Introducere în MATLAB
pentru prelucrarea semnalelor

1.1 Instalarea și documentație

MATLAB este un limbaj de programare de nivel înalt folosit pentru diferite calcule numerice, ce include atât o interfață de tip linie de comandă cât și o interfață de tip fișier de scripturi. Acesta este folosit optim atunci când se folosește calculul matriceal pentru a rezolva diferite probleme. De asemenea, este folosit cu precădere pentru analiza numerică, procesarea semnalelor sau reprezentări grafice în domeniul științelor inginerești.

MATLAB este un produs software proprietar. Pentru a-l instala vizitați adresa: https://www.mathworks.com/help/install/install-products.html și alegeți tipul de produs MATLAB pe care vreți să îl instalați. Fiind un produs proprietar veți avea nevoie de o licență pe care fie o cumpărați, fie vă este oferită de către facultate/universitate.

Documentația pentru a programa cu ajutorul MATLAB se găsește la adresa https://www.mathworks.com/help/MATLAB/.

1.2 Scurtă introducere în prelucrările matematice

Există două tipuri de programe în MATLAB:

- scripturi, care constau dintr-o secvență de comenzi ce utilizează funcții deja definite în MATLAB;
- funcții implicite sau definite de utilizator. Acestea au un număr specific de parametri de intrare și includ prelucrările asferente acestor parametri, rezultatele fiind stocate în parametri de ieșire.

La pornirea MATLAB se poate vizualiza calea către directorul curent (de lucru), împreună cu următoarele 4 ferestre (Figura 1):

- fereastra de comandă/consola (Command Window), unde se pot rula diferitele instrucțiuni;
- spațiul de lucru (Workspace), unde se pot vizualiza variabilele;
- folderul curent (Current Folder), unde se salvează eventualele fișiere;
- editorul de script-uri (Editor), unde se pot crea fișiere de tip script în directorul de lucru. Cu ajutorul acestora se pot rula o serie de comenzi succesive (apăsând F5) sau linii selectate (apăsând F9).

Dacă descrierea de mai sus nu coincide cu ceea ce găsiți când deschideți MATLAB, se poate proceda așa cum este indicat in Figura 2, schimbând layout-ul în modul default.

