Numerical solution of ODEs in 15 minutes

Thomas Tram

thomas.tram@epfl.ch

The first ODE method

Consider only *explicit* systems of *first order* equations with known *initial conditions*:

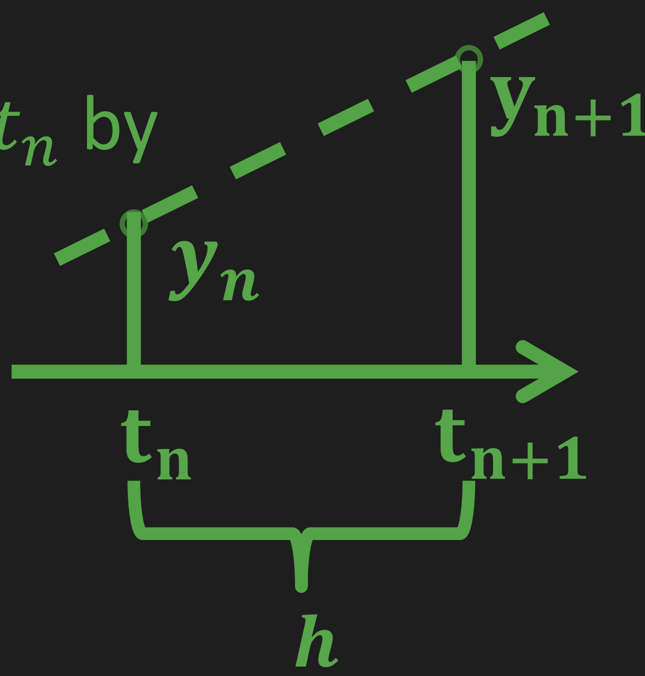
$$y' = f(t, y), \qquad y_i = y(t_i).$$

Approximate the derivative at time t_n by

$$y'_n \simeq \frac{y_{n+1} - y_n}{t_{n+1} - t_n},$$

we find the *forwards* Euler method:

