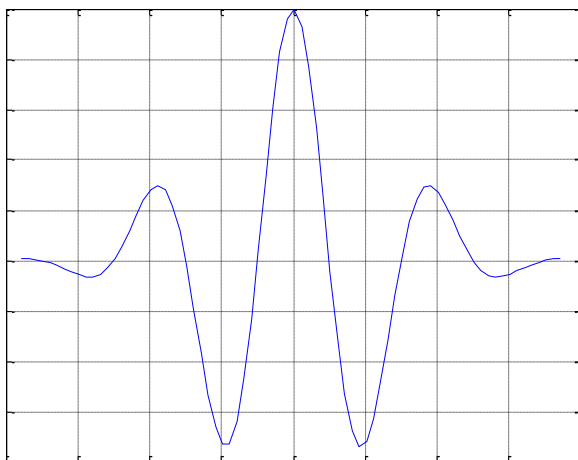


UNIVERSITATEA TEHNICĂ A MOLDOVEI

PRELUCRAREA SEMNALELOR

Partea 1

Indicații de laborator



Chișinău

2018

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Facultatea Calculatoare, Informatică și Microelectronică

Departamentul Ingineria Software și Automatică

PRELUCRAREA SEMNALELOR

Partea 1

Indicații de laborator

Chișinău

Editura „Tehnica-UTM”

2018

CUPRINS

1. Lucrarea nr. 1. Introducere în Matlab.....	4
2. Lucrarea nr. 2. Formarea semnalelor elementare în sistemul Matlab.....	41
3. Lucrarea nr. 3. Analiza spectrală a semnalelor.....	65
4. Lucrarea nr. 4. Eșantionarea și cuantizarea semnalelor. Interpolarea semnalelor eșantionate.....	88
Bibliografie.....	105

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 calcului numeric și reprezentărilor grafice. Elementul de bază cu care operează este matricea, de aici provenind și numele: MATrix LABoratory. Resursele sale de calcul și reprezentare grafică sunt bogate, permițând operații matematice fundamentale, analiza datelor, programare, reprezentări grafice 2D și 3D, realizarea de interfețe grafice etc. Din punct de vedere al construcției sale, MATLAB este alcătuit dintr-un nucleu de bază în jurul căruia sunt grupate TOOLBOX-urile. Acestea reprezintă niște aplicații specifice, fiind de fapt colecții extinse de funcții 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 și funcții 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 promptul MATLAB și se află la începutul fiecărei linii de comandă din

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

***Atenție:** Comenzile introduse anterior pot fi readuse în linia de comandă prin folosirea săgeților de la tastatură, ↑ și ↓ (căutarea se face ca într-o “listă”).*

1.2.1. Funcții MATLAB de interes general

- **help** – furnizează informații despre MATLAB și funcțiile acestuia.

Sintaxă:

`help nume` – furnizează informații despre `nume` (poate fi un nume de funcție sau un nume de director).

Exemplu:

`help fft` – furnizează informații despre transformata Fourier discretă.

- **who** – listează numele variabilelor din spațiul de lucru.
- **whos** – furnizează informații suplimentare referitoare la variabilele din spațiul de lucru (nume, dimensiune etc.).
- **format** – stabilește formatul extern de afișare al numerelor pe ecran.

Sintaxă:

`format opțiune` – parametrul `opțiune` 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 informații tastezi `help format`.

Exemple:

`format short`

`x=pi` → `x =`
3.1416

format long

x=pi → x = 3.14159265358979

Să se verifice și celelalte tipuri de format, folosind aceeași valoare x.

- **clear** – șterge din memorie una sau mai multe variabile.

Sintaxe:

clear v – șterge din memorie variabila v;

clear v1 v2 – șterge din memorie variabilele v1 și v2 (sintaxa e valabilă pentru oricâte variabile);

clear – șterge din memorie toate variabilele definite până în acel moment.

- **lookfor** – listează toate numele de fișiere care au în prima linie a help-ului cuvintele menționate ca argument, precum și prima linie din help.

Sintaxă:

lookfor cuvânt – listează toate numele de fișiere care conțin în prima linie a help-ului cuvânt, precum și prima linie din help.

Exemplu:

lookfor ifft – listează toate numele de fișiere care conțin în prima linie a help-ului ifft, precum și prima linie din help.

- **dir** – afișează numele tuturor fișierelor din directorul curent sau din orice alt director precizat ca argument.

Sintaxe:

dir – afișează numele tuturor fișierelor din directorul current;

dir nume – afișează numele tuturor fișierelor 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:\`).

Atenție: *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 și 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.

Atenție:

- *dacă după o linie de comandă urmează semnul “ ; ”, atunci rezultatul nu va mai fi afișat (excepție fac comenzile grafice);*
- *dacă în fața unei linii de comandă se pune semnul “ % ”, atunci se face abstracție de linia respectivă (este interpretată ca o linie de comentariu);*
- *dacă se dorește continuarea unei instrucțiuni pe linia următoare, se folosesc “...” urmate de enter.*

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

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

1.2.3. Matricea – element de bază în MATLAB

MATLAB lucrează numai cu un singur tip de obiecte, și anume, matrice numerice, având elemente reale sau complexe. Astfel, scalarii sunt priviți 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 (blank) sau virgule;
- liniile matricei se separă prin ; sau *enter*.

Exemplu:

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

```
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
```


$$C = [-1; -2] \quad \rightarrow \quad C = \begin{matrix} -1 \\ -2 \end{matrix}$$

Pentru o matrice M:

- $M(i, j)$ reprezintă elementul din matricea M corespunzător liniei i și 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 poziția i din vector.

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

$$\begin{array}{lll} A(2, 3) & \rightarrow & \text{ans} = 6 \\ A(4) & \rightarrow & \text{ans} = 5 \\ B(2) & \rightarrow & \text{ans} = 8 \\ C(2) & \rightarrow & \text{ans} = -2 \end{array}$$

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

$$\begin{array}{lll} A(2, 3) = 0 & \rightarrow & A = \begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 0 \end{matrix} \\ A(3, 3) = -3 & \rightarrow & A = \begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 0 \\ 0 & 0 & -3 \end{matrix} \\ C(3) = 6 & \rightarrow & C = \begin{matrix} -1 \\ -2 \\ 6 \end{matrix} \end{array}$$

Se pot construi matrice de dimensiuni mai mari pornind de la matrice de dimensiuni mai reduse. Pentru exemplificare vom folosi matricea A și vectorii B și C în ultima lor formă ($A - 3 \times 3$, $B - 1 \times 3$, $C - 3 \times 1$):

$$D = [A; B] \rightarrow D = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 0 \\ 0 & 0 & -3 \\ 7 & 8 & 9 \end{pmatrix}$$

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

Atenție: A și B au același număr de coloane (3) pentru a fi posibilă construcția.

$$E = [A, C] \rightarrow E = \begin{pmatrix} 1 & 2 & 3 & -1 \\ 4 & 5 & 0 & -2 \\ 0 & 0 & -3 & 6 \end{pmatrix}$$

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

Atenție: A și C au același număr de linii (3) pentru a fi posibilă construcția.

Dacă v este un vector atunci:

- $v(i:k)$ – selectează elementele de pe pozițiile $i, i+1, i+2, \dots, k$ ale vectorului v ; dacă $i > k$ atunci vectorul rezultat este gol (nu are niciun element).
- $v(i:j:k)$ – selectează elementele de pe pozițiile $i, i+j, i+2j, \dots$ până la k , ale vectorului v (selectează cu pasul j); dacă $j > 0$ și $i > k$ sau $j < 0$ și $i < k$ atunci vectorul rezultat este gol.
- $v([i, j, k])$ – selectează elementele de pe pozițiile i, j și 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 $i, i+j, i+2j, \dots$ până la k ale matricei M (selectează cu pasul j).
- $M(i:j:k, :)$ – selectează liniile $i, i+j, i+2j, \dots$ până la k ale matricei M .
- $M(i:j, k:l)$ – extrage submatricea formată cu elementele aflate la intersecția liniilor de la i la j și coloanelor de la k la l ale matricei M .
- $M(:, [i, j, k])$ – selectează coloanele i, j și k ale matricei M .
- $M([i, j, k], :)$ – selectează liniile i, j și k ale matricei M .
- $M([i, j, k], [l, m, n])$ – extrage submatricea formată cu elementele aflate la intersecția liniilor i, j și k și coloanelor l, m și n ale matricei M .
- $M(:, :)$ – selectează întreaga matrice M .
- $M(i:j)$ – selectează elementele de la i la j ale matricei M și le pune sub forma unui vector linie (elementele într-o matrice se numără pe coloane).
- $M(:)$ – selectează toate elementele matricei M și le aranjează sub forma unui vector coloană (pune coloanele matricei M una sub alta, sub forma unui vector coloană).

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

1.2.4. Vectori și matrice uzuale

- **Generarea vectorilor cu pas liniar**

Sintaxe:

- `v=inițial:pas:final` – se generează un vector linie `v` cu elementele, începând de la `inițial` la `final`, cu pasul egal cu `pas` (pasul poate fi și negativ, dar atunci valoarea inițială trebuie să fie mai mare decât valoarea finală);
- `v=inițial:final` – se generează un vector linie `v` cu elementele, începând de la `inițial` 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 și având un număr de elemente egal cu `număr_de_elemente`.

Verificați 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 logaritm**

Sintaxe:

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

Atenție: Dacă `maxim=pi`, atunci elementele vor fi distribuite logaritm între 10^{minim} și π .

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

Verificați 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 $n \times n$ cu toate elementele egale cu 0;

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

Verificați 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 $n \times n$;
- `eye(m,n)` – returnează o matrice de dimensiune $m \times n$, 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.

Verificați 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 distribuție uniformă**

Sintaxe:

- `rand(n)` – returnează o matrice de dimensiune $n \times n$, având drept elemente numere aleatoare cu distribuție uniformă între 0 și 1;
- `rand(m,n)` – returnează o matrice de dimensiune $m \times n$, având drept elemente numere aleatoare cu distribuție uniformă între 0 și 1;

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

Verificați 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 distribuție normală (gaussiană)*

Sintaxe:

- `randn(n)` – returnează o matrice de dimensiune $n \times n$, având drept elemente numere aleatoare cu distribuție normală (gaussiană) de medie nulă și varianță unitară;
- `randn(m,n)` – returnează o matrice de dimensiune $m \times n$, având drept elemente numere aleatoare cu distribuție normală de medie nulă și varianță unitară;
- `randn(size(M))` – returnează o matrice de dimensiunea matricei `M`, având drept elemente numere aleatoare cu distribuție normală (gaussiană) de medie nulă și varianță unitară.

Verificați 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 principală;

- `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 conține elementele de pe diagonala principală a matricei M ;
- `diag(M,k)` – returnează un vector coloană ce conține 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 și o matrice A :

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

```
A=randn(5)
```

Verificați 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)` – înlocuiește 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 verificați 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)** – înlocuiește 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 verificați următoarele *exemple*:

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

1.2.5. Dimensiunea unei matrice.

Determinantul și inversa unei matrice

- **length, size** – *Determinarea dimensiunii variabilelor*

Sintaxe:

Dacă **v** este un vector și **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 și numărul de coloane al matricei **M** (maximul dintre **m** și **n**);
- **[l,c]=size(M)** – returnează numărul de linii (**l**) și 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** și o matrice **C**:

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

Verificați 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)

Verificați 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.

Verificați folosind matricele M și N definite la punctul precedent.

E 1.1. *Exercițiu:*

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

- a) Ce lungime trebuie să aibă **b** astfel, încât să aibă sens înmulțirea (în sens matricial) **a*b**? Inițializați în MATLAB cei doi vectori și efectuați înmulțirea.
- b) Efectuați înmulțirea **b*a**.

1.2.6. Instrucțiuni de control logic

Operatori relaționali și operatori logici

- **if, else, elseif** – *Execuția condiționată*

Sintaxe:

```
- if expresie  
    instrucțiuni  
end
```

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

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

// dacă expresie este adevărată, se execută
instrucțiuni_1; dacă expresie este falsă, se execută
instrucțiuni_2.

```
- if expresie_1  
    instrucțiuni_1  
else if expresie_2  
    instrucțiuni_2  
end
```

// dacă expresie_1 este adevărată, se execută
instrucțiuni_1; dacă expresie_1 este falsă și
expresie_2 este adevărată, se execută instrucțiuni_2.

- **for** – *Repetarea unui număr de instrucțiuni de un anumit număr de ori*

Sintaxă:

```
- for index=inițial:pas:final  
    instrucțiuni  
end
```

// pentru index, parcurgând intervalul de la inițial la final cu pasul pas, se execută instrucțiuni.

- **while** – *Repetarea unui număr de instrucțiuni atâta timp cât o condiție specificată este adevărată*

Sintaxă:

```
- while expresie  
    instrucțiuni  
end
```

// cât timp expresie este adevărată, se execută instrucțiuni.