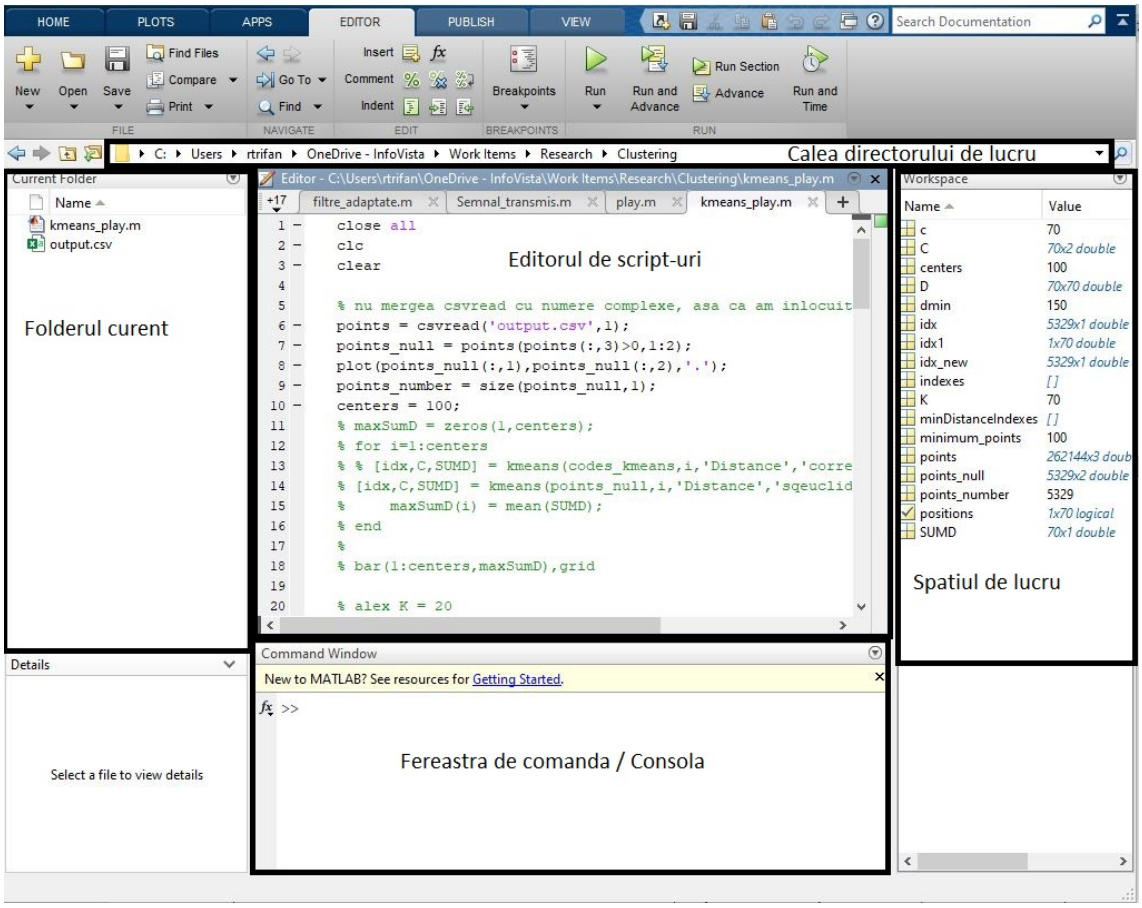


Fig. 1. Interfața MATLAB

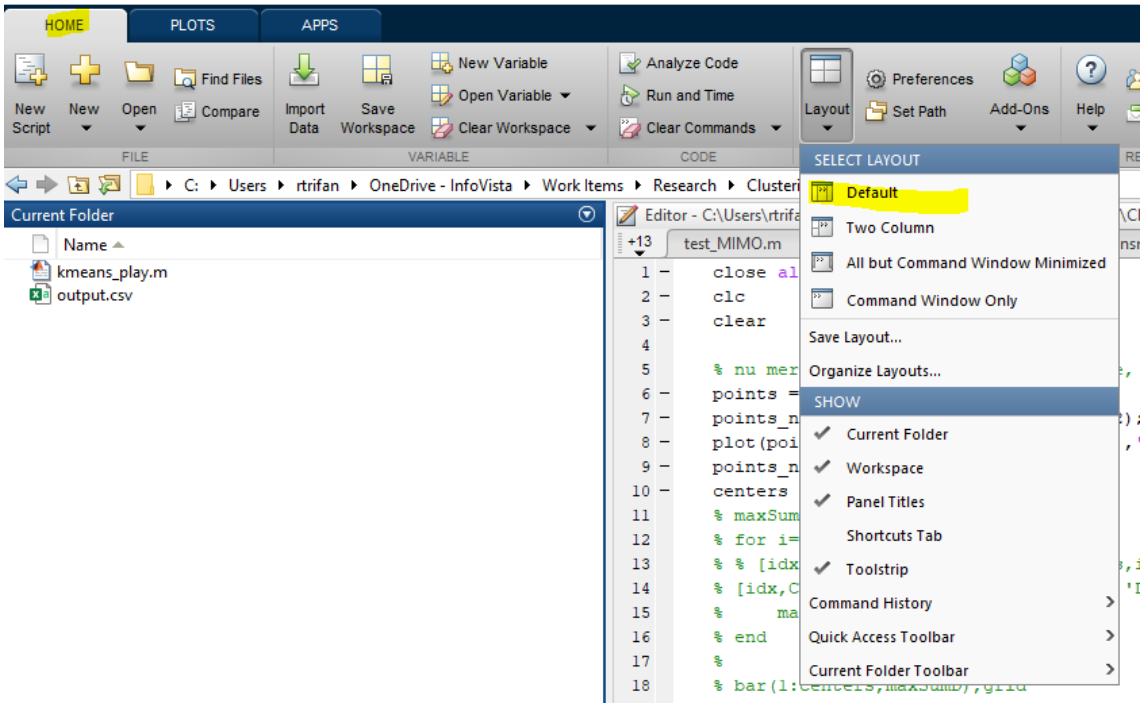


Fig. 2. Default layout

1.3 Funcții de bază în MATLAB

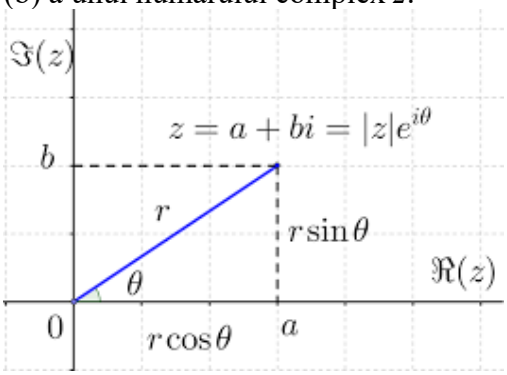
În Tabelul 1.1 sunt enumerate principalele funcții și parametri disponibili în MATLAB, împreună descrierea acestora și câteva exemple.

