with the order of the DE!)
- Ex.: explicit Euler is method of approximation order 1

siehe M. Hanke-Bourgeois, Grundl. der Num. Mathe. & des Wissenschftl. Rechnens, Seiten 560-565

Numeric Integration
Verlet Method

- Method is specifically designed for physical systems of order 2 with position $\underline{\boldsymbol{x}}$ and velocity $\vec{\boldsymbol{v}}$.
- Derivation:
  - Exploit Taylor series adding $+h$ and $-h$:

$$\underline{\mathbf{x}}(t_i+h)=\underline{\mathbf{x}}(t_i)+h\cdot\underline{\dot{\mathbf{x}}}(t_i)+\tfrac{1}{2}h^2\cdot\underline{\ddot{\mathbf{x}}}(t_i)+\tfrac{1}{6}h^3\cdot\underline{\dddot{\mathbf{x}}}(t_i)+O(h^4)$$

$$\underline{\mathbf{x}}(t_i-h)=\underline{\mathbf{x}}(t_i)-h\cdot\underline{\dot{\mathbf{x}}}(t_i)+\tfrac{1}{2}h^2\cdot\underline{\ddot{\mathbf{x}}}(t_i)-\tfrac{1}{6}h^3\cdot\underline{\dddot{\mathbf{x}}}(t_i)+O(h^4)$$

$$\underline{\mathbf{x}}(t_i+h)+\underline{\mathbf{x}}(t_i-h)=2\underline{\mathbf{x}}(t_i)+h^2\cdot\underline{\ddot{\mathbf{x}}}(t_i)+O(h^4),\text{ mit }\underline{\ddot{\mathbf{x}}}(t_i)=\vec{\boldsymbol{a}}(t_i)$$

  - i.e. the 3rd order term vanished and $\boldsymbol{x}(t)$ is approximated with approximation order 3:

$$\boxed{\underline{\mathbf{x}}_{i+1}=2\underline{\mathbf{x}}_i-\underline{\mathbf{x}}_{i-1}+h^2\frac{\vec{\boldsymbol{F}}_i}{m}+O(h^4)}$$

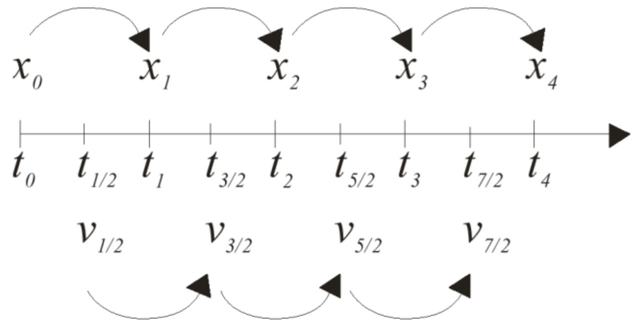
$\vec{\boldsymbol{v}}_{i+1}$ is not directly available

Numeric Integration
Verlet Method with Velocity

TECHNISCHE UNIVERSITÄT DRESDEN

Computergraphik und Visualisierung

- estimate velocity from $\vec{v} = \frac{x_{i+1} - x_i}{h}$
- Different interpretations are possible
  - forward difference … $\vec{v} = \vec{v}_i$
  - backward difference … $\vec{v} = \vec{v}_{i+1}$
  - central difference … $\vec{v} = \vec{v}_{i+\frac{1}{2}}$
- central difference can be interpreted on staggered temporal grid and have best approximation order:



Numeric Integration Verlet Method

Plugging $\vec{v}_{i-\frac{1}{2}} = \frac{\underline{x}_i - \underline{x}_{i-1}}{h}$ in yields:

$$\underline{x}_{i+1} = 2\underline{x}_i - \underline{x}_{i-1} + h^2 \frac{\vec{F}_i}{m}$$

$$\underline{x}_{i+1} = \underline{x}_i + h \frac{\underline{x}_i - \underline{x}_{i-1}}{h} + h^2 \frac{\vec{F}_i}{m}$$

$$\underline{x}_{i+1} = \underline{x}_i + h\vec{v}_{i-\frac{1}{2}} + h^2 \frac{\vec{F}_i}{m}$$

$$\underline{x}_{i+1} = \underline{x}_i + h\left(\vec{v}_{i-\frac{1}{2}} + h\frac{\vec{F}_i}{m}\right)$$

Verlet with velocity has app. ord. 2

$$\vec{v}_{i+\frac{1}{2}} = \vec{v}_{i-\frac{1}{2}} + h\frac{\vec{F}_i}{m}, \qquad \underline{x}_{i+1} = \underline{x}_i + h\vec{v}_{i+\frac{1}{2}}$$

Numeric Integration
Euler-Cromer Methode

also called semi-implicit

$$\boxed{\begin{aligned} \vec{\boldsymbol{v}}_{i+1} &= \vec{\boldsymbol{v}}_i + h\frac{\vec{\boldsymbol{F}}_i}{m} \\ \underline{\boldsymbol{x}}_{i+1} &= \underline{\boldsymbol{x}}_i + h\cdot\vec{\boldsymbol{v}}_{i+1} \end{aligned}}$$

