1 Introducere în MATLAB

1.1 Consideraţii generale

În cadrul acestei lucrări de laborator ne propunem să revedem câteva dintre comenzile MATLAB utile în prelucrarea numerică a semnalelor. Programul MATLAB este unul dintre cele mai rapide şi mai facile pentru rezolvarea numerică a problemelor. Există numeroase funcţii atât pentru calcul, cât şi pentru reprezentări grafice [1].

După lansarea în execuţie, programul MATLAB intră în modul de comandă, afişând prompterul `>>`, şi aşteaptă introducerea unei comenzi de la utilizator. De exemplu, comanda: `>> var = 0 : 6` generează variabila `var`, afişând în fereastra de comandă cele şapte elemente, de la `var(1) = 0` la `var(7) = 6`:

```
    var =
        0    1    2    3    4    5    6
```

Comenzile introduse anterior pot fi corectate sau reutilizate; prin apăsarea săgeţilor de pe tastatură apar în linia de comandă comenzile introduse anterior; se pot realiza corecţiile necesare iar, apoi, prin apăsarea tastei `[Enter]` se rulează comanda corectată.

În afara modului de lucru în linia de comandă, utilitarul MATLAB permite, de asemenea, lucrul cu programe incluse în fişiere, numite fişiere-M (`*.m`). Un program poate fi scris ca un fişier script sau ca un fişier funcţie. Un script este un fişier extern care conţine o secvenţă de comenzi MATLAB. După execuţia completă a unui fişier script, variabilele create rămân în zona de memorie a aplicaţiei. Dacă prima linie a fişierului conţine cuvântul `function`, fişierul respectiv este un fişier funcţie, caracterizându-se prin faptul că poate lucra cu argumente. La terminarea execuţiei unei funcţii, în memoria calculatorului rămân doar variabilele de ieşire ale acesteia.

MATLAB-ul lucrează cu două tipuri de ferestre: o fereastră de comenzi şi una de reprezentări grafice. La un moment dat, doar o fereastră de comenzi poate fi deschisă, în schimb, pot fi deschise mai multe ferestre de reprezentări grafice. Mediul de programare MATLAB este sensibil la tipul de litere (mici sau mari - "case sensitive"). Numele unei funcţii trebuie scris întotdeauna cu litere mici. Liniile de comentariu dintr-un script sau dintr-o funcţie sunt precedate de caracterul `%`.

1.1.1 Definirea variabilelor

Variabilelor le sunt atribuite valori numerice, scriindu-se direct expresia numerică. Dacă se tastează în linia de comandă: `>> var1 = 1+8`, rezultatul obţinut este:

```
var1 =
    9
```

Dacă la sfârşitul expresiei se pune caracterul punct-virgulă `;`, rezultatul nu mai este afişat în linia de comandă; de exemplu, `>> var1 = 1+8;`.

Unei variabile i se pot atribui formule ce utilizează operatori aritmetici definiţi în MATLAB (vezi tabelul 1.1 [2]) sau una sau mai multe mărimi definite anterior (chiar şi în comanda curentă). Presupunând că variabila `var1` a fost definită anterior, `>> var2 = var1^2` va returna valoarea:

```
var2 =
        81
```

Simbol	Descriere
+	Adunare
-	Scădere
*	Înmulţire
.*	Înmulţire de matrice
/	Împărţire
./	Împărţire de matrice
^	Ridicare la putere
.^	Ridicare la putere a unei matrice
'	Transpunere sau transpunere complex conjugată
.'	Transpunere a unei matrice
( )	Specificarea ordinii de evaluare a operaţiilor

Tabela 1.1: Operatori aritmetici

În MATLAB există variabile predefinite – acestea nu pot fi declarate şi sunt accesibile global în orice fişier-M. În general, aceste variabile speciale se introduc în codul funcţiilor, ele returnând valori scalare utile [3]. Câteva dintre variabilele şi constantele speciale sunt ilustrate în tabelul 1.2.

În MATLAB toate calculele se realizează cu precizie dublă. Formatul – modul în care MATLAB-ul afişează numerele – este stabilit prin comanda `format`. Pentru a vedea toate opţiunile tastaţi în linia de comandă `help format`. Sintaxa de apelare a acestei funcţii este `format opţiune`; câteva exemple sunt prezentate în tabelul 1.3 [4].