Tabelul 1.1 funcțiile de bază MATLAB

Funcție/Parametru	**Descriere și exemple**
`help nume_funcție`	Funcție folosita pentru descrierea diferitelor funcții folosite
`ans`	Numele implicit al rezultatului în workspace și command prompt
`pi`	Valoarea constantei pi
`Inf,` `NaN`	Infinit Eroare de tip nedeterminare - Not a Number

`+, -, *, /, ^,`	Adunare, scădere, înmulțire, împărțire, ridicare la putere unor numere/vectori/matrice
	Ex:
	`A = [1 2 3], B = [4 5 6]'`
	`A*B = 32`

`<, <=, >, >=, ==, ~=`	Mai mic, mai mic sau egal, mai mare, mai mare sau egal, identic, diferit
`&, | , ~`	Operatori logici ȘI, SAU, NU
`i, j, 1i, 1j`	$i = j = \sqrt{-1}$

`abs(z)`
`angle(z)`
`real(z)`
`imag(z)`
	Modulul (r), faza (θ), partea reala (a) si partea imaginara (b) a unui numărului complex *z*:
	 
	
	Ex:
	`z1 = 7;`
	`z2 = -7;`
	`z3 = 1i * 7;`
	`z4 = -1i * 7;`
	
	`real(z1), real(z2), real(z3), real(z4)`
	`imag(z1), imag(z2), imag(z3), imag(z4)`
	`abs(z1), abs(z2), abs(z3), abs(z4)`
	`angle(z1),angle(z2), angle(z3), angle(z4)`
	`% radiani`

`cos(x), sin(x), tan(x)`	Funcțiile cosinus, sinus și tangentă, unde x este unghiul definit în radiani

`acos(x), asin(x), atan(x)`	Inversul funcțiilor cosinus, sinus și tangentă, în radiani
`deg2rad(x), rad2deg(x)`	Conversia unghiului x din grade în radiani, respectiv din radiani în grade
`exp(x), log(x), log10(x)`	Funcția exponențială, logaritm natural și logaritmică în baza 10
`ceil(x), floor(x), round(x)`	Partea întreagă superioară, partea întreagă inferioară și rotunjirea la cel mai apropiat întreg a numărului real *x* Ex: `floor(2.6) = 2, ceil (2.6) = 3, round (2.6) = 3`

`A.*B`	Înmulțire element cu element a 2 matrice/vectori A și B
	Ex:
	`A = [1 2 3], B = [4 5 6]`
	`A.*B = [4 10 18]`
	
	`A = [7 8 9]; % vector linie`
	`B = [10; 20; 30]; % vector coloana`
	`z2 = A .* A`
	`z3 = A .* B`
	`z4 = B .* B`
	
	`A = [1 2 3; 4 5 6] % matrice 2 x 3`
	`B = [10, 20, 30; 40, 50, 60]% matrice 2x3`
	`A.*B`

`A./B`	Împărțire element cu element a 2 matrice/vectori A și B
`A.^B`	Ridicare la putere element cu element a 2 matrice/vectori A și B

`A'`	`A =`
	`7.0000 + 9.0000i  10.0000 + 5.0000i`
	`7.0000 + 7.0000i  2.0000 +10.0000i`
	`2.0000 + 2.0000i  1.0000 + 2.0000i`
	`2.0000 + 4.0000i  6.0000 + 9.0000i`
	
	Transpusa și complex conjugata unei matrice **A**
	`>> A'`
	`ans =`
	`7.0000 - 9.0000i  7.0000 - 7.0000i  2.0000 - 2.0000i  2.0000`
	`10.0000 - 5.0000i  2.0000 -10.0000i  1.0000 - 2.0000i  6.0000`

`A.'`	Transpusa unei matrice **A**
	`>> A.'`
	`ans =`
	`7.0000 + 9.0000i  7.0000 + 7.0000i  2.0000 + 2.0000i  2.0000 +`
	`10.0000 + 5.0000i  2.0000 +10.0000i  1.0000 + 2.0000i  6.0000 +`

`conj(A)`	Complex conjugata unei matrice **A**
	`>> conj(A)`
	`ans =`
	`7.0000 - 9.0000i  10.0000 - 5.0000i`
	`7.0000 - 7.0000i  2.0000 -10.0000i`
	`2.0000 - 2.0000i  1.0000 - 2.0000i`
	`2.0000 - 4.0000i  6.0000 - 9.0000i`

x=val_init:pas:val_fin	Generarea unui vector **x** cu primul element = `val_init` , ultimul element = `val_fin` și celelalte elemente generate cu pasul `pas`
	`>> i = 1:2:5`
	`i =`
	`1 3 5`

x= linspace(val_init,val_fin,n)	Generarea unui vector **x** cu `n` elemente generate uniform începând cu elementul `val_init` până la elementul `val_fin`
	`>> i = linspace(1,5,3)`
	`i =`
	`1 3 5`

A=[]	O matrice goală

A=[x1;x2]	O matrice cu liniile date de vectorii **x1** și **x2**
	`>> x1 = [1 3 5];`
	`>> x2 = [5 6 7];`
	`>> X = [x1;x2]`
	`X =`
	`1 3 5`
	`5 6 7`

A=[x1c, x2c]	O matrice cu liniile date de vectorii **x1c** și **x2c** (prin concatenare)
	`x1c =` `1 3 5`
	`x2c =` `5 6 7`
	`>> X = [x1c,x2c]`
	`X =`
	`1 5`
	`3 6`
	`5 7`

ones(N,M)
zeros(N,M)
eye (N,M)
	O matrice de unu cu N lunii și M coloane.
	O matrice de zero cu N lunii și M coloane
	O matrice cu N linii și M coloane cu toate elementele de pe diagonala principală unu și restul 0
	