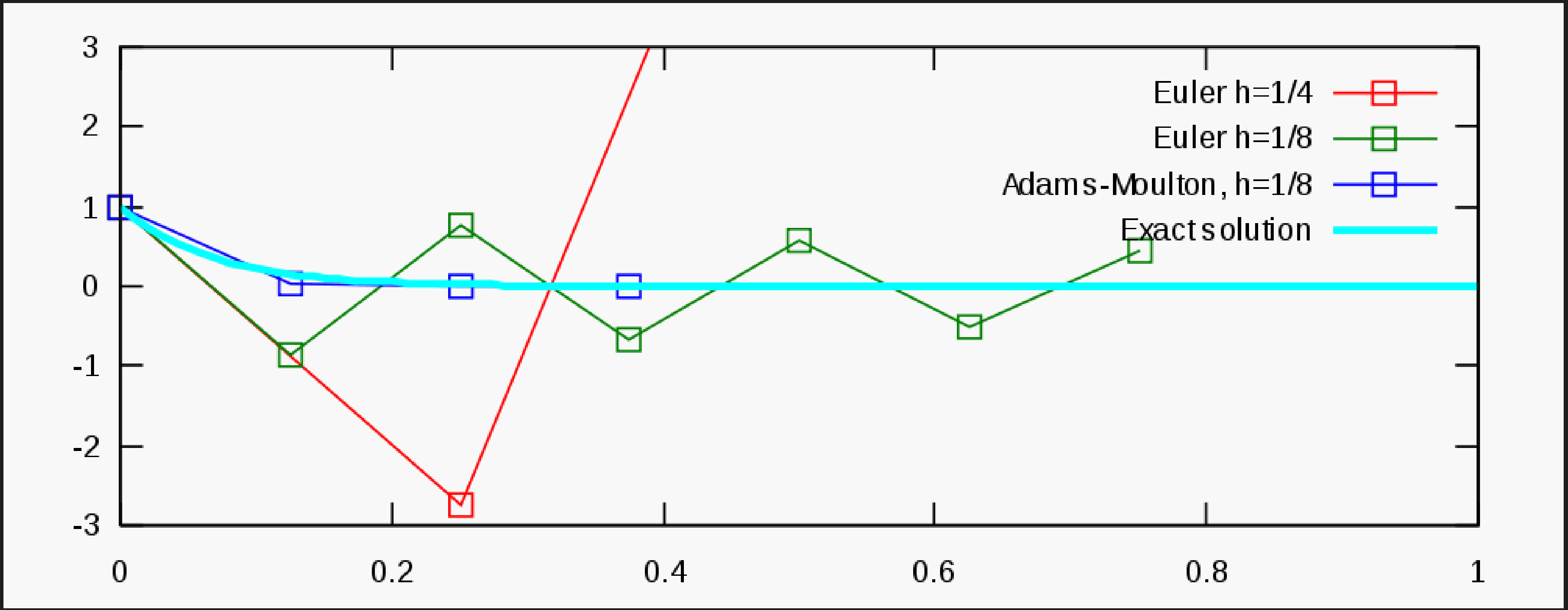
$$y_{n+1} = y_n + f(t_n, y_n)h$$



...but never use it!

