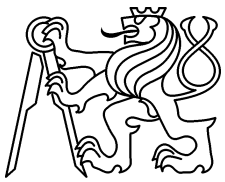
Jemný úvod do Matlabu a Simulinku

Částečně splněné požadavky na zápočet za 29932 sekund

B. Kovář, J. Přikryl, M. Pěnička, M. Vlček, L. Hodný



Obsah

1 Úvod

V tomto textu jsou shrnuty základní operace, funkce a principy práce v Matlabu a Simulinku, tak jak jsou vyžadovány na cvičeních předmětu Modelování systémů a procesů, přednášeného v letním semestru třetího ročníku. Pochopení ukázkových příkladů a samostatné zvládnutí neřešených pak stanovuje požadovanou míru znalostí nutných pro splnění praktické části zápočtu. Nedílnou součástí textu jsou odkazy na doporučená skripta [1].

2 Matlab

Matlab je programový systém společnosti The MathWorks, Inc. Název souboru je zkratka "MATrix LABoratory" a výstižně charakterizuje způsob práce s programem. Původně Matlab sloužil jako interface na metody numerických knihoven LINPACK a EISPACK. Později se stal komerčním produktem a pro svou jednoduchost se stal průmyslovým standardem v oblasti softwarových řešení automatického řízení (CADCS).

2.1 Přednosti Matlabu

Tím, že se Matlab stal světovým standardem v mnoha inženýrských oborech, je rozšířen po celém světe a existuje pro všechny operační systémy (Unix, Linux, Solaris, Windows, Mac OS). Matlab je možno relativně pohodlně rozšiřovat o další funkce a na Internetu je možné získat několik desítek toolboxů. S programem je možné začít pracovat i bez speciálních znalostí programování (nicméně, elementární znalosti programování jsou výhodou). V systému Matlab jsou zabudovány robustní numerické metody. Matlab je možné dále rozšiřovat o externí programy (MEX-soubory) naprogramované v C/C++ nebo Fortranu.

2.2 Příkazy a proměnné

Příkazy jsou v Matlabu ve tvaru *proměnná = výraz*. Výrazy se skládají z operátorů, speciálních znaků, funkcí a proměnných. Pokud chybí přiřazení proměnné, zavede se systémová proměnná *ans*. Umístěním středníku za výrazem potlačíme výstup na obrazovku. Mezery uvnitř výrazu jsou nepodstatné. Výsledkem je obecně matice, která se zobrazi na obrazovce. Jména proměnných a funkcí musí začínat písmenem. Matlab **rozlišuje malá a velká písmena** ve jménech proměnných, funkcí a konstant.

Příklad:

```
>> 1320 / 63
>> ans =
         20.9524
>> a = 1 + 1;
>> a = a + 1
>> a =
       3
```

2.3 Čísla a aritmetické výrazy

Matlab provádí všechny výpočty v dvojité přesnosti (pokud není definováno jinak). Priorita operací je stejná jako v matematice. Okolo symbolů *e,E* nesmí být mezera (např. 3.14159, 6.6345e23, 9.12E-20). V Matlabu je možné používat následující operátory:

- + sčítání,
- - odčítání,
- * násobení,
- / pravé dělení,
- \ levé dělení,
- ∧ umocňování
- () závorky upravují pořadí provádění operací.

2.4 Manipulace s maticemi a vektory

2.4.1 Generování vektoru

Vektor v Matlabu zadáváme výčtem prvků nebo definicí prvku vektoru na dané pozici (kapitola 2.4.3). Pro generování vektoru lze také použít notaci s dvojtečkou. Syntaxe je: `vektor = od : <krok> : do`. Tento způsob je vhodný pro zápis aritmetických posloupností. Transpozici vektoru provedeme takto: `u = u'`.

Příklad:

```
>> u = [1 2 3 4 5] % výčet prvků
>> x = 1:5 % notace s dvojtečkou
>> x =
       1  2  3  4  5
>> y = 0:pi/4:pi
>> y =
       0.0000 0.7854 1.5708 2.3562 3.1416
```

Cvičení 1:
1. Definujte vektor $\vec{a}$, který obsahuje prvky $3, 4, 5, 6, 7, 8$.
2. Definujte vektor $\vec{a}$, který obsahuje prvky $8, 7, 6, 5, 4, 3$.
3. Definujte vektor $\vec{a}$, který obsahuje pouze sudá čísla od 1 do 20.

2.4.2 Generování tabulky

Příklad:

```
>> x = [0.0:0.1:0.5]';
>> y = exp(-x).*cos(x);
>> [x y]
>> ans =
              0  1.0000
```

```
0.1000   0.9003
0.2000   0.8024
0.3000   0.7077
0.4000   0.6174
0.5000   0.5323
```

Cvičení 2:
1. Nadefinujte vektory $\vec{u}, \vec{v}, \vec{w}$ stejné délky a vytvořte z nich tabulku, kde v prvním sloupci bude vektor $\vec{u}$, ve druhém ,$\vec{v}$, a ve třetím $\vec{w}$.
2. Vytvořte tabulku, goniometrických funkcí od -2π do 2π.

2.4.3 Čtení a zápis prvku vektoru na dané pozici

K prvkům vektoru v Matlabu přistupujeme pomocí indexů, podobně jako v jiných programovacích jazycích. Na vektor se můžeme dívat jako na alokované pole, které má na rozdíl od jiných programovacích jazyků první index roven 1 a ne 0. Vektor můžeme tedy deklarovat i tím, že definujeme prvky na jednotlivých pozicích.

Příklad:

```
>> a(1) = 1; a(3) = 5; a(7) = 3;
>> a
>> ans =
         1 0 5 0 3
```

Čtení hodnoty vektoru na dané pozici provedeme obdobně přiřazením.

Příklad:

```
>> u = [1 3 5 7];
>> x = u(2)
x =
    3
```

2.4.4 Generování matice

Matice v Matlabu vytváříme obdobně jako vektory. Výčtem prvků, případně zápisem hodnoty prvku na danou pozici. Hodnoty matice čteme opět obdobně jako u vektorů. První index je řádkový, druhý index sloupcový.