- **Operatori relaționali**

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

- **Operatori logici**

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

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

Pentru secvențe lungi de comenzi se recomandă crearea și lansarea în execuție a unui program MATLAB. Acesta este un fișier “text”, având extensia .m și conținând succesiunea dorită de comenzi MATLAB. După creare, fișierul devine o nouă comandă externă MATLAB, care poate fi lansată în execuție prin simpla introducere de la tastatură a numelui fișierului (fără extensia .m).

Pentru crearea unui astfel de fișier se parcurg următorii pași:

- în bara de meniuri a ferestrei de comenzi se selectează File, iar în interiorul acesteia New, urmată de M-fișe. Se va deschide astfel o nouă fereastră (sesiune de editare cu editorul NOTEPAD) cu bară de meniuri proprie și cu numele Untitled. În acest fișier se scrie programul MATLAB drept;
- pentru a salva fișierul 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 fișierului (File name:) însoțit de extensia .m și locul unde dorim să îl salvăm (Save in:).

Atenție:

- *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, fișierul trebuie salvat (File, urmat de Save);*
- *denumirea fișierului nu trebuie să înceapă cu cifre sau să fie un cuvânt cheie (instrucțiune) Matlab. Denumiri de genul „ex_1.m” sau „progr_2.m” sunt suficiente.*

E 1.2. Exercițiu:

Se creează un fișier nou care trebuie salvat în directorul **d:/student/pns/nrgrupa**. Folosind sintaxele și indicațiile din secțiunile 1.2.6 și 1.2.7, elaborați un program MATLAB care să genereze un vector cu elemente aleatoare cu distribuție normală (gaussiană) și să afișeze elementele negative ale acestui vector.

1.2.8. Crearea funcțiilor MATLAB (crearea fișierelor funcție)

Dacă prima linie a unui fișier MATLAB (*.m) conține la început cuvântul `function`, atunci fișierul respectiv e declarat ca fișier funcție. Aceste fișiere pot fi adăugate ca funcții noi în structura MATLAB. Forma generală a primei linii a unui fișier

funcție este:

function

[parametrii_ieșire]=nume(parametrii_intrare)

cu următoarele semnificații:

- **function** – cuvânt cheie care declară fișierul ca fișier funcție;
- **nume** – numele funcției; reprezintă numele sub care se salvează fișierul .m (extensia .m nu face parte din nume); acest nume nu poate fi identic cu cel al unui fișier .m deja existent;
- **parametrii_ieșire** – reprezintă parametrii de ieșire ai funcției; trebuie separați prin virgulă și cuprinși între paranteze drepte;
- **parametrii_intrare** – reprezintă parametrii de intrare ai funcției; trebuie separați prin virgulă și cuprinși între paranteze rotunde.

1.2.9. Operații aritmetice

Operații asupra numerelor complexe

• Operatori aritmetici:

- **+** – adunare;
- **-** – scădere;
- ***** – înmulțire
- **.*** – înmulțire între două matrice (sau vectori) element cu element;
- **/** – împărțire;
- **./** – împărțire între două matrice (sau vectori) element cu element;
- **^** – ridicare la putere;
- **.^** – ridicare la putere a unei matrice (sau vector) element cu element;
- **'** – transpunere și conjugare;
- **.'** – transpunere.

Să se definească următoarele elemente:

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

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

Verificați și explicați 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 ,
 $b.^3$, $2/A$, $2./A$, $2/b$, $2./b$

- ***Operații asupra numerelor complexe:***

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

Verificați comenzile, utilizând drept argumente elementele A, B, a și b definite anterior.

1.2.10. Funcții matematice uzuale

- ***Funcțiile radical, exponențială și logaritm:***

- `sqrt` – extragere radical de ordinul 2 (rădăcina pătrată);
- `exp` – calculează exponențiala (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.

- ***Funcțiile trigonometrice directe:***

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

- ***Funcțiile trigonometrice inverse:***

- \arcsin – calculează arcsinus;
- \arccos – calculează arccosinus;
- \arctan – calculează arctangenta;
- $\operatorname{atan2}$ – calculează arctangenta dacă argumentul este complex;
- acot – calculează arccotangenta;
- asec – calculează arcsecanta;
- acsc – calculează arccosecanta.

- ***Funcțiile hiperbolice directe:***

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

- ***Funcțiile 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 informații despre modul de utilizare al acestor funcții folosiți comanda `help` însoțită de numele funcției dorite (vezi sintaxa de la pagina 2).

1.2.11. Funcții destinate analizei de date

- **sum,prod** – *Suma și produsul*

Sintaxe:

Dacă v este un vector și 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: $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$, $a = [1 \ 3 \ 8]$

Să se verifice următoarele *exemple*:

`sum(a)`

`sum(A)`

`sum(sum(A))`

`prod(a)`

`prod(A)`

`prod(prod(A))`

- **max,min** – *Maximul și minimul*

Sintaxe:

Dacă v este un vector și 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 și 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 și 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 și un vector linie p ce conține poziția 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 și un vector linie p ce conține poziția minimului respectiv în cadrul fiecărei coloane.

Verificați sintaxele anterioare, folosind matricea A și vectorul a definite mai sus.

- **mean** – *Media aritmetică*

Sintaxe:

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

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

Verificați sintaxele anterioare, folosind matricea A și vectorul a definite mai sus.

E 1.3. *Exercițiu:*

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

- media aritmetică a părților reale ale elementelor vectorului;
- un vector ce conține elementele vectorului inițial ridicate la pătrat;
- o matrice obținută din înmulțirea vectorului inițial cu transpusul său.

Atenție: pentru a nu se afișa rezultate intermediare din funcție sau elementele unor variabile se va folosi `;` la sfârșitul liniei respective de program.

1.2.12. Reprezentări grafice

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

Sintaxe:

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

- `plot(v)` – dacă v este un vector cu *elemente reale*, se vor reprezenta grafic elementele sale în funcție 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 funcție de partea reală (pe abscisă) și de partea imaginară (pe ordonată) a elementelor sale;
- `plot(M)` – se vor reprezenta pe același grafic coloanele matricei M (fiecare din coloanele matricei este privită ca un vector și reprezentat ca în sintaxa precedentă).

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

- `plot(x, y)` – se vor reprezenta grafic elementele vectorului y în funcție 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 același grafic coloanele matricei M în funcție de elementele vectorului x ; dacă lungimea vectorului x este egal cu numărul de coloane al matricei M , atunci se vor reprezenta pe același grafic liniile matricei M în funcție 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 același grafic coloanele matricei N în funcție de coloanele matricei M (coloana k din matricea N va fi reprezentată în funcție de coloana k din matricea M , unde $k = 1, 2, \dots$, numărul de coloane); *dacă cele două matrice nu au aceeași dimensiune, atunci reprezentarea nu este posibilă*.

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

- `plot(x1,y1,x2,y2,...,xn,yn)` - se vor reprezenta pe același grafic $y1$ în funcție de $x1$, $y2$ în funcție de $x2$, ... , yn în funcție de xn (pot fi vectori sau matrice); rămân valabile considerentele făcute în sintaxele anterioare referitor la cazurile când avem vectori sau matrice; *dacă x_i și y_i ($i = 1, 2, \dots, n$) nu au aceeași dimensiune, atunci reprezentarea nu este posibilă*.

Atenție: se pot folosi diverse linii, markere și culori de reprezentare a graficelor. Vezi `help plot`.

Funcția `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 funcției `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 și markere).

Vezi `help stem`.

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

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

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

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

1	2	3
4	5	6
7	8	9

Sintaxă:

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

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

Dacă se dorește vizualizarea numai a unei anumite porțiuni dintr-un grafic, corespunzătoare unor anumite intervale pe abscisa și ordonată, se va folosi comanda `axis`.

Sintaxa:

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

- **title, xlabel, ylabel, gtext, grid** – *Precizarea titlului graficului și a etichetelor axelor. Trasarea unei rețele 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 rețea 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 reținem graficul curent și să adăugăm în aceeași fereastră grafică următoarele reprezentări grafice se poate folosi funcția `hold`.

Sintaxă:

- `hold on` – reține graficul curent și adaugă în aceeași fereastră grafică următoarele reprezentări grafice;
- `hold off` – dacă se dorește în continuare reprezentarea în ferestre grafice separate (dezactivează comanda `hold on`).

Atenție:

Dacă se dorește 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 sfârșitul execuției programului va apărea numai ultima reprezentare grafică (se va folosi o singură fereastră grafică ce va fi “ștearsă” de fiecare dată la întâlnirea unei noi comenzi grafice).

E 1.4. Exercițiu

Elaborați un program MATLAB în care să generați și să reprezentați grafic folosind funcția **stem** următorii vectori:

- $\mathbf{z} = [0,0,0,0,0,1,0,0,\dots,0]$, vectorul \mathbf{z} , având lungimea 21. Reprezentarea grafică se va face în două „minifereestre” (funcția `subplot`), vectorul \mathbf{z} în funcție de $\mathbf{n}=0:20$, respectiv de $\mathbf{m}=-5:15$;
- $\mathbf{t}=|10-\mathbf{n}|$, reprezentat graphic în funcție de $\mathbf{n}=0:20$;
- $x_1 = \sin\left(\frac{\pi}{17}n\right)$, $-15 \leq n \leq 25$ și $x_2 = \cos\left(\frac{\pi}{\sqrt{23}}n\right)$, $0 \leq n \leq 50$.

Cele două secvențe vor fi reprezentate:

- în figura 1 – în același sistem de coordonate (pe același grafic);
- în figura 2 – folosind două “minifereestre” grafice plasate una sub alta.

Reprezentați cele două figure, folosind comanda `plot`, apoi încercați reprezentarea lor, folosind comanda `stem`.

Cu funcția **plot** se pot reprezenta grafic semnale sau funcții “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 variația semnalului reprezentat. De exemplu, dacă perioada semnalului e 0.01 secunde se poate alege variabila temporală cu pasul de 0.001s:

$$\mathbf{t} = 0:0.001:5 \text{ (secunde).}$$

Exemplu:

Să se reprezinte grafic cu funcția **plot** un semnal sinusoidal de frecvență 50 Hz, de durată 0.2 secunde și amplitudine 2. Se va alege rezoluția temporală 1ms.

```
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. Exercițiu:

- 1) Modificați pasul de variație a variabilei t la 0.01, apoi la 0.0002. Comentați diferențele;
- 2) Măsurați pe grafic perioada semnalului sinusoidal în cele 3 situații;
- 3) Generați un semnal **cosinusoidal** de frecvență 20 Hz pe care să-l reprezentați cu culoare roșie pe același grafic peste semnalul sinusoidal.

E 1.6. Exercițiu:

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 funcții în MATLAB se face cu diferențe finite (regresive, progresive sau centrate). Pentru exprimarea formulelor diferențelor finite, MATLAB pune la dispoziția utilizatorului funcția auxiliară `diff`.

Funcția MATLAB `diff` evaluează diferența elementelor succesive ale unui vector sau coloanelor unei matrice. Sintaxa funcției `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 funcția `diff`;
- `dim` are semnificația de dimensiune a lui x : implicit, funcția `diff` calculează diferențele după prima dimensiune a unei matrice x (de exemplu, pentru o matrice bidimensională diferențele se determină pe coloane); specificarea unei alte dimensiuni după care să se realizeze diferențele se poate face prin precizarea sa prin intermediul parametrului `dim`.

Fie f o funcție reală de variabilă reală derivabilă pe intervalul $[a, b]$ și $\{x_k\}_{k=1, 2, \dots, n}$ un șir strict crescător de puncte din intervalul $[a, b]$. Derivata df (notație MATLAB) de ordinul I a funcției f se poate aproxima cu diferențe finite în modul următor, în care x este vectorul punctelor $\{x_k\}$ și y este vectorul valorilor funcției f în punctele $\{x_k\}$:

• *în cazul utilizării diferențelor regressive sau progresive:*

$$df = \text{diff}(y) ./ \text{diff}(x)$$

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

Dacă se dorește ca aproximarea derivatei să se facă cu diferențe regressive, între vectorul x și vectorul df se va face asocierea: $x(2:n) \rightarrow df$, adică, aproximarea derivatei într-un punct $x(k)$ are valoarea $df(k-1)$, $k=2, \dots, n$.

Dacă se dorește ca derivata să fie aproximată cu diferențe progresive, între vectorii x și df se va considera asocierea:

$x(1:n-1) \rightarrow df$, adică, aproximarea derivatei într-un punct $x(k)$ are valoarea $df(k)$, $k=1, \dots, n-1$.

• *în cazul utilizării diferențelor centrate:*

$dx = \text{diff}(x)$; $dy = \text{diff}(y)$; $k = \text{length}(dx)$;

$$df = (dy(1:k-1) + dy(2:k)) ./ (dx(1:k-1) + dx(2:k))$$

Folosirea diferențelor finite duce la obținerea valorilor derivatei funcției f în punctele șirului (cu excepția unuia sau a ambelor capete ale șirului).

Exemplu:

Fie f o funcție 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 funcției f în punctele 1.5, 2.1 și 3.2, folosind diferențe regresive.

Soluție:

```
% 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 execuției acestei secvențe se obține:

```
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 și 1.45 derivata de ordinul I a funcției f precizată de vectorul $y=[100\ 90\ 86\ 62\ 74\ 81]$ pe intervalul $[-2:3]$:

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

valori, derivata se va aproxima prin interpolare cu funcții spline.

Se execută următoarea secvență 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
```

și se obțin 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 funcția f dată prin puncte de relațiile

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

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

Soluție: Se execută următoarea secvență 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.

