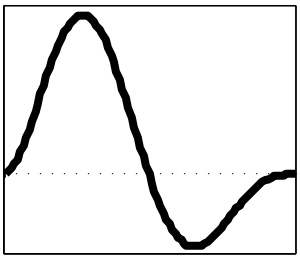
UNIVERSITATEA *TRANSILVANIA* BRAȘOV

KERTÉSZ Csaba-Zoltán **Laurențiu-Mihail IVANOVICI**

PROCESAREA DIGITALĂ A SEMNALELOR

ÎNDRUMAR DE LABORATOR



2009

Cuvânt înainte

În tehnica electronică actuală, în care o mare parte a semnalelor este transferată în domeniul numeric (digital), procedeele de prelucrare digitală a semnalelor au o deosebită importanță. Din această perspectivă la formarea specialiștilor din domeniul ingineriei electronice și a telecomunicațiilor, dar și a celor de la spscializările din domeniul de calculatoare și tehnologia informației, planul de învățământ prevede discipline sau module de discipline, care au ca subiect prelucrarea digitală a semnalelor, care are abrevierea bine cunoscută DSP (Digital Signal Processing).

Îndrumarul de laborator de față se adresează în primul rând studenților de la programele de studii ce aparțin domeniilor sus-menționate: Electronică aplicată, Tehnologii și sisteme de telecomunicații, Calculatoare, Tehnologia informației, Inginerie electrică și calculatoare, toate fiind studii de licență la Facultatea de Inginerie Electrică și Știința Calculatoarelor, la Universitatea Transilvania din Brașov.

Lucrările incluse în prezentul îndrumar se desfășoară în mediul software Matlab. Considerând că studenții nu au avut ocazia să cunoască în prealabil acest mediu software, prima lucrare (care poate avea o extensie elastică de 2-3 ședințe de laborator, în funcție de gradul de cunoaștere și/sau de avansare a formației de lucru) constituie o inițiere de bază, dacă este cazul o inițiere mai avansată, în Matlab prin exerciții și activitate individuală supravegheată.

În continuare lucrările au cursivitatea de a parcurge câteva din cele mai importante procedee de prelucrare digitală de semnale: eșantionarea, cuantizarea, convoluția, corelația, autocorelația, analiza în frecvență a semnalelor (domeniul Z, DFT, FFT), filtre digitale (analiză, sinteză). Lucrările sunt concepute astfel încât studenții să aibă posibilitatea de a lucra individual, să cunoască gradual noțiunile, să aibă câte un scurt breviar teoretic și să întocmească referatul de lucrarare.

Cu speranța că îndrumarul de laborator de față va fi util (lucrările de laborator au fost rulate de mai mulți ani), vom fi bucuroși să primim observații, sugestii de îmbunătățire pentru a crește gradul de impact și de profunzime a conoștințelor ce dorim a fi transmise studenților.

Autorii

Cuprins

Mulțumiri

Autorii doresc să mulțumească domnului profesor Iuliu Székely, conducătorul catedrei de Electronică și Calculatoare, din cadrul Facultății de Inginerie Electrică și Știința Calculatoarelor, Universitatea "Transilvania" din Brașov, pentru sprijinul acordat pentru publicarea lucrării de față.

De asemenea autorii mulțumesc studenților care au contribuit la îmbunătățirea conținutului acestui îndrumar de laborator, venind cu întrebări și comentarii pertinente de-a lungul anilor.

Lucrarea 1

Introducere în Matlab / Octave

Limbajul matlab este un limbaj de nivel înalt ce permite efectuarea calculelor matematice fără a fi nevoie de implementarea algoritmilor de calcul într-un limbaj de nivel jos. Limbajul matlab a fost creat de prof. Cleve B. Moler de la Universitatea din New Mexico pentru a permite un access ușor la bibliotecile de calcul matricial realizat în Fortran. Limbajul matlab permite realizarea calculelor matriciale printr-o formă simplă, apropiată de reprezentarea matematică [9], chiar și denumirea matlab provine din *MAT*rix *LAB*oratory.

Limbajul matlab este un limbaj interpretat, având nevoie de un interpretor pentru efectuarea propriu-zisă a calculelor. Cel mai răspândit interpretor este suita de programe **MATLAB**, oferit de firma MathWorks[1], o suită comercială, care pe lângă interpretor oferă și o sumedenie de biblioteci (numite toolbox-uri) pentru o gamă largă de domenii, printre care și procesarea semnalelor sau a imaginilor. În afară de interpretor, MATLAB integrează și toate uneltele necesare pentru crearea și rularea programelor: o interfață grafică ușor de folosit, un editor dedicat, un spațiu de lucru care oferă toate informațiile necesare despre datele folosite și interfețe pentru afișarea rezultatelor și interacțiunea cu utilizatorul.

O variantă de interpretor matlab open-source este **octave**[2] oferit prin intermediul proiectului GNU. Acest program este mai limitat decât varianta comercială, mai ales la capitolul de toolbox-uri oferite, existând totuși o serie de toolbox-uri realizate sub licența GPL colecționate prin proiectul **octave-forge**[3]. Nu există nici mediu integrat ca în cazul lui Matlab, având nevoie de programe auxiliare, cum ar fi un editor de texte de uz general pentru scrierea programelor sau gnuplot pentru afișarea rezultatelor.

Deși cele două programe nu sunt 100% compatibile, marea majoritate a programelor create într-unul din aceste programe pot fi rulate fără probleme și în celălalt. Diferențele importante, cum ar fi dotarea cu mai multe cu toolbox-uri a Matlabului, se poate sesiza numai în cazul problemelor specializate. Pentru realizarea lucrărilor de laborator din acest îndrumar sunt suficiente toolbox-urile de procesare a semnalelor și imaginilor

[1] http://www.mathworks.com
[2] http://www.gnu.org/software/octave
[3] http://octave.sourceforge.net

oferite de ambele variante de interpretor matlab.

1.1. Interpretarea programelor matlab

MATLAB este un program ce dispune de o interfață grafică care integrează câteva ferestre, printre care o fereastră *Command Window*, care oferă o linie de comandă folosită pentru executarea scripturilor matlab. Octave pe de altă parte este un program în linie de comandă ce poate fi executat într-un terminal cu ajutorul comenzii `octave`, după care prompterul terminalului va fi înlocuit cu unul similar cu acela din MATLAB.

Linia de comandă este caracterizată de un prompter care în cazul programului MATLAB este de forma:

```
>>
```

iar în cazul programului Octave:

```
octave:1>
```

În cele ce urmează acest prompter va fi indicat cu semnul:

```
»
```

După acest prompter se pot introduce comenzile matlab. Comanda introdusă este interpretată, după care este afișat răspunsul (dacă este cazul). De exemplu, comanda `pwd` va afișa calea către directorul curent:

```
» pwd
ans =
    /home/user
```

Rularea programelor matlab se face în directorul curent. De aceea directorul curent trebuie schimbat la directorul de lucru al utilizatorului (în care se află programele matlab ale acestuia):

```
» cd dsp
» pwd
ans =
    /home/user/dsp
```

Prin intermediul acestei linii de comandă pot fi executate toate operațiile oferite de limbajul matlab (care este un limbaj bazat pe linii de cod tocmai din această cauză), respectiv lansate programele scrise în limbajul matlab și stocate în fișiere aflate în directorul curent.

Pe lângă această linie de comandă, octave oferă și o integrare cu interpretorul de comenzi din Linux (bash) prin posibilitatea rulării scripturilor care încep în forma consacrată a scripturilor: `#!/usr/bin/octave`

Pentru a părăsi mediul matlab se poate folosi comanda **`exit`**.

1.2. Variabile

Elementul de bază al operațiilor matlab este matricea, toate variabilele sunt considerate ca fiind de tipul matrice. Un caz particular este când matricea respectivă conține un singur element, care poate fi considerat ca o valoare scalară. De asemenea, dacă toate dimensiunile unei matrici cu excepția uneia sunt 1, atunci vorbim de un vector. Aceste cazuri sunt tratate în mod special în anumite situații: de exemplu vectorii permit accesul la elementele lor prin precizarea unei singure poziții (indiferent dacă este un vector rând sau un vector coloană), iar valorile scalare pot fi extinse în orice dimensiune pentru a ușura operațiile cu matrici.

Intern matlab lucrează implicit cu valori de tip *double* (virgulă mobilă, dublă precizie), dar în anumite condiții poate fi precizat un tip cu reprezentare pe mai puțini biți (întreg reprezentat pe 8 sau 16 biți), care însă la efectuarea unei operații va fi transformat automat la o valoare *double*. De asemenea dacă rezultatul unei operații este un număr complex, variabila va fi transformată într-o structură de tip complex, care are două valori double: partea reală și partea imaginară. Transformările automate se fac numai în direcția creșterii preciziei, de aceea o varibilă care a devenit de tip complex va rămâne de tip complex pe durata vieții (până când este ștearsă sau suprascrisă).

Definirea unei variabile se face utilizâns operatorul `=` :

```
» x = 10
x =
     10
```

Numele de variabilă poate fi format din caracterele alfanumerice și _, pe prima poziție nefiind permise cifre. După fiecare operație, Matlab afișează rezultatul operației, care în cazul atribuirii unei variabile este variabila însăși. Pentru a nu afișa rezultatul, la sfârșitul comenzii trebuie adăugat operatorul `;` :

```
» y = 2;
```

1.3. Operatori

Matlab cunoaște operațiile matematice de bază reprezentate cu simbolurile: `+` (adunare), `-` (scădere), `*` (multiplicare), `/` (împărțire la stânga), `\` (împărțire la dreapta)

respectiv ^ (ridicare la putere). Spațiile goale (spațiu sau tabulator) nu au influență asupra modul de interpretare a operatorilor. Exemplu de folosire:

```
» 1+2
ans =
    3
» a = 2.5 * 2 - 1
a =
    4
» b = a / 2;
» c = b^5
c =
    32
```

Trebuie avut grijă că aceste operații sunt definite implicit ca operații matriciale, și funcționează ca operatori obișnuiți numai în cazul valorilor scalare.

Operații cu matrici

Pentru operații cu matricii mai sunt definiți următorii operatori: [] și (). Operatorul [] permite crearea matricilor prin enumerarea elementelor acestora între parantezele pătrate, inserând câte un spațiu între valorile de pe aceași linie și un caracter ; pentru delimitarea liniilor.

Astfel, pentru a crea un vector de tip linie se folosește comanda:

```
» v1 = [1 2 3]
v1 =
    1 2 3
```

pentru crearea unui vector de tip coloană comanda:

```
» v2 = [1;1;1]
v2 =
    1
    1
    1
```

iar pentru matrici multidimensionale:

```
» m = [1 2; 1 4]
m =
    1 2
    1 4
```

Un vector poate fi generat în mod automat folosind operatorul `:`. Aceasta va genera o serie de valori, sintaxa fiind: *valoare început* `:` *increment* `:` *valoare sfârșit*. De exemplu:

```
» i = 1:2:9
i =
    1  3  5  7  9
```

Dacă incrementul nu este precizat atunci se consideră valoarea implicită de 1:

```
» i = 1:4
i =
    1  2  3  4
```

Pentru extragerea unui element al unei matrici se folosește operatorului `()`:

```
» m(1,2)
ans =
    2
```

În cazul în care operatorul `()` se folosește pentru vectori cu mai multe elemente, rezultatul va fi o matrice cu valorile din pozițiile combinate ale parametrilor:

```
» m(1,1:2)
ans =
    1   2
» m([2 1], [2 1])
ans =
    4   1
    2   1
```

Operatorul `()` se poate folosi și pentru modificarea valorilor unor elemente din matrice:

```
» m(1,1:2) = [2 3]
m =
    2   3
    1   4
```

Operațiile aritmetice prezentate anterior sunt considerate a fi operații matriciale atunci când sunt executate asupra unor matrici. Astfel de exemplu operatorul `*` este folosit pentru înmulțirea matricială. Pentru a executa operația aritmetică element cu element operatorul trebuie precedat de `.`:

```
» a=[1 2; 0 1]; b=[2 1;4 3];
» a*b
ans =
```

```
    10     7
     4     3
» a.*b
ans =
     2     2
     0     3
```

Trebuie avut grijă la dimensiunile matricilor la executarea operațiilor aritmetice. De exemplu înmulțirea matricială presupune că numărul de coloane în prima matrice este egală cu numărul de rânduri din cea de a doua matrice. La fel când se folosesc operațiile element cu element cele două matrici trebuie să aibă exact aceași dimensiune. O singură excepție există la această condiție: dacă o matrice se înmulțește cu un scalar, atunci scalarul respectiv va fi automat convertit într-o matrice având aceleași dimensiuni ca celălalt operand și se va executa o înmulțire element cu element între cei doi operanzi.

Transpusa unei matrici se poate calcula cu operatorul `'` (transpusa complex conjugată) sau `.'` (transpusa fără conjugare):

```
» m'
ans =
    2   1
    3   4
```

Alte operații importante asupra matricilor sunt realizate prin intermediul unor funcții, de exemplu **`inv`** pentru calcularea inversei unei matrici sau **`det`** pentru calcularea determinantului:

```
» inv(m)
ans =
     0.8000   -0.6000
    -0.2000    0.4000
» det(m)
ans =
    5
```

1.4. Funcții

O funcție este un set de operații efectuate asupra unor argumente, care returnează un set de valori. Funcția este apelată sub forma:

```
[r1 r2 … rn] = nume_funcție(p1, p2, … pn)
```

unde r_1 … r_n sunt valorile returnate, iar p_1 … p_n sunt parametrii functiei. Dacă funcția returnează o singură valoare, atunci parantezele drepte nu sunt necesare. Dacă nu sunt

specificate variabilele care să fie returnate, atunci dacă funcția returnează unul sau mai multe valori, prima dintre acestea va fi salvată în variabila specială **ans**, iar restul se pierd.

Fișiere .m

O secvență repetată de comenzi se poate salva într-un fișier cu extensia *.m. Aceste fișiere pot fi rulate ulterior prin introducerea numelui fișierului (fără extensie) în linia de comandă. La introducerea unei comenzi, Matlab va verifica dacă există un fișier .m în directorul curent cu numele introdus, caz în care va interpreta comenzile din acel fișier. Dacă nu se găsește fișierul în directorul curent, Matlab va mai căuta într-o serie de directoare implicite (unde sunt stocate de altfel toate funcțiile disponibile în Matlab.

Funcțiile se definesc în fișiere .m, cu același nume ca funcția respectivă. Pentru ca o funcție să fie recunoscută ca atare, prima linie de cod trebuie să conțină cuvântul cheie **function** și să aibă următorul format:

```
function [r_1 r_2 … r_n] = nume_funcție(p_1 p_2 … p_n)
```

Parantezele drepte trebuie să apară chiar dacă funcția nu returnează nici o valoare.

Dacă fișierul începe cu un comentariu sau în liniile imediat următoare după definiția funcției există un comentariu, atunci comentariul respectiv va reprezenta documentația funcției, accesibilă prin comanda **help**.

Un exemplu pentru o funcție care calculează pătratul unei valori este:

```
%patrat(a) calculează patratul lui a
function [r] = patrat(a)
    r = a*a
return
```

Funcția poate fi apelată din linia de comandă astfel:

```
» patrat(2)
ans =
    4
```

iar pentru a vedea descrierea funcției se poate executa:

```
» help patrat
patrat(a)   calculează patratul lui a
```

Fiecare funcție din Matlab are inclusă o astfel de documentație accesibilă cu funcția **help**.

Structuri de control

Pentru a crea o buclă în matlab pot fi folosite instrucțiunile **for** sau **while**. Comanda **for** va parcurge un vector atribuind unei variabile *contor* valoare fiecărui element din *vector*. Sintaxa aceste comenzi este următoarea:

```
for contor = vector
    ...
end
```

Pentru a crea o variabilă care se incrementează (similar cu alte limbaje), se poate folosi operatorul : de creare a vectorilor:

```
for contor = început : pas : sfârșit
    ...
end
```

Comanda **while** va crea o buclă care se repetă atât timp cât o condiție logică este adevărată:

```
while condiție
    ...
end
```

Evaluarea condiționată a anumitor secvențe poate fi realizată prin intermediul comenzii **if**, având următoarea sintaxă:

```
if condiție1
    ...
else if condiție2
    ...
else
    ...
end
```

ramurile **else** sunt opționale, respectiv ramura **else if** poate fi prezentă de mai multe ori.

Condițiile logice în cazul comenzilor **while** respectiv **if** trebuie să fie valori logice ce pot fi realizate prin intermediul operatorilor logici: == (egal), ~= (diferit), < (mai mic), <= (mai mic sau egal), > (mai mare), >= (mai mare sau egal), & (și logic), | (sau logic) respectiv ~ (negare).

