

Ioan P. MIHU, Cătălina NEGHINĂ

Prelucrarea Digitală a Semnalelor

Aplicații didactice în Matlab

**Editura Universității “Lucian Blaga”
Sibiu, 2014**

Descrierea CIP a Bibliotecii Naționale a României

MIHU, IOAN P.

Prelucrarea digitală a semnalelor: aplicații didactice în MatLab /

Mihu Ioan P., Neghină Cătălina. - Sibiu : Editura Universității "Lucian Blaga" din Sibiu, 2014

Bibliogr.

ISBN 978-606-12-0796-1

I. Neghină, Cătălina

004.42 MATLAB

Copertă: Anna Enache

Toate drepturile acestei ediții sunt rezervate autorilor.

Prefață

În această lucrare sunt prezentate noțiunile fundamentale de prelucrare a semnalelor digitale, atât în domeniul timp cât și în domeniul frecvență, însoțite de exemple, exerciții și aplicații didactice. De asemenea, sunt prezentate elementele de bază ale programării în Matlab, un mediu de simulare flexibil și puternic folosit pe scară largă în mediul academic pentru cercetare. Prin implementarea parțială sau integrală în Matlab a exercițiilor și prin folosirea nu doar a semnalelor abstracte, dar și a semnalelor practice de tip sunet, imagine, EKG, acest îndrumar conferă o bună înțelegere a noțiunilor esențiale privitoare la prelucrarea digitală a semnalelor.

Lucrarea este structurată în 14 capitole și 5 anexe. Toate capitolele conțin:

- 1) o scurtă prezentare teoretică a noțiunilor
- 2) consolidarea noțiunilor prin exemple
- 3) exerciții și aplicații în Matlab

Anexele oferă informații suplimentare referitoare la realizarea interfețelor grafice în Matlab, posibilitatea citirii, redării și salvării semnalelor, construirea caracteristicii de frecvență și proiectarea filtrelor (FIR/IIR). Aplicația *DSP_Assistant* precum și alte aplicații implementate în Matlab sunt incluse pe CD-ul atașat acestui îndrumar.

Lucrarea este adresată în special studenților Facultății de Inginerie din cadrul Universității “Lucian Blaga” din Sibiu, ca suport pentru activitățile cursurilor *Procesarea Numerică a Semnalelor*, *Prelucrarea Digitală a Semnalelor*, *Proiectarea Filtrelor Numerice*, însă poate fi utilă nu doar studenților de profil electric, electronic, calculatoare și tehnologia informației, ci și oricărei persoane interesate de principiile prelucrării digitale a semnalelor.

Sibiu, 2014

Cuprins

Lucrarea 1. Noțiuni esențiale referitoare la filtrarea semnalelor	7
Lucrarea 2. Noțiuni introductive în Matlab	11
Lucrarea 3. Grafice și funcții în Matlab	21
Lucrarea 4. Interfață grafică în Matlab	31
Lucrarea 5. Conversia Analog Numerică. Teorema eșantionării.....	37
Lucrarea 6. Medierea și histograma.....	49
Lucrarea 7. Derivata/Diferențierea	57
Lucrarea 8. Corelația.....	65
Lucrarea 9. Convoluția.....	75
Lucrarea 10. Teorema Fourier	81
Lucrarea 11. Transformata Fourier Discretă (TFD)	87
Lucrarea 12. Proiectarea Filtrelor Nerecursive folosind TFTDI	95
Lucrarea 13. Proiectarea Filtrelor Nerecursive folosind Transformata Fourier Discretă Inversă ..	103
Lucrarea 14. Filtre Recursive (IIR).....	109
Anexa 1. Realizarea unei interfețe grafice în Matlab	119
Anexa 2. Citirea și redarea semnalelor în Matlab.....	123
Anexa 3. Construcția caracteristicii de frecvență	127
Anexa 4. Proiectarea filtrelor FIR folosind aplicația <i>DSP_Assistant</i>	133
Anexa 5. Proiectarea filtrelor IIR folosind aplicația <i>DSP_Assistant</i>	137

Lucrarea 1

Noțiuni esențiale referitoare la filtrarea semnalelor

Obiective: folosirea aplicației *DSP_Assistant* pentru: înțelegerea și determinarea caracteristicii de frecvență; filtrarea semnalelor cunoscând caracteristica de frecvență.

La modul general, un filtru este un bloc funcțional care are proprietăți selective față de elementele unei mulțimi: unele entități sunt lăsate să treacă iar altele sunt rejectate. Exemple de filtre din lumea reală: filtrul de cafea, sita, filtrele optice etc.

Pentru semnale electrice, filtrul are proprietăți selective față de tensiuni sinusoidale de frecvențe diferite și anume:

- Sinusoide de anumite frecvențe se propagă neatenuate
- Sinusoide de alte frecvențe vor fi atenuate
- Sinusoide de anumite frecvențe nu vor trece prin filtru

De ce folosim tensiuni sinusoidale? Pentru că sunt singurele semnale din natură care se propagă prin sisteme liniare fără să-și schimbe forma. **Sinusoidă intră, sinusoidă iese.** Sinusoida de la ieșire poate avea amplitudine diferită față de sinusoida de la intrare și poate fi defazată față de aceasta, dar tot o sinusoidă va fi. *Atenție!* Frecvența rămâne neschimbată.

Comportamentul filtrului față de tensiuni sinusoidale este dat de caracteristica de frecvență a filtrului. **Caracteristica de frecvență** a unui filtru este: exprimarea grafică a rezultatelor unui experiment simplu în care aducem la intrarea filtrului o singură sinusoidă de amplitudine unitară, a cărei frecvență o creștem lent. Pentru fiecare valoare a frecvenței măsurăm **amplitudinea** și **defazajul**, iar rezultatele le prezentăm sub formă grafică, obținând astfel: **caracteristica de amplitudine** a filtrului (modulul caracteristicii de frecvență), respectiv **caracteristica de fază** a filtrului.

Exemplu 1. În exemplul din *Figura 1*, se poate observa că dacă la intrarea filtrului se aduce o sinusoidă unitară având frecvența de 2256Hz, atunci la ieșire se va obține tot o sinusoidă unitară, deci valoarea caracteristicii de amplitudine pentru $F = 2256\text{Hz}$ va fi 1. Se obține în acest mod caracterica de amplitudine a unui *Filtru Trece Sus*.

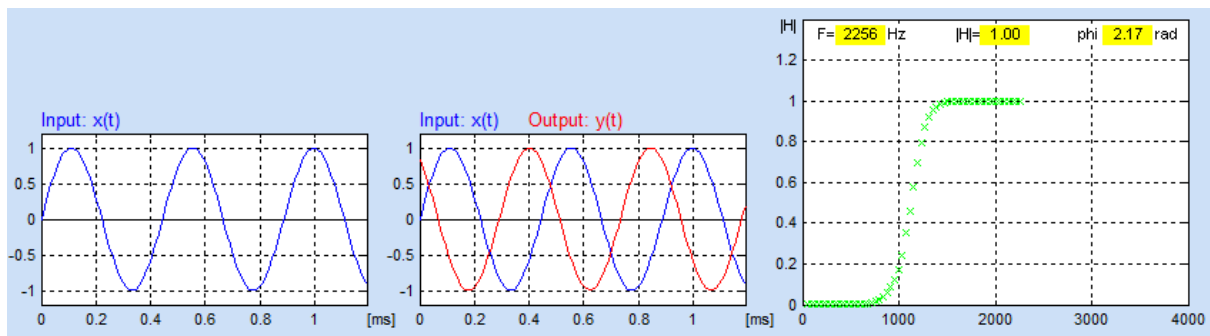


Figura 1. Caracteristica de amplitudine construită folosind aplicația *DSP_Assistant*

Exemplu 2. În exemplul din *Figura 2*, se poate observa că dacă la intrarea filtrului se aduce o sinusoidă unitară având frecvența de 1823Hz, atunci la ieșire se va obține o sinusoidă în antifază cu sinusoida de la intrare. Deci pentru $F = 1823\text{Hz}$, caracteristica de fază va avea valoarea -3.14 rad .

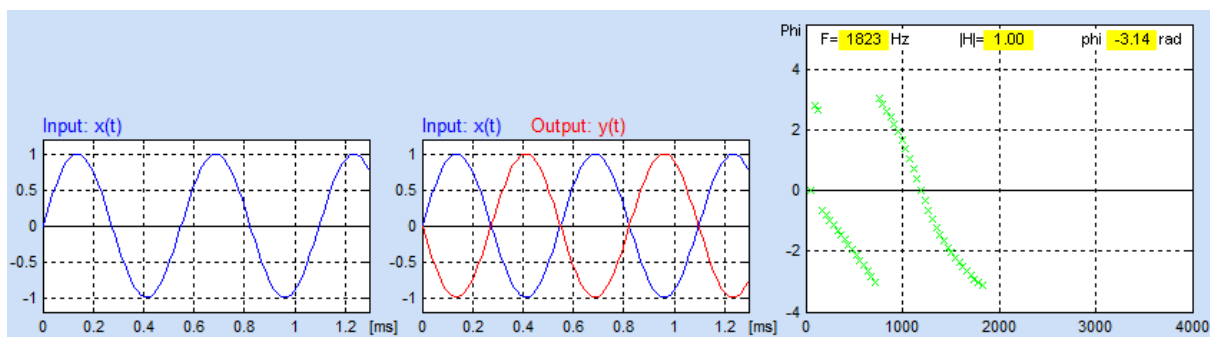


Figura 2. Caracteristica de fază construită folosind aplicația *DSP_Assistant*

Reciproc, dacă se cunoaște caracteristica de frecvență a unui filtru, aceasta ne permite să aflăm cum se comportă filtrul față de sinusoidale de frecvențe diferite aduse la intrarea sa.

Exemplu 3. Pentru un filtru având caracteristica de amplitudine ca cea din *Figura 3*, se observă că pentru frecvențe joase ($<800\text{ Hz}$) valoarea caracteristicii de amplitudine este 1, (în acest interval de frecvențe, la ieșirea filtrului se obține tot o sinusoidă unitară) urmează apoi o bandă de frecvențe în care valoarea caracteristicii de amplitudine scade (în acest interval de frecvențe, la ieșirea filtrului se obține o sinusoidă atenuată), iar de la frecvențe mai mari de 2000Hz valoarea caracteristicii de amplitudine este nulă (sinusoidalele cu frecvențe mai mari de 2000Hz vor fi rejectate). În aceste condiții, dacă se aduce la intrarea filtrului o sinusoidă cu frecvența de 1030Hz, atunci se va obține la ieșirea filtrului o sinusoidă atenuată, cu amplitudinea maximă de 0.66.

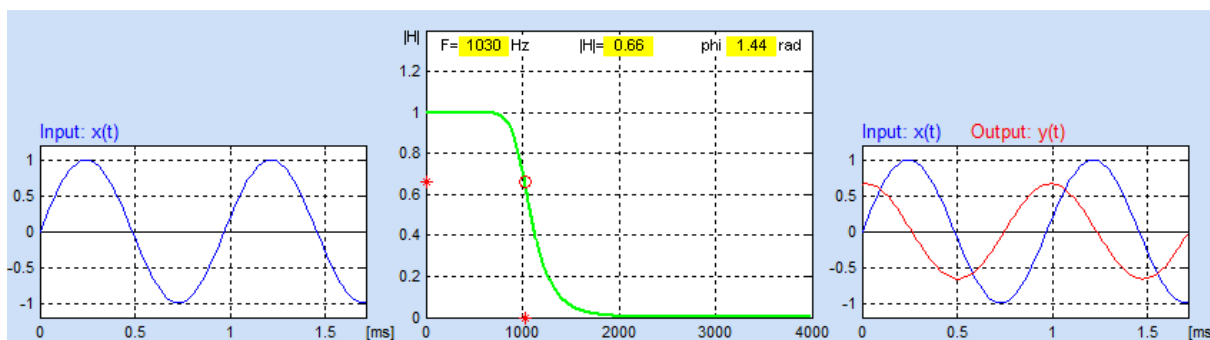


Figura 3. Răspunsul filtrului cunoscând caracteristica de frecvență (aplicația *DSP_Assistant*)

Caracteristica de frecvență arată comportamentul selectiv al filtrului față de sinusoidă de frecvențe diferite: le lasă să treacă mai mult sau mai puțin atenuate. În cele mai multe cazuri trebuie ca filtrul să fie descris la modul “trece” sau “nu trece”. Din acest motiv se impune definirea unei noi mărimi atașate caracteristicii de frecvență și anume **banda de trecere**. Aceasta reprezintă intervalul de frecvențe în care amplitudinea la ieșire este mai mare decât $\frac{1}{\sqrt{2}}|H_{max}|$.

Exemplu 4. Pentru un FTJ având caracteristica de amplitudine ca cea din *Figura 4*, se observă că pentru frecvența de 941Hz, valoarea caracteristicii de amplitudine este de 0.86 ($0.86 > \frac{1}{\sqrt{2}}$). În aceste condiții, dacă se aduce la intrarea filtrului o sinusoidă cu frecvența de 941Hz, atunci aceasta va trece.

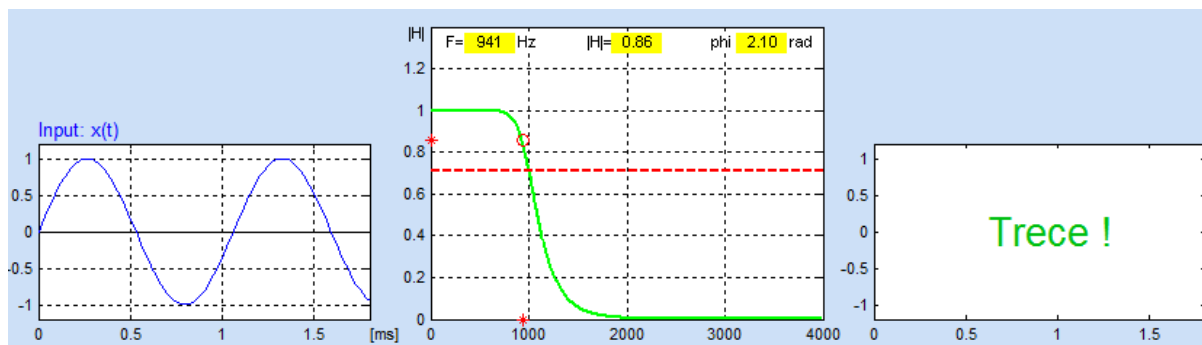


Figura 4. Banda de trecere. Implementare realizată cu aplicația *DSP_Assistant*

Cum se comportă însă filtrul pentru semnale altfel decât cele sinusoidale? Răspunsul poate fi dat dacă se cunosc două principii importante:

1. **Orice semnal este format din sinusoidă.** Mulțimea sinusoidelor din care este format un semnal se numește **spectru**. Spectrul se determină cu ajutorul *Teoremei Fourier* (pentru semnale periodice) și cu *Transformata Fourier* (pentru semnale neperiodice).
2. **Principiul superpoziției.** Fiecare cauză își produce propriul efect, fără a ține cont de prezența celorlalte. Astfel, fiecare sinusoidă a semnalului de intrare se propagă prin filtru, ca și cum ar fi singură. Deci fiecare sinusoidă din semnalul de intrare va trece prin filtru conform caracteristicii de frecvență a filtrului.

Exemplu 5. În exemplul din *Figura 5* se aduce la intrarea filtrului un semnal obținut din suma a două sinusoidă având frecvențele $F_1 = 500\text{Hz}$ și $F_2 = 1000\text{Hz}$. Caracteristica de amplitudine este cea a unui filtru *notch*, proiectat să rejeteze frecvența de 1000Hz. Se observă că la ieșirea filtrului a rămas doar sinusoidă cu frecvența de 500Hz.

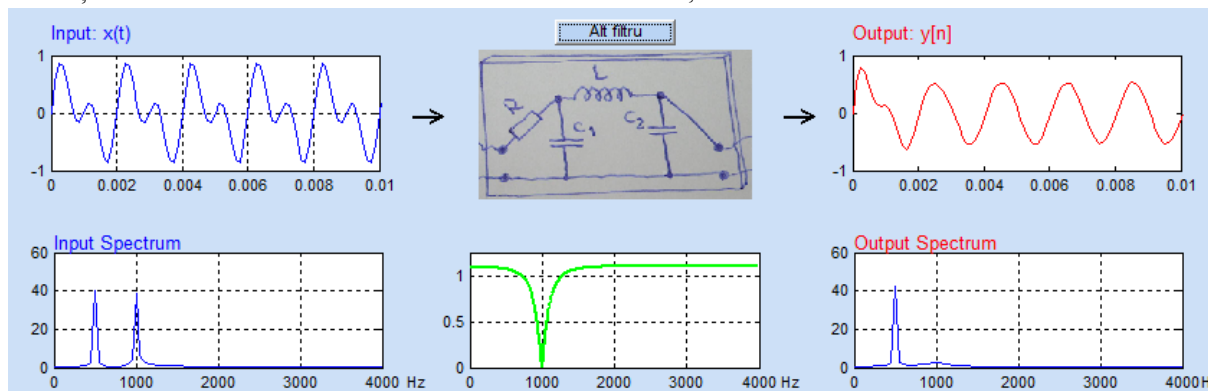


Figura 5. Filtrarea semnalelor nesinusoidale. Implementare realizată cu aplicația *DSP_Assistant*

Aplicații în Matlab

1. Să se deschidă aplicația *DSP_Assistant* (se selectează în *Current Directory* calea către aplicație, iar în *Command Window* se scrie `>> Main_Filters`). Se vor parcurge paginile 1-18.
2. Să se răspundă la întrebările din aplicație de la paginile 19, 20 și 21.

Lucrarea 2

Noțiuni introductive în Matlab

Obiective: însușirea unor elemente de bază din Matlab precum: definirea unei matrice, operații cu matrice, tipuri de date, variabile, instrucțiuni, fișiere script.

Matlabul este foarte util pentru realizarea simulărilor, având o colecție bogată de funcții ce permit implementarea cu ușurință a algoritmilor din diverse domenii precum: procesarea imaginilor, procesarea semnalelor audio, inteligență artificială etc

Interfața Matlabului (v. 7.7.0) arată ca în *Figura 1* și conține următoarele ferestre importante:
1) Current Directory; 2) Command Window; 3) Workspace; 4) Command History.

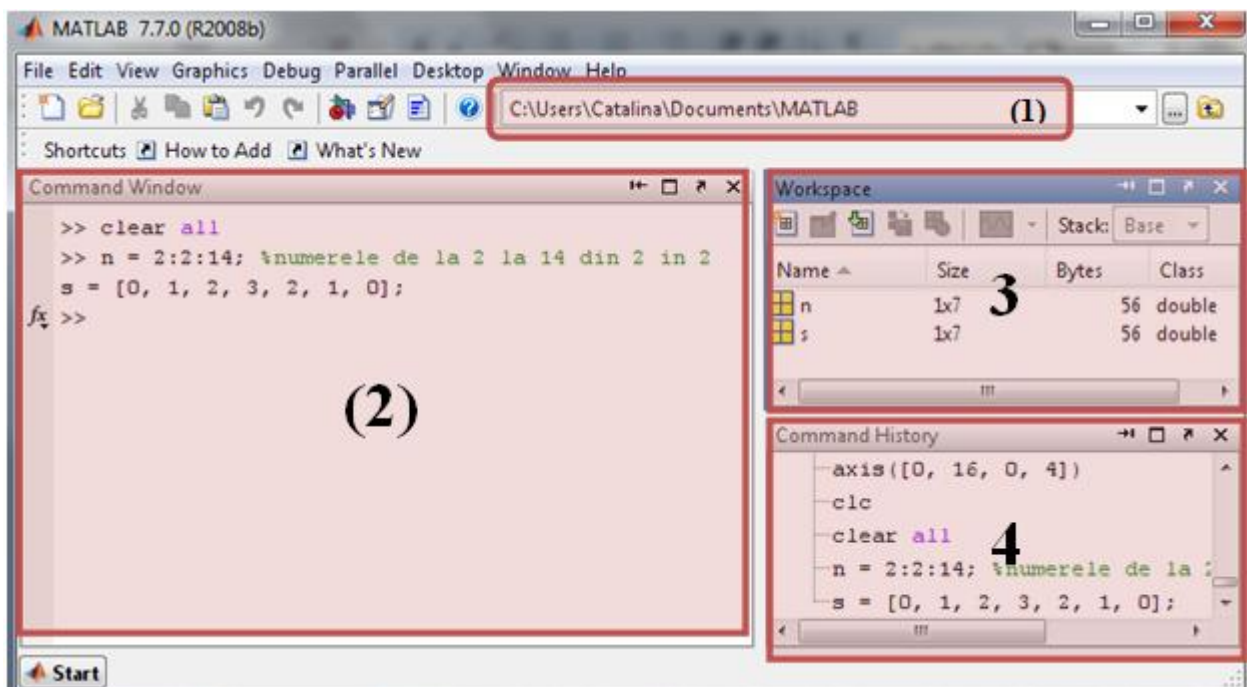


Figura 1. Interfață Matlab versiunea 7.7.0 (R2008b)

- 1) Fereastra *Current Directory* permite setarea directorului curent.
- 2) Fereastra *Command Window* permite executarea instrucțiunilor linie cu linie. Instrucțiunea se execută doar la apăsarea tastei *Enter*. Pentru a șterge toate liniile din *Command Window* se folosește comanda *clc*.

3) Fereastra *Workspace* permite vizualizarea tuturor variabilelor din aplicația curentă. Pentru a șterge toate variabilele din *Workspace* se folosește comanda *clear all*.

4) Fereastra *Command History* permite vizualizarea ultimelor instrucțiuni scrise în *Command Window*. Se poate accesa o instrucțiune din *Command History* prin dublu-click pe acea instrucțiune. Apăsând tasta ↑ (săgeată în sus) în fereastra *Command Window*, se vor afișa linii cu linii instrucțiunile din *Command History* în ordinea inversă în care au fost scrise.

Dacă doriți să porniți interfața în modul *Default*, accesați *Desktop/Desktop Layout/Default*.

Help și lookfor

Pentru a afla informații despre o funcție a Matlabului tastați în *Command Window* numele funcției precedat de *help*.

Exemplu: dacă doriți să aflați mai multe informații despre funcția *min*, tastați:

```
>> help min
```

și va apărea descrierea funcției:

```
MIN      Smallest component.  
For vectors, MIN(X) is the smallest element in X...
```

Dacă aveți nevoie de o funcție care bănuieți că ar trebui să fie deja implementată în Matlab, dar nu îi știți numele, o puteți căuta tastând în *Command Window* un cuvânt cheie pentru acea funcție, precedat de *lookfor*.

Exemplu: vreți să calculați media elementelor dintr-un vector dar nu știți ce funcție să folosiți. Un cuvânt cheie după care puteți căuta este *average*, așa că puteți scrie:

```
>> lookfor average
```

Matlabul va începe să afișeze toate funcțiile care pot avea legătură cu acest cuvânt cheie căutat.

```
localavfit          - Construct "average fit" model  
mean                - Average or mean value.
```

În acest caz observăm că a doua funcție găsită este *mean*, care din descriere constatăm că este funcția căutată. Pentru a opri Matlabul din rulare se folosește comanda CTRL + C.

Lucrul cu matrice în Matlab

Elementul de bază cu care lucrează Matlabul este **matricea**, acest lucru sugerându-l chiar numele de MATLAB care vine de la “**matrix laboratory**”. Matlabul este optimizat pentru calcul matriceal, operațiile cu matrice (așa cum se va vedea) fiind foarte ușor de realizat.

Observații:

- un vector linie este o matrice cu o singură linie.
- un vector coloană este o matrice cu o singură coloană.
- un scalar (un număr) este o matrice cu o linie și o coloană.

Pentru a introduce o matrice în Matlab trebuie să respectăm următorii pași:

- elementele de pe linii se separă prin *spațiu* sau *virgulă*.
- pentru a separa coloanele se folosește *punct și virgulă*.
- elementele matricei se scriu între *paranteze pătrate*.

Exemplu: Fie matricea $A = \begin{bmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{bmatrix}$.

Se scrie în *Command Window*:

```
>> A = [8 1 6; 3 5 7; 4 9 2]
```

Matlabul afișează în *Command Window*:

```
A =
     8     1     6
     3     5     7
     4     9     2
```

iar variabila A este salvată în *Workspace*.

- elementul de pe linia i și coloana j poate fi selectat prin comanda $A(i, j)$.

Exemplu: Pentru matricea A de mai sus, $A(2, 3)$ reprezintă selecția elementului de pe linia 2 și coloana 3, adică 7. Pentru a însuma elementele de pe ultima linie din matricea A putem scrie:

```
>> A(3,1) + A(3,2) + A(3,3)
ans =
    15
```

Atenție! În Matlab indexarea începe de la 1. Primul element al unui vector x este $x(1)$.

Operatorul două puncte (:)

Acest operator este extrem de important în Matlab și poate fi folosit în mai multe contexte.

Sintaxă: $x = \text{val_start} : \text{pas} : \text{val_stop}$

Se generează un vector x , cu valori între val_start și val_stop , din pas în pas . Dacă nu se scrie variabila pas , se consideră implicit $\text{pas} = 1$.

Exemplu 1: Se generează un vector x cu valori între 1 și 10.

```
>> x = 1 : 10
x =
     1     2     3     4     5     6     7     8     9    10
```

Exemplu 2: Se generează un vector x cu valori între 1 și 100, din 10 în 10.

```
>> x = 1 : 10 : 100
x =
     1    11    21    31    41    51    61    71    81    91
```

Sintaxă: $B = A(:, j)$

În acest caz operatorul “:” semnifică *toate liniile*. Cu alte cuvinte, în matricea B se salvează coloana j din matricea A.

Exemplu 1: Se salvează în matricea B, toate liniile și coloana 2 din matricea A

```
>> B = A(:, 2)
B =
     1
     5
     9
```

Exemplu 2: Se salvează în matricea C liniile de la 1 la 2 și coloana 2 din matricea A.

```
>> C = A(1 : 2, 2)
C =
     1
     5
```

Exemplu 3: Se însumează toate elementele de pe ultima linie din matricea A.

```
>> sum(A(3, :))
ans =
    15
```

Operații cu matrice

Cele mai uzuale operații cu matrice sunt:

	Operatori	Observații
+	Adunare	$A + B$ (A și B au aceleași dimensiuni)
-	Scădere	$A - B$ (A și B au aceleași dimensiuni)
*	Înmulțire	$A * B$ (numărul de coloane din A = numărul de linii din B)
.*	Înmulțire element cu element	$A.*B$ (A și B au aceleași dimensiuni)
./	Împărțire element cu element	$A./B$ (A și B au aceleași dimensiuni)
^	Ridicare la putere	Pentru matrice pătrate, $A^n = A * A * \dots * A$ (de n ori)
.^	Ridicare la putere element cu element	$A.^n$ (se ridică fiecare element din A la puterea n)
.'	Transpusa unei matrice	$A.'$; $A(m,n)$ devine $A(n,m)$
'	Conjugata transpusă a unei matrice	A' ; Pentru o matrice conținând numere reale, $A.' = A'$

Fie matricele:

```
>> A = [1 2 0; 0 -1 1];
>> B = [0 -1 2; 4 -2 5];
>> C = [1 0; -1 0; 2 1];
```

Exemplu 1:

```
>> D = A + B
D =
     1     1     2
     4    -3     6
```

Exemplu 2:

```
>> E = A - B
E =
     1     3    -2
    -4     1    -4
```

Exemplu 3:

```
>> F = A * C
F =
    -1     0
     3     1
```

Exemplu 4:

```
>> G = A .* B
G =
     0    -2     0
     0     2     5
```

Exemplu 5:

```
>> H = A ./ B
H =
    Inf    -2.0000     0
     0     0.5000    0.2000
```

Exemplu 6:

```
>> J = A.^3
J =
     1     8     0
     0    -1     1
```

Exemplu 7:

```
>> K = A.'
K =
     1     0
     2    -1
     0     1
```

Punct și virgulă (;)

În Matlab, spre deosebire de alte limbaje de programare, nu este obligatoriu ca o instrucțiune să se termine cu *punct și virgulă*. În Matlab, *punct și virgulă* la sfârșitul instrucțiunii are rolul de a suprima afișajul rezultatului în *Command Window*.

Exemplu:

```
>> a = [1 2 3 4 5]
a =
     1     2     3     4     5
>> b = [1 2 3 4 5];
>>
```

Se observă că spre deosebire de variabila *a*, variabila *b* nu a mai fost afișată în *Command Window*. Ambele variabile se găsesc însă în *Workspace*.

În Matlab există funcții care generează matrice extrem de utile precum:

- `zeros(m, n)` generează o matrice de dimensiune $m \times n$ conținând zerouri.
- `ones(m, n)` generează o matrice de dimensiune $m \times n$ cu valori unitare.
- `eye(m)` generează matricea unitate de dimensiune $m \times m$.
- `rand(m, n)` generează o matrice de dimensiune $m \times n$ cu valori random uniform distribuite în intervalul (0, 1).

Variabile

În Matlab, de cele mai multe ori nu trebuie ca o variabilă să fie declarată înainte de a fi folosită. Este suficient să i se atribuie direct o valoare, ca în exemplu:

```
>> variabila = sin(pi/2)
```

Observații:

- numele oricărei variabile trebuie să înceapă cu o literă.
- numele oricărei variabile nu trebuie să conțină caractere care nu sunt litere, cifre sau “_”.
- numele oricărei variabile nu trebuie să fie un cuvânt cheie al Matlabului. Pentru lista completă a cuvintelor cheie rulați comanda `>>iskeyword`.
- lungimea maximă a numelui unei variabile este dată de `>>namelengthmax`.
- deși nu este interzis, nu numiți o variabilă `sin`, `function`, `length`, `sum`, `max` sau orice altă denumire care reprezintă deja o funcție a Matlabului, deoarece acea funcție nu va mai putea fi folosită cât timp variabila cu același nume există în *Workspace*.

○ *Exemplu de așa nu:*

```
>> x = [ 1 7 2 3 9];
>> max = 10;
>> y = max(x)
```

??? *Index exceeds matrix dimensions.*

Funcția `max` a Matlabului (care determină maximumul unui vector) a fost înlocuită de variabila `max = 10`. Ca urmare, Matlabul va semnala eroare atunci când dorim să determinăm valoarea maximă din vectorul `x` (??? *Index exceeds matrix dimensions*).

- Matlabul este *case sensitive*, face distincție între litere mici și litere mari.
- este necesară declararea variabilelor înainte de a fi folosite doar în cazul variabilelor `global` sau `persistent`.
- în Matlab nu este obligatoriu să declarăm tipul sau dimensiunea variabilelor. Implicit o variabilă numerică va fi salvată pe **8 octeți**, iar tipul va fi **double**.

Tipuri de date în Matlab

În Matlab, există mai multe tipuri de date, dintre care cele mai uzuale sunt: *integer*, *double*, *char*, *cell* și *struct*.

1. Tipul de date **integer**, poate fi *cu semn* sau *fără semn*, pe 1, 2, 4 sau 8 octeți.

Tabel 1. Tipul de date integer

Tipul de date	Domeniul de valori	Funcția Matlab care realizează conversia
Signed 8-bit integer	-2^7 to 2^7-1	int8
Signed 16-bit integer	-2^{15} to $2^{15}-1$	int16
Signed 32-bit integer	-2^{31} to $2^{31}-1$	int32
Signed 64-bit integer	-2^{63} to $2^{63}-1$	int64
Unsigned 8-bit integer	0 to 2^8-1	uint8
Unsigned 16-bit integer	0 to $2^{16}-1$	uint16
Unsigned 32-bit integer	0 to $2^{32}-1$	uint32
Unsigned 64-bit integer	0 to $2^{64}-1$	uint64

Matlabul stochează implicit datele numerice în formatul *double*. Pentru a stoca în format *integer* trebuie făcută conversia din *double* în tipul *integer* dorit, ca în exemplu:

```
>> x = int16(325)
```

Dacă numărul ce se dorește a fi convertit este un număr zecimal, atunci acesta va fi rotunjit la cel mai apropiat întreg.

```
>> x = int8(32.4)
x =
    32
```

Dacă avem de stocat o imagine grayscale (cu valori întregi între 0 și 255) nu are sens să salvăm fiecare pixel al imaginii în format *double* (8 octeți), este suficient să salvăm în format *uint8*.

2. Tipul de date **double** (numere reale în dublă precizie cu virgulă mobilă). Este reprezentat pe 8 octeți (64 de biți) împărțiți astfel:

- 0 ÷ 51 partea fracționară
- 52 ÷ 62 exponent
- bitul 63 este pentru semn (0 pozitiv, 1 negativ)

Intervalul de valori pentru o variabilă *double* este:

- pentru valori negative între $-1.79769e+308$ și $-2.22507e-308$
- pentru valori pozitive între $2.22507e-308$ și $1.79769e+308$

Obs: $e+308 = 10^{308}$

Pentru a afla distanța dintre un număr și următorul număr în dublă precizie se folosește sintaxa `eps(număr)`.

Exemplu: pentru a afla distanța dintre 6 și următorul număr în dublă precizie se scrie:

```
>> format long
>> eps(6)
ans =
    8.881784197001252e-016
```


Rezultă că următorul număr în dublă precizie după 6 este $6 + \text{eps}(6)$.

```
>> 6+eps(6)
ans =
    6.0000000000000001
```

3. Tipul de date **char**. O variabilă *char* este definită pe 2 octeți. Mesajele (*string*) sunt interpretate de Matlab ca matrice (adesea vectori linie) de variabile *char*. Pentru a scrie un mesaj, acesta trebuie să fie încadrat de apostroafe.

```
>> mesaj = 'laborator PNS'
mesaj =
laborator PNS
```

Deoarece variabila *mesaj* conține 13 caractere, aceasta este stocată pe 26 de octeți.

Exemple de prelucrări de stringuri:

```
>> Mesaj1 = 'la';
Mesaj2 = 'tine';
%concatenare
Mesaj3 = [Mesaj1, Mesaj2];
%concatenare cu spatiu
Mesaj4 = [Mesaj1, ' ', Mesaj2];
%substrings
Sub0 = ' ';
Sub1 = Mesaj3(1:4);
Sub2 = 'tudi';
Sub3 = Mesaj3(5:6);
Mesaj5 = [Sub0 Sub1 Sub2 Sub3];
%substitutie
Mesaj6 = Mesaj5;
Mesaj6(1:4) = 'long';
%Matrice de char
Mesaj7 = [Mesaj5;Mesaj6]

Mesaj7 =
latitudine
longitudine
```

4. Tipul de date **cell**. O astfel de variabilă conține mai multe celule, fiecare celulă putând conține orice tip de dată. Pentru a defini o variabila *cell*, conținutul ei trebuie scris între acolade.

```
>> A = {'Punctaj =', 9.8}
A =
'Punctaj =' [9.8]
```

În exemplul de mai sus, variabila *A* conține o variabilă de tip șir de caractere (*char*) și o variabilă numerică (*double*).

Pentru a accesa un element al celulei, se folosește sintaxa *variabilă{index}*.

```
>> A{1}
ans =
Punctaj =
>> A{2}
ans =
    9.8
```

5. Tipul de date **struct**. `s = STRUCT('field1',VALUES1,'field2',VALUES2,...)`

Se realizează o structură cu câmpurile și valorile specificate.

```
>> s = struct('orase',{{'Sibiu','Brasov'}},'populatie',[154.220 276.914])
s =
    orase: {'Sibiu'  'Brasov'}
    populatie: [154.2200 276.9140]
```

Pentru a accesa câmpul *orase* se folosește sintaxa:

```
>> s.orase
ans =
    'Sibiu'    'Brasov'
```

Pentru a accesa valoarea '*Sibiu*' se accesează prima valoare a câmpului *orase* folosind sintaxa:

```
>> s.orase(1)
ans =
    'Sibiu'
```

Instrucțiunea FOR

Se utilizează pentru a realiza o structură repetitivă, condiționată anterior.

Sintaxă: `for var = val_start:pas:val_stop`
 [instrucțiuni]
end

Pentru *var* luând valori în intervalul $val_start \div val_stop$, din *pas* în *pas*, se execută blocul de instrucțiuni. Dacă nu se scrie variabila *pas*, se consideră *pas* = 1.

Exemplu: `>> for n = 1 : 8`
 `x(n) = 2^n;`
 end
`>> x`
x =
2 4 8 16 32 64 128 256

Instrucțiunea IF

Execută un set de instrucțiuni când o expresie este adevărată.

Sintaxă: `if(expresie)`
 [instrucțiuni1]
else
 [instrucțiuni2]
end

Dacă valoarea expresiei este edevărată, atunci se execută blocul de instrucțiuni1; altfel se execută blocul de instrucțiuni2.

Exemplu: `>> if(2^10 > 8^6)`
 `'2^10 > 8^6'`
 else
 `'2^10 < 8^6'`
 end
ans =
2^10 < 8^6

Fișiere script

Până acum toate exemplele date au fost scrise în *Command Window* deoarece au fost instrucțiuni simple și scurte și ne interesa rezultatul direct. Atunci când vrem să scriem însă un program, vom folosi fișiere script (cu extensia *.m*). Pentru a deschide un fișier nou, selectăm *File/New/Blank m-file*. Numele sub care salvăm un fișier trebuie să respecte următoarele reguli:

- să înceapă cu literă
- să NU conțină spații
- să NU fie numele unei funcții existente în Matlab

Pentru a rula un program, aveți mai multe opțiuni:

- F5
- săgeata verde având *tooltip* “*save and run*”
- *Debug/Run*

Este bine ca primele linii ale programului să fie:

```
clc
clear all
close all
```

deoarece atunci când se va rula programul:

- `clc` șterge tot ce era în *Command Window*. În acest fel vom vedea doar eventualele erori/rezultate ale rulării curente.
- `clear all` șterge toate variabilele din *Workspace*.
- `close all` închide toate ferestrele cu grafice.

Pentru a comenta o linie de cod se folosește semnul procent (%). Linia comentată va fi scrisă cu verde.

```
>> n=1:5; % n contine numerele de la 1 la 5
```

Pentru a comenta mai multe linii de cod, se selectează liniile și se folosește comanda CTRL + R. Pentru a decommenta se folosește combinația de taste CTRL + T.

Exemplu: să se scrie într-un fișier script toate calculele efectuate în exemplele de la operațiile cu matrice. Ce se va afișa în *Command Window*?

```
clc
clear all
close all
A = [1 2 0; 0 -1 1];
B = [0 -1 2; 4 -2 5];
C = [1 0; -1 0; 2 1];
D = A + B; % suma
E = A - B % diferenta
F = A * C; % inmultire
G = A.* B; % inmultire element cu element
H = A./B; % impartire element cu element
I = A.^3; % ridicare la putere element cu element
```

În *Command Window* se afișează doar conținutul matricei E, deoarece doar pentru acea instrucțiune nu este suprimat afișajul cu punct și virgulă.

Aplicații în Matlab

1. Să se scrie într-un fișier script următoarea matrice:

$$A = \begin{bmatrix} 1 & 0 & 5 & 2 \\ -1 & 3 & 4 & 0 \\ 2 & 3 & 1 & -3 \end{bmatrix}$$

și să se realizeze în Matlab următoarele operații:

- să se calculeze suma elementelor de pe linia 2 din matricea A.
- să se salveze în matricea B linia 1 din matricea A.
- să se salveze în matricea C coloana 3 din matricea A.
- să se salveze în matricea D coloanele 2 și 4 din matricea A.
- să se salveze în matricea E elementele din A de pe liniile 1, 3 și coloanele 1, 4.
- să se afișeze elementul de pe linia 2 coloana 4.

2. Să se genereze un vector x conținând elementele 3, 6, 9, 12, ..., 195, 198, 201 și să se calculeze în variabila y suma elementelor din x.

3. Să se genereze o matrice A de dimensiune 5 x 5 conținând valori random distribuite uniform în intervalul (0, 1). Să se genereze matricea unitate de dimensiune 5 x 5 și să se salveze în variabila B. Să se salveze în matricea C produsul dintre A și B. Să se salveze în matricea D produsul dintre A și B realizat element cu element.

4. Pentru a ocupa cât mai puțin spațiu de memorie, ce tip de date ar trebui folosit pentru a salva următoarele numere?

- a) 5; b) 120; c) -230 d) 21032 e) valoarea unui pixel dintr-o imagine grayscale

5) Câți octeți ocupă mesajul "Acesta este primul laborator de PNS" ?

6) Ce funcție a Matlabului se folosește pentru a calcula numărul elementelor unui vector?

7) Fie fișierul script *length.m* având următoarele linii de cod. Ceva este greșit. Ce anume?

```
clc
clear all
x = [1 5 7 3 4];
lungime_x = length(x)
```

8. Să se genereze o matrice A, astfel: $A[m, n] = m \cdot n$, știind că $m, n \in \overline{1 \dots 10}$.

9. Să se genereze matricea A cu numere complexe: $A[m, n] = m + j \cdot n$, știind că $m, n \in \overline{1 \dots 10}$. Să se calculeze matricea $B = A^T$.

10. Să se genereze semnalul $x[n] = e^{1/n}$, știind că $n \in \overline{1 \dots 10}$.

11. Să se genereze șirul lui Fibonacci $f(n) = f(n-1) + f(n-2)$, $f(1) = f(2) = 1$, pentru $n \in \overline{1 \dots 100}$. Să se calculeze șirul $g(n) = \frac{f(n+1)}{f(n)}$, pentru $n \in \overline{1 \dots 99}$.

Lucrarea 3

Grafice și funcții în Matlab

Obiective: reprezentarea graficelor și implementarea funcțiilor în Matlab.

Reprezentarea graficelor în Matlab

Pentru reprezentarea graficelor bidimensionale vom analiza funcțiile `stem` și `plot`.

Funcția `plot`

Funcția `plot` reprezintă grafic o secvență de date.

Sintaxă: `plot(x, y)`, reprezintă grafic elementele vectorului `y` în funcție de elementele vectorului `x`. Folosind proprietățile implicite ale funcției `plot`, rezultă un grafic conținând perechile $\{x(i), y(i)\}$ unite prin segmente, astfel încât să dea impresia de continuitate.

Atenție! Vectorii `x` și `y` trebuie să aibă aceleași dimensiuni.

Exemplu 1. Reprezentare grafică utilizând funcția `plot` cu parametri implicați

```
n = 2:2:14; %numerele de la 2 la 14 din 2 in 2
s = [0, 1, 2, 3, 2, 1, 0];
figure(1)
plot(n,s), title('Reprezentare grafica folosind plot')
xlabel('n'), ylabel('s'), axis([0, 16, 0, 4])
```

În `figure(1)` se va afișa următorul grafic.



Pentru a trasa un grafic cu funcția `plot` se pot folosi diverse combinații de linii, markere (puncte pe grafic) și culori, conform tabelului de mai jos. Pentru a seta grosimea liniei se folosește proprietatea `'LineWidth'`.

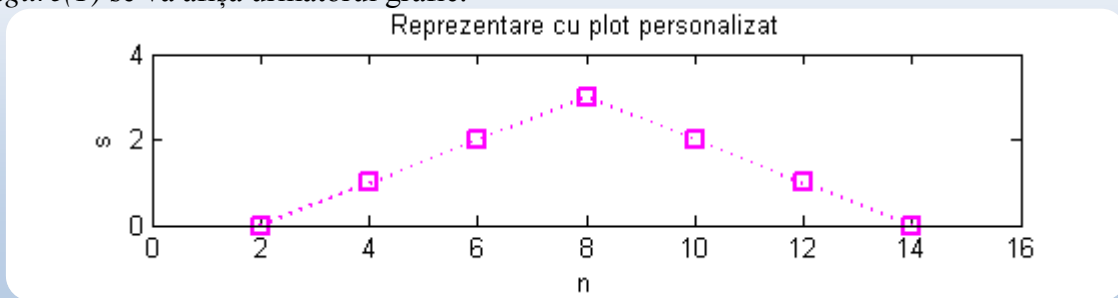
Tabel 1. Proprietăți ale funcției plot

Culori		Linii		Markere	
Simbol	Semnificație	Simbol	Semnificație	Simbol	Semnificație
r	roșu (red)	-	linie continuă	.	punct
g	verde (green)	:	linie punctată	o	cerc
b	albastru (blue)	-.	linie întreruptă	x	cruciuliță
c	cyan, (cyan)	--	linie întreruptă	+	plus
m	magenta, (magenta)	(none)	fără linie	*	steluță
y	galben, (yellow)			s	pătrat (square)
k	negru, (black)			d	romb (diamond)
w	alb, (white)				etc

Exemplu 2. Reprezentare grafică utilizând funcția plot cu diverși parametri (linie punctată, marker pătrat, culoare magenta, grosime linie 2)

```
n = 2:2:14; %numerele de la 2 la 14 din 2 in 2
s = [0, 1, 2, 3, 2, 1, 0];
figure(1)
% linie punctata, marker patrat, culoare magenta, grosime linie 2
plot(n,s, 'sm', 'linewidth',2), title('plot personalizat')
xlabel('n'), ylabel('s'), axis([0, 16, 0, 4])
```

În *figure(1)* se va afișa următorul grafic.



Pentru o altă culoare în afara celor menționate mai sus se folosește proprietatea ‘color’ urmată de codul *rgb* al culorii.

Exemplu: pentru a afișa un grafic folosind o nuanță de gri se folosește sintaxa:

```
plot(x, y, 'color', [0.5 0.5 0.5])
```

Exemplu: pentru a afișa un grafic folosind culoarea portocaliu se folosește sintaxa:

```
plot(x, y, 'color', [1.0 0.5 0.0])
```

Observație: dacă funcția plot se folosește cu un singur parametru, plot(y), în acest caz se consideră că pe axa Ox sunt numerele 1, 2, 3, ..., n, unde n reprezintă numărul de eșantioane din y (cu alte cuvinte, pe axa Ox este reprezentat indexul eșantionului).

Pentru mai multe informații despre funcția plot și proprietățile acesteia, rulați în *Command Window* >>help plot.

Funcția stem

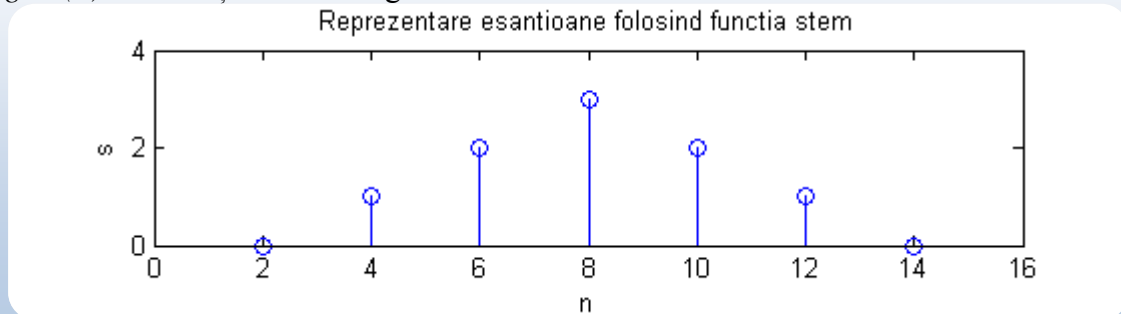
Funcția **stem** reprezintă grafic o **secvență** de date.

Sintaxă: stem(x,y) reprezintă grafic elementele vectorului y în funcție de elementele vectorului x. Spre deosebire de funcția plot, perechile {x(i), y(i)} sunt marcate prin cerculețe în vârful unor tije. *Atenție!* Vectorii x și y trebuie să aibă aceleași dimensiuni.

Exemplu 3. Reprezentare grafică utilizând funcția stem cu parametri implicați

```
n = 2:2:14; %numerele de la 2 la 14 din 2 in 2
s = [0, 1, 2, 3, 2, 1, 0];
figure(1)
stem(n,s), title('Reprezentare esantioane folosind functia stem')
xlabel('n'), ylabel('s'), axis([0, 16, 0, 4])
```

In *figure(1)* se va afișa următorul grafic.



Proprietățile funcției stem pot fi modificate în mod similar celor pentru funcția plot. Pentru mai multe informații despre funcția stem, rulați în *Command Window* >>help stem.

Reprezentarea graficelor în aceeași figură, în același sistem de coordonate

Pentru a reprezenta mai multe grafice în același sistem de coordonate, se folosește sintaxa:

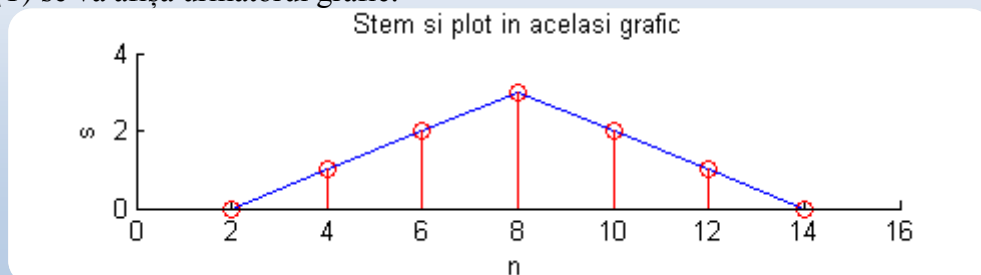
```
hold on
[reprezentare grafic_1]
[reprezentare grafic_2]
....
[reprezentare grafic_n]
hold off
```

De exemplu, pentru a afișa în aceeași figură, în același sistem de coordonate, graficele cu stem și plot din *Exemplele 1 și 2*, codul propus este cel din *Exemplu 4*.

Exemplu 4. Reprezentarea mai multor grafice utilizând hold on ... hold off

```
n = 2:2:14; %numerele de la 2 la 14 din 2 in 2
s = [0, 1, 2, 3, 2, 1, 0];
figure(1)
hold on
plot(n,s)
stem(n,s, 'r')
hold off
title('Stem si plot in acelasi grafic'), xlabel('n'), ylabel('s')
axis([0, 16, 0, 4])
```

In *figure(1)* se va afișa următorul grafic.



Reprezentarea graficelor în aceeași figură, în sisteme de coordonate diferite

Pentru a reprezenta mai multe grafice în aceeași figură, dar în sisteme de coordonate diferite se folosește funcția `subplot`.

Sintaxa: `subplot(m,n,p)`, împarte figura într-o matrice cu m linii, n coloane. Parametrul p reprezintă indicele de ordine al graficelor, numerotarea făcându-se de la stânga la dreapta și de sus în jos.

Exemplu: pentru a reprezenta într-o figură, 6 grafice organizate pe 2 linii și 3 coloane, structura este următoarea:

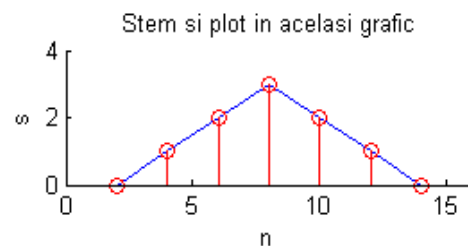
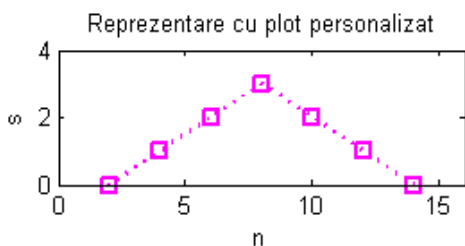
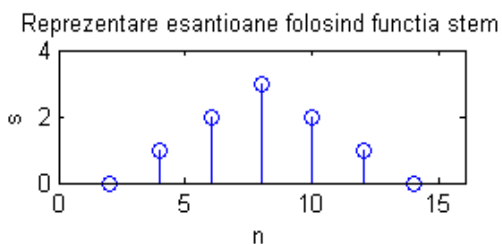
<code>subplot(2,3,1)</code>	<code>subplot(2,3,2)</code>	<code>subplot(2,3,3)</code>
<code>subplot(2,3,4)</code>	<code>subplot(2,3,5)</code>	<code>subplot(2,3,6)</code>

Observație: `subplot` indică doar zona în care se face afișarea, nu face și afișarea. Pentru afișare se folosește una dintre funcțiile `plot`, `stem` etc.

