Elemente de bază privind utilizarea pachetului de programe MATLAB

Pachetul de programe MATLAB (MATrix LABoratory) oferă un mediu performant având ca scop principal simplificarea modului de prelucrare numerică a datelor. Matlab-ul permite exprimarea soluţiilor unor aplicaţii cu caracter ştiinţific sau tehnico-ingineresc într-un limbaj simplu, apropiat de formularea analitică (matematica) a problemei în cauză. În acest mod, utilizatorul se poate concentra în primul rând asupra metodei de rezolvare fără a întâmpina dificultăţi majore în traducerea soluţiei în limbajul propriu MATLAB-ului.

Nucleul MATLAB încorporează un număr mare de funcţii predefinite destinate operaţiilor matematice de uz general (aritmetica în virgulă mobilă, calcul polinomial, funcţii trigonometrice, exponenţiale, logaritmice, speciale, etc.), manipulării tablourilor şi matricelor, rezolvării unor probleme de analiză numerică, reprezentări grafice 2D şi 3D, dezvoltării de programe în limbaj MATLAB şi interfaţării acestora cu sistemul de operare gazdă. La aceste funcţii predefinite disponibile în nucleul de bază se adaugă un număr însemnat de pachete de programe (toolbox-uri) acoperind domenii specifice (procesarea semnalelor, teoria controlului, statistica, reţele neurale, calcul simbolic, procesarea imaginilor, optimizare, etc.). Aceste toolbox-uri, ca şi programele utilizatorilor, sunt fişiere sursă scrise într-un limbaj de programare simplu propriu mediului MATLAB.

Nucleul MATLAB poate opera atât în regim de linie de comandă - execuţia imediată a comenzii introduse de utilizator - cât şi de interpretor - execuţia unui program aflat în formă sursă într-un fişier. Această particularitate a sporit extensibilitatea limbajului permiţând utilizatorilor să adauge funcţii proprii la setul deja existent (predefinit sau cele din toolbox-urile de firma). Flexibilitatea limbajului este avantajată şi de existenţa unui singur tip de dată de bază, tipul matriceal având ca elemente valori reale sau imaginare. Spre deosebire de alte limbaje, variabilele nu trebuie declarate şi nici dimensionate.

Varianta actuală a nucleului MATLAB este scrisă în întregime în C şi poate rula pe diferite tipuri de sisteme de calcul (IBM- PC, VAX, SUN, Macintosh). Ea oferă facilităţile unui sistem complet integrat având la bază un limbaj propriu şi un interpretor performant. Aceasta permite ca exprimarea unui algoritm relativ complicat să se facă într-o formă compactă, execuţia acestuia să se desfăşoare într-un timp acceptabil şi cu o acurateţe ridicată, iar afişarea soluţiei să se realizeze într-un format convenabil, alfanumeric, eventual însoţit de o formă grafică.

1. Elemente fundamentale de utilizarea a nucleului MATLAB

1.1. Lansarea în execuţie şi interfaţarea cu sistemul de operare

În varianta existentă pe reţeaua din laborator, nucleul Matlab se lansează în lucru prin intermediul shortcut-ului cu numele Matlab. Lansarea în execuţie se va face din directorul curent (d:\Matlab\users) şi tot aici se vor salva fişierele temporare necesare reluării unei sesiuni MATLAB.

Odată ajunşi sub controlul nucleului MATLAB, prompterul implicit este ">". Nucleul acceptă introducerea unei comenzi MATLAB interne sau externe. Comenzile de tip extern sunt de fapt funcţii scrise în limbaj MATLAB (forma sursă) care sunt încărcate, interpretate şi executate de nucleu. Execuţia unei comenzi externe poate fi realizată dacă fişierul sursă cu numele corespunzător şi extensie .M se afla în directorul curent sau în unul din directoarele specificate în calea de căutare a MATLAB- ului.

Exemplu:

- afişarea directorului de lucru: *dir*
- creerea unui fişier text se face cu ajutorul editorului notepad; fie acesta test.m
- lansarea lui în lucru: *test.*

În timpul unei sesiuni MATLAB, utilizatorul poate crea şi atribui valori unui număr de variabile aflate în memoria internă afectată nucleului. Pentru a salva aceste valori şi a relua ulterior execuţia cu acelaşi context de lucru se pot utiliza comenzi de salvare/restaurare de forma:

>*save* <nume-fişier> - salvează variabilele în fişierul cu numele precizat

>*load* <nume-fişier> - restaurează contextul de lucru folosind fişierul precizat

Extensia implicită pentru aceste fişiere este .MAT. Dacă numele fişierului este omis atunci implicit se foloseşte fişierul MATLAB.MAT.