Toate aceste structuri de control trebuie terminate cu comanda **end**, ceea ce permite ca aceste structuri să poate fi folosite atât în fișiere .m, cât și în linia de comandă, caz

în care interpretarea comenzilor introduse este suspendată până când se întâlnește comanda **end**, după care sunt realizate buclele respectiv instrucțiunile condiționate.

Trebuie remarcat faptul că matlabul fiind orientat asupra operațiilor cu matrici, aceste operații sunt optimizate și se execută mult mai rapid decât parcurgerea matricii și executarea operației pentru fiecare element în parte. Astfel o structură **for** folosită pentru parcurgerea unei matrici în vederea executării unei operații asupra elementelor sale se va executa mult mai lent decât operația matricială corespunzătoare. De aceea trebuie evitată pe cât posibil folosirea structurilor **for**

1.5. Grafice în Matlab

Pentru crearea graficelor se folosesc următoarele comenzi: **plot**, **xlabel**, **ylabel**, **title**, **grid**, **axis**, **subplot**. Comanda **plot** afișează un vector sub forma unui grafic. Sintaxa acestei comenzi este:

```
» plot(x,y)
```

care va afișa un grafic format din puncte obținute din perechi de valori din vectorii x și y. Vectorii x și y trebuie să aibă aceași dimensiune.

Pentru modificarea aspectului graficului, se pot specifica anumite opțiuni cu privire la forma, culoarea sau dimensiunea punctelor sau a liniilor ce unesc punctele graficului:

```
» plot(x,y,'option')
```

unde '*option*' este un text care poate conține culoarea, tipul de linie și/sau simbolul. Culoarea poate să fie `r` (roșu), `g` (verde), `b` (albastru), `c` (cian), `m` (magenta), `y` (galben), `w` (alb), `k` (negru). Tipul de linie poate fi `-` (linie continuă), `--` (line întreruptă), `-.` (linie punct linie), `:` (linie punctată). Linia poate conține markere pentru fiecare punct din y sub forma unor simboluri: `+`, `*`,`o`,`x`, triunghi, romb, etc.

De exemplu:

```
» plot(x,y,'r')
```

va trasa graficul cu o linie roșie

```
» plot(x,y,'b*')
```

va marca de puncte punctele (x, y) cu steluțe albastre

```
» plot(x,y,'g:')
```

va trasa un grafic cu linie punctată verde.

Dacă se dorește afișarea mai multor seturi de puncte pe același grafic, atunci funcția **plot** va primi ca argumente o listă de parametri de forma "x,y,['*option*']".

Titlul graficului se poate seta cu comanda **title**, textul afișat pe axele Ox și Oy se poate seta cu funcțiile **xlabel** respectiv **ylabel**, iar o rețea de linii ajutătoare se poate afișa cu **grid**.

Matlab scalează axele automat după valorile maxime din vectorii x și y. Dacă se dorește ca graficul să apară pe o scală specificată se folosește funcția **axis**, care are următoarea sintaxă:

```
» axis([xmin xmax ymin ymax])
```

care va scala axele între $xmin$ și $xmax$ (axa Ox) și între $ymin$ și $ymax$ (axa Oy).

Pentru a afișa mai multe grafice se folosește comanda **subplot** astfel:

```
» subplot(m,n,p),plot(x,y);
```

Aceasta va genera o serie de grafice aranjate în m rânduri și n coloane și va direcționa următoarea comandă **plot** pe poziția p (începând de la stânga sus).

Pentru a crea o suprafață 3D, trebuie să reprezentăm valorile înălțimii $z = f(x, y)$. Pentru afișarea acestei suprafețe se folosește comada **mesh** sau **surf**. Sintaxa este:

```
» mesh(x,y,z)
```

respectiv

```
» surf(x,y,z)
```

unde x, y și z sunt matrici de dimensiune $m \times n$ care reprezintă coordonatele fiecărui punct de pe suprafață. Parametrii x și y pot fi și vectori de lungime m respectiv n, caz în care se va realiza toate combinațiile între ele pentru calculul punctelor de pe suprafață. Comanda **mesh** va conecta punctele învecinate prin linii realizând un model wireframe al sufrafeței, iar comanda **surf** va desena mici suprafețe plane între punctele învecinate realizând astfel o aproximare a suprafeței.

Folosind parametrul c, al patrulea al funcției **mesh**, se pot specifica culorile punctelor de pe suprafața afișată:

```
» mesh(x,y,z,c)
```

unde c este o matrice de dimensiune $m \times n$ reprezentând valoarile culorilor punctelor din grafic. respectiv cu ajutorului unei palete de culori ce poate fi specificată cu comada **colormap**. În cazul în care c nu este specificată aceasta va lua valoarea implicită $c = z$.

Pentru a crea matricile x și y, care trebuie să aibă toate combinațiile a doi vectori reprezentând coordonatele Ox și Oy se poate folosi comanda **meshgrid**:

```
» [x y] = meshgrid(a,b)
```

unde a și b sunt doi vectori de dimensiune m resprectiv n.

Exemple de grafice

Pentru afișarea funcției $y = f(x) = x^2+x+1$ se folosește următoarea secvență rezultând graficul prezentat în figura 1.1:

```
» x = -2:0.1:2;
» y = x.^2 + x + 1;
» plot(x,y,'r');
» title('Exemplu plot');
» xlabel('x');
» ylabel('x^2+x+1');
```

Pentru afișarea unei suprafețe $z = f(x, y) = x^2 + xy + y^2$ se folosește următoarea secvență, rezultând graficul prezentat în Figura 1.2:

```
» [x y] = meshgrid(-2:0.1:2, -2:0.1:2);
» z = x.\^2 + x.*y + y.\^2;
» mesh(x,y,z);
» title('Exemplu mesh');
```

1.6. Exerciții

1. Implementați funcția `fib` care calculează recursiv al `n`-lea element din șirul lui Fibonacci definit cu formula:

$$fib(n) = \begin{cases} 1 & \text{dacă } n <= 2, \\ fib(n-1) + fib(n-2) & \text{altfel.} \end{cases}$$

   Creați o funcție care va genera în mod iterativ un vector cu șirul lui Fibonacci cu `n` elemente.

2. Rezolvați sistemul de ecuații de mai jos folosind metoda Cramer:

$$\begin{cases} 2x + y - z = 1 \\ x + 3y + z = 10 \\ x - y + 2z = 5 \end{cases}$$

   Rezolvați aceași ecuație prin împărțire matricială pornind de la scrierea matricială următoare:

$$\begin{bmatrix} 2 & 1 & -1 \\ 1 & 3 & 1 \\ 1 & -1 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 \\ 10 \\ 5 \end{bmatrix}$$

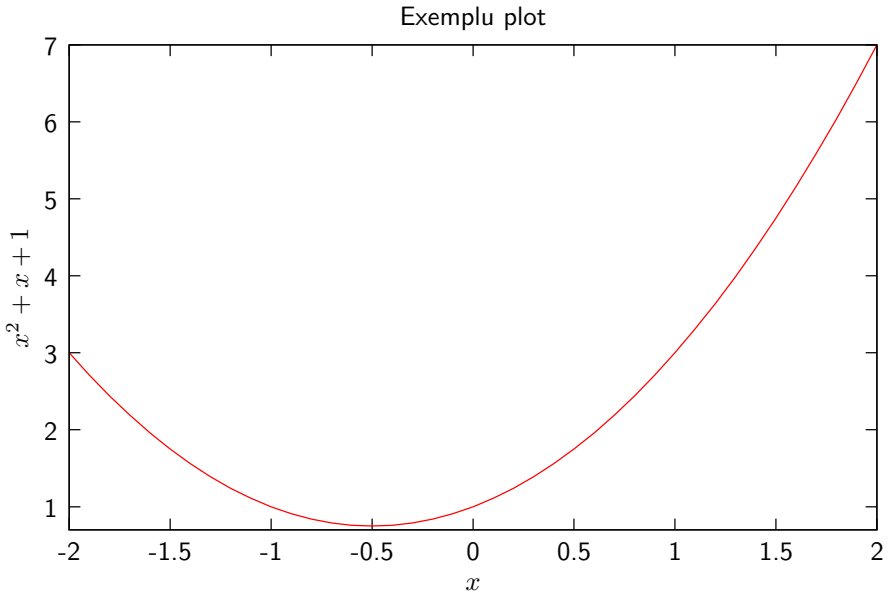


Figura 1.1. *Exemplu de grafic bidimensional*

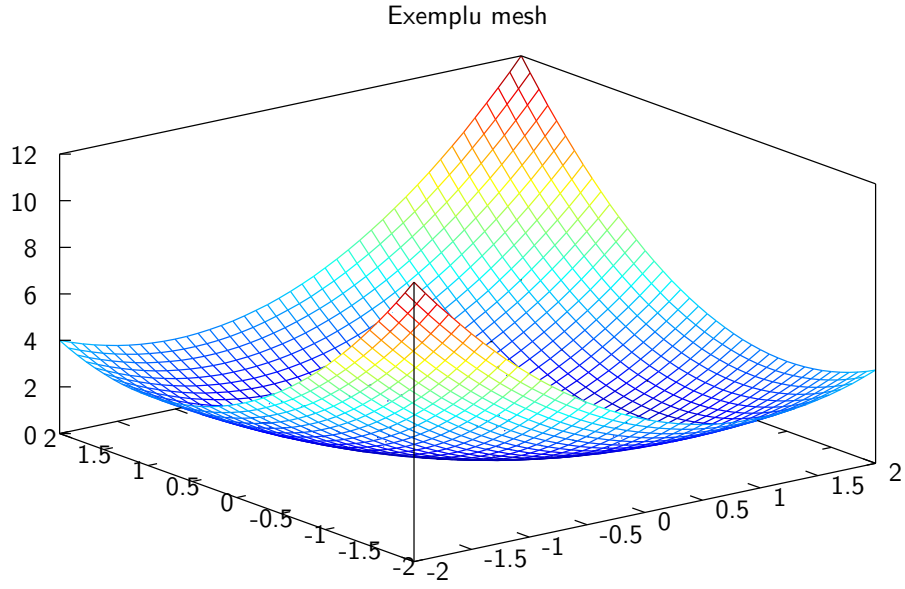


Figura 1.2. *Exemplu de grafic tridimensional*

3. Afișați o formă de undă sinusoidală definită prin formula

$$s = A \cdot sin(2 \cdot \pi \cdot f \cdot x)$$

   Modificați programul astfel încât să afișeze o formă de undă dreptunghiulară respectiv triunghiulară prin înlocuirea funcției `sin` cu `square` respectiv `sawtooth`.

4. Realizați o funcție care crează următoarele trei grafice în aceeași fereastră: o formă de undă sinusoidală, o formă de undă sinusoidală cu frecvența dublă față de prima și rezultatul însumării acestora.

Lucrarea 2

Discretizarea semnalelor

2.1. Semnale analogice și procesarea digitală

Semnalele analogice sunt semnale continue ce pot fi reprezentate din punct de vedere matematic ca funcții reale continue în timp. Pentru a putea procesa un semnal cu ajutorul unui procesor de uz general sau cu a unuia specializat[1], semnalele continue trebuie convertite într-un format potrivit cu reprezentarea datelor dintr-un procesor.

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

Discretizarea în timp poartă numele de eșantionare și se realizează cu ajutorul unui bloc de eșantionare-memorare (vezi figura 2.1). Discretizarea în amplitudine se numește cuantizare și este realizată cu un circuit de tip convertor analog-digital (CAD).

Circuitul de eșantionare-memorare va preleva valorile semnalului de intrare la anumite momente discrete de timp, iar convertorul analog-digital va converti aceste valori reale în reprezentarea lor numerică.

[1]Digital Signal Procesor – procesor digital de semnal

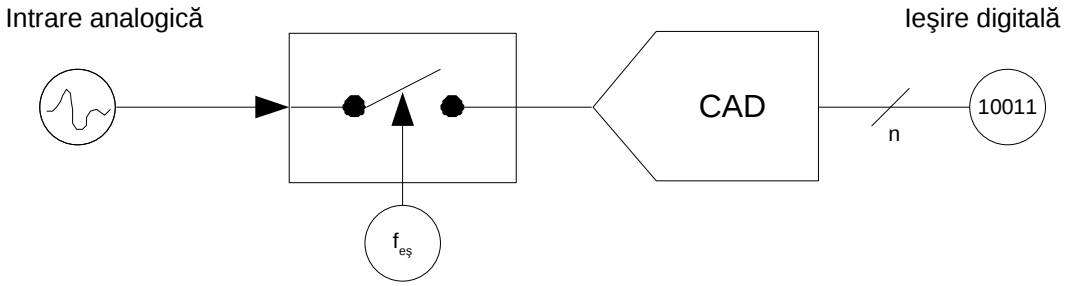


Figura 2.1. *Discretizarea semnalelor analogice*

2.2. Eșantionarea

Eșantionarea reprezintă discretizarea în timp a unui semnal continuu. Un eșantion reprezintă valoarea semnalului la un moment dat, bine precizat.

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

Frecvența de eșantionare $f_s = \frac{1}{T}$ reprezintă frecvența cu care sunt prelevate eșantioanele semnalului analogic. Pentru ca un semnal să poate fi refăcut complet din eșantioanele sale, conform teoremei eșantionării, trebuie satisfăcut criteriul Nyquist și anume că frecvența de eșantionare trebuie să fie cel puțin dublul frecvenței maxime f_m din spectrul semnalului:

$$f_s \geq 2 \cdot f_m \tag{2.1}$$

O condiție suplimentară este aceea ca spectrul semnalului să fie mărginit, de forma celui din figura 2.2:

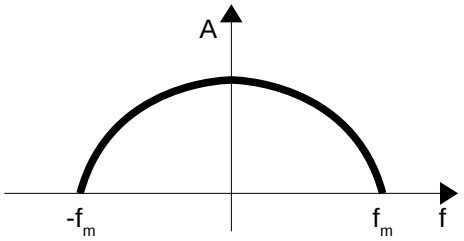


Figura 2.2. *Exemplu de spectru mărginit*

Șirul de eșantioane este echivalent unui vector în matlab, ale cărui elemente sunt valorile semnalului luate la momentele de timp corespunzătoare eșantionării. Trebuie menționat faptul că informația despre frecvența de eșantionare nu este stocată în vector ci numai se presupune cunoscută pentru calcule. Astfel orice vector din matlab poate fi considerat un semnal eșantionat presupunând că i se face corespondența cu un set de parametri de eșantionare.

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

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

În acest caz x va conține eșantioanele unui semnal sinusoidal, dar pentru a cunoște semnalul căruia acest vector corespunde, trebuie să știm unitatea metrică pentru t și frecvența de eșantionare f_s: astfel, acest semnal poate fi o sinusoidă cu durata de 1

secundă și eșantionat la f_s Hz, cu durata de 1 ms și eșantionat la f_s kHz, sau cu durata de 2 s și eșantionat la $\frac{f_s}{2}$ Hz.

Semnalul sinusoidal eșantionat din exemplu precedent este prezentat în Figura 2.3.

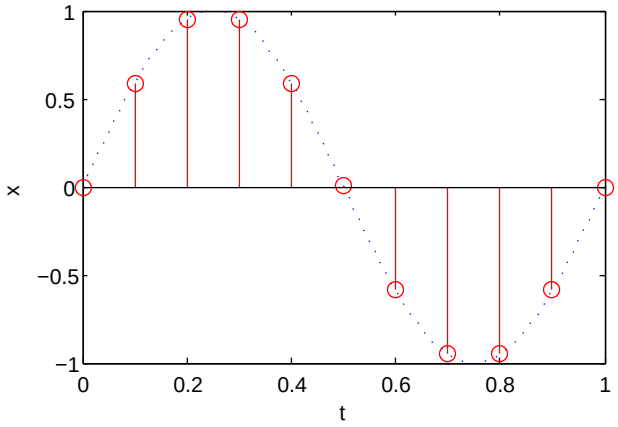


Figura 2.3. *Semnal sinusoidal eșantionat*

2.3. Cuantizarea

Cuantizarea reprezintă discretizarea în amplitudinea valorilor eșantioanelor. Din punct de vedere practic reprezintă aproximarea în trepte (numite nivele de cuantizare) a unui semnal x:

$$x \approx q_j, \text{ pentru } d_j \leq x < d_{j+1} \tag{2.2}$$

unde $\ldots d_{j-1}, d_j, d_{j+1} \ldots$ sunt nivelele de decizie având valoarea pentru cazul când q_j sunt întregi $d_j = q_j - 0,5$. Acest caz (prezent în convertoarele analog digitale) se numește cuantizare uniformă, iar pasul $q_{j+1} - q_j$ se numește cuantă.

