11MAMY – Cvičení 3

Modelování pomocí obyčejných diferenciálních rovnic
část druhá

Jan Přikryl, Jakub Ševčík, Martin Janda

ČVUT FD (Přikryl), ZČU FEL (Přikryl, Ševčík, Janda)

23. února 2023

Obsah cvičení

Úvodní poznámky

Toto cvičení využívá z části předpřipravených skriptů a funkcí, jež si můžete stáhnout ze stránek předmětu na `https://zolotarev.fd.cvut.cz/mamy/static/2023/mamy-2023-c-03.zip`.

Balíček `mamy-2023-c-03.zip` obsahuje pro vaše pohodlí implementaci základních numerických metod pro řešení ODR a souborů pro experimenty s PID regulátorem, které budeme potřebovat později:

- `ode1.m`
- `piw.m` – požadovaný průběh vstupů
- `piz.m` – simulace poruchy
- `pidemo.m` – šablona M-souboru k doplnění

Formát zápisu diferenciální rovnice

Příklad volání `ode45`

Knihovní metoda `ode45()` numericky řeší soustavu ODR, popsanou M-funkcí, s proměnným integračním krokem h_k určeným automaticky, pro daný interval hodnot parametru t a daný vektor počátečních podmínek:

```
[t,y] = ode45(@odefun,tspan,y0)
```

kde vstupní parametry jsou

`odefun` ... funkce $\mathbf{y}' = \mathbf{f}(t, \mathbf{y})$ se signaturou `dy=odefun(t,y)` reprezentující soustavu ODR; zápis `@odefun` je odkaz na danou funkci

`tspan` ... explicitně dané časové kroky t_k nebo interval $\langle t_0, t_n \rangle$

`y0` ... vektor počátečních podmek $\mathbf{y}_0$

Formát zápisu diferenciální rovnice

Příklad volání `ode45`

Knihovní metoda `ode45()` numericky řeší soustavu ODR, popsanou M-funkcí, s proměnným integračním krokem h_k určeným automaticky, pro daný interval hodnot parametru t a daný vektor počátečních podmínek:

```
[t,y] = ode45(@odefun,tspan,y0)
```

kde výstupní parametry jsou

- `t` ... vektor (nerovnoměrných) časových značek t_k
- `y` ... matice výstupních hodnot $\mathbf{y}_k$, po sloupcích

Ovečky a vlci

Viz přednáška ...

Ovečky a vlci

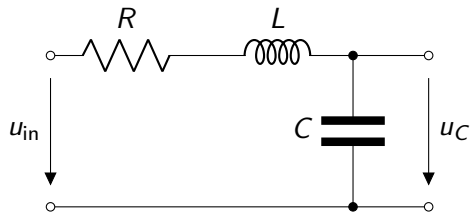
Bača

Příklad můžeme ještě zkomplikovat tak, že do modelu zahrneme baču, který v rámci péče o stádo sem tam zastřelí nějakého vlka a sem tam nějakou ovci porazí, aby měl skopové.
Paradoxně se může ukázat, že i když bača ujídá, tak pokud je schopen vlky dostatečně eliminovat, stádo se díky tomu rozroste na vyšší počet kusů – ovšem jen do té doby, než plně zaúřadují vlci.
Bača, pokud jeho vliv v rovnici vývoje stavu nepostihneme korektně, způsobí také to, že počty ovcí nebo vlků klesnou výrazně do záporna. I to lze ale v modelové funkci ošetřit vhodnými podmínkami.

RLC filtr

Uvažujme RLC obvod v následujícím zapojení:

$$\frac{\mathrm{d}}{\mathrm{d}t} i_L(t) = -\frac{R}{L} i_L(t) - \frac{1}{L} u_C(t) + \frac{1}{L} u_{\mathrm{in}}(t)$$
$$\frac{\mathrm{d}}{\mathrm{d}t} u_C(t) = \frac{1}{C} i_L(t)$$



Jde o lineární dynamický systém se stavem $\mathbf{x}(t) = (i_L(t), u_C(t))^{\mathsf{T}}$, jehož rovnice vývoje stavu je

$$\frac{\mathrm{d}}{\mathrm{d}t} \begin{bmatrix} i_L(t) \\ u_C(t) \end{bmatrix} = \begin{bmatrix} -\frac{R}{L} & -\frac{1}{L} \\ \frac{1}{C} & 0 \end{bmatrix} \begin{bmatrix} i_L(t) \\ u_C(t) \end{bmatrix} + \begin{bmatrix} \frac{1}{L} \\ 0 \end{bmatrix} u_{\mathrm{in}}(t)$$
$$\mathbf{x}'(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{b} u_{\mathrm{in}}(t)$$

RLC filtr

Zápis soustavy ODR v Matlabu

Model soustavy uložíme ve formě $\mathbf{x}' = \mathbf{f}(t, \mathbf{x}, \dots)$ do `rlc.m`:

```
function dxdt=rlc(t,x,u_in,A,b) % u_in(tau) je funkce!
  dxdt = ...;
end
```

RLC filtr

Zápis soustavy ODR v Matlabu

Model soustavy uložíme ve formě $\mathbf{x}' = \mathbf{f}(t, \mathbf{x}, \dots)$ do `rlc.m`:

```
function dxdt=rlc(t,x,u_in,A,b) % u_in(tau) je funkce!
  dxdt = A*x + b*u_in(t);
end
```

Analogicky s příkladem z minulého cvičení převdeme `dxdt=rlc(t,x,u_in,A,B)` na `dxdt=odefunc(t,x)`, abychom mohli použít odpovídající ODR řešiče v Matlabu.