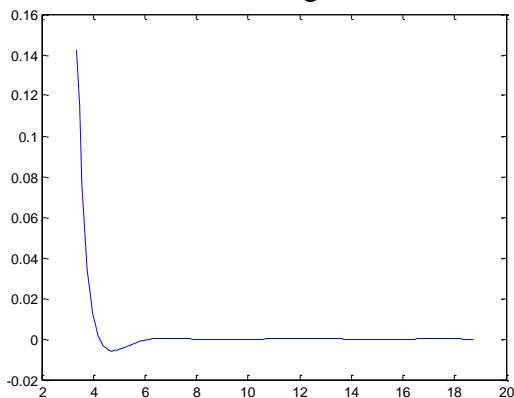


Fig.1.1. Graficul derivatei funcției

Pentru derivarea funcțiilor polinomiale, MATLAB pune la dispoziția utilizatorului o funcție specială numită `polyder`, cu sintaxa de apel:

`dc=polyder(c)`

unde:

- `c` reprezintă vectorul coeficienților funcției polinomiale, în ordine descrescătoare după puterea variabilei funcției;
- `dc` reprezintă vectorul coeficienților funcției polinomiale

obținute prin derivare, tot în ordine descrescătoare după puterea variabilei funcției.

Observații:

1. Derivarea produsului a două funcții polinomiale se poate face tot cu funcția MATLAB `polyder`, folosind sintaxa de apel:

`dc=polyder(c1,c2)`

`c1` fiind vectorul coeficienților primei funcții, iar `c2` vectorul coeficienților celei de a doua funcții.

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

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

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

Exemplu:

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

Să se calculeze derivata funcției în punctele -3, 0.2 și 4. Să se reprezinte grafic derivata funcției f pe intervalul $[-4,5]$.

Soluție: Se execută următoarea secvență de instrucțiuni 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 obțin 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

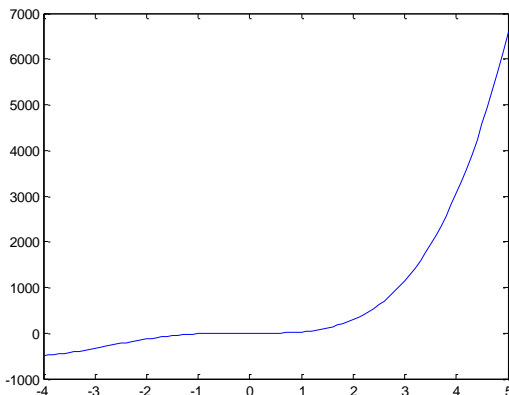


Fig.1.2. Graficul derivatei funcției polinomiale

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

Observație: 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 funcția f :

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

Soluție: Funcția, fiind cunoscută prin puncte, se va folosi metoda trapezelor pentru calculul integralei cerute. Se execută următoarea secvență 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:

$I = -0.0025.$

Funcția MATLAB `quad` realizează calculul integralei definite a unei funcții 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 funcția MATLAB). Funcția `quad` presupune că funcția de integrat f este cunoscută prin expresia sa analitică, $y = f(x)$. Două sintaxe de apel ale funcției `quad` sunt:

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

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

unde:

- `nume_fisier` reprezintă un șir de caractere ce conține numele fișierului-funcție în care a fost scrisă expresia funcției de integrat f ;

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

- `precizia` este un argument opțional 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 $\int_0^\pi \ln(x+1)\sin(x)dx$.

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

definește expresia funcției de integrat într-un fișier-funcție (de exemplu, f.m):

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

Pentru calculul integralei se apelează funcția MATLAB quad:

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

rezultând valoarea:

```
I = 1.8113.
```

E 1.7. Exercițiu:

1. Fie f o funcție 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 funcției f în punctele:

- a) -2, 0 și 2.5, folosind diferențe progresive;
- b) -1.5, 0 și 3, folosind diferențe regrese;
- c) -0.75, 0.25, 0.5 și 1, folosind diferențe centrate.

Să se reprezinte grafic în aceeași fereastră grafică derivata de ordinul I a funcției f , obținută prin aproximare cu cele trei tipuri de diferențe finite.

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

a) Să se calculeze derivata funcției f pentru valorile -2, -1.3, 0.1 și 2.45.

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

3. Să se calculeze $\int_0^\pi f(x)dx$, unde funcția f este dată prin relațiile:

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

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 posibilităților sistemului MATLAB în modelarea diferitor forme de semnale în vederea cercetării particularităților lor de bază.*

2.1. Introducere

Se numește **semnal** o mărime fizică măsurabilă, purtătoare de informație, care poate fi transmisă la distanță, recepționată și/sau prelucrată. În cele ce urmează se va aborda problema modelării formei semnalelor, fără a ne preocupa de conținutul în informație al semnalelor.

Un semnal **unidimensional**, numit și semnal 1D, este o funcție de timp, notată generic prin $x(t)$, $t \in \mathbb{R}$. De regulă, mărimea fizică variabilă, reprezentând semnalul, este o tensiune electrică. Totuși, în echipamentele de automatizări se utilizează și semnale de altă natură fizică, așa cum sunt, de exemplu: curentul electric, presiunea aerului instrumental, deplasarea unui corp solid. În telecomunicații, 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 și $y(t)$ semnalul obținut la ieșirea acestuia, numit și *răspuns* al sistemului la semnalul de intrare (fig. 2.1). Sistemele dinamice realizează prelucrarea semnalelor, conform cu funcțiunile realizate de echipamentele electronice în care sunt înglobate.

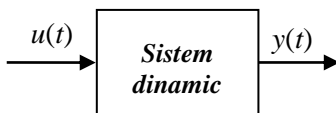


Fig. 2.1. Sistem dinamic

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

Semnalele pot fi *cu timp continuu* și *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 obține din cele analogice prin două operații:

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

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

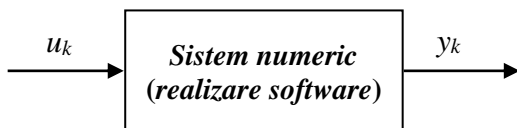


Fig. 2.2. Sistem numeric

Semnalele care au o evoluție ce nu este supusă hazardului se numesc **semnale deterministe**. Alături de acestea, se întâlnesc și semnalele **aleatoare**, a căror evoluție în timp este supusă hazardului, așa cum sunt perturbațiile care afectează sistemele de transmitere și prelucrare a informațiilor.

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

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

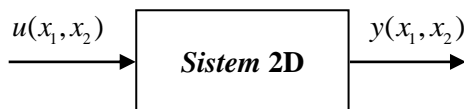


Fig. 2.3. Sistem 2D

Uneori este necesar să se extragă principalele proprietăți ale semnalelor (proprietățile cheie) prin aplicarea, asupra acestora, a unor algoritmi de prelucrare. Este posibil să se identifice proprietățile sistemului discret prin analiza semnalelor de ieșire la aplicarea anumitor semnale de intrare ale sistemului studiat. În acest caz, este necesar, pentru început, de a învăța generarea unor semnale de bază în mediul MATLAB și de a efectua operații elementare asupra lor, ceea ce determină sarcina inițială a acestei lucrări de laborator.

2.2. Formarea semnalelor impuls unitare

Funcția impuls Dirac sau **funcția delta**, numită și „impuls” unitar, notată $\delta(x)$, 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 obține un impuls de amplitudine infinită și de durată nulă, notat cu $\delta(t)$. Aria impulsului este egală cu 1 (de aceea se numește unitar), adică:

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

definește „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)' )
```

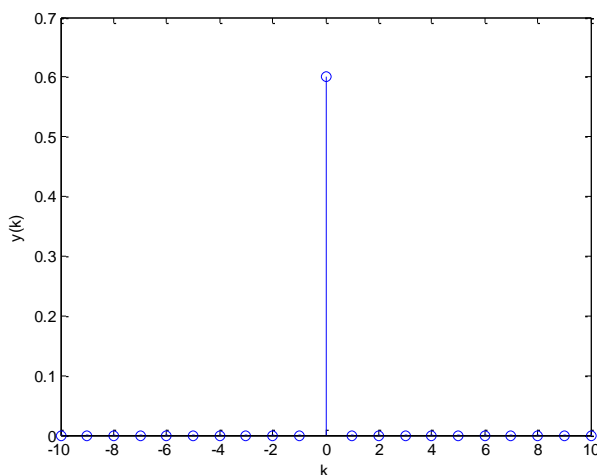


Fig.2.4. Funcția „impuls” unitar

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

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

Următorul program generează și 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]);
```

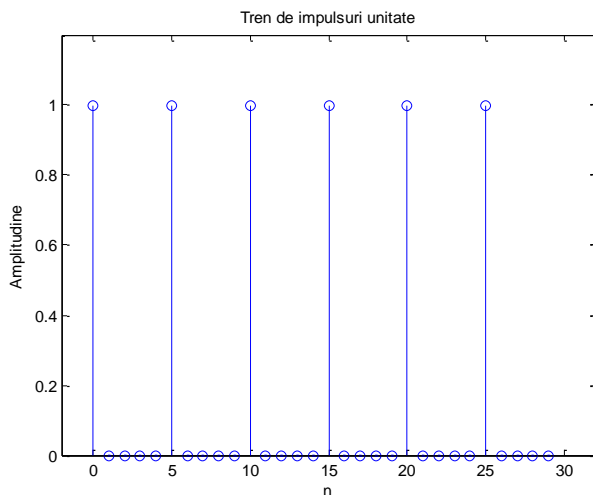


Fig.2.5. Tren de impulsuri unitare

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]);
```

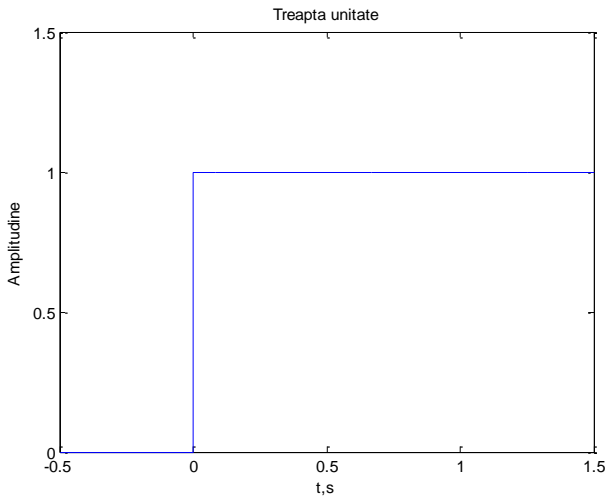


Fig.2.6. Semnalul treaptă unitară

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)')
```

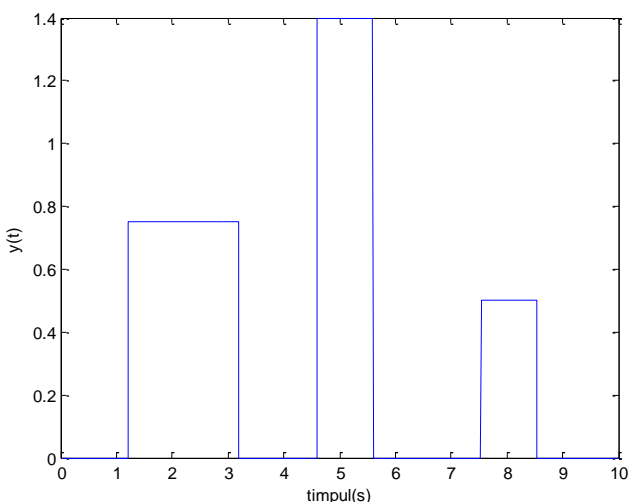


Fig.2.7. Impulsuri dreptunghiulare

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 același sens ca și în exemplul de mai sus. Argumentul s ($-1 \leq s \leq 1$) determină înclinarea triunghiului. Dacă $s=0$, sau dacă nu este indicat, atunci impulsul triunghiular este simetric. Funcția `y=tripuls(t,w)` formează un impuls simetric, de amplitudine unitară și durată w , centrat față 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)')

```

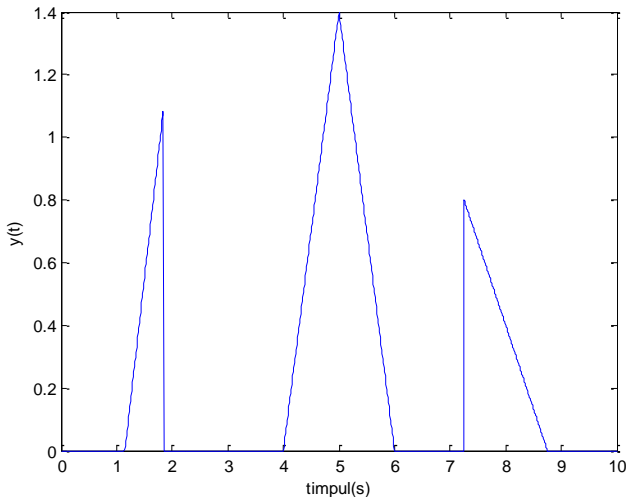


Fig.2.8. Impulsuri triunghiulare

Impulsul sinusoidal modulat de funcția 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=gauspuls('cutoff',fc,bw,bwr,tpe)

```

Funcția **yi=gauspuls(t,fc,bw)** formează secvența eșantioanelor unui impuls, calculate în momentele de timp date de vectorul t ; fc [Hz] determină frecvența purtătoare a semnalului; bw - lărgimea relativă a benzii de frecvență a semnalului în dB (ia valori pozitive). Implicit, se utilizează următoarele mărimi: $fc=1000\text{Hz}$ și $bw=0.5\text{ dB}$ ($bw=\Delta f/f_c$).

