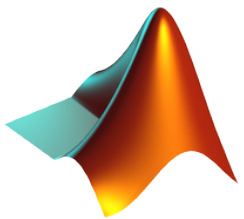
GNU Octave a Matlab – pole buněk, derivace a integrace, ukládání a čtení dat ze souborů

Programování F1400 + F1400a

doc. RNDr. Petr Mikulík, Ph.D.

podzimní semestr 2020



```
barvy = 'krgbymc';
x=0:10;
clf; hold on
for k=1:length(barvy)
    plot(x, k-x, [barvy(k), '-*']);
    popisek{k} = ['Barva: ', barvy(k)];
end
hold off
legend(popisek);
title('ukazka barev v grafu pomoci pole bunek')
grid
```

Pole buněk (cell array) – vektor (nejen) řetězců

- Potřebujeme-li proměnnou, která obsahuje **seznam** (též pole či nečíselný vektor) s položkami různých typů, např. (různě dlouhé) řetězce či řetězce i čísla, pak použijeme **číslovaný seznam** neboli **pole buněk**: pole indexované pomocí složených závorek. Příklad:

Příkazy:

```
clear

z{1} = 'silicon';
z{2} = 'germanium';

z{4} = 3.14;
z{5} = 1:6

% Totéž dostaneme i tímto zápisem:
z = { 'silicon', 'germanium', [], 3.14, 1:6 }
```

Výstup:

```
z =
{
  [1,1] = silicon
  [1,2] = germanium
  [1,3] = [](0x0)
  [1,4] =  3.1400
  [1,5] = 1   2   3   4   5   6
}
```

- **Přístup k prvkům:** složené závorky přistupují k hodnotám buněk, složené závorky k položkám vektoru:

```
p = z{1};        % hodnota buňky (řetězec 'silicon')
z{3} = 123.45    % nahraď současnou hodnotu jinou
z(3) = []        % smazání položky
zz = z(1:3:9)    % nové pole s menším počtem buněk
```

Pole buněk – zabudované (množinové) funkce a příklad

- **Třídění řetězců** je stejné jako u číselných vektorů:
  Příkaz `sort(z)`: setřídění položek pole buněk
  Příkaz `unique(z)`: vyjmutí duplicit v setříděném vektoru
- **Množinové funkce:**
  Příkaz `union(z1, z2)`: proveď sjednocení množin (množinový součet)
  Příkaz `intersect(z1, z2)`: proveď průnik množin (množinový součin, tj. vypiš shodné položky)
  Příkaz `setdiff(z1, z2)`: proveď množinový rozdíl (vypiš prvky jedinečné v `z1`)
  Příkaz `setxor(z1, z2)`: množinový rozdíl v obou polích (jedinečné v `z1` i `z2`)
  Příkaz `ismember(z1, z2)` či `ismember(z2, z1)`: které z položek si odpovídají
- **Příklad**

```
1 z1 = {'Si', 'Ge', 'Ar', 'He', 'O', 'Ga'}
2 z2 = {'Kr', 'Ar', 'Ga', 'As', 'Si', 'C', 'H'}

3 strcmp(z1, 'He')
4 sort(z1), sort(z2)

5 union(z1, z2)
6 intersect(z1, z2), ismember(z1, z2), ismember(z2, z1)
7 setdiff(z1, z2), setxor(z1, z2)
```

Pole buněk – příklad

- Když se kreslí grafy příkazem `plot()`, tak se předchozí graf vždy smaže. Tomu zabráníme příkazem `hold`:

```
hold on
x=-4:0.2:4; plot(x, x); plot(x, x.*x)
hold off
```

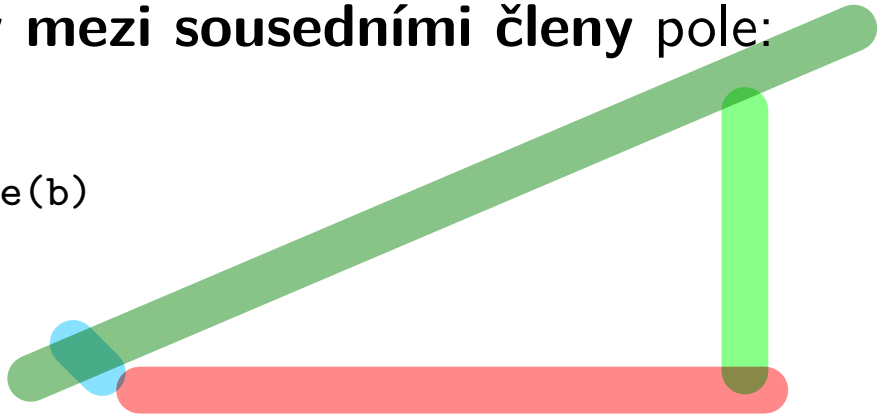
- Příklad: vykreslete tolik křivek, kolik je různých barev v běžném grafu (v Matlabí syntaxi), a ke grafu vyrobte legendu s názvem barvy.

