Ordinary Differential Equations

Laura Grigori

EPFL and PSI

slides based on lecture notes/slides from L. Dede/S. Deparis

November 26/December 3/10/17, 2025

EPFL

PAUL SCHERRER INSTITUT PSI

Plan

Plan

Numerical approximation of ODEs

Problem: Approximate numerically differential equations in the form:

$$F\left(y(t), \frac{dy(t)}{dt}, \frac{d^2y(t)}{dt^2}, \ldots, \frac{d^py(t)}{dt^p}\right) = 0,$$

with

- t an independent variable, often associated with the time variable,
- $y(t)$ the solution of the differential problem,
- p the order of the differential equation.

Focus on first-order problems for which $p = 1$, i.e.:

$$F\left(y(t), \frac{dy(t)}{dt}\right) = 0.$$

ODEs of order $p > 1$ can be recast into systems of ODEs of order $p = 1$.

Numerical approximation of ODEs

Problem: Approximate numerically differential equations in the form:

$$F\left(y(t), \frac{dy(t)}{dt}, \frac{d^2y(t)}{dt^2}, \ldots, \frac{d^py(t)}{dt^p}\right) = 0,$$

with

- t an independent variable, often associated with the time variable,
- $y(t)$ the solution of the differential problem,
- p the order of the differential equation.

Focus on first-order problems for which $p = 1$, i.e.:

$$F\left(y(t), \frac{dy(t)}{dt}\right) = 0.$$

ODEs of order $p > 1$ can be recast into systems of ODEs of order $p = 1$.

Numerical approximation of ODEs

Problem: Approximate numerically differential equations in the form:

$$F\left(y(t), \frac{dy(t)}{dt}, \frac{d^2y(t)}{dt^2}, \ldots, \frac{d^py(t)}{dt^p}\right) = 0,$$

with

- t an independent variable, often associated with the time variable,
- $y(t)$ the solution of the differential problem,
- p the order of the differential equation.

Focus on first-order problems for which $p = 1$, i.e.:

$$F\left(y(t), \frac{dy(t)}{dt}\right) = 0.$$

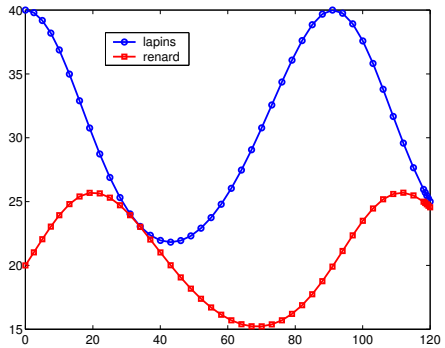
ODEs of order $p > 1$ can be recast into systems of ODEs of order $p = 1$.

Example and motivations

Consider two populations, y_1 and y_2, where y_1 are the prey and y_2 are the predators. The evolution of the two populations is described by the simultaneous differential equations

$$\begin{cases} y_1'(t) = C_1 y_1(t)\left[1 - b_1 y_1(t) - d_2 y_2(t)\right], \\ y_2'(t) = -C_2 y_2(t)\left[1 - b_2 y_2(t) - d_1 y_1(t)\right], \end{cases} \tag{1}$$

where C_1 and C_2 represent the growth rates of the two populations. The coefficients d_1 and d_2 govern the type of interaction between the two populations, while b_1 and b_2 are related to the available quantity of nutrients. The above equation are called the Lotka-Volterra equations.



The Cauchy problem

Definition (8.1)

Consider a continuous function $f : \mathbb{R}_+ \times \mathbb{R} \to \mathbb{R}$. For given $y_0 \in \mathbb{R}$, we search $y : t \in I \subset \mathbb{R}_+ \to y(t) \in \mathbb{R}$ that satisfies the following problem, called the *Cauchy problem*:

$$\begin{cases} y'(t) = f(t, y(t)) & \forall t \in I \\ y(t_0) = y_0 \end{cases} \tag{2}$$

where $y'(t) = \dfrac{dy(t)}{dt}$.

Model problem

The model problem is a Cauchy problem:

$$\begin{cases} \frac{dy}{dt}(t) = f(t, y(t)) & \forall t \in I \\ y(t_0) = y_0 \end{cases} \tag{3}$$

where $f(t, y) = \lambda y$ for some $\lambda \in \mathbb{R}$ and $\lambda < 0$.
Such a problem admits the solution:

$$y(t) = y_0 e^{\lambda(t - t_0)}, \text{ for all } t \in [t_0, t_f]$$

Often $I = (t_0, +\infty)$

Examples

- A Cauchy problem can be linear, such as:

$$\begin{cases} y'(t) = 3y(t) - 3t & \text{if } t > 0 \\ y(0) = 1 \end{cases} \tag{4}$$

with $f(t, v) = 3v - 3t$. The solution is $y(t) = (1 - 1/3)e^{3t} + t + 1/3$.

- We have also nonlinear problems, such as

$$\begin{cases} y'(t) = \sqrt[3]{y(t)} & \text{if } t > 0 \\ y(0) = 0 \end{cases} \tag{5}$$

with $f(t, v) = \sqrt[3]{v}$. This problem has got three following solutions : $y(t) = 0$, $y(t) = \sqrt{8t^3/27}$, $y(t) = -\sqrt{8t^3/27}$.

- For the following problem:

$$\begin{cases} y'(t) = 1 + y^2(t) & \text{if } t > 0 \\ y(0) = 0 \end{cases} \tag{6}$$

a solution is a function $y(t) = \tan(t)$ where $0 < t < \frac{\pi}{2}$, i.e. a local solution.

Well-posedness of the Cauchy problem

Theorem (Cauchy-Lipschitz, proposition 8.1 in the book)

If a function $f(t, y)$ is

1. *continuous with respect to both its arguments;*
2. *Lipschitz-continuous with respect to its second argument, that is, there exists a positive constant L (named Lipschitz constant) such that*

$$|f(t, y_1) - f(t, y_2)| \leq L|y_1 - y_2| \quad \forall y_1, y_2 \in \mathbb{R},\ \forall t \in I, \tag{7}$$

Then the solution $y = y(t)$ of the Cauchy problem (2) exists, is unique and belongs to $C^1(I)$.

Example

Consider a problem (4) and we check it exists a unique global solution. In this case $f(t, v) = 3v - 3t$ and we have:

$$|f(t, y_1) - f(t, y_2)| = |3y_1 - 3t - (3y_2 - 3t)| = |3y_1 - 3y_2| \leq 3|y_1 - y_2|$$

so

$$|f(t, y_1) - f(t, y_2)| \leq L|y_1 - y_2| \quad \forall y_1, y_2 \in \mathbb{R},\ \forall t > 0, \quad \text{where} \quad L = 3.$$

So f satisfies the assumptions of Theorem 1 and we can say that the problem (4) has got a unique global solution.

Well-posedness of the Cauchy problem

Remark: If the function $f(t, y) : I \times \mathbb{R} \to \mathbb{R}$ is C^1-continuous in the second argument y, then it is also Lipschitz continuous in the second argument. Indeed, we have:

$$|f(t, y_1) - f(t, y_2)| \leq \left(\max_{t \in I, y \in \mathbb{R}} \left| \frac{\partial f}{\partial y}(t, y) \right| \right) \cdot |y_1 - y_2|,$$

for which $L = \max_{t \in I, y \in \mathbb{R}} \left| \frac{\partial f}{\partial y}(t, y) \right|$.

Plan

Numerical differentiation

Let $y : [a, b] \to \mathbb{R}$ be C^1 and $t_n \in [a, b]$. The derivative $y'(t_n)$ is given by

$$\begin{aligned} y'(t_n) &= \lim_{h \to 0^+} \frac{y(t_n + h) - y(t_n)}{h}, \\ &= \lim_{h \to 0^+} \frac{y(t_n) - y(t_n - h)}{h}, \\ &= \lim_{h \to 0} \frac{y(t_n + h) - y(t_n - h)}{2h}. \end{aligned}$$

Let $t_0, t_1, \ldots, t_{N_h}$, $N_h + 1$ be equidistributed nodes at $[t_0, t_{N_h}]$. Let $h = (t_{N_h} - t_0)/N_h$ be the distance between two consecutive nodes. Let $(Dy)_n$ be an approximation of $y'(t_n)$. We say

- **Forward finite difference** if

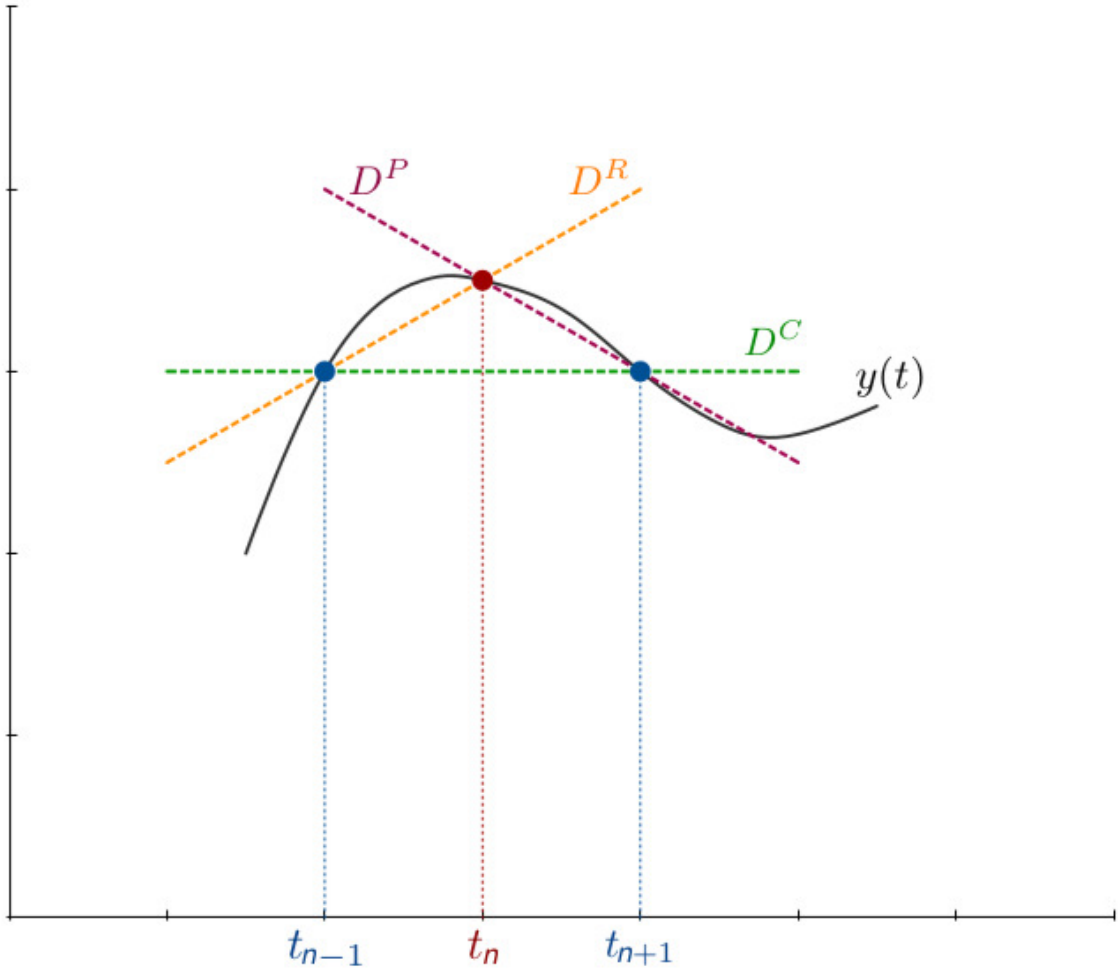
$$(Dy)_n^P = \frac{y(t_{n+1}) - y(t_n)}{h}, \qquad n = 0, \ldots, N_h - 1 \tag{8}$$

- **Backward finite difference** if

$$(Dy)_n^R = \frac{y(t_n) - y(t_{n-1})}{h}, \qquad n = 1, \ldots, N_h \tag{9}$$

- **Centered finite difference** if

$$(Dy)_n^C = \frac{y(t_{n+1}) - y(t_{n-1})}{2h}, \qquad n = 1, \ldots, N_h - 1 \tag{10}$$


D^P
D^R
D^C
$y(t)$
t_{n-1}
t_n
t_{n+1}

The error in the finite difference

Definition

The difference $\tau_n(h) = |y'(t_n) - (Dy)_n^P|$ is called truncation error in the point t_n. We say that τ_n is of order $p > 0$ if

$$\tau_n(h) \leq Ch^p,$$

for a positive constant C.

Thanks to the found estimation, the truncation error of the forward and the backward finite difference is of order 1; the truncation error of centered finite difference is of order 2.

Plan

The finite difference method

for approximating the Cauchy problem (Chapt. 8.2 in the book)

Let $\overline{I} = [t_0, t_f]$ be partitioned into N_h subintervals of equal size h,

$$t_0 < t_1 < \ldots < t_n < t_{n+1} < \ldots < t_f$$

where $h = t_{n+1} - t_n = \frac{t_f - t_0}{N_h}$ is the time step. We denote by

$$u_n \quad \text{an approximation of} \quad y(t_n).$$

In the Cauchy problem (2), for $t = t_n$, we have

$$y'(t_n) = f(t_n, y(t_n)).$$

We want to approximate the derivative $y'(t_n)$ in the point t_n. We can use a **finite difference differentiation**.

Forward Euler approximates $y'(t_n)$ with forward finite differences as $\frac{u_{n+1}-u_n}{h} = f(t_n, u_n)$

$$\begin{cases} u_{n+1} = u_n + hf(t_n, u_n) & \text{for } n = 0, 1, \ldots, N_h - 1 \\ u_0 = y_0 \end{cases} \tag{11}$$

Backward Euler approximates $y'(t_{n+1})$ with backward finite differences as $\frac{u_{n+1}-u_n}{h} = f(t_{n+1}, u_{n+1})$

$$\begin{cases} u_{n+1} = u_n + hf(t_{n+1}, u_{n+1}) & \text{for } n = 0, 1, \ldots, N_h - 1 \\ u_0 = y_0 \end{cases} \tag{12}$$

Remark

- *The forward Euler is explicit because u_{n+1} depends on u_n explicitly:*

$$\text{(forwardEuler)} \qquad u_{n+1} = u_n + hf(t_n, u_n).$$

- *The backward Euler is implicit because u_{n+1} is implicitly defined in terms of u_n:*

$$\text{(backwardEuler)} \qquad u_{n+1} = u_n + hf(t_{n+1}, u_{n+1}).$$

Backward Euler with Newton method

In general, for the backward Euler, we have to solve a nonlinear equation at each time step.
Fixed point iterations: Note that (backward Euler) is equivalent to a fixed point problem with

$$u_{n+1} = \phi(u_{n+1}) = u_n + hf(t_{n+1}, u_{n+1}) \tag{13}$$

We can solve this problem thanks to the following iterations

$$u_{n+1}^{k+1} = \phi(u_{n+1}^k), \quad k = 0, 1, 2, \ldots \tag{14}$$

The Newton method: Starting from the equation:

$$F(u_{n+1}) \equiv u_{n+1} - \phi(u_{n+1}) \equiv u_{n+1} - u_n - hf(t_{n+1}, u_{n+1}) = 0, \tag{15}$$

we use the following iterations:

$$u_{n+1}^{k+1} = u_{n+1}^k - \frac{F(u_{n+1}^k)}{F'(u_{n+1}^k)} = u_{n+1}^k - \frac{F(u_{n+1}^k)}{1 - \phi'(u_{n+1}^k)}, \quad k = 0, 1, 2, \ldots \tag{16}$$

In both cases, we have $\lim_{k \to \infty} u_{n+1}^k = u_{n+1}$.

Example

Consider the following differential equation

$$\begin{cases} y'(t) = -ty^2(t), & t > 0 \\ y(0) = 2. \end{cases} \tag{17}$$

We want to solve this equation using forward Euler and backward Euler methods, in the interval $[0, 4]$ with $N_h = 20$ subintervals (it is equivalent to a time step $h = 0.2$). We approximate the exact solution $y(t_n)$ at times $t_n = nh$, $n = 0, 1, \ldots 20$ (therefore $t_n = 0.2, 0.4, 0.6, \ldots$) by a numerical solution u_n.

