CIMPOERU ANA-MARIA

UNIVERSITATEA POLITEHNICA DIN BUCURESTI
FACULTATEA DE ELECTRONICĂ, TELECOMUNICAȚII SI TEHNOLOGIA INFORMAȚIEI

Proiect 3
Modulații Digitale

Tema Proiectului

Să se proiecteze un simulator pentru semnale cu modulaţie digitală care va conţine un modulator şi demodulatorul aferent, ce pot fi conectate între ele printr-un canal ideal, de caracteristică unitară, afectat de un zgomot aditiv gaussian alb, de medie nulă şi densitate spectrală de putere $N_0/2$ (prin intermediul unui raport semnal zgomot ce va fi dată de intrare, in dB). Simulatorul va permite totodată simularea transmisiunii şi în absenţa zgomotului. Simulatorul va fi deschis, permiţând conectarea şi altor tipuri de canale de comunicaţie, trebuind să se specifice formatul în care acestea pot fi simulate.

Simulatorul va permite aplicarea la intrare a unei secvenţe binare predefinite de date, furnizate într-un format adecvat, sau va permite generarea de semnale aleatoare binare de o lungime L ce constituie dată de intrare, secvenţă ce se va aplica modulatorului.

Semnalul modulat generat va putea fi reprezentat grafic pe un număr de perioade de bit K ce constituie dată de intrare, într-un domeniu precizat.

Detectorul utilizat va fi un detector optimal urmat de un detector cu prag, urmat de un bloc de calcul ce va permite comparaţia datelor estimate cu cele aplicate la emisie, incluzând un numărător de erori şi un estimator al probabilităţii de eroare.

Demodulatorul va putea lucra cu informaţii de sincronizare ideale (parametrii utilizaţi la emisie) sau va include un bloc de sincronizare (tact sau purtătoare) cu o structură cât mai simplă.

Se recomandă ca simulatorul să prezinte o interfaţă grafică pentru introducerea datelor şi afişarea/prezentarea rezultatelor simulării.

Se va analiza si se va propune explicit o metoda de a transpune modulatorul într-un FPGA.

Simulatorul va fi realizat într-un limbaj de programare / mediu de simulare care este la latitudinea celui care realizează tema.

Informatiile de timp se dau in perioade de bit iar cele de frecventa se raporteaza la rata de bit.

Datele Temei :

- Tipul modulației: QPSK
- Raportul semnal zgomot:RSZ=10...100 dB
- Tipul de zgomot care afectează canalul: zgomot alb gausian, de medie nulă şi densitate spectrală de putere $N_0/2$ (prin intermediul unui raport semnal zgomot ce va fi dată de intrare, in dB).
- Modulație în banda de bază, folosind cosinus ridicat
- Alpha = 0.3
- Sincronizare tact
- L = 10….100
- K = 10….100

Resusrse Folosite : MATLAB R2020a

Partea Teoretică :

Modulația QPSK

QPSK este o schemă de codificare de tipul M-ary, unde N = 2 și M= 4 (prin urmare, numele "cuaternar" însemnând "4"). Un modulator QPSK este un semnal binar (baza 2), folosit pentru a produce patru combinații diferite de intrare,: 00, 01, 10 și 11.

În modulația de fază în cuadratură (QPSK) două sinusoide (sinus și cosinus) sunt utilizate ca funcții de bază pentru modulație. Modulația este realizată prin varierea fazei funcțiilor de bază care depind de simbolurile mesajului transmis.

În QPSK, modulația este bazată pe simboluri, unde un simbol conține 2 biți. Prin urmare, cu QPSK, datele de intrare binare sunt combinate în grupuri de doi biți, numit dibiți. În modulator, fiecare cod de dibit generează una dintre cele patru faze de ieșire posibile (+45°, +135°, -45° și -135°) sau

(π/4, 3π/4, 5π/4, 7π/4)

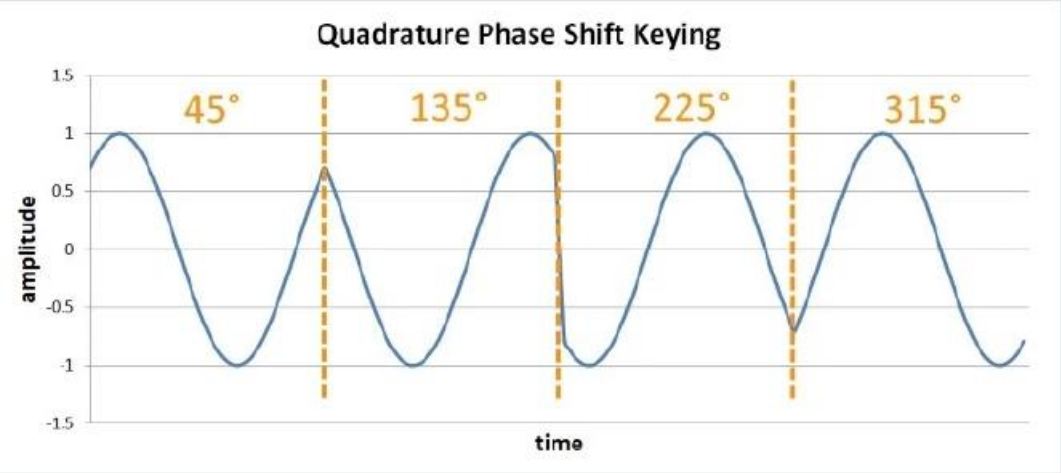


Fig. 2.11 Modulația QPSK

Sursa :

https://www.researchgate.net/publication/322926899_chapter_III_BPSK_and_QPSK_modulation_and_demodulation_with_Simulink

Ecuația de mai jos descrie tehncica de modulație QPSK :

$$s_i(t) = \sqrt{\left(\frac{2E_s}{T}\right)} * \cos\left[2\pi f_c t + (2n-1)\frac{\pi}{4}\right]; \qquad n = 1,2,3,4$$

QPSK este utilizat pentru transmiterea prin satelit a video MPEG2, modemuri de cablu, videoconferințe, sisteme de telefonie mobilă, și alte forme de comunicare digitală pe un suport RF.