	Ex: `eye(4) = eye(4,4)`
	`>> eye(4)`
	`ans =`
	`1 0 0 0`
	`0 1 0 0`
	`0 0 1 0`
	`0 0 0 1`

rand(N,M)	O matrice de numere aleatoare uniform distribuite in intervalul (0,1) cu `N` linii și `M` coloane

`randn(N,M)`
 	
`randi([min,max],N,M)`
	O matrice de numere aleatoare distribuite Gaussian (normal) de medie 0 si deviație standard 1, cu `N` linii și `M` coloane
	O matrice de numere întregi uniform distribuite in intervalul (`min`, `max`) cu `N` linii și `M` coloane

`A(i,j)`	Elementul unei matrice **A** de pe linia i și coloana j
	`>> A = randi(5,5)`
	
	`A =`
	
	`1 5 1 3 3`
	`5 4 2 1 2`
	`3 2 1 5 5`
	`3 3 1 5 2`
	`1 3 2 3 1`
	`>> A(1,3)`
	`ans =`
	`1`

`A(i,:)`	Linia i a matricei **A**
	`>> A(2,:)`
	`ans =`
	`5 4 2 1 2`

`A(i:j,:)`	Liniile de la i la j ale matricei **A**
	`>> A(2:3,:)`
	`ans =`
	`5 4 2 1 2`
	`3 2 1 5 5`

`A(i:k:j,:)`	Liniile i, i+k, i+2k,…,j ale matricei **A**
	`>> A(1:2:5,:)`
	`ans =`
	`1 5 1 3 3`
	`3 2 1 5 5`
	`1 3 2 3 1`

`A([i,j,k],:)`	Doar liniile i,j,k ale matricei **A**
	`>> A([1,3,5],:)`
	`ans =`
	`1 5 1 3 3`
	`3 2 1 5 5`
	`1 3 2 3 1`

`A(:,j)`	Coloana j a matricei **A**
	`>> A(:,2)`
	`ans =`
	`5`
	`4`
	`2`
	`3`
	`3`

`A(:,i:j)`	Coloanele de la i la j ale matricei **A**

	`>> A(:,2:3)`
	`ans =`
	`5 1`
	`4 2`
	`2 1`
	`3 1`
	`3 2`

A(:,i:k:j)	Coloanele i, i+k, i+2k,…,j ale matricei **A**
	`>> A(:,1:2:5)`
	`ans =`
	`1 1 3`
	`5 2 2`
	`3 1 5`
	`3 1 2`
	`1 2 1`

A(:,[i,j,k])	Doar coloanele i,j,k ale matricei **A**
	`>> A(:,[1,3,5])`
	`ans =`
	`1 1 3`
	`5 2 2`
	`3 1 5`
	`3 1 2`
	`1 2 1`

size(**A**)	Numărul de linii și de coloane al matricei **A**
	`A = []`
	`size(A)`
	`ans =`
	`0 0`
	`A = 72`
	`size(A)`
	`ans =`
	`1 1`
	`A = randn(3,6)`
	`size(A)`
	`ans =`
	`3 6`

length(**x**)	Numărul de linii/coloane al vectorului **x**
	`A = []`
	`length(A)`
	`ans =`
	`0`
	`A = 72`
	`length(A)`
	`ans =`
	`1`
	`A = randn(3,6)`
	`length(A) % cea mai mare dimensiune`
	`ans =`
	`6`

mean(**x**), sum(**x**), min(**x**), max(**x**)	Media, suma, minimum și maximul elementelor unui vector **x**
find(**x**) sau find(**x**~=0) find(condiție)	Returnează pozițiile din vectorul x care sunt diferite de 0 sau care îndeplinesc o anumita condiție:

	find(x) sau find(x~=0)
	`>> find(x)`
	`ans =`
	`2 3 4 5 6`
	find(x>5)
	`>> find(x>5)`
	`ans =`
	`4 5 6`

x(find(x)) sau **x(x~=0)**
find(condiție)
	Returnează elementele din vectorul x care sunt diferite de 0 sau care îndeplinesc o anumita condiție:
	x(find(x)) sau x(x~=0)
	`>> x(find(x))`
	`ans =`
	`2 4 6 8 10`
	`>> x(x~=0)`
	`ans =`
	`2 4 6 8 10`
	`>> x(x>5)`
	`ans =`
	`6 8 10`

for i=val_init:pas:val_fin
Instrucțiuni
end
	Bucla For
	`x = [7 8 9]; % linie`
	`y = [10; 20; 30]; % coloana`
	`r = 0; % inmultire cu acumulare`
	`for index=1:length(y)`
	`r = r + x(index)*y(index);`
	`end`
	`r % 500`

if condiție instrucțiuni else/elseif instrucțiuni end	Secvență de cauzalitate if/ else/elseif
While (condiție) Instrucțiuni end	Bucla While
plot(**x,y**) stem(**x,y**)	Grafic continuu (interpolat liniar) generat de punctele date de vectorii **x** și **y** Grafic discret (segmente) generat de punctele date de vectorii **x** și **y**
Figure	Generarea unei figuri pentru a afișa un grafic
grid	Peste figură se afișează o grilă
xlabel('text'), ylabel('text') title('text')	Text pentru axa Ox, respectiv Oy pentru o figură Titlu pentru o figură
clear, clc, clf, close all	Ștergerea variabilelor din memorie, ștergerea istoricului liniei de comanda, închiderea figurii curente, închiderea tuturor figurilor