Aproximarea unui semnal x între (-1,1) prin 17 cuante se realizează cu o funcție de genul celei din figura 2.4.

Funcția din figura 2.4 a fost generată cu secvența matlab următoare:

```
» x = -1:0.0001:1;
» q = round(x*8);
» plot(x,q)
```

Convertoarele analog digitale sunt de regulă capabile să reprezinte semnalele ca valori întregi în baza 2, de aceea sunt folosite 2^m cuante, unde m este numărul de biți de reprezentare. Numerele sunt reprezentate în cod NBCD pe intervalul $0 \ldots 2^m - 1$ pentru cazul unipolar, și pe intervalul $-2^{m-1} \ldots 2^{m-1} - 1$ pentru cazul bipolar.

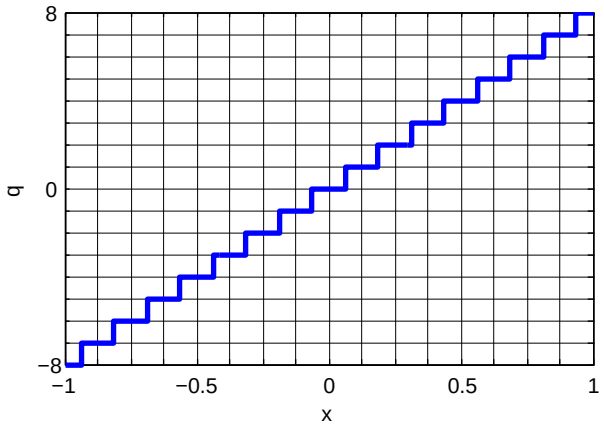


Figura 2.4. *Funcție de cuantizare uniformă*

După cuantizare semnalul original nu mai poate fi refăcut, cuantizarea introducân un zgomot intrinsec numit *zgomot de cuantizare*. Zgomotul de cuantizare reprezintă eroarea dintre semnalul cuantizat și cel original:

$$\varepsilon = |x_c[n] = x_{es}[n]| \tag{2.3}$$

și este de forma celui din figura 2.5.

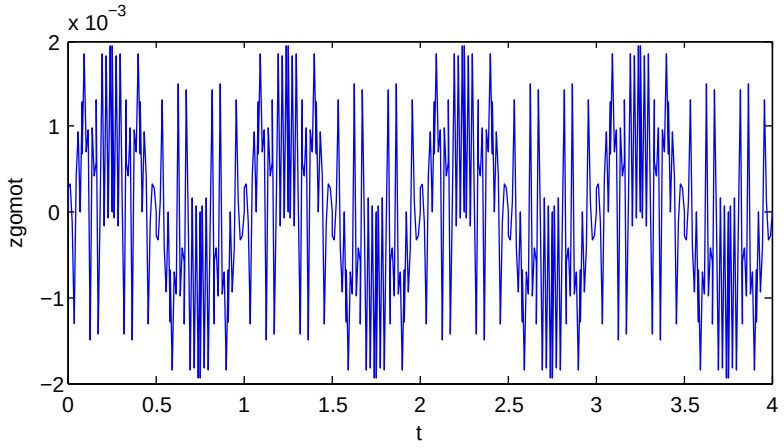


Figura 2.5. *Zgomotul de cuantizare*

Pentru evaluarea acestui zgomot, se folosește raportul semnal-zgomot[2]:

$$SNR = 20 \cdot \lg \frac{V_{ef\text{semnal}}}{V_{ef\text{zgomot}}} \tag{2.4}$$

[2]signal-to-noise ratio – SNR

În cazul unui CAD bipolar cu $N = 2^m$ nivele de cuantizare, pentru un semnal sinusoidal ideal raportul semnal zgomot este:

$$SNR = 20 \cdot \lg \frac{x_{max}/\sqrt{2}}{\underbrace{\Delta q/\sqrt{12}}_{zg.mediucuant.}} = 20 \cdot \lg \sqrt{3/2} \cdot N \approx 6,02 \cdot m + 1,76 \tag{2.5}$$

unde zgomotul mediu de cuantizare a rezultat din ecuația:

$$\begin{aligned} \sqrt{\frac{1}{N \cdot \Delta q} \sum_{j=0}^{N-1} \int_{d_j}^{d_{j+1}} (x - q_j)^2 \, \mathrm{d}x} &= \sqrt{\frac{1}{N \cdot \Delta q} \sum_{j=0}^{N-1} \frac{1}{3} \left[(d_{j+1} - q_j)^3 - (d_j - q_j)^3 \right]} = \\ &= \sqrt{\frac{1}{N \cdot \Delta q} \cdot \frac{1}{3} \cdot N \frac{1}{4} \cdot \Delta q^3} = \Delta q/\sqrt{12} \end{aligned} \tag{2.6}$$

2.4. Exerciții

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

2. Cuantizați acest semnal pe 8 respectiv, 16 biți, reprezentați semnalul cuantizat alături de zgomotul de cuantizare și calculați media pătratică a zgomotului și raportul semnal-zgomot.

Lucrarea 3

Schimbarea ratei de eșantionare

3.1. Importanța frecvenței de eșantionare

Pentru alegerea corectă a frecvenței de eșantionare trebuie satisfăcut criteriul Nyquist:

$$f_s \geq 2 \cdot f_m \tag{3.1}$$

Ca urmare a eșantionării, spectrul semnalului va fi periodicizat, iar respectarea criteriului Nyquist asigură faptul că replicile spectrale nu se vor suprapune (Figura 3.1).

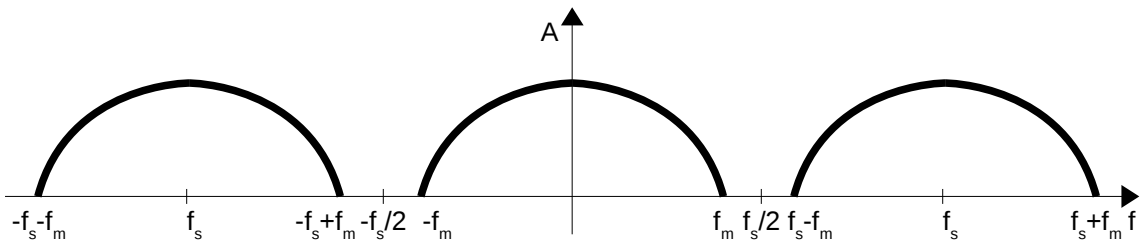


Figura 3.1. *Spectrul periodicizat al unui semnal corect eșantionat*

Dacă criteriul Nyquist nu este respectat, atunci replicile spectrale se vor suprapune, fenomen ce poartă numele de aliere (Figura 3.2). În acest caz, semnalul analogic original nu mai poate fi refăcut corect din eșantioanele sale.

Cu cât frecvența de eșantionare este mai mare, spectrele duplicate vor fi cu atât mai îndepărtate. Pentru a reface semnalul original din cel eșantionat este nevoie de filtrarea trece jos a semnalului eșantionat pentru a înlătura spectrele duplicate.

Filtrarea trece jos trebuie făcută la etapa de conversie digital analogică cu filtre analogice, ceea ce înseamnă că este foarte costisitoare folosirea unor filtre de ordin superior (necesară pentru a înlătura spectrele foarte apropiate). De aceea este de preferat folosirea unei frecvențe de eșantionare cât mai ridicate permițând folosirea unor filtre de ordin inferior.

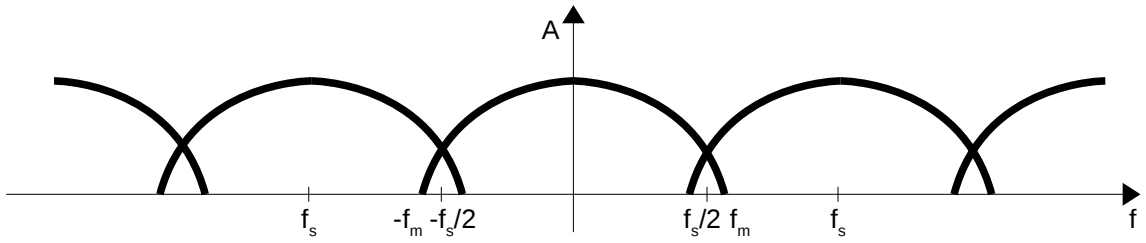


Figura 3.2. *Spectrul unui semnal subeșantionat, afectat de aliere*

Creșterea prea mare a frecvenței de eșantionare însă va necesita circuite de eșantionare cuantizare mai performante (și implicit mai scumpe).

De aceea se practică schimbarea ratei de eșantionare în interiorul blocului de procesare digitală unde filtrarea de ordin superior poate fi ușor implementată la un cost extrem de redus.

3.2. Creșterea ratei de eșantionare

Creșterea ratei de eșantionare se mai numește și interpolare, și este realizată prin introducerea unor eșantioane suplimentare între eșantioanele curente ale semnalului. În Figura 3.3 puteți observa semnalul original, reprezentat de pătrățele albastre, și eșantioanele ce trebuie adăugate reprezentate cu $\times$.

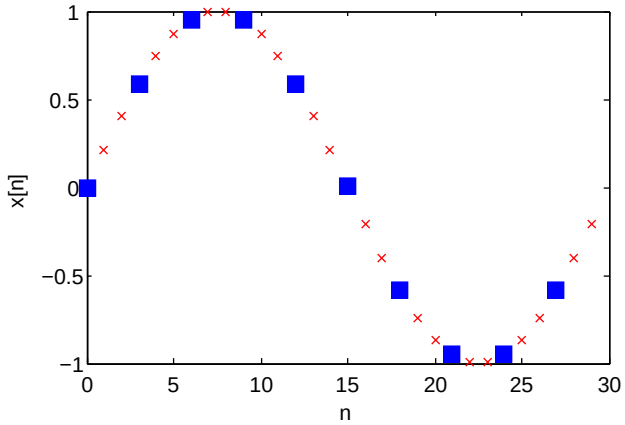


Figura 3.3. *Exemplu de interpolare*

Pentru a calcula valorile eșantionelor ce trebuie adăugate avem două posibilități:

1. calculăm valorile cu o metodă de interpolare (liniară, pătratică, Lagrange)
2. introducem valori de 0, după care filtrăm semnalul cu un fitru trece jos de ordine superioară

Ambele metode au avantaje și dezavantaje din punct de vedere viteză și precizie. Practic se preferă folosirea celei de a doua metode, pentru că filtrarea trece jos oricum trebuie realizată pentru refacerea semnalului analogic.

3.3. Scăderea ratei de eșantionare

Scăderea ratei de eșantionare sau decimarea este realizată prin renunțarea la anumite eșantioane. Este o tehnică complementară cu interpolarea (pentru exemplul din Figura 3.3, eliminăm eșantioanele marcate cu $\times$, rămânând doar eșantioanele marcate cu pătrățele).

Trebuie avut însă grijă pentru evitarea apariției fenomenului de aliere: dacă există frecvențe mai mari decât $\frac{1}{2}$ din noua frecvență de eșantionare, acestea se vor suprapune frecvențelor joase distrugând semnalul. Pentru a evita acest lucru înainte de decimare, semnalul original trebuie trecut printr un filtru trece jos cu frecvența de tăiere de $\frac{f_s}{2}$.

3.4. Schimbarea ratei de eșantionare cu un factor rațional

Interpolarea și decimarea sunt folosite pentru schimbarea ratei de eșantionare cu un factor întreg.

În cele mai multe cazuri însă este necesar să se schimbe rata de eșantionare cu un factor rațional. Un exemplu clasic este conversia între o rata de eșantionare a semnalelor audio de la 44100Hz (audio CD) la 48kHz (digital audio).

Pentru realizarea unei asemena schimbări semnalul original trebuie interpolat la o frecvență care reprezintă cel mai mic multiplu comun al celor două frecvențe de eșantionare, după care trebuie decimat la noua frecvență de eșantionare. Cele două filtre trece jos (de la sfârșitul etapei de interpolare și de la începutul etapei de eșantionare) se pot combina într unul singur (vezi Figura 3.4).

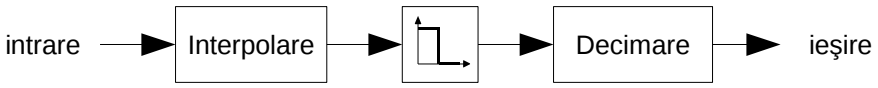


Figura 3.4. *Schimbarea ratei de eșantionare cu un factor rațional*

3.5. Exerciții

1. Realizați în Matlab interpolarea unui semnal la o frecvență de eșantionare de 4 ori mai mare prin introducerea de 0-uri și filtrarea trece jos. Observați semnalul în fiecare etapă.

Pentru filtrarea trece jos (a unui semnal x) puteți folosi secvența:

```
» [b a] = cheby1(5, 0.5, fsmic / fsmare);
» xout = filter(b,a,x);
```

2. Observați efectul de aliere prin afișarea pe același grafic a unui semnal eșantionat corespunzător și același semnal decimat la o frecvență mai mică decât cea corespunzătoare criteriului Nyquist.

3. Convertiți un semnal eșantionat la 44100 Hz la o rată de eșantionare de 48000 Hz. Găsiți calea optimă pentru a realiza conversia. Pentru a găsi calea optimă porniți de la ideea că interpolarea cu un factor foarte mare va necesita foarte multe calcule.

   Pentru a calcula cel mai mic multiplu comun a două numere folosiți funcția `lcm`, iar pentru factorizarea unui număr funcția `factor`.

Lucrarea 4

Analiza spectrală a semnalelor

4.1. Transformata z

Transformata z este o tehnică similară cu transformata Laplace, fiind o formă generalizată a transformatei Fourier. Aceasta este folosită la analiza spectrală atât a semnalelor continue cât și pentru semnale discrete. Transformata Laplace are următoarea formă:

$$X(\sigma, \omega) = \int_{-\infty}^{+\infty} x(t) e^{-\sigma t} e^{-j\omega t} \, \mathrm{d}t \tag{4.1}$$

În această formulă exponentul $\sigma + j\omega$ poate fi schimbat cu valoarea complexă s:

$$X(s) = \int_{-\infty}^{+\infty} x(t) e^{-st} \, \mathrm{d}t \tag{4.2}$$

Pentru a ajunge la formula transformatei z, discretizăm semnalul $x(t)$ și înlocuim e^s cu r:

$$X(r, \omega) = \sum_{n=-\infty}^{+\infty} x[n] r^{-n} e^{-j\omega n} \tag{4.3}$$

sau înlocuind $r \cdot e^{j\omega}$ cu z:

$$X(z) = \sum_{n=-\infty}^{+\infty} x[n] z^{-n} \tag{4.4}$$

Se poate observa că transformata z spre deosebire de transformata Laplace este reprezentată în coordonate polare. Distanța de la origine r este valoarea atenuării exponențiale a semnalului, iar ω este frecvența semnalului.

De exemplu 3 semnale de 50Hz, eșantionate la 8 kHz și având atenuări corespunzătoare cu $r = 1.1$ (semnal atenuat în timp), $r = 1$ (semnal constant în timp) respectiv $r = 0.9$ (semnal amplificat în timp) sunt prezentate în Figura 4.1. Lângă fiecare semnal este prezentat în coordonate polare și spectrul corespunzător calculat de-a lungul

cercului unitate. Se poate observa concentrarea punctelor se apropie de origine cu cât valoare lui r este mai mic.

