Laborator 2 Metode Numerice
- Elemente fundamentale de programare în Matlab -

1 Instrucţiuni şi comenzi Matlab

În Matlab există un singur tip de date, reprezentat de o matrice ce poate conţine şi elemente de tip complex. Numerele sunt de asemenea interpretate ca o matrice cu un singur element. Datele de tip numeric pot lua valori întregi, reale sau complexe. Tipul variabilelor nu trebuie însă declarat, Matlab-ul le va interpreta în funcţie de prelucrările sau operaţiile aplicate. De asemenea, nici dimensiunile matricelor nu trebuiesc declarate, ele vor fi recunoscute în mod automat. Declararea variabilelor se face cu nume formate din cel mult 19 caractere alfanumerice, primul caracter fiind obligatoriu reprezentat de o literă.

În cele ce urmează, este prezentată o listă cu constantele şi variabilele predefinite din Matlab:

- `pi` - constanta π;
- `i,j` - unităţile imaginare;
- `Inf` - infinit (valoare care se obţine ca urmare a unei împărţiri cu zero);
- `eps` - epsilon-ul maşinii, eroarea relativă specifică de rotunjire;
- `NaN` - Not a Number, rezultat obţinut ca urmare a unor nedeterminări de forma $\frac{0}{0}$, $\frac{\infty}{\infty}$;
- `ans` - Answer, variabilă ce reţine conţinutul ultimei operaţii din fereastra de comenzi Matlab pentru care nu s-a făcut o atribuire cu un alt nume;
- `nargin` - indică numărul de argumente de intrare pe care le are funcţia în interiorul căreia se utilizează această variabilă;
- `nargout` - indică numărul de argumente de ieşire pe care le are funcţia în interiorul căreia se utilizează această variabilă.

Constantele şi variabilele predefinite nu pot fi utilizate ca nume pentru scripturile Matlab.

Variabilele globale se declară cu ajurorul comenzii: `global var1 var 2 var3 ... varn.`

Pentru a vizualiza lista variabilelor dintr-o sesiune de lucru, precum şi memoria folosită, se foloseşte comanda: `who`. Pentru obţinerea mai multor informaţii despre o variabilă, folosim comanda: `whos.`

De asemenea, putem elibera memoria de una sau mai multe variabile din sesiunea curentă astfel:

`clear var` - eliberează memoria de variabila `var`;

`clear a b c` - eliberează memoria de variabilele `a,b,c`;

`clear` - este eliberată memoria de toate variabilele.

Comanda `clc` curăţă zona de lucru de rezultatele obţinute anterior lansării ei în execuţie. Variabilele declarate într-o sesiune de lucru Matlab ocupă memoria pe măsură ce sunt definite.

Pentru a insera comentarii într-o secvenţă de cod Matlab, folosim caracterul "%".

Operatori şi funcţii Matlab

Operatorii aritmetici sunt: adunarea (+), scăderea (-), înmulţirea (*), împărţirea la stânga (\), împărţirea la dreapta (/), ridicarea la putere (^), transpusa unei matrice (‘). În cazul în care dorim să realizăm calcule cu valori numerice, regulile de calcul rămân cele cunoscute şi vom folosi operatorii aritmetici uzuali. Există însă câteva considerente pe care trebuie să le luăm în calcul dacă dorim să folosim vectori sau matrice.

Adunarea şi scăderea pot fi implementate pentru două sau mai multe matrice cu aceeaşi dimensiune, respectiv pentru o matrice şi un număr, caz în care valoarea numerică va fi adunată sau scăzută din toate elementele matricei.

Înmulţirea poate fi realizată: între două matrice care respectă regulile aritmetice de înmulţire (numărul de coloane ale primei matrice este egal cu numărul de linii al celei de-a doua matrice), respectiv între o matrice şi un număr. Înmulţirea poate fi realizată şi între două matrice egale ca dimensiune, acest lucru realizându-se element cu element şi folosind simbolul ".*" în loc de "*".

Împărţirea poate fi realizată sub mai multe forme:

- împărţire la stânga: $x = A\backslash b$ ($x = A^{-1} \cdot b$, soluţia ecuaţiei matriceale $Ax = b$);
- împărţire la dreapta: $x = C/A$ ($x = C \cdot A^{-1}$, soluţia ecuaţiei matriceale $xA = C$);
- împărţirea unei matrice la un număr: $a = A/nr$;
- împărţirea a două matrice de dimensiuni egale, element cu element, folosind simbolul "./": $X = A./B$.

