Lucrarea 1

Introducere în MATLAB

1. Considerente generale

1.1. Prezentarea mediului MATLAB

Prelucrarea numerică semnalelor este necesară în multe aplicații: industriale, telecomunicații, biomedicină etc.

Scopul acestor lucrări de laborator este învățarea metodelor de prelucrarea numerică a semnalelor unidimensionale. Instrumentul folosit este programul **MATLAB** deoarece este simplu de învățat şi utilizat, foarte răspândit în mediile universitare şi poate fi mereu actualizat. El integrează analiza numerică, calculul matriceal, procesarea semnalelor şi reprezentările grafice.

MATLAB (**MAT**rix **LAB**oratory) este un program interactiv, destinat prelucrării numerice a datelor furnizate sub formă vectorială sau matriceală.

MATLAB-ul include si aplicații specifice, numite **TOOLBOX-uri.** Acestea sunt colecții extinse de funcții MATLAB care dezvoltă mediul de programare de la o variantă la alta, pentru a rezolva probleme specifice. În cazul procesării semnalelor se va lucra mai ales cu **Signal Processing Toolbox.**

1.2. Moduri de lucru în MATLAB

După lansarea în execuție, programul MATLAB intră în *modul de comandă,* afişând prompterul >> , şi aşteaptă introducerea unei comenzi de către utilizator.

De exemplu, comanda:

```
>> v = 0 : 10
```

va crea variabila v afişând cele 11 elemente ale vectorului linie v, de la v[1]=0 la v[11]=10.

În afara modului de lucru în linie de comandă, MATLAB-ul lucrează cu programe conținute în fişiere. Fişierele ce conțin instrucțiuni MATLAB se numesc *fişiere **M*** (au extensia .**m**).

Un program MATLAB poate fi scris sub forma fişierelor *script* sau a fişierelor *function*. Un fişier *script* este un fişier extern care conține o secvență de comenzi MATLAB. După execuția completă a unui fişier *script*, variabilele create de acest tip de fişier rămân în zona de memorie a aplicației.

Dacă prima linie a fişierului conține cuvântul “function" fişierul respectiv este fişier funcție, care se caracterizează prin faptul că poate lucra cu argumente. La terminarea execuției unei funcții, în memoria calculatorului nu rămân decât variabilele de ieşire ale acesteia.

1.3. Lansarea în lucru a programului MATLAB. Ferestre de lucru

Se selectează icoana specifică programului MATLAB şi apoi MATLAB.exe. Pe monitor va apărea fereastra de comenzi ca în figura 1.

Figura 1 - Fereastra de comenzi a MATLAB-ului

Selecția unei comenzi din bara de comenzi sau meniul principal se poate face cu mouse-ul sau cu ajutorul săgeților, prin deplasarea zonelor active sau prin tastarea literei marcate în fiecare subcomandă.

Figura.2. Meniul ferestrei de comenzi

Meniul principal:

Pentru a deschide un fişier în Editorul/Debugger-ul MATLAB-ului din meniul de comandă se procedează în felul următor:

- pentru un fişier nou, se selectează **File**, apoi **New**, apoi **New M-file**.
- pentru un fişier existent se selectează **File,** apoi **Open M-file** si apoi se selectează fişierul dorit.

a) b)

Figura 3. Selectarea din meniul principal a unui fişier nou (a) sau a unuia existent (b)

Tot din meniul principal se pot seta anumite proprietăți, legate de formatul dorit a fi afişat, fonturi si opțiuni de copiere.

