CUPRINS

LUCRAREA nr. **1**

INTRODUCERE ÎN MATLAB

Obiective*: însușirea elementelor de bază din sistemul MATLAB precum: definirea unei matrice, operații cu matrice, tipuri de date, variabile, instrucțiuni, fișiere script, reprezentarea graficelor și implementarea funcțiilor în Matlab.*

1.1. Introducere

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

1.2. Comenzi şi funcţii principale în MATLAB

MATLAB operează cu două tipuri de ferestre:

1) fereastra de comenzi;
2) fereastra pentru reprezentări grafice.

La deschiderea programului MATLAB, pe ecranul calculatorului va apărea fereastra de comenzi, având în partea de sus bara de meniuri aferentă. Simbolul ">>" reprezintă prompterul MATLAB şi se află la începutul fiecărei linii de comandă din

fereastra de comenzi. Ferestrele pentru reprezentări grafice vor apărea când se va cere prin comenzi specifice afişarea unor grafice.

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

1.2.1. Funcţii MATLAB de interes general

- **`help`** – furnizează informaţii despre MATLAB şi funcţiile acestuia.
  Sintaxă:
  `help nume` – furnizează informaţii despre `nume` (poate fi un nume de funcţie sau un nume de director).
  Exemplu:
  `help fft` – furnizează informaţii despre transformata Fourier discretă.
- **`who`** – listează numele variabilelor din spaţiul de lucru.
- **`whos`** – furnizează informaţii suplimentare referitoare la variabilele din spaţiul de lucru (nume, dimensiune etc.).
- **`format`** – stabileşte formatul extern de afişare al numerelor pe ecran.
  Sintaxă:
  `format opţiune` – parametrul `opţiune` poate fi:
  - `short` – 5 cifre // formatul implicit
  - `long` – 15 cifre
  - `short e` – 5 cifre + exp (puteri ale lui 10)
  - `long e` – 15 cifre + exp (puteri ale lui 10)

  etc.

  Pentru mai multe informaţii tastaţi `help format`.

Exemple:

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

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

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

- **`clear`** – şterge din memorie una sau mai multe variabile.
  Sintaxe:
  `clear v` – şterge din memorie variabila `v`;
  `clear v1 v2` – şterge din memorie variabilele `v1` şi `v2` (sintaxa e valabilă pentru oricâte variabile);
  `clear` – şterge din memorie toate variabilele definite până în acel moment.
- **`lookfor`** – listează toate numele de fişiere care au în prima linie a help-ului cuvintele menţionate ca argument, precum şi prima linie din help.
  Sintaxă:
  `lookfor cuvânt` – listează toate numele de fişiere care conţin în prima linie a help-ului `cuvânt`, precum şi prima linie din help.
  Exemplu:
  `lookfor ifft` – listează toate numele de fişiere care conţin în prima linie a help-ului `ifft`, precum şi prima linie din help.
- **`dir`** – afişează numele tuturor fişierelor din directorul curent sau din orice alt director precizat ca argument.
  Sintaxe:
  `dir` – afişează numele tuturor fişierelor din directorul current;
  `dir nume` – afişează numele tuturor fişierelor din directorul `nume`;
- **`cd`** – returnează numele directorului curent sau schimbă directorul de lucru.

Sintaxe:

`cd` – returnează numele directorului current;

`cd c:\matlab\nume_director` – schimbă directorul de lucru în `nume_director` (presupunând că MATLAB este instalat pe `c:\`).

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

1.2.2. Variabile şi constante speciale în MATLAB

- **`ans`** – variabilă creată automat în care este returnat rezultatul unui calcul, atunci când expresia nu a avut asignat un nume.

  Exemplu (se va tasta direct în fereastra de comenzi):

```
3      →    ans=          // nu s-a alocat niciun nume
                3
X=2    →    x=            // s-a alocat numele x
                2
```

- **`pi`** – valoarea π.
- **`Inf`** – reprezentarea lui $+\infty$.
- **`NaN`** – reprezentarea lui 0/0 sau ∞/∞.
- **`i`** sau **`j`** – folosite în reprezentarea numerelor complexe.

Atenţie**:**

- *dacă după o linie de comandă urmează semnul " ; ", atunci rezultatul nu va mai fi afişat* (*excepţie fac comenzile grafice*);
- *dacă în faţa unei linii de comandă se pune semnul " % ", atunci se face abstracţie de linia respectivă* (*este interpretată ca o linie de comentariu*);
- *dacă se doreşte continuarea unei instrucţiuni pe linia următoare, se folosesc "…" urmate de* enter.

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

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

1.2.3. Matricea – element de bază în MATLAB

MATLAB lucrează numai cu un singur tip de obiecte, şi anume, matrice numerice, având elemente reale sau complexe. Astfel, scalarii sunt priviţi ca matrice de dimensiune 1 x 1, iar vectorii ca matrice de dimensiune 1 x n (dacă este vector linie) sau n x 1 (dacă este vector coloană).

Reguli privind modul de definire a matricelor:

- elementele matricei sunt cuprinse între paranteze drepte [ ];
- elementele unei linii se separă prin pauză (blanc) sau virgule;
- liniile matricei se separă prin ; sau *enter*.

Exemplu:

Fie matricea $A=\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix}$ şi vectorii $B=[7 \quad 8 \quad 9]$, $C=\begin{bmatrix}-1\\ -2\end{bmatrix}$

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

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

Pentru o matrice M:

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

Pentru un vector v:

- `v(i)` reprezintă elementul de pe poziţia `i` din vector.

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

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

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

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

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

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

// s-a construit matricea D de dimensiune 4 x 3, prin adăugarea vectorului B la matricea A (ca ultimă linie).

Atenţie**:** *A şi B au acelaşi număr de coloane* (*3*) *pentru a fi posibilă construcţia.*

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

// s-a construit matricea E de dimensiune 3 x 4, prin adăugarea vectorului C la matricea A (ca ultimă coloană).

Atenţie**:** *A şi C au acelaşi număr de linii* (*3*) *pentru a fi posibilă construcţia.*

Dacă v este un *vector* atunci:

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

- `v(:)` – dacă vectorul este linie, atunci el devine coloană; dacă vectorul este coloană, atunci el rămâne nemodificat.

Dacă M este o *matrice* atunci:

- `M(:,j)` – selectează coloana j a matricei M.
- `M(i,:)` – selectează linia i a matricei M.
- `M(:,i:j)` – selectează coloanele de la i la j ale matricei M.
- `M(i:j,:)` – selectează liniile de la i la j ale matricei M.
- `M(:,i:j:k)` – selectează coloanele i, i + j, i + 2j , … până la k ale matricei M (selectează cu pasul j).
- `M(i:j:k,:)` – selectează liniile i, i + j, i + 2j , … până la k ale matricei M.
- `M(i:j,k:l)` – extrage submatricea formată cu elementele aflate la 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 şi k ale matricei M.
- `M([i,j,k],:)` – selectează liniile i, j şi k ale matricei M.
- `M([i,j,k],[l,m,n])` – extrage submatricea formată cu elementele aflate la intersecţia liniilor i, j şi k şi coloanelor l, m şi n ale matricei M.
- `M(:,:)` – selectează întreaga matrice M.
- `M(i:j)` – selectează elementele de i la j ale matricei M şi le pune sub forma unui vector linie (elementele într-o matrice se numără pe coloane).
- `M(:)` – selectează toate elementele matricei M şi le aranjează sub forma unui vector coloană (pune coloanele matricei M una sub alta, sub forma unui vector coloană).

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

1.2.4. Vectori şi matrice uzuale

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

Sintaxe:

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

Verificaţi următoarele *exemple*:

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

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

Sintaxe:

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

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

Verificaţi următoarele *exemple*:

```
g=logspace(1,2)
r=logspace(1,pi)
h=logspace(1,2,5)
k=logspace(0,pi,5)
```

- ***Matricea goală***

Sintaxa:

- `x=[]` – generează o matrice goală (fără niciun element).

Exemplu:

```
x=[]        →     x =
                        []
```

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

Sintaxe:

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

Verificaţi următoarele *exemple*:

```
ones(3)
ones(1,5)
ones(5,1)
ones(3,2)
ones(size(D))    // unde D este matricea definită anterior.
```

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

Sintaxe:

- `zeros(n)` – returnează o matrice de dimensiune n x n cu toate elementele egale cu 0;

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

Verificaţi următoarele *exemple*:

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

• **`eye`** - ***Matricea identitate***

Sintaxe:

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

Verificaţi următoarele *exemple*:

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

• **`rand`** - ***Matricea cu numere aleatoare cu distribuţie uniformă***

Sintaxe:

- `rand(n)` – returnează o matrice de dimensiune n x n, având drept elemente numere aleatoare cu distribuţie uniformă între 0 şi 1;
- `rand(m,n)` – returnează o matrice de dimensiune m x n, având drept elemente numere aleatoare cu distribuţie uniformă între 0 şi 1;

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

Verificaţi următoarele *exemple*:

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

• **`randn`** - ***Matricea cu numere aleatoare cu distribuţie normală* (*gaussiană*)**

Sintaxe:

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

Verificaţi următoarele *exemple*:

```
randn(3)
randn(1,5)
randn(5,1)
randn(3,2)
randn(size(D))  // unde D este matricea definită anterior.
```

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

Sintaxe:

Dacă v este un *vector* (linie sau coloană) atunci:

- `diag(v)` – returnează o matrice pătrată diagonală cu elementele vectorului v pe diagonala principal;

- `diag(v,k)` – returnează o matrice pătrată cu elementele vectorului v pe diagonala k deasupra celei principale, dacă k > 0, sau sub cea principală dacă k < 0; restul elementelor sunt 0.

Dacă M este o *matrice* atunci:

- `diag(M)` – returnează un vector coloană ce conţine elementele de pe diagonala principală a matricei M;
- `diag(M,k)` – returnează un vector coloană ce conţine elementele din matricea M de pe diagonala k deasupra celei principale, dacă k > 0, sau sub cea principală, dacă k < 0.

Se va defini un vector linie `a` şi o matrice `A`:

```
a=randn(1,5)
A=randn(5)
```

Verificaţi următoarele *exemple*:

```
diag(a)
diag(a,1)
diag(a,-1)
diag(a,2)
diag(a,-2)
diag(A)
diag(A,2)
diag(A,-2)
diag(diag(A))
```

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

Sintaxe:

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

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

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

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

Sintaxe:

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

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

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

1.2.5. Dimensiunea unei matrice. Determinantul şi inversa unei matrice

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

Sintaxe:

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

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

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

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

Verificaţi următoarele *exemple*:

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

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

Sintaxă:
Dacă M este o matrice pătratică (numărul de linii = numărul de coloane), atunci - `det(M)` – calculează determinantul matricei M.

Se vor defini două matrice:

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

Verificaţi următoarele *exemple*:

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

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

Sintaxă:
Dacă M este o matrice pătratică cu determinantul diferit de zero atunci - `inv(M)` – calculează inversa matricei M.

Verificaţi folosind matricele M şi N definite la punctul precedent.

E 1.1. ***Exerciţiu:***

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

a) Ce lungime trebuie să aibă **b** astfel, încât să aibă sens înmulţirea (în sens matricial) **a*b**? Iniţializaţi în MATLAB cei doi vectori şi efectuați înmulţirea.
b) Efectuați înmulţirea **b*a**.

1.2.6. Instrucţiuni de control logic
Operatori relaţionali şi operatori logici

- **`if,else,elseif`** – ***Execuţia condiţionată***

Sintaxe:

```
- if expresie
   instrucţiuni
end
```

// dacă `expresie` este adevărată, se execută `instrucţiuni`; dacă `expresie` este falsă, se trece după `end`.

```
-  if expresie
      instrucţiuni_1
   else
      instrucţiuni_2
   end
```

// dacă `expresie` este adevărată, se execută `instrucţiuni_1`; dacă `expresie` este falsă, se execută `instrucţiuni_2`.

```
-  if expresie_1
      instrucţiuni_1
   else if expresie_2
      instrucţiuni_2
   end
```

// dacă `expresie_1` este adevărată, se execută `instrucţiuni_1`; dacă `expresie_1` este falsă şi `expresie_2` este adevărată, se execută `instrucţiuni_2`.

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

Sintaxă:

```
- for index=iniţial:pas:final
      instrucţiuni
   end
```

// pentru `index`, parcurgând intervalul de la `iniţial` la `final` cu pasul `pas`, se execută `instrucţiuni`.

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

Sintaxă:

```
- while expresie
    instrucţiuni
  end
```

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

- ***Operatori relaţionali***

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

- ***Operatori logici***

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

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

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

Pentru crearea unui astfel de fişier se parcurg următorii *paşi*:

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

Atenţie:

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

E 1.2. ***Exerciţiu***:

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

1.2.8. Crearea funcţiilor MATLAB (crearea fişierelor funcţie)

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

funcţie este:

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

cu următoarele semnificaţii:

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

1.2.9. Operaţii aritmetice
Operaţii asupra numerelor complexe

- ***Operatori aritmetici:***

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

Să se definească următoarele elemente:

$$A=\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{bmatrix},\quad B=\begin{bmatrix}1-i & 2+i & 0\\ -i & 3 & i\end{bmatrix},$$

$$a=[1+i \quad 3 \quad 0],\qquad b=\begin{bmatrix}-1/3\\ -2^2\end{bmatrix}.$$

Verificaţi şi *explicaţi* următoarele *exemple*:

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

- ***Operaţii asupra numerelor complexe:***

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

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

1.2.10. Funţii matematice uzuale

- ***Funcţiile radical, exponenţială şi logaritm:***

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

- ***Funcţiile trigonometrice directe:***

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

- ***Funcţiile trigonometrice inverse:***

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

- ***Funcţiile hiperbolice directe:***

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

- ***Funcţiile hiperbolice inverse:***

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

Pentru informaţii despre modul de utilizare al acestor funcţii folosiţi comanda `help` însoţită de numele funcţiei dorite (vezi sintaxa de la pagina 2).

1.2.11. Funţii destinate analizei de date

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

Sintaxe:
Dacă v este un vector şi M este o matrice atunci:

- `sum(v)` – calculează suma elementelor vectorului v;
- `prod(v)` – calculează produsul elementelor vectorului v;
- `sum(M)` – returnează un vector linie, având ca elemente suma elementelor fiecărei coloane din matricea M;
- `prod(M)` – returnează un vector linie, având ca elemente produsul elementelor fiecărei coloane din matricea M.

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

Să se verifice următoarele *exemple*:

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

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

Sintaxe:
Dacă v este un vector şi M este o matrice atunci:

- `max(v)` – returnează elementul maxim al vectorului v;
- `min(v)` – returnează elementul minim al vectorului v;
- `[m,p]=max(v)` – returnează elementul maxim al vectorului (`m`), precum şi indicele elementului maxim (`p`);

dacă există maxime multiple, se returnează indicele primului dintre ele;

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

Verificaţi sintaxele anterioare, folosind matricea `A` şi vectorul `a` definite mai sus.

- **`mean`** – ***Media aritmetică***

Sintaxe:

Dacă v este un vector şi M este o matrice atunci:

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

Verificaţi sintaxele anterioare, folosind matricea `A` şi vectorul `a` definite mai sus.

E 1.3. ***Exerciţiu***:

Elaborați un program MATLAB care generează un vector cu elemente complexe. Elaborați (un alt fişier) o funcţie MATLAB care, având drept parametru de intrare vectorul cu valori complexe, returnează ca parametri de ieşire:

- media aritmetică a părţilor reale ale elementelor vectorului;
- un vector ce conţine elementele vectorului iniţial ridicate la pătrat;
- o matrice obţinută din înmulţirea vectorului iniţial cu transpusul său.

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

1.2.12. Reprezentări grafice

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

Sintaxe:

Dacă v este un vector şi M este o matrice atunci:

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

Dacă x şi y sunt doi vectori de *aceeaşi lungime* şi N este o matrice de *aceeaşi dimensiune* cu matricea M atunci:

- `plot(x,y)` – se vor reprezenta grafic elementele vectorului y în funcţie de elementele vectorului x; *dacă lungimea vectorului* x *nu este egală cu lungimea vectorului* y, *reprezentarea nu este posibilă*;

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

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

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

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

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

Vezi `help stem`.

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

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

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

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

1	2	3
4	5	6
7	8	9

Sintaxă:

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

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

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

Sintaxa:

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

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

Sintaxe:

- `title('text')` – plasează deasupra graficului, ca titlu, textul `text`;
- `xlabel('text')` – textul `text` devine eticheta de pe abscisă;
- `ylabel('text')` – textul `text` devine eticheta de pe ordonată;
- `grid` – trasează pe grafic o reţea de linii, înlesnind astfel citirea graficului;
- `gtext('text')` – plasează pe grafic textul `text` (folosind mouse-ul). Toate aceste sintaxe urmează după comanda de reprezentare grafică.

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

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

Sintaxă:

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

***Atenţie*:**
Dacă se doreşte în cadrul unui program reprezentarea mai multor grafice în ferestre separate, fiecare comandă grafică va trebui să fie precedată de un nume de forma `figure(n)`, *unde* `n` *este numărul figurii respective. În caz contrar, la sfârşitul execuţiei programului va apărea numai ultima reprezentare grafică (se va folosi o singură fereastră grafică ce va fi "ştearsă" de fiecare dată la întâlnirea unei noi comenzi grafice).*

E 1.4. ***Exerciţiu***

Elaborați un program MATLAB în care să generaţi şi să reprezentaţi grafic folosind funcţia **stem** următorii vectori:

a) $\boldsymbol{z}$ = [0,0,0,0,0,1,0,0,...,0], vectorul $\boldsymbol{z}$, având lungimea 21. Reprezentarea grafică se va face în două „miniferestre" (funcția `subplot`), vectorul z în funcție de **n=0:20**, respectiv de **m=-5:15**;

b) **t=|10-n |**, reprezentat graphic în fincție de **n=0:20**;

c) $x_1 = sin\left(\frac{\pi}{17}n\right)$, -15≤*n*≤25 și $x_2 = cos\left(\frac{\pi}{\sqrt{23}}n\right)$, 0≤*n*≤50.

Cele două secvenţe vor fi reprezentate:

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

Reprezentaţi cele două figure, folosind comanda `plot`, apoi încercaţi reprezentarea lor, folosind comanda `stem`.

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

`t = 0:0.001:5` (secunde).

Exemplu:

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

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

E 1.5. ***Exerciţiu*:**

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

E 1.6. ***Exerciţiu*:**

Să se genereze vectorul `x` conținând valorile 1, 2, 3, ..., 99, 100 și vectorul `y` având valorile 2, 4, 6, 8, ..., 198, 200:

a) să se reprezinte grafic `y` în funcție de `x`, folosind funcția `stem`;
b) să se reprezinte grafic `y` în funcție de `x`, folosind funcția `plot`;
c) să se reprezinte grafic în aceeași figură, în același sistem de coordonate, `y` în funcție de `x`, folosind funcțiile `plot` și `stem` (se vor folosi culori diferite);
d) să se reprezinte grafic în aceeași figură, în sisteme de coordonate diferite, `y` în funcție de `x`, folosind funcția `plot` și `y` în funcție de `x`, folosind funcția `stem`.

1.2.13. Derivare și integrare

Aproximarea derivatei unei funcţii în MATLAB se face cu diferenţe finite (regresive, progresive sau centrate). Pentru exprimarea formulelor diferenţelor finite, MATLAB pune la dispoziţia utilizatorului funcţia auxiliară `diff`.

Funcţia MATLAB `diff` evaluează diferenţa elementelor succesive ale unui vector sau coloanelor unei matrice. Sintaxa funcţiei `diff` prezintă trei forme de apel:

```
y=diff(f)
y=diff(f,x)
y=diff(f,x,dim)
```

unde:

- `f` este un vector, o matrice sau o funcție reală;

- `x` este un întreg strict pozitiv, prin intermediul căruia se precizează de câte ori se va apela recursiv funcţia `diff`;

- `dim` are semnificaţia de dimensiune a lui `x`: implicit, funcţia `diff` calculează diferenţele după prima dimensiune a unei matrice `x` (de exemplu, pentru o matrice bidimensională diferenţele se determină pe coloane); specificarea unei alte dimensiuni după care să se realizeze diferenţele se poate face prin precizarea sa prin intermediul parametrului `dim`.