1. Odkaz na konkrétní prvek matice

```
>> a(3,3) = a(1,3) + a(3,1);
```

2. Definice matice (5,1), která se skládá z prvků A(1,3), A(2,3), ..., A(5,3)

```
>> A(1:5,3);
```

3. Definice matice (5,4), která se skládá z prvních pěti řad a sloupců 7 až 10 matice A

```
>> A(1:5,7:10);
```

4. Všechny prvky 3. sloupce matice A

```
>> A(:,3);
```

5. Nahrazení 2., 4. a 7. sloupce matice A prvními třemi sloupci matice B

```
>> A = [1 2 3 4 5 6 7 8 9; ...
        1 2 3 4 5 6 7 8 9; ...
        1 2 3 4 5 6 7 8 9];
>> B = [0 0 0 13 13; ...
        0 0 0 13 13; ...
        0 0 0 13 13];
>> A(:,[2 4 7]) = B(:,1:3)
>> A =
       1 0 3 0 5 6 0 8 9
       1 0 3 0 5 6 0 8 9
       1 0 3 0 5 6 0 8 9
```

6. Prohození sloupců matice A

```
>> A = A(:,9:-1:1);
>> A =
       9 8 0 6 5 0 3 0 1
       9 8 0 6 5 0 3 0 1
       9 8 0 6 5 0 3 0 1
```

7. Převod matice na vektor

```
>> A = [1 2; 3 4; 5 6];
>> B = A(:)'
>> B =
       1 3 5 2 4 6
```

8. Prázdná matice. Prázdná matice `x=[]` má nulový rozměr. Vymazání 1. a 3. sloupce matice A se tedy provede

```
>> A(:,[1 3]) = [];
```

Cvičení 3:
1. Je dán vektor $\vec{u} = \{9\ 11\ 13 \cdots\ 97\}^T$. Jak tento vektor nejefektivněji zadáte.
2. Zadejte matici A, s dimenzí 8×7, která má všechny prvky nulové, pouze prvky $[1, 4]$ a $[4, 6]$ jsou rovny 1.
3. Zadejte libovolnou matici A. Jak vytvoříte vektor $\vec{u}$, který bude obsahovat všechny prvky druhého sloupce matice A.
4. Zadejte libovolné matici A a B se stejnou dimenzí. V matici A nahraďte první sloupec, posledním sloupcem matice B.

2.4.5 Komplexní čísla a matice

Komplexní čísla jsou povolena ve všech operacích a funkcích Matlabu. Pokud i nebo j používáme jako proměnné, je třeba nadefinovat jinou imaginární jednotku, např.

```
>> ii = sqrt(-1);
>> z = 2 + 3*ii;
```

Zápis matice s komplexními prvky se tedy provádí:

```
>> A = [1 2;3 4] + i*[-5 6; 7 -8];
>> A = [1-5*i 2+6*i; 3+7*i 4-8*i];
```

2.4.6 Operace s maticemi

Zápis matic je obdobný jako v matematice. Základní operace s maticemi jsou:

1. Transpozice `B = A'`

```
>> A = [1 2;3 4]
>> A =
       1 2
       3 4
>> B = A'
>> B =
       1 3
       2 4
```

2. Součet a rozdíl. Matice musí mít stejný rozměr $(m, n) + (m, n) = (m, n)$

```
>> C = A + B;
>> C = A - B;
```

3. Součin. Matice musí mít příslušný rozměr $(m, n) * (n, k) = (m, k)$. Jedná se o vektorový součin, jehož výsledkem je skalár.

```
>> C = A * B;
```

4. Podíl. V Matlabu rozlišujeme pravé dělení `C = A/B`, které je definováno `A*inv(B)` a levé dělení `C = A\B`, definovené `inv(A)*B`. Pravé dělení je také možné definovat jako `B/A = (A'\B')'`.

5. Umocňování `C = A^p`, resp. `C = p^A` je definováno, pokud `A` je čtvercová matice a `p` skalár.

2.5 Operace s poli

Tyto operace odpovídají operacím "prvku s prvkem". Operace jsou stejné jako s maticemi. Jedná se o tyto operátory: `.' .* .\ ./ .^`

Příklad:

```
>> x = [1 2 3], y = [4 5 6];
>> z = x .\ y
>> z =
       4.0000 2.5000 2.0000
>> z = x .^ y
>> z =
       1   32   729
```

Cvičení 4:
1. Zadejte matici A a B. Jak provedete vektorový součin? Jaké musí být dimenze matice A a B?
2. Od všech prvků matice A odečtěte konstantu 3.
3. Je dán vektor $\vec{u} = \{1\ 3\ 5\ 7\}$ a vektor $\vec{v} = \{2\ 4\ 6\ 8\}^T$. Jak vypočtete součin vektoru $\vec{u}$ a vektoru $\vec{v}$. Jak by součin vypadal v případě operace "prvek s prvkem".

2.6 Relace

Relace je možné provádět mezi maticemi stejných rozměrů. Porovnání je prováděno mezi odpovídajícími si prvky. Výsledkem je matice samých 0 a 1. Hodnota 1 odpovídá pravdivé relaci, hodnota 0 nepravdivé. Relační operátory jsou:

- `<` menší než,
- `<=` menší nebo rovno,
- `>` větší než,
- `>=` větší nebo rovno,
- `==` rovno,
- $\sim$`=` nerovno,

Příklad:

```
>> x = [1 4 8], y = [-4 5 0];
>> x < y
>> ans =
         0   1   0
```

Cvičení 5:
1. Je dána matice `A=[1 2 3; 4 5 6; 1 2 3]`. Jak zjistíte, zda jsou hodnoty prvků v prvním řádku matice `A` menší než hodnoty prvků ve třetím sloupci matice A. Zapište výsledek této operace.
2. Je dán vektor $\vec{u} = \{1\ 2\ 3\}$. Výsledkem operace $\vec{u} \neq \vec{v}$ je vektor $\{1\ 0\ 1\}$. Jak zadáte libovolný vektor $\vec{v}$, který odpovídá zadané operaci.
3. Je dán vektor $\vec{u} = \{2\ 2\ 3\}$ a vektor $\vec{v} = \{1\ 2\ 1\}$. Napište relační operátor, pomocí kterého zjistíte, zda je vektor $\vec{u}$ roven vektoru $\vec{v}$. Zapište výsledek této operace.