- se selectează **File**, apoi **Preferences** ca în figura 4

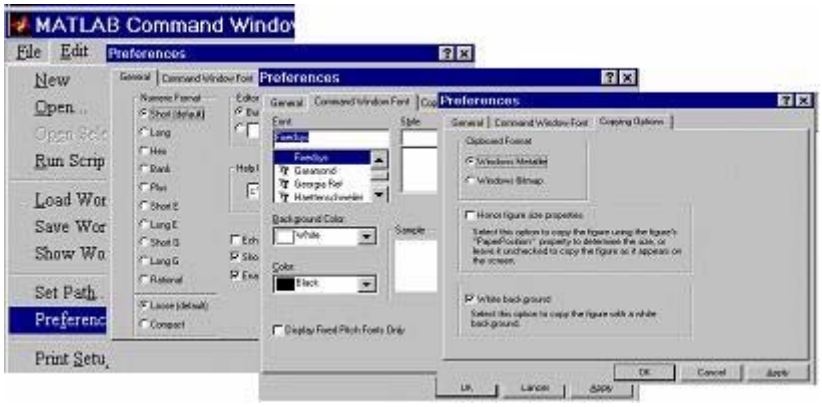

Figura 4. Selectarea din meniul principal a comenzii Preferences

MATLAB-ul lucrează cu două tipuri de ferestre: *o fereastră de comenzi* şi una de *reprezentări grafice*.

La un moment dat poate fi deschisă numai o fereastră de comenzi. Fereastra grafică este utilizată în reprezentarea grafică a datelor. Pot fi deschise mai multe ferestre grafice în acelaşi timp.

Selectarea ferestrei grafice se face în modul următor: din meniul **File** se selectează **New** apoi **Figure**. Pe monitor va apărea o fereastră grafică ca în fig. 5.

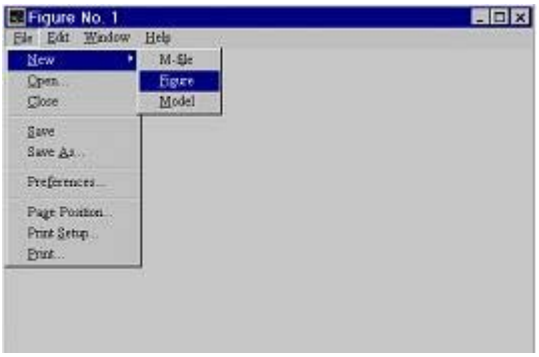

Figura 5. Subdomeniul File al ferestrei grafice

Selectarea comenzii **File** (din aceasta fereastră) urmată de **New**, **Figure** sau **New Figure** determină trecerea într-o fereastră grafică nouă.

1.4. Informaţii generale. Funcţii de control general

- Implicit, mediul de programare MATLAB este sensibil la tipul de litere (mari sau mici), dar există comenzi care fac trecerea între modurile sensibil şi nesensibil. Numele de funcţie este obligatoriu să fie scris cu litere mici.
- Liniile de comentariu dintr-un fişier script / funcţie sunt precedate de caracterul **%**.
- Pentru ajutor se tastează **help** pentru meniul întreg sau **help** urmat de denumirea funcţiei sau fişierului **.m**. (Exemplu: `help fft`**).**
- Numărul de cifre zecimale în care sunt afişate diverse variabile nu reprezintă precizia în care sunt efectuate calculele. Pentru a schimba formatul afişat se tastează **`format short e`** pentru 5 cifre zecimale, **`format long e`** pentru 15 cifre zecimale şi **`format bank`** pentru plasarea a două cifre zecimale la dreapta punctului zecimal. Acelaşi lucru se poate face din meniu, selectând **File**, **Preferences**, **General**, iar aici formatul dorit (a se revedea fig. 4).

- **`cd`** apelează directorul curent.
- **`dir`** listează fişierele din directorul curent**.**
- **`dir numedirector`** listează fişierele din directorul "numedirector".
- Comenzile **`who`** si **`whos`** furnizează numele variabilelor ce au fost definite în spaţiul de lucru MATLAB.
  - **`who`** afişează variabilele curente din memorie;
  - **`whos`** afişează variabilele, dimensiunile lor, precum si tipul acestora (reale sau complexe).
- **`what`** listeaza fişierele M, MAT si MEX din directorul curent.