Funcția **yi=gauspuls(t,fc,bw,bwr)** calculează secvența eșantioanelor semnalului cu următorii parametri: amplitudinea unitară, lățimea relativă a benzii de frecvență 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, $bwr = -6 \text{ dB}$.

Funcția **[yi,yq]=gauspuls(...)** calculează doi vectori. Vectorul yi conține eșantioanele semnalului inițial, iar vectorul yq conține eșantioanele semnalului cu un defazaj de 90° față de cel inițial.

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

Funcția **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 înfășurătoarei în raport cu nivelul ei maximal. Implicit $tpe = -60 \text{ dB}$.

Construim curba impulsului Gaussian cu frecvența purtătoare 50 kHz și banda relativă de frecvență 60%, considerând pasul de calcul 1 MHz. Pentru calculul și construcția impulsului se utilizează intervalul de timp, pe durata căruia amplitudinea impulsului se micșorează 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)')
```

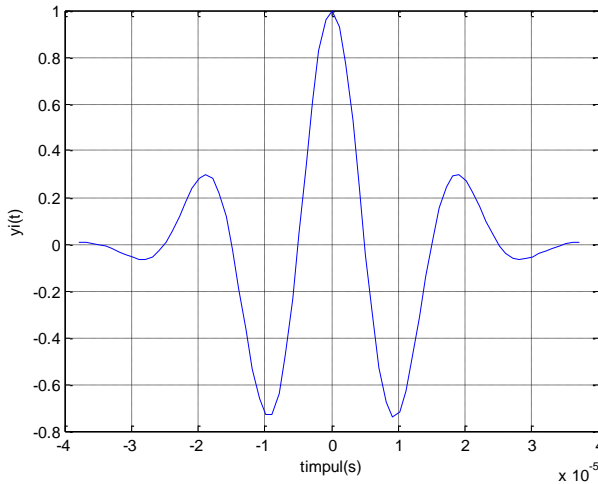


Fig. 2.9. Impuls Gaussian

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

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

Această funcție reprezintă o transformare inversă Fourier a impulsului dreptunghiular de amplitudine unitară și 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)')
```

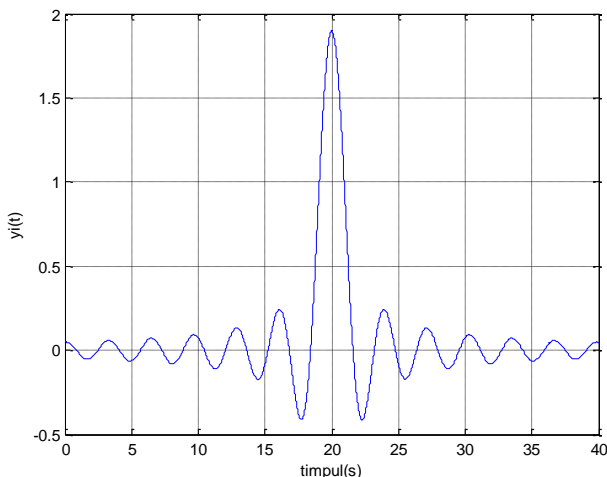


Fig. 2.10. Funcția $\text{sinc}(t)$

2.3. Formarea oscilațiilor periodice

Semnalele armonice și poliarmonice pot fi generate în MATLAB cu ajutorul operatorilor trigonometrice $\sin(x)$ și $\cos(x)$.

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].

Observație: 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 – pulsația [rad/s].

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

- f – frecvența semnalului [Hz];
- T – perioada semnalului [s]. *Observație:* $f = 1/T$;
- φ_0 – faza inițială [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;
```

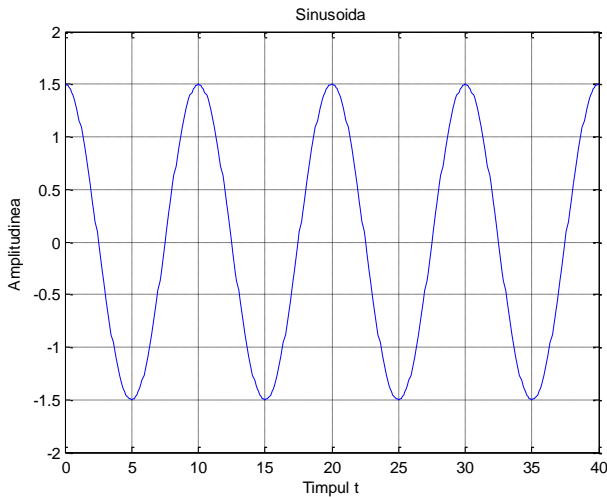


Fig.2.11.Semnal sinusoidal

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

zgomot cu distribuție normală, valoarea medie zero și varianța 0.5V, folosind o frecvență 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
```

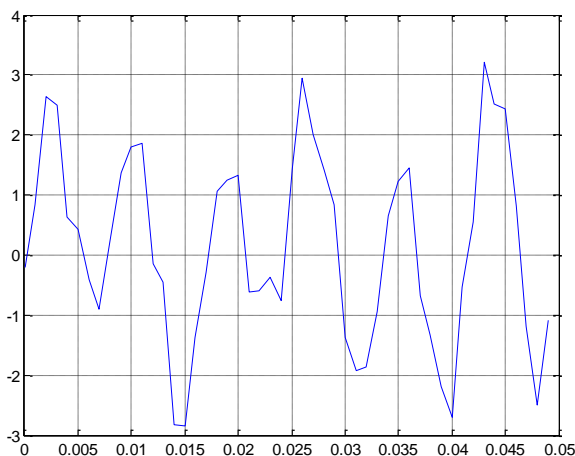


Fig.2.12. Semnal poliarmonic afectat de zgomot

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

```
y=square(t);  
y=square(t,duty).
```

Funcția **square(t)** formează oscilații dreptunghiulare de perioadă 2π și amplitudine ± 1 .

Funcția **y=square(w*t,duty)** formează o secvență de impulsuri dreptunghiulare cu durata semiundei pozitive, determinată de parametrul `duty` (coeficient de umplere) în procente față de perioadă și frecvența 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
```

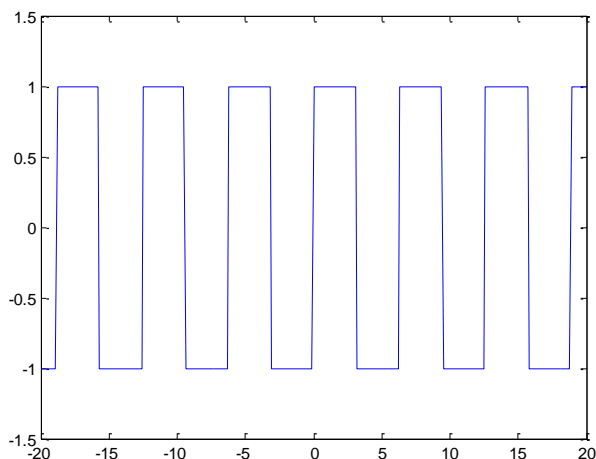


Fig.2.13. Secvență periodică de impulsuri rectangulare

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

Funcția **x=sawtooth(t)** formează semnalul de forma "dinte de fereastră", care ia valoarea -1 în momentele de timp multiple cu 2π și crescând linear pe intervalul 2π cu panta $1/\pi$.

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

$2\pi * \text{width}$, iar pe urmă scade de la 1 până la -1 pe intervalul de la $2\pi * \text{width}$ până la 2π . Dacă $\text{width}=0.5$, atunci se formează o undă simetrică.

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

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

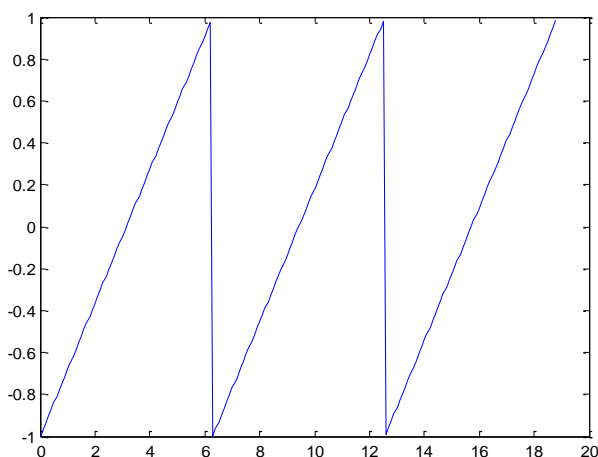


Fig.2.14. Secvență periodică de impulsuri "dinte de ferestru"

Procedura **pulstran** permite a forma oscilații care sunt secvențe 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 și poate avea una

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

Funcția **$y = \text{pulstran}(t, d, p, F_s)$** permite a determina impulsul cu secvența eșantioanelor date în vectorul p . Frecvența de discretizare este dată de parametrul F_s . La folosirea funcției **$y = \text{pulstran}(t, d, p)$** , frecvența 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
```

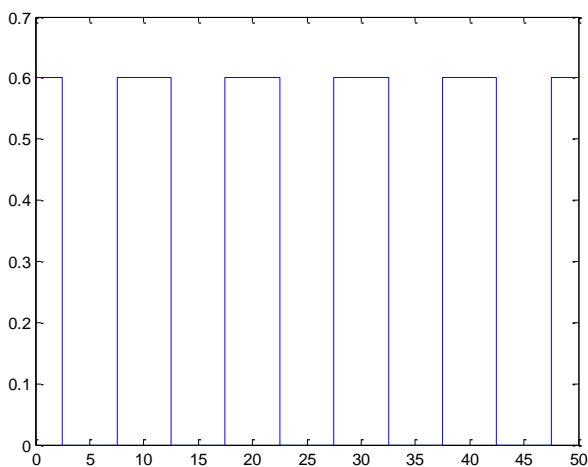


Fig. 2.15. Utilizarea funcției **pulstran** în cazul impulsurilor dreptunghiulare

Pentru o succesiune de impulsuri triunghiulare:

```
t=0:0.01:50;  
d=[0:10:50];  
y1=0.8*pulstran(t,d,'tripuls',5);  
plot(t,y1),  
grid
```

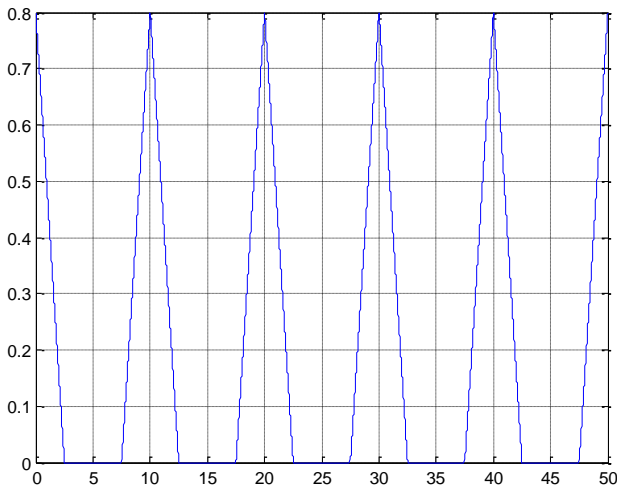


Fig. 2.16. Utilizarea funcției `pulstran` pentru o succesiune periodică de impulsuri triunghiulare

