COL865: Special Topics in Computer Applications

Physics-Based Animation

5 — Time integration

Paper discussions this Thursday

1. Selle et al., “A Mass Spring Model for Hair Simulation”, 2008
2. Liu et al., “Fast Simulation of Mass-Spring Systems”, 2013

Lead: me

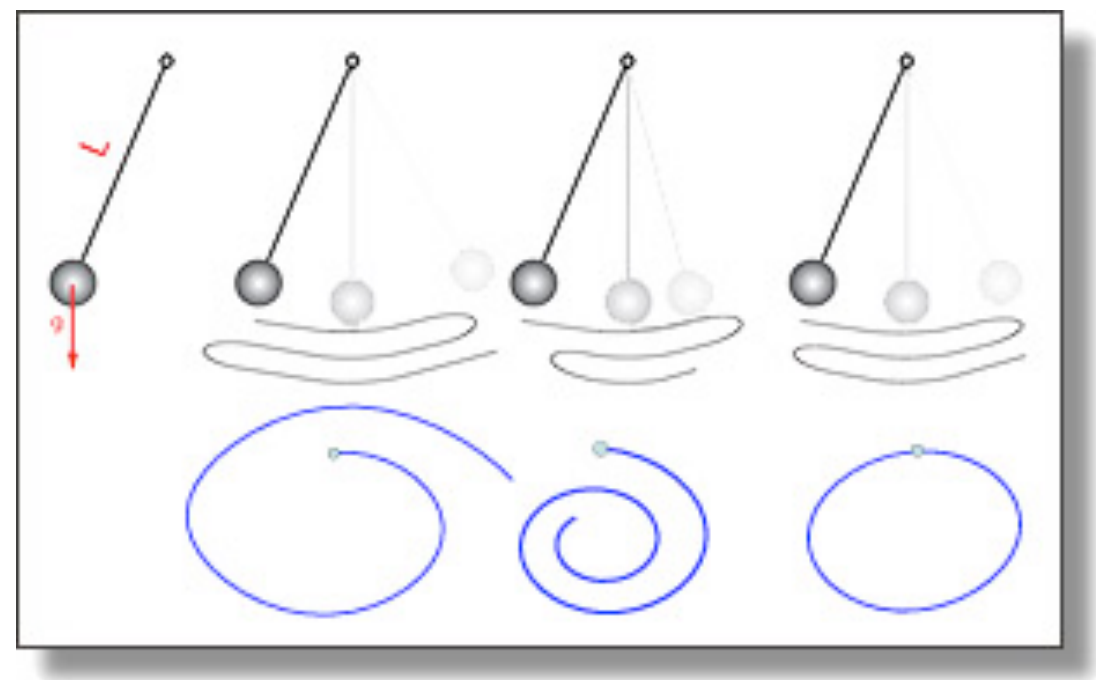
Your job:

- Read both papers before Thursday’s class
- Come prepared with questions, comments, ideas (at least one)

Today

Time integration

- Forward and backward Euler, Runge-Kutta methods, implicit methods, leapfrog and symplectic Euler



- Accuracy and stability analysis
- Reading: *Numerical Algorithms* Ch. 15

Forward Euler

$$y'(t) = \varphi(t, y(t))$$

Choose $t = t_0$, use forward difference $y'(t^0) \approx (y^1 - y^0)/\Delta t$

$$y^1 = y^0 + \varphi(t^0, y^0)\, \Delta t$$

Drawbacks:

- Based on **first-order accurate** discretization of time derivative
- Can be **unstable** if forces are stiff / time step is large

Test problem

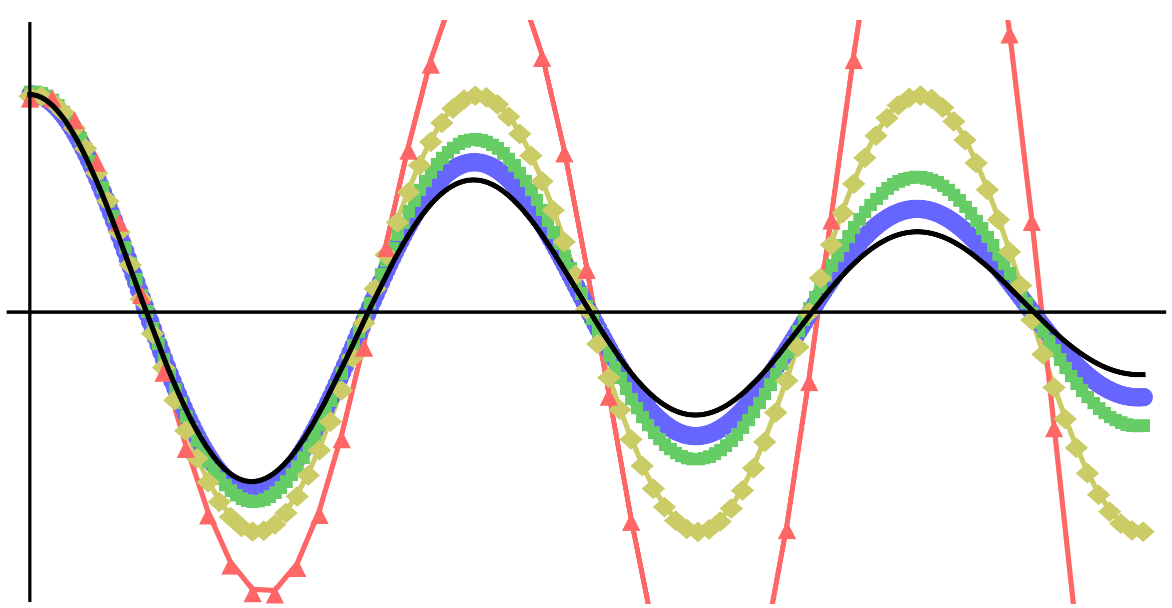
Consider a damped harmonic oscillator

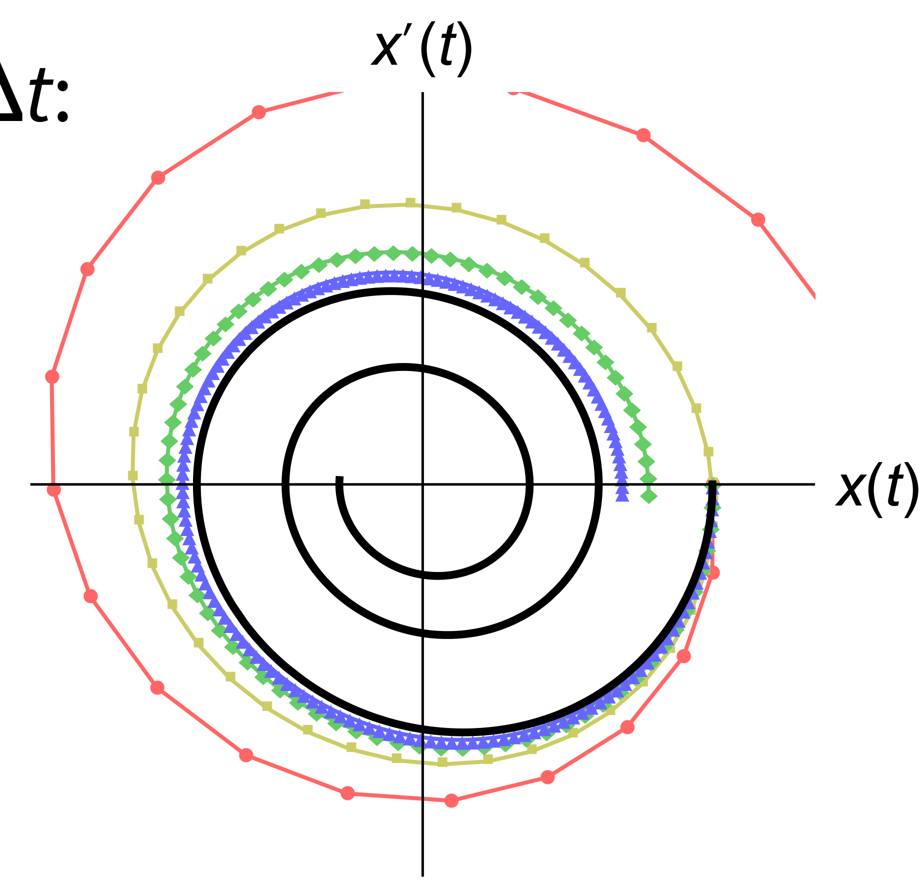
$$x'' = -kx - cx'$$

Analytical solution:

$$x(t) = e^{-ct/2}(a_1 \cos \omega t + a_2 \sin \omega t)$$
$$\omega = \sqrt{k - c^2/4}$$