De exemplu, pentru a afișa în aceeași figură, în sisteme de coordonate diferite, graficele din exemplele 1, 2, 3 și 4, codul propus este cel din *Exemplu 5*.

Exemplu 5. Reprezentarea mai multor grafice utilizând `subplot`

```
n = 2:2:14; %numerele de la 2 la 14 din 2 in 2
s = [0, 1, 2, 3, 2, 1, 0];
subplot(2,2,1)
    stem(n,s), title('Reprezentare esantioane folosind functia stem')
    xlabel('n'), ylabel('s'), axis([0, 16, 0, 4])
subplot(2,2,2)
    plot(n,s), title('Reprezentare grafica folosind functia plot')
    xlabel('n'), ylabel('s'), axis([0, 16, 0, 4])
subplot(2,2,3)
    plot(n,s,':sm', 'linewidth',2), title('Reprezentare cu plot personalizat')
    xlabel('n'), ylabel('s'), axis([0, 16, 0, 4])
subplot(2,2,4)
    hold on
    plot(n,s)
    stem(n,s,'r')
    hold off
    title('Stem si plot in acelasi grafic'), xlabel('n'), ylabel('s')
```



Funcții

Fișierele *M*-file pot fi fișiere *script* sau *funcții*. Fișierele *script* sunt fișiere simple conținând o secvență de instrucțiuni. *Funcțiile* acceptă parametri de intrare și returnează parametri de ieșire. Variabilele din interiorul unei funcții sunt variabile locale. Sintaxa unei funcții este:

```
function [out1, out2, ...] = functionName(in1, in2, ...)
```

unde:

- `out1, out2, ...` reprezintă parametrii de ieșire și sunt salvați într-un vector.
- `in1, in2, ...` reprezintă parametrii de intrare.
- `functionName` este numele funcției.

Observații:

- în mod obligatoriu prima linie din codul unei funcții este cea în care se declară funcția.
- numele funcției trebuie să coincidă cu numele fișierului *script* în care este salvată funcția (exceptând extensia *.m*).
- este recomandat ca imediat după linia de declarare a funcției să existe comentarii sugestive referitoare la modul de utilizare al funcției, parametrii de intrare și de ieșire, etc (aceste comentarii vor fi afișate la comanda `>> help functionName`).
- terminarea unei funcții cu `end` este opțională.
- nu este obligatoriu ca o funcție să aibă parametri de ieșire.
- nu este obligatoriu ca o funcție să aibă parametri de intrare.

Exemplu 6. Să se implementeze în Matlab o funcție numită *sinusoida*, care să primească următorii parametri de intrare:

- Amplitudinea maximă (*A*)
- Frecvența (*F*)
- Frecvența de eșantionare (*F_s*)
- Faza inițială (*faza*)
- Durata semnalului în secunde (*durata*)

Funcția va calcula și va reprezenta grafic sinusoida $x[n]$ generată cu parametrii primiți la intrare.

Observație: formula unui semnal sinusoidal continuu este:

$$x(t) = A \cdot \sin(2\pi \cdot F \cdot t + \varphi_0) \text{ unde:}$$

- A = amplitudinea maximă
- F = frecvența
- φ_0 = faza inițială

În urma conversiei analog numerice (ce presupune eșantionare cu frecvența de eșantionare F_s și cuantizare) rezultă un semnal discret/numeric având formula:

$$x[n \cdot T_s] = A \cdot \sin(2\pi \cdot F \cdot nT_s + \varphi_0)$$

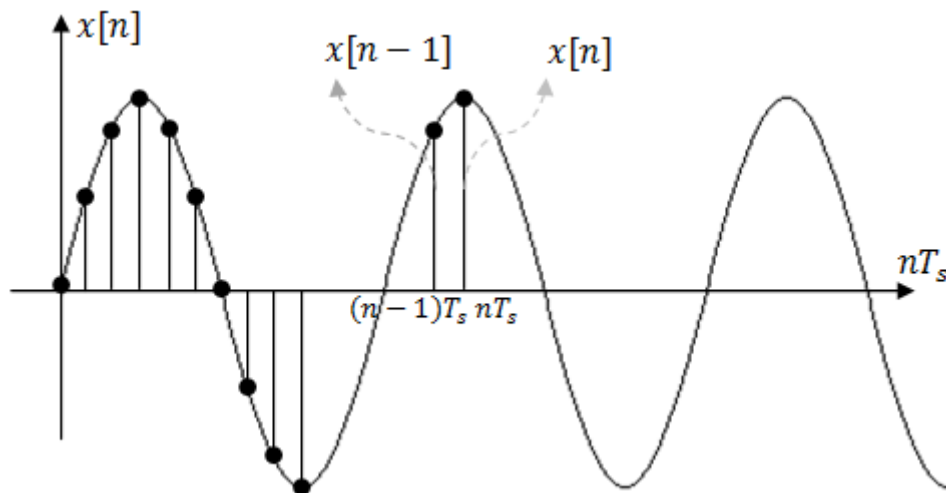


Figura 1. Semnal sinusoidal eșantionat

Formula sinusoidelor discrete se mai poate scrie în funcție de frecvența de eșantionare F_s :

$$x[n] = A \cdot \sin\left(2\pi \cdot n \cdot \frac{F}{F_s} + \varphi_0\right)$$

Observații:

- T (perioada) reprezintă intervalul de timp după care semnalul se repetă.
- F (frecvența de repetiție) reprezintă numărul de oscilații dintr-o secundă $F = 1/T$.
- T_s (perioada de eșantionare) reprezintă distanța în timp dintre două eșantioane.
- F_s (frecvența de eșantionare) reprezintă numărul de eșantioane dintr-o secundă.

Vectorul $x[n]$ este cel ce trebuie calculat și reprezentat grafic de funcția sinusoida. Codul Matlab propus este următorul.

```
function sinusoida(A, F, Fs, faza, durata)
% generare sinusoida avand parametrii de intrare: A, F, Fs, faza, durata
% A = Amplitudinea maxima
% F = Frecventa
% Fs = Frecventa de esantionare
% faza = Faza initiala
% durata = Durata semnalului (in secunde)

t = 0:1/Fs:durata;
x = A*sin(2*pi*F*t+faza);
figure(1)
hold on
plot(t,x,'r')
stem(t, x)
hold off
```

Pentru a apela această funcție din *Command Window*, trebuie ca mai întâi să schimbăm *Current Directory* să fie folderul în care am salvat fișierul *sinusoida*. Apoi vom scrie în *Command Window*:

```
>> help sinusoida
```

și vor apărea comentariile de la începutul funcției.

Pentru a apela funcția, se scrie numele funcției, iar între paranteze se dau valori parametrilor de intrare.

Exemplu: dacă dorim să reprezentăm grafic o sinusoidă cu frecvența de 2Hz, amplitudinea de 2V, fază inițială nulă, frecvența de eșantionare de 50Hz și durata de 2secunde, apelăm funcția sinusoida cu următorii parametri:

```
>> sinusoida(2, 2, 50, 0, 2)
```

Rezultatul este cel din *Figura 2*.

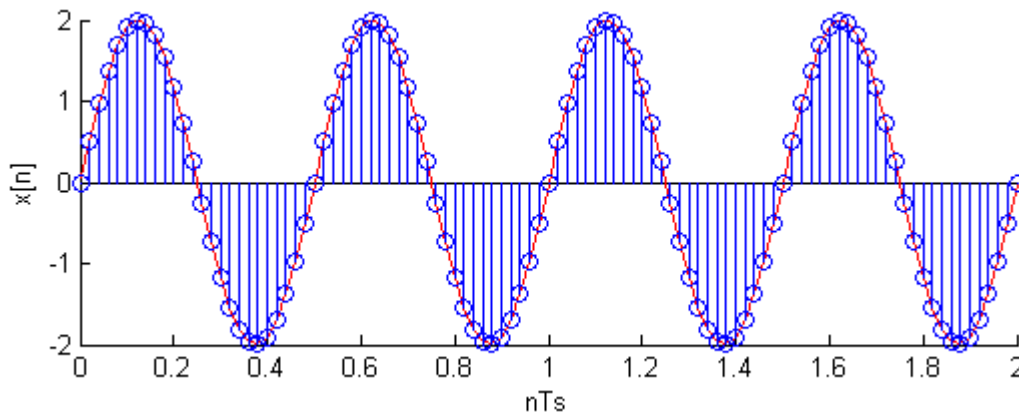


Figura 2. Sinusoida rezultată în urma apelării funcției `sinusoida(2, 2, 50, 0, 2)`

Aplicații în Matlab

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

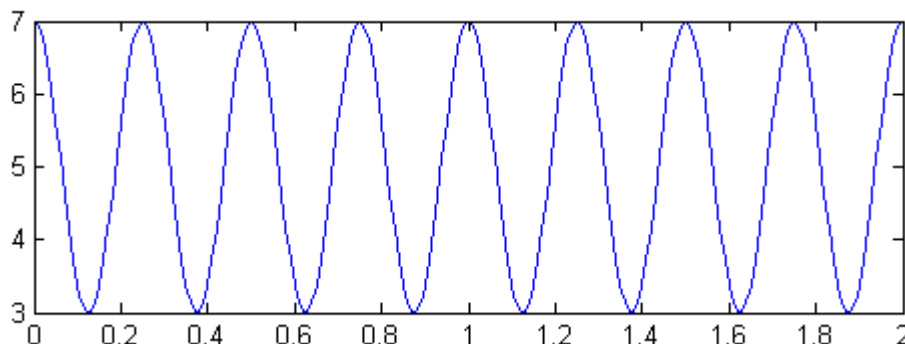
a) să se reprezinte grafic y în funcție de x folosind funcția `stem`.

b) să se reprezinte grafic y în funcție de x folosind funcția `plot`.

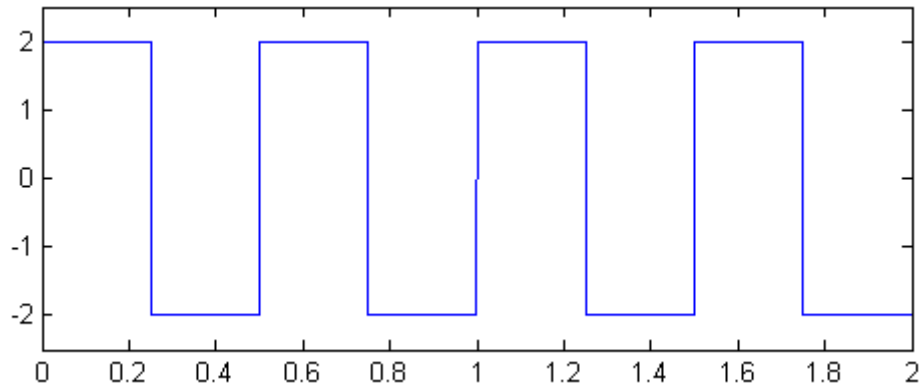
c) să se reprezinte grafic în aceeași figură, în același sistem de coordonate, y în funcție de x folosind funcțiile `plot` și `stem` (se vor folosi culori diferite).

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`.

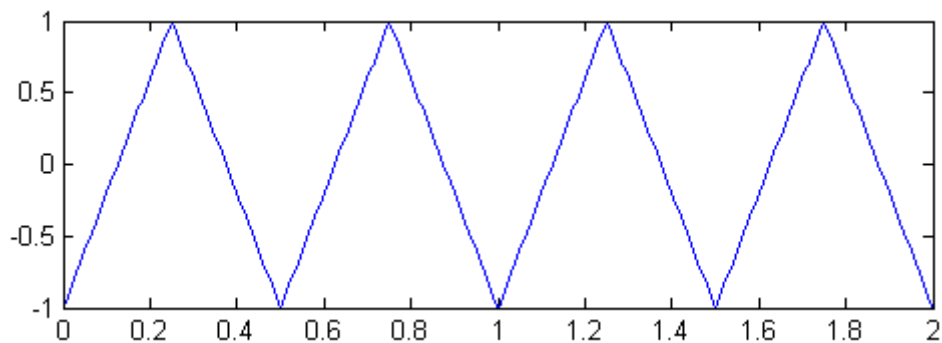
2. Să se genereze în Matlab următorul semnal sinusoidal.



3. Să se genereze în Matlab următorul semnal dreptunghiular.



4. Să se genereze în Matlab următorul semnal triunghiular.



5. Să se implementeze o funcție numită *medie*, care să calculeze media elementelor unui vector. Funcția va primi la intrare vectorul x ce se dorește a fi mediat și va întoarce ca parametru de ieșire valoarea y ce reprezintă media. Să se compare valoarea lui y cu rezultatul obținut de funcția `mean` a Matlabului.

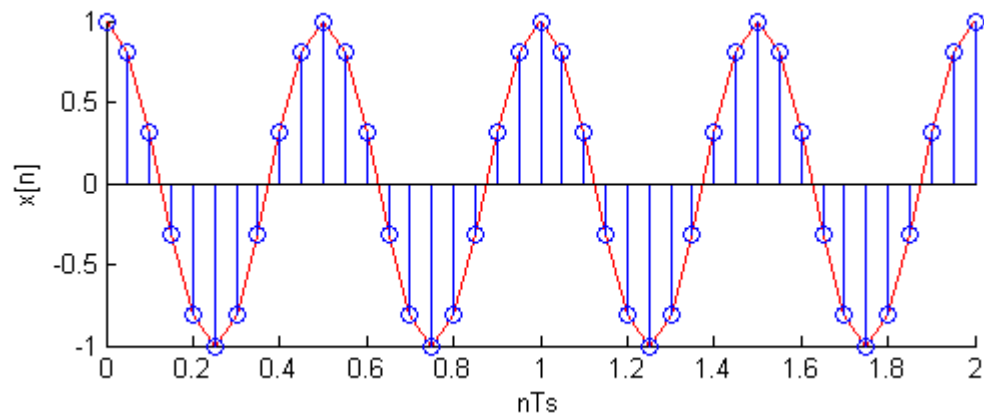
6. Să se realizeze o funcție care să oglindească orizontal o matrice pătrată (flip orizontal).

$$\text{Indiciu: } B = A \cdot \begin{bmatrix} 0 & \dots & 1 \\ \vdots & 1 & \vdots \\ 1 & \dots & 0 \end{bmatrix}$$

7. Să se realizeze o funcție care să oglindească vertical o matrice pătrată (flip vertical).

$$\text{Indiciu: } B = \begin{bmatrix} 0 & \dots & 1 \\ \vdots & 1 & \vdots \\ 1 & \dots & 0 \end{bmatrix} \cdot A$$

8. Cu ce parametri trebuie să se apeleze funcția sinusoida (din *Exemplu 6*), astfel încât rezultatul apelării să fie următorul?



9. Fie un semnal $x[n]$ format din două sinusoida având frecvențele f_{sol} (frecvența notei muzicale SOL) și f_{mi} (frecvența notei muzicale MI). Semnalul este generat astfel:

```
f_sol = 440/(1.0595^2); %Hz
f_mi = 440/(1.0595^5); %Hz
Fs = 8000; %Hz
d1 = 1; %s
d2 = 0.5; %s
dp = 0.2; %s
```

```
sin_sol_1 = sin(2*pi*f_sol*[0:1/Fs:d1]);
sin_sol_2 = sin(2*pi*f_sol*[0:1/Fs:d2]);
sin_mi_1 = sin(2*pi*f_mi*[0:1/Fs:d1]);
sin_mi_2 = sin(2*pi*f_mi*[0:1/Fs:d2]);
pauza = zeros(size([0:1/Fs:dp]));
```

```
x = [sin_sol_1, pauza, sin_mi_1, pauza, sin_sol_2, pauza, sin_sol_2, pauza,
sin_mi_1, pauza];
x = [x, sin_sol_2, pauza, sin_sol_2, pauza, sin_mi_2, pauza, sin_mi_2, pauza,
sin_sol_2, pauza, sin_sol_2, pauza, sin_mi_1, pauza];
```

```
sound(x)
```

Recunoașteți melodia?

Lucrarea 4

Interfață grafică în Matlab

Obiective: prezentarea unor noțiuni de bază pentru realizarea unei interfețe grafice în Matlab (GUI); implementarea unui GUI care să permită generarea și afișarea unei sinusoide dinamice.

GUI (Graphical User Interface)

Pentru a deschide o interfață grafică în Matlab se tastează comanda *guide* în fereastra *Command Window* (sau din *Toolbar* se selectează *File/New/GUI*). Pentru a porni o nouă interfață se selectează *Create New GUI/Blank GUI (Default)*. Pentru a deschide o interfață deja existentă se selectează *Open Existing GUI* și apoi se deschide interfața dorită. O interfață arată precum cea din *Figura 1*.

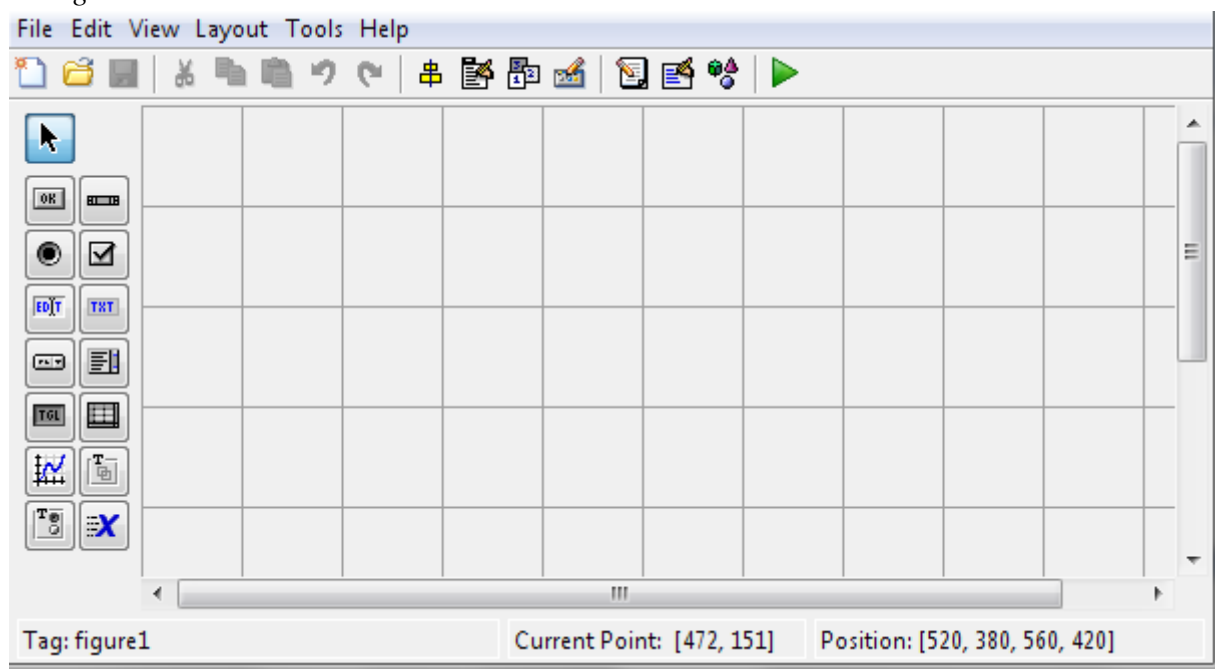


Figura 1. GUI (Graphical User Interface)

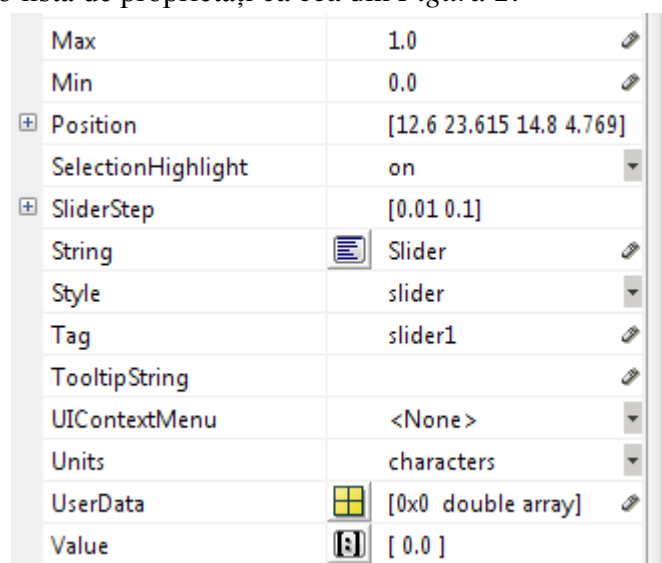
O interfață conține o zonă de lucru (centrală) și o zonă cu obiecte grafice (în stânga). Obiectele disponibile sunt: *Select*, *PushButton*, *Slider*, *Radio Button*, *Check Box*, *Edit Text* etc. La salvarea unei interfețe se vor crea două fișiere cu același nume dar cu extensii diferite: un fișier **.fig* (în care se află obiectele grafice și zona de lucru) și un fișier **.m* (ce conține codul Matlab).

Observații:

- cele două fișiere trebuie să fie în același folder.
- ambele fișiere trebuie să aibă același nume (diferă doar extensiile). Dacă se dorește redenumirea fișierelor, indicat ar fi să se salveze mai întâi fișierul **.fig* cu noul nume dorit, fișierul **.m* generându-se apoi automat cu noua denumire.
- de fiecare dată când se fac modificări în fișierul **.fig* acesta trebuie salvat, pentru a se actualiza și codul (fișierul **.m*).
- fișierul M-file conține următoarele funcții:
 - o funcție principală având aceeași denumire cu cea a fișierului M-file.
 - funcția `OpeningFcn` precedată de numele fișierului M-file.
Această funcție se apelează imediat ce s-a rulat programul, înainte de a accesa orice obiect. Aici se inițializează variabilele.
 - funcția `OutputFcn` precedată de numele fișierului M-file.
 - funcții asociate obiectelor introduse în interfață.
- pentru a accesa din cod un obiect grafic, trebuie ca *Tag*-ul acestuia să fie precedat de cuvântul cheie `handles`.
- toate proprietățile asociate unui obiect pot fi accesate atât din interfață (fișierul **.fig*) cât și din codul Matlab (fișierul **.m*) și pot fi modificate dinamic din codul Matlab.
- pentru a citi valoarea proprietății unui obiect se folosește funcția `get`.
`variabila = get(handles.tag_obiect, 'proprietate')`
- pentru a seta valoarea proprietății unui obiect se folosește funcția `set`.
`set(handles.tag_obiect, 'proprietate', valoare)`

Obiectul grafic *Slider*

Slider-ul permite parcurgerea cu pas constant a unui interval de valori *Min÷Max*. Pentru a introduce un *Slider* în interfață, se dă click pe butonul având *tooltip*-ul *Slider* și apoi se trasează un dreptunghi în zona de lucru, având dimensiunea *Slider*-ului dorit. Dacă se dă dublu click pe *Slider* se va deschide o listă de proprietăți ca cea din *Figura 2*.



Max	1.0	
Min	0.0	
Position	[12.6 23.615 14.8 4.769]	
SelectionHighlight	on	
SliderStep	[0.01 0.1]	
String	Slider	
Style	slider	
Tag	slider1	
TooltipString		
UIContextMenu	<None>	
Units	characters	
UserData	[0x0 double array]	
Value	[0.0]	

Figura 2. Proprietățile asociate unui *Slider*

Dintre toate proprietățile asociate unui *Slider*, următoarele ne interesează în mod special:

- *Tag*: reprezintă numele asociat *Slider*-ului (este de preferat ca *Tag*-ul să fie sugestiv. *Exemplu*: dacă se realizează un *Slider* din care se modifică frecvența unui semnal, atunci ar putea fi numit *sliderF*).
- *Min*: valoarea minimă de la care pornește *Slider*-ul.
- *Max*: valoarea maximă până la care merge *Slider*-ul.
- *SliderStep*: pasul cu care se modifică *Slider*-ul. Pentru a modifica *Slider*-ul să meargă din 1 în 1, parametrul *SliderStep* pentru x se calculează ca $1/(Max - Min)$.
Exemplu: dacă $Min = 0$ și $Max = 20$, atunci *SliderStep* trebuie să fie $1/20 = 0.05$.
- *Value*: reprezintă valoarea indicată de *Slider*. *Value* trebuie să fie tot timpul în intervalul $[Min, Max]$. *Atenție*: dacă selectați $Min = 5$ și $Max = 20$, atunci parametrul *Value* trebuie să fie setat în intervalul $[5, 20]$. Implicit *Value* = 0, iar în acest caz Matlabul va semnala eroare.

Toate proprietățile pot fi accesate și modificate dinamic din cod (fișierul M-file).

Dacă *Tag*-ul *Slider*-ului este *sliderF*, în momentul în care se salvează fișierul *.fig se vor crea în cod 2 funcții: *sliderF_CreateFcn* și *sliderF_Callback*. La fiecare mișcare a *Slider*-ului se apelează funcția *sliderF_Callback*. În această funcție putem scrie codul prin care se citește valoarea selectată cu *Sliderul sliderF*.

Pentru a salva într-o variabilă *F* valoarea selectată cu *sliderF*, se folosește sintaxa:

```
F = get(handles.sliderF, 'Value');
```

Pentru a seta valoarea lui *sliderF* cu o anumită valoare (de exemplu 5), se folosește sintaxa:

```
set(handles.sliderF, 'value', 5);
```

Obiectul grafic *Static Text*

Obiectul grafic *Static Text* este utilizat pentru afișarea unei valori. Dintre toate proprietățile asociate unui *Static Text*, următoarele ne interesează în mod special:

- *Tag*: reprezintă numele asociat obiectului grafic *Static Text* (este de preferat ca *Tag*-ul să fie sugestiv. *Exemplu*: dacă într-un *Static Text* se afișează valoarea unei frecvențe, atunci *Tag*-ul poate fi *valF*).
- *String*: valoarea scrisă în acest câmp este cea afișată.

Pentru a scrie din cod o valoare *F* într-un câmp *Static Text* având *Tag*-ul *valF*, se folosește sintaxa:

```
set(handles.valF, 'String', F)
```

Obiectul grafic *Axes*

Acest obiect permite afișarea graficelor și imaginilor într-un GUI. Modul de utilizare este foarte simplu. Se selectează obiectul *Axes* și apoi se trasează în zona de lucru o suprafață în care se dorește reprezentarea grafică. Pentru *Axes* este importantă proprietatea *Tag*, care reprezintă numele asociat obiectului.

Exemplu 1. Să se realizeze în Matlab o interfață grafică (GUI) în care să se afișeze o sinusoidă cu următorii parametri:

- Amplitudine $A = 2V$
- Frecvență de eșantionare $F_s = 100Hz$
- Durata semnalului $durata = 2s$
- Faza inițială $faza = 0 rad$
- Frecvența sinusoidei se selectează cu un *Slider* și poate avea valori în intervalul $0 \div 10Hz$, cu pas constant de $1Hz$.

Se va implementa un GUI conținând următoarele obiecte:

- *Slider* cu proprietățile: $Tag = sliderF$, $Min = 0$, $Max = 10$, $SliderStep = 0.1$
- *Static Text* în care se afișează valoarea frecvenței selectată cu *Slider*-ul, având următoarele proprietăți: $Tag = valF$ și $String = 0$ (deoarece *Slider*-ul pornește de la 0). Se vor mai adăuga două componente de tip *Static Text* în care se va scrie “ $F=$ ” și “ Hz ”. Tag -urile acestor două obiecte nu sunt importante, deoarece nu vor fi accesate din fișierul M-file.
- O zonă în care se afișează graficul, având $Tag = axes1$.

Interfața se va salva cu denumirea *sinusoida.fig*. Automat se va crea și fișierul M-file *sinusoida.m*.

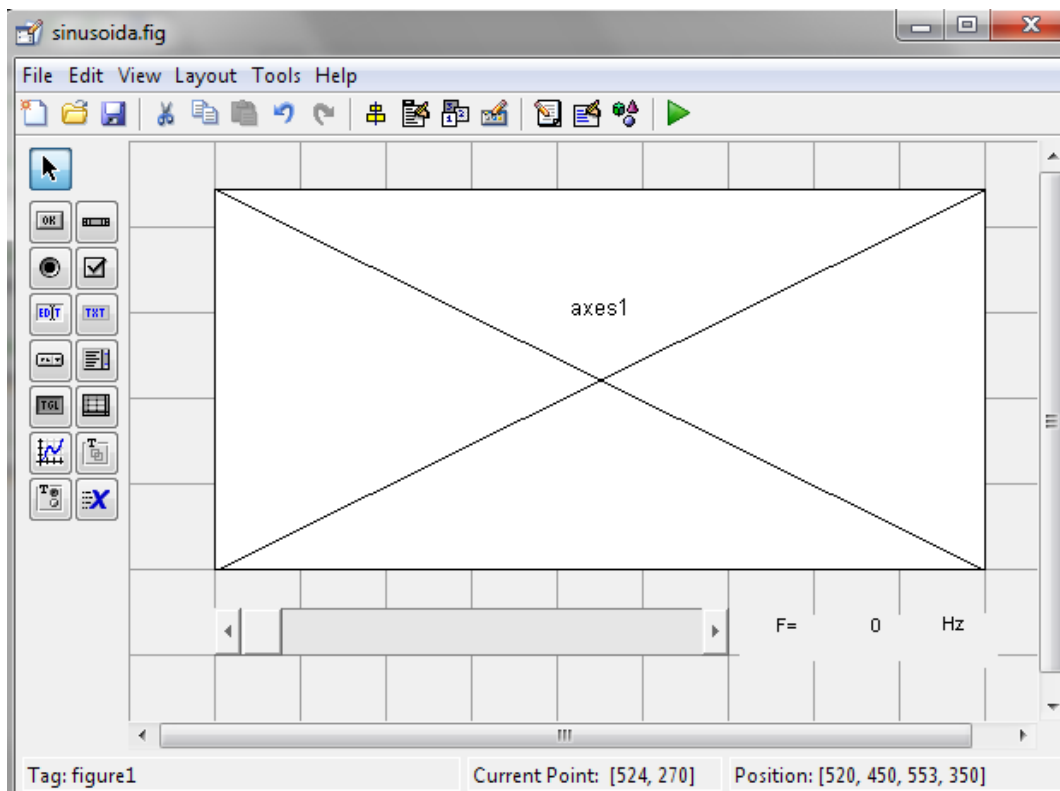


Figura 3. Interfață *sinusoida.fig*

Conținutul fișierului *sinusoida.m* este următorul:

```
function varargout = sinusoida(varargin)
% SINUSOIDA M-file for sinusoida.fig
% --- Executes just before sinusoida is made visible.
function sinusoida_OpeningFcn(hObject, eventdata, handles, varargin)
% --- Outputs from this function are returned to the command line.
function varargout = sinusoida_OutputFcn(hObject, eventdata, handles)
% --- Executes on slider movement.
function sliderF_Callback(hObject, eventdata, handles)
% --- Executes during object creation, after setting all properties.
function sliderF_CreateFcn(hObject, eventdata, handles)
```

La fiecare mișcare a *Slider*-ului se apelează funcția `sliderF_Callback`. În această funcție se poate scrie codul prin care se citește valoarea selectată cu *Slider*-ul `sliderF` și se afișează în câmpul de *Static Text* având *Tag*-ul `valF`. Deoarece variabila în care se salvează valoarea frecvenței trebuie să fie recunoscută și în alte funcții, aceasta se declară variabilă globală (`global F`) în toate funcțiile în care se folosește. Pentru o bună structurare a codului este indicat să se genereze și să se reprezinte grafic sinusoida într-o funcție separată, de exemplu funcția `grafic`. După citirea valorii frecvenței se apelează funcția `grafic(handles)`. În aceste condiții, funcția `sliderF_Callback` este următoarea:

```
function sliderF_Callback(hObject, eventdata, handles)
global F % se declara variabila globala F
% se salveaza in F valoarea selectata cu sliderul
F = get(handles.sliderF, 'Value');
% se scrie in campul Static Text valoarea frecventei F
set(handles.valF, 'String', F);
grafic(handles)
```

Funcția `grafic(handles)` în care se generează și se reprezintă grafic sinusoida este următoarea:

```
function grafic(handles)
global F
A = 2;
Fs = 100;
durata = 2;
t = 0:1/Fs:durata;
x = A*sin(2*pi*F*t);
axes(handles.axes1)
plot(t,x)
```

Dacă se dorește ca frecvența F să aibă o valoare implicită diferită de zero (de exemplu $F = 2\text{Hz}$), se modifică funcția `sinusoida_OpeningFcn` astfel:

```
global F
F = 2; % se initializeaza F cu valoarea 2
% se pozitioneaza sliderul pe pozitia corespunzatoare
set(handles.sliderF, 'Value', F);
% se afiseaza in campul de static text valoarea 2
set(handles.valF, 'String', F);
% se apeleaza functia ce face reprezentarea grafica
grafic(handles)
```

La rularea programului *sinusoida.m*, rezultatul este cel din *Figura 4*.

Observație: implementarea nu este unică, existând multe alte moduri de a realiza o aplicație cu aceeași funcționalitate în Matlab.

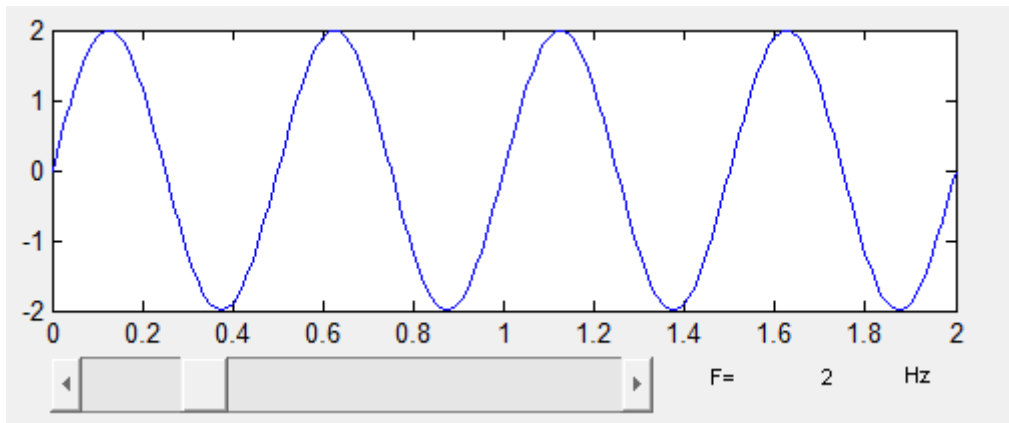


Figura 4. Sinusoidă cu frecvență ce poate fi modificată din *Slider*

Aplicații în Matlab

1. Pentru interfața realizată în *Exemplu 1*, să se mai adauge:

- *Slider* pentru frecvența de eșantionare (între 10 și 200Hz, cu pasul de 1Hz).
- *Slider* pentru amplitudine (între 0 și 5 V, cu pasul de 1V).
- *Static Text* în care să se afișeze frecvența de eșantionare selectată cu *Slider*-ul.
- *Static Text* în care să se afișeze amplitudinea selectată cu *Slider*-ul.

Parametrii implicați să fie $F = 10\text{Hz}$, $F_S = 100\text{Hz}$, $A = 2\text{V}$.

2. Să se realizeze o interfață GUI în care să se afișeze 2 sinusoidale $s1$ și $s2$ cu următorii parametri:

- frecvența de eșantionare $F_S = 1000\text{Hz}$.
- durata = 2 secunde.
- faze inițiale nule.
- frecvența să se modifice dintr-un *Slider*, din 1 în 1, în intervalul $0 \div 20\text{Hz}$.
- amplitudinea să se modifice dintr-un *Slider*, din 1 în 1, în intervalul $0 \div 5\text{V}$.

În câmpuri de *Static Text* se vor afișa valorile selectate din *Slider*-ele de amplitudine și frecvență. La pornirea aplicației, toate *Slider*-ele și câmpurile *StaticText* vor porni cu valori diferite de zero, iar graficele vor fi reprezentate corespunzător valorilor alese.

3. Să se genereze și să se reprezinte grafic semnalul $s3$ obținut din produsul sinusoidelor $s1$ și $s2$ generate la *Problema 2*. Apoi în câmpuri *Static Text*, în dreptul graficului semnalului $s3$ să se afișeze:

- valoarea amplitudinii semnalului $s3$.
- valoarea frecvenței semnalului $s3$.

Indiciu: $\sin(2a) = 2 \sin(a) \cdot \cos(a)$.

4. Pentru *Problema 9* de la *Lucrarea 3*, să se realizeze o interfață grafică în care să se poată modifica din *Slidere* frecvențele sinusoidelor, durata notelor și durata pauzei.

Lucrarea 5

Conversia Analog Numerică

Teorema eșantionării

Obiective: înțelegerea conceptului de eșantionare; realizarea eșantionării unui semnal sinusoidal; înțelegerea conceptului de cuantizare; cuantizarea unui semnal sinusoidal; înțelegerea conceptului de zgomot de cuantizare.

Clasificarea semnalelor

După natura continuă sau discontinuă a domeniului de definiție și a celui de valori, semnalele se pot clasifica conform Tabelului 1.

Tabel 1. Clasificarea semnalelor după natura continuă sau discontinuă a domeniului de definiție și a celui de valori

		Domeniu de valori	
		Continuu	Discret
Domeniu de definiție (timpul)	Continuu	Semnale continue în timp continuu (<i>analogice</i>)	Semnale discrete în timp continuu
	Discret	Semnale continue în timp discret	Semnale discrete în timp discret (<i>numerice</i>)

În continuare ne propunem să prelucrăm semnalele în Matlab, ca urmare singurele tipuri de semnale cu care putem lucra (din cele patru menționate în Tabelul 1) sunt semnalele numerice. Dacă semnalul inițial este analogic, pentru a-l transforma într-un semnal numeric trebuie făcută *Conversia Analog Numerică (CAN)*, realizată prin:

- **eșantionare** = discretizarea domeniului timp.
- **cuantizare** = discretizarea domeniului de valori.

Eșantionarea

Eșantionarea reprezintă discretizarea domeniului de definiție (timp). În urma eșantionării, semnalul este definit doar în anumite momente ale domeniului de definiție, dar amplitudinea poate lua orice valoare reală în domeniul de valori.

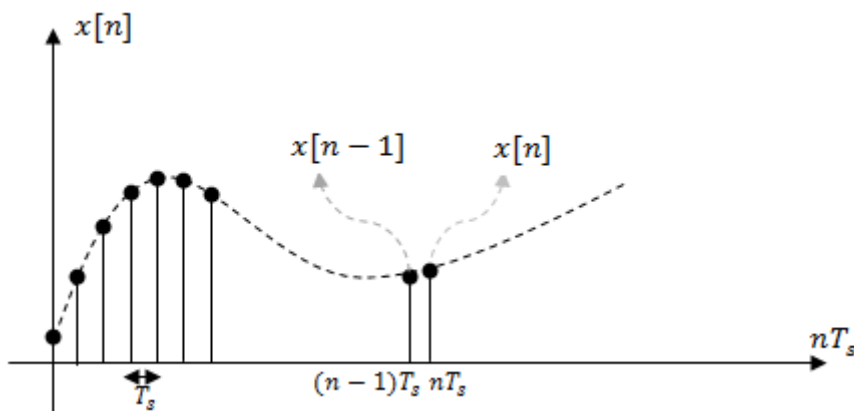


Figura 1. Semnal eșantionat cu perioada de eșantionare T_s

Observații:

- nu este obligatoriu ca distanța (în timp) între eșantioane să fie constantă, însă folosirea unui pas de eșantionare constant oferă avantaje în prelucrarea ulterioară a semnalelor.
- pentru eșantionare uniformă (cu pas constant), intervalul de timp dintre două eșantioane se numește perioadă de eșantionare și se notează T_s (s de la *sampling* = eșantionare în engleză).
- frecvența (rata) de eșantionare reprezintă numărul de eșantioane obținute în unitatea de timp (o secundă) și este în mod uzual notată cu $F_s = 1/T_s$. F_s se măsoară în *hertzi* (Hz) sau eșantioane pe secundă.
- numărul de eșantioane M dintr-o perioadă poate fi calculat ca $M = T/T_s = F_s/F$.

Reamintim:

- pentru un semnal periodic, perioada T a semnalului reprezintă intervalul de timp după care semnalul se repetă.
- pentru un semnal periodic, frecvența de repetiție a semnalului reprezintă numărul de perioade ale semnalului dintr-o secundă $F = 1/T$.

Observații:

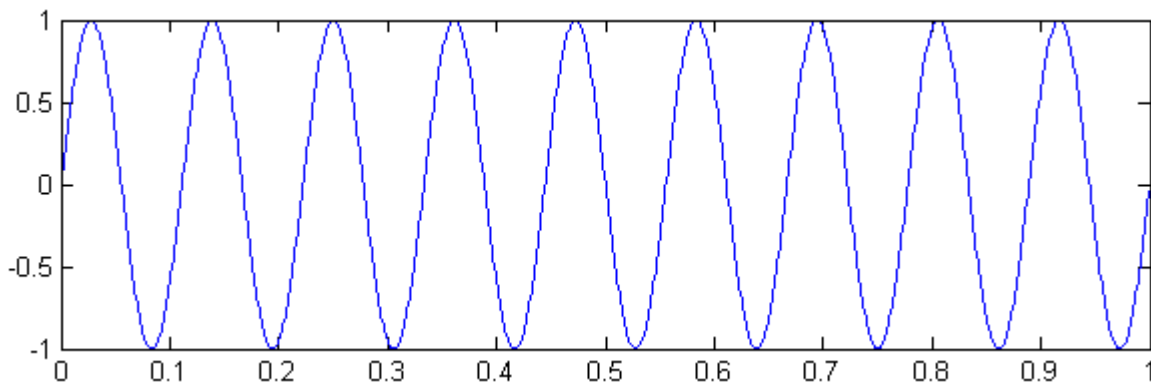
- dacă dorim ca semnalul eșantionat să se apropie cât mai mult de semnalul real, atunci trebuie să luăm un număr cât mai mare de eșantioane, deci să creștem frecvența de eșantionare F_s . Dezavantajul constă în spațiul mare de memorie ocupat.
- dacă dorim un număr mai mic de eșantioane (spațiu de memorie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

Teorema eșantionării. Pentru ca un semnal să poate fi reconstituit din eșantioanele sale, trebuie ca frecvența de eșantionare să fie cel puțin dublul frecvenței maxime din spectrul semnalului.

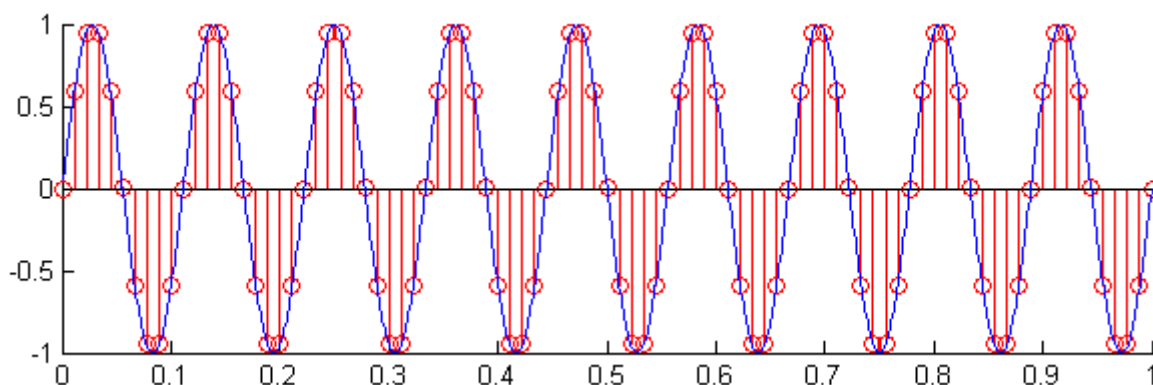
$$F_s \geq 2F_{max}$$

Exemplul următor ilustrează grafic importanța teoremei eșantionării.

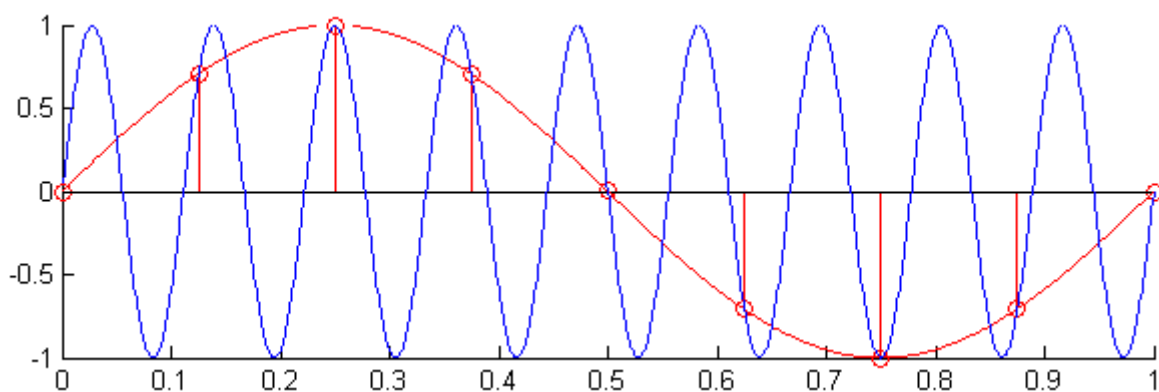
Exemplu. Fie un semnal sinusoidal $s(t)$ având frecvența $F = 9\text{Hz}$ (Figura 2.a). Acest semnal a fost eșantionat cu $F_{s_1} = 90\text{Hz}$ (Figura 2.b). Se observă că din eșantioanele rezultate poate fi refăcut $s(t)$. În Figura 2.c, semnalul $s(t)$ a fost eșantionat cu $F_{s_2} = 8\text{Hz}$ (nu se respectă teorema eșantionării) și se poate observa că aceleași eșantioane rezultă atât prin eșantionarea semnalului $s(t)$, cât și prin eșantionarea unei alte sinusoide cu frecvență mai mica, de 1Hz . În acest caz, semnalul $s(t)$ nu mai poate fi refăcut, formula de reconstituire generând sinusoida de 1Hz .



a) semnalul $s(t)$ cu frecvența $F = 9\text{Hz}$



b) semnalul $s(t)$ eșantionat cu $F_{s_1} = 90\text{Hz}$



c) semnalul $s(t)$ eșantionat cu $F_{s_2} = 8\text{Hz}$

Figura 2. Semnalul $s(t)$ (Figura 2.a) eșantionat respectând teorema eșantionării (Figura 2.b) și eșantionat nerespectând teorema eșantionării (Figura 2.c)

Eșantionarea unui semnal sinusoidal

Formula unui semnal sinusoidal continuu este:

$$x(t) = A \cdot \sin(\omega t + \varphi_0) \quad (1)$$

unde:

- A = amplitudinea maximă
- φ = faza; $\varphi = \omega t + \varphi_0$; $[\varphi]_{S.I} = 1 \text{ rad}$
- ω = pulsația; $\omega = 2\pi \cdot F$; $[\omega]_{S.I} = 1 \text{ rad/s}$
- φ_0 = faza inițială; $[\varphi_0]_{S.I} = 1 \text{ rad}$

Înlocuind $\omega = 2\pi \cdot F$, obținem:

$$x(t) = A \cdot \sin(2\pi \cdot F \cdot t + \varphi_0) \quad (2)$$

În urma eșantionării rezultă semnalul numeric (vectorul):

$$x[n \cdot T_s] = A \cdot \sin(2\pi \cdot F \cdot nT_s + \varphi_0) \quad (3)$$

Formula sinusoidelor discrete se mai poate scrie în funcție de frecvența de eșantionare F_s :

$$x[n] = A \cdot \sin\left(2\pi \cdot n \cdot \frac{F}{F_s} + \varphi_0\right) \quad (4)$$

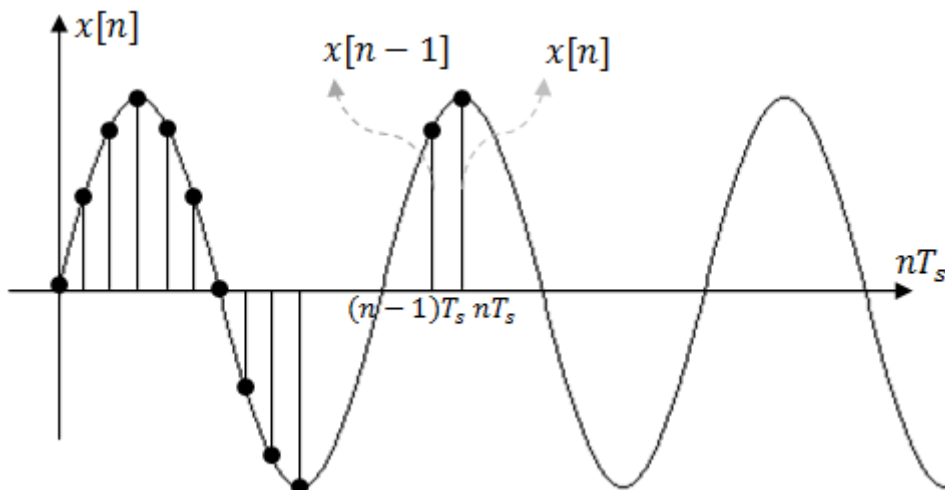


Figura 3. Semnal sinusoidal eșantionat

Exercițiu 1. Fie semnalul $u(t) = \sin(2\pi \cdot 100 \cdot t + \pi/3)$. Care este frecvența minimă de eșantionare, astfel încât semnalul să poată fi reconstituit din eșantioanele sale?

Soluție: $F = 100\text{Hz} \rightarrow F_s \geq 200\text{Hz}$.

Exercițiu 2. Fie semnalul $u(t) = \sin(2\pi \cdot 4 \cdot t) + \sin(2\pi \cdot 6 \cdot t)$. Cât trebuie să fie minimă frecvența de eșantionare, astfel încât să se respecte teorema eșantionării? Care este frecvența de repetiție a semnalului $u(t)$?

Soluție: frecvențele din spectru sunt $F_1 = 4\text{Hz}$ și $F_2 = 6\text{Hz}$, deci frecvența maximă din spectru este $F_{max} = 6\text{Hz} \rightarrow F_s \geq 12\text{Hz}$.

Frecvența de repetiție a semnalului $u(t)$ este $F = 2\text{Hz}$, așa cum se observă și în *Figura 4* (frecvența de repetiție se determină ca *c.m.m.d.c* dintre F_1 și F_2).

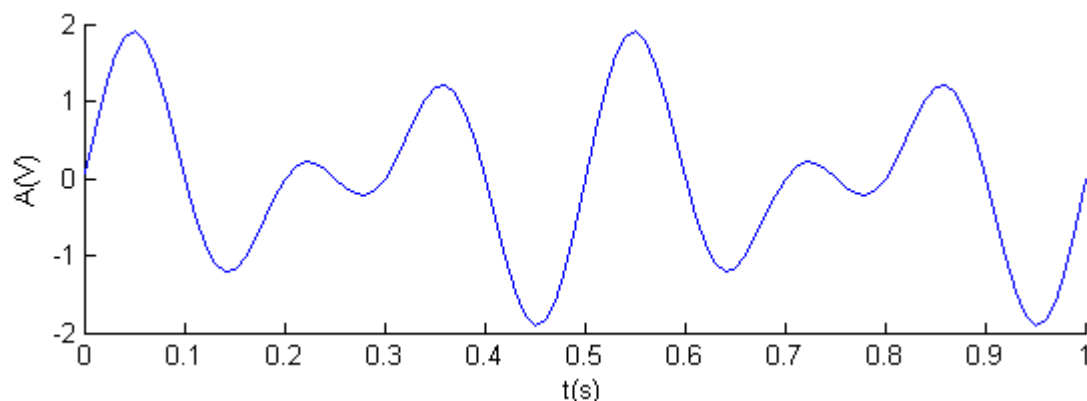


Figura 4. Reprezentarea grafică a semnalului $u(t) = \sin(2\pi \cdot 4 \cdot t) + \sin(2\pi \cdot 6 \cdot t)$

Observație: chiar dacă din punct de vedere matematic este suficient ca F_s să fie minim $2F_{max}$, pentru ca reprezentarea grafică a semnalelor să fie sugestivă este recomandat ca F_s să fie minim $10F_{max}$.