Fie `f` o funcţie reală de variabilă reală derivabilă pe intervalul `[a,b]` şi $\{x_k\}_{k=1,2,..,n}$ un şir strict crescător de puncte din intervalul `[a,b]`. Derivata `df` (notaţie MATLAB) de ordinul I a funcţiei `f` se poate aproxima cu diferenţe finite în modul următor, în care `x` este vectorul punctelor $\{x_k\}$ şi `y` este vectorul valorilor funcţiei `f` în punctele $\{x_k\}$:

• în cazul utilizării diferenţelor regresive sau progresive:

`df=diff(y)./diff(x)`

`df` are lungimea cu o unitate mai mică decât vectorii `x` şi `y`.

Dacă se doreşte ca aproximarea derivatei să se facă cu diferenţe regresive, între vectorul `x` şi vectorul `df` se va face asocierea: `x(2:n)` → `df`, adică, aproximarea derivatei într-un punct `x(k)` are valoarea `df(k-1), k=2,...,n`.

Dacă se doreşte ca derivata să fie aproximată cu diferenţe progresive, între vectorii `x` şi `df` se va considera asocierea: `x(1:n-1)` → `df`, adică, aproximarea derivatei într-un punct `x(k)` are valoarea `df(k), k=1,...,n-1`.

• în cazul utilizării diferenţelor centrate:

```
dx=diff(x); dy=diff(y); k=length(dx);
df=(dy(1:k-1)+dy(2:k))./(dx(1:k-1)+dx(2:k))
```

Folsirea diferenţelor finite duce la obţinerea valorilor derivatei funcţiei `f` în punctele şirului (cu excepţia unuia sau a ambelor capete ale şirului).

Exemplu:

Fie `f` o funcţie reală de o variabilă reală, precizată de următorii vectori:

```
x=[1 1.5 2.1 2.4 3 3.2];
y=[56 41 30 37 42 40].
```

Să se aproximeze derivata de ordinul I a funcţiei `f` în punctele `1.5`, `2.1` şi `3.2`, folosind diferenţe regresive.

Soluţie:

```
% vectorii ce determina functia
x=[1 1.5 2.1 2.4 3 3.2];
y=[56 41 30 37 42 40];
% aproximarea derivatei
df=diff(y)./diff(x);
% afisarea derivatei in punctele dorite
vx=[1.5 2.1 3.2];
for i=1:length(vx)
 % determinarea pozitiei pe care se afla vx(i)
in x
 k=find(x==vx(i));
 % afisarea derivatei cu diferente regresive
 disp(['f''(' num2str(vx(i)) ') = '
num2str(df(k-1))])
end
```

În urma execuţiei acestei secvenţe se obţine:

```
f'(1.5) = -30
f'(2.1) = -18.3333
f'(3.2) = -10
```

Exemplu:

Să se aproximeze în punctele -2, -1, -0.5, 0.5, 1 şi 1.45 derivata de ordinul I a funcţiei `f` precizată de vectorul `y=[100 90 86 62 74 81]` pe intervalul `[-2:3]`:

Soluţie: Deoarece se doreşte aproximarea derivatei funcţiei `f` şi în primul punct, se va utiliza formula cu diferenţe progresive. Această formulă va putea fi folosită pentru a calcula derivata în punctele -2, -1 şi 1, deoarece acestea fac parte din elementele $\{x_i\}$. Pentru punctele -0.5, 0.5 şi 1.45, care nu apar în șirul de

valori, derivata se va aproxima prin interpolare cu funcţii spline. Se execută următoarea secvenţă MATLAB:

```
% datele initiale
x=-2:3;
y=[100 90 86 62 74 81];
% aproximarea derivatei
df=diff(y)./diff(x);
% afisarea derivatei in punctele x(i) dorite
vx1=[-2 -1 1];
for i=1:length(vx1)
 % determinarea pozitiei pe care se afla vx(i)
in x
 k=find(x==vx1(i));
 % afisarea derivatei cu diferente progresive
 disp(['f''(' num2str(vx1(i)) ') = '
num2str(df(k))])
end
% afisarea derivatei in punctele -0.5, 0.5 si
1.45
vx2=[-0.5 0.5 1.45]; n=length(x);
der=spline(x(1:n-1),df,vx2);
for i=1:length(der)
 disp(['f''(' num2str(vx2(i)) ')='
num2str(der(i))])
end
```

şi se obţin rezultatele:

```
f'(-2) = -10
f'(-1) = -4
f'(1) = 12
f'(-0.5)=-18.6719
f'(0.5)=-9.7344
f'(1.45)=23.3022
```

Exemplu:

Fie funcţia `f` dată prin puncte de relaţiile

$$f(x_i) = \frac{sin(x_i)}{i^2+1} cos\left(\frac{i}{x_i}\right), x_i = \pi + i\frac{\pi}{30}, i = 1,2,\dots,150.$$

Să se reprezinte grafic derivata de ordinul I a funcţiei pe intervalul $[x_2; x_{149}]$, folosind diferenţe centrate.

Soluţie: Se execută următoarea secvenţă de program Matlab:

```
% generarea vectorilor x si y
for i=1:150
 x(i)=pi+i*pi/30;
 y(i)=sin(x(i))/(i^2+1)*cos(i/x(i));
end
% aproximarea derivatei cu diferente centrate
dx=diff(x); dy=diff(y); k=length(dx);
df=(dy(1:k-1)+dy(2:k))./(dx(1:k-1)+dx(2:k));
% reprezentarea grafica a derivatei
n=length(x);
plot(x(2:n-1),df)
```

Graficul derivatei este redat în figura 1.1.

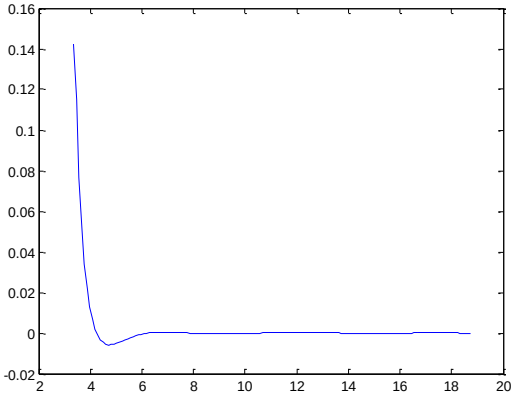


Fig.1.1. Graficul derivatei funcţiei

Pentru derivarea funcţiilor polinomiale, MATLAB pune la dispoziţia utilizatorului o funcţie specială numită `polyder`, cu sintaxa de apel:

`dc=polyder(c)`

unde:

- `c` reprezintă vectorul coeficienţilor funcţiei polinomiale, în ordine descrescătoare după puterea variabilei funcţiei;
- `dc` reprezintă vectorul coeficienţilor funcţiei polinomiale

obţinute prin derivare, tot în ordine descrescătoare după puterea variabilei funcţiei.

Observaţii:

1. Derivarea podusului a două funcţii polinomiale se poate face tot cu funcţia MATLAB `polyder`, folosind sintaxa de apel:

`dc=polyder(c1,c2)`

`c1` fiind vectorul coeficienţilor primei funcţii, iar `c2` vectorul coeficienţilor celei de a doua funcţii.

2. Funcţia `polyder` poate fi folosită şi pentru calculul derivatei unei fracţii polinomiale P/Q, rezultatul fiind exprimat tot sub forma unei fracţii polinomiale (M/N)=(P/Q)':

`[m,n]=polyder(p,q)`

unde `p, q, m, n` sunt vectorii coeficienţilor corespunzători polinoamelor `P, Q, M,` respectiv `N`.

Exemplu:

Fie funcţia polinomială `f`: $f(x)=x^5+7\cdot x^4-3\cdot x^2+x-1$.

Să se calculeze derivata funcţiei în punctele -3, 0.2 şi 4. Să se reprezinte grafic derivata funcţiei `f` pe intervalul [-4,5].

Soluţie: Se execută următoarea secvenţă de instrucţiuni MATLAB:

```
% vectorul coeficientilor functiei polinomiale
c=[1 7 0 -3 1 -1];
% vectorul coeficientilor derivatei
dc=polyder(c);
% valorile derivatei in punctele -3, 0.4 si 4
vx=[-3 0.2 4];
format short g
der=polyval(dc,vx)
% reprezentarea grafica a derivatei pe [-4,5]
x=-4:0.1:5; df=polyval(dc,x);
plot(x,df)
```

Se obţin valorile derivatei în punctele -3, 0.4, respectiv 4:

```
der =  -332      0.032     3049
```

Graficul derivatei pe intervalul [-4,5] este redat în figura 1.2

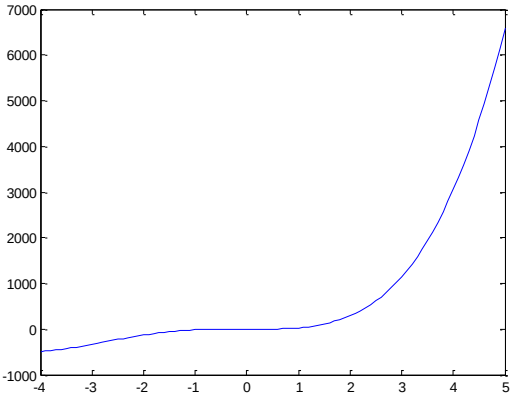


Fig.1.2. Graficul derivatei funcţiei polinomiale

Pentru **calculul integralelor** definite prin ***metoda trapezelor***, în MATLAB se utilizează funcţia `trapz`. Această funcţie presupune că funcţia de integrat `f` este precizată sub formă de valori numerice, $\{y_k=f(x_k)\}, k=1,...,n$, în punctele echidistante $\{x_k\}, k=1,...,n$ $(x_1=a,\ x_n=b)$ ale intervalului de integrare `[a,b]`. Sintaxa de apel a funcţiei `trapz` este:

`I=trapz(x,y),`

unde: - `x` reprezintă vectorul valorilor $\{x_k\}$;
- `y` reprezintă vectorul valorilor $\{y_k=f(x_k)\}$;
- `I` reprezintă aproximarea cu metoda trapezelor a integralei definite $\int_a^b f(x)dx$.

Observaţie: `n` reprezintă aici numărul de puncte. Prin urmare, pasul de integrare, calculat implicit, este dat de formula:

$$h = \frac{b-a}{n-1}.$$

Exemplu:

Să se calculeze $\int_{\pi+\frac{\pi}{30}}^{6\pi} f(x)dx$ pentru funcţia `f`:

$$f(x_i) = \frac{sin(x_i)}{i^2+1} cos\left(\frac{i}{x_i}\right), x_i = \pi + i\frac{\pi}{30}, i = 1,2,\dots,150.$$

Soluţie: Funcţia, fiind cunoscută prin puncte, se va folosi metoda trapezelor pentru calculul integralei cerute. Se execută următoarea secvenţă MATLAB:

```
% generarea vectorilor x si y
for i=1:150
 x(i)=pi+i*pi/30;
 y(i)=sin(x(i))/(i^2+1)*cos(i/x(i));
end
% calculul integralei folosind metoda trapezelor
I=trapz(x,y)
```

Rezultând următoarea valoare a integralei:

`I = -0.0025.`

Funcţia MATLAB `quad` realizează calculul integralei definite a unei funcţii prin ***metoda adaptativ-recursivă Simpson*** (o variantă mai performantă a metodei lui Simpson, pasul de parcurgere a intervalului de integrare este calculat implicit de către funcţia MATLAB). Funcţia `quad` presupune că funcţia de integrat `f` este cunoscută prin expresia sa analitică, `y = f(x)`. Două sintaxe de apel ale funcţiei quad sunt:

`I=quad(nume_fisier,a,b)`;

`I=quad(nume_fisier,a,b,precizia)`,

unde:

- `nume_fisier` reprezintă un şir de caractere ce conţine numele fişierului-funcţie în care a fost scrisă expresia funcţiei de integrat `f`;

- `a` şi `b` reprezintă limitele de integrare (capetele intervalului `[a,b]` pe care se realizează integrarea);

- `precizia` este un argument opţional prin care se poate modifica precizia implicită 10^{-6};

- `I` reprezintă aproximarea integralei definite $\int_a^b f(x)dx$.

Exemplu:

Să se calculeze integrala $\int_0^{\pi} \ln(x+1)\sin(x)dx$.

Soluţie: Funcţia de integrat, având expresia cunoscută, integrala ei va fi calculată cu ajutorul metodei lui Simpson. Se

defineşte expresia funcţiei de integrat într-un fişier-funcţie (de exemplu, `f.m`):

```
function y=f(x)
y=log(x+1).*sin(x)
```

Pentru calculul integralei se apelează funcţia MATLAB `quad`:

```
I=quad('f',0,pi),
```

rezultând valoarea:

```
I = 1.8113.
```

E 1.7. ***Exerciţiu*:**

1. Fie `f` o funcţie reală de o variabilă reală, precizată de următorii vectori:

```
x=[-2 -1.5 0 0.5 2.5 3];
y=[6 9 11 10 7 5].
```

Să se aproximeze derivata de ordinul I a funcţiei `f` în punctele:

a) -2, 0 şi 2.5, folosind diferenţe progresive;

b) -1.5, 0 şi 3, folosind diferenţe regresive;

c) -0.75, 0.25, 0.5 şi 1, folosind diferenţe centrate.

Să se reprezinte grafic în aceeaşi fereastră grafică derivata de ordinul I a funcţiei `f`, obţinută prin aproximare cu cele trei tipuri de diferenţe finite.

2. Fie funcţia polinomială f : $\boldsymbol{R} \to \boldsymbol{R}$, $f(x)=x^4+13\cdot x^3-7\cdot x^2+x-1$.

a) Să se calculaze derivata funcţiei *f* pentru valorile -2, -1.3, 0.1 şi 2.45.

b) Să se reprezinte grafic derivata funcţiei *f* pe intervalul [-2,3].

3. Să se calculeze $\int_0^{\pi} f(x)dx$, unde funcţia `f` este dată prin relaţiile:

$$f(x_j) = \frac{j\cdot x_j^2}{x_j-1} - \frac{2}{j+1},\ x_j = -1.1 + 0.1\cdot j,\ j = 1,2,\dots,11.$$

4. Să se calculeze integrala $\int_{\pi/3}^{\pi/2} \frac{1}{\sin(x)+\cos(x)}\,dx$.

LUCRAREA nr. **2**

FORMAREA SEMNALELOR ELEMENTARE ÎN SISTEMUL MATLAB

***Obiective**: studierea posibilităţilor sistemului MATLAB în modelarea diferitor forme de semnale în vederea cercetării particularităţilor lor de bază.*

2.1. Introducere

Se numeşte ***semnal*** o mărime fizică măsurabilă, purtătoare de informaţie, care poate fi transmisă la distanţă, recepţionată şi/sau prelucrată. În cele ce urmează se va aborda problema modelării formei semnalelor, fără a ne preocupa de conţinutul în informaţie al semnalelor.

Un semnal ***unidimensional***, numit şi semnal 1D, este o funcţie de timp, notată generic prin $x(t)$, $t \in \Re$. De regulă, mărimea fizică variabilă, reprezentând semnalul, este o tensiune electrică. Totuşi, în echipamentele de automatizări se utilizează şi semnale de altă natură fizică, aşa cum sunt, de exemplu: curentul electric, presiunea aerului instrumental, deplasarea unui corp solid. În telecomunicaţii, semnalul 1D este întotdeauna o tensiune electrică variabilă în timp.

Semnalele se pot aplica unor circuite sau, în general, unor sisteme dinamice. Fie $u(t)$ semnalul aplicat la intrarea unui sistem şi $y(t)$ semnalul obţinut la ieşirea acestuia, numit şi *răspuns* al sistemului la semnalul de intrare (fig. 2.1). Sistemele dinamice realizează prelucrarea semnalelor, conform cu funcţiunile realizate de echipamentele electronice în care sunt înglobate.

Fig. 2.1. Sistem dinamic

Exemplificăm câteva operaţii uzuale de prelucrare a semnalelor: integrarea unui semnal, derivarea acestuia, filtrarea (extragerea unor componente spectrale ale semnalului sau, după caz, eliminarea componentelor parazite), modulaţia semnalelor etc. De fapt, cele mai multe echipamente electronice sunt formate din lanţuri de sisteme dinamice, care realizează prelucrări consecutive ale semnalelor, conform unei „tehnologii" care determină funcţiunile realizate de echipamentul respectiv.

Semnalele pot fi ***cu timp continuu*** şi ***cu timp discret***. În circuitele analogice de procesare, semnalele sunt cu timp continuu, fiind adesea numite ***semnale analogice***.

Semnalele numerice pot fi generate de echipamente numerice sau se pot obţine din cele analogice prin două operaţii:

- *eşantionarea* semnalului, adică discretizarea timpului t cu un pas T_e, numit perioadă de eşantionare. Semnalul cu timp discret, $x(kT_e)$, este notat adesea cu $x(k)$, unde k reprezintă timpul discret, adică pasul curent de eşantionare;
- *cuantizarea* semnalului, adică discretizarea amplitudinii eşantioanelor $x(k)$. Se alege un pas de cuantizare, q, iar rezultatul operaţiei de cuantizare este un număr întreg, N, astfel încât produsul $q \cdot N$ să fie cât mai apropiat de amplitudinea eşantionului cuantizat.

Cele două operaţii se realizează uzual în cadrul unui convertor analogic/numeric (A/N). La ieşirea acestuia se obţine un şir de valori numerice, $\{x_k\}$, aferente momentelor de timp discrete k. Acest şir reprezintă un semnal numeric. Într-un sistem numeric (fig. 2.2), procesarea semnalului de intrare, $\{u_k\}$, în vederea obţinerii răspunsului $\{y_k\}$ se realizează prin mijloace software.

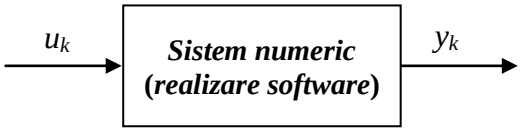


Fig. 2.2. Sistem numeric

Semnalele care au o evoluţie ce nu este supusă hazardului se numesc ***semnale deterministe***. Alături de acestea, se întâlnesc şi semnalele ***aleatoare***, a căror evoluţie în timp este supusă hazardului, aşa cum sunt perturbaţiile care afectează sistemele de transmitere şi prelucrare a informaţiilor.

Din clasa ***semnalelor unidimensionale*** menţionăm: semnalul vocal, semnalul radio (modulat în amplitudine sau în frecvenţă), semnalele furnizate de traductoare ale mărimilor fizice uzuale (temperatură, viteză ş.a.) etc.

Semnalele ***bidimensionale***, numite şi semnale 2D, sunt, de regulă, imagini. Fie $u(x_1, x_2)$ un semnal bidimensional, în raport cu coordonatele spaţiale x_1 şi x_2. Mărimea u reflectă valoarea nivelului de gri în punctul de coordonate x_1 şi x_2. Ca şi în cazul semnalelor unidimensionale, modelarea matematică a semnalelor 2D vizează facilitarea descrierii operaţiilor de prelucrare. Aceste operaţii de prelucrare se realizează cu ajutorul sistemelor 2D (fig. 2.3). Semnalul de ieşire din sistem, $y(x_1, x_2)$, se obţine prin aplicarea unor operaţii specifice (filtrare, extragere contur etc.) aplicate semnalului de intrare $u(x_1, x_2)$.

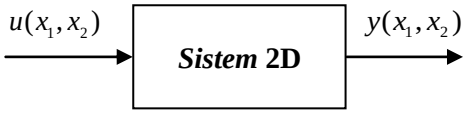


Fig. 2.3. Sistem 2D

Uneori este necesar să se extragă principalele proprietăţi ale semnalelor (proprietăţile cheie) prin aplicarea, asupra acestora, a unor algoritmi de prelucrare. Este posibil să se identifice proprietăţile sistemului discret prin analiza semnalelor de ieşire la aplicarea anumitor semnale de intrare ale sistemului studiat. În acest caz, este necesar, pentru început, de a învăţa generarea unor semnale de bază în mediul MATLAB şi de a efectua operaţii elementare asupra lor, ceea ce determină sarcina iniţială a acestei lucrări de laborator.

2.2. Formarea semnalelor impuls unitare

Funcția impuls Dirac sau **funcția delta**, numită și „**impuls" unitar**, notată **δ(x)**, nu este o funcție obișnuită, ci o *funcție generalizată* (sau o *distribuție*).

Un **„impuls" unitar** analitic se definește astfel:

$$\delta = \begin{cases} 1, & k = 0, \\ 0, & k \neq 0. \end{cases}$$

Prin trecere la limită se obţine un impuls de amplitudine infinită şi de durată nulă, notat cu ***δ(t)***. Aria impulsului este egală cu 1 (de aceea se numeşte unitar), adică:

$$\int_{-\infty}^{\infty} \delta(t)dt = 1,$$

defineşte „mărimea" impulsului.

