Laborator 1 - RNSF

Introducere în OCTAVE

1. Introducere

GNU Octave reprezintă un limbaj de programare de nivel înalt, destinat cu precădere calculelor numerice și reprezentărilor grafice în domeniul științei și ingineriei. Acesta integrează analiza numerică, calculul matriceal, procesarea semnalului și reprezentările grafice. Octave reprezintă implementarea opensource pentru o parte din bibliotecile de funcții MATLAB.

Elementul de bază cu care operează Octave este matricea. În acest limbaj de programare nu se declară variabile (ca în cazul limbajului C) și nici nu se folosesc tipuri de date predefinite, deoarece orice set de date manipulat de utilizator este văzut ca o matrice (o zonă continuă de memorie). Toate programele scrise în Octave sunt interpretate (nu compilate) și sunt executate linie cu linie. Octave pune la dispoziție o consolă pentru lansarea în execuție a comenzilor, însă se pot edita și fișiere sursă cu ajutorul editorului de text.

2. Programul Octave

La deschiderea programului Octave, va apărea pe ecranul calculatorului fereastra de comenzi, în care simbolul „>" reprezintă prompterul Octave care se află la începutul fiecărei linii de comandă. În dreptul acestui simbol se poate scrie comanda dorită, aceasta fiind rulată prin apăsarea tastei ENTER.

Exemplu 2.1(tastarea unei comenzi):

```
> x=3

x =

     3
```

În urma acestei comenzi, în fereastra Workspace se poate observa apariția variabilei *x*, împreună cu o serie de informații corespunzătoare variabilei respective, informații precum: numele variabilei, dimensiunea variabilei etc.

Dacă se va introduce caracterul *;* după intrucțiune, în fereastra de comandă nu se va mai afișa comanda introdusă, însă variabila *x* va fi stocată în memoria Octave și poate fi vizualizată în fereastra Workspace.

Pe lângă fereastra de comandă, Octave pune la dispoziție și posibilitatea de utilizare a fișierelor sursă cu ajutorul editorului de text. Fișierul poate fi salvat cu extensia ***.m***, care este, în esență, un fișier ASCII, deci poate fi modificat cu ajutorul oricărui instrument de modificare a textului.

Pentru crearea unui fișier script se poate utiliza comanda *File →New*.

În cadrul acestui laborator, se vor utiliza fișiere text.

3. Vectori și Matrice

Octave oferă o soluție simplă de a defini un vector, utilizând sintaxa:

valoare_inițială:valoare_incrementare:valoare_sfârșit

Exemplu 3.1:

```
> vect = 1:2:9

vect =

     1     3     5     7     9
```

Această linie de cod definește o variabilă *vect*, care reprezintă un vector ce conține valorile 1,3,5,7 și 9. Cu alte cuvinte, vectorul pornește de la valoarea inițială: 1, incrementează valoarea inițială la fiecare pas cu valoarea de incrementare: 2 și se oprește când ajunge sau înainte de a depăși valoarea finala: 9.
În cazul în care nu este specificat pasul de incrementare, acesta este implicit 1.

Exemplu 3.2

```
> vect = 1:5

vect =

     1     2     3     4     5
```

Exercițiu 3.1: Să se definească un vector care conține toți multiplii de 3 de la 1 la 100.

Matricile pot fi definite prin separarea elementelor aparținând unei linii cu spațiu liber și terminarea unei linii (trecerea la linia următoare) cu ajutorul simbolului ***;***.

Exemplu 3.3

```
> A = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]

A =

    16     3     2    13
     5    10    11     8
     9     6     7    12
     4    15    14     1
```

Exercițiu 3.2: Să se definească o matrice, *M*, de trei linii și două coloane, având pe prima coloană numere în ordine crescătoare, iar pe cea de a doua coloană numere în ordine descrescătoare.

Elementele unei matrici se definesc în paranteze drepte **[ ]**, iar pentru accesarea elementelor matricii se folosesc paranteze rotunde: **( )**.

Exemplu 3.4:

```
> A(2,3)

ans =

    11
```

Exercițiu 3.3: Să se extragă din matricea *M* creată la exercițiul anterior elementul de pe linia 3 și coloana 2.

În cazul limbajului de programare Octave, **indecșii sunt numerotați pornind de la 1**, nu de la 0, ca în cazul altor limbaje precum C.

Limbajul Octave permite, de asemenea, și extragerea unei întregi coloane sau linii din matrice folosinf operatorul **:**.

Exemplu 3.5:

```
> A(:,3)

ans =

     2
    11
     7
    14
```

Uilizând această comandă, au fost extrase toate elementele din matricea A corespunzătoare coloanei 3.

Exemplu 3.6:

```
> A(2,:)

ans =

     5    10    11     8
```

Exercițiu 3.4: Să se extragă din matricea *M* creată la *Exercițiu 3.2* elementele de pe ultima coloană.

Asemănător extragerii unei coloane, poate fi extrasă din matrice și o linie, conform exemplului anterior.

De asemenea, se poate crea și o submatrice cu elementele matricei A.

Exemplu 3.7:

```
> A(2:4,3:4)

ans =

    11     8
     7    12
    14     1
```

Exercițiu 3.5: Să se extragă din matricea *M* creată la *Exercițiu 3.2* elementul de pe ultimile două linii și cele două coloane.

Această comandă a extras din matricea A toate elementele corespunzătoare liniilor de la 2 la 4 (2,3 și 4) de pe coloanele 3 și 4.