Funcţie	Valoare returnată
ans	Cel mai recent rezultat (variabilă); dacă unei expresii nu îi este atribuită o variabilă de ieşire, MATLAB-ul stochează automat rezultatul în ans
eps	Acurateţea relativă pentru aritmetica în virgulă mobilă; aceasta este toleranţa utilizată la calcule (valoarea implictă este $2.2204e-016$)
pi	Variabilă permanentă căreia îi este asignată valoarea $\pi = 3.141592653589\ldots$
i, j	Variabilă folosită pentru introducerea numerelor compleexe ($\sqrt{-1}$)
inf	Infinit; calculele de genul $n/0$, unde $n \in \mathbb{R}^*$, dau rezultatul inf
NaN	*Not a Number*, o valoare numerică invalidă; expresiile de genul $0/0$ şi inf/inf dau rezultatul NaN; de asemenea, dacă $n \in \mathbb{C}$ cu partea reală zero, atunci $n/0$ returnează o valoare cu partea reală NaN

Tabela 1.2: Variabile/constante speciale

Opţiune	Rezultat	Exemplu
+	+, −, blanc	+
bank	Reprezentare cu 2 zecimale	3.14
hex	Reprezentarea hexazecimală a unui număr binar cu precizie dublă	$400921fb54442d18$
long	Pentru aritmetica în virgulă fixă: 15 digiţi – precizie dublă; 8 digiţi – precizie simplă	3.14159265358979
long e	Pentru aritmetica în virgulă mobilă: 15 digiţi – precizie dublă; 8 digiţi – precizie simplă	$3.141592653589793e+00$
long g	Reprezentare în virgulă fixă/mobilă: 15 digiţi – precizie dublă; 8 digiţi – precizie simplă (care este mai uşor de citit)	3.14159265358979
rat	Raport de întregi cu valori mici	355/113
short	Virgulă fixă, cu 5 digiţi	3.1416
short e	Virgulă mobilă, cu 5 digiţi	$3.1416e+00$
short g	Reprezentare în virgulă fixă/mobilă: 5 digiţi (care este mai uşor de citit)	3.1416

Tabela 1.3: Câteva valori permise pentru parametrul `opţiune` şi un exemplu pentru π

1.1.2 Funcţii predefinite

Câteva dintre funcţiile predefinite în MATLAB sunt ilustrate în tabelul 1.4. Pentru informaţii despre utilizarea acestor funcţii folosiţi comanda `help` urmată de numele funcţiei dorite.

Dacă `v` este un vector şi `M` o matrice, sintaxele pentru funcţiile destinate analizei de date sunt prezentate în tabelul 1.5.

Funcţie	**Descriere**
Operaţii cu privire la numere complexe	
abs	Valoare absolută (modul)
angle	Fază, în radiani
conj	Valoare complex conjugată
real	Parte reală a unui număr complex
imag	Parte imaginară a unui număr complex
Funcţii trigonometrice	
sin	Sinus a unui argument în radiani
cos	Cosinus a unui argument în radiani
tan	Tangentă a unui argument în radiani
cot	Cotangentă a unui argument în radiani
Radical, exponenţială şi logaritm	
sqrt	Radical de ordin 2
exp	Exponenţială (puteri ale numărului e)
log	Logaritm natural
log2	Logaritm în baza 2
log10	Logaritm în baza 10
pow2	Puteri ale lui 2
Operaţii de bază	
round	Rotunjire la cel mai apropiat întreg
floor	Rotunjire spre $-\infty$
fix	Rotunjire spre zero
ceil	Rotunjire spre $+\infty$

Tabela 1.4: Câteva funcţii elementare predefinite în MATLAB

1.1.3 Scalari, vectori şi matrice

MATLAB este un pachet de programe care lucrează numai cu un singur tip de obiecte, matrice numerice rectangulare, cu elemente reale sau complexe; scalarii sunt asimilaţi matricelor cu o linie şi o coloană iar vectorii sunt asimilaţi matricelor cu o linie sau o coloană. În MATLAB datele numerice şi nenumerice se stochează un pic altfel dar, pentru început, este convenabil să privim totul ca fiind matrice; se poate lucra cu întreaga matrice rapid şi uşor [4].

Matricele pot fi introduse în mai multe moduri: introducerea explicită a listei de elemente, încărcarea din fişiere de date externe, generarea utilizând funcţii predefinite, crearea matricelor cu funcţii proprii. Cea mai simplă metodă constă în utilizarea unei liste explicite, respectând următoarele reguli:

- elementele unei linii trebuie separate prin spaţii libere sau virgulă `,`;
- liniile se separă prin semnul punct-virgulă `;`;
- elementele matricei sunt cuprinse între paranteze drepte `[]`.

Sintaxă	Descriere
sum(v)	Returnează suma elementelor vectorului v
prod(v)	Returnează produsul elementelor vectorului v
sum(M)	Returnează un vector linie având ca elemente suma elementelor fiecărei coloane din matricea M
prod(M)	Returnează un vector linie având ca elemente produsul elementelor fiecărei coloane din matricea M
max(v)/ min(v)	Returnează elementul cel mai mare/mic din vectorul v
[m, p] = max(v)/ [m, p] = min(v)	Returnează elementul cel mai mare/mic din v în variabila m şi indicele corespunzător în variabila de ieşire p; dacă există mai multe valori maxime/minime, se returnează indicele primului element găsit
max(M)/ min(M)	Returnează un vector linie având ca elemente maximul/minimul elementelor din fiecare coloană a matricei M
[m, p] = max(M)/ [m, p] = min(M)	Returnează un vector linie m având ca elemente maximul/minimul elementelor din fiecare coloană a matricei M şi poziţia maximului/ minimului în cadrul fiecărei coloane în p; dacă există mai multe valori, se returnează indicele primului element găsit
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 matricea M