Pentru a forma un impuls unitar, poziționat în momentul de timp t=0, se utilizează următoarea secvență de program:

```
k=-10:10;
y=zeros(size(k));
y(11)=0.6;
figure; stem(k,y);
xlabel('k'); ylabel('y(k)')
```

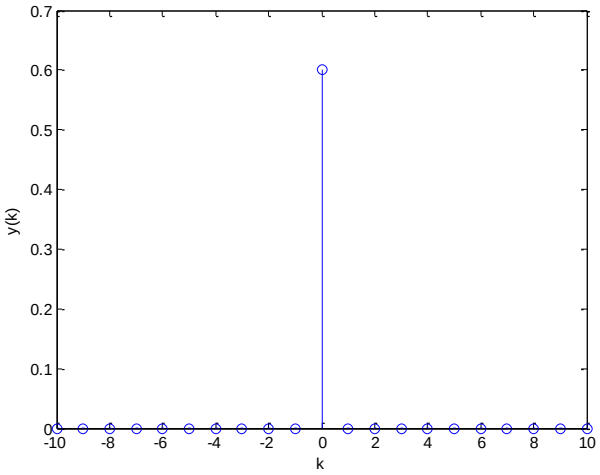


Fig.2.4. Funcția „impuls" unitar

Impulsurile $\delta[n - n_0]$ pot fi folosite la construirea trenurilor de impulsuri periodice, de amplitudine A_l, perioadă P şi lungime M*P:

$$s[n] = \sum_{l=0}^{M-1} A_l \delta[n - lP].$$

Următorul program generează şi reprezintă un tren periodic de impulsuri:

```
P=5; M=6;
%generarea impulsului generator, de
lungime P
d=[1;zeros(P-1,1)]; y=d*ones(1,M);
%generarea trenului de impulsuri de
lungime P*M
tren=y(:);
%reprezentare grafica
n=0:M*P-1;
stem(n,tren);
xlabel('n');ylabel('Amplitudine');
title('Tren de impulsuri unitate');
axis([-2 32 0 1.2]);
```

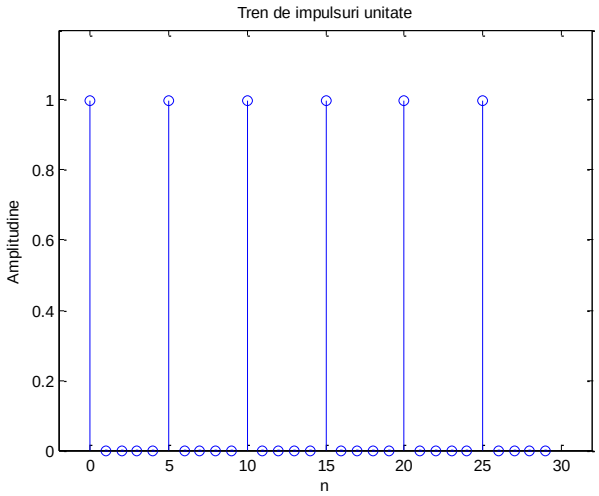


Fig.2.5. Tren de impulsuri unitare

Semnalul **treaptă unitară** analitic se definește astfel:

$$u(t)=\begin{cases}1 & \text{pt. } t\geq 0\\ 0 & \text{pt. } t<0\end{cases}$$

În sistemul MATLAB funcția treaptă unitară poate fi formată utilizând procedura `ones`.

```
t=-0.5:0.002:1.5;
U=[zeros(1,251),ones(1,750)];
plot(t,U);
xlabel('t,s');ylabel('Amplitudine');
title('Treapta unitara');
axis([-0.5 1.5 0 1.5]);
```

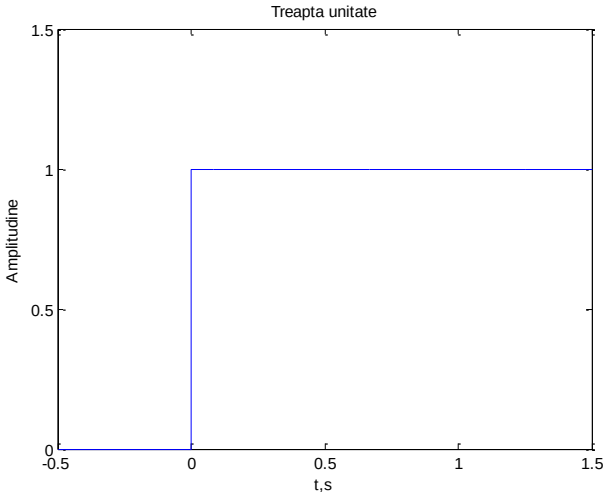


Fig.2.6. Semnalul treaptă unitară

Impulsul unitar dreptunghiular (rectangular) poate fi format cu ajutorul funcției `rectpuls`, de forma: **`y=rectpuls(t,w)`** - care permite a forma vectorul *y* al valorilor semnalului impuls rectangular de amplitudine unitară și durată `w`, centrat față de *t*=0 după vectorul dat *t* al momentelor de timp. Dacă durata impulsului *w* nu este dată, atunci valoarea ei se

ia egală cu 1. În figura 2.7 se prezintă rezultatul îndeplinirii scriptului de mai jos, care constă din trei impulsuri dreptunghiulare consecutive de amplitudine și durată diferită.

```
t=0:0.01:10;
y=0.75*rectpuls(t-2.2,2)+1.4*rectpuls(t-
5.1)+0.5*rectpuls(t-8.04);
plot(t,y),grid;
xlabel('timpul(s)');ylabel('y(t)')
```

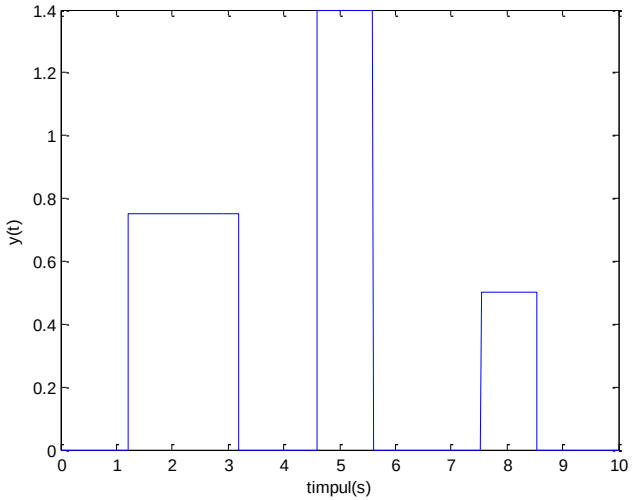


Fig.2.7. Impulsuri dreptunghiulare

Impulsul de formă triunghiulară și amplitudine unitară se realizează cu ajutorul procedurii `tripuls`, care are forma: **`y=tripuls(i,w,s)`**. Argumentele `y, t, w,` au acelaşi sens ca şi în exemplul de mai sus. Argumentul `s` ($-1 \le s \le 1$) determină înclinarea triunghiului. Dacă $s=0$, sau dacă nu este indicat, atunci impulsul triunghiular este simetric. Funcţia `y=tripuls(t,w)` formează un impuls simetric, de amplitudine unitară și durată `w`, centrat faţă de $t=0$. În figura 2.8 se prezintă un exemplu de formare a unei succesiuni de semnal, alcătuită din trei impulsuri:

```
t=0:0.01:10;
y=1.1*tripuls(t-1.5,0.7,1)+1.4*tripuls(t-
5,2)+0.8*tripuls(t-8,1.5,-1);
plot(t,y); grid
xlabel('timpul(s)')
ylabel('y(t)')
```

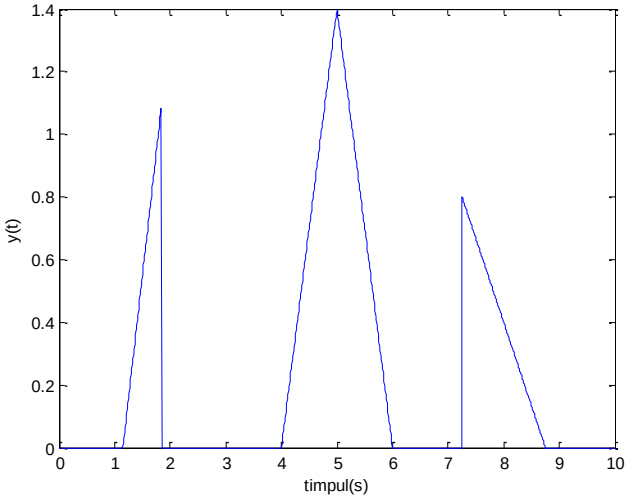


Fig.2.8. Impulsuri triunghiulare

Impulsul sinusoidal modulat de funcţia Gauss se execută prin procedura `gauspuls`, aceasta poate avea următoarele forme:

```
yi=gauspuls(t,fc,bw)
yi=gauspuls(t,fc,bw,bwr)
[yi,yq]=gauspuls(...)
[yi,yq,ye]=gauspuls(...)
tc=gauspuIs('cutoff',fc,bw,bwr,tpe)
```

Funcţia **`yi=gauspuls(t,fc,bw)`** formează secvenţa eşantioanelor unui impuls, calculate în momentele de timp date de vectorul *`t`*; `fc` [Hz] determină frecvenţa purtătoarei semnalului; `bw` - lărgimea relativă a benzii de frecvenţă a semnalului în dB (ia valori pozitive). Implicit, se utilizează următoarele mărimi: *fc=1000Hz* şi *bw=0.5 dB* ($bw=\Delta f/f_c$).

Funcţia **yi=gauspuls(t,fc,bw,bwr)** calculează secvenţa eşantioanelor semnalului cu următorii parametri: amplitudinea unitară, lăţimea relativă a benzii de frecvenţă `bw`, determinată de nivelul de atenuare `bwr` [dB] în raport cu amplitudinea maximală a spectrului semnalului. Parametrul `bwr` trebuie să ia valori negative, deoarece determină atenuarea necesară a funcției spectrale față de valoarea ei maximală. Implicit, `bwr=-6` *dB*.

Funcţia **[yi,yq]=gauspuls(...)** calculează doi vectori. Vectorul *yi* conţine eşantioanele semnalului iniţial, iar vectorul *yq* conţine eşantioanele semnalului cu un defazaj de 90^0 față de cel inițial.

Funcţia **[yi,yq,ye]=gauspuls(...)** calculează adăugător semnalul `ye`, ce reprezintă anvelopa impulsului.

Funcţia **tc=gauspuls('cutoff,fc,bw,bwr,tpe)** calculează timpul `tc`, corespunzător momentului de timp în care amplitudinea semnalului scade până la `tpe` [dB] față de valoarea maximală. Mărimea `tpe` trebuie să ia valori negative, deoarece setează atenuarea necesară a înfășuratoarei în raport cu nivelul ei maximal. Implicit `tpe=-60dB`.

Construim curba impulsului Gaussian cu frecvenţa purtătoarei 50 kHz şi banda relativă de frecvenţă 60%, considerând pasul de calcul 1 MHz. Pentru calculul şi construcţia impulsului se utilizează intervalul de timp, pe durata căruia amplitudinea impulsului se micşorează până la nivelul -40 dB în raport cu valoarea lui maximală.

```
tc = gauspuls('cutoff', 50e3, 0.6, [], -40);
t = -tc:1e-6:tc;
yi = gauspuls(t,50e3,0.6);
plot(t,yi);
grid on
xlabel('timpul(s)')
ylabel('yi(t)')
```

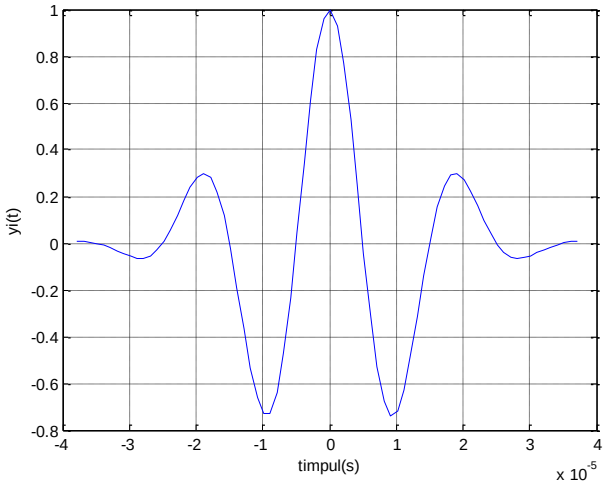


Fig. 2.9. Impuls Gaussian

Procedura ***sinc*** permite de a calcula valoarea funcţiei *sinc(t)* față de elementele vectorului sau matricei de intrare. Funcția este determinată de următoarea expresie:

$$sinc(t) = \begin{cases} 1, & t = 0, \\ \dfrac{\sin(\pi t)}{\pi t}, & t \neq 0. \end{cases}$$

Această funcție reprezintă o transformare inversă Fourier a impulsului dreptunghiular de amplitudine unitară şi durată 2π:

$$sinc(t) = \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{j\omega t} d\omega.$$

Exemplu:

```
t=0:0.01:40;
yi=1.9*sinc(pi*(t-20)/5);
plot(t,yi); grid;
xlabel('timpul(s)');
ylabel('yi(t)')
```

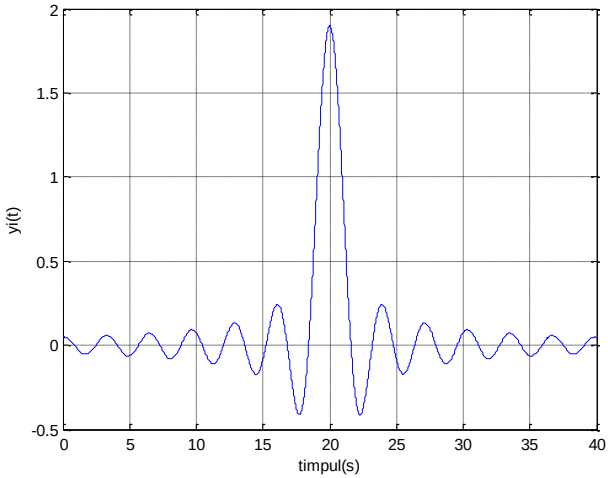


Fig. 2.10. Funcția *sinc(t)*

2.3. Formarea oscilațiilor periodice

Semnalele armonice și poliarmonice pot fi generate în MATLAB cu ajutorul operatorilor trigonometrici *sin(x)* şi *cos(x).*

Expresia analitică a semnalului sinusoidal este:

$$x(t) = A \sin(\omega_0 t + \varphi_0).$$

Considerând că *x(t)* descrie un semnal electric (tensiune sau curent), parametrii ce-l definesc sunt următorii:

- A – amplitudinea semnalului [V] sau [A].

Observaţie: valoarea efectivă a semnalului este definită astfel:

$$A_{ef} = \frac{\sqrt{2}}{2} A \approx 0{,}707\, A;$$

- $\varphi = (\omega_0 t + \varphi_0)$ – faza [rad];
- ω_0 - pulsaţia [rad/s].

Cum $\omega_0 = (2\pi)/T = 2\pi f$, se pot pune în evidenţă alţi doi parametri:

- f - frecvenţa semnalului [Hz];
- T – perioada semnalului [s]. *Observaţie:* f = 1/T;
- φ_0 - faza iniţială [rad].

Scriptul pentru generarea semnalului sinusoidal:

```
clf;
t=0:0.1:40;
f=0.1;
phase=0;
A=1.5;
arg=2*pi*f*t-phase;
x=A*cos(arg);
plot(t,x);
axis([0 40 -2 2]);
grid;
title('Sinusoida');
xlabel('Timpul t');
ylabel('Amplitudinea');
axis;
```

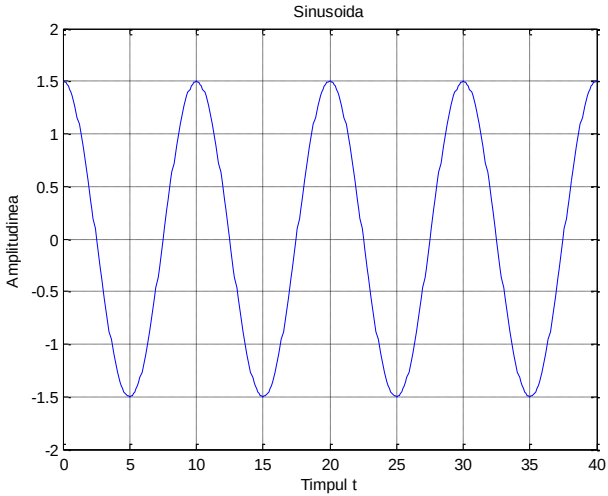


Fig. 2.11. Semnal sinusoidal

Formarea unui **semnal poliarmonic** în intervalul de timp $0 \leq t \leq 1s$ care constă din suma unei oscilaţii armonice cu amplitudinea de 1V şi frecvenţa 50Hz și a unei oscilaţii cu amplitudinea de 2V şi frecvenţa de 120Hz, afectat de un semnal de

zgomot cu distribuție normală, valoarea medie zero şi varianța 0.5V, folosind o frecvenţă de discretizare de 1000Hz, adică intervalul de discretizare T=0.001s.

Setul de comenzi necesar pentru crearea unui astfel de semnal este următorul:

```
t=0:0.001:1;
y=sin(2*pi*50*t)+2*sin(2*pi*120*t);
randn('state',0);
yn=y+0.5*randn(size(t));
plot(t(1:50), yn(1:50));grid
```

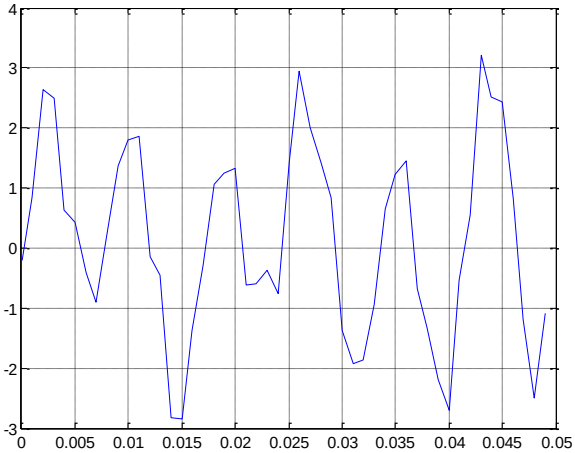


Fig.2.12. Semnal poliarmonic afectat de zgomot

Generarea unei **secvenţe periodice de impulsuri dreptunghiulare** se efectuează cu ajutorul procedurii `square`. Adresarea către ea are forma:

`y=square(t);`

`y=square(t,duty).`

Funcţia **`square(t)`** formează oscilaţii dreptunghiulare de perioadă 2π şi amplitudine ± 1.

Funcţia **y=square(w*t,duty)** formează o secvenţă de impulsuri dreptunghiulare cu durata semiundei pozitive, determinată de parametrul `duty` (coeficient de umplere) în procente faţă de perioadă şi frecvenţa unghiulară determinată de parametrul `w`. De regulă se ia `duty`=50.

```
t=-20:0.1:20;
y=square(t,50);
plot(t,y);
axis([min(t),max(t),-1.5,1.5]),grid
```

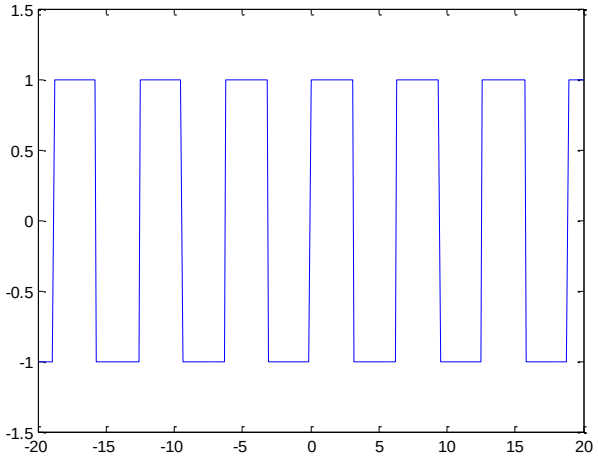


Fig.2.13. Secvenţă periodică de impulsuri rectangulare

Formarea **semnalelor "dinte de ferestrău" şi triunghiulare** de amplitudine ±1 şi perioadă 2π se efectuează prin procedura ***sawtooth***:

Funcţia **x=sawtooth(t)** formează semnalul de forma "dinte de ferestrău", care ia valoarea -l în momentele de timp multiple cu 2π şi crescând liniar pe intervalul 2π cu panta 1/π**.**

Funcţia **x=sawtooth(t,width)** formează semnalul "dinte de ferestrău" modificat. Parametrul `width` ia valori în limitele de la 0 la 1 şi determină o parte a perioadei în care semnalul creşte. Semnalul creşte de la -l până la 1, pe intervalul de la 0 până la

2π*width, iar pe urmă scade de la 1 până la -l pe intervalul de la 2π*width până la 2π. Dacă width=0.5, atunci se formează o undă simetrică.