In Matlab, the forward Euler method can be used by:

```
>> h = 0.2;                                    % the time step
>> u(1) = 2;                                   % the initial value
>> t = [0:h:4];                                % vector of time t(n)
>> for n=1:20;                                 % loop ‘‘for’’
     u(n+1) = u(n) + h * ( -t(n) * u(n)^2 );
   end;
>> plot(t,u);                                  % we draw the graph
```

We can also use the functions `feuler` and `beuler`:

- *Forward Euler*

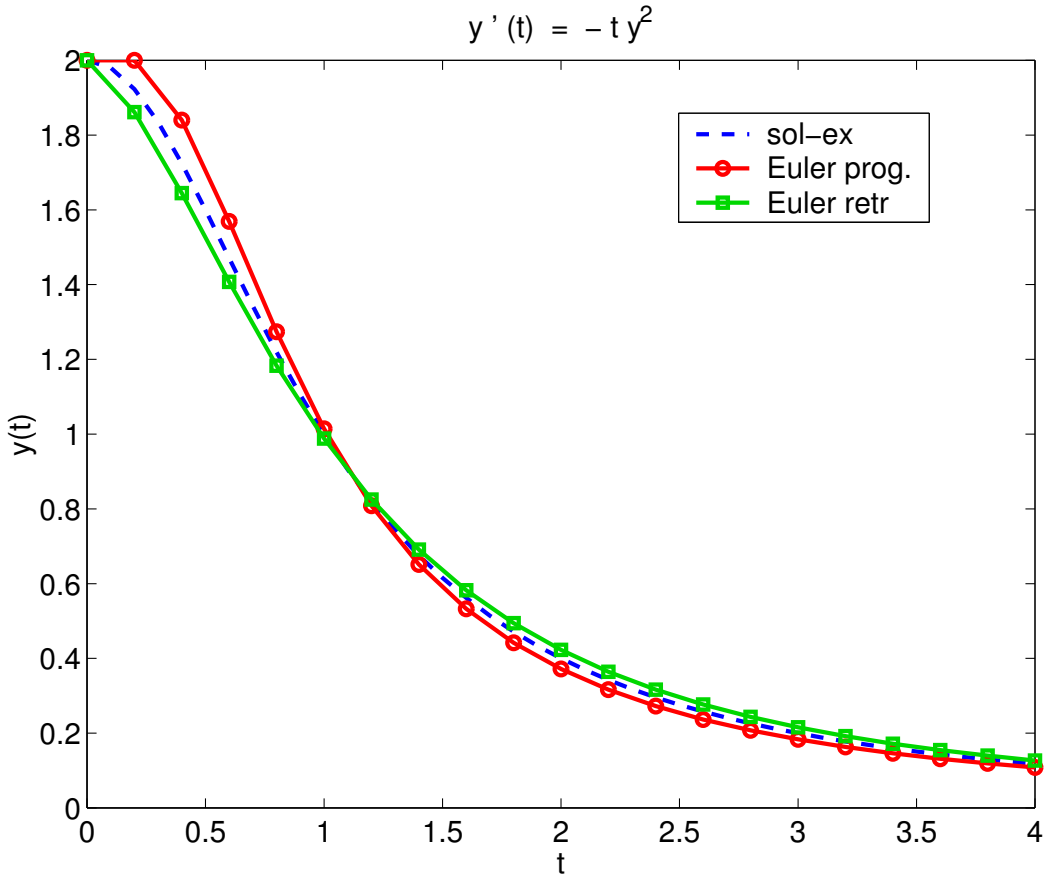
```
>> f = @(t,y) -t.*y.^2;
>> Nh = 20; tspan = [0 4]; y0 = 2;
>> [t_EP, y_EP] = feuler(f, tspan, y0, Nh);
```

  Output variables `t_EP` and `y_EP` contain sequences of the times t_n and the values u_n respectively.

- *Backward Euler*
  The function `beuler` uses the same syntax:

```
>> [t_ER, y_ER] = beuler(f, tspan, y0, Nh);
```

Comparison between the exact solution and those obtained by forward and backward Euler methods.



- Euler prog.: Forward Euler
- Euler retr.: Backward Euler

Crank-Nicolson method

Derived from the integral of the ODE using the trapezoidal rule, that is

$$\int_{t_n}^{t_{n+1}} \frac{dy}{dt} = \int_{t_n}^{t_{n+1}} f(t, y(t)) \implies y_{n+1} = y_n + \int_{t_n}^{t_{n+1}} f(t, y(t))$$

Crank-Nicolson method

$$\begin{cases} u_{n+1} = u_n + \dfrac{h}{2}\left[f(t_n, u_n) + f(t_{n+1}, u_{n+1})\right], & \text{for } n = 0, 1, \ldots, N_h - 1, \\ u_0 = y_0 \end{cases} \tag{18}$$

Crank–Nicolson method is an implicit method.

Crank-Nicolson method

Derived from the integral of the ODE using the trapezoidal rule, that is

$$\int_{t_n}^{t_{n+1}} \frac{dy}{dt} = \int_{t_n}^{t_{n+1}} f(t, y(t)) \implies y_{n+1} = y_n + \int_{t_n}^{t_{n+1}} f(t, y(t))$$

Crank-Nicolson method

$$\begin{cases} u_{n+1} = u_n + \dfrac{h}{2}\left[f(t_n, u_n) + f(t_{n+1}, u_{n+1})\right], & \text{for } n = 0, 1, \ldots, N_h - 1, \\ u_0 = y_0 \end{cases} \tag{18}$$

Crank–Nicolson method is an implicit method.

Crank-Nicolson method

Since Crank-Nicolson is implicit, it requires solving the nonlinear equation for each $n = 0, 1, \ldots, N_h - 1$:

$$\text{find } u_{n+1} : \; F_n^{\mathrm{CN}}(u_{n+1}) = 0 \quad \text{for all } n = 0, 1, \ldots, N_h - 1.$$

with $u_0 = y_0$, where

$$F_n^{\mathrm{CN}}(y) := y - u_n - \frac{h}{2}\left[f(t_n, u_n) + f(t_{n+1}, y)\right].$$

If Newton method is used, then one needs the first derivative of $F_n^{\mathrm{CN}}(y)$, which reads

$$(F_n^{\mathrm{CN}})'(y) = 1 - \frac{h}{2}\frac{\partial f}{\partial y}(t_{n+1}, y).$$

Heun method

Heun method is an explicit method which consists of two stages: compute u_{n+1}^* as in forward Euler and then compute u_{n+1} as in Crank-Nicolson but replacing u_{n+1} by u_{n+1}^*

$$\begin{cases} u_{n+1}^* = u_n + h\, f(t_n, u_n), \\ u_{n+1} = u_n + \frac{h}{2}\left[f(t_n, u_n) + f(t_{n+1}, u_{n+1}^*)\right], \text{ for } n = 0, 1, \ldots, N_h - 1 \\ u_0 = y_0 \end{cases} \tag{19}$$

Error analysis of the methods

Definition

The error associated with the numerical approximation of the Cauchy problem at t_n is $e_n := |y_n - u_n|$ for some $n = 0, 1, \ldots, N_h$. If one has:

$$e_n \leq Ch^p,$$

where $C > 0$ is independent of h, then the method has convergence order $p > 0$ (order of accuracy of the method).

- If the solution of the Cauchy problem is $y \in C^2(I)$, then the forward and backward Euler methods converge with order $p = 1$ in h.
- If the solution of the Cauchy problem is $y \in C^3(I)$, then the Crank–Nicolson and Heun methods converge with order $p = 2$ in h.

Error analysis of the methods

Definition

The error associated with the numerical approximation of the Cauchy problem at t_n is $e_n := |y_n - u_n|$ for some $n = 0, 1, \ldots, N_h$. If one has:

$$e_n \leq Ch^p,$$

where $C > 0$ is independent of h, then the method has convergence order $p > 0$ (order of accuracy of the method).

- If the solution of the Cauchy problem is $y \in C^2(I)$, then the forward and backward Euler methods converge with order $p = 1$ in h.
- If the solution of the Cauchy problem is $y \in C^3(I)$, then the Crank–Nicolson and Heun methods converge with order $p = 2$ in h.

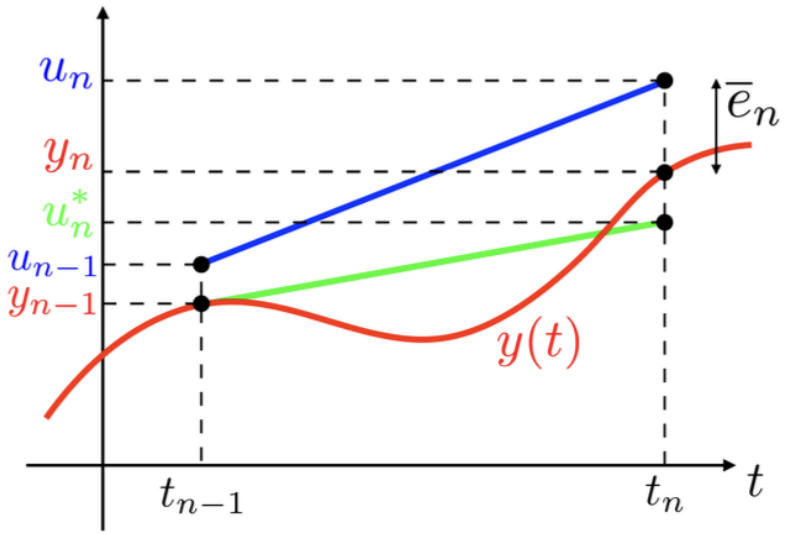
Error analysis of the methods

The error $\bar{e}_n = y_n - u_n = (y_n - u_n^*) + (u_n^* - u_n)$ depends on two contributions: Consider Forward Euler, computed from y_{n-1} and u_{n-1},

$$u_n^* = y_{n-1} + hf(t_{n-1}, y_{n-1})$$

$$u_n = u_{n-1} + hf(t_{n-1}, u_{n-1})$$

- $\tau_n(h) = (y_n - u_n^*)/h$ is called the local truncation error. It converges to 0 because the error of numerical differentiation converges to 0.
- $u_n - u_n^*$ represents propagation of the error from the previous step. We can prove it converges to 0 using that f is Lipschitz continuous.



Plan

Stability conditions

The choice of time step h is not arbitrary. For forward Euler, we will see later that if h is not small enough then stability problems may arise.

For example, if we consider the problem

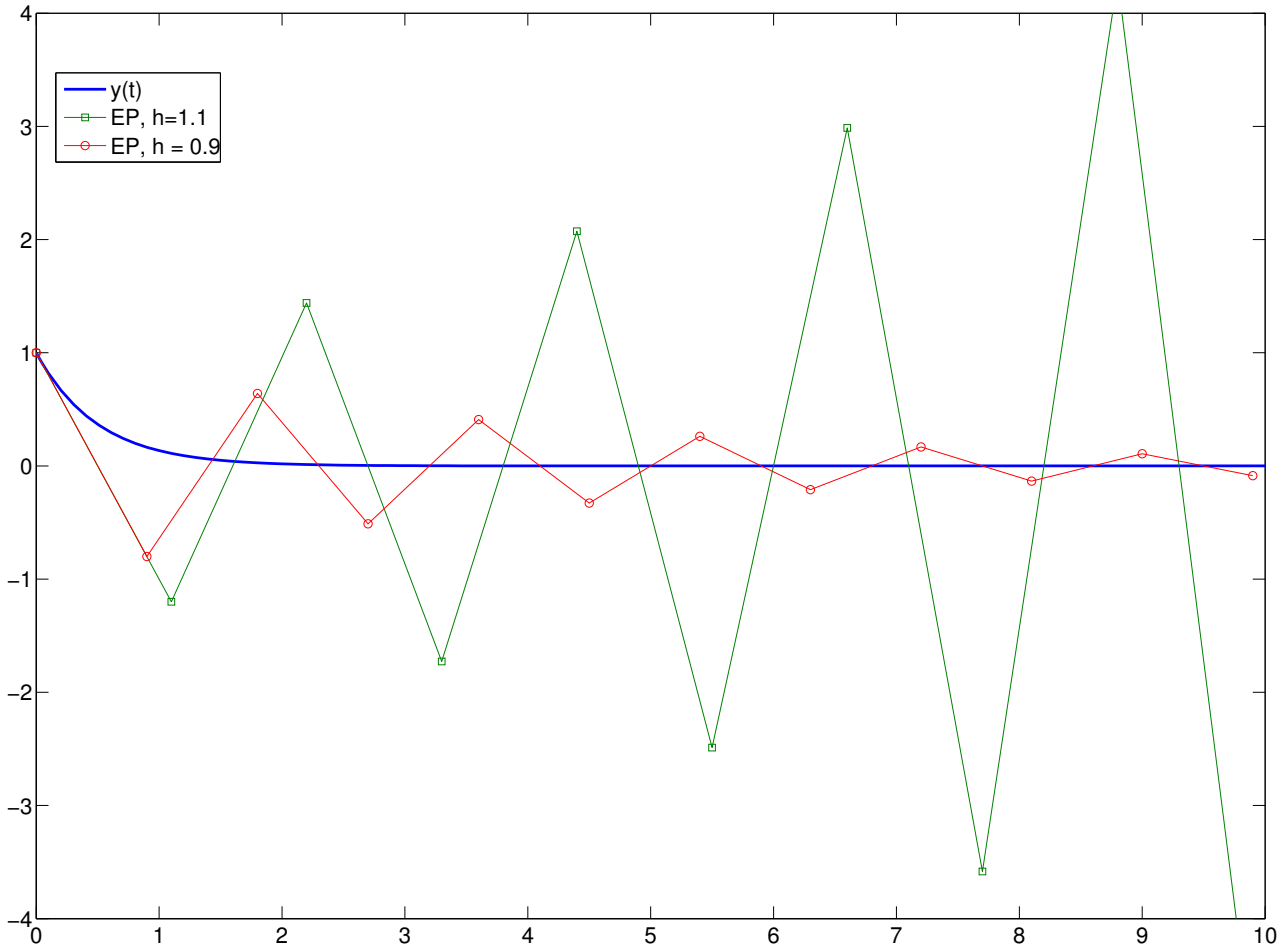
$$\begin{cases} y'(t) = -2y(t) & \text{for } t \in \mathbb{R}_+ \\ y(0) = 1, \end{cases} \tag{20}$$

then the solution is

$$y(t) = e^{-2t},$$

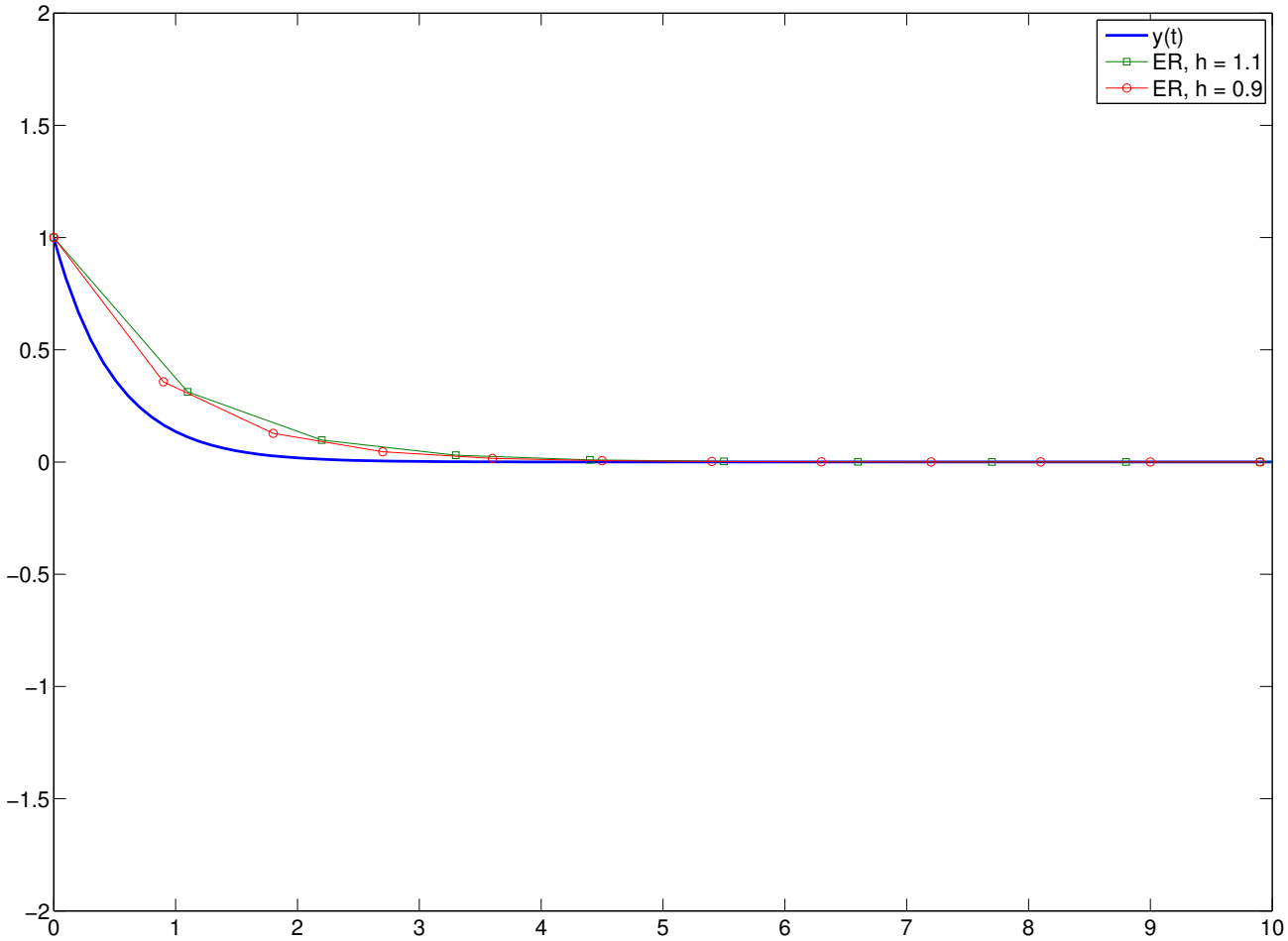
We can observe that behavior with respect to h of forward and backward Euler methods are very different.