Diagrama constelației QPSK va arăta punctele constelației situate atât pe axa x, cât și pe axa y. Acesta înseamnă că semnalul modulat QPSK va avea o componentă în fază (I) și, de asemenea, o componentă în cuadratură (Q).

Motivul pentru care am ales utilizarea 45°, 135°, 225° și 315° este acela că acestea sunt mai usor de generat folosind tehnicile de modulare I/Q (în fază / în cuadratură), deoarece însumarea semnalelor I și Q, care sunt inversate sau neinversate, rezultă aceste patru schimbări de fază.

I	Q	Schimbări de fază I+Q
Neinversat	Neinversat	45°
Inversat	Neinversat	135°
Inversat	Inversat	225°
Neinversat	Inversat	315°

Tabel 2.2 Schimbările celor 4 faze în QPSK

Un modulator QPSK poate fi implementat după cum urmează. Un multiplexor (sau convertor serial - paralel) este utilizat pentru a separa biții impari si pari de biții de informatie generați. Fiecare dintre biții impari (componenta în cuadratură) și biții pari (componenta în fază) sunt convertiți în format NRZ în mod paralel. Semnalul de pe brațul în fază este multiplicat cu componenta cosinusoidală, iar semnalul de pe brațul cuadraturii este înmulțit cu componenta sinusoidală. Semnalul modulat QPSK este obținut prin adăugarea semnalului de la ambele brațe în fază și în cuadratură.

Demodulația QPSK:

Pentru demodulatorul QPSK, se folosește tehnica de detecție coerentă , în care informațiile privind frecvența și faza purtătoare trebuie să fie cunoscute de către receptor. Aceasta se poate realiza prin utlizarea unui PLL (circuit cu buclă închisă = phase lock loop) la recepție. Un PLL blochează, în esență, la frecvența de transport de intrare și urmărește variațiile de frecvență și de fază. Pentru următoarea simulare, nu este folosit un PLL, dar în schimb, presupunem că receptorul este în sincronizare perfectă. În scopu demonstrative vom presupune că recuperarea fazei purtătoare se face prin utilizarea frecvențelor de referință generate de receptor $[\cos(2\pi f_c t)]$ şi $[\sin(2\pi f_c t)]$.

În demodulator, semnalul primit este multiplicat de frecvență de referință a generatorului $[\cos(2\pi f_c t)]$ şi $[\sin(2\pi f_c t)]$ în componente separate (în fază și în cuadratură). Rezultatul înmulțit pe fiecare component este integrat pe o perioadă de un bit cu ajutorul unui integrator. Un detector de prag ia o decizie cu privire la fiecare bit integrat pe baza unui prag. În cele din urmă, biții de pe componenta în fază (biții pari) și de pe componenta în cuadratură (biți impari) sunt relegați pentru a forma fluxul de informații detectat.

Avantaje QPSK :

În comparație cu schemele de modulare care transmit un bit pe simbol, QPSK este avantajos în ceea ce privește eficiența lățimii de bandă. De exemplu, imaginați-vă un semnal analogic baseband într-un sistem BPSK (binar schimbare de fază keying). BPSK folosește două schimbări de fază posibile în loc de patru, și astfel se poate transmite doar un bit pe simbol. Semnalul benzii de bază are o anumită frecvență, iar în timpul fiecărei perioade de simbol, un bit poate fi transmis. Un sistem QPSK poate utiliza un semnal de

banda de bază de aceeași frecvență, dar transmite doi biți în timpul fiecărei perioade simbol. Astfel, eficiența lățimii de bandă este (ideal) mai mare cu un factor de doi.

Descrierea Simulatorului și a metodei alese :

Am ales să implementez simulatorul folosind Matlab, deoarece este un program complex ce are funcții multiple.

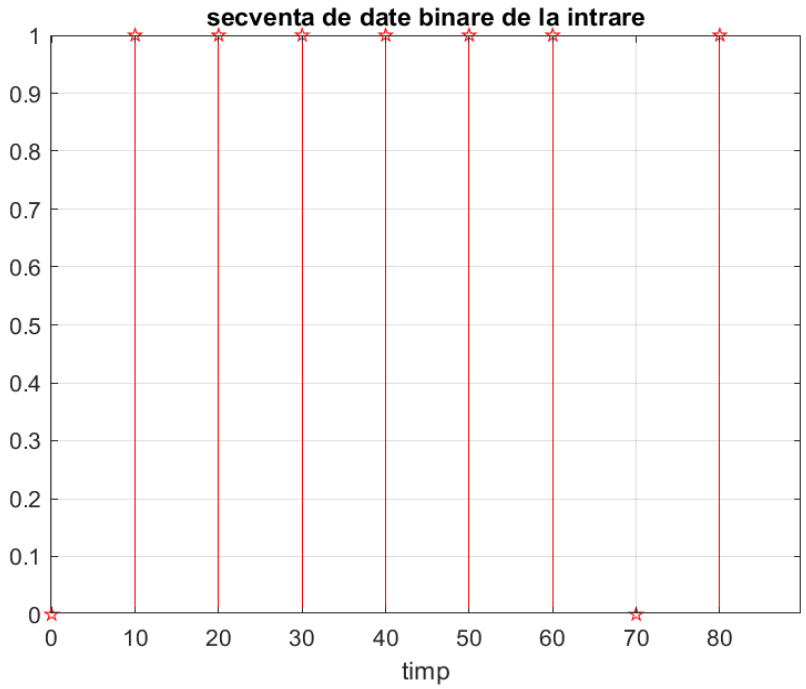
Codul scris în fereastra Editor este umrătorul :