Funcţia **sawtooth(w*t)** formează semnalul de forma "dinte de ferestrău" cu frecvenţa unghiulară determinată de parametrul w. Prezentăm un exemplu de generare a unui semnal de tip „dinte de ferestrău" pe intervalul 0-6π (fig. 2.14):

```
t=0:0.1:6*pi;
x=sawtooth(t);
plot(t,x); grid
```

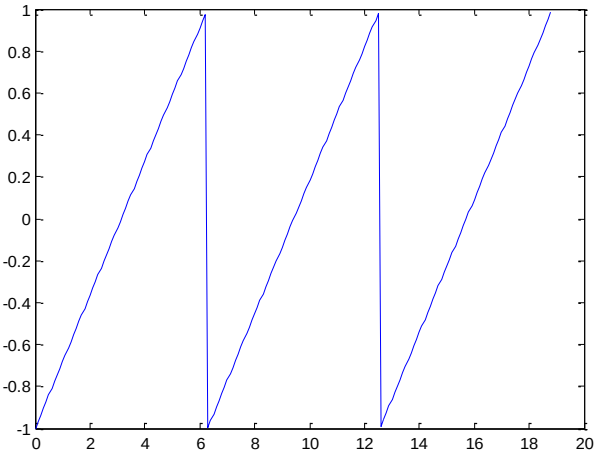


Fig.2.14. Secvenţă periodică de impulsuri "dinte de ferestrău"

Procedura pulstran permite a forma oscilaţii care sunt secvenţe ale impulsurilor rectangulare, triunghiulare sau gaussiene. Adresarea către ea are forma:

```
y=pulstran(t,d,'func');
y=pulstran(t,d,'func',p1,p2,...);
y=pulstran(t,d,p,Fs);
y=pulstran(t,d,p).
```

Aici parametrul d determină valoarea vectorului momentelor de timp corespunzătoare axei de simetrie a impulsurilor; parametrul func determină forma impulsului şi poate avea una

din următoarele sensuri: *rectpuls* (pentru impuls dreptunghiular), *tripuls* (pentru impuls triunghiular), *gauspuls* (sinusoidă modulată cu funcţia Gauss). Semnalul de ieşire *y* se calculează pentru valorile argumentului, date de vectorul *t*, după formula: *y=func(t-d(1))+func(t-d(2))+...* Numărul impulsurilor în diapazonul dat al argumentelor este *length(d)*. Parametrii *p1*, *p2*... determină parametrii necesari ai impulsului în funcție de forma de adresare către procedura, care determină acest impuls.

Funcţia **`y=pulstran(t,d,p,`$\mathbf{F_s}$`)`** permite a determina impulsul cu secvenţa eşantioanelor date în vectorul `p`. Frecvenţa de discretizare este dată de parametrul F_s. La folosirea funcţiei **`y=pulstran(t,d,p)`**, frecvenţa de discretizare se ia egală cu 1Hz. Mai jos sunt prezentate trei exemple de utilizare a procedurii `pulstran`:

Pentru o succesiune de impulsuri dreptunghiulare:

```
t=0:0.01:50;
d=[0:10:50];
y=0.6*pulstran(t,d,'rectpuls',5);
plot(t,y), grid
```

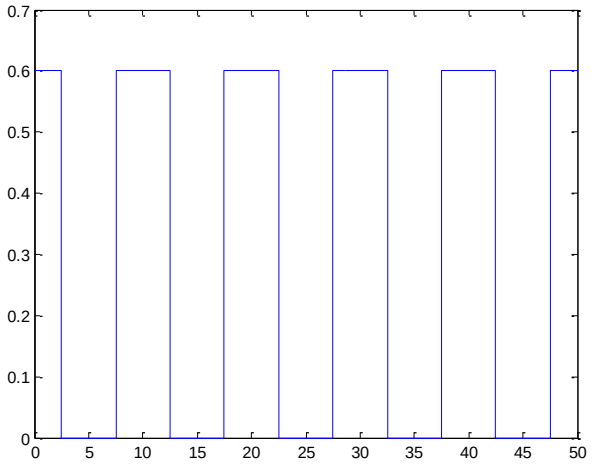


Fig. 2.15. Utilizarea funcţiei `pulstran` în cazul impulsurilor dreptunghiulare

Pentru o succesiune de impulsuri triunghiulare:

```
t=0:0.01:50;
d=[0:10:50];
yl=0.8*pulstran(t,d,'tripuls',5);
plot(t,yl),
grid
```

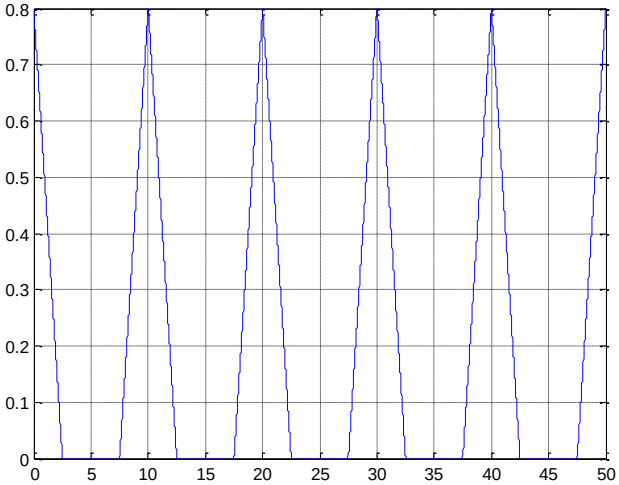


Fig. 2.16. Utilizarea funcţiei `pulstran`
pentru o succesiune periodică de impulsuri triunghiulare

Pentru o succesiune periodică de impulsuri gaussiene:

```
t=0:0.01:50;
d=[0:10:50];
y2=0.7*pulstran(t,d,'gauspuls',1,0.5);
plot(t,y2),
grid
```

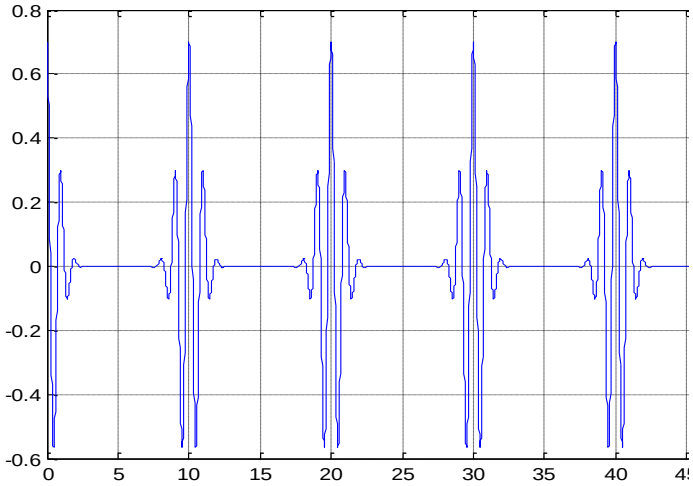


Fig. 2.17. Utilizarea funcţiei `pulstran`
pentru o succesiune periodică de impulsuri gaussiene.

O cosinusoidă, frecvenţa căreia variază liniar în timp, se generează cu ajutorul procedurii `chirp`:

`y=chirp(t,f0,t1,f1).`

Aceasta formează eşantioanele semnalului cosinusoidal, frecvenţa căruia creşte liniar pentru momentele de timp date de vectorul `t`; f_0 - frecvenţa instantanee în momentul de timp t=0; f_1 - frecvenţa instantanee în momentul de timp t=l. Frecvenţele f_0 şi f_1 se dau în Hz. De exemplu, pentru f_0=0, t_1=l, f_1=100, utilizăm următorul script:

```
t=0:0.001:1;
y=0.8*chirp(t);
plot(t,y);
grid
```

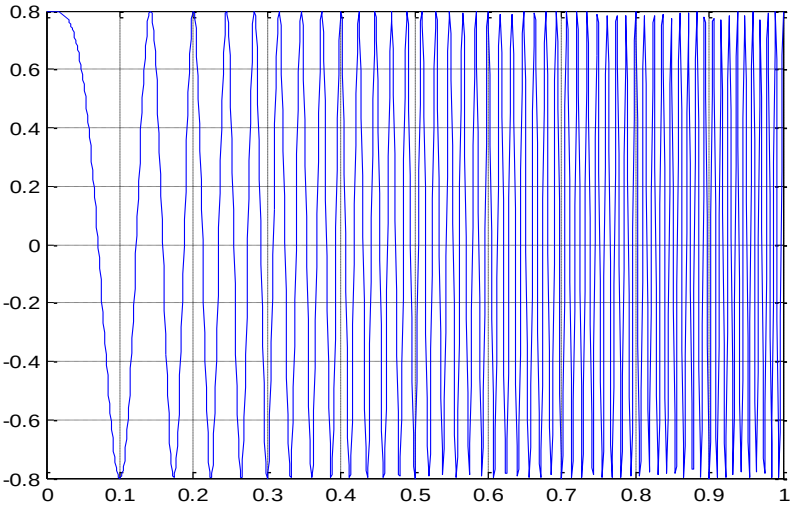


Fig. 2.18. Formarea unei cosinusoide cu frecvenţa variabilă în timp

2.4. Formarea semnalelor exponențiale

Unul dintre principalele tipuri de semnale elementare este **semnalul exponeţial**. Acest semnal poate fi generat cu ajutorul următorilor operatori MATLAB: .^ şi *exp*.

Funcția **`v=exp(Z)`** calculează exponenta mărimilor elementelor vectorului `Z`. Pentru mărimi complexe $z=x+iy$ este adevărată expresia Euler:

$$e^z=e^x(cos(y)+i\ sin(y)),$$

care reprezintă un semnal cosinusoidal, amplitudinea căruia variază după o exponentă.

Dacă elementele vectorului *Z* sunt mărimi reale, atunci se obține un semnal exponențial.

Calculul valorilor exponențiale ale elementelor unei matrice *Z* se realizează cu ajutorul unei funcții speciale **`expm(Z)`**.

Generarea unui semnal exponenţial complex poate fi efectuată cu ajutorul următorului script:

```
clf;
c=-(1/12)+i*(pi/6);
K=2;
t=0:0.1:40;
x=K*exp(c*t);
subplot(2,1,1);
plot(t,real(x));
xlabel('Timpul t');
ylabel('Amplitudinea');
title('Partea reala'); grid;
subplot(2,1,2);
plot(t,imag(x));
xlabel('Timpul t');
ylabel('Amplitudinea');
title('Partea imaginara'); grid
```

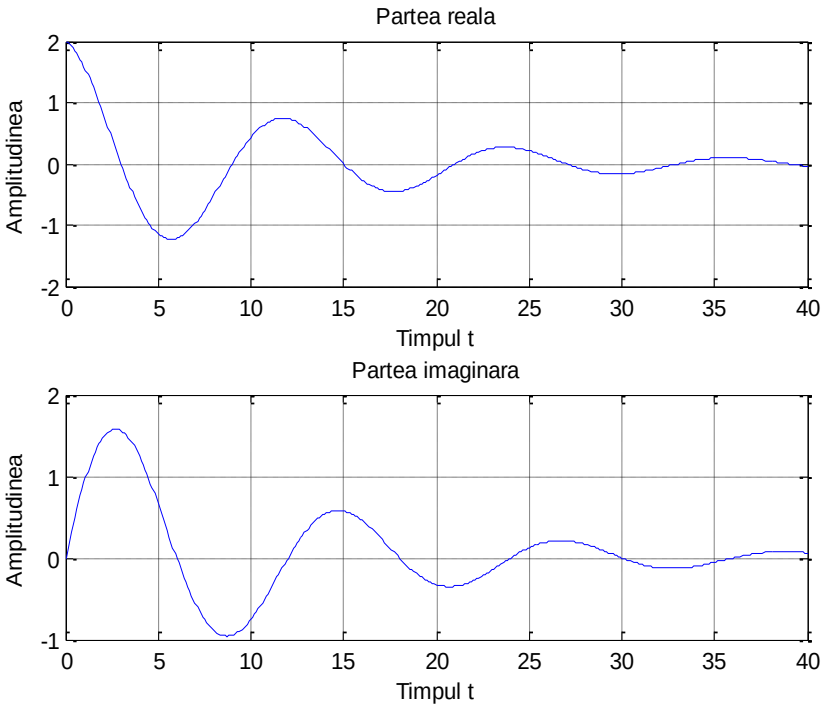


Fig. 2.19. Semnal exponenţial complex

Un **semnal sinusoidal, variind exponențial**, poate fi realizat și în urma utilizării următoarei funcții analitice:

$$y=e^{xt}\sin(\omega_0 t+\varphi_0).$$

Exemplu:

```
t=0:0.001:1.223;
T=0.001;
k=101:1224;
Z=zeros(1,100);
x=[Z,exp(-(k*T-0.1)/0.2).*sin(2*pi*(k*T-
0.1)/0.16)];
plot(t,x); grid on;
title('Proces tranzitoriu');
xlabel('Time(s)');
ylabel('Amplitudine')
```

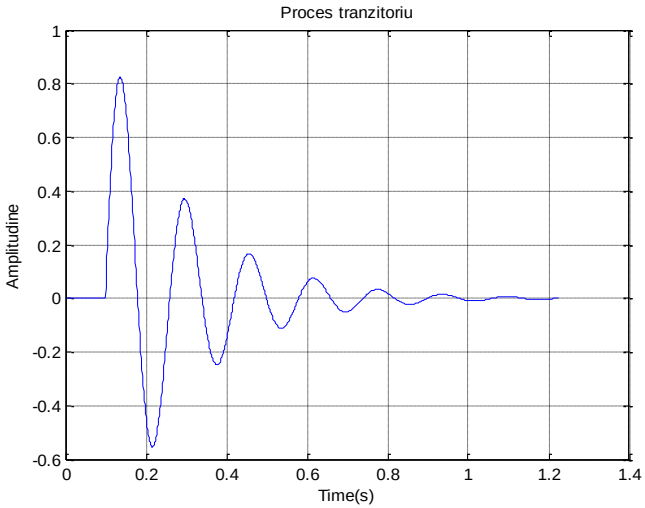


Fig. 2.20. Semnal sinusoidal variind exponențial

O secvență exponențială definită real se descrie prin următoarea funcție analitică $y=ka^x$.

Exemplu:

```
clf;
t = -10:10;
a = 1.2;
K = 2;
x = K*a.^t;
plot(t,x);
title('Secventa exponentiala reala');
xlabel('n');ylabel('Amplitudine');
grid
```

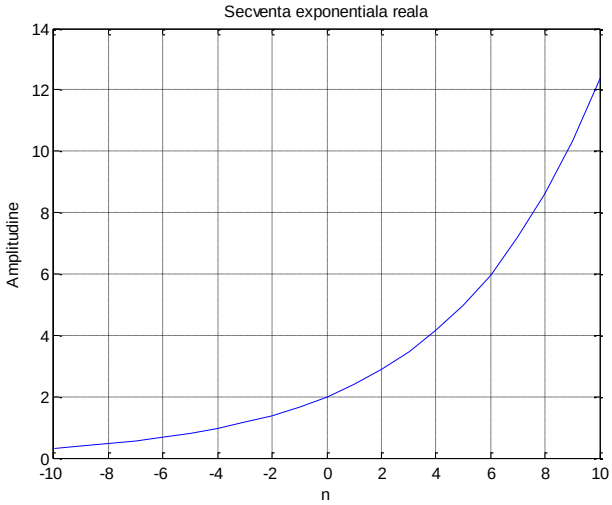


Fig. 2.21. Secvență exponențială reală

E 2.1. ***Exerciţii:***

1. Să se genereze şi să se reprezinte următoarele secvenţe.

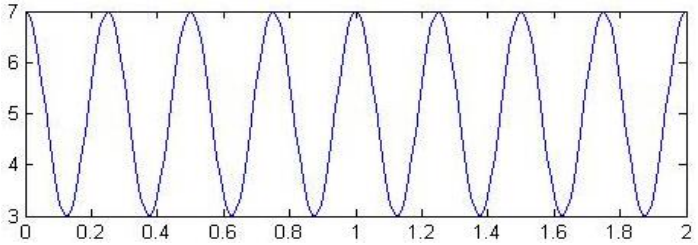
$x_1[n]=0.8\ \delta[n]$,	$-15 \le n \le 15$
$x_2[n]=0.9\ \delta\ [n-5]$,	$1 \le n \le 20$
$x_3[n]=1.5\ \delta\ [n-333]$,	$300 \le n \le 350$
$x_4[n]=4.9\ \delta\ [n+7]$,	$-10 \le n \le 0$
$x_5[n]=4\ u[n]$,	$-10 \le n \le 10$

$x_6[n]=1.4\ u[n-7]$, $-5\leq n\leq 20$

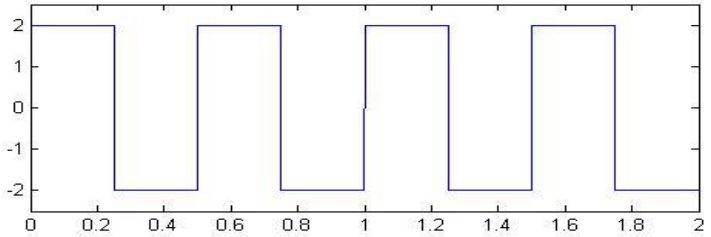
$x_7[n]=2.3\ u[n+5]$, $-15\leq n\leq 10$

Pentru trei cazuri să se construiască un tren de impulsuri unitare periodice.

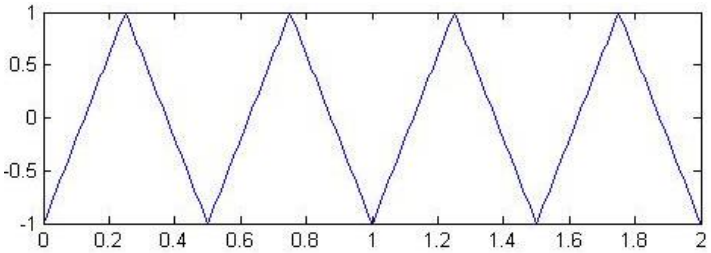
2. Să se genereze în MATLAB următorul semnal sinusoidal.



3. Să se genereze în MATLAB următorul semnal dreptunghiular.



4. Să se genereze în MATLAB următorul semnal triunghiular.



5. Să se creeze în intervalul $0s\leq t\leq 256s$ o oscilaţie armonică cu amplitudinea unitate, perioada T=50 *sec* și faza inițială $\pi/3$.

6. Să se creeze în intervalul $0s \leq t \leq 1s$ următoarele semnale exponenţiale:
   a) **5exp(-6t);**
   b) **exp(5t),**
   folosind frecvenţa rezoluției temporale $f_d = 1000Hz$.
7. Să se creeze în intervalul $-10s \leq t \leq 10s$ un impuls exponenţial descris de următoarea expresie $\boldsymbol{x(t)=Br^t}$, unde $B=1$, $r=0.8$.
8. Să se creeze în intervalul $-10s \leq t \leq 10s$ un semnal sinusoidal dat de expresia:
$$x(t) = 2sin\left(\frac{2\pi}{12}t\right).$$
9. Să se creeze un semnal sinusoidal atenuat, pe baza înmulţirii exponentei atenuatoare formate în punctul 7 și a semnalului sinusoidal creat în punctul 8, ambele obținute pentru intervalul $-10s \leq t \leq 10s$.
10. Să se creeze un impuls dreptunghiular de amplitudine unitate și durata $1s$, amplasat simetric față de originea de coordonate $t=0$ ($-0.5s \leq \tau \leq 0.5s$) descris în intervalul de timp $-1s \leq t \leq 1s$, utilizând rezoluția temporală $t=2ms$.
    Sugestie: impulsul dreptunghiular poate fi creat cu ajutorul diferenţei a două funcţii de tip "treaptă unitară" deplasate în timp cu un interval egal cu durata impulsului.
11. A forma o succesiune periodică de impulsuri dreptunghiulare cu amplitudinea $A=\pm 1$, viteza unghiulară $\omega=\pi/4$ și coeficientul de umplere 30% în intervalul $-10s \leq t \leq 10s$.
12. Pe intervalul $-10s \leq t \leq 10s$ să se reprezinte grafic în aceeași figură, în sisteme de coordonate diferite un semnal exponenţial complex $x(t)=\exp((-0.1+j0.3)t)$.