2.7 Logické operace

Logické operace lze provádět mezi maticemi stejných rozměrů. Logické operace jsou prováděny mezi prvky matic, které jsou obvykle 0 nebo 1. Logické operátory jsou:

- `&` konjunkce (AND),
- `|` disjunkce (OR),
- $\sim$ negace (NOT),

V Matlabu jsou zabudovány ještě další relační a logické funkce, např. `any, all`.

2.8 Řetězce

Řetězce jsou uzavřeny v apostrofech a ukládány jako vektory.

Příklad:

```
>> x = 'modelovani'
>> x =
       modelovani
>> x = [x, ' systemu a procesu']
>> x =
       modelovani systemu a procesu
>> size(x)
>> ans =
          1  27
```

2.9 Informace o proměnných

Informace o proměnných a alokované paměti udávají funkce `who` a `whos`. Alokované proměnné je možné uložit do souboru a později načíst k dalšímu zpracování pomocí funkcí `save` a `load`. Vymazání proměnné provádí příkaz `clear`.

Příklad:

```
>> x = [1 2 3; 4 5 6; 7 8 9]
>> x =
       1  2  3
       4  5  6
       7  8  9
>> save moje_matice.mat x;
>> clear;
>> load moje_matice.mat;
>> x
>> x =
       1  2  3
       4  5  6
       7  8  9
```

2.10 Help

Matlab obsahuje velmi kvalitní nápovědu v HTML a PDF. Často je mnohem efektivnější použít zabudovaný help, který poskytuje dostatek informací o všech objektech Matlabu.

Příklad:

```
>> help plot
```

2.11 Funkce v Matlabu

Matlab má v současné době několik stovek interních funkcí. Mezi ty nejdůležitější (a na cvičeních nejpoužívanější) patří:

Obecné funkce

`help`	on-line nápověda
`who`	seznam proměnných
`size`	dimenze matice
`length`	délka vektoru
`^C`	ukončení výpočtu
`chdir`	změna adresáře
`save`	uložení proměnné do souboru
`load`	načtení proměnné ze souboru
`clear`	de-alokace proměnných

Matematické funkce

Goniometrické a ostatní matematické funkce jsou definovány stejně jako v matematice (např. `sin, cos, sqrt, exp`).

Ostatní funkce

`clc`	vymaže obrazovku
`echo`	zobrazí komentář
`plot`	grafický výstup
`ones(m,n)`	jednotková matice (m,n)
`zeros(m,n)`	nulová matice (m,n)
`rand(m,n)`	matice (m,n) náhodných čísel
`roots`	výpočet vlastních čísel

Funkce jsou buď vnitřní, nebo se jedná o tzv. M-funkce, které jsou uloženy na disku. Uživatel může vytvářet vlastní M-funkce (pomocí příkazů Matlabu, nebo pomocí mex-funkcí naprogramovaných jazykem C) a tím rozšiřovat funkčnost Matlabu.

Funkce mohou mít více vstupních a výstupních parametrů. Příklad zápisu funkce s jedním výstupním parametrem je `y=sin(x)`. Funkce však může vracet i více parametrů (např. `[m,n]=size(X)`).

2.12 Grafické funkce

Visualizace dat je silnou stránkou Matlabu. Základní operace s grafy jsou velmi jednoduché a snadno zapamatovatelné. Pokročilejší práce s grafy, jejich nastavení a export do různých grafických formátů je možné nalézt v nápovědě. Základní příkaz pro tvorbu grafů je `plot`. Tento příkaz vykreslí čárový graf, který spojuje body se souřadnicemi $[x, y]$. Souřadnice jsou definovány vektory x a y.

Příklad:

```
>> t = pi*(0:0.02:2);
>> plot(t,sin(t))
```

O tom, že se jedná skutečně o čárou spojené body se můžete přesvědčit příkazem

```
>> plot(t,sin(t),'o-')
```

Pokud nyní necháme vykreslit červenou barvou funkci $cos(t)$,

```
>> plot(t,cos(t),'r')
```

předchozí křivka bude vymazána. Pro přidání křivky do již existujícího grafu použijeme příkaz `hold`.

Příklad:

```
>> plot(t,sin(t),'b')
>> hold on
>> plot(t,cos(t),'r')
```

Mezi další užitečné příkazy pro práci s grafy patří například:

Příkaz	Funkce
`figure`	otevře nové grafické okno
`subplot`	více grafů v jednom grafickém okně
`semilogx`, `semilogy`, `loglog`	logaritmické měřítko osy x a y
`legend`	legenda pro grafy s více funkcemi
`print`	tisk na tiskárně

Tabulka 1: Další funkce pro tvorbu grafů

Příklad:

```
>> t = 2*pi*(0:0.01:1);
>> plot(t,sin(t))
>> xlabel('t')
>> ylabel('amplituda')
>> title('Jednoduchý harmonický oscilátor')
```

Obdobně jako vykreslujeme grafy funkcí $y = f(x)$ (2D grafy), můžeme v Matlabu tvořit grafy funkcí $y = f(x, y)$ (3D grafy).

Příklad:

```
>> x = pi*(0:0.02:1);
>> y = 2*x;
>> [X,Y] = meshgrid(x,y);
>> surf(X,Y,sin(X.2+Y))
```

Grafické výstupy Matlabu můžeme nejen zobrazovat na obrazovce, tisknout na tiskárně, ale i ukládat do souborů pro další použití v jiných (např. publikačních) programech. Graf, jako obrázek, uložíme příkazem `saveas(figure,jmeno)`. Podporovány jsou tyto grafické formáty – bitmapové GIF, JPEG, PNG, TIFF, BMP a vektorové EPS a WMF. Obrázek je ukládán do právě používaného adresáře.

Příklad:

```
>> figure(1)
>> t = pi*(0:0.02:2);
>> plot(t,sin(t))
>> saveas(1,'sinus.bmp');
```

2.13 M-soubory (M-files)

Příkazy Matlabu jsou prováděny buď ihned (v command window) nebo mohou být uloženy do m-souboru, do textového souboru s příponou .m, a prováděny sekvenčně. Příkazy m-souboru se mohou odkazovat na jiné m-soubory, případně rekurzivně volat sami sebe. M-soubory jsou typu *script* nebo *m-funkce*.