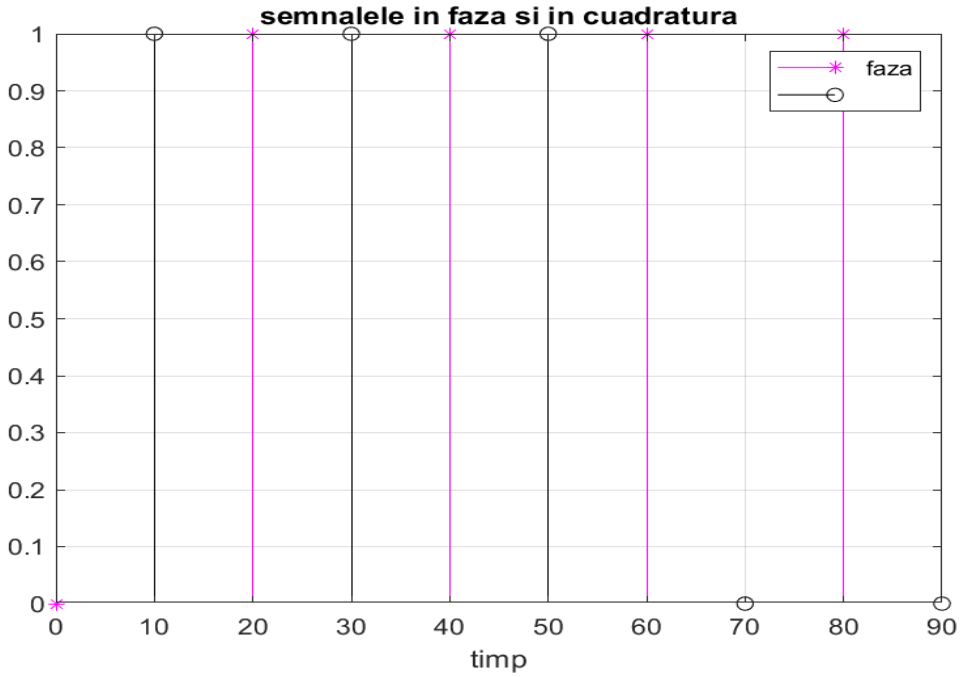
```
clear all;
close all;


L=10;         %lungimea secventei de intrare (valoarea data L, valoare intre 10 si
100) in cazul cand secventa este aleatoare
alfa=0.3;          %factorul de rotunjire al cosinusului ridicat
Tb=10;               %perioada de bit (K intre 10 si 100)
Ts=2*Tb;            %perioada de simbol
Fs=1/Ts;            %frecventa datelor de la intrarea filtrelor
Fes=10*Fs;           %frecventa de esantionare a filtrelor
RSZ=10;             %raportul semnal zgomot al canalului (RSZ intre 10 si 100 [dB] )
delay1=3*Ts;              %intarzierea introdusa de filtru
delay2=delay1*Ts;     %intarzierea de propagare a semnalului prin filtre

zgomot=1;           %daca zgomot=1 avem zgomot, daca zgomot=0 nu avem zgomot

% Date de intrare:

% Secventa binara aleatoare:
date_in(1:2)=randsrc(1,2,[0 1]);
for i=3:2:L-1
    date_in(i:i+1)=randsrc(1,2,[0 1]);
    while (date_in(i)~=date_in(i-2))&(date_in(i+1)~=date_in(i-1))

        date_in(i:i+1)=randsrc(1,2,[0 1]);
    end
end



tdate_in=[0:length(date_in)-1].*Tb;  %timpul corespunzator datelor de intrare
figure(1)
stem(tdate_in,date_in,'rp'),xlabel('timp'),grid on,title('secventa de date binare de
la intrare')


%Separarea componentelor in faza si in cuadratura:

date_f=date_in(1:2:length(date_in)); %componenta in faza
tdate_f=tdate_in(1:2:length(tdate_in));
date_q=date_in(2:2:length(date_in));  %componenta in cuadratura
tdate_q=tdate_in(2:2:length(tdate_in));

figure(2)
stem(tdate_f,date_f,'m*'),grid on,xlabel('timp'),title('semnalele in faza si in
cuadratura')
hold on,stem(tdate_q,date_q,'k'),hold off
legend('faza','','cuadratura','')
```