Forward Euler solutions with different Δt:



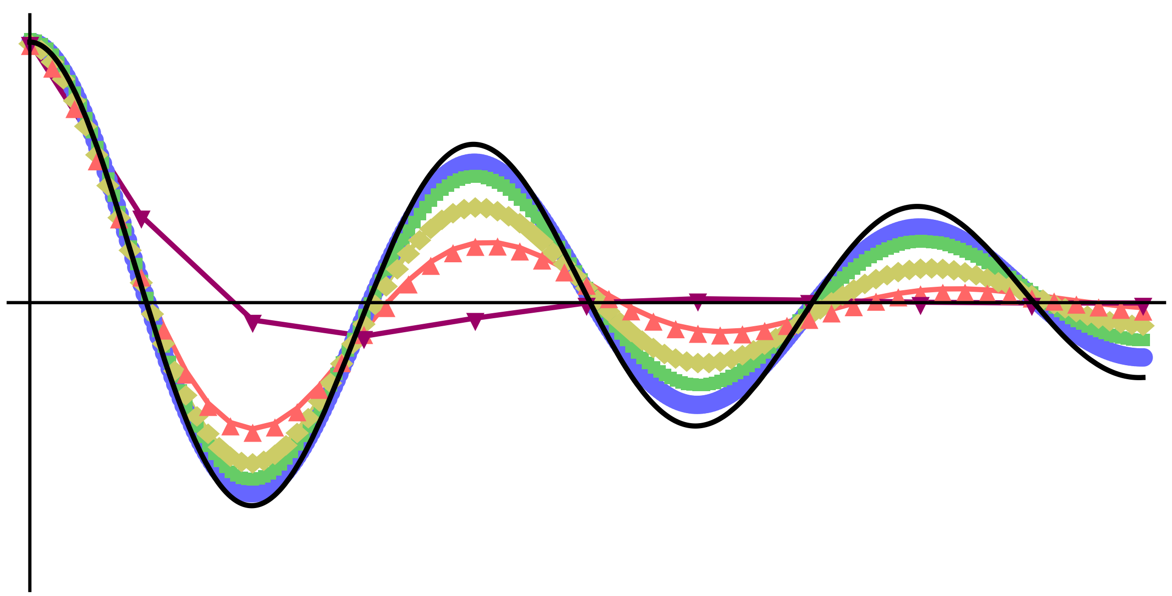


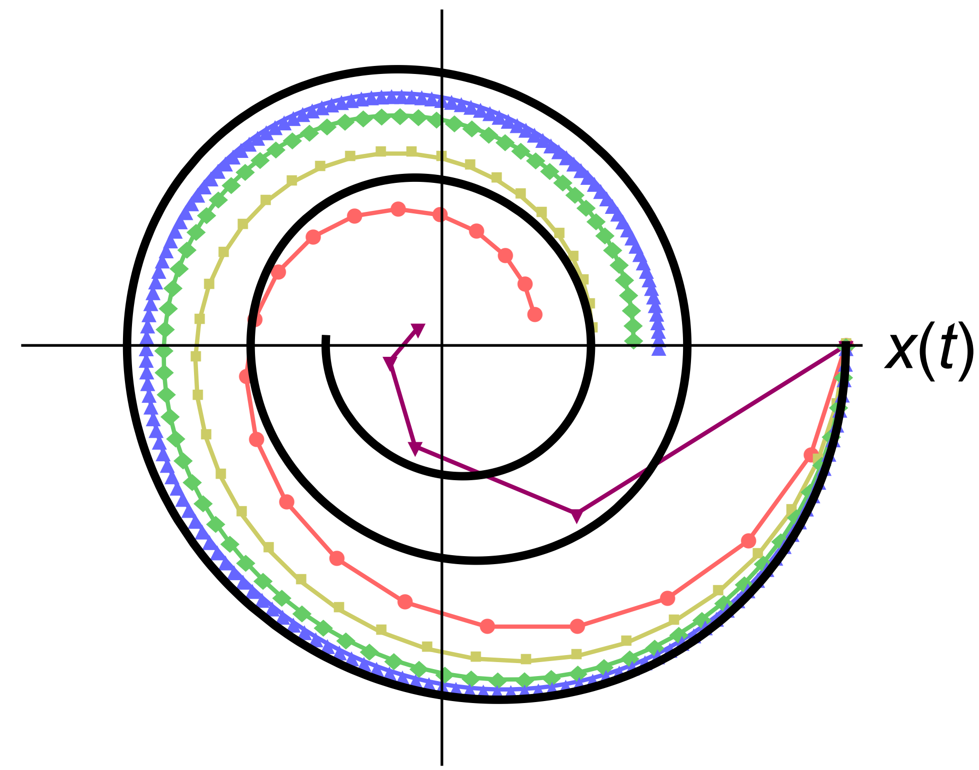
Backward Euler

$$(y^1 - y^0)/\Delta t \approx y'(t^1) = \varphi(t^1, y^1)$$

$$y^1 - \varphi(t^1, y^1)\,\Delta t = y^0$$

Also known as “**implicit Euler**” (vs. FE = explicit Euler)





Accuracy still O(Δt), but can take arbitrarily large time steps!

Drawback: Lots of **artificial dissipation**, especially for large Δt

Accuracy and stability analysis

Accuracy analysis

An oversimplified analysis:
(for rigorous version, see *Numerical Analysis* Ch. 5.2)

Taylor series: $y(t_1) = y(t_0) + y'(t_0)\,\Delta t + \tfrac{1}{2}\, y''(t_0)\,\Delta t^2 + \cdots$

Forward Euler: $y^1 = y^0 + y'(t_0)\,\Delta t$

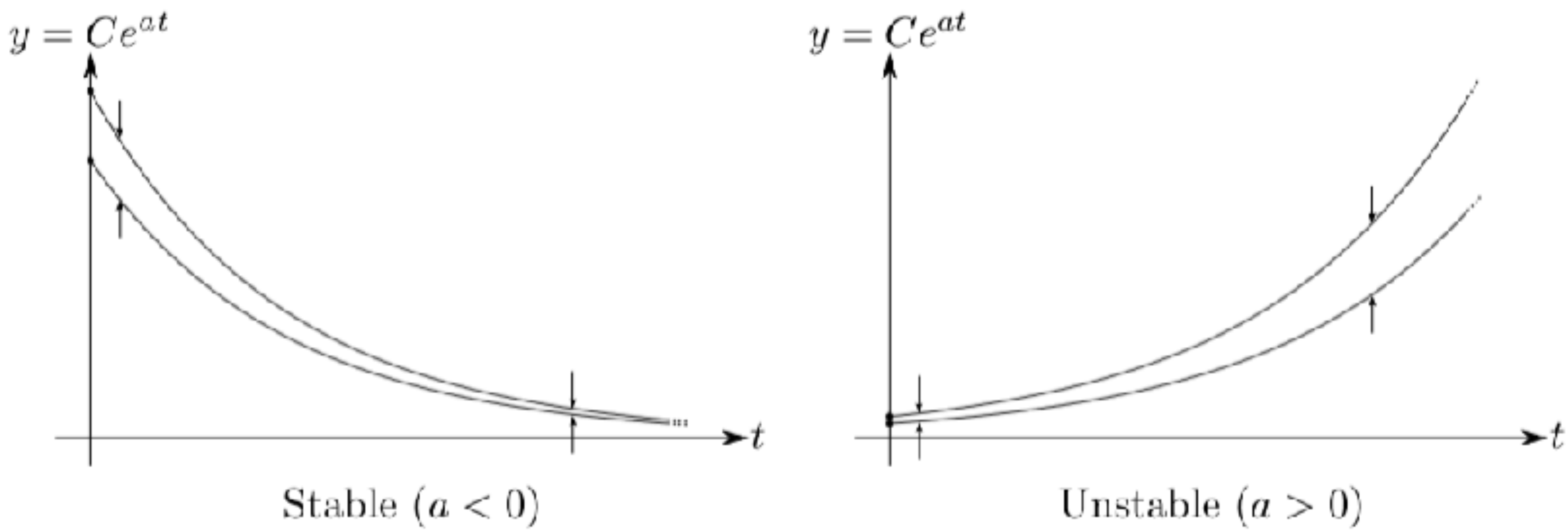
Local error between computed y^1 and true $y(t_1) = O(\Delta t^2)$

Error will accumulate over $O(1/\Delta t)$ time steps, so **global error** is $O(\Delta t)$: forward Euler is first-order accurate

Same analysis for backward Euler: also first-order

Stability

ODE is **stable** if two nearby solutions remain nearby



We want numerical solution of stable ODE to also be stable

- How to determine if an ODE is stable?
- How to determine if a numerical method is stable?