LUCRAREA nr. **3**

ANALIZA SPECTRALĂ A SEMNALELOR

***Obiective**: analiza spectrală a semnalelor periodice prin dezvoltare în serie Fourier şi a semnalelor neperiodice prin aplicarea transformatei Fourier.*

3.1. Introducere

Pentru a înțelege cât mai bine modul în care circuitele electronice prelucrează semnalele electrice (de tensiune sau de curent), este importantă în primul rând studierea semnalelor. Studierea semnalelor în domeniul timp, de exemplu cu ajutorul osciloscopului, poate fi completată cu o metodă absolut diferită: *studiul în domeniul frecvențelor (analiza spectrală).* Studiul în frecvenţă a semnalelor și circuitelor s-a dovedit a fi o abordare foarte productivă în electronică și automatizări. Metoda are ca punct de pornire constatarea că semnalele periodice pot fi descompuse în sume de componente armonice cu ajutorul dezvoltării în serie Fourier.

3.2. Semnale periodice. Serii Fourier

Semnale periodice sunt acele semnale pentru care

$$x(t) = x(t + kT), (\forall)\ t \in R,$$

unde k este un număr natural: k=0, 1, 2, …, iar T reprezintă perioada semnalului.

Un semnal periodic $x(t)$ de perioadă T poate fi dezvoltat în serie Fourier dacă satisface condițiile lui Dirichlet: să fie uniform, finit și să aibă un număr finit de discontinuități și de maxime pe durata unei perioade.

Seria Fourier are următoarele forme.

Forma trigonometrică

$$x(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty}[a_n \cos(n\omega_0 t) + b_n \sin(n\omega_0 t)], \quad (3.1)$$

unde $f_0 = \frac{1}{T} = \frac{\omega_0}{2\pi}$ reprezintă frecvenţa de repetiţie a semnalului periodic (frecvenţa fundamentală).

Frecvența de repetiție a unui semnal constituit dintr-o sumă de semnale sinusoidale se determină ca cel mai mic divizor comun al frecvențelor componentelor sinusoidale.

Coificienții a_0, a_n, b_n se calculează prin relațiile

$$a_0 = \int_{t_0}^{t_0+T} x(t)dt; \quad (3.2)$$

$$a_n = \frac{2}{T}\int_{t_0}^{t_0+T} x(t)\cos(n\omega_0 t)dt; \quad (3.3)$$

$$b_n = \frac{2}{T}\int_{t_0}^{t_0+T} x(t)\sin(n\omega_0 t)dt. \quad (3.4)$$

Forma armonică Este o formă mai compactă, utilizând numai funcţii cosinusoidale:

$$x(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} A_n \cos(n\omega_0 t + \varphi_n); \quad (3.5)$$

$$A_n = \sqrt{a_n^2 + b_n^2}\ ;\ \varphi_n = -arctg\frac{b_n}{a_n}. \quad (3.6)$$

Deci, seria Fourier armonică dă o descompunere a semnalului periodic *x(t)* într-o sumă de semnale cosinusoidale ale căror frecvenţe sunt multipli frecvenţei de repetiţie a semnalului periodic. Aceste componente se mai numesc armonici.

Forma complexă sau exponențială

$$x(t) = \sum_{n=-\infty}^{\infty} C_n e^{jn\omega_0 t}\ ; \quad (3.7)$$

$$C_n = \frac{1}{T}\int_{t_0}^{t_0+T} x(t)e^{-jn\omega_0 t}dt. \quad (3.8)$$

Coeficienţii C_n sunt mărimi complexe şi, deci, pot fi reprezentaţi prin modul şi fază:

$$C_n = C(n\omega_0) = |C_n| e^{j\phi_n}. \tag{3.9}$$

Relaţiile de legătură dintre coeficienţii C_n şi coeficienţii seriilor trigonometrică şi armonică sunt:

$$\begin{aligned} C_n &= C(n\omega_0) = \frac{a_n - jb_n}{2}; \\ C_{-n} &= C(-n\omega_0) = \frac{a_n + jb_n}{2}; \\ |C_n| &= |C_{-n}| = \frac{A_n}{2}; \quad C_0 = \frac{a_0}{2}; \\ \varphi_n &= \phi_n = -arctg\frac{b_n}{a_n}. \end{aligned} \tag{3.10}$$

Alegerea limitelor de integrare în evaluarea coeficienţilor seriilor Fourier este arbitrară, esenţial este ca integrarea să se se facă pe durata unei perioade (de la $-T/2$ la $T/2$ sau de la 0 la T etc.). Pentru semnale pare seria trigonometrică coincide cu cea armonică, deoarece $b_n = 0$. Pentru semnale impare, respectiv $a_0 = a_n = 0$.

Caracterizarea în domeniul frecvenţă a semnalului periodic se face prin reprezentarea spectrelor de amplitudini şi faze. Se pot reprezenta fie diagramele spectrale asociate seriei armonice, $A_n(n\omega_0)$ şi $\varphi_n(n\omega_0)$, fie diagramele spectrale asociate seriei complexe, $|C_n(n\omega_0)|$ şi $\phi_n(n\omega_0)$. Deoarece semnalele periodice sunt exprimate prin sume discrete de semnale elementare, rezultă că spectrele de amplitudini şi faze vor fi discrete. La spectrele asociate seriei armonice, liniile spectrale vor fi localizate la frecvenţele: $0, \omega_0, 2\omega_0, 3\omega_0, ...$ etc., în timp ce la spectrele asociate seriei exponenţiale liniile spectrale vor fi localizate la $0, \pm\omega_0, \pm 2\omega_0, \pm 3\omega_0, ...$ Semnificaţia frecvenţelor negative rezultă din funcţia:

$$2A\cos(\omega t) = Ae^{j\omega t} + Ae^{-j\omega t}.$$

Două componente exponenţiale, localizate simentric faţă de

axa ordonatelor, se însumează şi dau o componentă cosinusoidală de frecvenţă pozitivă. În figura 3.1 sunt reprezentate diagramele spectrale de amplitudine ale semnalului periodic *x(t),* determinate în conformitate cu *a)* SFA şi *b)* SFC.

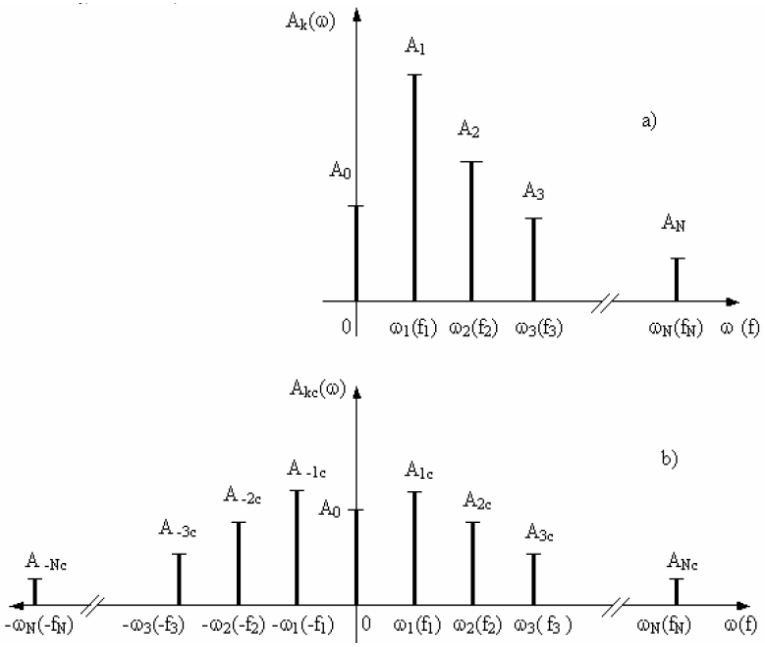


Fig.3.1. Diagramele spectrale de amplitudine ale semnalului *x(t) - a)* SFA, *b)* SFC

Teoretic, spectrele semnalelor periodice se întind de la $f = 0$ la $f = \infty$. Practic, spectrele sunt limitate din cauza descreşterii amplitudinilor componentelor, ceea ce permite limitarea seriei la un termen, începând cu care amplitudinea componentelor este neglijabilă. Trunchierea seriei la un anumit termen depinde de cerinţele impuse tipului de comunicaţie care utilizează semnalul respectiv. Prin urmare, analiza spectrală a unui semnal permite a stabili lăţimea benzii de frecvenţe efectiv ocupată de semnal.

3.3. Semnale neperiodice. Transformata Fourier

Din analiza formei spectrului semnalului periodic în funcție de perioada sa *T*, se observă că pe măsură ce *T* crește, componentele din spectrul semnalului se "îndesesc". Acest lucru este natural, întrucât creșterea lui *T* este echivalentă cu scăderea frecvenței fundamentale $\omega_0=2\pi/T$, deci, cu scăderea intervalului de frecvențe dintre două componente succesive. Evident, la limită, când $T\rightarrow\infty$, componentele frecvențiale se "contopesc", iar spectrul semnalului devine de natură continuă. Astfel, trecerea la limită, când $T \to \infty$, aduce relația (3.8) la forma:

$$F\{x(t)\} = X(\omega) = \int_{-\infty}^{+\infty} x(t)e^{-j\omega t}dt, \qquad (3.11)$$

cunoscută ca ***transformata Fourier directă în timp continuu*** a semnalului *x(t),* de data aceasta neperiodic. Integrala este convergentă dacă semnalul satisface condițiile Dirichlet pe orice interval finit.

Există și ***transformarea Fourier inversă***, analogă seriilor Fourier pentru semnalele periodice (3.7):

$$F^{-1}\{X(\omega)\} = x(t) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} X(\omega)e^{j\omega t}d\omega, \qquad (3.12)$$

care reconstituie semnalul în domeniul timp *x(t),* din exprimarea lui absolută, echivalentă informațional, din domeniul frecvențelor $X(\omega)$.

Atât pentru semnalele periodice, cât și pentru cele neperiodice, prin tratarea Fourier se obțin spectrele semnalelor, care sunt proiecții ale acestor semnale pe baze în spațiul semnalelor alcătuite din sinusoide.

Este important, pentru înțelegerea noțiunilor, să observăm similitudinile și diferențele dintre relațiile (3.11) și (3.12) și cele care descriu descompunerea în serie Fourier complexă a unui semnal periodic, respectiv (3.7) și (3.8). Se observă că semnificația valorilor $X(\omega)$ este similară cu cea a coeficienților C_n, cu singura diferență că, în cazul transformatei Fourier, numărul de

cosinusoide în care se descompune semnalul devine infinit nenumărabil. În rest, semnificația valorilor $X(\omega)$ este aceeași - modulul $|X(\omega)|$ și faza $\varphi(\omega)$ ale cantității complexe

$$X(\omega) = |X(\omega)| \exp(j\varphi(\omega)) \quad (3.13)$$

sunt amplitudinea, respectiv faza cosinusoidei de frecvență ω ce intră în descompunerea spectrală a semnalului $x(t)$. Într-adevăr, observând că, în ipoteza unui semnal $x(t)$ cu valori reale, valorile transformatei Fourier situate simetric față de 0 sunt complex conjugate:

$$X(-\omega) = X^*(\omega) \leftrightarrow \begin{cases} |X(-\omega)| = |X(\omega)|, \\ \varphi(-\omega) = -\varphi(\omega). \end{cases} \quad (3.14)$$

Transformata Fourier în timp discret (DTFT – *Discrete Time Fourier Transform*) a unei secvenţe $x[n]$ este dată prin relaţia:

$$X(e^{j\omega}) = \sum_n x[n] e^{-j\omega n}, \quad (3.15)$$

unde ω este pulsaţia normată:

$$\omega = 2\pi \frac{F}{F_s},$$

F este frecvenţa nenormată (exprimată în Hz), iar F_S - *frecvenţa de eşantionare.*

Similar, frecvenţa normată este:

$$f = \frac{F}{F_s}.$$

Funcţia $X(e^{j\omega})$ este periodică de perioadă 2π. Deci, este suficient să cunoaştem comportarea sa în intervalul $[-\pi, \pi)$ (*interval de bază*). Datorită faptului că această funcţie este continuă, variabila ω putând lua o infinitate de valori, nu este posibilă o implementare pe o maşină de calcul.

Pentru a realiza totuşi o analiză în frecvenţă se utilizează ***transformata Fourier discretă*** TFD (DFT – *Discret Fourier Transform*), obţinută prin discretizarea variabilei ω pe intervalul $[0, 2\pi)$ în N puncte:

$$\omega_k = 2\pi \frac{k}{N}, \; k = 0, 1, \ldots, N-1 \, .$$

Astfel, *transformata Fourier discretă* a unei secvenţe $x[n]$ este dată de relaţia:

$$X(k) = \sum_n x[n]\, e^{-j\frac{2\pi}{N}kn}. \qquad (3.16)$$

Există și TFD inversă, care permite a determina valorile eșantioanelor $x[n]$ din valorile coeficienților $X(k)$:

$$x[n] = \sum_n X(k)\, e^{j\frac{2\pi}{N}kn}. \qquad (3.17)$$

În figură 3.2 se reprezintă spectrul unui semnal în funcţie de pulsaţie sau frecvenţă normată şi corespondenţa cu frecvenţa analogică. De asemenea, se observă corespondenţa dintre componentele spectrale de indice k calculate cu TFD şi spectrul reprezentat în pulsaţii normate.

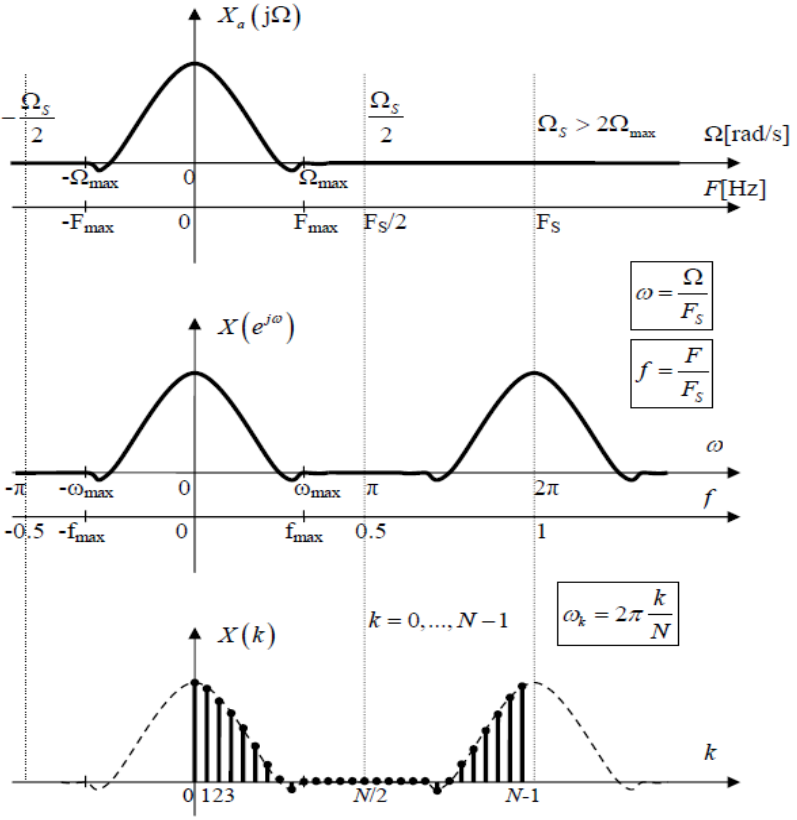


Fig.3.2. Spectrul unui semnal determinat în baza transformatei Fourier analogice în timp discret și a transformatei Fourier discrete

În MATLAB, pentru calculul transformatei Fourier discrete se foloseşte funcţia **`fft`**. Denumirea sa reprezintă prescurtarea de la *Fast Fourier Transform* (transformata Fourier rapidă) şi că foloseşte pentru calcul un algoritm rapid.

***Sintaxe*:**

`y = fft(x)`

- dacă `x` este un vector, se returnează un vector `y` de aceeaşi dimensiune cu vectorul `x` ce conţine valorile transformatei Fourier discrete aplicată elementelor vectorului `x`; lungimea transformatei Fourier (numărul de puncte *N* în care se calculează transformata) este egală în acest caz cu lungimea vectorului `x`;
- dacă `x` este o matrice se va returna matricea `y` de aceeaşi dimensiune cu matricea `x`; coloana *i* din matricea `y` va conţine valorile transformatei Fourier discrete aplicată elementelor coloanei *i* din matricea `x`.

`y = fft(x,N)`

- aceleaşi considerente ca în sintaxa precedentă cu deosebirea că în acest caz se specifică şi numărul de puncte *N* în care se calculează transformata. Dacă `length(x)<n`, elementele masivului `x` se completează cu zerouri. Dacă `lenglh(x)>n`, elementele în plus se şterg.

`x = ifft(y)`

- calculează transformata Fourier inversă consecutivităţii date în vectorul `y`. Dacă `y` este matrice, atunci fiecare coloană se transformă separat.

`x = ifft(y,N)`

- efectuează transformata Fourier inversă din N puncte. Dacă `length(y)<N`, elementele ce lipsesc ale masivului se completează cu zerouri. Dacă `length(y)>N`, elementele de prisos se elimină.

`y = fftshift(x)`

- efectuează regruparea masivului de ieşire al transformatei

Fourier pentru construirea graficelor speclrelor semnalelor în formă obişnuită, cu deplasarea frecvenţei nule în centrul spectrului.

Pentru a folosi procedura **fft** cu scopul de a obţine prezentarea semnalului în domeniul frecvenţă, reieşind din reprezentarea lui în domeniul timp, este necesar a efectua următorii paşi:

1. După valoarea aleasă a perioadei de eşantionare T_s, să se calculeze mărimea F_{max} a gamei de frecvenţe (în Hz) conform formulei:
$$F_{max}=1 / T_s.$$
2. În conformitate cu durata procesului T, să se calculeze `df` după formula:
$$df=l/T.$$
3. După valorile calculate să se formeze vectorul valorilor frecvenţelor în care se va calcula imaginea Fourier, de exemplu:
$$f=0:df:F_{max}$$
Ca rezultat al utilizării procedurii **fft** se va obţine valoarea vectorului y_k, adică TFD a succesiunii x_n dată în domeniul timp.

Pentru a obţine spectrul semnalelor în formă obişnuită, cu deplasarea frecvenţei nule în centrul spectrului, este necesar a mai efectua următorii paşi:

1. Rezultatele acţiunii procedurii **fft** se înmulţesc cu factorul T_s şi se supun acţiunii procedurii **fftshift,** care schimbă cu locurile prima şi a doua jumătate a vectorului obţinut.
2. Vectorul frecvenţelor se reconstruieşte după algoritmul:
$$f=-F_{max}/2:df:F_{max}/2.$$

Afişarea graficelor spectrelor discrete ale semnalelor periodice se recomandă a fi efectuată cu comanda **stem**, iar a spectrelor continue ale semnalelor neperiodice cu comanda **plot**.

3.4. Spectrele semnalelor periodice

***Exemplul* 3.1:**

Se propune descompunerea unui semnal periodic *s(t)* în serie Fourier armonică. Se vor analiza mai multe semnale:

- semnale armonice și combinații liniare de semnale armonice:
- semnale dreptunghiulare;
- semnale de tip „dinte de ferestrău".

La execuţie, programul cere următoarele argumente: *T* - perioada, *N* - numărul de armonici pentru aproximare, precum şi tipul semnalului *s(t)*: sinusoidal, dreptunghiular sau „dinte de ferestrău". Timpul este modelat printr-un vector de dimensiunea 1024.