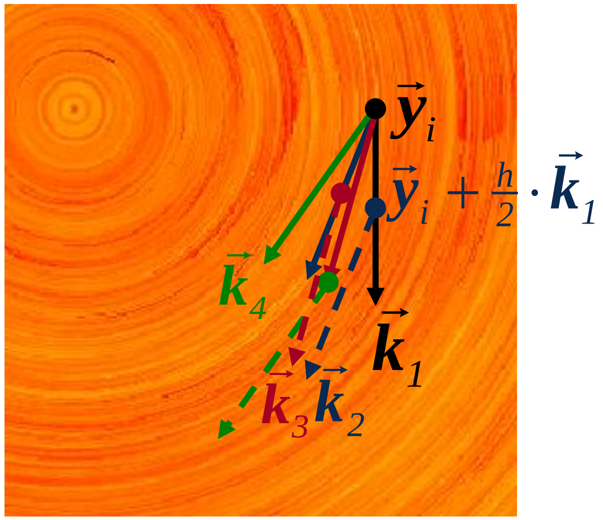
- Method is of approx. order 1 but is symplectic, what means that it is energy preserving and therefore more stable than other methods of approximation order 1.

Numeric Integration Runge-Kutta Methods

- Family of methods with methods for different approximation order
- App. ord. n builds on n function evaluations
- Very popular is RK4 (Runge Kutta 4. App.Ord.)

$$\vec{y}_{i+1} = \vec{y}_i + h \cdot \tfrac{1}{6}\left(\vec{k}_1 + 2\vec{k}_2 + 2\vec{k}_3 + \vec{k}_4\right) + O\left(h^5\right)$$

$$\begin{aligned}
\vec{k}_1 &= \vec{f}\left(t_i, \vec{y}_i\right)\\
\vec{k}_2 &= \vec{f}\left(t_i + \tfrac{h}{2}, \vec{y}_i + \tfrac{h}{2} \cdot \vec{k}_1\right)\\
\vec{k}_3 &= \vec{f}\left(t_i + \tfrac{h}{2}, \vec{y}_i + \tfrac{h}{2} \cdot \vec{k}_2\right)\\
\vec{k}_4 &= \vec{f}\left(t_i + h, \vec{y}_i + h \cdot \vec{k}_3\right)
\end{aligned}$$



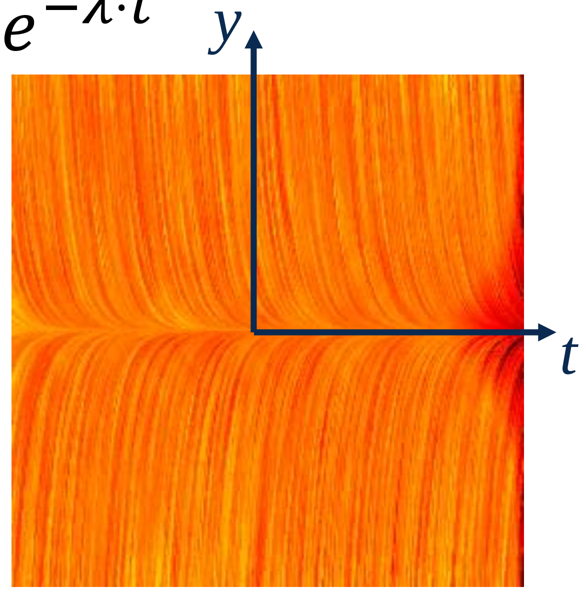
Numeric Integration A-Stability

Dahlquist

- A-Stability is analyzed on simple test case – the following 1st order DE: $\dot{y}(t) = -\lambda \cdot y(t)$
- Analytic solution: $y(t) = e^{-\lambda \cdot t}$
- Figure shows plot of

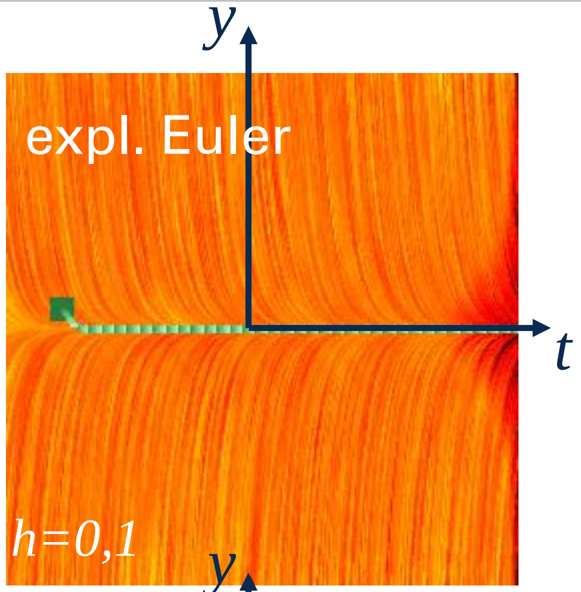
$$\frac{d}{dt}\begin{pmatrix} t \\ y(t) \end{pmatrix} = \begin{pmatrix} 1 \\ -\lambda \cdot y(t) \end{pmatrix}$$