Stability analysis

Consider an **autonomous** linear ODE (RHS independent of *t*)

$$\mathbf{y}' = \mathbf{A}\mathbf{y}$$

Take eigendecomposition, $\mathbf{A} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^{-1}$
($\boldsymbol{\Lambda}$ is diagonal, both $\mathbf{Q}$ and $\boldsymbol{\Lambda}$ may be complex)

Express $\mathbf{y}$ in eigenbasis: $\mathbf{z} = \mathbf{Q}^{-1}\mathbf{y}$

$$\mathbf{z}' = \boldsymbol{\Lambda}\mathbf{z}$$

All components of $\mathbf{z}$ are decoupled!

$$\begin{gathered} z_1' = \lambda_1 z_1 \\ z_2' = \lambda_2 z_2 \\ \vdots \end{gathered}$$

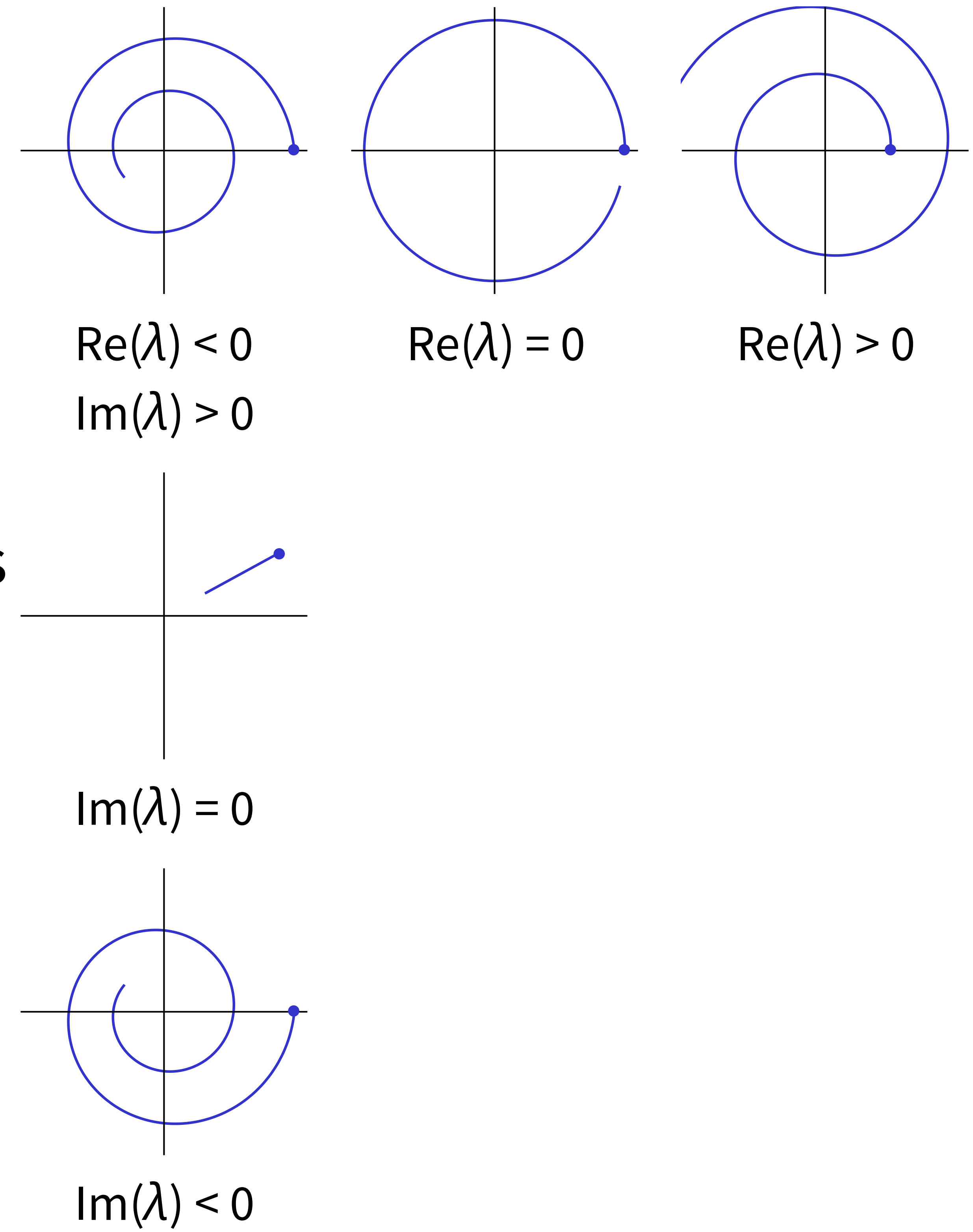
Stability analysis in 1 variable

$$z' = \lambda z$$

Solution: $z(t) = e^{\lambda t} z(0)$

- $\mathrm{Im}(\lambda)$ = frequency of oscillations
- $\mathrm{Re}(\lambda)$ = rate of growth/decay

ODE is stable if $\mathrm{Re}(\lambda) \leq 0$



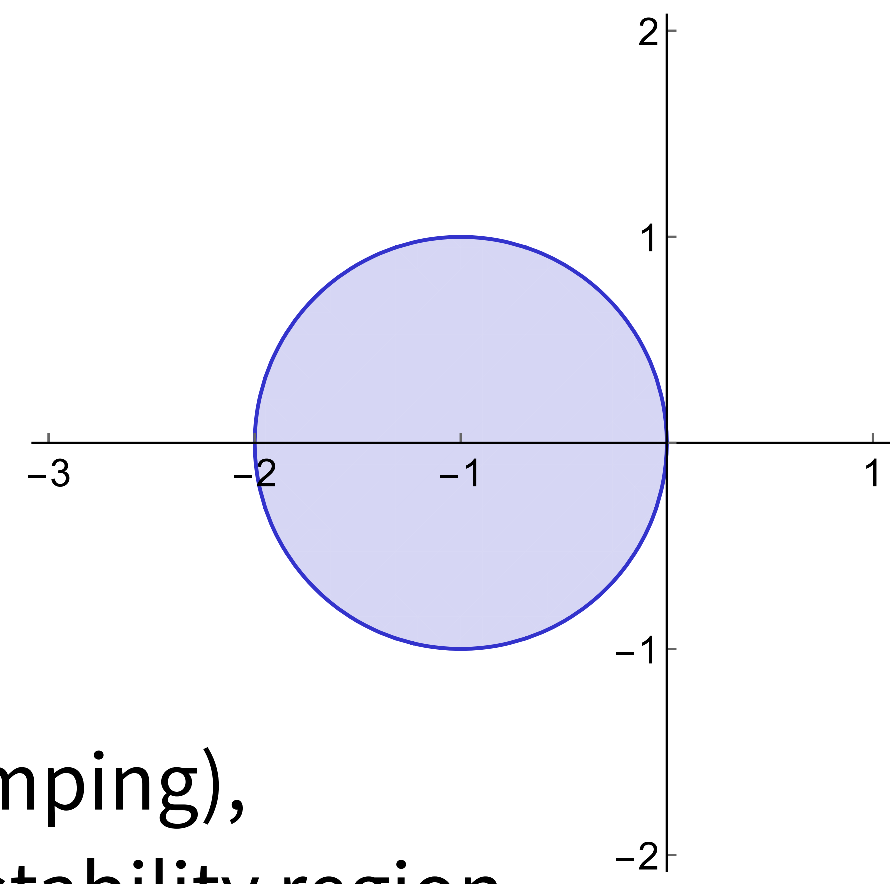
Stability analysis of forward Euler

Forward Euler: $z^1 = (1 + \lambda\, \Delta t)\, z^0$

$\Rightarrow z^n = (1 + \lambda\, \Delta t)^n\, z^0$

FE solution is stable if $|1 + \lambda\, \Delta t| \leq 1$

$\lambda\, \Delta t$ must lie in the **stability region:**



- With larger λ (stiffer springs / more damping), Δt must become smaller to remain in stability region
- If $\mathrm{Re}(\lambda) = 0$ (no damping), forward Euler is not stable for any Δt!