Rezultatele se obțin prin intermediul a patru diagrame:

- semnalul *s(t)*;
- descompunerea în armonici;
- fazele fiecărei armonici;
- recompunerea semnalului *s(t)* prin însumarea armonicilor calculate. Această diagramă permite compararea semnalului recompus cu cel inițial și validarea rezultatelor obținute.

```
%Descompunerea unui semnal periodic s(t) in
serie Fourier:
%T=perioada [sec], N=nr. de armonici
T = input('Setati perioada T [sec]: ');
N = input('Setati nr. de armonici: ');
tip = input('Alegeti tipul semnalului (sin[s],
dreptunghiular[d], sau ferestrau[f]): ', 's');
W=2*pi/T;              %pulsatia fundamentala
t=0:T/1022:T+T/1022;
if strcmp(tip,'s')
    s=sin(W*t);        % semnal s(t) sinusoidal
else
    for j=1:1024
        if strcmp(tip,'d')
```

```
            if j<512  %semnal dreptunghiular
                s(j)=1;
            else
                s(j)=-1;
            end
        elseif strcmp(tip,'f')
               %semnal dinte de ferestrau
               s(j)=j/500-1;
        end
    end
end
%valoarea medie
val_medie=trapz(t,s)/T;
%valoarea efectiva
val_efectiva=sqrt(trapz(t,s.^2)/T);
timp=t-T/2;
for i=1:N
    %primii N coef. trigonometrici
    a(i)=2*trapz(t,s.*cos(i*W*t))/T;
    b(i)=2*trapz(t,s.*sin(i*W*t))/T;
    %coeficientii formei armonice
    A(i)=sqrt(a(i)^2+b(i)^2);
%defazajele formei armonice
    F(i)=atan2(b(i),a(i));
    f(i)=i/T;
end
r=val_medie;
for j=1:N
    r=r+A(j)*cos(j*W*t-F(j));
end
subplot(223); plot(t,r);
title('semnalul reconstruit (verificare)');
xlabel('t [sec]');
axis([min(t) max(t) (min(r)-0.02*(max(r)-
min(r))) (max(r)+0.02*(max(r)-min(r)))]);
grid;
subplot(221); plot(t,s);
title('semnalul s(t)'); xlabel('t [sec]'); grid;
```

```
axis([min(t) max(t) (min(r)-0.02*(max(r)-
min(r))) (max(r)+0.02*(max(r)-min(r)))]);
subplot(222); stem(f,A);
title('Armonicile A(n)*cos[n*2*pi*f*t-Fi(n)]');
xlabel('f [Hz]'); grid;
subplot(224); stem(f,F/(pi));
title('defazajele Fi(f)'); xlabel('f [Hz]');
ylabel('x pi   [rad]');
grid;
```

Pentru exemplificarea execuției programului au fost construite reprezentările grafice pentru semnalul exprimat printr-o succesiune de impulsuri dreptunghiulare de perioadă T=7s, aproximat prin 20 armonici (fig. 3.3).

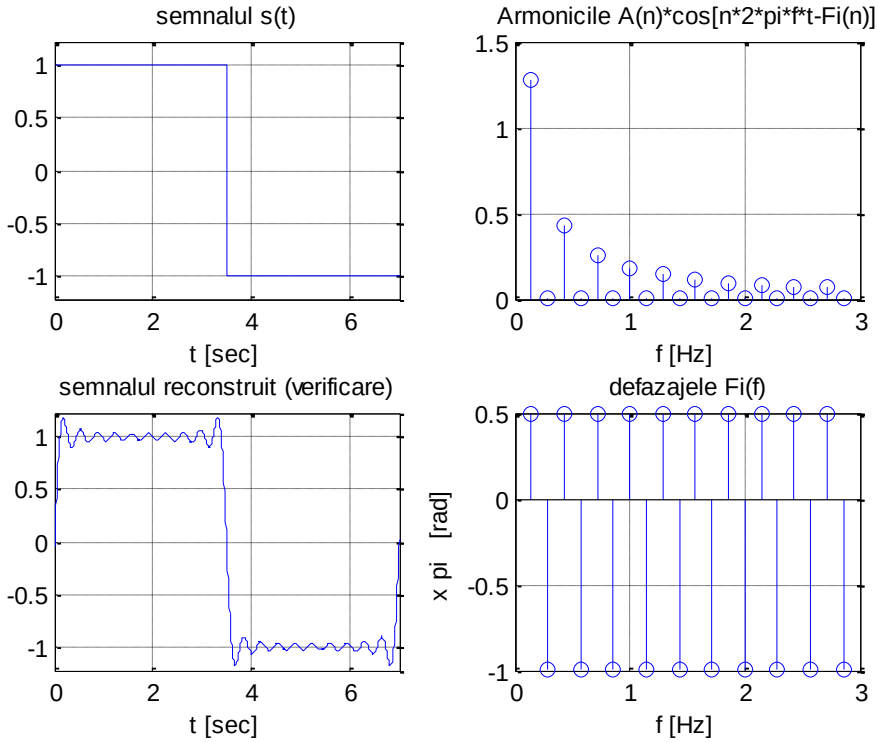


Fig.3.3. Descompunerea în serie Fourier a unui semnal dreptunghiular aproximat prin 20 de armonici

E3.1. ***Exerciţiu*:**

Utilizând scriptul din *Exemplul* 3.1, să se efectueze descompunerea în serie Fourier a trei tipuri de semnale periodice: armonic, dreptunghiular și dinte de ferestrău, pentru două valor ale numărului de armonici de aproximare N_1 și N_2, unde $N_2 \approx 3N_1$. Să se explice rezultatele obținute.

***Exemplul* 3.2:**

Se propune descompunerea unui tren de impulsuri dreptunghiulare în serie Fourier complexă.

La execuţie, programul cere introducerea următorilor parametri ai semnalului: perioada T, durata τ, amplitudinea A și numărul de armonici pentru aproximare N.

Rezultatele se obțin prin intermediul a trei diagrame:

- spectrul de amplitudine;
- spectrul de faze;
- recompunerea semnalului *x(t)* prin însumarea componentelor spectrale calculate.

```
%parametrii trenului de impulsuri
T = input('Setati perioada T [sec]: ');
tau = input('Setati durata impulsului tau [sec]:
');
Amplit = input('Setati amplitudinea A [V]: ');
Ni = input('Setati nr. de armonici N: ');
% Pasul de selectare a numarului de armonici
n=Ni;
% numarul de armonici pentru aproximarea finala
Nf=3*n;
w0=2*pi/T;
f0=1/T;
B=Nf+1;
% calculul parametrilor modelului spectral
A=zeros(1,B);phi=zeros(1,B);
for i=1:B,
      alf=(i-1)*w0*tau/2;
```

```
      alf=alf/pi;
      A(1,i)=abs(Amplit*tau*sinc(alf)/T);
      phi(1,i)=-angle(sinc(alf));
end;
%se calculeaza vectorul ind, necesar in
reprezentarea grafica a spectrului
for i=1:B,
      ind(i)=(i-1)*f0;
end;
%reprezentarea spectrului SFC (numai pentru
frecvente pozitive)
subplot(221);
stem(ind,A(1,:));
title('spectrul SFC al trenului de impulsuri');
xlabel('f [Hz]');
grid;
subplot(222);
stem(ind,phi(1,:));
title('defazajele Fi(f)');
xlabel('f [Hz]'); ylabel('x pi  [rad]');
grid;
%generarea trenului de impulsuri si
reprezentarea lui grafica
x1=zeros(1,((T*1000/2)-(tau*1000/2)));
x2=Amplit*ones(1,(tau*1000));
x3=zeros(1,((T*1000/2)-(tau*1000/2)));
x=[x1 x2 x3];
dt=0.001;t=[-T/2+dt:dt:T/2];
subplot(223);
h=plot(t,x); %set(h,'LineWidth',T);
axis([-T/2 T/2 -1.5 1.2*Amplit]);grid;hold on;
%calculul semnalelor deduse pe baza spectrului
determinat
%se utilizeaza Ni, 2*Ni si 3*Ni armonici in
spectru;
%aceste semnale se reprezinta pe un grafic comun
%cu cel al trenului de impulsuri
for j=Ni:n:Nf,
    xy=A(1)*ones(1,(T*1000));
```

```
    for i=1:j,
        xy=xy+2*A(1,i+1)*cos(i*w0*t+phi(1,i+1));
    end;
end;
plot(t,xy,'k');grid;
title('semnalul initial si reconstruit');
xlabel('t [sec]');
axis([-T/2 T/2 -1.5 1.2*Amplit]);grid;
```

Pentru exemplificarea execuției programului au fost construite reprezentările grafice pentru semnalul exprimat printr-o succesiune de impulsuri dreptunghiulare de perioadă T=3s, durata τ=0,5s, amplitudinea A=5V (fig. 3.4).

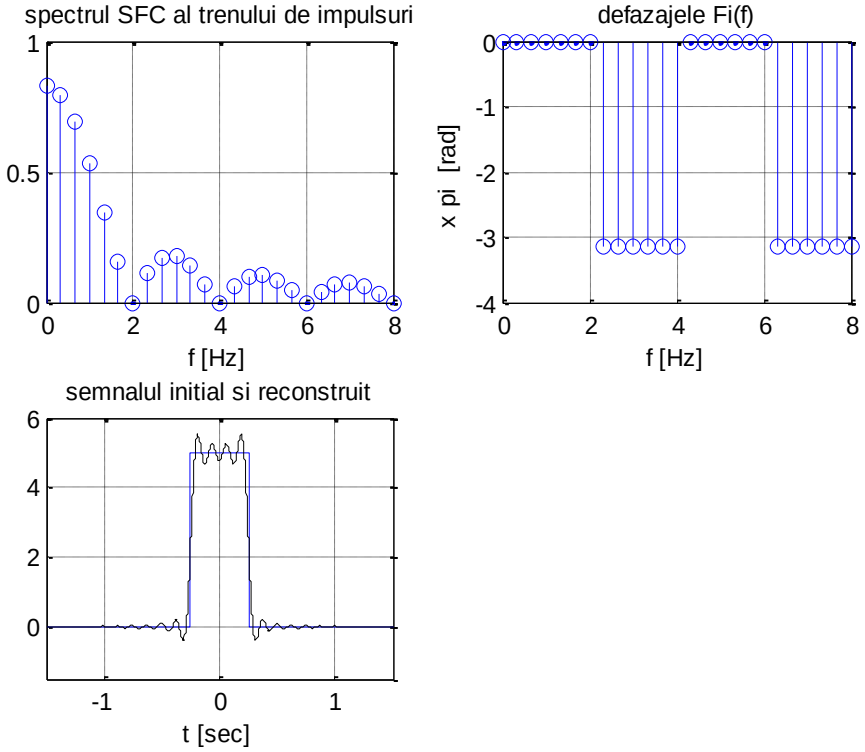


Fig.3.4. Descompunerea în serie Fourier complexă a unui tren de impulsuri dreptunghiulare

E3.2. ***Exerciţiu*:**

Utilizând scriptul din *Exemplul* 3.2, studiaţi spectrul unui tren de impulsuri dreptunghiulare pentru diverse valori ale parametrilor semnalului: perioada *T*, durata τ și amplitudinea *A*. Să se analizeze și să se explice rezultatele obținute.

***Exemplul* 3.3:**

Să se analizeze spectrul semnalului poliarmonic, alcătuit din trei oscilaţii armonice cu frecvenţele 1, 2 şi 3 Hz şi amplitudinile respectiv 3, 1 şi 4:

$$y(t) = 3\cos(2\pi t) + \sin(4\pi t) + 4\cos(6\pi t).$$

Să se afișeze graficele semnalului, spectrului de amplitudine (modulul spectrului) şi a părţilor reale şi imaginare ale spectrului.

Soluție:

```
Ts=0.01;T=10; t=0:Ts:T;
y=3*cos(2*pi*t)+sin(4*pi*t)+4*cos(6*pi*t);
subplot(211); plot(t,y); grid
df=1/T; Fm=1/Ts; len=length(t);
f=-Fm/2:df:Fm/2;
x=fft(y)/len;
xs=fftshift(x);
A=abs(xs);
s1=len/2-50; s2=len/2+50;
subplot(212); stem(f(s1:s2), A(s1:s2)); grid
xlabel('frecventa(Hz)'); ylabel('Modulul')
```

Graficele semnalului şi spectrului de amplitudine sunt reprezentate în figura 3.5.

Pentru construcţia spectrului de faze este suficient ca să înlocuim operatorul calculului modulului **`A=abs(xs)`** cu operatorul calculului unghiului de fază **`P=angle(xs)`**, cu schimbările corespunzătoare în operatorii de afişare a graficelor `stem` şi `ylabel`.

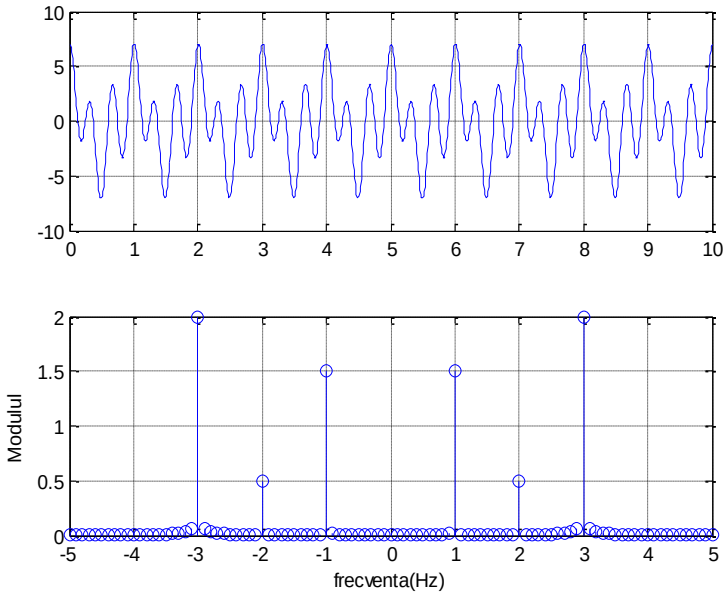


Fig.3.5. Modulul spectrului semnalului poliarmonic corespunzător seriei Fourier complexe

Vom evidenţia partea reală şi cea imaginară ale spectrului compiex:

```
Ts=0.01;T=10; t=0:Ts:T;
y=3*cos(2*pi*t)+sin(4*pi*t)+4*cos(6*pi*t);
subplot(311); plot(t,y); grid
df=1/T; Fm=1/Ts; len=length(t);
f=-Fm/2:df:Fm/2;
x=fft(y)/len;
xs=fftshift(x);
Re=real(xs); Im=imag(xs);
s1=len/2-50; s2=len/2+50;
subplot(312)
plot(f(s1:s2),Re(s1:s2));grid
ylabel('Partea reala')
subplot(313)
plot(f(s1:s2),Im(s1:s2));grid
ylabel('Partea imaginara')
```

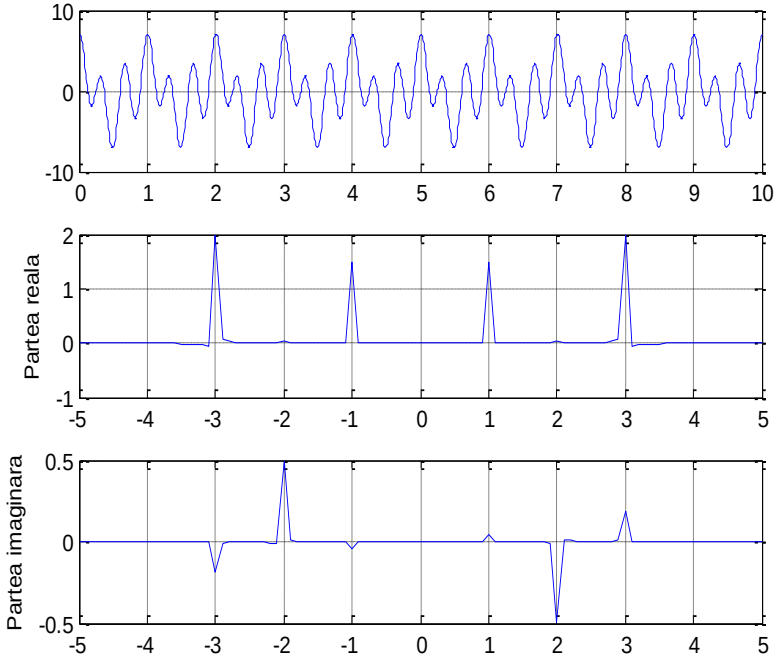


Fig. 3.6. Partea reală şi cea imaginară ale spectrului complex al semnalului poliarmonic.

E3.3. ***Exerciţiu:***

Să se calculeze și să se construiască caracteristicile spectrale de amplitudini și de faze ale unor semnale periodice, recomandate de profesor, pentru diverse valori ale parametrilor ce caracterizează aceste semnale. Să se analizeze și să se explice rezultatele obținute.

3.5. Spectrele semnalelor neperiodice

***Exemplul* 3.4:**

Scriptul următor calculează transformata Fourier analogică, utilizând expresia (3.11) a semnalului impuls rectangular, simetric față de origine, de durată `tau` și de arie unitară:

```
%parametrii impulsului rectangular
tau=1;Amplit=1/tau;
%generarea impulsului rectangular
a=tau;
tm=6;
x1=zeros(1,((tm*1000/2)-(tau*1000/2)));
x2=Amplit*ones(1,(tau*1000));
x3=zeros(1,((tm*1000/2)-(tau*1000/2)));
x=[x1 x2 x3];
dt=0.001;t=[-tm/2+dt:dt:tm/2];
subplot(211);
h=plot(t,x); %set(h,'LineWidth',T);
title('impuls rectangular x(t)');
xlabel('t [sec]');
axis([-tm/2 tm/2 -0.1 1.2*Amplit]);
grid;hold on;
% sunt declarate variabilele simbolice
syms x w
% se calculeaza integrala Fourier
wmax=30;
int(Amplit*exp(-j*w*x),-a/2,a/2);
subplot(212);
% se reprezinta grafic
ezplot(ans,[-wmax wmax])
title('transformata Fourier a impulsului
rectangular X(w)');
xlabel('w');
axis([-wmax wmax -0.5 1]);
grid;hold on
u=-wmax:wmax:wmax;
y=0.0*u;
% se traseaza orizontala y=0
plot(u,y)
hold off;
```

Reprezentările grafice ale semnalului şi funcția de densitate spectrală corespunzătoare sunt date în figura 3.7.

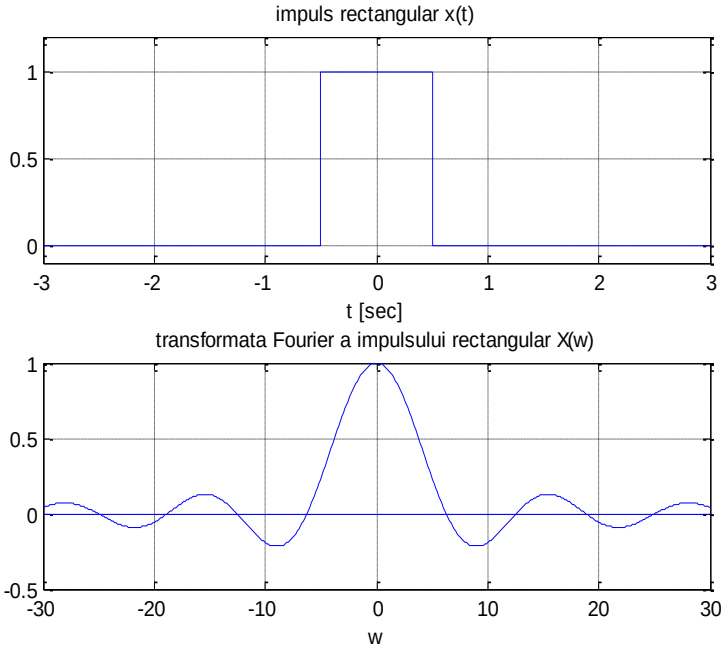


Fig. 3.7. Forma de undă și spectrul de frecvențe ale semnalului impuls dreptunghiular

***Exemplul* 3.5:**

Utilizând procedura **`fft`**, să se construiască funcția de densitate spectrală a unui semnal impuls unitar dreptunghiular centrat în originea de coordinate, cu amplitudinea A=0.8 şi durata w=0.5*s*. Aplicăm pasul de discretizare T_s=0.01*s*, iar durata analizei T= 1*s*:

```
%Generarea semnalului impuls unitar
dreptunghiular
Ts=0.01; T=1; A=0.85; w=0.5;
N=T/Ts;t=-T/2:Ts:T/2;
```

```
y=A*rectpuls(t,w);
subplot(311); plot(t,y); grid;
title('Impuls unitar dreptunghiular');
xlabel('Timpul,sec.');
%Aplicarea procedurii fft
x=fft(y)/N; df=1/T; Fm=1/Ts;
a=abs(x);f=0:df:Fm;
subplot(312); plot(f,a);grid;
title('Functia de densitate spectrala
(procedura fft)');
xlabel('Frecventa,Hz');
ylabel('Modulul')
%Aplicarea procedurii fftshift
xp=fftshift(x);
a=abs(xp);f1=-Fm/2:df:Fm/2;
subplot(313);plot(f1,a),grid;
title('Functia de densitate spectrala
(procedura fftshift)');
xlabel('Frecventa,Hz');
ylabel('Modulul')
```

Reprezentările grafice ale semnalului şi spectrului de amplitudini sunt date în figura 3.8.