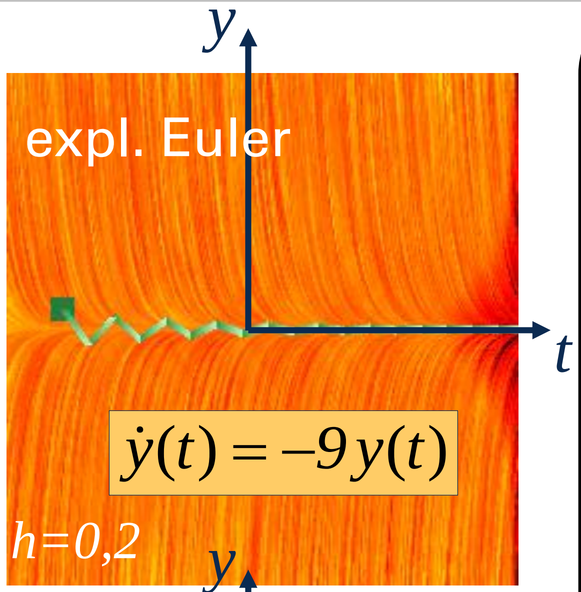
with λ=9



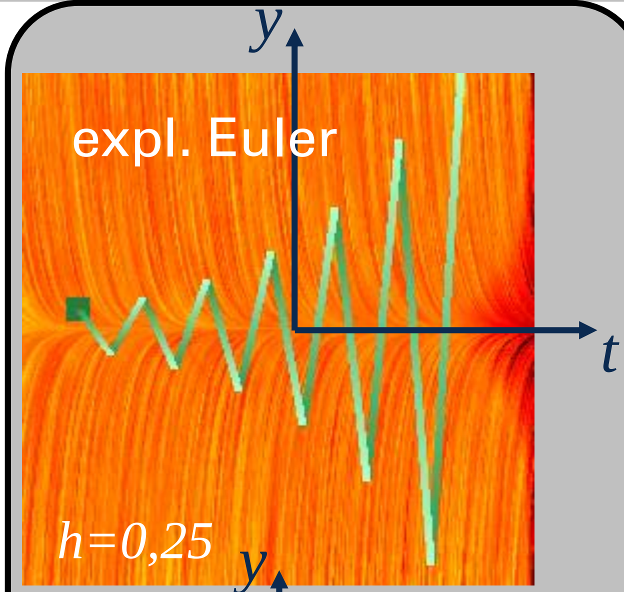
- Definition: a numeric integration method is called A-stable, iff generated approximations converge for any step width and any λ towards the t-axis.

