LUCRAREA 1
INTRODUCERE ÎN MATLAB

1.1. Introducere

MATLAB este un pachet de programe dedicat calculului numeric şi reprezentărilor grafice. Elementul de bază cu care operează este matricea, de aici provenind şi numele său: MATrix LABoratory. Resursele sale de calcul şi reprezentare grafică sunt bogate, permițând operații matematice fundamentale, analiza datelor, programare, reprezentări grafice 2D şi 3D, realizarea de interfețe grafice etc. Din punct de vedere al construcției sale, MATLAB este alcătuit dintr-un nucleu de bază în jurul căruia sunt grupate TOOLBOX-urile. Acestea reprezintă nişte aplicații specifice, fiind de fapt colecții extinse de funcții MATLAB care dezvoltă mediul de programare de la o versiune la alta, pentru a rezolva probleme din diverse domenii. În prelucrarea numerică a semnalelor cel mai des utilizat este toolbox-ul SIGNAL PROCESSING.

1.2. Comenzi şi funcții principale în MATLAB

MATLAB operează cu două tipuri de ferestre:
1. fereastra de comenzi;
2. fereastra pentru reprezentări grafice.

La deschiderea programului MATLAB va apărea pe ecranul calculatorului fereastra de comenzi, având în partea de sus bara de meniuri aferentă. Simbolul ">>" reprezintă prompterul MATLAB şi se află la începutul fiecărei linii de comandă din fereastra de comenzi. Ferestrele pentru reprezentări grafice vor apărea atunci când se va cere prin comenzi specifice afişarea unor grafice.

***Atenție*:** *Comenzile introduse anterior pot fi readuse în linia de comandă prin folosirea săgeților de la tastatură, ↑ şi ↓ (căutarea se face ca într-o "listă").*

1.2.1. Funcții MATLAB de interes general

- **`help`** – furnizează informații despre MATLAB şi funcțiile acestuia.
  Sintaxă:
  `help nume` – furnizează informații despre `nume` (poate fi un nume de funcție sau un nume de director).
  Exemplu:
  `help fft` – furnizează informații despre transformata Fourier discretă.

- **`who`** – listează numele variabilelor din spațiul de lucru.

- **`whos`** – furnizează informații suplimentare referitoare la variabilele din spațiul de lucru (nume, dimensiune etc.).

- **`format`** – stabileşte formatul extern de afişare al numerelor pe ecran.
  Sintaxa:
  `format opțiune` – parametrul `opțiune` poate fi:
  - `short` – 5 cifre // formatul implicit
  - `long` – 15 cifre
  - `short e` – 5 cifre + exp (puteri ale lui 10)
  - `long e` – 15 cifre + exp (puteri ale lui 10)
  - etc.

  Pentru mai multe informații tastați `help format`.
  Exemple:

```
format short
x=pi        →   x =
                    3.1416
format long
x           →   x =
                    3.14159265358979
```

  Să se verifice şi celelalte tipuri de format folosind aceeaşi valoare `x`.

- **`clear`** – şterge din memorie una sau mai multe variabile.
  Sintaxe:
  `clear v` – şterge din memorie variabila `v`.
  `clear v1 v2` – şterge din memorie variabilele `v1` şi `v2` (sintaxa e valabilă pentru oricâte variabile).
  `clear` – şterge din memorie toate variabilele definite până în acel moment.

- **`lookfor`** – listează toate numele de fişiere care au în prima linie a help-ului cuvintele menționate ca argument, precum şi prima linie din help.

Sintaxă:
`lookfor cuvânt` – listează toate numele de fişiere care conţin în prima linie a help-ului `cuvânt`, precum şi prima linie din help.
Exemplu:
`lookfor ifft` – listează toate numele de fişiere care conţin în prima linie a help-ului `ifft`, precum şi prima linie din help.

- **`dir`** – afişează numele tuturor fişierelor din directorul curent sau din orice alt director precizat ca argument.
  Sintaxe:
  `dir` – afişează numele tuturor fişierelor din directorul curent.
  `dir nume` – afişează numele tuturor fişierelor din directorul `nume`.

- **`cd`** – returnează numele directorului curent sau schimbă directorul de lucru.
  Sintaxe:
  `cd` – returnează numele directorului curent.
  `cd c:\matlab\nume_director` – schimbă directorul de lucru în `nume_director` (presupunând că MATLAB este instalat pe `c:\`).

***Atenţie*:** *Un anumit program MATLAB aflat într-un anumit director nu poate fi rulat decât dacă directorul respectiv este directorul de lucru*.

1.2.2. Variabile şi constante speciale în MATLAB

- **`ans`** – variabilă creată automat în care este returnat rezultatul unui calcul, atunci când expresia nu a avut asignat un nume.
  Exemplu (se va tasta direct în fereastra de comenzi):
  `3` → `ans = 3` // nu s-a alocat nici un nume
  `x=2` → `x = 2` // s-a alocat numele `x`
- **`pi`** – valoarea π.
- **`Inf`** – reprezentarea lui +∞.
- **`NaN`** – reprezentarea lui 0/0 sau ∞/∞.
- **`i`** sau **`j`** – folosite reprezentarea numerelor complexe.

***Atenție*:**
- *dacă după o linie de comandă urmează semnul " ; " atunci rezultatul nu va mai fi afişat* (*excepție fac comenzile grafice*);
- *dacă în fața unei linii de comandă se pune semnul " % " atunci se face abstracție de linia respectivă* (*este interpretă ca o linie de comentariu*);
- *dacă se doreşte continuarea unei instrucțiuni pe linia următoare se folosesc "…" urmate de* enter;

Verificați următoarele *exemple* (se va tasta direct în fereastra de comenzi):

```
p=pi;                 // nu se va afişa valoarea p (dar există în memorie)
q=pi/2                // va afişa valoarea q
%r=pi/4               // nu se ia în considerare această linie
v=r/2                 // va rezulta o eroare deoarece nu îl cunoaşte pe r
s=1+2+3+...  enter
4+5+6                 // instrucțiunea se continuă şi pe linia următoare
```

1.2.3. Matricea – element de bază în MATLAB

MATLAB lucrează numai cu un singur tip de obiecte şi anume matrice numerice, având elemente reale sau complexe. Astfel scalarii sunt priviți ca matrice de dimensiune 1 x 1, iar vectorii ca matrice de dimensiune 1 x n (dacă este vector linie) sau n x 1 (dacă este vector coloană).
Reguli privind modul de definire a matricelor:
- elementele matricei sunt cuprinse între paranteze drepte [ ]
- elementele unei linii se separă prin pauză (blanc) sau virgulă
- liniile matricei se separă prin ; sau *enter*.

