Numerical solution of Ordinary Differential Equations

Thomas Tram

Institute of Gravitation and Cosmology

October 10, 2014

Not advanced!

Disclaimer!

Contrary to what the program states, this lecture will not be advanced!

A taste...

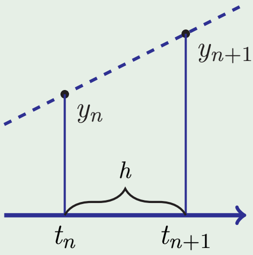
However, I will try to give you a taste for the art of solving differential equations numerically.

What is a differential equation?

General form

- A first order ODE can be written as $\mathbf{y}'(t) = f(t, \mathbf{y})$.
- We consider initial value problems: $\mathbf{y}(t_0) = \mathbf{y}_0$.

Derivatives



Forward Euler

- I can estimate the derivative at t_n by

$$\mathbf{y}'(t_n) \simeq \frac{\mathbf{y}_{n+1} - \mathbf{y}_n}{t_{n+1} - t_n}. \tag{1}$$

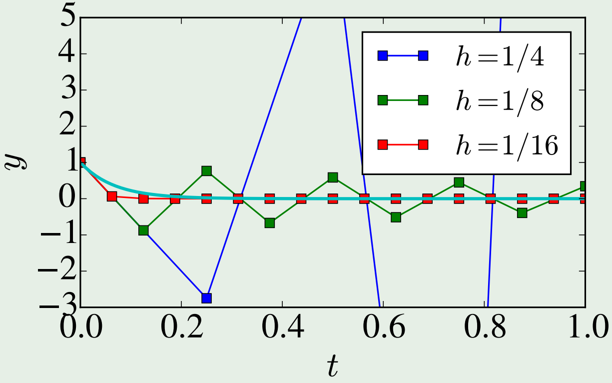
- This gives me the **forward Euler** method:

$$\mathbf{y}_{n+1} = \mathbf{y}_n + f(t_n, \mathbf{y}_n)h. \tag{2}$$

You should never use the forward Euler method

Instability of the forward Euler method

Consider a test equation $y' = -15y$ with the analytic solution $y(t) = y(0)\mathrm{e}^{-15t}$:



The problem of stiffness

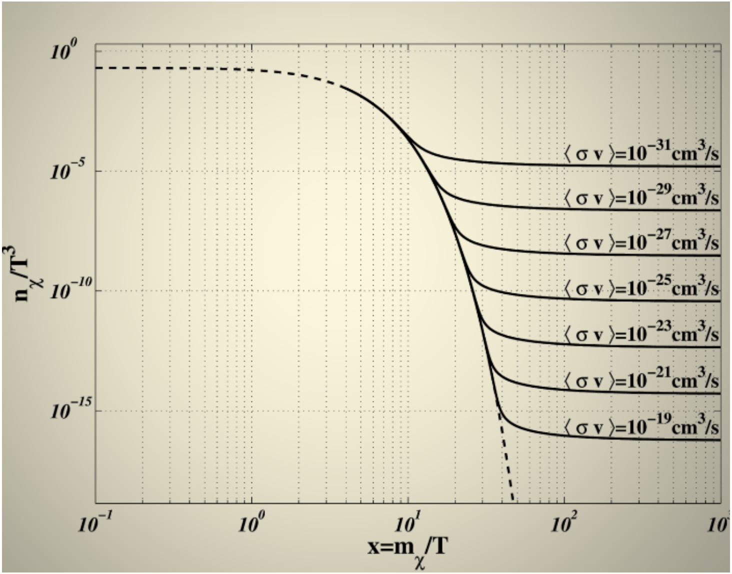
Different time scales

- the dynamic time scale is different from the time scale of interest.
- Cosmology: $\tau_{\text{int.}}$ vs. τ_{H_0}.
- Example from before: $\tau_{\text{int.}} = \frac{1}{15}$ vs. $[0, 1]$

Equilibrium

- a trivial equilibrium solution exists.
- Example from cosmology: Tight coupling limit
- In example from before: $y(t) \to 0$

Similar to WIMP freeze-out



Stability analysis of the forward Euler method

The test equation

Consider the test equation $y' = ay$ with the analytic solution $y(t) = y(0)\mathrm{e}^{at}$. Apply the **forward Euler** method to this equation:

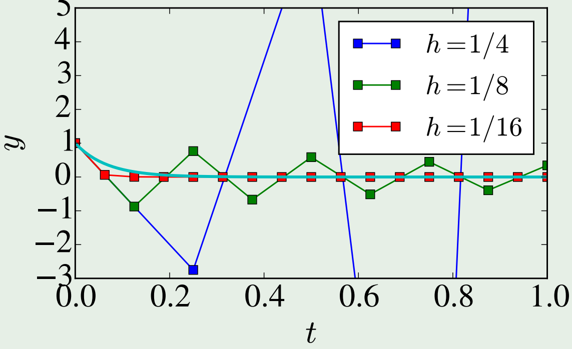
$$
\begin{aligned}
y_{n+1} &= y_n + f(t_n, y_n)h, \\
&= y_n + ay_n h, \\
&= (1 + ah)y_n.
\end{aligned}
$$

So the forwards Euler method will remain bounded **if and only** if $||1 + ah|| \leq 1$. Taking $a = -15$ requires $h \leq \frac{2}{15}$ for stability **at any time**. This is bad!

Forward Euler method exercise

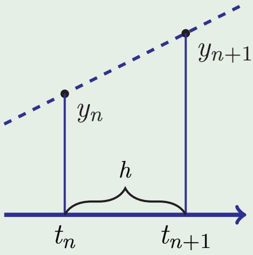
Forward Euler exercise!

Code your own forward Euler method in Python or C and reproduce this figure:



Backward Euler

Derivatives



Backward Euler

- I can estimate the derivative at t_{n+1} by

$$\mathbf{y}'(t_{n+1}) \simeq \frac{\mathbf{y}_{n+1} - \mathbf{y}_n}{t_{n+1} - t_n}. \tag{3}$$

- This gives me the **backward Euler** method:

$$\mathbf{y}_{n+1} = \mathbf{y}_n + f(t_{n+1}, \mathbf{y}_{n+1})h. \tag{4}$$

Implicit method

At each time-step, we must solve a system of non-linear algebraic equations!

Stability analysis of backwards Euler

The test equation revisited

Consider again the test equation $y' = ay$, but now apply the **backward Euler** method:

$$\begin{aligned} y_{n+1} &= y_n + f(t_{n+1}, y_{n+1})h, \\ &= y_n + a y_{n+1} h, \\ &= \frac{1}{1 - ah} y_n. \end{aligned}$$

Stability

If $\Re(a) < 0$ the solution is decaying. The **backwards Euler** method will remain bounded for **any** positive value of h!

Best method for perturbations?

Explicit methods

- **Easy** to code ODE-solver.
- **Fast** (well) after tight coupling.
- Stiffness must be **removed by hand** using TCA.
- **Not robust** against new physics.

Implicit methods

- **Fast** even without TCA.
- Very **robust** against users.
- Can be **slow** due to algebraic system.
- More **difficult** to code.

evolver_ndf15.c

Features of the primary ODE-solver in CLASS

`evolver_ndf15.c`: multistep extension of backwards Euler. Speed relies on

- Variable order 1-5.
- Adaptive step size.
- Interpolation of output values while keeping step size optimal.
- Recycling of Jacobians for Newtons method.
- Sparse LU decompositions.