11MAMY LS 2019/2020
Cvičení č. 3: 1. 4. 2020

Shrnutí úvodu do Matlabu. Dynamické systémy.

Jan Přikryl

1. dubna 2020

Modelování dynamických systémů v Matlabu.

Obsah

1 Matlab – základní operace

Matlab jako kalkulačka, význam proměnné `ans`, ukládání výsledků do proměnných, základní početní operace (+,-,*,/) například:

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

Zadávání vektorů:

```
>> u = [1 2 3 4 5] % vycet prvku
```

```
>> x = 1:5 % notace s dvojteckou
>> x =
       1 2 3 4 5
>> y = 0:pi/4:pi
>> y =
       0.0000 0.7854 1.5708 2.3562 3.1416
```

Řádkový a sloupcový vektor:

```
>> 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
```

Čtení a zapisování prvků vektoru:

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

Zadávání matice výčtem prvků, indexování řádků a sloupců. Operace typu násobení matic, sčítání matic, násobení vektorů, atp. Více příkladů, řešených i neřešených, studenti najdou v dokumentu „Jemný úvod do Matlabu a Simulinku“, který je dostupný na stránkách předmětu v podkladech pro první cvičení. Seznámit se s těmito příklady měli za domácí úkol.

2 Matlab – základní funkce

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
`eye(m)`	jednotková matice (m,m)
`ones(m,n)`	matice (m,n) jedniček
`zeros(m,n)`	nulová matice (m,n)
`rand(m,n)`	matice (m,n) náhodných čísel
`roots`	výpočet vlastních čísel

3 Grafické možnosti

Základní příkazy:

`figure`	obrázek
`plot`	graf
`legend`	legenda grafu
`title`	titulek grafu
`xlabel, ylabel`	popisky os
`xlim, ylim`	limity os
`subplot(m,n,p)`	podobrázek

Úkol 1. Nakreslete do jednoho obrázku graf funkce

$$y = t \cdot \mathrm{e}^{-\frac{1}{2}t}.$$

Řešení:

```
t = [0:0.1:5];
y = t.*exp(0.5*t);
figure(1);
plot(t,y);
title('Obrazek funkce v Matlabu');
xlabel('t');
ylabel('f(t)');
```

□

Úkol 2. Nakreslete do jednoho obrázku graf funkcí

$$y = f_1(t) = \frac{1}{4}t,$$
$$y = f_2(t) = \mathrm{e}^{-\frac{1}{2}t},$$
$$y = f_3(t) = f_1(t) \cdot f_2(t) = \frac{1}{4}t \cdot \mathrm{e}^{-\frac{1}{2}t}.$$

Řešení:

```
t = [0:0.1:5];
y1 = 0.25*t;
y2 = exp(0.5*t);
y3 = y1 .* y2;
figure(2);
plot(t,y1,t,y2,t,y3);
title('Obrazek tri funkci v Matlabu');
xlabel('t');
ylabel('y');
legend('f_1(t)','f_2(t)','f_3(t)');
```

□

4 Zobrazení signálu

Zmínit zvuk jako soubor harmonických signálů, skládání harmonických.

Úkol 3. Složit dvě frekvence dohromady, zobrazit pomocí `subplot()` a přehrát pomocí `sound()`. Přidat šum, zobrazit a přehrát.

Řešení:

```
t = [0:0.001:0.25];
s1 = sin(2*pi*50*t);
s2 = sin(2*pi*120*t);
s = s1 + s2;
%s = s + 2*randn(size(t));
figure(1)
subplot(1,3,1), plot(t,s1);
subplot(1,3,2), plot(t,s2);
subplot(1,3,3), plot(t,s);
```

□

5 Maticový zápis časově posuntých dat

Diferenční rovnice je rovnice, generující posloupnosti, například na základě diskrétně změřených data (časové řady). Hodnota n-tého členu posloupnosti se spočte z několika předchozích členů, je to obdoba diferenciální rovnice. Diferenčními rovnicemi popisujeme chování diskrétních systémů.