Tabela 1.5: Funcţii MATLAB pentru analiza de date

Elementele matricelor pot fi orice numere reale sau complexe, sau orice alte variabile. Elementele matricei `A` pot fi identificate utilizând comanda `A`(i, j) (elementul de la intersecţia liniei `i` cu coloana `j`) – este nevoie de doi indici; pentru a ne referi la un element dintr-un vector este nevoie doar de un indice. Matricea $\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ şi vectorii $\mathbf{B} = \begin{bmatrix} 7 & 8 & 9 \end{bmatrix}$ şi $\mathbf{C} = \begin{bmatrix} -1 \\ -2 \end{bmatrix}$ pot fi introduşi în MATLAB astfel:

```
>> A=[1,2,3; 4,5,6]
```

```
>> A=[1 2 3; 4 5 6]
```

```
>> A=[1 2 3 Enter
   4 5 6]
```

toate sintaxele returnând:

```
A =
    1    2    3
    4    5    6
```

```
>> B = [7 8 9]
B =
    7    8    9
```

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

Dacă se doreşte schimbarea elementelor unei matrice sau adăugarea altor elemente, fără a rescrie întreaga matrice, se procedează astfel:

```
>> A(1, 3) = 0
A =
    1   2   0
    4   5   6
```

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

```
>> C(3) = 1
C =
   -1
   -2
    1
```

Matricele de dimensiuni mari pot fi obţinute din matrice de dimensiuni mai mici. Vom folosi pentru exemplificare matricea `A` şi vectorii `B` şi `C` definiţi anterior, în ultima lor formă:

```
>> D = [A; B]
D =                 % matricea D (4x3) s-a obţinut prin adăugarea vectorului B la
    1    2    0     % matricea A (ca ultimă linie);
    4    5    6     % A şi B trebuie să aibă acelaşi număr de coloane
    0    0   -2
    7    8    9
>> E = [A, C]
E =                       % matricea E (3x4) s-a obţinut prin adăugarea vectorului C la
    1    2    0   -1      % matricea A (ca ultimă coloană);
    4    5    6   -2      % A şi C trebuie să aibă acelaşi număr de linii
    0    0   -2    1
```

Dacă `v` este un vector şi `M` o matrice avem sintaxele prezentate în tabelul 1.6.

Dimensiunea, determinantul şi inversa unei matrice pot fi determinate utilizând sintaxele din tabelul 1.7.

În tabelul 1.8 sunt ilustrate sintaxele corespunzătoare celor mai utilizaţi vectori şi celor mai folosite matrice.

1.1.4 Construirea unei funcţii

Funcţiile sunt mult mai eficiente decât scripturile, deoarece permit utilizatorului să creeze noi comenzi MATLAB [5]. Forma generală a primei linii corespunzătoare unui fişier funcţie este:

```
function [iesire1, iesire2, ...] = nume(intrare1, intrare2, ...)
```

- `function` – cuvânt cheie care declară fişierul ca fişier funcţie (obligatoriu);
- `nume` – numele funcţiei (numele fişierului-M); acest nume nu poate fi identic cu cel al unui fişier `*.m` deja existent;

Sintaxă	Descriere
v(i:k)	Selectează elementele de pe poziţiile i, i+1, i+2, ..., k ale vectorului v; dacă i>k, vectorul rezultat este gol
v(i:j:k)	Selectează elementele de pe poziţiile i, i+j, i+2j, ..., k (cu pasul j); dacă j>0 şi i>k sau, dacă j>0 şi i<k, vectorul rezultat este gol
v([i, j, k])	Selectează elementele de pe poziţiile i, j, k
v(:)	Dacă vectorul este un vector linie devine vector coloană; dacă este vector coloană atunci rămâne nemodificat
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, ..., k ale matricei M
M(i:j:k, :)	Selectează liniile i, i+j, i+2j, ..., k ale matricei M
M(i:j, k:l)	Extrage submatricea formată cu elementele aflate la intersecţia liniilor de la i la j şi coloanelor de la k la l ale matricei M
M(:, [i, j, k])	Selectează coloanele i, j, k ale matricei M
M([i, j, k], :)	Selectează liniile i, j, k ale matricei M
M([i, j, k], [l, m, n])	Extrage submatricea formată cu elementele aflate la intersecţia liniilor i, j, k şi coloanelor l, m, n
M(:, :)	Selectează întreaga matrice M
M(:)	Selectează toate elementele matricei M şi le returnează într-un vector coloană