Exemplu: Fie matricea $A=\begin{pmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{pmatrix}$ şi vectorii $B=\begin{pmatrix}7 & 8 & 9\end{pmatrix}$, $C=\begin{pmatrix}-1\\ -2\end{pmatrix}$

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

```
C=[-1;-2]          →        C =
                                -1
                                -2
```

Pentru o matrice M:
- `M(i,j)` reprezintă elementul din matricea M corespunzător liniei `i` şi coloanei `j`
- `M(i)` reprezintă elementul `i` din matrice, numărarea elementelor făcându-se pe coloane.

Pentru un vector v:
- `v(i)` reprezintă elementul de pe poziția `i` din vector.

Exemplu: Pentru A, B, C definite anterior verificați:

```
A(2,3)             →        ans =
                                 6
A(4)               →        ans =
                                 5
B(2)               →        ans =
                                 8
C(2)               →        ans =
                                -2
```

Dacă dorim să schimbăm elementele unei matrice sau să mai adăugăm alte elemente, fără a mai rescrie întreaga matrice, procedăm ca în exemplul următor:

```
A(2,3)=0           →        A =
                                 1     2     3
                                 4     5     0
A(3,3)=-3          →        A =
                                 1     2     3
                                 4     5     0
                                 0     0    -3
C(3)=6             →        C =
                                -1
                                -2
                                 6
```

Se pot construi matrice de dimensiuni mai mari pornind de la matrice de dimensiuni mai reduse. Pentru exemplificare vom folosi matricea A şi vectorii B şi C în ultima lor formă (A – 3 x 3, B – 1 x 3, C – 3 x 1):

```
D=[A;B]          →        D =
                              1     2     3
                              4     5     0
                              0     0    -3
                              7     8     9
```

// s-a construit matricea D de dimensiune 4 x 3, prin adăugarea vectorului B la matricea A (ca ultimă linie).
Atenție: *A şi B au acelaşi număr de coloane* (*3*) *pentru a fi posibilă construcția.*

```
E=[A,C]          →        E =
                              1     2     3    -1
                              4     5     0    -2
                              0     0    -3     6
```

// s-a construit matricea E de dimensiune 3 x 4, prin adăugarea vectorului C la matricea A (ca ultimă coloană).
Atenție: *A şi C au acelaşi număr de linii* (*3*) *pentru a fi posibilă construcția.*

Dacă v este un *vector* atunci:

- `v(i:k)` – selectează elementele de pe pozițiile i, i + 1, i + 2, … , k ale vectorului v; dacă i > k atunci vectorul rezultat este gol (nu are nici un element).
- `v(i:j:k)` – selectează elementele de pe pozițiile i, i + j, i + 2j, … până la k, ale vectorului v (selectează cu pasul j); dacă j > 0 şi i > k *sau* j < 0 şi i < k atunci vectorul rezultat este gol.
- `v([i,j,k])` – selectează elementele de pe pozițiile i, j şi k.
- `v(:)` – dacă vectorul este linie atunci el devine coloană; dacă vectorul este coloană atunci el rămâne nemodificat.

Dacă M este o *matrice* atunci:

- `M(:,j)` – selectează coloana j a matricei M.
- `M(i,:)` – selectează linia i a matricei M.
- `M(:,i:j)` – selectează coloanele de la i la j ale matricei M.
- `M(i:j,:)` – selectează liniile de la i la j ale matricei M.
- `M(:,i:j:k)` – selectează coloanele i, i + j, i + 2j , … până la k ale matricei M (selectează cu pasul j).
- `M(i:j:k,:)` – selectează liniile i, i + j, i + 2j , … până la k ale matricei M.
- `M(i:j,k:l)` – extrage submatricea formată cu elementele aflate la intersecția liniilor de la i la j şi coloanelor de la k la l ale matricei M.
- `M(:,[i,j,k])` – selectează coloanele i, j şi k ale matricei M.
- `M([i,j,k],:)` – selectează liniile i, j şi k ale matricei M.

- `M([i,j,k],[l,m,n])` – extrage submatricea formată cu elementele aflate la intersecția liniilor i, j şi k şi coloanelor l, m şi n ale matricei M.
- `M(:,:)` – selectează întreaga matrice M.
- `M(i:j)` – selectează elementele de i la j ale matricei M şi le pune sub forma unui vector linie (elementele într-o matrice se numără pe coloane).
- `M(:)` – selectează toate elementele matricei M şi le pune sub forma unui vector coloană (pune coloanele matricei M una sub alta, sub forma unui vector coloană).

Verificați sintaxele de mai sus folosind matricea A şi vectorii B şi C din exemplele anterioare sau construind alte matrice şi vectori.

1.2.4. Vectori şi matrice uzuale

- ***Generarea vectorilor cu pas liniar***

Sintaxe:
- `v=inițial:pas:final` – se generează un vector linie `v` cu elementele începând de la `inițial` la `final`, cu pasul egal cu `pas` (pasul poate fi şi negativ dar atunci valoarea inițială trebuie să fie mai mare decât valoarea finală).
- `v=inițial:final` – se generează un vector linie `v` cu elementele începând de la `inițial` la `final`, cu pasul egal cu 1.
- `v=linspace(minim,maxim,număr_de_elemente)` – se generează un vector linie v cu elementele începând de la `minim` la `maxim`, cu pas constant şi având un număr de elemente egal cu `număr_de_elemente`.

