Chapter 3: Numerical Solutions of Ordinary Differential Equations

Perla Mallouk

perla.mallouk@math.cnrs.fr

MAP5, Université Paris Cité

Numerical methods – General concepts

Consider any first-order ODE in explicit form, with an initial condition:

$$\begin{cases} y'(t) = f(t, y), \\ y(t_0) = y_0. \end{cases}$$

Numerical methods – General concepts

Consider any first-order ODE in explicit form, with an initial condition:

$$\begin{cases} y'(t) = f(t, y), \\ y(t_0) = y_0. \end{cases}$$

Main idea of the Euler scheme

Approximate $y'(t)$ by a difference quotient:

$$y'(t) \approx \frac{y(t + \Delta t) - y(t)}{\Delta t},$$

which yields

$$y(t + \Delta t) \approx y(t) + \Delta t \; f(t, y).$$

Numerical methods – The Forward Euler method

Set $t_n = t_0 + n\,\Delta t$.

Definition (Forward Euler)

$$\begin{cases} y_{n+1} = y_n + \Delta t\ f(t_n, y_n), \quad n = 0, 1, \dots \\ y_0 \text{ given by the initial condition.} \end{cases}$$

Numerical methods – The Forward Euler method

Set $t_n = t_0 + n\,\Delta t$.

Definition (Forward Euler)

$$\begin{cases} y_{n+1} = y_n + \Delta t\; f(t_n, y_n), \quad n = 0, 1, \dots \\ y_0 \text{ given by the initial condition.} \end{cases}$$

Consequence: starting from an initial position y_0 at time t_0, and fixing a time step $\Delta t > 0$, one can compute the approximations

$$y_1 \approx y(t_0 + \Delta t), \quad y_2 \approx y(t_0 + 2\,\Delta t), \quad \dots$$

iteratively and in an **explicit** manner.

Numerical methods – The Backward Euler method

Set $t_n = t_0 + n\,\Delta t$.

Definition (Backward Euler)

$$\begin{cases} y_{n+1} = y_n + \Delta t\ f(t_{n+1}, y_{n+1}), & n = 0, 1, \ldots \\ y_0 \text{ given by the initial condition.} \end{cases}$$

Numerical methods – The Backward Euler method

Set $t_n = t_0 + n\,\Delta t$.

Definition (Backward Euler)

$$\begin{cases} y_{n+1} = y_n + \Delta t\; f(t_{n+1}, y_{n+1}), & n = 0, 1, \dots \\ y_0 \text{ given by the initial condition.} \end{cases}$$

Consequence: starting from an initial position y_0 at time t_0, and fixing a time step $\Delta t > 0$, one can compute the approximations

$$y_1 \approx y(t_0 + \Delta t), \quad y_2 \approx y(t_0 + 2\,\Delta t), \quad \dots$$

iteratively, but only after solving an **implicit** equation.

Numerical methods – Stability

Definition (absolute stability)

For the test equation $y'(t) = -\lambda y(t)$, $\lambda > 0$ (whose exact solution $y(t) = y_0 e^{-\lambda t} \to 0$), a numerical scheme is **stable** for a given Δt if the approximations y_n also $\to 0$ as $n \to \infty$.

Numerical methods – Stability

Definition (absolute stability)

For the test equation $y'(t) = -\lambda y(t)$, $\lambda > 0$ (whose exact solution $y(t) = y_0 e^{-\lambda t} \to 0$), a numerical scheme is **stable** for a given Δt if the approximations y_n also $\to 0$ as $n \to \infty$.

Forward Euler: $y_{n+1} = (1 - \lambda \Delta t)\, y_n \;\Rightarrow\; y_n = (1 - \lambda \Delta t)^n y_0$.

$$\text{Stable} \iff |1 - \lambda \Delta t| < 1 \iff 0 < \Delta t < \frac{2}{\lambda}.$$

Numerical methods – Stability

Definition (absolute stability)

For the test equation $y'(t) = -\lambda y(t)$, $\lambda > 0$ (whose exact solution $y(t) = y_0 e^{-\lambda t} \to 0$), a numerical scheme is **stable** for a given Δt if the approximations y_n also $\to 0$ as $n \to \infty$.