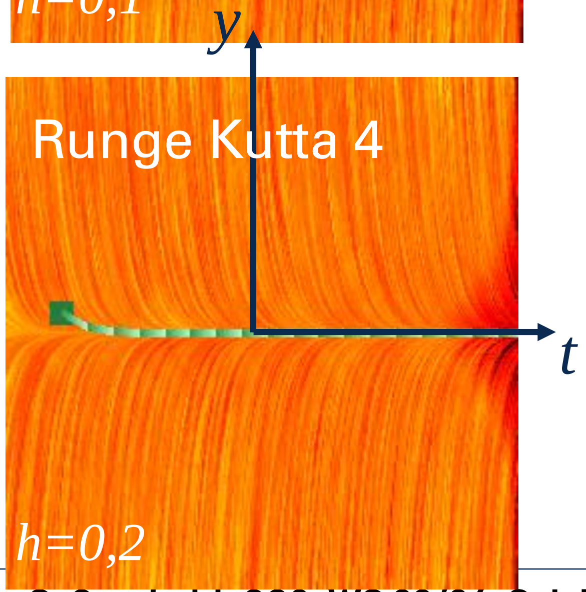
Numeric Integration
Stability

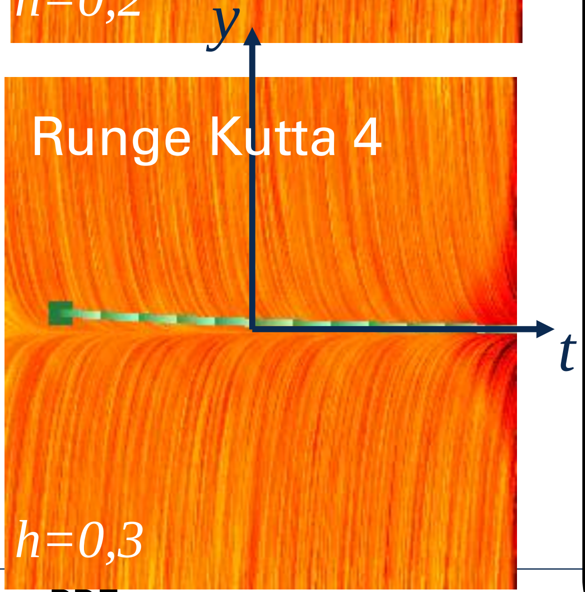


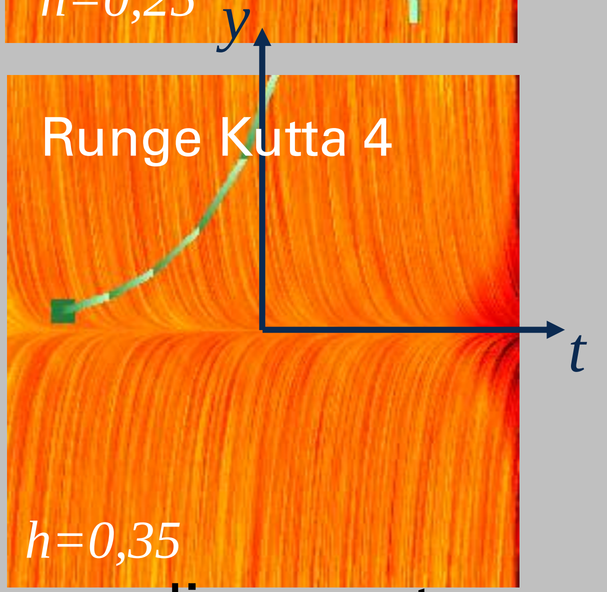


$$\dot{y}(t) = -9\,y(t)$$



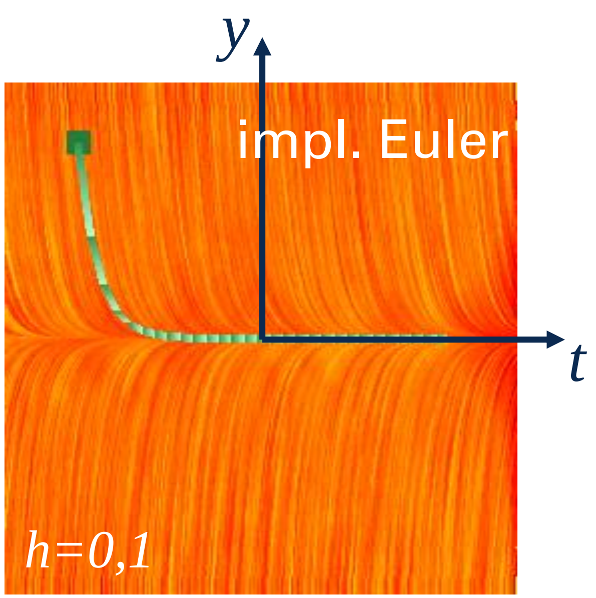


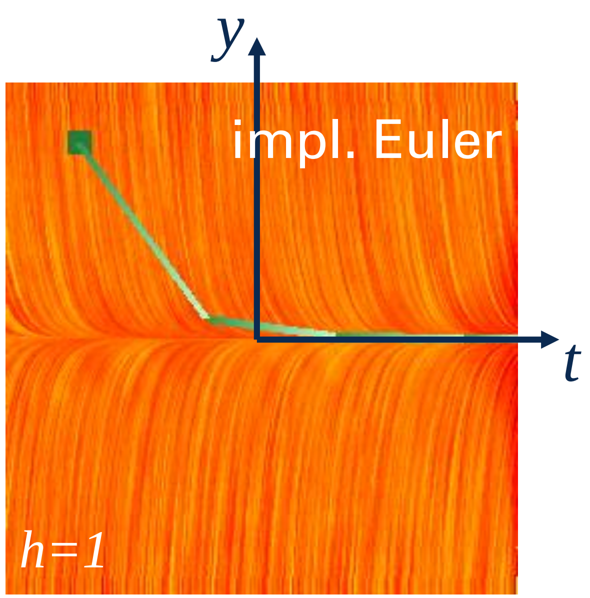


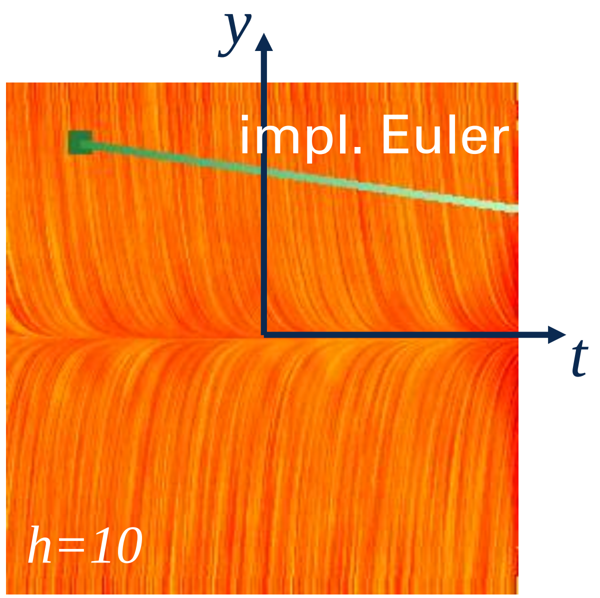


divergent

Numeric Integration
Stability – Implicit Euler







$$\dot{y}(t) = -\lambda \cdot y(t), \quad \lambda > 0$$

$$y_{i+1} = y_i + h \cdot \dot{y}(t_{i+1})$$

$$y_{i+1} = y_i - h \cdot \lambda \cdot y_{i+1}$$

$$(1 + h \cdot \lambda) \cdot y_{i+1} = y_i$$

$$\boxed{y_{i+1} = y_i / (1 + h \cdot \lambda)}$$

$$y_{i+1} < y_i \Leftrightarrow 1 + h \cdot \lambda > 1 \Leftrightarrow h \cdot \lambda > 0$$

Numeric Integration Stability – Implicit Euler

- The DE $\dot{y}(t) = -\lambda \cdot y(t)$ is called stiff as explicit methods do not always converge.
- The implicit Euler Method is A-stable
- The general form:

$$\vec{y}_{i+1} = \vec{y}_i + h \cdot \vec{f}\left(t_{i+1}, \vec{y}_{i+1}\right)$$

  yields a system of typically non-linear equations in the components of $\vec{y}_{i+1}$
- A Newton-Iteration can be used to solve the system of equations
- For stiff problems this additional work pays off as larger steps can be taken

