Základy algoritmizace a programování

Příklady v MATLABu

Přednáška 10
30. listopadu 2009

Řídící instrukce – if – else

C

Matlab

Podmíněný příkaz

```
if ( podmínka )
   { příkazy }
else
   {  příkazy }
```

```
if podmínka
    příkazy
elseif podmínka2
         příkazy
...
      else
         příkazy
end
```

C: příkazy se sdružují do **bloku** pomocí složených závorek

MATLAB: posloupnost příkazů je ukončena slovem `end`

Relační a logické operátory

C		MATLAB
$>, <, >=, <=$	větší, menší, ...	$>, <, >=, <=$
$= =$	rovnost	$= =$,
&&, ||	konjunkce, disjunkce	&, |
!, !=	negace, nerovnost	~, ~=

V MATLABu POZOR!

Jsou li A, B **matice**, nemusí být pravdivá ani jedna z podmínek:
A = = B, A $>$ B, A $<$B , A ~= B
protože pravdivé jsou, když jsou splněny **pro všechny** prvky matic

V MATLABu navíc:

```
any
all
isequal
isempty
```

Řídící instrukce – cykly

C | Matlab

Cyklus while

C:

```
while (podmínka)
{ příkazy }
```

Matlab:

```
while podmínka
    příkazy
end
```

Cyklus for

C:

```
for (inicializace;podmínka;změna)
{  příkazy }
```

Matlab:

```
for proměnná = výraz
    příkazy
end
```

C: příkazy se sdružují do **bloku** pomocí složených závorek
MATLAB: posloupnost příkazů je ukončena slovem `end`

Řídící instrukce – přepínač

C | Matlab

Přepínač switch

C

```
switch (výraz)
{
case hodnota1,hodnota2:
  příkazy; break;
case hodnota3:
  příkazy; break;
...
default:
...
}
```

Matlab

```
switch výraz
case{ hodnoty1}
    příkazy
case  {hodnoty2}
    příkazy
...
otherwise
    příkazy
end
```

C: Není li `break;` výpočet pokračuje dál.

MATLAB: `break` není nutný, **vždy** je vybrán **jediný** case !
`break` lze použít pro předčasné ukončení cyklu.

M–files

Program v MATLABu – posloupnost příkazů, které mohou být uloženy v souborech. Takovým souboům s instrukcemi říkáme **m–files**.

- **Program** je obyčejný textový soubor, který lze vyvtvořit v **textovém** editoru, můžeme použít editor MATLABu.
- **Příkazy** (instrukce) se zapisují stejně jako v příkazovém řádku, oddělují se znaky **; , , ,** nebo **"enter"**
- **Konec programu** nemá žádný speciální znak, jednoduše poslední příkaz. Předčasné ukončení – příkaz **return**
- Komentář (poznámky) : od znaku % do konce řádku
- Komentáře na začátku souboru jsou dostupné z **help**u pomocí **help jméno_programu**

Typy **m**–souborů

Scénáře (skripty)

- Pracují s daty v pracovním prostředí (workspace)
- Nemají vstupní argumenty, nemohou vracet výsledky
- Použití : odladění a orientační výpočty

Funkce

- Pracují s vlastními – **lokálními** proměnnými nebo s globálními daty
- Mohou mít vstupní argumenty a mohou vracet výsledky
- Rozšiřují možnosti jazyka MATLAB

Funkce

- Hlavička funkce
  1. `function [y1,y2,...,ym] = ff(x1,x2,...,xn)`
  2. `function y = ff(x1,x2,...,xn)`
  3. `function ff(x1,x2,...,xn)`
  4. `function [y1,y2,...,ym] = ff`
- Volání funkce
  1. `[y1,y2,...,ym] = ff(x1,x2,...,xn)`
  2. `y = ff(x1,x2,...,xn)`
  3. `ff(x1,x2,...,xn)`
  4. `[ y1,y2,...,ym] = ff(x1,x2,...,xn)`
- Globální proměnné – `global a1 a2 ...`
  musí být uvedeno ve **všech** funkcích, které tyto proměnné používají
- Lokální funkce – podfunkce
  Soubor – funkce může obsahovat **několik** funkcí.
  První (hlavní) je dostupná zvně.
  Ostatní – lokální nebo podfunkce – mohou být volány pouze z hlavní funkce nebo z některé z lokálních funkcí v tomto souboru.