Stability conditions (forward Euler)



EP stands for Forward Euler

Stability conditions (backward Euler)



ER stands for Backward Euler

Stability of the numerical methods: zero-stability

(Chapt. 8.2.6 in the book)

Zero-stability: Property of a method to control the propagation of numerical perturbations for bounded intervals I s.t. $|I| < +\infty$ which are relatively small.

Example with Forward Euler:

$$\begin{cases} u_{n+1} = u_n + hf(t_n, u_n) & \text{for } n = 0, 1, \ldots, N_h - 1 \\ u_0 = y_0 \end{cases} \tag{21}$$

Perturbed system

$$\begin{cases} z_{n+1} = z_n + h(f(t_n, z_n) + \rho_n) & \text{for } n = 0, 1, \ldots, N_h - 1 \\ z_0 = y_0 + \rho \end{cases} \tag{22}$$

Zero-stability

Definition (8.3.)

A numerical method for the approximation of ODEs is *zero-stable* if

$$\exists h_0 > 0, C > 0, \varepsilon_0 > 0 \text{ s.t., } \forall h \in (0, h_0] \text{ and } \forall \varepsilon \in (0, \varepsilon_0],$$

$$\text{if } |\rho_n| \leq \varepsilon \ \forall n = 0, \ldots, N_h, \text{ then } |z_n - u_n| \leq C\varepsilon \ \forall n = 0, \ldots, N_h, \text{ where}$$

- ρ_n is size of the perturbation introduced at the step t_n,
- z_n is solution that would be obtained by applying the numerical method to a perturbed ODE,
- C is constant independent of h but dependent on $|I|$,
- ε is the maximum size of the perturbation.

Based on Lax–Richtmeyer equivalence Theorem 1.1, a consistent method for the approximation of ODEs is convergent if and only if is zero–stable.
All previous methods are consistent, convergent and zero-stable.

Zero-stability

Definition (8.3.)

A numerical method for the approximation of ODEs is *zero-stable* if

$$\exists h_0 > 0, C > 0, \varepsilon_0 > 0 \text{ s.t., } \forall h \in (0, h_0] \text{ and } \forall \varepsilon \in (0, \varepsilon_0],$$

$$\text{if } |\rho_n| \leq \varepsilon \ \forall n = 0, \ldots, N_h, \text{ then } |z_n - u_n| \leq C\varepsilon \ \forall n = 0, \ldots, N_h, \text{ where}$$

- ρ_n is size of the perturbation introduced at the step t_n,
- z_n is solution that would be obtained by applying the numerical method to a perturbed ODE,
- C is constant independent of h but dependent on $|I|$,
- ε is the maximum size of the perturbation.

Based on Lax–Richtmeyer equivalence Theorem 1.1, a consistent method for the approximation of ODEs is convergent if and only if is zero–stable.
All previous methods are consistent, convergent and zero-stable.

Zero-stability

Definition (8.3.)

A numerical method for the approximation of ODEs is *zero-stable* if

$$\exists h_0 > 0, C > 0, \varepsilon_0 > 0 \text{ s.t., } \forall h \in (0, h_0] \text{ and } \forall \varepsilon \in (0, \varepsilon_0],$$

$$\text{if } |\rho_n| \leq \varepsilon \ \forall n = 0, \ldots, N_h, \text{ then } |z_n - u_n| \leq C\varepsilon \ \forall n = 0, \ldots, N_h, \text{ where}$$

- ρ_n is size of the perturbation introduced at the step t_n,
- z_n is solution that would be obtained by applying the numerical method to a perturbed ODE,
- C is constant independent of h but dependent on $|I|$,
- ε is the maximum size of the perturbation.

Based on Lax–Richtmeyer equivalence Theorem 1.1, a consistent method for the approximation of ODEs is convergent if and only if is zero–stable.
All previous methods are consistent, convergent and zero-stable.

The absolute stability (on unbounded intervals)

For given $\lambda \in \mathbb{R}$, $\lambda < 0$, we consider the model problem:

$$\begin{cases} y'(t) = \lambda y(t) & \text{for } t \in \mathbb{R}_+ \\ y(t_0) = y_0 \end{cases} \tag{23}$$

The solution is

$$y(t) = y_0 e^{\lambda(t-t_0)}. \qquad \text{In particular,} \quad \lim_{t\to\infty} y(t) = 0.$$

Let $t_0 < t_1 < \ldots < t_n < t_{n+1} < \ldots$ such that $t_n = nh$ and where the *time step* $h > 0$ is fixed.

We say that a numerical scheme associated to the model problem is absolutely stable if $\lim_{n\to\infty} u_n = 0$.

- *unconditionally* if $\lim_{n\to\infty} u_n = 0$ for all $h > 0$,
- *conditional* if
  $\lim_{n\to\infty} u_n = 0$ for all $h > 0$ such that $h < h_{\max}$, for some $h_{\max} > 0$.

The absolute stability (on unbounded intervals)

For given $\lambda \in \mathbb{R}$, $\lambda < 0$, we consider the model problem:

$$\begin{cases} y'(t) = \lambda y(t) & \text{for } t \in \mathbb{R}_+ \\ y(t_0) = y_0 \end{cases} \tag{23}$$

The solution is

$$y(t) = y_0 e^{\lambda(t-t_0)}. \qquad \text{In particular,} \quad \lim_{t\to\infty} y(t) = 0.$$

Let $t_0 < t_1 < \ldots < t_n < t_{n+1} < \ldots$ such that $t_n = nh$ and where the *time step* $h > 0$ is fixed.

We say that a numerical scheme associated to the model problem is absolutely stable if $\lim_{n\to\infty} u_n = 0$.

- *unconditionally* if $\lim_{n\to\infty} u_n = 0$ for all $h > 0$,
- *conditional* if
  $\lim_{n\to\infty} u_n = 0$ for all $h > 0$ such that $h < h_{\max}$, for some $h_{\max} > 0$.

The absolute stability (on unbounded intervals)

For given $\lambda \in \mathbb{R}$, $\lambda < 0$, we consider the model problem:

$$\begin{cases} y'(t) = \lambda y(t) & \text{for } t \in \mathbb{R}_+ \\ y(t_0) = y_0 \end{cases} \tag{23}$$

The solution is

$$y(t) = y_0 e^{\lambda(t-t_0)}. \qquad \text{In particular,} \quad \lim_{t\to\infty} y(t) = 0.$$

Let $t_0 < t_1 < \ldots < t_n < t_{n+1} < \ldots$ such that $t_n = nh$ and where the *time step* $h > 0$ is fixed.

We say that a numerical scheme associated to the model problem is absolutely stable if $\lim_{n\to\infty} u_n = 0$.

- *unconditionally* if $\lim_{n\to\infty} u_n = 0$ for all $h > 0$,
- *conditional* if
  $\lim_{n\to\infty} u_n = 0$ for all $h > 0$ such that $h < h_{\max}$, for some $h_{\max} > 0$.

Stability function

Consider our model problem:

$$\begin{cases} y'(t) = \lambda y(t) & \text{for } t \in \mathbb{R}_+, \lambda \in \mathbb{R}, \lambda < 0 \\ y(t_0) = y_0 \end{cases} \tag{24}$$

Definition (8.5)

The stability function associated with a numerical method is the complex function $R(z) : \mathbb{C} \to \mathbb{C}$ such that, when applied to the model problem we have:

$$u_n = R(h\lambda)u_{n-1} = [R(h\lambda)]^n y_0 \quad \text{for } n = 0, 1, \ldots$$

Proposition

A method is absolutely stable if and only if $|R(h\lambda)| < 1$.

Stability function

Consider our model problem:

$$\begin{cases} y'(t) = \lambda y(t) & \text{for } t \in \mathbb{R}_+, \lambda \in \mathbb{R}, \lambda < 0 \\ y(t_0) = y_0 \end{cases} \tag{24}$$

Definition (8.5)

The stability function associated with a numerical method is the complex function $R(z) : \mathbb{C} \to \mathbb{C}$ such that, when applied to the model problem we have:

$$u_n = R(h\lambda)u_{n-1} = [R(h\lambda)]^n y_0 \quad \text{for } n = 0, 1, \ldots$$

Proposition

A method is absolutely stable if and only if $|R(h\lambda)| < 1$.

Forward Euler for the model problem

- For the **forward Euler**:

$$u_{n+1} = (1 + \lambda h)u_n, \qquad \text{where} \quad u_n = (1 + \lambda h)^n y_0, \quad \forall n \geq 0. \tag{25}$$

Stability function:

$$R^{FE}(z) = 1 + z$$

If $1 + \lambda h < -1$, then $|u_n| \to \infty$ when $n \to \infty$, therefore forward Euler is *unstable*.
To ensure stability, we need to *limit the time step* h, by imposing the **stability condition** :

$$\boxed{|1 + \lambda h| < 1 \text{ hence } 0 < h < 2/|\lambda|.}$$

Backward Euler

- For the **backward Euler**:

$$u_{n+1} = \left(\frac{1}{1-\lambda h}\right) u_n \qquad \text{and therefore} \qquad u_n = \left(\frac{1}{1-\lambda h}\right)^n y_0, \quad \forall n \geq 0.$$

Stability function:

$$R^{BE}(z) = \frac{1}{1-z}$$

Because $\lim_{n\to\infty} u_n = 0$, it is unconditionally stable (it is stable for any $h > 0$).

Crank-Nicolson

- For the **Crank-Nicolson**:

$$u_{n+1} = u_n + \frac{h}{2}(\lambda u_n + \lambda u_{n+1})$$

$$u_{n+1} = \left(\frac{1 + (h\lambda)/2}{1 - (h\lambda)/2}\right) u_n \qquad \text{and therefore} \qquad u_n = \left(\frac{1 + (h\lambda)/2}{1 - (h\lambda)/2}\right)^n y_0, \quad \forall n \geq 0.$$

Stability function:

$$R^{CN}(z) = \frac{1 + z/2}{1 - z/2}$$

Because $h > 0, \lambda < 0$ then $|R^{CN}(h\lambda)| < 1$ and $\lim_{n\to\infty} u_n = 0$, it is unconditionally stable (it is stable for any $h > 0$).

Heun method

- For the **Heun method**:

$$\begin{aligned} u_{n+1} &= u_n + \frac{h}{2} \cdot \left(f(t_n, u_n) + f(t_{n+1}, u^*_{n+1}) \right) \\ &= u_n + \frac{h}{2} \cdot \left(\lambda u_n + \lambda u^*_{n+1} \right) \right) = u_n + \frac{h}{2} (\lambda u_n + \lambda (u_n + h\lambda u_n)) \end{aligned}$$

$$u_{n+1} = \left(1 + h\lambda + \frac{(h\lambda)^2}{2} \right) u_n$$

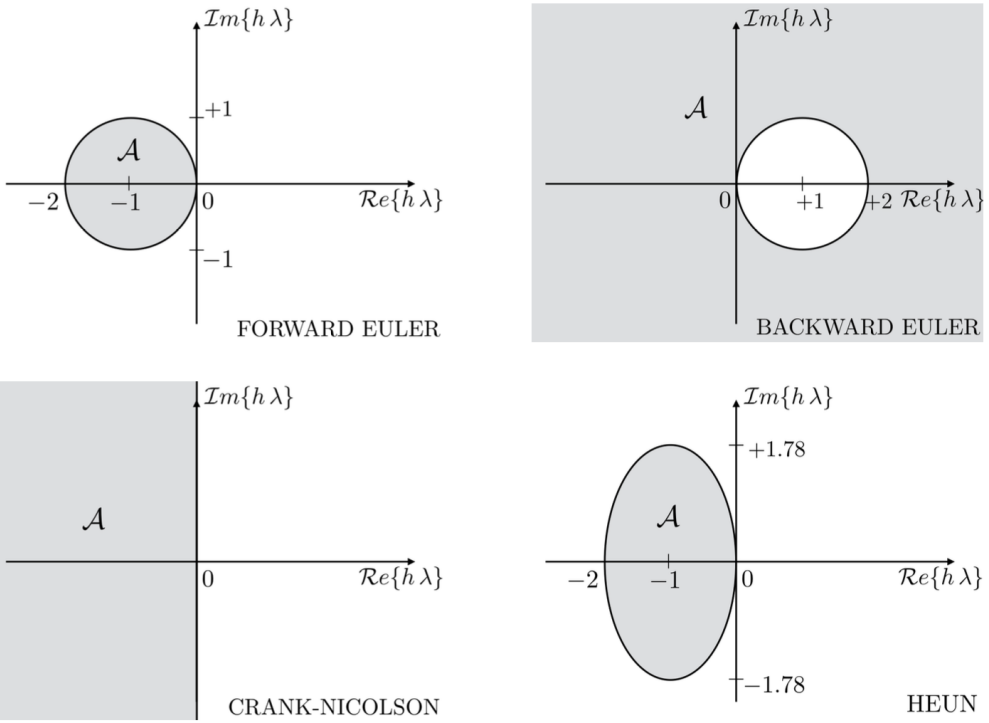
Stability function:

$$R^H(z) = 1 + z + \frac{z^2}{2}$$

Setting $|R^H(h\lambda)| < 1$ we obtain Heun is conditionally absolutely stable if

$$0 < h < h_{max} \text{ with } h_{max} = \frac{2}{|\lambda|}$$

Region of absolute stability



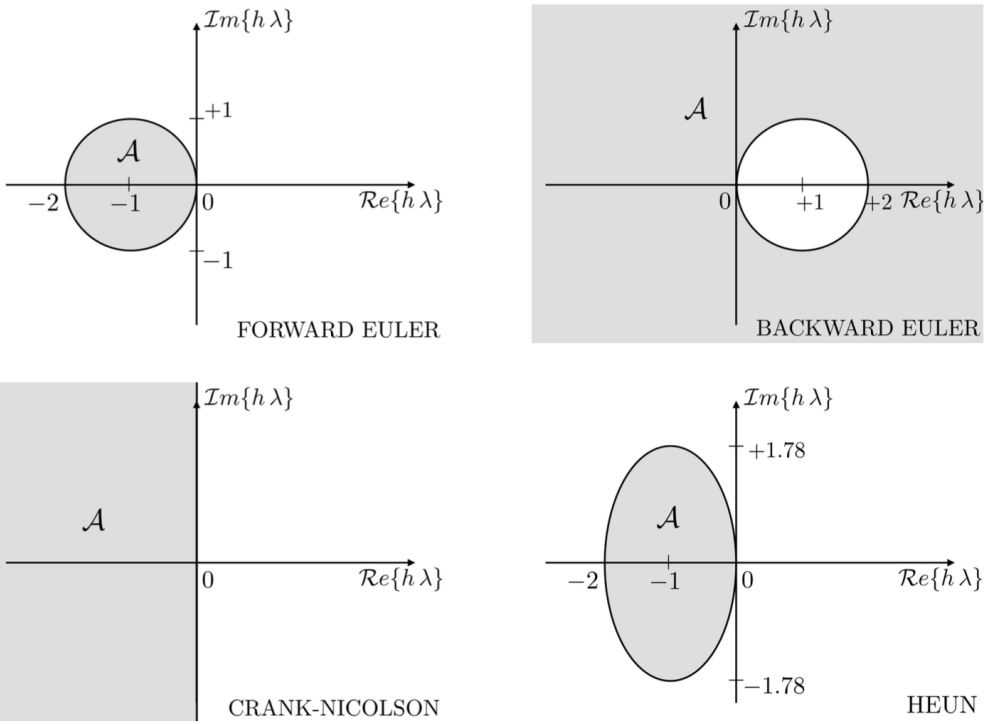
Definition

The region of absolute stability of a numerical method applied to the model problem is the set in the complex plane

$$\mathcal{A} := \{z \in \mathbb{C} : |R(z)| < 1\},$$

where $R(z) : \mathbb{C} \to \mathbb{C}$ is the stability polynomial.