2.13.1 Script soubory

Scriptové soubory jsou sekvencemi příkazů ve kterých jsou všechny proměnné globální.

Příklad:

```
x = [-pi:0.1:pi]';
y = sin(x);
plot(y)
```

2.13.2 M-funkce

M-funkce rozšiřují možnosti Matlabu, protože umožňují definovat nové funkce. Proměnné jsou uvnitř funkce lokální a s prostředím Matlabu komunikují pomocí vstupních a výstupních parametrů. Nová funkce se definuje příkazem `function` a seznamem vstupních a výstupních parametrů (bez tohoto příkazu by se jednalo o script soubor).

Příklad:

```
function y=prumer(x)
% vstupni parametr vektor x
% vystupni parametr aritmeticky prumer
% pouziti y=prumer(x)
y=0;
soucet=0;
for i=1:length(x)
    soucet=soucet+x(i);
end
y=soucet/length(x);
```

2.13.3 Mex-funkce

Z Matlabu lze volat i externí funkce napsané v jazycích C nebo Fortran. Funkce naprogramovaná v jazyce C se skládá ze dvou částí. První, tzv. gateway interface, zajišťuje komunikaci s Matlabem a je obdobou *main()* funkce v jazyce C. Druhá část je vlastní kód. Výhodou mex funkcí je jejich mnohonásobně vyšší rychlost oproti m-funkcím.

2.14 Programování M-souborů

Při programování m-souborů je často požadováno opakované provádění skupiny příkazů. V případě, že je počet opakování dán, je vhodné použít příkaz `for`.

```
for <parametr>=<vyraz>
    <prikazy>
end
```

Příkazy cyklu se mohou vkladat do sebe, každý příkaz `for` je ukončen příkazem `end`.

Příklad:

```
for i=1:m
    for j=1:n
        x(i,j)=0;
        y(i,j)=0;
    end
end
```

V případě, že počet opakování není předem dán, je možné použít příkaz `while`. Skupina příkazů je prováděna tolikrát, dokud je splněna logická podmínka.

```
while <podminka>
    <prikazy>
end
```

Příklad:

```
n=1;
while prod(1:n) < 1000
    n=n+1;
end;
```

Příkaz `if` umožňuje větvení programu podle logické podmínky.

```
if <podminka>
    <prikaz>
elseif <podminka>
    <prikaz>
    ...
else
    <prikaz>
end
```

Příklad:

```
j=a;
for i=1:n
    if X(i,j)>0
       m=m+1;
    elseif X(i,j)<0
       m=m-1;
    else
       break;
    end
end
```

Příkaz `break` umožňuje přerušit provádění `for` nebo `while` cyklů. Příkaz `return` předá řízení volající funkci, případně umožní návrat do command window. Příkaz `input` umožňuje vstup dat z obrazovky.

```
<promenna> = input('<text>'[,'s'])
```

Příklad:

```
jmeno=input('Vase jmeno:','s');
rok=input('Rok narozeni:');
```

Příkaz `eval` interpretuje znakový řetězec.

Příklad:

```
string='clc,clear,a=[1 2;3 4],...
b=[0 0; 1 0], c=a*b'
eval string
a =
     1     2
     3     4
b =
     0     0
     1     0
c =
     2     0
     4     0
```

2.15 Koeficienty Čebyševových polynomů

Čebyševovy polynomy mohou být odvozeny takto:
Nechť $x = cos(\varphi), |x| \leq 1$ a předpokládejme, že existují polynomy $T_k(x) = cos(k\varphi)$, $cos(\varphi) = x$. Řešením diferenciální rovnice, pro $|x| \leq 1$,

$$(1 - x^2)y'' - xy' + n^2 y = 0, \quad y = T_n(x), \quad n \geq 0 \tag{1}$$

dostaneme tyto polynomy:

$$\begin{aligned} T_0(x) &= 1, \\ T_1(x) &= x, \\ T_2(x) &= 2x^2 - 1, \\ T_3(x) &= 4x^3 - 3x, \\ T_4(x) &= 8x^4 - 8x^2 + 1, \\ T_5(x) &= 16x^5 - 20x^3 + 5x \ldots, \end{aligned} \tag{2}$$

Tyto polynomy splňují rekurentní rovnici

$$\begin{aligned} T_0(x) &= 1, \\ T_1(x) &= x, \\ T_{k+1}(x) &= 2xT_k(x) - T_{k-1}(x), \ k = 1, \ldots, n, \end{aligned} \tag{3}$$

Model v Matlabu

```
function F=T(n)
% ************************************
% * Koeficienty Cebysevovych polynomu *
% *                                   *
% ************************************
pocitadlo = 1;
```

```
arg = [1 0];
a = [1];
b = [1 0];
shift = [0 0 1];
if n == 0
   F = a;
elseif n == 1
   F = b;
else
   while pocitadlo < n
      c = conv(2*arg,b) - conv(shift,a);
      a = b;
      b = c;
      pocitadlo = pocitadlo + 1;
   end
   F = c;
end
```

2.16 Jednoduchý dynamický model Nabídka-Poptávka

Vyjděme z ekonomického modelu, kdy nabídka v n-tém časovém intervalu $s(n)$ je přímo úměrná ceně produktu v minulém časovém intervalu $p(n-1)$ a naopak poptávka v n-tém časovém intervalu $dem(n)$ klesá se současnou cenou produktu $p(n)$, to znamená, že je úměrná $-dp(n)$. Zavedeme-li proměnnou $u(n)$, která charakterizuje počet vyrobených kusů v čase n, můžeme systém popsat soustavou tří diferenčních rovnic.

$$\begin{aligned} s(n) &= c \cdot p(n-1) + a \cdot u(n), \\ dem(n) &= -d \cdot p(n) + b \cdot u(n), \\ s(n) &= p(n). \end{aligned} \tag{4}$$

Pro obecně zvolené parametry a, b, c, d a $u(n) = 1(n)$ můžeme v Matlabu vytvořit následující model.