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
```

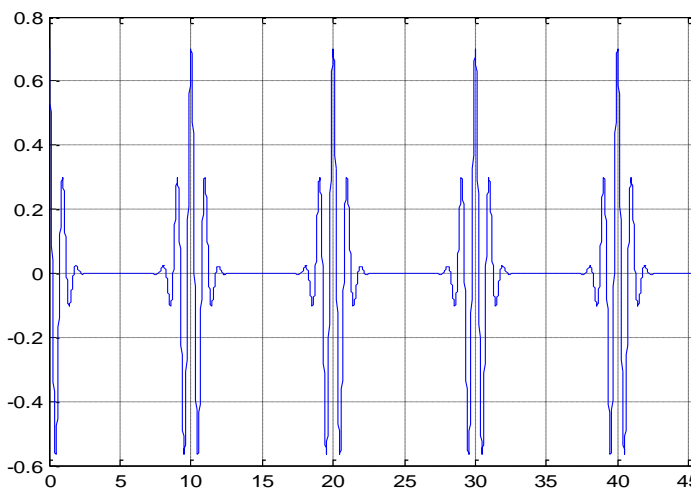


Fig. 2.17. Utilizarea funcției `pulstran` pentru o succesiune periodică de impulsuri gaussiene.

O sinusoidă, frecvența căreia variază liniar în timp, se generează cu ajutorul procedurii `chirp`:

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

Aceasta formează eșantioanele semnalului sinusoidal, frecvența căruia crește liniar pentru momentele de timp date de vectorul t ; f_0 - frecvența instantanee în momentul de timp $t=0$; f_1 - frecvența instantanee în momentul de timp $t=1$. Frecvențele f_0 și f_1 se dau în Hz. De exemplu, pentru $f_0=0$, $t_1=1$, $f_1=100$, utilizăm următorul script:

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

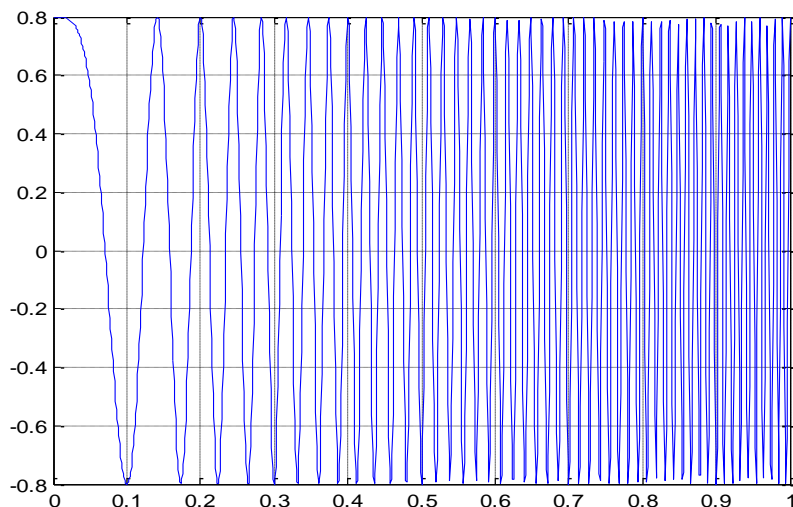


Fig. 2.18. Formarea unei cosinusoide cu frecvență variabilă în timp

2.4. Formarea semnalelor exponențiale

Unul dintre principalele tipuri de semnale elementare este **semnalul exponențial**. Acest semnal poate fi generat cu ajutorul următorilor operatori MATLAB: `.^` și `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 exponență.

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 exponențial 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

```

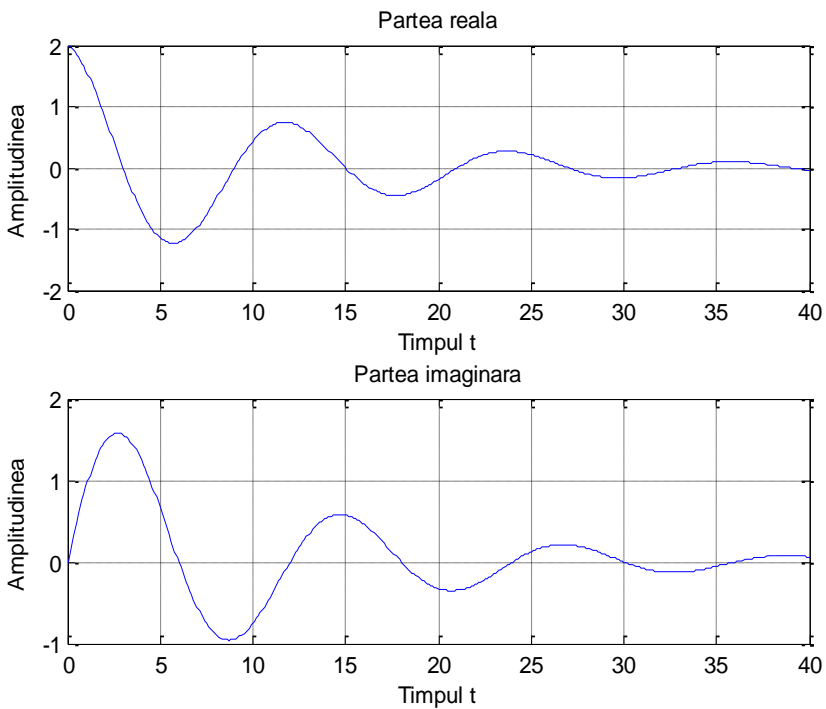


Fig. 2.19. Semnal exponențial complex

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')
```

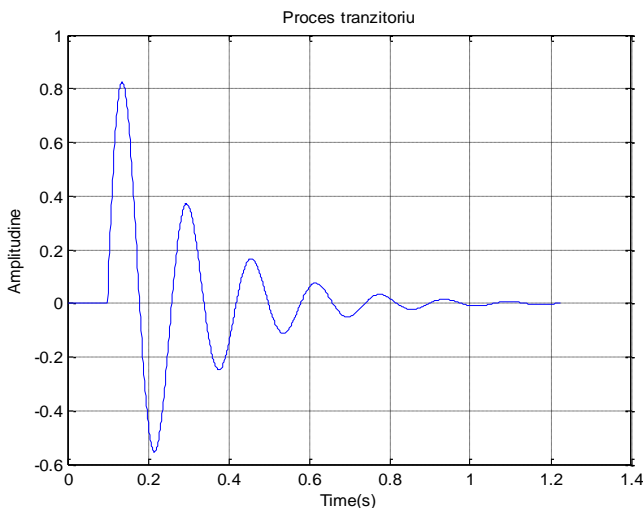


Fig. 2.20. Semnal sinusoidal variind exponențial

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
```

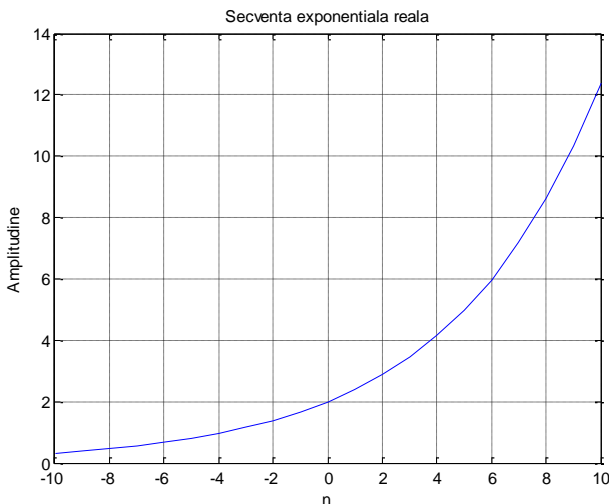


Fig. 2.21. Secvență exponențială reală

E 2.1. *Exerciții:*

1. Să se genereze și să se reprezinte următoarele secvențe.

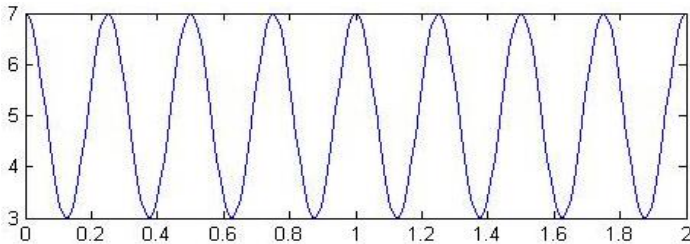
$x_1[n]=0.8 \delta[n],$	$-15 \leq n \leq 15$
$x_2[n]=0.9 \delta[n-5],$	$1 \leq n \leq 20$
$x_3[n]=1.5 \delta[n-333],$	$300 \leq n \leq 350$
$x_4[n]=4.9 \delta[n+7],$	$-10 \leq n \leq 0$
$x_5[n]=4 u[n],$	$-10 \leq n \leq 10$

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

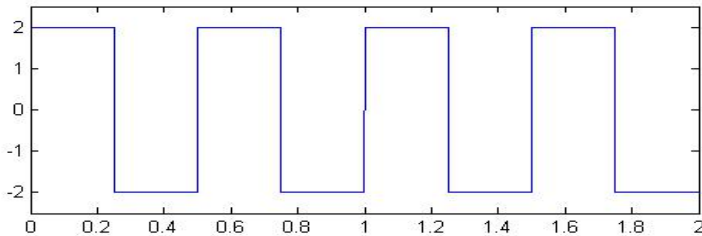
$$x_7[n] = 2.3 u[n+5], \quad -15 \leq n \leq 10$$

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

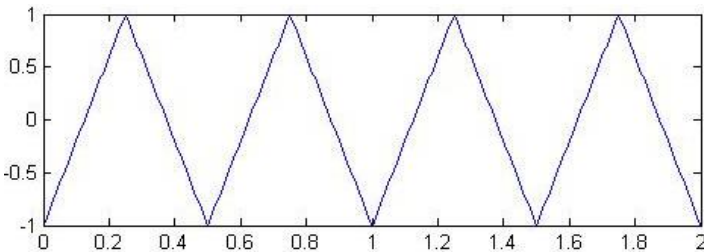
2. Să se genereze în MATLAB următorul semnal sinusoidal.



3. Să se genereze în MATLAB următorul semnal dreptunghiular.



4. Să se genereze în MATLAB următorul semnal triunghiular.



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

6. Să se creeze în intervalul $0s \leq t \leq 1s$ următoarele semnale exponențiale:

a) **$5\exp(-6t)$** ;

b) **$\exp(5t)$** ,

folosind frecvența rezoluției temporale $f_d = 1000Hz$.

7. Să se creeze în intervalul $-10s \leq t \leq 10s$ un impuls exponențial descris de următoarea expresie $x(t) = Br^t$, unde $B=1$, $r=0.8$.

8. Să se creeze în intervalul $-10s \leq t \leq 10s$ un semnal sinusoidal dat de expresia:

$$x(t) = 2\sin\left(\frac{2\pi}{12}t\right).$$

9. Să se creeze un semnal sinusoidal atenuat, pe baza înmulțirii 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 durată $1s$, 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 diferenței a două funcții de tip "treaptă unitară" deplasate în timp cu un interval egal cu durată 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 exponențial 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 și 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 frecvență 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: $k=0, 1, 2, \dots$, 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 frecvența de repetiție a semnalului periodic (frecvența 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.

Coeficienții 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 funcții cosinusoidale:

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

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

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

Forma complexă sau exponențială

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

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

Coeficienții C_n sunt mărimi complexe și, deci, pot fi reprezentați prin modul și fază:

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

Relațiile de legătură dintre coeficienții C_n și coeficienții seriilor trigonometrică și armonică sunt:

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

Alegerea limitelor de integrare în evaluarea coeficienților seriilor Fourier este arbitrară, esențial este ca integrarea să 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 frecvență a semnalului periodic se face prin reprezentarea spectrelor de amplitudini și faze. Se pot reprezenta fie diagramele spectrale asociate seriei armonice, $A_n(n\omega_0)$ și $\phi_n(n\omega_0)$, fie diagramele spectrale asociate seriei complexe, $|C_n(n\omega_0)|$ și $\phi_n(n\omega_0)$. Deoarece semnalele periodice sunt exprimate prin sume discrete de semnale elementare, rezultă că spectrele de amplitudini și faze vor fi discrete. La spectrele asociate seriei armonice, liniile spectrale vor fi localizate la frecvențele: $0, \omega_0, 2\omega_0, 3\omega_0, \dots$ etc., în timp ce la spectrele asociate seriei exponențiale liniile spectrale vor fi localizate la $0, \pm\omega_0, \pm 2\omega_0, \pm 3\omega_0, \dots$. Semnificația frecvențelor negative rezultă din funcția:

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

Două componente exponențiale, localizate simetric față de

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

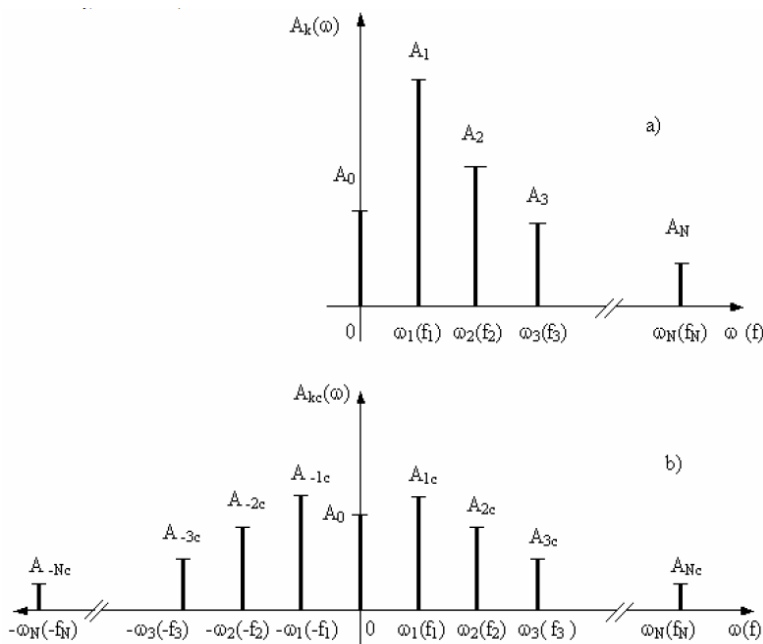


Fig.3.1. Diagramele spectrale de amplitudine ale semnalului $x(t)$ - a) SFA, b) SFC

Teoretic, spectrele semnalelor periodice se întind de la $f = 0$ la $f = \infty$. Practic, spectrele sunt limitate din cauza descreșterii 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 cerințele impuse tipului de comunicație care utilizează semnalul respectiv. Prin urmare, analiza spectrală a unui semnal permite a stabili lățimea benzii de frecvențe 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 successive. Evident, la limită, când $T \rightarrow \infty$, componentele frecvențiale se “contopesc”, iar spectrul semnalului devine de natură continuă. Astfel, trecerea la limită, când $T \rightarrow \infty$, aduce relația (3.8) la forma:

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

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):

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

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 sinusoidale.