De asemenea, pentru ridicarea la putere avem următoarele situaţii:

- comanda `A^n`, unde $n \in \mathbb{Z}_+$ semnifică faptul că matricea `A` se înmulţeşte cu ea însăşi de `n` ori;
- comanda `A.^n` va genera o nouă matrice de aceeaşi dimensiune cu `A` în care fiecare element a lui A va fi ridicat la puterea `n;`
- comanda `A.^B` este folosită pentru a realiza ridicarea la putere, element cu element, a două matrice de diumensiuni egale `A` şi `B.`

Operatorii relaţionali folosiţi în Matlab sunt: mai mic (<), mai mare (>), mai mic sau egal (<=), mai mare sau egal (>=), diferit (∼=), identic (==). Aceştia sunt folosiţi în cadrul condiţiilor de test, rezultatul returnat având valoarea de adevăr true (1) sau false (0). Operatorii relaţionali pot fi folosiţi şi în lucrul cu matrice de aceeaşi dimensiune, rezultatul returnat va fi tot o matrice cu elemente de 1 sau 0, corespunzătoare rezultatului returnat prin aplicarea operatorilor relaţionali.

Operatorii logici folosiţi în secvenţele de cod Matlab sunt: negaţia (∼), şi logic (&), sau logic (!).

Instrucţiuni în Matlab

Ca în cazul celorlalte limbaje de programare, principalele instrucţiuni pe care le putem folosi sunt cele uzuale, şi anume: `if`, `for`, `while`. Fiecare dintre aceste instrucţiuni se încheie cu sintagma `end`. Prezentăm în continuare sintaxa generală pentru fiecare dintre acestea.

$\rightsquigarrow$Pentru instrucţiunea `if`, avem următoarele variante de construcţie:

a) if expresie_logică
  set de instrucţiuni;
end

b) if expresie_logică
  set de instrucţiuni 1;
else
  set de instrucţiuni 2;
end

c) if expresie_logică 1
  set de instrucţiuni 1;
elseif expresie_logica 2
  set de instrucţiuni 2;
....
else
  set de instrucţiuni n;
end

Exemplul 1. Să se genereze o matrice $A = (a_{ij})_{1 \leq i,j \leq n}$ cu n linii şi n coloane, ale cărei elemente să fie generate de regulile de mai jos:

$$A = \begin{cases} 1, \text{ dacă } i > j \\ -1, \text{ dacă } i < j \\ 2, \text{ dacă } i = j. \end{cases}$$

Soluţie.

```
n=3;
for i=1:n
 for j=1:n
     if i>j
        a(i,j)=1;
     elseif i<j
        a(i,j)=-1;
     else
        a(i,j)=2;
     end
 end
end
display(a)
```

Vom obţine rezultatele:

```
a =
 2  -1  -1
 1   2  -1
 1   1   2
```

⇝Pentru instrucţiunea `for`, avem următoarele sintaxă generală:

a) for contor = expresie
    set de instrucţiuni;
  end

b) for contor = valoare_iniţială:pas:valoare_finală
    set de instrucţiuni;
  end

Exemplul 2. Să se calculeze suma $S = 1 + 2^2 + 3^2 + ... + n^2$, pentru o valoare n citită de la tastatură.

Soluţie.

```
n=input('dati valoarea lui n:  ');
S=0;
for i=1:n
 S=S+i^2;
end
disp(S)
% numele scriptului ce conţine secvenţele de mai sus va fi exempluFor.m
% vom considera n=10
```

După compilarea script-ului, în fereastra de comandă se va afişa mesajul prin care se cere inserarea valorii dorite pentru n şi se va realiza calculul:

```
>> exempluFor
dati valoarea lui n:  10
 385
```

⇝Instrucţiunea `while` este folosită cu următoarea sintaxă generală:

```
while expresie
        set de instrucţiuni
end
```

Aici, `expresie` poate fi o expresie logică.

Exemplul 3. Să se genereze primele 10 pătrate perfecte consecutive folosind instrucţiunea while.

Soluţie.

```
nr=1;
while nr<=10
```

```
 pp=nr^2
 nr=nr+1;
end.
```

Alte instrucţiuni pe care le putem folosi în secvenţele corespunzătoare celorlalte instrucţiuni prezentate anterior sunt:

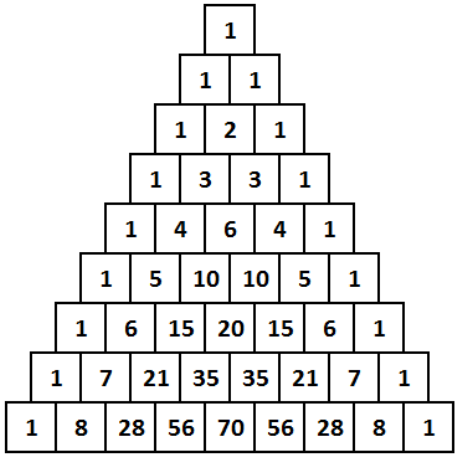
1) Instrucţiunea `break`: este folosită pentru a ieşi dintr-o buclă înainte ca aceasta să-şi încheie execuţia;