  Text pro legendu bude pole řetězců obsahující popisky jednotlivých křivek. Vygeneruj seznam řetězců `popisek{1}` až `popisek{n}` pro legendu grafu:

```
barvy = 'krgbymc';      % barvy: blacK Red Green Blue Yellow Magenta Cyan
x=0:10;                 % nejaka osa x
clf; hold on            % smaz soucasny graf a nastav prekreslovani
for k=1:length(barvy)   % projdi vsechna pismenka barev
    plot(x, k-x, [barvy(k), '-*']);      % nejaka linearni funkce y=y(x)
   popisek{k} = ['Barva: ', barvy(k)]  % vyrob popisek pro legendu grafu
end
hold off              % ukonci prekreslovani
legend(popisek);      % zobraz legendu (popis krivek)
```

Numerická derivace

- Funkce `diff()` – **rozdíly mezi sousedními členy** pole:



```
octave> a1 = [1 2 3 4 5];
octave> b=diff(a1), c=unique(b)
b =
   1   1   1   1
c =
   1
octave> a2 = [1 4 8 16 32];
octave> diff(a2)
ans =
    3    4    8   16
```

- **Numerická derivace** – použití funkce `diff()`:

```
1 x = linspace(-10,10,50);          % vzorkovani osy x
2 y = x.*x;                         % funkce y = x^2

3 deriv = diff(y) ./ diff(x);       % vypocet numericke derivace
4 xderiv = x(1:end-1);              % pole deriv ma o jeden bod mene nez pole x, y
5 % vyzkousejte si: length(x), length(y), length(deriv)

6 plot(x,y, xderiv,deriv);          % vykresli funkci a jeji derivaci
7 legend('funkce x^2', 'jeji derivace');
```

Funkce eval() a příklad na numerickou derivaci

- Funkce `eval(příkaz)` vyhodnotí **řetězec jako příkaz**, jako by byl tento příkaz zadán na příkazové řádce:

```
eval('a=3; b=4; c=sqrt(a*a+b*b)')
cmd='plot(1:10)'
eval(cmd)
```

- **Příklad**: `eval` a `diff()` pro graf funkcí, vzorec zadaný řetězcem:

```
1  x = linspace(-4*pi,4*pi,150);        % vzorkovani osy x
2  yfce = { '0*x', '2*x+1', 'x.*x', 'sin(x).*sin(x)' }
3  clear ydata deriv
4  xderiv = x(1:end-1);

5  for k=1:length(yfce)
6     cmd = [ 'y=', yfce{k}, ';' ];
7     % cmd = sprintf('y=%s;', yfce{k});  % jiný zápis výše uvedeného
8     eval(cmd);
9     d = diff(y) ./ diff(x);     % vypocet numericke derivace
10    ydata(k,:) = y;      % funkcni hodnoty
11    deriv(k,:) = d;      % derivace
12 end
13 figure(1); plot(x,ydata);         % vykresli funkci a jeji derivaci
14 legend(yfce); title('puvodni funkce')
15 figure(2); plot(xderiv,deriv); % vykresli derivace
16 legend(yfce); title('derivace')
```

Numerická integrace

- **Numerická integrace**: výpočet plochy pod křivkou danou analytickým vzorcem, zjemňování až je dosaženo maximální či zadané přesnosti (lichoběžníková metoda, Simpsonova metoda aj.), viz numerická matematika.
- **Numerická integrace** v Octave/Matlab: optimalizovaná funkce `quad()`. Integrovanou funkci do ni vkládáme jako **řetězec**, nebo jako **anonymní (bezejmennou) funkci**, nebo jako **normální funkci** (uvedenou níže nebo ve vlastním souboru `nazevfunkce.m`):

```
function t_integrace
quad('cos(x)', 0, pi/2)        % 1. způsob: funkce jako řetězec
quad(@(x) cos(x), 0, pi/2)     % 2. způsob: "anonymní" funkce
quad(@moje_funkce, 0, pi/2)    % 3. způsob: vhodné pro složitější funkce
end

function y=moje_funkce(x)
y = cos(x);
end
```

- **Dvojný a trojný integrál**

```
% dvojný integrál:
dblquad(@(x,y) x.*x+y.*y, 0,2, 1,4)

% trojný integrál:
triplequad(@(x,y,z) x+y+z, 0,2, 1,5, -2,2)
```

Zápis a čtení z/do souboru – textové datové soubory

- Zápis a čtení matic aj. z/do textového souboru – GNU Octave i Matlab

```
save -ascii matice.dat a      % ulozi matici jako citelny textovy soubor
load matice.dat               % nacte matici ulozenou v textovem souboru
load('matice.dat')            % nacte matici ulozenou v textovem souboru
b=importdata('matice.dat')    % jiny zpusob (o neco rychlejsi)
```

- Zápis a čtení matic aj. z/do textového souboru – pouze GNU Octave

```
save -text vsechno.dat a x y  % ulozi vse jako textovy soubor (pouze GNU Octave)
load vsechno.dat              % nacte vse z textoveho souboru (pouze GNU Octave)
```