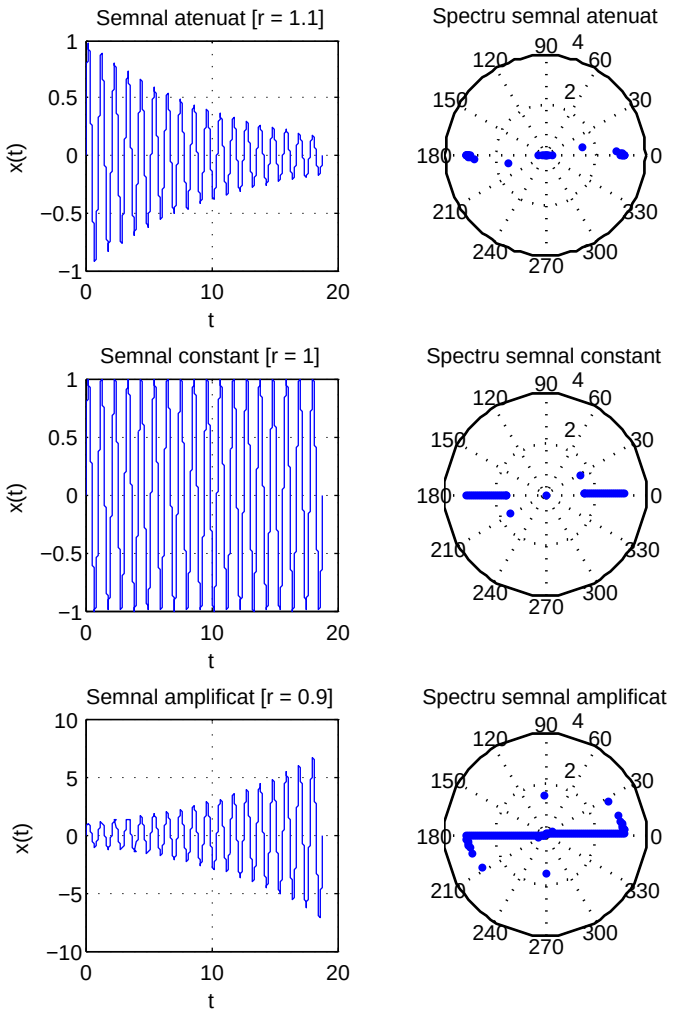


Figura 4.1. *Semnale pentru diferite valori ale lui r și spectrele polare asociate*

Semnalul constant și periodic va avea spectrul pe cercul unitar și este identic cu spectrul Fourier. Pentru a calcula spectrul a unui semnal putem folosi funcția Matlab `czt`. De exmplu pentru un semnal sinusoidal:

```
» t = 0 : 1/8000 : 1023/8000;
» x = sin(2*pi*500*t);
```

Putem calcula transformata z cu:

```
» z = czt(x);
» z = z / (length(z)/2);
```

Normalizarea la jumătatea lungimii vectorului este folosită pentru a avea informații corecte despre magnitudinea spectrală a semnalului. Dacă ne interesează doar forma spectrului normalizarea nu este necesară.

Transformata z conține valori complexe, de aceea pentru vizualizarea spectrului trebuie afișat magnitudinea și faza acestor valori. Pentru calculul magnitudinii se folosește funcția **abs** iar pentru calculul fazei funcțiile **angle** și **unwrap**:

```
» zm = abs(z);
» zf = unwrap(angle(z));
```

Spectrul semnalului calculat conține același număr de eșantioane ca semnalul original organizat în felul următor: prima jumătate a eșantioanelor indică spectrul frecvențelor între 0 și jumătatea frecvenței de eșantionare, iar celelalte eșantioane indică spectrul frecvențelor oglindite (frecvențele negative).

Așadar pentru a afișa corect spectrul se poate folosi următoarea secvență:

```
» w = 0 : 8000/1024 : 8000*1023/1024;
» subplot(2,1,1),plot(w,zm),subplot(2,1,2),plot(w,zf);
```

Graficul rezultat din aceste comenzi este prezentat în Figura 4.2.

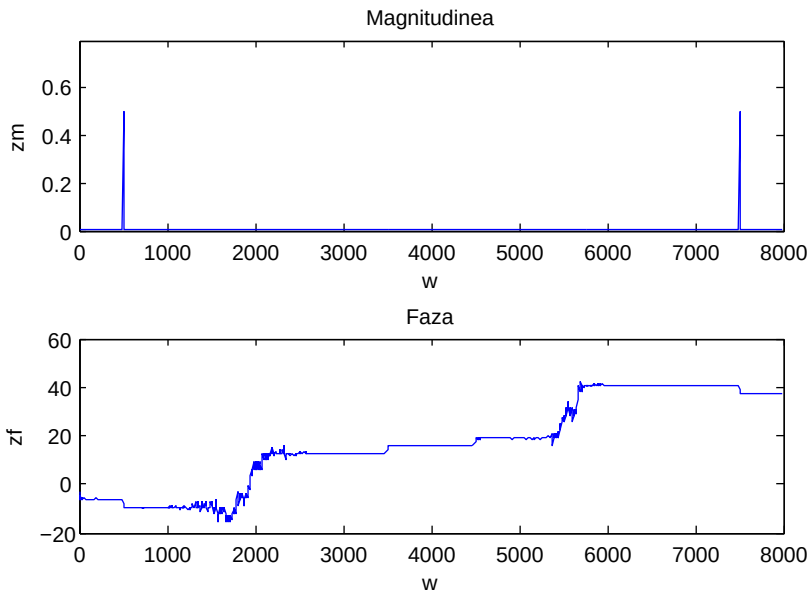


Figura 4.2. *Magnitudinea și faza spectrală a unui semnal sinusoidal de 500 Hz*

Spectrul calculat de funcția `czt` în mod implicit este spectrul complet de pe cercul

unitar, adică identic cu spectrul Fourier. Se pot calcula însă și doar anumite porțiuni ale spectrului și la alte distanțe față de origine, modificând parametrii funcției `czt`.

4.2. Răspunsul în frecvență al unui filtru

Transformata z este folosită îndeosebi pentru calculul răspunsului în frecvență ale filtrelor. Pentru un filtru oarecare este valabilă următoarea expresie:

$$\begin{aligned} a(1)\cdot y(n) = b(1)\cdot x(n) + b(2)\cdot x(n-1) + \cdots + b(B)\cdot x(n-B+1) - \\ - a(2)\cdot y(n-1) - \cdots - a(A)\cdot y(n-A+1) \end{aligned} \tag{4.5}$$

Cunoscând transformatele z ale semnalelor $x[n]$ și $y[n]$, adică $X[z]$ și $Y[z]$ se pot calcula funcția de transfer $H[z]$ a sistemului filtrului ca fiind:

$$H[z] = \frac{Y[z]}{X[z]} \tag{4.6}$$

Expresia lui $H[z]$ se poate scrie mai departe ca fiind:

$$H[z] = \frac{b_1 + b_2\cdot z^{-1} + b_3\cdot z^{-2} + \ldots}{1 - a_2\cdot z^{-1} - a_3\cdot z^{-2} - \ldots} \tag{4.7}$$

(normalizat la a_1).

Rescriind relația de mai sus ca:

$$H[z] = \frac{(z-z_1)(z-z_2)(z-z_3)\ldots}{(z-p_1)(z-p_2)(z-p_3)\ldots} \tag{4.8}$$

pot fi identificați zerourile z_1, z_2, z_3, ... și polii p_1, p_2, p_3, ... funcției de transfer.

Pornind de la polii și zerourile dorite, se pot calcula coeficienții filtrului.

Pentru calculul răspunsului în frecvență a unui filtru se poate folosi funcția Matlab `freqz` sau, pentru a afișa direct polii și zerourile, funcția `zplane`.

4.3. Exerciții

1. Generați câteva semnale sinusoidale, dreptunghiulare respectiv triunguläre cu frecvențe de bază de 50, 100 și 1000 Hz și frecvența de eșantionare de 8 kHz și observați spectrul acestor semnale.

2. Generați un semnal alcătuit din mai multe sinusoide (sau alte forme de undă) și filtrați acest semnal. Afișați răspunsul în frecvență al filtrului folosit și spectrele semnalului original și cel filtrat.

   Pentru a realiza filtrarea semnalului folosiți secvența:

```
» [b a] = cheby1(5, 0.5, 50/4000);
» y = filter(b, a, x);
```

Lucrarea 5

Transformata Fourier Rapidă

5.1. Transformata Fourier Discretă

Transformata Fourier discretă este varianta eșantionată a transformatei Fourier continue. Transformata Fourier discretă a unui semnal $x[n]$ format din N eșantioane este calculată cu formula:

$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot e^{-j\frac{2\pi}{N}kn}, \quad k = \overline{0, N-1} \tag{5.1}$$

Pentru calculul unei singure valori din spațiul transformatei sunt necesare N operații, prin urmare complexitatea algoritmului de calcul al transformatei Fourier pentru o secvență de N eșantioane, este $O(N^2)$. În 1965 Cooley și Tukey au inventat un algoritm de calcul rapid al tranformatei Fourier[1] care reduce ordinul acestor calcule la $O(N \log_2 N)$ prin exploatarea proprietăților transformatei Fourier (simetriei și periodicității funcțiilor *sin* și *cos*) și refolosirea coeficienților la înmulțirile complexe.

5.2. Algoritmul FFT cu decimare în timp

Se exprimă spectrul eșantionat separat în funcție de secvența pară respectiv cea impară a semnalului:

$$X(k) = \sum_{n=0}^{\frac{N}{2}-1} x(2n) w_N^{2kn} + \sum_{n=0}^{\frac{N}{2}-1} x(2n+1) w_N^{k(2n+1)} \tag{5.2}$$

unde s-a notat:

$$w_N^{2kn} = e^{-j\frac{2\pi}{N}2kn} = w_{\frac{N}{2}}^{kn} \tag{5.3}$$

[1]Transformata Fourier Rapidă – Fast Fourier Transform (FFT)

Relația devine astfel:

$$X(k) = \underbrace{\sum_{n=0}^{\frac{N}{2}-1} x(2n) w_{\frac{N}{2}}^{kn}}_{X_{par}(k)} + \left[\underbrace{\sum_{n=0}^{\frac{N}{2}-1} x(2n+1) w_{\frac{N}{2}}^{kn}}_{X_{impar}(k)} \right] \cdot w_N^k = X_{par}(k) + w_N^k \cdot X_{impar}(k) \quad (5.4)$$

Am notat cu $X_{par}(k)$ transformata Fourier discretă a eșantioanelor pare ale secvenței $x(n)$ iar cu $X_{impar}(k)$ a celor impare.

Dat fiind faptul că secvența FFT este simetrică față de mijloc avem:

$$X_{par}(k + \frac{N}{2}) = X_{par}(k) \quad (5.5)$$

$$w_N^{k+\frac{N}{2}} \cdot X_{impar}(k + \frac{N}{2}) = -w_N^k \cdot X_{impar}(k) \quad (5.6)$$

Astfel secvența FFT cu $k = \overline{0, \frac{N}{2} - 1}$ pentru secvența $x(n),\ n = \overline{0, N-1}$:

$$\begin{cases} X(k) & = X_{par}(k) + w_N^k \cdot X_{impar}(k) \\ X(k + \frac{N}{2}) & = X_{par}(k) - w_N^k \cdot X_{impar}(k) \end{cases} \quad (5.7)$$

Această formulă poate fi reprezentată grafic ca aripile unei fluture (de aici vine denumirea celulei din FFT: butterfly) reprezentată grafic în Figura 5.1.

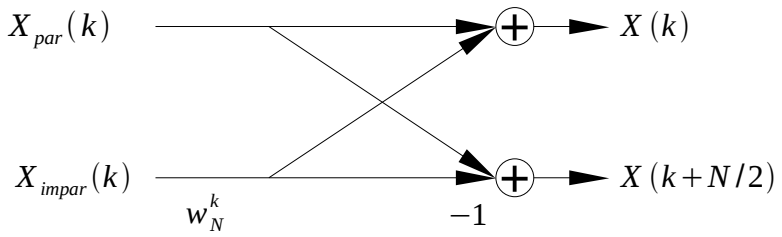


Figura 5.1. *Celulă de tip fluture*

În Figura 5.2 se poate vedea "fluturele" de calcul pentru algoritmul FFT pentru 8 eșantioane.

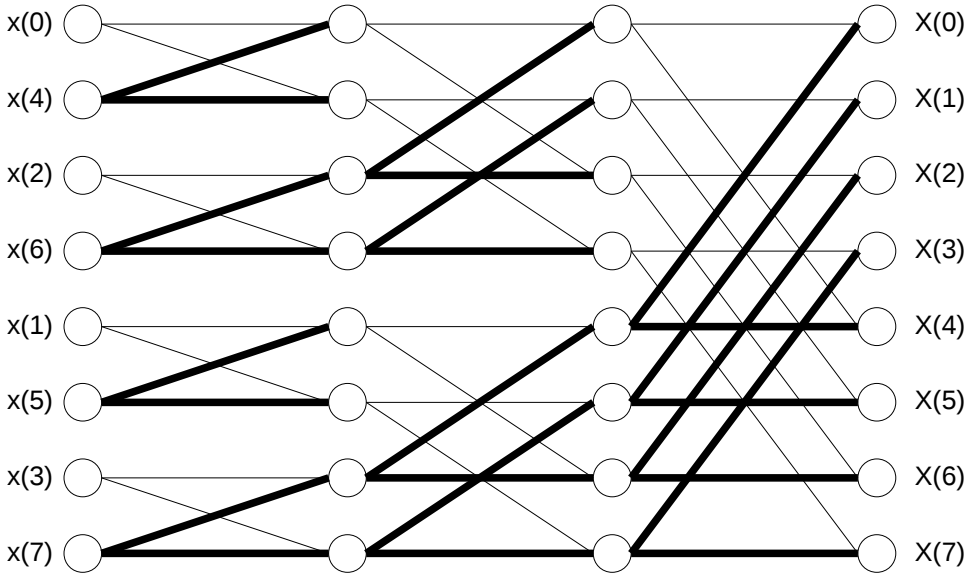


Figura 5.2. *Fluturele de calcul al FFT cu decimare în timp pentru 8 eșantioane*

5.3. Algoritmul FFT cu decimare în frecvență

În acest caz dezvoltăm termenii pari și impari ai transformatei Fourier:

$$\begin{aligned} X(2k) &= \sum_{n=0}^{N-1} x(n) w_N^{2kn} = \sum_{n=0}^{\frac{N}{2}-1} x(n) w_N^{2kn} + \sum_{n=0}^{\frac{N}{2}-1} x(n+\frac{N}{2}) w_N^{2k(n+\frac{N}{2}} \\ &= \sum_{n=0}^{\frac{N}{2}-1} x(n) w_N^{2kn} + \sum_{n=0}^{\frac{N}{2}-1} x(n+\frac{N}{2}) w_N^{2kn} \underbrace{w_N^{2k\frac{N}{2}}}_{1} \\ &= \sum_{n=0}^{\frac{N}{2}-1} x(n) w_{\frac{N}{2}}^{kn} + \sum_{n=0}^{\frac{N}{2}-1} x(n+\frac{N}{2}) w_{\frac{N}{2}}^{kn} \end{aligned} \tag{5.8}$$

$$\begin{aligned} X(2k+1) &= \sum_{n=0}^{N-1} x(n) w_N^{(2k+1)n} = \sum_{n=0}^{\frac{N}{2}-1} x(n) w_N^{(2k+1)n} + \sum_{n=0}^{\frac{N}{2}-1} x(n+\frac{N}{2}) w_N^{(2k+1)(n+\frac{N}{2})} \\ &= \sum_{n=0}^{\frac{N}{2}-1} x(n) w_N^{2kn} w_N^n + \sum_{n=0}^{\frac{N}{2}-1} x(n+\frac{N}{2}) w_N^{2kn} w_N^n \underbrace{w_N^{2k\frac{N}{2}}}_{1} \underbrace{w_N^{\frac{N}{2}}}_{-1} \\ &= \sum_{n=0}^{\frac{N}{2}-1} x(n) w_{\frac{N}{2}}^{kn} w_N^n - \sum_{n=0}^{\frac{N}{2}-1} x(n+\frac{N}{2}) w_{\frac{N}{2}}^{kn} w_N^n \end{aligned} \tag{5.9}$$

ceea ce poate fi notat simplificat ca:

$$\begin{cases} X(2k) & = X_{inf}(k) + X_{sup}(k) \\ X(2k+1) & = X_{inf}(k) \cdot w_N^n - X_{sup}(k) \cdot w_N^n \end{cases} \tag{5.10}$$

unde s-a notat cu $X_{inf}(k)$ transformata Fourier a primelor $\frac{N}{2}$ eșantioane și cu $X_{sup}(k)$ transformata Fourirer a ultimelor $\frac{N}{2}$ eșantioane.

Fluturele pentru algoritmul cu decimare în frecvență este reprezentat în Figura 5.3.