RLC filtr

Zápis soustavy ODR v Matlabu

Model soustavy uložíme ve formě $\mathbf{x}' = \mathbf{f}(t, \mathbf{x}, \dots)$ do `rlc.m`:

```
function dxdt=rlc(t,x,u_in,A,b) % u_in(tau) je funkce!
  dxdt = A*x + b*u_in(t);
end
```

Analogicky s příkladem z minulého cvičení převdeme `dxdt=rlc(t,x,u_in,A,B)` na `dxdt=odefunc(t,x)`, abychom mohli použít odpovídající ODR řešiče v Matlabu.

Použijeme tzv. anonymní funkci:

```
A = ...; % zadáme prvky matice A
b = ...; % definujeme vektor q
% Odkážeme se na funkci u_in(t)=rlcu(t)
odefunc = @(t,x) rlc(t,x,@rlcu,A,b); % A,b jsou konstanty
```

RLC filtr ($R = 10\,\mathrm{k\Omega}$, $L = 100\,\mathrm{mH}$, $C = 100\,\mu\mathrm{F}$)

```
clear all; close all;
% Parametry obvodu
R = ...; L = ...; C = ...; % Ohmů, Henry, Faradů
t0 = 0; tn = 10; % Doba řešení
% Matice modelu obvodu
A = ...; % zadáme prvky matice A
b = ...; % definujeme vektor b
x0 = [0;0]; % Počáteční podmínky
% Diferenciální rovnice
odefunc = @(t,x) rlc(t,x,@rlcu,A,b);
% Řešení
[t,x] = ode45(odefunc, [t0 tn], x0);
% Obrázek
f = figure(1);
subplot(...); plot(...); legend(...); xlabel(...); ylabel(...);
```

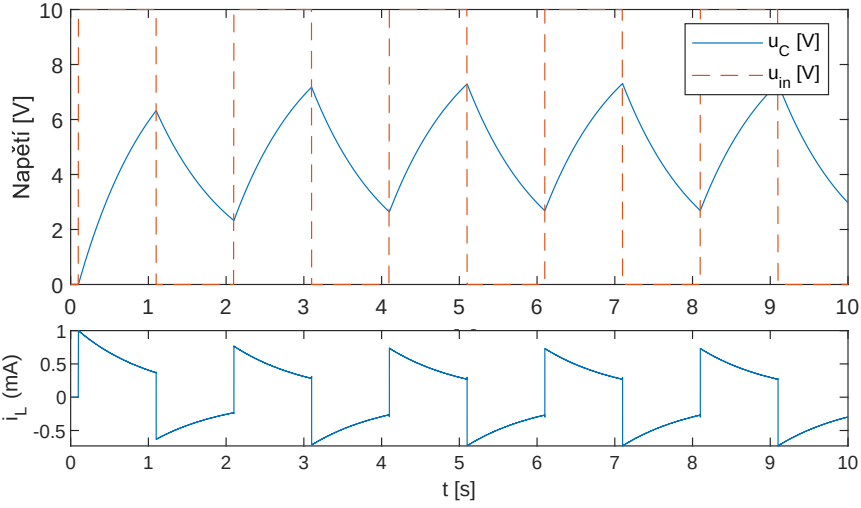
RLC filtr ($R = 10\,\mathrm{k\Omega}$, $L = 100\,\mathrm{mH}$, $C = 100\,\mu\mathrm{F}$)

```
clear all; close all;
% Parametry obvodu
R = 10000; L = 0.1; C = 1e-4; % Ohmů, Henry, Faradů
t0 = 0; tn = 10; % Doba řešení
% Matice modelu obvodu
A = [-R/L -1/L; 1/C 0]; % prvky matice A
b = [1/L; 0]; % definujeme vektor b
x0 = [0;0]; % Počáteční podmínky
% Diferenciální rovnice
odefunc = @(t,x) rlc(t,x,@rlcu,A,b);
% Řešení
[t,x] = ode45(odefunc, [t0 tn], x0);
% Obrázek
f = figure(1);
%% (... pokračuje)
```

RLC filtr ($R = 10\,\mathrm{k\Omega}$, $L = 100\,\mathrm{mH}$, $C = 100\,\mu\mathrm{F}$)

```
%% (pokračuje ...)
% Obrázek
f = figure(1);
subplot(3, 1, [1 2]); % 2x výška
plot(t, x(:,2), t, rlcu(t), '--');
ylabel('Napětí␣[V]');
xlabel('t␣[s]');
legend({'u_C␣[V]', 'u_{in}␣[V]'});
%
subplot(3,1,3);
plot(t,x(:,1)*1000);
xlabel('t␣[s]');
ylabel('i_L␣(mA)');
```

RLC filtr ($R = 10\,\mathrm{k\Omega}$, $L = 100\,\mathrm{mH}$, $C = 100\,\mu\mathrm{F}$)



Stabilita Eulerovy metody

Příklad „tuhé“ ODR

Nedostatky Eulerovy metody demonstrujeme na ODR tvaru

$$y'(t) = -15y(t), \quad y(0) = 1 \qquad \Leftrightarrow \qquad y(t) = \mathrm{e}^{-15t}.$$

Funkci nejprve implementujeme jako `dydt=fstiff(t,y)` v `fstiff.m`:

```
function dydt = fstiff(t,y)
    dydt = ... % Doplňte sami
end
```

Stabilita Eulerovy metody

Příklad „tuhé“ ODR

Nedostatky Eulerovy metody demonstrujeme na ODR tvaru

$$y'(t) = -15y(t), \quad y(0) = 1 \qquad \Leftrightarrow \qquad y(t) = \mathrm{e}^{-15t}.$$