Numeric Integration
Midpoint methods

- simplest explicit midpoint method as modified Euler:

$$\vec{\boldsymbol{y}}_{i+1} = \vec{\boldsymbol{y}}_i + h\vec{\boldsymbol{f}}\left(t_{i+\frac{1}{2}}, \vec{\boldsymbol{y}}_i + \frac{h}{2}\vec{\boldsymbol{f}}(t_i, \vec{\boldsymbol{y}}_i)\right)$$

- simplest implicit midpoint method:

$$\vec{\boldsymbol{y}}_{i+1} = \vec{\boldsymbol{y}}_i + h\vec{\boldsymbol{f}}\left(t_{i+\frac{1}{2}}, \frac{1}{2}(\vec{\boldsymbol{y}}_i + \vec{\boldsymbol{y}}_{i+1})\right)$$

- Both methods are of approximation order 2
- implicit midpoint is also symplectic

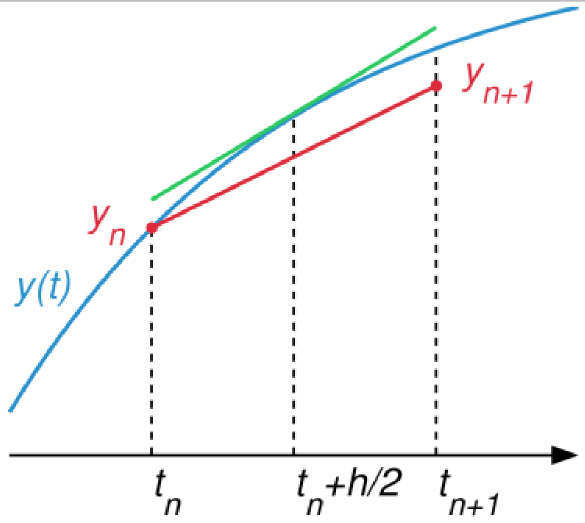


Illustration of the midpoint method assuming that $\vec{\boldsymbol{y}}_n$ equals the exact value $\vec{\boldsymbol{y}}(t_n)$. The midpoint method computes $\vec{\boldsymbol{y}}_{n+1}$ so that the red chord is approximately parallel to the tangent line at the midpoint (the green line).

Numeric Integration
Adaptation of Stepwidth

- let **`step`** be a k^{th} app.ord. method
- estimate step error ε by comparing to two steps with half stepwidth:

$$\vec{y}_{i+1} = \mathbf{step}\left(h, t_i, \vec{y}_i\right) = \underline{\vec{y}}_i(t_{i+1}) + ch^{k+1} + O(h^{k+2})$$

$$\vec{y}^{\,*}_{i+1} = \mathbf{step}\left(\tfrac{h}{2}, t_{i+\frac{1}{2}}, \mathbf{step}\left(\tfrac{h}{2}, t_i, \vec{y}_i\right)\right) \approx \underline{\vec{y}}_i(t_{i+1}) + 2c\left(\tfrac{h}{2}\right)^{k+1} + O(h^{k+2})$$

$$\Rightarrow \varepsilon = \left\|\vec{y}^{\,*}_{i+1} - \vec{y}_{i+1}\right\|_\infty = g(c)h^{k+1} + O\left(h^{k+2}\right)$$

- if ε is larger than toleranz τ discard step and re-estimate h, otherwise estimate h for next step:

$$\boxed{h_{\text{new}} = h \cdot \sqrt[k+1]{\rho \cdot \frac{\tau}{\epsilon}},}$$ with safety factor $\rho < 1$

Numeric Integration
Adaptation of Stepwidth

- Derivation of stepwidth update:

$$\varepsilon(h) \approx g(c)h^{k+1}$$

$$\Rightarrow g(c) \approx \frac{\varepsilon(h)}{h^{k+1}}$$

$$\varepsilon(h_{\text{new}}) \approx g(c)h_{\text{new}}^{k+1} < \tau$$

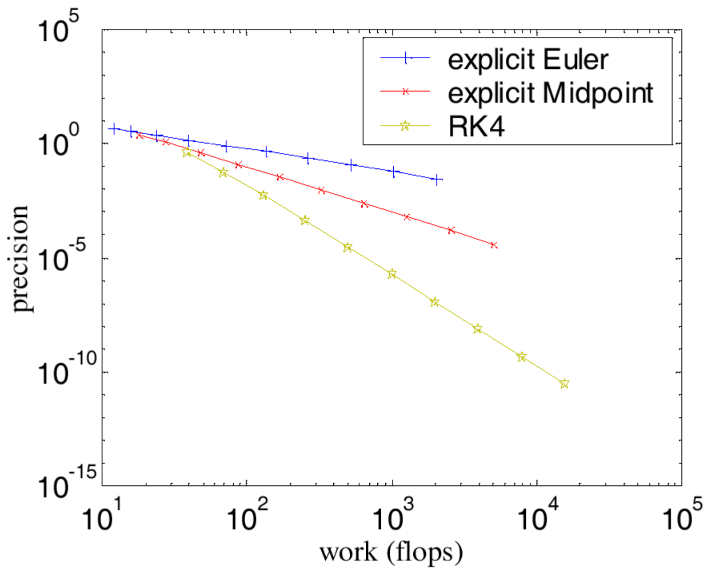
$$\varepsilon(h)\frac{h_{\text{new}}^{k+1}}{h^{k+1}} < \tau$$

$$h_{\text{new}}^{k+1} < h^{k+1}\frac{\tau}{\varepsilon(h)}$$

$$h_{\text{new}} < \sqrt[k+1]{h^{k+1}\frac{\tau}{\varepsilon(h)}} = h \cdot \sqrt[k+1]{\frac{\tau}{\varepsilon(h)}}$$