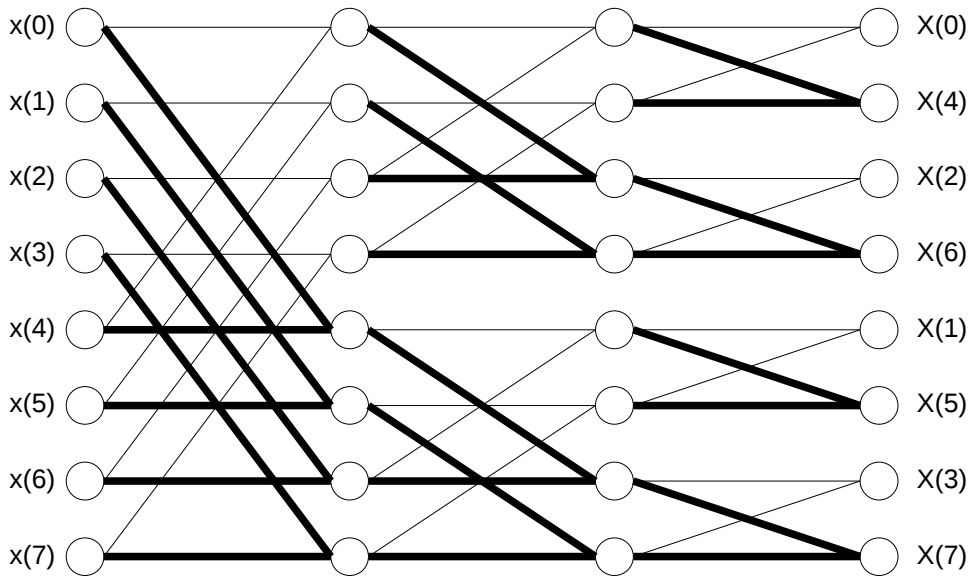


Figura 5.3. *Fluturele de calcul al FFT cu decimare în frecvență pentru 8 eșantioane*

5.4. Exerciții

1. Implementați algoritmul FFT (una din cele două variante) și comparați rezultatul și timpul de execuție cu timpul de execuție al algoritmului clasic ne-optimizat și cel al funcției `fft` din Matlab.

2. Calculați spectrul Fourier pentru câteva semnale cunoscute (de exemplu o sumă de două su trei sinusoide de diverse frecvențe, un semnal dreptunghiular, etc.) folosind funcția `fft` din Matlab. Vizualizați spectrul de amplitudine și cel de fază.

Lucrarea 6

Filtrarea semnalelor

Filtrarea este o operație fundamentală de procesare dintr-un sistem de procesare a semnalelor. Filtrarea este utilizată de regulă pentru eliminarea zgomotului care afectează un semnal. În funcție de tipul filtrului (trece jos, trece bandă sau trece sus), acesta va lăsa să treacă anumite frecvențe ale semnalului, dintr-o bandă specificată, rejectând frecvențele din afara benzii de trecere.

6.1. Noțini teoretice

Operația de filtrare reprezintă de regulă trecerea unui semnal $x(t)$ printr-un sistem liniar invariant în timp, a cărui funcție pondere $h(t)$ este cunoscută (vezi Figura 6.1)

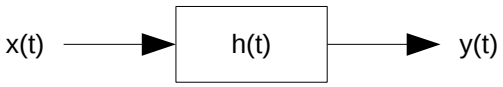


Figura 6.1. *Operația de filtrare*

Un sistem se numește liniar, dacă respectă principiul superpoziției, adică dacă la intrarea sistemului aplicăm o combinație liniară de două (sau mai multe) semnale $ax_1(t) + bx_2(t),\ a, b \in \mathbb{R}$, atunci semnalul de la ieșirea sistemului poate fi scris ca fiind $ay_1(t) + by_2(t)$, unde $y_1(t)$ și $y_2(t)$ reprezintă răspunsurile sistemului la semnalele $x_1(t)$ respectiv $x_2(t)$.

Dacă răspunsul sistemului este mereu același pentru un același semnal de intrare, indiferent de momentul de timp la care este aplicat semnalul respectiv la intrarea sistemului, atunci sistemul se numește invariant în timp.

Sistemul are asociat o funcție pondere $h(t)$, care reprezintă răspunsul la impuls al sistemului, adică ieșirea corespunzătoare unui impuls Dirac aplicat la intrare. Operația realizată de un bloc de filtrare nu este altceva decât o operație de convoluție dată de

următoarea formulă:

$$y(t) = h(t) * x(t) = \int_{-\infty}^{+\infty} h(\tau) \cdot x(t-\tau)\,\mathrm{d}\tau = \int_{-\infty}^{+\infty} h(t-\tau) \cdot x(\tau)\,\mathrm{d}\tau \tag{6.1}$$

În domeniul spectral, produsul de convoluție se transformă în produs simplu:

$$Y(\omega) = H(\omega) \cdot X(\omega) \tag{6.2}$$

unde $Y(\omega)$, $H(\omega)$ și $X(\omega)$ reprezintă spectrele Fourier ale lui $y(t)$, $h(t)$ respectiv $y(t)$. $H(\omega)$ poartă numele de funcție de transfer a sistemului/filtrului

Convoluția poate fi imaginată ca o fereastră glisantă (fig. 6.2) reprezentată de funcția ponder $h(t)$, care se deplasează peste semnalul de la intrare, realizându-se o sumă a eșantioanelor de intrare ponderate cu coeficienții filtrului. Suma nu este altceva decât răspunsul filtrului la un moment dat.

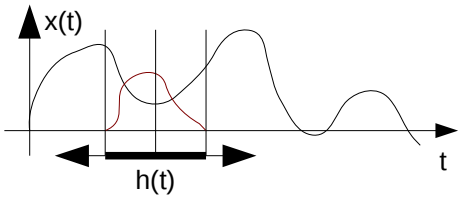


Figura 6.2. *Convoluția*

În domeniul discret convoluția are următoarea formă:

$$y(n) = \sum_{k=-\infty}^{\infty} h(k) \cdot x(n-k) \tag{6.3}$$

și descrie rezultatul când funcția x trece prin sistemul caracterizat prin funcția pondere h.

Dacă funcția pondere a sistemului (răspunsul la impuls) este un vector de lungime K atunci convoluția devine:

$$y(n) = \sum_{k=0}^{K} h(k) \cdot x(n-k) \tag{6.4}$$

adică echivalent cu înmulțirea a două polinoame cu coeficienți din vectorii x și h, ceea ce este simplu de realizat cu ajutorul unui procesor.

Când răspunsul la impuls al blocului respectiv are lungimea infinită, nu se poate realiza convoluția cu funcția de transfer. În acest caz se folosesc două seturi de coeficienți, unul care se convolvă cu eșantioanele de la intrare iar celălalt cu ieșirile anterioare, astfel încât răspunsul la impuls al întregului sistem să fie exact răspunsul necesar.

Astfel filtrarea se poate descrie cu formula:

$$\begin{aligned} a(1)\cdot y(n) = b(1)\cdot x(n) + b(2)\cdot x(n-1) + \cdots + b(B)\cdot x(n-B+1) - \\ - a(2)\cdot y(n-1) - \cdots - a(A)\cdot y(n-A+1) \end{aligned} \tag{6.5}$$

Această formulă poate fi realizată fizic prin folosirea unor blocuri de întârziere și amplificare asupra intrării și ieșirii, și însumarea acestor semnale rezultate, ca în Figura 6.3.

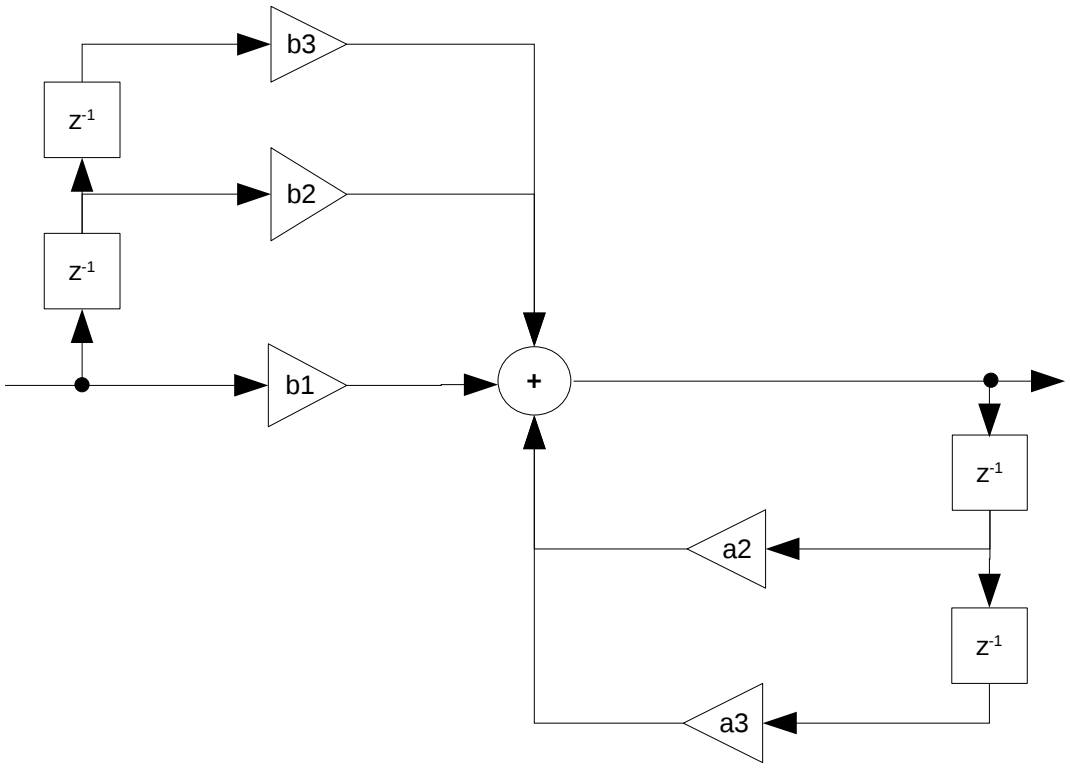


Figura 6.3. *Implementarea unui filtru cu blocuri de întârziere și amplificare*

6.2. Filtrarea în matlab

Filtrarea unui semnal se realizează în matlab cu funcția `filter`:

```
» y = filter(b,a,x);
```

În acest caz x este vectorul ce conține semnalul de la intrare, iar b și a sunt doi vectori ce conțin coeficienții filtrului. Dacă vrem să realizăm o filtrare cu o funcție de transfer de lungime finită (adică o convoluție) putem folosi următorul apel al funcției `filter`:

```
» y = filter(h,1,x);
```

în care apare 1 în loc de a, pentru că $a(1)$ trebuie neapărat să fie definit pentru că la această valoare se face normarea rezultatului.

6.3. Exerciții

1. Implementați o funcție de filtrare cu formatul `y = myfilter(b,a,x)` și comparați rezultatele cu cele obținute folosind funcția matlab **`filter`**.

2. Observați efectul filtrării asupra unui semnal folosind un filtru Cebîșev. Pentru semnal de test puteți folosi semnalul generat la lucrarea 2, sau orice semnal alcătuit din sinusoide de frecvențe diferite. Pentru determinarea coeficienților acestui filtru, folosiți următoarea funcție matlab, apelată cu parametrii indicați în exemplu:

   ```
   » [b a] = cheby1(5,0.5,50/4000);
   ```

Lucrarea 7

Proiectarea filtrelor în Matlab

7.1. Filtre FIR

Filtrele FIR[1] sunt filtre de convoluție la care semnalul de intrare este convolut cu coeficienții sau funcția pondere a filtrului:

$$y_k = b_1 \cdot x_k + b_2 \cdot x_{k-1} + \dots \tag{7.1}$$

unde x_k sunt eșantioanele semnalului la intrare, y_k reprezintă răspunsul filtrului la momentul k (eșantionul k al semnalului y de la ieșirea filtrului), iar valorile b_k reprezintă coeficienții filtrului.

Proiectarea unui filtru FIR se bazează pe diverse tipuri de ferestre (dreptunghiulară, triunghiulară, Hamming, Hann, Chebyshev, Kaiser).

În Matlab, proiectarea unui filtru se face cu comanda `fir1` respectiv `fir2`.

Comanda `fir1` are următorarele forme:

```
» fir1(N, Wn);
» fir1(N, Wn, 'high');
» fir1(N, Wn, 'stop');
» fir1(N, Wn, wind);
```

unde N este ordinul filtrului (filtrul va avea N+1 coeficienți), Wn este frecvența de tăiere raportată la frecvența maximă a semnalului (frecvența de eșantionare / 2). Dacă Wn este un vector de 2 sau mai multe elemente atunci filtrul va fi un filtru trece bandă cu banda între cele perechile de valori din Wn. Dacă este specificat `'high'` (în cazul în care Wn este scalar) atunci filtrul va fi un filtru trece sus în loc de filtru trece jos, iar dacă este specificat `'stop'` (în cazul în care Wn este vector de mai multe elemente) atunci filtrul va deveni filtru oprește bandă. `wind` este o fereastră folosită la generarea filtrului. Implicit fereastra folosită este o fereastră Hamming. Diferite tipuri de ferestre pot fi generate cu comenzile: `boxcar`, `bartlett`, `hamming`, `hann`, `kaiser`, `chebwin`, `blackman`.

Comanda `fir2` are următoarea sintaxă:

[1] engl – Finite Impulse Response (raspuns finit la impuls)

```
» fir2(N, F, M)
```

unde N este ordinul filtrului, iar F și M reprezintă răspunsul dorit în frecvență al filtrului specificat sub formă grafică (ca și cum ar fi desenat cu o comandă gen `plot(F,M)`). Comanda `fir2` va genera un filtru care aproximează acest răspuns în frecventă.

Funcția de transfer (coeficienții) și răspunsul în frecvență a unor filtre trece jos, trece bandă și trece sus este prezentat în figura 7.1.

7.2. Filtre IIR

Filtrele IIR[2] sunt filtre recusive la care pentru calcularea valorii curente a semnalului de ieșire sunt folosite atât valorile semnalului de intrare cât și valorile vechi ale semnalului de ieșire. Astfel se poate realiza un răspuns aproape "infinit" fără a utiliza foarte mulți coeficienți. Ecuația care implementează un filtru IIR este următoarea:

$$\begin{aligned} a(1)\cdot y(n) = b(1)\cdot x(n) + b(2)\cdot x(n-1) + \cdots + b(B)\cdot x(n-B+1) - \\ - a(2)\cdot y(n-1) - \cdots - a(A)\cdot y(n-A+1) \end{aligned} \tag{7.2}$$

Avantajul acestor filtre este că se poate realiza o filtrare cu răspuns lung cu calcule puține, dezavantajul fiind că nu avem control la fel de precis asupra răspunsului filtrului, putând apărea "ripluri[3]" (oscilații nedorite) în banda de trecere și în banda de tăiere. De asemenea filtrul poate deveni instabil dacă coeficientul pentru eșantioanele de ieșire depășește 1, caz în care semnalul la ieșire va intra în saturație.

În Matlab proiectarea filtrelor IIR se pot face cu funcțiile `butter`, `cheby1`, `cheby2`, `ellip` și `yulewalk`.

Funcția `butter` generează un filtru Butterworth și are următoarea sintaxă:

```
» [b a] = butter(N, Wn);
» [b a] = butter(N, Wn, 'high');
» [b a] = butter(N, Wn, 'stop');
```

Parametrii sunt aceiași ca la filtre FIR.

Funcția `cheby1` și `cheby2` generează filtre Chebyshev de tipul I respectiv II:

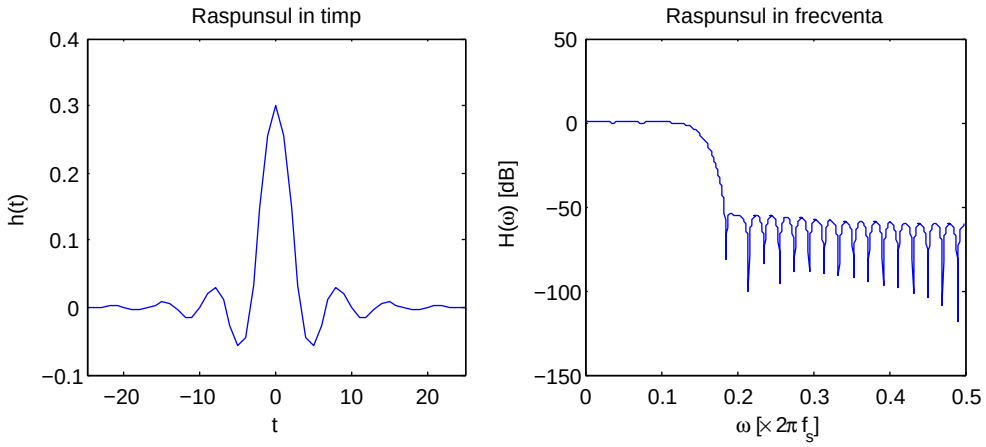
```
» [b a] = cheby1(N, Rp, Wn);
» [b a] = cheby2(N, Rs, Wn);
```

Filtrul Chebyshev de tipul I prezintă ripluri în banda de trecere, diferența maximă a acestor ripluri de la 0 dB poate fi specificată prin parametrul Rp. Filtrul Chebyshev de tipul II prezintă ripluri în banda de tăiere, diferența minimă între aceste ripluri și linia de 0 dB poate fi specificată prin parametrul Rs.