1.5. Fişiere M

Fişierele M pot fi privite ca macro-uri ale comenzilor MATLAB salvate în fişiere cu extensia **.m.,** adică *numefisier***.m** . Un fişier M poate fi fie o funcţie cu variabile de intrare şi ieşire, fie o listă de comenzi.

MATLAB cere ca fişierele M să fie salvate fie în directorul de lucru sau într-un director care este specificat în lista căilor din MATLAB. Pentru a putea accesa fişierul M dintr-un anumit director, trebuie să adăugam directorul/fişierul la calea MATLAB.

Se procedează în felul următor:

- se tastează butonul **Path Browser** din butoanele de comandă

- apare fereastra
- se tastează butonul **Browse**; apare fereastra:

- se caută directorul respectiv şi se tastează **OK**, apoi **Add to path** din fereastra anterioară.

path returnează căile cu care lucrează MATLAB-ul (în care sunt căutate fişierele apelate)

Exemplul 1

```
» path
          MATLABPATH

     C:\MATLABR11\toolbox\matlab\general
     C:\MATLABR11\toolbox\matlab\ops
     C:\MATLABR11\toolbox\matlab\lang
     ......
```

addpath dirname adaugă directorul `dirname` la calea curentă din MATLAB. Această comandă este echivalentă cu procedura prezentată anterior de adăugare a unei noi căi.

Exemplul 2

Comanda `addpath c:\nume` are ca efect adăugarea noii căi, adică se vor putea apela fişierele cu extensia **.m** din directorul ce are calea `c:\nume`.

Fişierele M pot utiliza variabile definite de utilizator, variabile definite cu comanda **input.** De exemplu, să presupunem că vrem să rulăm un fişier M pentru diferite valori ale unei variabile N. Atunci, în fişierul M se utilizează următoarea comandă: `N=input('N=').`

1.6. Definirea variabilelor

Variabilelor li se atribuie valori numerice, tipărindu-se direct expresia numerică. De exemplu

```
a=1+2
```

conduce la rezultatul:

```
a=
     3
```

Rezultatul nu se afişează dacă se pune punct şi virgulă la sfârşitul expresiei, de exemplu

```
a=1+2;
```

În MATLAB se utilizează următorii operatori aritmetici:

+	adunare
-	scădere
*	multiplicare
/	împărțire
^	ridicare la putere
'	transpus

Unei variabile i se poate atribui o formulă ce utilizează operatorii menționați anterior şi una sau mai multe mărimi definite anterior chiar în cadrul comenzii curente. De exemplu, presupunând că `a` este definită anterior:

```
b=3^a
```

va returna valoarea:

```
b=
     27
```

Există variabile predefinite:

```
i=sqrt(-1)
```

```
j=sqrt(-1)
pi=3,1416...
```
etc.

Exemplul 3

```
y=2*(1+4*j)
```
conduce la
```
y=
     2.0000+8.0000i
```
Pentru definirea variabilelor există un număr de funcții predefinite, dintre care, cele mai uzuale pentru prelucrarea semnalelor, sunt prezentate în Tabelul 1.

Tabelul 1.

abs	valoarea absolută
angle	faza ataşată unui număr complex, în radiani
cos	funcția cos, cu argumentul în radiani
sin	funcția sin, cu argumentul în radiani
exp	funcția exponențială cu baza **e**

De exemplu, cu `y` din exemplul anterior, comanda:
```
c=abs(y)
```
conduce la:
```
c=
     8.2462
```
sau:
```
u=angle(y)
```
la:
```
u=
     1.3258
```
Se face observația că funcția **exp** se poate utiliza cu numere complexe, de exemplu, cu acelaşi y, comanda:
```
d=exp(y)
```
conduce la:
```
d=
     -1.0751+7.3104i
```
fapt evident, verificabil utilizând formula lui Euler

$$d = e^{y} = e^{\mathrm{Re}\, y + j\, \mathrm{Im}\, y} = e^{2}\left(\cos 8 + j \sin 8\right)$$