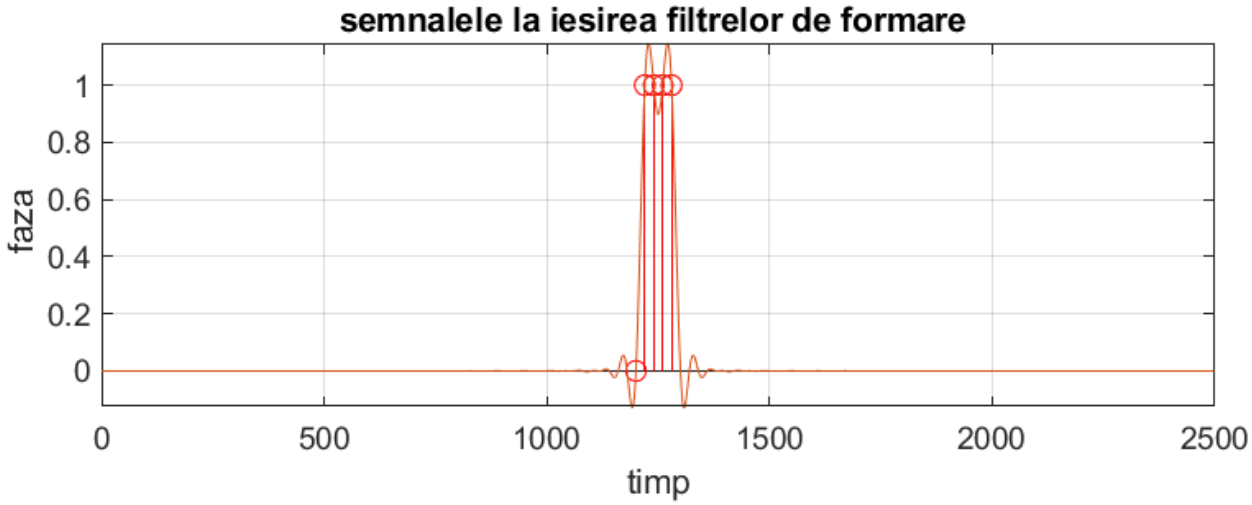
```
%Filtrarea cu cos ridicat:

[date_filtr_f,tfiltr]=rcosflt(date_f,Fs,Fes,'normal',alfa,delay1);
[date_filtr_q,tflitr]=rcosflt(date_q,Fs,Fes,'normal',alfa,delay1);

%reprezentarea datelor dupa filtare;

    tdate_f=tdate_f+delay2;  %se adauga timpul de propagare prin filtru
    tdate_q=tdate_q+delay2-Tb; %se adauga timpul de propagare prin filtru
    figure(3)
    subplot(2,1,1),stem(tdate_f,date_f,'r');hold on;grid on;title('semnalele la
iesirea filtrelor de formare ')
    subplot(2,1,1),plot(tfiltr,date_filtr_f),hold off,ylabel(' faza'),xlabel('timp')
    subplot(2,1,2),stem(tdate_q,date_q,'r'),hold on,grid on
    subplot(2,1,2),plot(tfiltr,date_filtr_q);hold off;ylabel('
cuadratura'),xlabel('timp')


%Modularea in banda de baza :

date_mod_f=date_filtr_f;
date_mod_q=date_filtr_q;

figure(4)
subplot(2,1,1),plot(tfiltr,date_mod_f,'r'),grid on,xlabel('timp'),ylabel('in
faza'),title('componentele semnalului dupa modulare'),axis tight
subplot(2,1,2),plot(tfiltr,date_mod_q,'r'),grid,xlabel('timp'),ylabel('in
cuadratura'),axis tight

%Semnalul la iesirea din modulator QPSK:

date_mod=date_mod_f-date_mod_q;
figure(5)
plot(tfiltr,date_mod,'r'),grid on,xlabel('timp'),title('semnalul la iesirea din
modulatorul QPSK')

%Simularea canalului:

if zgomot==1
    date_rec_zg=awgn(date_mod,RSZ,'measured'); %semnalul la care se adauga zgomot prin
fct "awgn"
elseif zgomot==0
    date_rec=date_mod; %semnalul nu este afectat de zgomot
else
    error('Eroare la introducerea zgomotului: RSZ trebuie sa fie intre 1 si
inf(infinit)')
end
trec=tfiltr;

figure(6)
plot(trec,date_rec_zg),grid,xlabel('timp'),title('semnalul dupa trecerea prin canal')

%Sincronizarea,demodularea si filtrarea:

date_dem_f=date_rec_zg;
date_dem_q=date_rec_zg;

figure(7)
subplot(2,1,1),plot(trec,date_dem_f),grid on,xlabel('timp'),ylabel('in
faza'),title('componentele semnalului dupa demodulare'),axis tight
subplot(2,1,2),plot(trec,date_dem_q),grid,xlabel('timp'),ylabel('in cuadratura'),axis
tight
```

