MATLAB/Octave

Skriptovací jazyk

Proč MATLAB:
-Rychlý vývoj
- Vhodný pro matematické operace
- Java například na předzpracování hrozně pomalá, ale vyvíjet všechno v C/C++ trvá dlouho
- Dnešní trend C/C++ na výpočetně náročné operace, Skriptovací jazyk(MATLAB, Python, Perl) na různé předzpracování a na prototypování. -> mnohem kratší vývojový čas při zachování efektivity

MATLAB = MATrix LABoratory - Základní datový typ je matice. V podstatě všechno je matice

- číslo je matice 1x1
- pole (vektor) matice n x 1 resp. 1 x n
- text je matice jeden řádek textu = jeden řádek matice, řádek je vektor znaků
- černobílý obrázek je matice intenzit pixelů o rozměrech šířka x výška
- barevný obrázek je trojrozměrná matice šířka x výška x 3 (RGB)
- atd…

Octave je volně dostupný ořezaný MATLAB

MATLAB má mnohem více high-level funkcí, Octave je syntakticky bližší C-like jazykům

Základní operace

`addpath` - přidá složku do cesty kde hledá skripty (load path).
`pwd`
`cd`
`disp`
`who`
`clc; clear all; close all;` vždy první příkazy skriptu, uklidí po předchozích skriptech
ans = výsledek poslední operace
.
.
.

Konvence

A, B, C, X, Y ... - matice
a, b, c, x, y … - vektory

Vytvoření matice

1) výčtem prvků v hranatých závorkách

```
A = [1, 2, 3; 4, 5, 6; 7, 8, 9]
```

Matice 3x3 s prvky 1-9

prvky na řádcích oddělené čárkou, řádky středníkem

Čárky můžeme vynechat:

```
A = [1 2 3; 4 5 6; 7 8 9]
```

2) Speciální matice

funkce `zeros()` a `ones()` vytvoří matici o zadaných rozměrech se samými nulami respektive jedničkami.

```
zeros(10)

ones(3, 3)

rand(4, 3, 2)

randn(3, 3)

diag(x)
```

Funkce `eye(n)` vytvoří jednotkovou matici n x n

```
size(X)
length(X)
```

Matematicko-logické operace

a) Matice skalár

```
A + 2;
2 * A
```

Aplikuje se postupně na všechny prvky matice

b) Matice matice

`A + B`

`A * B` maticové násobení

`A .* B` násobení po prvcích

`A ^ 2` umocnění na druhou
`A .^ 2` umocnění jednotlivých prvků

A / B ekvivalentní A * B^{-1}
A \ B ekvivalentní A^{-1} * B

`&&`
`||`
`==`
`~=`
`!=` možné pouze v Octave (v MATLABu ne)

Zkrácené zápisy jako `A++` `A += 5` atd. jsou opět možné pouze v Octave -> nepoužívat

inv(A)
pinv(A)
transpose(A)
A'
det(A)

Příklad

x = [2 5 1 6]
a = x + 16
c = sqrt(x) **or** c = x.^(0.5)
d = x.^2 **or** d = x.*x

lineární regrese A \ b

Hledání prvků:
funkce find vrací pole indexů nenulových prvků
Pokud tedy chceme zjistit, na kterém indexu vektoru `v` se nachází číslo 3:
`find(v == 3)`
výsledkem operace `v == 3` je vektor o stejné délce jako `v`, který obsahuje jedničky na indexech, kde v obsahuje hodnotu 3 a nuly na ostatních.