1.7. Definirea scalarilor, vectorilor şi matricelor

MATLAB-ul este un pachet de programe care lucrează numai cu un singur tip de obiecte, matrice numerice rectangulare, cu elemente reale sau complexe. În acest sens scalarii sunt asimilați matricelor cu o linie şi o coloană (1 × 1), iar vectorii sunt asimilați matricelor cu o linie (1 × n) sau o coloană (n × 1).

Introducerea matricelor în mediul de lucru se face prin una din metodele:

- introducerea explicită;
- generarea prin instrucțiuni şi funcții;
- crearea de fişiere M;
- încărcarea din fişiere de date externe.

Cea mai simplă metodă constă în *utilizarea unei liste explicite*. Trebuie respectate următoarele reguli:

- elementele unei linii trebuie separate prin spații libere sau virgulă;
- liniile se separă prin punct-virgulă ”;”;

- elementele matricei sunt cuprinse între paranteze drepte "[ ] ".

Elementele matricelor pot fi numere reale sau complexe sau orice altă variabilă MATLAB.

Elementele unei matrice **A** pot fi identificate, în MATLAB, prin notația `A(i,j)` şi semnifică elementul de la intersecția liniei *i* cu coloana *j*. Pentru a face referire la un element al matricei sunt necesari doi indici, iar referirea la un element al unui vector se face cu un singur indice.

Exemplul 4

Matricele sunt definite introducând elementele linie cu linie:

```
A=[1 2 3; 3 6 8]
```

Această comandă creează matricea

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 6 & 8 \end{bmatrix}$$

Comanda `A(2,1)` afişează rezultatul:

```
ans =

     3
```

Se remarcă faptul că dacă se atribuie o valoare unui element care ocupă o poziție în afara dimensiunii maxime a unei matrice sau unui vector, dimensiunea acesteia/acestuia este mărită automat până la valoarea indicelui noului element, iar elementele nedefinite sunt setate la valoarea zero.

Exemplul 5

Comanda: `A(2,5)=4`

creează matricea:

$$A = \begin{bmatrix} 1 & 2 & 3 & 0 & 0 \\ 3 & 6 & 8 & 0 & 4 \end{bmatrix}$$

Comanda `size(A)` returnează dimensiunea matricei A.

Se definesc mai multe matrice particulare:

- matricea vidă `A=[ ]`
- matricea nulă de ordin n × m : `A= zeros(n,m)`
- matricea unitară de ordin n × m : `A= ones(n,m)`
- matricea cu diagonală 1, de ordin n × n `A= eye(n)`

Vectorii pot fi definiți în două moduri:

- introducerea explicită a componentelor vectorului;

Exemplul 6

Comanda:

```
v=[1 2 3 4 5 ]
```

creează un vector cu o linie şi 5 coloane. Comanda:

```
v(6)=6
```

conduce la vectorul:

v=[1 2 3 4 5 6]

Fie vectorul $c = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$

Aşadar $c(2) = 2$.

- Al doilea mod este utilizat pentru a crea vectori cu elemente egal spațiate prin comanda:

```
k=amin:pas:amax
```

unde amin este valoarea primului element din vector, pas este valoarea incrementului, amax limita până la care pot lua valori componentele vectorului. Dacă incrementul lipseşte, atunci implicit valoarea sa este 1.

Exemplul 7

```
k=0:0.5:5.1
```

creează un vector cu elementele 0, 0.5, 1, 1.5,....5 .

Comanda `length(v)` returnează lungimea vectorului v.