Funkci nejprve implementujeme jako `dydt=fstiff(t,y)` v `fstiff.m`:

```
function dydt = fstiff(~,y) % první parametr ignorujeme
    dydt = -15*y;
end
```

Stabilita Eulerovy metody

Příklad „tuhé“ ODR

Nedostatky Eulerovy metody demonstrujeme na ODR tvaru

$$y'(t) = -15y(t), \quad y(0) = 1 \qquad \Leftrightarrow \qquad y(t) = \mathrm{e}^{-15t}.$$

Funkci nejprve implementujeme jako `dydt=fstiff(t,y)` v `fstiff.m`:

```
function dydt = fstiff(~,y) % první parametr ignorujeme
    dydt = -15*y;
end
```

Úlohu budeme řešit metodou `ode1()` pro $t_0 = 0$, $t_n = 1$ a pro kroky $h \in (1/4, 1/8, 1/16)$. Výsledné průběhy řešení porovnáme s „přesným“ řešením.

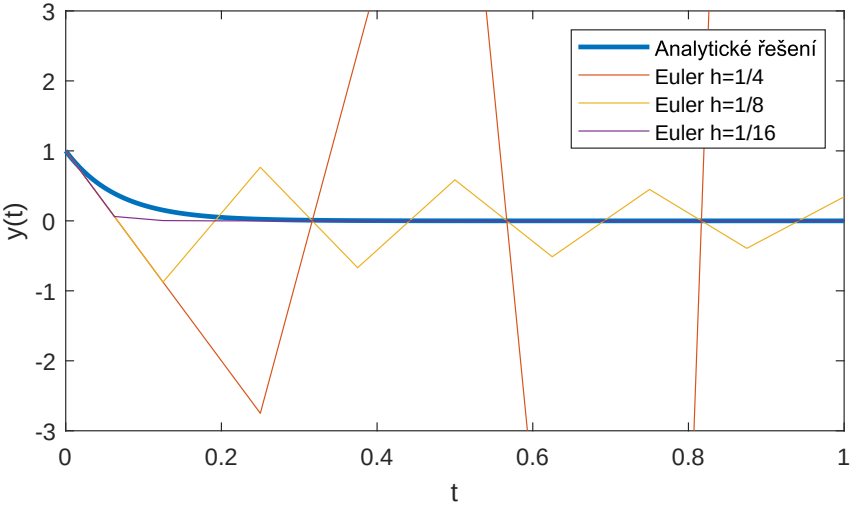
Stabilita Eulerovy metody

```
clear all; close all;
% Euler, různé kroky
[t4,y4] = ode1(...); % Jak voláme ode1()? Jaký krok?
[t8,y8] = ode1(...); % Jaký krok?
[t16,y16] = ode1(...); % Jaký krok?
% "Přesné" řešení
t = ...; % Rovnoměrně 1000 bodů mezi 0 a 1
y = ...; % Výpočet známé y(t) podle t
% Obrázek
f = figure(1);
plot(...); % Jak vykreslíme 4 průběhy s jinými t?
ylim([-3, 3]); % Abychom viděli okolí přesného řešení
legend(...);
xlabel(...);
ylabel(...);
```

Stabilita Eulerovy metody

```
clear all; close all;
% Euler, různé kroky
t0 = 0; tn = 1; y0 = 1; % Parametry
[t4,y4] = ode1(@fstiff, t0, 1/4, tn, y0); % h = 1/4
[t8,y8] = ode1(@fstiff, t0, 1/8, tn, y0); % h = 1/8
[t16,y16] = ode1(@fstiff, t0, 1/16, tn, y0); % h = 1/16
% "Přesné" řešení
t = linspace(0,1,1000); y = exp(-15*t);
% Obrázek
f = figure(1);
plot(t, y, t4, y4, t8, y8, t16, y16);
ylim([-3, 3]);
legend('Analytické␣řešení', 'Euler␣h=1/4', 'Euler␣h=1/8', 'Euler␣
h=1/16');
xlabel('t'); ylabel('y(t)');
```

Stabilita Eulerovy metody



Stabilita Eulerovy metody

Proč dochází k oscilacím?

Z přednášek víme, že řešení v jednotlivých krocích leží na průbězích odpovídajícím jiným počátečním podmínkám úlohy, než byly ty původně zadané.

Pro krok $h = 1/8$ máme pouze první pár hodnot (t_0, y_0) správně. **Q:** Proč?

Stabilita Eulerovy metody

Proč dochází k oscilacím?

Z přednášek víme, že řešení v jednotlivých krocích leží na průbězích odpovídajícím jiným počátečním podmínkám úlohy, než byly ty původně zadané.

Pro krok $h = 1/8$ máme pouze první pár hodnot (t_0, y_0) správně. **Q:** Proč? **A:** Protože y_0 je počáteční podmínka.

Stabilita Eulerovy metody

Proč dochází k oscilacím?

Z přednášek víme, že řešení v jednotlivých krocích leží na průbězích odpovídajícím jiným počátečním podmínkám úlohy, než byly ty původně zadané.

Pro krok $h = 1/8$ máme pouze první pár hodnot (t_0, y_0) správně. **Q:** Proč? **A:** Protože y_0 je počáteční podmínka.