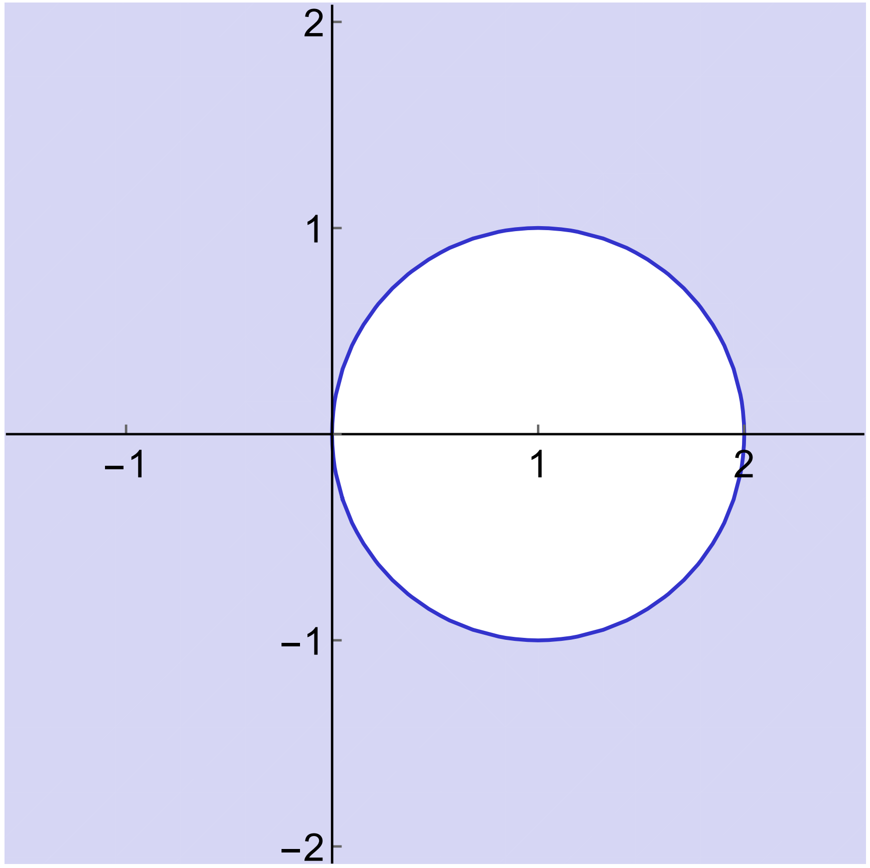
Stability analysis of backward Euler

Backward Euler: $z^1 = (1 - \lambda\,\Delta t)^{-1}\, z^0$

$\Rightarrow z^n = (1 - \lambda\,\Delta t)^{-n}\, z^0$

Stable if $|1 - \lambda\,\Delta t|^{-1} \leq 1\ldots$
Always true if $\mathrm{Re}(\lambda) \leq 0$!

BE is **unconditionally stable**,
or **A-stable**



Review

For any autonomous linear ODE:

- Forward Euler is stable if $|1 + \lambda\,\Delta t| \leq 1$
- Backward Euler is always stable if ODE is stable

Stability condition must hold for **every** eigenvalue λ_i …
Bad news for FE if even a single extremely stiff force in system

Runge-Kutta methods

Higher-order methods

Recall interpretation of ODE as quadrature:

$$y_1 - y_0 = \int_{t_0}^{t_1} \varphi\left(t, y(t)\right) \mathrm{d}t$$

Forward Euler = quadrature point at start of interval

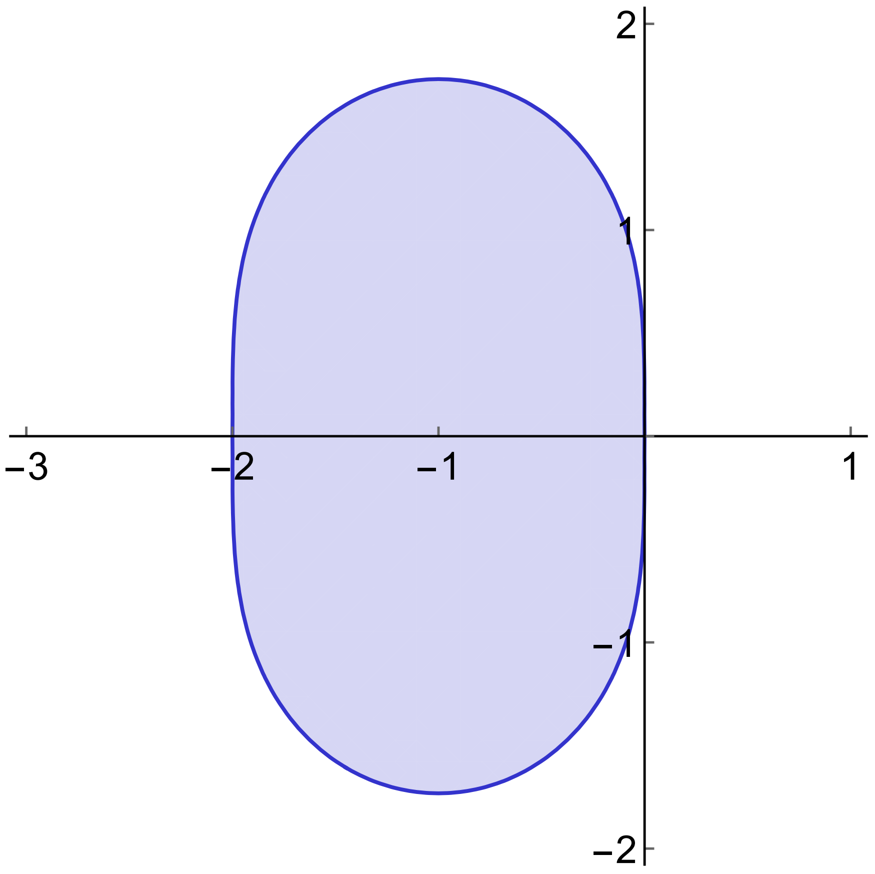
$$y^1 = y^0 + \varphi(t^0, y^0)\, \Delta t$$

Midpoint method = quadrature point at center. But what's $y^{½}$?
Take **explicit** approximation: FE step of length $\Delta t/2$

$$y^{½} = y^0 + \varphi(t^0, y^0)\, \Delta t/2$$
$$y^1 = y^0 + \varphi(t^{½}, y^{½})\, \Delta t$$

Explicit midpoint method

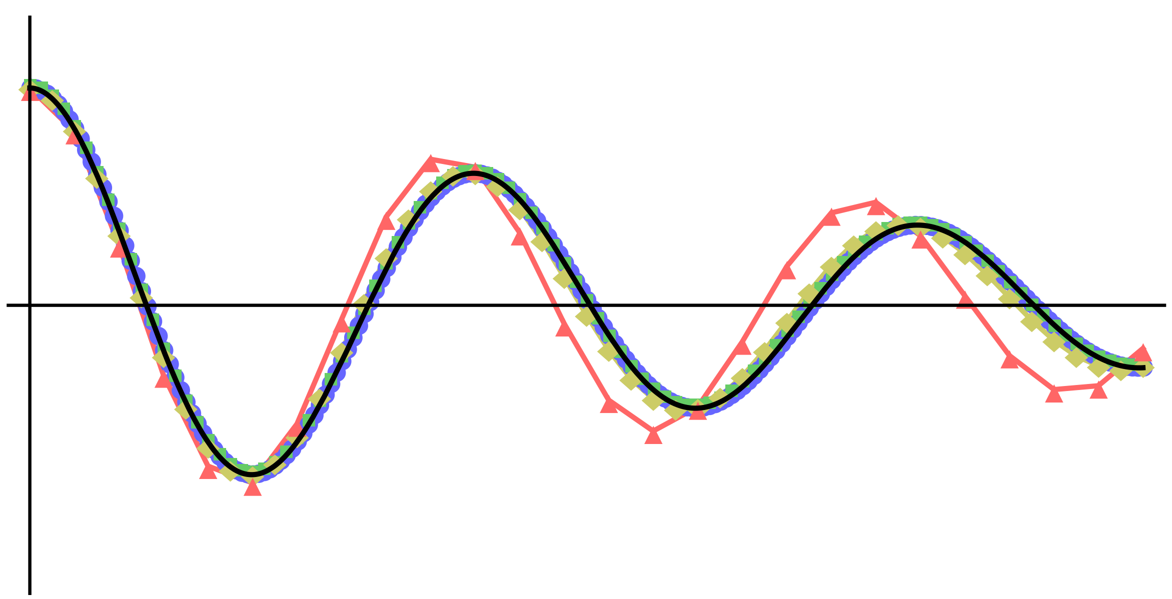
$$y^{½} = y^0 + \varphi(t^0, y^0)\,\Delta t/2$$
$$y^1 = y^0 + \varphi(t^{½}, y^{½})\,\Delta t$$