Exerciții

Se creează un director de lucru (pe Desktop). Se copiază calea acestui director în MATLAB. Apoi se creează un nou fișier de script care se salvează in directorul curent.

Să se genereze o matrice pătratică A cu 10 linii și 10 coloane, ce conține valori întregi uniform distribuite in intervalul 1 : 20.

1. Să se creeze matrice pătratice noi, fiecare cu 10 linii și 10 coloane, pornind de la matricea A, astfel încât să conțină:
   a. doar elementele pare;
   b. doar elementele de pe diagonala principală.
2. Să se construiască un vector de numere complexe care are partea reală formată din linia 2 a matricei A, iar partea imaginară formată din coloana 4 a matricei A. Să se determine:
   a. vectorul care conține modulele elementelor vectorului de numere complexe și vectorul care conține fazele elementelor vectorului de numere complexe;
   b. suma produsului element cu element al vectorului de numerele complexe cu conjugatul său (*încercați obținerea sumei dorite fără bucla FOR*).
3. Funcțiile `min, max, sum` și `mean` pot lucra atât pe vectori cat și pe matrice. Când sunt apelate pe matrice, ele pot fi aplicate fie pe rânduri, fie pe coloanele acesteia. De asemenea, funcțiile `min` și `max` pot returna și indexul valorii minime sau maxime.
   a. să determine valoarea maxima a fiecărei coloane din matricea A;
   b. să se normeze fiecare coloana a matricei A la valorile maxime calculate la punctul 3.a.;
   c. să se determine valoarea maximă globală și să se găsească poziția sa în matricea A.

Exemple

1. Având două numere complexe, z_1 și z_2:
   a. afișați partea reală și partea imaginară ale z_1 și z_2;
   b. calculați numărul real a, reprezentat de partea reală a sumei dintre z_1 și conjugatul lui z_2;
   c. calculați unghiul reprezentat de suma unghiurilor corespunzătoare lui z_1 și z_2 în planul complex și convertiți valoarea în grade;
   d. calculați logaritmul natural al numărului a.

Exemplu de folosire a instrucțiunilor (rezolvarea exercițiului):

```
clear all; clf
z1=3+j*5;
z2=-9+3*j;
z1real=real(z1);
z1imaginar=imag(z1);
a=real(z1+conj(z2));
```

```
SumUnghiRadiani=angle(z1)+angle(z2);
SumUnghiGrade=SumUnghiRadiani*180/pi;
LogNatural=log(a);
```

1.4 Generarea semnalelor în MATLAB

În MATLAB un semnal este reprezentat printr-un vector (pentru semnale unidimensionale), o matrice (pentru semnale bidimensionale) sau o secvență de matrice (în cazul în care este nevoie de 3 sau mai multe dimensiuni). Aceste structuri de date conțin valorile eșantioanelor semnalelor continue, care pot depinde fie de timp, fie de două variabile spațiale (de exemplu o imagine).

1.4.1 Semnale armonice

Semnalul sinusoidal este definit în felul următor:

$$x(t) = sin(\omega_0 t),$$

unde $\omega_0 = 2\pi f_0$ este pulsația semnalului sinusoidal,

f_0 este frecvența semnaului sinusoidal,

t este variabila de tip timp a funcției sinus.

Pentru a genera și vizualiza un semnal sinusoidal de frecvență 10Hz se poate utiliza codul de mai jos:

```
close all
clc
clear

% frecventa in Hz
f0 = 10;
omega0= 2*pi*f0;
pas = 0.001;
% Tmax = 500 ms
T_max= 500*10^-3;
% generarea vectorului de timp
t = 0:pas:T_max;
% generarea semnalului in functie de timp
x = sin(omega0*t);

figure (1), plot (t, x), grid;
xlabel('Timp(s)');
ylabel('Amplitudine');
title('x = sin (omega0*t)');
```

Se poate observa că au fost definite întâi momentele de timp pentru care s-a făcut evaluarea funcției sinus, în felul următor:

```
t = 0:pas:T_max;
```

Este important ca valoarea pasului să fie suficient de mică pentru ca afișarea să se realizeze corect. Astfel, am considerat pasul ca fiind 0,001.