Exemplu de identificare a parametrilor unei sinusoidă numerice

Pentru sinusoida din *Figura 5*, parametrii sunt următorii:

- frecvența de repetiție: $F = 2\text{Hz}$ (sunt 2 oscilații complete ale sinusoidă într-o secundă).
- numărul de eșantioane dintr-o secundă: $F_s = 32\text{Hz}$.
- numărul de eșantioane dintr-o perioadă: $F_s/F = 16$.
- perioada $T = 1/F = 0.5\text{s}$.
- perioada de eșantionare $T_s = 1/F_s = 0.03125\text{s}$.
- amplitudine unitară.
- fază inițială nulă.

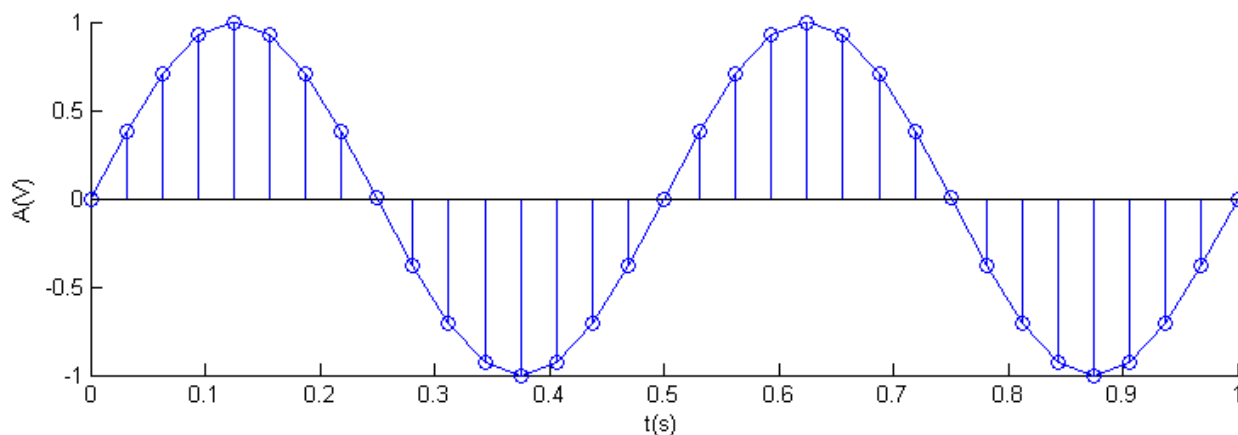


Figura 5. Sinusoidă numerică

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 folosite două metode pentru cuantizarea uniformă sunt *cuantizarea prin rotunjire* respectiv *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.

Observație: când semnalul de intrare (eșantionul analogic) se află în domeniul de lucru al cuantizatorului, eroarea de reprezentare variază în intervalul $-q/2 \div q/2$. Acest tip de eroare, denumită *eroare de cuantizare*, apare atunci 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.

Observație: când semnalul de intrare (eșantionul analogic) se află în domeniul de lucru al cuantizatorului, 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.

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

Tabel 2. Cuantizare 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

Exemplu 2. 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 6.c* (cuantizare prin rotunjire) și semnalul din *Figura 6.d* (cuantizare prin trunchiere).

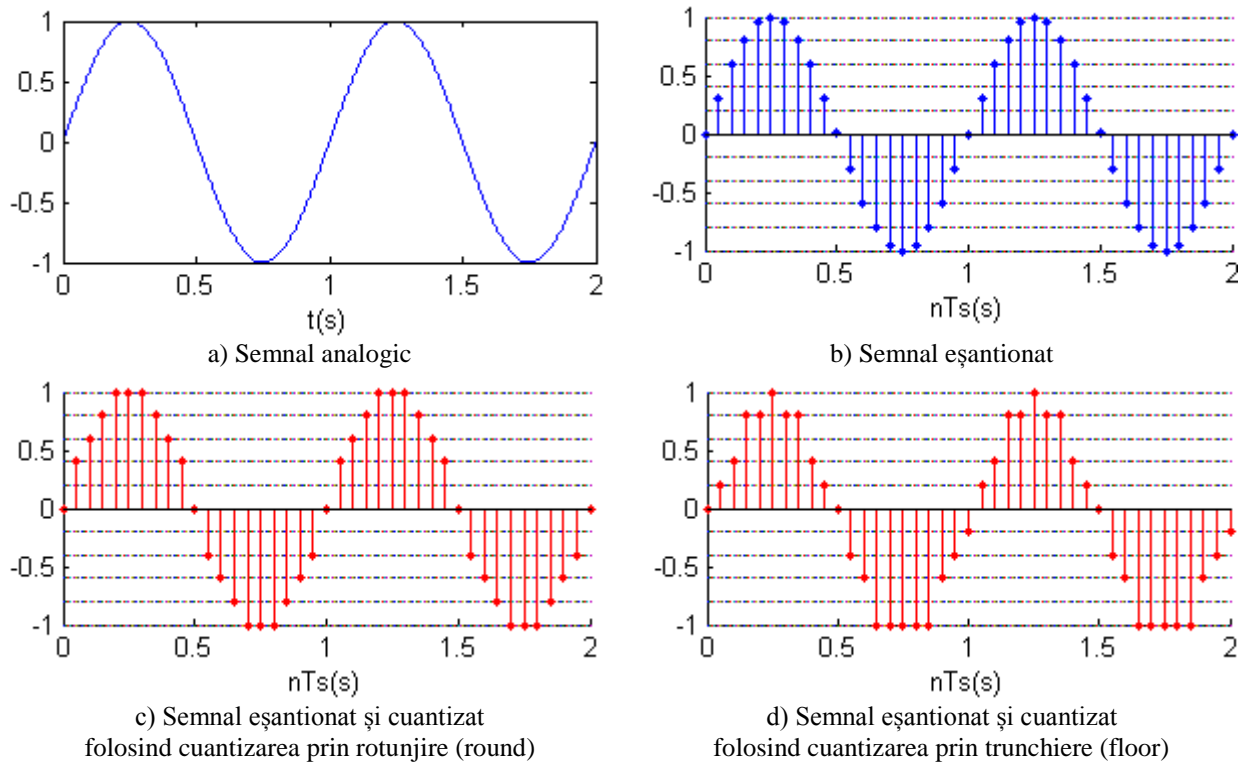


Figura 6. Cuantizare prin rotunjire și trunchiere

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

Exemplu 3. Fie un semnal sinusoidal cuantizat folosind metoda prin rotunjire (*Figura 7.a*). Zgomotul de cuantizare este cel din *Figura 7.b*.

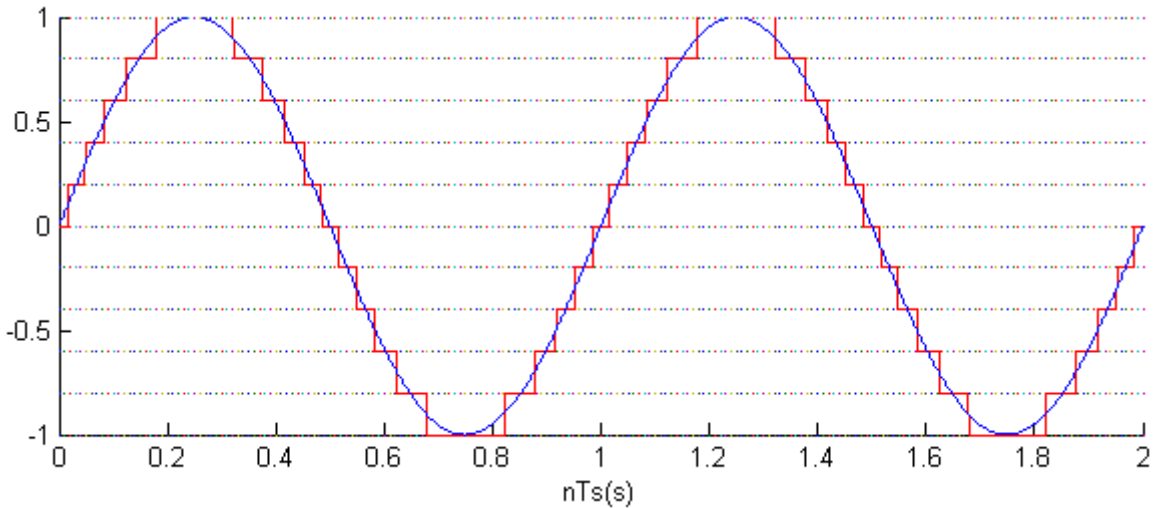


Figura 7.a. Semnal cuantizat folosind metoda prin rotunjire

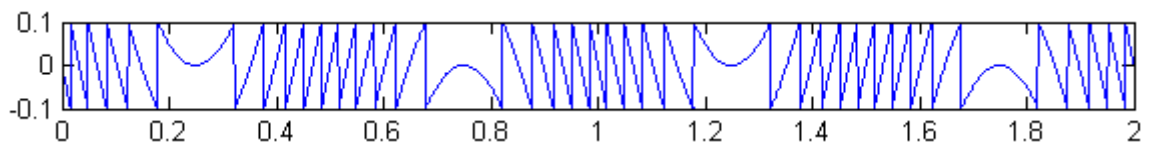


Figura 7.b. Zgomotul de cuantizare

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

Nivel 0 = 0

Nivel 1 = $1/3$

Nivel 2 = $2/3$

Nivel 3 = 1

În Tabelul 3, sunt trecute valorile obținute în urma cuantizării prin rotunjire pentru 10 eșantioane ale semnalului $x[n] = \frac{n}{n+1.5}$

Tabel 3. Cuantizare 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 3*, 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 cu formula:

$$b = \lceil \log_2 N \rceil, \text{ unde:}$$

- b = numărul de biți necesari indicării nivelului de cuantizare.
- N = numărul de niveluri de cuantizare.
- $\lceil \cdot \rceil$ = rotunjire înspre 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 5. Care ar trebui să fie numărul minim de biți per eșantion pentru stocarea semnalului din *Exemplul 2*?

Soluție: în *Exemplul 2* sunt $N = 11$ niveluri de cuantizare, deci numărul minim de biți per eșantion necesar este $b = \lceil \log_2 11 \rceil = 4$.

Aplicații în Matlab

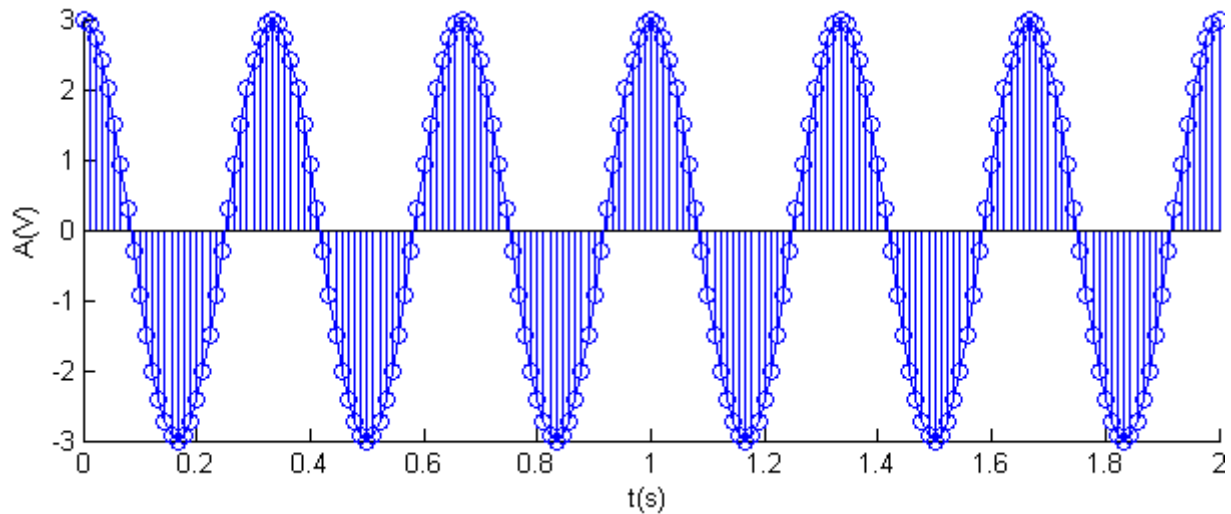
1. Să se parcurgă paginile 22 – 32 din aplicația *DSP_Assistant* (se selectează în *Current Directory* calea către aplicație, iar în *Command Window* se scrie *Main_Filters*).

2. Fie semnalul $u(t) = 3 \cdot \sin(2\pi \cdot 5 \cdot t)$, având durata de 2 secunde. Să se eșantioneze acest semnal cu $F_s = 100\text{Hz}$ și să se reprezinte grafic eșantioanele semnalului.

```
durata = 2;  
F = 5;  
A = 3;  
Fs = 100;  
t = 0:1/Fs:durata;  
u = A*sin(2*pi*F*t);  
figure(1), stem(t,u)
```

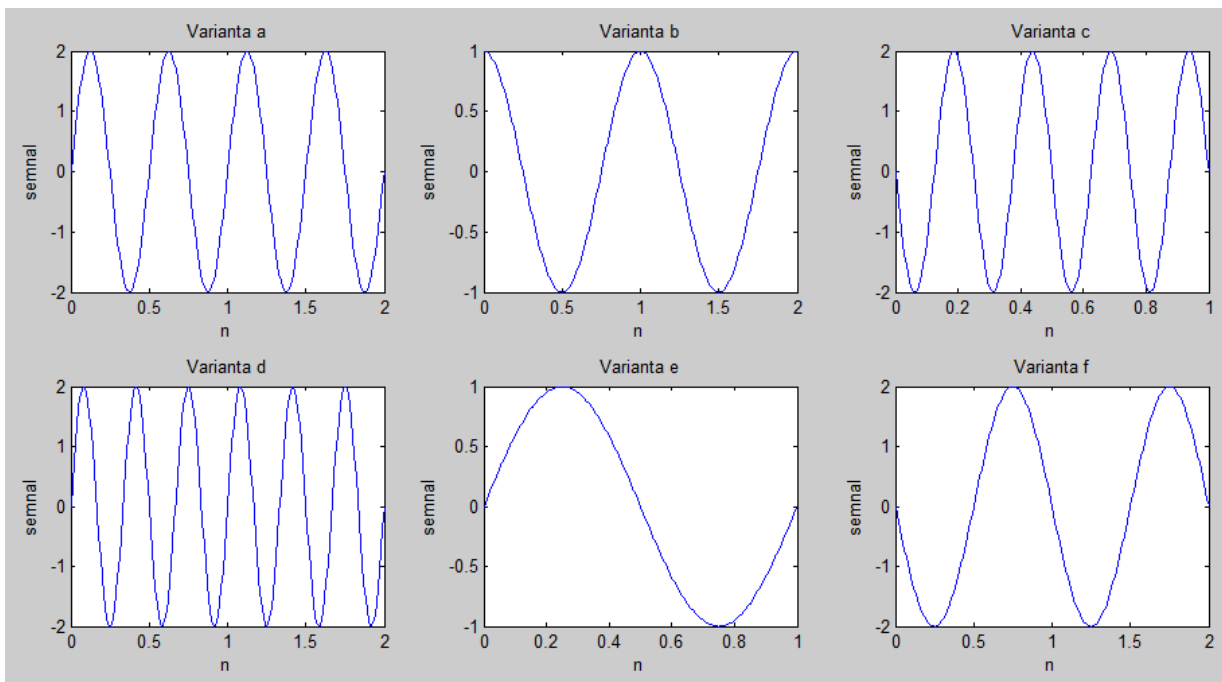
3. Să se genereze în Matlab semnalul din figura următoare.

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



4. Ce se va afișa în *Figure(1)*, în urma rulării următoarelor linii de cod ? (încercuți varianta corectă!)

```
A = 2;
F = 3;
Fs = 100;
durata = 2;
n = 0:1/Fs:durata;
semnal = A*sin(2*pi*F*n);
figure(1),plot(n, semnal)
```



5. Care trebuie să fie frecvența minimă de eșantionare pentru următorul semnal $u(t)$, astfel încât să se respecte teorema eșantionării?

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

6. Fie semnalul $u(t) = 2 \cdot \sin(40\pi t) + 2 \cdot \sin(60\pi t) + 2 \cdot \sin(120\pi t)$. 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 $u(t)$ și să se reprezinte grafic.
 - care este frecvența de repetiție a semnalului $u(t)$?
7. Fie semnalul sinusoidal $s(t) = \sin(2\pi \cdot 9 \cdot t)$. Să se eșantioneze acest semnal cu $F_{s1} = 8\text{Hz}$ și cu $F_{s2} = 90\text{Hz}$ și să se reprezinte grafic. În care dintre cele două cazuri poate fi reconstituit semnalul $s(t)$ din eșantioanele sale?
8. 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?
9. 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.
10. 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` a Matlabului.

$$z = 0.2 \cdot \text{randn}(1, 15);$$
- Să se cuantizeze semnalul z folosind un cuantizor uniform cu nivelurile $\left\{0, \pm \frac{1}{4}, \pm \frac{2}{4}, \pm \frac{3}{4}, \pm 1\right\}$.
11. Să se realizeze un GUI în care să se afișeze o sinusoidă cu următorii parametri: amplitudinea de 5V, frecvența de 2Hz, faza inițială nulă, durata semnalului de 2 secunde. Frecvența de eșantionare (F_s) se modifică dintr-un *Slider* în intervalul 4Hz ÷ 204Hz (pasul cu care variază F_s este de 1Hz). Se observă cum se modifică semnalul eșantionat în funcție de F_s .
12. Să se realizeze un GUI în care să se afișeze un semnal obținut din suma a două sinusoidă având următorii parametri: $A = 5\text{V}$, $F_1 = 4\text{Hz}$, $F_2 = 6\text{Hz}$, fază inițială nulă, durată = 2s. Frecvența de eșantionare F_s se modifică dintr-un *Slider* în intervalul 4Hz ÷ 204Hz. Se observă cum se modifică semnalul eșantionat în funcție de F_s .

Lucrarea 6

Medierea și histograma

Obiective: înțelegerea conceptului de mediere; realizarea medierii pas cu pas pentru un semnal didactic; medierea unui semnal sinusoidal afectat de zgomot; medierea unei imagini afectate de zgomot; realizarea histogramei unui semnal unidimensional; realizarea histogramei unei imagini.

1. Medierea

Medierea este cea mai simplă și utilizată procesare în timp și poate fi folosită cu succes pentru îndepărtarea zgomotului uniform sau a zgomotului alb gaussian așa cum se va putea vedea în aplicațiile următoare.

Formula medierii pentru o fereastră de mediere de M eșantioane este:

$$y[n] = \frac{1}{M} \sum_{k=0}^{M-1} x[n-k] \quad (1)$$

- $x[n]$ este semnalul ce se dorește a fi mediat
- $y[n]$ este semnalul mediat
- M este lungimea ferestrei de mediere (numărul de eșantioane din $x[n]$ care se mediază)

Semnalul $y[n]$ se calculează ca medie a ultimelor M eșantioane ale semnalului $x[n]$.

Exemplu 1. Fie semnalul $x = [3, 3, 3, 6, 6, 6, 9, 9, 9]$.

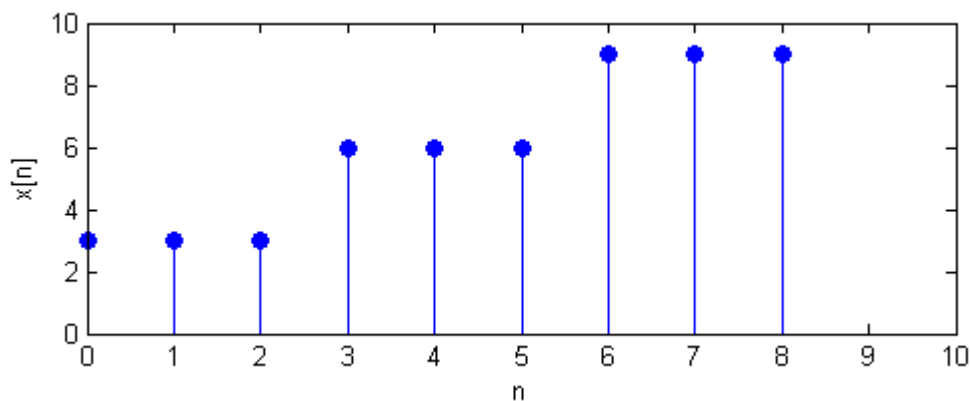


Figura 1. Semnalul $x[n]$

Să se medieze folosind o fereastră de mediere de $M = 3$ eşantioane.

Pentru $M = 3$ formula medierii devine: $y[n] = \frac{x[n]+x[n-1]+x[n-2]}{3}$

- pentru $n = 0 \rightarrow y[0] = \frac{x[0]+x[-1]+x[-2]}{3}$

În componența lui $y[0]$ intră eşantioanele $x[-1]$ și $x[-2]$ care nu se cunosc. În acest caz putem folosi una dintre convențiile:

1. *Zero-padding*: se extinde semnalul cu eşantioane nule (completăm semnalul către stânga cu $M - 1$ zerouri.)

În acest caz $x[-1] = 0$, $x[-2] = 0$ iar $y[0] = \frac{3+0+0}{3} = 1$.

2. Atâta timp cât nu se cunosc toate eşantioanele necesare pentru a-l calcula pe $y[n]$, valoarea lui $y[n]$ se consideră nulă. În acest caz $y[0] = 0$.

Observație: folosind această convenție, primele $M - 1$ eşantioane din $y[n]$ vor fi nule.

În ambele cazuri, primele $M - 1$ eşantioane din $y[n]$ vor fi imprecise.

În continuare vom folosi convenția zero-padding.

- pentru $n = 1 \rightarrow y[1] = \frac{x[1]+x[0]+x[-1]}{3} = \frac{3+3+0}{3} = 2$
- pentru $n = 2 \rightarrow y[2] = \frac{x[2]+x[1]+x[0]}{3} = \frac{3+3+3}{3} = 3$

Se observă că începând de la $n = 2$ (pentru cazul general $n = M - 1$) dispunem de toate eşantioanele semnalului $x[n]$, deci putem calcula fără nicio convenție $y[n]$.

Se obține în final semnalul $y = [1, 2, 3, 4, 5, 6, 7, 8, 9]$.

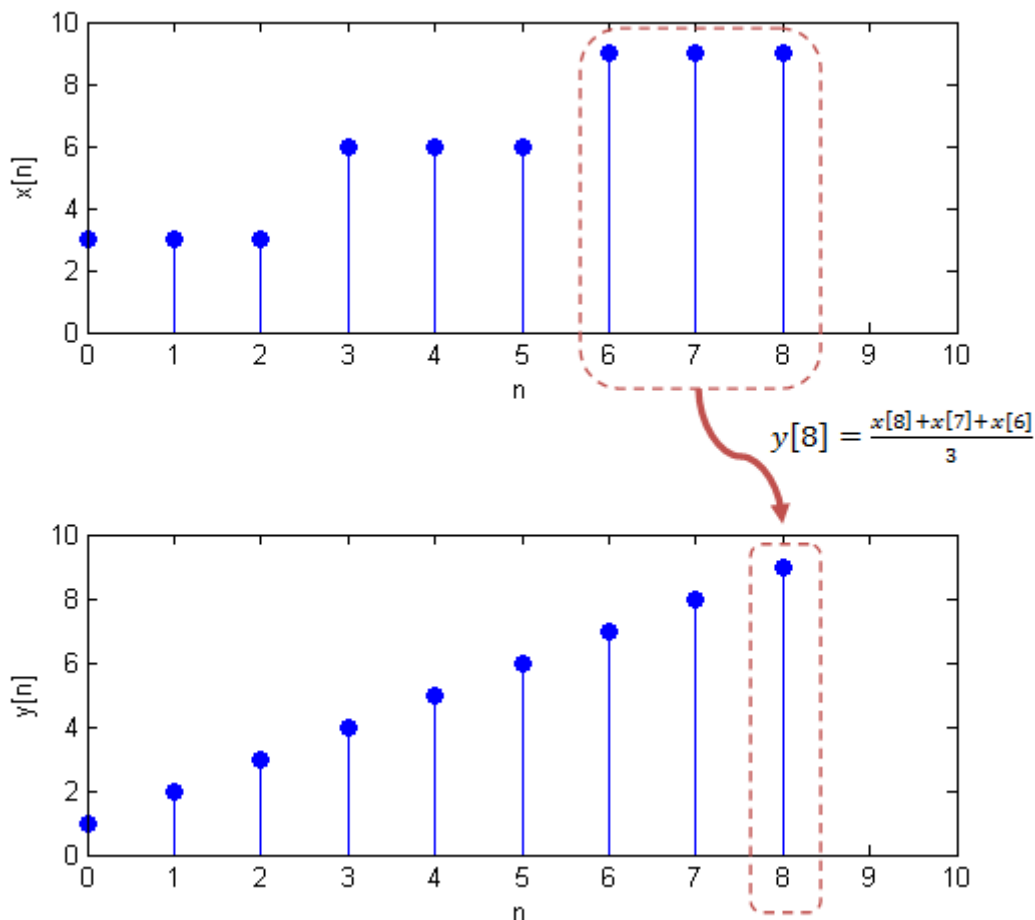


Figura 2. Rezultat mediere (pentru primele 2 eşantioane s-a folosit convenția zero-padding)

Exemplu 2. Fie un semnal sinusoidal peste care s-a adăugat zgomot uniform (semnalul reprezentat cu roșu în *Figura 3*). Se poate observa că zgomotul este (parțial) eliminat (semnalul reprezentat cu verde în *Figura 3*) în urma aplicării medierii.

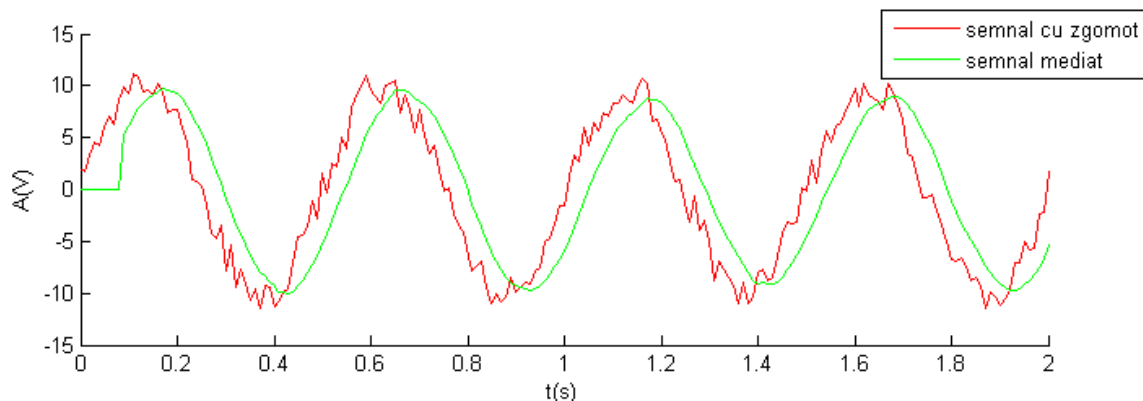
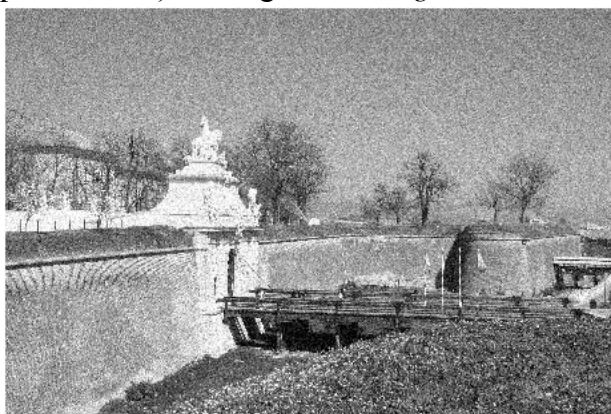


Figura 3. Medierea unei sinusoide cu zgomot

În domeniul timp, acest efect poate fi explicat prin observația că rezultatul medierii semnalului cu zgomot este compus din rezultatele medierii separate a semnalului pur și a zgomotului: $E[s+zg] = E[s] + E[zg]$. Dacă semnalul fără zgomot variază lent, atunci orice M eșantioane consecutive ale acestui semnal sunt aproape identice iar media lor este foarte aproape de valoarea eșantionului M , deci semnalul pur este foarte puțin distorsionat de mediere. Pe de altă parte, eșantioanele de zgomot sunt fie pozitive, fie negative, iar prin mediere acestea tind să se anuleze (*Atenție!* Observațiile sunt valabile doar pentru zgomot aditiv de medie nulă, de exemplu zgomot alb sau uniform centrat în 0).

În domeniul frecvență, efectul medierii poate fi explicat prin analiza spectrală a semnalelor. Astfel, semnalul util care variază lent are componente doar pe frecvențe joase, în timp ce zgomotul are componente în întreg domeniul spectral. Filtrul corespunzător procesului de mediere are caracteristica de frecvență specifică *filtrelor trece jos* (nelăsând să treacă majoritatea componentelor zgomotului).

Exemplu 3. Pentru a efectua medierea asupra unei imagini grayscale (conținând doar niveluri de gri între 0 și 255) este nevoie să se facă medierea atât pe linii cât și pe coloane, deci este nevoie de o fereastră de mediere pătrată/dreptunghiulară. Dacă se aplică medierea asupra unei imagini grayscale afectate de zgomot (ca cea din *Figura 4.a*), folosind o fereastră de mediere de 5x5 pixeli, se obține imaginea din *Figura 4.b*.



a) Imaginea cu zgomot



b) Imaginea mediată

Figura 4. Exemplu de mediere a unei imagini

Se poate observa că în urma medierii zgomotul a fost mult diminuat.

Pentru semnale care variază rapid, medierea poate avea și efecte secundare, ca uniformizarea semnalului prin atenuarea și ”rotunjirea” minimelor/maximelor. În cazul imaginilor cu contururi accentuate, acest efect dă impresia de blur.

2. Histograma

Histograma este utilă pentru a vedea cum sunt distribuite valorile unui semnal și este des utilizată în statistică, procesarea imaginilor etc. Histograma împarte domeniul de valori al unui semnal $x[n]$ în M intervale de obicei egale și întoarce numărul de valori din fiecare interval. Cu alte cuvinte axa Ox a histogramei reflectă intervalul de valori al semnalului $x[n]$, iar axa Oy reflectă numărul de elemente care intră în cadrul fiecărui interval. Reprezentarea grafică se face de obicei sub forma unor bare verticale.

Exemplu 1. Fie semnalul $x = [-2, 1, 6, 4, 0, 2, 6, 4, 5, -2, 3, 4, -3]$. Se dorește vizualizarea histogramei în cazul în care se împarte domeniul de valori în $M = 3$ intervale egale.

Valoarea minimă a semnalului $x[n]$ este $x_{min} = -3$ iar valoarea maximă este $x_{max} = 6$. Împărțindu-se intervalul $[-3, 6]$ în $M = 3$ intervale egale, vor rezulta intervalele: $[-3, 0]$, $(0, 3]$ și $(3, 6]$. În intervalul $[-3, 0]$ sunt 4 valori, în intervalul $(0, 3]$ sunt 3 valori iar în intervalul $(3, 6]$ sunt 6 valori. Histograma este cea din *Figura 5*.

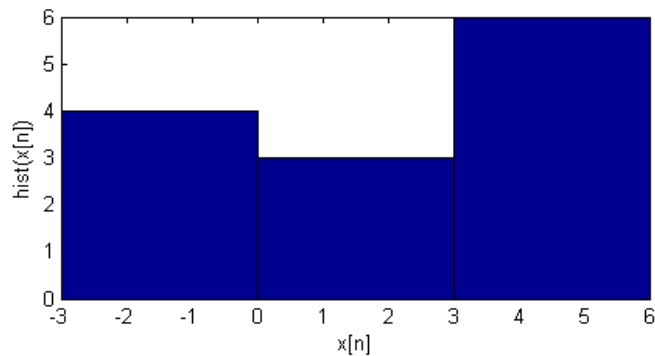


Figura 5. Histograma semnalului $x[n]$ pentru $M = 3$ intervale

Particularizând, histograma poate fi utilizată și pentru a afla de câte ori se repetă fiecare valoare a semnalului din intervalul $x_{min} : x_{max}$. Pentru semnalul $x[n]$ anterior, valoarea -3 apare o singură dată, -2 apare de de 2 ori, -1 nu apare, 0 apare o dată, etc. Histograma este cea din *Figura 6*.

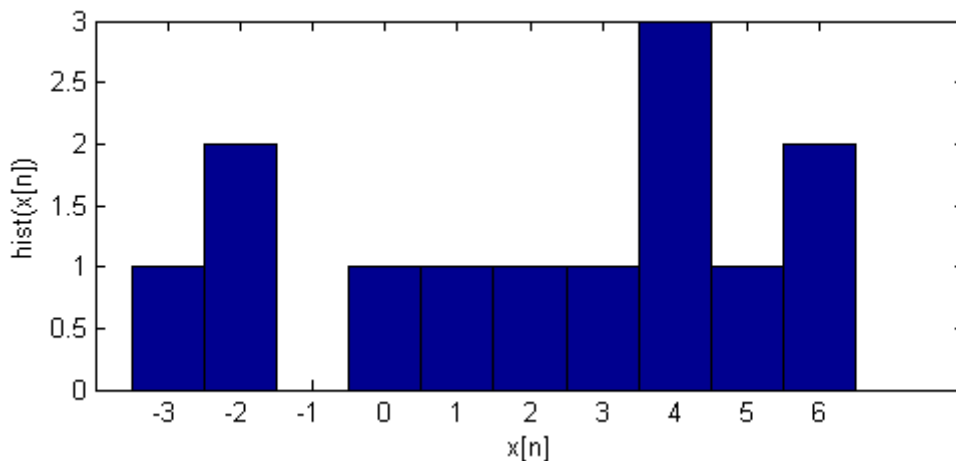


Figura 6. Histograma semnalului $x[n]$, pentru a vizualiza de câte ori apare fiecare valoare din intervalul $x_{min} : x_{max}$

Exemplu 2. Fie un semnal sinusoidal $x(t) = 2\sin(2\pi t)$, cu durata de 3 secunde și eșantionat cu $F_s = 20\text{Hz}$. Să se afișeze eșantioanele semnalului $x[n]$ și histograma care să reflecte de câte ori se repetă fiecare valoare a semnalului $x[n]$.

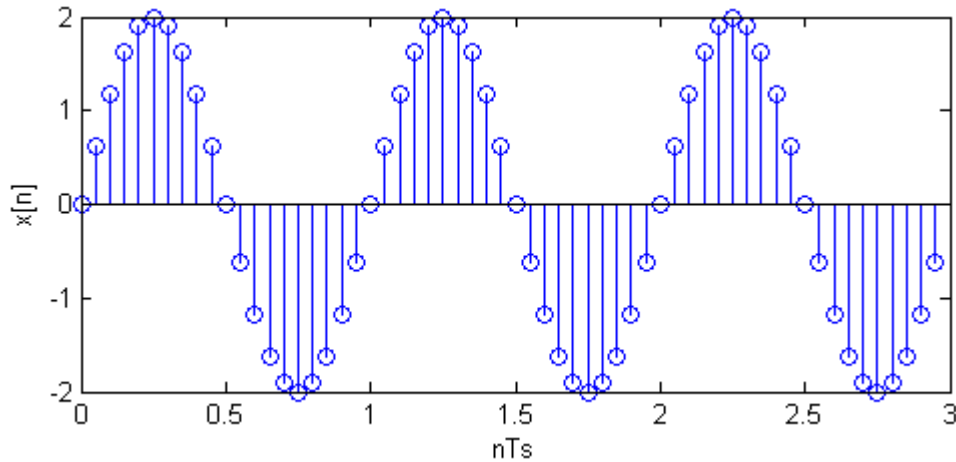


Figura 7. Semnal sinusoidal $x[n]$

Valoarea minimă a semnalului $x[n]$ este -2 iar valoarea maximă este 2 . Celelalte valori ale eșantioanelor sunt: $0.6180, 1.1756, 1.6180, 1.9021, 0, -0.6180, -1.1756, -1.6180, -1.9021$. Valorile 2 și -2 se repetă de câte 3 ori. Toate celelalte valori se repetă de câte 6 ori. Histograma este cea din Figura 8.

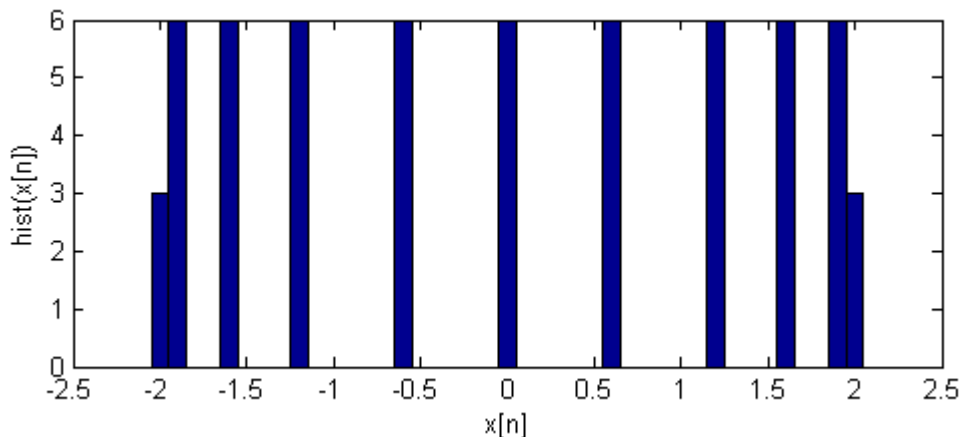


Figura 8. Histograma semnalului $x[n]$ din Figura 7

Exemplu 3. Vizualizarea și interpretarea histogramei unei imagini.

Pentru o imagine grayscale¹, histograma reprezintă numărul de pixeli pentru fiecare nivel de gri. În procesarea imaginilor, histograma este foarte utilă, permițând corectarea problemelor de expunere, majoritatea aparatelor foto afișând histograma imaginii capturate.

În Figura 9.a este afișată o imagine grayscale iar în Figura 9.b este histograma acestei imagini.

¹ O imagine grayscale are 256 niveluri de gri (de la 0 la 255, unde 0 reprezintă negru și 255 reprezintă alb; dacă imaginea este normalată între 0 și 1, atunci albul este reprezentat de valoarea 1). Imaginea grayscale este o matrice, fiecare element al matricei reprezentand nivelul de gri al unui pixel.



Figura 9. a. Imagine grayscale

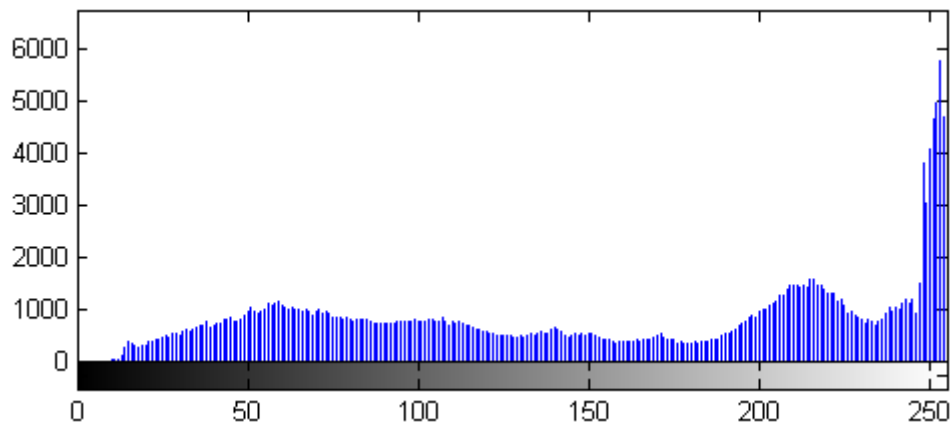


Figura 9.b. Histograma imaginii din Figura 9.a

Din histogramă (*Figura 9.b*) ne putem da seama că imaginea are mulți pixeli cu valori apropiate de 255 (alb), ceea ce indică o posibilă supra-expunere.

Aplicații în Matlab

1. Să se implementeze o funcție numită *mediere* care să realizeze medierea unui semnal unidimensional (vector linie). Funcția primește 2 parametri de intrare (x = semnalul ce se dorește a fi mediat și M = lungimea ferestrei de mediere) și întoarce semnalul y rezultat în urma medierii. Pentru primele $M - 1$ eșantioane se folosește convenția *zero-padding*.

```
function y = mediere(x, M)
% se completeaza semnalul la stanga cu M-1 zerouri (conventia zero-padding)
x_pad = [zeros(1,M-1), x];
for k = M:length(x_pad)
    y(k-M+1) = sum(x_pad(k-M+1:k))/M;
end
```

2. Să se genereze și să se reprezinte grafic în Matlab semnalul $x = [3, 3, 3, 6, 6, 6, 9, 9, 9]$.

Indiciu: pentru a afla numărul elementelor dintr-un vector se poate folosi funcția `length`.

```

x = [3, 3, 3, 6, 6, 6, 9, 9, 9];
n = 0 : length(x)-1; % n = [0, 1, 2, 3, 4, 5, 6, 7, 8]
figure(1),
stem(n,x,'filled'), axis([0, 10, 0, 10]), xlabel('n'), ylabel('x[n]')

```

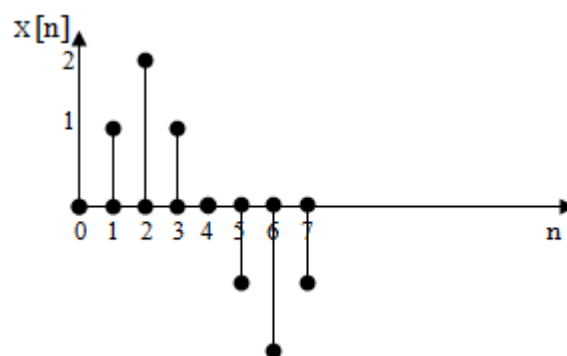
3. Folosind funcția creată la *punctul 1*, să se medieze semnalul $x[n]$ generat la *punctul 2* cu o fereastră de mediere de $M = 3$ eșantioane. Să se reprezinte grafic semnalul mediat.

4. Fie secvența $x[n]$ de mai jos. Folosind medierea pe o fereastră de $M = 2$ eșantioane, se cere:

a) să se determine secvența $y[n]$ obținută în urma medierii.

b) să se reprezinte grafic secvența $y[n]$.

c) știind că funcția `rand` a Matlabului generează valori random în intervalul (0..1), care este sintaxa în Matlab pentru a genera un semnal de aceeași dimensiune cu cea a vectorului $x[n]$ și cu valori în intervalul (1..3)?



5. Fie un semnal sinusoidal $u[n]$ cu următorii parametri: frecvența de 3Hz, frecvența de eșantionare de 300Hz, durata de 1s, amplitudinea de 10V și fază inițială nulă. Sinusoidei $u[n]$ i se adaugă zgomot uniform (valori random în intervalul [-2, 2]). Să se reprezinte grafic semnalul cu zgomot și semnalul cu zgomot mediat folosind o fereastră de mediere de $M = 10$ eșantioane.

6. Să se realizeze o interfață grafică (GUI) care să conțină un sistem de axe (*Axes*) pentru afișarea graficelor și un *Slider* cu ajutorul căruia să se poată modifica numărul eșantioanelor pe care se face medierea (să se poată selecta între 1 și 21 de eșantioane).

a) să se genereze o sinusoidă ($u[n]$) cu următorii parametri: frecvența de 2Hz, frecvența de eșantionare de 200Hz, durata de 2s, amplitudinea de 10V și fază inițială nulă. Să se reprezinte grafic sinusoida.

b) să se genereze un semnal ($zg[n]$) cu valori random în intervalul [-2, 2].

Indiciu. Pentru a genera valori random în intervalul (0, 1) se poate folosi funcția `rand`.

c) să se genereze o sinusoidă cu zgomot ($u_zg[n]$), prin însumarea sinusoidelor de la *punctul a*, cu semnalul random de la *punctul b* ($u_zg[n] = u[n] + zg[n]$).

d) să se genereze semnalul $u_zg[n]$ obținut la *punctul c*. Numărul eșantioanelor pe care se face medierea este stabilit din *Slider*.

e) să se reprezinte grafic, în același sistem de coordonate, cu culori diferite, cele 3 grafice: sinusoida originală $u[n]$, sinusoida cu zgomot $u_zg[n]$ și semnalul mediat.

f) ce se constată că se întâmplă cu semnalul mediat, atunci când mărim dimensiunea ferestrei de mediere? Putem spune că medierea este o filtrare? Ce fel de filtrare?

7. Să se citească în Matlab o imagine căreia să i se adauge zgomot alb gaussian. Folosind o fereastră de mediere de $n \times n$ pixeli, să se medieze imaginea. *Indicii:*

- pentru a citi o imagine se folosește funcția `imread`.
- pentru a adăuga zgomot alb gaussian se poate folosi funcția `imnoise` sau `awgn`.

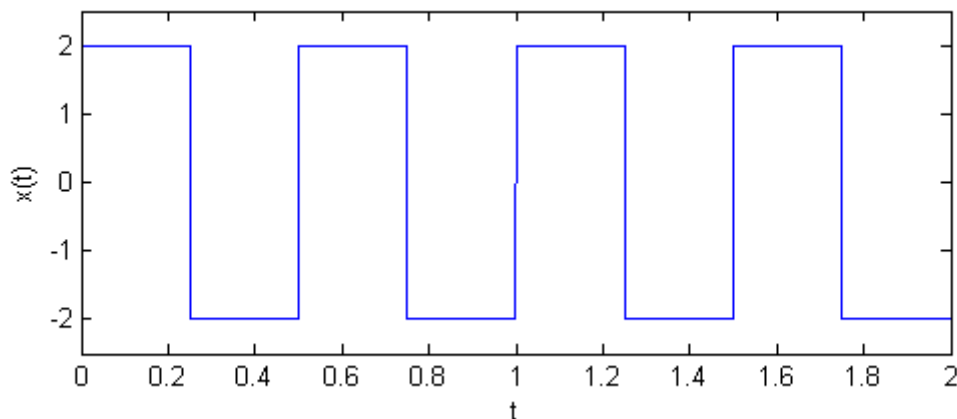
8. Fie un semnal sinusoidal $x(t) = 3\sin(2\pi t)$, cu durata de 2 secunde și eșantionat cu $F_s = 20\text{Hz}$. Să se afișeze eșantioanele semnalului $x[n]$ și histograma acestuia.

Indiciu: pentru a calcula histograma se poate folosi funcția `hist` a Matlabului.

```
F = 1;
Fs = 20;
durata = 2;
A = 3;
t = 0:1/Fs:durata-1/Fs;
x = A*sin(2*pi*F*t);
figure(1)
stem(t,x), xlabel('nTs'), ylabel('x[n]')
figure(2)
hist(x, [min(x):0.1:max(x)])
xlabel('valori x[n]'), ylabel('histograma x[n]')
```

9. Să se implementeze o funcție numită *histograma* care să aibă 2 parametri de intrare (semnalul $x[n]$ căruia i se reprezintă histograma și numărul de intervale M). Atunci când se apelează funcția *histograma* (x, M), să se afișeze histograma semnalului $x[n]$. Rezultatul obținut să se compare cu cel al funcției `hist` a Matlabului.

10. Fie următorul semnal dreptunghiular.



Dacă semnalul este eșantionat cu $F_s = 20\text{Hz}$, cum va arăta histograma?

11. Fie o imagine alb-negru sub forma unei table de șah, dimensiunea fiecărui pătrat fiind de 20×20 pixeli. Cum arată histograma acestei imagini? *Indicii:*

- imaginea este o matrice cu 160 de linii și 160 de coloane. Pătratele albe conțin doar pixeli cu valoarea 255, iar pătratele negre conțin doar pixeli cu valoarea 0.
- pentru a afișa histograma unei imagini se poate folosi funcția `imhist` a Matlabului.

Lucrarea 7

Derivata/Diferențierea

Obiective: înțelegerea conceptului de derivată/diferențiere; realizarea diferențierii pas cu pas pentru un semnal didactic cu scopul obținerii punctelor de extrem; diferențierea unui semnal obținut din suma a două sinusoides; diferențierea unei imagini.

Derivata este o metodă fundamentală de analiză a semnalelor cu diverse aplicații practice cum ar fi: detecția extremelor (maxime și minime) ale unei funcții, detecția contururilor obiectelor unei imagini, determinarea vitezei obiectelor în mișcare etc

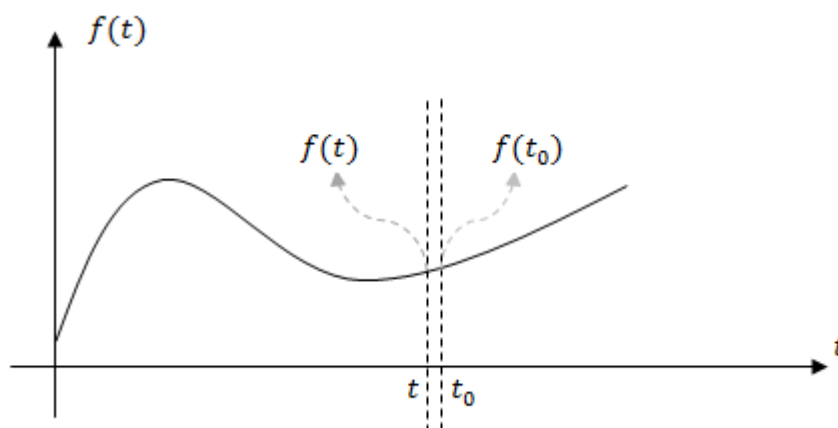


Figura 1. Derivata într-un punct

Derivata într-un punct, pentru o funcție continuă $f(t)$, se scrie:

$$\left. \frac{df(t)}{dt} \right|_{t_0} = \lim_{t \rightarrow t_0} \frac{f(t) - f(t_0)}{t - t_0} \quad (1)$$

și este o mărime a vitezei cu care funcția variază (crește sau descrește) în jurul punctului t_0 . Pentru semnale discrete, termenul de *derivare* este înlocuit cu cel de *diferențiere*.

Pentru un semnal discret $x[n]$, obținut în urma eșantionării cu perioada de eșantionare T_s , formula diferențierii este:

$$y[n] = \frac{x[n] - x[n - 1]}{T_s} \quad (2)$$

Cu alte cuvinte se face diferența a două eșantioane consecutive iar rezultatul obținut se împarte la distanța în timp dintre cele 2 eșantioane, adică la perioada de eșantionare T_s .

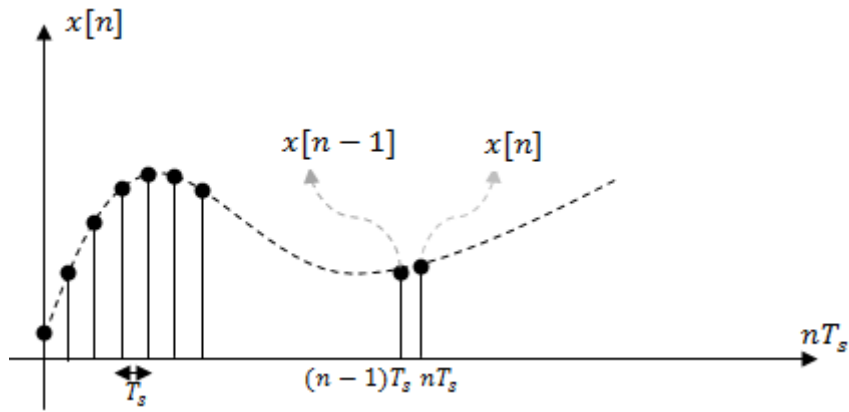


Figura 2. Semnal eșantionat

Exemplu 1. Fie sinusoida $x(t) = \sin(4\pi t)$. Să se determine *punctele de extrem*² ale acestei funcții (punctele de maxim și de minim), știind că semnalul este eșantionat cu $F_s = 16\text{Hz}$.