Concatenarea matricelor poate fi realizată cu ușurință prin utilizarea parantezelor drepte **[ ]**, asemănător cu concatenarea elementelor în vederea definirii unei matrice.

Exemplu 3.8

```
> A1 = [1 2 3; 6 7 8; 11 12 13];
> B1 = [4 5; 9 10; 14 15];
> C1 = [A1 B1]

C1 =

     1     2     3     4     5
     6     7     8     9    10
    11    12    13    14    15

> A2 = [1 6 11; 2 7 12; 3 8 13];
> B2 = [4 9 14; 5 10 15];
> C2 = [A2; B2]

C2 =

     1     6    11
     2     7    12
     3     8    13
     4     9    14
     5    10    15
```

Exercițiu 3.6: Să se concateneze matricea *C2* prezentă în exemplu cu o coloană ce conține cifrele [16 17 18 19 20], apoi cu o linie ce conține cifrele [6 11 16 21].

Limbajul Octave permite crearea ușoară a **matricei unitate, matricei cu conținut numai de zerouri și matricei cu conținut numai de unu**, utilizând funcțiile: ***eye***, ***zeros*** și ***ones***.

Exemplu 3.9:

```
> eye(3)

ans =

     1     0     0
     0     1     0
     0     0     1

> zeros(2,3)

ans =

     0     0     0
     0     0     0

> ones(3,2)

ans =

     1     1
     1     1
     1     1
```

Exercițiu 3.7: Să se definească următoarea matrice, utilizând funcțiile ***eye***, ***ones*** și ***zeros***:

$$M = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Inițializarea aleatoare a unei matrice poate fi realizată cu ajutorul funcțiilor ***rand*** sau ***randn***. Funcția ***rand*** inițializeaza matricea cu valori cuprinse între 0 și 1, valorile având o distribuție uniformă, în timp ce ***randn*** se utilizează pentru inițializarea matricei cu valori extrase din distribuția normală standard (unde media, μ, este 0, iar deviația standard, σ, este 1).

Exemplu 3.10:

```
> rand(3)

ans =

    0.1170    0.2462    0.5466
    0.8147    0.3427    0.5619
    0.3249    0.3757    0.3958

> randn(2,3)

ans =

   -0.1526    0.4583    0.6201
    0.0337    1.2816   -0.2867
```

Exercițiu 3.8: Să se genereze o matrice, *R*, de 2 coloane și 6 linii, în care prima coloană să conțină numere între 3 și 12, iar a doua coloană să conțină numere între 4 și 20.
Pentru alocarea aleatoare a valorilor în intervalul (minim; maxim), se poate utiliza următoarea ecuație:

$$a = (\text{maxim} - \text{minim}) * [0;\ 1] + \text{minim}$$

4. Operații de bază

4.1 Dimensiunea unei matrice

Pentru a afla dimensiunea unei matrice, se utilizează funcția ***size***.

Exemplu 4.1:

```
> size(A)

ans =

     4     4
```

4.2 Transpunerea unei matrice

Pentru transpunerea unei matrice se utilizează simbolul ’.

Exemplu 4.2:

```
> A'

ans =

    16     5     9     4
     3    10     6    15
     2    11     7    14
    13     8    12     1
```

Exercițiu 4.1: Se dă matricea *M*:

$$M = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 3 & 4 & 5 \\ 4 & 5 & 6 \end{bmatrix}$$

Să se determine dimensiunea matricei *M*.
Să se transpună matricea *M*, dupa care să se determine din nou dimensiunea acesteia.

4.3 Funcția sum

Funcția ***sum*** se poate utiliza pentru sumarea tuturor elementelor de pe linie sau de pe coloana unei matrice.

Exemplu 4.3

```
> sum(A)

ans =

    34    34    34    34
```

În cazul în care nu se specifică dimensiunea pe care să se realizeze suma elementelor matricei, aceasta se va realiza implicit pe coloane (instrucțiunea fiind echivalentă cu instrucțiunea ***sum(A,1)***).
Pentru calcularea sumei elementelor de pe fiecare linie, se va utiliza instrucțiunea ***sum(A,2)***.

Pentru calcularea sumei tuturor elementelor din matrice se poate utiliza instrucțiunea: ***sum(sum(A))***.

Exemplu 4.4

```
> sum(sum(A))

ans =

   136
```

Exercițiu 4.2: Să se calculeze suma elementelor de pe fiecare linie a matricei *M*, definite în Exercițiu 4.1 și să se calculeze suma tuturor elementelor matricei *M*.

4.4 Funcțiile min și max

Funcțiile ***min*** și ***max*** pot fi utilizate în cadrul limbajului Octave pentru determinarea valorii minime, respectiv maxime, de pe liniile sau coloanele unei matrice.

Exemplu 4.5

```
> min(A)
ans =
     4     3     2     1

> max(A)
ans =
    16    15    14    13

> min(A,[],2)
ans =
     2
     5
     6
     1

> max(A,[],2)
ans =
    16
    11
    12
    15


> min(min(A))
ans =
     1

> max(max(A))
ans =
    16
```

În cazul funcțiilor ***min*** și ***max***, al doilea parametru de intrare nu este specific dimensiunii pe care se dorește calcularea minimului, motiv pentru care va fi înlocuit cu o variabilă goală: [], urmând ca al treilea parametru de intrare să corespundă dimensiunii.