După executarea codului se obține imaginea din Figura 3:

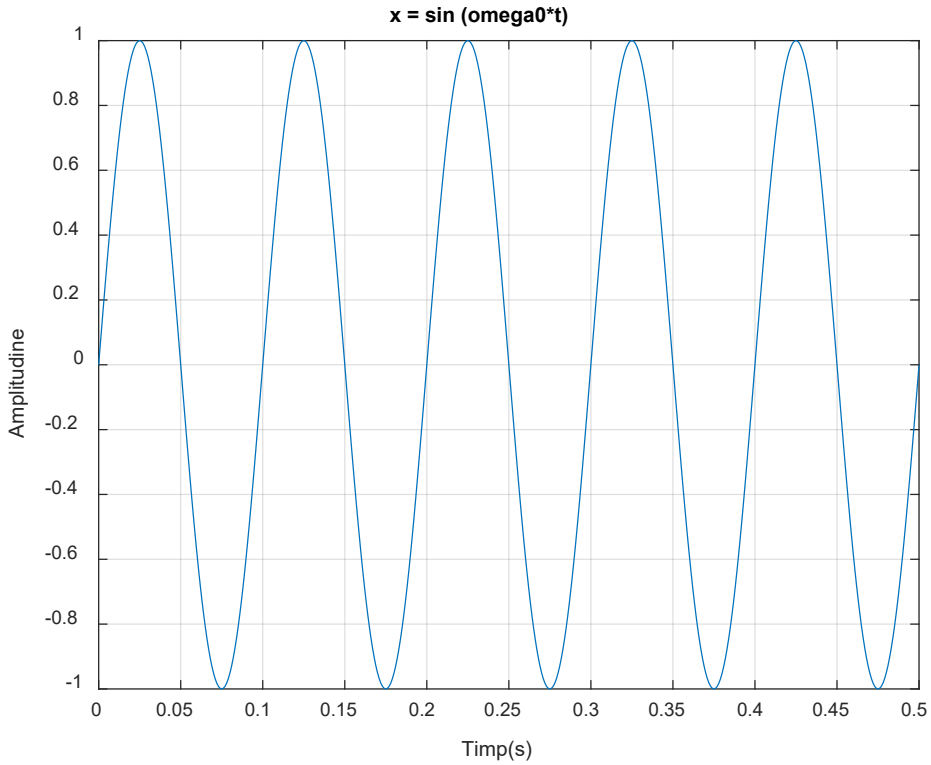


Fig. 3. Semnal sinusoidal

Exerciții:

1. Ce se întâmplă cu precizia afișării când valoarea pasului devine 0.01?
2. Ce se întâmplă cu precizia afișării când valoarea frecvenței devine 100Hz?

Hint: F0< FNyquist = Fs/2
Ts = pas = 1/Fs

3. Reprezentați semnalul armonic anterior marcând valorile intermediare cu „o”. Ce operație execută funcția plot?

```
figure(1), plot(t,x,'o-'), grid
```

4. Reprezentați următoarele semnale:

$$x_1[n] = \left(\frac{1}{2}\right)^n - \left(-\frac{1}{2}\right)^n, pentru\ 0 \le n \le 10$$

$$x_2[n] = \ln\left|\cos\left(\frac{n\pi}{15}\right) - \sin\left(\frac{n\pi}{15}\right)\right|, pentru - 20 \le n \le 20$$

$$x_3[n] = (-1)^n \cos\left(\frac{n\pi}{15}\right), pentru\ 0 \le n \le 10$$

1.4.2 Semnale dreptunghiulare

Semnalele dreptunghiulare periodice se pot crea prin două metode:

- cu ajutorul funcției ***square*** pentru crearea directă a semnalului periodic;
- cu ajutorul funcției ***rectpuls*** pentru crearea unui singur dreptunghi, care este apoi este periodizat prin concatenare.

Pentru a genera și vizualiza semnale dreptunghiulare create prin cele două metode menționate anterior se poate utiliza codul de mai jos:

```
clc
clear all
close all

pas = 0.0001;
tmin=-5;
tmax=5;
t1 = tmin:pas:tmax;
frecv=1;
x1 = square(2*pi*frecv*t1);

t2max=0.5;
t2min=-0.5;
t2p = t2min:pas:t2max-pas;
x2p = rectpuls(t2p, 0.5);

nr_perioade=10;
x2=[];
for i=1:nr_perioade
    x2=[x2,x2p];
end
t2 = nr_perioade*t2min:pas:nr_perioade*t2max-pas;
x3 = x2 *2 - 1;
size(x2p)
size(x2)
size(t2)

figure
subplot(4,1,1), plot(t2p,x2p)
xlabel('Timp [s]'), ylabel('Nivel semnal'), grid
subplot(4,1,2), plot(t1,x1)
xlabel('Timp [s]'), ylabel('Nivel semnal'), grid
subplot(4,1,3), plot(t2,x2)
xlabel('Timp [s]'), ylabel('Nivel semnal'), grid
subplot(4,1,4), plot(t2,x3)
xlabel('Timp [s]'), ylabel('Nivel semnal'), grid
```