1.8. Operații cu matrice

Fie X şi Y două matrice. Se definesc următoarele operații

a) adunarea şi scăderea, definite pentru cazul în care dimensiunile matricei X sunt egale cu dimensiunile matricei Y:

```
Z=X+Y
Z=X-Y
```

b) înmulțirea, definită atunci când X are dimensiunile (m × n) iar Y (n × p)

```
Z=X*Y
```

c) împărțirea la dreapta, definită atunci când X are dimensiunile (n × n) iar Y (n × n)

```
Z=X/Y
```

şi este identică cu $X*Y^{-1}$ (Y^{-1} este inversa matricii Y).

d) înmulțirea element cu element (între componentele a două matrice de aceleaşi dimensiuni)

```
Z=X.*Y
```

e) împărțirea la stânga

```
Z=X\Y
```

este identică cu $X^{-1}*Y$ (X^{-1} este inversa matricii X).

f) ridicarea la putere

```
Z=X^p
```

reprezintă ridicarea la puterea *p* a matricei X. Expresia X^p are sens numai pentru matrice pătratice şi *p* scalar.

g) ridicarea la putere a elementelor unui vector/matrice

```
Z=X.^p
```

h) ridicarea la puterea elementelor unui vector/matrice a unui scalar

```
Z=p.^X
```

i) transpunerea

```
Z=X'
```

Acest lucru face ca pentru o matrice cu dimensiunea n × m să se obțină o matrice Z cu dimensiunea m × n.

1.9. Operații cu vectori

Operațiile cu vectori se efectuează prin particularizarea regulilor de la operații cu matrice impunând ca una din dimensiuni sa fie egală 1, acolo unde dimensiunea permite acest lucru.

a) Produsul scalar a doi vectori se poate calcula în următoarele moduri:

`z= sum(a.*b)` dacă vectorii `a` şi `b` sunt amândoi fie vectori coloană fie vectori linie sau `z=(a*b')` dacă `a` şi `b` sunt vectori linie.

Observație: vectorii a şi b trebuie să fie de aceeaşi dimensiune.

1.10. Construirea unei funcții

Mediul de programare Matlab oferă posibilitatea creării unei funcții ce foloseşte funcții Matlab, funcții definite anterior, operatori predefiniți etc. Numele fişierului în care este scrisă funcția trebuie să fie identic cu numele funcției. De asemenea, prima linie a fişierului trebuie să conțină sintaxa caracteristică definiției unei funcții:

```
function „var_returnate” = „nume_functie”(„var_intrare”)
```

De exemplu, o funcție ce realizează suma a două variabile `x` şi `y`, rezultatul fiind returnat în variabila `z,` se realizează astfel:

```
function z=xplusy(x,y);
z=x+y;
```

Fişierul astfel scris se salvează sub denumirea `xplusy.m` . Funcția definită anterior poate fi apelată din linia de comenzi Matlab, dintr-un script sau dintr-o altă funcție. De exemplu:

>>a=1;b=2; c=xplusy(a,b);

1.11. Reprezentări grafice elementare

Funcțiile de bază MATLAB pentru reprezentările grafice sunt:

- **plot**
- **loglog**
- **semilogx**
- **semilogy**

Pentru reprezentarea graficelor în coordonate liniare se utilizează funcția **plot.** Funcția **plot** se apelează cu una din sintaxele:

- **plot (y)**

Dacă argumentul `y` este complex, **plot(y)** este echivalent cu:

plot(real(y), imag(y)).

Dacă `y` este un vector real (linie sau coloană), atunci funcția **plot** trasează graficul y=y(i), unde i=1,2,…L este numărul de ordine al elementului y, L fiind lungimea vectorului `y`.

- **plot (x,y)** reprezintă vectorul y funcție de vectorul x.

Dacă `x` este vector, iar `y` este matrice, atunci coloanele lui `y` sunt trasate în funcție de vectorul `x`, rezultatul fiind reprezentarea mai multor grafice în aceeaşi fereastră grafică.

Dacă x şi y sunt matrice de aceeaşi dimensiune, atunci se reprezintă coloanele lui y în funcție de coloanele lui x.