Tabela 1.6: Sintaxe MATLAB pentru selectarea elementelor vectorului/matricei

Sintaxă	Descriere
length(v)	Returnează lungimea (numărul de elemente) vectorului v
[l, c] = size(v)	Una dintre dimensiuni va fi egală cu 1; dacă v este un vector linie l=1; dacă v este un vector coloană c=1
length(M)	Returnează valoarea celei mai mari dimensiuni a matricei M
[l, c] = size(M)	Returnează dimensiunea matricei M în variabile separate l şi c
det(M)	Returnează determinantul matricei pătratice M; dacă M conţine doar valori întregi, rezultatul este de asemenea un întreg
inv(M)	Returnează inversa unei matrice pătratice M; va apărea un mesaj de eroare dacă M este o matrice singulară

Tabela 1.7: Dimensiunea unei matrice. Determinant şi inversă

- `iesire1, iesire2, ...` – parametrii de ieşire ai funcţiei; aceştia trebuie separaţi prin virgule şi cuprinşi între paranteze drepte; dacă funcţia nu are parametri de ieşire parantezele drepte şi semnul egal nu au nici un rost;
- `intrare1, intrare2, ...` – parametrii de intrare ai funcţiei; aceştia trebuie separaţi prin virgule şi cuprinşi între paranteze rotunde; dacă funcţia

Sintaxă	**Descriere**
v = initial:pas:final	Produce un vector linie v cu elemente începând de la initial la final, cu pasul pas (pas poate avea valoare pozitivă sau negativă)
v = initial:final	Produce un vector linie v cu elemente începând de la initial la final, cu pasul 1
v = linspace(a, b, n)	Generează un vector linie v cu elemente începând de la a la b, cu pas constant, şi având un număr de elemente egal cu n
v = logspace(a, b)	Generează un vector linie v având 50 de elemente distribuite logaritmic între 10^a şi 10^b
v = logspace(a b, n)	Generează un vector linie cu n elemente distribuite logaritmic între 10^a şi 10^b
x = [ ]	Generează o matrice goală (fără nici un element)
ones(n)	Returnează o matrice nXn cu toate elementele 1
ones(m, n)	Returnează o matrice mXn cu toate elementele 1
ones(size(M))	Returnează o matrice de dimensiunea lui M doar cu elemente 1
zeros(n)	Returnează o matrice nXn cu toate elementele 0
zeros(m, n)	Returnează o matrice mXn cu toate elementele 0
zeros(size(M))	Returnează o matrice de dimensiunea lui M doar cu elemente 0
eye(n)	Returnează o matrice identitate nXn
eye(m, n)	Returnează o matrice mXn: 1 pe diagonala principală, 0 în rest
eye(size(M))	Returnează o matrice identitate de dimensiunea matricei M
rand(n)	Returnează o matrice nXn cu valori aleatoare, distribuite uniform în intervalul $(0, 1)$
rand(m, n)	Returnează o matrice mXn cu valori aleatoare distribuite uniform
rand(size(M))	Returnează o matrice cu valori aleatoare, distribuite uniform, de dimensiunea matricei M
randn(n)	Produce o matrice nXn cu valori aleatoare, cu distribuţie normală (Gaussiană) (medie 0, varianţă $\sigma^2 = 1$, deviaţie standard $\sigma = 1$)
randn(m, n)	Produce o matrice mXn cu valori aleatoare, cu distribuţie normală
randn(size(M))	Produce o matrice cu valori aleatoare, cu distribuţie normală, de dimensiunea matricei M
diag(v)	Returnează o matrice diagonală nXn, cu valorile vectorului v pe diagonala principală
diag(v, k)	Returnează o matrice simbolică pătratică de ordin n+abs(k) (n - numărul de elemente al lui v), cu elementele lui v pe diagonala k; k=0 semnifică diagonala principală
diag(M)	Dacă M este o matrice simbolică pătratică, returnează diagonala principală a matricei M
diag(M, k)	Dacă M este o matrice simbolică pătratică, returnează un vector coloană cu elementele diagonalei k din matricea M

Tabela 1.8: Vectori şi matrice uzuale

nu are parametri de intrare, parantezele rotunde şi semnul egal nu au nici un rost.

1.1.5 Instrucţiuni de control logic

1. Instrucţiunea condiţională `if` , clauza `else` , clauza `elseif`

```
if ExprLogica      % dacă expresia logică ExprLogica este adevărată (este evaluată ca
   instructiuni    % 1 logic), MATLAB-ul execută toate instrucţiunile instructiuni
end                % dintre if şi end; dacă condiţia este falsă (evaluată ca 0 logic),
                   % MATLAB-ul sare peste instrucţiunile dintre liniile if şi end,
                   % şi trece la executarea liniei ce urmează după instrucţiunea end
```

```
if ExprLogica1     % dacă expresia logică ExprLogica1 este adevărată, MATLAB-ul
   instructiuniA   % execută toate instrucţiunile instructiuniA dintre if şi clauza
else               % else; dacă condiţia este falsă, MATLAB-ul sare peste toate
   instructiuniB   % instrucţiunile dintre liniile if şi else, şi trece la executarea
end                % instrucţiunilor instructiuniB dintre clauza else şi end
```