Funcţia "help" permite obţinerea unor informaţii cu caracter general despre comenzile interne şi externe MATLAB. Ea poate fi apelată în mai multe forme:

- *help* - oferă informaţii despre elementele limbajului MATLAB şi a fişierelor .M din directorul curent.
- *help* <nume-funcţie> - oferă informaţii despre funcţia în cauză.

Odată cu livrarea "toolbox"-urilor MATLAB, sunt încorporate şi o serie de fişiere .M cu meniuri demonstrative privind capabilităţile diferitelor grupe particulare de funcţii. Lansarea meniului demonstrativ se face cu comanda: *demo*

1.2. Variabile şi spaţiu de lucru

În cadrul unei sesiuni MATLAB, deci atâta timp cât ne aflam sub prompter ">", putem defini variabile de tip constantă numerică, vector sau matrice, dându-le nume diferite, desemnate atât cu majuscule cât şi cu minuscule (MATLAB-ul este "case-sensitive").

Tastarea returului de car <CR> va determina afişarea rezultatului sau acţiunii liniei introduse (principiul interpretorului). Dacă rezultatului liniei introduse nu i s-a desemnat un nume prin atribuiri explicite: "variabila=expresie", deci are doar forma "expresie", va fi afişat numele "ans" (answer) ce desemnează ultimul rezultat nenominalizat (valoarea expresiei). Cu excepţia salvărilor explicite, spaţiul de lucru ce conţine toate variabilele de lucru dintr-o sesiune MATLAB, se pierde odată cu părăsirea voită sau accidentală a mediului.

Vizualizarea, din cadrul unei sesiuni, a spaţiului de lucru se face cu comanda: *who*, iar a listei fişierelor .M şi .MAT din directorul curent cu comanda: *what* .

Comanda *exist('nume')* atestă prezenţa variabilei *'nume'* în spaţiul de lucru prin returnarea valorii 1; în caz contrar se va returna 0, iar dacă 'nume' este numele unui fişier .M se va returna valoarea 2.

În spaţiul de lucru sunt create implicit o serie de variabile şi constante cu caracter special după cum urmează:

- ans -cel mai recent răspuns;
- eps -precizia relativă în virgulă flotantă;
- pi = 3.1415926;
- inf –infinit;
- NAN - simbol folosit pentru un număr ce nu poate fi reprezentat;
- flops - numărul de operaţii în virgulă mobilă efectuate;
- nargin - numărul de argumente de intrare ale funcţiei;
- nargout - număr de argumente de ieşire ale funcţiei.

1.3 Introducerea şi manipularea vectorilor şi matricilor

Masivele (vectori şi matrici) pot fi introduse în MATLAB astfel:

1. ca lista explicită de elemente;
2. construite cu ajutorul funcţiilor şi instrucţiunilor specifice;
3. create cu ajutorul fişierelor .M;
4. încărcate din fişiere externe.

Spaţiul de memorare se aloca automat, la orice nouă definire de variabilă, până la completarea spaţiului de lucru, a cărui dimensiune depinde de calculatorul folosit.

Cel mai simplu mod de introducere a matricilor este introducerea elementelor pe linie. Separatorul dintre linii este ; , iar întregul masiv va fi cuprins între paranteze pătrate [...]: *> A = [1 2 3; 4 5 6; 7 8 9]*.

Orice instrucţiune MATLAB urmată de ';' va determina inhibarea afişării pe ecran a rezultatului respectivei instrucţiuni; în caz contrar acesta, sau acţiunea specifica, vor fi afişate.

Elementele unei matrici pot fi şi expresii:

*>x = [-1.3 sqrt(3); (1+2+3)*5 4/5]*

sau chiar masive:

>r = [10 11 12]

>A = [A; r]

Elementele unui masiv sunt nominalizate prin înscrierea poziţiei elementului în paranteze rotunde. Comanda: *>A(1,2)*

furnizează elementul de pe linia 1, coloana 2. Putem referi submasive mai mici în cadrul unor masive mai mari astfel: *>A=A(1:3,:)*

extrage submatricea formata din liniile 1, 2, 3 indiferent de indicele coloanei.

Vectorii pot fi generaţi prin comenzi de forma: *>x=1:5*

care duce la generarea unui vector x=[1 2 3 4 5].

Dacă este necesar se poate specifica incrementul, de exemplu:

>y=0 : pi/4 : pi

furnizează un vector y=[0.000 0.7854 1.5706 2.3562 3.1416].