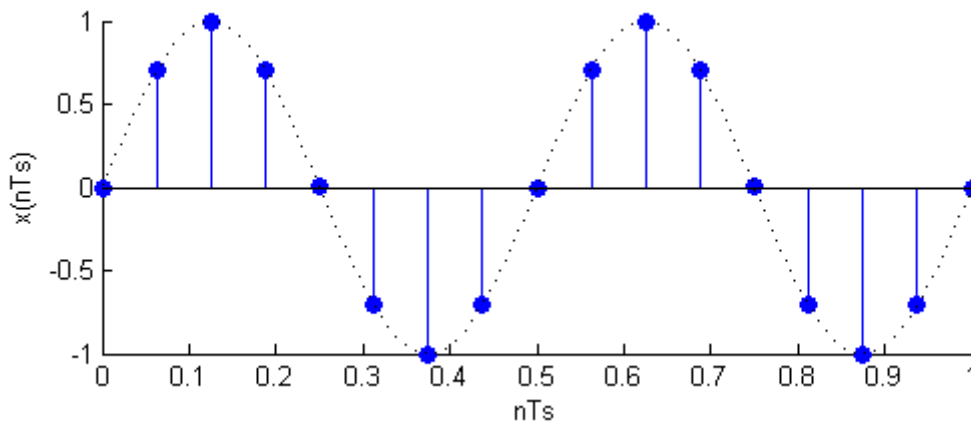


Figura 3. a. Semnalul $x[n]$

Se constată că sinusoida $x(t)$ are amplitudine maximă unitară, fază inițială nulă și frecvența $F = 2\text{Hz}$. În urma eșantionării³ cu $F_s = 16\text{Hz}$, se obține semnalul $x[n]$ cu următoarele valori:

- $x[0] = 0$
- $x[1] = \sin(2 \cdot \pi \cdot F \cdot 1 \cdot T_s) = \sin\left(\frac{\pi}{4}\right) = \frac{\sqrt{2}}{2}$
- $x[2] = \sin(2 \cdot \pi \cdot F \cdot 2 \cdot T_s) = \sin\left(\frac{\pi}{2}\right) = 1$ etc

Se obține astfel vectorul $x = [0, \frac{\sqrt{2}}{2}, 1, \frac{\sqrt{2}}{2}, 0, -\frac{\sqrt{2}}{2}, -1, -\frac{\sqrt{2}}{2}, 0 \dots]$ reprezentat grafic în Figura 3.a.

² Pentru a calcula punctele de extrem ale unei funcții, se calculează derivata de ordin întâi a funcției respective. Se egalează apoi derivata cu zero și se rezolvă ecuația. Valorile găsite sunt punctele de extrem ale funcției.

Exemplu: $f(x) = x^2 - 3x + 2 \Rightarrow f'(x) = 2x - 3; f'(x) = 0 \Rightarrow 2x - 3 = 0 \Rightarrow x = \frac{3}{2}$

Rezultă că funcția $f(x)$ are un punct de extrem în $x = \frac{3}{2}$.

³ Formula după care se eșantionează o sinusoidă $u(t)$ este $u[n] = U_{max} \sin(2 \cdot \pi \cdot F \cdot n \cdot T_s + \varphi_0)$

Semnalul $y[n]$ obținut în urma derivării semnalului $x[n]$ are următoarele valori:

- $y[0] = \frac{x[0]-x[-1]}{T_s}$.

Observație: deoarece semnalul $x[n]$ pornește de la $n = 0$, vom presupune că $x[-1] = 0$.

În aceste condiții, $y[0] = 0$

- $y[1] = \frac{x[1]-x[0]}{T_s} = \frac{0.707-0}{0.0625} = 11.31$

- $y[2] = \frac{x[2]-x[1]}{T_s} = \frac{1-0.707}{0.0625} = 4.68$

Se obține astfel semnalul $y[n]$ din *Figura 3.b*.

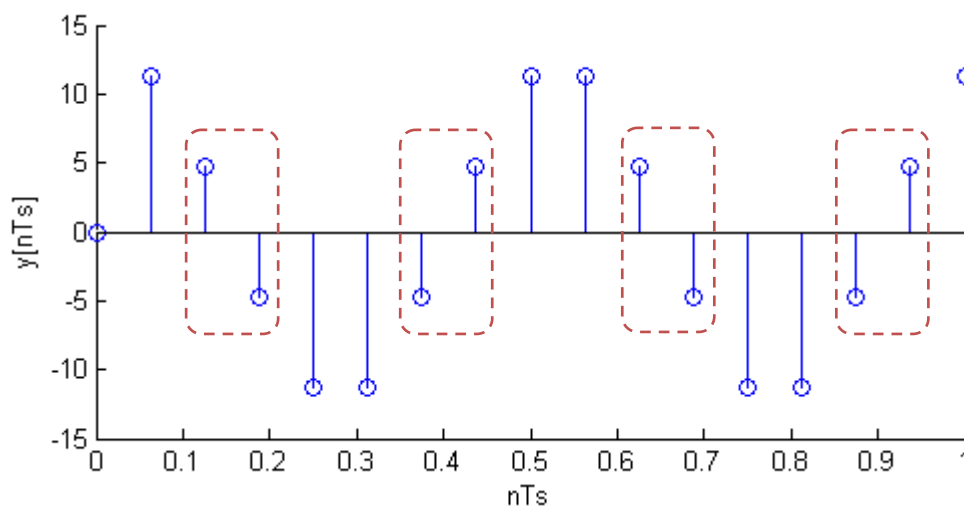


Figura 3.b. Semnalul $y[n]$ obținut în urma derivării semnalului $x[n]$

Observație: Pentru a calcula punctele de extrem ale unui semnal continuu, se verifică în ce puncte derivata este nulă. Pentru un semnal numeric, pentru a detecta extremele se verifică trecerile prin zero ale derivatei. Cu alte cuvinte se verifică când derivata își schimbă semnul (trece de la valori pozitive la valori negative sau de la valori negative la valori pozitive).

Pentru semnalul din exemplul anterior, se observă că al 3-lea eșantion al derivatei are valoare pozitivă iar al 4-lea are valoare negativă. Deci aici are loc o trece prin zero a derivatei, cu alte cuvinte eșantionul al 3-lea al semnalului $x[n]$ este punct de extrem. La fel se întâmplă și cu al 7-lea și al 8-lea eșantion. Unul este negativ iar celălalt pozitiv, deci al 7-lea eșantion este punct de extrem. Rezultă astfel punctele de extrem ale semnalului $x[n]$ ca în *Figura 3.c*.

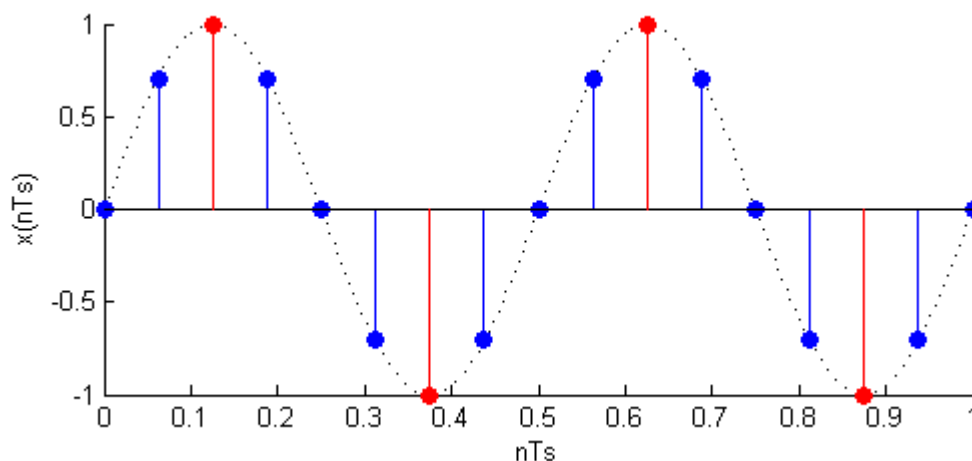


Figura 3.c. Detectia punctelor de extrem

Exemplu 2. Fie un semnal $x[n]$ obținut din suma a două sinusoides unitare cu faza inițială nulă, de frecvențe $F_1 = 2\text{Hz}$ și $F_2 = 50\text{Hz}$ eșantionate cu $F_s = 2000\text{Hz}$ (Figura 4.a). Semnalul $x[n]$ a fost apoi diferențiat, rezultând semnalul din Figura 4.b.

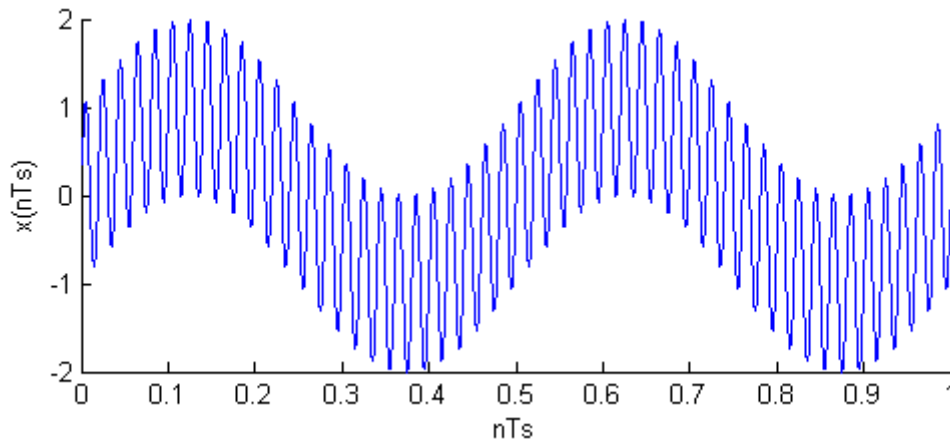


Figura 4.a. Semnalul $x[n]$ obținut din suma a două sinusoides de frecvențe $F_1 = 2\text{Hz}$ și $F_2 = 50\text{Hz}$

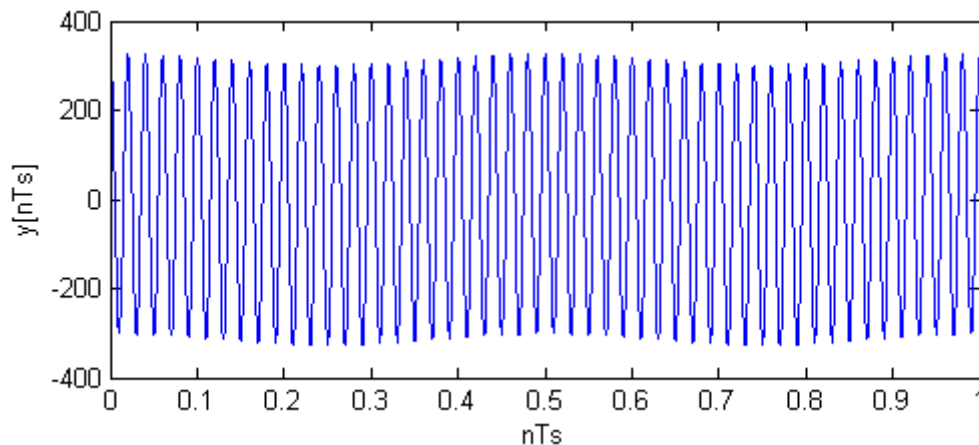


Figura 4.b. Semnalul $y[n]$ obținut în urma diferențierii semnalului $x[n]$

Observație: se constată că în urma diferențierii a rămas doar semnalul cu frecvența $F_2 = 50\text{Hz}$. Putem trage astfel concluzia că diferențierea are proprietăți selective față de semnale de frecvențe diferite, comportându-se ca un *filtru trece sus*.

Exemplu 3. Dacă dorim identificarea conturilor dintr-o imagine, putem folosi diferențierea. În figurile de mai jos sunt trei exemple de identificare a conturilor folosind diferențierea pe verticală (Figura 5.b – se scad câte două linii consecutive), diferențierea pe orizontală (Figura 5.c – se scad câte două coloane consecutive) și diferențierea pe orizontală și pe verticală (Figura 5.d) pentru o imagine alb-negru.

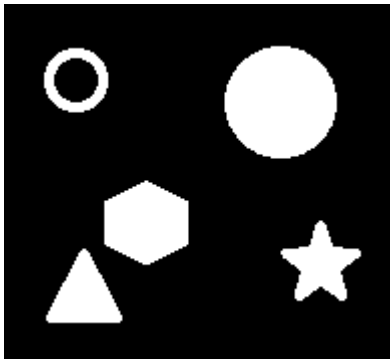


Figura 5. a
Imagine originală

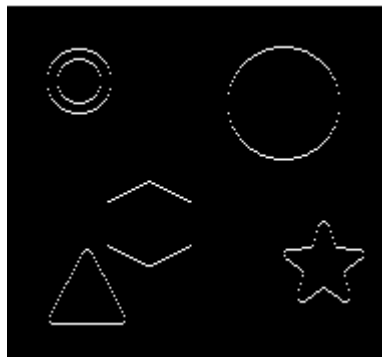


Figura 5. b
Identificare contururi folosind
diferențierea pe verticală

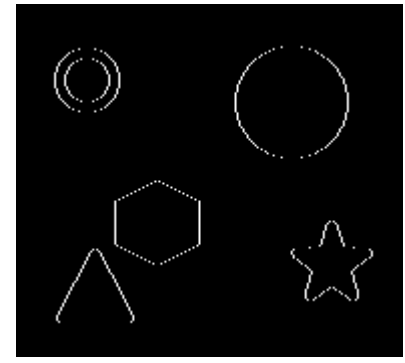


Figura 5. c
Identificare contururi folosind
diferențierea pe orizontală

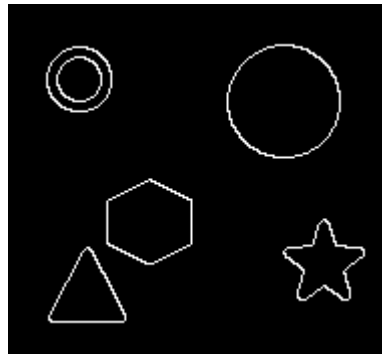


Figura 5. d
Identificare contururi folosind diferențierea pe orizontală și pe verticală

Observație: O imagine alb-negru de dimensiune $M \times N$ pixeli este practic o matrice de dimensiune $M \times N$, conținând doar valori de 0 și 1 (sau 255). În exemplul de mai sus, fundalul este reprezentat de pixeli având valoarea 0 iar obiectele sunt reprezentate de pixeli având valoarea 1. În urma diferențierii pe verticală, se scad elementele a două linii consecutive: $y[m, n] = x[m, n] - x[m - 1, n]$. Dacă pe o coloană sunt doi pixeli consecutivi de valoare 0 sau 1, în urma diferențierii pe verticală se obține valoarea 0. Dacă pe o coloană sunt însă doi pixeli consecutivi având valorile 0 și 1, sau 1 și 0, acest lucru semnifică o trecere de la fundal la obiect (respectiv de la obiect la fundal), deci este un contur. În acest caz, în urma diferențierii pe verticală se obține valoarea absolută 1. Făcând diferențierea pe verticală, nu se vor detecta contururile verticale.

1. Să se implementeze în Matlab o funcție numită *diferențiere*, care să realizeze diferențierea unui semnal unidimensional (vector linie). Funcția primește 2 parametri de intrare (semnalul ce se dorește a fi diferențiat și frecvența de eșantionare) și întoarce ca parametru de ieșire semnalul obținut în urma diferențierii.

```
function y = diferentiere(x, Fs)
y(1) = 0;
for n = 2:length(x)
    y(n) = (x(n) - x(n-1)) * Fs;
end
```

Observație! Să se implementeze codul de mai sus fără a se folosi ciclul *for*.

2. Să se genereze și să se afișeze grafic o sinusoidă $x[n]$, cu următorii parametri: frecvența $F = 2\text{Hz}$, frecvența de eșantionare $F_s = 40\text{Hz}$, amplitudine maximă unitară, fază inițială nulă și durată de 1s. Câte eșantioane va avea sinusoida?

```
F = 2;
Fs = 40;
durata = 1;
t = 0:1/Fs:durata-1/Fs;
x = sin(2*pi*F*t);
figure(1)
stem(t,x, 'filled')
xlabel('nTs'), ylabel('x(nTs)')
```

Observație! Semnalul $x[n]$ va avea $F_s \cdot \text{durata}$ eșantioane, adică 40 de eșantioane.

3. Să se diferențieze semnalul $x[n]$ generat anterior, folosind funcția *diferențiere*.

4. Să se găsească punctele de extrem ale semnalului $x[n]$ generat anterior, folosind rezultatul diferențierii de la *punctul 3*.

Observație! Pentru a verifica trecerile prin zero ale derivatei, se pot înmulți două valori consecutive ale derivatei. Dacă rezultatul este negativ, atunci s-a descoperit un punct de extrem. În codul propus mai jos, indicele eșantionului de extrem este salvat în vectorul *poz_extrem*.

```
poz_extrem = [];
for i = 1:length(y)-1
    if (y(i+1)*y(i)<0)
        poz_extrem = [poz_extrem, i];
    end
end
```

Observație! Să se implementeze codul de mai sus fără a se folosi ciclul *for*.

5. Să se reprezinte în același grafic semnalul $x[n]$ de la *punctul 2* și eșantioanele din punctele de extrem determinate la *punctul 4*.

```
figure(2)
hold on
    plot(t,x)
    stem(t(poz_extrem), x(poz_extrem), 'r')
hold off
xlabel('nTs'), ylabel('x(nTs)')
```

6. Să se repete cerințele anterioare, pentru $F_s = 200\text{Hz}$.

7. Fie funcția $f(x) = x^2 - 3x + 2$. Să se reprezinte grafic funcția pentru $x \in [0,3]$. Folosind diferențierea să se determine punctul de extrem și să se reprezinte grafic.

8. Fie funcția $f(x) = x^3 - 6x^2 + 11x - 6$.

a) Să se reprezinte grafic funcția pentru $x \in [0,4]$.

b) Folosind diferențierea să se determine punctele de extrem și să se reprezinte grafic.

c) Să se determine intervalele pentru care funcția este convexă/concavă prin determinarea semnului derivatei de ordin 2.

9. Fie două sinusoides $x_1[n]$ și $x_2[n]$ având următorii parametri: amplitudine unitară, fază inițială nulă, frecvențe $F_1 = 2\text{Hz}$ și $F_2 = 50\text{Hz}$, frecvență de eșantionare $F_s = 2000\text{Hz}$ și durata = 1 secundă. Fie $x[n] = x_1[n] + x_2[n]$. Se cere:

- Să se reprezinte în aceeași figură, în sisteme de coordonate diferite, semnalele $x_1[n]$, $x_2[n]$ și $x[n]$.
- Să se reprezinte grafic semnalul $y[n]$ obținut în urma diferențierii lui $x[n]$.
- Care este frecvența de repetiție a semnalului $y[n]$?

10. Să se genereze în Matlab o tablă de șah, având numai pătrățele albe și negre, dimensiunea unui pătrățel fiind de 50×50 pixeli. Să se aplice apoi:

- Diferențierea pe linii. Ce se obține?
- Diferențierea pe coloane. Ce se obține?
- Diferențierea pe linii și coloane. Ce se obține?

Indiciu: tabla de șah va fi o matrice de dimensiune 400×400 , conținând doar valori de 0 (pentru pătrățelele negre) și 1 (pentru pătrățelele albe).

Lucrarea 8

Corelația

Obiective: înțelegerea conceptului de corelație și corelație normată; realizarea corelației pas cu pas pentru un semnal didactic; utilizarea corelației pentru identificarea unui șablon într-un semnal sinusoidal afectat de zgomot; utilizarea corelației pentru detecția bătăilor inimii într-un semnal EKG; utilizarea corelației pentru detecția unei sigle într-o imagine.

De multe ori este util să cunoaștem similaritatea dintre două semnale, iar corelația ne poate ajuta în acest sens. În mod frecvent corelația este întâlnită în aplicații radar (pentru identificarea semnalelor reflectate de avioane), în procesarea imaginilor (pentru identificarea obiectelor) etc. Cu alte cuvinte corelația este adesea folosită pentru a identifica o formă cunoscută (șablon) într-un semnal dat.

Formula folosită pentru calculul corelației a două semnale unidimensionale este:

$$y[n] = \sum_{k=0}^{M-1} h[k] \cdot x[n+k] \quad (1)$$

- $h[k]$ = șablon
- M = numărul de eșantioane ale șablonului
- $x[n]$ = semnalul în care se caută șablonul

Formula de mai sus funcționează însă numai în cazul semnalelor pentru care cunoaștem de la început valorile tuturor eșantioanelor. Pentru prelucrarea semnalelor în timp real, *formula 1* nu se poate folosi deoarece este nevoie de eșantioane viitoare. În acest caz se va întârzia semnalul $x[n]$ cu $M-1$ eșantioane, formula devenind:

$$y[n] = \sum_{k=0}^{M-1} h[k] \cdot x[n+k-(M-1)] \quad (2)$$

În continuare vom calcula corelația folosindu-ne de *formula 2*.

Pentru a putea interpreta rezultatele obținute în urma corelației trebuie ca acestea să fie normate.

Pentru normare se folosește *formula 3*.

$$y_{\text{normat}}[n] = \frac{y[n]}{\sqrt{\sum_{k=0}^{M-1} (h[k])^2} \cdot \sqrt{\sum_{k=0}^{M-1} (x[n+k-(M-1)])^2}} \quad (3)$$

În urma normării se obțin valori în intervalul $[-1, 1]$ unde:

- **1** reprezintă corelație pozitivă perfectă: *Exemple:*
 - $h = [1, 2, 3]$ și $x = [1, 2, 3]$ au corelație pozitivă perfectă deoarece semnalele sunt identice.
 - $h = [1, 2, 3]$ și $x = [2, 4, 6]$ au corelație pozitivă perfectă deoarece $x[n]$ este o variantă amplificată a lui $h[n]$.
- **0** reprezintă corelație nulă, adică semnalele sunt complet independente. *Exemplu:* când un semnal este generat random.
- **-1** reprezintă corelație negativă perfectă. *Exemplu:* semnale în antifază.

Exemplu 1. Fie semnalul discret $x[n]$ având următoarele valori:

$x = [2, 3, -1, 1, 2, 3, 0, 1, -2, 1, -1, 2, 4, 6, -1, -2, -3]$.

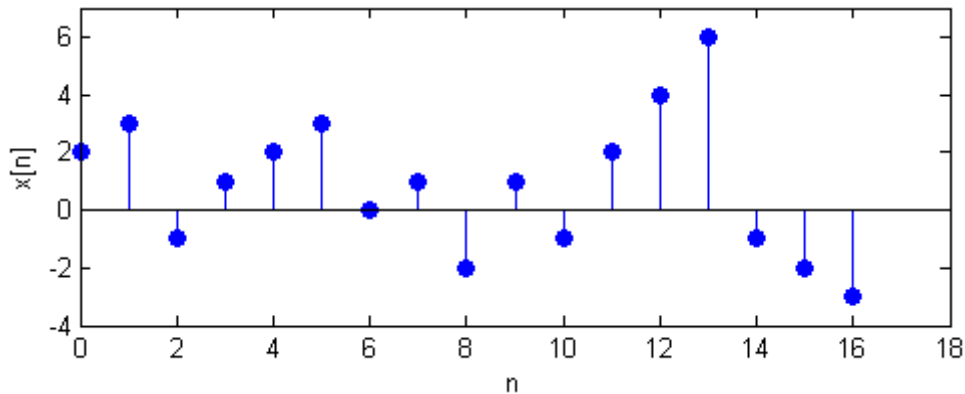


Figura 1. Semnalul $x[n]$

Se dorește să se identifice apariția șablonului $h = [1, 2, 3]$ în semnalul $x[n]$.

Aplicând corelația se obține:

- pentru $n = 0 \rightarrow y[0] = h[0] \cdot x[-2] + h[1] \cdot x[-1] + h[2] \cdot x[0]$

În componența lui $y[0]$ intră eșantioanele $x[-1]$ și $x[-2]$ care nu se cunosc. În acest caz putem folosi una dintre convențiile:

1. Atâta timp cât nu se cunosc toate eșantioanele necesare pentru a-l calcula pe $y[n]$, valoarea lui $y[n]$ se consideră nulă. În acest caz $y[0] = 0$.

Observație: folosind această convenție, primele $M - 1$ eșantioane din $y[n]$ vor fi nule.

2. *Zero-padding:* se extinde semnalul cu eșantioane nule.

În acest caz $x[-1] = 0$, $x[-2] = 0$ iar $y[0] = 1 \cdot 0 + 2 \cdot 0 + 3 \cdot 2 = 6$

Observație: folosind această convenție, este nevoie să completăm semnalul către stânga cu $M - 1$ zerouri.

În ambele cazuri, primele $M - 1$ eșantioane din $y[n]$ vor fi imprecise.

În continuare vom folosi convenția zero-padding.

- pentru $n = 1 \rightarrow y[1] = h[0] \cdot x[-1] + h[1] \cdot x[0] + h[2] \cdot x[1] = 13$
- pentru $n = 2 \rightarrow y[2] = h[0] \cdot x[0] + h[1] \cdot x[1] + h[2] \cdot x[2] = 5$

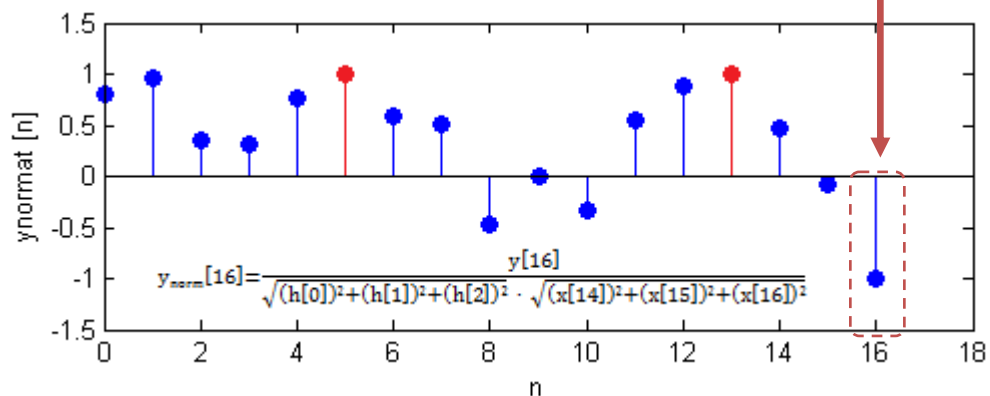
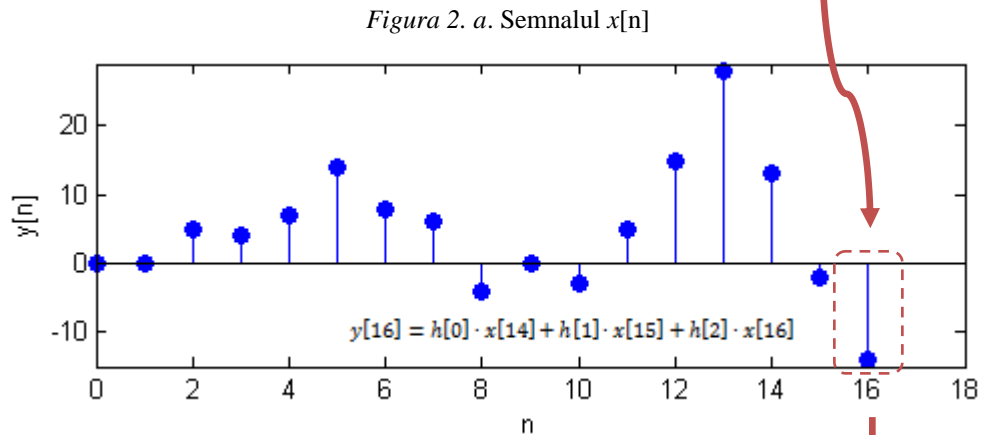
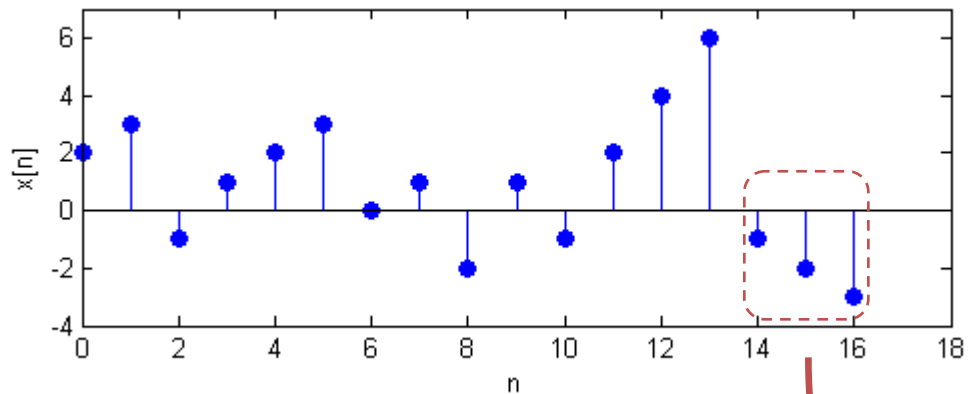
Se observă că începând de la $n = 2$ (pentru cazul general $n = M - 1$) dispunem de toate eșantioanele semnalului $x[n]$, deci putem calcula fără nicio convenție $y[n]$.

Vom obține în final $y = [6, 13, 5, 4, 7, 14, 8, 6, -4, 0, -3, 5, 15, 28, 13, -2, -14]$ (Figura 2.c).

În continuare, pentru a putea interpreta rezultatele obținute în urma corelației va trebui ca acestea să fie normate folosind formula 3. Se obțin astfel valorile:

- $n = 0 \rightarrow y_{\text{norm}}[0] = \frac{y[0]}{\sqrt{(h[0])^2+(h[1])^2+(h[2])^2} \cdot \sqrt{(x[-2])^2+(x[-1])^2+(x[0])^2}} = \frac{6}{\sqrt{14} \cdot \sqrt{14}} = 0.8$
- $n = 1 \rightarrow y_{\text{norm}}[1] = \frac{y[1]}{\sqrt{(h[0])^2+(h[1])^2+(h[2])^2} \cdot \sqrt{(x[-1])^2+(x[0])^2+(x[1])^2}} = \frac{13}{\sqrt{14} \cdot \sqrt{13}} = 0.96$
- etc
- $n = 5 \rightarrow y_{\text{norm}}[5] = \frac{y[5]}{\sqrt{(h[0])^2+(h[1])^2+(h[2])^2} \cdot \sqrt{(x[3])^2+(x[4])^2+(x[5])^2}} = \frac{14}{\sqrt{14} \cdot \sqrt{14}} = 1$
- etc
- $n = 13 \rightarrow y_{\text{norm}}[13] = \frac{y[13]}{\sqrt{(h[0])^2+(h[1])^2+(h[2])^2} \cdot \sqrt{(x[11])^2+(x[12])^2+(x[13])^2}} = \frac{28}{\sqrt{14} \cdot \sqrt{56}} = 1$
- etc

În urma normării se obține semnalul din *Figura 2.c*.



Analizând *Figura 2.c* se observă că în urma normării s-au obținut valori maxime pentru $y[5]$ și $y[13]$ ($y[5] = 1$ și $y[13] = 1$).

- Pentru a-l calcula pe $y[5]$, s-au folosit eșantioanele $x[3]$, $x[4]$ și $x[5]$, deci șablonul $h[n]$ se aseamănă perfect cu secvența $x[3]$, $x[4]$, $x[5]$.
- Pentru a-l calcula pe $y[13]$, s-au folosit eșantioanele $x[11]$, $x[12]$ și $x[13]$, deci șablonul $h[n]$ se aseamănă perfect cu secvența $x[11]$, $x[12]$, $x[13]$.

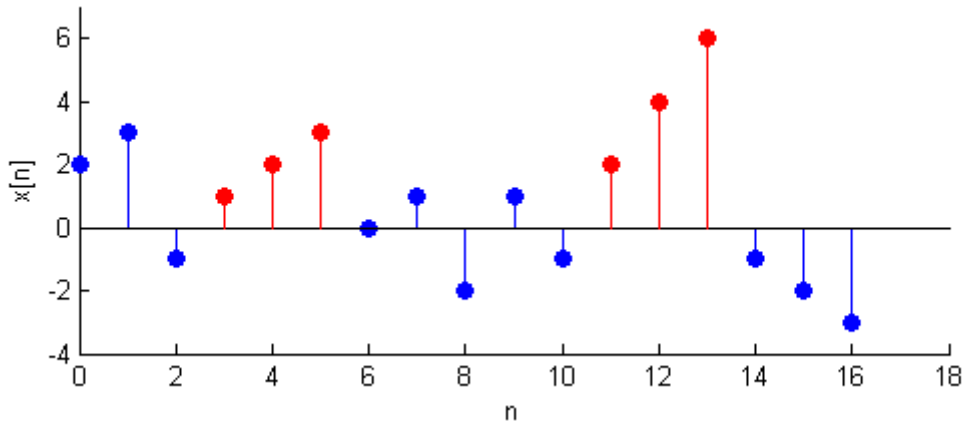


Figura 3. Identificare șablon $h[n]$ în semnalul $x[n]$

Exemplu 2. Fie semnalul sinusoidal $x(t) = 5 \sin(4\pi t)$ care se eșantionează. Se copiază în vectorul șablon $h[n]$ jumătate din cea de-a treia perioadă a semnalului $x[n]$. Semnalului $x[n]$ i se adaugă zgomot uniform, obținându-se astfel semnalul $x_zg[n]$. Să se identifice de câte ori apare $h[n]$ în $x_zg[n]$.

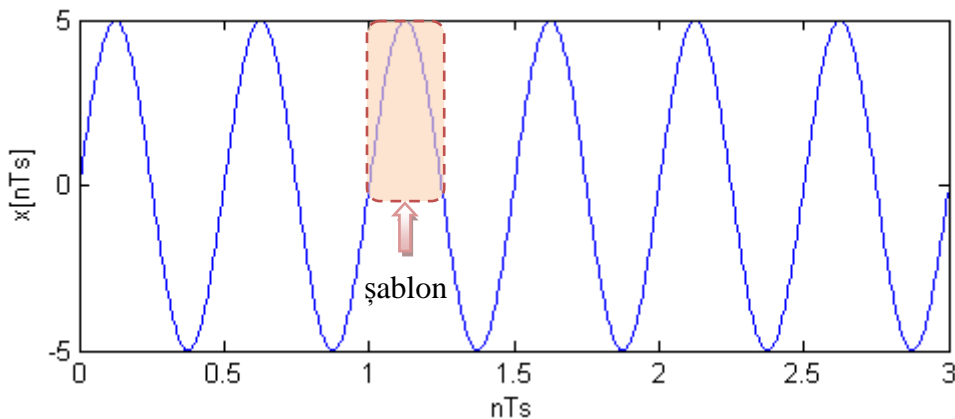


Figura 4. Semnalul $x[n]$ și șablonul $h[n]$

În urma corelației normate nu se va găsi nicăieri valoarea 1, deoarece semnalului $x[n]$ din care s-a selectat șablonul i s-a adăugat zgomot. Valoarea maximă a corelației normate a scăzut la 0.9949. Pentru acest exemplu este suficient să punem pragul corelației normate egal cu 0.99 (o valoare de altfel foarte apropiată de 1). În acest caz se va identifica șablonul $h[n]$ în semnalul $x_zg[n]$, ca în *Figura 6*.

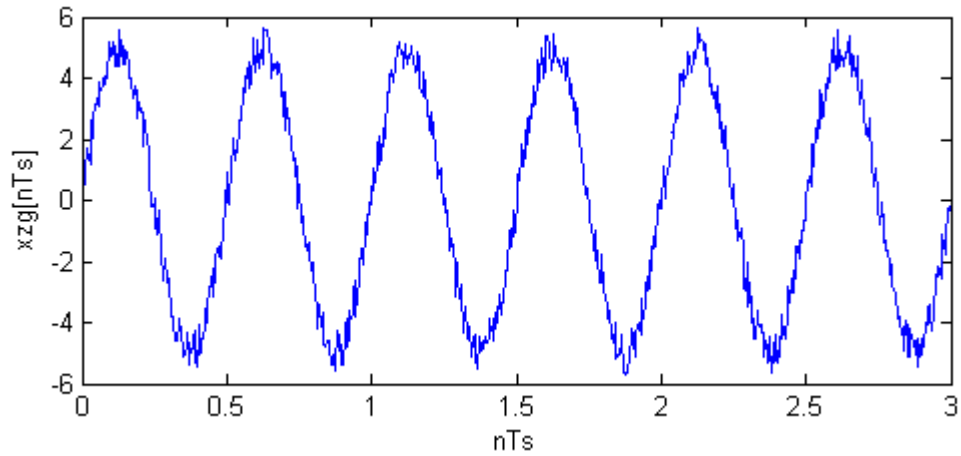


Figura 5. Semnalul $x_{zg}[n]$

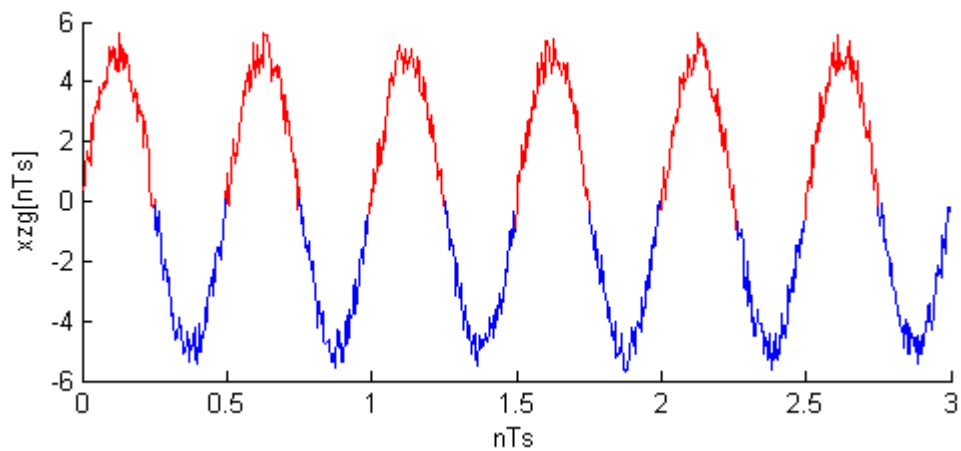


Figura 6. Identificarea șablonului $h[n]$ în semnalul $x_{zg}[n]$

Observație: chiar dacă semnalul este afectat de zgomot, șablonul poate fi identificat fără probleme.

Exemplu 3. Fie un semnal ECG (electrocardiogramă) în care se dorește identificarea bătăilor inimii. Se salvează în șablon o bătaie a inimii și apoi folosind corelația normată se identifică și celelalte bătăi ale inimii.

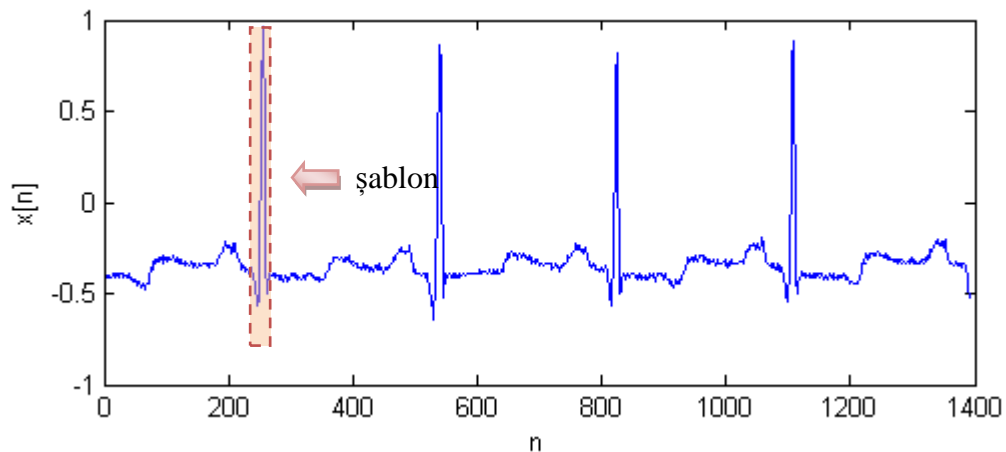


Figura 7. Semnal ECG și șablon

Folosind un prag al corelației normale de 0.98, se identifică bătăile inimii ca în *Figura 8*.

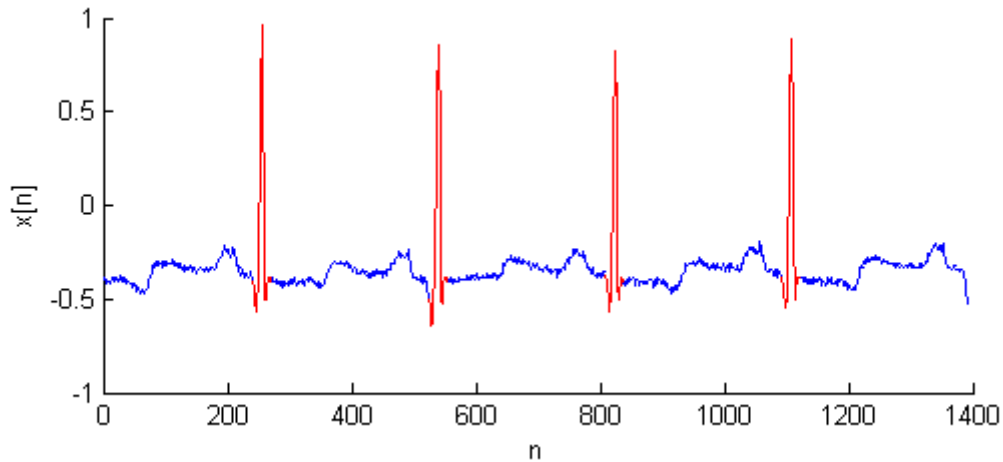



Figura 8. Identificarea bătăilor inimii într-un semnal ECG

Exemplu 4. Fie o imagine ce conține siglele mai multor companii iar cerința constă în identificarea siglei unei anumite companii. În acest caz șablonul este sigla iar semnalul este imaginea. Deoarece semnalul nu mai este unul unidimensional (ca în exemplele anterioare) nu se mai poate folosi *formula 2*. Cel mai simplu este ca imaginea să fie transformată într-o imagine grayscale (în cazul în care era color) și apoi să se folosească funcția `normxcorr2` a Matlabului (această funcție face corelație bidimensională normalată).



Figura 9. Imagine cu diverse sigle

Se alege ca șablon sigla unei companii (de exemplu Lufthansa ). În urma aplicării corelației normale bidimensionale, se identifică unde apare sigla companiei Lufthansa în imagine, precum în *Figura 10*.

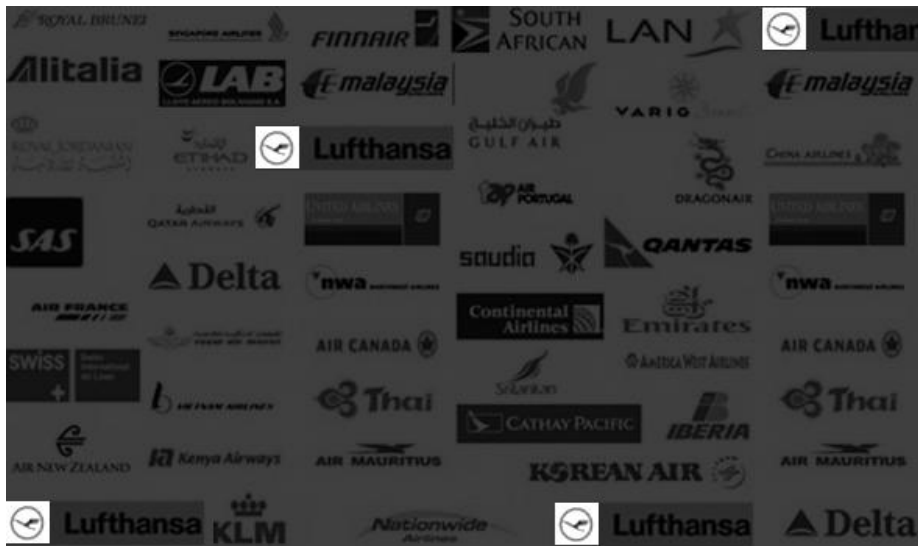


Figura 10. Identificarea siglei șablon în imagine

Aplicații în Matlab

1. Să se implementeze în Matlab o funcție denumită *corelație*, care să aibă ca parametri de intrare două semnale $x[n]$ și $h[n]$ (vectori linie) iar ca parametri de ieșire corelația ($y[n]$) și corelația normată ($y_{norm}[n]$) a semnalelor primite la intrare.

```
function [y, y_norm] = corelatie(x, h)
M = length(h);
% se foloseste conventia zero-padding: se completeaza semnalul x[n] cu M - 1
% zerouri spre stanga
x_zp = [zeros(1,M-1), x];
N = length(x_zp);
for n = M : N
    y(n-M+1) = 0;
    for k = 1 : M
        % se calculeaza corelatia
        y(n-M+1) = y(n-M+1) + h(k) * x_zp(n+k-M);
    end
    % se calculeaza corelatia normata
    y_norm(n-M+1) = y(n-M+1) / (sqrt(sum(h.^2)) * sqrt(sum(x_zp(n-M+1:n).^2)));
end
```

2. Să se genereze și să se afișeze sinusoida $x(t) = 5\sin(4\pi t)$ eșantionată cu $F_s = 244\text{Hz}$ și durata de 3 secunde.

```
F = 2;
Fs = 244;
durata = 3;
A = 5;
t = 0:1/Fs:durata-1/Fs;
x = A*sin(2*pi*F*t);
figure(1)
plot(t,x), xlabel('nTs'), ylabel('x[n]')
```

3. Să se salveze în vectorul șablon $h[n]$ jumătate din prima perioadă a semnalului $x[n]$ generat la punctul anterior. Să se calculeze și să se reprezinte grafic corelația și corelația normalată dintre $h[n]$ și $x[n]$ folosind funcția *corelație* de la punctul 1.

```
% se salveaza in sablon jumatate din prima perioada a semnalului x[n]
h = x(1:Fs/(2*F));
M = length(h);
[y, y_norm] = corelatie(x,h);
figure(2)
plot(t,y), xlabel('nTs'), ylabel('y[n]')
figure(3)
plot(t,y_norm), xlabel('nTs'), ylabel('y_norm[n]')
```

4. Să se reprezinte grafic de fiecare dată când șablonul $h[n]$ este identificat în semnalul $x[n]$. *Observație!* Trebuie mai întâi să identificăm la ce momente de timp s-a obținut corelație normalată maximă (teoretic trebuie să fie corelație egală cu 1, deoarece șablonul se potrivește perfect cu prima jumătate a oricărei perioade; practic însă nu este exact 1, deoarece intervin rotunjirile de calcul ale Matlabului, de exemplu $\sin(\pi)$ nu este zero ci $1.2246 \cdot 10^{-16}$). Pentru a găsi indicii elementelor dintr-un vector care respectă anumite criterii, se poate folosi funcția *find*.

```
poz = find(y_norm>=0.9999);
figure(4)
hold on
plot(t,x)
for i = 1:length(poz)
    plot(t(poz(i)-M+1:poz(i)), x(poz(i)-M+1:poz(i)), 'r')
end
hold off
xlabel('nTs'), ylabel('x[nTs]')
```

5. Semnalului $x[n]$ anterior să i se adauge zgomot uniform în intervalul $[-0.7, 0.7]$. Apoi să se repete pașii 2, 3 și 4. (*Atenție!* Șablonul să nu conțină zgomot).

```
zg = -0.7+1.4*rand(1,length(x));
x = x + zg;
```

Care va fi pragul pentru corelația normalată, astfel încât să se identifice de fiecare dată când șablonul $h[n]$ apare în semnalul $x[n]$?

6. Să se genereze un semnal $x[n]$ format din suma a 3 sinusoidă având amplitudine unitară, fază inițială nulă și frecvențele $F_1 = 10\text{Hz}$, $F_2 = 20\text{Hz}$, $F_3 = 40\text{Hz}$. Se alege frecvența de eșantionare astfel încât să se respecte teorema eșantionării. Se salvează într-un șablon $h[n]$ jumătate dintr-o perioadă a semnalului $x[n]$. Să se detecteze și să se reprezinte grafic de fiecare dată când șablonul $h[n]$ este găsit în semnalul $x[n]$.

7. Să se citească imaginea *sigle.jpg*. Să se selecteze o siglă și să se caute în toată imaginea folosind corelația. (*Observație:* În următorul cod Matlab, imaginea *sigla.jpg* s-a obținut prin selecția din imaginea *sigle.jpg* a siglei companiei Lufthansa)

```
imag_color = imread('imagine','jpg');
imag = rgb2gray(imag_color);
imag = double(imag)/255;
logo_color = imread('sigla','jpg');
logo = rgb2gray(logo_color);
logo = double(logo)/255;
[logo_rows, logo_cols] = size(logo);
```



```

corelatie = normxcorr2(logo, imag);
[poz_logo_rows, poz_logo_cols] = find(corelatie > 0.99);
imag_new = imag;
for i = 1:length(poz_logo_rows)
    imag_new(poz_logo_rows(i)-logo_rows:poz_logo_rows(i),poz_logo_cols(i)-
    logo_cols:poz_logo_cols(i))=...
        imag(poz_logo_rows(i)-logo_rows:poz_logo_rows(i),poz_logo_cols(i)-
    logo_cols:poz_logo_cols(i))*6;
end
imag_new = imag_new/4;
figure(1), imshow(logo_color)
figure(2), imshow(imag_color)
figure(3), imshow(imag_new)

```

8. Să se încarce semnalul EKG cu denumirea *EKG.mat*.

- Să se reprezinte grafic semnalul EKG
- Să se salveze într-un șablon o bătaie a inimii
- Să se identifice toate bățile inimii din semnalul EKG

Lucrarea 9

Convoluția

Obiective: înțelegerea conceptului de convoluție, realizarea convoluției pas cu pas pentru un semnal didactic; utilizarea convoluției pentru filtrarea unei sinusoidă afectate de zgomot; utilizarea convoluției pentru filtrarea unui semnal obținut din suma a două sinusoidă.

Filtrul numeric reprezintă un algoritm/formulă, care pornind de la un șir de numere numit *semnal numeric de intrare* produce un alt șir de numere numit *semnal numeric de ieșire*.

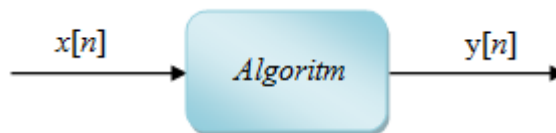


Figura 1. Filtrarea semnalului digital $x[n]$

Filtrele numerice pot fi împărțite în două clase:

- *filtre cu Răspuns Finit la Impuls* (RFI) sau FIR (**F**inite **I**mpulse **R**esponse).
Exemplu: $y[n] = 3 \cdot x[n] - x[n - 2]$
- *filtre cu Răspuns Infinit la Impuls* (RII) sau IIR (**I**nfinite **I**mpulse **R**esponse).
Exemplu: $y[n] = 3 \cdot x[n] - x[n - 2] - y[n - 1]$

Filtrarea în timp se realizează folosind **convoluția**. Pentru filtrele FIR, convoluția dintre două semnale discrete $x[n]$ și $h[n]$ este descrisă de relația:

$$y[n] = \sum_{k=0}^{M-1} h[k] \cdot x[n - k] = h[n] * x[n] \quad (1)$$

unde $h[n]$ reprezintă coeficienții filtrului iar M reprezintă numărul de eșantioane ale lui $h[n]$. Convoluția mai poartă denumirea și de *produs de convoluție*.