Dacă funcţia ce trebuie evaluată are mai multe niveluri de instrucţiuni `if-else` , este dificil de determinat expresia logică adevărată, care selectează grupul de istrucţiuni ce urmează a fi executat; în acest caz se foloseşte clauza `elseif` .

```
if ExprLogica1      % dacă ExprLogica1 este adevărată, se execută doar
   instructiuniA    % instructiuniA; dacă ExprLogica1 este falsă şi dacă
elseif ExprLogica2  % ExprLogica2 este adevărată, se execută doar instructiuniB;
   instructiuniB    % dacă ExprLogica1 este falsă şi ExprLogica2 este de
elseif ExprLogica3  % asemenea falsă, şi dacă ExprLogica3 este adevărată,
   instructiuniC    % se execută doar instructiuniC; dacă mai multe expresii logice
end                 % sunt adevărate, prima expresie adevărată determină grupul
                    % de instrucţiuni care se execută prima dată; dacă toate
                    % expresiile logice sunt false, nu se execută nici un grup de
                    % instrucţiuni dintre liniile if şi end
```

2. Instrucţiunea repetitivă `for` – repetă grupul de instrucţiuni de un anumit număr de ori

```
for index = expresie
   instructiuni
end
```

- `index` – nume contor;
- `expresie` – o matrice, un vector sau un scalar;
- de cele mai multe ori `expresie` este de forma `initial:pas:final` (`initial` – prima valoare pentru `index` ; `pas` – incrementul pentru `index` (dacă

este omis se consideră valoarea implicită 1); `final` – valoarea maximă pentru `index`);

- pentru `index` parcurgând intervalul de la `initial` la `final` cu incrementul `pas`, se execută grupul de instrucţiuni `instructiuni` de un număr de ori egal cu $\left[\frac{\text{final} - \text{initial}}{\text{pas}}\right]_*$ ($[\,]_*$ – parte întreagă); `instructiuni` pot fi orice expresii MATLAB.

La utilizarea buclei `for` trebuie respectate anumite reguli [3]: indexul buclei `for` trebuie să fie o variabilă; dacă expresia este o matrice goală, bucla nu se execută, şi se va trece la executarea liniei ce urmează după instrucţiunea `end`; dacă expresia este un scalar, bucla se execută o singură dată, cu indexul dat de valoarea scalarului; dacă expresia este un vector linie, bucla se execută de un număr de ori egal cu numărul elementelor vectorului, de fiecare dată indexul având valoarea egală cu următorul element din vector; dacă expresia este o matrice, indexul va avea la fiecare iteraţie valorile conţinute în următoarea coloană a matricei; la terminarea ciclurilor `for`, indexul are ultima valoare folosită.

3. Instrucţiunea repetitivă `while` – repetă grupul de instrucţiuni atât timp cât expresia de control este adevărată

```
while expresie
   instructiuni
end
```

4. Instrucţiunea `Break` – încheie executarea unei bucle `for` sau a unei bucle `while`; la întâlnirea unei instrucţiuni `break`, execuţia continuă cu instrucţiunea ce urmează după buclă; în cazul unor cicluri imbricate, `break` comandă ieşirea din ciclul cel mai interior.

5. Operatori relaţionali şi logici – în cadrul unui algoritm, de cele mai multe ori este necesară selecţia grupului de instrucţiuni ce urmează a fi executat, condiţionată de valoarea de adevăr a unei expresii. Instrucţiunile condiţionale utilizează operatori relaţionali şi logici.

   În MATLAB există şase operatori relaţionali folosiţi la compararea a două matrice cu aceeaşi dimensiune (vezi tabelul 1.9). Un operator relaţional compară două matrice sau expresii matriceale, element cu element. Rezultatul este tot o matrice cu dimensiunea egală cu cea a matricelor comparate, cu elementele: **1** – dacă relaţia este adevărată; **0** – dacă relaţia este falsă.

Pentru combinarea a două sau mai multor expresii logice se folosesc operatorii logici din tabelul 1.10.

Operator	
<	Mai mic
<=	Mai mic sau egal
>	Mai mare
>=	Mai mare sau egal
==	Egal
∽=	Diferit

Tabela 1.9: Operatori relaţionali

Operator		Prioritate
∽	Nu	1
&&	Şi	2
||	Sau	3

Tabela 1.10: Operatori logici

1.1.6 Reprezentări grafice

Dacă `v`, `x` şi `y` sunt vectori (`x` şi `y` au aceeaşi lungime) şi `M` şi `N` sunt două matrice cu aceeaşi dimensiune, câteva sintaxe utilizate la reprezentarea grafică sunt ilustrate în tabelul 1.11 [4].