Pentru construcţia spectrului de faze este suficient ca să înlocuim operatorul calculului modulului **`abs(.)`** cu operatorul calculului unghiului de fază **`angle(.)`**, cu schimbările corespunzătoare în operatorii de afişare a graficelor `stem` şi `ylabel`.

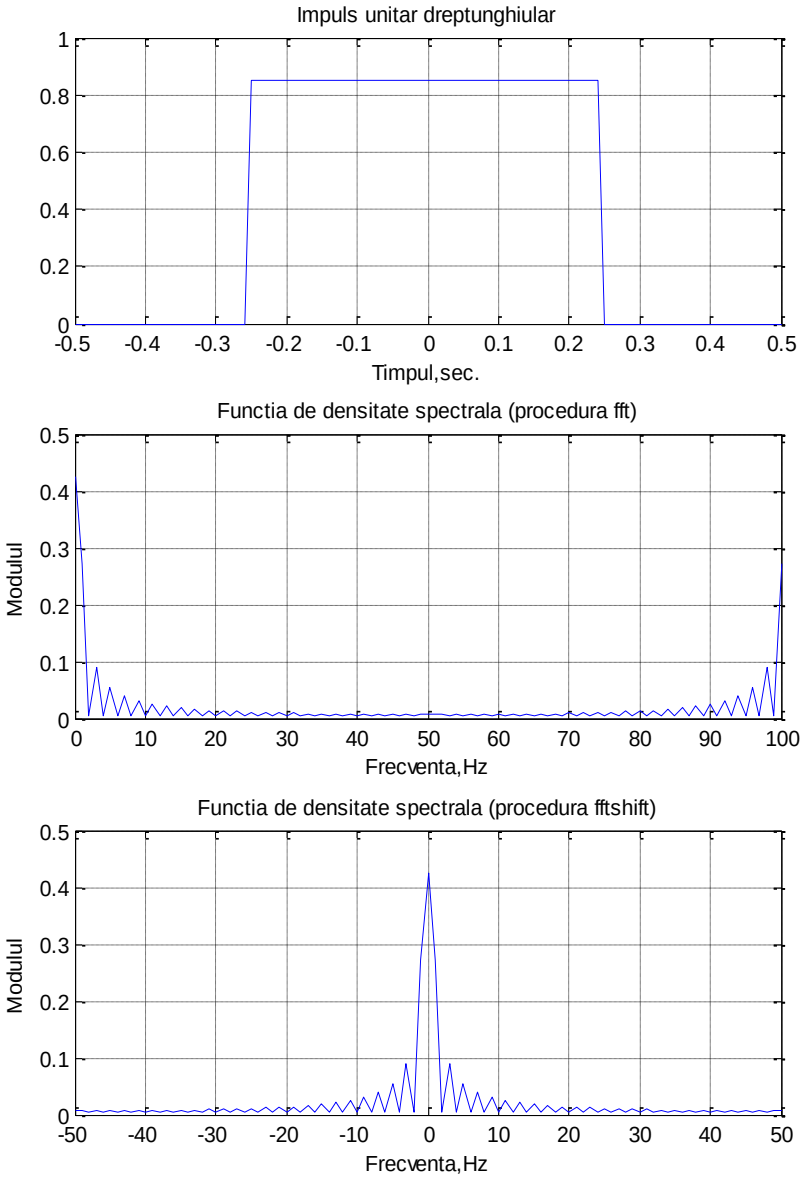


Fig. 3.8. Forma de undă și spectrul de frecvențe ale semnalului impuls dreptunghiular

E3.4. ***Exerciţiu:***

Să se calculeze și să se construiască caracteristicile spectrale de amplitudini și de faze ale unor semnale neperiodice, recomandate de profesor, pentru diverse valori ale parametrilor ce caracterizează aceste semnale și, de asemenea, pentru cazul deplasării semnalului în timp și în frecvență. Să se analizeze și să se explice rezultatele obținute.

Întrebări de control

1. Cum apare noţiunea de frecvenţă negativă?
2. Cum se schimbă spectrul unei succesiuni periodice de impulsuri la schimbarea perioadei impulsurilor?
3. Cum se schimbă spectrul unei succesiuni periodice de impulsuri la schimbarea duratei impulsurilor?
4. Ce caracter are spectrul semnalului neperiodic?
5. Ce particularităţi are densitatea spectrală a semnalului real?
6. Care este legătura dintre durata impulsului şi lărgimea spectrului său?
7. Cum se calculează amplitudinile şi fazele spectrelor semnalelor neperiodice?
8. În ce constă asemănarea şi deosebirea spectrelor semnalelor discrete şi analogice?
9. În ce constă şi cum se manifestă suprapunerea spectrelor la discretizarea semnalului?

LUCRAREA nr. **4**

EŞANTIONAREA ŞI CUANTIZAREA SEMNALELOR. INTERPOLAREA SEMNALELOR EŞANTIONATE

***Obiective**: înțelegerea conceptelor de eșantionare, cuantizare, de zgomot de cuantizare și interpolare a semnalelor eșantionate.*

4.1. Semnale analogice și procesarea digitală

Semnalele analogice sunt semnale continue ce pot fi reprezentate din punct de vedere matematic ca funcții reale continue în timp. Pentru a putea procesa un semnal cu ajutorul unui procesor de uz general sau cu a unuia specializat, semnalele continue trebuie convertite într-un format potrivit cu reprezentarea datelor dintr-un procesor.

Pentru conversia unui semnal analogic într-un semnal digital, trebuie realizate două operații de bază: discretizarea în timp și discretizarea în amplitudine a semnalului.

Discretizarea în timp poartă numele de eșantionare și se realizează cu ajutorul unui bloc de eșantionare-memorare (vezi figura 4.1). Discretizarea în amplitudine se numește cuantizare și este realizată cu un circuit de tip convertor analog-digital (CAD).

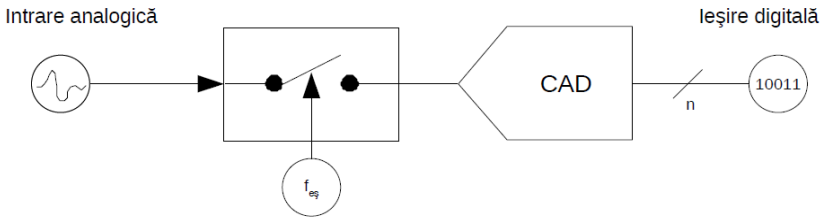


Fig. 4.1. Discretizarea semnalelor analogice

Circuitul de eșantionare-memorare va preleva valorile semnalului de intrare în anumite momente discrete de timp, iar

convertorul analog-digital va converti aceste valori reale în reprezentarea lor numerică.

4.2. Eșantionarea

Eșantionarea reprezintă discretizarea în timp a unui semnal continuu. În urma eșantionării, semnalul este definit doar în anumite momente ale domeniului de definiție, iar amplitudinea poate lua orice valoare reală în domeniul de valori. Un eșantion reprezintă valoarea semnalului la un moment dat, bine precizat.

Rezultatul eșantionării unui semnal continuu $x(t)$ este un șir de eșantioane $x[nT_s]$, unde T_s este intervalul de timp dintre două eșantioane și se numește perioadă de eșantionare, în cazul unei eșantionări uniforme (figura 4.2).

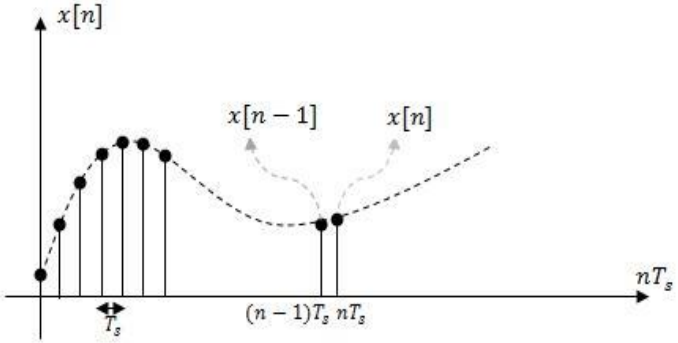


Fig. 4.2. Semnal eșantionat cu perioada de eșantionare T_s

Teorema eşantionării uniforme (Shannon) stabileşte că un semnal de bandă limitată este univoc determinat de valorile sale considerate la momente de timp echidistante, dacă distanţa T_s dintre două momente succesive satisface relaţia:

$$T_s \leq \frac{1}{2 f_m},$$

unde f_m reprezintă frecvenţa maximă din spectrul semnalului.

Frecvența de eșantionare $f_s = 1/T_s$ reprezintă frecvența cu care sunt prelevate eșantioanele semnalului analogic. Pentru ca un

semnal să poate fi refăcut complet din eșantioanele sale, conform teoremei eșantionării, trebuie satisfăcut criteriul Nyquist și anume că frecvența de eșantionare trebuie să fie cel puțin dublul frecvenței maxime f_m din spectrul semnalului:

$$f_s \geq 2 \cdot f_m \tag{4.1}$$

O condiție suplimentară constă în faptul ca spectrul semnalului să fie mărginit (de bandă limitată), de forma celui din figura 4.3.

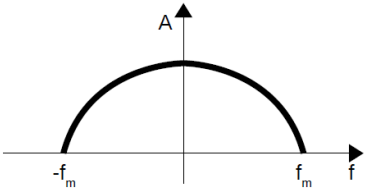


Fig. 4.3. Exemplu de spectru mărginit

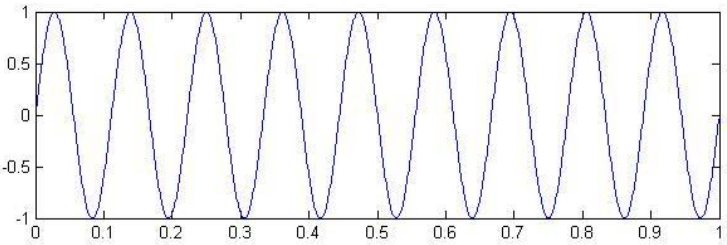
Observații:

- dacă dorim ca semnalul eșantionat să se apropie cât mai mult de semnalul real, atunci trebuie să luăm un număr cât mai mare de eșantioane, deci, să creștem frecvența de eșantionare f_s. Dezavantajul constă în spațiul mare de memorie ocupat;
- dacă dorim un număr mai mic de eșantioane (spațiu de memorie ocupat mai mic, dar și o posibilă pierdere de informație), atunci putem scădea frecvența de eșantionare f_s. Însă pentru a putea reface semnalul real din eșantioanele sale, frecvența de eșantionare nu poate fi scăzută oricât, ea trebuie să respecte teorema eșantionării.

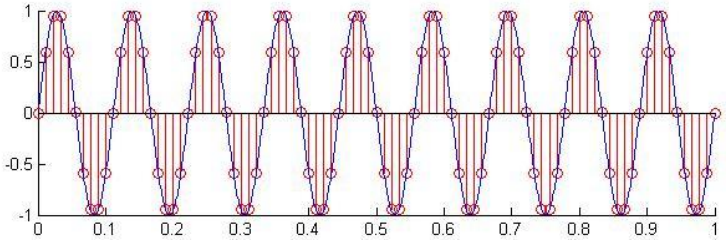
***Exemplul* 4.1:**

Fie un semnal sinusoidal $x(t)$ având frecvența f = 9Hz (*figura 4.4, a*). Acest semnal a fost eșantionat cu f_{s1}=90Hz (*figura 4.4, b*). Se observă că din eșantioanele rezultate poate fi refăcut $x(t)$. În *figura 4.4, c*, semnalul $x(t)$ a fost eșantionat cu f_{s2} = 8Hz (nu se respectă teorema eșantionării) și se poate observa că aceleași eșantioane rezultă atât prin eșantionarea semnalului $x(t)$, cât și prin eșantionarea unei alte sinusoide cu frecvență mai mică, de 1Hz. În

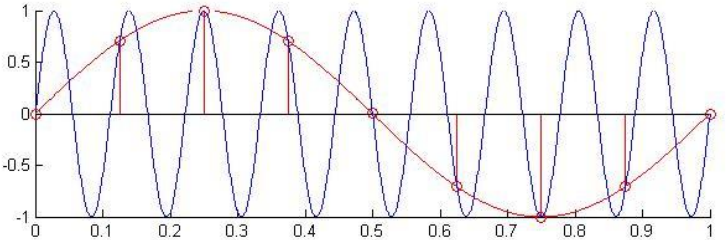
acest caz, semnalul $x(t)$ nu mai poate fi refăcut, formula de reconstituire generând sinusoida de 1Hz.



a) semnalul $x(t)$ cu frecvența f = 9Hz



b) semnalul $x(t)$ eșantionat cu f_{s1} = 90Hz



c) semnalul $x(t)$ eșantionat cu f_{s2} = 8Hz

Fig. 4.4. Semnalul $x(t)$ (*figura 4.4, a*), eșantionat respectând teorema eșantionării (*figura 4.4, b*) și eșantionat nerespectând teorema eșantionării (*figura 4.4, c*).

Șirul de eșantioane este echivalent unui vector în MATLAB, ale cărui elemente sunt valorile semnalului luate la momentele de timp corespunzătoare eșantionării. Trebuie menționat că informația despre frecvența de eșantionare nu este stocată în vector, ci numai se presupune cunoscută pentru calcule. Astfel, orice vector din

MATLAB poate fi considerat un semnal eșantionat presupunând că i se face corespondența cu un set de parametri de eșantionare.

Pentru a genera un semnal știind funcția care descrie semnalul analogic, se va genera în prealabil un vector de timp care reprezintă momentele de timp la care se va realiza practic eșantionarea:

```
t = 0 : 1/fs : 1;
x = sin (2*pi*f*t);
```

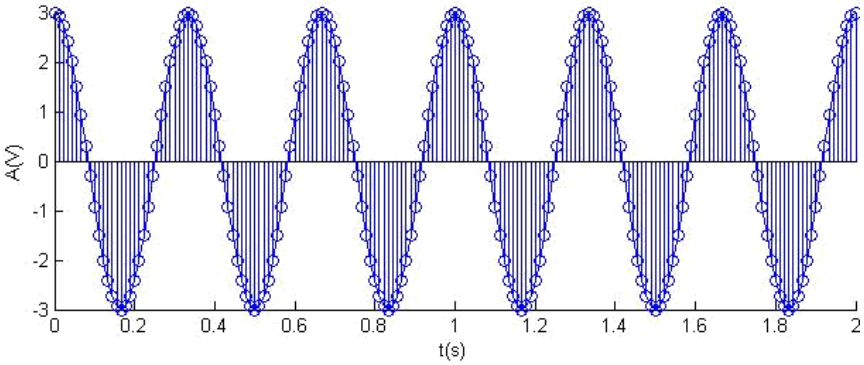
În acest caz, *x* va conține eșantioanele unui semnal sinusoidal, dar pentru a cunoaște semnalul căruia acest vector corespunde, trebuie să cunoaștem unitatea metrică pentru *t* și frecvența de eșantionare f_s. Pentru reprezentarea grafică a unui semnal discret se utilizează funcţia `stem`.

E4.1. ***Exerciţiu:***

1. Reprezentați grafic 1024 de eșantioane ale unui semnal alcătuit din 2 sinusoide (una cu frecvența de 50 Hz, defazajul 0 și amplitudinea 0.5 V, iar cealaltă cu frecvența de 230 Hz, defazajul $\pi/3$ și amplitudinea 0.2 V), folosind o frecvență de eșantionare de 8 kHz.

2. Să se genereze în MATLAB semnalul din figura de mai jos.

Indiciu: pornind de la figură, trebuie să se identifice toți parametrii sinusoidei (amplitudine, frecvență, frecvență de eșantionare, durată, fază inițială). Frecvența f_s a fost aleasă astfel, încât să fie 30 de eșantioane într-o perioadă.



3. Fie semnalul:

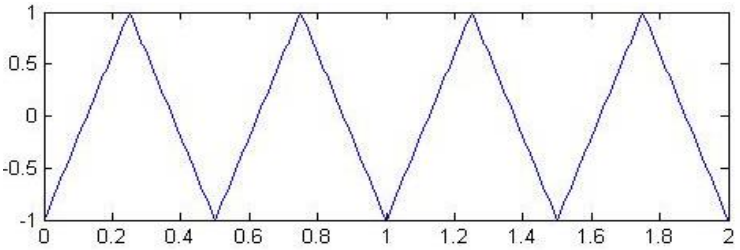
$$x(t) = 10\sin\left(200\pi t + \frac{\pi}{2}\right) + 20\sin(100\pi t) - 40\sin\left(300\pi t - \frac{\pi}{4}\right).$$

Cerințe:

- care este frecvența minimă de eșantionare astfel încât să se respecte teorema eșantionării;
- alegând o frecvență de eșantionare de 10 ori mai mare decât cea determinată la punctul anterior, să se eșantioneze semnalul $x(t)$ și să se reprezinte graphic;
- care este frecvența de repetiție a semnalului $x(t)$?

4. Fie semnalul sinusoidal $x(t) = 3\sin(2\pi \cdot 5t)$. Să se eșantioneze acest semnal cu f_{s1} = 4Hz și cu f_{s2} = 50Hz și să se reprezinte grafic. În care dintre cele două cazuri poate fi reconstituit semnalul $x(t)$ din eșantioanele sale?

5. Să se genereze și să se prezinte în MATLAB, în două ferestre separate, semnalul din figura următoare și semnalul discretizat.



Indiciu: pornind de la figură, trebuie să se identifice toți parametrii semnalului triunghiular (amplitudine, frecvență, frecvență de eșantionare, durată). Frecvența f_s se alege astfel încât să fie 30 de eșantioane într-o perioadă.

4.3. Cuantizarea

În urma eșantionării unui semnal continuu se obține un semnal definit doar în anumite momente ale domeniului de definiție, dar amplitudinea poate lua orice valoare reală în domeniul de valori. Însă sistemele digitale nu pot prelucra semnale

cu valori într-un domeniu continuu. De aceea, este nevoie de o nouă procesare, și anume, de *cuantizare*.

Prin cuantizare, fiecărui eşantion i se alocă o valoare dintr-un set finit de valori. Distanţa dintre două niveluri consecutive de cuantizare se numește *pas de cuantizare*. În funcție de valoarea pasului de cuantizare, cuantizarea poate fi:

- *cuantizare neuniformă*: dacă pasul de cuantizare variază;
- *cuantizare uniformă*: dacă pasul de cuantizare este constant. Acestă variantă de cuantizare este întâlnită la majoritatea convertoarelor A/D. Cele mai des folosite două metode pentru cuantizarea uniformă sunt *cuantizarea prin rotunjire* și *cuantizarea prin trunchiere.*

Cuantizarea prin rotunjire (round)

Dacă notăm cu q distanţa dintre două niveluri de cuantizare consecutive, atunci valoarea unui eşantion analogic ce va fi cuantizat, folosind cuantizarea prin rotunjire, va fi cel mai apropiat nivel de cuantizare disponibil. Pentru a genera în MATLAB un semnal cuantizat prin rotunjire, se utilizează funcția **`round(x)`**.

Observație: când semnalul de intrare (eşantionul analogic) se află în domeniul de lucru al cuantizorului, eroarea de reprezentare variază în intervalul -q/2÷q/2. Acest tip de eroare, denumită *eroare de cuantizare*, apare când valoarea eşantionului analogic este situată între două niveluri disponibile de cuantizare.

Cuantizarea prin trunchiere (floor)

Dacă notăm cu q distanţa dintre două niveluri de cuantizare consecutive, atunci valoarea unui eşantion analogic ce va fi cuantizat, folosind cuantizarea prin trunchiere, va fi cel mai apropiat nivel de cuantizare disponibil, care este valoric inferior eșantionului. Pentru a genera în MATLAB un semnal cuantizat prin trunchiere, se utilizează funcția **`floor(x)`**.

Observație: când semnalul de intrare (eșantionul analogic) se află în domeniul de lucru al cuantizorului, eroarea de reprezentare variază în intervalul –q ÷ 0.

Pentru ambele tipuri de cuantizare, când semnalul de intrare se află în afara domeniului de cuantizare, toate eșantioanele care depășesc limita maximă a intervalului (L_{max}) vor lua valoarea L_{max}, iar eșantioanele care au valori mai mici decât limita minimă a intervalului (L_{min}) vor lua valoarea L_{min}. Această eroare se numește *eroare de depășire* și crește nelimitat în funcție de semnalul de intrare.