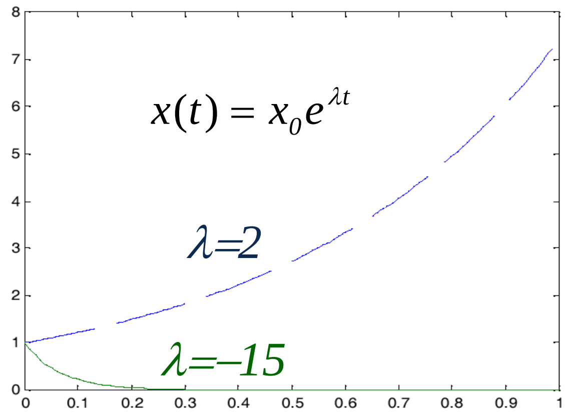
Numeric Integration Efficiency Analysis

- Hauth, M., Etzmuß, O., & Straßer, W. (2003). Analysis of numerical methods for the simulation of deformable models. *The Visual Computer*, *19*(7-8), 581-600.
- Compare methods through the number of floating point operations necessary to achieve a given precision (error bound)
- Fraction of precision over number of necessary operations (slope in diagrams) is measure of efficiency of a method

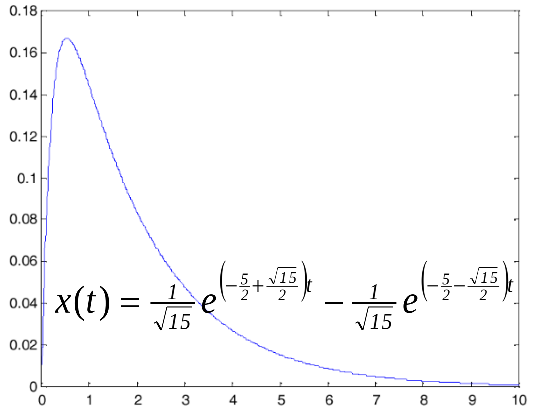


Numeric Integration Efficiency Analysis

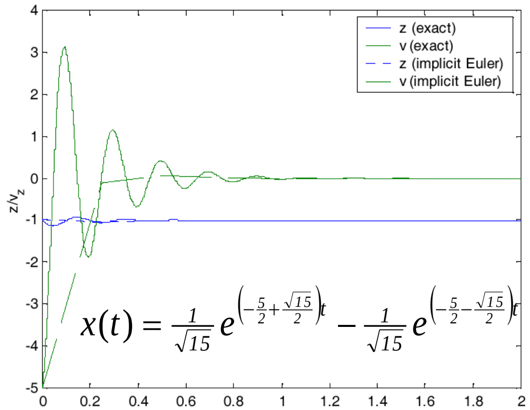
- Hauth et al. use the stiff DE of A-stability analysis with two parameter settings and the damped harmonic oscillator with and without gravity
- For all DEs the analytic solution is known and can be used to compute precision of methods exactly