Model v Matlabu

```
% ********************************
% * Nabídka a poptávka           *
% * (jednoduchý dynamický model) *
% ********************************
clear
a = 100;
b = 120;
c = 3;
d = 4;
N = 50;
% vstupní posloupnost
u = ones(1,N);
```

```
p(1) = (b-a)/d;
% rovnice pro cenu p(n)
for k = 2:N
    p(k) = -c/d*p(k-1)+(b-a)/d*u(k);
end
figure(1)
subplot(3,1,1), stem(p)
% nabídka s(n)
for m = 1:N
    s(m) = c*p(m) + a*u(m);
end
subplot(3,1,2), plot(p,s,'*')
% poptávka dem(n)
for m = 1:N
    dem(m) = -d*p(m) + b*u(m);
end
subplot(3,1,3), plot(p,dem,'o',p,s,'*')
```

3 Simulink

Simulink, součást Matlabu, je nástroj pro modelování a analyzování dynamických systémů. Podporovány jsou lineární i nelineární systémy, spojité i diskrétní modely, případně jejich kombinace.

3.1 Úvod do Simulinku

Modelování probíhá v grafickém uživatelském prostředí (GUI) převážně pomocí myši. Model se skládá z blokových diagramů, které v tomto prostředí vytváříme obdobně, jako bychom je kreslili tužkou na papír.

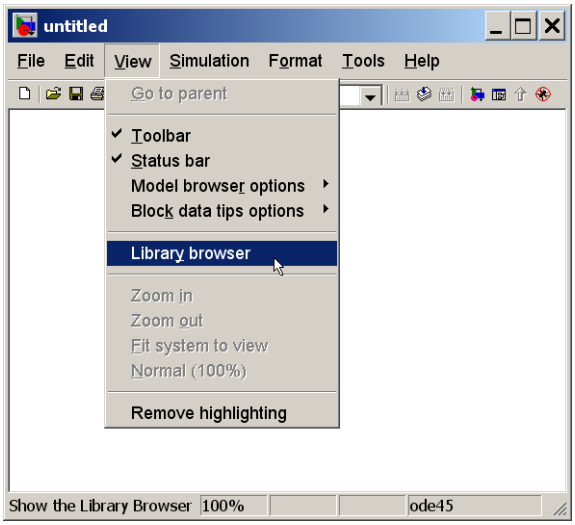
Součástí Simulinku je obsáhlá knihovna různých vstupů, výstupů, lineárních a nelineárních komponent a konektorů umožňujících spojovat jednotlivé bloky. Stejně jako v případě Matlabu je možné vytvářet vlastní bloky (vytvářením subsystémů z existujících bloků, přípaně naprogramováním nových bloků v jazyce C pomocí S-funkcí).

Po vytvoření modelu je možno z integrovaného menu spustit jeho simulaci. Použitím výstupních bloků je možné zobrazovat, měnit parametry a analyzovat výsledky během simulace. Stejně tak je možné ukládat výsledky do proměnných Matlabu a dále je zpracovávat, případně použít pro vizualizaci. Protože Simulink je integrován v Matlabu, je možné simulovat, analyzovat a upravovat modely v obou prostředích.

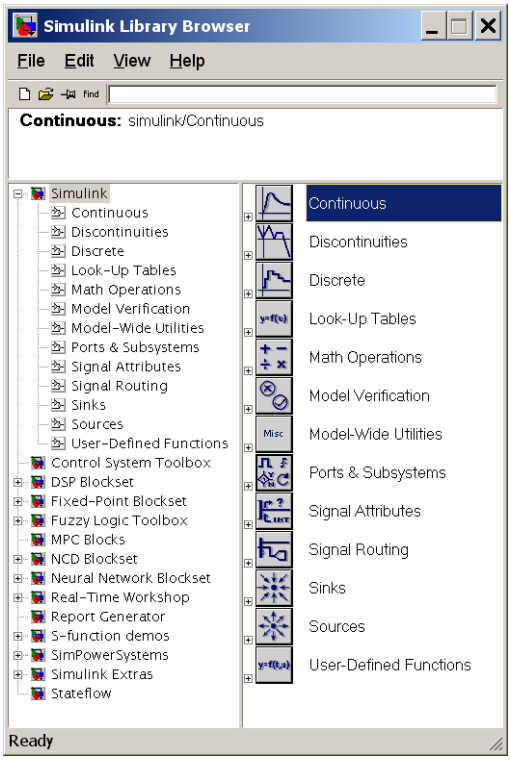
3.1.1 Spuštění Simulinku

Okno pro vytváření modelů (viz obrázek 1) v Simulinku otevřeme pomocí menu
`File`→`New`→`Model`,
knihovnu stavebních bloků (viz obrázek 2) pak zobrazíme z tohoto okna pomocí
`View`→`Library browser`.
Alternativně lze téhož efektu dosáhnout příkazem `simulink` z příkazové řádky Matlabu

(v mém případě tento příkaz ovšem pouze otevře okno s knihovnou stavebních bloků Simulinku).



Obrázek 1: Okno pro kreslní modelu v Simulinku



Obrázek 2: Knihovna stavebních bloků pro Simulink

3.1.2 Jednoduché příklady

Nejjednodušší příklad: zobrazení funkce $\sin(t)$ pomocí bloku zdroje `Sine Wave` a bloku výstupu `Scope`. Vyzkoušejte si ovládání parametrů bloků: nastavení délky periody, nastavení amplitudy, nastavení fáze, automatické zobrazení celého grafu v případě, že překračuje implicitně nastavené meze.

Jak pomocí zdroje `Sine Wave` získám funkci $\cos(t)$? Pomocí bloku `Mux` zobrazte oba dva průběhy v jednom grafu. Zobrazte součet $\sin(t) + \cos(t)$, zobrazte násobek dvou funkcí.

3.1.3 Kružnice

Vyzkoušejte si, co se stane při použití bloku výstupu `XYGraph` a dvou vstupů `Sine Wave`, z nichž jeden simuluje funkci $\sin(t)$ a druhý funkci $\cos(t)$. Proč je výstupem simulace jednotková kružnice?