***Exemplul* 4.2:**

Fie următoarele niveluri de cuantizare disponibile {-6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4}. În *tabelul* 4.1 sunt incluse valorile obținute în urma cuantizării prin rotunjire și prin trunchiere pentru diverse valori ale unui semnal analogic.

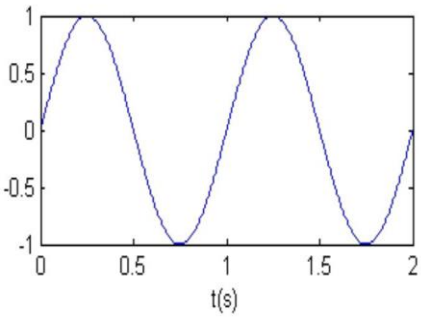
Tabel 4.1. Cuantizarea prin rotunjire și prin trunchiere

Valoare eșantion analogic	**Cuantizare prin rotunjire (round)**		**Cuantizare prin trunchiere (floor)**	
	Valoare	**Eroare**	**Valoare**	**Eroare**
- 1.2	- 1	0.2	- 2	- 0.8
2.63	3	0.37	2	- 0.63
2	2	0	2	0
- 2.61	- 3	- 0.39	- 3	- 0.39
3.5	4	0.5	3	- 0.5
6	4	- 2	4	- 2
- 10	- 6	4	- 6	4

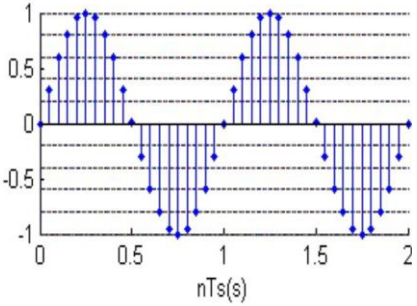
***Exemplul* 4.3:**

Dacă se dorește cuantizarea uniformă cu pasul de cuantizare q = 0.2, în intervalul L_{min} = -1 și L_{max} = 1, nivelurile de cuantizare posibile sunt {-1, -0.8, -0.6, -0.4, -0.2, 0, 0.2, 0.4, 0.6, 0.8, 1}. În urma cuantizării unui semnal sinusoidal (de amplitudine unitară,

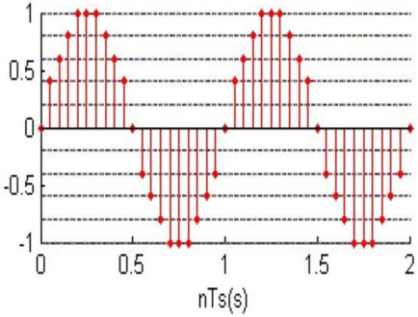
fază inițială nulă, eșantionat cu T_s = 0.05*s*) se obține semnalul din *figura 4.5, c* (cuantizare prin rotunjire) și semnalul din *figura 4.5, d* (cuantizare prin trunchiere).



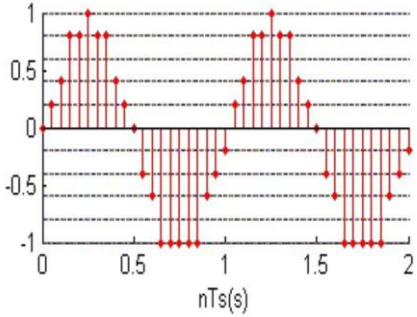
a) Semnal analogic



b) Semnal eșantionat



c) Semnal eșantionat și cuantizat folosind cuantizarea prin rotunjire (round)



d) Semnal eșantionat și cuantizat folosind cuantizarea prin trunchiere (floor)

Fig.4.5. Cuantizarea prin rotunjire și trunchiere

Zgomotul de cuantizare este un semnal obținut prin diferența dintre semnalul rezultat în urma cuantizării și semnalul original.

***Exemplul* 4.4:**

Fie un semnal sinusoidal cuantizat, folosind metoda prin rotunjire (*figura 4.6, a*). Dacă prin N se notează numărul de niveluri de cuantizare, atunci semnalul din *figura 4.6, a* se generează cu ajutorul următorului script:

```
N=6;
t = 0:0.001:2;
x=sin(2*pi*t);
y = round((N-1)*x)/(N-1);
plot(t,y,t,x);
grid
```

Zgomotul de cuantizare este dat în *figura 4.6, b*.

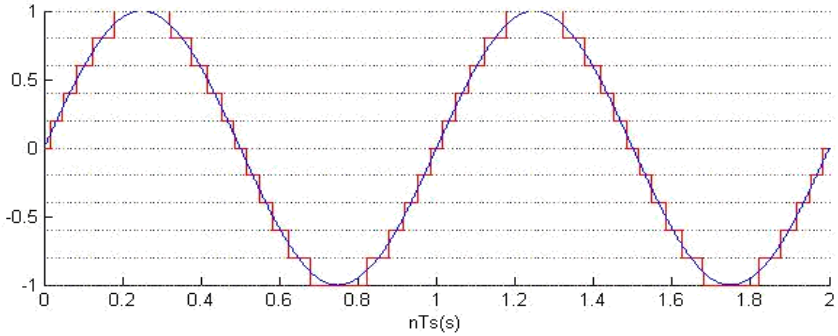


Fig.4.6, *a*. Semnal cuantizat folosind metoda prin rotunjire

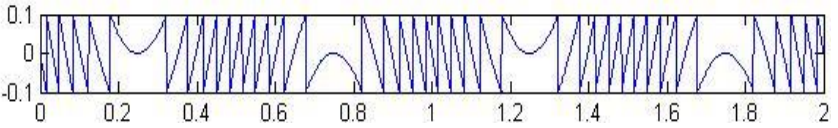


Fig.4.6, *b*. Zgomotul de cuantizare

În urma cuantizării, fiecare eșantion al semnalului are o valoare din setul finit de valori al nivelelor de cuantizare. Astfel, pentru descrierea unui eșantion este suficientă precizarea nivelului de cuantizare corespunzător.

***Exemplul* 4.5:**

Fie următoarele niveluri de cuantizare disponibile: {0, 1/3, 2/3, 1}.

Nivel 0 = 0	Nivel 2 = 2/3
Nivel 1 = 1/3	Nivel 3 = 1

În *tabelul* 4.2 sunt incluse valorile obținute în urma cuantizării prin rotunjire pentru 10 eșantioane ale semnalului:

$$x[n] = \frac{n}{n + 1.5}.$$

Tabel 4.2. Cuantizarea prin rotunjire

Valoare indice n	Valoare eșantion analogic	Cuantizare prin rotunjire (round)	Nivel cuantizare (explicit)	Nivel cuantizare (binar)
0	0.000	0	Nivel 0	00
1	0.400	1/3	Nivel 1	01
2	0.571	2/3	Nivel 2	10
3	0.667	2/3	Nivel 2	10
4	0.727	2/3	Nivel 2	10
5	0.769	2/3	Nivel 2	10
6	0.800	2/3	Nivel 2	10
7	0.823	2/3	Nivel 2	10
8	0.842	1	Nivel 3	11
9	0.857	1	Nivel 3	11

Așa cum se poate observa în *tabelul* 4.2, nivelul de cuantizare poate fi indicat cu un număr mic de biți (2 biți în cazul exemplului), astfel încât întregul semnal poate fi stocat într-un spațiu mic de memorie.

Numărul (minim) de biți prin care trebuie indicat nivelul de cuantizare se calculează prin formula:

$$b = \lceil log_2 N \rceil,$$

unde:

- b = numărul de biți necesari indicării nivelului de cuantizare;
- N = numărul de niveluri de cuantizare;
- $\lceil \ \rceil$= rotunjire spre infinit = cel mai mic întreg mai mare sau egal cu valoarea dintre paranteze. Rotunjirea este necesară, deoarece numărul de biți b trebuie să fie număr natural.

Deoarece pentru indicarea fiecărui nivel este nevoie de b biți, iar valoarea fiecărui eșantion este unul dintre niveluri, b este totodată și **numărul de biți per eșantion**, folosit pentru stocarea semnalului.

***Exemplul* 4.6:**

Care ar trebui să fie numărul minim de biți *per eșantion* pentru stocarea semnalului din *Exemplul* 4.3?

Soluție: în *Exemplul 4.3* sunt N=11 niveluri de cuantizare, deci, numărul minim de biți *per eșantion* necesar este $b = \lceil log_2 11 \rceil = 4$.

E4.2. ***Exerciţiu:***

1. Un semnal cu durata de 2 minute este eșantionat cu frecvența de eșantionare de 4kHz. Câte eșantioane vor rezulta? Dacă fiecare eșantion este stocat pe 2 octeți, ce memorie vor ocupa toate eșantionale generate?

2. Să se cuantizeze semnalele $x[n] = \frac{17}{n}$ și $y[n] = -\frac{17}{n}$, $n = 1 \dots 40$, folosind metodele *floor* și *round*. Se cunosc nivelurile de cuantizare $\{0, \pm 1, \pm 2, \dots, \pm 17\}$. Să se reprezinte grafic semnalele originale $x[n]$ și $y[n]$, precum și semnalele obținute în urma cuantizării. Să se calculeze și să se reprezinte grafic zgomotul de cuantizare.

3. Să se genereze 15 eșantioane de zgomot alb gaussian cu media zero și dispersia 0,2. *Indiciu*: pentru a genera zgomot alb gaussian se poate folosi funcția `randn`:

```
z = 0.2*randn(1,15).
```

Să se cuantizeze semnalul z folosind un cuantizor uniform cu nivelurile $\left\{0, \pm\frac{1}{4}, \pm\frac{2}{4}, \pm\frac{3}{4}, \pm 1\right\}$.

4. Cuantizați semnalul din *Exercițiul* 4.1.1 pe 8 respectiv, 16 biți. Reprezentați semnalul cuantizat alături de zgomotul de cuantizare.

4.4. Interpolarea semnalelor eșantionate

Reconstrucția semnalului $x(t)$ în domeniul timp din eșantioanele $x[n]$ se realizează prin relaţia de interpolare:

$$x(t) = \sum_{n=-\infty}^{+\infty} x[nT]\, sinc[\omega_m(t-nT)],$$

unde funcţia *sinc* se defineşte astfel:

$$sinc(x) = \frac{\sin(x)}{x}$$

cu argumentul $x = \omega_m(t-nT)$. Graficul acestei funcţii este reprezentat în figura 4.7.

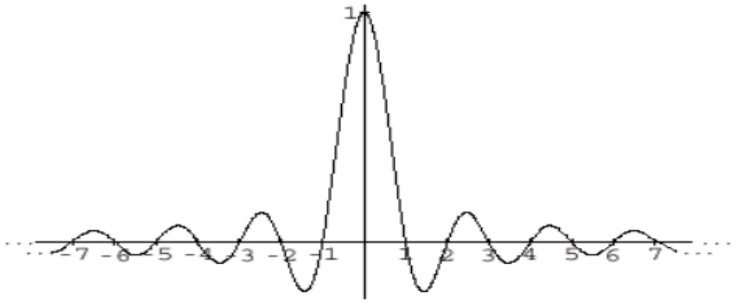


Fig.4.7. Reprezentarea grafică a funcţiei $sinc(x)$

Figura 4.8 este o ilustrare a modului cum se poate recupera un semnal rectangular din eşantioanele sale prelevate cu o anumită periodicitate.

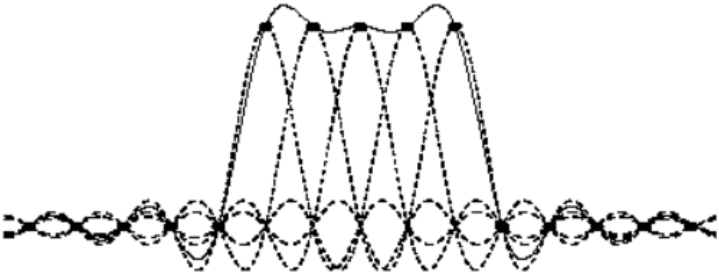


Fig.4.8. Reconstituirea unui impuls rectangular
din eşantioanele sale

Sunt luate în considerare cinci eşantioane nenule prelevate pe durata semnalului, alte eşantioane fiind, evident, nule deoarece sunt situate în afara suportului compact al semnalului rectangular. Însumarea celor cinci funcţii *sinc* multiplicate cu valorile eşantioanelor [...,0,1,1,1,1,1,0,...] formează curba cu linie plină din figură a semnalului rectangular eşantionat. Semnalul "reconstituit" este nenul acolo und ar trebuie să fie nul, este ondulat acolo unde ar trebuie să fie constant, are creşteri şi descreşteri în timp finit acolo unde ar trebui să varieze brusc. Dar această aproximare sau una întrucâtva mai bună ar putea fi satisfăcătoare sub aspect practic-ingineresc sau ar putea fi corectată pentru a fi utilizată practic.

Dincolo de aspectul aproximativ comentat mai este de discutat non-cauzalitatea funcţiilor *sinc* din formula de interpolare. Funcţia *sinc* ataşată unui eşantion "există" prin valori nenule încă înainte ca eşantionul să fie prelevat/măsurat. Ceea ce este, desigur, absurd. De aceea, în operaţiile practice de reconstituire a unui semnal din eşantioanele sale există o lipsă de informaţie la începutul secvenţei de eşantioane şi la finalul ei, lipsă care aduce la aproximări şi mai grosiere în zonele apropiate capetelor intervalului finit pe care semnalul este evaluat. Erorile de aproximare pentru zonele extreme se numesc: "efecte de capăt".

***Exemplul* 4.7:**

În *script*-ul următor se propune eşantionarea, apoi reconstrucţia unui semnal, sumă a doua sinusoide:

$$s(t) = a_1 \cos(\omega_1 t) + a_2 \cos(\omega_2 t - \varphi),$$

care este, evident, de bandă limitată.

Introduceţi în calculator, cu editorul ***Matlab***, *script*-ul care urmează:

```
% optiune de reprezentare (rep=0) grafic unic,
(rep>0) grafice separate
tmax=10;   % timpul maxim de reprezentare
fi=pi/2;   % faza componentei secundare
```

```
f1=1;      % frecventa componentei primare
f2=2;      % frecventa componentei secundare
% pasul utilizat la reprezentarea grafica
pasmic=0.001;
fes=4.1;         % frecventa de esantionare
rep=1;
tes=1/fes;       % perioada esantioanelor
a1=2;
a2=1;
t=0:pasmic:tmax;
% pregatirea graficului 1 (semnal original)
y=a1*cos(2*pi*f1*t)+a2*cos(2*pi*f2*t-fi);
t1=0:tes:tmax;
n=round(tmax/tes)+1;
% pregatirea graficului 2 (esantionarea)
y1=a1*cos(2*pi*f1*t1)+a2*cos(2*pi*f2*t1-fi);
% pregatirea graficului 3 (reconstituire)
y2=y1(1)*sinc(t/tes);
for k=1:(n-1)
y2=y2+y1(k+1)*sinc(t/tes-k);
end
if rep>0
     subplot(4,1,1)
   plot(t,y)     % trasarea graficului 1 (semnal)
   hold on
   ylabel('Original')
   title('ESANTIONAREA SEMNALELOR')
   plot([0 tmax],[0 0])
   for i=1:n
            subplot(4,1,2)
      % trasarea graficului 2 (esantionarea)
       plot([t1(i) t1(i)],[0 y1(i)],'b:')
   hold on
      ylabel('Esantionat')
plot([0 tmax],[0 0])
end
   subplot(4,1,3)
      % trasarea graficului 3 (reconstituire)
   plot(t,y2,'k')
```

```
   hold on
plot([0 tmax],[0 0])
 for i=1:n
plot([t1(i) t1(i)],[0 y1(i)],'b:')
   end
   ylabel('Reconstituire')
   subplot(4,1,4)
% trasarea graficului 4 (diferente)
   plot(t,y-y2,'r')
   hold on
plot([0 tmax],[0 0])
   ylabel('Diferente')
   xlabel('Timp(s)')
else
   plot(t,y) % trasarea graficului 1 (semnal)
 hold on
plot([0 tmax],[0 0])
 for i=1:n
% trasarea graficului 3 (reconstituire)
      plot([t1(i) t1(i)],[0 y1(i)],'b:')
 end
   plot(t,y2,'k')
% trasarea graficului 4 (diferente)
   plot(t,y-y2,'r')
plot([0 tmax],[0 0])
title('ESANTIONAREA SEMNALELOR')
ylabel('Semnal(albastru)/Reconstituire(negru)/
Diferente(rosu)')
   xlabel('Timp(s)')
end
```

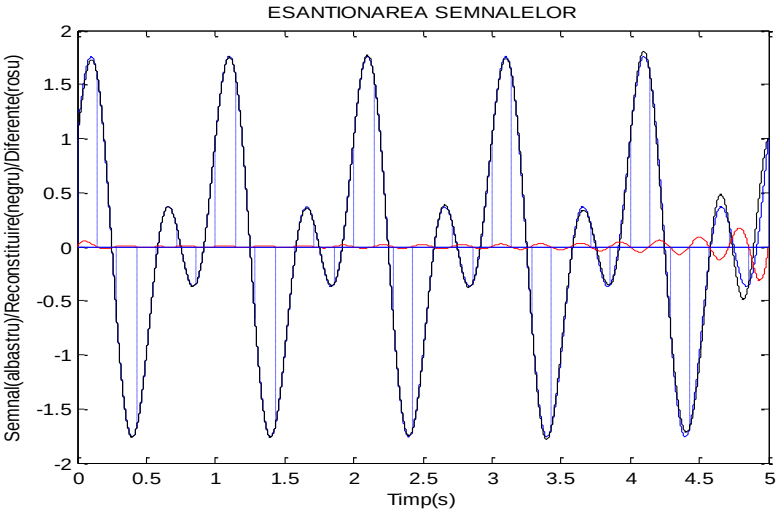


Fig.4.9. Eşantionarea şi reconstrucţia unui semnal, sumă a doua sinusoide

E4.3. ***Exerciţiu:***

Chestiuni de studiu în baza *Exemplului* 4.7.

a) Executarea calculelor cu valorile recomandate în antetul *script*-ului, reprezentarea semnalelor în formă originară şi după reconstrucţie, precum şi a diferenţelor dintre ele, în spaţii grafice diferite sau în acelaşi spaţiu.

b) Executarea de calcule cu valorile diferite pentru frecvenţele f_1 şi f_2, eventual pentru amplitudini diferite ale sinusoidelor componente, utilizând pentru fiecare caz trei valori ale frecvenţei de eşantionare: mai mică, egală şi mai mare (de 10 ori) ca frecvenţa Nyquist.

c) Aprecierea vizuală şi/sau cantitativ a modificărilor semnalului reconstituit, calitatea reconstrucţiei semnalului în zona de început şi de final a intervalului de timp observat.

d) Studierea şi a altor semnale de bandă limitată, recomandate de profesor.

Bibliografie

1. Strîmbu C. *Semnale și circuite electronice – Analiza și prelucrarea semnalelor*. Brașov: Academia Forțelor Aeriene „Henri Coandă”, 2007. – 120 p.
2. Ceangă E.; Munteanu Iu.; Bratcu Antoneta, Culea M. *Semnale, circuite și sisteme. Partea I: Analiza semnalelor*. Galați: Editura „Academica”, 2001. – 201 p.
3. Idriceanu S. *Teoria informației și transmisiuni de date. Ciclu de prelegeri. Partea I.* Chișinău: Editura „Tehnica-UTM”, 1996. – 151 p.
4. Dumitrescu B. *Prelucrarea semnalelor: breviar teoretic, probleme rezolvate, ghid MatLab*. București: Universitatea ”Politehnica” București, 2006. – 182 p.
5. Mihu I.; Neghină Cătălina *Prelucrarea digitală a semnalelor: aplicaţii didactice în MatLab*. Sibiu: Editura Universităţii "Lucian Blaga", 2014. – 139 p.
6. Kertesz Csaba-Zoltán; Ivanovici Laurențiu-Mihail *Procesarea digitală a semnalelor. Îndrumar de laborator*. Brașov: Editura Universitatea Transilvania, 2009. – 73 p.
7. Chițul I.; Bejan N.; Nistiriuc P. *Teoria transmisiunii informației. Îndrumar de laborator. Partea 1*. Chișinău: Editura „Tehnica-UTM”, 2003. – 64 p.
8. Kalechman M. *Practical Matlab Applications for Engineers*. CRC Press, Boca Raton, FL, 2008.