- Zápis a čtení matic z/do textových datových souborů **csv** (přenos data s tabulkovým procesorem)

```
data = csvread('data.csv')
data = textread('data.csv', 'delimiter', ',', 'commentstyle', 'shell')

viz tez csvwrite(...)
```

Zápis a čtení z/do souboru – binární mat soubory

- **Zápis do a čtení z** nativního matlabího binárního souboru pomocí příkazů **save** a **load**: zadává se soubor a seznam proměnných.

```
a=1./hilb(5); x=1:10; y=x.*x; % vytvor nejake promenne

save vsechno.mat        % uloz vsechny promenne
save xy.mat x y         % uloz jen dve promenne
save -v7 xy.mat x y     % uloz, pouzij komprimovany format (mensi soubory)

clear                   % vymaz vsechny promenne
load vsechno.mat        % nacte vsechny promenne ulozene do souboru
whos                    % ukaz, co jsi nacetl

clear
load xy.mat             % nacte ze souboru jen promenne x, y
ted=load('xy.mat')      % nacte soubor a vytvori strukturu s prvky ted.x a ted.y
whos                    % ukaz, co jsi nacetl
```

Zápis do souboru

- Zápis do souboru: **otevřeme soubor** pro zápis `fopen(...,'w')`, **zapisujeme do souboru** `fprintf()`, nakonec **soubor zavřeme** `fclose()`.
  Pozn.: V C se to dělá stejně, proměnná typu soubor se deklaruje `FILE* fid`.
- Příklad:

Příkazy:

```
fid = fopen('vystup.dat', 'w');
fprintf(fid, 'ahoj svete\n');
fprintf(fid, '1+2*3 = %i\n', 1+2*3);
fprintf(fid, '2*pi = %20.16f\n', 2*pi);
for k=1:3
    fprintf(fid, '%g\t%g\n', k, 1.2*k);
end
fclose(fid);
```

Soubor vystup.dat obsahuje:

```
ahoj svete
1+2*3 = 7
2*pi =   6.2831853071795862
1 1.2
2 2.4
3 3.6
```

- Chceme-li připsat něco na konec (existujícího) souboru, pak:

Příkazy:

```
fid = fopen('vystup.dat', 'a');
fprintf(fid, 'The end.\n');
fclose(fid);
```

Výstup:

```
ahoj svete
1+2*3 = 7
2*pi =   6.2831853071795862
1 1.2
2 2.4
3 3.6
The end.
```

Čtení z textového souboru

- Nejjednodušší je načtení matice čísel z textového souboru příkazem `a=load('soubor.dat')`. Pak můžeme matici přeformátovat na vektor příkazem `b=a(:)` nebo na jinak tvarovanou matici `c=reshape(a,2,prod(size(a))/2)`.
- Čtení ze souboru: otevřeme existující soubor pro čtení `fopen(...,'r')`, čteme čísla, řetězce nebo řádky pomocí `fscanf()`, `fgetl()`, pak zavřeme `fclose()`. *Pozn.: V C se to dělá stejně, proměnná typu soubor se deklaruje jako* `FILE*`.
- Příklad – **načtení všech čísel ze souboru najednou:**

```
fid = fopen('cisla.dat', 'r');     % otevri vstupni soubor
a = fscanf(fid, '%g')              % nacti vsechna cisla do jednoho vektoru
fclose(fid);                       % zavri vstupni soubor
```

- **Čtení a zpracování textového souboru řádek po řádku**, každý řádek je tedy řetězec:

```
fid = fopen('vstup.dat', 'r');              % otevri vstupni soubor
radek=0;                                    % pocitadlo radku
while 1
    s = fgetl(fid);                         % nebo: s = fgets(fid)
    if (s==-1) break; end                   % nalezen konec souboru
    radek = radek+1;                        % zvetsi pocitadlo radku
    fprintf('Radek %i: %s\n', radek, s);    % nebo: fprintf('Radek %i: %s',radek,s);
end
fclose(fid);                                % zavri soubor
```

Načtení celého textového souboru najednou

- **Načtěte celý textový soubor najednou a zpracujte ho poté:**
  - Zobraz dialog pro výběr souboru pomocí `uigetfile()`.
  - Načti celý textový soubor najednou do jedné řetězcové proměnné `s`.
  - Tuto rozděl na řádky do pole řetězců `lines`.
  - Načtené řádky vypiš.
  - Řádky pak vhodným způsobem zpracuj.

```
1  [fajl, cesta] = uigetfile ('*.*', 'Pick a file'); % zobraz dialogove okno
2
3  fid = fopen(fajl, 'r');                % otevri soubor
4  s = char(fread(fid, 'uchar')');        % nacti cely soubor najednou
5  fclose(fid);                           % zavri soubor
6
7  lines = regexp(s, '\n', 'split')       % rozdel nacteny soubor na radky
8
9  for k=1:length(lines)                  % vypis nactene radky
10     fprintf('Radek %2i | %s\n', k, lines{k})
11 end
12
13 % Nyni muzeme napr. vypsat vsechny radky, ktere obsahuji pismeno 'a',
14 % a spocitat pocet vyskytu. (Pozn.: pouzijte strfind()).
```