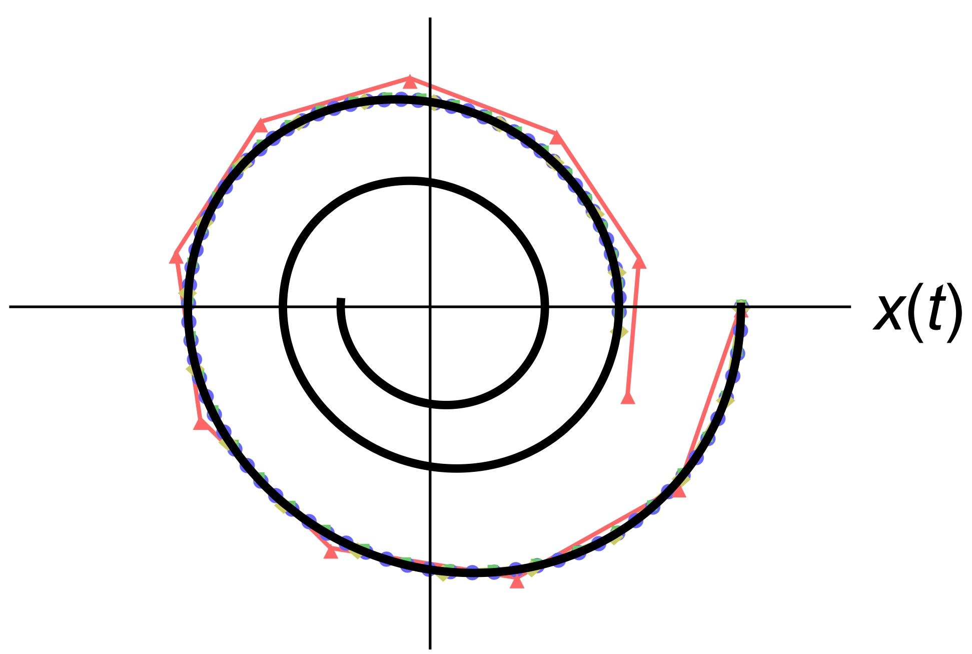


Second-order accurate

- Even though $y^{½}$ is only first-order… why?

Stability region: $|1 + \lambda\,\Delta t + ½\,(\lambda\,\Delta t)^2| \le 1$





Runge-Kutta methods

Equivalent form:

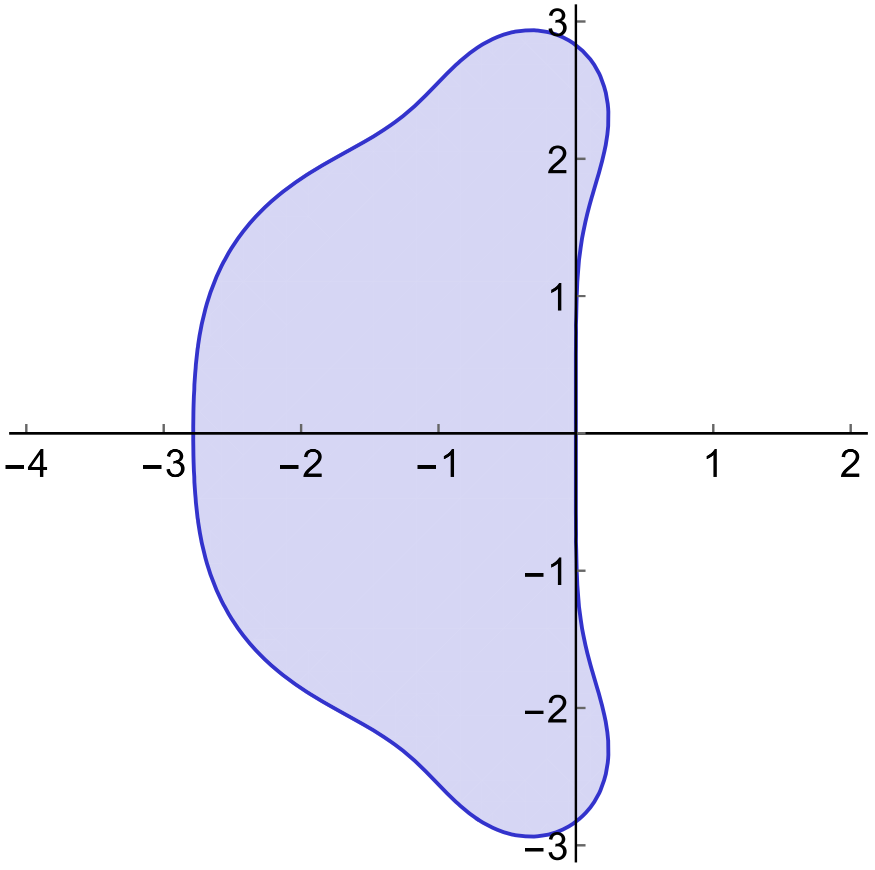
$$\varphi^0 = \varphi(t^0, y^0)$$
$$\varphi^{1/2} = \varphi(t^0 + \tfrac{1}{2}\,\Delta t, y^0 + \tfrac{1}{2}\,\varphi^0\,\Delta t)$$
$$y^1 = y^0 + \varphi^{1/2}\,\Delta t$$

Higher-order generalization: Evaluate φ at various quadrature points, chosen using previously evaluated values of φ

$$\varphi_0 = \varphi(t^0 + 0\,\Delta t, y^0)$$
$$\varphi_1 = \varphi(t^0 + c_1\,\Delta t, y^0 + (a_{10}\,\varphi_0)\,\Delta t)$$
$$\varphi_2 = \varphi(t^0 + c_2\,\Delta t, y^0 + (a_{20}\,\varphi_0 + a_{21}\,\varphi_1)\,\Delta t)$$
$$\vdots$$
$$y^1 = y^0 + (b_0\,\varphi_0 + b_1\,\varphi_1 + b_2\,\varphi_2 + \cdots)\,\Delta t$$

RK4: "the" Runge-Kutta method

$$\varphi_0 = \varphi(t^0 + 0\,\Delta t, y^0)$$
$$\varphi_1 = \varphi(t^0 + \tfrac{1}{2}\,\Delta t, y^0 + \tfrac{1}{2}\,\varphi_0\,\Delta t)$$
$$\varphi_2 = \varphi(t^0 + \tfrac{1}{2}\,\Delta t, y^0 + \tfrac{1}{2}\,\varphi_1\,\Delta t)$$
$$\varphi_3 = \varphi(t^0 + 1\,\Delta t, y^0 + \varphi_2\,\Delta t)$$
$$y^1 = y^0 + (\tfrac{1}{6}\,\varphi_0 + \tfrac{1}{3}\,\varphi_1 + \tfrac{1}{3}\,\varphi_2 + \tfrac{1}{6}\,\varphi_3)\,\Delta t$$



Fourth-order accurate

Reduces to Simpson's rule if $\varphi(t)$ independent of y

Butcher tableaus

Compact way of expressing RK methods

$$\varphi_0 = \varphi(t^0 + 0\,\Delta t, y^0)$$
$$\varphi_1 = \varphi(t^0 + c_1\,\Delta t, y^0 + (a_{10}\,\varphi_0)\,\Delta t)$$
$$\varphi_2 = \varphi(t^0 + c_2\,\Delta t, y^0 + (a_{20}\,\varphi_0 + a_{21}\,\varphi_1)\,\Delta t)$$
$$\vdots$$
$$y^1 = y^0 + (b_0\,\varphi_0 + b_1\,\varphi_1 + b_2\,\varphi_2 + \cdots)\,\Delta t$$

$$\begin{array}{c|cccc}
0 & & & & \\
c_1 & a_{10} & & & \\
c_2 & a_{21} & a_{22} & & \\
\vdots & \vdots & \vdots & \ddots & \\
\hline
 & b_0 & b_1 & b_2 & \cdots
\end{array}$$

Example: RK4

$$\begin{array}{c|cccc}
0 & & & & \\
1/2 & 1/2 & & & \\
1/2 & 0 & 1/2 & & \\
1 & 0 & 0 & 1 & \\
\hline
 & 1/6 & 1/3 & 1/3 & 1/6
\end{array}$$

Butcher tableaus

- What is the tableau for explicit midpoint?
- **Heun's method** is given by the following tableau.
  How does it work?

$$\begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & 1/2 & 1/2 \end{array}$$

- What quadrature scheme does it look like?