Exercițiu 4.3: Să se determine valorile minime și maxime de pe fiecare linie a matricei *M* definită în Exercițiu 4.1. Să se determine și valorile minime și maxime a întregii matrice *M*.

5. Instrucțiuni

5.1 Instrcțiunea if

Sintaxa utilizată în Octave pentru instrucțiunea if este următoarea:

```
if expresie
    instrucțiuni
end
```

Efect: dacă expresia este adevărată, atunci se execută intrucțiuni, altfel, dacă expresia este falsă, nu se execută nicio instrcțiune.

```
if expresie
    instrucțiuni_1
else
    instrucțiuni_2
end
```

Efect: dacă expresia este adevărată, se execută instrucțiuni_1, altfel, se execută instrucțiuni_2.

Exemplu 5.1:

```
> x = 122;

if mod(x,2) == 0 % Functia mod returneaza restul impartitii variabilei x la 2
    fprintf('Numarul este par \n')
else
    fprintf('Numarul este impar \n')
end

Numarul este par
```

Exercițiu 5.1: Se dorește să se determine dacă un număr *x* este inclus în intervalul (15;35). În cazul în care numărul se află în intervalul dat, să se afișeze „*Numărul se află în interval*”, altfel să se afișeze mesajul „*Numarul NU se află în acest interval*”.

5.2 Switch

Sintaxa:

```
switch expresie
    case const_1
        instructiune_1
    case const_2
        instructiune_2
    case const_n
        instructiune_n
end
```

Efect: Rezultatul expresiei va conduce la una dintre constante: const_1, const_2, …,const_n, urmând să fie executată instrucțiunea corespunzătoare constantei rezultate în urma expresiei.

```
switch expresie
    case const_1
        instructiune_1
    case const_2
        instructiune_2
    case const_n
        instructiune_n
   otherwise
        instructiune_implicita
end
```

Efect: În cazul în care rezultatul expresiei va conduce la una dintre constante: const_1, const_2, …,const_n, în continuare va fi executată instrucțiunea corespunzătoare constantei rezultate în urma expresiei, altfel, se va merge pe ramura otherwise și se va executa instrucțiunea implicită.

Exemplu 5.2:

```
calificativ = 'B';

   switch(calificativ)
   case 'I'
      fprintf('Insuficient!\n' );
   case 'S'
      fprintf('Suficient\n' );
   case 'B'
      fprintf('Bine\n' );
   case 'FB'
      fprintf('Foarte bine\n' );
   otherwise
      fprintf('Calificativ necunoascut\n' );
   end
```

```
Bine
```

Exercițiu 5.2: Să se creeze un meniu de băuturi care să conțină: apă minerală (3 lei), apă plată (3 lei), coca-cola (5 lei), fanta (5 lei), suc de portocale (8 lei). În funcție de alegerea făcută, să se afișeze mesajul: *„Nota de plata este de x lei"* sau *„Această alegere nu se află în meniu"*, în cazul în care alegerea facută nu corespunde niciuneia dintre variante.

5.3 For

Sintaxă:

```
for index=val_init:pas:val_fin
    instrucțiuni
end
```

Efect: Pentru index pornind de la valoarea inițială (val_init) până la valoarea finală (val_fin) cu pasul pas se execută instrucțiuni

Exemplu 5.3:

```
% Calcularea sumei numerelor pare de la 1 la 10 - instructiunea for
suma = 0;

for i=2:2:10
    suma = suma + i;
end

suma =

    30
```

Exercițiu 5.3: Să se definească un vector de 20 de elemente utilizând instrucțiunea ***for***, iar fiecare element să fie un multiplu de 5.

5.4 While

Sintaxă:

```
while expresie
    instrucțiuni
end
```

Efect: Cât timp expresia este adevărată, se execută instrucțiuni.

Exemplu 5.4:

```
% Calcularea sumei numerelor pare de la 1 la 10 - instructiunea while
suma = 0;
i = 10;

while i
    suma = suma + i;
    i = i - 2;
end

suma =

     30
```

Exercițiu 5.4: Să se calculeze factorialul unui număr utilizând instrucțiunea ***while***.

6. Funcții

Sintaxa:

```
function [lista_var_iesire] = nume_fct(lista_var_intrare)
    corp_functie
end
```

Unde:

- *nume_fct* reprezintă numele funcției
- *lista_var_intrare* reprezintă lista variabilelor de intrare
- *lista_var_iesire* reprezintă lista variabilelor care vor fi transmise ca parametrii de ieșire din funcție
- *corp_functie* reprezintă instrucțiunile din interiorul funcției care vor fi executate în momentul apelului acesteia

O funcție în Octave este asemănătoare la nivel funcțional cu o funcție dintr-un alt limbaj de programare, precum C/C++: primește parametrii de intrare, execută instrucțiuni, returnează rezultate.

Această funcție trebuie salvată într-un fișier cu **acelasi** nume corespunzător funcției (nume_fct).

Apelul funcției se realizează utilizând sintaxa: [lista_var_iesire] = nume_fct(lista_var_intrare).

Exemplu 6.1(crearea funcției):

```
function [ medie ] = calcul_medie (v)
% Calcularea mediei elementelor dintr-un vector

    medie = 0;

    for i=1:length(v)
        medie = medie + v(i);
    end

    medie = medie/length(v);

end
```

Exemplu 6.2 (apelarea funcției):

```
vector = [1 3 6 9 12];
val_medie = calcul_medie (vector)

val_medie =

    6.2000
```

Exercițiu 6.1: Să se creeze o funcție cu numele *dist_Eulcidiana* care calculează distanța Euclidiană dintre doi vectori, primiți ca parametrii de intrare. Distanța Euclidiană se calculează conform relaiei:

$$d = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2}$$

unde x și y reprezintă doi vectori de dimensiune N. Pentru obținerea radicalului, sepoate utiliza funcția ***sqrt***.