Exemplu 1. Fie semnalele $x = [3, 3, 3, 6, 6, 6, 9, 9, 9]$ și $h = [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$. Să se calculeze produsul de convoluție $y[n] = h[n] * x[n]$.

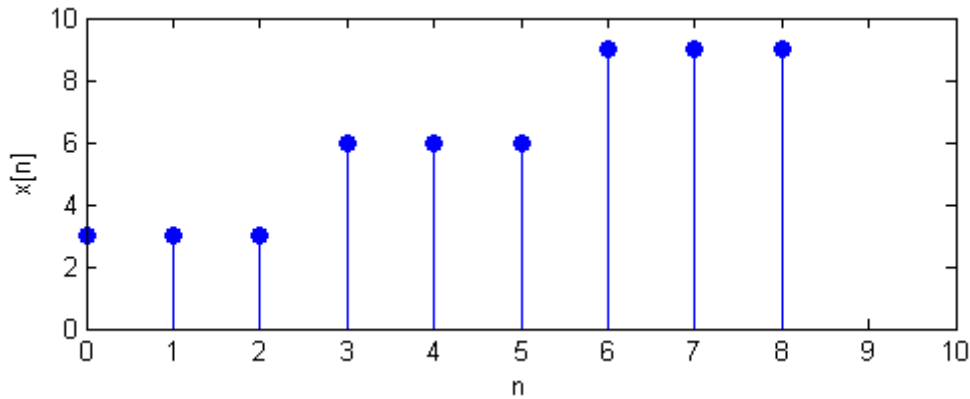


Figura 2. Semnalul $x[n]$

Deoarece semnalul $h[n]$ are 3 valori, rezultă că $M = 3$. Pentru început se va extinde semnalul $x[n]$ cu $M - 1$ eşantioane nule către stânga și $M - 1$ eşantioane nule către dreapta.

Valorile semnalului obținut în urma convoluției sunt:

- Pentru $n = 0 \rightarrow y[0] = h[0] \cdot x[0] + h[1] \cdot x[-1] + h[2] \cdot x[-2]$.
Rezultă $y[0] = \frac{1}{3} \cdot 3 + \frac{1}{3} \cdot 0 + \frac{1}{3} \cdot 0 = 1$
- Pentru $n = 1 \rightarrow y[1] = h[0] \cdot x[1] + h[1] \cdot x[0] + h[2] \cdot x[-1]$
 $y[1] = \frac{1}{3} \cdot 3 + \frac{1}{3} \cdot 3 + \frac{1}{3} \cdot 0 = 2$
- ...
- Pentru $n = 8 \rightarrow y[8] = h[0] \cdot x[8] + h[1] \cdot x[7] + h[2] \cdot x[6]$
 $y[8] = \frac{1}{3} \cdot 9 + \frac{1}{3} \cdot 9 + \frac{1}{3} \cdot 9 = 9$
- Pentru $n = 9 \rightarrow y[9] = h[0] \cdot x[9] + h[1] \cdot x[8] + h[2] \cdot x[7]$
 $y[9] = \frac{1}{3} \cdot 0 + \frac{1}{3} \cdot 9 + \frac{1}{3} \cdot 9 = 6$
- Pentru $n = 10 \rightarrow y[10] = h[0] \cdot x[10] + h[1] \cdot x[9] + h[2] \cdot x[8]$
 $y[10] = \frac{1}{3} \cdot 0 + \frac{1}{3} \cdot 0 + \frac{1}{3} \cdot 9 = 3$

Se obține astfel semnalul $y = [1, 2, 3, 4, 5, 6, 7, 8, 9, 6, 3]$.

Observații:

- pentru a putea efectua convoluția, semnalul $x[n]$ trebuie completat la extremități (atât la dreapta cât și la stânga) cu câte $M - 1$ zerouri (tehnică cunoscută sub denumirea de *zero-padding*). Dacă semnalul $h[n]$ are M eşantioane și semnalul $x[n]$ are N eşantioane (înainte de a folosi *zero-padding*), atunci semnalul $y[n]$ obținut în urma convoluției va avea $N + M - 1$ eşantioane.
- primele $M - 1$ eşantioane și ultimele $M - 1$ eşantioane obținute în urma convoluției vor fi imprecise, datorită eşantioanelor introduse prin *zero-padding*.

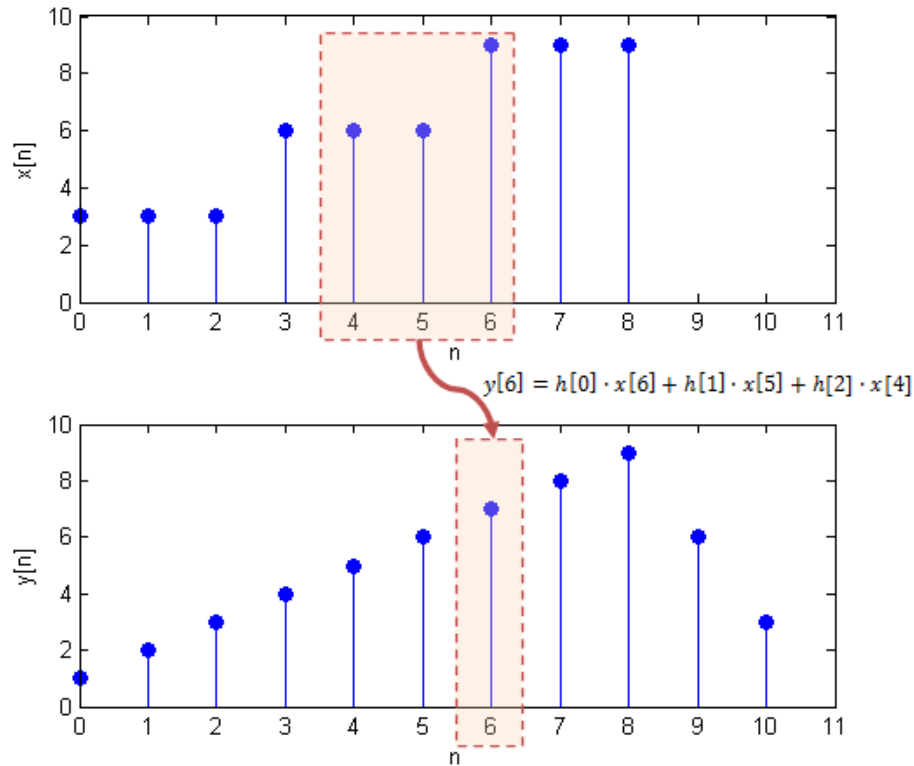


Figura 3. Semnalul $y[n]$ obținut în urma convoluției semnalelor $x[n]$ și $h[n]$

Exemplu 2. Fie o sinusoidă $x[n]$ peste care s-a adăugat zgomot uniform. Să se filtreze semnalul folosind următorul set de coeficienți $h = \left[\frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5} \right]$.

Aplicând formula 1 se obține:

$$y[n] = \frac{1}{5} \cdot x[n] + \frac{1}{5} \cdot x[n - 1] + \frac{1}{5} \cdot x[n - 2] + \frac{1}{5} \cdot x[n - 3] + \frac{1}{5} \cdot x[n - 4],$$

adică media ultimilor 5 eșantioane din semnalul $x[n]$. Dar așa cum am menționat în *Lucrarea 6*, medierea se comportă ca un *filtru trece jos* în urma căruia frecvențele înalte (zgomotul) vor fi atenuate, după cum se observă și în *Figura 4*.

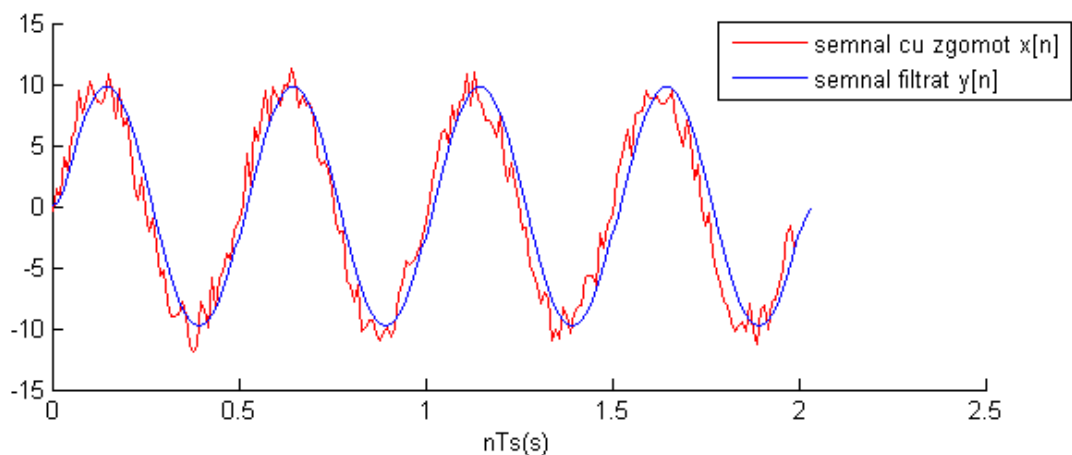


Figura 4. Filtrarea semnalului sinusoidal cu zgomot

Exemplu 3. Fie un semnal $x[n]$ obținut din suma a două sinusoides unitare cu faza inițială nulă, de frecvențe $F_1 = 2\text{Hz}$ și $F_2 = 50\text{Hz}$ (*Figura 5.a*). Să se filtreze semnalul $x[n]$ folosind următorul set de coeficienți $h = [1, -1]$.

Aplicând *formula 1* se obține: $y[n] = x[n] - x[n - 1]$, adică formula diferențierii amplificată cu T_s . Dar așa cum am menționat în *Lucrarea 7*, diferențierea se comportă ca un *filtru trece sus*. Ne așteptăm astfel ca în urma convoluției să se obțină un semnal care să conțină sinusoida de frecvență înaltă, sinusoida de frecvență joasă fiind mult atenuată, așa cum se poate observa și în *Figura 5.b*.

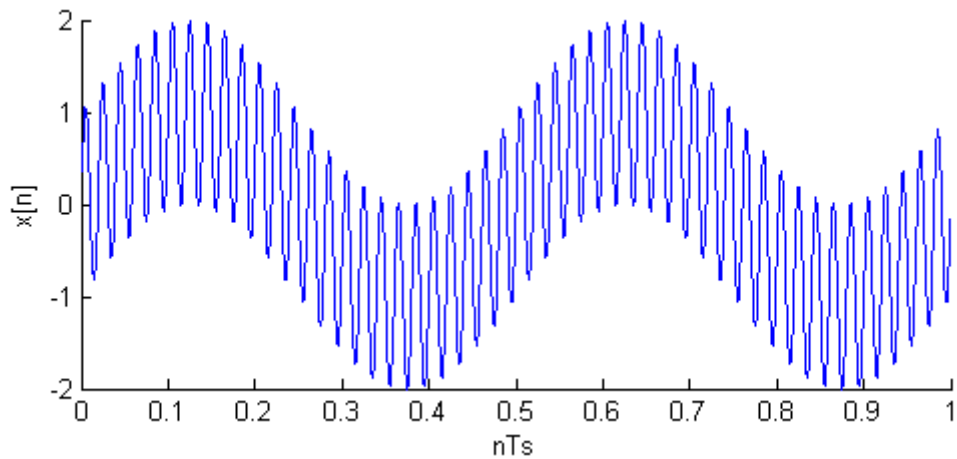


Figura 5.a. Semnalul $x[n]$ obținut din suma a două sinusoida de frecvențe $F_1 = 2\text{Hz}$ și $F_2 = 50\text{Hz}$

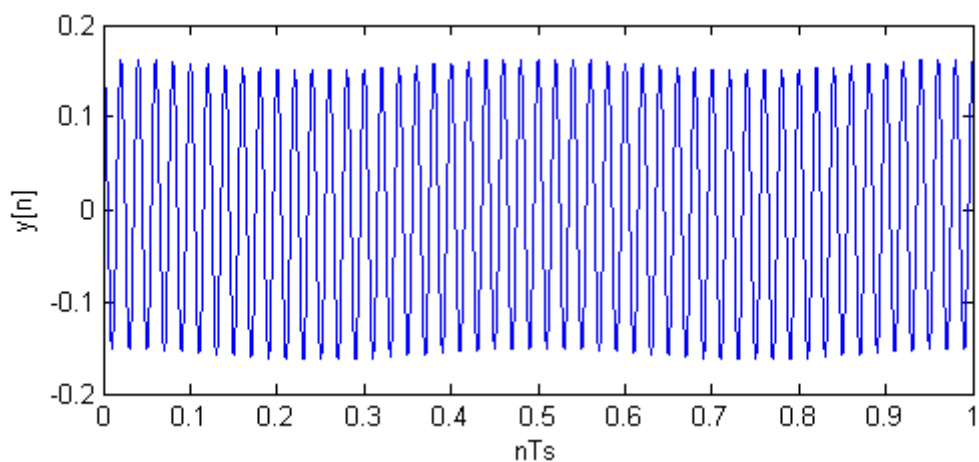


Figura 5.b. Semnalul $y[n]$ obținut în urma convoluției dintre $x[n]$ și $h = [-1, 1]$

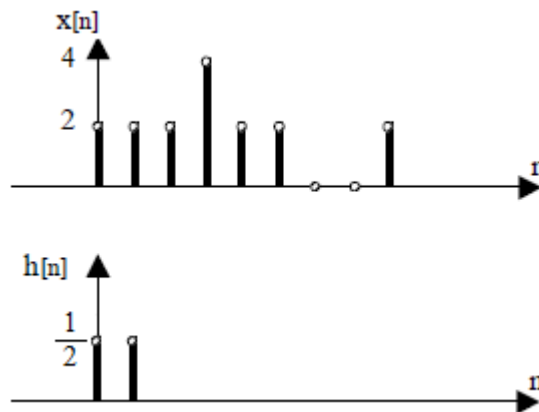
Aplicații în Matlab

1. Să se implementeze în Matlab o funcție denumită *convoluție*, care să primească ca parametri de intrare două semnale (vectori unidimensionali) și să aibă ca parametru de ieșire produsul de convoluție a celor două semnale.

```
function y = convolutie(x, h)
M = length(h);
x = [zeros(1, M-1), x, zeros(1, M-1)];
N = length(x);
for n = M:length(x)
    y(n-M+1) = 0;
    for k = 1:M
        y(n-M+1) = y(n-M+1) + h(k)*x(n-k+1);
    end
end
```

2. Folosind funcția de convoluție implementată anterior, să se calculeze produsul de convoluție dintre semnalele $x = [3, 3, 3, 6, 6, 6, 9, 9, 9]$ și $h = [1/3, 1/3, 1/3]$. Câte elemente va avea semnalul obținut în urma convoluției?

3. Fie semnalele $x[n]$ și $h[n]$ de mai jos.



Cerințe:

- Să se calculeze semnalul $y[n]$, ca produs de convoluție între $x[n]$ și $h[n]$.
- Să se calculeze semnalul $y[n]$, ca produs de convoluție între $h[n]$ și $x[n]$. Ce se observă?
- Să se reprezinte grafic semnalul $y[n]$.

4. Fie un semnal sinusoidal $x(t) = 10 \cdot \sin(4\pi t)$ eșantionat cu $F_s = 200\text{Hz}$ peste care se adaugă zgomot uniform în intervalul $[-1, +1]$.

- Să se reprezinte grafic sinusoida cu zgomot.

```
Fs = 200;
F = 2;
durata = 1;
A = 10;
t = 0:1/Fs:durata;
x = A*sin(2*pi*F*t);
zg = -1 + 2*rand(1, length(x));
x_zg = x + zg;
figure(1), plot(t, x_zg), xlabel('nTs (s)'), ylabel('x_zg[nTs]')
```

- Să se filtreze sinusoida cu zgomot cunoscând coeficienții filtrului $h = [1/4, 1/4, 1/4, 1/4]$. Să se reprezinte grafic semnalul obținut în urma filtrării. Ce observați?
- Să se filtreze sinusoida cu zgomot cunoscând coeficienții filtrului $h = [1, -1]$. Să se reprezinte grafic semnalul obținut în urma filtrării. Ce observați?

5. Fie un semnal $x[n]$ obținut din suma a două sinusoida având frecvențele $F_1 = 5\text{Hz}$ și $F_2 = 50\text{Hz}$. Ambele sinusoida au amplitudine maximă unitară, fază inițială nulă și au fost eșantionate cu $F_s = 200\text{Hz}$.

- Să se reprezinte grafic semnalul $x[n]$.

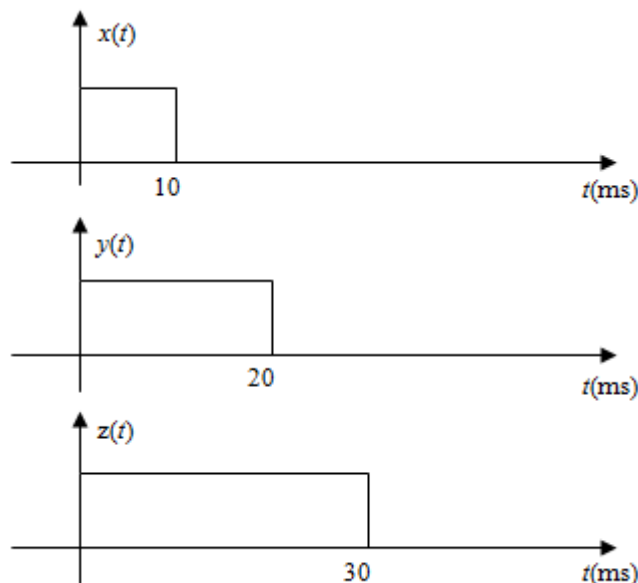
```

Fs = 200;
F1 = 5;
F2 = 50;
durata = 1;
t = 0:1/Fs:durata;
x1 = sin(2*pi*F1*t);
x2 = sin(2*pi*F2*t);
x = x1 + x2;
figure(1), plot(t,x), xlabel('nTs (s)'), ylabel('x[nTs]')

```

- Să se filtreze semnalul $x[n]$, cunoscând coeficienții filtrului $h = [1, -1]$. Să se reprezinte grafic semnalul obținut în urma filtrării. Care este frecvența semnalului rezultat?

6. Fie următoarele semnale:



Știind că toate cele 3 semnale sunt eșantionate cu $F_s = 1\text{kHz}$, să se calculeze:

- $a[n] = (x[n] * y[n]) * z[n]$ și $b[n] = x[n] * (y[n] * z[n])$. Ce observați?
- $c[n] = (x[n] * z[n]) + (y[n] * z[n])$ și $d[n] = (x[n] + y[n]) * z[n]$. Ce observați?

Lucrarea 10

Teorema Fourier

Obiective: înțelegerea Teoremei Fourier; utilizarea descompunerii în serie Fourier a unui semnal dreptunghiular.

Reamintim de ce sunt semnalele sinusoidale atât de importante: *deoarece sunt singurele care se propagă nedistorsionate prin sisteme liniare*. Pot fi atenuate sau amplificate, pot fi defazate, însă tot sinusoidale rămân. Ce se întâmplă însă dacă avem semnale continue și periodice care nu sunt sinusoidale? În acest caz se poate folosi teorema Fourier pentru a descompune semnalul respectiv într-o sumă de sinusoidale.

Teorema Fourier

Orice semnal **continuu** și **periodic** de perioadă T_0 poate fi scris ca o **sumă infinită** de sinusoidale (armonici) plus o componentă continuă.

$$u(t) = U_0 + \sum_{k=1}^{\infty} S_k \sin(2\pi F_k t + \varphi_k) \quad (1)$$

unde:

- U_0 este componenta continuă
- S_k este amplitudinea sinusoidalei de ordin k
- φ_k este faza inițială a sinusoidalei de ordin k
- F_k este frecvența sinusoidalei de ordin k

Frecvența de repetiție a semnalului, numită și **frecvență fundamentală** este $F_0 = \frac{1}{T_0}$.

Atenție! F_k , **frecvența sinusoidalei de ordin k** , este **multiplu întreg al frecvenței fundamentale**: $F_k = kF_0$.

Deoarece frecvența sinusoidelor este multiplu întreg al frecvenței fundamentale, aceste sinusoidale se mai numesc și **armonici**. Descompunerea lui $u(t)$ în serie de armonici este așadar:

$$u(t) = U_0 + \underbrace{S_1 \sin(2\pi F_0 t + \varphi_1)}_{\substack{\text{armonica de ordin } 1 \\ \text{(fundamentală)}}} + \underbrace{S_2 \sin(2\pi(2F_0)t + \varphi_2)}_{\substack{\text{armonica} \\ \text{de ordin } 2}} + \dots + \underbrace{S_n \sin(2\pi(nF_0)t + \varphi_n)}_{\substack{\text{armonica} \\ \text{de ordin } n}} + \dots$$

În practică nu vom putea lucra cu o infinitate de armonici, de aceea vom păstra un număr finit de armonici a căror sumă să se apropie cât mai fidel de semnalul original.

Pentru a calcula U_0 , S_k și φ_k se utilizează formulele:

$$U_0 = \frac{1}{T} \int_0^T u(t) dt \quad (2)$$

$$S_k = \sqrt{A_k^2 + B_k^2} \quad (3)$$

$$\varphi_k = \arctg \frac{B_k}{A_k} \quad (4)$$

unde:

$$A_k = \frac{2}{T} \int_0^T u(t) \sin(2\pi k F_0 t) dt, B_k = \frac{2}{T} \int_0^T u(t) \cos(2\pi k F_0 t) dt \quad (5)$$

Spectrul de frecvență al unui semnal

Se numește spectru de frecvență al unui semnal $u(t)$, mulțimea tuturor sinusoidelor în care poate fi descompus semnalul. Spectrul de frecvență poate fi generat folosind *Teorema Fourier* (pentru semnale periodice) sau *Transformata Fourier* (pentru semnale neperiodice), iar informația este de obicei prezentată în două grafice: *spectru de amplitudine* și *spectru de fază*.

Spectrul de amplitudine arată dependența dintre modulul amplitudinilor armonicilor și frecvența acestora. Amplitudinile armonicilor descresc spre zero când frecvențele tind către infinit.

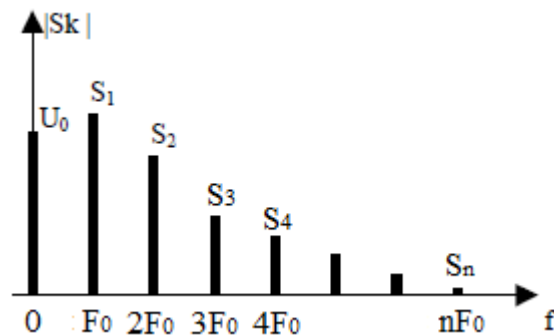


Figura 1. Spectru de amplitudine (generic)

Pentru un semnal continuu și periodic spectrul este discret, frecvențele armonicilor având doar valori multiplu întreg a frecvenței fundamentale.

Spectrul de fază arată dependența dintre faza inițială a armonicilor și frecvența acestora.

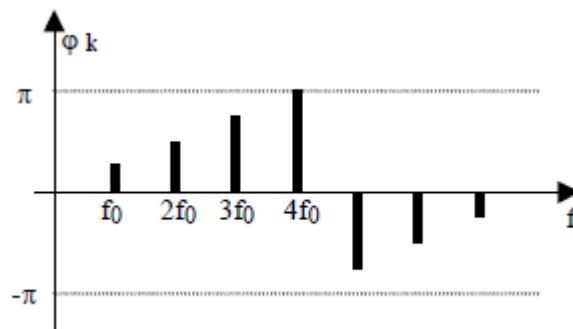


Figura 2. Spectru de fază (generic)

Exemplu. Descompunerea în serie Fourier a unui semnal dreptunghiular.

Fie următorul semnal continuu și periodic cu $T_0 = 0.5s$.

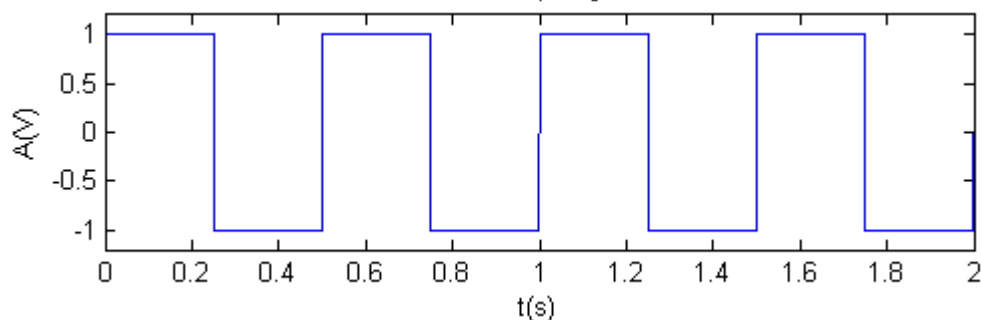


Figura 3. Semnal dreptunghiular

Deoarece semnalul are în fiecare perioadă valorile 1V și -1V pentru intervale egale de timp, media va fi de 0V. Așadar componenta continuă U_0 va fi nulă (pentru verificare se va folosi formula 2).

Aplicând formula 5 și formula 3, se determină amplitudinile sinusoidelor de ordin k :

$$S_k = \begin{cases} \frac{4}{k\pi}, & \text{pentru } k \text{ impar} \\ 0, & \text{pentru } k \text{ par} \end{cases}$$

Aplicând formula 4, se determină fazele inițiale ale sinusoidelor. Se obține astfel $\varphi_k = 0$.

Astfel, dezvoltarea în serie Fourier a semnalului dreptunghiular $u(t)$ este:

$$u(t) = \frac{4}{\pi} \sin(2\pi F_0 t) + 0 + \frac{4}{3\pi} \sin(2\pi(3F_0)t) + 0 + \frac{4}{5\pi} \sin(2\pi(5F_0)t) + 0 + \dots$$

Armonica fundamentală are frecvența $F_0 = 2\text{Hz}$, amplitudinea $\frac{4}{\pi}$ și fază inițială nulă.

Reprezentarea armonicilor fundamentale peste semnalul original arăta ca în Figura 4.

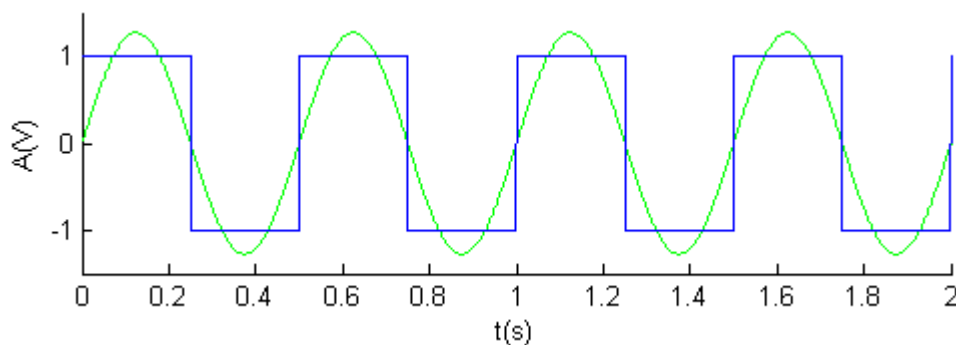


Figura 4. Armonica fundamentală (cu verde)

Dacă se păstrează primele 5 armonici, semnalul rezultat va fi cel din Figura 5.

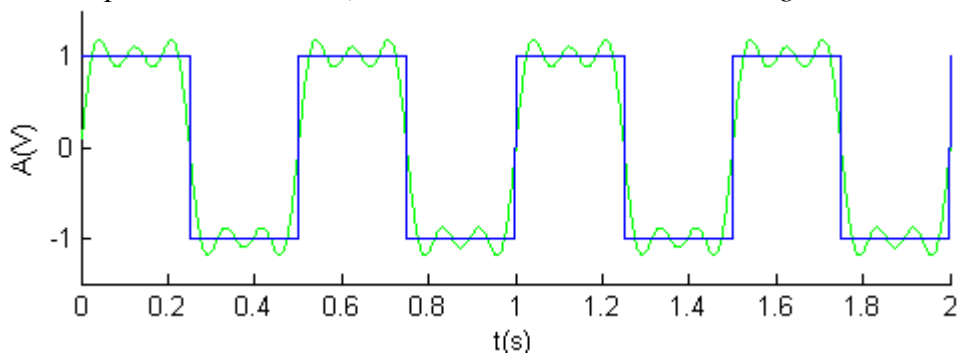


Figura 5. Suma primelor 5 armonici (cu verde)

Dacă se păstrează primele 50 de armonici, semnalul rezultat va fi cel din *Figura 6*.

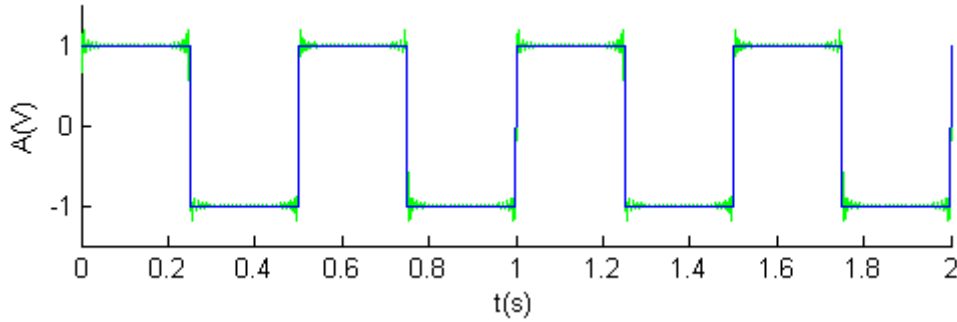


Figura 6. Suma primelor 50 de armonici (cu verde)

Se observă că semnalul rezultat se apropie tot mai mult de semnalul real (original) cu cât creștem numărul armonicilor însumate.

Dacă se păstrează doar primele 10 armonici, spectrul de amplitudine al semnalului obținut este următorul.

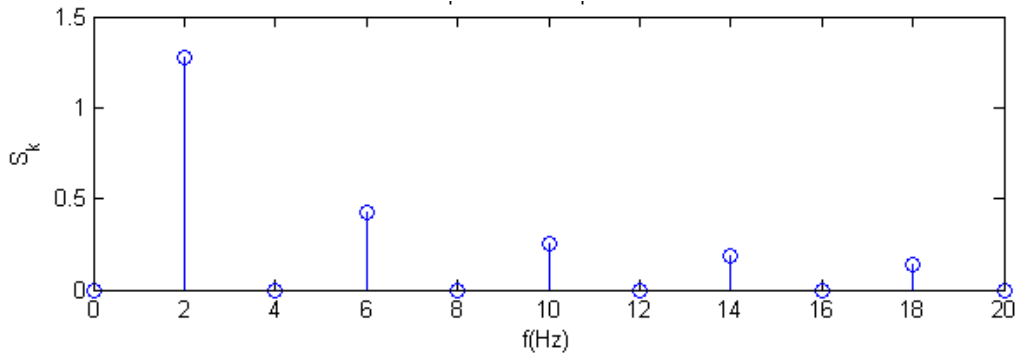


Figura 7. Spectrul de amplitudine al semnalului obținut din primele 10 armonici

Observații:

- componenta continuă ($F = 0\text{Hz}$) este nulă
- armonica fundamentală ($F_1 = F_0 = 2\text{Hz}$) are amplitudinea $S_1 = \frac{4}{\pi}$
- cea de-a doua armonică ($F_2 = 2F_0 = 4\text{Hz}$) are amplitudinea $S_2 = 0$
- cea de-a treia armonică ($F_3 = 3F_0 = 6\text{Hz}$) are amplitudinea $S_3 = \frac{4}{3\pi}$

Dacă semnalul dreptunghiular $u(t)$ având $F_0 = 2\text{Hz}$ ar fi trecut printr-un FTJ ideal cu frecvența de tăiere de 7Hz , atunci semnalul filtrat va conține doar componenta continuă și armonicile de ordin 1, 2 și 3. Armonicile de ordin $k > 3$ au frecvența mai mare de 7Hz și nu vor fi lăsate să treacă. Deci la ieșirea filtrului s-ar obține semnalul:

$$u_{\text{filtrat}}(t) = \frac{4}{\pi} \sin(2\pi F_0 t) + 0 + \frac{4}{3\pi} \sin(2\pi(3F_0)t).$$

1. Să se genereze în Matlab și să se reprezinte grafic un semnal dreptunghiular de perioadă $T_0 = 0.5\text{s}$ și durată = 2s ca cel din *Figura 3*.

Indiciu: pentru a genera un semnal dreptunghiular se poate folosi funcția `square`.

```
T = 0.5; % perioada semnalului dreptunghiular
Ts = 0.001; % perioada de esantionare
Durata = 2; % durata semnalului dreptunghiular
F = 1/T; % frecventa semnalului dreptunghiular
t = 0:Ts:Durata;
s_drept = square(2*pi*F*t); % semnalul dreptunghiular
figure(1), plot(t, s_drept), axis([0, Durata, -1.2, 1.2])
xlabel('t(s)'), ylabel('A(V)')
```

2. Să se reprezinte grafic în aceeași figură și în același sistem de coordonate atât semnalul dreptunghiular cât și armonica fundamentală.

```
armonica_fundamentala = 4/pi*sin(2*pi*F*t);
figure(2)
hold on
    plot(t, armonica_fundamentala, 'g')
    plot(t, s_drept)
hold off
```

3. Să se reprezinte grafic în aceeași figură dar în sisteme de coordonate diferite, primele 5 armonici. Să se reprezinte în același sistem de coordonate atât semnalul dreptunghiular cât și cel obținut din suma celor 5 armonici.

```
u1 = 4/pi*sin(2*pi*F*t); % armonica de ordin 1
u2 = 0*sin(2*pi*2*F*t); % armonica de ordin 2
u3 = 4/(3*pi)*sin(2*pi*3*F*t); % armonica de ordin 3
u4 = 0*sin(2*pi*4*F*t); % armonica de ordin 4
u5 = 4/(5*pi)*sin(2*pi*5*F*t); % armonica de ordin 5
u = u1 + u2 + u3 + u4 + u5; % suma primelor 5 armonici
figure(3)
subplot(6,1,1), plot(t,u1), title('u1'), xlabel('t(s)'), ylabel('A(V)')
subplot(6,1,2), plot(t,u2), title('u2'), xlabel('t(s)'), ylabel('A(V)')
subplot(6,1,3), plot(t,u3), title('u3'), xlabel('t(s)'), ylabel('A(V)')
subplot(6,1,4), plot(t,u4), title('u4'), xlabel('t(s)'), ylabel('A(V)')
subplot(6,1,5), plot(t,u5), title('u5'), xlabel('t(s)'), ylabel('A(V)')
subplot(6,1,6), title('u=u1+u3+u5')
hold on
    plot(t,u, 'g')
    plot(t, s_drept)
hold off
xlabel('t(s)'), ylabel('A(V)')
```

4. Să se reprezinte grafic în aceeași figură, în același sistem de coordonate, atât semnalul dreptunghiular cât și cel obținut din suma primelor 100 de armonici.

```
nr_armonici = 100;
u = 0;
for k = 1 : 2 : nr_armonici
    u_k = 4/(k*pi)*sin(2*pi*k*F*t);
    u = u + u_k;
end
figure(4), hold on
    plot(t,u, 'g'), axis([0, Durata, -1.5, 1.5])
    plot(t, s_drept)
hold off
xlabel('t(s)'), ylabel('A(V)')
```

5. Să se reprezinte spectrul de amplitudine al semnalului obținut din suma primelor 100 de armonici. *Indiciu:* pentru a testa dacă un număr este par se poate folosi funcția `rem`.

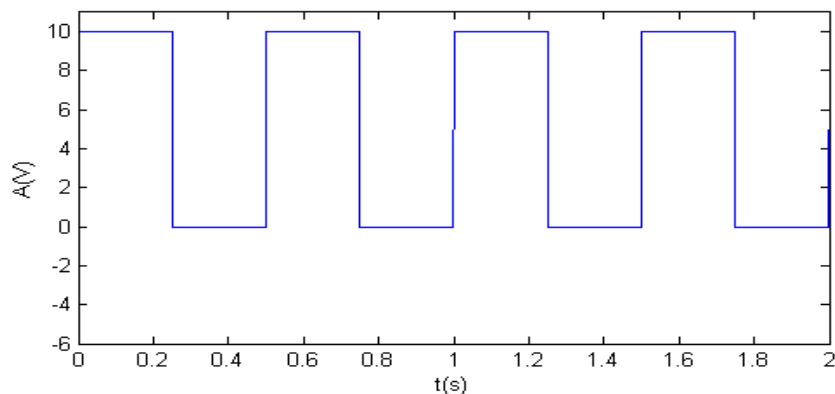
```
for k = 1 : nr_armonici
    f(k) = k*F;
    if(rem(k,2)==0)
        A(k) = 0;
    else
        A(k) = 4/(k*pi);
    end
end
end
f=[0,f]; % se adauga frecventa F = 0Hz
A=[0,A]; % se adauga valoarea componentei continue
figure(5), stem(f, A), title('Spectru de amplitudini'), xlabel('f(Hz)')
```

6. Să se realizeze o interfață grafică (GUI) cu următoarele elemente:

- un *Slider* cu ajutorul căruia să se poată modifica numărul de armonici păstrate (între 0 și 100).
- grafic pentru suma armonicilor păstrate.
- grafic pentru spectrul semnalului obținut din suma armonicilor păstrate.

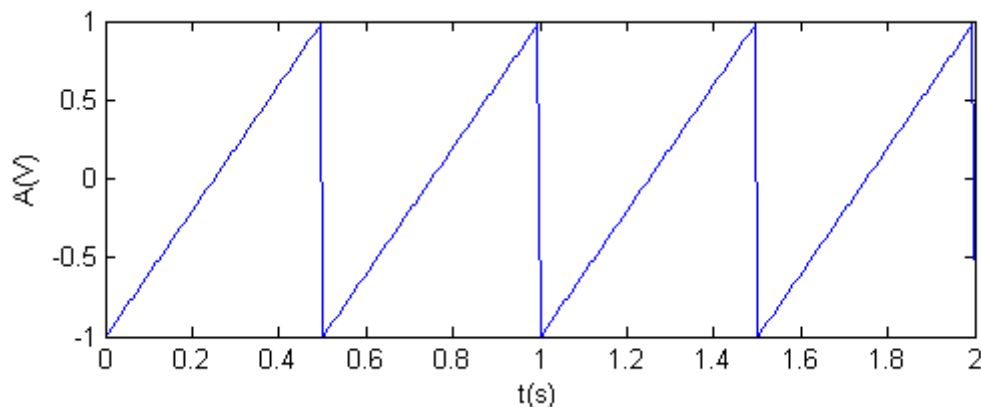
7. Dacă semnalul dreptunghiular de perioadă $T_0 = 0.5s$ este adus la intrarea unui FTJ ideal, cu frecvența de tăiere de 201Hz, cum va arăta semnalul filtrat?

8. Repetați cerințele punctelor 1, 2, 3, 4 și 5 pentru următorul semnal dreptunghiular.



9. Repetați cerințele punctelor 1, 2, 3, 4 și 5 înlocuind semnalul dreptunghiular cu următorul semnal.

Indiciu. Pentru a genera un semnal triunghiular se poate folosi funcția `sawtooth`.



Lucrarea 11

Transformata Fourier Discretă (TFD)

Obiective: calculul Transformatei Fourier Discrete a unui semnal; determinarea spectrului unui semnal folosind TFD; determinarea caracteristicii de frecvență a filtrului folosind TFD.

Orice semnal este format din sinusoid. Mulțimea sinusoidelor din care este format un semnal se numește **spectru**. Spectrul se determină folosind *Teorema Fourier* (pentru semnale periodice) și *Transformata Fourier* (pentru semnale neperiodice).

Transformata Fourier Discretă (TFD) permite trecerea din domeniu timp în domeniu frecvență. TFD transformă N eșantioane ale unui semnal din domeniul timp, în N valori complexe din domeniul frecvență.

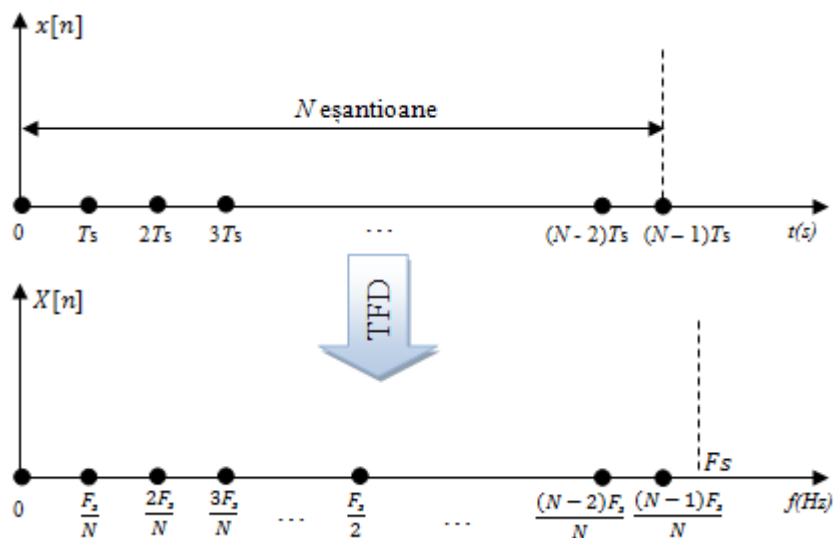


Figura 1. Transformarea domeniului timp în domeniu frecvență folosind TFD

Formula de calcul a *Transformatei Fourier Discrete* este:

$$X[n] = \sum_{k=0}^{N-1} x[k] \cdot e^{-j \frac{2\pi \cdot n \cdot k}{N}}, \text{ pentru } n = 0, 1, 2, \dots, N-1 \quad (1)$$

Exemplu 1. Fie semnalul $x[n] = [1, 0, 2, 1]$. În urma aplicării TFD rezultă:

$$X[0] = x[0] \cdot e^{-j\frac{2\pi \cdot 0 \cdot 0}{4}} + x[1] \cdot e^{-j\frac{2\pi \cdot 0 \cdot 1}{4}} + x[2] \cdot e^{-j\frac{2\pi \cdot 0 \cdot 2}{4}} + x[3] \cdot e^{-j\frac{2\pi \cdot 0 \cdot 3}{4}}$$

$$= x[0] + x[1] + x[2] + x[3] = 4$$

$$X[1] = x[0] \cdot e^{-j\frac{2\pi \cdot 1 \cdot 0}{4}} + x[1] \cdot e^{-j\frac{2\pi \cdot 1 \cdot 1}{4}} + x[2] \cdot e^{-j\frac{2\pi \cdot 1 \cdot 2}{4}} + x[3] \cdot e^{-j\frac{2\pi \cdot 1 \cdot 3}{4}}$$

$$= x[0] + x[1] \cdot e^{-j\frac{\pi}{2}} + x[2] \cdot e^{-j\pi} + x[3] \cdot e^{-j\frac{3\pi}{2}}$$

Folosind formula $e^{jx} = \cos(x) + j \cdot \sin(x)$ se obține:

$$e^{-j\frac{\pi}{2}} = \cos\left(\frac{\pi}{2}\right) - j\sin\left(\frac{\pi}{2}\right) = -j$$

$$e^{-j\pi} = \cos(\pi) - j\sin(\pi) = -1$$

$$e^{-j\frac{3\pi}{2}} = \cos\left(\frac{3\pi}{2}\right) - j\sin\left(\frac{3\pi}{2}\right) = j$$

$$X[1] = 1 + 0 \cdot (-j) + 2 \cdot (-1) + 1 \cdot j = -1 + j$$

În același mod se determină $X[2] = 2$ și $X[3] = -1 - j$

Se observă că în urma aplicării TFD, se obțin valori complexe. Dacă dorim să determinăm spectrul de amplitudine, atunci va trebui să calculăm modulul lui $X[n]$.

Se obține $|X[n]| = [4, \sqrt{2}, 2, \sqrt{2}]$.

Transforma Fourier Discretă furnizează spectrul unui semnal discret. Cele N valori din domeniul frecvență obținute în urma transformatei TFD vor fi multiplicare prin periodicitate, atât în domeniul frecvențelor negative cât și în domeniul frecvențelor pozitive.

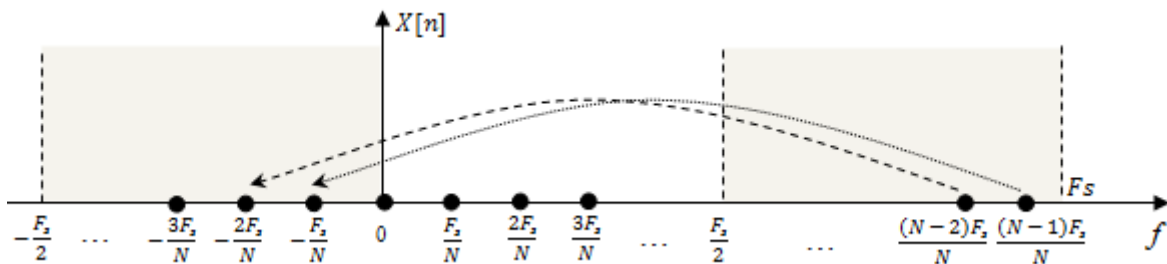


Figura 2. Domeniul frecvențelor în intervalul $-\frac{F_s}{2} \div \frac{F_s}{2}$

Observații:

- TFD a unui semnal furnizează spectrul semnalului.
- TFD a coeficienților filtrului furnizează caracteristica de frecvență a filtrului.

Funcții utile în Matlab pentru a realiza TFD

- pentru a calcula *Transformata Fourier Discretă* a unui semnal $x[n]$, se poate folosi funcția `fft(x)` (FFT este acronim de la *Fast Fourier Transform*)

```
>> x = [1 2 0 1];
X = fft(x)
X =
    4.0000    1.0000 - 1.0000i   -2.0000    1.0000 + 1.0000i
```

- pentru a calcula modulul unui număr complex X în Matlab se poate folosi funcția `abs(X)`

```
>> abs(X)
ans =
    4.0000    1.4142    2.0000    1.4142
```

- pentru a realiza shift-are semnalului obținut în urma TFD, se folosește funcția `fftshift(X)`

```
>> x = [1 0 -2 1 1];
```



```

>> X = fft(x)
X = 1.0000 2.1180+2.7144i -0.1180-2.2654i -0.1180+2.2654i 2.1180-2.7144i
>> fftshift(X)
ans =
-0.1180+2.2654i 2.1180-2.7144i 1.0000 2.1180+2.7144i -0.1180-2.2654i

```

- pentru a genera N puncte echidistante în intervalul $[a, b]$, se folosește funcția

```

linspace(a, b, N)
>> n = linspace(-2, 2, 5)
n =
-2 -1 0 1 2

```

Pe scurt, pentru a reprezenta în Matlab spectrul unui semnal $x[n]$ eșantionat cu frecvența de eșantionare F_s , se folosește sintaxa:

```

axa_fft = linspace(-Fs/2, Fs/2, length(x));
stem(axa_fft, fftshift(abs(fft(x))))

```

Exemplu 2. Fie sinusoida $x(t) = \sin(400\pi t)$ eșantionată cu $F_s = 2000\text{Hz}$. Spectrul acestei sinusoidă este cel din *Figura 3*.

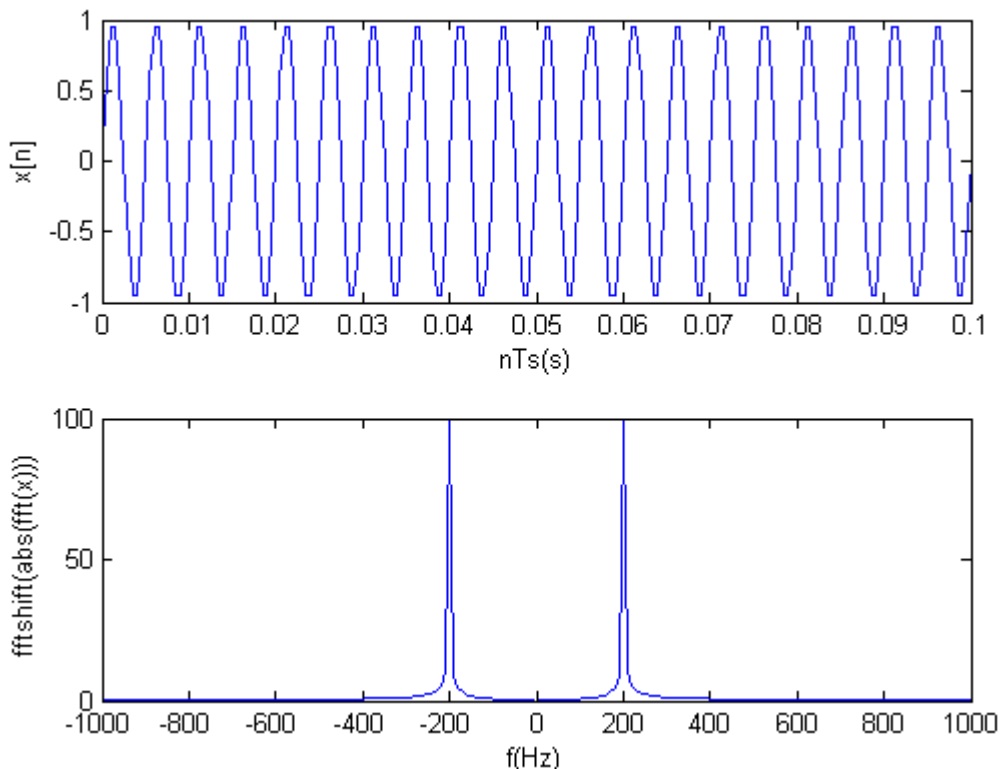


Figura 3. Semnalul $x[n]$ și spectrul semnalului $x[n]$

Exemplu 3. Fie trei sinusoidă $x_1[n]$, $x_2[n]$ și $x_3[n]$ având frecvențele $F_1 = 200\text{Hz}$, $F_2 = 700\text{Hz}$ și $F_3 = 1200\text{Hz}$, amplitudine unitară și fază inițială nulă. Semnalele au fost eșantionate cu $F_s = 10\text{kHz}$ și însumate, obținându-se astfel semnalul $x[n]$ ($x[n] = x_1[n] + x_2[n] + x_3[n]$). Semnalele $x_1[n]$, $x_2[n]$, $x_3[n]$ și $x[n]$ sunt reprezentate în *Figura 4.a* iar spectrele lor în *Figura 4.b*.