- **plot (x,y,'linie tip') 'linie tip'** fiind o succesiune de caractere (vezi Tabelul 2) care specifică tipul liniei cu care este trasat graficul.

- **plot (x1,y1,x2,y2,…)** reprezintă simultan mai multe grafice, în acelaşi sistem de coordonate şi în aceeaşi fereastră grafică.

Graficele se pot trasa utilizând linii şi markere de diferite culori. Acest lucru este prezentat în Tabelul 2.

Tabelul 2

Linii tip şi markere MATLAB			Culori –cod MATLAB	
continuă	-	+	Galben	y
întreruptă	--	*	Mov	m
două puncte	:	°	albastru-deschis	c
linie-punct	-.	x	Roşu	r
		.	Verde	g
			Albastru	b
			Alb	w
			Negru	k

1.12. Reprezentarea discretă a datelor

Reprezentarea discretă a datelor se face cu funcția **stem** sub forma unor linii terminate cu cerculeț. Se apelează cu una din sintaxele:

- **stem(y)** trasează un grafic y=y[i], i=1,2,3 ....L din linii terminate cu cerculeț
- **stem(x,y)** trasează un grafic cu linii terminate cu cerculeț, cu locațiile specificate de vectorul x. Valorile lui x trebuie să fie egal spațiate.
- **stem(x,y,'linie-tip')** este similară funcției **plot(x,y,'linie tip'),** cu deosebirea că se trasează graficul cu linii terminale cu cerculeț.

Observație: atât la funcția **plot** cât şi la funcția **stem** lungimile vectorilor de pe abscisă şi ordonată trebuie sa fie egale.

1.13. Personalizarea graficelor

Pentru plasarea în câmpul graficelor a unor texte, etichete ale axelor, precum şi a titlului se utilizează următoarele funcții:

- **title('text')** comandă prin care titlul graficului se plasează deasupra acestuia; 'text' fiind un şir de caractere care reprezintă titlul graficului;
- **xlabel('text')** precizează eticheta axei x; 'text' fiind un şir de caractere care reprezintă numele axei, unitatea de măsură etc.;
- **ylabel('text')** precizează eticheta axei y; 'text' fiind un şir de caractere care reprezintă numele axei, unitatea de măsură etc.;
- **grid on** trasează o rețea de linii orizontale şi verticale pe grafic
- **grid off** elimină rețeaua de linii orizontale şi verticale trasate pe grafic de grid on

Exemplul 8

1. Să se reprezinte grafic funcția $f(t)=\sin(2\pi 50t)$, cu culoarea neagră şi linie-punct şi $g(t)=-f(t)$ cu markere * de culoare verde. Să se scrie titlul "graficele funcțiilor f(t) si g(t)", pe axa x să se scrie „t", iar pe axa y să se scrie "f(t) şi g(t)".

```
t=0:.001:0.02
f=sin(2*pi*50*t)
g=-f
plot(t,f,'-.k',t,g,'*g'),grid on
title('Graficele functiilor f(t) si g(t)')
xlabel('t'), ylabel('f(t) si g(t)')
```