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 secvențe $x[n]$ este dată prin relația:

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

unde ω este pulsația normalată:

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

F este frecvența nenormată (exprimată în Hz), iar F_s - *frecvența de eșantionare*.

Similar, frecvența normalată este:

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

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

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

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

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

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

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

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

În figură 3.2 se reprezintă spectrul unui semnal în funcție de pulsație sau frecvență normată și corespondența cu frecvența analogică. De asemenea, se observă corespondența dintre componentele spectrale de indice k calculate cu TFD și spectrul reprezentat în pulsații normate.

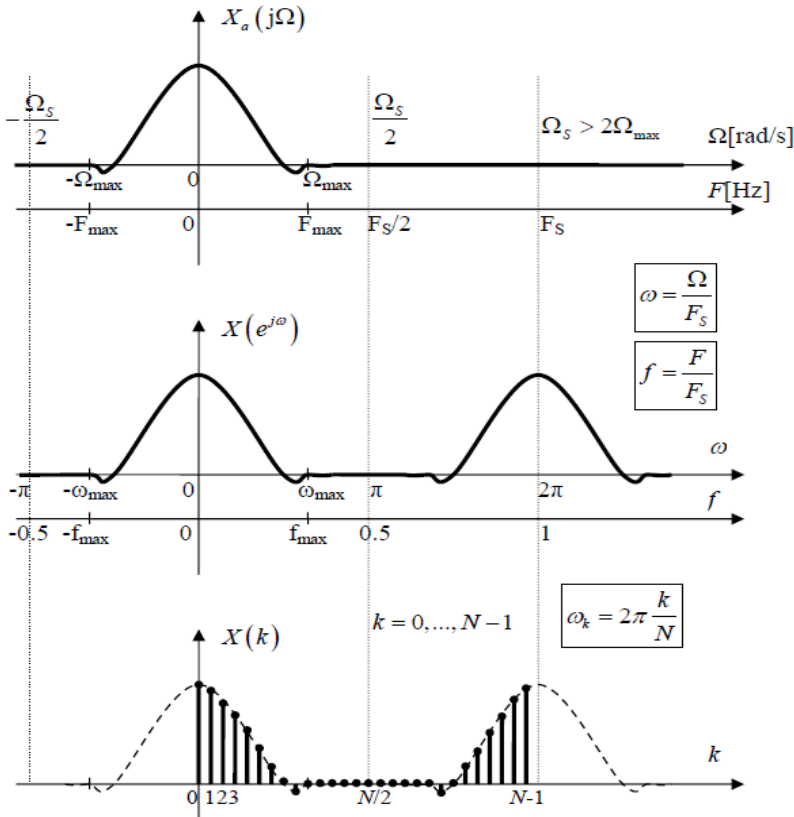


Fig.3.2. Spectrul unui semnal determinat în baza transformatei Fourier analogice în timp discret și a transformatei Fourier discrete

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

Sintaxe:

y = fft(x)

- dacă x este un vector, se returnează un vector y de aceeași dimensiune cu vectorul x ce conține 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 aceeași dimensiune cu matricea x ; coloana i din matricea y va conține valorile transformatei Fourier discrete aplicată elementelor coloanei i din matricea x .

y = fft(x,N)

- aceleași considerente ca în sintaxa precedentă cu deosebirea că în acest caz se specifică și numărul de puncte N în care se calculează transformata. Dacă $\text{length}(x) < n$, elementele masivului x se completează cu zerouri. Dacă $\text{length}(x) > n$, elementele în plus se șterg.

x = ifft(y)

- calculează transformata Fourier inversă consecutivității 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 $\text{length}(y) < N$, elementele ce lipsesc ale masivului se completează cu zerouri. Dacă $\text{length}(y) > N$, elementele de prisos se elimină.

y = fftshift(x)

- efectuează regruparea masivului de ieșire al transformatei

Fourier pentru construirea graficelor spectrelor semnalelor în formă obișnuită, cu deplasarea frecvenței nule în centrul spectrului.

Pentru a folosi procedura **fft** cu scopul de a obține prezentarea semnalului în domeniul frecvență, reieșind din reprezentarea lui în domeniul timp, este necesar a efectua următorii pași:

1. După valoarea aleasă a perioadei de eșantionare T_s , să se calculeze mărimea F_{\max} a gamei de frecvențe (în Hz) conform formulei:

$$F_{\max} = 1 / T_s.$$

2. În conformitate cu durata procesului T , să se calculeze df după formula:

$$df = 1/T.$$

3. După valorile calculate să se formeze vectorul valorilor frecvențelor în care se va calcula imaginea Fourier, de exemplu:

$$f = 0 : df : F_{\max}$$

Ca rezultat al utilizării procedurii **fft** se va obține valoarea vectorului y_k , adică TFD a succesiunii x_n dată în domeniul timp.

Pentru a obține spectrul semnalelor în formă obișnuită, cu deplasarea frecvenței nule în centrul spectrului, este necesar a mai efectua următorii pași:

1. Rezultatele acțiunii procedurii **fft** se înmulțesc cu factorul T_s și se supun acțiunii procedurii **fftshift**, care schimbă cu locurile prima și a doua jumătate a vectorului obținut.
2. Vectorul frecvențelor se reconstruiește după algoritmul:

$$f = -F_{\max}/2 : df : F_{\max}/2.$$

Afișarea 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 ferestru”.

La execuție, programul cere următoarele argumente: T - perioada, N - numărul de armonici pentru aproximare, precum și tipul semnalului $s(t)$: sinusoidal, dreptunghiular sau „dinte de ferestru”. 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 ferestru[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. trigonometrice
    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).

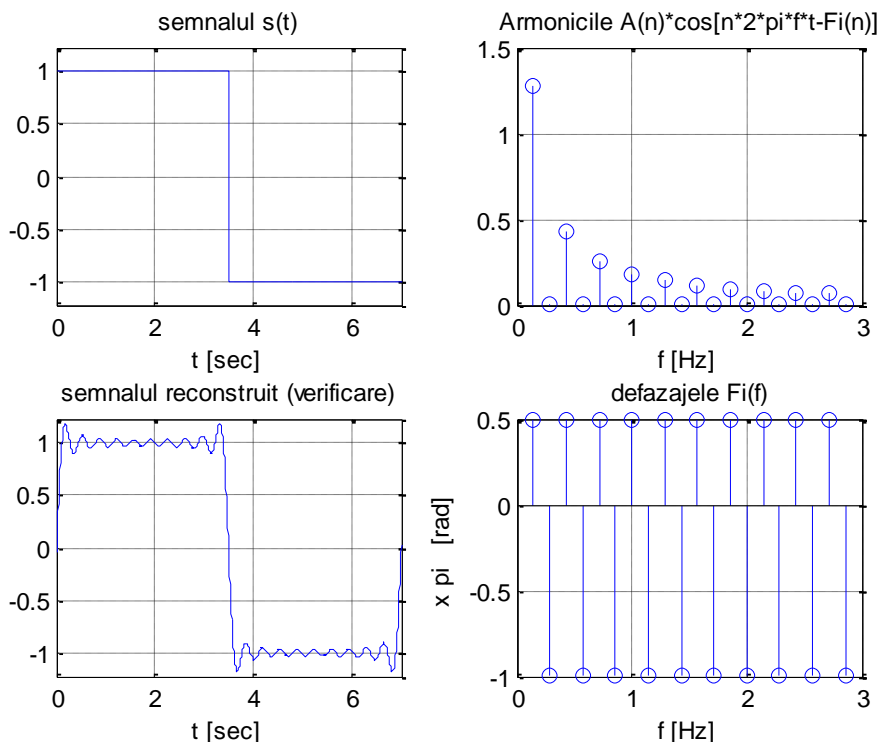


Fig.3.3. Descompunerea în serie Fourier a unui semnal dreptunghiular aproximat prin 20 de armonici

E3.1. Exercițiu:

Utilizând scriptul din *Exemplul 3.1*, să se efectueze descompunerea în serie Fourier a trei tipuri de semnale periodice: armonic, dreptunghiular și dinte de ferestru, pentru două valori 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 execuție, 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 $\tau=0,5s$, amplitudinea $A=5V$ (fig. 3.4).

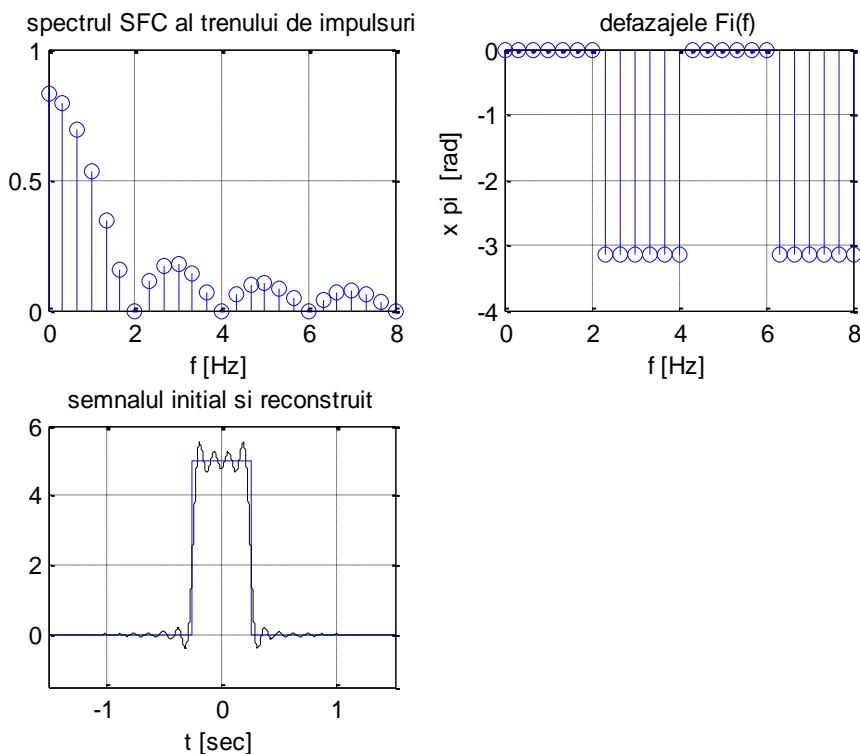


Fig.3.4. Descompunerea în serie Fourier complexă a unui tren de impulsuri dreptunghiulare

E3.2. Exercițiu:

Utilizând scriptul din *Exemplul 3.2*, studiați 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 oscilații armonice cu frecvențele 1, 2 și 3 Hz și amplitudinile respectiv 3, 1 și 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) și a părților reale și 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 și spectrului de amplitudine sunt reprezentate în figura 3.5.

Pentru construcția spectrului de fază 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 afișare a graficelor **stem** și **ylabel**.