Ostatní hodnoty $y_1, y_2, \ldots, y_8$ jsou velmi nepřesné, protože odpovídají polohám řešení rovnice $y' = -15y$ pro jiné počáteční podmínky, jde tedy o funkce

$$y_i(t) = c_i y(t) = c_i \cdot \mathrm{e}^{-15t}, \quad i = 1, 2, \ldots, 8.$$

Stabilita Eulerovy metody

Proč dochází k oscilacím?

Z přednášek víme, že řešení v jednotlivých krocích leží na průbězích odpovídajícím jiným počátečním podmínkám úlohy, než byly ty původně zadané.

Pro krok $h = 1/8$ máme pouze první pár hodnot (t_0, y_0) správně. **Q:** Proč? **A:** Protože y_0 je počáteční podmínka.

Ostatní hodnoty $y_1, y_2, \ldots, y_8$ jsou velmi nepřesné, protože odpovídají polohám řešení rovnice $y' = -15y$ pro jiné počáteční podmínky, jde tedy o funkce

$$y_i(t) = c_i y(t) = c_i \cdot \mathrm{e}^{-15t}, \quad i = 1, 2, \ldots, 8.$$

Nakresleme si v Matlabu ilustrační obrázek: Koeficienty c_i dokážeme spočítat z hodnot (t_1, y_1) až (t_8, y_8). **Q:** Jak?

Stabilita Eulerovy metody

Proč dochází k oscilacím?

Z přednášek víme, že řešení v jednotlivých krocích leží na průbězích odpovídajícím jiným počátečním podmínkám úlohy, než byly ty původně zadané.

Pro krok $h = 1/8$ máme pouze první pár hodnot (t_0, y_0) správně. **Q:** Proč? **A:** Protože y_0 je počáteční podmínka.

Ostatní hodnoty $y_1, y_2, \ldots, y_8$ jsou velmi nepřesné, protože odpovídají polohám řešení rovnice $y' = -15y$ pro jiné počáteční podmínky, jde tedy o funkce

$$y_i(t) = c_i y(t) = c_i \cdot \mathrm{e}^{-15t}, \quad i = 1, 2, \ldots, 8.$$

Nakresleme si v Matlabu ilustrační obrázek: Koeficienty c_i dokážeme spočítat z hodnot (t_1, y_1) až (t_8, y_8). **Q:** Jak? **A:** $y_i = c_i \exp(-15t_i)$ takže $c_i = y_i / \exp(-15t_i)$.

Stabilita Eulerovy metody

Vykreslení průběhu nepřesných řešení

```
clear all; close all;
% Euler, krok pouze 1/8
[t8,y8] = ode1(...); % Viz minulý příklad
% "Přesné" řešení
t = ...; % Viz minulý příklad
y = ...; % Viz minulý příklad
% Dopočet c_1 až c_8
c = zeros(1,8);
for i = 2:9 % Matlab indexuje y od 1! y_1 je y(2)!
    c(i-1) = .../...; % Pro i=2 máme c_1, zmatek ...
end
%% Ještě bude pokračovat dále
```

Stabilita Eulerovy metody

Vykreslení průběhu nepřesných řešení

```
clear all; close all;
% Euler, krok pouze 1/8
[t8,y8] = ode1(@fstiff, 0, 1/8, 1, 1); % Viz minulý příklad
% "Přesné" řešení
t = linspace(0,1,1000);
y = exp(-15*t);
% Dopočet c_1 až c_8
c = zeros(1,8);
for i = 2:9 % Matlab indexuje y od 1! y_1 je y(2)!
    c(i-1) = y8(i)/exp(-15*t8(i)); % Pro i=2 máme c_1, zmatek ...
end
%% Ještě bude pokračovat dále
```

Stabilita Eulerovy metody

Vykreslení průběhu nepřesných řešení

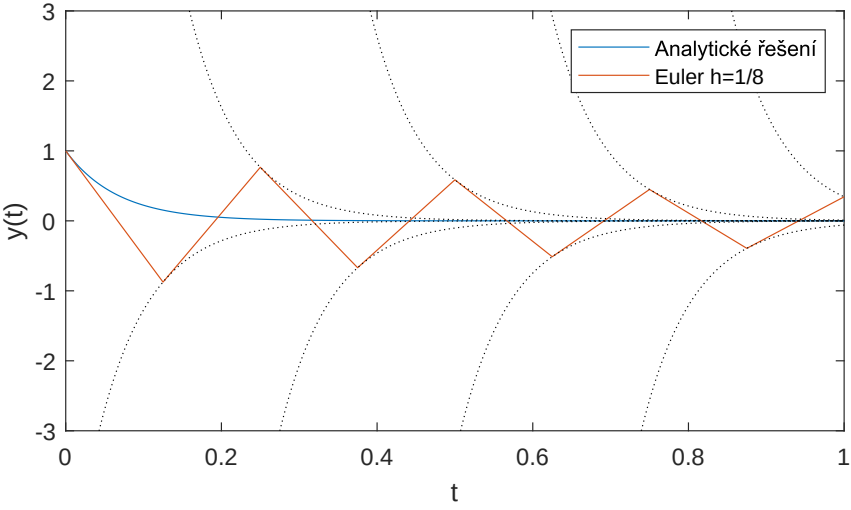
```
% Obrázek
f = figure(1);
plot(t, y, t8, y8); % Přesné a Eulerovo řešení s krokem 1/8
hold on; % Nemaž původně vykreslené
for i = 1:8 % Vykresli průběhy nepřesných řešení
    plot(...); % Jednotlivé c_i * y(t) černě tečkovaně
end
hold off; % Další plot() už zase maže
ylim([-3, 3]); % Stále kvůli přehlednosti
legend('Analytické řešení', 'Euler h=1/8');
xlabel('t');
ylabel('y(t)');
```