Na příkladu dat z mýtných bran si ale napřed ukážeme, jak si do matice uložit časově posunutá data tak, abychom diferenční rovnici

$$y[n] = ax[n-1] + bx[n-2] + cx[n-3]$$

převedli na

$$y[n] = ax_1[n] + bx_2[n] + cx_3[n].$$

Data si lze stáhnout z http://zolotarev.fd.cvut.cz/static/mamy/count0187.mat.

`for i=1:len(x) ... end`	cyklus
`while x1-x2 < eps ... end`	cyklus
`if x1<x2 ... else ... end`	rozhodovací blok
`function [y1,y2]=fce(x1,x2) ... end`	funkce v `.m` souboru `fce.m`

Tabulka 1: Základní řídicí konstrukce Matlabu

Úkol 4. Načtěte data o počtu vozidel na mýtné bráně 187 ze souboru count0187.mat. Uložte je do matice o pěti sloupích tak, aby v jednom řádku byly uvedeny hodnoty za posledních pět minut (znamená to, že řádky pro 1, 2, 3 a 4 minutu vynecháte, protože nemáte dostatečně stará data).

Řešení:

```
% toto vyrobi promennou ‘count0187‘,
% ta je ale mozna jeste ve formatu struktury a ma
% polozky jako ‘count0187.count‘, ‘count0182.speed‘
% a podobne
load(’count0187.mat’)
% data obsahuji celkem 1440 minut
% o pet minut zpozdena
x4 = count0187(1:1440-4);
% o ctyri minuty zpozdena
x3 = count0187(2:1440-3);
% o tri minuty zpozdena
x2 = count0187(3:1440-2);
% o dve minuty zpozdena
x1 = count0187(4:1440-1);
% za minulou minutu
x0 = count0187(5:1440-0);
% a do matice s nimi
D = [ x0, x1, x2, x3, x4 ];
```

□

6 Základní řídicí konstrukce jayzka Matlab

Příkazy jsou v tabulce 1. Někteří by si je měli pamatovat z algoritmizace (14ADS a další kódy), jenže ten předmět nemají všichni a podle zpráv Michala Jeřábka z K614 toho studenti moc nastudovaného nemají.

U cyklů si prosím zapamatujte, že **for** používáme v případě, že předem víme, kolikrát chceme výpočet opakovat, zatímco **while** je vhodný cyklus pro iterativní výpočet, u něhož neznáme předem počet opakování.

Úkol 5. Třeba sčítání hodnot v poli. Jaký typ cyklu použijeme v případě, že je naším úkolem sečíst číselné hodnoty ve vektoru délky N prvků?

Řešení:
Máme předem daný počet opakování N, půjde o ***for*** *cyklus. Ignorujeme to, že Matlab má příkaz* `sum()`.

```
%% mysum.m
function s=mysum(v)
  n=length(v);
  s=0;
  for i=1:n
    s=s+v(i);
  end
end
```

□

Úkol 6. Mějme diferenční rovnici $y[n+1] = 1/2\, y[n]$ s počáteční podmínkou $y[0] = 10$. Jaký typ cyklu použijeme v případě, že je naším úkolem najít takovou hodnotu n, pro niž je $y[n] < \varepsilon = 0{,}001$?

Řešení:
Nemáme předem daný počet opakování, víme, že nějaká hodnota $y[n]$ musí být nakonec menší, než dané ε, půjde proto o cyklus ***while****.*
Jak budou vypadat iterace? Asi takto:

$$\begin{aligned} y[0] &= 10 \\ y[1] &= y[0]/2 = 5 \\ y[2] &= y[1]/2 = 2.5 \\ y[3] &= y[2]/2 = 1.25 \end{aligned}$$