Cvičení 6:
Namodelujte systém zobrazující rovnici kružnice s poloměrem r, danou rovnicemi

$$\begin{aligned} x &= r\ \sin t \\ y &= r\ \cos t \end{aligned}$$

Cvičení 7:
Namodelujte systém zobrazující logaritmickou spirálu

$$\begin{aligned} x &= e^{-kt}\ \sin t \\ y &= e^{-kt}\ \cos t \end{aligned}$$

Jak byste nasimulovali systém generující obraz *archimedovy spirály*, případně *asteroidy*? Asteroida je popsána soustavou rovnic

$$\begin{aligned} x &= \sin^3 t \\ y &= \cos^3 t \end{aligned}$$

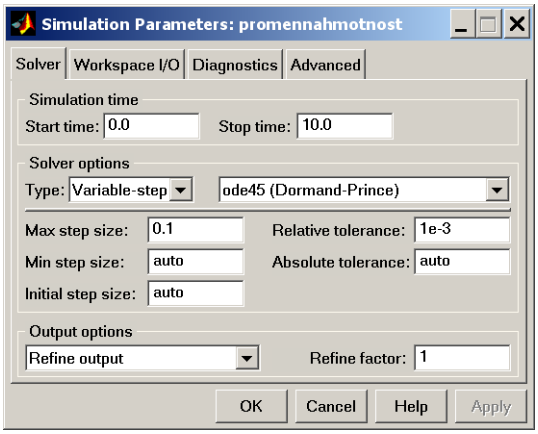
a Archimedova spirála je popsána soustavou

$$\begin{aligned} x &= t\ \sin t \\ y &= t\ \cos t \end{aligned}$$

3.1.4 Parametry simulace

Pokud se vám v předchozích příkladech zobrazovaly jenom podivné hranaté tvary, nijak nepřipomínající to, co jste předpokládali, že uvidíte, je to patrně tím, že simulace probíhala buď příliš krátce, nebo byl časový krok simulace příliš velký.

Okno pro nastavení parametrů simulace (viz obrázek 3) lze otevřít buď klávesovou zkratkou `Ctrl+E` nebo z menu v okně modelu pomocí volby `Simulation`→`Simulation parameters...`.



Obrázek 3: Nastavení parametrů simulace

V tomto okně můžete nastavit, jestli simulace probíhá s proměnným časovým krokem, či s krokem pevným, a parametry časového kroku – počáteční krok (tedy hodnotu, s níž simulace začíná), minimální a maximální časový krok (pro simulaci s proměnným časovým krokem). Volbou položek `Start time` a `Stop time` volíte délku doby simulace. Vyzkoušejte si to.

3.1.5 Nabídka-poptávka

V příkladu 5.2 na stranách 99–100 je ukázán model ekonomického systému vývoje ceny na základě nabídky a poptávky. Ve skriptu se tento příklad řeší pomocí $\mathcal{Z}$-transformace, ale na obrázku 5.12 můžete vidět model původní diferenční rovnice, z níž model vychází (viz též odvození rovnice (2.8) na straně 14). Ve skriptu znázorněný model počítá s pevnými hodnotami – upravte jej pro uživatelem zadané hodnoty parametrů a, b, c a d.

3.1.6 Fakulta

Základní model předpovídající počet studentů absolvujících studium například na FD ČVUT je opět popsán ve skriptu v příkladu 5.3 na stranách 100–103. Upravte tento model tak, aby zohledňoval fakt, že po nedokonční první etapy studia někteří studenti končí, a že do každého vyššího ročníku mohou přistoupit studenti z jiných škol.

3.2 Modelování v Simulinku

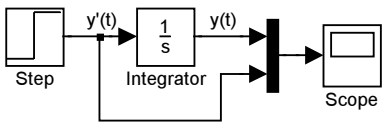
Většinu podkladů nalezenete ve skriptech [1]. Cílem této kapitoly je seznámit vás s modelováním jednoduchých dynamických systémů popsaných vnějším a vnitřním popisem.

Pro samostudium doporučujeme kapitoly 3.3 (stránky 54–59), a v nich probrané příklady a příklady 5.4 a 5.5 na stranách 103–105.

3.2.1 Integrátor a jednotkové zpoždění

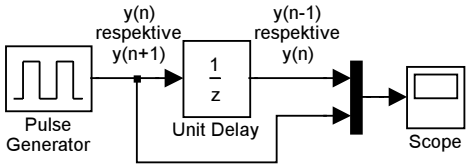
Při simulaci spojitých systémů využíváme bloku `Integrator`, jenž, jak jeho název napovídá, integruje vstupní signál v čase. Funkci bloku demonstruje jednoduché schéma znázorněné na obrázku 4. Přivedeme-li na vstup integrátoru signál $a(t)$, bude výstup

tohoto bloku odpovídat signálu $\int a(\tau)d\tau$, a obráceně: je-li na vstupu $\frac{da(t)}{dt}$, na výstupu bloku bude $a(t)$.



Obrázek 4: Jednoduchý příklad ukazující funkci integrátoru.

Při simulaci diskrétních systémů pak využíváme bloku `Unit Delay`, jenž vzorkuje vstupní signál každou celou časovou jednotku simulace a navzorkovanou hodnotu v čase o jednotku zpozdí – to, co jsme přivedli na vstup toto bloku v okamžiku $t = \nu$ nalezneme na výstupu tohoto bloku až v čase $n = \nu+1$ a tato hodnota bude na výstupu pro všechna $n < \nu + 2$. Funkci bloku demonstruje schéma znázorněné na obrázku 5. Přivedeme-li na vstup integrátoru signál $a(n)$, bude výstup tohoto bloku odpovídat signálu $a(n-1)$, a obráceně, je-li na vstupu $a(n+1)$, na výstupu obdržíme $a(n)$.



Obrázek 5: Jednoduchý příklad ukazující funkci bloku `Unit Delay`,

3.2.2 Diferenciální rovnice

Vytvořme jednoduchý model systému popsaného diferenciální rovnicí prvního řádu

$$y'(t) + a \cdot y(t) = \sin(t). \tag{5}$$

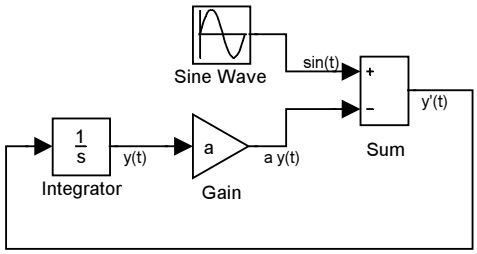
Celou rovnici nejprve přepíšeme tak, aby na levé straně byla nejvyšší derivace, v tomto případě $y(t)$,

$$y'(t) = \sin(t) - a \cdot y(t),$$

vzhledem k tomu, že rovnice musí platit i při modelování, musí být rozdíl veličin $u(t)$ a $a \cdot y(t)$ roven derivaci $y'(t)$. Průběh $y(t)$ ovšem neznáme – lze jej ale získat pomocí integrátoru z vypočtené derivace, kterou máme na výstupu. Výsledné schéma je na obrázku 6.