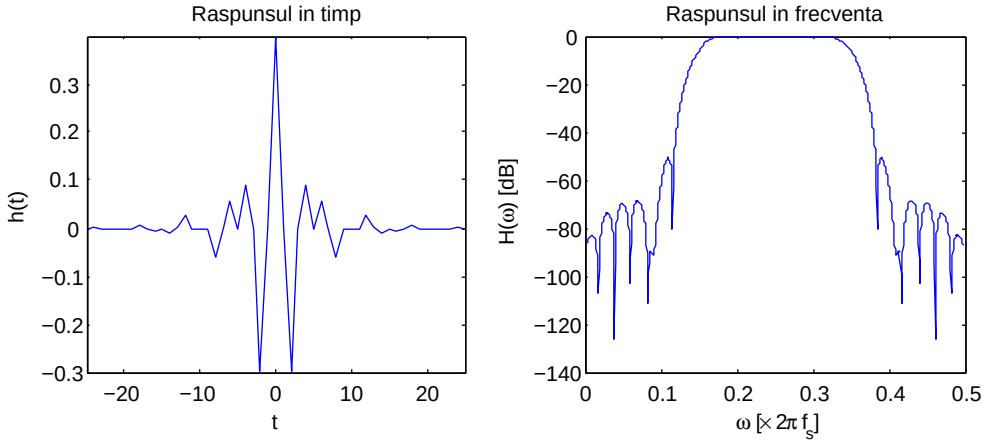
Funcția `ellip` generează un filtru elliptic:

[2] engl. – Infinite Impulse Response (răspuns infinit la impuls)

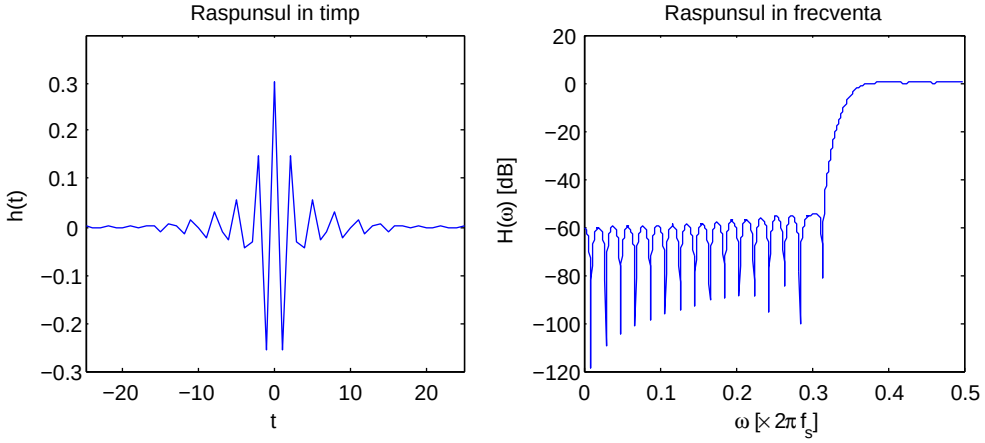
[3] engl. – ripple (undișoare)



(a) Filtru trece jos



(b) Filtru trece bandă



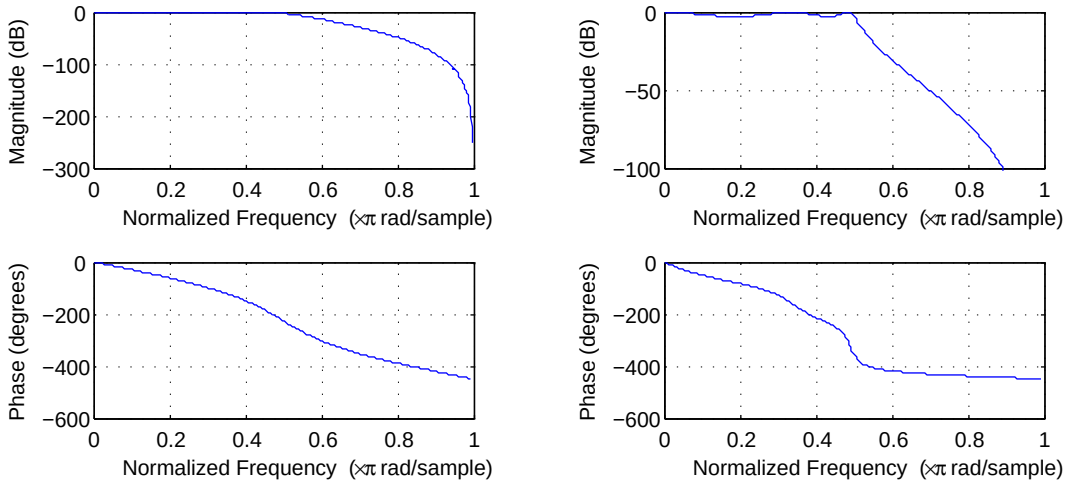
(c) Filtru trece sus

Figura 7.1. *Funcția de transfer și răspunsul în frecvență a unor filtre trece jos (a), trece bandă (b) și trece sus (c)*

```
» [b a] = ellip(N, Rp, Rs, Wn);
```

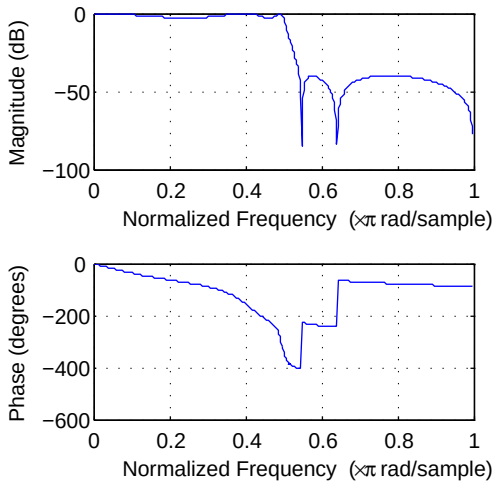
Acest filtru prezintă ripluri atât în banda de trecere, cât și în banda de tăiere, aceste ripluri pot fi controlate cu parametrii Rp și Rs similar cu filtrele Chebyshev.

Un exemplu pentru aceste filtre reprezentate grafic prin intermediul comenzii `freqz` poate fi vizualizat în figura 7.2.

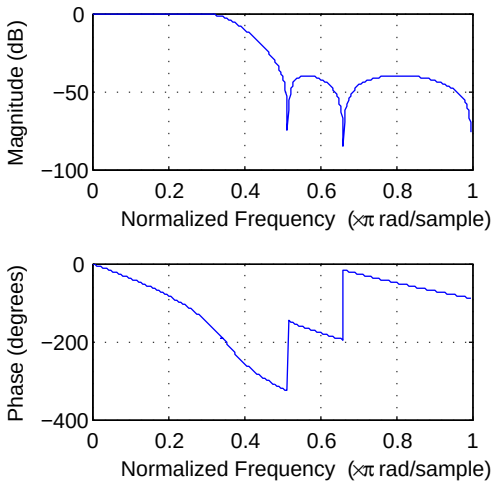


(a) Butterworth

(b) Chebyshev I



(c) Elliptic



(d) Chebyshev II

Figura 7.2. *Răspunsurile în frecvență a filtrelor Butterworth, Chebyshev de tipul I și II respectiv elliptic*

Funcția `yulewalk` generează un filtru IIR care aproximează un răspuns în frecvență precizat în mod similar cu funcția `fir2`:

```
» [b a] = yulewalk(N, F, M)
```

Pentru a calcula ordinul minim al unui filtru IIR care corespunde anumitor constrângeri asupra riplurilor în banda de trecere și banda de tăiere și frecvențele limită se pot folosi funcțiile `buttord`, `cheb1ord`, `cheb2ord` și `ellipord`.

7.3. Exerciții

1. Generați câteva filtre FIR cu ajutorul comenzilor `fir1` și `fir2`, de diferite tipuri (trece jos, trece bandă, trece sus), pentru diferite frecvențe folosind diferite ferestre și comparați răspunsul la impuls, răspunsul în frecvență și rezultatul filtrării unor semnale compuse din mai multe sinusoide și zgomot.

2. Generați câteva filtre IIR cu comenzile `butter`, `cheby1`, `cheby2`, `ellip` și `yulewalk`, de diferite tipuri (trece jos, trece bandă, trece sus), pentru diferite frecvențe folosind diferite ferestre și comparați răspunsul la impuls, răspunsul în frecvență și rezultatul filtrării unor semnale compuse din mai multe sinusoide și zgomot.

3. Separați semnalul de 300 Hz din semnalul stocat în fișierul `s.mat` (frecvența de eșantionare pentru acest semnal fiind de 8 kHz). Vizualizați semnalul înainte și după implementarea operație de filtrare.

Lucrarea 8

Filtrul adaptat la semnal

8.1. Breviar teoretic

Filtrul adaptat la semnal este un tip aparte de filtru, proiectarea acestuia făcându-se prin specificarea funcției pondere în domeniul timp, nu prin specificarea funcției de tranfer în domeniul frecvență ca la filtrele prezentate anterior. Acest filtru se folosește de regulă pentru detecție, de exemplu într-un lanț de transmisiune, mai exact, atunci când se cunoaște forma semnalului original transmis, care la recepție este afectat de zgomot.

Pentru un semnal determinist de durată finită, $s(t)$, se definește filtrul adaptat la semnal ca fiind filtrul liniar invariant în timp care are următoarea funcție de transfer:

$$h(t) = K \cdot s(-(t - t_0)) \tag{8.1}$$

în care K și t_0 sunt constante reale oarecare, cu constrângerea că t_0 trebuie astfel ales încât filtrul să fie *cauzal*. Un filtru (sau un semnal) se spune că aste cauzal, dacă $h(t) = 0$ pentru $t < 0$.

Un exemplu de filtru adaptat la semnal este prezentat în Figura 8.1 [1].

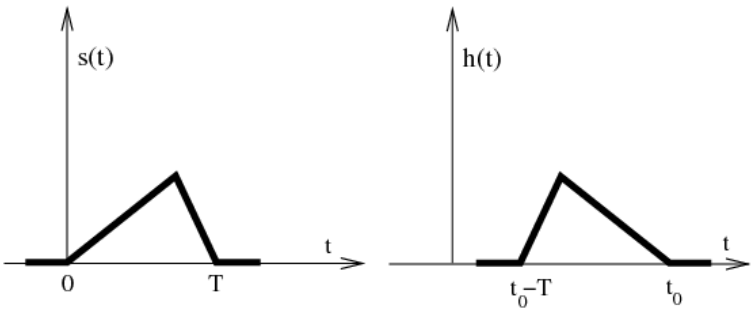


Figura 8.1. *Exemplu de filtru adaptat la semnal - semnalul original și funția pondere a filtrului.*

Funcția de transfer a filtrului adaptat la semnal este:

$$H(\omega) = K \int_{-\infty}^{\infty} h(t) \cdot e^{-j\omega t} dt = K \int_{-\infty}^{\infty} s(-(t - t_0)) \cdot e^{-j\omega t} dt \tag{8.2}$$

Făcând schimbarea de variabilă $\tau = -(t - t_0)$ obținem:

$$H(\omega) = -K \int_{\infty}^{-\infty} s(\tau) \cdot e^{-j\omega(t_0 - \tau)} d\tau = K \cdot e^{-j\omega t_0} \int_{-\infty}^{\infty} s(t) \cdot e^{j\omega\tau} d\tau \tag{8.3}$$

$$H(\omega) = KS^*(\omega) e^{-j\omega t_0} \tag{8.4}$$

unde $S(\omega)$ este transformata Fourier a semnalului $s(t)$, iar $S^*(\omega)$ complex conjugata acesteia.

O proprietate foarte importantă a acestui filtru este faptul că maximizează raportul semnal/zgomot [1].

8.2. Desfășurarea lucrării

Generați un semnal de 1000 de eșantioane, care conține un impuls dreptunghiular de lățime 100 de eșantioane, ca cel din Figura 8.2.

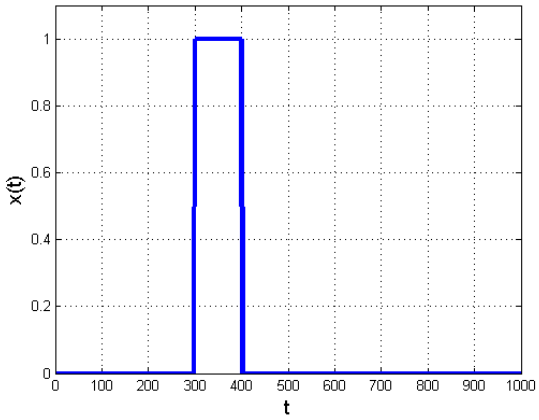


Figura 8.2. *Semnal impuls dreptunghiular.*

Pentru aceasta veți folosi următoarea secvență Matlab:

```
» x = zeros(1,1000);
» x(300:400) = 1;
```

Peste acest semnal se va suprapune zgomot, de exemplu, folosind funcția Matlab **rand**:

```
» x = x + rand(1,1000);
```

Semnalul afectat de zgomot va fi de forma celui din Figura 8.3.

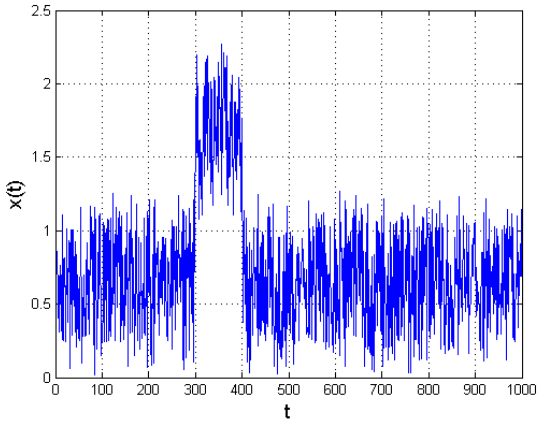


Figura 8.3. *Semnalul afectat de zgomot.*

Știind că semnalul $x(t)$ recepționat, conține un impuls dreptunghiular de durată 100 eșantioane, vom proiecta un filtru adaptat la semnal, având funcția pondere $h(t)$ ca cea din Figura 8.4, folosind secvența Matlab următoare.

```
» h = zeros(1,300);
» h(100:200)=1;
```

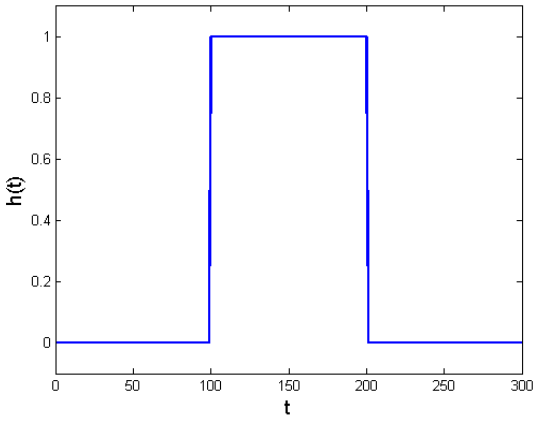


Figura 8.4. *Funcția pondere a filtrului adaptat la semnal.*

Semnalul $y(t)$ de la ieșirea filtrului adaptat la semnal (vezi Figura 8.5) se va calcula cu ajutorul funcției Matlab **conv**, care va realiza convoluția dintre semnalul $x(t)$ și funcția pondere $h(t)$ a filtrului.

```
» y = conv(x,h);
```

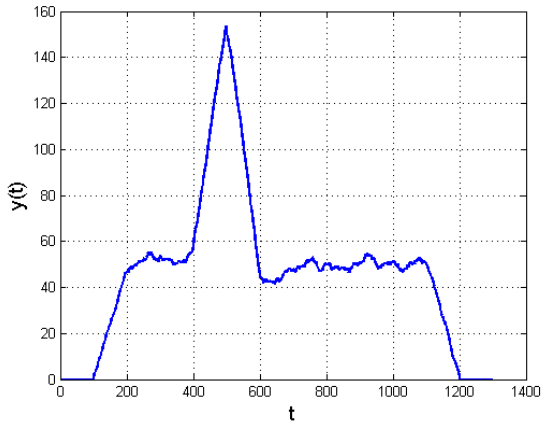


Figura 8.5. *Semnalul $y(t)$ de la ieșirea filtrului adaptat la semnal.*

Valoarea maximă din semnalul $y(t)$ indică poziția în care filtrul adaptat la semnal a detectat un impuls dreptunghiular similar cu funcția pondere, adică locul în care potrivirea dintre semnalul $x(t)$ de la intrarea filtrului și "semnalul" $h(t)$ a fost cea mai bună. Semnalul original ar putea fi recuperat prin prăguirea semnalului $y(t)$.