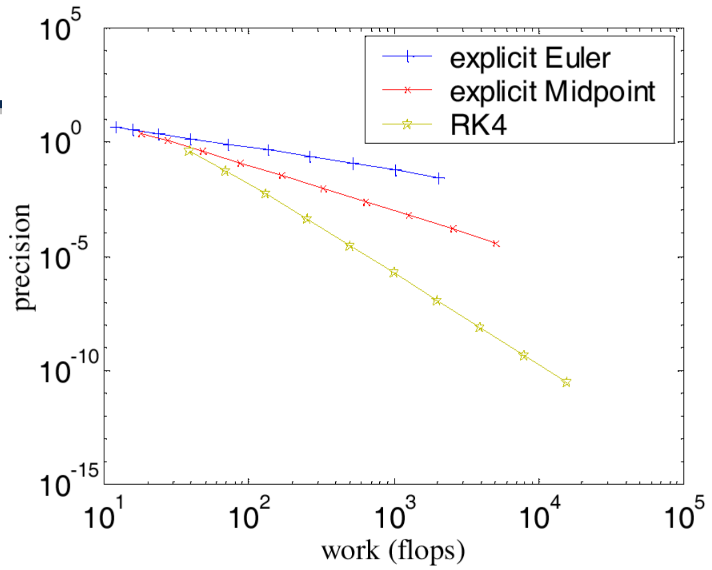
$$x(0)=1 \quad \dot{x}=\lambda x$$



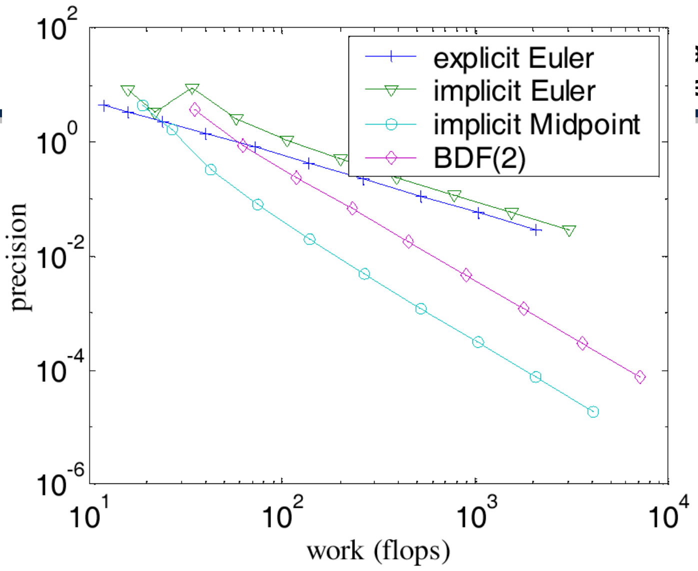
$$x(0)=0 \quad \ddot{x}=\tfrac{\lambda}{2}x+\lambda\dot{x}$$
$$\dot{x}(0)=1 \quad \lambda=-5$$



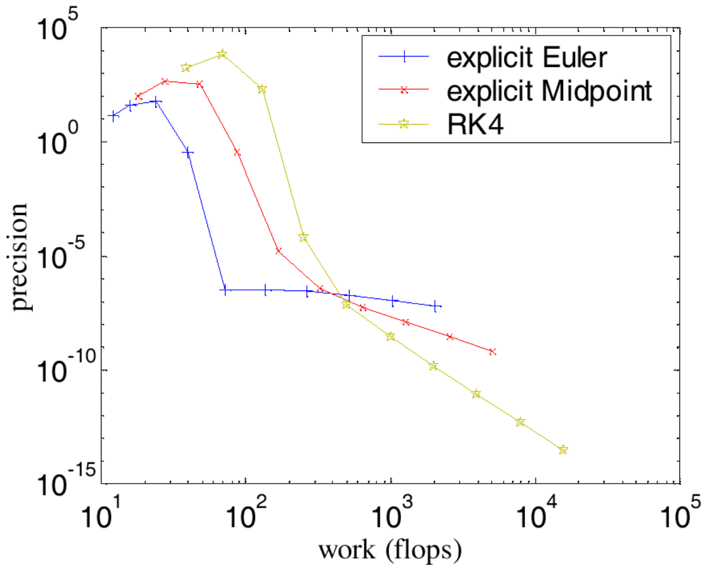
$$x(0)=0, \dot{x}(0)=-5 \quad \ddot{x}=\tfrac{k}{m}(l_0-x)+\tfrac{d}{m}\dot{x}+g$$
$$k=1000, l=-1, d=10, g=-10, m=1$$