Forward Euler: $y_{n+1} = (1 - \lambda \Delta t)\, y_n \;\Rightarrow\; y_n = (1 - \lambda \Delta t)^n y_0.$

$$\text{Stable} \iff |1 - \lambda \Delta t| < 1 \iff 0 < \Delta t < \frac{2}{\lambda}.$$

Backward Euler: $y_{n+1} = \dfrac{y_n}{1 + \lambda \Delta t} \;\Rightarrow\; y_n = \dfrac{y_0}{(1 + \lambda \Delta t)^n}.$

Always stable, for any $\Delta t > 0$.

Numerical methods – Stability

Definition (absolute stability)

For the test equation $y'(t) = -\lambda y(t)$, $\lambda > 0$ (whose exact solution $y(t) = y_0 e^{-\lambda t} \to 0$), a numerical scheme is **stable** for a given Δt if the approximations y_n also $\to 0$ as $n \to \infty$.

Forward Euler: $y_{n+1} = (1 - \lambda \Delta t)\, y_n \;\Rightarrow\; y_n = (1 - \lambda \Delta t)^n y_0.$

$$\text{Stable} \iff |1 - \lambda \Delta t| < 1 \iff 0 < \Delta t < \frac{2}{\lambda}.$$

Backward Euler: $y_{n+1} = \dfrac{y_n}{1 + \lambda \Delta t} \;\Rightarrow\; y_n = \dfrac{y_0}{(1 + \lambda \Delta t)^n}.$

Always stable, for any $\Delta t > 0$.

Summary: FE is only **conditionally stable** ($\Delta t < 2/\lambda$), while BE is **unconditionally stable**, but more costly since it's implicit.

Numerical methods – A simple example

Consider the following ODE:

$$\begin{cases} y'(t) = -\lambda y(t), \\ y(0) = 1. \end{cases}$$

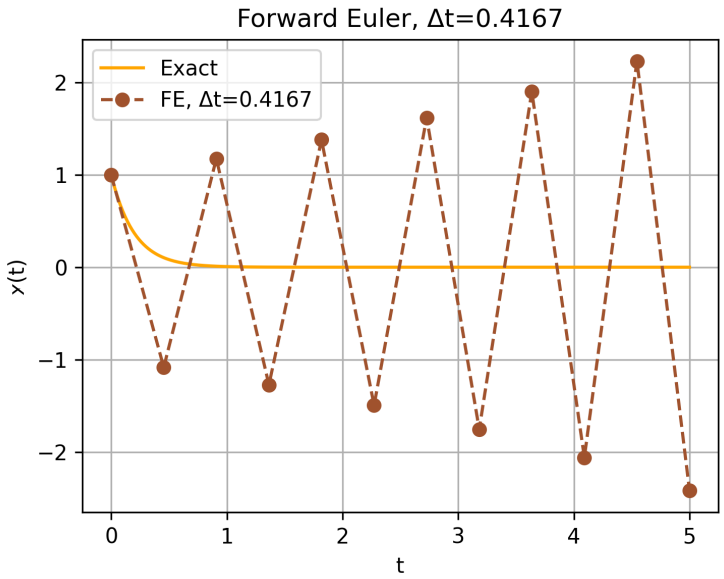
Numerical methods – A simple example

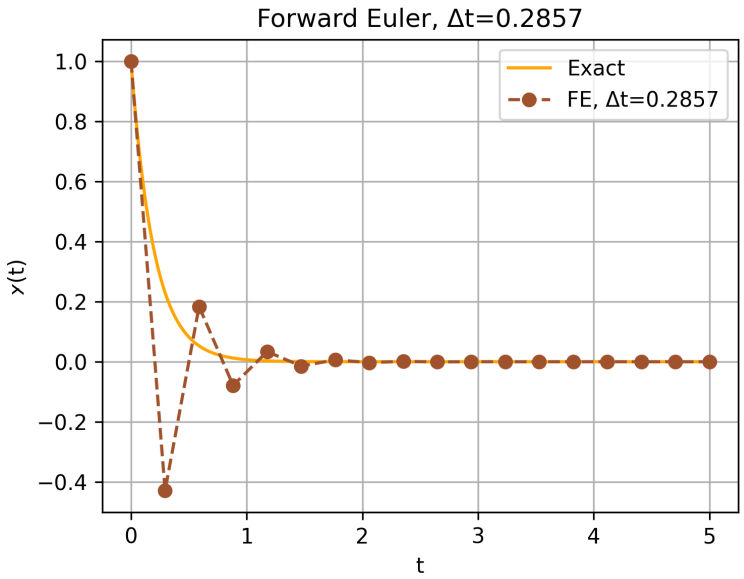
Consider the following ODE:

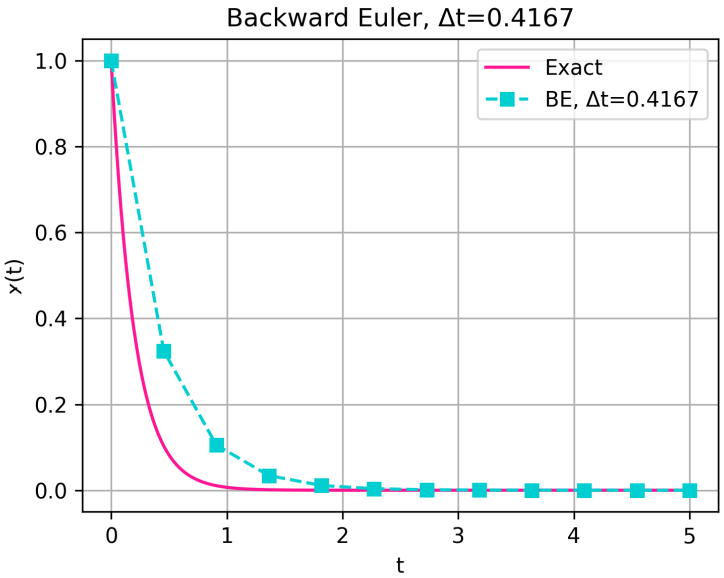
$$\begin{cases} y'(t) = -\lambda y(t), \\ y(0) = 1. \end{cases}$$

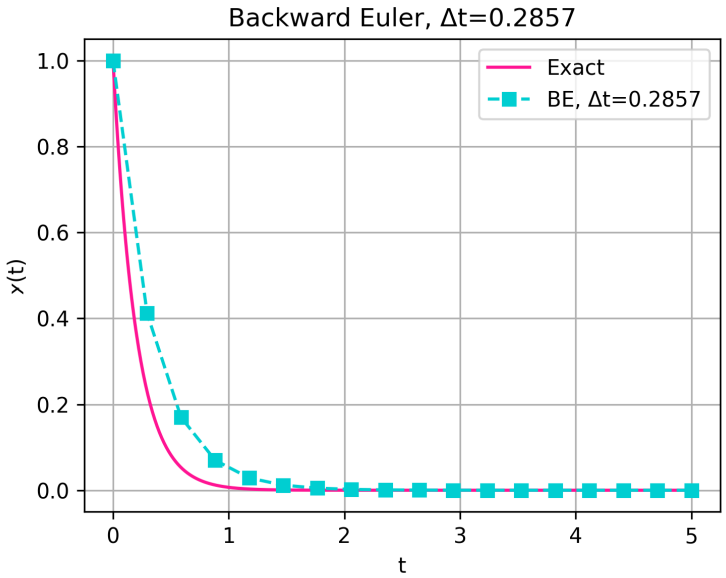
We will solve this equation with both methods, FE and BE, for $\lambda = 5$ and two different time steps: $\Delta t = 0.4167$ and $\Delta t = 0.2857$.

Numerical methods – A simple example









Numerical methods – Comments

Questions:

- What are the **similarities** and the **differences** between the previous examples?
- What are the **difficulties**?

Numerical methods – Comments

Questions:

- What are the **similarities** and the **differences** between the previous examples?
- What are the **difficulties**?