Verificați următoarele *exemple*:
```
v=3:7:40
u=5:10
d=17:-3:4
l=linspace(17,58,4)
q=linspace(pi,-pi,6)
```

- ***Generarea vectorilor cu pas logaritmic***

Sintaxe:
- `v=logspace(minim,maxim)` – se generează un vector linie `v` având 50 de elemente distribuite logaritmic între 10^{minim} şi 10^{maxim}.
- `v=logspace(minim,maxim,număr_de_elemente)` – se generează un vector linie `v` având `număr_de_elemente` elemente distribuite logaritmic între 10^{minim} şi 10^{maxim}.

***Atenție*:** *Dacă* `maxim=pi` *atunci elementele vor fi distribuite logaritmic între* 10^{minim} *şi* π.

Verificați următoarele *exemple*:
```
g=logspace(1,2)
r=logspace(1,pi)
h=logspace(1,2,5)
k=logspace(0,pi,5)
```

- ***Matricea goală***

Sintaxa:
- `x=[]` – generează o matrice goală (fără nici un element)

Exemplu:
```
x=[]      →        x =
                       []
```

- **`ones`** **-** ***Matricea unitate***

Sintaxe:
- `ones(n)` – returnează o matrice de dimensiune n x n cu toate elementele egale cu 1.
- `ones(m,n)` – returnează o matrice de dimensiune m x n cu toate elementele egale cu 1.
- `ones(size(M))` – returnează o matrice de dimensiunea matricei M cu toate elementele egale cu 1.

Verificați următoarele *exemple*:
```
ones(3)
ones(1,5)
ones(5,1)
ones(3,2)
ones(size(D))          // unde D este matricea definită anterior (vezi pag. 6)
```

- **`zeros`** **-** ***Matricea zero***

Sintaxe:
- `zeros(n)` – returnează o matrice de dimensiune n x n cu toate elementele egale cu 0.
- `zeros(m,n)` – returnează o matrice de dimensiune m x n cu toate elementele egale cu 0.
- `zeros(size(M))` – returnează o matrice de dimensiunea matricei M cu toate elementele egale cu 0.

Verificați următoarele *exemple*:

```
zeros(3)
zeros(1,5)
zeros(5,1)
zeros(3,2)
zeros(size(D))          // unde D este matricea definită anterior (vezi pag. 6)
```

- **eye** - ***Matricea identitate***

Sintaxe:

- `eye(n)` – returnează o matrice identitate de dimensiune n x n.
- `eye(m,n)` – returnează o matrice de dimensiune m x n având elementele primei diagonale egale cu 1 iar restul elementelor egale cu 0.
- `eye(size(M))` – returnează o matrice de dimensiune egală cu dimensiunea matricei M, având elementele primei diagonale egale cu 1 iar restul elementelor egale cu 0.

Verificați următoarele *exemple*:

```
eye(3)
eye(2,3)
eye(3,2)
eye(size(D))          // unde D este matricea definită anterior (vezi pag. 6)
```

- **rand** - ***Matricea cu numere aleatoare cu distribuție uniformă***

Sintaxe:

- `rand(n)` – returnează o matrice de dimensiune n x n având drept elemente numere aleatoare cu distribuție uniformă între 0 şi 1.
- `rand(m,n)` – returnează o matrice de dimensiune m x n având drept elemente numere aleatoare cu distribuție uniformă între 0 şi 1.
- `rand(size(M))` – returnează o matrice de dimensiunea matricei M având drept elemente numere aleatoare cu distribuție uniformă între 0 şi 1.

Verificați următoarele *exemple*:

```
rand(3)
rand(1,5)
rand(5,1)
rand(3,2)
rand(size(D))          // unde D este matricea definită anterior (vezi pag. 6)
```

- **randn** - ***Matricea cu numere aleatoare cu distribuție normală (gaussiană)***

Sintaxe:

- `randn(n)` – returnează o matrice de dimensiune n x n având drept elemente numere aleatoare cu distribuție normală (gaussiană) de medie nulă şi varianța unitară.

- `randn(m,n)` – returnează o matrice de dimensiune m x n având drept elemente numere aleatoare cu distribuție normală de medie nulă şi varianța unitară.
- `randn(size(M))` – returnează o matrice de dimensiunea matricei M având drept elemente numere aleatoare cu distribuție normală (gaussiană) de medie nulă şi varianța unitară.

Verificați următoarele *exemple*:
```
randn(3)
randn(1,5)
randn(5,1)
randn(3,2)
randn(size(D))        // unde D este matricea definită anterior (vezi pag. 6)
```

- **`diag`** - ***Matricea diagonală***

Sintaxe:
Dacă v este un *vector* (linie sau coloană) atunci
- `diag(v)` – returnează o matrice pătrată diagonală, cu elementele vectorului v pe diagonala principală.
- `diag(v,k)` – returnează o matrice pătrată cu elementele vectorului v pe diagonala k deasupra celei principale, dacă k > 0, sau sub cea principală dacă k < 0; restul elementelor sunt 0.

Dacă M este o *matrice* atunci
- `diag(M)` – returnează un vector coloană ce conține elementele de pe diagonala principală a matricei M.
- `diag(M,k)` – returnează un vector coloană ce conține elementele din matricea M de pe diagonala k deasupra celei principale, dacă k > 0, sau sub cea principală, dacă k < 0.

Se va defini un vector linie `a` şi o matrice `A`:
```
a=randn(1,5)
A=randn(5)
```
Verificați următoarele *exemple*:
```
diag(a)
diag(a,1)
diag(a,-1)
diag(a,2)
diag(a,-2)
diag(A)
diag(A,2)
diag(A,-2)
diag(diag(A))
```

- **`tril`** - ***Matricea inferior triunghiulară***

Sintaxe:

- `tril(M)` – extrage matricea inferior triunghiulară din matricea M (anulează toate elementele matricei M de deasupra diagonalei principale).
- `tril(M,k)` – înlocuieşte cu 0 toate elementele de deasupra diagonalei k din matricea M (raportarea se face la diagonala principală – vezi sintaxa de la `diag`).