7. Operații artitmetice

7.1 Operatorii de adunare și scădere

Limbajul Octave oferă posibilitatea de a realiza cu ușurință operații atât asupra vectorilor cât și asupra matricelor.

Astfel că, limbajul Octave permite ca operatorul + și - să adune sau să scadă două matrice element cu element.

Exemplu 7.1:

```
> A = [1 2 1; 0 1 1; 2 1 1];
> B = [0 0 1; 1 1 0; 1 1 2];
> A+B

ans =

     1     2     2
     1     2     1
     3     2     3

> A-B

ans =

     1     2     0
    -1     0     1
     1     0    -1
```

7.2 Operatorul de înmulțire

Utilizând operatorul de înmulțire *, acesta va realiza înmulțirea matriceală dintre matricea A și matricea B, în timp ce opartorul .* va înmulți element cu element cele două matrice.

Se dau două matrice *A*, de dimensiune NxM și *B*, de dimensiune MxP:

$$A = [a_{ij}] = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1M} \\ a_{21} & a_{22} & \cdots & a_{2M} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{NM} \end{bmatrix}$$

$$B = [b_{ij}] = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1P} \\ b_{21} & b_{22} & \cdots & b_{2P} \\ \vdots & \vdots & \ddots & \vdots \\ b_{M1} & b_{M2} & \cdots & b_{MP} \end{bmatrix}$$

Înmulțirea matriceală se realizează conform relației:

$$C = A*B = [c_{ij}] = \left[\sum_{k=1}^{M} a_{ik} \cdot b_{kj}\right]$$

În timp ce înmulțirea element cu element se realizează conform ecuației:

$$D = A.*B = [d_{ij}] = [a_{ij} \cdot b_{ij}]$$

Exemplu 7.2

```
>> A*B

ans =

     3     3     3
     2     2     2
     2     2     4

>> A.*B

ans =

     0     0     1
     0     1     0
     2     1     2
```

Exercițiu 7.1: Să se înmulțească matriceal următoarele matrice *M* și *N*:

$$M = \begin{bmatrix} 10 & 3 & 5 \\ 8 & 4 & 9 \end{bmatrix}, N = \begin{bmatrix} 3 & 4 \\ 5 & 9 \\ 14 & 3 \end{bmatrix}$$

Rezultat:

$$M*N = \begin{bmatrix} 115 & 82 \\ 170 & 95 \end{bmatrix}$$

8. Citirea datelor dintr-un fișier și salvarea satelor într-un fișier

Încărcarea datelor dintr-un fișier având extensia ***.mat*** se poate realiza cu ajutorul funcției ***load.*** Această funcție primește ca parametru de intrare numele fișierului care se dorește a fi încărcat în workspace. În cazul în care fișierul **NU** se găsește în folderul curent, atunci este necesară **specificarea întregii adrese la care se găsește fișierul respectiv.**

Exemplu 8.1

```
load('matrice.mat');
load('C:\RNSF\matrice.mat')
```

Salvarea varabilelor din workspace se poate realiza cu ajutorul funcției ***save***. Primul parametru de intrare al acestei funcții este reprezentat de numele fișierului, iar al doilea parametru reprezintă variabila care se dorește a fi salvată. În cazul încare cel de-al doilea parametru lipsește, se va salva în fișierul respectiv toate variabilele prezente în workspace

Exemplu 8.2:

```
save('matrice.mat','A')
```

9. Reprezentarea grafică în Octave

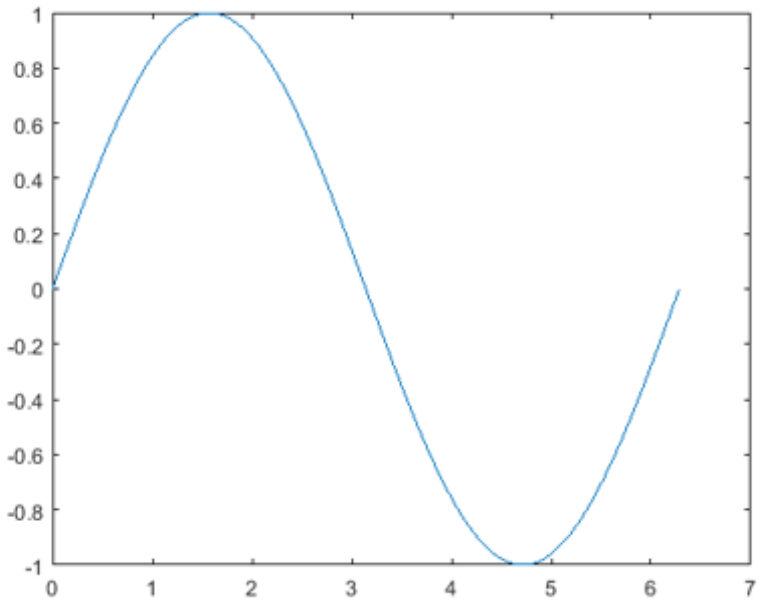
9.1 Funcția *plot*

Pentru reprezentarea grafică a unui semnal unidimensional se poate utiliza funcția ***plot***.