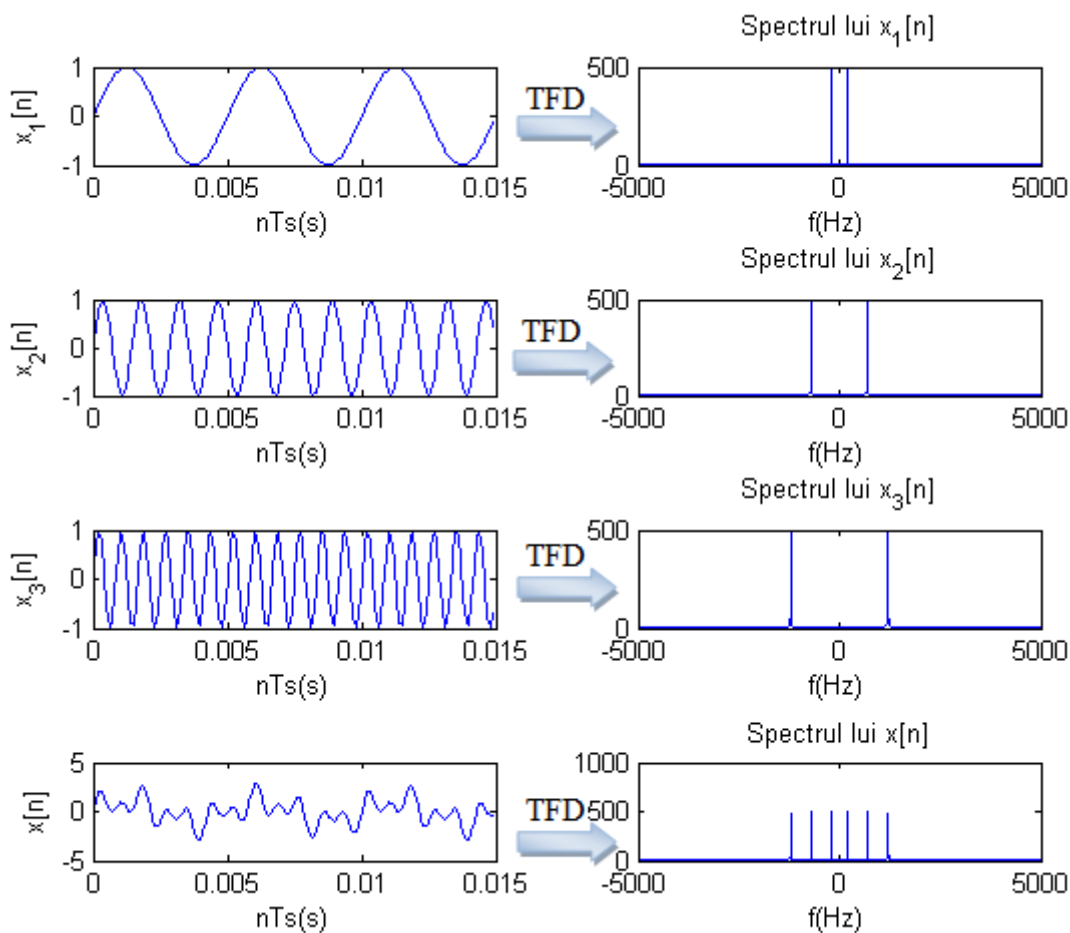


Figura 4.a. Semnalele $x_1[n]$, $x_2[n]$, $x_3[n]$ și $x[n]$
 $(x[n] = x_1[n] + x_2[n] + x_3[n])$

Figura 4.b. Spectrele semnalelor $x_1[n]$, $x_2[n]$, $x_3[n]$ și $x[n]$

Exemplu 4. Fie coeficienții unui filtru obținuți prin eșantionarea cu $Ts = 0.1s$ a unui semnal $\text{sinc}(x)$ de frecvență 3Hz între $t_{min} = -1s$ și $t_{max} = 1s$.

$$h = [0, -0.056, 0.039, 0.045, -0.084, 0, 0.126, -0.104, -0.156, 0.505, 1, 0.505, -0.156, -0.104, 0.126, 0, -0.084, 0.045, 0.039, -0.056, 0]$$

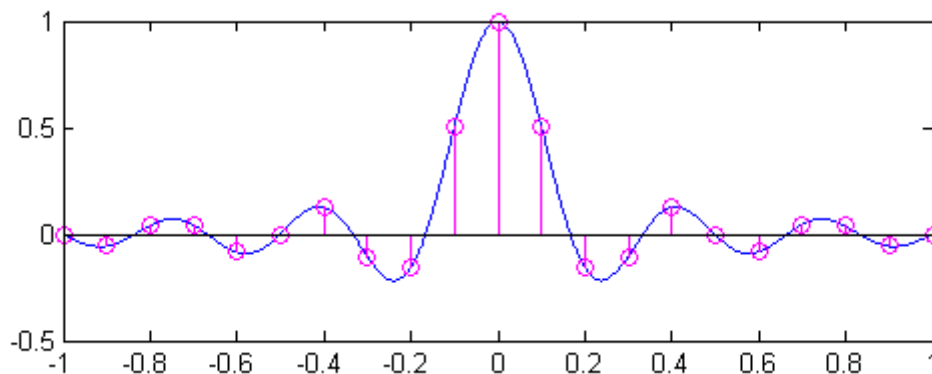


Figura 5. Coeficienții filtrului obținuți prin eșantionarea semnalului $\text{sinc}(x)$

Așa cum am menționat, Transformata Fourier Discretă a coeficienților filtrului furnizează caracteristica de frecvență a filtrului. Modulul transformatei TFD reprezintă caracteristica de amplitudine a filtrului. Pentru coeficienții h de mai sus, caracteristica de amplitudine este cea din Figura 6.

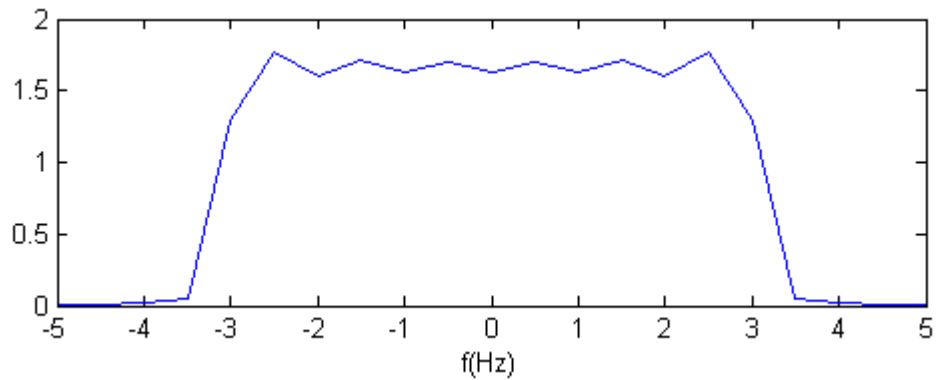


Figura 6. Caracteristica de amplitudine a filtrului definit de coeficienții din Figura 5

Din Figura 6 putem trage concluzia că este vorba despre un Filtru Trece Jos având frecvența de tăiere $F_t = 3\text{Hz}$.

Exemplu 5. Fie un semnal $x[n]$ având următorul spectru de amplitudine:

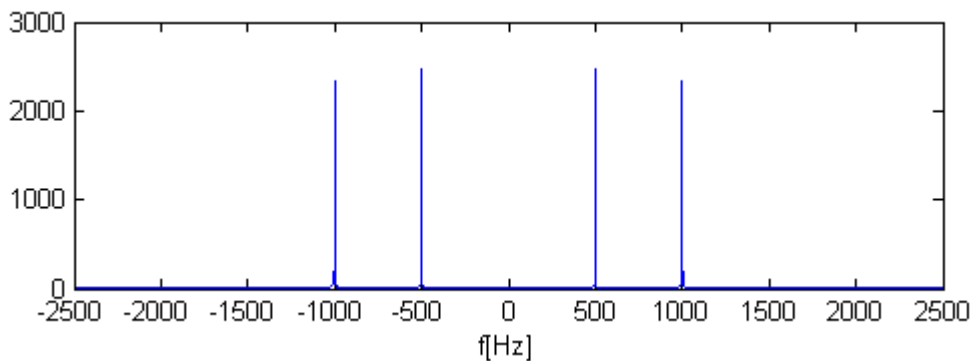


Figura 7. Spectrul semnalului $x[n]$

Analizând spectrul de amplitudine al semnalului putem trage următoarele concluzii:

- semnalul $x[n]$ este format din suma a două sinusoidे având frecvențele $F_1 = 500\text{Hz}$ și $F_2 = 1000\text{Hz}$.
- frecvența de eșantionare este $F_s = 5000\text{Hz}$.

Ce se întâmplă dacă semnalul $x[n]$ este trecut printr-un filtru ideal având următoarea caracteristică de amplitudine?

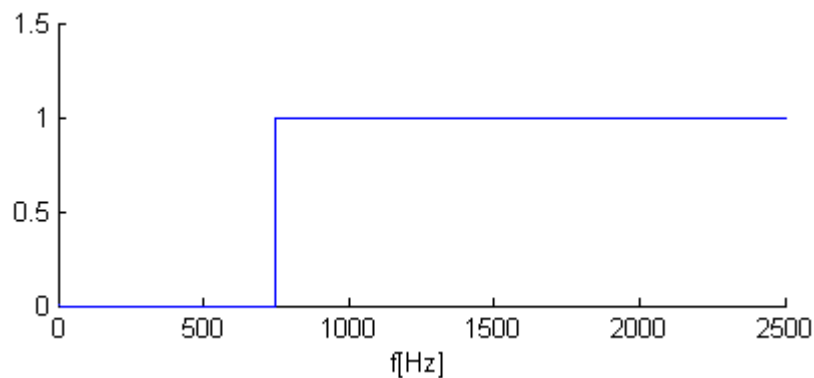


Figura 8. Caracteristica de amplitudine a unui FTS ideal

Conform *principiului superpoziției*, fiecare cauză își produce propriul efect, fără a ține cont de prezența celorlalte. Astfel, fiecare sinusoidă a semnalului de intrare se propagă prin filtru ca și cum ar fi singură. Deci fiecare sinusoidă din semnalul de intrare va trece prin filtru conform caracteristicii de frecvență a filtrului.

Conform caracteristicii de amplitudine din *Figura 8*, este vorba despre un Filtru Trece Sus (FTS) ideal, cu frecvența de tăiere de 750Hz, ceea ce înseamnă că trec doar frecvențele mai mari de 750Hz. Deci la ieșirea din filtru se va obține o sinusoidă cu frecvența de 1000Hz.

Aplicații în Matlab

1. Să se deschidă aplicația *DSP_Assistant* (se selectează în *Current Directory* calea către aplicație, iar în *Command Window* se va scrie *Main_Filters*). Se vor parcurge paginile 36-45.

2. Să se implementeze o funcție numită *tfd*, care să aibă ca parametru de intrare un semnal discret $x[n]$ și care să întoarcă *Transformata Fourier Discretă* a acestuia.

```
function y = tfd(x)
N = length(x);
for n = 0 : N-1
    y(n+1) = 0;
    for k = 0:N-1
        y(n+1) = x(k+1)*exp(-i*2*pi*k*n/N) + y(n+1);
    end
end
end
```

Obs: Rezultatul obținut cu funcția *tfd* să fie comparat cu rezultatul obținut de funcția *fft*.

3. Fie sinusoida $x(t) = \sin(400\pi t)$ eșantionată cu $F_s = 2000$ Hz. Să se reprezinte grafic această sinusoidă precum și spectrul ei.

```
Fs = 2000;
F = 200;
durata = 0.1;
t = 0:1/Fs:durata;
x= sin(2*pi*F*t)
figure(1)
plot(t,x) % semnalul in timp
figure(2)
axa_fft = linspace(-Fs/2, Fs/2, length(x));
plot(axa_fft, fftshift(abs(fft(x)))) % spectrul semnalului
```

4. Fie trei sinusoida $x_1[n]$, $x_2[n]$ și $x_3[n]$ având frecvențele $F_1 = 200$ Hz, $F_2 = 700$ Hz și $F_3 = 1200$ Hz, amplitudine unitară și fază inițială nulă. Semnalele au fost eșantionate cu $F_s = 10$ kHz și însumate, obținându-se astfel semnalul $x[n]$ ($x[n] = x_1[n] + x_2[n] + x_3[n]$). Să se reprezinte grafic semnalele $x_1[n]$, $x_2[n]$, $x_3[n]$, $x[n]$ și spectrele lor.

5. Fie coeficienții unui filtru obținuți prin eșantionarea cu $T_s = 0.1s$ a unui semnal $\text{sinc}(x)$ de frecvență 3Hz între $t_{min} = -1s$ și $t_{max} = 1s$.

- Să se determine coeficienții filtrului.

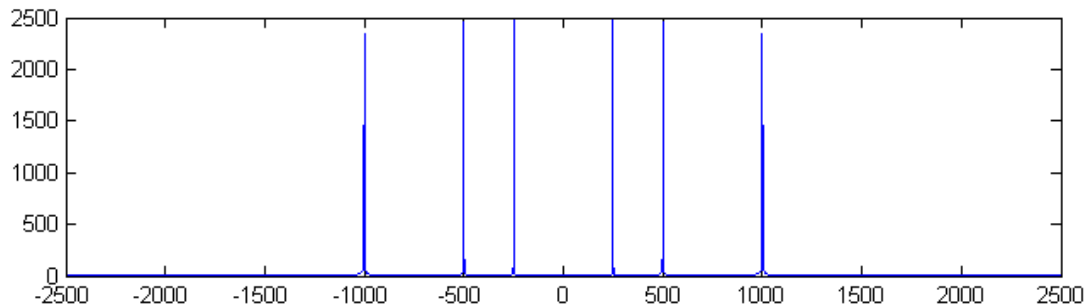
```
t_max = 1;
t_min = -t_max;
Fs = 10;
t = t_min : 1/Fs : t_max;
f = 3; %Hz
h = sinc(2*f*t); % Matlab SINC = Sin(pi*x)/(pi*x) function.
```

- Să se reprezinte grafic coeficienții filtrului.
- Să se determine și să se reprezinte grafic caracteristica de amplitudine a filtrului.

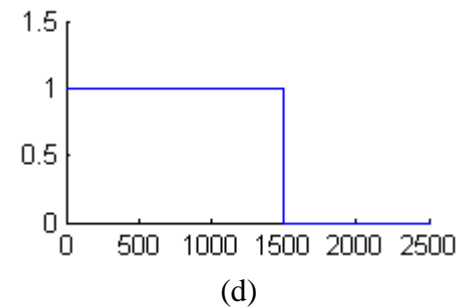
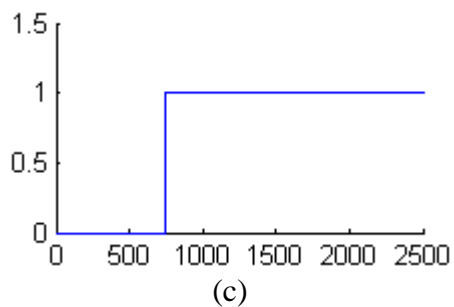
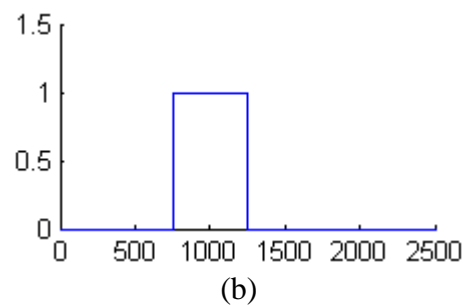
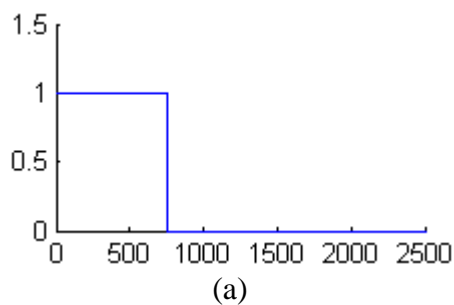
```
axa_fft = linspace(-Fs/2, Fs/2, length(h));
figure(), plot(axa_fft, fftshift(abs(fft(h)))), xlabel('f (Hz)')
```

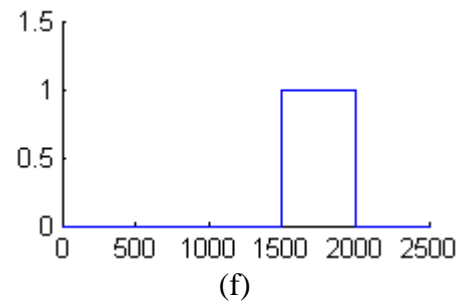
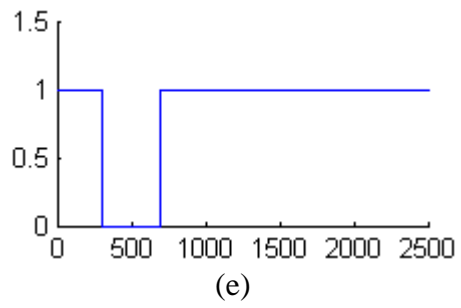
- Să se precizeze tipul filtrului.
- Să se repete experimentul pentru $T_s = 0.01s$, $t_{min} = -10s$ și $t_{max} = 10s$.

6. Fie un semnal $x[n]$ având următorul spectru de amplitudine:



- Care sunt frecvențele sinusoidelor ce intră în componența semnalului $x[n]$?
- Dacă semnalul $x[n]$ ar fi trecut prin diverse filtre ideale având caracteristicile de amplitudine de mai jos, cum ar arăta spectrul de amplitudine al semnalului obținut la ieșirea filtrului?





7. Fie un Filtru Trece Jos ideal, cu frecvența de tăiere $F_t = 800\text{Hz}$. La intrarea filtrului se aduce semnalul $s(t) = 20 \cdot \sin(600\pi t) + 50 \cdot \sin(2000\pi t)$ ce se eșantionează cu $F_s = 5\text{kHz}$. Cum arată spectrul de amplitudine al semnalului obținut la ieșirea filtrului?

Lucrarea 12

Proiectarea Filtrelor Nerecursive (FIR) folosind Transformata Fourier în Timp Discret Inversă

Obiective: înțelegerea pașilor de proiectare a unui filtru FIR folosind Transformata Fourier în Timp Discret Inversă (TFTDI); implementarea unui filtru trece jos (FTJ) și utilizarea acestuia pentru filtrarea unui semnal audio.

1. Filtru Trece Jos (FTJ)

Caracteristica ideală de amplitudine a unui FTJ arată ca în *Figura 1* și se interpretează astfel: toate semnalele sinusoidale având frecvența în intervalul $0 \div Ft$ trec fără a fi amplificate sau atenuate, iar cele având frecvența cuprinsă în intervalul $Ft \div Fs/2$ sunt rejectate.

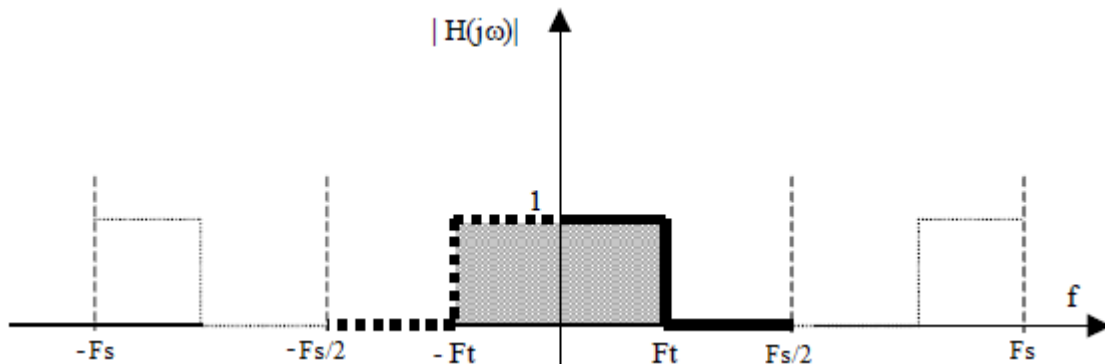


Figura 1. Caracteristica ideală de amplitudine a unui FTJ

Cunoscând caracteristica de amplitudine, se pot determina coeficienții filtrului aplicând *Transformata Fourier în Timp Discret Inversă* (TFTDI). Transformata TFTDI pornește de la o transformată Fourier continuă și obține un semnal în timp discret.

Se obține astfel formula de calcul a coeficienților:

$$h[n] = 2 \cdot \frac{F_t}{F_s} \cdot \text{sinc} \left(n \cdot 2\pi \cdot \frac{F_t}{F_s} \right) \quad (1)$$

Observații:

- În urma transformatei TFTDI vor rezulta o infinitate de coeficienți, dintre care se păstrează doar cei mai semnificativi N coeficienți; N reprezintă **ordinul filtrului**.
- Cei N coeficienți se obțin prin eșantionarea funcției $\text{sinc}(x)$, astfel încât jumătate dintre coeficienți să fie în stânga originii iar cealaltă jumătate să fie în dreapta originii. Depinzând de paritatea lui N , rezultă două situații:

○ N este număr impar. În acest caz eșantionarea se face în punctele:

$$n = \left\{ -\frac{(N-1)}{2}; -\frac{(N-1)}{2} + 1; \dots; -2; -1; 0; +1; +2; \dots; \frac{(N-1)}{2} - 1; \frac{(N-1)}{2} \right\}$$

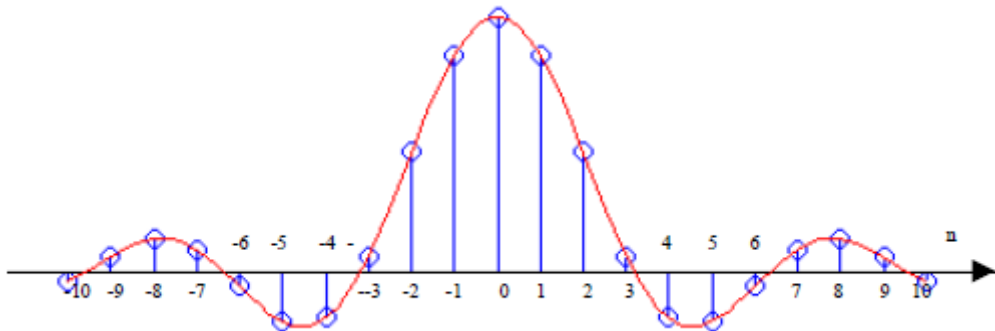


Figura 2. Eșantionarea funcției $\text{sinc}(x)$ pentru N impar ($N = 21$)

○ N este număr par. În acest caz eșantionarea se face în punctele:

$$n = \left\{ -\frac{(N-1)}{2}; -\frac{(N-1)}{2} + 1; \dots; -1.5; -0.5; +0.5; +1.5; \dots; \frac{(N-1)}{2} - 1; \frac{(N-1)}{2} \right\}$$

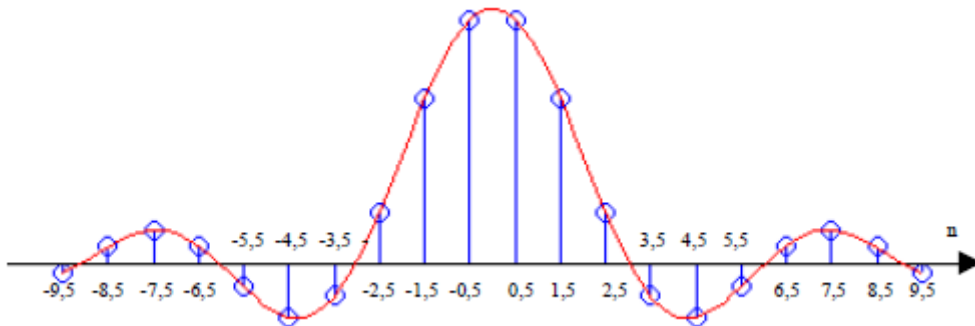


Figura 3. Eșantionarea funcției $\text{sinc}(x)$ pentru N par ($N = 20$)

Cunoscând coeficienții $h[n]$, se poate filtra semnalul $x[n]$ folosind produsul de convoluție dintre $h[n]$ și $x[n]$, obținând semnalul filtrat $y[n]$.

$$y[n] = \sum_{k=0}^{M-1} h[k] \cdot x[n-k] \quad (2)$$

Observație: pentru a calcula convoluția este nevoie ca indicele coeficienților să pornească de la 0. Din acest motiv, semnalul $h[n]$ va trebui întârziat cu $\frac{N-1}{2}$ eșantioane, adică $h\left[-\frac{(N-1)}{2}\right]$ va deveni $h[0]$, $h\left[-\frac{(N-1)}{2} + 1\right]$ va deveni $h[1]$, $h[0]$ va deveni $h\left[\frac{N-1}{2}\right]$, $h\left[\frac{N-1}{2}\right]$ va deveni $h[N-1]$.

Cunoscând coeficienții filtrului, se poate determina caracteristica reală de amplitudine a filtrului, aplicând *Transforma Fourier Discretă* asupra coeficienților. Cu cât numărul coeficienților este mai mare, cu atât caracteristica reală de amplitudine se apropie mai mult de caracteristica ideală, așa cum se poate observa în *Figura 4*.

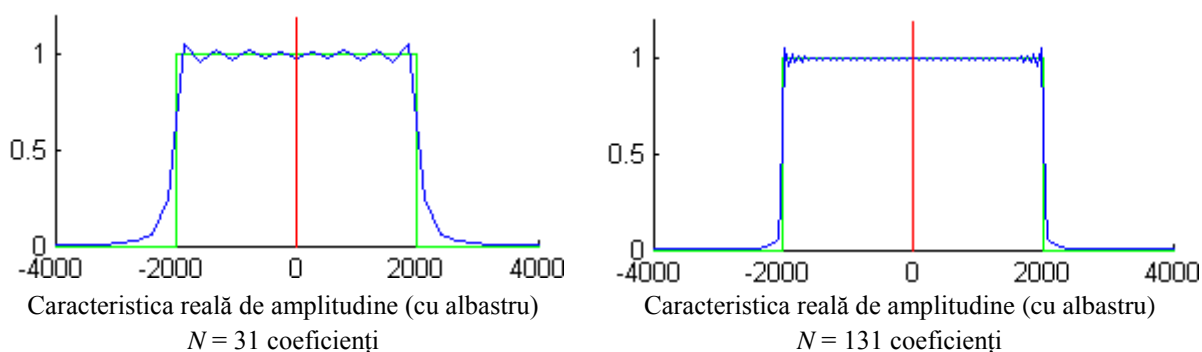


Figura 4. FTJ de ordin diferit

2. Filtru Trece Sus (FTS)

Caracteristica ideală de amplitudine a unui FTS arată ca în *Figura 5* și se interpretează astfel: toate semnalele sinusoidale având frecvența în intervalul $0 \div Ft$ nu trec, iar cele având frecvența cuprinsă în intervalul $Ft \div Fs/2$ trec fără a fi amplificate sau atenuate.

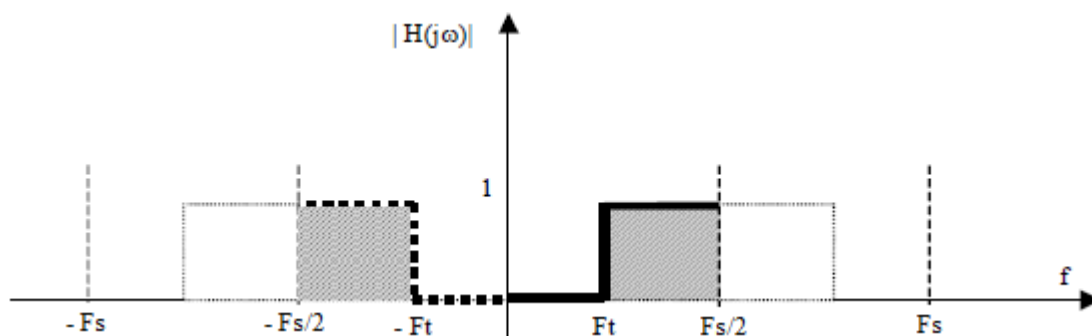


Figura 5. Caracteristica ideală de amplitudine a unui FTS

Ca și în cazul FTJ, se aplică TFTDI caracteristicii ideale de amplitudine și se obține următoarea formulă de calcul a coeficienților filtrului:

$$h[n] = \text{sinc}(\pi n) - 2 \cdot \frac{F_t}{F_s} \cdot \text{sinc}\left(n \cdot 2\pi \cdot \frac{F_t}{F_s}\right) \quad (3)$$

3. Filtru Trece Bandă (FTB)

Caracteristica ideală de amplitudine a unui FTB arată ca în *Figura 6* și se interpretează astfel: toate semnalele sinusoidale având frecvența în intervalul $0 \div Ft_1$ și $Ft_2 \div Fs/2$ nu trec, iar cele având frecvența cuprinsă în intervalul $Ft_1 \div Ft_2$ trec fără a fi amplificate sau atenuate.

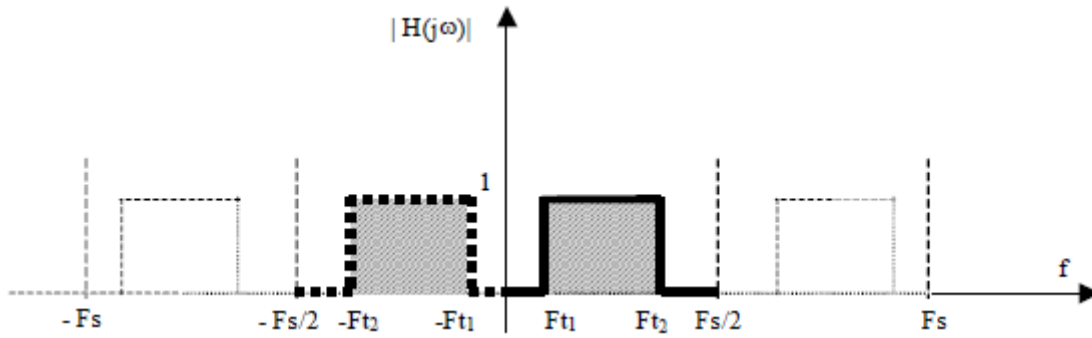


Figura 6. Caracteristica ideală de amplitudine a unui FTB

Pentru a calcula coeficienții filtrului se va aplica următoarea formulă:

$$h[n] = 2 \cdot \frac{F_{t2}}{F_s} \cdot \text{sinc}\left(n \cdot 2\pi \cdot \frac{F_{t2}}{F_s}\right) - 2 \cdot \frac{F_{t1}}{F_s} \cdot \text{sinc}\left(n \cdot 2\pi \cdot \frac{F_{t1}}{F_s}\right) \quad (4)$$

4. Filtru Oprește Bandă (FOB)

Caracteristica ideală de amplitudine a unui FOB arată ca în Figura 7 și se interpretează astfel: toate semnalele sinusoidale având frecvența în intervalul $0 \div Ft_1$ și $Ft_2 \div Fs/2$ trec fără a fi amplificate sau atenuate, iar cele având frecvența cuprinsă în intervalul $Ft_1 \div Ft_2$ nu trec.

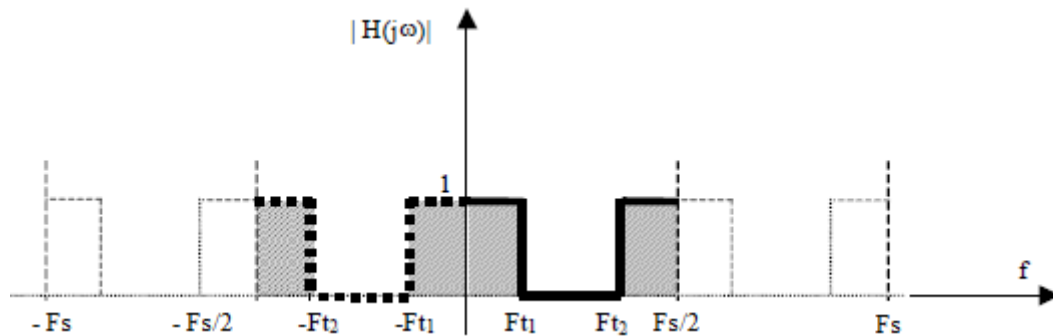


Figura 7. Caracteristica ideală de amplitudine a unui FOB

Pentru a calcula coeficienții filtrului se va aplica următoarea formulă:

$$h[n] = \text{sinc}(\pi n) - 2 \cdot \frac{F_{t2}}{F_s} \cdot \text{sinc}\left(n \cdot 2\pi \cdot \frac{F_{t2}}{F_s}\right) + 2 \cdot \frac{F_{t1}}{F_s} \cdot \text{sinc}\left(n \cdot 2\pi \cdot \frac{F_{t1}}{F_s}\right) \quad (5)$$

Pașii necesari pentru realizarea filtrelor nerecursive folosind transformata Fourier

- Se alege tipul de filtru dorit (FTJ, FTS, FTB, FOB) precizând: numărul de coeficienți N , frecvența de eșantionare F_s , frecvența de tăiere F_t (pentru FTJ și FTS), frecvențele de tăiere F_{t1} și F_{t2} (pentru FOB și FTB).
- Se folosesc relațiile 1, 3, 4, 5 pentru a calcula coeficienții filtrului.
- Se calculează caracteristica reală de amplitudine (calculând TFD a coeficienților filtrului) și se compară cu cea ideală. Dacă rezultatul nu este mulțumitor, se schimbă parametrii filtrului și se recalculează coeficienții.
- Se folosesc cei N coeficienți pentru a filtra semnalul dorit (se aplică formula 2).

Observații:

- TFD a semnalului furnizează spectrul de frecvență al semnalului
- TFD a coeficienților filtrului furnizează caracteristica de frecvență a filtrului

Exemplu. Fie un semnal audio $x[n]$ (voce cu zgomot), eșantionat cu $F_s = 8\text{kHz}$, având următorul spectru de amplitudine.

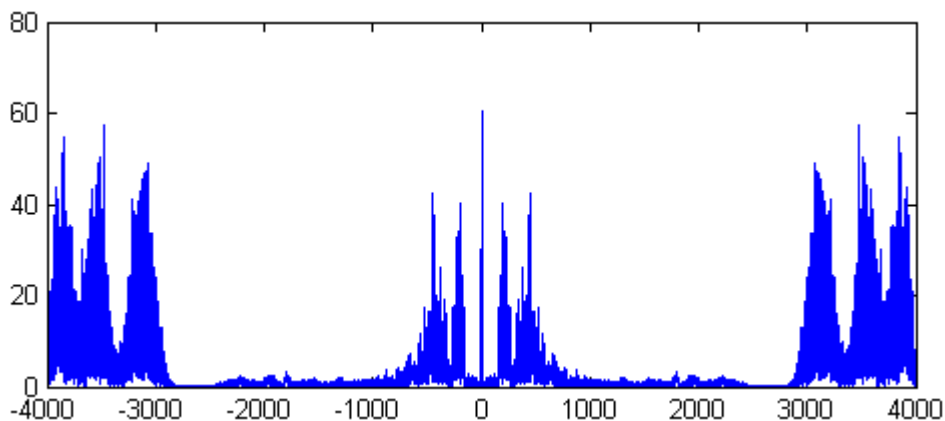


Figura 8. Spectrul semnalului $x[n]$

Ascultând semnalul și privind spectrul putem întui că vocea este pe frecvențe joasă (până în jurul frecvenței de 2600Hz), iar zgomotul pe frecvențe înalte (mai mari de 2900Hz). Ne dorim să filtrăm acest semnal astfel încât să auzim cât mai bine vocea, deci va trebui să rejectăm zgomotul (frecvențele înalte). Vom proiecta astfel un filtru FTJ, care să lase să treacă doar sinusoidalele cu frecvențe mai mici de 2800Hz (frecvență de tăiere determinată empiric din spectrul semnalului). Caracteristica ideală de amplitudine a filtrului FTJ cu $F_t = 2800\text{Hz}$ este reprezentată grafic în Figura 9.

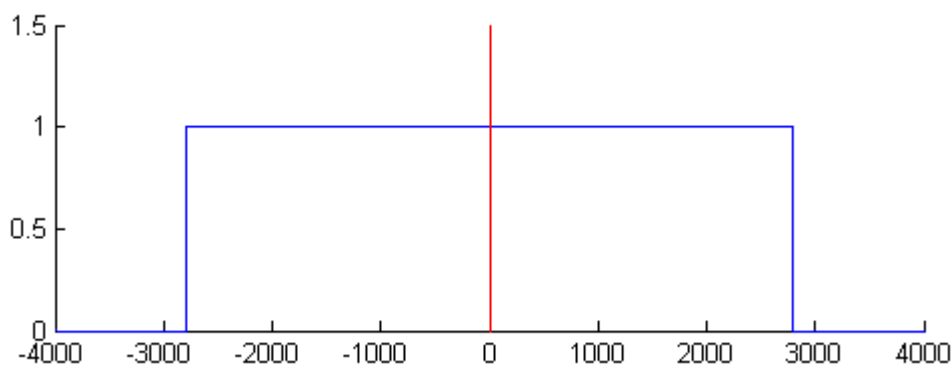


Figura 9. Caracteristica ideală de amplitudine a unui FTJ cu $F_t = 2800\text{Hz}$

În Figura 10 sunt reprezentați cei mai semnificativi 101 coeficienți ai filtrului, calculați cu formula 1, pentru $N = 101$.

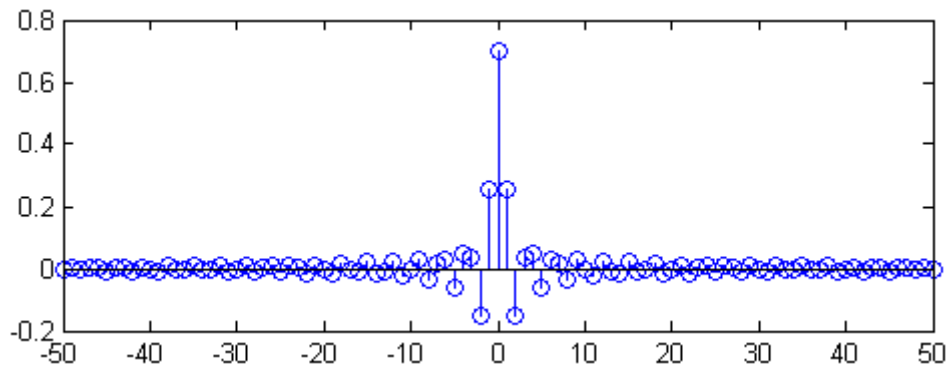


Figura 10. Reprezentarea grafică a celor $N = 101$ coeficienți ai filtrului

Aplicând TFD coeficienților filtrului, se va obține caracteristica reală de frecvență a filtrului. În Figura 11 este reprezentat grafic modulul caracteristicii de frecvență.

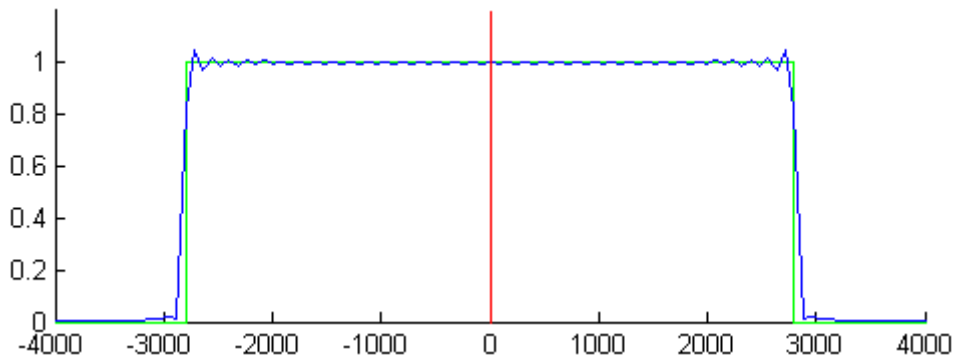


Figura 11. Caracteristica reală de amplitudine (cu albastru) a unui FTJ cu $Ft = 2800\text{Hz}$

Folosind cei $N = 101$ coeficienți determinați anterior se va filtra semnalul $x[n]$ utilizând formula 2. Va rezulta astfel semnalul filtrat $y[n]$, având următorul spectru.

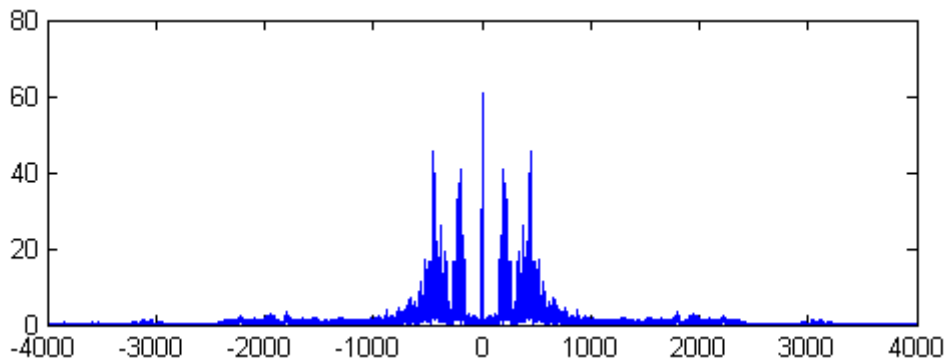


Figura 12. Spectrul semnalului obținut în urma filtrării

Din Figura 12 se poate constata că zgomotul a fost mult atenuat, ceea ce ne și propusesem.

1. Să se deschidă un fișier *M-file* și să se salveze cu denumirea *proiectareFTJ.m*

2. În folderul în care s-a salvat fișierul *proiectareFTJ.m* să se copieze semnalul *semmat.mat*.

3. Să se încarce fișierul *semmat.mat*

```
% incarcare semnal  
load 'semmat'
```

Se constată că în *Workspace* au apărut 2 variabile: variabila *semmat* care conține eșantioanele semnalului cu care vom lucra mai departe și variabila *Fs* care reprezintă frecvența cu care semnalul a fost eșantionat.

4. Să se asculte semnalul. Care este mesajul semnalului?

```
% redare semnal  
sound(semmat, Fs)
```

5. Să se reprezinte grafic spectrul de amplitudine al semnalului. Să se identifice pe ce frecvențe este zgomotul. Ce fel de filtru trebuie implementat pentru a diminua zgomotul de fundal?

```
% spectru semnal  
axa_fft = linspace(-Fs/2, Fs/2, length(semmat));  
figure(1)  
plot(axa_fft, fftshift(abs(fft(semmat)))), title('Spectru semnal')
```

Din spectru se poate observa că frecvențele zgomotului încep de la aproximativ 2800Hz. Deci va trebui implementat un FTJ cu $F_t = 2800\text{Hz}$.

6. Să se reprezinte grafic caracteristica ideală de amplitudine a filtrului FTJ cu $F_t = 2800\text{Hz}$.

```
% caracteristica ideala de amplitudine pentru FTJ cu Ft=2800Hz  
Ft = 2800; % Ft = frecventa de taiere  
H_ideal(-Fs/2+Fs/2+1:-Ft+Fs/2+1)=0;  
H_ideal(-Ft+Fs/2+1:Ft+Fs/2+1)=1;  
H_ideal(Ft+Fs/2+1:Fs/2+Fs/2+1)=0;  
figure(2)  
hold on  
plot(-Fs/2:Fs/2, H_ideal, 'b'), axis([-Fs/2, Fs/2, 0, 1.5])  
plot([0,0], [0, 2], 'r')  
hold off  
title('caracteristica ideala de amplitudine')
```

7. Să se calculeze cei mai semnificativi $N = 101$ coeficienți și să se reprezinte grafic.

```
% calcul coeficienti  
N = 101;  
for n = -(N-1)/2:(N-1)/2  
    h(n+(N-1)/2+1)=2*Ft/Fs*sinc(2*n*Ft/Fs);  
end  
figure(3)  
stem(-(N-1)/2:(N-1)/2,h), title('Coeficienti filtru');
```

Observație: Formula matematică a lui $\text{sinc}(x)$ este $\text{sinc}(x) = \frac{\sin(x)}{x}$ (pentru $x \neq 0$). În Matlab însă, $\text{sinc}(x) = \frac{\sin(\pi x)}{\pi x}$. Implementarea formulei $h[n] = 2 \cdot \frac{F_t}{F_s} \cdot \text{sinc}\left(n \cdot 2\pi \cdot \frac{F_t}{F_s}\right)$ în Matlab trebuie să țină cont de faptul că π apare deja în formulă.

8. Să se reprezinte grafic caracteristica reală de amplitudine comparativ cu caracteristica ideală de amplitudine.

```
% caracteristica reala si cea ideala de amplitudine
H_real = fft(h);
axa_fft2 = linspace(-Fs/2, Fs/2, length(h));
figure(4)
hold on
plot(-Fs/2:Fs/2, H_ideal, 'g'), axis([-Fs/2, Fs/2, 0, 1.2])
plot(axa_fft2, fftshift(abs(H_real)))
plot([0,0], [0, 1.5], 'r')
hold off
title('Caracteristica ideala si cea reala de amplitudine')
```

9. Să se filtreze semnalul folosind coeficienții obținuți la *punctul 7*. Să se asculte semnalul filtrat.

```
y = conv(semnal, h);
sound(y, Fs)
```

10. Să se reprezinte grafic spectrul de amplitudine al semnalului filtrat.

```
axa_fft3 = linspace(-Fs/2, Fs/2, length(y));
figure(5)
plot(axa_fft3, fftshift(abs(fft(y))))
title('Spectru semnal filtrat')
```

11. Să se realizeze o interfață grafică (GUI) pentru un filtru FTJ care să conțină următoarele elemente:

- un sistem de axe în care să fie reprezentat spectrul semnalului original.
- un sistem de axe în care să fie reprezentate caracteristica reală și cea ideală de amplitudine a filtrului.
- un sistem de axe în care să fie reprezentat spectrul semnalului filtrat.
- un *Slider* din care să se modifice numărul de coeficienți ai filtrului.
- un câmp de *EditText* din care să se poată modifica frecvența de tăiere.

12. Să se filtreze semnalul de la *punctul 1*, astfel încât să treacă doar zgomotul.

13. Fie trei sinusoida $x_1[n]$, $x_2[n]$ și $x_3[n]$ având frecvențele $F_1 = 200\text{Hz}$, $F_2 = 700\text{Hz}$ și $F_3 = 1200\text{Hz}$, amplitudine unitară și fază inițială nulă. Semnalele au fost eșantionate cu $F_s = 10\text{kHz}$ și însumate, obținându-se astfel semnalul $x[n]$ ($x[n] = x_1[n] + x_2[n] + x_3[n]$). Să se filtreze semnalul $x[n]$ astfel încât:

- să treacă doar sinusoida de frecvență $F_1 = 200\text{Hz}$.
- să treacă doar sinusoida de frecvență $F_2 = 700\text{Hz}$.
- să treacă doar sinusoida de frecvență $F_3 = 1200\text{Hz}$.
- să treacă doar sinusoida de frecvențe $F_1 = 200\text{Hz}$ și $F_3 = 1200\text{Hz}$.

Lucrarea 13

Proiectarea Filtrelor Nerecursive (FIR) folosind Transformata Fourier Discretă Inversă (TFDI)

Obiective: înțelegerea pașilor de proiectare a unui filtru FIR folosind Transformata Fourier Discretă Inversă (TFDI); implementarea unui Filtru Trece Jos (FT) și utilizarea acestuia pentru filtrarea unui semnal audio.

În lucrarea anterioară, am proiectat filtre FIR folosind TFTDI astfel: am pornit de la o caracteristică de amplitudine de tip **continuu** și aplicând *Transformata Fourier în Timp Discret Inversă* (TFTDI) am determinat coeficienții filtrului. În această lucrare, vom porni de la o caracteristică de amplitudine de tip **discret** și aplicând *Transformata Fourier Discretă Inversă* (TFDI) vom determina coeficienții filtrului. Pe scurt, diferența dintre cele două metode este redată schematic în figura de mai jos.

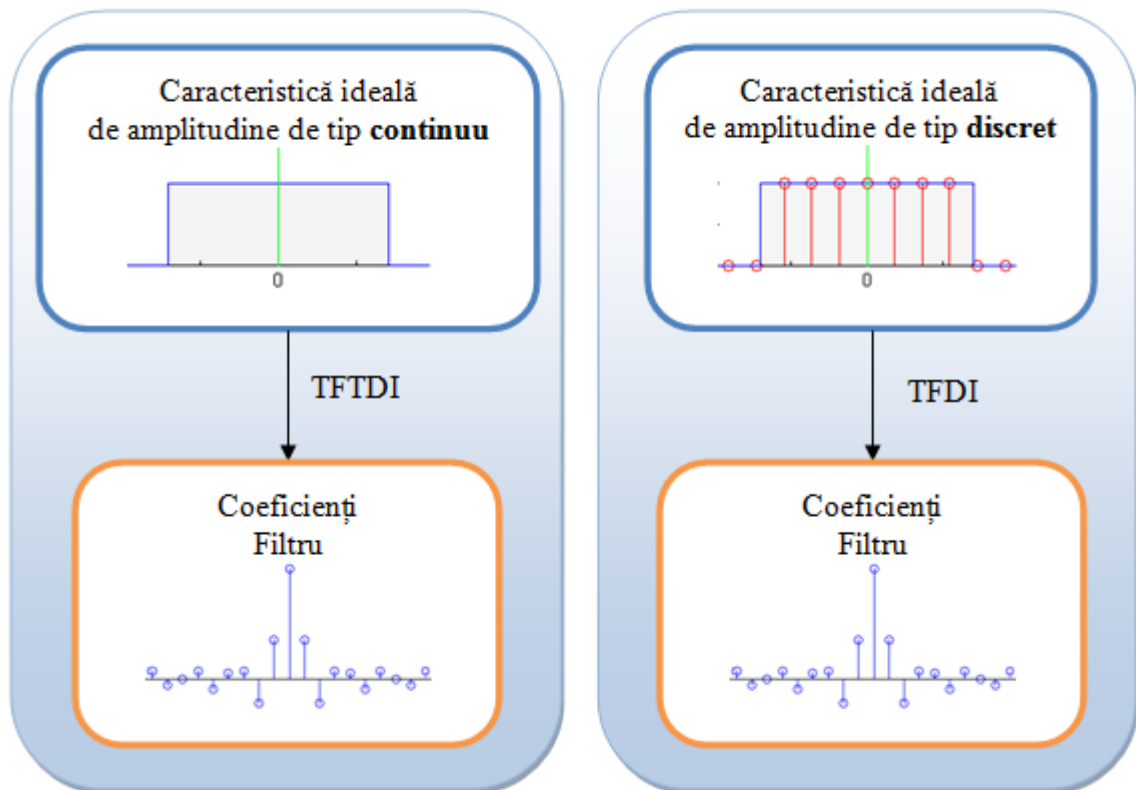


Figura 1. Calculul coeficienților filtrului FIR folosind

a) Transformata Fourier în Timp Discret Inversă b) Transformata Fourier Discretă Inversă

Pentru a construi caracteristica de amplitudine de tip discret, se vor alege N puncte echidistante în intervalul $-F_s/2 \div F_s/2$, astfel încât:

- Distanța dintre două puncte în frecvență să fie F_s/N .
- Să nu fie plasate puncte în extremitățile domeniului de frecvență, adică în $-F_s/2$ și $F_s/2$.

Depinzând de paritatea lui N , există 2 situații:

- N este număr impar. În acest caz frecvențele vor fi:
 $\left\{ -\frac{(N-1)F_s}{2N}, -\frac{(N-3)F_s}{2N}, \dots, -\frac{2F_s}{N}, -\frac{F_s}{N}, 0, \frac{F_s}{N}, \frac{2F_s}{N}, \dots, \frac{(N-3)F_s}{2N}, \frac{(N-1)F_s}{2N} \right\}$
- N este număr par. În acest caz frecvențele vor fi:
 $\left\{ -\frac{(N-1)F_s}{2N}, -\frac{(N-3)F_s}{2N}, \dots, -\frac{3F_s}{2N}, -\frac{F_s}{2N}, \frac{F_s}{2N}, \frac{3F_s}{2N}, \dots, \frac{(N-3)F_s}{2N}, \frac{(N-1)F_s}{2N} \right\}$

În *Figura 2* este prezentat un exemplu de construcție a unei caracteristici discrete de amplitudine folosind $N = 9$ puncte. Se poate observa că această caracteristică discretă nu este altceva decât eșantionarea în frecvență a caracteristicii ideale de amplitudine a unui FTJ.

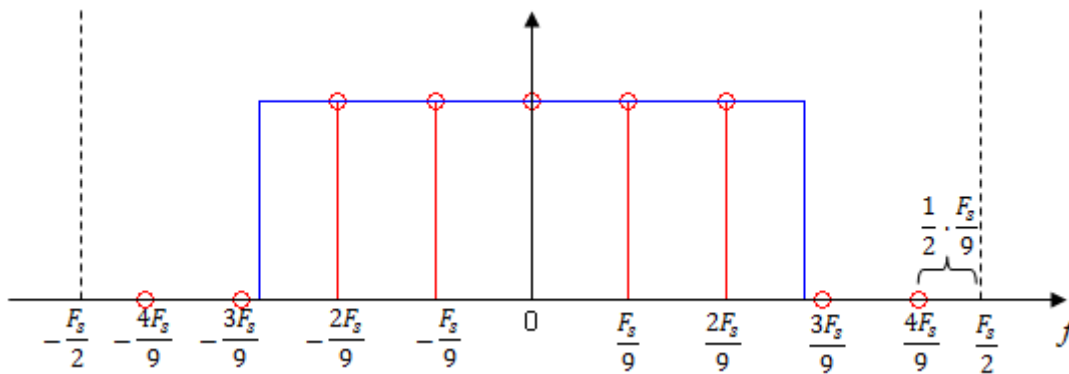


Figura 2. Eșantionarea în frecvență pentru $N = 9$ puncte

Pentru exemplul de mai sus, vor rezulta următoarele valori pentru caracteristica discretă de amplitudine: $H = [0, 0, 1, 1, 1, 1, 1, 0, 0]$.