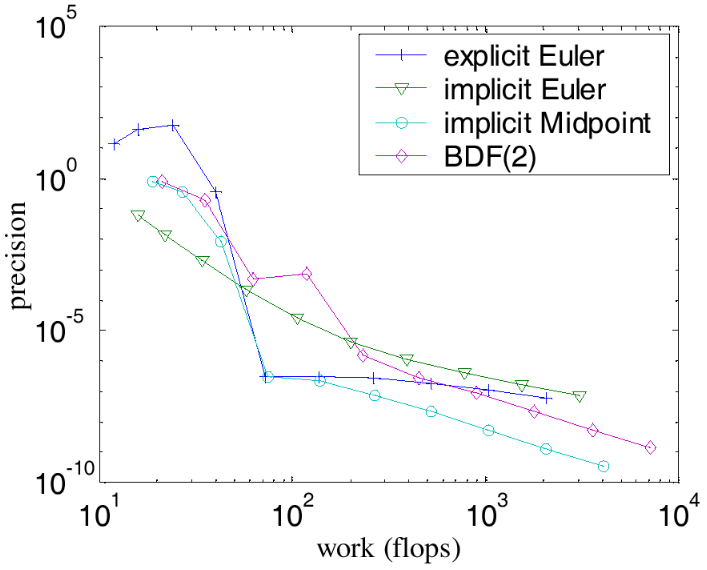
$$\lambda = 2$$

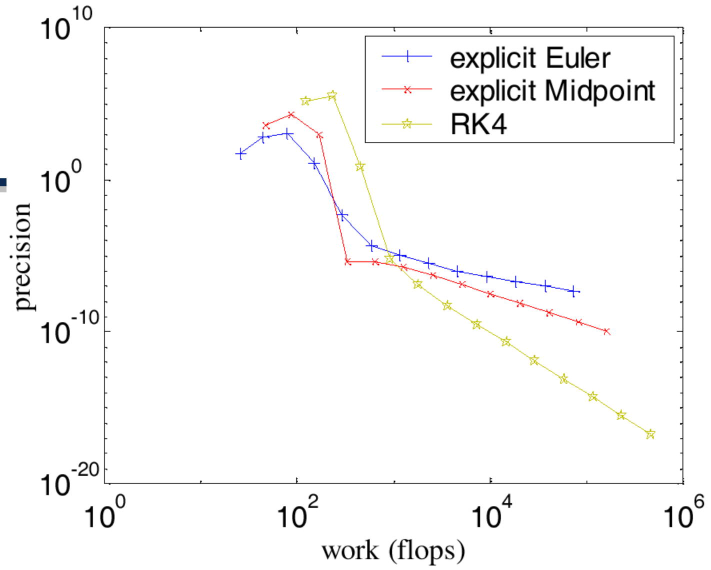


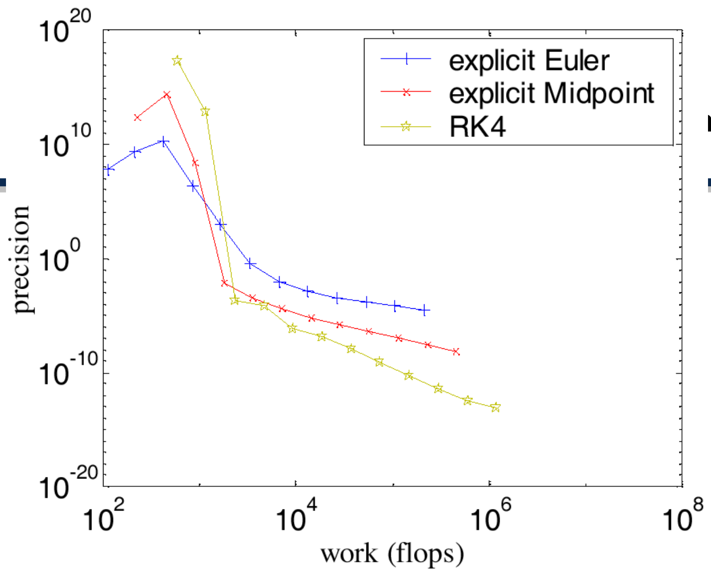
$$x(0) = 1 \quad \dot{x} = \lambda x \quad \lambda = 2$$



$$\lambda = -15$$

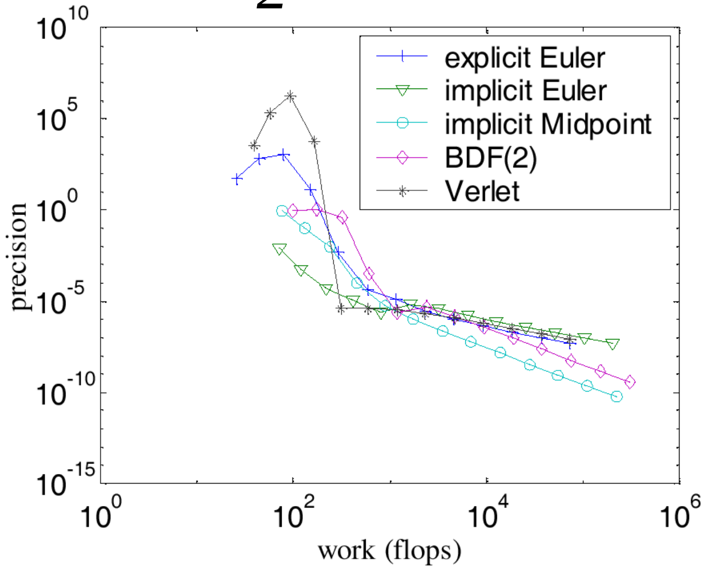


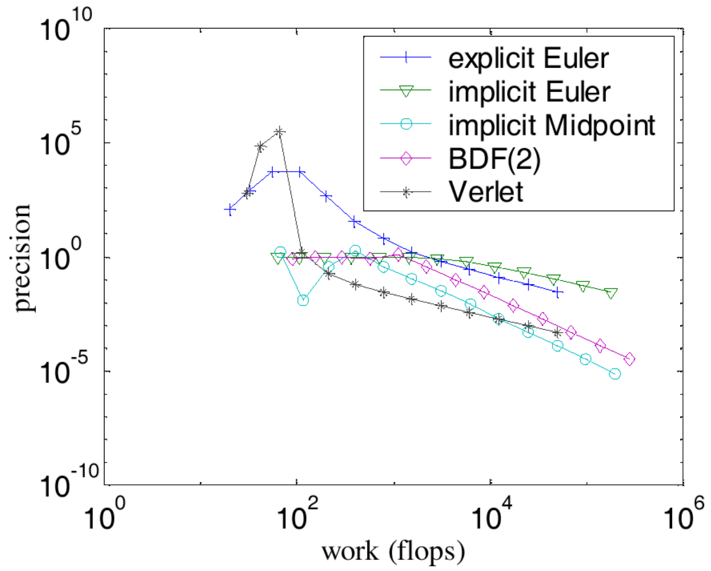




$$\ddot{x} = \frac{\lambda}{2} x + \lambda \dot{x} \quad \lambda = -5$$

$$\ddot{x} = \frac{k}{m}(l_0 - x) + \frac{d}{m}\dot{x} + g$$





Summary

- an initial value problem is composed of a DE and an initial value
- ordinary DEs only depend on single variable
- order k of a DE is highest derivative
- order k DE can be transformed to linear system of order 1 DEs.
- numeric integration methods can be characterized by their approximation order and whether they are stable
- implicit methods are stable but lead to system of non-linear equations to be solved
- step width can be adapted
- most efficient scheme depends on application

Physics Engines in JavaScript

3D

- ammo.js: https://github.com/kripken/ammo.js
- cannon.js: https://schteppe.github.io/cannon.js
- oimo.js: https://github.com/lo-th/Oimo.js

2D

- verlet-js: https://github.com/subprotocol/verlet-js
- matter-js: https://brm.io/matter-js

- overview page: https://www.tapirgames.com/blog/open-source-physics-engines