Exemplu 9.1:

```
> x = 0:pi/100:2*pi;
y = sin(x);
plot(x,y)
```

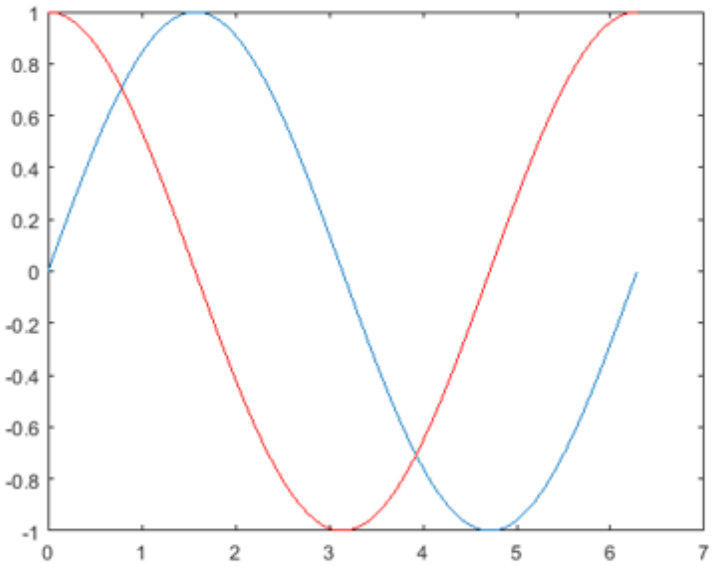
Programul va afișa următoarea imagine:



În cazul în care se dorește suprapunerea a două grafice, se poate utiliza funcția ***hold on***.

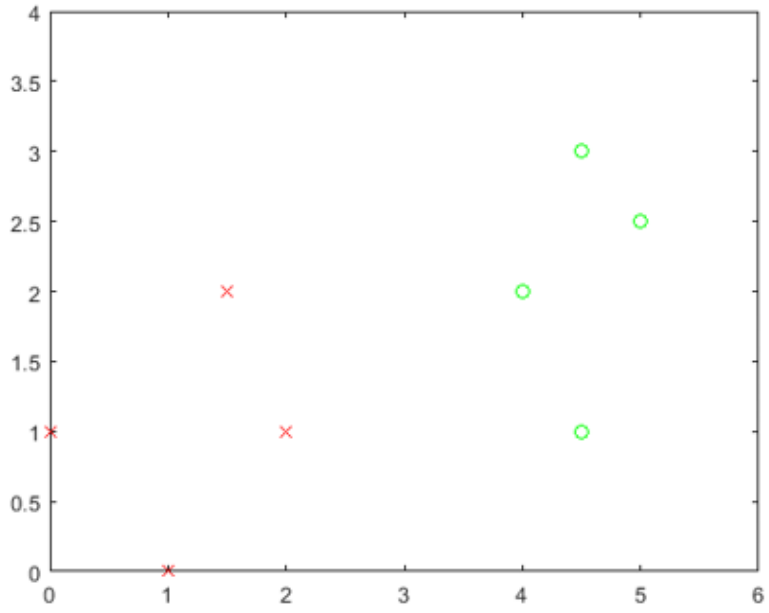
Exemplu 9.2:

```
> x = 0:pi/100:2*pi;
y = sin(x);
plot(x,y)
hold on, plot(x, cos(x), 'r')
```



Exemplu 9.3:

```
> x1=[0 1 1.5 2];
> y1=[1 0 2 1];
> x2 = [4 4.5 4.5 5];
> y2 = [2 1 3 2.5];
> plot(x1,y1,'xr')
> hold on, plot(x2,y2,'og')
> axis([0 6 0 4])
```



Exercițiu 9.1: Să se definească următorii vectori: *x1*, ce conține 100 de valori alocate aleator în intervalul (10; 30), *x2*, ce conține 100 de valori alocate aleator în intervalul (5; 20), *y1*, ce conține 50 de valori alocate aleator înintervalul (15; 35) și *y2*, ce conține 50 de valori alocate aleator în intervalul (10; 30). Să se reprezinte pe același grafic vectorii *x2* în funcție de *x1* și *y2* în funcție de *y1*, cele două reprezentări având culori diferite.

10.Desfășurarea lucrării

Exercițiu 4.1: Se dau vectorii:

x = [5 7 3 2 1 6];
y = [7 6 12 9 8 10];
Să se calculeze media aritmetică a celor doi vectori precum si variația și covariația acestora.
Formula varianței este:

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_x)^2$$

Unde:

- N reprezintă numărul de elemente ale vectorului x
- μ_x reprezintă media vectorului x

Formula covarianței dintre doi vectori este:

$$\sigma_{x,y}^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_x)(y_i - \mu_y)$$

Pasul 1: Se deschide un script nou utilizând comanda *File →New*. Se va salva într-un folder denumit Grupa [Nr. Grupei] fișierul cu denumirea Exercitiu10_1.
Pasul 2: Se vor defini cei doi vectori: *x* și *y*.

```
x = [5 7 3 2 1 6];
y = [7 6 12 9 8 10];
```

Pasul 3: Se vor determina dimensiunile celor doi vectori *x* și *y* utilizând funcția ***size***. Rezultatele obținute vor fi stocate în variabilele *N* și respectiv *M*. Rezultate: N = 6, M = 6.

```
N = size(x,2);
M = size(y,2);
```