Region of absolute stability



Definition

A numerical method is *A-stable* if it is unconditionally absolutely stable for the model problem for all $\lambda \in \mathbb{C}$ such that $\mathrm{Re}(\lambda) < 0$.

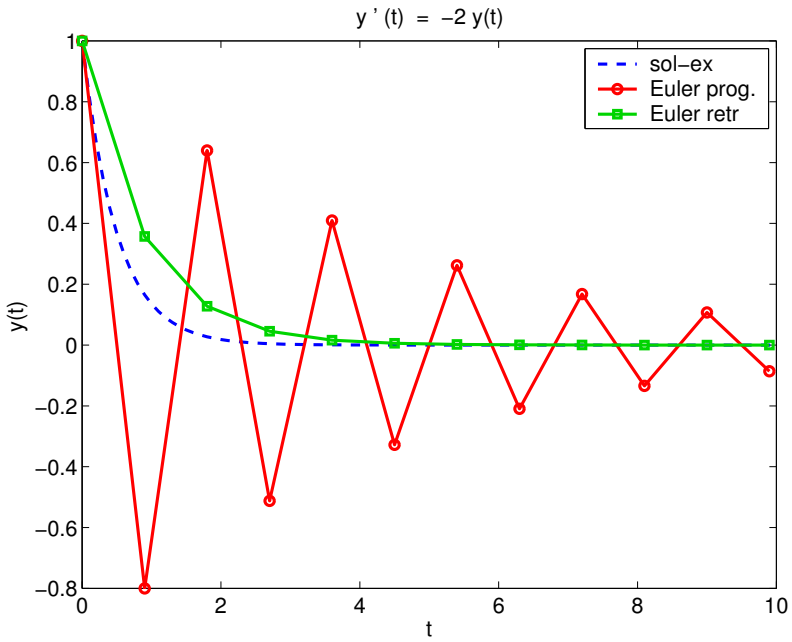
- Backward Euler and Crank-Nicolson methods are *A-stable*.
- Forward Euler and Heun methods are not *A-stable*.

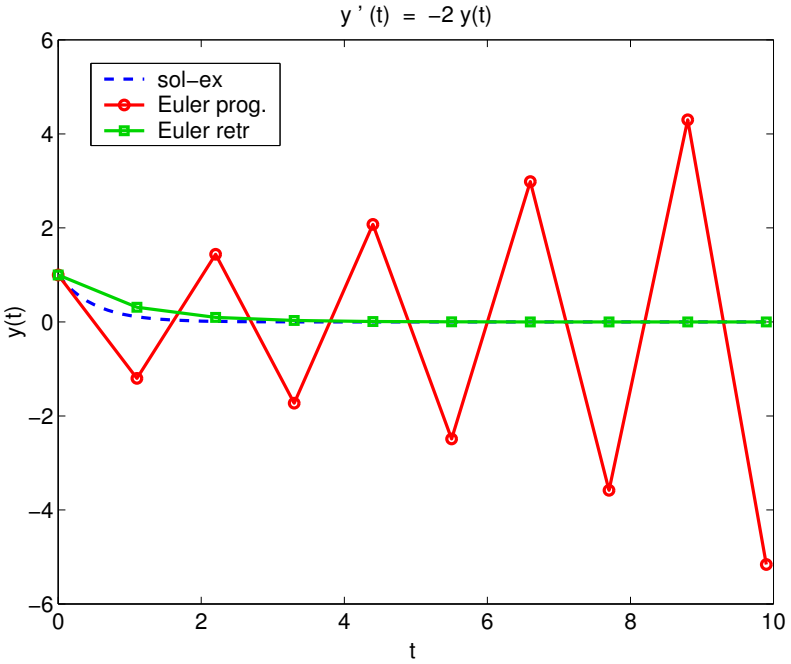
Example

Let's solve the problem (20) for $\lambda = -2$ and $y_0 = 1$ at interval $[0, 10]$ using forward and backward Euler methods with $h = 0.9$ and $h = 1.1$. Here are the Matlab/Octave commands for the case $h = 0.9$. Note that, even if $f(t, y)$ does not depend on t, it must be defined in Matlab/Octave as a function of (t, y).

```
>> f = @(t,x) -2*x; h=0.9; tspan=[0 10]; Nh = 10/h; y0=1;
>> [t_ep, y_ep] = feuler(f, tspan, y0, Nh);
>> [t_er, y_er] = beuler(f, tspan, y0, Nh);
>> t = linspace(0, 10, 11); sol_ex = @(t) exp(-2*t);
>> plot(t, sol_ex(t), 'b', t_ep, y_ep, 'ro-', t_er, y_er', 'go-')
```

The following figure shows obtained solutions for $h = 0.9$ (on the left) and $h = 1.1$ (on the right) and the exact solution.





Example

Comparison of solutions that we obtain by the forward and backward Euler methods for $h = 0.9$ (on the left, stable) and $h = 1.1$ (on the right, unstable) (stability condition for forward Euler: $|\lambda| = 2 \Rightarrow h < 2/|\lambda| = 1$).

Plan

Runge-Kutta methods

- One-step methods for the numerical approximation of ODEs
- The approximate solution u_{n+1} is determined by evaluating $f(t, y)$ at s stages in the interval $[t_n, t_{n+1}]$.
- The general Runge-Kutta method for approximating the Cauchy problem is:

$$\begin{cases} u_{n+1} = u_n + h \sum_{i=1}^{s} b_i K_i, \quad n = 0, 1, \ldots, N_h - 1, \\ u_0 = y_0, \end{cases}$$

where

$$K_i := f\left(t_n + c_i h, u_n + h \sum_{j=1}^{s} a_{ij} K_j\right), \quad i = 1, \ldots, s, \tag{8.9}$$

for some coefficients $\mathbf{c} = (c_1, \ldots, c_s)^T \in \mathbb{R}^s$, $\mathbf{b} = (b_1, \ldots, b_s)^T \in \mathbb{R}^s$, and $A \in \mathbb{R}^{s \times s}$, with $(A)_{ij} = a_{ij}$ for $i, j = 1, \ldots, s$.

Runge-Kutta and Butcher's array

For a given Runge-Kutta method,

$$\begin{cases} u_{n+1} = u_n + h\sum_{i=1}^{s} b_i K_i, \quad n = 0, 1, \ldots, N_h - 1, \\ u_0 = y_0, \end{cases} \tag{26}$$

where

$$K_i := f\left(t_n + c_i h, u_n + h\sum_{j=1}^{s} a_{ij} K_j\right), \quad i = 1, \ldots, s, \tag{8.9}$$

the coefficients are stored in the so-called Butcher's array as:

$$\begin{array}{c|c} c & A \\ \hline & \mathbf{b}^T \end{array}$$

- If A is stored from the bottom-left "corner," the Runge-Kutta method is explicit, if $a_{ij} = 0$ for $j \geq i$ for all $i = 1, \ldots, s$;
- otherwise, the Runge-Kutta method is implicit.

Runge-Kutta and Butcher's array

For a given Runge-Kutta method,

$$\begin{cases} u_{n+1} = u_n + h\sum_{i=1}^{s} b_i K_i, \quad n = 0, 1, \ldots, N_h - 1, \\ u_0 = y_0, \end{cases} \tag{26}$$

where

$$K_i := f\left(t_n + c_i h, u_n + h\sum_{j=1}^{s} a_{ij} K_j\right), \quad i = 1, \ldots, s, \tag{8.9}$$

the coefficients are stored in the so-called Butcher's array as:

$$\begin{array}{c|c} c & A \\ \hline & \mathbf{b}^T \end{array}$$

- If A is stored from the bottom-left "corner," the Runge-Kutta method is explicit, if $a_{ij} = 0$ for $j \geq i$ for all $i = 1, \ldots, s$;
- otherwise, the Runge-Kutta method is implicit.

Runge-Kutta methods of order 1

Explicit Runge-Kutta with $s = 1$ (RK1) In this case, Eq. (26) becomes:

$$\begin{cases} u_{n+1} = u_n + hb_1 K_1, \quad n = 0, 1, \ldots, N_h - 1, \\ u_0 = y_0, \end{cases}$$

where

$$K_1 = f(t_n + c_1 h, u_n + ha_{11} K_1).$$

By setting $c_1 = 0$, $b_1 = 1$, and $a_{11} = 0$, i.e., with the following Butcher's array:

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

RK1 corresponds to forward Euler

Runge-Kutta methods of order 1

Explicit Runge-Kutta with $s = 1$ (RK1) In this case, Eq. (26) becomes:

$$\begin{cases} u_{n+1} = u_n + hb_1 K_1, \quad n = 0, 1, \ldots, N_h - 1, \\ u_0 = y_0, \end{cases}$$

where

$$K_1 = f(t_n + c_1 h, u_n + ha_{11} K_1).$$

By setting $c_1 = 0$, $b_1 = 1$, and $a_{11} = 0$, i.e., with the following Butcher's array:

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

RK1 corresponds to forward Euler

Runge-Kutta methods of order 1

Implicit Runge-Kutta with $s = 1$ In this case, Eq. (26) becomes:

$$\begin{cases} u_{n+1} = u_n + hb_1 K_1, \quad n = 0, 1, \ldots, N_h - 1, \\ u_0 = y_0, \end{cases}$$

where

$$K_1 = f(t_n + c_1 h, u_n + ha_{11} K_1).$$

With the following Butcher's array,

$$\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}$$

we obtain backward Euler

Runge-Kutta methods of order 1

Implicit Runge-Kutta with $s = 1$ In this case, Eq. (26) becomes:

$$\begin{cases} u_{n+1} = u_n + hb_1 K_1, \quad n = 0, 1, \ldots, N_h - 1, \\ u_0 = y_0, \end{cases}$$

where

$$K_1 = f(t_n + c_1 h, u_n + ha_{11} K_1).$$

With the following Butcher's array,

$$\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}$$

we obtain backward Euler

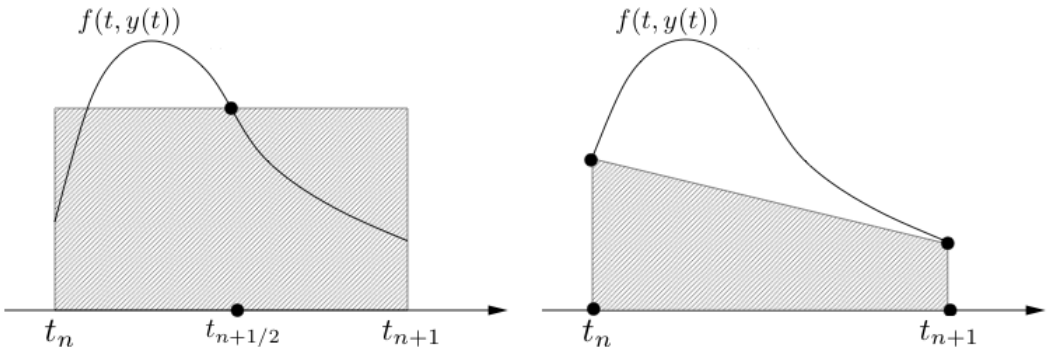
Runge-Kutta methods of order 2

If we integrate the equation $y'(t) = f(t, y(t))$ between t_n and t_{n+1}, we obtain:

$$y(t_{n+1}) - y(t_n) = \int_{t_n}^{t_{n+1}} f(t, y(t))dt. \tag{27}$$

Remark

Numerical integration methods (Chapt. 4.2 in the book)



We want to approximate the integral of the function $f(t, y(t))$. If we use the midpoint formula, we approximate the area below the curve by the area of a rectangle that has as a basis h and as a height the value of the function at time $t_n + h/2$ (see figure on the left). If we use the trapezoidal formula, we approximate the area below the curve by the area of a trapezoid that has as basis both values of the function at times t_n and t_{n+1} and as a height h (see figure on the right).

Using trapezoidal formula, we find the following implicit method, that is called **Crank-Nicolson or trapezoidal method** :

$$\boxed{u_{n+1} - u_n = \frac{h}{2}\left[f(t_n, u_n) + f(t_{n+1}, u_{n+1})\right], \quad \forall n \geq 0.} \tag{28}$$

This method is unconditionally stable when it is applied to the model problem (20).

If we modify the schema (28) (changing to explicit) then we obtain the **Heun method**:

$$\boxed{u_{n+1} - u_n = \frac{h}{2}\left[f(t_n, u_n) + f(t_{n+1}, u_n + hf(t_n, u_n))\right].} \tag{29}$$

Both methods (Crank-Nicolson and Heun) are of order 2 with respect to h.

Runge-Kutta methods of order 2

Explicit Runge-Kutta with $s = 2$ (RK2) In this case, Eq. (26) becomes:

$$\begin{cases} u_{n+1} = u_n + hb_1 K_1 + hb_2 K_2, \quad n = 0, 1, \ldots, N_h - 1, \\ u_0 = y_0, \end{cases}$$

where

$$K_1 = f\left(t_n + c_1 h, u_n + ha_{11} K_1 + ha_{12} K_2\right), \quad K_2 = f\left(t_n + c_2 h, u_n + ha_{21} K_1 + ha_{22} K_2\right).$$

Then, we consider the following Butcher's array:

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

for which we obtain the RK2 method, i.e., the Heun method (8.7), where:

$$K_1 = f(t_n, u_n), \quad K_2 = f(t_{n+1}, u_n + hK_1).$$

RK2 corresponds to Heun method

Runge-Kutta methods of order 2

Explicit Runge-Kutta with $s = 2$ (RK2) In this case, Eq. (26) becomes:

$$\begin{cases} u_{n+1} = u_n + hb_1 K_1 + hb_2 K_2, \quad n = 0, 1, \ldots, N_h - 1, \\ u_0 = y_0, \end{cases}$$

where

$$K_1 = f\left(t_n + c_1 h, u_n + ha_{11} K_1 + ha_{12} K_2\right), \quad K_2 = f\left(t_n + c_2 h, u_n + ha_{21} K_1 + ha_{22} K_2\right).$$

Then, we consider the following Butcher's array:

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

for which we obtain the RK2 method, i.e., the Heun method (8.7), where:

$$K_1 = f(t_n, u_n), \quad K_2 = f(t_{n+1}, u_n + hK_1).$$

RK2 corresponds to Heun method

Runge-Kutta methods of order 2

Runge-Kutta with $s = 2$ In this case, Eq. (26) becomes:

$$\begin{cases} u_{n+1} = u_n + hb_1 K_1 + hb_2 K_2, \quad n = 0, 1, \ldots, N_h - 1, \\ u_0 = y_0, \end{cases}$$

where

$$K_1 = f\left(t_n + c_1 h, u_n + ha_{11} K_1 + ha_{12} K_2\right), \quad K_2 = f\left(t_n + c_2 h, u_n + ha_{21} K_1 + ha_{22} K_2\right).$$

Consider the following Butcher's array:

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

this corresponds to Crank-Nicolson method

Runge-Kutta methods of order 2

Runge-Kutta with $s = 2$ In this case, Eq. (26) becomes:

$$\begin{cases} u_{n+1} = u_n + hb_1 K_1 + hb_2 K_2, \quad n = 0, 1, \ldots, N_h - 1, \\ u_0 = y_0, \end{cases}$$

where

$$K_1 = f\left(t_n + c_1 h, u_n + ha_{11} K_1 + ha_{12} K_2\right), \quad K_2 = f\left(t_n + c_2 h, u_n + ha_{21} K_1 + ha_{22} K_2\right).$$

Consider the following Butcher's array:

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

this corresponds to Crank-Nicolson method

Runge-Kutta method of order 4

Obtained by considering the integration of the Simpson method:

$$\begin{cases} u_{n+1} = u_n + \dfrac{h}{6}(K_1 + 2K_2 + 2K_3 + K_4), \\ u_0 = y_0 \end{cases} \quad \text{where:} \begin{cases} K_1 = f(t_n, u_n), \\ K_2 = f(t_n + \dfrac{h}{2}, u_n + \dfrac{h}{2}K_1), \\ K_3 = f(t_n + \dfrac{h}{2}, u_n + \dfrac{h}{2}K_2), \\ K_4 = f(t_{n+1}, u_n + hK_3). \end{cases}$$