Stabilita Eulerovy metody

Vykreslení průběhu nepřesných řešení

```
% Obrázek
f = figure(1);
plot(t, y, t8, y8); % Přesné a Eulerovo řešení s krokem 1/8
hold on; % Nemaž původně vykreslené
for i = 1:8
    plot(t, c(i)*y, ':k'); % Jednotlivé c_i * y(t) černě tečkovaně
end
hold off; % Další plot() už zase maže
ylim([-3, 3]); % Stále kvůli přehlednosti
legend('Analytické řešení', 'Euler h=1/8');
xlabel('t');
ylabel('y(t)');
```

Nepřesná řešení u Eulerovy metody



Implicitní a explicitní Eulerova metoda

Jako závěrečný experiment s Eulerovou metodou porovnejme řešení ODR

$$y'(t) = -15y(t), \quad y(0) = 1$$

explicitní Eulerovou metodou `ode1()` a implicitní Eulerovou metodou `ode1i()` s krokem $h = 1/4$.

Z přednášek víme, že implicitní Eulerova metoda je pro stabilní ODR metodou *bezpodmínečně stabilní*, nemá omezení na velikost integračního kroku h a měla by tedy i pro $h = 1/4$ poskytnout snesitelně přesné řešení.

Q: V čem spočívá nevýhoda explicitních metod?

Implicitní a explicitní Eulerova metoda

Jako závěrečný experiment s Eulerovou metodou porovnejme řešení ODR

$$y'(t) = -15y(t), \quad y(0) = 1$$

explicitní Eulerovou metodou `ode1()` a implicitní Eulerovou metodou `ode1i()` s krokem $h = 1/4$.

Z přednášek víme, že implicitní Eulerova metoda je pro stabilní ODR metodou *bezpodmínečně stabilní*, nemá omezení na velikost integračního kroku h a měla by tedy i pro $h = 1/4$ poskytnout snesitelně přesné řešení.

Q: V čem spočívá nevýhoda explicitních metod?
A: Jsou výpočetně náročné, protože v každém kroku k navíc numericky řešíme nelineární rovnici pro $\mathbf{y}_{k+1}$.

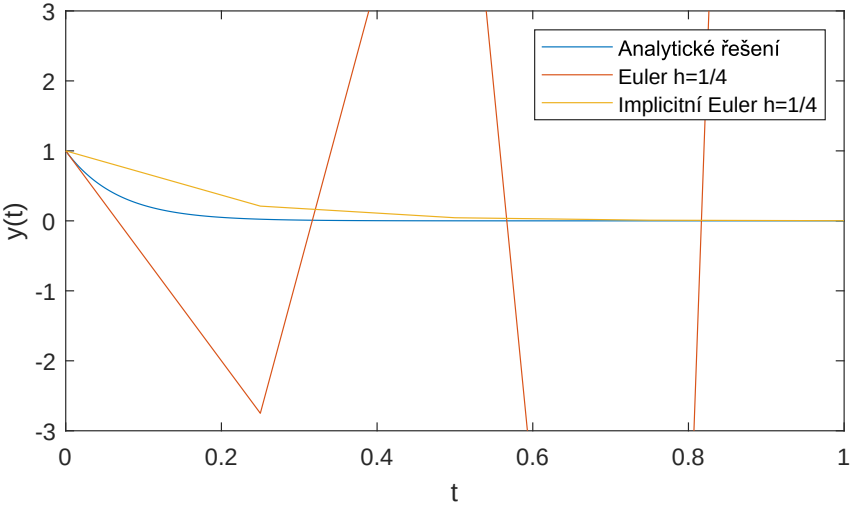
Implicitní a explicitní Eulerova metoda

```
clear all; close all;
% Euler, krok pouze 1/4
t0 = ...; tn = ...; y0 = ...; % Parametry
[t4e,y4e] = ode1(...); % Explicitní, viz minulý příklad
[t4i,y4i] = ode1i(...); % Implicitní, stejné parametry
% "Přesné" řešení
t = ...; % Viz minulý příklad
y = ...; % Viz minulý příklad
% Obrázek
f = figure(1);
plot(...); % Opět snad už umíme...
ylim([-3, 3]); % Abychom viděli okolí přesného řešení
legend(...);
xlabel(...);
ylabel(...);
```

Implicitní a explicitní Eulerova metoda

```
clear all; close all;
% Euler, krok pouze 1/4
t0 = 0; tn = 1; y0 = 1; % Parametry
[t4e,y4e] = ode1(@fstiff, t0, 1/4, tn, y0); % Explicitní
[t4i,y4i] = ode1i(@fstiff, t0, 1/4, tn, y0); % Implicitní
% "Přesné" řešení
t = linspace(0,1,1000);
y = exp(-15*t);
% Obrázek
f = figure(1);
plot(t, y, t4e, y4e, t4i, y4i);
ylim([-3, 3]);
legend('Analytické␣řešení', 'Euler␣h=1/4', 'Implicitní␣Euler␣h=1/4');
xlabel('t');
ylabel('y(t)');
```

Implicitní a explicitní Eulerova metoda



Dodatky

Lorenzův atraktor

Lorenzův atraktor je popsán soustavou

$$\begin{aligned} x'(t) &= a\,(y(t) - x(t)) \\ y'(t) &= x(t)\,(b - z(t)) - y(t) \\ z'(t) &= c\,z(t) + x(t)y(t) \end{aligned}$$

s parametry například $a = 10$, $b = 28$, $c = \frac{8}{3}$ a počátečními podmínkami například $x(0) = 5$, $y(0) = 5$, $z(0) = 5$.