1.4. Operatori aritmetici, funcţionali şi relaţionali

Operatorii aritmetici de baza pentru lucrul cu expresii şi masive sunt:

+ adunare;
- scădere;
* înmulţire;
/ împărţire dreapta;
\ împărţire stânga;
^ ridicare la putere;

Împărţirea / şi \ este analogă pentru expresii, dar provoacă ieşiri diferite în cadrul calculului matricial:

- A\B este echivalent cu înmulţirea la stânga cu inversa lui A (sau soluţia ecuaţiei A*X=B);
- A/B este echivalent cu înmulţirea la dreapta cu inversa lui A (sau soluţia ecuaţiei X*A=B);

În afara acestor operatori exista şi operatori specifici lucrului cu masive:

a) operatorul de transpunere ('). Comenzile:
   >A = [1 2 3; 4 5 6; 7 8 9]
   >B = A'
   vor determina ca B sa fie egal cu: B = [1 4 7; 2 5 8; 3 6 0]
   Dacă A este o matrice complexă, atunci A' va produce matricea transpusă, complex-conjugată. Pentru a obţine doar transpusa, fără conjugare, se recomandă folosirea comenzii: *conj*(*A'*).
b) radical. Comanda: *>sqrt*(*A*)
   extrage radicalul din fiecare element al masivului;
c) exponenţiere. Comanda: *>exp*(*A*)
   realizează operaţia de exponenţiere fiecărui element al masivului;
d) logaritmare. Comanda: *>log*(*A*)
   determina obţinerea logaritmului natural al elementelor lui A,
   iar comanda *>log10*(*A*)
   este varianta pentru logaritmare în baza 10;
e) *expm*(*A*) - este matricea exponenţiala a lui A;
f) *det*(*A*) - determinantul matricii A;
g) *trace*(*A*)- urma matricii A;
h) *inv*(*A*) - inversa matricii A;
i) *rank*(*A*) - rangul matricii A;
j) *size*(*A*) - determina dimensiunea lui A;
k) *length*(*A*) - pentru argument vector, va returna lungimea sa;

Acestora li se mai adaugă operatorii de câmp:

a) .* (înmulţire cu punct) - are ca efect înmulţirea a doua masive, de aceeaşi dimensiune, element cu element. Exemplificare:
   >x = [1 2 3]; y = [4 5 6];
   >z = x . y*
   z = [4 10 18]
b) ./ (împărţire cu punct, la dreapta).;
c) .\ (împărţirea cu punct, la stânga). Exemplificare:
   >z = x .\ y; z = [4.0000 2.5000 2.0000]
d) .^ (ridicarea la putere cu punct) - va ridica la puterea indicată fiecare element al unui masiv;

Exista 6 operatori relaţionali pentru compararea a 2 matrici de aceeaşi dimensiune; compararea se face pe perechi de elemente corespunzătoare:

a) < mai mic;
b) <= mai mic sau egal;
c) >= mai mare sau egal;
d) == egal;
e) ~= diferit;

Aceşti operatori sunt valabili şi pentru variabilele de tip constantă sau expresie.

1.5. Manipularea matricilor

În continuare vor fi prezentate câteva comenzi ce acţionează asupra matricilor în întregime sau numai asupra unor porţiuni a acestora:

a. *rot90* - rotirea cu 90 grade;

b. *diag*(*A*, *k*) - extrage elementele de pe diagonala *k* a unei matrici sau formează o matrice cu elementele desemnate în vectorul *A* plasate pe diagonala *k*
c. *tril*(*A*) - reţine elementele de sub diagonala principală, zerorizându-le pe celelalte;
d. *triu*(*A*) - reţine elementele de deasupra diagonalei principale, zerorizându-le pe celelalte.

Operatori pe coloană acţionează rând pe rând, asupra elementelor fiecărei coloane a unui masiv. Rezultatul este un vector linie:

a) *max*(*A*) - determină elementul maxim pe fiecare coloană;
b) *min*(*A*) - determină elementul minim pe fiecare coloană;
c) *mean*(*A*) - determină valoarea medie pe fiecare coloană;
d) *median*(*A*) - determină valoarea mediană pe fiecare coloană;
e) *sum*(*A*) - suma elementelor pe coloană;
f) *prod*(*A*) - produsul elementelor pe coloană;
g) *cumsum*(*A*) - suma cumulativă a elementelor pe coloană;
h) *cumprod*(*A*) - produsul cumulativ a elementelor pe coloană;
i) *cov*(*A*) - matricea de covarianţă (fiecare coloană este considerată o variabilă şi fiecare linie o observaţie);
j) *std*(A) - deviaţia standard (*sqrt*(*diag*(*cov*(*x*))));

Observaţie:
Dacă argumentul din funcţiile de mai sus este un vector, rezultatul este un scalar.