În MATLAB se pot utiliza diferite tipuri de linii, markere şi culori pentru reprezentările grafice (vezi tabelul 1.12). Funcţia `stem` realizează o reprezentare în formă discretă a datelor. În reprezentările 2-D, liniile pleacă de pe axa-x.

Se pot realiza şi reprezentări grafice în coordonate logaritmice; pentru aceasta se folosesc funcţiile `loglog`, `semilogx` şi `semilogy`. Sintaxele rămân aceleaşi ca la reprezentările în coordonate liniare, singura diferenţă fiind dată de modul de scalare a axelor. Funcţia `loglog` scalează ambele axe (abscisa şi ordonata) folosind logaritmul în baza 10, ca atare, pe axe vom avea puteri ale lui 10. Funcţia `semilogx` realizează acelaşi tip de scalare, dar doar pe abscisă, iar pentru `semilogy` scalarea se realizează doar pe ordonată.

Dacă se doresc mai multe reprezentări în aceeaşi fereastră grafică se poate utiliza comanda `subplot`. Aceasta împarte fereastra grafică curentă în mai multe "subferestre". De exemplu, dacă dorim să împărţim fereastra grafică în 2×3 subgrafice, vom avea următoarea ordine: $\begin{array}{|c|c|c|}\hline 1 & 2 & 3 \\ \hline 4 & 5 & 6 \\ \hline \end{array}$. Funcţia `subplot(m, n, p)` crează o axă în subfereastra `p` corespunzătoare figurii împărţită într-o matrice de dimensiune `mxn` [6]. Noile axe devin axe curente. Graficele pot fi personalizate. Dacă se doreşte vizualizarea numai a unei anumite porţiuni dintr-un grafic, corespunzătoare unor anumite intervale pe abscisă şi ordonată, se poate folosi comanda `axis`, imediat după comanda `plot` (sau orice altă funcţie care are ca rezultat reprezentarea grafică). Sintaxa acesteia este:

```
axis([x0 x1 y0 y1])  % setează limitele de pe axa x şi y
```

Sintaxă	**Descriere**
plot(v)	Dacă v este un vector cu elemente reale, se reprezintă grafic elementele sale în funcţie de indici (primul indice este 1 iar ultimul este egal cu lungimea vectorului); dacă v are valori complexe, reprezentarea se face în funcţie de partea reală (pe abscisă) şi de partea imaginară (pe ordonată)
plot(M)	Se reprezintă pe acelaşi grafic coloanele matricei M funcţie de indice, dacă M are valori reale; dacă M are valori complexe, sintaxa este echivalentă cu plot(real(M), imag(M))
plot(x, y)	Se reprezintă grafic elementele vectorului y funcţie de elementele vectorului x; lungimile celor doi vectori trebuie să fie egale
plot(x, M)	Dacă lungimea lui x este egală cu numărul de linii din M, se reprezintă grafic coloanele matricei M funcţie de x; dacă lungimea lui x este egală cu numărul de coloane din M, se reprezintă liniile matricei M funcţie de x; dacă lungimea lui x nu este egală nici cu numărul de linii, nici cu numărul de coloane, atunci reprezentarea nu este posibilă
plot(M, N)	Se reprezintă coloanele lui N funcţie de coloanele lui M (coloana k a matricei N se reprezintă funcţie de coloana k a lui M); dimensiunile celor două matrice trebuie să fie egale
plot(x1, y1, x2, y2, ..., xn, yn)	Se reprezintă pe acelaşi grafic y1 funcţie de x1, y2 funcţie de x2, ..., yn funcţie de xn (pot fi vectori sau matrice); rămân valabile considerentele de la sintaxele anterioare

Tabela 1.11: Reprezentare grafică în coordonate liniare

Se poate preciza titlul graficului (`title`), se poate eticheta abscisa (`xlabel`), ordonata (`ylabel`), sau se poate plasa un anumit text pe grafic. Toate aceste comenzi se utilizează după comanda de reprezentare grafică (vezi tabelul 1.13). Un anumit text poate fi plasat pe grafic în unul dintre cele două moduri: folosind comanda `text` sau `gtext` . Prima comandă implică cunoaşterea coordonatelor unde se doreşte amplasarea textului: `text(xcor, ycor, 'textstring')` . Pentru utilizarea celei de a doua funcţii, nu trebuie cunoscute coordonatele: `gtext('textstring')` şi, apoi, cu ajutorul mouse-ului se poziţionează textul în locul dorit.