Převeďte tento zápis do formy funkce `dsdt=lorenz(t,s)` se stavovým vektorem `s`, použitelné jako `odeset` v Matlabu, vyřeště pomocí `ode45` a vykreslete trajektorii.

Lorenzův atraktor

Funkce

```
function sdot = lorenz(t, s, a, b, c)
% LORENZ Stavové rovnice Lorenzova atraktoru
% Params e.g. a = 10, b = 28, c = 8/3.

    sdot = [ ...;
             ...;
             ... ];
end
```

Lorenzův atraktor

Funkce

```
function sdot = lorenz(t, s, a, b, c)
% LORENZ Stavové rovnice Lorenzova atraktoru
% Params e.g. a = 10, b = 28, c = 8/3.

    sdot = [ -a*s(1) + a*s(2);
              b*s(1) - s(2) - s(1)*s(3);
             -c*s(3) + s(1)*s(2) ];
end
```

Lorenzův atraktor

Funkce

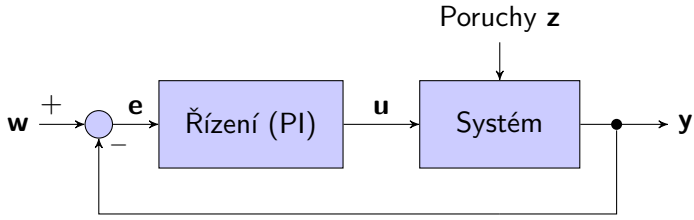
Vypočteme standardně pomocí `ode45`. Výstupem je matice o třech sloupcích, reprezentujících polohu stavového bodu dynamického systému v třírozměrném prostoru.

Vykreslíme pomocí `plot3`:

```
plot3(...);
```

Simulace obvodu s PI regulátorem

Budeme pracovat s modelem následujícího obvodu:

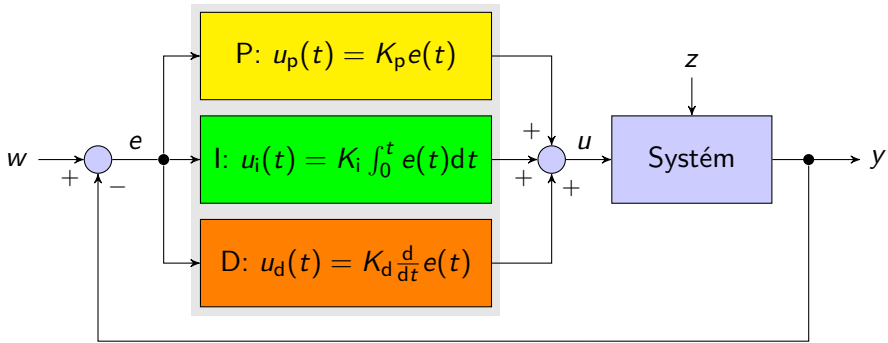


Vlastní regulovaný systém sestává z dvou aperiodických bloků prvního řádu, zapojených sériově. Poruchová veličina **z** představuje škálovaný jednotkový skok. Chceme zjistit, jak systém po poruše udrží požadovanou hodnotu výstupu **y**.

Simulace obvodu s PI regulátorem

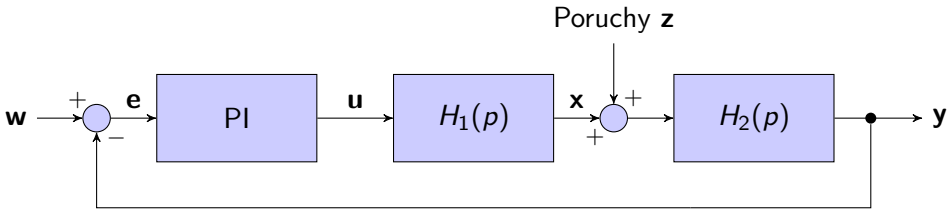
PID regulátor

Připomeňme si, že PI regulátor je ořezanou formou PID regulátoru, jehož zpětnovazebné zapojení pro jednu řídicí veličinu vypadá přibližně takto:



Simulace obvodu s PI regulátorem

Modelovaný obvod

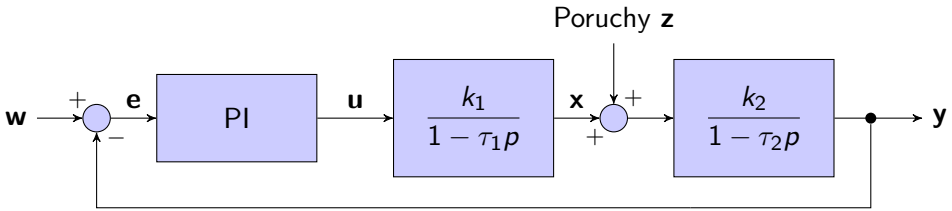


Obvod sestává z

- PI regulátoru s konstantami K_r a τ_i a ze
- dvou bloků prvního řádu s exponenciální odezvou, definovanou zesíleními k_1 a k_2 a časovými konstantami τ_1 a τ_2.