Generarea matricilor utile se face folosind următoarele funcţii :

a) *zeros*(*M*, *N*) - generează o matrice nulă cu *M* linii şi *N* coloane (la matricile pătrate se poate omite numărul de coloane);
b) *ones*(*M*, *N*) - generează o matrice cu elemente unitare;
c) *rand*(*M*, *N*) - generează o matrice cu elemente aleatoare;
d) *eye*(*M*, *N*) - generează o matrice identitate.

1.6. Comenzi speciale

Facilitează lucrul în cadrul sesiunii de lucru :

a) clear *X* - şterge variabila *X* din spaţiul de lucru;
b) clear - şterge toate variabilele din spaţiul de lucru;
c) clc - şterge ecranul de comanda;
d) ... - semnul de continuare a unei instrucţiuni pe a doua linie;
e) % - plasat la începutul liniei, desemnează o linie de tip comentariu;
f) disp - afişarea datelor şi mesajelor sub forma :
disp(A) - afişează matricea *A*;
disp('mesaj') - afişează mesaj.

1.7. Exerciţii

a. Generarea un vector x cu elementele cuprinse între 0.0 şi 3.0, cu pasul de incrementare de 0.2: x = (0.0:0.2:3.0)';
b. Generarea unui vector y ale cărui elemente să fie produsul dintre:
   - sinusul elementului corespunzător din primul vector;
   - rezultatul exponenţierii elementului luat cu semn schimbat din primul vector: *y = sin(x).*exp(-x);*
c. Să se formeze matricea ce are drept coloane vectorii *x* şi *y: A=[x y];*
d. Să se determine dimensiunea acestei matrici: *size(A);*
e. Sa se anuleze, în matrice, elementele de pe linia 5: *A(5,:) = [0 0]*

1.8. Probleme propuse

a. Să se genereze o matrice pătratica, de dimensiune 4, ce are elemente aleatoare, uniform distribuite în intervalul (0,1).
b. Să se determine rangul acestei matrici.
c. Să se determine determinantul acestei matrici.
d. Să se transpună elementele matricii.
e. Să se rezolve sistemul : *A * X = b,* unde: *A* = [1 2 3; 4 5 6; 7 8 0] şi *b*= [5; 8; -7]
f. Să se formeze vectorul ce are ca elemente elementele de pe diagonala principală a lui A.
g. Să se genereze vectorii: *x*=[-1 0 2]' şi *y* = [-2 -1 1]' şi să se realizeze operaţiile:
   - *A = x * y'*
   - *B = y * x'*

- Înmulţirea elementelor lui *x* cu constanta *pi*;
- Ridicarea lui *B* la puterea a doua;
- Determinarea inversei lui *A*;
- Determinarea matricii ce are ca elemente radicalul pătratic din elementele lui *A*;
- Partea superior triunghiulară a lui *A*;
- Maximul valorilor medii pe coloane pentru *A* şi *B*;
- Elementele maxime pe linii, respectiv pe coloane şi maximul absolut pentru matricele *A*, *B*;
- Urma matricelor *A*, *B*;

h. Să se construiască o matrice pătratică, de ordin 5, tridiagonală, cu elementele unitare, de forma:

$$\mathbf{z} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$ folosind o singură comanda MATLAB.

2. Facilităţi MATLAB

Se urmăreşte o trecere în revistă a facilităţilor grafice şi de lucru cu fişiere în cadrul mediului MATLAB. De asemenea, se vor detalia instrucţiunile ce exercită controlul asupra execuţiei comenzilor, formatele de editare şi modul în care se creează şi se operează cu fişierele .M.

2.1. Structuri pentru controlul execuţiei comenzilor

2.1.1. Bucla for

Forma generală a unei bucle **for** este:

for *variabila=start* **:** *pas* **:** *final*
    instrucţiuni
end

unde:

- *variabila* reprezintă variabila de contorizare a buclei;
- *start* reprezintă valoarea atribuită iniţial variabilei de contorizare;
- *pas* reprezintă pasul de incrementare al variabilei de contorizare;
- *final* reprezintă valoarea finală atribuită variabilei de contorizare;
- *instrucţiuni* reprezintă o succesiune de una sau mai multe comenzi MATLAB;
- **end** marchează sfârşitul buclei **for.**

Dacă instrucţiunile sunt urmate de caracterul "**;**" se inhibă afişarea rezultatelor intermediare din bucla **for** (se recomandă folosirea lui).