8.3. Probleme

1. Generați un semnal care conține câteva impulsuri de formă dreptunghiulară, de aceeași lățime. Suprapuneți zgomot peste acest semnal, folosind funțiile Matlab **`rand`** și **`randn`**. Filtrați semnalul cu un filtru adaptat la semnal, proiectat în prealabil. Vizualizați rezultatul. Ce observați?

2. Aceeași problemă ca mai sus, pentru cazul în care semnalul conține impulsuri de formă triunghiulară.

Lucrarea 9

Analiza statistică a semnalelor

Scopul lucrării este acela de a familiariza studenții cu generarea semnalelor aleatoare cu parametri cunoscuți și cu o distribuție dată, determinarea funcției de repartiție și a densității de probabilitate, calcularea mediei și a variației.

9.1. Breviar teoretic

Un semnal aleator este un proces care se desfășoară în timp și este guvernat de legi probabilistice. Din punct de vedere matematic, un semnal aleator este o funcție de două variabile $\xi(k,t) = \xi^{(k)}(t)$, unde k ia valori în spațiul eșantioanelor, iar t ia valori pe axa reală a timpului. Funcția $\xi^{(k)}(t)$ face parte din mulțimea sau clasa de semnale $\xi(t)$ și se numește o „realizare particulară" a procesului $\xi(t)$.

Pentru a caracteriza un semnal aleator la un moment de timp t arbitrar sau pentru o realizare particulară, se folosesc funcțiile de repartiție și cea de densitate de probabilitate. Alte mărimi caracteristice larg utilizate sunt momentele statistice, cele mai importante fiind media și varianța.

Funcția de repartiție, definită într-un punct x, este probabilitatea ca variabila aleatoare la momentul t să fie mai mică sau egală decât pragul x:

$$F_\xi(x,t) = P\{\xi(t) \leq x\}$$

Densitatea de probabilitate este derivata funcției de repartiție, și anume:

$$w_\xi(x,t) = \frac{dF_\xi(x,t)}{dx}$$

Valori medii statistice larg utilizate în diverse aplicații:

1. Valoarea medie

$$\overline{\xi(t)} = \int_{-\infty}^{+\infty} x w_\xi(x,t) dx$$

2. Valoarea pătratică medie

$$\overline{\xi^2(t)} = \int_{-\infty}^{+\infty} x^2 w_\xi(x,t)dx$$

3. Varianța

$$\sigma^2(t) = \overline{\xi^2(t)} - \overline{\xi(t)}^2$$

9.2. Desfășurarea lucrării

Se vor genera trei tipuri de semnale aleatoare: $x(n)$ cu distribuție uniformă, $y(n)$ cu distribuție normală (Gaussiană) și $z(n)$ cu distribuție Rayleigh. Aceste trei semnale aleatoare vor avea media m și varianța σ^2 specificate și vor fi generate folosind următoarele formule [8]:

$$x(n) = m + \sqrt{3}(2\xi - 1)\sqrt{\sigma} \tag{9.1}$$

$$y(n) = m + \sqrt{2}\sqrt{-ln\xi}cos(2\pi\eta)\sqrt{\sigma} \tag{9.2}$$

$$z(n) = m + \sqrt{2}\sqrt{-ln\xi}\sqrt{\sigma} \tag{9.3}$$

unde funcția ξ reprezintă o variabilă aleatoare distribuită uniform în intervalul $[0, 1]$, obținută cu ajutorul funcţiei Matlab **rand**. Același lucru este valabil și pentru variabila η. De exemplu, puteți genera semnalele x, y și z ca fiind formate din $N = 1000$ eșantioane, pe baza a N valori ale variabilei aleatoare ξ (vezi Figura 9.1). Pentru aceasta puteți utiliza comanda Matlab următoare:

```
» xi = rand(1,1000);
```

Pentru generarea variabilei aleatoare cu distribuție Gaussiană puteți folosi secvența Matlab următoare:

```
» N = 1000;
» eta = rand(1,N);
» xi = rand(1,N);
» y = m + sqrt(2)*sqrt(-1*log(xi)).*cos(2*pi*eta)*sigma;
```

Pentru generarea unei variabile aleatoare cu distribuție Rayleigh, se folosește comanda următoare.

```
» z = m + sqrt(2)*sqrt(-1*log(xi))*s;
```

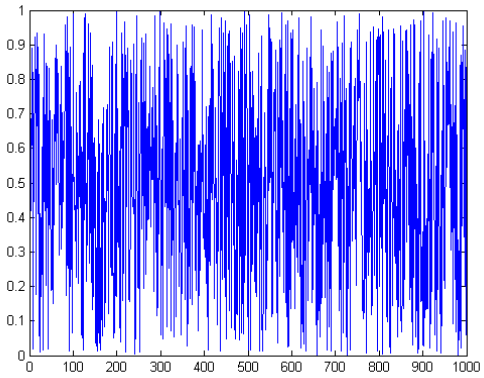


Figura 9.1. *Reprezentarea grafică a valorilor lui ξ.*

Se vor vizualiza diferite realizări particulare ale celor trei semnale, folosind funțiile **`figure`** și **`plot`** din Matlab, pentru mai multe valori ale mediei și ale variației.

Cu ajutorul funcțiilor **`mean`** și **`var`** din Matlab se vor calcula mediile și variațele celor trei semnale și se vor compara cu mediile și variațele specificate. Cum explicați diferențele?

Se va determina funcția de repartiție pentru fiecare din cele 3 realizări particulare ale semnalelor aleatoare, după algoritmul de mai jos:

- Se alege un număr N de nivele de cuantizare (de exemplu N = 100).
- În funcție de valorile minime și maxime ale semnalelor, determinate cu funcțiile Matlab **`min`**, respectiv **`max`**, se va calcula pasul de cuantizare:

$$\delta = \frac{max - min}{N}$$

- Se va genera un șir de N valori discrete ale lui x în intervalul $[min, max]$, cu pasul δ, după formula:

$$x_j = min + j\delta$$

  pentru $j = 1..N$.

- Se vor determina valorile funcției de repartiție, pornind de la definiție:

$$F_\xi(x_j) = P\{\xi \leq x_j\}$$

  pentru cele N valori discrete ale lui x în intervalul $[min, max]$, cu pasul δ. Probabilitatea se va calcula ca frecvență relativă de apariție, cu alte cuvinte ca raport între numărul de valori mai mici decât un anume x și numărul total de valori ale semnalului.

Se va reprezenta grafic funcția de repartiție.

Pornind de la definiție, pe baza funcției de repartiție se va determina funția de densitate de probabilitate, astfel:

$$w_\xi(x_j) = \frac{dF_\xi(x_j)}{dx_j}$$

Pentru cazul discret, se va folosi o aproximare a derivatei continue, dată de formula:

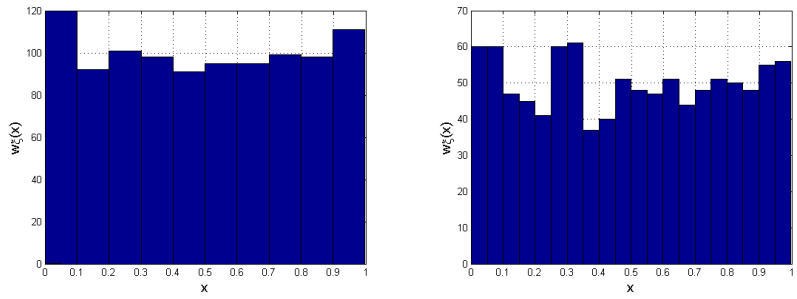
$$w_j = \frac{F_j - F_{j-1}}{x_j - x_{j-1}} = \frac{F_j - F_{j-1}}{\delta}$$

pentru $j = 2..N$.

Reprezentați grafic funcția de densitate de probabilitate.

Să se calculeze histograma valorilor semnalelor, folosind funcția Matlab **`hist`** și să se reprezinte grafic. Histograma reprezintă un estimat al funcției de densitate de probabilitate. Comparați rezultatul obținut cu graficul anterior. Ce observați?

În figurile 9.2 și 9.3 puteți observa forma histogramei pentru o variabilă aleatoare distribuită uniform, respectiv pentru una distribuită normal, pentru doă valori ale lui N. Cum explicați diferențele? De notat faptul că funcția **`hist`** calculează histograma ne-normată.



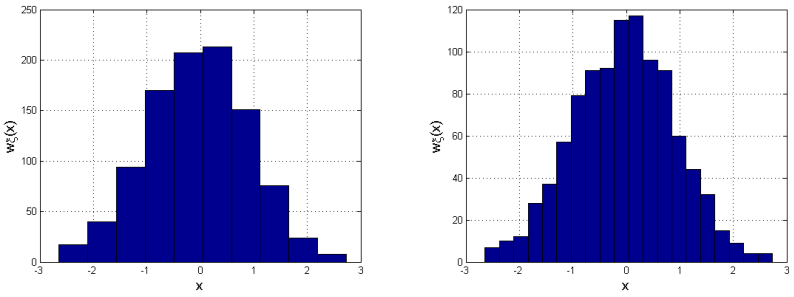
(a) N=10 (b) N=20

Figura 9.2. *Histograma unei variabile aleatoare distribuite uniform.*

Histograma cumulativă $h_c(x)$ reprezintă estimatul funcției de repartiție și se calculează pe baza histogramei $h(x)$ folosind formula:

$$h_c(x) = \int_{-\infty}^{x} h(\tau)d\tau$$

În cazul discret, integrala se transformă într-o sumă, iar formula de calcul a histogramei cumulative devine:



(a) N=10 (b) N=20

Figura 9.3. *Histograma unei variabile aleatoare distribuite normal.*

$$h_x(x) = \sum_{k=-\infty}^{x} h(k)$$

Calculați și reprezentați grafic histograma cumulativă. Ce observați?

9.3. Probleme

1. Determinați și reprezentați grafic histograma pentru următoarea secvență de valori aleatoare reprezentate pe 2 biți: 1 1 3 2 3 2 0 0 0 0 0 1 1 0 3 0 2 0 0 0 0 3 3 3.

2. Calculați și reprezentați grafic histograma cumulativă.

Lucrarea 10

Corelația semnalelor

Scopul lucrării este acela de a studia funcțiile de autocorelație și intercorelație în cazul unor semnale aleatoare cu densitate de probabilitate uniformă, normală și Rayleigh.

10.1. Breviar teoretic

Funcția de autocorelație

Funcția de autocorelație pentru un semnal aleator $\xi(t)$ se definește ca fiind corelația dintre $\xi(t_1)$ și $\xi(t_2)$, $\forall t_1, t_2 \in \mathbb{R}$, adică dintre ξ la momentul de timp t_1 și ξ la momentul t_2:

$$R_\xi(t_1, t_2) = \overline{\xi(t_1)\xi(t_2)} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x_1 x_2 w_2(x_1, x_2; t_1, t_2) dx_1 dx_2 \tag{10.1}$$

În ipoteza că semnalul $\xi(t)$ este staționar, atunci funcția de autocorelație va depinde doar de diferența de timp dintre t_1 și t_2:

$$R_\xi(\tau) = \overline{\xi(t)\xi(t+\tau)} \tag{10.2}$$

unde $\tau = t_1 - t_2$.

Un estimat al funcției de autocorelație [7] pentru o secvență aleatoare $x(n)$ este dat de formula:

$$R_\xi(m) = \frac{1}{N}\sum_{n=0}^{N-m-1} x(n)x(n+m) \tag{10.3}$$

Proprietățile funcției de autocorelație

Funcția de autocorelație are următoarele proprietăți:

1. Funcția de autocorelație este pară:

$$R_\xi(\tau) = R_\xi(-\tau)$$

2. Funcția de autocorelație este maximă în origine:

$$R_\xi(0) \geq |R_\xi(\tau)|$$

3. În ipoteza că nu există componente periodice sau deterministe, valoarea funcției de autocorelație la infinit este egală cu pătratul mediei semnalului:

$$R_\xi(\infty) = \overline{\xi}^2$$

4. Media pătratică și varianța semnalului se obțin din funcția de autocorelație astfel:

$$\overline{\xi^2} = R_\xi(0)$$

$$\sigma_\xi^2 = R_\xi(0) - R_\xi(\infty)$$

5. Dacă semnalul aleator este periodic, atunci și funcția lui de autocorelație este periodică, având aceeași perioadă:

$$\xi(t) = \xi(t+T) \rightarrow R_\xi(\tau) = R_\xi(\tau+T)$$

Funcția de intercorelație

Fie $\xi(t)$ și $\eta(t)$ două semnale staționare. Funcția de intercorelație între $\xi(t)$ și $\eta(t)$ se definește ca fiind corelația dintre ξ la momentul de timp t_1 și η la momentul t_2:

$$R_{\xi\eta}(t_1, t_2) = \overline{\xi(t_1)\eta(t_2)} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x_1 y_2 w_2(x_1, y_2; t_1, t_2) dx_1 dy_2 \tag{10.4}$$

În ipoteza de staționaritate, funcția de intercorelație va depinde la rândul ei doar de diferența dintr momentele de timp t_1 și t_2:

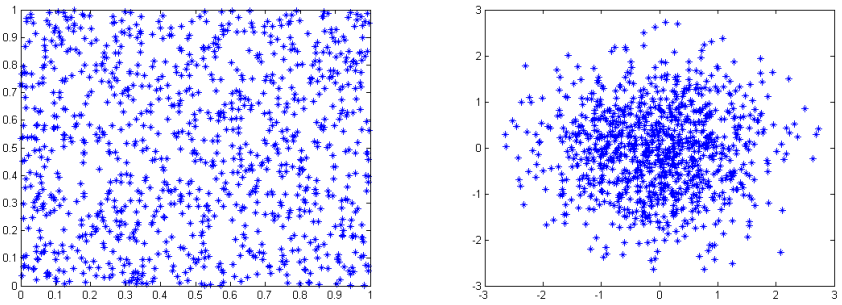
$$R_{\xi\eta}(\tau) = \overline{\xi(t)\eta(t+\tau)} \tag{10.5}$$

Un estimat al funcției de intercorelație [7] pentru două secvențe aleatoare discrete $x(n)$ și $y(n)$ este dat de formula:

$$R_{\xi\eta}(m) = \frac{1}{N}\sum_{n=0}^{N-m-1} x(n)y(n+m) \tag{10.6}$$

Diagrama în spațiul stărilor

Dacă ne interesează să apreciem calitativ dependența dintre două variabile aleatoare, ξ și η, atunci când nu cunoaștem funțiile de densitate de probabilitate care caracterizează perechea de variabile, se poate folosi *diagrama în spațiul stărilor*. Aceasta reprezintă un nor de puncte într-un spațiu bidimensional, format din perechi $(\xi_{(i)}, \eta_{(i)})$ ale realizărilor particulare ale perechii de variabile. Forma acestui nor furnizează informații despre corelația dintre cele două semnale: un nor format din puncte haotic distribuite în spațiu indică variabile independente din punct de vedere statistic (vezi Figura 10.1), pe când unul *ordonat* indică o anumită dependență între cele două variabile.



(a) (b)

Figura 10.1. *Diagrama de stare pentru două variabile independente distribuite a) uniform; b) normal.*

Diagrama în spațiul stărilor asociată unui semnal aleator este o reprezentare bidimensională a unei variabile aleatoare care reprezintă eșantionul de la momentul t al semnalului în funcție de variabila aleatoare asociată semnalului la momentul $t - p$. Reprezentarea se face pe o durată de observație T.

Cu ajutorul diagramei în spațiul stărilor putem vizualiza legătura între cele două variabile aleatoare pentru diferite intervale de "întârziere" p, obținând o evaluare calitativă referitoare la corelația conținută în semnal și la natura (aleatoare sau deterministă) a semnalului.

Pentru semnalele deterministe aspectul diagramei corespunde legăturilor funcționale dintre eșantioanele semnalului, diagrama fiind o reprezentare grafică a acestora. Pentru semnalele aleatoare aspectul diagramei se prezintă sub forma unei mulțimi de puncte care pot fi incluse într-un contur închis cu o anumită formă. Această formă ne poate da informații cu privire la corelația dintre eșantioane.