Consider a test equation

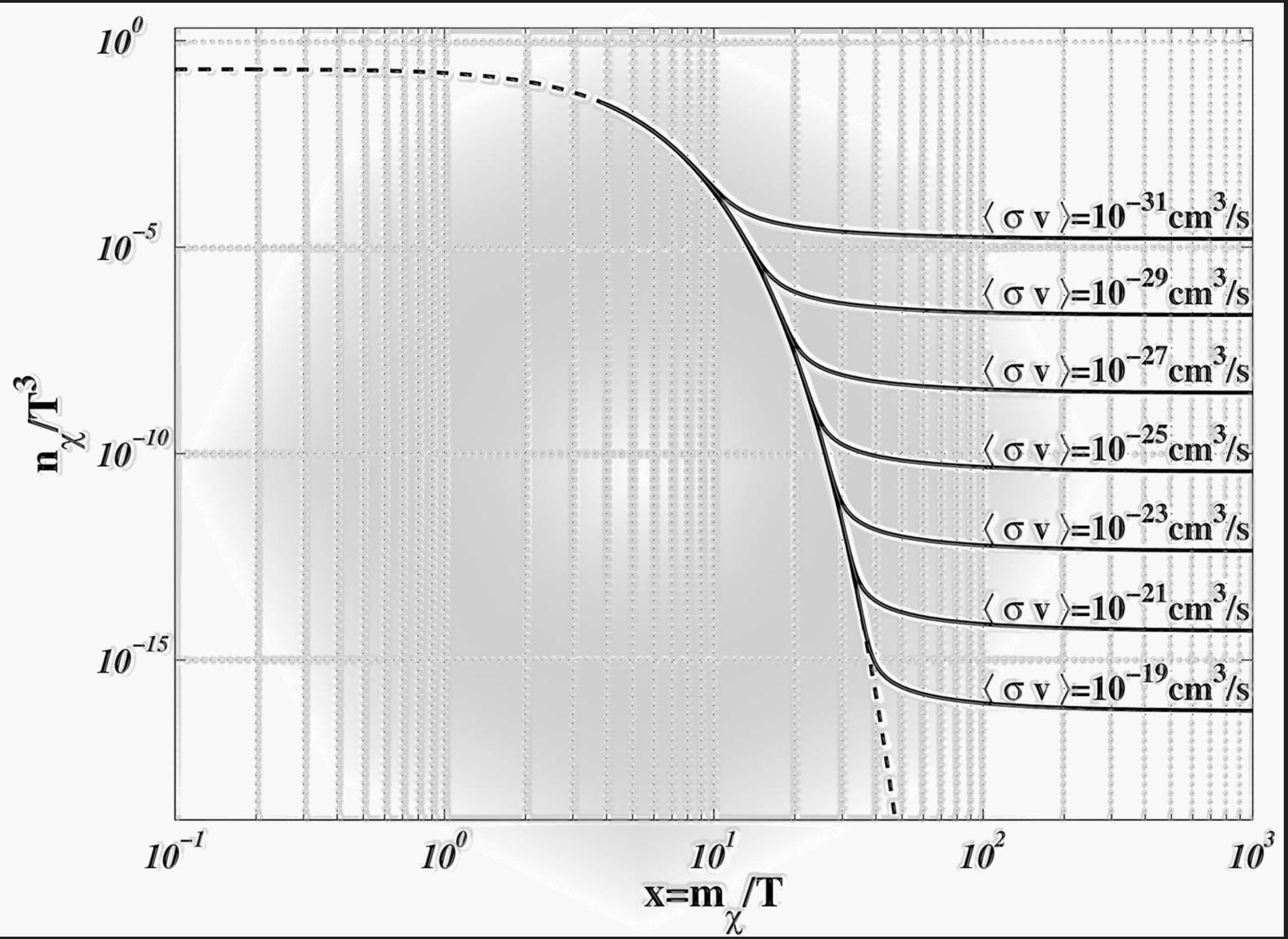
$$y' = -15y, \qquad y(t) = y(0)e^{-15t}$$