Pentru matricea `A` definită anterior verificați următoarele *exemple*:

```
tril(A)
tril(A,1)
tril(A,-1)
```

- **`triu`** - ***Matricea superior triunghiulară***

Sintaxe:

- `triu(M)` – extrage matricea superior triunghiulară din matricea M (anulează toate elementele matricei M de sub diagonala principală)
- `triu(M,k)` – înlocuieşte cu 0 toate elementele de sub diagonala k din matricea M (raportarea se face la diagonala principală – vezi sintaxa de la `diag`).

Pentru matricea `A` definită anterior verificați următoarele *exemple*:

```
triu(A)
triu(A,1)
triu(A,-1)
```

1.2.5. Dimensiunea unei matrice. Determinantul şi inversa unei matrice

- **`length,size`** – ***Determinarea dimensiunii variabilelor***

Sintaxe:

Dacă v este un vector şi M este o matrice m x n atunci

- `length(v)` – returnează numărul de elemente (lungimea) vectorului v.
- `length(M)` – returnează maximul dintre numărul de linii şi numărul de coloane al matricei M (maximul dintre m şi n).
- `[l,c]=size(M)` – returnează numărul de linii (`l`) şi numărul de coloane (`c`) pentru matricea M.
- `[l,c]=size(v)` – în acest caz una dintre dimensiuni va fi egală cu 1; dacă v este un vector linie atunci `l` = 1, iar dacă este coloană atunci `c` = 1.

Se va defini un vector linie `a`, un vector coloană `b` şi o matrice `C`:

```
a=randn(1,5)
b=randn(5,1)
C=randn(3,4)
```

Verificaţi următoarele *exemple*:

```
length(a)
length(b)
length(C)
size(a)
size(b)
size(C)
```

- **`det`** – ***Determinantul unei matrice***

Sintaxă:

Dacă M este o matrice pătratică (numărul de linii = numărul de coloane) atunci

- `det(M)` – calculează determinantul matricei M.

Se vor defini două matrice:

```
M=randn(4)
N=randn(4,3)
```

Verificaţi următoarele *exemple*:

```
det(M)
det(N)
```

- **`inv`** – ***Inversa unei matrice***

Sintaxă:

Dacă M este o matrice pătratică cu determinantul diferit de zero atunci

- `inv(M)` – calculează inversa matricei M.

Verificaţi folosind matricele M şi N definite la punctul precedent.

E1. ***Exerciţiu.***

Fie vectorii linie $\mathbf{a} = [0,\ 0.1,\ 0.2, \cdots, 2]$ şi coloană $\mathbf{b} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$.

a) Ce lungime trebuie să aibă **b** astfel ca să aibă sens înmulţirea (în sens matricial) **a*b**? Iniţializaţi în MATLAB cei doi vectori şi realizaţi înmulţirea. Ce operaţie se efectuează?
b) Realizaţi înmulţirea **b*a**.
c) Realizaţi înmulţirea element cu element a celor doi vectori. Ce rezultat se obţine?

1.2.6. Instrucțiuni de control logic. Operatori relaționali şi operatori logici

- **`if,else,elseif`** – ***Execuția condiționată***

Sintaxe:

- `if expresie`
  `instrucțiuni`
  `end`
  // dacă `expresie` este adevărată se execută `instrucțiuni`;
  dacă `expresie` este falsă se trece după `end`.
- `if expresie`
  `instrucțiuni_1`
  `else`
  `instrucțiuni_2`
  `end`
  // dacă `expresie` este adevărată se execută `instrucțiuni_1`;
  dacă `expresie` este falsă se execută `instrucțiuni_2`.

- `if expresie_1`
  `instrucțiuni_1`
  `elseif expresie_2`
  `instrucțiuni_2`
  `end`
  // dacă `expresie_1` este adevărată se execută `instrucțiuni_1`;
  dacă `expresie_1` este falsă şi `expresie_2` este adevărată se execută `instrucțiuni_2`.

- **`for`** – ***Repetarea unui număr de instrucțiuni de un anumit număr de ori***

Sintaxă:

- `for index=inițial:pas:final`
  `instrucțiuni`
  `end`
  // pentru `index` parcurgând intervalul de la `inițial` la `final` cu pasul `pas` se execută `instrucțiuni`.

- **`while`** – ***Repetarea unui număr de instrucțiuni atâta timp cât o condiție specificată este adevărată***

Sintaxă:

- `while expresie`
  `instrucțiuni`
  `end`

// cât timp `expresie` este adevărată se execută `instrucţiuni`.

- ***Operatori relaţionali***

- < - mai mic
- <= - mai mic sau egal
- > - mai mare
- >= - mai mare sau egal
- == - identic
- ~= - diferit

- ***Operatori logici***

- & - operatorul ŞI logic
- | - operatorul SAU logic
- ~ - operatorul NU logic

1.2.7. Crearea programelor MATLAB (crearea fişierelor de comenzi *.m)

Pentru secvenţe lungi de comenzi se recomandă crearea şi lansarea în execuţie a unui program MATLAB. Acesta este un fişier "text" având extensia .m şi conţinând succesiunea dorită de comenzi MATLAB. După creare, fişierul devine o nouă comandă externă MATLAB, care poate fi lansată în execuţie prin simpla introducere de la tastatură a numelui fişierului (fără extensia .m).
Pentru crearea unui astfel de fişier se parcurg următorii *paşi*:

- în bara de meniuri a ferestrei de comenzi se selectează File, iar în interiorul acesteia New, urmată de M-file. Se va deschide astfel o nouă fereastră (sesiune de editare cu editorul NOTEPAD) cu bară de meniuri proprie şi cu numele Untitled. În acest fişier se scrie programul MATLAB dorit.
- pentru a salva fişierul astfel creat sub un alt nume se selectează din bara de meniuri a noii ferestre comanda File, urmată de Save As...; va apărea o nouă fereastră de dialog în care vom preciza numele fişierului (File name:) însoţit de extensia .m şi locul unde dorim să îl salvăm (Save in:).