Determinarea coeficienților filtrului se face aplicând TFDI asupra caracteristicii discrete de amplitudine. Se obține astfel formula de calcul a coeficienților:

$$h[n] = \frac{1}{N} \sum_{k=-\frac{N-1}{2}}^{\frac{N-1}{2}} H[k] \cdot \cos\left(\frac{2\pi \cdot k \cdot n}{N}\right), \text{ unde } n = -\frac{N-1}{2} \dots \frac{N-1}{2} \quad (1)$$

Observații:

- În urma transformatei TFDI vor rezulta N coeficienți unde N este numărul de puncte în care a fost definită caracteristica discretă de amplitudine. N reprezintă **ordinul filtrului**.
- Pentru exemplul din *Figura 2*, cei N coeficienți obținuți reprezintă eșantioane ale funcției $\text{sinc}(x)$; jumătate dintre coeficienți sunt în stânga originii iar cealaltă jumătate sunt în dreapta originii.

Cunoscând coeficienții $h[n]$, se poate filtra semnalul $x[n]$ folosind produsul de convoluție dintre $h[n]$ și $x[n]$, obținând semnalul filtrat $y[n]$.

$$y[n] = \sum_{k=0}^{M-1} h[k] \cdot x[n - k] \quad (2)$$

Pași necesari pentru realizarea filtrelor FIR folosind TFDI

- Se construiește caracteristica discretă de amplitudine precizând: numărul de coeficienți N , frecvența de eșantionare F_s și valorile caracteristicii discrete de amplitudine în cele N puncte.
- Se folosește *formula 1* pentru a calcula coeficienții filtrului.
- Se folosesc cei N coeficienți pentru a filtra semnalul dorit (se aplică *formula 2*).

Exemplu. Fie un semnal audio $x[n]$ (voce cu zgomot), eșantionat cu $F_s = 8\text{kHz}$, având următorul spectru de amplitudine.

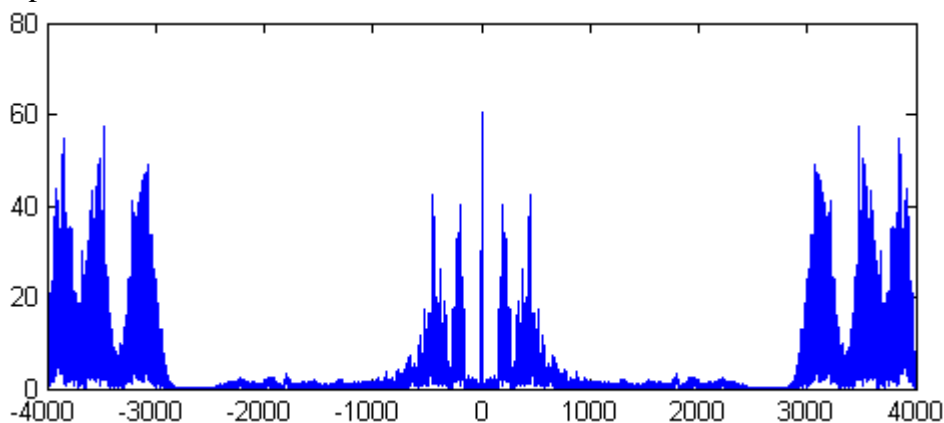


Figura 3. Spectrul semnalului $x[n]$

Ascultând semnalul și privind spectrul putem întui că vocea este pe frecvențe joasă (până în jurul frecvenței de 2600Hz), iar zgomotul pe frecvențe înalte (mai mari de 2900Hz). Ne dorim să filtrăm acest semnal astfel încât să auzim cât mai bine vocea, deci va trebui să rejectăm zgomotul (frecvențele înalte). Vom proiecta astfel un filtru FTJ care să lase să treacă doar sinusoidalele cu frecvențe mai mici de 2800Hz (frecvență de tăiere determinată empiric din spectrul semnalului).

Alegem un număr N de coeficienți și eșantionăm în frecvență caracteristica ideală de amplitudine a unui FTJ. Pentru $N = 11$ coeficienți, caracteristica discretă de amplitudine este cea din Figura 4.

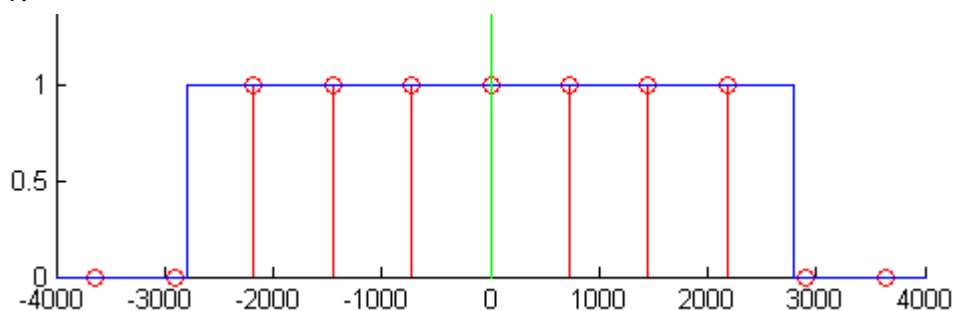


Figura 4. Caracteristica discretă de amplitudine a unui FTJ

Aplicând *formula 1* pentru $N = 11$, se vor obține coeficienții reprezentați grafic în *Figura 5*.

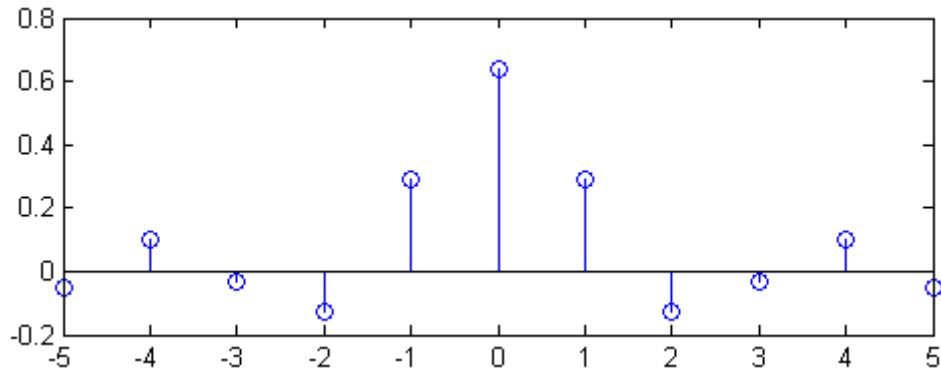


Figura 5. Reprezentarea grafică a celor $N = 11$ coeficienți ai filtrului

Folosind cei 11 coeficienți determinați anterior, se filtrează semnalul $x[n]$ utilizând *formula 2*. Rezultă astfel semnalul filtrat $y[n]$ având spectrul de amplitudine reprezentat grafic în *Figura 6*.

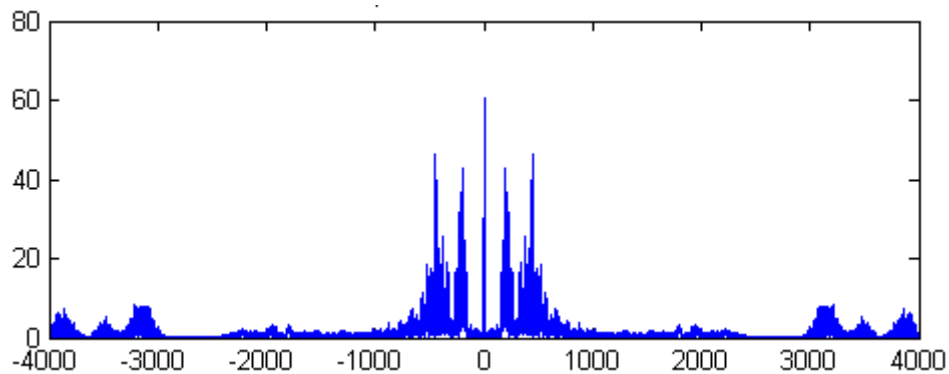


Figura 6. Spectrul semnalului obținut în urma filtrării cu $N = 11$ coeficienți

Analizând *Figura 6* se poate constata că zgomotul a fost atenuat, însă pentru o filtrare mai eficientă trebuie să creștem numărul coeficienților. Astfel, pentru $N = 101$ coeficienți, se poate constata că zgomotul este mult mai bine rejectat.

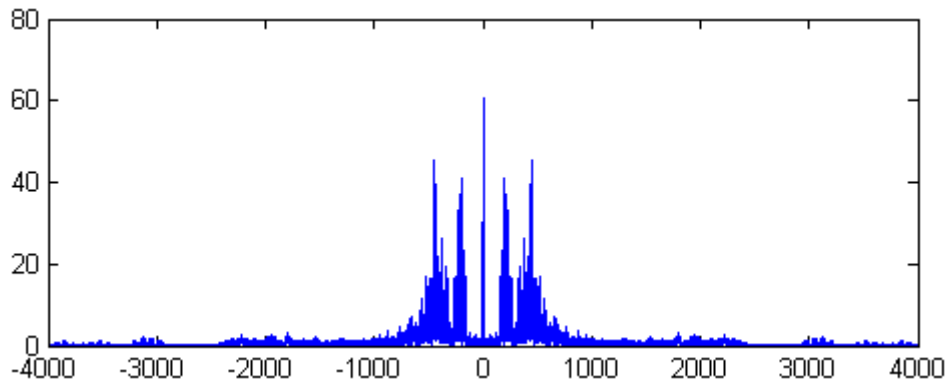


Figura 7. Spectrul semnalului obținut în urma filtrării cu $N = 101$ coeficienți

1. Să se deschidă un fișier *M-file* și să se salveze cu denumirea *proiectareFTJ.m*

2. În folderul în care s-a salvat fișierul *proiectareFTJ.m* să se copieze semnalul *semnal.mat*.

3. Să se încarce fișierul *semnal.mat*.

```
% incarcare semnal
load 'semnal'
```

Se constată că în *Workspace* au apărut 2 variabile: variabila *semnal* care conține eșantioanele semnalului cu care vom lucra mai departe și variabila *Fs* care reprezintă frecvența cu care semnalul a fost eșantionat

4. Să se asculte semnalul. Care este mesajul semnalului?

```
sound(semnal, Fs)
```

5. Să se reprezinte spectrul semnalului. Să se identifice pe ce frecvențe este zgomotul. Ce fel de filtru trebuie implementat pentru a diminua zgomotul de fundal?

```
axa_fft = linspace(-Fs/2, Fs/2, length(semnal));
figure(1)
plot(axa_fft, fftshift(abs(fft(semnal)))), title('Spectru semnal')
```

Din spectru se poate observa că frecvențele zgomotului încep de la aproximativ 2800Hz. Deci va trebui implementat un FTJ cu $F_t = 2800\text{Hz}$.

6. Să se reprezinte grafic caracteristica ideală de amplitudine a filtrului FTJ cu $F_t = 2800\text{Hz}$.

```
% caracteristica ideala de amplitudine pentru FTJ cu Ft=2800Hz
Ft = 2800; % Ft = frecventa de taiere
H_ideal(-Fs/2+Fs/2+1:-Ft+Fs/2+1)=0;
H_ideal(-Ft+Fs/2+1:Ft+Fs/2+1)=1;
H_ideal(Ft+Fs/2+1:Fs/2+Fs/2+1)=0;
figure(2)
hold on
plot(-Fs/2:Fs/2, H_ideal, 'b'), axis([-Fs/2, Fs/2, 0, 1.5])
plot([0,0], [0, 2], 'r')
hold off
title('caracteristica ideala de amplitudine')
```

7. Să se construiască caracteristica discretă de amplitudine a filtrului folosind $N = 101$ coeficienți. Să se reprezinte în același grafic caracteristica ideală și cea discretă de amplitudine.

```
% caracteristica discreta de amplitudine
N = 101; % numar coeficienti filtru
for k = -(N-1)/2:(N-1)/2
    if(abs(k*Fs/N)<Ft)
        H_discret(k+(N-1)/2+1) = 1;
    else
        H_discret(k+(N-1)/2+1) = 0;
    end
end
axa_Hdiscret = linspace(-Fs/2+Fs/(2*N), Fs/2-Fs/(2*N), N);
```

```

figure(3)
hold on
stem(axes_Hdiscret, H_discret, 'r')
plot(-Fs/2:Fs/2, H_ideal)
plot([0 0], [0 1.5], 'g')
axis([-Fs/2, Fs/2, 0, 2])
title('Hideal si Hdiscret')
hold off

```

8. Să se calculeze coeficienții filtrului folosind *formula 1* și să se reprezinte grafic. Coeficienții filtrului reprezintă eșantioane ale cărei funcții matematice?

```

% calcul coeficienti
for n = -(N-1)/2:(N-1)/2
    h(k+(N-1)/2+1) = 0;
    for k = -(N-1)/2:(N-1)/2
        h(n+(N-1)/2+1) = h(n+(N-1)/2+1) + H_discret(k+(N-1)/2+1)*cos(2*pi*k*n/N);
    end
end
h = h/N;
figure(4), stem(-(N-1)/2:(N-1)/2, h), title('Coeficienti filtru')

```

9. Să se filtreze semnalul folosind coeficienții obținuți la *punctul 8*. Să se asculte semnalul filtrat.

```

% filtrare semnal
y = conv(semnal, h);
sound(y, Fs)

```

10. Să se reprezinte grafic spectrul semnalului filtrat.

```

% spectrul semnalului filtrat
axes_fft3 = linspace(-Fs/2, Fs/2, length(y));
figure(5)
plot(axes_fft3, fftshift(abs(fft(y))))
title('Spectru semnal filtrat')

```

11. Să se filtreze semnalul de la *punctul 1*, astfel încât să treacă doar zgomotul, semnalul util (vocea) fiind rejectat.

Lucrarea 14

Filtre Recursive (IIR)

Obiective: determinarea funcției de transfer utilizând transformata \mathcal{Z} ; implementarea ecuațiilor cu diferențe finite folosind diagrame; identificarea ordinului unui filtru; implementarea unui filtru notch utilizând metoda poli-zero-uri.

Un filtru recursiv (*Infinite Impulse Response* – IIR) este un sistem numeric descris în domeniul timp de următoarea relație (ecuație cu diferențe finite):

$$y[n] = \sum_{k=0}^N b_k \cdot x[n-k] - \sum_{i=1}^M a_i \cdot y[n-i] \quad (1)$$

- $x[n]$ reprezintă o secvență de date de intrare
- $y[n]$ reprezintă o secvență de date de ieșire
- b_k și a_i reprezintă coeficienții filtrului

Observație: În cazul în care toți coeficienții a_i sunt nuli, formula 1 descrie un filtru FIR.

Transformata \mathcal{Z}

Pentru analiza ecuațiilor cu diferențe finite precum și a răspunsului în frecvență a filtrelor numerice, instrumentul de bază folosit este transformata \mathcal{Z} , definită astfel:

$$X(z) = \mathcal{Z}\{x[n]\} = \sum_{n=-\infty}^{\infty} x[n] \cdot z^{-n} \quad (2)$$

unde z este o variabilă complexă.

Dintre proprietățile transformatei \mathcal{Z} menționăm întârzierea, definită astfel:

$$\mathcal{Z}\{x[n-m]\} = z^{-m} \cdot X(z) \quad (3)$$

Funcția de transfer

$\mathbf{H}(z)$ se numește **funcția de transfer** a sistemului și este definită astfel: $H(z) = \frac{Y(z)}{X(z)}$.

Pornind de la ecuația cu diferențe finite (formula 1), se obține:

$$y[n] + a_1 y[n-1] + \dots + a_M y[n-M] = b_0 x[n] + b_1 x[n-1] + \dots + b_N x[n-N]$$

Aplicând transformata \mathcal{Z} (formula 2) și folosind proprietatea de întârziere (formula 3), rezultă:

$$Y[z] + a_1 z^{-1} Y[z] + \dots + a_M z^{-M} Y[z] = b_0 X[z] + b_1 z^{-1} X[z] + \dots + b_N z^{-N} X[z]$$

Se ajunge astfel la formula generală a funcției de transfer pentru un filtru IIR:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + \dots + b_N z^{-N}}{1 + a_1 z^{-1} + \dots + a_M z^{-M}} \quad (4)$$

Observații:

- La numărătorul funcției de transfer se găsesc coeficienții lui $x[n]$ iar la numitorul funcției de transfer se găsesc coeficienții lui $y[n]$.
- Rădăcinile numărătorului se numesc **zerouri** ale funcției de transfer.
- Rădăcinile numitorului se numesc **poli** ai funcției de transfer.
- **Polii amplifică sinusoide de anumite frecvențe.**
- **Zerourile atenuează sinusoide de anumite frecvențe.**
- Un filtru FIR nu are poli, are numai zerouri.
- Pentru a obține numai coeficienți reali ai filtrului (b_k și a_i) trebuie ca atât polii cât și zerourile să fie perechi complex conjugate.

Diagrama de implementare a ecuațiilor cu diferențe finite

Modul de implementare al algoritmului recursiv (formula 1), cunoscând funcția de transfer este reprezentat grafic în Figura 1 și poartă denumirea de *forma de realizare directă I*.

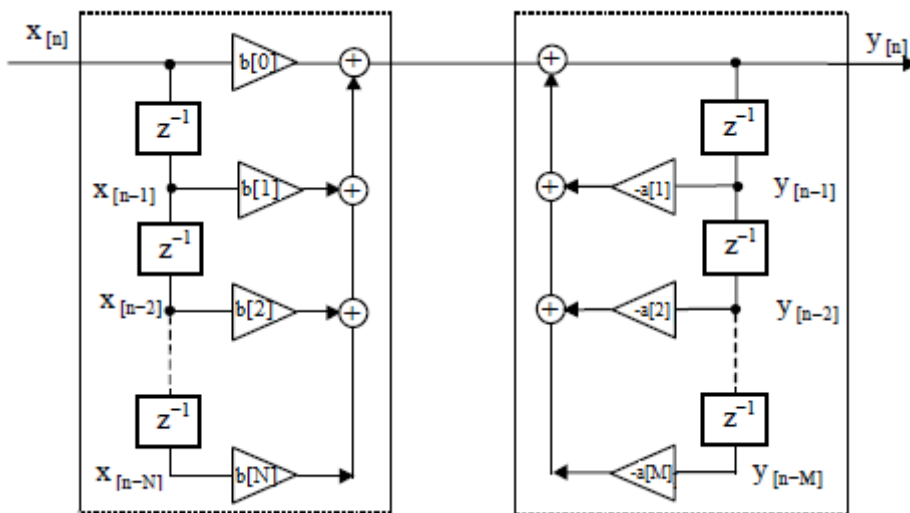


Figura 1. Diagrama de implementare. Forma directă I

Observație: blocurile notate cu z^{-1} reprezintă celule de întârziere.

Ordinul filtrului reprezintă numărul maxim al celulelor de întârziere dintre cele aflate la intrare și cele de la ieșire.

Lagătura dintre transformată Z și TFD

Transformata Fourier Discretă poate fi considerată un caz special de transformată Z , unde variabila z este evaluată pe cercul unitate în planul complex.



Figura 2. Legătura dintre transformata Z și transformata Fourier

Reprezentarea frecvențelor în planul z

Variabila complexă z poate fi scrisă:

$$z = e^{j\omega Ts} = \cos\left(2\pi \frac{f}{F_s}\right) + j \cdot \sin\left(2\pi \frac{f}{F_s}\right) \quad (5)$$

În *Figura 3* este reprezentat planul complex z , în care cercul unitate corespunde axei frecvențelor din *Transformata Fourier*.

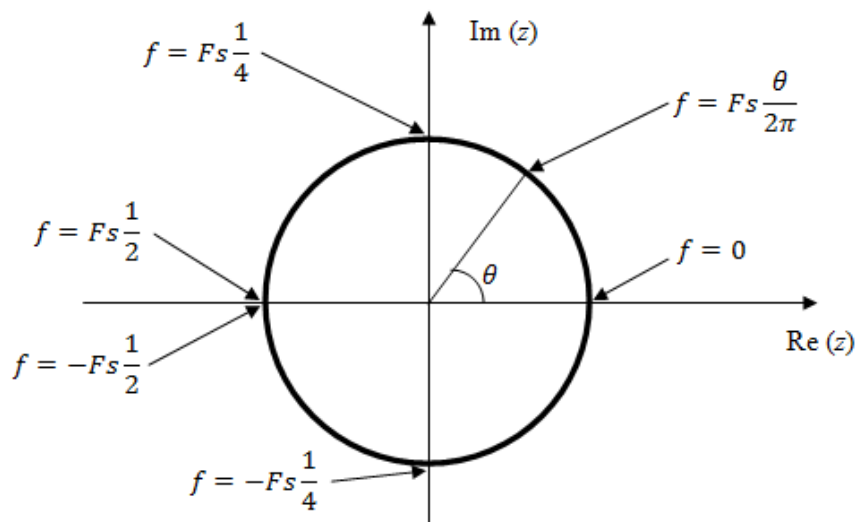


Figura 3. Poziția semnalelor sinusoidale în planul z

Observație: pe semicercul superior se găsesc frecvențele din intervalul $[0 \div F_s/2]$. Semicercul inferior este alocat frecvențelor negative.

Exercițiu 1. Fie funcția de transfer $H(z) = \frac{(3-z^{-1})^2}{(1+\sqrt{3}\cdot j-z^{-1})(1-\sqrt{3}\cdot j-z^{-1})}$

- să se determine polii și zerourile funcției de transfer și să se reprezinte în planul z .
- pornind de la $H(z)$ să se deducă ecuația cu diferențe finite.
- să se deseneze schema bloc a filtrului (forma directă I).
- care este ordinul filtrului?

Rezolvare:

$$\text{a) polii reprezintă rădăcinile numitorului, deci } \begin{cases} 1 - \sqrt{3} \cdot j - z^{-1} = 0 \Rightarrow p_1 = (1 + \sqrt{3} \cdot j)/4 \\ 1 + \sqrt{3} \cdot j - z^{-1} = 0 \Rightarrow p_2 = (1 - \sqrt{3} \cdot j)/4 \end{cases}$$

$$\text{zerourile reprezintă rădăcinile numărătorului, deci } (3 - z^{-1})^2 = 0 \Rightarrow z_1 = z_2 = \frac{1}{3}$$

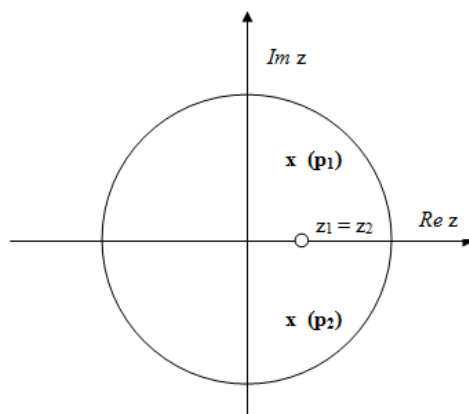


Figura 4. Reprezentarea polilor și zerourilor în planul z

$$b) H(z) = \frac{Y(z)}{X(z)} \Rightarrow \frac{9-6z^{-1}+z^{-2}}{4-2z^{-1}+z^{-2}} = \frac{Y(z)}{X(z)} \Rightarrow X(z)(9-6z^{-1}+z^{-2}) = Y(z)(4-2z^{-1}+z^{-2})$$

$$9X(z) - 6X(z)z^{-1} + X(z)z^{-2} = 4Y(z) - 2Y(z)z^{-1} + Y(z)z^{-2}$$

Folosind teorema întârzierii se obține:

$$9x[n] - 6x[n-1] + x[n-2] = 4y[n] - 2y[n-1] + y[n-2]$$

Ecuția cu diferențe finite a filtrului IIR este:

$$y[n] = \frac{9}{4}x[n] - \frac{3}{2}x[n-1] + \frac{1}{4}x[n-2] + \frac{1}{2}y[n-1] - \frac{1}{4}y[n-2]$$

c) Forma directă I

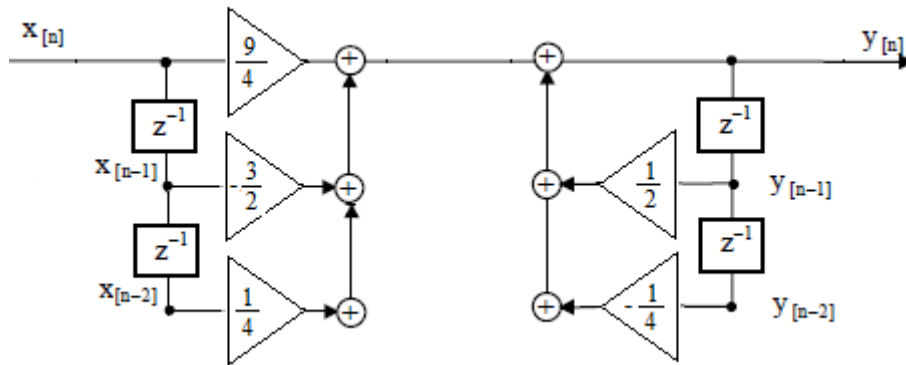


Figura 5. Diagrama. Forma directă I

d) ordinul filtrului este 2.

Exercițiu 2. Fie un sistem definit de următoarea relație: $y[n] = x[n] - x[n-1]$.

- relația de mai sus descrie un filtru FIR sau IIR?
- care este ordinul filtrului?
- să se determine funcția de transfer a sistemului.
- să se determine modulul caracteristicii de frecvență a filtrului și să se reprezinte grafic.

Rezolvare:

- relația este nerecursivă, deci este un filtru FIR.
- ordinul filtrului este 1, (există o singură întârziere a semnalului $x[n]$).
- aplicând transformata Z, se obține $Y(z) = X(z) - z^{-1}X(z) \Rightarrow Y(z) = X(z)(1 - z^{-1})$.

Funcția de transfer va fi $H(z) = \frac{Y(z)}{X(z)} = 1 - z^{-1}$.

d) Caracteristica de frecvență se scrie $H(e^{j\omega Ts}) = H(z)|_{z=e^{j\omega Ts}} = 1 - e^{-j\omega Ts}$.

Folosind formula Euler ($e^{jx} = \cos(x) + j \cdot \sin(x)$) rezultă:

$$H(e^{j\omega Ts}) = 1 - \cos(\omega Ts) + j \cdot \sin(\omega Ts)$$

Modulul caracteristicii de frecvență va fi:

$$|H(e^{j\omega Ts})| = \sqrt{(1 - \cos(\omega Ts))^2 + (\sin(\omega Ts))^2}$$

$$|H(e^{j\omega Ts})| = \sqrt{2 - 2\cos\left(\frac{2\pi \cdot f}{Fs}\right)}$$

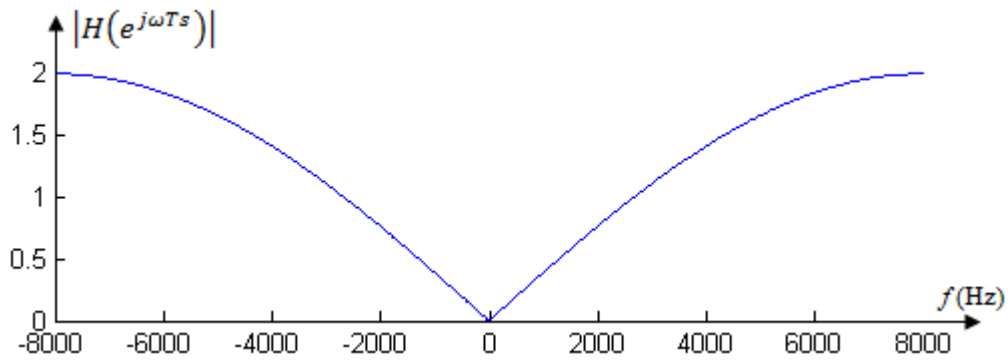


Figura 6. Caracteristica de amplitudine a filtrului descris de relația $y[n] = x[n] - x[n - 1]$, $F_s = 16\text{kHz}$

Din grafic se poate observa că s-a obținut caracteristica de amplitudine a unui FTS, ceea ce era de așteptat, deoarece relația de la care am pornit este cea de derivare (iar derivarea se comportă precum un filtru FTS).

Implementarea unui filtru *notch*, folosind metoda poli-zero

Un filtru *notch* este astfel proiectat încât să rejeteze o anumită frecvență. Așa cum am menționat anterior, zerourile funcției de transfer anulează sinusoidă de anumite frecvențe. Deci pentru proiectarea unui filtru *notch* care să rejeteze o sinusoidă de frecvență F , este nevoie să plasăm un *zero* pe cercul unitate, pe frecvența F . Dacă dorim să obținem coeficienți reali, atunci este nevoie de o pereche de *zerouri* complex conjugate, ca în Figura 7.

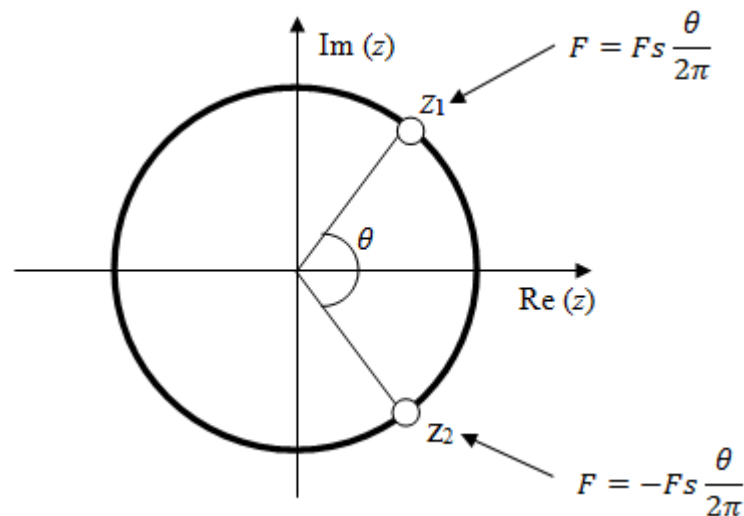


Figura 7. Plasarea a două zerouri complex conjugate pe cercul unitate

Fie cele două zerouri: $z_1 = e^{j\theta}$ și $z_2 = e^{-j\theta}$, unde $\theta = 2\pi \cdot \frac{F}{F_s}$ iar F este frecvența ce se dorește a fi rejecată. Funcția de transfer se scrie:

$$H(z) = (1 - z_1 \cdot z^{-1})(1 - z_2 \cdot z^{-1})$$

$$H(z) = 1 - 2 \cdot \cos\theta \cdot z^{-1} + z^{-2}$$

Rezultă ecuația cu diferențe finite: $y[n] = x[n] - 2 \cdot \cos\theta \cdot x[n - 1] + x[n - 2]$.

Coeficienții filtrului *notch* sunt:

- coeficienții lui x : $B = [1, -2 \cdot \cos\theta, 1]$.
- coeficienții lui y : $A = [1]$.

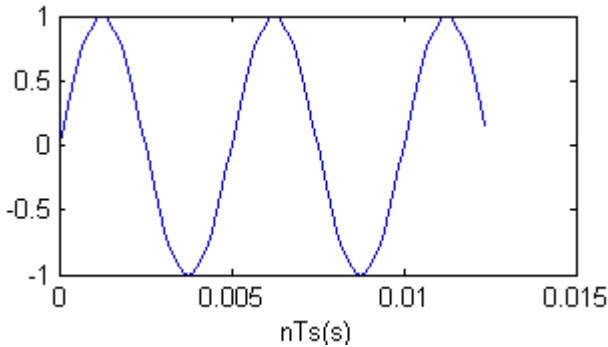
Observații:

- pentru a realiza în Matlab filtrarea unui semnal cunoscând coeficienții filtrului (B este setul de coeficienți ai lui x și A este setul de coeficienți ai lui y) se folosește funcția `filter(B, A, x)`.
- pentru a trasa caracteristica de frecvență se folosește funcția `freqz(B, A)`.

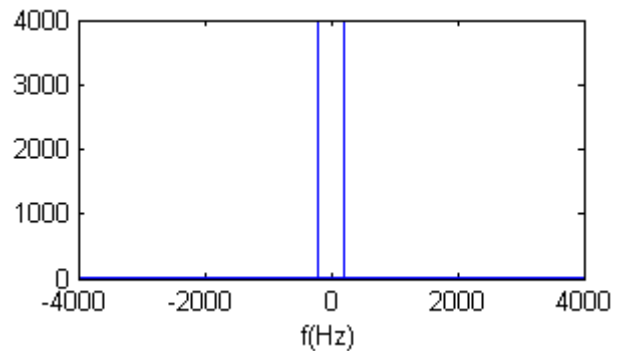
Exemplu. Fie două sinusoidale $x_1[n]$ și $x_2[n]$ având frecvențele $F_1 = 200\text{Hz}$ și $F_2 = 600\text{Hz}$, amplitudine unitară și fază inițială nulă. Semnalele au fost eșantionate cu $F_s = 8\text{kHz}$ și însumate, obținându-se astfel semnalul $x[n]$ ($x[n] = x_1[n] + x_2[n]$). Să se filtreze semnalul $x[n]$ astfel încât să treacă doar sinusoida de frecvență $F_1 = 200\text{Hz}$.

Rezolvare:

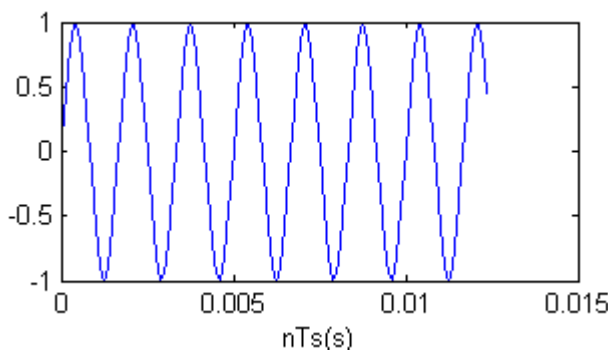
Semnalul $x[n]$ conținând sinusoidale de frecvențe F_1 și F_2 este reprezentat grafic în *Figura 8.e*. Pentru a păstra doar sinusoida de frecvență F_1 , vom proiecta un filtru *notch* care să rejeteze frecvența $F_2 = 600\text{Hz}$. Cele două zerouri sunt: $z_1 = e^{j\theta}$ și $z_2 = e^{-j\theta}$, unde $\theta = 2\pi \cdot \frac{F_2}{F_s}$. Rezultă coeficienții $B = [1, -2 \cdot \cos\theta, 1]$ și $A = [1]$. Folosind funcția `filter` a Matlabului, obținem semnalul $y = \text{filter}(B, A, x)$, reprezentat grafic în *Figura 8.g*.



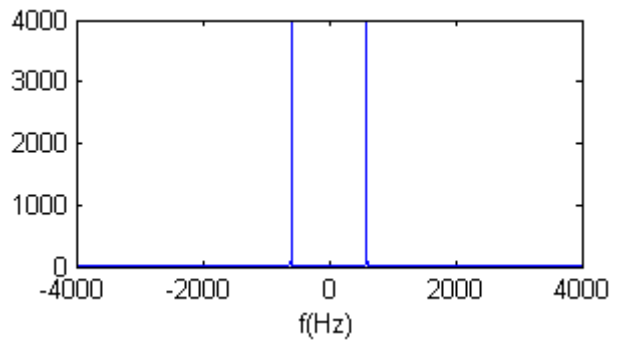
a) Semnalul $x_1[n]$ reprezentat în timp



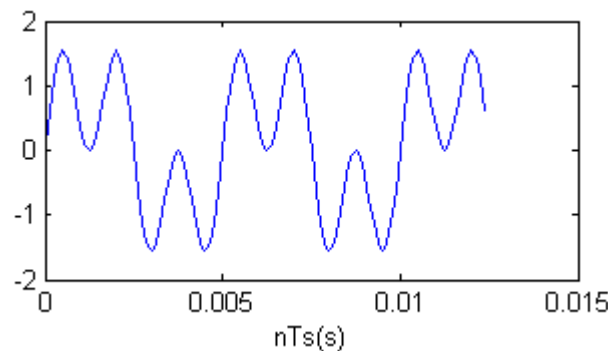
b) Spectrul semnalului $x_1[n]$



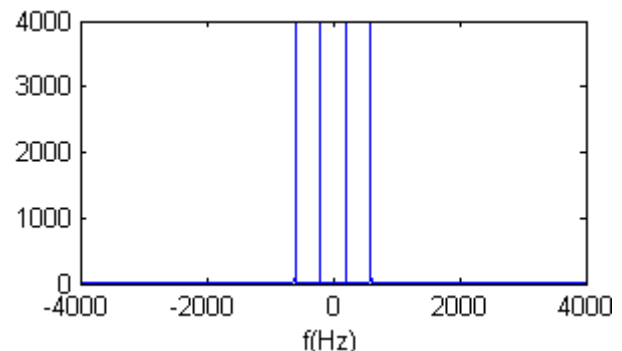
c) Semnalul $x_2[n]$ reprezentat în timp



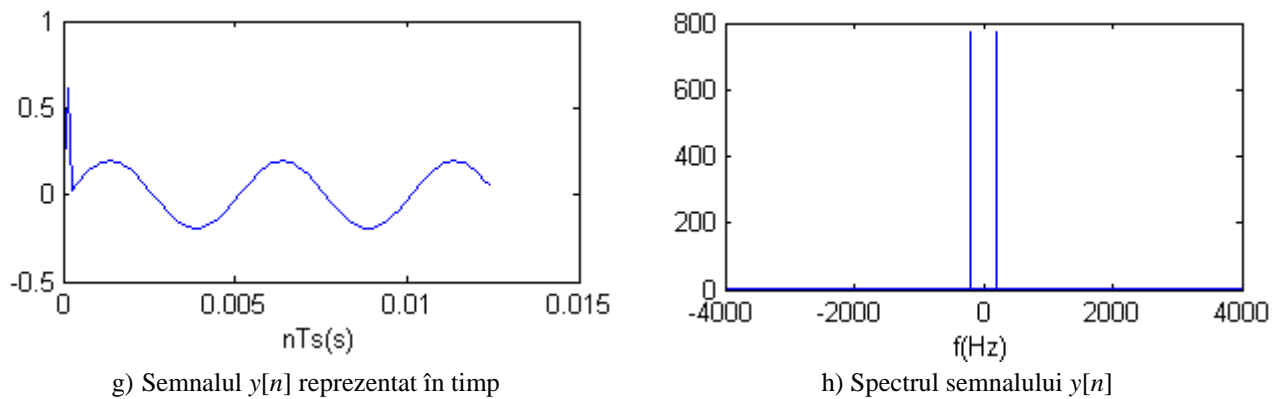
d) Spectrul semnalului $x_2[n]$



e) Semnalul $x[n]$ reprezentat în timp



f) Spectrul semnalului $x[n]$



g) Semnalul $y[n]$ reprezentat în timp

h) Spectrul semnalului $y[n]$

Figura 8. Semnalele $x_1[n]$, $x_2[n]$, $x[n]$ ($x[n] = x_1[n] + x_2[n]$) și spectrele lor

Așa cum se poate observa din *Figura 8.g*, în urma filtrării a trecut sinusoida de 200Hz, atenuată însă de aproximativ 5 ori. Pentru a înțelege de ce s-a întâmplat acest lucru, trebuie să analizăm caracteristica de amplitudine a filtrului *notch* implementat (*Figura 9* obținută folosind funcția `freqz` a Matlabului).

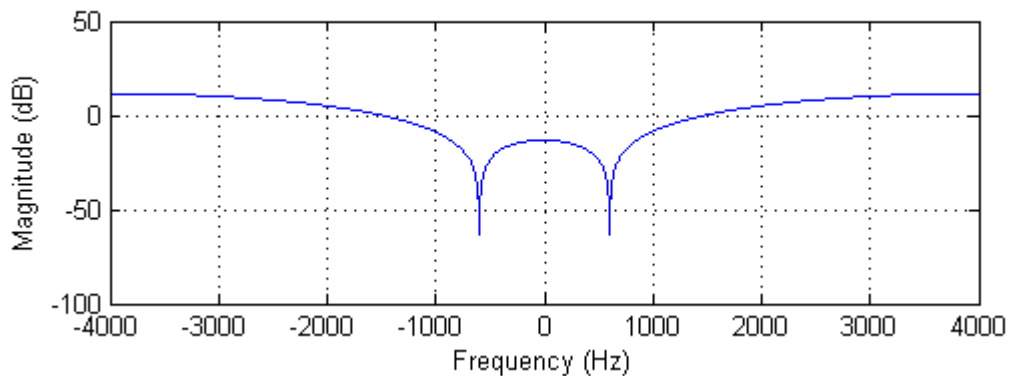


Figura 9. Caracteristica de amplitudine a filtrului *notch* având coeficienții $B = [1, -2 \cdot \cos\theta, 1]$ și $A = [1]$, proiectat să rejeteze frecvența de 600Hz

Conform caracteristicii de amplitudine, sinusoida cu frecvența de 600Hz este mult atenuată ($\cong -63dB \Rightarrow 20\log|A| = -63dB \Rightarrow |A| \cong 0.7 \cdot 10^{-3}$), iar sinusoida cu frecvența de 200Hz are în urma filtrării $\cong -14.27dB \Rightarrow 20\log|A| = -14.27dB \Rightarrow |A| \cong 0.2$.

Exerciții

Exercițiu 1. Fie un filtru IIR definit de relația $y[n] = 2 \cdot x[n] - y[n - 1]$. Știind că $y[0] = 0$ și că $x[n]$ este un semnal constant de valoare 2, să se determine $y[101]$.

Exercițiu 2. Fie funcția de transfer $H(z) = \frac{(2-z^{-1})(2+z^{-1})}{4+z^{-2}}$.

- să se determine polii și zerourile funcției de transfer.
- pornind de la $H(z)$ să se deducă ecuația cu diferențe finite.
- să se reprezinte grafic schema bloc a filtrului (forma directă I).
- care este ordinul filtrului?

Exercițiu 3. Fie un sistem definit de următoarea relație:

$$y[n] = \frac{1}{3}(x[n] + x[n - 1] + x[n - 2]).$$

- relația de mai sus descrie un filtru FIR sau IIR?
- care este ordinul filtrului?
- să se determine funcția de transfer a sistemului.
- să se determine modulul caracteristicii de frecvență a filtrului și să se reprezinte grafic.

Aplicații în Matlab

1. Fie un sistem descris de următoarea ecuație cu diferențe finite: $y[n] = x[n] - x[n - 1]$. Să se reprezinte grafic caracteristica de amplitudine, cunoscând $F_s = 8000$ și $N = 1000$ puncte.

Soluție 1: folosind formula determinată în *Exemplu 2* ($|H(e^{j\omega T_s})| = \sqrt{2 - 2\cos\left(\frac{2\pi \cdot f}{F_s}\right)}$).

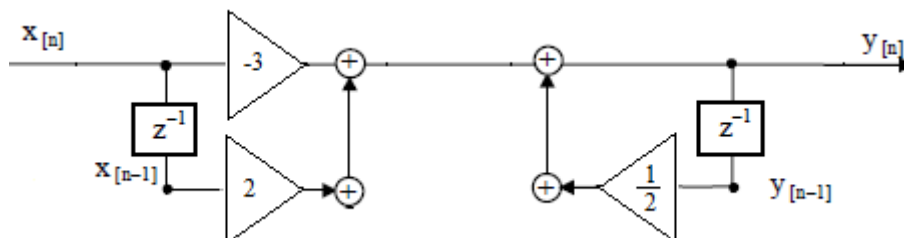
```
Fs = 8000;  
F = linspace(-Fs/2, Fs/2, 1000);  
% varianta 1, folosind formula  
H = sqrt(2 - 2*cos(2*pi*F/Fs));  
figure(1), plot(F, H)
```

Soluție 2: folosind funcția *freqz* a Matlabului

```
% varianta 2, folosind functia freqz a Matlabului  
figure(2), freqz([1,-1],1, F,Fs)
```

2. Fie un sistem descris de următoarea ecuație cu diferențe finite: $y[n] = x[n] + x[n - 1]$. Să se reprezinte grafic caracteristica de amplitudine, cunoscând $F_s = 8000$ și $N = 1000$ puncte.

3. Fie un filtru având următoarea diagramă de implementare.



- Să se determine funcția de transfer a filtrului.
- Să se reprezinte grafic caracteristica de frecvență a filtrului. Ce fel de filtru este?

Indiciu: `>> freqz([-3, 2], [1, -0.5], 1024)`

4. Fie două sinusoida $x_1[n]$ și $x_2[n]$ având frecvențele $F_1 = 200\text{Hz}$ și $F_2 = 600\text{Hz}$, amplitudine unitară fază inițială nulă și durată de 1 secundă. Semnalele au fost eșantionate cu $F_s = 8\text{kHz}$ și însumate, obținându-se astfel semnalul $x[n]$ ($x[n] = x_1[n] + x_2[n]$). Să se filtreze semnalul $x[n]$ astfel încât să treacă doar sinusoida de frecvență $F_1 = 200\text{Hz}$. Să se afișeze:

- semnalul $x_1[n]$ și spectrul semnalului $x_1[n]$.
- semnalul $x_2[n]$ și spectrul semnalului $x_2[n]$.
- semnalul $x[n]$ și spectrul semnalului $x[n]$.
- semnalul $y[n]$ (obținut în urma filtrării) și spectrul semnalului $y[n]$.
- caracteristica de frecvență a filtrului folosind funcția *freqz*.

```
F1 = 200; % frecventa sinusoida x1
F2 = 600; % frecventa sinusoida x2
Fs = 8000; % frecventa de esantionare

t = 0:1/Fs:1;
x1 = sin(2*pi*F1*t);
x2 = sin(2*pi*F2*t);
x = x1 + x2;

teta = 2*pi*F2/Fs;
B = [1 -2*cos(teta) 1]; % coeficientii lui x[n]
A = 1; % coeficientul lui y[n]
y = filter(B,A,x); % y reprezinta semnalul obtinut in urma filtrarii

figure(1), plot(t(1:100), x1(1:100)), xlabel('nTs (s)'), ylabel('x1[n]')
figure(2), plot(linspace(-Fs/2,Fs/2,length(x1)), fftshift(abs(fft(x1)))),
xlabel('f (Hz)'), ylabel('fftshift(abs(fft(x1)))')
figure(3), plot(t(1:100), x2(1:100)), xlabel('nTs (s)'), ylabel('x2[n]')
figure(4), plot(linspace(-Fs/2,Fs/2,length(x2)), fftshift(abs(fft(x2)))),
xlabel('f (Hz)'), ylabel('fftshift(abs(fft(x2)))')
figure(5), plot(t(1:100), x(1:100)), xlabel('nTs (s)'), ylabel('x[n]')
figure(6), plot(linspace(-Fs/2,Fs/2,length(x)), fftshift(abs(fft(x))),
xlabel('f (Hz)'), ylabel('fftshift(abs(fft(x)))')
figure(7), plot(t(1:100), y(1:100)), xlabel('nTs (s)'), ylabel('y[n]')
figure(8), plot(linspace(-Fs/2,Fs/2,length(y)), fftshift(abs(fft(y))),
xlabel('f (Hz)'), ylabel('fftshift(abs(fft(y)))')
% afisarea caracteristicii de frecventa
figure(9), freqz(B,A,[linspace(-Fs/2,Fs/2,length(y))],Fs)
```

5. Să se filtreze semnalul de la problema anterioară astfel încât sinusoida cu frecvența de 200Hz să fie rejectată. Să se afișeze semnalul $y[n]$ (obținut în urma filtrării) și spectrul acestuia.

6. Să se încarce semnalul *zgomot.wav*.












- să se reprezinte spectrul semnalului.
- să se identifice cele 3 frecvențe pe care este zgomotul.
- să se proiecteze un filtru *notch* care să rejecteze frecvența centrală.

Anexa 1

Realizarea unei interfețe grafice în Matlab

Obiective: prezentarea obiectelor grafice disponibile într-o interfață (GUI); realizarea unei interfețe grafice demonstrative ce utilizează obiectele grafice: *Slider*, *Push Button*, *Button Group*, *Check Box*, *Edit Text*, *Static Text*, *Pop-up Menu* și *Listbox*.

Obiectele grafice disponibile într-o interfață GUI (**G**raphical **U**ser **I**nterface) sunt prezentate pe scurt în *Tabelul 1*.

Componentă	Iconiță	Descriere
Push Button		<i>Push Button</i> generează o acțiune atunci când este apăsat. De exemplu, un buton de OK ar putea închide o fereastră de dialog.
Slider		<i>Slider</i> -ul permite parcurgerea cu pas constant a unui interval de valori. Utilizatorul poate deplasa <i>Slider</i> -ul dând <i>click</i> și apoi trăgând de <i>Slider</i> , sau prin apăsarea săgeților de la capetele <i>Slider</i> -ului. Poziția <i>Slider</i> -ului indică poziția relativă în intervalul specificat.
Check Box		<i>Check Box</i> permite selecția unei opțiuni dintr-o listă de opțiuni independente.
Radio Button		<i>Radio Button</i> permite selecția unei opțiuni. Este similar cu <i>Check Box</i> , cu observația că <i>Radio Buttons</i> se exclud reciproc în cadrul unui grup de butoane (<i>Button Group</i> ); adică la selectarea unui <i>Radio Button</i> , butonul selectat anterior se deselectionează automat.
Edit Text		<i>Edit Text</i> este o componentă care permite utilizatorului să introducă valori ce sunt interpretate ca șiruri de caractere.
Static Text		<i>Static Text</i> afișează linii de text.
Pop-Up Menu		<i>Pop-up Menu</i> deschide pentru afișare o listă de opțiuni atunci când utilizatorul dă <i>click</i> pe săgeată.
List Box		<i>List Box</i> afișează o listă de opțiuni și permite utilizatorului să selecteze una sau mai multe opțiuni.
Axes		<i>Axes</i> permite reprezentarea graficelor și imaginilor.
Panel		<i>Panel</i> aranjează componentele unui GUI în grupuri, pentru o mai bună organizare a interfeței. Un <i>Panel</i> poate avea titlu și margini.

În continuare ne propunem să realizăm o interfață funcțională ca cea din *Figura 1*. Interfața următoare este de fapt un demo, pentru a arăta cum se comportă fiecare obiect grafic și cum se poate programa.