În aceeaşi figură se pot reprezenta mai multe grafice, unul peste altul, punând

Tip linie		Tip marker		Culoare	
–	Continuă (predefinită)	+	Plus	r	Roşu
		∘	Cerc	g	Verde
––	Întreruptă	∗	Steluţă	b	Albastru
:	Punctată	·	Punct	c	Cyan
–.	Linie-punct	×	Cruce	m	Magenta
		'square' or s	Pătrat	y	Galben
		'diamond' or d	Romb	k	Negru
		^ / v	Triunghi cu vârful în sus/jos	w	Alb
		> / <	Triunghi cu vârful în dreapta/stânga		
		'pentagram' or p	Stea cu cinci vârfuri		
		'hexagram' or h	Stea cu şase vârfuri		

Tabela 1.12: Tipuri de linii, markere şi culori

Syntaxă	Descriere
title('text')	Plasează deasupra graficului, la mijloc, ca titlu text
xlabel('text')	Textul text devine eticheta de pe abscisă
ylabel('text')	Textul text devine eticheta de pe ordonată

Tabela 1.13: Etichetarea graficelor

mai mulţi parametri funcţiei `plot`. Dacă se doreşte acest lucru, este bine ca diferitele grafice să fie diferenţiate prin culori şi markere diferite. Se poate realiza acelaşi lucru folosind comenzile `hold on` (reţine graficul curent şi adaugă în aceeaşi fereastră grafică următoarele reprezentări grafice) şi `hold off` (dezactivează comanda `hold on`).

Dacă în cadrul unui program se doreşte 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.

1.2 Desfăşurarea lucrării

1. Să se verifice toate formatele de afişare pentru valoarea `x = pi` (vezi tabelul 1.3).
2. Să se verifice următoarele exemple scriind direct în fereastra de comandă:

```
>> p = pi;       % valoarea p nu va fi afișată (există în zona de memorie)
>> % r = pi/4    % această linie nu este luată în considerare
>> v = r/2       % va apărea un mesaj de eroare, deoarece r nu este recunoscut
>> s = 1+2+3 [Enter]  % instrucțiunea va continua pe linia următoare
```

```
+4+5+6
```

3. Se consideră matricea $\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ şi vectorii $\mathbf{B} = \begin{bmatrix} 7 & 8 & 9 \end{bmatrix}$ şi $\mathbf{C} = \begin{bmatrix} -2 & -1 \end{bmatrix}^T$. Să se verifice sintaxele din tabelul 1.6.

4. Să se afle elementele vectorilor:

```
v = 0:10:40
u = 2:8
d = 20:-2:2
```

```
linspace(4, 16, 4)
linspace(pi,-pi,3)
logspace(1, 3)
```

```
logspace(1, pi)
logspace(1, 2, 6)
logspace(0, pi, 3)
```

5. Să se afle elementele matricelor:

```
>> ones(2)
>> ones(3, 2)
>> ones(size(D))
>> zeros(4)
```

```
>> zeros(5, 2)
>> eye(2)
>> eye(size(D))
>> rand(2)
```

```
>> rand(1, 4)
>> rand(size(D))
>> randn(2)
>> randn(1, 4)
```

6. Să se genereze două matrice `M = randn(5)` şi `N = randn(3, 3)`, şi să se afle valorile determinanţilor acestora.

7. Să se genereze un vector linie `a = randn(1, 4)` şi o matrice `A = randn(4)`. Să se verifice matricele:

```
>> diag(a)
>> diag(a, 1)
```

```
>> diag(a, -1)
>> diag(A)
```

```
>> diag(A, -2)
>> diag(diag(A))
```

8. Să se genereze o matrice `C = randn(3, 4)`, un vector linie `a = randn(1, 5)` şi un vector coloană `b = randn(5, 1)`. Să se afle lungimea acestora (`length`) şi dimensiunea lor (`size`).

9. O funcţie care realizează media aritmetică a valorilor vectorului `x`, poate fi scrisă astfel:

```
function m = medieAritm(x)
n = length(x); % lungimea vectorului x
m = sum(x)/n; % media aritmetică a valorilor vectorului x
y = ['Media aritmetica este:', num2str(m)];
disp(y);
```

Fişierul astfel scris se salvează sub denumirea `medieAritm.m` [7]. Funcţia definită anterior poate fi apelată din linia de comandă MATLAB, dintr-un script sau dintr-o altă funcţie:

```
>> x = [6 5 4 3 2 1]; medieAritm(x);
```

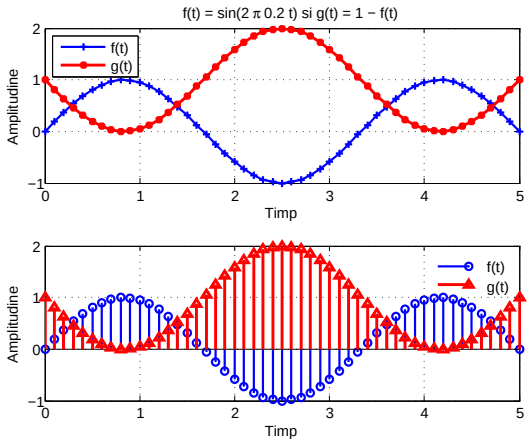

Figura 1.1: Graficele pentru $f(t)$ şi $g(t)$

```
Media aritmetica este: 3.5
```