După executarea codului se obține imaginea din Figura 4:

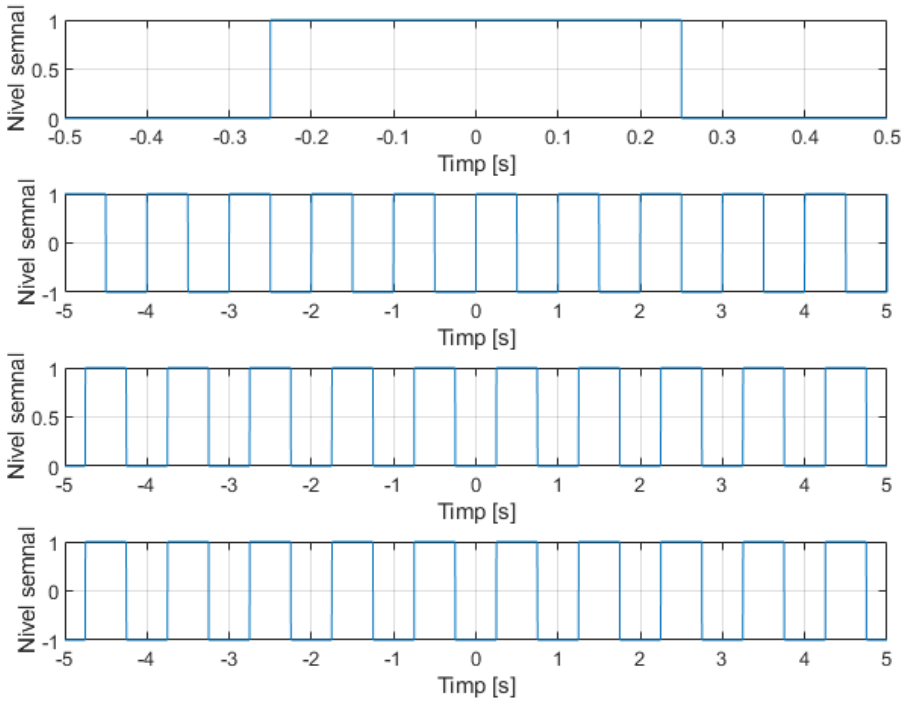


Fig. 4 Semnale dreptunghiulare

1.4.3 Semnale triunghiulare

Repetati exercițiul anterior pentru a genera și vizualiza semnale triunghiulare folosind funcțiile ***tripuls***, respectiv ***sawtooth(t,0.5)***.

1.5 Definirea de funcții în MATLAB

Funcțiile în MATLAB se definesc folosind sintaxa:

```
function [y1,y2,…yn]=nume_funcție(x1,x2,…,xn)
```

unde x1,x2,…,xn sunt parametrii de intrare,

[y1,y2,…,yn] este vectorul parametrilor care rezultă în urma prelucrării parametrilor de intrare.

De exemplu instrucțiunea `[M,N] = size(X)` ia ca parametru de intrare o matrice `X` și, printr-o serie de prelucrări efectuate în funcția `size()`, returnează un vector cu 2 elemente:

- primul element: numărul de linii M;
- cel de-al doilea element: numărul de coloane N.

Funcțiile sunt esențiale pentru modularizarea programelor complexe. Prin utilizarea funcțiilor, puteți transforma un script cu un număr mare de instrucțiuni într-un program mai organizat, alcătuit din mai multe funcții definite în fișiere separate. Această abordare de organizare a codului vă permite să reutilizați

porțiuni de cod, contribuind astfel la o dezvoltare mai eficientă și mai ușor de întreținut.

Atunci când vă creați propriile funcții în MATLAB, puteți realiza acest lucru prin crearea de noi fișiere cu extensia .m în directorul în care dezvoltați scriptul principal. Este de preferat ca aceste noi fișiere să aibă aceeași denumire cu denumirea funcției respective.

Orice fișier .m care definește o funcție începe cu sintaxa:

```
[y1,y2,…yn]=nume_funcție(x1,x2,…,xn)
```

După această sintaxă, este recomandat să se ofere explicații pentru fiecare parametru de ieșire (y1, y2, ...) și pentru fiecare parametru de intrare (x1, x2, ...), precum și să se descrie prelucrările realizate de funcția definită.