Exemplu:

```
n=10;
for i=1:n
        x(i)=i^2;
end;
```

Pot exista şi situaţii în care este necesară folosirea mai multor bucle **for** imbricate; în aceste cazuri buclele trebuie să fie strict incluse una în cealaltă şi să respecte forma generală de scriere.

Exemplu:

```
m=6;   n=8;
for i=1:m
        for j=1:n
                A(i,j)=1/(i+j-1);
        end
end
A
```

Observaţie:

Neintroducerea terminatorului **end** în bucla **for** determină neexecuţia buclei. Sistemul rămâne în stare de aşteptare până la introducerea lui **end**.

2.1.2. Bucla WHILE

Aceasta permite ca o instrucţiune sau un grup de instrucţiuni să fie repetat de un număr neprecizat de ori în funcţie de o condiţie logică. Forma sa generală este:

while *condiţie*
    instrucţiuni
end

Exemplu:

Să se determine primul număr întreg *n,* pentru care factorialul (n!) este un număr de 5 cifre:

```
n=1;
while prod(1:n)<1.e4
    n=n+1;
end
n
```

Bucla va fi executată atât timp cât condiţia (*n*!<1.e4) este îndeplinită.

2.1.3. Instrucţiunea IF

Instrucţiunea IF este folosită pentru luarea unei decizii multiple. Forma sa generală este:

if *condiţie-1*
    instrucţiuni-1
elseif *condiţie-2*
    instrucţiuni-2
. . . . . .
else
    instrucţiuni-rest
end

iar forma cea mai simplă:

if *condiţie*
    instrucţiuni
end

Dacă *condiţia-1* este îndeplinită, se vor executa *instrucţiuni-1*; dacă însă ea nu este îndeplinită, se testează *condiţia-2*; dacă aceasta este satisfăcută, se va executa grupul 2 de instrucţiuni ş.a.m.d. Dacă nici una din condiţii nu este îndeplinită, se execută grupul *instrucţiuni-rest*.

În general condiţia apare sub forma:

expresie operator-relaţional expresie

unde operatorul relaţional poate fi: <, <=, >, >=, ==, ~=.

Exemplu:

Să se construiască o matrice pătratică de dimensiune *n* = 4 în care elementele de pe diagonala principală sunt egale cu 2, cele de pe diagonala de deasupra şi de dedesubtul diagonalei principale sunt egale cu 1 iar restul cu 0.

```
for i=1:n
    for j=1:n
        if i==j
            a(i,j)=2;
        elseif abs(i-j)==1
            a(i,j)=-1;
        else
            a(i,j)=0;
        end
    end
end
```

Observaţie:

Ori de câte ori este posibil, în vederea obţinerii vitezei maxime de lucru sub MATLAB, se încearcă transformarea buclelor **for** şi **while** în operaţii matriciale.

Exemplu:

Secvenţa de mai jos poate fi transformată astfel:

```
i=0;
   for t=0:0.01:10
     i=i+1;
     y(i)=sin(t);
   end
```

```
t=0:0.01:10;
y=sin(t);
```

O astfel de transformare determină o creştere a vitezei de lucru de aproximativ 25 de ori. O altă metodă de mărire a vitezei de lucru este cea de prealocare a oricăror vectori pentru care se doreşte determinarea elementelor.

Exemplu:

```
y=zeros(1,100);
for i=1:100
   y(i)=2*i^2;
end
```

Fără prealocare, interpretorul MATLAB-ului redimensionează, la fiecare pas din **for** vectorul iniţial mărindu-i dimensiunea cu o unitate; acest pas este eliminat la prealocare, memoria fiind utilizată mult mai eficient. Principalul inconvenient în memorarea în spaţiul de lucru derivă din faptul că memoria tinde să devină fragmentată în cursul unei sesiuni de lucru, astfel încât, deşi există suficient spaţiu de stocare, acesta nu este contiguu pentru a permite stocarea unei variabile de dimensiune mare. Spaţiul de memorare disponibil, afişat în urma lansării comenzii **who** reprezintă spaţiul contiguu disponibil pentru memorare; spaţiul disponibil total este sensibil mai mare. Acest spaţiu devine vizibil mai mare dacă ştergem variabila de dimensiune maximă din spaţiul de lucru.

2.1.4. Instrucţiunea BREAK

Această instrucţiune determină ieşirea forţată dintr-o buclă **for**, **while** sau dintr-un **if**.