Pasul 3: Se calculează media variabilei *x* cu ajutorul funcției ***mean*** și se salvează rezultatul obținut în variabila *mu_x*. Rezultatul obținut: mu_x = 4.

```
mu_x = mean(x);
```

Pasul 4: Se calculează media variabilei *y* cu ajutorul funcției ***mean*** și se salvează rezultatul obținut în variabila *mu_y*. Rezultatul obținut: mu_y = 8.66.

```
mu_y = mean(y);
```

Pasul 5: Se va calcula diferența dintre vectorul *x* și variabila *mu_x* obținută anterior la Pasul 3, utilizând operatorul – și se va salva rezultatul în variabila *dif_x*. Rezultatul obținut va fi de forma: dif_x =
1 3 -1 -2 -3 2

```
dif_x = x - mu_x;
```

Pasul 6: Fiecare element din variabila *dif_x* va fi ridicat la pătrat utilizând operatorul **.^**, iar rezultatul va fi salvat în variabila *patrat_x*. Rezultatul obținut: patrat_x = 1 9 1 4 9 4

```
patrat_x = dif_x.^2;
```

Pasul 7: Toate elementele variabilei *pătrat_x* vor fi însumate cu ajutorul funcției ***sum***, iar rezultatul va fi salvat în variabila *suma_x*. Rezultatul obținut: suma_x = 28.

```
suma_x = sum(patrat_x);
```

Pasul 8: În continuare, se va calcula varianța vectorului *x* prin împărțirea rezultatului obținut în variabila *suma_x* la *N*. Rezultatul obținut va fi salvat in variabila *sigma_x*. Rezultatul obținut în urma calculării variației pentru x este: 4.66.

```
sigma_x = suma_x/N;
```

Pasul 9: Se repetă pașii pornind de la Pasul 5 până la Pasul 8 pentru vectorul *y*. Rezultatul obținut în urma calculării variației pentru y este: 3.88.

```
dif_y = y - mu_y;
patrat_y = dif_y.^2;
suma_y = sum(patrat_y);
sigma_y = suma_y/M;
```

Pasul 11: Pentru calcularea covarianței dintre cei doi vectori, *x* și *y*, inițial se vor înmulții variabilele calculate anterior *dif_x* și *dif_y* utilizând operatorul **.***. Rezultatul va fi stocat în vectorul *dif_xy*. Rezultatul obținut este: dif_xy = -1.6667 -8.0000 -3.3333 -0.6667 2.0000 2.6667.

```
dif_xy = dif_x.*dif_y;
```

Pasul 12: În continuare se vor aduna elementele obținute în vectorul *dif_xy* utilizând funcția ***sum***. Rezultatul va fi stocat în variabila *suma_xy*. Rezultatul obținut: suma_xy = -9.

```
suma_xy = sum(dif_xy);
```

Pasul 13: În final, se va obține covarianța celor doi vectori, stocată în variabila *sigma_xy*, prin împărțirea variabilei *suma_xy* la *N*. Rezultatul obținut în urma calculării covariației dintre vectori: sigma_xy = -1.5.

```
sigma_xy = suma_xy/N;
```

Exercițiu 4.2: Se dau vectorii:

x=[2 3 0 9 7 8];

w=[0.2 1.5 -0.1 -0.7 0.7 0.02];

Să se calculeze valoarea lui R, aplicând formula:

$$R = \sum_{i=1}^{N} x_i w_i$$

Rezolvare:

Pasul 1: Se deschide un script nou utilizând comanda *File →New*. Se va salva într-un folder denumit Grupa [Nr. Grupei] fișierul cu denumirea Exercitiu10_2.

Pasul 2: Se vor defini cei doi vectori: x și w.

```
x=[2 3 0 9 7 8];
w=[0.2 1.5 -0.1 -0.7 0.7 0.02];
```

Pasul 3: Se vor determina dimensiunile celor doi vectori x și w utilizând funcția ***size***. Rezultatele obținute vor fi stocate în variabilele N și respectiv M. Rezultate: N = 6, M = 6.

```
N = size(x,2);
M = size(w,2);
```

Pasul 4: Se vor înmulți element cu element cei doi vectori: x și w. Rezultatul obțiut se va salva în vectorul *inmultire_xw*. Rezultat: inmultire_xw = 0.4000 4.5000 0 -6.3000 4.9000 0.1600.

```
inmultire_xw = x.*w;
```

Pasul 5: În final, se va salva în variabila R rezultatul final prin însumarea elementelor din vectorul *inmultire_xw* utilizând funcția ***sum***. Rezultatul obținut: R = 3.66.

```
R = sum(inmultire_xw);
```

Pasul 6: Se va calcula valoarea lui R utilizând înmulțirea matriceala și se va salva în variabila *R_m* rezultatul obținut. Înmulțirea matriceală se va realiza conform ecuației: **R_m = x*w'**. Rezultatul obținut: R_m = 3.66.

```
R_m = x*w';
```

Exercițiu 4.3: Se dă matricea *M*:

$$M = \begin{bmatrix} 4.5 & 1 & 1 \\ 1.5 & 2 & 0 \\ 4 & 3 & 1 \\ 5 & 2.5 & 1 \\ 4 & 2 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 2 & 1 & 0 \end{bmatrix}$$

Unde:

- prima coloană reprezintă punctele de cordonate *x*
- a doua coloană reprezintă punctele de coordonate *y*
- iar a treia coloană reprezintă o etichetă a fiecarui punct de coordonate [x;y]

Exemplu: Primul punct (corespondent primei linii din matrice) prezintă coordonatele [4.5;1] și are eticheta 1.