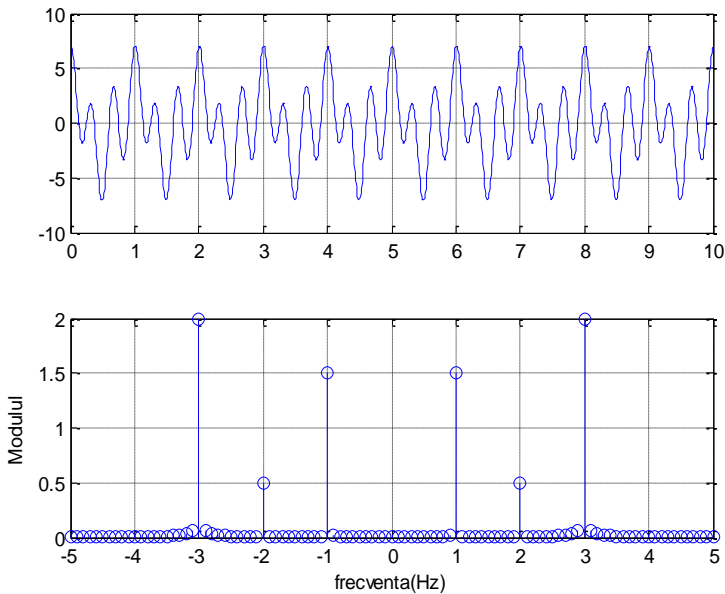


Fig.3.5. Modulul spectrului semnalului poliarmonic
corespunzător seriei Fourier complexe

Vom evidenția partea reală și cea imaginară ale spectrului complex:

```
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')
```

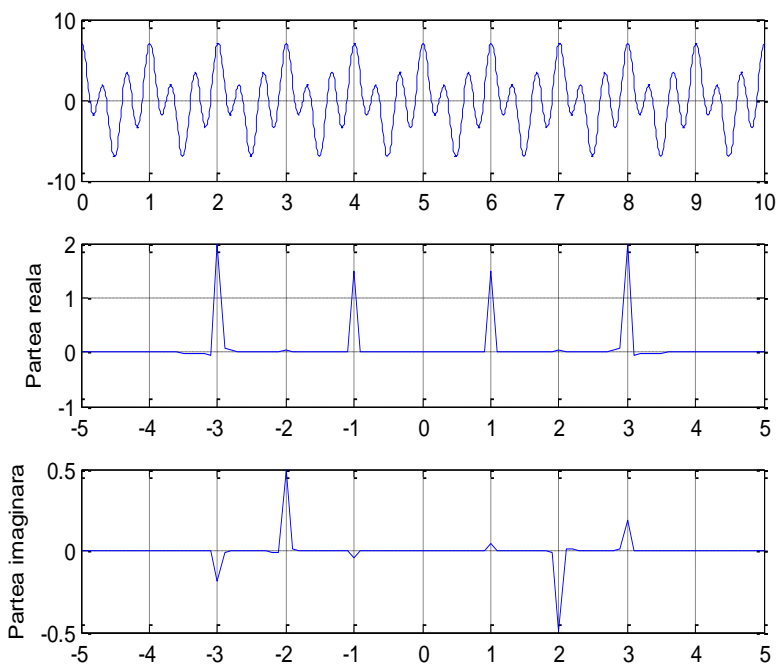


Fig. 3.6. Partea reală și cea imaginară ale spectrului complex al semnalului poliarmonic.

E3.3. Exercițiu:

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 τ 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 și funcția de densitate spectrală corespunzătoare sunt date în figura 3.7.

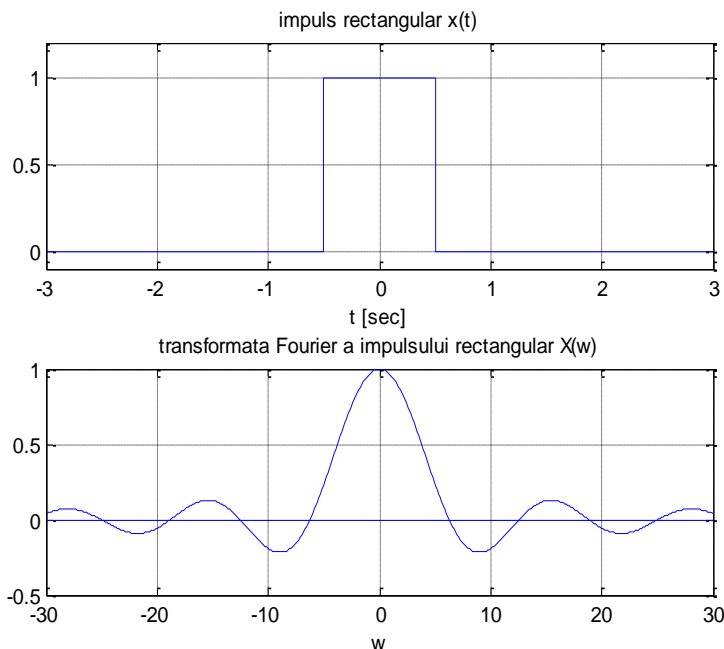


Fig. 3.7. Forma de undă și spectrul de frecvențe ale semnalului impuls dreptunghiular

Exemplul 3.5:

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

```
%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 și spectrului de amplitudini sunt date în figura 3.8.

Pentru construcția 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 afișare a graficelor **stem** și **ylabel**.

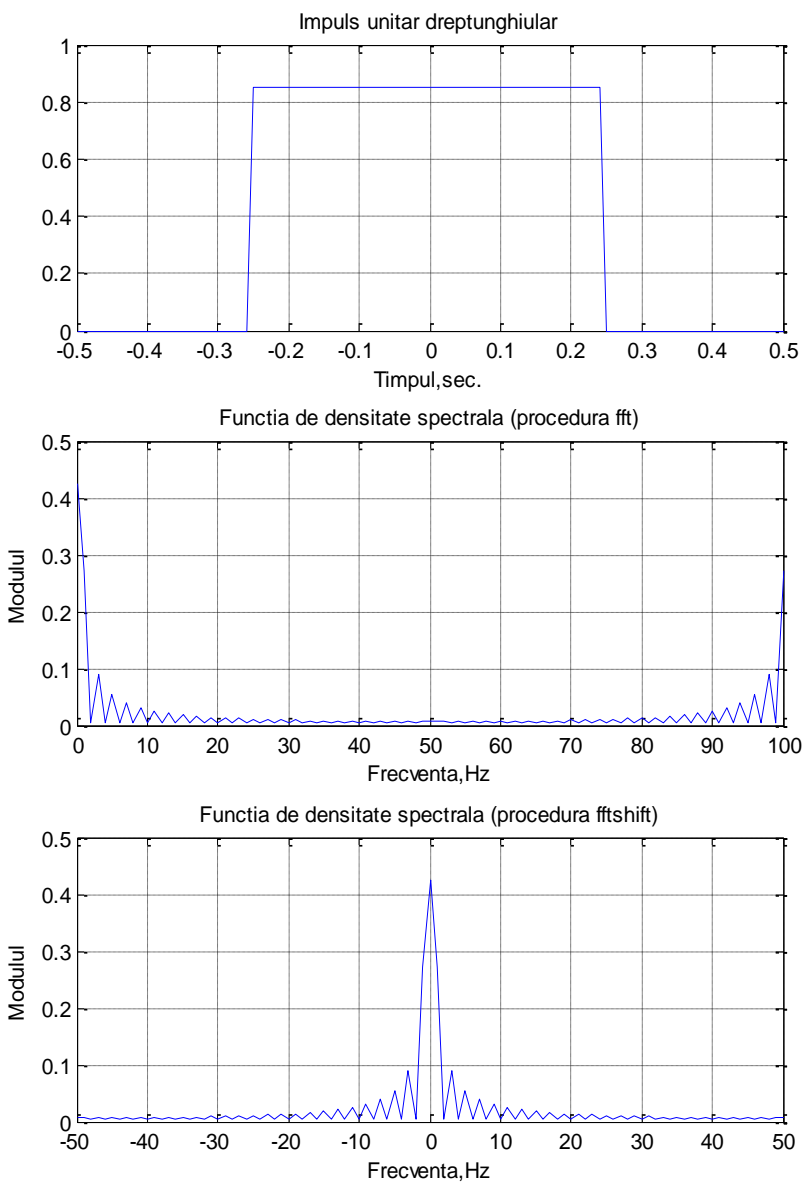


Fig. 3.8. Forma de undă și spectrul de frecvențe ale semnalului impuls dreptunghiular

E3.4. *Exercițiu:*

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 noțiunea de frecvență 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 particularități are densitatea spectrală a semnalului real?
6. Care este legătura dintre durata impulsului și lărgimea spectrului său?
7. Cum se calculează amplitudinile și fazele spectrelor semnalelor neperiodice?
8. În ce constă asemănarea și deosebirea spectrelor semnalelor discrete și analogice?
9. În ce constă și cum se manifestă suprapunerea spectrelor la discretizarea semnalului?

LUCRAREA nr. 4

EȘANTIONAREA ȘI CUANTIZAREA SEMNALELOR. INTERPOLAREA SEMNALELOR EȘANTIONATE

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 unui 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).

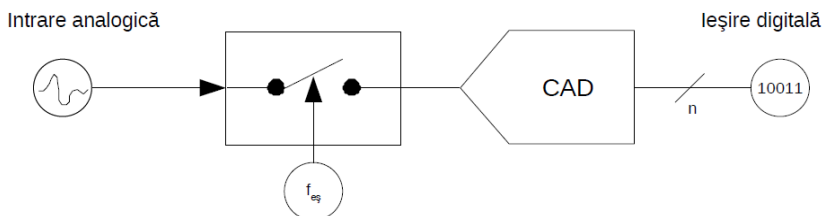


Fig. 4.1. Discretizarea semnalelor analogice

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

convectorul 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).

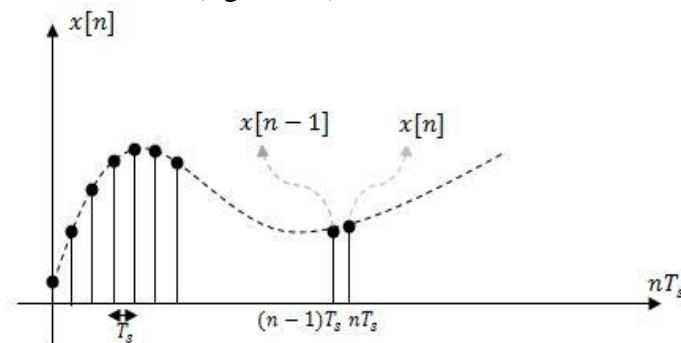


Fig. 4.2. Semnal eșantionat cu perioada de eșantionare T_s

Teorema eșantionării uniforme (Shannon) stabilește că un semnal de bandă limitată este univoc determinat de valorile sale considerate la momente de timp echidistante, dacă distanța T_s dintre două momente succesive satisface relația:

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

unde f_m reprezintă frecvența 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 eşantioanele sale, conform teoremei eşantionării, trebuie satisfăcut criteriul Nyquist şi anume că frecvenţa de eşantionare trebuie să fie cel puţin dublul frecvenţei maxime f_m din spectrul semnalului:

$$f_s \geq 2 \cdot f_m \quad (4.1)$$

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

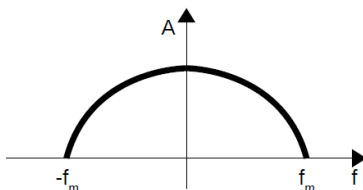


Fig. 4.3. Exemplu de spectru mărginit

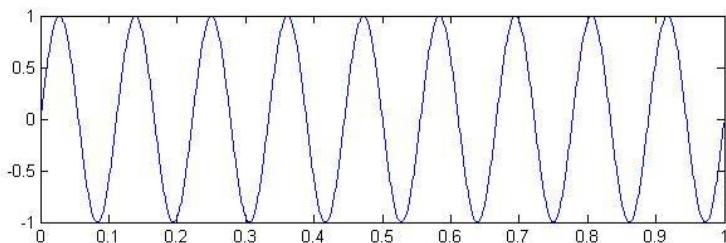
Observații:

- dacă dorim ca semnalul eşantionat să se apropie cât mai mult de semnalul real, atunci trebuie să luăm un număr cât mai mare de eşantioane, deci, să creştem frecvenţa de eşantionare f_s . Dezavantajul constă în spaţiul mare de memorie ocupat;
- dacă dorim un număr mai mic de eşantioane (spaţiu de memorie ocupat mai mic, dar şi o posibilă pierdere de informaţie), atunci putem scădea frecvenţa de eşantionare f_s . Însă pentru a putea refăce semnalul real din eşantioanele sale, frecvenţa de eşantionare nu poate fi scăzută oricât, ea trebuie să respecte teorema eşantionării.

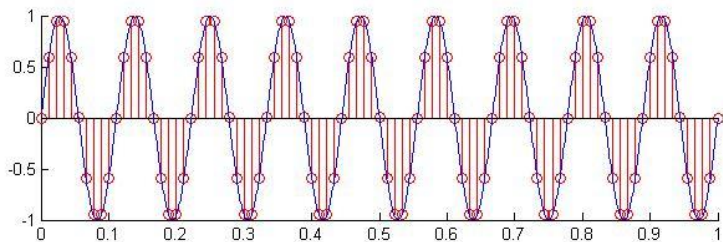
Exemplul 4.1:

Fie un semnal sinusoidal $x(t)$ având frecvenţa $f = 9\text{Hz}$ (figura 4.4, a). Acest semnal a fost eşantionat cu $f_{s1}=90\text{Hz}$ (figura 4.4, b). Se observă că din eşantioanele rezultate poate fi refăcut $x(t)$. În figura 4.4, c, semnalul $x(t)$ a fost eşantionat cu $f_{s2} = 8\text{Hz}$ (nu se respectă teorema eşantionării) şi se poate observa că aceleaşi eşantioane rezultă atât prin eşantionarea semnalului $x(t)$, cât şi prin eşantionarea unei alte sinusoidă cu frecvenţă mai mică, de 1Hz . În

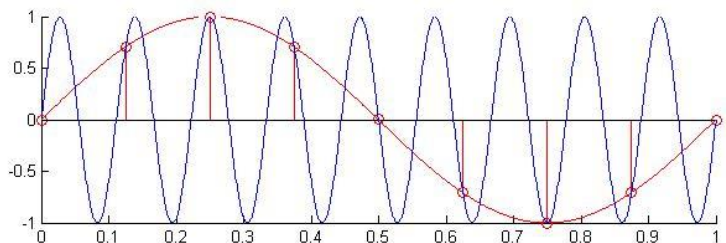
acest caz, semnalul $x(t)$ nu mai poate fi refăcut, formula de reconstituire generând sinusoida de 1Hz.



a) semnalul $x(t)$ cu frecvența $f = 9\text{Hz}$



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



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

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);
```