What went wrong?

- Different time scales
  - the dynamic time scale is different from the time scale of interest.
  - Cosmology: $\tau_{\text{int}}\ vs\ \tau_{H_0}$
  - Example from before: $\tau_{\text{int}} = \frac{1}{15}\ vs\ [0,1]$
- Equilibrium
  - a trivial equilibrium solution exists.
  - Cosmology: Tight coupling limit
  - Example from before: $y(t) \to 0$

Similar to WIMP Freeze-Out



The test equation

Consider the test equation

$$y' = ay, \qquad y(t) = y(0)e^{at}.$$

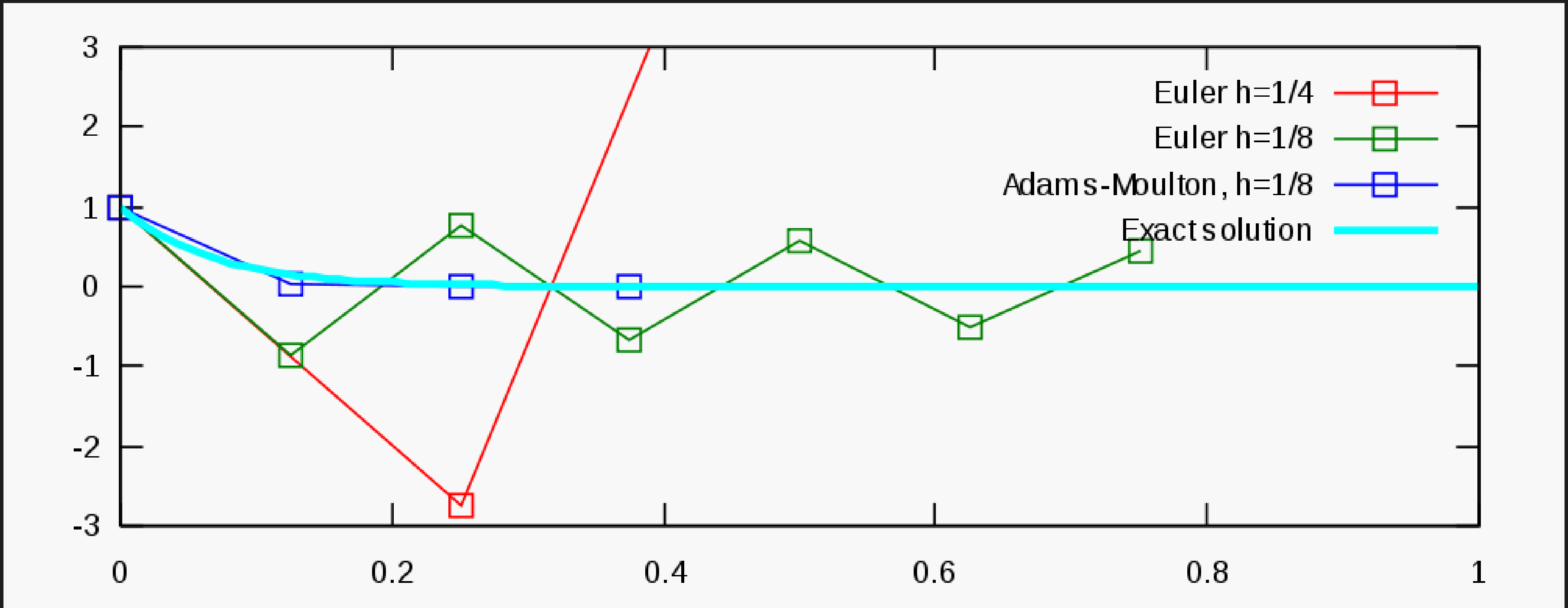
The forwards Euler method reads

$$\begin{aligned} y_{n+1} &= y_n + f(t, y_n)h \\ &= y_n + ay_nh \\ &= (1 + ah)y_n \end{aligned}$$

Remaining bounded for $\mathrm{Re}(a) < 0$ requires $\|1 + ah\| \leq 1$. Thus $a = -15$ requires $h < \frac{2}{15}$.

Stability issue revisited

So we must have $h < \frac{2}{15}$, **even during equilibrium evolution!**



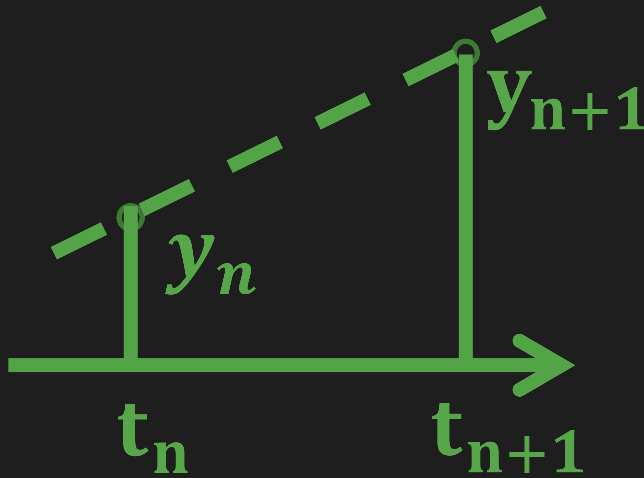
Let's try something different...

Approximate the derivative at time t_{n+1} by

$$y'_{n+1} \simeq \frac{y_{n+1} - y_n}{t_{n+1} - t_n}$$

This leads to the *backwards* Euler method:

$$y_{n+1} = y_n + f(t_{n+1}, y_{n+1})h$$



Backwards Euler

The equation

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \boldsymbol{f}(t_{n+1}, \boldsymbol{y}_{n+1})h$$

is in general a system of non-linear, coupled equations. Bad idea?

Consider $y' = ay$:

$$y_{n+1} = y_n + a y_{n+1} h \Rightarrow$$

$$y_{n+1} = \frac{1}{1 - ah} y_n.$$

Always stable for $\boldsymbol{Re(a) < 0}$!

Best method for perturbations?