10. Să se reprezinte grafic funcţia $f(t) = \sin(2\pi 0.3t)$, cu culoare albastră şi linie-plus, şi funcţia $g(t) = 1 - f(t)$, cu culoare roşie şi linie-steluţă. Să se eticheteze axa-x `Timp`, iar axa-y `Amplitudine`; titlul graficului trebuie să fie `f(t) = sin(2π0.3t) si g(t) = 1 - f(t)`. Pentru reprezentarea grafică să se utilizeze comanda `plot` în subfigura 1 şi comanda `stem` în subfigura 2. Fiecare subfereastră trebuie să conţină legendă [7] (ca în figura 1.1). Un fişier script denumit `fgPlot.m` poate fi scris astfel:

```
% Reprezentare grafica a doua functii, etichetarea axelor, adaugarea unui titlu
t = 0:0.1:5; f = sin(2*pi*0.3*t);  g = 1-f;
figure(1); subplot(211); plot(t, f, '-+b', 'LineWidth', 1.5);
hold on    plot(t, g, '-*r', 'LineWidth', 2);
hold off; grid; xlabel('Timp'); ylabel('Amplitudine');
legend('boxon'); l1 = legend('f(t)', 'g(t)', 2);
title('f(t) = sin(2\pi0.2t) and g(t) = 1-f(t)');
subplot(212); stem(t, f, 'ob', 'LineWidth',1.5);
hold on       stem(t, g, '^r', 'LineWidth', 2);
hold off; grid; xlabel('Timp'); ylabel('Amplitudine');
legend('boxoff'); l2 = legend('f(t)', 'g(t)', 1);
```

1.3 Exerciţii

1. Să se genereze un vector cu pas liniar, între 3 şi 9, cu increment 2.
2. Să se genereze un vector cu pas liniar, cu 13 elemente, între 3 şi 9.
3. Să se genereze un vector cu 9 elemente distribuite logaritmic, între 10^{-3}

şi 10^3.

4. Se consideră vectorul `y = 3:0.9:123`. Să se evalueze lungimea vectorului şi să se genereze un nou vector cu toate elementele 1, cu lungimea egală cu cea a vectorului `y`.

5. Se consideră matricele $\mathbf{A} = \begin{bmatrix} 3 & 2 & 1 \\ 8 & 4 & 5 \\ 0 & 2 & 0 \end{bmatrix}$ şi $\mathbf{B} = \begin{bmatrix} 2 & 3 & 4 \\ 1 & 1 & 1 \\ 2 & 3 & 2 \end{bmatrix}$, şi scalarul $m = 4$. Să se evalueze în MATLAB:

   - $\mathbf{C} = \mathbf{A} + \mathbf{B}$;
   - $\mathbf{D} = \mathbf{A} - \mathbf{B}$;
   - $\mathbf{E} = \mathbf{C} + m$;
   - $\mathbf{F} = \mathbf{A} \cdot \mathbf{B}$;
   - $\mathbf{G} = \mathbf{B} \cdot m$;
   - $\mathbf{H} = \mathbf{A}^T$;
   - $\mathbf{I} = \mathbf{B}^T$;
   - $\mathbf{J} = \mathbf{A}/\mathbf{B}$;
   - $\mathbf{K} = \mathbf{A} \setminus \mathbf{B}$;
   - $\mathbf{L} = \mathbf{C}^m$.

   Să se verifice dacă $\mathbf{J} = \mathbf{A} \cdot \mathbf{B}^{-1}$ şi dacă $\mathbf{K} = \mathbf{A}^{-1} \cdot \mathbf{B}$. Să se utilizeze formatul `long e`.

6. Să se evalueze produsul scalar al vectorilor: $\mathbf{a} = \begin{bmatrix} 1 & 2 \end{bmatrix}$, $\mathbf{b} = \begin{bmatrix} -3 & 3 \end{bmatrix}$.

7. Pentru matricele $\mathbf{A} = \begin{bmatrix} 9 & 8 & 7 \\ 6 & 5 & 4 \\ 3 & 2 & 1 \end{bmatrix}$ şi $\mathbf{B} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix}$ să se evalueze produsul element cu element.

8. Să se reprezinte grafic $x(n) = \sin\left(2\pi\frac{1}{5}n\right)$, $n = \overline{0,10}$, folosind comanda `stem`. Graficul trebuie reprezentat cu steluţe de culoare roşie; să se eticheteze axele şi să se adauge un titlu graficului.

9. Să se scrie o funcţie MATLAB `bplusa.m` care să realizeze suma a două variabile `a` şi `b`:

```
function suma = bplusa(a, b);
```

10. Să se scrie o funcţie `boria.m` care să evalueze produsul a doi vectori `a` şi `b`:

```
function produs = boria(a, b);
```

11. Să se scrie o funcţie MATLAB `medieGeom.m`, cu care să se evalueze media geometrică a doi scalari `a` şi `b`:

```
function mgeometrica = medieGeom(a, b);
```