Simulace obvodu s PI regulátorem

Modelovaný obvod



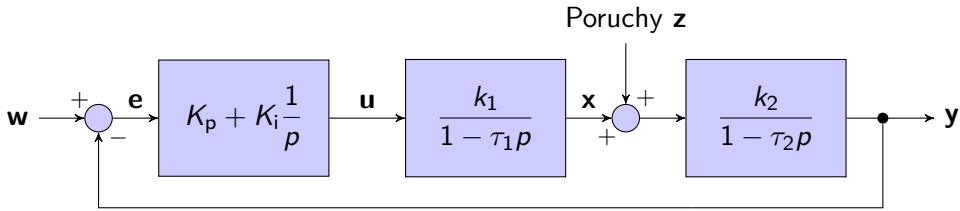
Bloky jsou reprezentovány funkcemi

$$-\tau_1 x'(t) + x(t) = k_1 u(t), \qquad -\tau_2 y'(t) + y(t) = k_2[x(t) + z(t)]$$

$$x'(t) = \frac{1}{\tau_1}[x(t) - k_1 u(t)] \qquad y'(t) = \frac{1}{\tau_2}[y(t) - k_2 x(t) - k_2 z(t)]$$

Simulace obvodu s PI regulátorem

Modelovaný obvod



PI regulátoru odpovídá rovnice

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\,\mathrm{d}\tau$$

$$u(t)' = K_p e'(t) + K_i e(t)$$

$$u(t)' = -(1 + K_p) y'(t) - \frac{1}{\tau_i} u(t)$$

Simulace obvodu s PI regulátorem

Soustava ODR

Model obvodu zahrnuje tři komponenty $x(t)$, $y(t)$ a $u(t)$:

$$x'(t) = \frac{1}{\tau_1}x(t) - \frac{k_1}{\tau_1}u(t) \qquad \text{výstup prvního bloku}$$

$$y'(t) = -\frac{k_2}{\tau_2}x(t) + \frac{1}{\tau_2}y(t) - \frac{k_2}{\tau_2}z(t) \qquad \text{výstup systému}$$

$$u'(t) = -(1 + K_\mathrm{p})y'(t) - \frac{1}{\tau_\mathrm{i}}u(t) \qquad \text{výstup PI řízení}$$

Simulace obvodu s PI regulátorem

Soustava ODR v maticové formě

Model obvodu je lineární a v maticové formě lze zapsat ve stavovém prostoru se stavovým vektorem $\mathbf{x} = (x, y, u)^\mathsf{T}$ jako

$$\mathbf{x}'(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{q}z(t) + \mathbf{r}e(t)$$

kde

$$\mathbf{A} = \begin{bmatrix} -1/\tau_1 & 0 & k_1/\tau_1 \\ k_2/\tau_2 & -1/\tau_2 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{q} = \begin{bmatrix} 0 \\ -k_2/\tau_2 \\ 0 \end{bmatrix}, \quad \mathbf{r} = \begin{bmatrix} K_\mathrm{r} k_2/\tau_2 \\ 0 \\ K_\mathrm{r}/\tau_\mathrm{i} \end{bmatrix}.$$

Q: Jaký přesně je vztah K_r (zesílení regulátoru), τ_i (časové konstanty integračního regulátoru) a konstant K_p a K_i?

Simulace obvodu s PI regulátorem

Soustava ODR v maticové formě

Model obvodu je lineární a v maticové formě lze zapsat ve stavovém prostoru se stavovým vektorem $\mathbf{x} = (x, y, u)^\mathsf{T}$ jako

$$\mathbf{x}'(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{q}z(t) + \mathbf{r}e(t)$$

kde

$$\mathbf{A} = \begin{bmatrix} -1/\tau_1 & 0 & k_1/\tau_1 \\ k_2/\tau_2 & -1/\tau_2 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{q} = \begin{bmatrix} 0 \\ -k_2/\tau_2 \\ 0 \end{bmatrix}, \quad \mathbf{r} = \begin{bmatrix} K_\mathrm{r} k_2/\tau_2 \\ 0 \\ K_\mathrm{r}/\tau_\mathrm{i} \end{bmatrix}.$$

Q: Jaký přesně je vztah K_r (zesílení regulátoru), τ_i (časové konstanty integračního regulátoru) a konstant K_p a K_i?
A: Dvě různé formulace PID: $K_\mathrm{p} = K_\mathrm{r}$, $K_\mathrm{i} = K_\mathrm{r}/\tau_\mathrm{i}$.

Simulace obvodu s PI regulátorem

Zápis soustavy ODR v Matlabu

Model soustavy uložíme ve formě $\mathbf{x}' = \mathbf{f}(t, \mathbf{x}, \dots)$ do `pictrl.m`:

```
function dxdt=pictrl(t,x,w,z,A,q,r) % w(tau), z(tau) jsou funkce!
  dxdt = ...;
end
```

Simulace obvodu s PI regulátorem

Zápis soustavy ODR v Matlabu

Model soustavy uložíme ve formě $\mathbf{x}' = \mathbf{f}(t, \mathbf{x}, \dots)$ do `pictrl.m`:

```
function dxdt=pictrl(t,x,w,z,A,q,r) % w(tau), z(tau) jsou funkce!
  dxdt = A*x + q*z(t) + r*(w(t)-x(2));
end
```

Bude třeba nějak převést `dxdt=pictrl(t,x,w,z,A,q,r)` na `dxdt=odefunc(t,x)`, abychom mohli použít odpovídající ODR řešiče v Matlabu.

Simulace obvodu s PI regulátorem

Zápis soustavy ODR v Matlabu

Model soustavy uložíme ve formě $\mathbf{x}' = \mathbf{f}(t, \mathbf{x}, \dots)$ do `pictrl.m`:

```
function dxdt=pictrl(t,x,w,z,A,q,r) % w(tau), z(tau) jsou funkce!
  dxdt = A*x + q*z(t) + r*(w(t)-x(2));
end
```

Bude třeba nějak převést `dxdt=pictrl(t,x,w,z,A,q,r)` na `dxdt=odefunc(t,x)`, abychom mohli použít odpovídající ODR řešiče v Matlabu.