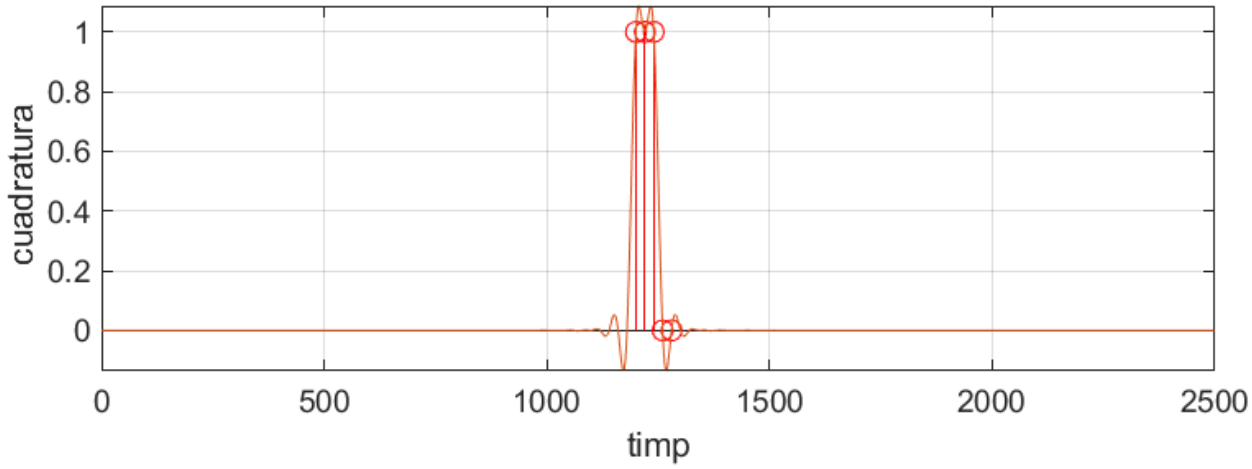
```
filtru=fir1(20,0.1);
date_dem_fil_f=filter(filtru,1,date_dem_f);
date_dem_fil_q=filter(filtru,1,date_dem_q);

tdate_f=tdate_f+Ts;  %se adauga un timp de intarziere prin filtru
tdate_q=tdate_q+Ts;  %se adauga un timp de intarziere prin filtru
figure(8)
subplot(2,1,1),stem(tdate_f,date_f,'r');hold on;grid on;title('semnalele demodulate')
subplot(2,1,1),plot(trec,date_dem_fil_f,'b'),hold off,ylabel('in faza'),xlabel('timp')
%legend('original','','decodat','',1)
subplot(2,1,2),stem(tdate_q,date_q,'r'),hold on,grid on
subplot(2,1,2),plot(trec,date_dem_fil_q,'b');hold off;ylabel('in
cuadratura'),xlabel('timp')
%legend('original','','decodat','',1)

%Esantionarea semnalului receptionat:

for i=1:length(tdate_f)
    for j=1:length(trec)
        if tdate_f(i)==trec(j)
            date_es_f(i)=date_dem_fil_f(j);
        end
    end
end

for i=1:length(tdate_q)
    for j=1:length(trec)
        if tdate_q(i)==trec(j)
            date_es_q(i)=date_dem_fil_q(j);
        end
    end
end

figure(9)
subplot(2,1,1),stem(date_es_f,'b'),hold on,grid on,title('semnalul
receptionat(albastru) si cel original(rosu)')
subplot(2,1,1),stem(date_f,'r*'),hold off,ylabel('in faza')
subplot(2,1,2),stem(date_es_q,'b'),hold on,grid on
stem(date_q,'r*'),hold off,ylabel('in cuadratura'),xlabel('numarul de ordine')


%Decizia:

for i=1:length(date_es_f)
    if date_es_f(i)<=0.5
        date_estimate_f(i)=0;
    else
        date_estimate_f(i)=1;
    end
end


for i=1:length(date_es_q)
    if date_es_q(i)<=0.5
        date_estimate_q(i)=0;
    else
        date_estimate_q(i)=1;
    end
end


for i=1:length(date_estimate_f)
    date_estimate(2*i-1)=date_estimate_f(i);
```

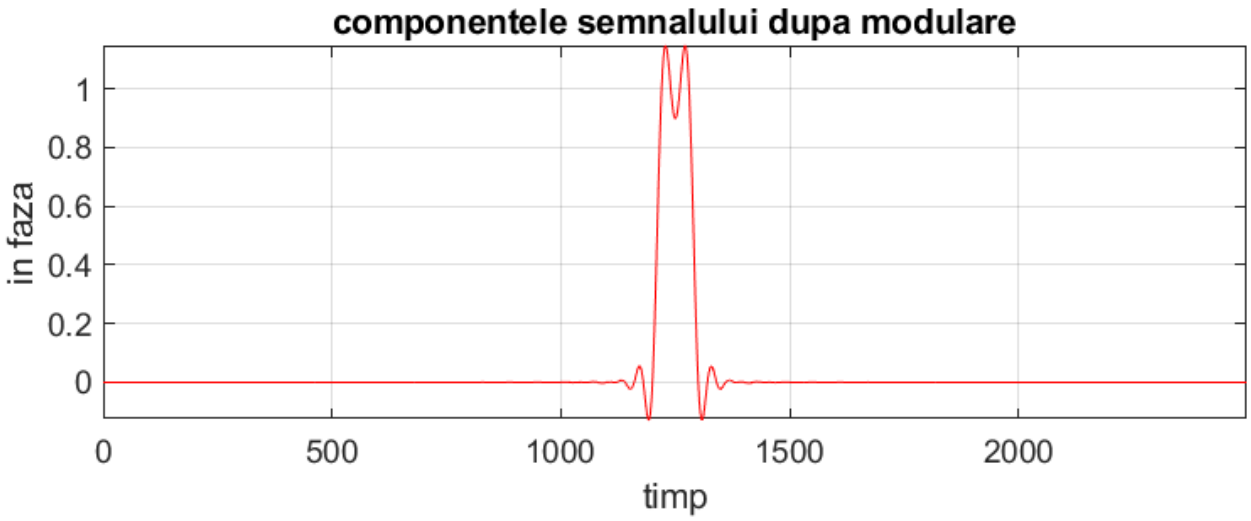
```
end
for i=1:length(date_estimate_q)
    date_estimate(2*i)=date_estimate_q(i);
end

figure(10)
stem(date_estimate,'o'),xlabel('numarul de ordine'),grid on,hold on,title('datele de
intrare si cele estimate la receptie')
stem(date_in,'r.'),hold off
%legend('semnalul estimat','','semnalul original','',0)


%Determinarea nr de erori

nr_er=0;
for i=1:L
    if date_in(i)~=date_estimate(i)
        nr_er=nr_er+1;
    end
end
nr_er
prob_de_eroare=nr_er/length(date_estimate)
```