Indexování a operátor :

V MATLABu a Octave se indexuje od 1 a index se píše do kulatých závorek
A(1, 2) prvek v prvním řádku a druhém sloupci
matici je možné indexovat i jedním indexem, prvky jsou pak číslované takto:

```
1 4 7
2 5 8
3 6 9
```

Matici je možné indexovat opět maticí. Výsledkem je potom matice o stejných rozměrech jako indexační matice, kde je každý prvek nahrazen prvkem z indexované matice na odpovídajícím indexu.
Příklad:

```
      1 2 3
A =   4 5 6
      7 8 9

i = 1 4 6

A(i) = 1 2 8
```

Operátor :

Obecně slouží k vytvoření aritmetické posloupnosti

```
A = 1: 10
```
vytvoří vektor s prvky `1 2 3 4 5 6 7 8 9 10`

v případě třech parametrů je druhý parametr krok (default 1)

```
A = 1: 2 : 10
A = 1 3 5 7 9
```

krok nemusí být celočíselný

```
A = 1 : 0.1 : 2
```

ani kladný

```
A = 10 : -1 : 1
```

jelikož dvojtečkový operátor vytvoří vektor, je jím možné indexovat (za předpokladu, že jsou meze celočíselné a krok také).

Navíc při indexování lze využít další věci

klíčové slovo `end`. značí konec indexované oblasti (pole)

```
a = [1 2 3 4 5]

a(3 : end)
3 4 5
a(0 : end)
1 2 3 4 5
```

předchozí příklad je pro vektor sice nesmyslný, má ale velké využití u dvou a více rozměrných matic kde vybíráme například celý sloupec. Používá se tak často že pro něj byla vytvořena zkratka a můžeme psát pouze:

`A(:, 2)` vybere druhý sloupec matice `A`

Příklad

x = [2 5 1 6]
    b = x(1:2:end) + 3

A = [2 4 1 ; 6 7 2 ; 3 5 9]
    x1 = A(1,:)
    y = A(end-1:end,:)
    c = sum(A)
    d = sum(A,2) **or** d = sum(A')'
    N = size(A,1), e = std(A)/sqrt(N)
Vyberte z vektoru pouze prvky na sudém indexu
Vytvořte jednotkovou matici bez použití funkcí eye() a diag()

Podmínky a cykly

```
if podmínka
    příkazy
elseif podmínka
    příkazy
else
    příkazy
end
```

```
for i = 1:10 #prochází přes všechny prvky pole přiřazeného do i
    příkazy
end

while podmínka
    příkazy
end
```

Vizualizace

`plot(x, y)` - napojí se na poslední vizualizační okno (v případě, že žádné neexistuje, vytvoří ho) a vykreslí do něj xy graf.

`hist(x)` - opět do posledního okna vykreslí histogram.
`figure` - vytvoří nové vizualizační okno.
`hold on`
`hold off` - pokud je vypnuto, vykreslením nových hodnot smažeme hodnoty předchozí. Když je zapnuto, vykreslíme další křivku, bez odstranění té předchozí.

```
title()
xlabel()
ylabel()
legend()
axis(xMin, xMax, yMin, yMax)
```

Funkce

```
function [výstupní parametry] = nazev(vstupní parametry)
     příkazy
end
```

návratová hodnota se vrací inicializací výstupních parametrů. výstupních parametrů může být více v takovém případě se funkce volá takto:

```
[a, b] = funkce(x);
```

Příklad

```
function [ y ] = fact( x )
     if(x == 0)
          y = 1;
     else
          y = x * factorial(x - 1);
     end
end
```

Buňky (cell)

Nehomogenní pole, které může obsahovat několik polí různých rozměrů a typů

```
A = [1 2 3; 1 2 3; 1 2 3];      pole
C = {A sum(A) det(A) };         buňka
```

indexuje se složenými závorkami

```
C{1}                            A
C{2}                            3 6 9
C{3}                            0
```

Struktury

Hierarchické datové typy.

```
a = "field2";
x.a = 1;   ve struktuře x vytvoří člen a a přiřadí mu hodnotu 1
x.(a) = 2;
x
     ⇒ x =
        {
          a =  1
          field2 =  2
        }
```

Řetězce

Jednořádkový text je v matlabu řádkový vektor znaků

Víceřádkový text je matice znaků, nevýhodou je, že jednotlivé řádky musí být stejně dlouhé.

Funkce `char()` spojí řetězce a doplní je na konci bílými znaky tak, aby byly řádky stejně dlouhé. Zároveň čísla převede na znaky.

Funkce `double()` převede chary zpět na čísla

http://www.facstaff.bucknell.edu/maneval/help211/exercises.html