Explicit method

Pros:

- Easy to code ODE-solver
- Fast (well) after tight coupling

Cons:

- Stiffness must be removed by hand by TCA
- Not robust against new physics

Implicit method

Pros:

- Eliminate the need for TCA
- Very robust against users

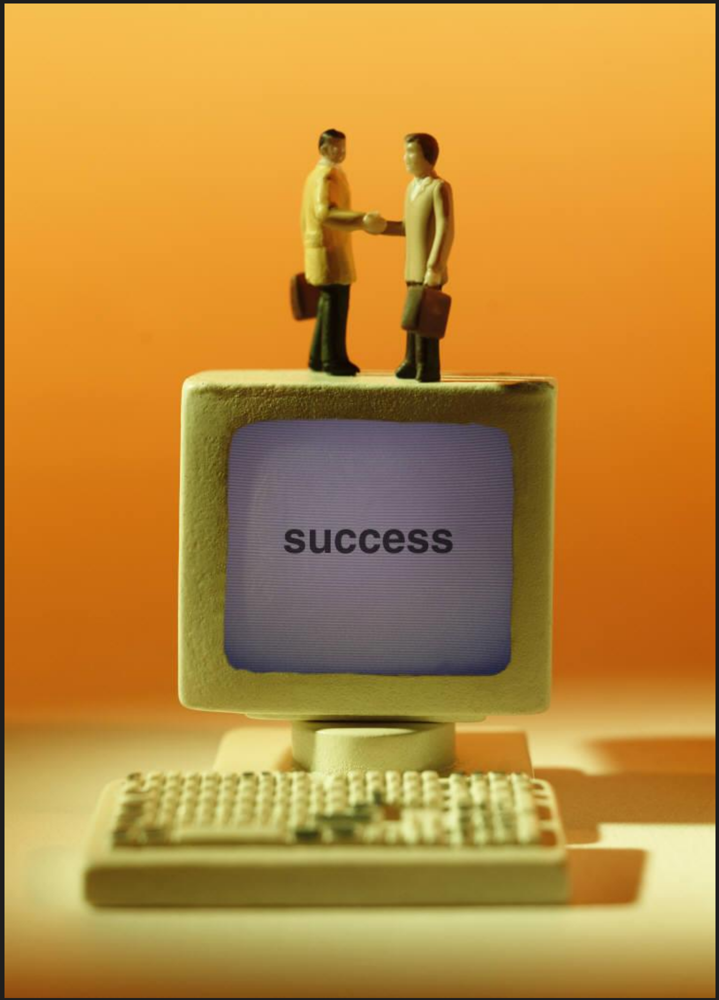
Cons:

- Can be slow due to algebraic system
- More difficult to code

evolver_ndf15.c

ndf15: multistep extension of backwards Euler.

- Speed relies on
  - Variable order 1-5
  - Adaptive step size
  - Dense output
  - Recycle Jacobians for Newtons method
  - Sparse LU decompositions