Atenţie:

- *pentru a rula un program MATLAB trebuie ca directorul în care a fost salvat să fie directorul de lucru* (*vezi comanda* `cd`*, pagina 3*).
- *pentru a fi luate în considerare eventualele modificări făcută într-un program MATLAB, înainte de o nouă rulare, fişierul trebuie salvat* (File*, urmat de* Save).
- *denumirea fişierului nu trebuie să înceapă cu cifre sau sa fie un cuvânt cheie (instrucţiune) Matlab. Denumiri de genul „ex_1.m" sau „progr_2.m" sunt suficiente.*

E2. ***Exerciţiu***:
Se creează un fişier nou care trebuie salvat în directorul **d:/student/pns/***nrgrupa*
Folosind sintaxele şi indicaţiile din secţiunile 1.2.6. şi 1.2.7. realizaţi un program MATLAB care să genereze un vector cu elemente aleatoare cu distribuţie normală (gaussiană) şi să afişeze elementele negative ale acestui vector.

1.2.8. Crearea funcţiilor MATLAB (crearea fişierelor funcţie)

Dacă prima linie a unui fişier MATLAB (*.m) conţine la început cuvântul `function` atunci fişierul respectiv e declarat ca fişier funcţie. Aceste fişiere pot fi adăugate ca funcţii noi în structura MATLAB. Forma generală a primei linii a unui fişier funcţie este:

```
function [parametrii_ieşire]=nume(parametrii_intrare)
```

cu următoarele semnificaţii:
- `function` – cuvânt cheie care declară fişierul ca fişier funcţie.
- `nume` – numele funcţiei; reprezintă numele sub care se salvează fişierul .m (extensia .m nu face parte din `nume`); acest `nume` nu poate fi identic cu cel al unui fişier .m deja existent.
- `parametrii_ieşire` – reprezintă parametrii de ieşire ai funcţiei; trebuie separaţi prin virgulă şi cuprinşi între paranteze drepte.
- `parametrii_intrare` – reprezintă parametrii de intrare ai funcţiei; trebuie separaţi prin virgulă şi cuprinşi între paranteze rotunde.

1.2.9. Operaţii aritmetice. Operaţii asupra numerelor complexe

- ***Operatori aritmetici***

- `+` - adunare
- `-` - scădere
- `*` - înmulţire
- `.*` - înmulţire între două matrice (sau vectori) element cu element
- `/` - împărţire
- `./` - împărţire între două matrice (sau vectori) element cu element
- `^` - ridicare la putere
- `.^` - ridicare la putere a unei matrice (sau vector) element cu element
- `'` - transpunere şi conjugare
- `.'` - transpunere

Să se definească următoarele elemente:

$$A=\begin{pmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{pmatrix},\ B=\begin{pmatrix}1-i & 2+i & 0\\ -i & 3 & i\end{pmatrix},\ a=\begin{pmatrix}1+i & 3 & 0\end{pmatrix},\ b=\begin{pmatrix}1/3\\ 2^2\end{pmatrix}$$

Verificați şi *explicați* următoarele *exemple*:

```
A+B , A+3 , B-2 , B-i , a+a , A*B , A.*B , A*B.' , a.*a ,
a*a , A*a , A*a', A*a.', A'*b , B' , B.' , A' , A.' , A^2 ,
A.^2 , 1-b , b'*b , b*b , b.*b , b^3 , b.^3 , 2/A , 2./A ,
2/b , 2./b .
```

- ***Operații asupra numerelor complexe***

- `abs` – calculează valoarea absolută (modulul).
- `angle` – calculează faza.
- `conj` – calculează complex conjugatul.
- `real` – extrage partea reală.
- `imag` – extrage partea imaginară.

Verificați comenzile utilizând drept argumente elementele A, B, a şi b definite anterior.

1.2.10. Funții matematice uzuale

- ***Funcțiile radical, exponențială şi logaritm***

- `sqrt` – extragere radical de ordinul 2 (rădăcina pătrată).
- `exp` – calculează exponețiala (puteri ale numărului *e*).
- `log` – calculează logaritmul natural (logaritm în baza *e*).
- `log2` – calculează logaritmul în bază 2.
- `log10` – calculează logaritmul zecimal (logaritm în baza 10).
- `pow2` – calculează puteri ale lui 2.

- ***Funcțiile trigonometrice directe***

- `sin` – calculează sinusul
- `cos` – calculează cosinusul
- `tan` – calculează tangeta
- `cot` – calculează cotangenta
- `sec` – calculează secanta
- `csc` – calculează cosecanta

- ***Funcțiile trigonometrice inverse***

- `asin` – calculează arcsinus

- `acos` – calculează arccosinus
- `atan` – calculează arctangenta
- `atan2` – calculează arctangenta dacă argumentul este complex
- `acot` – calculează arccotangenta
- `asec` – calculează arcsecanta
- `acsc` – calculează arccosecanta

- ***Funcțiile hiperbolice directe***

- `sinh` – calculează sinusul hiperbolic
- `cosh` – calculează cosinusul hiperbolic
- `tanh` – calculează tangeta hiperbolică
- `coth` – calculează cotangenta hiperbolică
- `sech` – calculează secanta hiperbolică
- `csch` – calculează cosecanta hiperbolică

- ***Funcțiile hiperbolice inverse***

- `asinh` – calculează arcsinus hiperbolic
- `acosh` – calculează arccosinus hiperbolic
- `atanh` – calculează arctangenta hiperbolică
- `acoth` – calculează arcotangenta hiperbolică
- `asech` – calculează arcsecanta hiperbolică
- `acsch` – calculează arccosecanta hiperbolică

Pentru informații despre modul de utilizare al acestor funcții folosiți comanda `help` însoțită de numele funcției dorite (vezi sintaxa de la pagina 2).