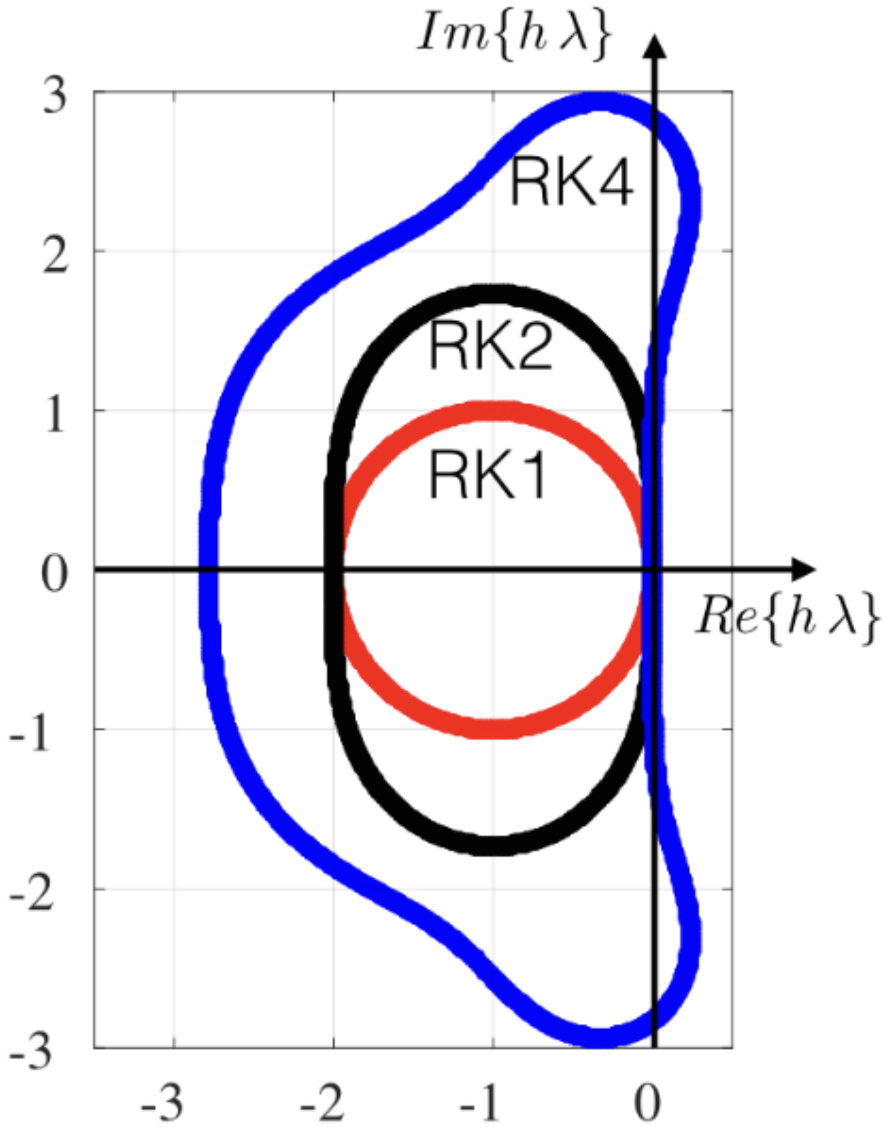
and the corresponding Butcher's array is:

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

In the following table we summarize the characteristics of the methods:

Method	*Explicit/Implicit*	*Stability*	*w.r.to h*
Forward Euler	Explicit	Conditionally	1
Backward Euler	Implicit	Unconditionally	1
Crank–Nicolson	Implicit	Unconditionally	2
Heun	Explicit	Conditionally	2
Runge–Kutta	Explicit	Conditionally	4

Regions of absolute stability $\mathcal{A}$ for RK1, RK2, RK4



Example

Let us consider the Cauchy problem

$$\begin{cases} y'(t) = -y(0.1 - \cos(t)), & t > 0 \\ y(0) = 1. \end{cases} \tag{30}$$

We solve this problem by the forward Euler and Heun methods on the interval $[0, 12]$ with a time step $h = 0.4$.

```
>> f = @(t,y) (cos(t) - 0.1)*y;
>> h = 0.4; tspan = [0 12]; y0 = 1; Nh = 12/h;
>> % forward Euler
>> [t_ep, y_ep] = feuler(f, tspan, y0, Nh);
>> % Heun
>> [t_heun, y_heun] = heun(f, tspan, y0, Nh);
```

Example

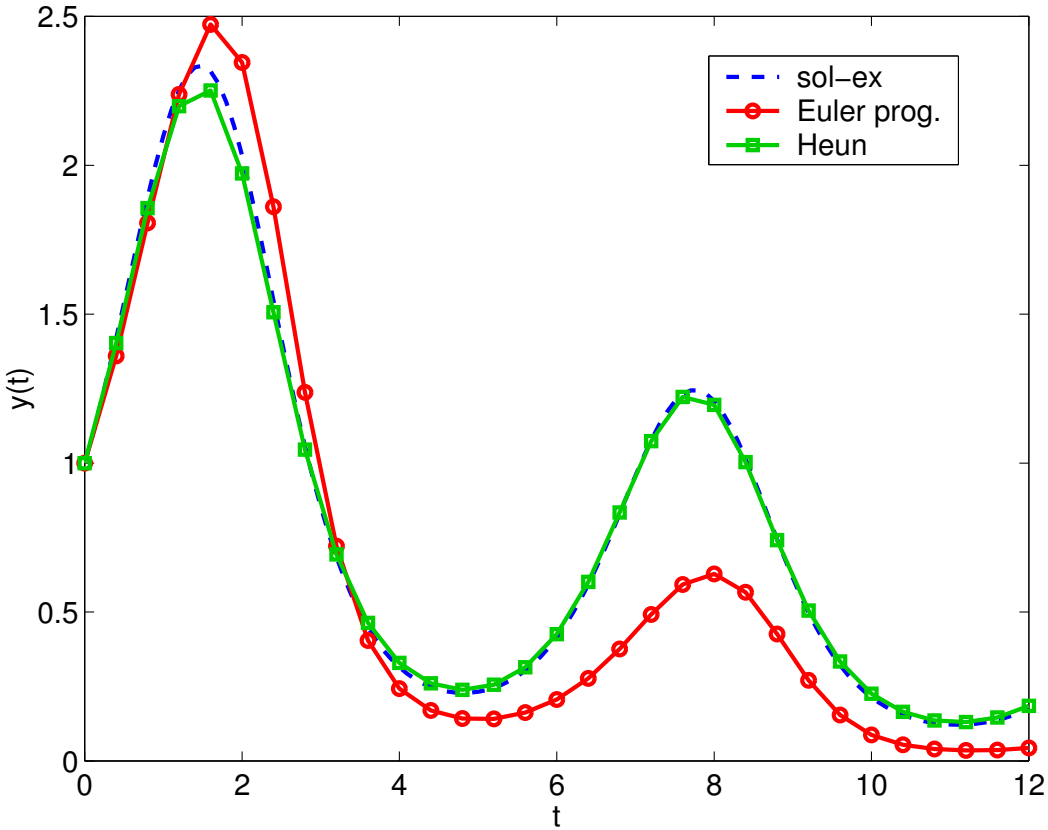
The first of the following figures shows the solutions obtained by both methods and the exact solution $y(t) = e^{-0.1t+\sin(t)}$. Note that the solution obtained by the Heun method is much more precise than the forward Euler method.

Moreover, we can see that if we reduce the time step, the solution obtained by the forward Euler method approximates the exact solution. The second figure shows the solutions obtained with $h = 0.4, 0.2, 0.1, 0.05$ using the following commands:

```
>> sol_ex = @(t) exp(-0.1*t + sin(t));
>> t = [0:0.01:12];
>> plot(t, sol_ex(t), 'b--'); hold on;
>> h=0.4; Nh = 12/h;
>> for i=1:4
     [t_ep, y_ep] = feuler(f, tspan, y0, Nh);
     plot(t_ep, y_ep)
     Nh = Nh*2;
   end
```

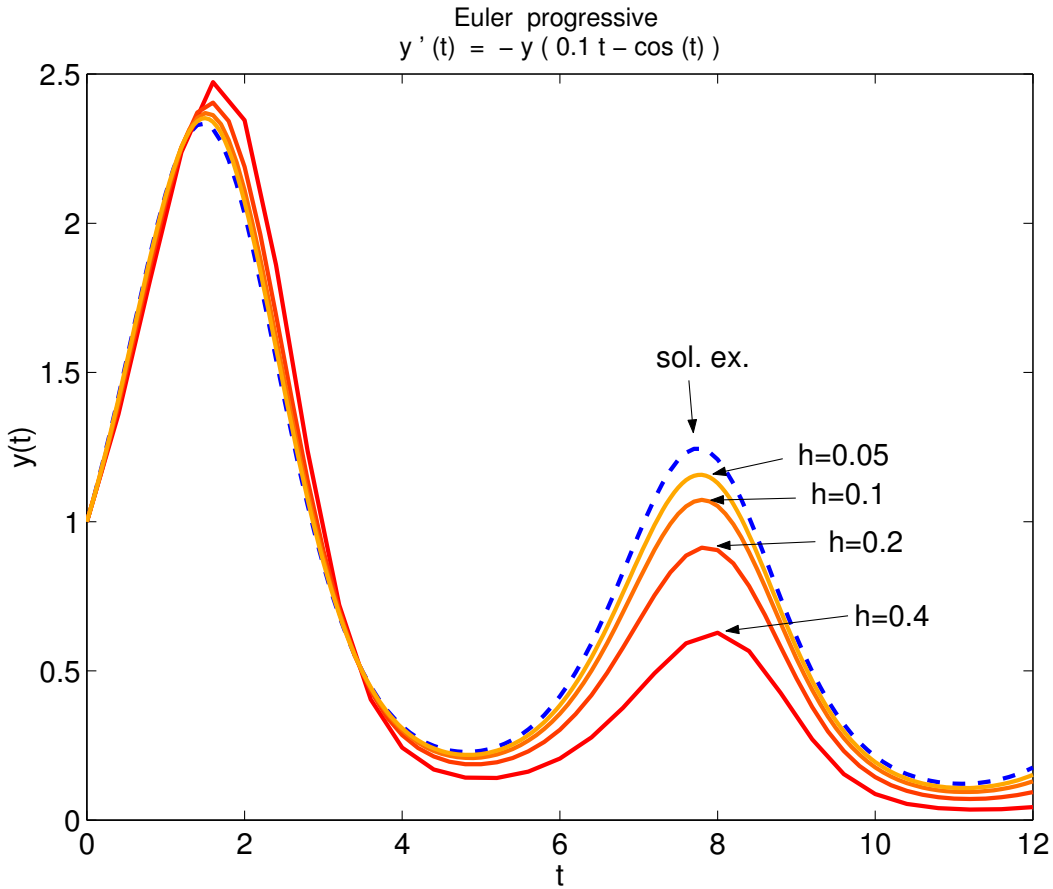
Example

Comparison of solutions obtained by the forward Euler and Heun methods for $h = 0.4$.



Example

Solutions obtained by the forward Euler method for different time steps.



Example

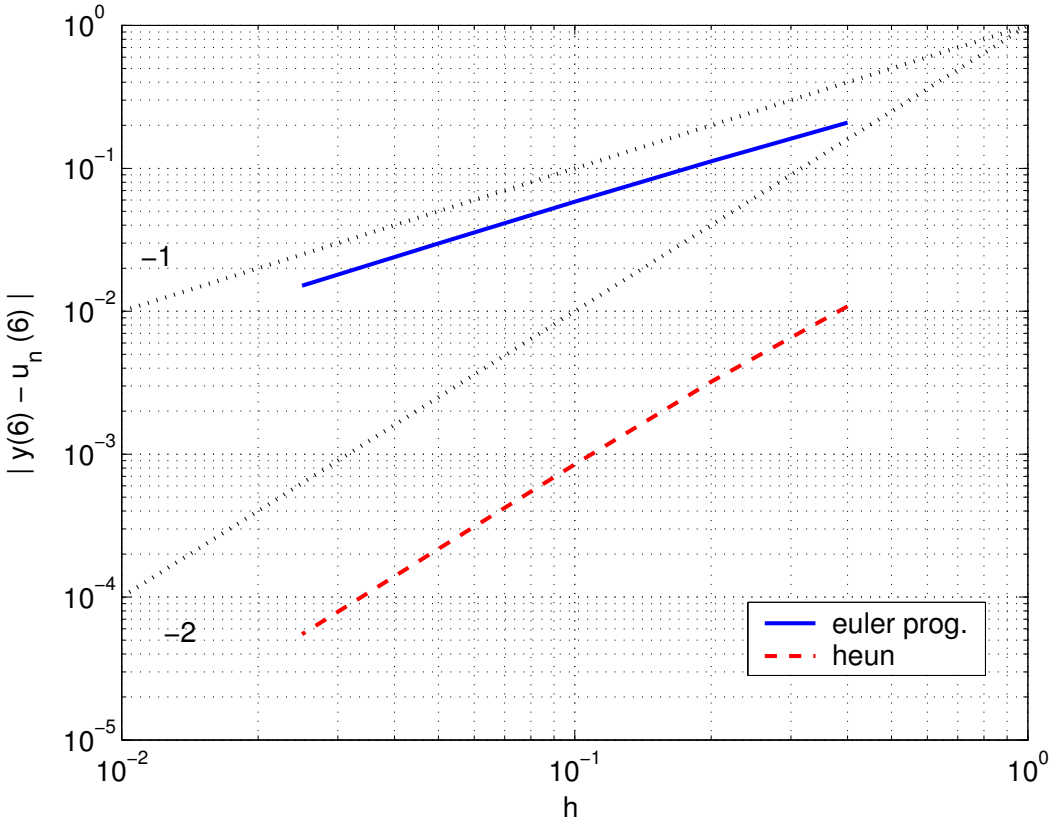
We want to estimate the order of convergence of these two methods. For this we will solve the problem with different time steps and we will compare the results obtained at time $t = 6$ with the exact solution.

```
>> h=0.4; Nh = 12/h; t=6; y6 = sol_ex(t);
>> for i=1:5
    % foreward Euler
    [t_ep, y_ep] = feuler(f, tspan, y0, Nh);
    err_ep(i) = abs(y6 - y_ep(fix(Nh/2)+1));
    % Heun
    [t_heun, y_heun] = heun(f, tspan, y0, Nh);
    err_heun(i) = abs(y6 - y_heun(fix(Nh/2)+1));
    Nh = Nh*2;
   end
>> h=[0.4, 0.2, 0.1, 0.05, 0.025];
>> loglog(h,err_ep,'b',h,err_heun,'r')
```

The following figure shows, in logarithmic scale, errors of both methods depending on h. Clearly, the forward Euler method converges with order 1 and Heun method converges with order 2.

Example

Errors of the forward Euler and Heun methods in the calculation of $y(6)$. Note that a scale is logarithmic.



Plan

Multistep methods for ODEs

- The approximate solution u_{n+1} is obtained by using $u_n, \ldots, u_{n-p}$ for some $p \geq 0$, with $p+1$ being the number of steps,
- A multistep method for approximating the Cauchy problem (2) is:

$$u_{n+1} = \sum_{j=0}^{p} a_j u_{n-j} + h \sum_{j=-1}^{p} b_j f(t_{n-j}, u_{n-j}), \quad n = p, \ldots, N_h - 1, \quad (8.10)$$

given $u_0, \ldots, u_p$, for some coefficients $\{a_j\}_{j=0}^{p}$ and $\{b_j\}_{j=-1}^{p}$, which determine the multistep method.

If $b_{-1} = 0$, the method is explicit; otherwise, it is implicit.

Multistep methods for ODEs

A multistep method is consistent if and only if:

$$-\sum_{j=0}^{p} a_j = 1 \quad \text{and} \quad \sum_{j=0}^{p} j a_j + \sum_{j=-1}^{p} b_j = 1.$$

One-step methods $p = 0$

From eq. (8.10) of multistep methods we obtain:

$$u_{n+1} = a_0 u_n + h b_{-1} f(t_{n+1}, u_{n+1}) + h b_0 f(t_n, u_n), \quad n = 0, 1, \ldots, N_h - 1,$$

given u_0.

- Forward Euler: $a_0 = 1$, $b_{-1} = 0$, and $b_0 = 1$
- Backward Euler: $a_0 = 1$, $b_{-1} = 1$, and $b_0 = 0$
- Crank–Nicolson: $a_0 = 1$, $b_{-1} = \frac{1}{2}$, and $b_0 = \frac{1}{2}$

One-step methods $p = 0$

From eq. (8.10) of multistep methods we obtain:

$$u_{n+1} = a_0 u_n + h b_{-1} f(t_{n+1}, u_{n+1}) + h b_0 f(t_n, u_n), \quad n = 0, 1, \ldots, N_h - 1,$$

given u_0.