Cvičení 8:
Zkuste si stejným způsobem vytvořit schéma k rovnici

$$y'''(t) + b \cdot y''(t) + a \cdot y(t) = \sin(t). \tag{6}$$



Obrázek 6: Simulační schéma pro rovnici (5).

3.2.3 Diferenční rovnice

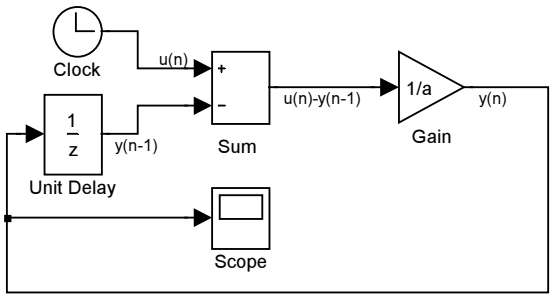
Mějme systém popsaný diferenční rovnicí

$$y(n-1) + a \cdot y(n) = u(t). \tag{7}$$

Před modelováním si opět rovnici můžeme upravit na

$$y(n) = \frac{1}{a}\left[u(t) - y(n-1)\right].$$

a pomocí bloku `Unit Delay` z $y(n)$ vyrobíme $y(n-1)$. Výsledek pro $u(t) = t$ je na obrázku 7.



Obrázek 7: Simulační schéma pro rovnici (7).

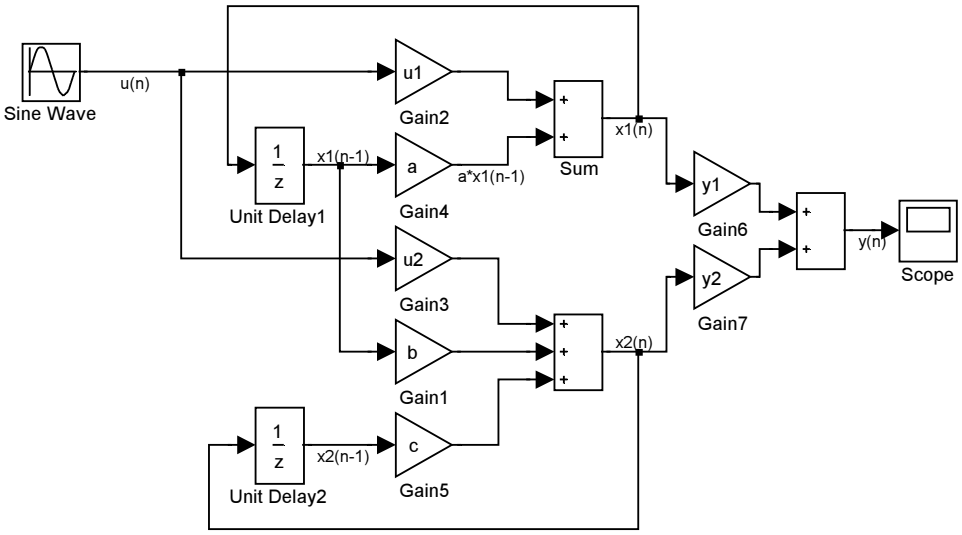
Cvičení 9:
Zkuste si nyní samostatně namodelovat obdobný systém, popsaný rovnicí:

$$y(n+1) + a \cdot y(n) = 1(t). \tag{8}$$

3.2.4 Vnitřní popis systému

Doposud jsme se zabývali modely systémů popsaných vnějším popisem. V některých případech je ovšem lepší použít vnitřní popis. Obrázek 8 ukazuje model diskrétního systému popsaného soustavou

$$\begin{aligned} x_2(n) &= bx_1(n-1) + cx_2(n-1) + u_2u(n) \\ x_1(n) &= ax_1(n-1) + u_1u(n) \\ y(n) &= y_1x_1(n) + y_2x_2(n) \end{aligned} \tag{9}$$



Obrázek 8: Model vnitřního popisu systému podle soustavy (9).

Reference

[1] Svítek, M., Borka, J., Vlček M.: Modelování systémů a procesů. Učební text Fakulty dopravní ČVUT, vydavatelství ČVUT, Praha, 2001.

Řešení Matlab

Cvičení 1:

1. Definujte vektor $\vec{a}$, který obsahuje prvky $3, 4, 5, 6, 7, 8$.

```
>> a = [3 4 5 6 7 8]
>> a = 3:8
```

2. Definujte vektor $\vec{a}$, který obsahuje prvky $8, 7, 6, 5, 4, 3$.

```
>> a = [8 7 6 5 4 3]
>> a = 8:-1:3
```

3. Definujte vektor $\vec{a}$, který obsahuje pouze sudá čísla od 1 do 20.

```
>> a = 2:2:20
```

Cvičení 2:

1. Nadefinujte vektory $\vec{u}, \vec{v}, \vec{w}$ stejné délky a vytvořte z nich tabulku, kde v prvním sloupci bude vektor $\vec{u}$, ve druhém ,$\vec{v}$, a ve třetím $\vec{w}$.

```
>> u = [1 0 0 1]
>> v = [1 1 1 1]
>> w = [1 0 1 0]
>> T = [u' v' w']
```

2. Vytvořte tabulku, goniometrických funkcí od -2π do 2π.

```
>> t = -2*pi:0.1:2*pi;
>> s = sin(t);
>> c = cos(t);
>> Tabulka = [t' s' c']
```

Cvičení 3:

1. Je dán vektor $\vec{u} = \{9\ 11\ 13 \cdots\ 97\}^T$. Jak tento vektor nejefektivněji zadáte.

```
>> u = 9:2:97
```

2. Zadejte matici A, s dimenzí 8×7, která má všechny prvky nulové, pouze prvky $[1, 4]$ a $[4, 6]$ jsou rovny 1.

```
>> A = zeros(8,7)
>> A(1,4) = 1;
>> A(4,6) = 1;
>> A
```

3. Zadejte libovolnou matici A. Jak vytvoříte vektor $\vec{u}$, který bude obsahovat všechny prvky druhého sloupce matice A.

```
>> A = [1 2 3 4; 1 1 1 1; 2 2 2 2];
>> u = A(:,2)
```

4. Zadejte libovolné matici A a B se stejnou dimenzí. V matici A nahraďte první sloupec, posledním sloupcem matice B.

```
>> A = [1 2 3 4; 1 1 1 1; 2 2 2 2];
>> B = [1 1 1 1; 0 0 0 0; 1 1 1 1];
>> A(:,1) = B(:,4);
>> A
```

Cvičení 4:

1. Zadejte matici A a B. Jak provedete vektorový součin? Jaké musí být dimenze matice A a B?

```
>> A = [1 2 3; 1 2 3];
>> B = [1 2; 3 4; 5 6];
>> s = A * B
(m,n)*(n,k)=(m,k), výsledkem operace je skalár
```

2. Od všech prvků matice A odečtěte konstantu 3.

```
>> A = [1 2 3 4; 1 1 4 4; 4 4 4 4];
>> A = A - 3;
```

3. Je dán vektor $\vec{u} = \{1\ 3\ 5\ 7\}$ a vektor $\vec{v} = \{2\ 4\ 6\ 8\}^T$. Jak vypočtete součin vektoru $\vec{u}$ a vektoru $\vec{v}$. Jak by součin vypadal v případě operace "prvek s prvkem".

```
>> s = u * v'
>> S = u .* v
```

Cvičení 5:

1. Je dána matice `A=[1 2 3; 4 5 6; 1 2 3]`. Jak zjistíte, zda jsou hodnoty prvků v prvním řádku matice `A` menší než hodnoty prvků ve třetím sloupci matice A. Zapište výsledek této operace.

```
>> A(1,:)<A(:,3)'
>> ans =
>>        1     1     0
```

2. Je dán vektor $\vec{u} = \{1\ 2\ 3\}$. Výsledkem operace $\vec{u} \neq \vec{v}$ je vektor $\{1\ 0\ 1\}$. Jak zadáte libovolný vektor $\vec{v}$, který odpovídá zadané operaci.

```
např.
>> v = [5 2 5]
```

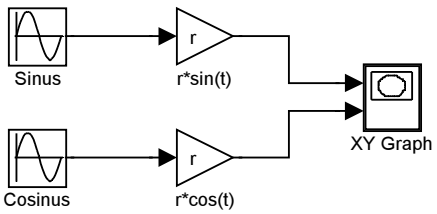
3. Je dán vektor $\vec{u} = \{2\ 2\ 3\}$ a vektor $\vec{v} = \{1\ 2\ 1\}$. Napište relační operátor, pomocí kterého zjistíte, zda je vektor $\vec{u}$ roven vektoru $\vec{v}$. Zapište výsledek této operace.

```
>> u == v
ans =
      0 1 0
```

Řešení Simulink

Cvičení 6:
Namodelujte systém zobrazující rovnici kružnice s poloměrem r, danou rovnicemi

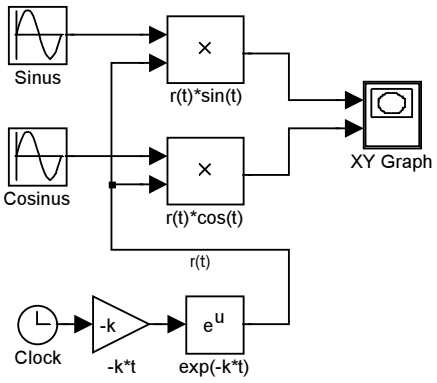
$$\begin{aligned} x &= r\ \sin t \\ y &= r\ \cos t \end{aligned}$$



Obrázek 9: Kružnice o poloměru r.

Cvičení 7:
Namodelujte systém zobrazující logaritmickou spirálu

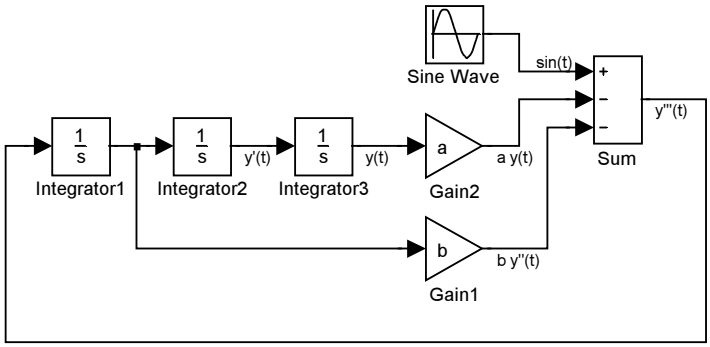
$$\begin{aligned} x &= e^{-kt}\ \sin t \\ y &= e^{-kt}\ \cos t \end{aligned}$$



Obrázek 10: Logaritmická spirála.

Cvičení 8:
Zkuste si stejným způsobem vytvořit schéma k rovnici

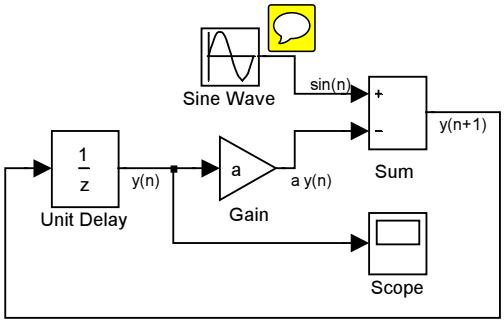
$$y'''(t) + b \cdot y''(t) + a \cdot y(t) = \sin(t). \tag{10}$$



Obrázek 11: Model systému popsaného rovnicí (10).

Cvičení 9:
Zkuste si nyní samostatně namodelovat obdobný systém, popsaný rovnicí:

$$y(n+1) + a \cdot y(n) = 1(t). \tag{11}$$



Obrázek 12: Model systému popsaného rovnicí (11).