Remarks:

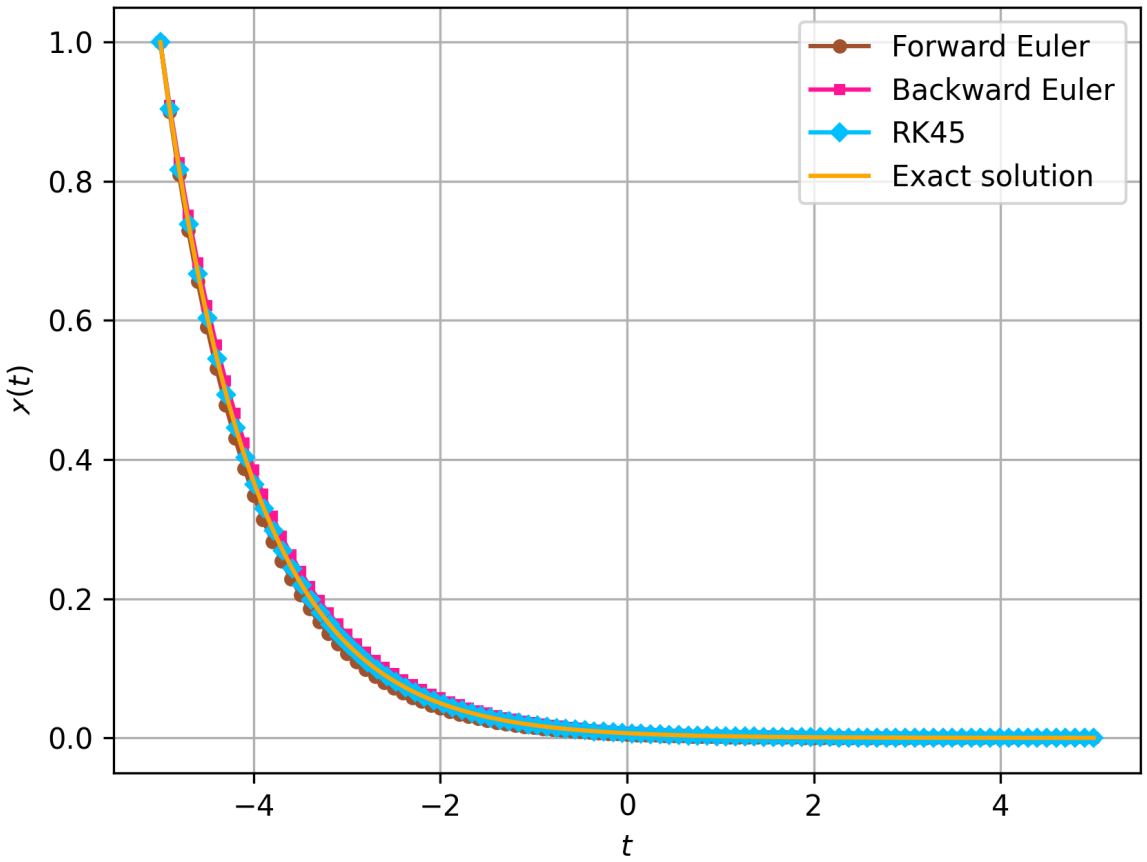
- **FE:** the numerical solution oscillates – need to decrease Δt;
  BE: no oscillations, better stability, but still not very accurate.
- **FE:** y_{n+1} depends exclusively on the value y_n previously computed;
  $\Rightarrow$ **explicit** method.
  BE: y_{n+1} depends also on itself through the value $f(t_{n+1}, y_{n+1})$;
  $\Rightarrow$ **implicit** method, more costly.
- To achieve a higher order of accuracy: use more sophisticated schemes, such as **the Runge–Kutta methods**.

Numerical methods – Practice

- Write a Python code using `NumPy` to solve numerically the equation $y' = -y$, with initial condition $y(0) = 1$, by performing 100 steps of the Euler scheme with stepsize $\Delta t = 0.1$. Draw the graph of this numerical solution. On the same graph, also plot the exact solution and compare.
- Use the function `solve_ivp` with `method='RK45'` to solve the same equation with the same initial condition, over the range $0 \leq t \leq 5$. On the same graph, draw this new numerical solution and comment.
- Use `solve_ivp` to solve numerically the equation $y' = t\cos(y)$ for the initial condition $y(0) = 1$ over the range $0 \leq t \leq 10$.
- Still for the equation $y' = t\cos(y)$, draw on the same graph the numerical solutions obtained with the different initial conditions $y(0) = 0$, $y(0) = 0.1$, $y(0) = 0.2$, ... $y(0) = 1$.