Exemplu

Definiți funcția treaptă folosind MATLAB, data de expresia:

$$u(n) = \begin{cases} 1 & , \quad n \geq 0 \\ 0 & , \quad n < 0 \end{cases}$$

Utilizând proprietatea de deplasare în timp, se poate scrie:

$$u(n - n_0) = \begin{cases} 1 & , \quad n \geq n_0 \\ 0 & , \quad n < n_0 \end{cases}$$

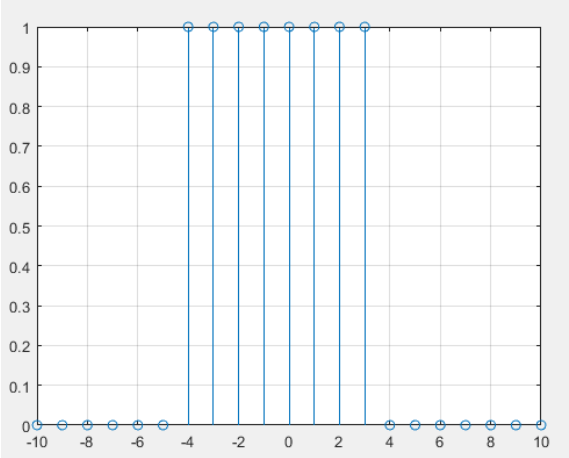
Astfel, se poate crea o funcţie MATLAB pentru definirea secvenţelor discrete de tip treaptă unitate, având un suport temporal finit:

```
function [y,n] = treapta(ni,ns,n0)
% Treapta unitate in timp discret
% Parametrii de iesire:
% y = u(n-n0) (vector linie) pe suportul ni:ns
% n = suportul temporal ni:ns
% Parametrii de intrare:
% ni = limita inferioara a suportului temporal
% ns = limita superioara a suportului temporal
% n0 = indicele din u(n-n0)
N = ns-ni+1;
Y = zeros (1, N);
y(n0-ni+1:N) = 1;
n = ni:ns;
end
```

Exerciții

Să se definească şi să se reprezinte grafic secvenţele:

1. $y_1[n]=u[n]$
2. $y_2[n]=0.7\cdot(u[n+3]-u[n-3])$
3. $y_3[n]=u[n]+0.5u[n-4]-0.5u[n+4]$

4. $y_4[n]$: 

Exerciții suplimentare

1. Generați și reprezentați grafic semnalele: *sm(t)* – sinus redresat monoalternanță, *sd(t)* – sinus redresat dublă alternanță.
2. Creați o matrice pătratică de dimensiuni 10x10 ce va conține valori întregi distribuite aleator în intervalul 1:10, folosind instrucțiunea `M = round(10*rand(10,10))`.
   a. calculați suma elementelor din colțurile matricei M;
   b. calculați suma elementelor matricei M;
   c. creați o funcție care calculează suma elementelor de pe fiecare linie a matricei M și returnează un vector coloană cu aceste sume.
3. Creați o funcție care construiește matricea identitate, considerând parametrii de intrare: m – numărul de linii și n – numărul de coloane ale matricei construite.
4. Creați o funcție care are ca parametru de intrare o matrice A și două valori: o valoare nouă m și una veche n. Funcția va returna matricea de la intrare cu valoarea veche schimbată cu valoarea nouă și numărul de elemente schimbate.
5. Creați o funcție care are la intrare o matrice A și returnează doi vectori, un vector care conține valorile pare din matricea A și un vector care conține valorile impare din matricea A.

Anexa 1

În Tabelul 1.2 sunt prezentate alte funcții (mai avansate) folosite în MATLAB.

Tabelul 1.2 Alte funcții MATLAB

Funcție/Parametru	Descriere și exemple
`cosh(x), sinh(x), tanh(x)`	Funcțiile cosinus hiperbolic, sinus hiperbolic și tangentă hiperbolică
`acosh(x), asinh(x), atang(x)`	Inversul funcțiilor cosinus hiperbolic, sinus hiperbolic și tangentă hiperbolică
`Whos`	Afișarea variabilelor din Workspace
`pause,` `pause(n)`	Pauză în rularea instrucțiunilor Pauză pentru n secunde în rularea instrucțiunilor
`Subplot`	Sub figură în interiorul aceleași figuri
`hold on, hold off`	Afișarea următoarelor grafice peste un grafic existent (on) sau următoarele grafice trebuie să fie independente de graficul curent (off)
`axis ([x_inf x_sup y_inf y_sup])`	Axele figurilor sunt afișate între limitele x_inf și x_sup (pe axa OX) iar pe axa OY, y_inf și y_sup

`xlabel(´nume_axa_OX´),`
`ylabel(nume_axa_OY),`
`title(nume_figură),`
`legend (parametrii)`
	Titlurile axelor, titlul figurii, legenda figurii

`save, load`	Salvarea sau încărcarea datelor intre-un/dintr-un fișier
`cd,` `pwd`	Schimba directorul curent, Afișarea numelui directorului curent
`input ("introduceți de la tastatura >>")`	Citirea valorii unei variabile de la tastatura

Bibliografie

1. Introducere în prelucrarea semnalelor folosind Python / Valentin-Adrian Niţă, Radu Alexandru Badea, Răzvan-Eusebiu Crăciunescu – Timişoara: Editura Politehnica, 2022
2. Mateescu, Adelaida, Dumitriu, N., Stanciu, L., Semnale, circuite şi sisteme, Teora, București, 2001
3. https://www.mathworks.com/