Exemplu:

Să se determine precizia de calcul pentru maşina de lucru:

```
eps=1;
for i=1:1000
      eps=eps/2;
      if (eps+1<=1)
        break;
      end
end
eps=eps*2;
```

2.2. Fişiere de comenzi indirecte (.M) şi fişiere funcţii

MATLAB-ul este capabil să execute secvenţe de comenzi memorate în fişiere cu extensie .M. Un fişier.M este format dintr-o succesiune de instrucţiuni MATLAB ce pot include şi referiri la alte fişiere .M, incluzând chiar fişierul în cauză (apel recursiv). Lansarea în lucru a unui astfel de fişier se face prin introducerea numelui său. În acest fel se pot memora şi executa secvenţe mai lungi de comenzi fără ca ele să fie introduse de fiecare dată de la tastatură. Prelucrările efectuate afectează în mod global zona de variabile.

Există şi un al doilea tip de fişiere .M, care permite extinderea pachetului MATLAB şi construire de noi funcţii. În acest caz se specifică argumentele funcţiei iar variabilele folosite au caracter local. Ambele tipuri de fişiere .M sunt fişiere text ASCII şi pot fi create cu ajutorul unui editor de texte.

2.2.1. Fişiere de comenzi indirecte ("script files")

La apelarea unui fişier script, MATLAB-ul va executa instrucţiunile cuprinse în acesta, linie cu linie, neaşteptând nici o comandă de la tastatură.

Exemplu:

```
%se calculează numerele lui Fibonnaci
      f=[1 1];
       i=1;
      while f(i)+f(i+1)<1000
            f(i+2)=f(i)+f(i+1);
            i=i+1;
      end
```

După execuţia unui fişier.M controlul revine MATLAB-ului. Meniurile demonstrative încorporate în MATLAB reprezintă un exemplu de folosire a fişierelor.M.

2.2.2. Fişiere funcţii

Dacă un fişier.M specifică o funcţie, atunci prima linie trebuie să conţină cuvântul **function**. Un fişier funcţie se deosebeşte de un fişier document prin faptul că el poate să transfere argumente cu spaţiul de lucru, iar variabilele definite în interiorul său sunt locale, neoperând global asupra spaţiului de lucru.

Exemplu:

Funcţia MEAN ce calculează valoarea medie, pe coloane, într-o matrice:

```
function y=mean(x)
% MEAN calculează valoarea medie a unui vector sau media, pe
% coloane, intr-o matrice; in ultimul caz va returna  un vector linie
[m,n]=size(x);
if m==1
    m=n  % izolam cazul vector linie
end
y=sum(x)/m;
```

Exemplu de apel:
```
>z=1:99;
>mean(z)
>ans=50
```

Exemplul prezentat pune în evidentă următoarele:

- prima linie declară numele funcţiei, argumentele de intrare şi cele de ieşire; fără această linie, fişierul ar fi de tip **script**;
- simbolul % marchează restul liniei drept comentariu; comentariile vor fi afişate la introducerea comenzii "help mean";
- variabilele m,n sunt locale funcţiei **mean** şi nu vor exista în spaţiul de lucru după părăsirea funcţiei (sau, dacă există cu acest nume, nu vor fi schimbate prin apelul funcţiei);
- la apelul lui **mean**, funcţiei i se transmite argumentul efectiv z, pe poziţia parametrului formal corespunzător x.

Observaţie:

Există 2 variabile permanente nedetaliate la descrierea spaţiului de lucru:

- nargin - numărul argumentelor de intrare într-o funcţie;
- nargout - numărul argumentelor de ieşire dintr-o funcţie;

În momentul apelului pentru prima dată intr-o sesiune de lucru a unui fişier funcţie, acesta va fi compilat şi plasat în memorie. El va putea fi ulterior reapelat, fără a fi necesară o recompilare. Codul respectiv va fi păstrat în memoria de lucru pe durata întregii sesiuni dacă lipsa de spaţiu nu face necesară ştergerea sa. La un apel oarecare în cadrul unei sesiuni de lucru a unui nume oarecare, nucleul MATLAB face următoarele verificări:

- verifică dacă acesta este o variabilă în spaţiul de lucru;
- verifică dacă acesta este o funcţie;
- verifică daca numele coincide cu numele unui fişier.M în:
  - directorul curent;
  - calea de directoare specificată in MATLABPATH.

2.2.3. Comenzi utile in lucrul cu fişiere funcţii

În mod normal, execuţia unui fişier.M nu determină afişarea comenzilor sale pe ecranul de comandă. Pentru a produce această imprimare se foloseşte una din comenzile:

```
>echo
>echo on        % activeazăafişarea in ecou;
>echo off       % inhibă afişarea in ecou;
```

Aceasta poate fi util pentru vizualizare, demonstraţii şi depanare. Furnizarea datelor de către utilizator, atât în sesiunea de lucru, cât şi în interiorul funcţiilor se face prin comanda: *variabila* = input('*mesaj*'), unde:

- *variabila* reprezintă variabila care va căpăta ca valoare expresia introdusă de utilizator;
- *mesaj* reprezintă mesajul afişat pe monitor înainte de citirea unei valori.