Použijeme tzv. anonymní funkci:

```
A = ...; % zadáme prvky matice A
q = ...; % definujeme vektor q
r = ...; % definujeme vektor r
% Odkážeme se na funkce w(t)=piw(t) a z(t)=piz(t)
odefunc = @(t,y) pictrl(t,x,@piw,@piz,A,q,r); % A,q,r jsou konstanty
```

Simulace obvodu s PI regulátorem

Funkce $w(t)$ a $z(t)$

Funkce reprezentující řídicí vstup w a poruchu z jsou dány jako

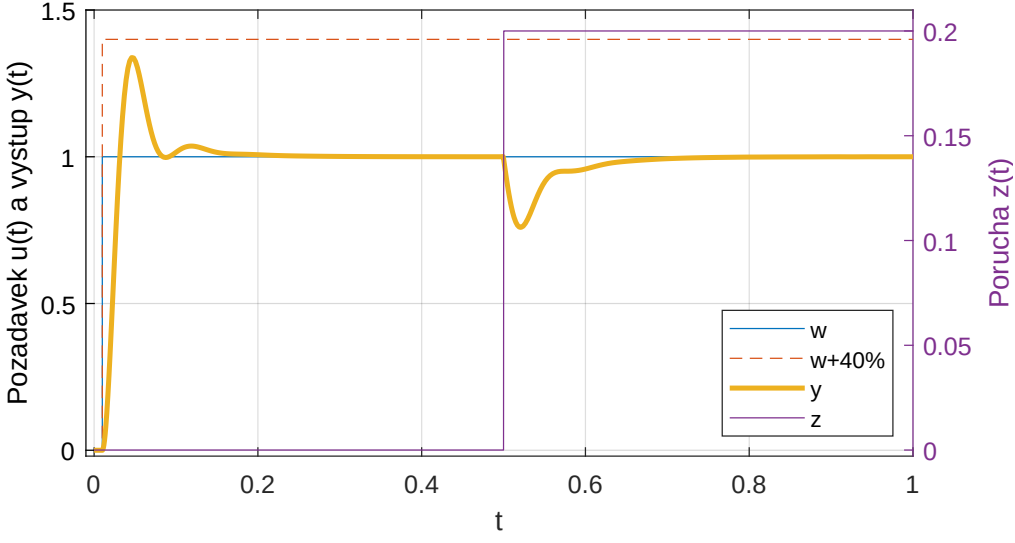
$$w(t) = \mathbb{1}(t - 0{,}005) \qquad z(t) = 0{,}2\,\mathbb{1}(t - 0{,}5)$$

a máte je k dispozici jako `piw()` a `piz()`.

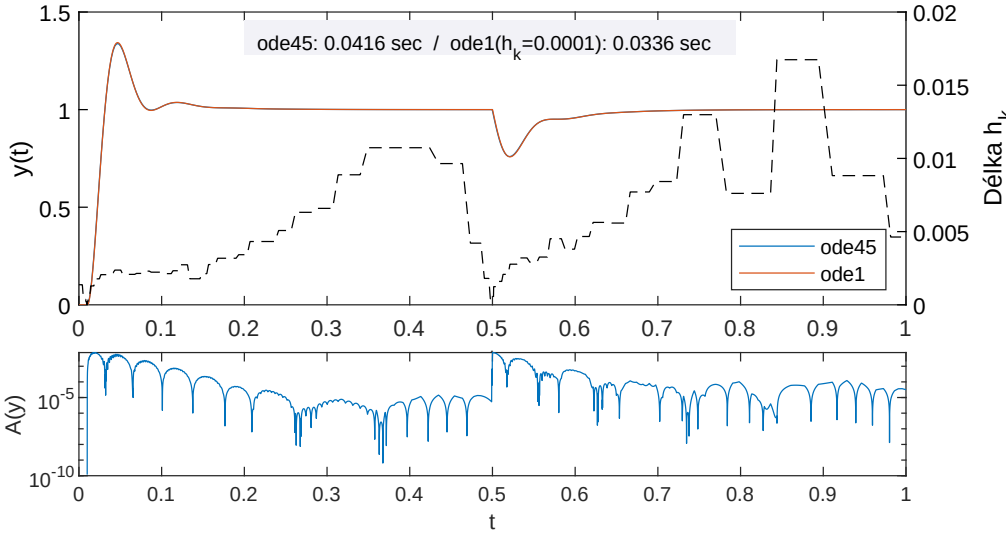
Pro urychlení práce máte také připraven k doplnění skript `pidemo.m`, v němž je třeba doplnit označené části kódu:

- Zkonstruovat matici **A** a vektory **q** a **r**
- Doplnit definici anonymní funkce `odefunc()`
- Určit odpovídající hodnoty $w(t)$ a $z(t)$ pro t dané výstupem `ode45()`

Vstup a výstup PI



Velikost kroku ODE45 a rozdíl s ODE1



Harmonické kmity

Použít a porovnat `ode1()`, `ode1i()`, `ode2h()`, `ode2m()`, `ode4()` a `ode45()` na řešení kmitů RLC obvodu v `harmonic.m`.

SIR model

Porovnat řešení SIR modelu pomoc `ode1()`, `ode1i()`, `ode2h()`, `ode2m()`, `ode4()` a `ode45()` a původního řešení z prvního cvičení.

Model epidemie je v `sirfunc.m`.