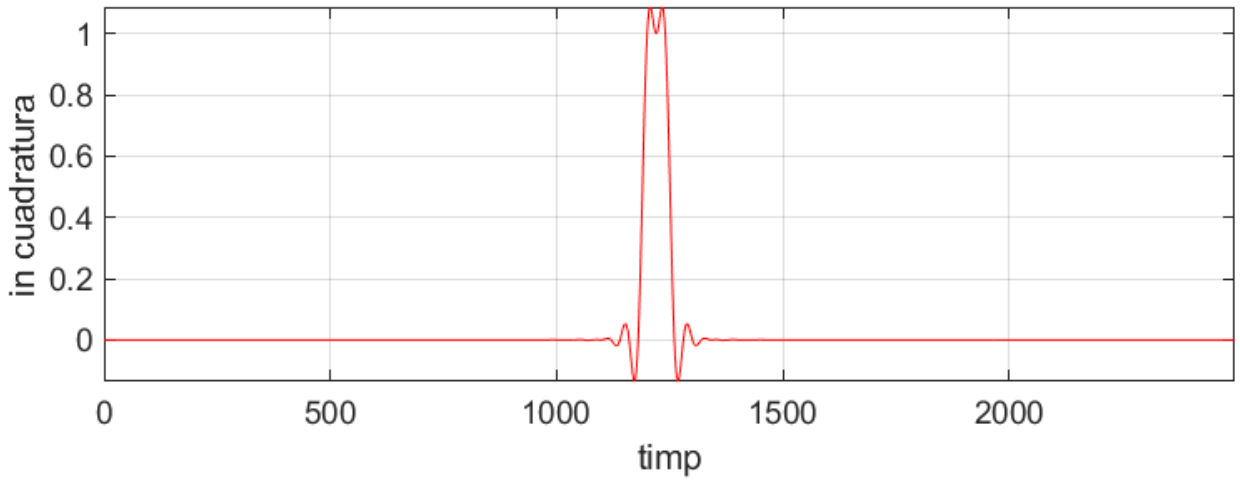


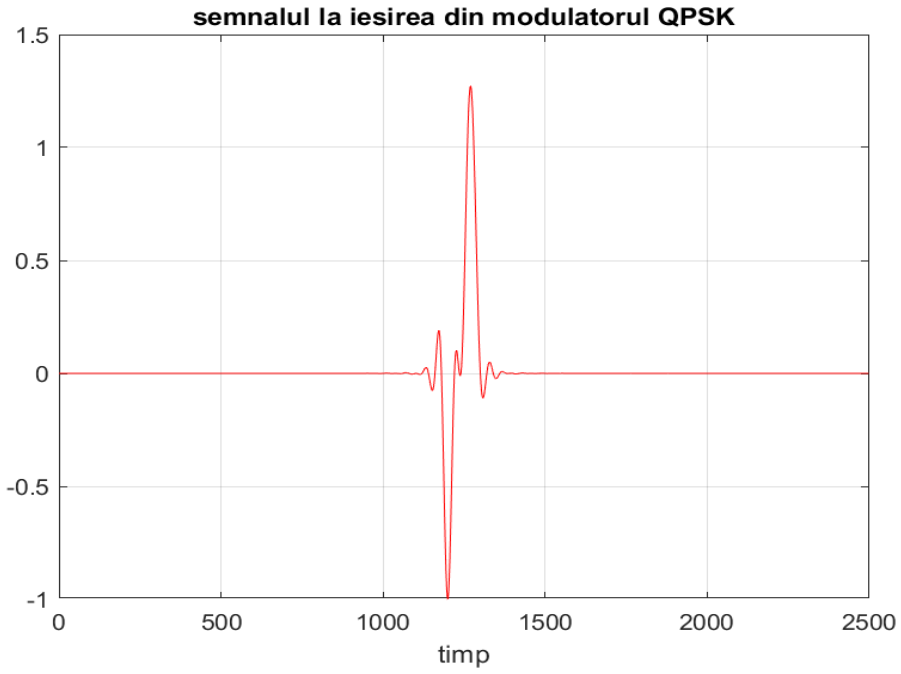

semnalele in faza si in cuadratura
faza
timp
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
10
20
30
40
50
60
70
80
90

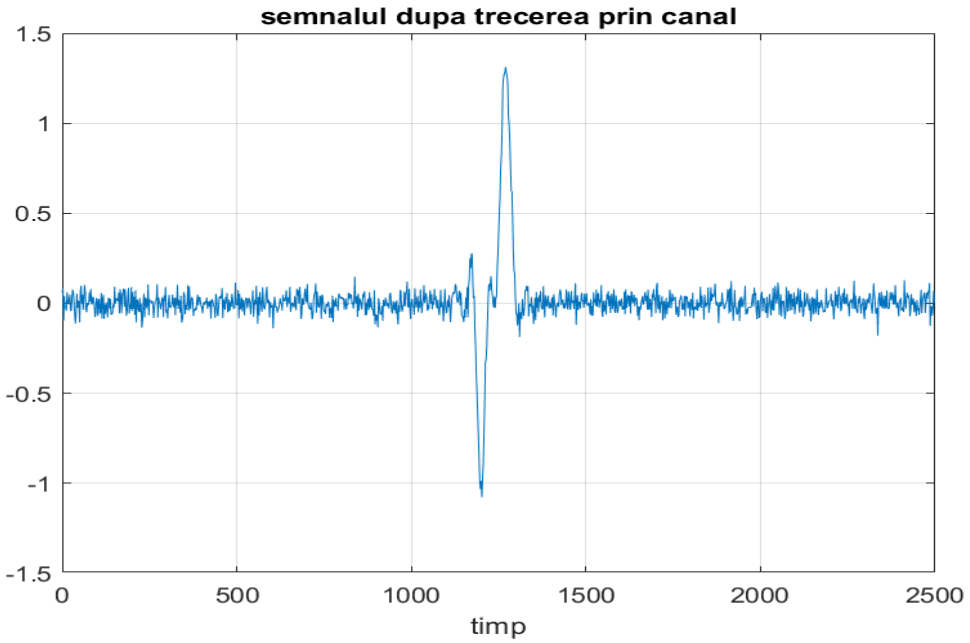

semnalele la iesirea filtrelor de formare
faza
timp
0
0.2
0.4
0.6
0.8
1
0
500
1000
1500
2000
2500