2.3. Funcţii grafice în MATLAB

2.3.1. Grafica în coordonate rectangulare

Dacă y este un vector, comanda: **plot(y)** produce afişarea elementelor lui y în funcţie de indexul elementelor.

Observaţie: Datele sunt autoscalate şi axele x şi y sunt desenate pe ecran. Desenul este afişat pe ecranul grafic; pe sistemele cu varianta MATLAB sub Windows ambele ecrane - de comanda şi grafic- pot fi vizualizate simultan.

Acestei comenzi de desenare simplă i se pot adaugă:

- inscriptionarea unui titlu;
- etichetarea axelor;
- trasarea verticalelor şi orizontalelor în dreptul fiecărei diviziuni de pe axe;

Exemplu:

```
y = [0 48 84 1 91 6 14];
plot(y);
title('Primul desen'), xlabel('Abscisa'),
ylabel('Ordonata'), grid
```

Dacă *x* şi *y* sunt vectori de aceeaşi lungime, comanda: *plot(x,y)* desenează elementele lui *y* funcţie de elementele lui *x*.

Exemplu:

```
t = 0:0.05:4*pi;
y = sin(t);
plot(t,y)
```

Pentru a suprapune mai multe grafice pe aceeaşi fereastră, se va proceda astfel:

- dacă y este o matrice şi x este un vector, plot(x,y) va trasa graficele corespunzătoare liniilor sau coloanele lui y în funcţie de vectorul x, folosind caractere diferite pentru fiecare dintre ele;
- dacă x este o matrice şi y un vector se aplică o regulă similară trasându-se graficele corespunzătoare liniilor lui x în funcţie de vectorul y;
- dacă x şi y sunt matrici de aceeaşi dimensiune, se va trasa graficul corespunzător coloanelor lui y funcţie de coloanele lui x.

Un alt mod de a realiza desene multiple este exemplificat prin comanda:

*plot***(***x1,y1,x2,y2,...,xn,yn***)**

unde *(x1,y1),(x2,y2),...* sunt perechi de vectori. În acest fel se obţine reprezentarea grafică pentru fiecare pereche (x,y). cest tip de grafic are avantajul că permite afişarea simultană a vectorilor de lungimi diferite. Fiecare pereche de vectori va folosi un alt tip de linie pentru afişare. Se poate indica tipul de caracter cu care se face trasare. Valorile posibile sunt:

solid	-
intrerupt(dashed)	--
doua puncte(dotted)	:
linie-punct(dashdot)	-.
punct(point)	.
plus	+
stea(star)	*
cerc(circle)	o
x-mark	x

Pentru monitoare color se poate utiliza o opţiune de culoare: 'w' (white), 'r' (red), 'g' (green), 'b' (blue).

2.3.2. Grafică în coordonate polare şi logaritmice

Această facilitate permite folosirea altor tipuri de coordonate. Funcţiile specifice sunt similare cu cele prezentate mai sus.

- *polar(theta,rho)* - reprezintă grafic, în coordonate polare, unghiul "*theta*" (în radiani) în funcţie de raza "*rho*";
- *loglog(x,y)* - realizează grafice în coordonate logaritmice pentru ambele axe;
- *semilogx(x,y)* - realizează grafice reprezentând pe *x* în coordonate logaritmice şi pe *y* liniar.
- *semilogy(x,y)* - analog ca mai sus inversând *x* cu *y***.**

2.3.3. Suprafeţe mesh 3-D şi grafice de tip contur

Comanda următoarea creează imaginea tridimensionala a matricii Z: *mesh(Z)*.

O suprafaţă mesh este definită prin coordonatele de cotă z determinate de elementele matricei deasupra unei suprafeţe x-y, plane. Funcţia *mesh* poate fi astfel folosită pentru vizualizarea matricilor de dimensiune mare, imposibil de reprezentat sugestiv prin forma numerică. De asemenea, această comandă poate fi folosită pentru reprezentarea grafică a funcţiilor ce depind de două variabile: z=f(x,y). Pentru a putea

face reprezentarea se creează mai întâi, o reţea plană, echidistant divizată pe cele două axe sub forma unei matrici:

Exemplu:

```
x = -8:0.5:8;
 y = x';
X = ones(y)*x;  % creează o matrice cu linii identice
Y = y*ones(x);
R =sqrt(X.^2 +Y.^2)+eps; % conţine distanţele de la centrul suprafeţei plane;
Z = sin(R)./R;
mesh(Z)
```

Generarea matricilor *X* şi *Y* poate fi optimizată prin apelul funcţiei *meshdom*.