Implicit methods

Second-order implicit methods

EM, Heun's method: second-order, not unconditionally stable

Instead of approximating future *y*'s, make them **implicit** in y^1

Implicit midpoint:

$$y^1 = y^0 + \varphi\left(\frac{t^0 + t^1}{2}, \frac{y^0 + y^1}{2}\right)\Delta t$$

Trapezoidal method:

$$y^1 = y^0 + \left(\frac{\varphi(t^0, y^0) + \varphi(t^1, y^1)}{2}\right)\Delta t$$

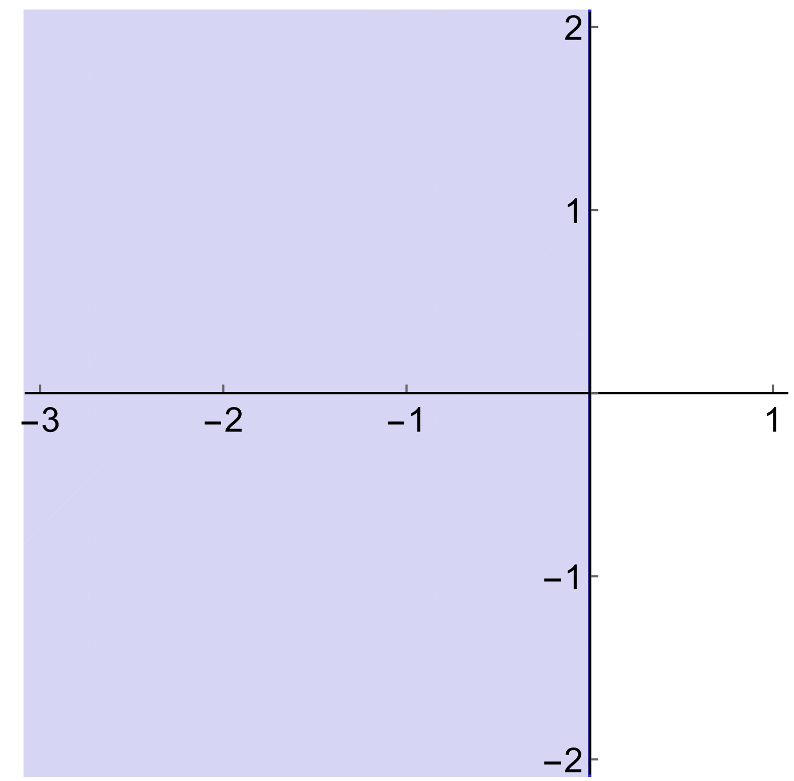
Both second-order accurate, equivalent for linear problems

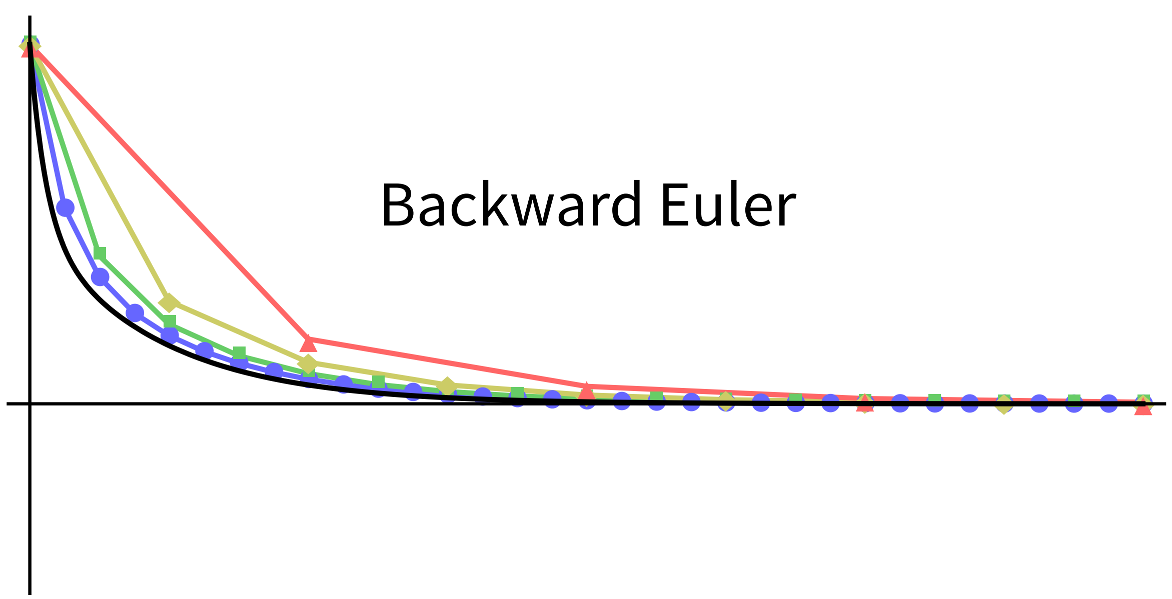
Stability

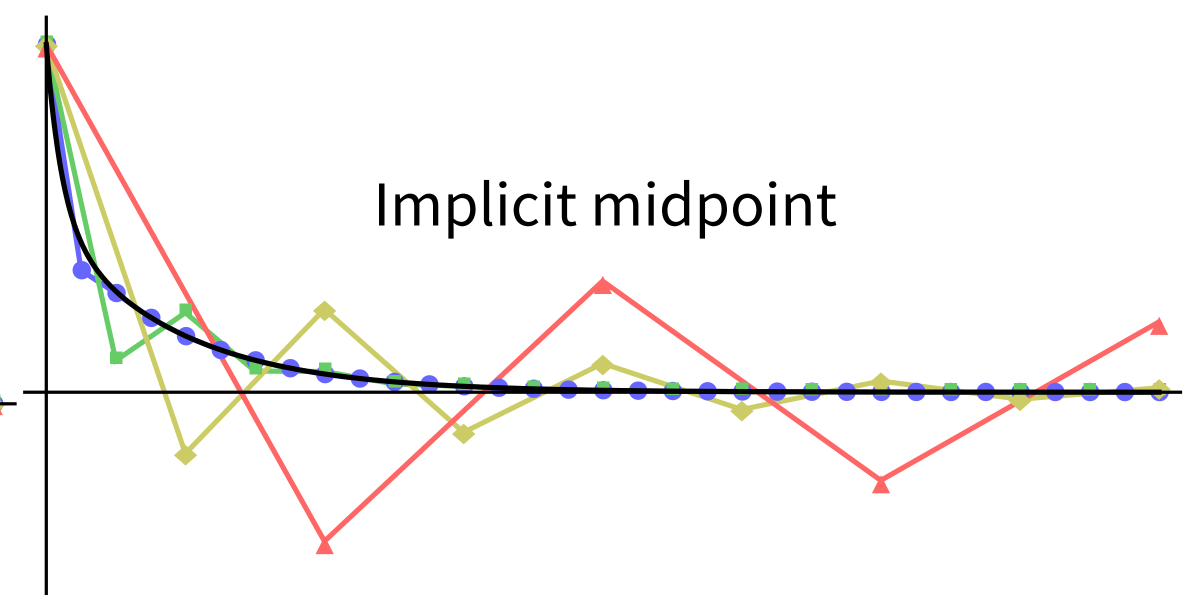
Implicit midpoint and trapezoidal method are unconditionally stable