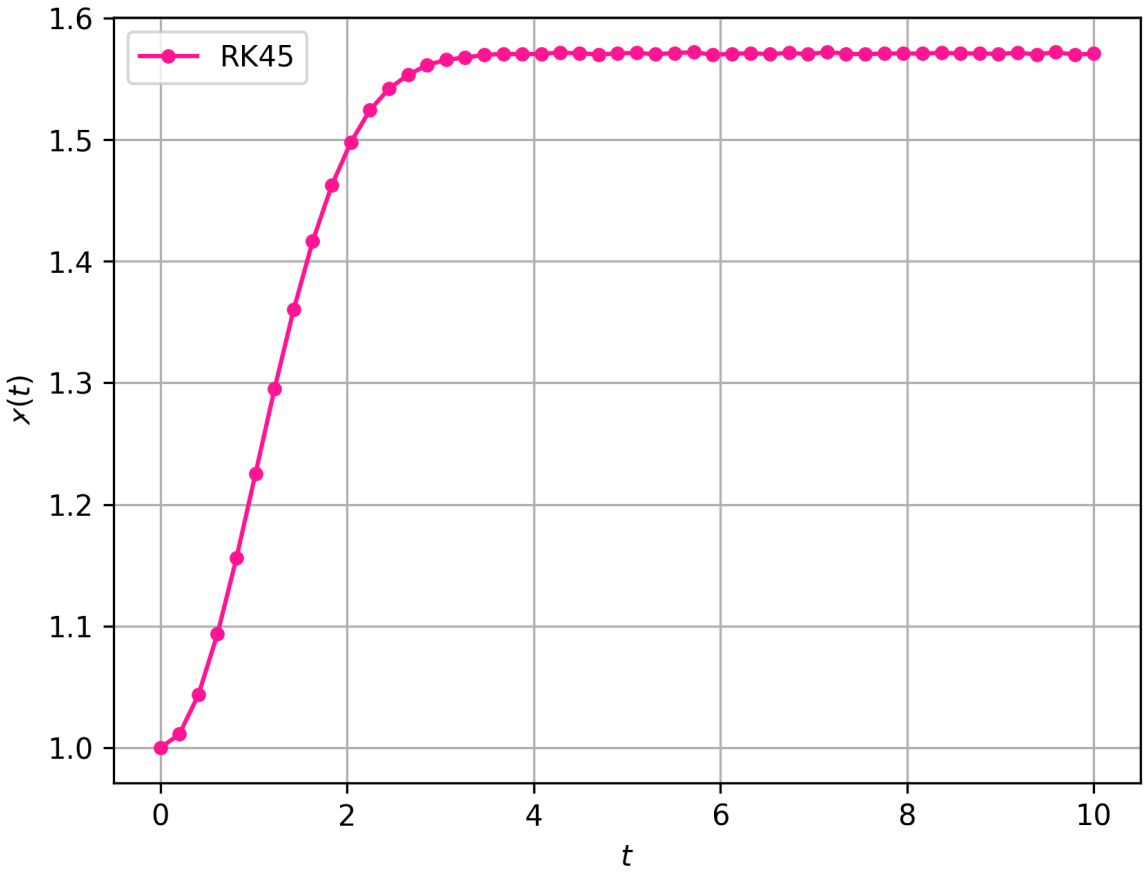
Numerical methods – Practice (correction)

- Solving numerically the equation $y' = -y$, with initial condition $y(0) = 1$: Forward Euler, Backward Euler and RK45 methods.



Numerical methods – Practice (correction)

- Solving numerically the equation $y' = t\cos(y)$, with initial condition $y(0) = 1$: RK45 method.



Numerical methods – Practice (correction)

- Solving numerically the equation $y' = t\cos(y)$, with different initial conditions: RK45 method.