1.2.11. Funții destinate analizei de date

- **`sum,prod`** – ***Suma şi produsul***

Sintaxe:
Dacă v este un vector şi M este o matrice atunci
- `sum(v)` – calculează suma elementelor vectorului v.
- `prod(v)` – calculează produsul elementelor vectorului v.
- `sum(M)` – returnează un vector linie având ca elemente suma elementelor fiecărei coloane din matricei M.
- `prod(M)` – returnează un vector linie având ca elemente produsul elementelor fiecărei coloane din matricei M.

Se vor defini următoarele elemente: $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$, $a = \begin{pmatrix} 1 & 3 & 8 \end{pmatrix}$

Să se verifice următoarele *exemple*:

```
sum(a)
sum(A)
sum(sum(A))
prod(a)
prod(A)
prod(prod(A))
```

- **`max,min`** – ***Maximul şi minimul***

Sintaxe:

Dacă v este un vector şi M este o matrice atunci

- `max(v)` – returnează elementul maxim al vectorului v.
- `min(v)` – returnează elementul minim al vectorului v.
- `[m,p]=max(v)` – returnează elementul maxim al vectorului (`m`) precum şi indicele elementului maxim (`p`); dacă există maxime multiple se returnează indicele primului dintre ele.
- `[m,p]=min(v)` – returnează elementul minim al vectorului (`m`) precum şi indicele elementului minim (`p`); dacă există minime multiple se returnează indicele primului dintre ele.
- `max(M)` – returnează un vector linie având ca elemente maximul elementelor din fiecare coloană a matricei M.
- `min(M)` – returnează un vector linie având ca elemente minimul elementelor din fiecare coloană a matricei M.
- `[m,p]=max(M)` – returnează un vector linie `m` având ca elemente maximul elementelor din fiecare coloană a matricei M precum şi un vector linie `p` ce conține poziția maximului respectiv în cadrul fiecărei coloane.
- `[m,p]=min(M)` – returnează un vector linie `m` având ca elemente minimul elementelor din fiecare coloană a matricei M precum şi un vector linie `p` ce conține poziția minimului respectiv în cadrul fiecărei coloane.

Verificați sintaxele anterioare folosind matricea A şi vectorul a definite mai sus.

- **`mean`** – ***Media aritmetică***

Sintaxe:

Dacă v este un vector şi M este o matrice atunci

- `mean(v)` – calculează media aritmetică a elementelor vectorului v.
- `mean(M)` – returnează un vector linie având ca elemente media aritmetică a elementelor fiecărei coloane din matricei M.

Verificați sintaxele anterioare folosind matricea A şi vectorul a definite mai sus.

E3. ***Exerciţiu***:
Realizaţi un program MATLAB care generează un vector cu elemente complexe. Realizaţi (un alt fişier) o funcţie MATLAB care având drept parametru de intrare vectorul cu valori complexe returnează ca parametri de ieşire:
- media aritmetică a părţilor reale ale elementelor vectorului;
- un vector ce conţine elementele vectorului iniţial ridicate la pătrat;
- o matrice obţinută din înmulţirea vectorului iniţial cu transpusul său.

Atenţie:
Pentru a nu se afişa rezultate intermediare din funcţie sau elementele unor variabile se va folosi **;** la sfârşitul liniei respective de program.

1.2.12. Reprezentări grafice

- **`plot,stem`** – ***Reprezentări grafice în coordonate liniare***

Sintaxe:
Dacă v este un vector şi M este o matrice atunci
- `plot(v)` – dacă v este un vector cu *elemente reale* se vor reprezenta grafic elementele sale în funcţie de indici (primul indice este 1, ultimul indice este egal cu lungimea vectorului); dacă v este un vector cu *elemente complexe* atunci reprezentarea sa se va face în funcţie de partea reală (pe abscisă) şi de partea imaginară (pe ordonată) a elementelor sale.
- `plot(M)` – se vor reprezenta pe acelaşi grafic coloanele matricei M (fiecare din coloanele matricei este privită ca un vector şi reprezentat ca în sintaxa precedentă);

Dacă x şi y sunt doi vectori de *aceeaşi lungime* şi N este o matrice de *aceeaşi dimensiune* cu matricea M atunci
- `plot(x,y)` – se vor reprezenta grafic elementele vectorului y în funcţie de elementele vectorului x; *dacă lungimea vectorului* x *nu este egală cu lungimea vectorului* y *reprezentarea nu este posibilă*.
- `plot(x,M)` – dacă lungimea vectorului x este egal cu numărul de linii al matricei M atunci se vor reprezenta pe acelaşi grafic coloanele matricei M în funcţie de elementele vectorului x; dacă lungimea vectorului x este egal cu numărul de coloane al matricei M atunci se vor reprezenta pe acelaşi grafic liniile matricei M în funcţie de elementele vectorului x; *dacă lungimea vectorului* x *nu este egală cu una din dimensiunile matricei* M *atunci reprezentarea nu este posibilă*.
- `plot(M,N)` – se vor reprezenta pe acelaşi grafic coloanele matricei N în funcţie de coloanele matricei M (coloana k din matricea N va fi reprezentată funcţie de coloana k din matricea M, unde k = 1, 2, … , numărul de coloane); *dacă cele două matrice nu au aceeaşi dimensiune atunci reprezentarea nu este posibilă.*

Pentru reprezentarea mai multor grafice în aceeaşi fereastră grafică, utilizând o singură comandă, folosim

- `plot(x1,y1,x2,y2,...,xn,yn)` - se vor reprezenta pe acelaşi grafic y1 în funcție de x1, y2 în funcție de x2, … ,yn în funcție de xn (pot fi vectori sau matrice); rămân valabile considerentele făcute în sintaxele anterioare, referitor la cazurile când avem vectori sau matrice; *dacă* xi *şi* yi (i =1, 2,…,n) *nu au aceeaşi dimensiune atunci reprezentarea nu este posibilă*.