- Forward Euler: $a_0 = 1$, $b_{-1} = 0$, and $b_0 = 1$
- Backward Euler: $a_0 = 1$, $b_{-1} = 1$, and $b_0 = 0$
- Crank–Nicolson: $a_0 = 1$, $b_{-1} = \frac{1}{2}$, and $b_0 = \frac{1}{2}$

One-step methods $p = 0$

From eq. (8.10) of multistep methods we obtain:

$$u_{n+1} = a_0 u_n + h b_{-1} f(t_{n+1}, u_{n+1}) + h b_0 f(t_n, u_n), \quad n = 0, 1, \ldots, N_h - 1,$$

given u_0.

- Forward Euler: $a_0 = 1$, $b_{-1} = 0$, and $b_0 = 1$
- Backward Euler: $a_0 = 1$, $b_{-1} = 1$, and $b_0 = 0$
- Crank–Nicolson: $a_0 = 1$, $b_{-1} = \frac{1}{2}$, and $b_0 = \frac{1}{2}$

One-step methods $p = 0$

From eq. (8.10) of multistep methods we obtain:

$$u_{n+1} = a_0 u_n + h b_{-1} f(t_{n+1}, u_{n+1}) + h b_0 f(t_n, u_n), \quad n = 0, 1, \ldots, N_h - 1,$$

given u_0.

- Forward Euler: $a_0 = 1$, $b_{-1} = 0$, and $b_0 = 1$
- Backward Euler: $a_0 = 1$, $b_{-1} = 1$, and $b_0 = 0$
- Crank–Nicolson: $a_0 = 1$, $b_{-1} = \frac{1}{2}$, and $b_0 = \frac{1}{2}$

Common multistep methods

- AB3 (explicit Adam–Bashforth) method with order of accuracy 3, which is a 3-step method ($p = 2$). From multistep eq. (8.10), we have:

$$u_{n+1} = u_n + \frac{h}{12}\left[23f(t_n, u_n) - 16f(t_{n-1}, u_{n-1}) + 5f(t_{n-2}, u_{n-2})\right], \quad n = 2, .., N_h$$

for which the coefficients are:

$$a_0 = 1, \quad a_1 = a_2 = 0, \quad b_{-1} = 0, \quad b_0 = \frac{23}{12}, \quad b_1 = -\frac{16}{12}, \quad b_2 = \frac{5}{12}.$$

- AM4 (implicit Adam–Moulton) method with order of accuracy 4, which is a 3-step method ($p = 2$). From multistep eq. (8.10), we have:

$$u_{n+1} = u_n + \frac{h}{24}\left[9f(t_{n+1}, u_{n+1}) + 19f(t_n, u_n) - 5f(t_{n-1}, u_{n-1}) + f(t_{n-2}, u_{n-2})\right],$$

with the coefficients:

$$a_0 = 1, \quad a_1 = a_2 = 0, \quad b_{-1} = \frac{9}{24}, \quad b_0 = \frac{19}{24}, \quad b_1 = -\frac{5}{24}, \quad b_2 = \frac{1}{24}.$$

Common multistep methods

- AB3 (explicit Adam–Bashforth) method with order of accuracy 3, which is a 3-step method ($p = 2$). From multistep eq. (8.10), we have:

$$u_{n+1} = u_n + \frac{h}{12}\left[23f(t_n, u_n) - 16f(t_{n-1}, u_{n-1}) + 5f(t_{n-2}, u_{n-2})\right], \quad n = 2, .., N_h$$

for which the coefficients are:

$$a_0 = 1, \quad a_1 = a_2 = 0, \quad b_{-1} = 0, \quad b_0 = \frac{23}{12}, \quad b_1 = -\frac{16}{12}, \quad b_2 = \frac{5}{12}.$$

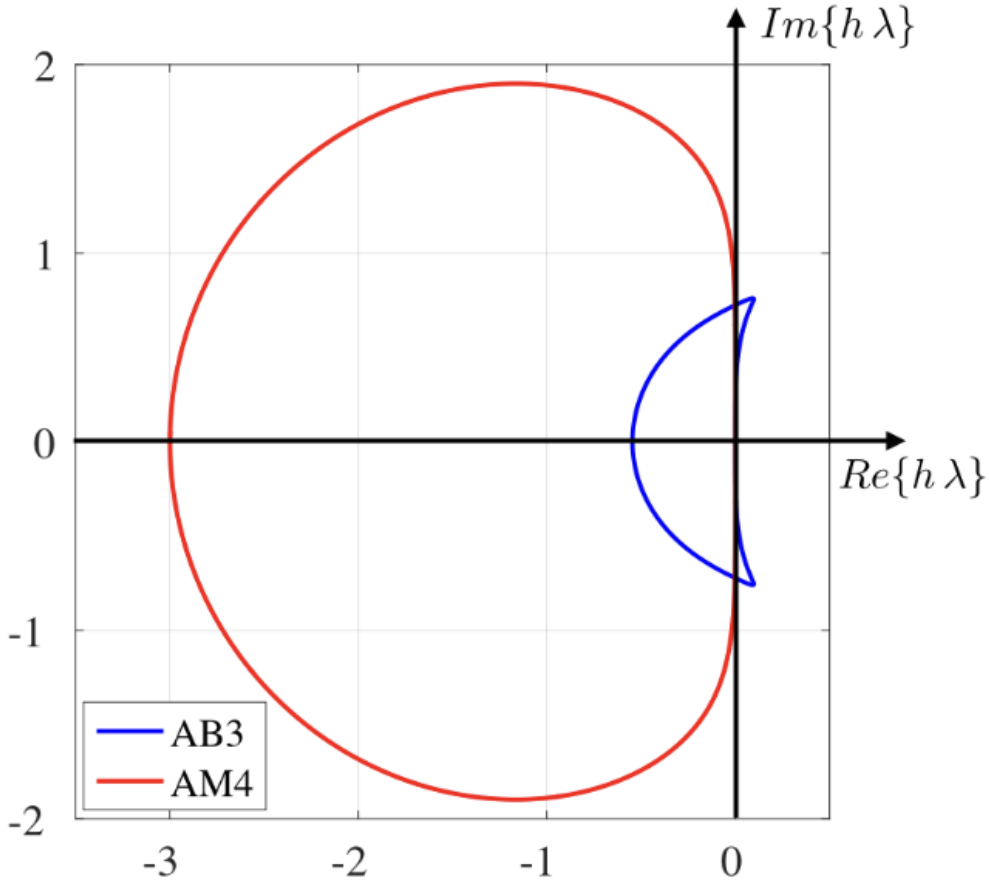
- AM4 (implicit Adam–Moulton) method with order of accuracy 4, which is a 3-step method ($p = 2$). From multistep eq. (8.10), we have:

$$u_{n+1} = u_n + \frac{h}{24}\left[9f(t_{n+1}, u_{n+1}) + 19f(t_n, u_n) - 5f(t_{n-1}, u_{n-1}) + f(t_{n-2}, u_{n-2})\right],$$

with the coefficients:

$$a_0 = 1, \quad a_1 = a_2 = 0, \quad b_{-1} = \frac{9}{24}, \quad b_0 = \frac{19}{24}, \quad b_1 = -\frac{5}{24}, \quad b_2 = \frac{1}{24}.$$

Regions of absolute stability $\mathcal{A}$ for AB3 and AM4



AB3 and AM4 are consistent and zero-stable. They are also conditionally absolutely stable.

Plan

Systems of differential equations

(Chapt. 8.3 in the book)

Definition

Let us consider the interval $I = (t_0, t_f) \subset \mathbb{R}$, then the vector–valued Cauchy problem reads:

$$\text{find } \mathbf{y} : I \to \mathbb{R}^m \text{ s.t.: } \begin{cases} \frac{d\mathbf{y}}{dt}(t) = \mathbf{f}(t, \mathbf{y}(t)) & \text{for all } t \in I, \\ \mathbf{y}(t_0) = \mathbf{y}_0, \end{cases} \tag{31}$$

where $m \geq 1$, $\mathbf{f}(t, \mathbf{y}) : I \times \mathbb{R}^m \to \mathbb{R}^m$ is given and assumed to be continuous in both arguments. We have:

$$\mathbf{y}(t) = \begin{bmatrix} y_1(t) \\ \vdots \\ y_m(t) \end{bmatrix}, \quad \text{and} \quad \mathbf{f}(t, \mathbf{y}) = \begin{bmatrix} f_1(t, \mathbf{y}) \\ \vdots \\ f_m(t, \mathbf{y}) \end{bmatrix};$$

Systems of differential equations

Definition

Consider $\mathbf{f}(t, \mathbf{y}) = A\mathbf{y}(t) + \mathbf{g}(t)$ and obtain the following system of ODEs which is non-homogeneous with constant coefficients.

$$\begin{cases} \frac{d\mathbf{y}(t)}{dt} = A\mathbf{y}(t) + \mathbf{g}(t), t > 0, \\ \mathbf{y}(0) = \mathbf{y}_0, \end{cases} \tag{32}$$

where $A \in \mathbb{R}^{m \times m}$ and $\mathbf{g}(t) \in \mathbb{R}^m$.

If $\mathbf{g}(t) = 0$ for all $t \in (t_0, t_f]$ the system is in homogeneous form.

If A has m distinct eigenvalues λ_j and associated eigenvectors v_j, $j = 1, \ldots, m$, the solution is $\mathbf{y} = \sum_{j=1}^{m} C_j e^{\lambda_j t} \mathbf{v}_j$, where C_j depends on the initial data.

Definition

Consider $\mathbf{f}(t, \mathbf{y}) = A\mathbf{y}(t) + \mathbf{g}(t)$ and obtain the following system of ODEs which is non-homogeneous with constant coefficients.

$$\begin{cases} \frac{d\mathbf{y}(t)}{dt} = A\mathbf{y}(t) + \mathbf{g}(t), t > 0, \\ \mathbf{y}(0) = \mathbf{y}_0, \end{cases} \tag{32}$$

where $A \in \mathbb{R}^{m \times m}$ and $\mathbf{g}(t) \in \mathbb{R}^m$.

If $\mathbf{g}(t) = 0$ for all $t \in (t_0, t_f]$ the system is in homogeneous form.

If A has m distinct eigenvalues λ_j and associated eigenvectors v_j, $j = 1, \ldots, m$, the solution is $\mathbf{y} = \sum_{j=1}^{m} C_j e^{\lambda_j t} \mathbf{v}_j$, where C_j depends on the initial data.

Definition

Consider $\mathbf{f}(t, \mathbf{y}) = A\mathbf{y}(t) + \mathbf{g}(t)$ and obtain the following system of ODEs which is non-homogeneous with constant coefficients.

$$\begin{cases} \frac{d\mathbf{y}(t)}{dt} = A\mathbf{y}(t) + \mathbf{g}(t), t > 0, \\ \mathbf{y}(0) = \mathbf{y}_0, \end{cases} \tag{32}$$

where $A \in \mathbb{R}^{m \times m}$ and $\mathbf{g}(t) \in \mathbb{R}^m$.

If $\mathbf{g}(t) = 0$ for all $t \in (t_0, t_f]$ the system is in homogeneous form.

If A has m distinct eigenvalues λ_j and associated eigenvectors v_j, $j = 1, \ldots, m$, the solution is $\mathbf{y} = \sum_{j=1}^{m} C_j e^{\lambda_j t} \mathbf{v}_j$, where C_j depends on the initial data.

Linear system

Example

The system

$$\begin{cases} y_1'(t) &= -2y_1(t) + y_2(t) + e^{-t} \\ y_2'(t) &= 3y_1(t) - 4y_2(t) \end{cases} \tag{33}$$

with initial conditions $y_1(0) = y_{10}$, $y_2(0) = y_{20}$, can be written as (32), where

$$\mathbf{y}(t) = \begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix}, \quad A = \begin{bmatrix} -2 & 1 \\ 3 & -4 \end{bmatrix}, \quad \mathbf{g}(t) = \begin{bmatrix} e^{-t} \\ 0 \end{bmatrix}, \quad \mathbf{y}_0 = \begin{bmatrix} y_{10} \\ y_{20} \end{bmatrix}.$$

Let $h > 0$ be the time step; for $n \in \mathbb{N}$, we set $t_n = nh$, $\mathbf{g}_n = \mathbf{g}(t_n)$ and we denote by $\mathbf{u}_n$ an approximation of the exact solution $\mathbf{y}(t_n)$ at time t_n.

Θ-method

Let $\theta \in \mathbb{R}$ such that $\theta \in [0, 1]$; then, the θ–method for the approximation of the Cauchy problem (2) is:

$$\begin{cases} \mathbf{u}_{n+1} = \mathbf{u}_n + h\left[(1-\theta)\mathbf{f}(t_n, \mathbf{u}_n) + \theta \mathbf{f}(t_{n+1}, \mathbf{u}_{n+1})\right] & \text{for } n = 0, \ldots, N_h - 1, \\ \mathbf{u}_0 = \mathbf{y}_0. \end{cases} \tag{34}$$

The θ–method is

- explicit for $\theta = 0$,
- implicit for $\theta \in (0, 1]$.

Θ-method

Let $\theta \in \mathbb{R}$ such that $\theta \in [0,1]$; then, the θ–method for the approximation of the Cauchy problem (2) is:

$$\begin{cases} \mathbf{u}_{n+1} = \mathbf{u}_n + h\left[(1-\theta)\mathbf{f}(t_n, \mathbf{u}_n) + \theta \mathbf{f}(t_{n+1}, \mathbf{u}_{n+1})\right] & \text{for } n = 0, \ldots, N_h - 1, \\ \mathbf{u}_0 = \mathbf{y}_0. \end{cases} \tag{34}$$

The θ–method is

- explicit for $\theta = 0$,
- implicit for $\theta \in (0,1]$.

From the numerical point of view, the methods introduced in the scalar case can be extended to systems of differential equations. For example, the forward Euler method (11) becomes:

$$\begin{cases} \dfrac{\mathbf{u}_{n+1} - \mathbf{u}_n}{h} = \mathbf{f}(t_n, \mathbf{u}_n) = A\mathbf{u}_n + \mathbf{g}_n \qquad \text{for } n = 0, 1, 2, \ldots, N_h - 1 \\ \mathbf{u}_0 = \mathbf{y}_0 \ , \end{cases} \tag{35}$$

while the backward Euler method (12) becomes:

$$\begin{cases} \dfrac{\mathbf{u}_{n+1} - \mathbf{u}_n}{h} = \mathbf{f}(t_{n+1}, \mathbf{u}_{n+1}) = A\mathbf{u}_{n+1} + \mathbf{g}_{n+1} \qquad \text{for } n = 0, 1, 2, \ldots, N_h - 1 \\ \mathbf{u}_0 = \mathbf{y}_0 \ , \end{cases} \tag{36}$$

FE, BE, and CN as Θ-methods

One iteration of Θ-methods is:

$$\begin{cases} \mathbf{u}_{n+1} = \mathbf{u}_n + h\left[(1-\theta)f(t_n, \mathbf{u}_n) + \theta f(t_{n+1}, \mathbf{u}_{n+1})\right] \\ \mathbf{u}_0 = \mathbf{y}_0. \end{cases}$$

We obtain:

forward Euler ($\Theta = 0$)
$$\begin{cases} \mathbf{u}_{n+1} = \mathbf{u}_n + hA\mathbf{u}_n + h\mathbf{g}_n = (I + hA)\mathbf{u}_n + h\mathbf{g}_n \\ \mathbf{u}_0 = \mathbf{y}_0 \end{cases}$$

backward Euler ($\Theta = 1$)
$$\begin{cases} (I - hA)\mathbf{u}_{n+1} = \mathbf{u}_n + h\mathbf{g}_{n+1} \\ \mathbf{u}_0 = \mathbf{y}_0 \end{cases}$$

Crank-Nicolson ($\Theta = 1/2$)
$$\begin{cases} (I - \frac{h}{2}A)\mathbf{u}_{n+1} = (I + \frac{h}{2}A)\mathbf{u}_n + \frac{h}{2}(\mathbf{g}_n + \mathbf{g}_{n+1}) \\ \mathbf{u}_0 = \mathbf{y}_0 \end{cases}$$

Implicit methods (BE and CN), solve a linear system at each step with the matrix $I - hA$ and $I - \frac{h}{2}A$, respectively.