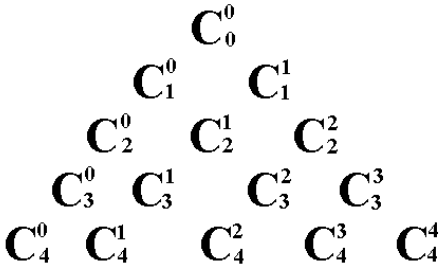
2) Instrucţiunea `error`: este folosită pentru a afişa mesaje la întâlnirea unei erori. Aceasta are sintaxa generală

`error('mesaj eroare')`, va afişa şirul 'mesaj eroare' şi va întrerupe forţat execuţia scriptului sau secvenţei în care apare.

Exemplul 4. Să se genereze în Matlab triunghiul lui Pascal.

Observaţie. Triunghiul lui Pascal este un aranjament geometric al coeficienţilor binomiali, numit astfel în onoarea matematicianului francez Blaise Pascal (19 iunie 1623 – 19 august 1662) deoarece el a fost prima persoană care a descoperit importanţa tuturor modelelor din componenţa acestuia. Acest triunghi a fost prima oară descoperit de matematicianul chinez Jia Xian undeva prin secolul al XI-lea! Înălţimea şi laturile triunghiului conţin cifra 1, iar fiecare număr de pe o linie n reprezintă suma celor 2 numere de pe linia superioară $n-1$.





* http://gandirelogica.blogspot.ro/2012/01/triunghiul-lui-pascal.html

Soluţie.

```
function A=TriunghiPascal(n)
%TriunghiPascal - triunghiul lui Pascal
A=zeros(n,n+1);
A(:,1)=ones(n,1); A(1,2)=1;
for i=2:n
 for j=2:i+1
 A(i,j)=A(i-1,j-1)+A(i-1,j);
 end
end
```

2 Recursivitate în Matlab

Funcţiile pot fi recursive, adică ele se pot autoapela, direct sau indirect. Recursivitatea este un instrument puternic, deşi nu toate calculele descrise în manieră recursivă pot fi implementate întotdeauna eficient în mod recursiv.

Exemplul 1. Să se implementeze în Matlab algoritmul lui Euclid de determinare a celui mai mare divizor comun a două numere.
Soluţie.

```
function g=DeterminCMMDC(m,n)
%DeterminCMMDC - functie pt determinarea celui mai mare divizor
comun
g=cmmdc(m,n);

function g=cmmdc(m,n)
%CMMDC - cel mai mare divizor comun
%apel g=cmmdc(m,n)
%m,n intregi
%pentru o implementare profesionala vezi functia MATLAB GCD
if ~isequal(m,round(m))||~isequal(n,round(n))
 error('argumentele nu sunt intregi');
end
if (m==0) & (n==0),
 g=0;
 return
end
if mod(m,n)==0
 g=n;
 return
else
 g=cmmdc(n,mod(m,n));
end
```

3 Probleme propuse

1. Calculaţi în mod eficient suma

$$S_n = \sum_{k=1}^{n} \frac{1}{k^5},$$

pentru $k = \overline{1,100}$.
2. Să se determine valoarea lui $n!$, pentru o valoare n număr natural citit de la tastatură ($n! = 1 \cdot 2 \cdot 3 \cdot ... \cdot n,\ 0! = 1$).
3. Să se calculeze suma

$$S = 1 \cdot 1! + 2 \cdot 2! + ... + n \cdot n!,$$

pentru $n \in \mathbb{N}^*$ citit de la tastatură.
4. Scrieţi un program care să afişeze numerele dintre 10 şi 100 care se divid cu 15.
5. Să se genereze o matrice pătratică de ordin $n \in \mathbb{N}$ în care valoarea fiecărui element corespunde produsului indicilor de linie şi coloană corespunzători poziţiei elementului în matrice. De exemplu, pentru $n = 3$, s-ar obţine matricea:

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 3 & 6 & 9 \end{bmatrix}.$$

6. Scrieţi un program care determină dacă un număr este prim.
7. Să se scrie un program care afişează toate numerele prime mai mici decât un număr dat $N \in \mathbb{N}^*, N \geq 2$, citit de la tastatură.