Se dorește reprezentarea pe un grafic a tuturor punctelor din matrice, astfel încât punctele ce prezintă eticheta 0 să fie reprezentate cu albastru, iar cele ce prezintă eticheta 1 să fie reprezentate cu roșu.

Rezolvare

Pasul 1: Se deschide un script nou utilizând comanda *File →New*. Se va salva într-un folder denumit Grupa [Nr. Grupei] fișierul cu denumirea Exercitiu10_3.

Pasul 2: Se va defini matricea *M*.

```
M = [4.5 1 1;
1.5 2 0;
4. 3 1;
5 2.5 1;
4 2 1;
0 1 0;
1 0 0;
2 1 0];
```

Pasul 3: Se vor determina dimensiunile matricei utilizând funcția size. Rezultatele obținute vor fi stocate în variabilele *N* și respectiv *P*. Rezultate: N = 8, P = 3.

```
[N, P] = size(M);
```

Pasul 4: Se va creea o submatrice cu elementele ce prezintă eticheta 1, denumită *E1* și o submatrice ce prezintă eticheta 0, denumită *E0*, utilizând instrucțiunea ***if*** în felul următor:

1. Se vor inițializa cele două matrice, *E1*și *E0*, astfel: E1 = [ ], E0 = [ ].
2. Se vor parcurge liniile matricei cu ajutorul instrucțiunii ***for***.
3. Dacă elementul de pe ultima coloană a liniei respective prezintă eticheta 1 (operatorul utilizat pentru verificarea egalității este ==), atunci se vor salva în matricea *E1* elementele din *M* de pe linia parcursă și coloanele 1 și 2. Salvarea elementelor se va realiza prin concatenarea matricei anterioare *E1* cu linia curentă.
4. Altfel, se vor salva în matricea *E0* elementele din *M* de pe linia parcursă și coloanele 1 și 2. Salvarea elementelor se va realiza prin concatenarea matricei anterioare *E0* cu linia curentă.

```
E1 = [];
E0 = [];
for i=1:N
    if M(i,P) == 1
        E1 = [E1; M(i,1:2)];
    else
        E0 = [E0; M(i,1:2)];
    end
end
```

Rezultate obținute: E1 = 4.5000 1.0000
4.0000 3.0000
5.0000 2.5000
4.0000 2.0000
E0 = 1.5000 2.0000
0 1.0000
1.0000 0
2.0000 1.0000

Pasul 5: În continuare, se va salva în vectorul *x0* elementele de pe coloana 1 a matricei *E0* și în vectorul *y0* elementele de pe coloana 2 a matricei *E0*.

```
x0 = E0(:,1);
y0 = E0(:,2);
```

Rezultate obținute: x0 = 1.5000 0 1.0000 2.0000
y0 = 2.0000 1.0000 0 1.0000.

Pasul 6: Se va salva în vectorul *x1* elementele de pe coloana 1 a matricei *E1* și în vectorul *y1* elementele de pe coloana 2 a matricei *E1*.

```
x1 = E1(:,1);
y1 = E1(:,2);
```

Rezultate obținute: x1 = 4.5000 4.0000 5.0000 4.0000
y1 = 1.0000 3.0000 2.5000 2.0000

Pasul 7: Se vor reprezenta cu ajutorul funcției plot elementele vectorului *y0* în funcție de elementele vectorului *x0* cu markerul 'x' și culoarea albastra, 'b'.

```
figure, plot(x0,y0,'x')
```

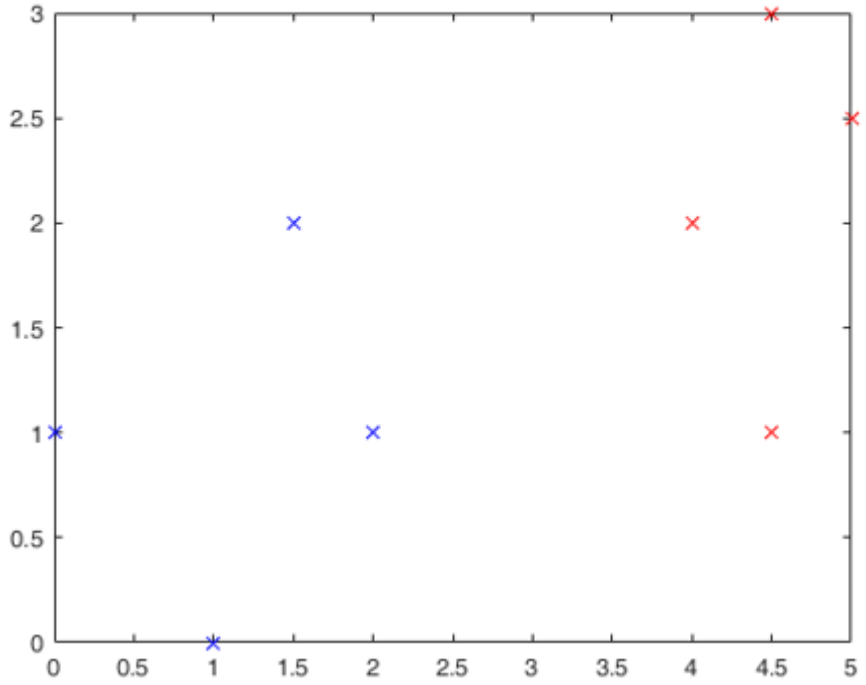
Pasul 8: Se va reține plotul anterior utilizând funcția ***hold***.

```
hold on
```

Pasul 8: Se vor reprezenta cu ajutorul funcției plot elementele vectorului *y1* în funcție de elementele vectorului *x1* cu markerul 'x' și culoarea roșie, 'r'.

```
plot(x1,y1,'xr')
```

Rezultat final:



Exercițiu 4.4: Să se realizeze o funcție permutare_linii care permută două linii (schimbă cele două linii între ele) dintr-o matrice. Funcția va primi ca parametrii de intrare matricea și cele două linii care urmează să fie permutate și va avea ca parametru de ieșire matricea permutată.