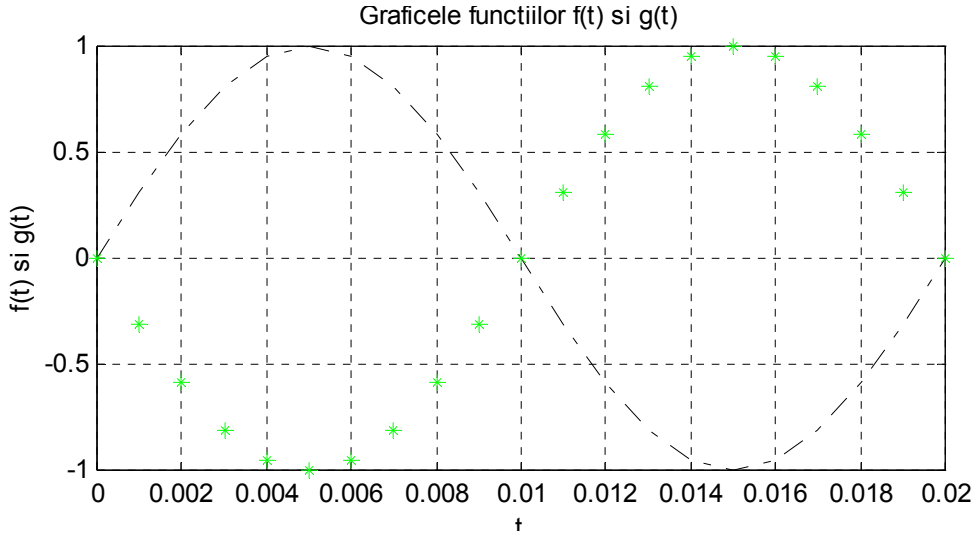

Figura 6. Graficul funcțiilor f(t) şi g(t)

2. Aplicații propuse

1. Operarea cu matrice, vectori şi scalari

a) Se dau matricele:

$$A = [2 \quad 3 \quad 4; \quad 5 \quad 2 \quad 9; \quad 16 \quad 0 \quad 0]$$

$$B = [1 \quad 2 \quad 3; \quad 1 \quad 1 \quad 1; \quad 2 \quad 3 \quad 2]$$

şi scalarul p=2. Să se calculeze în MATLAB:

1.	$C1 = A + B$
2.	$C2 = A - B$
3.	$C3 = A + p$
4.	$C4 = A * B$
5.	$C5 = A * p$
6.	$D = A'$
7.	$E = B'$
8.	$Z = A / B$
9.	$W = A \setminus B$
10.	$Y = A \hat{} p$

Să se verifice dacă $Z = A * B^{-1}$ şi $W = A^{-1} * B$.

b) Să se genereze un vector cu pas liniar, cu limitele: amin = 2, amax = 10, pas = 2.

c) Să se genereze un vector cu pas liniar, cu limitele: amin = 2, amax = 10, N = 5, unde N este numărul de elemente.

d) Să se genereze un vector cu N=5 elemente distribuite logaritmic între decadele 10^{-2} si 10^{2}.

e) Fie un vector `y=1:0.11:123`. Să se calculeze lungimea vectorului `y` şi să se genereze un vector cu toate elementele 1 de lungime egală cu lungimea vectorului `y`.

f) Să se efectueze produsul scalar al vectorilor:

$$a = [2 \quad 3]$$
$$b = [-4 \quad 4]$$

g) Să se calculeze produsul element cu element al matricilor:

$$A = [1 \quad 2 \quad 3; \quad 4 \quad 5 \quad 6; \quad 7 \quad 8 \quad 9]$$
$$B = [0 \quad 1 \quad 0; \quad 0 \quad 1 \quad 0; \quad 0 \quad 1 \quad 0]$$

2. Reprezentarea grafică

Să se reprezinte grafic funcţia discretă:

$$x(n) = \sin\left(2\pi \frac{1}{10} n\right) \quad pentru\ n = \overline{0,20}$$

Graficul să fie de culoare roşie, reprezentat cu steluţe (*) (a se face apel la comanda `help stem`) . Să se scrie titlul şi identificările axelor.

3. Realizarea unui script

Să se scrie un program în MATLAB care să calculeze produsul scalar al vectorilor

$$\vec{a} = 3\vec{i} - 4\vec{j}$$
$$\vec{b} = \vec{i} + 2\vec{j} - 2\vec{k}$$

4. Realizarea unei funcţii

Să se creeze o funcţie, denumită **yprodx.m,** prin care se face produsul scalar a doi vectori linie x şi y, de aceeaşi lungime.