Příklady – vytvoření m–funkcí

- Funkce y = f(x) – soubor f.m

```
 function y = f(x)
y = x^3 - 2*x - 5
```

- Funkce puleni – řešení f(x) = 0 – soubor puleni.m

```
function x = puleni(f, a, b, presnost)
if f(a) * f(b) > 0
  disp('Chybny interval')
  break
while abs(b-a) > presnost
  x = (a+b)/2
  if f(x) = = 0
      break
  elseif f(x)*f(a) > 0
      a=x
  else
      b=x
  end %if
end %while
```

Eulerova metoda – jako m.funkce v souboru euler.m

```
 function [x y] = euler(x0,y0,h,N)
%od počátečního bodu [x0,y0] s krokem h N kroků
%vráti 2 vektory - x-ové hodnoty a y-ové hodnoty
x(1) = x0
y(1) = y0
for i=2:N
y(i) = y(i-1)+h*f(x(i-1), y(i-1))
x(i) = x(i-1)+h
end
% lokální funkce
function yy = f (x,y)
yy = 3*x^2/(2*y)
```

Příklad – použití funkce jako parametru

```
x=puleni(@f,1,3,0.0001)
```

Příklady práce s maticemi

- Načtení hodnot ze souboru
- Zjištění rozměrů matice

```
[mA nA] = size(A)
[mB nB] = size(B)
```

- Vytvoření matice jiného rozměru

```
B=reshape(B, mm, nn)
```

  POZOR, počet čísel v původní matici B **musí** být **přesně** `mm*nn`

- Určení hodnosti matic:

```
hA = rank(A)
hB = rank(B)
```

- Je matice čtvercová nebo obdélníková?

```
if mA = = nA
disp('Matice A je ctvercova')
else
disp('Matice A je obdelnikova')
end
```

Příklady práce s maticemi

- Je matice regulární?

```
if hA = =mA
disp('Matice A je regularni')
```

- Je matice symetrická?

```
if A = =A')
disp('Matice A je symetricka')
```

- Determinant matice

```
detA = det(A)
```

- Vlastní čísla a vlastní vektory matice

```
spektrumA = eig(A)
```

spektrum matice je množina všech vlastních čísel

```
[V D ] = eig(A)
```

výsledky: V matice, jejíž sloupce jsou vlastní vektory, odpovídající vlastním číslům, ktará jsou na diagonále matice D

- Inverzní matice

```
invA = inv(A)
```

Některé vlastnosti matic

- Norma matice
  - řádková
    `normaR = max(sum(A'))`
  - sloupcová
    `normaS = max(sum(A))`
  - Euklidovská
    `normaE = sqrt(sum(sum(A.^2)))`
- Číslo podmíněnosti matice
  `podminenost = cond(A)`
  číslo podmíněnosti = největší vlastní číslo/nejmenší vlastní číslo
- Ostře diagonální matice *pro všechny řádky platí: absolutní hodnota čísla na hlavní diagonále je větší než součet absolutních hodnot zbývajících čísel v řádku*
- Pozitivně definitní matice
  Všechny hlavní minory matice jsou kladné

Řešení soustavy lineárních rovnic

Gaussova eliminační metoda
Dána matice soustavy A.
(*předpokládáme, že matice je regulární, a tedy existuje jediné řešení*)
Vektor pravé strany `b` musí být **sloupec**, musí mít tolik prvků, kolik má matice A řádků.
Potom řešení soustavy rovnic:
`x=A\b`
Příklad:

$$A = \begin{pmatrix} 4 & 2 & -1 \\ 1 & 2 & 3 \\ 2 & 0 & 1 \end{pmatrix}, \quad b = \begin{pmatrix} 6 \\ -1 \\ 1 \end{pmatrix}$$

```
A = [4 2 -1; 1 2 3; 2 0 1]
b = [6; -1; 1]
x=A\b
```

(x = 1 0.5 -1)