Stability region: $\mathrm{Re}(\lambda) \le 0$

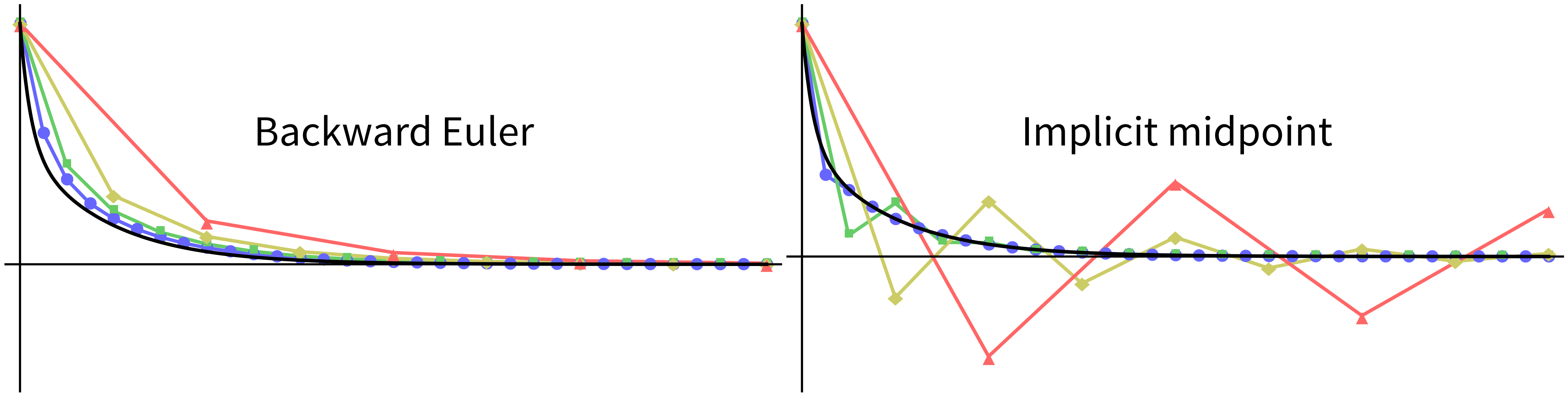
Artificial oscillations for large time steps







L-stability



A-stability: If $\mathrm{Re}(\lambda) \le 0$, then $|y^1|/|y^0| \le 1$

L-stability: If $\mathrm{Re}(\lambda) \le 0$ and $\Delta t \to \infty$, then $|y^1|/|y^0| \to 0$

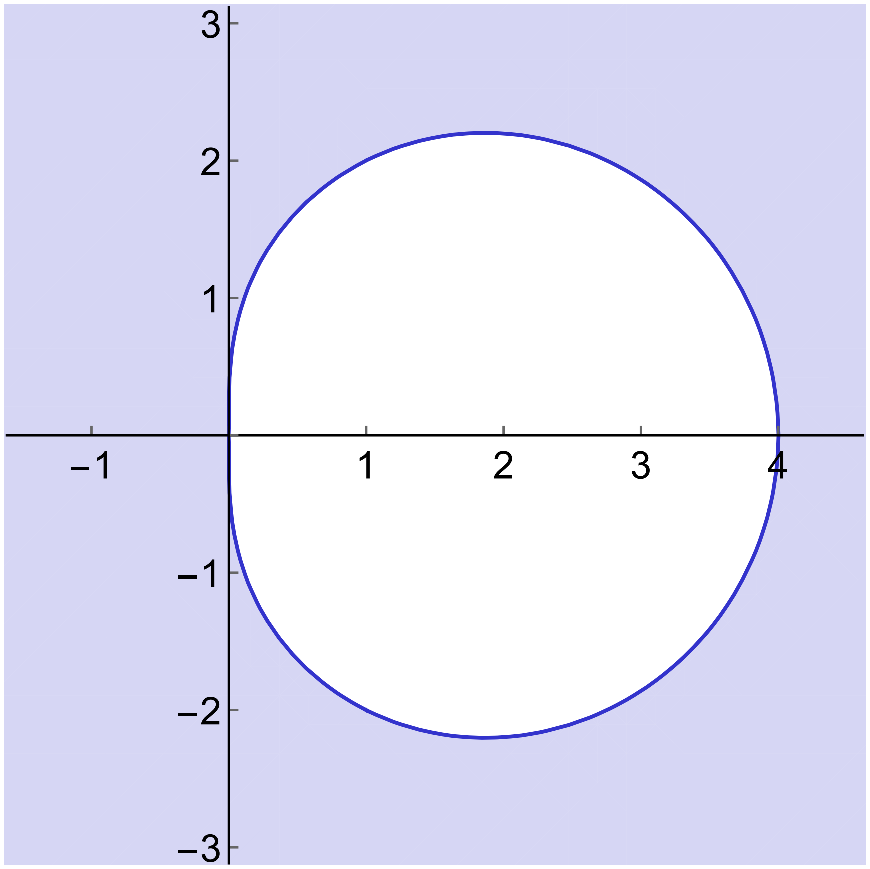
Backward Euler is L-stable, but implicit midpoint and trapezoidal method are not

BDF2

What if we use **second-order** finite difference approximation of y'?

- Centered differences is no good. Why?
- Second-order backward differences:

$$y'(t^{n+1}) = (3/2\, y^{n+1} - 2\, y^n + 1/2\, y^{n-1})/\Delta t$$



Second-order accurate, A-stable, L-stable, less dissipative than BE

This is a **multistep method**: uses two previous states y^n and y^{n-1}

Downsides: Needs to be "kickstarted" with one-step method, not good near nonsmooth points (e.g. collisions)

Symplectic methods

Verlet/leapfrog integration

Suppose $\mathbf{f}$ independent of $\mathbf{v}$. Then we can just write $\mathbf{x}'' = \mathbf{M}^{-1}\,\mathbf{f}(\mathbf{x})$.

Apply three-point formula for second derivative:

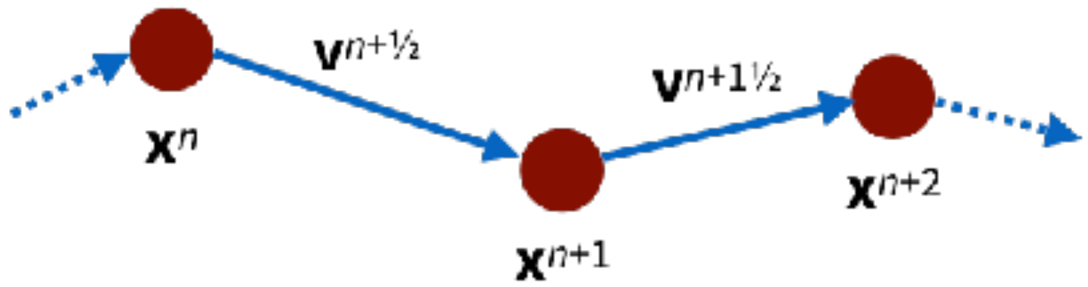
$$(\mathbf{x}^{n+1} - 2\,\mathbf{x}^{n} + \mathbf{x}^{n-1})/\Delta t^2 = \mathbf{M}^{-1}\,\mathbf{f}(\mathbf{x}^{n})$$

$$\Rightarrow \mathbf{x}^{n+1} = 2\,\mathbf{x}^{n} - \mathbf{x}^{n-1} + \mathbf{M}^{-1}\,\mathbf{f}(\mathbf{x}^{n})\,\Delta t^2$$

Another interpretation: Let $(\mathbf{x}^{n} - \mathbf{x}^{n-1})/\Delta t = \mathbf{v}^{n-½}$

$$\mathbf{v}^{n+½} = \mathbf{v}^{n-½} + \mathbf{M}^{-1}\,\mathbf{f}(\mathbf{x}^{n})\,\Delta t$$
$$\mathbf{x}^{n+1} = \mathbf{x}^{n} + \mathbf{v}^{n+½}\,\Delta t$$