***Atenție*:**
Se pot folosi diverse linii, markere şi culori de reprezentare a graficelor. Vezi `help plot`.

Funcția `stem` realizează o reprezentare în formă "discretă" a datelor. Pentru variantele de MATLAB 5 sau MATLAB 6 se pot folosi oricare din sintaxele prezentate la `plot`, exceptând ultima sintaxă. Pentru variantele de MATLAB 4 argumentele de intrare ale funcției `stem` nu pot fi decât vectori, iar reprezentarea grafică nu se poate face decât cu o singură culoare (se pot folosi însă mai multe tipuri de linii şi markere).
Vezi `help stem`.

- **`loglog,semilogx,semilogy`** – ***Reprezentări grafice în coordonate logaritmice***

Pentru acest tip de reprezentări se folosesc funcțiile `loglog`, `semilogx` şi `semilogy`. Sintaxele rămân aceleaşi ca la funcția `plot`, singura deosebire fiind modul de scalare al axelor. Astfel funcția `loglog` scalează ambele axe (abscisa şi ordonata) folosind logaritmul în bază 10, deci pe axe vom avea puteri ale lui 10. Funcția `semilogx` realizează acelaşi tip de scalare însă numai pe abscisă iar funcția `semilogy` procedează în acelaşi mod însă numai pe ordonată.

- **`subplot`** – ***Divizarea ferestrei grafice***

Dacă dorim ca fereastra grafică să conțină mai multe reprezentări grafice se poate folosi funcția `subplot` care împarte fereastra grafică în mai multe "miniferestre", în fiecare dintre acestea putând fi plasat câte un grafic. Fereastra grafică este astfel privită sub forma unei matrice cu m linii şi n coloane, deci în total m·n "miniferestre". Numărarea acestor "miniferestre" se face pe linii. De exemplu, dacă vrem să împărțim fereastra grafică în 3·3 = 9 "miniferestre" vom avea următoarea ordine:

1	2	3
4	5	6
7	8	9

Sintaxa:

- `subplot(m,n,p)` – împarte fereastra grafică într-o matrice m x n (m·n "miniferestre"), iar p reprezintă numărul fiecărei "miniferestre" în matricea grafică respectivă (numărarea se face pe linii); sintaxa respectivă este urmată de comanda propriu-zisă de afişare a graficului, care poate fi oricare din cele prezentate până acum (`stem`, `plot`, `loglog`, `semilogx`, `semilogy`).

- **`axis`** – ***Schimbarea limitelor axelor***

Dacă se doreşte vizualizarea numai a unei anumite porțiuni dintr-un grafic, corespunzătoare unor anumite intervale pe abscisă şi ordonată, se va folosi comanda `axis`.

Sintaxa:

- `axis([x0 x1 y0 y1])` – pe abscisa se va vizualiza între valorile x0 şi x1 iar pe ordonată între y0 şi y1; această sintaxă se plasează după comanda de reprezentare grafică.

- **`title,xlabel,ylabel,gtext,grid`** – ***Precizarea titlului graficului şi a etichetelor axelor. Trasarea unei rețele pe grafic. Plasarea unui text pe grafic.***

Sintaxe:

- `title('text')` – plasează deasupra graficului, ca titlu, textul `text`.
- `xlabel('text')` – textul `text` devine eticheta de pe abscisă.
- `ylabel('text')` – textul `text` devine eticheta de pe ordonată.
- `grid` – trasează pe grafic o rețea de linii, înlesnind astfel citirea graficului.
- `gtext('text')` – plasează pe grafic textul `text` (folosind mouse-ul).

Toate aceste sintaxe urmează după comanda de reprezentare grafică.

- **`hold`** – ***Suprapunerea succesivă a graficelor***

Dacă dorim să reținem graficul curent şi să adăugăm în aceeaşi fereastră grafică următoarele reprezentări grafice se poate folosi funcția `hold`.

Sintaxa:

- `hold on` – reține graficul curent şi adaugă în aceeaşi fereastră grafică următoarele reprezentări grafice.
- `hold off` – dacă se doreşte în continuare reprezentarea în ferestre grafice separate (dezactivează comanda `hold on`).

***Atenție*:**

Dacă se doreşte în cadrul unui program reprezentarea mai multor grafice în ferestre separate, fiecare comandă grafică va trebui să fie precedată de un nume de forma `figure(n)`, *unde* `n` *este numărul figurii respective. În caz*

contrar, la sfârşitul execuţiei programului va apărea numai ultima reprezentare grafică (se va folosi o singură fereastră grafică ce va fi "ştearsă" de fiecare dată la întâlnirea unei noi comenzi grafice).

***Exemple*:** Se vor tasta direct în fereastra de comenzi, *explicând* rezultatele:

```
x=0:0.2:2*pi;
size(x)
s=sin(x);
size(s)
plot(s),grid,title('sinus'),xlabel('n')
stem(s),grid,title('sinus'),xlabel('n')
n=linspace(min(x),max(x),length(x));
size(n)
plot(n,s,'r*'),grid,title('sinus'),xlabel('n')
hold on
stem(n,s),grid,xlabel('n'),ylabel('amplitudine')
hold off
figure(1)
plot(n,s),grid,axis([0 pi min(s) max(s)])
figure(2)
plot(n,s,n,s-pi/2),grid
```

// se închid ambele ferestre grafice şi se continuă tastarea în fereastra de comenzi

```
c=cos(x);
subplot(2,1,1),stem(n,s),title('sinus'),grid
subplot(2,1,2),stem(n,c),title('cosinus'),grid
M=[c;s];
plot(n,M),grid
N=[n;n];
plot(N,M),grid
plot(N',M'),grid
z=1:1000;
p=z.^2;
plot(z,p),grid
loglog(z,p),grid
semilogx(z,p),grid
semilogy(z,p),grid
gtext('Ultimul grafic!!!')
```

// cu ajutorul mouse-ului plasați pe grafic textul respectiv, în poziția dorită.