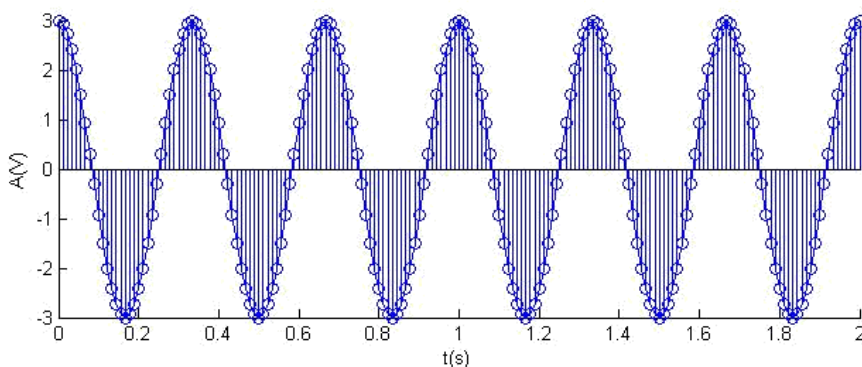
În acest caz, x va conține eșantioanele unui semnal sinusoidal, dar pentru a cunoaște semnalul căruia acest vector corespunde, trebuie să cunoaștem unitatea metrică pentru t și frecvența de eșantionare f_s . Pentru reprezentarea grafică a unui semnal discret se utilizează funcția `stem`.

E4.1. Exercițiu:

1. Reprezentați grafic 1024 de eșantioane ale unui semnal alcătuit din 2 sinusoidi (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 sinusoidi (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:

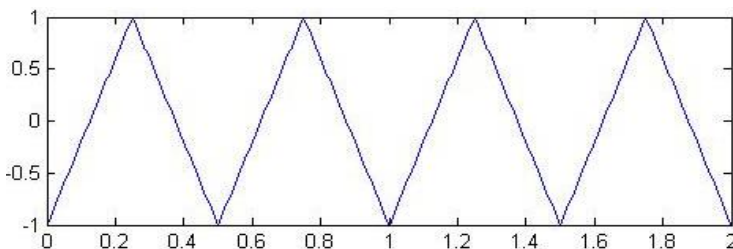
$$x(t) = 10 \sin\left(200\pi t + \frac{\pi}{2}\right) + 20 \sin(100\pi t) - 40 \sin\left(300\pi t - \frac{\pi}{4}\right).$$

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 $f_{s1} = 4\text{Hz}$ și cu $f_{s2} = 50\text{Hz}$ 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 eșantion i se alocă o valoare dintr-un set finit de valori. Distanța 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. Această 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 distanța dintre două niveluri de cuantizare consecutive, atunci valoarea unui eșantion 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 (eșantionul analogic) se află în domeniul de lucru al cuantizorului, eroarea de reprezentare variază în intervalul $-q/2 \div q/2$. Acest tip de eroare, denumită *eroare de cuantizare*, apare când valoarea eșantionului analogic este situată între două niveluri disponibile de cuantizare.

Cuantizarea prin trunchiere (floor)

Dacă notăm cu q distanța dintre două niveluri de cuantizare consecutive, atunci valoarea unui eșantion 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 $-q \div 0$.

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 $T_s = 0.05s$) se obține semnalul din figura 4.5, c (cuantizare prin rotunjire) și semnalul din figura 4.5, d (cuantizare prin trunchiere).

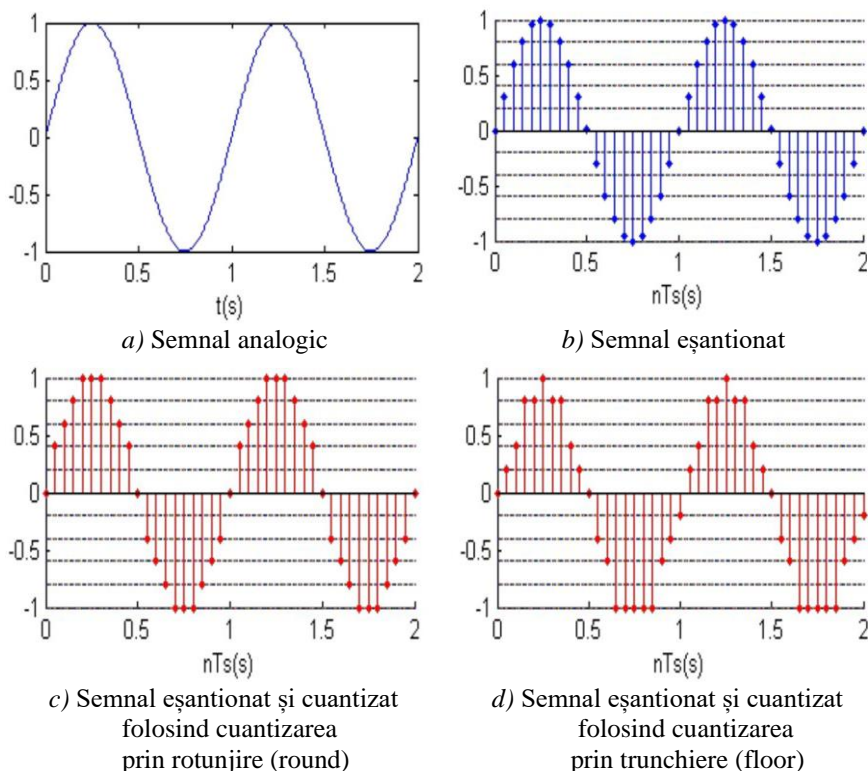


Fig.4.5. Cuantizarea prin rotunjire și trunchiere

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.

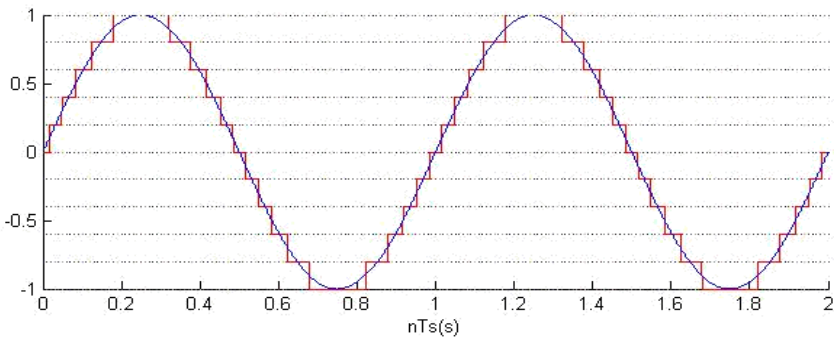


Fig.4.6, a. Semnal cuantizat folosind metoda prin rotunjire

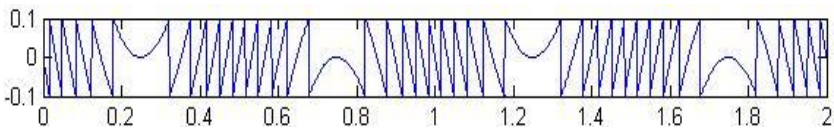


Fig.4.6, b. Zgomotul de cuantizare

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;
- $\lceil \cdot \rceil$ = 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. Exercițiu:

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 eșantioanele 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 relația de interpolare:

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

unde funcția *sinc* se definește astfel:

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

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

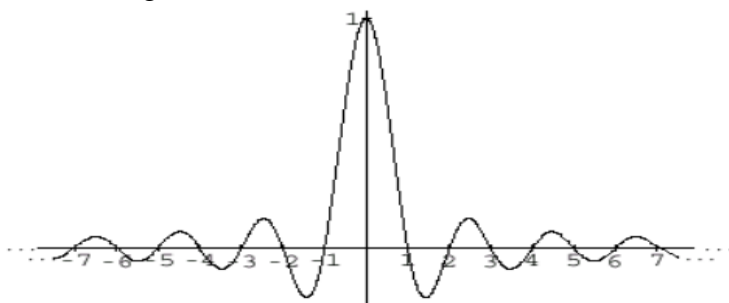


Fig.4.7. Reprezentarea grafică a funcției *sinc*(*x*)

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

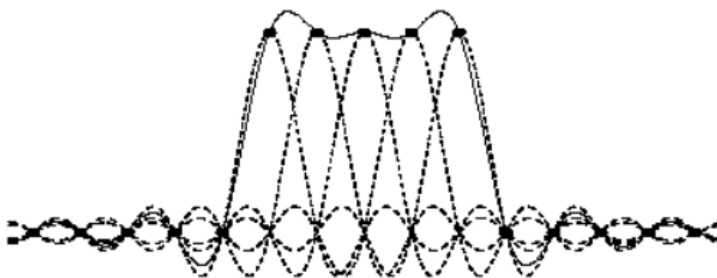


Fig.4.8. Reconstituirea unui impuls rectangular din eșantioanele sale

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* înmulţite cu valorile eşantioanelor [...0,1,1,1,1,0,...] formează curba cu linie plină din figură a semnalului rectangular eşantionat. Semnalul “reconstituit” este nenul acolo unde 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

```

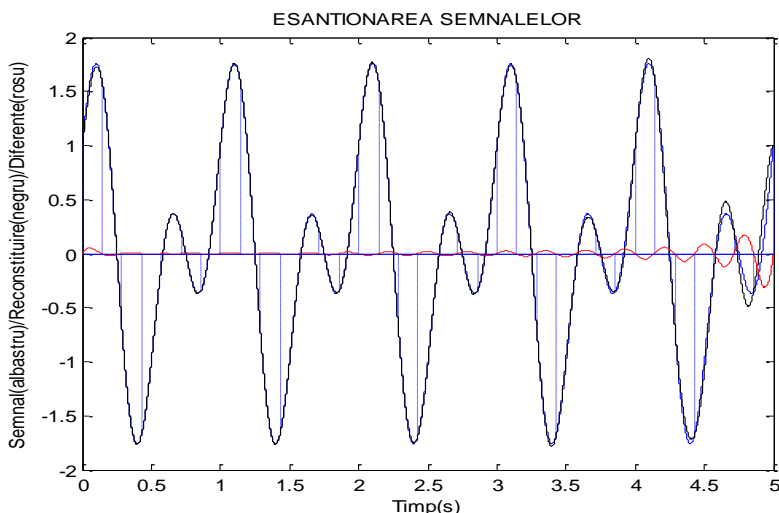


Fig.4.9. Eșantionarea și reconstrucția unui semnal, sumă a doua sinusoida

E4.3. Exercițiu:

Chestiuni de studiu în baza Exemplului 4.7.

- Executarea calculelor cu valorile recomandate în antetul *script*-ului, reprezentarea semnalelor în formă originală și după reconstrucție, precum și a diferențelor dintre ele, în spații grafice diferite sau în același spațiu.
- Executarea de calcule cu valorile diferite pentru frecvențele f_1 și f_2 , eventual pentru amplitudini diferite ale sinusoidelor componente, utilizând pentru fiecare caz trei valori ale frecvenței de eșantionare: mai mică, egală și mai mare (de 10 ori) ca frecvența Nyquist.
- Aprecierea vizuală și/sau cantitativ a modificărilor semnalului reconstituit, calitatea reconstrucției semnalului în zona de început și de final a intervalului de timp observat.
- Studierea și 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: aplicații didactice în MatLab*. Sibiu: Editura Universității "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.