Observaţie:

Opţiunile comenzii mesh permit diferite unghiuri de vedere şi depărtări de suprafaţa generată.

O altă categorie de grafice îl constituie cele de tip "contur". Efectul lor se poate observa după secvenţa de forma:

```
p = rand(4,7);
contour(p)
```

2.3.4. Controlul ecranului

Comutarea între ecranul grafic şi cel de comandă precum şi divizarea ecranului grafic sunt disponibile în MATLAB cu ajutorul următoarelor comenzi:

- *shg* - comută pe ecranul grafic;
- *clg* - şterge ecran grafic;
- *home* - mută cursorul în coltul stânga-sus al ecranului (analoga clc);
- *subplot*(*m,n,p*) - subdivizează ecranul grafic; parametrii *m, n, p* au următoarea semnificaţie:
  m reprezintă numărul de grafice pe linie;
  n reprezintă numărul de grafice pe coloană;
  p reprezintă poziţia primului grafic inscripţionat în partiţia ecran realizată;

Exemplu:

```
subplot(2, 1, 1), plot(abs([2 -7 3 5 17 -25]))
subplot(2, 1, 2), plot(rand(1,10));
```

Observaţie:

1. Se pot scrie mai multe instrucţiuni pe aceeaşi linie, separate prin virgule.
2. În unele situaţii se renunţă la scalarea automată a graficului în favoarea unei scalări manuale, cu ajutorul comenzilor:
   ```
   v = [x_min, x_max, y_min, y_max];
   axis(v)
   ```
3. Pentru a evita efectul deformant introdus de ecran se poate utiliza comanda: `axis('square')`
4. Menţinerea unui grafic pe ecran în vederea suprapunerii unui alt grafic poate fi obţinută cu ajutorul funcţiei hold.

2.4. Probleme propuse

1. Să se scrie o funcţie MATLAB, *rangk*(*x*, *k*) care să permită determinarea elementului aflat pe rangul *k* (dacă *x* este un vector) sau a vectorului de elemente aflate pe rangul *k* în coloanele matricei (dacă *X* este o matrice).

```
function y=rangk(x, k)  % RANGK elementele de pe rangul K
% Daca X este un vector, RANGK(X,K) furnizează elementul de pe
% rangul K din X. Daca X este o matrice, RANGK(X,K) furnizează un vector %linie continând elementele de pe rangul K al fiecărei coloane
if nargin ~= 2
        error('argument missing');
end
[m,n]=size(x);
if(n == 1) x=x';
        temp=m;
        m=n;
        n=temp;
end
```

```
if(k > n | k < 1)
        error('invalid rang');
end
if (m==1)
        y=x(k);
else
        y=x(k,:);
end
```

2. Să se vizualizeze folosind acelaşi ecran grafic următoarele funcţii:

a) $f(x)=x^3+4x-15$; $g(x)=\sin x+\sin 2x+\sin 4x$;

b) $f(x)=e^{-1/x}\cos x$; $g(x)=\cos x+\cos 3x+\cos 5x$; $h(x)=tgx/x$,

c) $f(x)=\begin{cases}1+5x^{-1}-3x^{-4} & pentru\ x\in[1,50)\\ x^{1/2} & pentru\ x\in[50,100]\end{cases}$

3. Să se deseneze suprafaţa mesh şi contur pentru matricea 25x25:

- matrice de numere aleatoare uniform distribuite în intervalul (0,1);
- matrice de numere normal distribuite de medie zero şi dispersie 1;

Obs. Pentru generarea numerelor aleatoare se folosesc funcţiile Matlab *rand* şi *randn*.

4. Să se genereze un vector de numere aleatoare uniform distribuite în intervalul (1,100); să se împartă intervalul [1,100] în 10 intervale echidistante şi să se determine câte numere aleatoare sunt în fiecare din cele 10 intervale.

5. Să se scrie o funcţie MATLAB, *coincid*(*x*,*y*) care să primească ca argument doi vectori *x*, *y* şi să întoarcă numărul de coincidente între elementele acestora. Argumentele care lipsesc se vor prelua de la tastatură.

6. Să se modifice funcţia *RANGK* descrisă anterior astfel încât dacă numărul de argumente este 1, vectorul sau matricea *x* se citesc dintr-un fişier ASCII al cărui nume se preia de la tastatură. Verificaţi existenţa fişierului în cauză. (indicaţie: folosiţi funcţiile LOAD, EXIST.)