Zprogramovat to lze s pomocí jediné proměnné, `y = y/2` *(vlevo je nová hodnota y, vpravo stará, proto kdysi Niklaus Wirth zaváděl v Pascalu ono :=, aby bylo jasné, že jde o přiřazení a ne o rovnost ... tohle sice máte umět z algoritmizace, ale nejsem si vůbec jist, že to většina studentů pobrala).*

```
%% mylim.m
function y=mylim(eps)
  n=0
  y=10;
  while y>=eps
    y=y/2;
  end
end
```

□

Úkol 7 (Metoda půlení intervalu)**.** Hledání kořene funkce základní iterční metodou – metodou bisekce (půlení intervalu, pokud nevíte, oč jde, koukněte třeba na https://cs.wikipedia.org/wiki/P%C5%AFlen%C3%AD_interval%C5%AF nebo Google). Jaký typ cyklu použijeme v případě, že je naším úkolem najít pravděpodobný kořen funkce $f(x) = x^2 + \sin x$ s chybou $\varepsilon \leq 0{,}001$?

Řešení:
Nemáme předem daný počet opakování, víme, že nějaký náš odhad chyby musí být nakonec menší, než dané ε, půjde proto o cyklus ***while****. Ignorujeme to, že Matlab má knihovní funkci* `fzero()`*, která je obecnější a konverguje rychleji.*

```
%% myfnc.m
function yy=myfnc(xx)
  yy=xx*xx+sin(xx)
end

%% myzero.m
function x0=myzero(eps)
  %% pocatecni uzavera ("bracket"), druhy koren je x0=0
  %% a,b,myfnc() by mely byt parametry, ale tady na to kasleme
  a = -1.0;
  b = -0.5;
  %% presnost odhadu je dana delkou uzavery
  while abs(a-b)>eps
    %% (a+b)/2 je numericky nestabilni, proto tato konstrukce
    m=a+(b-a)/2;
    %% fubkcni hodnota na jednom kraji a uprostred
    fa = myfnc(a);
    fm = myfnc(m);
    %% pokud je mezi a a m koren, maji fa a fm opacna znamenka
    if fa*fm < 0
      %% koren je mezi a a m, nahradime b
      b = m;
    else
      %% koren neni mezi a a m, je tedy mezi m a b
      a = m;
    end
  end
  %% posledni odhad korene je stred uzavery
  x0=a+(b-a)/2;
end
```

□

7 Klouzavý průměr

Naprogramovat funkci `y=ravg(x,w)` kde `x` je vektor vstupních dat, `w` je šířka okna (počet průměrovaných hodnot) a `y` je výstup. Zobrazit x a y v grafu v různých barvách – y třeba červeně a tlustší čarou. K průměrování použít funkci `mean()`. Počáteční hodnoty pro $i < w$ replikovat. Jako vstup studenti použijí dopravní data, uložená v souboru http://zolotarev.fd.cvut.cz/static/msap/data.mat.

Funkce uložená v souboru ravg.m bude něco jako

```
function y = ravg ( x, w )
y = x;
for j = w:length(x)
    i = j-w+1;
    y(j) = mean(x(i:j));
end
```

a její vykreslení potom

```
>> clear all
>> load data.mat
>> y = ravg ( data, 10 );
>> plot(data);
>> hold on;
>> plot(y,'r','linewidth',2);
```

8 Hod férovou mincí

Simulace hodu férovou mincí s pravděpodobností pannna-orel 50–50. Naprogramovat funkci `[s,p,o]=coin(n)` simulující n hodů mincí. Výstupem je vektor (řetězec) s obsahující symboly 'P' a 'O' a celá čísla p a o udávající, kolikrát padla panna a kolikrát orel.

Funkce bude něco jako

```
function [s,p,o] = coin(n)
s = char ( zeros ( 1, n ));
p = 0;
o = 0;
for i = 1:n
  r = rand();
  if r < 0.5
    s(i) = 'P';
    p = p + 1;
  else
```

```
    s(i) = 'O';
    o = o + 1;
  end
end
```