Exemplu:

$$M = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \\ 13 & 14 & 15 \end{bmatrix}$$

Rezultatul permutării liniei 2 cu linia 4 este:

$$M\_perm = \begin{bmatrix} 1 & 2 & 3 \\ 10 & 11 & 12 \\ 7 & 8 & 9 \\ 4 & 5 & 6 \\ 13 & 14 & 15 \end{bmatrix}$$

Rezolvare:

I. Crearea funcției

Pasul 1: Se deschide o nouă funcție utilizând comanda *File →New Function.* Se va salva într-un folder denumit Grupa [Nr. Grupei] fișierul cu denumirea permutare_linii.

Pasul 2: Se va modifica fișierul deschis astfel: în locul numelui implicit Unitiled, se va scrie numele funcției: *permutare_linii*. În locul argumentelor de intrare, input_args, se va scrie numele matricii, *M* și variabilele *i* și *j*, corespunzătoare celor două linii care vor fi permutate. În locul argumentelor de ieșire, output_args, se va scrie denumirea matricei rezultate în urma permutării: *M*.

```
function [ M ] = permutare_linii( M, i, j )

end
```

Pasul 3: Se va salva într-un vector, *V_i*, linia *i* a matricei *M*.

```
V_i = M(i,:);
```

Pasul 4: Se va salva într-un vector, *V_j*, linia *j* a matricei *M*.

```
V_j = M(j,:);
```

Pasul 5: Se va înlocui linia *i* a matricei *M* cu vectorul *V_j*.

```
M(i,:) = V_j;
```

Pasul 6: Se va înlocui linia *j* a matricei *M* cu vectorul *V_i*.

```
M(j,:) = V_i;
```

Pasul 7: Funcția va arăta în final:

```
function [ M ] = permutare_linii( M, i, j )

    V_i = M(i,:);
    V_j = M(j,:);
    M(i,:) = V_j;
    M(j,:) = V_i;

end
```

II. Apelarea funcției.

Pasul 1: Se deschide un script nou utilizând comanda *File →New*. Se va salva într-un folder denumit Grupa [Nr. Grupei] fișierul cu denumirea Exercitiu10_4.

Pasul 2: Se va defini matricea *M* din exemplu.

```
M = [1 2 3; 4 5 6; 7 8 9; 10 11 12; 13 14 15];
```

Pasul 3: Se va apela funcția astfel:

```
[ M_perm] = permutare_linii( M, i, j );
```

Exercițiu 4.5: Să se realizeze o funcție cu denumirea histograma care primește ca parametru de intrare un vector și returnează histograma acestuia. Să se reprezinte grafic histograma utilizând funcția ***stem***.
Histograma reprezintă o functie ce asociaza fiecarui element posibil din vector probabilitatea sa de aparitie.

Rezolvare:
Pasul 1: Se deschide un script nou utilizând comanda *File →New*. Se va salva într-un folder denumit Grupa [Nr. Grupei] fișierul cu denumirea histograma.
Pasul 2: Se va modifica fișierul deschis astfel: în locul numelui implicit Unitiled, se va scrie numele funcției: *histograma.* În locul argumentelor de intrare, input_args, se va scrie numele vectorului de intrare, *v*. În locul argumentelor de ieșire, output_args, se va scrie denumirea matricei ce contine histograma: *h*.
Pasul 3: În corpul funcției se vor determina minimul și maximul vectorului de intrare valori ce vor fi stocate în variabilele *minim* și respectiv *maxim*.
Pasul 4: Prima linie a histogramei *h* va conține valorile de la minim la maxim cu pasul de 1.
Pasul 5: A doua linie a histogramei *h* se va inițializa cu zero utilizând funcția ***zeros***.
Pasul 5: Se parcurge vectorul de intrare, *v*, cu ajutorul instrucțiunii ***for***.
Se determină poziția elementului *i* din vector pe prima linie a histogramei *h* cu ajutorul funcției ***find*** și se salvează în variabila *idx* astfel: `idx = find(h(1,:)==v(i));`
Pe linia 2 a histogramei *h*, se incrementează cu o unitate elementul corespunzător indexului *idx* determinat anterior.
Se termină instrucțiunea ***for*** cu ajutorul cuvântului cheie *end*.

Apelarea funcției:
Pasul 1: Se deschide un script nou utilizând comanda *File →New*. Se va salva într-un folder denumit Grupa [Nr. Grupei] fișierul cu denumirea Exercitiu10_5.
Pasul 2: Se va defini vectorul *V* din exemplu.
Pasul 3: Se va apela funcția astfel:
h = histograma(v);
Pasul 4: Se reprezintă grafic histograma cu ajutorul funcției ***stem***, având pe axa x elementele de pe prima linie a matricei *h*, iar pe axa y elementele de pe cea de a doua linie a matricei *h*.