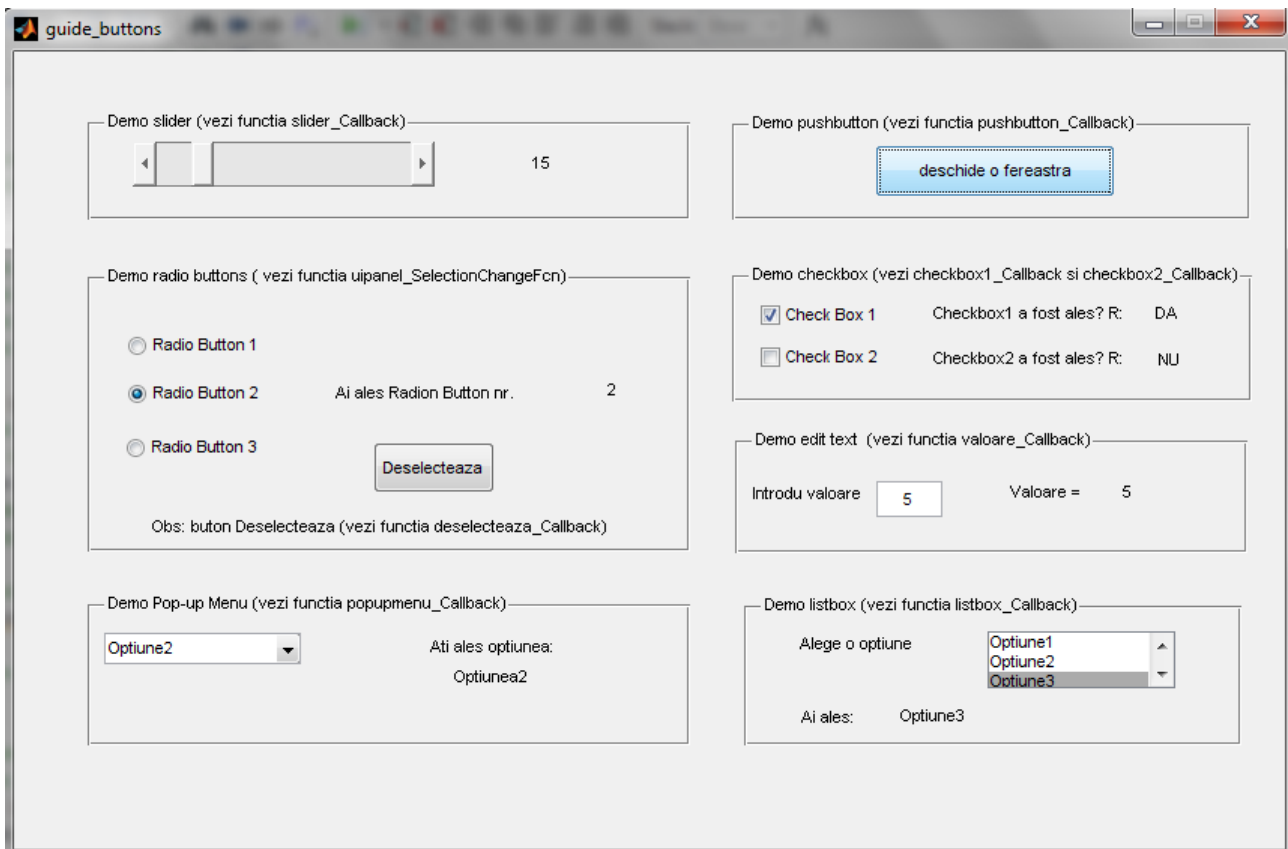


Figura 1. Interfață grafică demonstrativă în Matlab ce conține diverse obiecte grafice

În continuare vom analiza pe rând obiectele grafice folosite în interfața demonstrativă din *Figura 1*.

1. Slider. *Tag = slider.* Slider-ul are asociată o componentă de tip *Static Text* cu *Tag*-ul *val_slider*, în care se afișează valoarea selectată. Conținutul funcției *slider_Callback* este următorul:

```
function slider_Callback(hObject, eventdata, handles)
val = round(get(handles.slider, 'Value'));
% se salveaza in variabila val valoarea selectata cu Sliderul
set(handles.val_slider, 'String', val);
% se afiseaza in campul de Static Text valoarea variabilei val
```

2. Push Button. *Tag = pushbutton.* La apăsarea butonului se deschide o fereastră cu mesajul *'Acest buton deschide o fereastră'* și cu titlul *'Demo pushbutton'*. Funcția *pushbutton_Callback* asociată butonului conține următoarele linii de cod:

```
function pushbutton_Callback(hObject, eventdata, handles)
msgbox('Acest buton deschide o fereastră', 'Demo pushbutton')
```


3. Button Group. *Tag = uipanel.* Conține 3 butoane având *Tag*-urile *radiobutton1*, *radiobutton2* și *radiobutton3*. Grupului de butoane îi este asociată o componentă de tip *Static Text* cu *Tag*-ul *val_button* în care se va afișa indicele butonul radio selectat. Funcția *uipanel_SelectionChangeFcn* asociată grupului de butoane este următoarea.

```
function uipanel_SelectionChangeFcn(hObject, eventdata, handles)
content = get(hObject, 'Tag')
% valorile posibile pentru variabila content sunt tagurile celor 3 butoane:
% radiobutton1, radiobutton2 si radiobutton3
switch (content)
    case 'radiobutton1'
        set(handles.val_button, 'String', 1);
        % daca se selecteaza radiobutton1 se afiseaza valoarea 1
    case 'radiobutton2'
        set(handles.val_button, 'String', 2);
        % daca se selecteaza radiobutton1 se afiseaza valoarea 2
    case 'radiobutton3'
        set(handles.val_button, 'String', 3);
        % daca se selecteaza radiobutton1 se afiseaza valoarea 3
    otherwise
        set(handles.val_button, 'String', 0);
end
```

Observație! Nu toate versiunile de Matlab generează în mod automat funcția *_SelectionChangeFcn*. Pentru a insera această funcție în fișierul *M-file* se procedează astfel: în fișierul **.fig* se dă click dreapta pe grupul de butoane, apoi *View Callbacks/SelectionChangeFcn*.

4. Check Box. Sunt 2 obiecte *Check Box* utilizate (*Check Box1* și *Check Box2*). Vom analiza obiectul cu numele *Check Box1*. *Tag = checkbox1* și are asociată componenta de tip *Static Text* cu *Tag*-ul *val_check1*, în care se va scrie 'DA', dacă *Check Box1* a fost selectat; 'NU' dacă *Check Box1* a fost deselectat. Valoarea implicită este '?'. Codul funcției *checkbox1_Callback* este:

```
function checkbox1_Callback(hObject, eventdata, handles)
val = get(handles.checkbox1, 'Value') % variabila val poate lua valorile:
% 0 ==> daca s-a deselectat checkbox1
% 1 ==> daca s-a selectat checkbox1
if (val == get(handles.checkbox1, 'Max'))
    set(handles.val_check1, 'String', 'DA');
    % daca val = 1 se afiseaza 'DA', adica s-a selectat checkbox1
else
    set(handles.val_check1, 'String', 'NU');
    % daca val = 0 se afiseaza 'NU', adica s-a deselectat checkbox1
end
```

5. Edit Text. *Tag = valoare.* Valoarea introdusă în câmpul de *Edit Text* este afișată în câmpul *Static Text* alăturat, cu *Tag*-ul *val_edit*. Codul funcției *valoare_Callback* este următorul.

```
function valoare_Callback(hObject, eventdata, handles)
val = str2double(get(handles.valoare, 'String'));
% se salveaza in variabila val valoarea introdusa in campul de Edit Text
% inainte de a fi salvata, valoarea este convertita din string in double
% folosind functia str2double
set(handles.val_edit, 'String', val);
% se afiseaza in camp de Static Text valoarea variabilei val
```

6. Pop-up Menu. *Tag = popupmenu.* Conține 3 opțiuni (*Opțiune1, Opțiune2 și Opțiune3*). Opțiunea selectată va fi afișată în câmpul *Static Text* alăturat având *Tag*-ul *val_menu*. Codul funcției *popupmenu_Callback* este cel de mai jos.

```
function popupmenu_Callback(hObject, eventdata, handles)
index_selected = get(handles.popupmenu, 'Value')
% variabila index_selected poate lua valorile:
% 1==> daca nu s-a ales nimic (practic s-a selectat 'Alege o optiune')
% 2==> daca s-a selectat "Optiune1"
% 3==> daca s-a selectat "Optiune2"
% 4==> daca s-a selectat "Optiune3"
switch index_selected
    case 1
        set(handles.val_menu, 'String', '?');
        % daca s-a selectat 'Alege o optiune' se va afisa '?'
    case 2
        set(handles.val_menu, 'String', 'Optiunea1');
        % daca s-a selectat Optiune1, se afiseaza 'Optiunea1'
    case 3
        set(handles.val_menu, 'String', 'Optiunea2');
        % daca s-a selectat Optiune2, se afiseaza 'Optiunea2'
    case 4
        set(handles.val_menu, 'String', 'Optiunea3');
        % daca s-a selectat Optiune3, se afiseaza 'Optiunea3'
end
```

7. Listbox. *Tag = listbox.* Conține 3 opțiuni (*Opțiune1, Opțiune2 și Opțiune3*). Opțiunea selectată va fi afișată în câmpul *Static Text* alăturat având *Tag*-ul *val_listbox*. Codul funcției *listbox_Callback* este următorul.

```
function listbox_Callback(hObject, eventdata, handles)
index_selected = get(handles.listbox, 'Value')
% variabila index_selected poate lua valorile:
% 1==> daca s-a selectat "Optiune1"
% 2==> daca s-a selectat "Optiune2"
% 3==> daca s-a selectat "Optiune3"

contents = get(handles.listbox, 'String')
% variabila contents este o variabila de tip cell
% contents contine cele 3 optiuni {'Optiune1', 'Optiune2', 'Optiune3'}

set(handles.val_listbox, 'String', contents{index_selected});
% daca se selecteaza "Optiune2", atunci index_selected = 2
% se afiseaza contents{2}, adica Optiune2
```

Anexa 2

Citirea și redarea semnalelor în Matlab

Obiective: citirea și redarea semnalelor în Matlab (imagine și semnal audio); salvarea variabilelor în fișiere *.mat; încărcarea variabilelor din fișiere *.mat.

1. Citirea și redarea unui semnal audio WAVE

Pentru a citi un semnal audio *.wav, se folosește funcția wavread.

Sintaxă: [Y, Fs, NBITS] = WAVREAD(FILE)

Y = semnalul citit

Fs = frecvența cu care a fost eșantionat semnalul

NBITS = număr biți/eșantion

Exemplu 1. Citirea semnalului zgomot.wav.

```
>>[Y,Fs,NBITS] = wavread('zgomot');
```

Se obțin următoarele variabile:

- semnalul Y, de dimensiune 510876x2 (510876 linii și 2 coloane, deoarece sunt două canale)
- Fs = 44100Hz
- NBITS = 16 biți = 2 octeți

Eșantioanele semnalului zgomot.wav sunt reprezentate pe 2.043.504 octeți (2·510876·2).

Dacă ne uităm însă în proprietățile fișierului zgomot.wav, observăm că Size = 2.043.548 octeți.

Diferența de 44 octeți o reprezintă header-ul semnalului *.wav, în care sunt salvate diverse informații despre fișier (Fs, NBITS etc).

Numărul de octeți ai unui fișier WAVE se determină astfel:

$$nr_canale \cdot nr_eșantioane_per_canal \cdot nr_octeți_per_eșantion + 44octeți$$

Pentru a reda un semnal audio, se folosește funcția sound. **Sintaxă:** SOUND(Y, Fs)

2. Citirea și afișarea unei imagini în Matlab

Pentru a citi o imagine în Matlab se poate utiliza funcția imread.

Sintaxă: imagine = IMREAD(nume_imagine, extensie)

Exemplu 2. Citirea unei imagini cu numele desert și extensia jpg.

```
>>imagine = imread('desert','jpg');
```

Pentru a afișa o imagine se pot folosi funcțiile: image, imshow etc.

Exemplu 3. Afișarea iamginii citite anterior.

```
>>image(imagine)
```

3. Salvarea în fișiere *.mat a variabilelor din Matlab

Pentru salvarea variabilelor în fișiere *.mat se folosește funcția save.

Sintaxă: save(nume_fisier, var_1, var_2, ..., var_n)

Exemplu 4. pentru a salva variabilele Y și Fs din *Exemplu 1* într-un fișier numit *parametri.mat*, vom scrie:

```
>> save('parametri','Y','FS')
```

4. Încărcarea variabilelor din fișiere mat-file

Pentru a încărca variabilele dintr-un fișier *.mat, se folosește funcția load.

Sintaxă: load(nume_fisier)

Exemplu 5. Încărcarea semnalului *parametri.mat* salvat în *Exemplu 4*.

```
>> load('parametri')
```

Se observă că în *Workspace* s-au încărcat variabilele Y și Fs .

Exemplu 6. Realizarea unui interfețe grafice care permite încărcarea unei imagini și a unui fișier audio WAVE, de la o adresă ce poate fi selectată.

Interfața conține două butoane cu *Tag*-urile *image* și *sunet*. Când se apasă butonul de încărcare al imaginii/sunetului apare fereastra *Select File to Open*, de unde se poate alege calea către imaginea/sunetul dorit.

Dacă se dorește încărcarea unei imagini sau a unui fișier audio WAVE de la o adresă ce poate fi selectată, se folosește funcția `uigetfile`.

Funcțiile asociate celor două butoane sunt:

```
function imagine_Callback(hObject, eventdata, handles)
[nume cale] = uigetfile({'*.jpg; *.bmp; *.tiff'});
% sunt vizibile numai imaginile cu extensiile 'jpg','bmp','tiff'
imagine = imread(strcat(cale,nume));
axes(handles.axes1);
image(imagine), title(nume)
```

```
function sunet_Callback(hObject, eventdata, handles)
[nume cale] = uigetfile({'*.wav'});
sunet = wavread(strcat(cale,nume));
axes(handles.axes2);
plot(sunet(:,1)), zoom xon, title(nume)
```

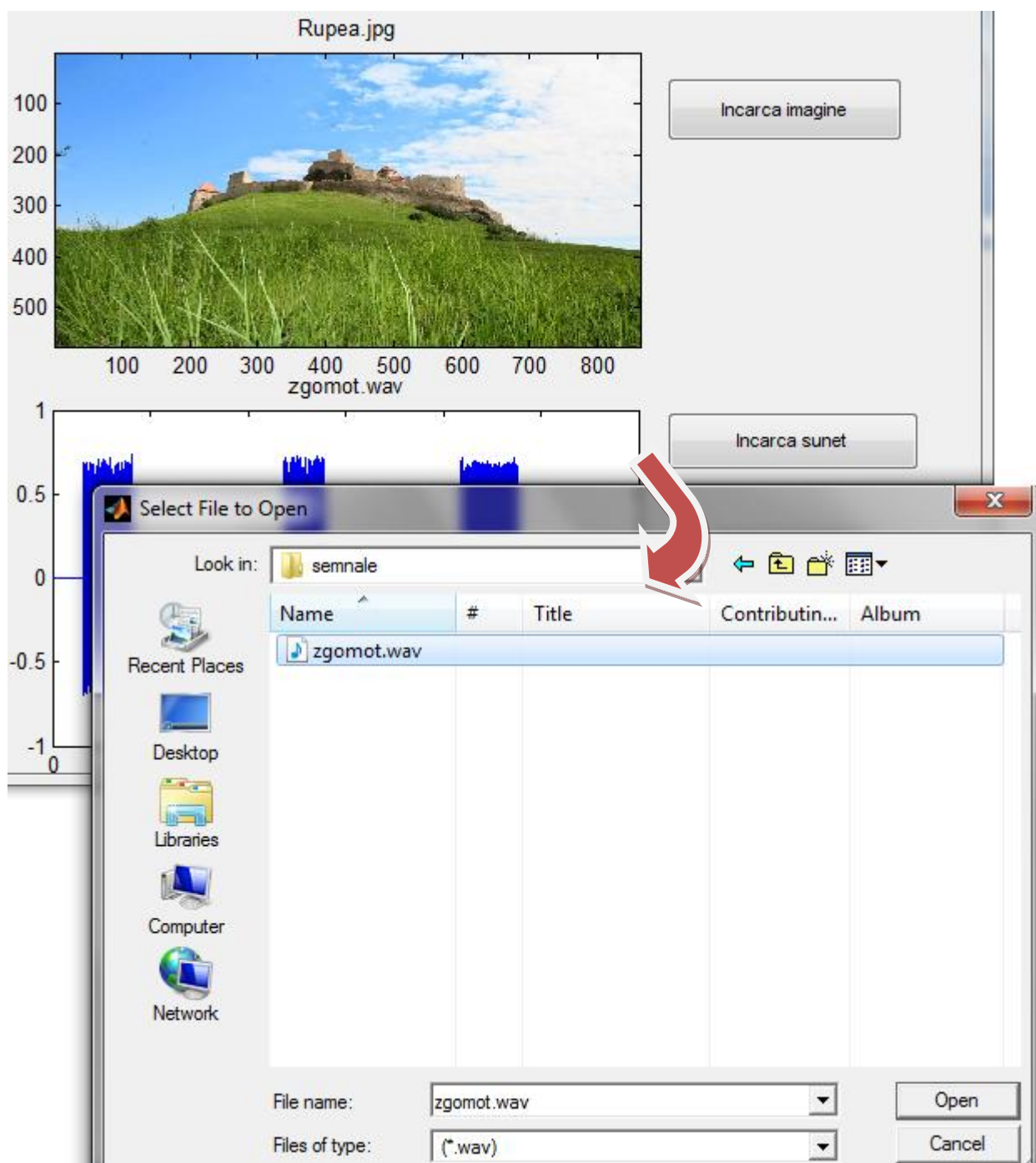


Figura 1. Încărcarea unui sunet de la o adresă selectată

Anexa 3

Construcția caracteristicii de frecvență

Obiective: construcția caracteristicii de frecvență pentru filtrele FTJ, FTS, FTB și FOB folosind TFTDI; compararea caracteristicilor reale și ideale de amplitudine.

Reamintim din *Lucrarea 11*:

Pașii necesari pentru realizarea filtrelor FIR folosind TFTDI sunt:

- Se alege tipul de filtru dorit (FTJ, FTS, FTB, FOB) precizând: numărul de coeficienți N , frecvența de eșantionare F_s , frecvența de tăiere F_t (pentru FTJ și FTS), frecvențele de tăiere F_{t1} și F_{t2} (pentru FOB și FTB).
- Se folosesc *relațiile 1, 3, 4 și 5* pentru a calcula coeficienții filtrului.
- Se calculează caracteristica reală de amplitudine (calculând TFD a coeficienților filtrului) și se compară cu cea ideală. Dacă rezultatul nu este mulțumitor, se schimbă parametrii filtrului și se recalculează coeficienții.

În continuare vom implementa caracteristica ideală și cea reală de amplitudine pentru fiecare dintre cele 4 filtre (FTJ, FTS, FTB, FOB).

1. Filtru Trece Jos (FTJ)

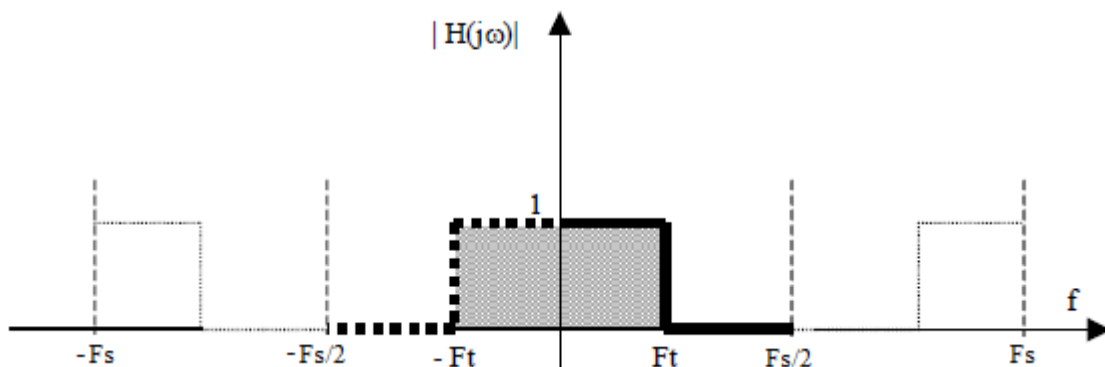


Figura 1. Caracteristica ideală de amplitudine a unui FTJ

Formula de calcul a coeficienților:

$$h[n] = 2 \cdot \frac{F_t}{F_s} \cdot \text{sinc} \left(n \cdot 2\pi \cdot \frac{F_t}{F_s} \right) \quad (1)$$

Pentru un FTJ, având $F_t = 1000\text{Hz}$, $F_s = 5000\text{Hz}$ și numărul de coeficienți $N = 101$, codul Matlab de implementare a caracteristicii ideale de amplitudine și a caracteristicii reale de amplitudine este următorul.

```

% ***** caracteristica de amplitudine pentru FTJ *****
% *****
Fs = 5000;
Ft = 1000;
N = 101;
% caracteristica ideala de amplitudine
H_ideal(-Fs/2+Fs/2+1:-Ft+Fs/2+1) = 0;
H_ideal(-Ft+Fs/2+1:Ft+Fs/2+1) = 1;
H_ideal(Ft+Fs/2+1:Fs/2+Fs/2+1) = 0;
% aflare coeficienti FTJ
for n = (-(N-1)/2):(N-1)/2
    h(n + (N-1)/2 + 1) = 2*Ft/Fs*sinc(n*2*Ft/Fs);
end
% constructie H real versus H ideal (FTJ)
H_real = fft(h);
figure(1)
hold on
plot(linspace(-Fs/2, Fs/2, length(H_ideal)), H_ideal, 'r')
plot(linspace(-Fs/2, Fs/2, length(H_real)), fftshift(abs(H_real)))
plot([0 0], [0 1.2], 'g')
title('Caracteristica reala si cea ideala de amplitudine pentru FTJ')
hold off

```

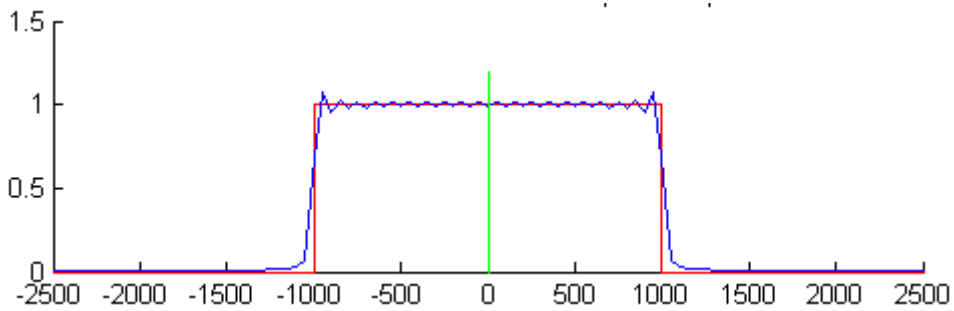


Figura 2. Caracteristica reală(cu albastru) și cea ideală(cu roșu) de amplitudine pentru FTJ

2. Filtru Trece Sus (FTS)

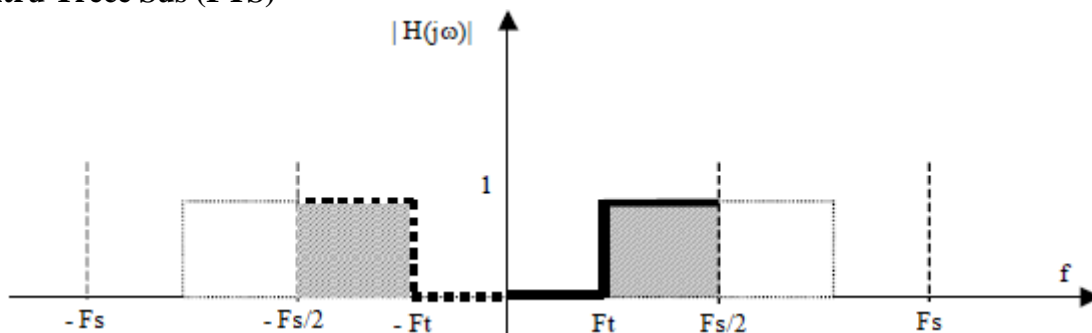


Figura 3. Caracteristica ideală de amplitudine a unui FTS

Formula de calcul a coeficienților:

$$h[n] = \text{sinc}(\pi n) - 2 \cdot \frac{F_t}{F_s} \cdot \text{sinc}\left(n \cdot 2\pi \cdot \frac{F_t}{F_s}\right) \quad (2)$$

Pentru un FTS, având $F_t = 1000\text{Hz}$, $F_s = 5000\text{Hz}$ și numărul de coeficienți $N = 101$, codul Matlab de implementare a caracteristicii ideale de amplitudine și a caracteristicii reale de amplitudine este următorul.

```

% ***** caracteristica de amplitudine pentru FTS *****
% *****
Fs = 5000;
Ft = 1000;
N = 101;
% caracteristica ideala de amplitudine
H_ideal(-Fs/2+Fs/2+1:-Ft+Fs/2+1) = 1;
H_ideal(-Ft+Fs/2+1:Ft+Fs/2+1) = 0;
H_ideal(Ft+Fs/2+1:Fs/2+Fs/2+1) = 1;
% aflare coeficienti FTS
for n = -(N-1)/2:(N-1)/2
    h(n + (N-1)/2 + 1) = sinc(n)-2*Ft/Fs*sinc(n*2*Ft/Fs);
end
% constructie H real versus H ideal (FTS)
H_real = fft(h);
figure(2)
hold on
plot(linspace(-Fs/2, Fs/2, length(H_ideal)), H_ideal, 'r')
plot(linspace(-Fs/2, Fs/2, length(H_real)), fftshift(abs(H_real)))
plot([0 0], [0 1.2], 'g')
title('Caracteristica reala si cea ideala de amplitudine pentru FTS')
hold off

```

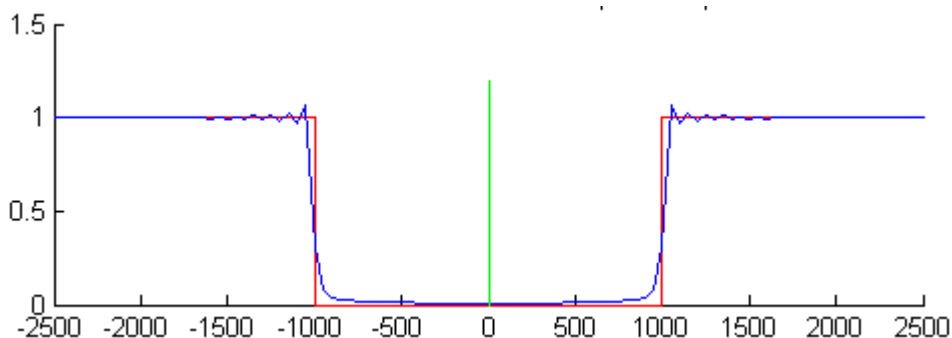


Figura 4. Caracteristica reală (cu albastru) și cea ideală (cu roșu) de amplitudine pentru FTS

3. Filtru Trece Bandă (FTB)

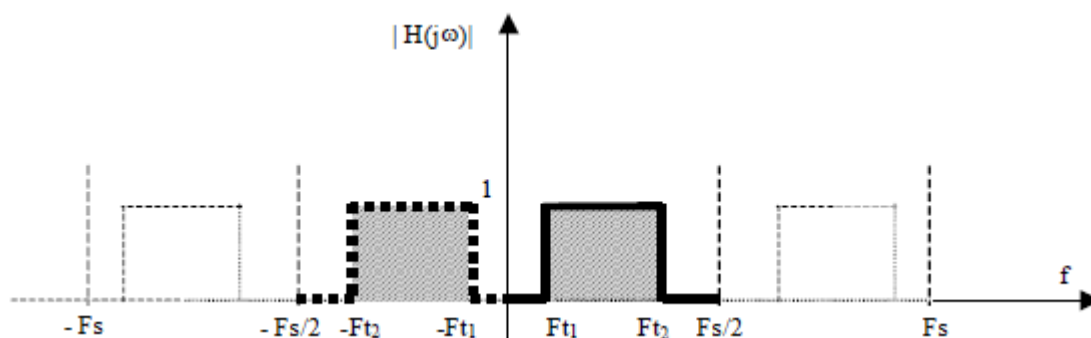


Figura 5. Caracteristica ideală de amplitudine a unui FTB

Formula de calcul a coeficienților:

$$h[n] = 2 \cdot \frac{F_{t2}}{F_s} \cdot \text{sinc}\left(n \cdot 2\pi \cdot \frac{F_{t2}}{F_s}\right) - 2 \cdot \frac{F_{t1}}{F_s} \cdot \text{sinc}\left(n \cdot 2\pi \cdot \frac{F_{t1}}{F_s}\right) \quad (3)$$

Pentru un FTB, având $F_{t1} = 500\text{Hz}$, $F_{t2} = 1500\text{Hz}$, $F_s = 5000\text{Hz}$ și numărul de coeficienți $N = 101$, codul Matlab de implementare a caracteristicii ideale de amplitudine și a caracteristicii reale de amplitudine este următorul.

```

% ***** caracteristica de amplitudine pentru FTB *****
% *****
Fs = 5000;
Ft1 = 500;
Ft2 = 1500;
N = 101;
% caracteristica ideala de amplitudine
H_ideal(-Fs/2+Fs/2+1:-Ft1+Fs/2+1) = 0;
H_ideal(-Ft2+Fs/2+1:-Ft1+Fs/2+1) = 1;
H_ideal(-Ft1+Fs/2+1:Ft1+Fs/2+1) = 0;
H_ideal(Ft1+Fs/2+1:Ft2+Fs/2+1) = 1;
H_ideal(Ft2+Fs/2+1:Fs/2+Fs/2+1) = 0;
% aflare coeficienti FTB
for n = -(N-1)/2:(N-1)/2
    h(n + (N-1)/2 + 1) = 2*Ft2/Fs*sinc(n*2*Ft2/Fs)-2*Ft1/Fs*sinc(n*2*Ft1/Fs);
end
% constructie H real versus H ideal (FTB)
H_real = fft(h);
figure(3)
hold on
plot(linspace(-Fs/2, Fs/2, length(H_ideal)), H_ideal, 'r')
plot(linspace(-Fs/2, Fs/2, length(H_real)), fftshift(abs(H_real)))
plot([0 0], [0 1.2], 'g')
title('Caracteristica reala si cea ideala de amplitudine pentru FTB')
hold off

```

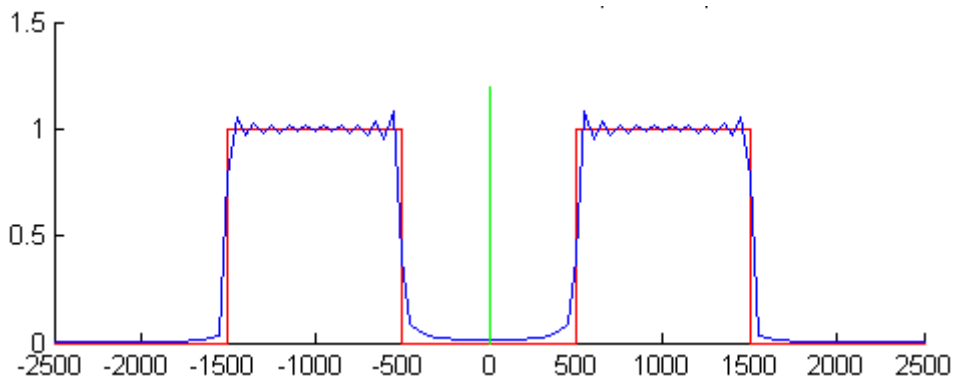


Figura 6. Caracteristica reală (cu albastru) și cea ideală (cu roșu) de amplitudine pentru FTB

4. Filtru Oprește Bandă

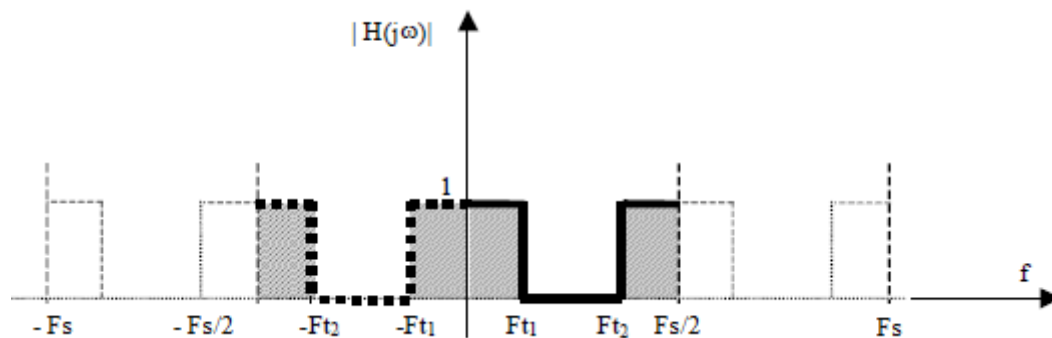


Figura 7. Caracteristica ideală de amplitudine a unui FOB

Formula de calcul a coeficienților:

$$h[n] = \text{sinc}(\pi n) - 2 \cdot \frac{F_{t2}}{F_s} \cdot \text{sinc}\left(n \cdot 2\pi \cdot \frac{F_{t2}}{F_s}\right) + 2 \cdot \frac{F_{t1}}{F_s} \cdot \text{sinc}\left(n \cdot 2\pi \cdot \frac{F_{t1}}{F_s}\right) \quad (4)$$

Pentru un FTB, având $F_{t1} = 500\text{Hz}$, $F_{t2} = 1500\text{Hz}$, $F_s = 5000\text{Hz}$ și numărul de coeficienți $N = 101$, codul Matlab de implementare a caracteristicii ideale de amplitudine și a caracteristicii reale de amplitudine este următorul.

```
% ***** caracteristica de amplitudine pentru FOB *****
Fs = 5000;
Ft1 = 500;
Ft2 = 1500;
N = 101;
% *****
% caracteristica ideala de amplitudine
H_ideal(-Fs/2+Fs/2+1:-Ft1+Fs/2+1) = 1;
H_ideal(-Ft2+Fs/2+1:-Ft1+Fs/2+1) = 0;
H_ideal(-Ft1+Fs/2+1:Ft1+Fs/2+1) = 1;
H_ideal(Ft1+Fs/2+1:Ft2+Fs/2+1) = 0;
H_ideal(Ft2+Fs/2+1:Fs/2+Fs/2+1) = 1;

% aflare coeficienti FOB
for n = -(N-1)/2:(N-1)/2
    h(n + (N-1)/2 + 1) = sinc(n) - 2*Ft2/Fs*sinc(n*2*Ft2/Fs) +
    2*Ft1/Fs*sinc(n*2*Ft1/Fs);
end

% constructie H real versus H ideal (FOB)
H_real = fft(h);
figure(4)
hold on
plot(linspace(-Fs/2, Fs/2, length(H_ideal)), H_ideal, 'r')
plot(linspace(-Fs/2, Fs/2, length(H_real)), fftshift(abs(H_real)))
plot([0 0], [0 1.2], 'g')
title('Caracteristica reala si cea ideala de amplitudine pentru FOB')
hold off
```

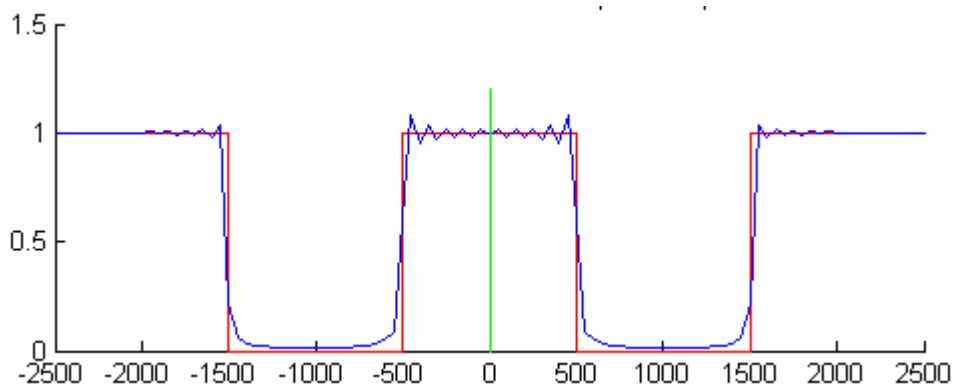



Figura 8. Caracteristica reală (cu albastru) și cea ideală (cu roșu) de amplitudine pentru FOB

Anexa 4

Proiectarea filtrelor FIR folosind aplicația *DSP_Assistant*

Obiective: folosirea aplicației *DSP_Assistant* pentru proiectarea diverselor filtre FIR (folosind TFTDI și TFDI); filtrarea semnalelor audio folosind coeficienții generați.

Se deschide aplicația *DSP_Assistant* (se selectează în *Current Directory* calea către aplicație, iar în *Command Window* se scrie `>>Main_Filters`). Se selectează butonul . Se va alege apoi filtrul FIR.

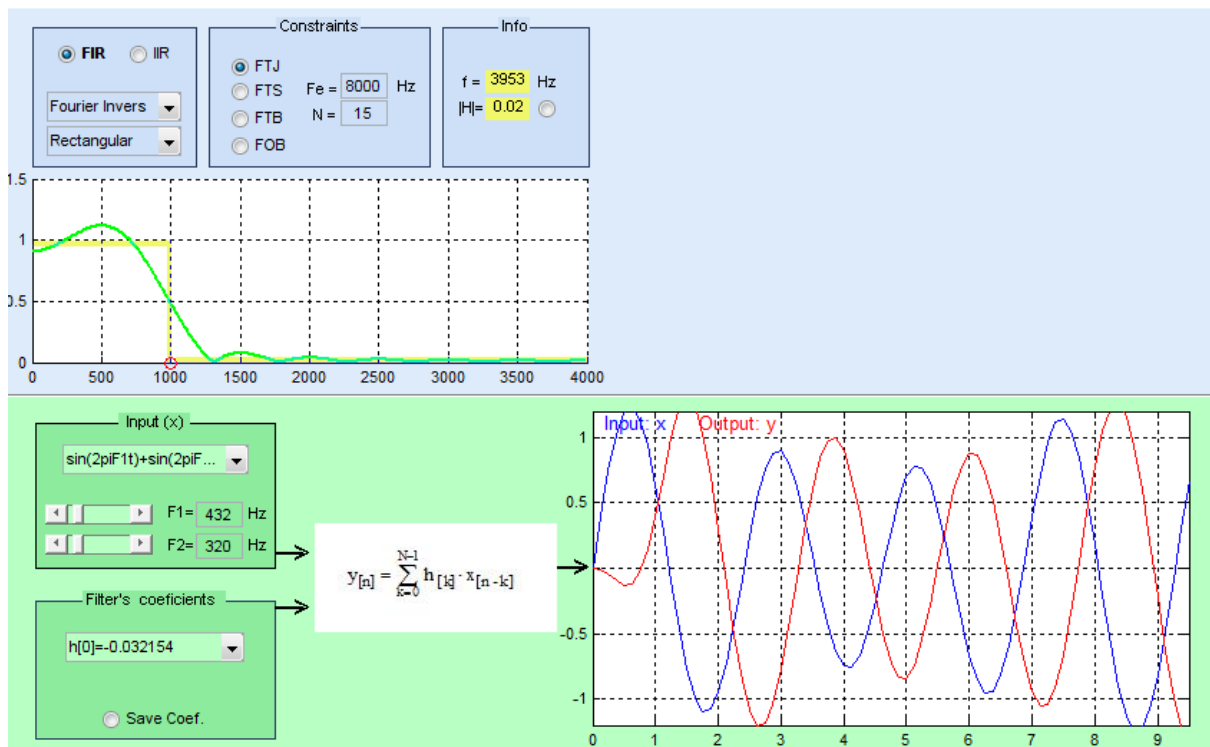


Figura 1. Aplicația *DSP_Assistant*. Metoda *Fourier Invers*

În continuare se pot proiecta filtre FIR folosind atât metoda TFDI cât și TFDI.

1. Metoda Fourier Invers (adică metoda TFTDI)

- Se pot folosi filtrele FTJ, FTS, FTB, FOB.
- Frecvența de tăiere se modifică prin modificarea poziției punctului roșu de pe caracteristica de amplitudine. Cu galben este reprezentată grafic caracteristica ideală de amplitudine iar cu verde este caracteristica reală de amplitudine.
- Se pot modifica parametrii F_e (frecvența de eșantionare) și N (numărul de coeficienți ai filtrului).
- Se pot alege diverse semnale de intrare (semnal sinusoidal, sumă de sinusoidale, chirp, semnal treaptă, dreptunghiular, triunghiular).
- Sunt generați cei N coeficienți ai filtrului care se pot apoi salva.

Aplicație 1. Se vor selecta parametrii: $F_t = 500\text{Hz}$, $N = 49$, $F_s = 8000\text{Hz}$. Se va alege ca semnal de intrare o sumă de sinusoidale cu frecvențele $F_1 = 200\text{Hz}$ și $F_2 = 700\text{Hz}$ și se va filtra semnalul folosind un filtru FTS. Cum va arăta semnalul la ieșirea din filtru? Se va folosi apoi un filtru FTJ. Ce se observă?

Aplicație 2. Să se încarce în Matlab fișierul *semnal.mat* și să se asculte semnalul. Care este mesajul? Să se reprezinte spectrul semnalului și să se identifice pe ce frecvențe este zgomotul. Folosind aplicația *DSP_Assistant* să se proiecteze un filtru (să se calculeze coeficienții filtrului) astfel încât zgomotul de fundal să fie diminuat. Folosind coeficienții rezultați să se filtreze semnalul original.

2. Metoda Fourier Discret (adică metoda TFDI)

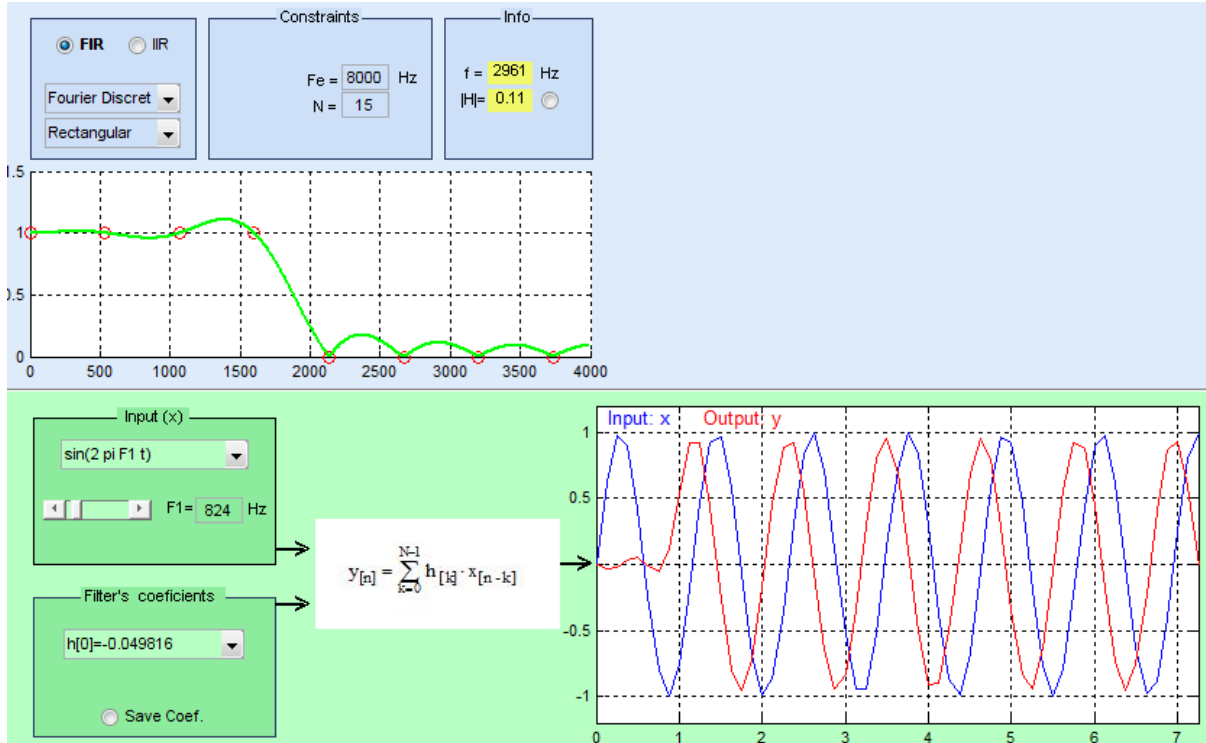


Figura 2. Aplicația *DSP_Assistant*. Metoda *Fourier Discret*

Caracteristici:

- Se poate modifica valoarea lui N (numărul coeficienților filtrului)
- Caracteristica de amplitudine se construiește prin modificarea pozițiilor celor $(N+1)/2$ cercelete roșii (dacă N este impar) sau $N/2$ cercelete roșii dacă N este par.

Aplicație 3. Se va alege ca semnal de intrare o sumă de sinusoidale cu frecvențele $F_1 = 200\text{Hz}$ și $F_2 = 700\text{Hz}$. Să se proiecteze un filtru care să lase să treacă doar sinusoida de frecvență F_1 .

Aplicație 4. Să se încarce în Matlab fișierul *semnal.mat* și să se asculte semnalul. Care este mesajul? Să se reprezinte spectrul semnalului și să se identifice pe ce frecvențe este zgomotul. Folosind aplicația *DSP_Assistant* să se proiecteze un filtru (să se calculeze coeficienții filtrului) astfel încât zgomotul de fundal să fie diminuat. Folosind coeficienții rezultați să se filtreze semnalul original.

Anexa 5

Proiectarea filtrelor IIR folosind aplicația *DSP_Assistant*

Obiectiv: folosirea aplicației *DSP_Assistant* pentru proiectarea filtrelor IIR folosind metoda poli-zerouri.

Se deschide aplicația *DSP_Assistant* (se selectează în *Current Directory* calea către aplicație, iar în *Command Window* se scrie `>> Filtre_IIR_Metoda_Poli_Zerouri`).

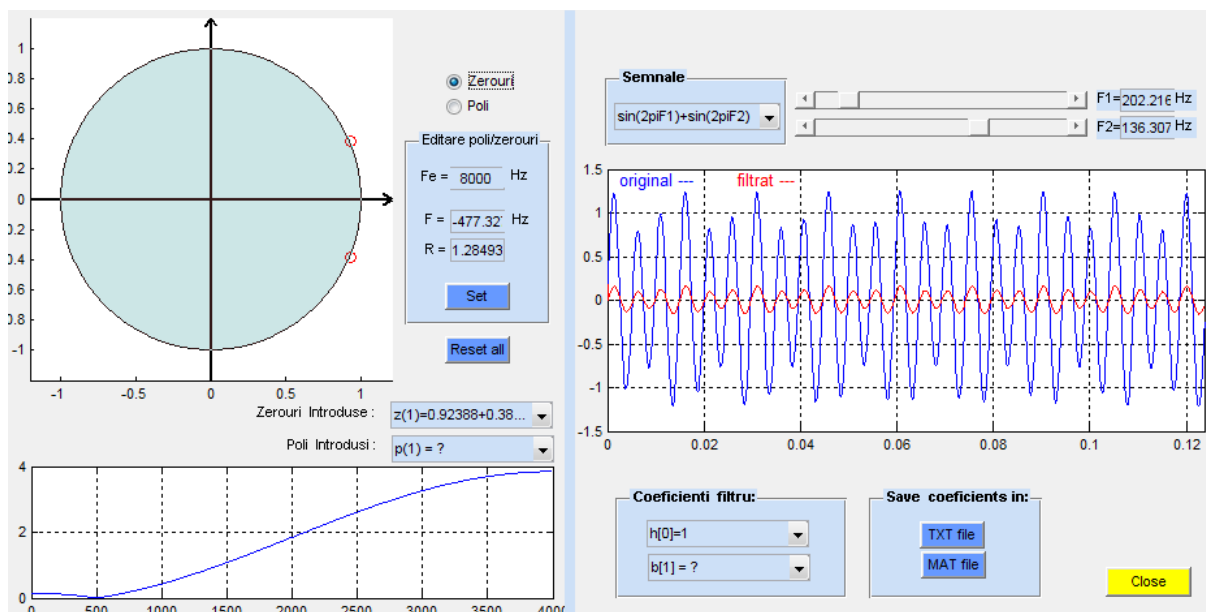


Figura 1. Aplicația *DSP_Assistant*. Proiectare IIR folosind metoda poli & zerouri

Se poate construi astfel caracteristica de frecvență prin alegerea poliilor și zerourilor.

- Pentru a plasa poliile și zerourile, sunt 2 opțiuni:

1) Click pe suprafața în care este reprezentat planul z . Cu cerculețe sunt marcate zerourile iar cu steluțe sunt poliile.

2) Scriind în câmpul de *Edit Text* valoarea dorită a frecvenței F și a razei R și apoi apăsând *Set*.

- Toate zerourile și toți poliile pot fi șterși apăsând butonul *Reset all*.
- Frecvența de eșantionare F_e poate fi modificată din câmpul de *Edit Text*.

- Coeficienții filtrului (h = coeficienții lui x și b = coeficienții lui y) se pot salva într-un fișier *.txt sau *.mat.
- Se pot selecta diverse semnale de intrare: sumă de sinusoidă, semnal dreptunghiular, triunghiular etc. Se reprezintă grafic atât semnalul original cât și cel obținut în urma filtrării.

Construcția unui filtru *notch* folosind metoda *poli-zero*

Știm că un filtru *notch* este astfel proiectat încât să rejeteze o anumită frecvență și mai știm că zerourile funcției de transfer anulează sinusoidă de anumite frecvențe. Deci pentru proiectarea unui filtru *notch* care să rejeteze o sinusoidă de frecvență F , este nevoie să plasăm un *zero* pe cercul unitate, pe frecvența F . Dacă dorim să obținem coeficienți reali, atunci este nevoie de o pereche de *zerouri* complex conjugate, ca în *Figura 2*.

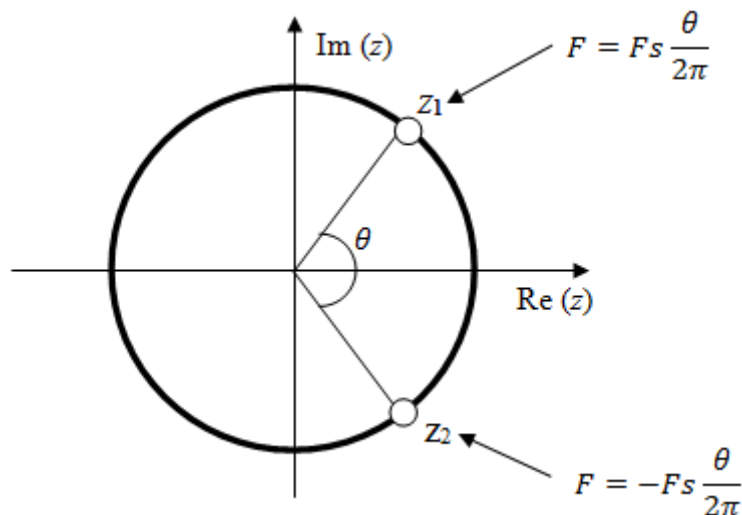


Figura 7. Plasarea a două zerouri complex conjugate pe cercul unitate

Aplicație 1. Să se proiecteze un filtru care să rejeteze frecvențele de 1000Hz și 3000Hz.

Aplicație 2. Se va alege ca semnal de intrare o sumă de sinusoidă cu frecvențele $F_1 = 200\text{Hz}$ și $F_2 = 700\text{Hz}$. Folosind metoda poli-zero să se rejeteze sinusoida cu frecvența F_2 .

Aplicație 3. Se încarcă în Matlab fișierul *zgomot.wav* și i se reprezintă spectrul. Folosind apoi aplicația *DSP_Assistant* și metoda *poli-zero*, să se determine coeficienții filtrului astfel încât să se rejeteze frecvența de 12kHz. Să se reprezinte spectrul semnalului filtrat.

Bibliografie

1. Ioan P. Mihu, *Procesarea Numerică a Semnalelor*, ISBN973-632-185, Editura Alma Mater din Sibiu, 2005
2. Alan V. Oppenheim, *Discrete –Time Signal Processing*, ISBN 0 – 13- 754920-2, Published by Prentice Hall, 1998
3. **MATLAB®** *Getting Started Guide*. © COPYRIGHT 1984–2011 by The MathWorks, Inc.
4. **MATLAB®** *Creating Graphical User Interfaces*. © COPYRIGHT 2000–2011 by The MathWorks, Inc.
5. Charles A. Bouman, *Purdue Digital Signal Processing Labs*, 2008
6. Ioan. P. Mihu, Aplicația de laborator *DSP_Assistant*