Various slides not used

Runge-Kutta methods

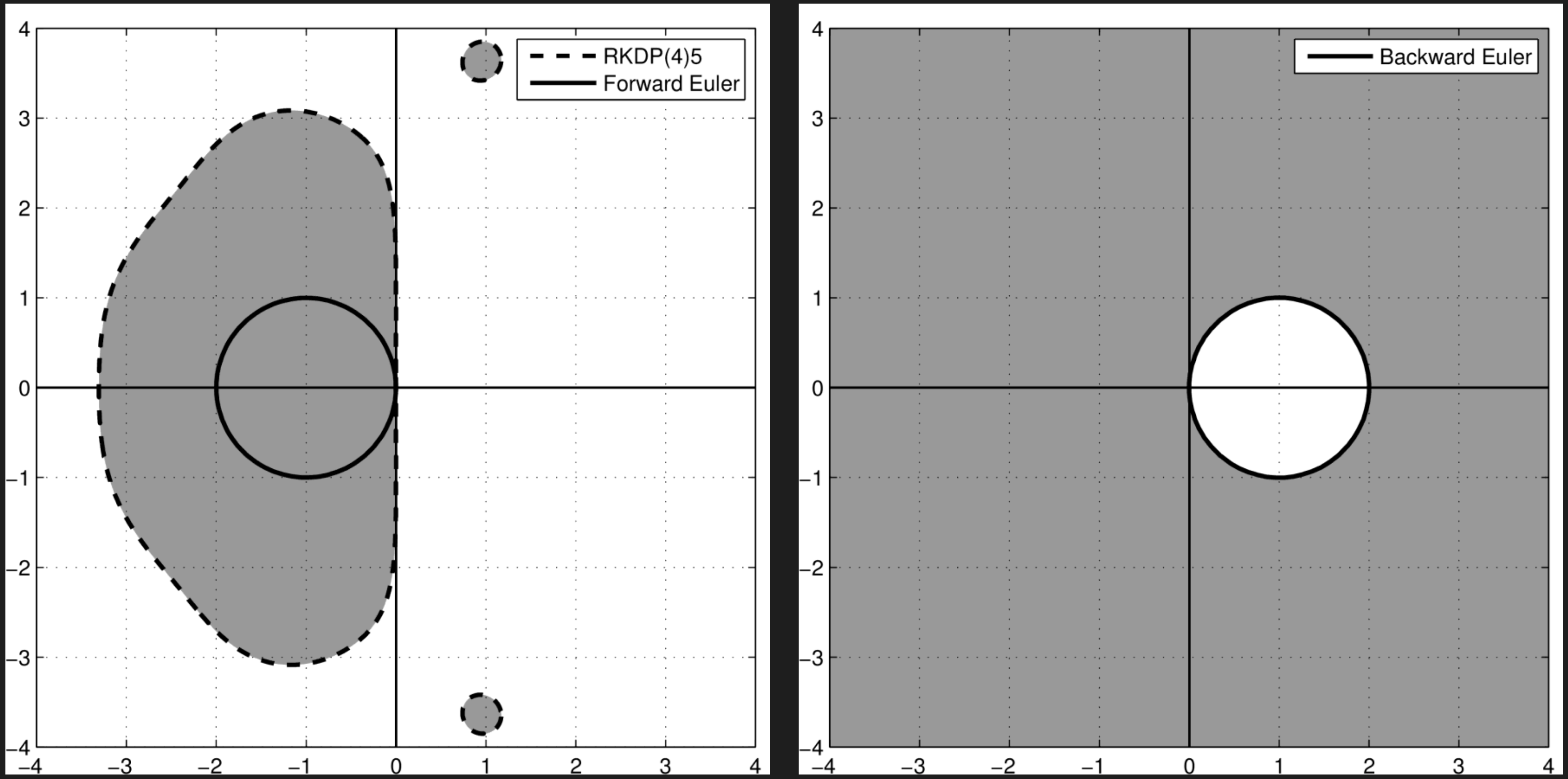
- Definition of a s-stage Runge-Kutta method
- Butcher tableau
- Explicit methods
  - Euler, RK4
- Embedded methods
  - RKDP(4)5
- Implicit Runge-Kutta?
  - BE, Radau..

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \sum_{i=1}^{s} b_i \boldsymbol{k}_i \,,$$

$$\boldsymbol{k}_i \equiv f\left(t_n + c_i h, \boldsymbol{y}_n + \sum_{j=1}^{s} a_{ij} \boldsymbol{k}_j \right).$$

$$\begin{array}{c|cccc}
c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\
c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\
\hline
 & b_1 & b_2 & \cdots & b_s \\
 & b_1^* & b_2^* & \cdots & b_s^*
\end{array}$$

Stability domains for various methods



BDF/NDF variable order methods

Define the backwards difference operator:

$$\nabla^0 y_n \equiv y_n,$$
$$\nabla^{j+1} y_n \equiv \nabla^j y_n - \nabla^j y_{n-1}.$$

The BDF formula or order k:

$$\sum_{j=1}^{k} \frac{1}{j} \nabla^j y_{n+1} = hf(t_{n+1}, y_{n+1})$$

The case $k = 1$ is the BE method.

`ndf15` is the method used in `CLASS`