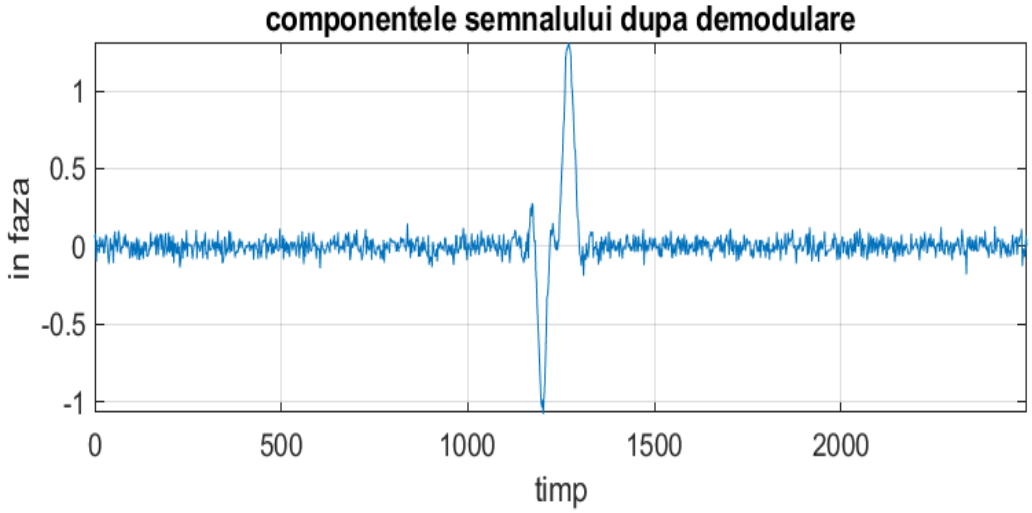

cuadratura
timp
0
0.2
0.4
0.6
0.8
1
0
500
1000
1500
2000
2500

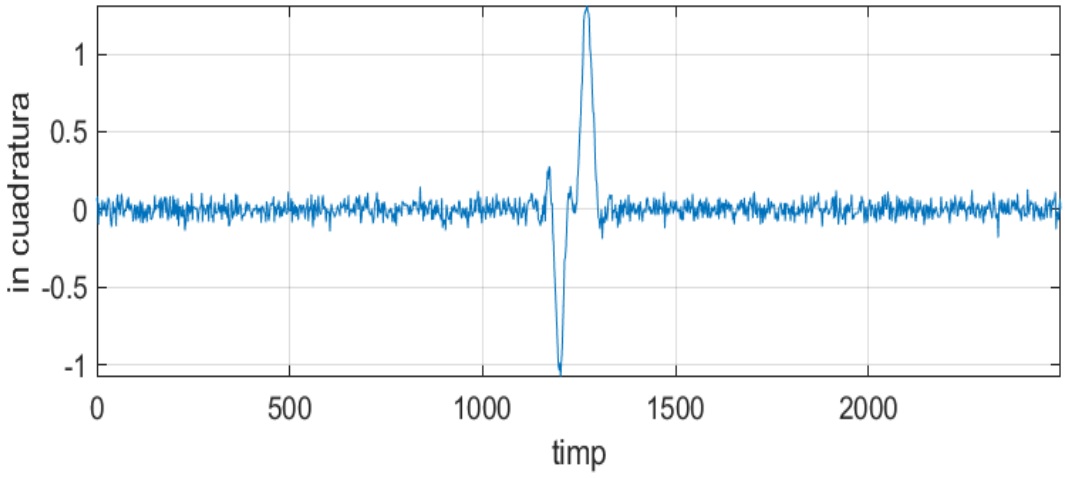

componentele semnalului dupa modulare
in faza
timp
0
0.2
0.4
0.6
0.8
1
0
500
1000
1500
2000