10.2. Desfășurarea lucrării

Se vor genera trei secvențe de numere aleatoare, cu media și dispersia cunoscute (vezi lucrarea precedentă). Pentru fiecare din cele trei secvențe aleatoare se va reprezenta

grafic diagrama în spațiul stărilor pentru diverse valori ale lui p. Acest lucru se va realiza folosind următoarea instrucțiune Matlab: `plot( x(p:n), x(1:n-p) )`, unde n este dimensiunea secvenței aleatoare.

Pentru fiecare din cele trei secvențe aleatoare se va determina și reprezenta grafic funcția de autocorelație, folosind funcția Matlab `xcorr`. Să se implementeze o funcție de calcul a autocorelației, pe baza formulei:

$$R_\xi(m) = \frac{1}{N} \sum_{n=0}^{N-m-1} x(n)x(n+m)$$

Se determina și reprezenta grafic funcția de intercorelație pentru două dintre secevențele aleatoare generate anterior, folosind funcția Matlab `xcorr`. Să se implementeze o funcție de calcul al intercorelației pe baza formulei:

$$R_{\xi\eta}(m) = \frac{1}{N} \sum_{n=0}^{N-m-1} x(n)y(n+m)$$

Se va considera cazul unui semnal determinist obținut la ieșirea unui sistem descris de ecuația:

$$x(n) = cx(n-1)[1 - x(n-1)]$$

Să se genereze acest semnal determinist, pornind de la $x(0) = 0.2$ și $c = 4$. Să se reprezinte grafic diagrama în spațiul stărilor pentru această secvență deterministă. Să se determine și reprezinte grafic funcția sa de autocorelație.

Calculați funcția de auto-corelație pentru un semnal cu distribuție Gaussiană, folosind funcția Matlab `xcorr`. Rezolvare:

```
» x = randn(1,1000);
» plot( xcorr(x,x) );
```

10.3. Probleme

1. Cum explicați diferențele de aspect între diagramele în spațiul stărilor pentru cele trei tipuri de semnale aleatoare?

2. Cum explicați aspectul diagramei în spațiul stărilor pentru semnalul determinist.

3. Verificați proprietățile funcției de autocorelație pentru fiecare caz analizat în lucrare.

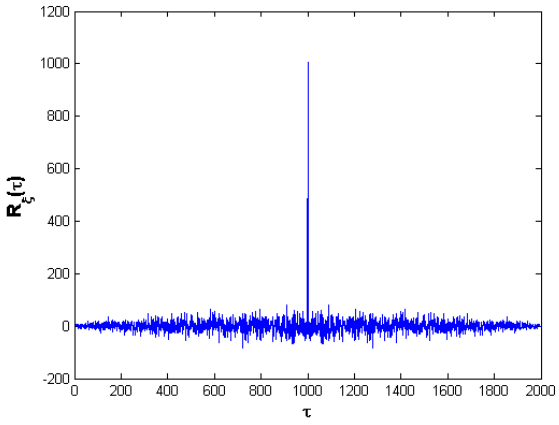


Figura 10.2. *Funcția de autocorelație pentru un semnal aleator distribuit normal.*

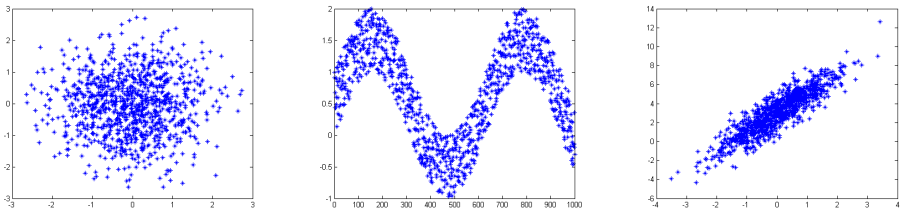
Lucrarea 11

Dreapta de regresie

11.1. Breviar teoretic

În această lucrare ne propunem să studiem cantitativ dependența statistică dintre două variabile aleatoare, concret să determinăm gradul de dependență liniară dintre ele prin calcularea coeficientului de corelație și determinarea dreptei de regresie.

Reamintim faptul că diagrama în spațiul stărilor pentru două variabile aleatoare permitea aprecierea calitativă a dependenței dintre acestea. În Figura 11.1 puteți observa forma norului de puncte pentru trei cazuri: a) atunci când cele două variabile aleatoare sunt independente din punct de vedere statistic; b) când între variabile există o anumită dependență funcțională, dar acestea sunt decorelate și c) când variabilele sunt corelate, și deci sunt și dependente.



(a) (b) (c)

Figura 11.1. *Forma norului de puncte pentru două variabile a) independente statistic; b) dependente dar decorelate și c) dependente și corelate.*

Dacă diagrama în spațiul stărilor indică faptul că între cele două variabile aleatoare ar exista o dependență aproximativ liniară, de forma celei din Figura 11.1c) atunci se poate determina dreapta de regresie (vezi Figura 11.3) care ar putea aproxima cel mai bine această dependență [1].

Ecuația dreptei de regresie, care minimizează eroarea pătratică medie [1] este:

$$\hat{\eta} = \frac{K_{\xi\eta}}{\sigma_\xi^2}(\xi - \overline{\xi}) + \overline{\eta} \tag{11.1}$$

unde $K_{\xi\eta}$ reprezintă covariația dintre cele două variabile, $\overline{\xi}$ este media statistică a lui ξ, iar $\overline{\eta}$ media lui η.

Covariația dintre ξ și η reprezintă momentul centrat mixt de ordinul doi, $K_{\xi\eta} = \overline{(\xi - \overline{\xi})(\eta - \overline{\eta})}$ și diferă de corelație doar printr-o constantă: $K_{\xi\eta} = R_{\xi\eta} - \overline{\xi}\overline{\eta}$. Două variabile se numesc *decorelate* atunci când $K_{\xi\eta} = 0$. Două variabile aleatoare independente sunt implicit decorelate.

Coeficientul de corelație

Pornind de la expresia erorii pătratice medii minime dintre variabila aleatoare η și cea care o aproximează cel mai bine, $\hat{\eta}$:

$$\varepsilon_{min} = \overline{(\eta - \hat{\eta})^2} = \sigma_\eta^2 \left[1 - \left(\frac{K_{\xi\eta}}{\sigma_\xi \sigma_\eta}\right)^2\right] \tag{11.2}$$

se definește coeficientul de corelație, notat cu $\rho_{\xi\eta}$:

$$\rho_{\xi\eta} = \frac{K_{\xi\eta}}{\sigma_\xi \sigma_\eta} \tag{11.3}$$

și care reprezintă valoarea normată a covariației dintre cele două variabile aleatoare. În acest fel, coeficientul de corelație permite cuantificarea absolută a gradului de dependență liniară dintre ξ și η.

Dacă rescriem expresia erorii pătratice medii ca fiind:

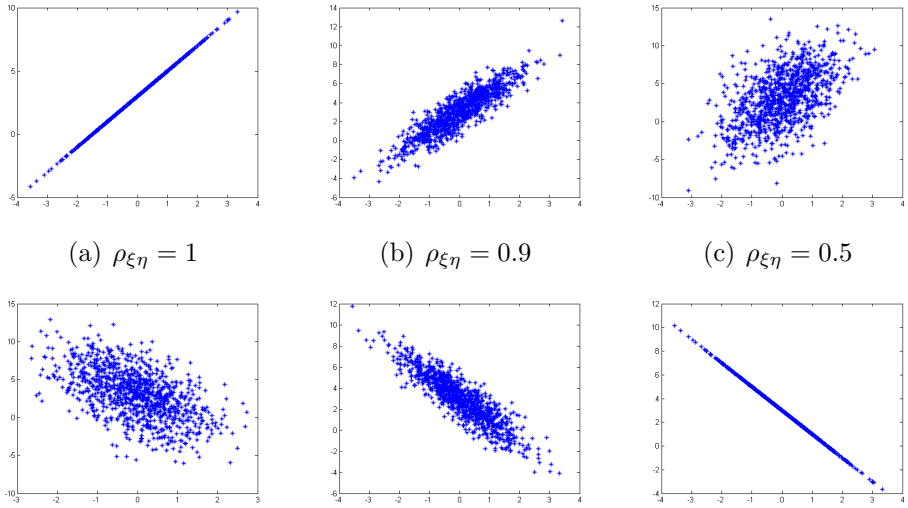
$$\varepsilon_{min} = \sigma_\eta^2(1 - \rho_{\xi\eta}^2) \tag{11.4}$$

se poate observa că $\rho_{\xi\eta}$ trebuie să fie de modul subunitar, pentru ca eroarea pătratică să fie pozitivă (fiind o sumă de cantități pozitive): $|\rho_{\xi\eta}| \leq 1$. Cu cât $|\rho_{\xi\eta}|$ este mai aproape de 1, cu atât eroarea de aproximare a lui η cu $\hat{\eta}$ e mai mică, deci gradul de dependență liniară dintre variabile este mai mare. Daca $|\rho_{\xi\eta}| = 1$ atunci $\varepsilon = 0$ – norul de puncte este chiar o dreaptă, dependența dintre cele două variabile fiind perfect liniară.

În Figura 11.2 puteți observa forma norului de puncte, sau a diagramei în spațiul stărilor, pentru două variabile aleatoare ξ și η, pentru diverse valori ale gradului de dependență liniară dintre acestea.

11.2. Desfășurarea lucrării

Să se genereze un semnal aleator $x(n)$ cu distribuție normală, și să se construiască un semnal $y(n)$ de forma: $y(n) = a \cdot x(n) + b + k \cdot \xi$, unde ξ este o variabilă aleatoare



(a) $\rho_{\xi\eta} = 1$ (b) $\rho_{\xi\eta} = 0.9$ (c) $\rho_{\xi\eta} = 0.5$

(d) $\rho_{\xi\eta} = -0.5$ (e) $\rho_{\xi\eta} = -0.9$ (f) $\rho_{\xi\eta} = -1$

Figura 11.2. *Forma diagramei în spațiul stărilor pentru ξ și η, pentru diverse valori ale coeficientului de corelație.*

distribuită normal, iar k un factor de ponderare. Să se reprezinte grafic diagrama în spațiul stărilor, pentru cele două semnale. Să se calculeze coeficientul de corelație dintre cele două semnale, folosind funcția Matlab **`corrcoef`** și să se determine dreapta de regresie, folosind funcția **`robustfit`**. Să se reprezinte grafic dreapta de regresie, în aceeași figură cu norul de puncte.

```
» x = randn(1,10000);
» y = 2*x + 3 + randn(1,1000);
» corrcoef(x,y)
ans =

    1.0000    0.8949
    0.8949    1.0000

» b = robustfit(x,y,'ols')
ans =
    3.0232
    2.0082
```

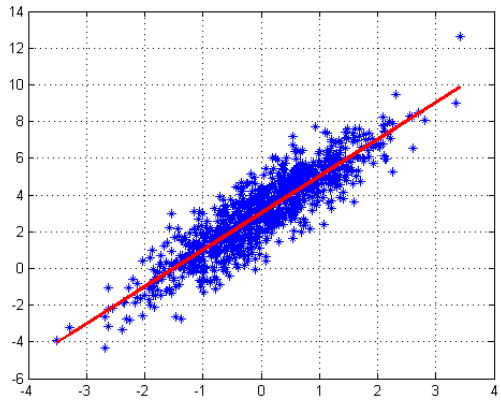


Figura 11.3. *Diagrama în spațiul stărolor și reapta de regresie.*

11.3. Probleme

1. Repetați cerințele din secțiunea de desfășurare a lucrării pentru diverse distribuții ale semnalelor $x(n)$ și $y(n)$, variind factorul k de ponderare și dând diverse valori constantelor a și b.

2. Calculați dreapta de regresie folosind celelalte metode disponibile pentru funcția `robustfit`: Andrews, Cauchy, Talwar etc.

3. Implementați o funcție care să calculeze coeficientul de corelație, pe baza formulei sale de definiție.

Anexă

Lista funcțiilor Matlab des întâlnite:

`abs`	Modul
`acos`	Arccosinus
`all`	Verifică dacă toate valorile unei matrici sunt nenule
`angle`	Faza unei valori complexe
`any`	Verifică dacă cel puțin o valoare dintr-o matrice este nenulă
`asin`	Arcsinus
`atan`	Arctangentă
`axis`	Modificarea axelor unui grafic
`butter`	Crearea unui filtru Butterworth
`cd`	Schimbare director de lucru
`ceil`	Rotunjire în sus
`cheby1`	Crearea unui filtru Chebyshev de tipul I
`cheby2`	Crearea unui filtru Chebyshev de tipul II
`colormap`	Crearea unei hărți de culoare pentru afișarea suprafețelor
`conv`	Convoluția între două semnale
`cos`	Cosinus
`czt`	Transformata z
`det`	Determinant
`diag`	Creare o matrice cu valori nenule pe diagonală
`ellip`	Crearea unui filtru eliptic
`exist`	Existența unei variabile
`exit`	Ieșire din Matlab
`exp`	Exponențială
`fft`	Trasformat Fourier Rapidă
`filter`	Filtrarea unui semnal
`find`	Găsește valorile nenule unei matrici
`fir1`	Crearea unui filtru FIR

`floor`	Rotunjire în jos
`freqz`	Răspunsul în frecvență a unui filtru
`grid`	Afișarea liniilor ajutătoare pe grafic
`help`	Afișarea documentației asociate unei funcții
`inv`	Inversa unei matrici
`isempty`	Verfică dacă o matrice este vidă
`mesh`	Afișarea unei suprafețe prin model wireframe
`meshgrid`	Crearea matricilor din combinația unor vectori pentru afișarea suprafețelor
`length`	Lungimea unui vector
`log`	Logaritm natural
`log10`	Logaritm bază 10
`log2`	Logaritm bază 2
`ones`	Creare o matrice cu valori de 1
`plot`	Afișarea unui grafic
`pwd`	Afișare director de lucru
`sawtooth`	Crearea unei forme de undă triunghiulară
`sin`	Sinus
`size`	Dimensiunea unei matrici
`sqrt`	Radical
`square`	Crearea unei forme de undă dreptunghiulară
`subplot`	Crearea mai multor grafice pe o figură
`surf`	Afișarea unei suprafețe solide
`rand`	Creare o matrice cu numere aleatoare
`rem`	Restul împărții
`repmat`	Creare unei matrici mai mari prin replicarea unei matrici
`round`	Rotunjire
`tan`	Tangentă
`title`	Modificarea titlului graficului
`unwrap`	Corectarea unui vector de faze
`xlabel`	Modificarea etichetei de pe axa Ox
`ylabel`	Modificarea etichetei de pe axa Oy
`zeros`	Creare o matrice cu valori de 0

Pentru o descriere mai detaliată a acestor funcții consultați documentația lor disponibilă folosind comanda `help`.

Bibliografie

[1] Ciuc, M., Vertan, C., *Prelucrarea Statistică a Semnalelor*, Editura MatrixRom, 2005

[2] Gâlmeanu, H., *Bazele procesării și transmiterii semnalelor II – Îndrumar de laborator*, Editura Universității Transilvania, Brașov, 2002

[3] Kalechman, M., *Practical Matlab Applications for Engineers*, CRC Press, Boca Raton, FL, 2008

[4] Kalechman, M., *Practical Matlab Basics for Engineers*, CRC Press, Boca Raton, FL, 2008

[5] Murgan, A.T., Dogaru, R., Comaniciu, C., *Teoria Transmisiunii Informației: Detecția, Estimarea și Filtrarea Semnalelor Aleatoare – lucrări practice*, Editura POLITEHNICA București, 1995

[6] Murgan, A.T., Spânu, I., Gavăt, I., Sztojanov, I., Neagoe, V.E., Vlad, A., *Teoria Transmisiunii Informației – probleme*, Editura Didactică și Pedagogică, București, 1983

[7] Oppenheim, A.V., Schafer, R.W., *Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, New Jersey, 1975

[8] Press, W.H., Flannery, B.P., Teukolsky, S.A., Vetterling, W.T., *Numerical Recipes in C - The Art of Scientific Computing*, 2 ed., Cambridge University Press, 1992

[9] Smith, J.O., *Introduction to Matlab and Octave*, Center for Computer Research in Music and Acoustics, Stanford University, Stanford, CA, februarie 2009

[10] Smith, S.W., *The Scientist and Engineer's Guide to Digital Signal Processing*, 2 ed., California Technical Publishing, San Diego, 1999

[11] Spătaru, A., *Teoria Transmisiunii Informației*, Editura Didactică și Pedagogică, București, 1983

[12] Stahel, A., *Octave at HTI Biel*, University of Engineering and Information Technology, Biel, 2008