FE, BE, and CN as Θ-methods

One iteration of Θ-methods is:

$$\begin{cases} \mathbf{u}_{n+1} = \mathbf{u}_n + h\left[(1-\theta)f(t_n, \mathbf{u}_n) + \theta f(t_{n+1}, \mathbf{u}_{n+1})\right] \\ \mathbf{u}_0 = \mathbf{y}_0. \end{cases}$$

We obtain:

forward Euler ($\Theta = 0$)
$$\begin{cases} \mathbf{u}_{n+1} = \mathbf{u}_n + hA\mathbf{u}_n + h\mathbf{g}_n = (I + hA)\mathbf{u}_n + h\mathbf{g}_n \\ \mathbf{u}_0 = \mathbf{y}_0 \end{cases}$$

backward Euler ($\Theta = 1$)
$$\begin{cases} (I - hA)\mathbf{u}_{n+1} = \mathbf{u}_n + h\mathbf{g}_{n+1} \\ \mathbf{u}_0 = \mathbf{y}_0 \end{cases}$$

Crank-Nicolson ($\Theta = 1/2$)
$$\begin{cases} (I - \frac{h}{2}A)\mathbf{u}_{n+1} = (I + \frac{h}{2}A)\mathbf{u}_n + \frac{h}{2}(\mathbf{g}_n + \mathbf{g}_{n+1}) \\ \mathbf{u}_0 = \mathbf{y}_0 \end{cases}$$

Implicit methods (BE and CN), solve a linear system at each step with the matrix $I - hA$ and $I - \frac{h}{2}A$, respectively.

FE, BE, and CN as Θ-methods

One iteration of Θ-methods is:

$$\begin{cases} \mathbf{u}_{n+1} = \mathbf{u}_n + h\left[(1-\theta)f(t_n, \mathbf{u}_n) + \theta f(t_{n+1}, \mathbf{u}_{n+1})\right] \\ \mathbf{u}_0 = \mathbf{y}_0. \end{cases}$$

We obtain:

forward Euler ($\Theta = 0$)
$$\begin{cases} \mathbf{u}_{n+1} = \mathbf{u}_n + hA\mathbf{u}_n + h\mathbf{g}_n = (I + hA)\mathbf{u}_n + h\mathbf{g}_n \\ \mathbf{u}_0 = \mathbf{y}_0 \end{cases}$$

backward Euler ($\Theta = 1$)
$$\begin{cases} (I - hA)\mathbf{u}_{n+1} = \mathbf{u}_n + h\mathbf{g}_{n+1} \\ \mathbf{u}_0 = \mathbf{y}_0 \end{cases}$$

Crank-Nicolson ($\Theta = 1/2$)
$$\begin{cases} (I - \frac{h}{2}A)\mathbf{u}_{n+1} = (I + \frac{h}{2}A)\mathbf{u}_n + \frac{h}{2}(\mathbf{g}_n + \mathbf{g}_{n+1}) \\ \mathbf{u}_0 = \mathbf{y}_0 \end{cases}$$

Implicit methods (BE and CN), solve a linear system at each step with the matrix $I - hA$ and $I - \frac{h}{2}A$, respectively.

FE, BE, and CN as Θ-methods

One iteration of Θ-methods is:

$$\begin{cases} \mathbf{u}_{n+1} = \mathbf{u}_n + h\left[(1-\theta)f(t_n, \mathbf{u}_n) + \theta f(t_{n+1}, \mathbf{u}_{n+1})\right] \\ \mathbf{u}_0 = \mathbf{y}_0. \end{cases}$$

We obtain:

forward Euler ($\Theta = 0$)
$$\begin{cases} \mathbf{u}_{n+1} = \mathbf{u}_n + hA\mathbf{u}_n + h\mathbf{g}_n = (I + hA)\mathbf{u}_n + h\mathbf{g}_n \\ \mathbf{u}_0 = \mathbf{y}_0 \end{cases}$$

backward Euler ($\Theta = 1$)
$$\begin{cases} (I - hA)\mathbf{u}_{n+1} = \mathbf{u}_n + h\mathbf{g}_{n+1} \\ \mathbf{u}_0 = \mathbf{y}_0 \end{cases}$$

Crank-Nicolson ($\Theta = 1/2$)
$$\begin{cases} (I - \frac{h}{2}A)\mathbf{u}_{n+1} = (I + \frac{h}{2}A)\mathbf{u}_n + \frac{h}{2}(\mathbf{g}_n + \mathbf{g}_{n+1}) \\ \mathbf{u}_0 = \mathbf{y}_0 \end{cases}$$

Implicit methods (BE and CN), solve a linear system at each step with the matrix $I - hA$ and $I - \frac{h}{2}A$, respectively.

The error associated with the numerical approximation of the Cauchy problem (2) at $t = t_n$ is $e_n := \|\mathbf{y}_n - \mathbf{u}_n\|_2$ for some $n = 0, 1, \ldots, N_h$. If

$$e_n \leq Ch^p,$$

C a positive constant independent of h, then the method has convergence order $p > 0$.

Method	*Explicit/Implicit*	Order of convergence
Forward Euler	Explicit	1 (if $\mathbf{y} \in C^2(I)$)
Backward Euler	Implicit	1 (if $\mathbf{y} \in C^2(I)$)
Crank–Nicolson	Implicit	2 (if $\mathbf{y} \in C^3(I)$)
Heun	Explicit	2 (if $\mathbf{y} \in C^3(I)$)

Heun method for systems of ODEs

Heun method is not a θ–method, but can be deduced by applying the reasoning to a system of ODEs, to obtain:

$$\begin{aligned}
\mathbf{u}_{n+1}^{*} &= \mathbf{u}_n + h\mathbf{f}(t_n, \mathbf{u}_n), \\
\mathbf{u}_{n+1} &= \mathbf{u}_n + \frac{h}{2}\left[\mathbf{f}(t_n, \mathbf{u}_n) + \mathbf{f}(t_{n+1}, \mathbf{u}_{n+1}^{*})\right], \ n = 0, \ldots, N_h - 1, \\
\mathbf{u}_0 &= \mathbf{y}_0.
\end{aligned}$$

Stability

Consider a non-homogeneous system of ODEs with constant coefficients,

$$\mathbf{f}(t, \mathbf{y}) = A\mathbf{y} + \mathbf{g}(t).$$

Assume that eigenvalues of A satisfy

$$\mathrm{Re}\{\lambda_i(A)\} < 0 \text{ for all } i = 1, \ldots, m,$$

then a numerical method is absolutely stable if if the condition on the stability function R (as in Def. 8.5 in lecture notes) is satisfied for all the eigenvalues, that is,

$$|R(h\lambda_i(A))| < 1, \quad \text{for } i = 1, \ldots, m,$$

Stability

Under the assumptions in the previous slide, we obtain:

- Forward Euler is conditionally absolutely stable for $h > 0$ s.t.

$$|1 + h\lambda_i(A)| < 1, \quad \text{for } i = 1, \ldots, m,$$

or

$$h < \frac{2}{\max_{j=1,\ldots,p} |\lambda_j|} = \frac{2}{\rho(A)}\,, \tag{37}$$

where $\rho(A)$ is the spectral radius of A,

- Heun method is conditionally absolutely stable for $h > 0$ s.t.

$$|1 + h\lambda_i(A) + \frac{(h\lambda_i(A))^2}{2}| < 1, \quad \text{for } i = 1, \ldots, m.$$

- Backward Euler and Crank-Nicolson are unconditionally absolutely stable.

Example

Linear system

The system

$$\begin{cases} y_1'(t) &= -2y_1(t) + y_2(t) + e^{-t} \\ y_2'(t) &= 3y_1(t) - 4y_2(t) \end{cases} \tag{38}$$

with initial conditions $y_1(0) = y_{10}$, $y_2(0) = y_{20}$, can be written as (32), where

$$\mathbf{y}(t) = \begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix}, \quad A = \begin{bmatrix} -2 & 1 \\ 3 & -4 \end{bmatrix}, \quad \mathbf{g}(t) = \begin{bmatrix} e^{-t} \\ 0 \end{bmatrix}, \quad \mathbf{y}_0 = \begin{bmatrix} y_{10} \\ y_{20} \end{bmatrix}.$$

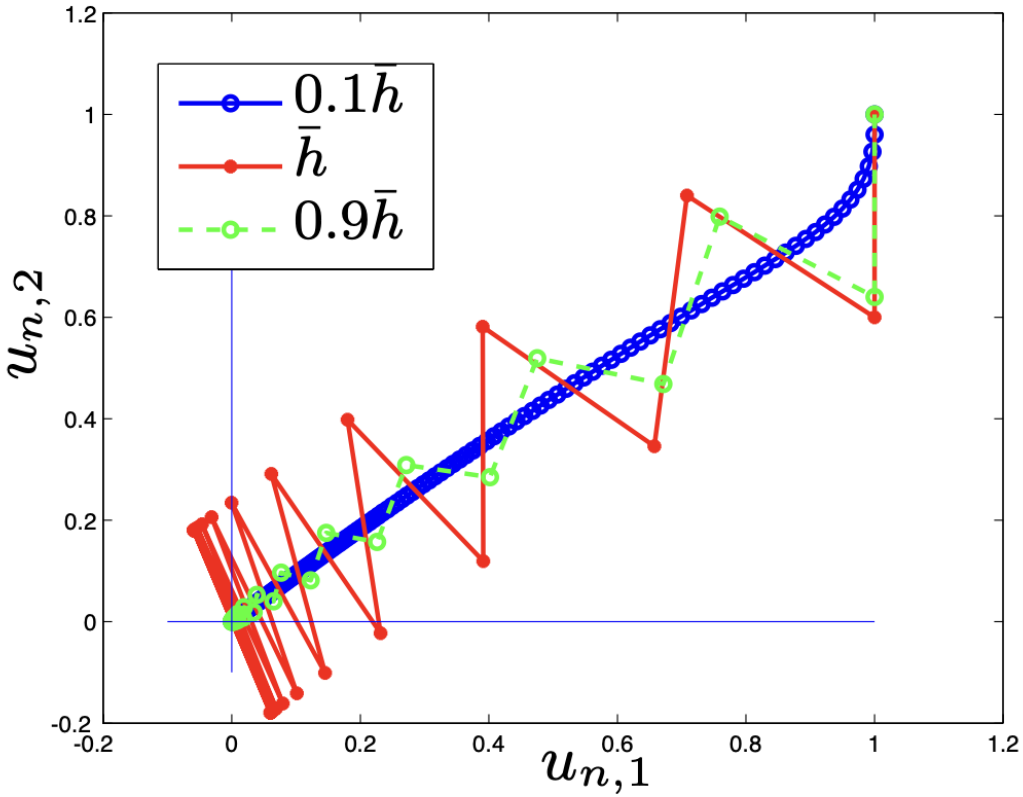
Let $h > 0$ be the time step; for $n \in \mathbb{N}$, we set $t_n = nh$, $\mathbf{g}_n = \mathbf{g}(t_n)$ and we denote by $\mathbf{u}_n$ an approximation of the exact solution $\mathbf{y}(t_n)$ at time t_n.

The forward Euler method is explicit (there is no linear system to solve), but it is stable conditionally. In this case, the eigenvalues of A are $\lambda_1 = -1$ and $\lambda_2 = -5$; they are strictly negative, so the condition (37) on h is satisfied. $\rho(A) = 5$, so the stability condition is

$$h < \bar{h} = \frac{2}{5}.$$

Example

Behavior of the forward Euler method for the system ((38)) with initial condition $\mathbf{y}_0 = [1, 1]^T$ and different values of the time step h.




```
A = [-2 1; 3 -4];
dy = @(t,y,A) A*y + [exp(-t); 0];
h_bar = 2 / max(abs(eig(A)));
[t,y]=feuler(dy,[0,10],[1;1],10/(0.1*h_bar),A);
plot(y(1,:), y(2,:)); hold on;
[t,y]=feuler(dy,[0,10],[1;1],10/(h_bar),A);
plot(y(1,:), y(2,:), 'ro');
[t,y]=feuler(dy,[0,10],[1;1],10/(0.9*h_bar),A);
plot(y(1,:), y(2,:), 'go--');
```

We could also consider the case of a nonlinear system of the form

$$\mathbf{y}'(t) = \mathbf{F}(t, \mathbf{y}(t)),$$

(for example the system (1)). Consider the Jacobian matrix

$$J(t) = \frac{\partial \mathbf{F}}{\partial \mathbf{y}}(t, \mathbf{y}(t))$$

and assume $Re\{\lambda_i(J(t))\} < 0$ for all $i = 1, \ldots, m$.
Then the backward Euler method is unconditionally stable, while the forward Euler method is stable under condition (37), where $A = \frac{\partial \mathbf{F}}{\partial \mathbf{y}}$.

Example

The nonlinear system

$$\begin{aligned} y_1'(t) &= -2y_1(t) + \sin(y_2(t)) + e^{-t}\sin(t), \\ y_2'(t) &= \cos(y_1(t)) - 4y_2(t), \end{aligned} \tag{39}$$

with initial conditions $y_1(0) = y_{10}$, $y_2(0) = y_{20}$, can be written as

$$\mathbf{y}'(t) = \mathbf{F}(t, \mathbf{y}(t)),$$

where

$$\mathbf{F}(t, \mathbf{y}(t)) = \begin{bmatrix} -2y_1(t) + \sin(y_2(t)) + e^{-t}\sin(t) \\ \cos(y_1(t)) - 4y_2(t) \end{bmatrix}.$$

Let $h > 0$ be the time step; for $n \in \mathbb{N}$, we set $t_n = nh$ and we denote by $\mathbf{u}_n$ an approximation of the exact solution $\mathbf{y}(t_n)$ at time t_n.

Example

The forward Euler, backward Euler and Crank-Nicolson methods for approximating the solution $\mathbf{y}(t)$ of (39) are written respectively:

$$\text{forward Euler} \quad \begin{cases} \mathbf{u}_{n+1} = \mathbf{u}_n + h\mathbf{F}(t_n, \mathbf{u}_n), \\ \mathbf{u}_0 = \mathbf{y}_0, \end{cases}$$

$$\text{backward Euler} \quad \begin{cases} \mathbf{u}_{n+1} + h\mathbf{F}(t_{n+1}, \mathbf{u}_{n+1}) = \mathbf{u}_n, \\ \mathbf{u}_0 = \mathbf{y}_0, \end{cases}$$

$$\text{Crank-Nicolson} \quad \begin{cases} \mathbf{u}_{n+1} - \frac{h}{2}\mathbf{F}(t_{n+1}, \mathbf{u}_{n+1}) = \mathbf{u}_n + \frac{h}{2}\mathbf{F}(t_n, \mathbf{u}_n), \\ \mathbf{u}_0 = \mathbf{y}_0. \end{cases}$$

It should be noted that at each step of the methods of BE and CN, we must solve a nonlinear system.