E4. ***Exerciţiu*:**
Realizaţi un program MATLAB în care să generaţi şi să reprezentaţi grafic folosind funcţia **stem** următorii vectori:

a) $z = [0,0,0,0,0,1,0,0,...,0]$, vectorul **z** având lungimea 21. Reprezentarea grafică se va face în două "miniferestre" (funcţia subplot) vectorul **z** în funcţie de **n = 0:20** respectiv de **m= –5:15**.
b) $\mathbf{t} = |10 - \mathbf{n}|$, reprezentat grafic în funcţie de **n = 0:20**.
c) $x_1 = \sin\left(\frac{\pi}{17}n\right)$, $-15 \le n \le 25$ şi $x_2 = \cos\left(\frac{\pi}{\sqrt{23}}n\right)$, $0 \le n \le 50$

cele două secvenţe vor fi reprezentate în:
- figura 1 – în acelaşi sistem de coordonate (pe acelaşi grafic);
- figura 2 – folosind două "miniferestre" grafice plasate una sub alta.

Reprezentaţi cele două figuri folosind comanda `plot` iar apoi încercaţi reprezentarea lor folosind comanda `stem`.

Cu funcţia **plot** se pot reprezenta grafic semnale sau funcţii "continue" deoarece se unesc cu linie continuă valorile care se reprezintă. Astfel se pot reprezenta semnale continue alegând variabila timp cu pasul mai mic decât variaţia semnalului reprezentat. De exemplu dacă perioada semnalului e 0.01 secunde se poate alege variabila temporală cu pasul de 0.001s: t = 0:0.001:5 (secunde).

Exemplu:
Să se reprezinte grafic cu funcţia **plot** un semnal sinusoidal de frecvenţă 50 Hz, de durată 0.2 secunde şi amplitudine 2. Se va alege rezoluţia temporală 1ms.

```
F = 50;
t = 0:0.001:0.2;
s = 2*sin(2*pi*F*t);
plot(t,s,'.-'),xlabel('Timp [s]'),grid
```

E5. ***Exerciţiu*:**
a) Modificaţi pasul de variaţie a variabilei **t** la 0.01 şi apoi la 0.0002. Comentaţi diferenţele.
b) Măsuraţi pe grafic perioada semnalului sinusoidal în cele 3 situaţii.
c) Generaţi un semnal **cosinusoidal** de frecvenţă 20 Hz pe care să-l reprezentaţi cu culoare roşie pe acelaşi grafic peste semnalul sinusoidal.

Un semnal discret provine din eşantionarea unui semnal continuu cu o perioadă de eşantionare T_s. Valorile stocate în vectorul care reprezintă semnalul discret corespund valorilor amplitudinii semnalului continuu la momente întregi de perioada de eşantionare $n \cdot T_s,\ n \in \mathbb{Z}$.

$$s(n) = s_c(nT_s)$$

De exemplu pentru un semnal sinusoidal de frecvență F_0

$$s_c(t) = A_0 \sin(2\pi \cdot F_0 \cdot t)$$

Semnalul eşantionat va fi:

$$s(n) = A_0 \sin(2\pi \cdot F_0 \cdot n \cdot T_s) = A_0 \sin\left(2\pi \frac{F_0}{F_s} n\right)$$

Notând frecvența normată $f_0 = \frac{F_0}{Fs}$, respectiv pulsația normată $\omega_0 = 2\pi f_0 = 2\pi \frac{F_0}{F_s}$ rezultă:

$$s(n) = A_0 \sin(2\pi \cdot f_0 \cdot n) = A_0 \sin(\omega_0 n)$$

Exemplu:

Să se reprezinte grafic cu funcția **stem** un semnal discret obținut prin eşantionarea unui semnal sinusoidal de frecvență 300 Hz, de durată 10 milisecunde şi amplitudine 1. Frecvența de eşantionare F_s = 4kHz. Câte eşantioane are semnalul discret?

```
F0 = 300; Fs = 4000;
W0 = 2*pi*F0/Fs;
N = 10*4;       // Numar de esantioane N=10ms*4kHz
n = 0:N-1;
s = 2*sin(w0*n);
stem(n,s),grid
```

E6. ***Exercițiu*:**

Realizați un program care să genereze un semnal binar aleator (valori de 0 şi 1), perioada de bit: 0.5 ms eşantionat cu frecvența de eşantionare F_s = 12 kHz.

a) Reprezentați cu funcția plot semnalul continuu în timp absolut (în milisecunde).
b) Reprezentați cu funcția stem semnalul discret în funcție de n.

Tema de casă.

Se va genera un semnal cu rezolutie temporara de 2ms, 20ms 200ms pentru un semnal continuu de tipul:

1. Semnal dreptunghiular periodic cu
   - Perioadă: 2 s.
   - Factor de umplere: 25%.
   - Nivel maxim: +0.5.
   - Nivel minim: -1.
2. Semnal triunghiular periodic
   - Perioadă: 5 s.
   - Nivel maxim: +1.
   - Nivel minim: -2.
   - Panta+ 1 V/s *Observație*: Panta– [V/s] rezultă din calcule
3. Semnal dreptunghiular multinivel, aleator
   - Durata fiecărui nivel: 0,25 s.
   - Nivelurile
     a) {-1, 1}
     b) {-3,-1,1,3}
     c) {-5,-3,-1,1,3,5}
     d) {-7,-5,-3,-1,1,3,5,7}
4. Semnal sinusoidal redresat mono alternanță
   - Perioada semnalului sinusoidal inițial (neredresat) 3 s.
   - Amplitudine: 0.8.
5. Semnal sinusoidal redresat dublă alternanță
   - Perioada semnalului sinusoidal neredresat: 4 s.
   - Amplitudine: 1.5.