Also called “leapfrog” integration



Symplectic Euler

Compare forward Euler:

$$\mathbf{v}^1 = \mathbf{v}^0 + \mathbf{M}^{-1}\,\mathbf{f}(\mathbf{x}^0, \mathbf{v}^0)\,\Delta t$$
$$\mathbf{x}^1 = \mathbf{x}^0 + \mathbf{v}^0\,\Delta t$$

Backward Euler:

$$\mathbf{v}^1 = \mathbf{v}^0 + \mathbf{M}^{-1}\,\mathbf{f}(\mathbf{x}^1, \mathbf{v}^1)\,\Delta t$$
$$\mathbf{x}^1 = \mathbf{x}^0 + \mathbf{v}^1\,\Delta t$$

Symplectic Euler: implicit in $\mathbf{v}$, explicit in $\mathbf{x}$

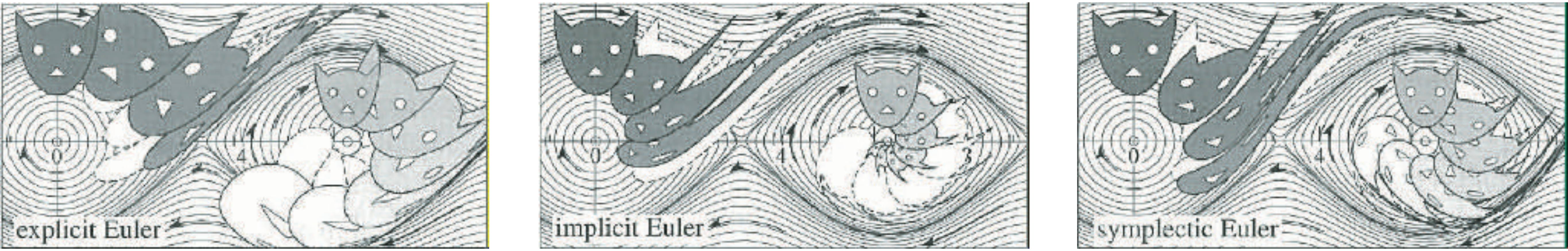
$$\mathbf{v}^1 = \mathbf{v}^0 + \mathbf{M}^{-1}\,\mathbf{f}(\mathbf{x}^0, \mathbf{v}^1)\,\Delta t$$
$$\mathbf{x}^1 = \mathbf{x}^0 + \mathbf{v}^1\,\Delta t$$

Using $\mathbf{f}(\mathbf{x}^0, \mathbf{v}^0)$ is usually good enough — then no solve needed

Symplecticity

“Symplectic” is a technical term with a complicated meaning

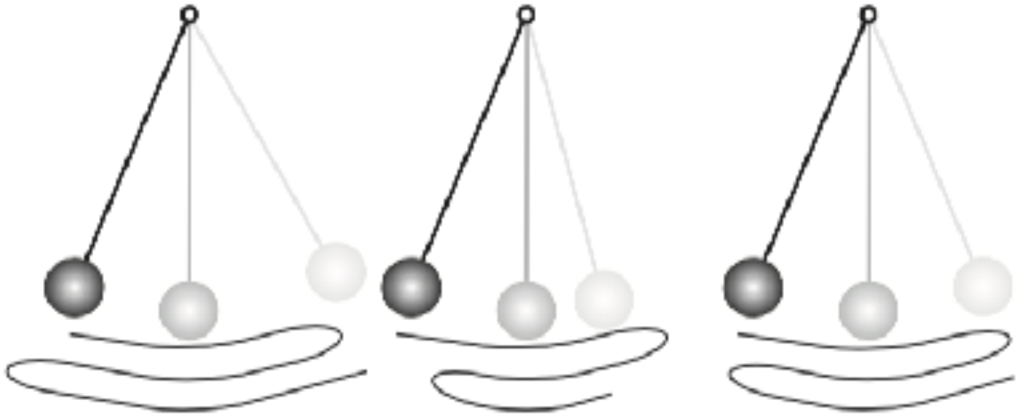
In absence of damping, symplectic methods conserve volumes in phase space (even for nonlinear systems!)



[Hairer et al. 2002]

Tend to conserve energy in the long term (**if** they remain stable)

Leapfrog, SE, IM are symplectic



[Stern and Desbrun 2006]

Summary

Summary of integration methods

Name	Accuracy	Exp/imp	Stability	Symplectic
Forward Euler	1st order	explicit	conditional	no
Backward Euler	1st order	implicit	L-stable	no
Explicit midpoint, Heun’s method	2nd order	explicit	conditional	no
RK4	4th order	explicit	conditional	no
Implicit midpoint, trapezoidal	2nd order	implicit	A-stable	IM: yes, TR: no
BDF2	2nd order	implicit	L-stable	no
Leapfrog, symplectic Euler	LF: 2nd, SE: 1st	explicit / semi-implicit	conditional	yes

Summary of integration methods

Which to use?

- Unconditional stability on stiff problems: BE, BDF2
- Long-term energy conservation: SE, IM, TR
- Fast and easy to implement: SE, EM, Heun's
- High-accuracy reference solution for validation: RK4

FE: just don't

Implementation notes

Always try to decouple model and integrator

Model should provide methods to:

- get number of DOFs n,
- get/set current state vector $\mathbf{y} \in \mathbb{R}^n$,
- evaluate current time derivative $\boldsymbol{\varphi}(\mathbf{y}) \in \mathbb{R}^n$

Then you can switch integrators as needed, reuse integrator code for different problems, etc.

Implementation notes

e.g. Explicit midpoint:

```
y0 = model.getState()
φ0 = model.getDerivative()
model.setState(y0 + φ0 Δt/2)
φh = model.getDerivative()
model.setState(y0 + φh Δt)
```

See Pixar notes for more

Implementation notes

For some integrators (BE, SE, …), may need to provide more:

- get/set current position $\mathbf{x}$, velocity $\mathbf{v} \in \mathbb{R}^n$,
- get inertia matrix $\mathbf{M} \in \mathbb{R}^{n \times n}$,
- compute current force $\mathbf{f}(\mathbf{x}, \mathbf{v}) \in \mathbb{R}^n$

Adaptive time stepping

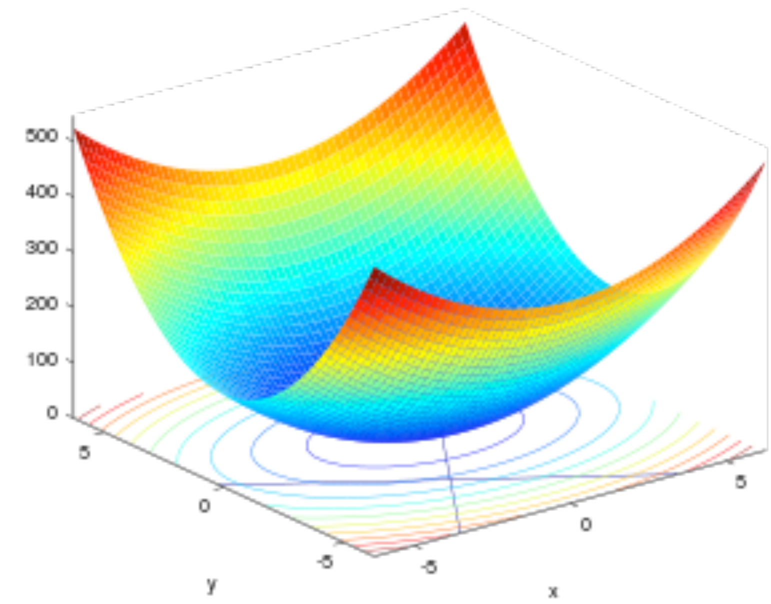
If **f** is changing rapidly, may need to reduce Δt for accuracy/ stability. How to know when?

- **In numerical analysis**: see Burden & Faires Ch. 5.5
- **In graphics**: problem-dependent time step criteria
  - Collisions not resolved $\Rightarrow$ reduce Δt
  - Continuum mechanics (elasticity, fluids, etc.) $\Rightarrow$ stability criteria (e.g. CFL condition)

Caveats: Adaptivity is hard to do for BDF2, breaks properties of symplectic methods

Next class

Equation solving and optimization



- Solving systems of equations for implicit methods
- Numerical optimization for robust quasistatic simulation
- Solomon, *Numerical Algorithms*, Ch. 8;
  Nocedal and Wright, *Numerical Optimization*, Ch. 2, 3