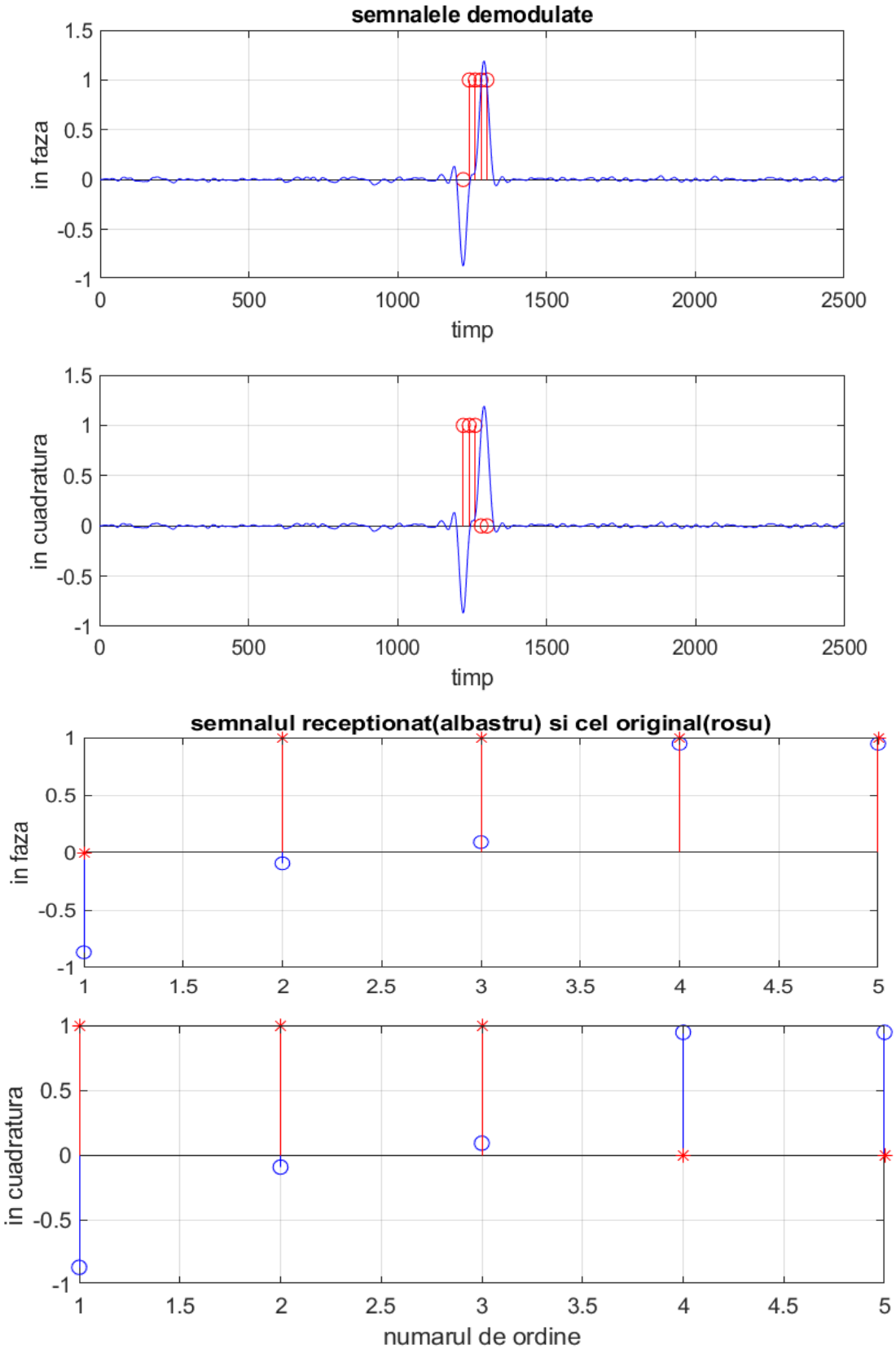

in cuadratura
timp
0
0.2
0.4
0.6
0.8
1
0
500
1000
1500
2000

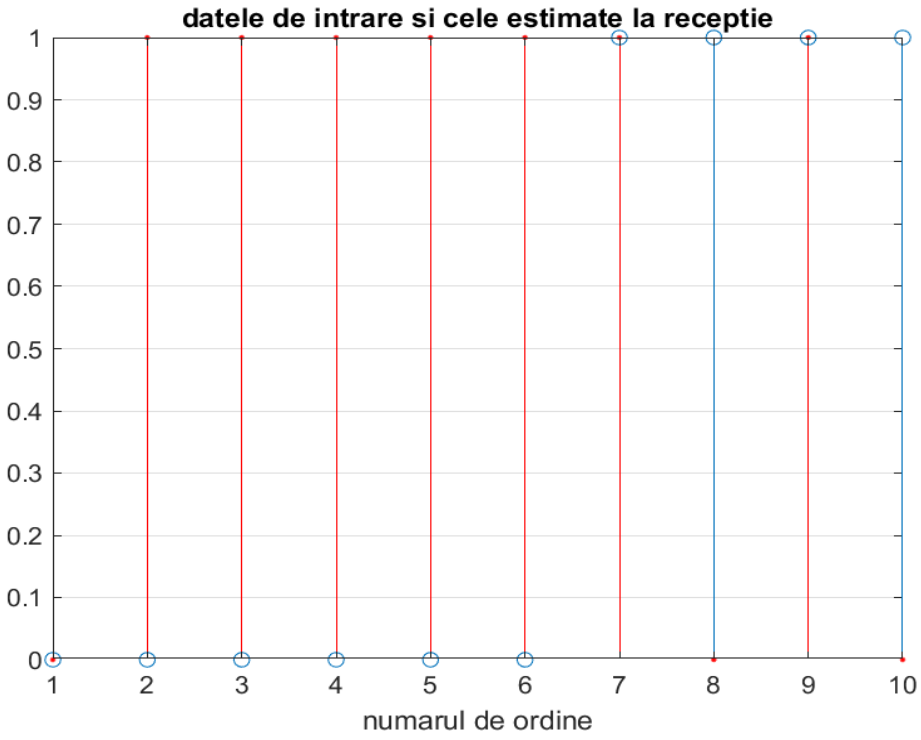

semnalul la iesirea din modulatorul QPSK
timp
-1
-0.5
0
0.5
1
1.5
0
500
1000
1500
2000
2500


semnalul dupa trecerea prin canal
1.5
1
0.5
0
-0.5
-1
-1.5
0
500
1000
1500
2000
2500
timp


componentele semnalului dupa demodulare
in faza
1
0.5
0
-0.5
-1
0
500
1000
1500
2000
timp


in cuadratura
1
0.5
0
-0.5
-1
0
500
1000
1500
2000
timp


semnalele demodulate
in faza
timp
in cuadratura
timp
semnalul receptionat(albastru) si cel original(rosu)
in faza
in cuadratura
numarul de ordine


datele de intrare si cele estimate la receptie
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1
2
3
4
5
6
7
8
9
10
numarul de ordine