Example

The forward Euler method is explicit (there is not system to solve), but it is stable conditionally. In this case, the jacobian of $\mathbf{F}$ is given by

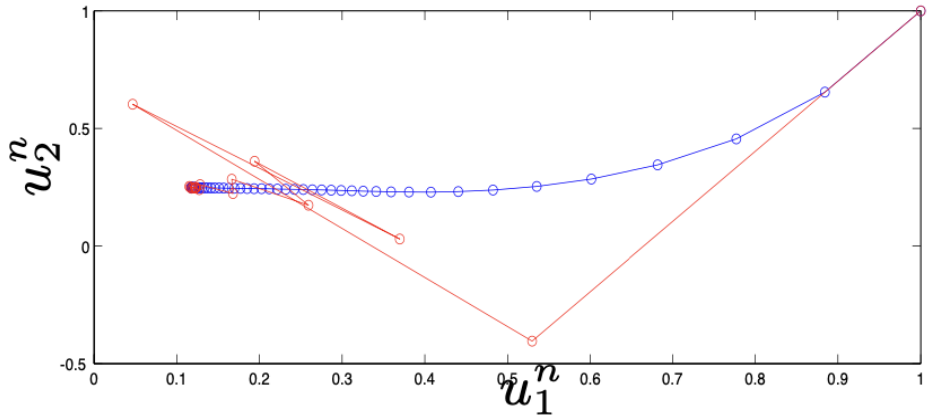
$$J = \frac{\partial \mathbf{F}}{\partial \mathbf{y}} = \begin{bmatrix} -2 & \cos y_2 \\ -\sin y_1 & -4, \end{bmatrix}$$

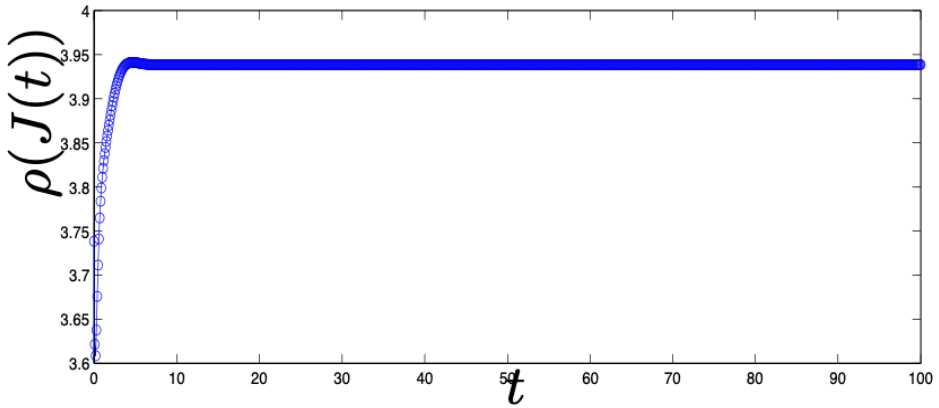
and the eigenvalues are $\lambda_{1,2} = -3 \pm \sqrt{1 - \sin y_1 \cos y_2}$; They are strictly negative, in particular $-3 - \sqrt{2} < \lambda_{1,2} < -3 + \sqrt{2} < 0$, and $\rho(J) < 3 + \sqrt{2}$. Therefore the stability condition is

$$h < \bar{h} = \frac{2}{\rho(J)}, \quad \text{for example if} \quad h < \frac{2}{3 + \sqrt{2}} \simeq 0.453.$$

Example

Behaviour of the forward Euler method for the system (39) with initial condition $\mathbf{y}_0 = [1, 1]^\top$: $h = 0.1$ (blue) and $h = 0.8\bar{h}$ (red). If we take $h \geq \bar{h}$, we can observe the instability of the method.





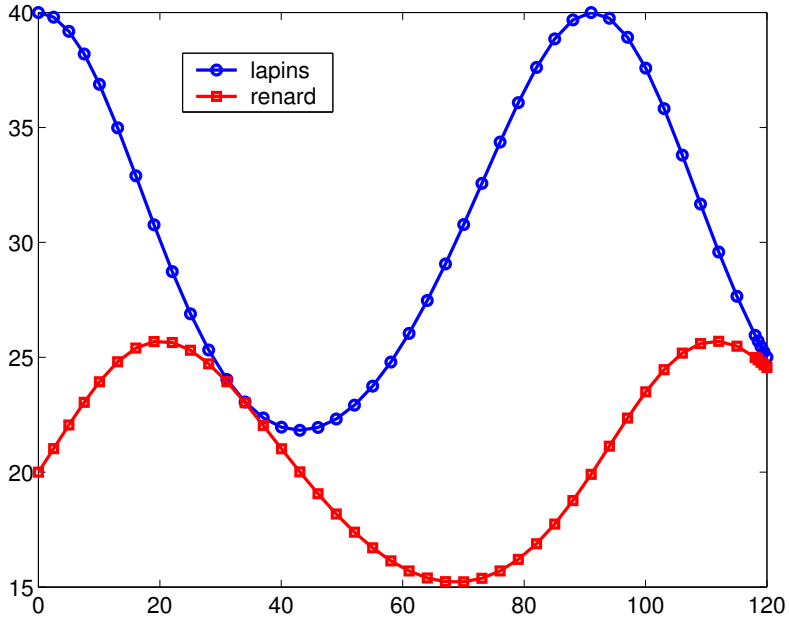
Example

We used the following commands:

```
dy = @(t,y) [-2*y(1) + sin(y(2)) + exp(-t)*sin(t); ...
             cos(y(1)) - 4*y(2)]
[t,y] = feuler(dy, [0,100],[1; 1], 100/0.1);
subplot(2,1,1); plot(y(1,:), y(2,:),'o-'); hold on;

J = @(y) [-2, cos(y(2)); -sin(y(1)), - 4];
for i = 1:size(t,2);
  rho(i) = max(abs(eig(feval(J, y(:,i)))));
end

subplot(2,1,2);
plot(t,rho, 'o-');

h_bar = 2/max(rho);

[t,y] = feuler(dy, [0,100],[1; 1], 100/(0.8*h_bar));
subplot(2,1,1);
plot(y(1,:), y(2,:), 'ro-');
```

Applications

We return to the example given at the beginning of the chapter.



Example

(prey-predator) (1) We consider the system (1). Let us take an initial population $y_1(0)$ of 40 rabbits, a population $y_2(0)$ of 20 foxes and the Lotka-Volterra equations:

$$\begin{cases} y_1'(t) = 0.08\, y_1(t) - 0.004\, y_1(t) y_2(t), \\ y_2'(t) = -0.06\, y_2(t) + 0.002\, y_1(t) y_2(t). \end{cases} \tag{40}$$

We want to study the evolution of the two populations over a period of 10 years. Let us define the vectors:

$$\mathbf{y}(t) = \begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix}, \qquad \mathbf{F}(t, \mathbf{y}) = \begin{bmatrix} 0.08\, y_1(t) - 0.004\, y_1(t) y_2(t) \\ -0.06\, y_2(t) + 0.002\, y_1(t) y_2(t) \end{bmatrix},$$

We can rewrite the system (40) in the general form:

$$\mathbf{y}'(t) = \mathbf{F}(t, \mathbf{y}), \quad t > 0, \qquad \mathbf{y}(0) = [y_1(0), y_2(0)]^T. \tag{41}$$

Example

All the methods that we have seen so far are applicable to the system (41). For example, the forward Euler method can be written as

$$\frac{\mathbf{u}_{n+1} - \mathbf{u}_n}{h} = \mathbf{F}(t_n, \mathbf{u}_n),$$

which is equivalent to the system of equations

$$\begin{cases} \dfrac{u_{n+1,1} - u_{n,1}}{h} = 0.08\, u_{n,1} - 0.004\, u_{n,1} u_{n,2}, \quad n \geq 0 \\ \dfrac{u_{n+1,2} - u_{n,2}}{h} = -0.06\, u_{n,2} + 0.002\, u_{n,1} u_{n,2}, \quad n \geq 0 \\ u_{0,1} = y_1(0), \quad u_{0,2} = y_2(0). \end{cases}$$

Example

The command `heun` can solve system of differential equations. First we must write a function that define the system:

```
>> fun2 = @(t,y) [ 0.08*y(1) - 0.004*y(1)*y(2);
                  -0.06*y(2) + 0.002*y(1)*y(2) ]
```

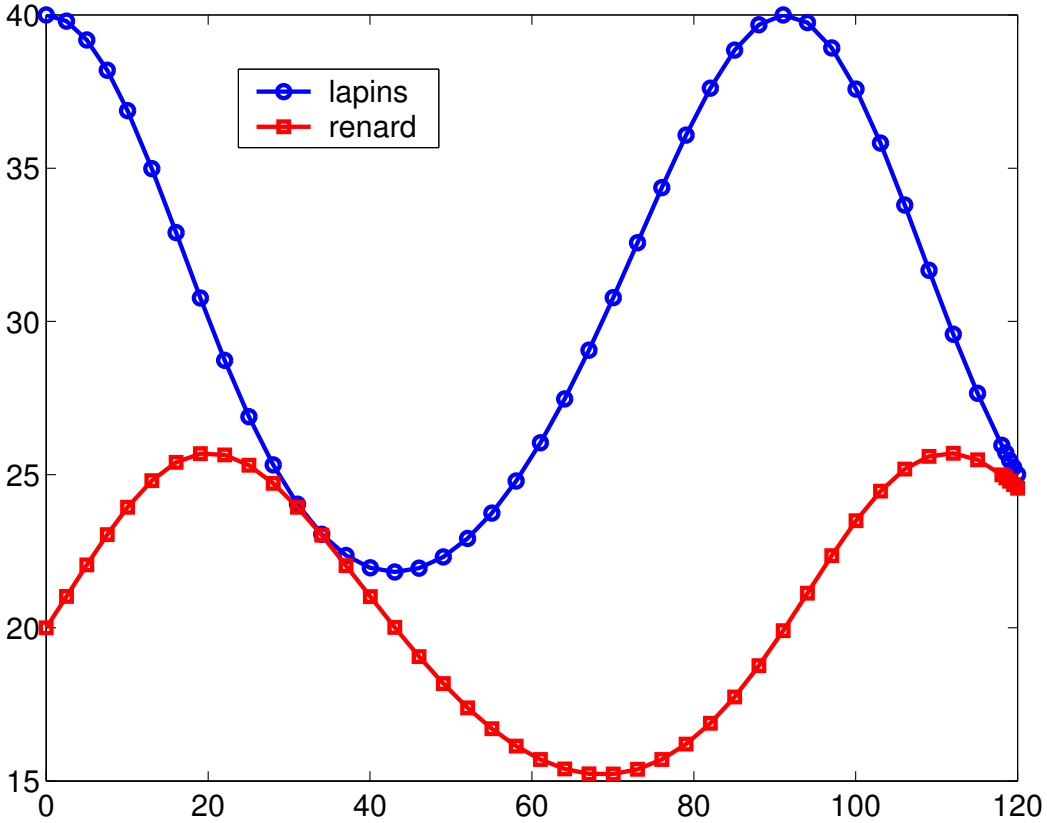
Then we can solve the system using:

```
>> y0=[40 20]; tspan=[0 120]; Nh=40;
>> [t,y] = heun(fun2, tspan, y0, Nh);
>> plot(t,y(:,1),'b', t,y(:,2),'r')
```

The first column of `y` contains the solution y_1, while the second column contains y_2. The following figure shows the evolution of the two populations.

Example

Evolution of populations of rabbits and foxes over 10 years.



Plan

Second order ODEs

Given $I = (t_0, t_f) \subset \mathbb{R}$, then a second-order ODE reads:

$$\text{find } y : I \to \mathbb{R} \text{ s.t.: } \begin{cases} \frac{d^2y}{dt^2}(t) = f\left(t, y(t), \frac{dy}{dt}(t)\right), & \text{for all } t \in I, \\ \frac{dy}{dt}(t_0) = y_{1,0}, \\ y(t_0) = y_{0,0}, \end{cases} \tag{42}$$

where $f(t, y, w_2) : I \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ and $y_{0,0}$ and $y_{1,0}$ is initial data.

Introduce $w_2(t) : I \to \mathbb{R}$ s.t. $w_2(t) = \frac{dy}{dt}(t)$ for all $t \in I$. Obtain:

$$\text{find } y, w_2 : I \to \mathbb{R} \text{ s.t.: } \begin{cases} \frac{dw_2}{dt}(t) = f(t, y(t), w_2(t)), & \text{for all } t \in I, \\ \frac{dy}{dt}(t) = w_2(t), & \text{for all } t \in I, \\ w_2(t_0) = y_{1,0}, \\ y(t_0) = y_{0,0}. \end{cases} \tag{43}$$

Second order ODEs

Given $I = (t_0, t_f) \subset \mathbb{R}$, then a second-order ODE reads:

$$\text{find } y : I \to \mathbb{R} \text{ s.t.:} \begin{cases} \frac{d^2y}{dt^2}(t) = f\left(t, y(t), \frac{dy}{dt}(t)\right), & \text{for all } t \in I, \\ \frac{dy}{dt}(t_0) = y_{1,0}, \\ y(t_0) = y_{0,0}, \end{cases} \tag{42}$$

where $f(t, y, w_2) : I \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ and $y_{0,0}$ and $y_{1,0}$ is initial data.

Introduce $w_2(t) : I \to \mathbb{R}$ s.t. $w_2(t) = \frac{dy}{dt}(t)$ for all $t \in I$. Obtain:

$$\text{find } y, w_2 : I \to \mathbb{R} \text{ s.t.:} \begin{cases} \frac{dw_2}{dt}(t) = f(t, y(t), w_2(t)), & \text{for all } t \in I, \\ \frac{dy}{dt}(t) = w_2(t), & \text{for all } t \in I, \\ w_2(t_0) = y_{1,0}, \\ y(t_0) = y_{0,0}. \end{cases} \tag{43}$$

Second order ODEs

Thus we need to solve the system of ODEs:

$$\text{find } y, w_2 : I \to \mathbb{R} \text{ s.t.:} \begin{cases} \frac{dw_2}{dt}(t) = f(t, y(t), w_2(t)), & \text{for all } t \in I, \\ \frac{dy}{dt}(t) = w_2(t), & \text{for all } t \in I, \\ w_2(t_0) = y_{1,0}, \\ y(t_0) = y_{0,0}. \end{cases} \tag{44}$$

that can be written as the Cauchy problem for a system of ODEs (31):

$$\mathbf{y}(t) = \begin{bmatrix} w_2(t) \\ y(t) \end{bmatrix}, \quad \mathbf{f}(t, \mathbf{y}) = \begin{bmatrix} f(t, y, w_2) \\ w_2(t) \end{bmatrix}, \quad \mathbf{y}_0 = \begin{bmatrix} y_{1,0} \\ y_{0,0} \end{bmatrix},$$

where $\mathbf{y} : I \to \mathbb{R}^2$ and $\mathbf{f}(t, \mathbf{y}) : I \times \mathbb{R}^2 \to \mathbb{R}^2$.

General high order ODEs

Consider $I = (t_0, t_f) \subset \mathbb{R}$. A high-order ODE of order $m \geq 2$ is:

$$\text{find } y : I \to \mathbb{R} \text{ s.t.:} \begin{cases} \frac{d^m y}{dt^m}(t) = f\left(t, y(t), \frac{dy}{dt}(t), \ldots, \frac{d^{m-1}y}{dt^{m-1}}(t)\right), \text{ for all } t \in I, \\ \frac{d^{m-1}y}{dt^{m-1}}(t_0) = y_{m-1,0}, \\ \vdots \\ y(t_0) = y_{0,0}, \end{cases} \tag{45}$$

where $f(t, y, w_2, \ldots, w_m) : I \times \mathbb{R} \times \mathbb{R} \times \cdots \times \mathbb{R} \to \mathbb{R}$ has $m+1$ arguments, and $\{y_{k,0}\}_{k=0}^{m-1}$ is the initial data.

General high order ODEs

Introduce the auxiliary variables $w_k(t) : I \to \mathbb{R}$ s.t.:

$$w_k(t) = \frac{d^{k-1}y}{dt^{k-1}}(t) \quad \text{for all } t \in I, \quad \text{and for } k = 2, \ldots, m.$$

Rewrite general high-order ODE as a system of ODEs:

$$\text{find } y, w_2, \ldots, w_m : I \to \mathbb{R} \text{ s.t.:} \begin{cases} \frac{dw_m}{dt}(t) = f\left(t, y(t), w_2(t), \ldots, w_m(t)\right), & \text{for all } t \in I, \\ \frac{dw_{m-1}}{dt}(t) = w_m(t), & \text{for all } t \in I, \\ \vdots \\ \frac{dy}{dt}(t) = w_2(t), & \text{for all } t \in I, \\ w_m(t_0) = y_{m-1,0}, \\ \vdots \\ w_2(t_0) = y_{1,0}, \\ y(t_0) = y_{0,0}. \end{cases}$$

General high order ODEs

$$\text{find } y, w_2, \dots, w_m : I \to \mathbb{R} \text{ s.t.:} \begin{cases} \frac{dw_m}{dt}(t) = f\left(t, y(t), w_2(t), \dots, w_m(t)\right), \text{ for all } t \in I, \\ \frac{dw_{m-1}}{dt}(t) = w_m(t), \quad \text{for all } t \in I, \\ \vdots \\ \frac{dy}{dt}(t) = w_2(t), \quad \text{for all } t \in I, \\ w_m(t_0) = y_{m-1,0}, \\ \vdots \\ w_2(t_0) = y_{1,0}, \\ y(t_0) = y_{0,0}. \end{cases}$$

The system can be recast in the general form of the Cauchy problem (31):

$$\mathbf{y}(t) = \begin{bmatrix} w_m(t) \\ \vdots \\ w_2(t) \\ y(t) \end{bmatrix}, \quad \mathbf{f}(t, \mathbf{y}) = \begin{bmatrix} f(t, y, w_2, \dots, w_m) \\ w_m \\ \vdots \\ w_2 \end{bmatrix}, \quad \mathbf{y}_0 = \begin{bmatrix} y_{m-1,0} \\ \vdots \\ y_{1,0} \\ y_{0,0} \end{bmatrix}.$$

where $\mathbf{y} : I \to \mathbb{R}^m$ and $\mathbf{f}(t, \mathbf{y}) : I \times \mathbb{R}^m \to \mathbb{R}^m$.