Documentatie Tema 1 TD

Membrii echipa: Bratfalean Dragos, Goia Calin Daniel

Cerinta proiect: Realizarea unei simulari a transmisiei QAM rectangulare (o constelatie la alegere) in MatLab. Filtrare, generare zgomot, calcul BER si SER. Se considera purtatoare si tact sincronizate.

Etape realizare proiect:

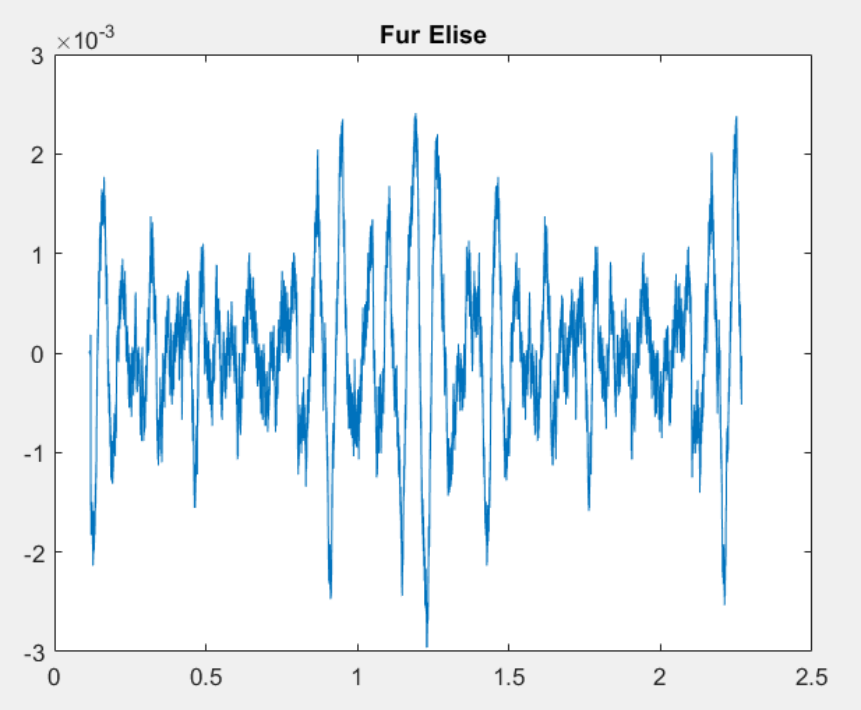
1. Citirea semnalului audio si afisarea lui.

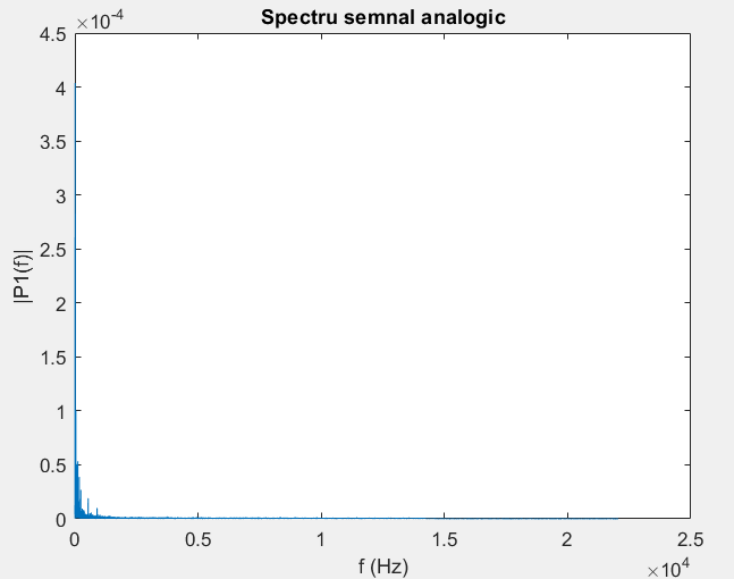
```
%% Read the audio signal

song = "Fur_Elise.mp3";
[y,Fs] = audioread(song);
channel1=y(5000 : 100000, 1);
```

```
%% Plot the audio signal

figure('Name', 'Audio Signal');
t=5000 / Fs : 1 / Fs : 100000 / Fs;
plot(t, channel1);
title("Fur Elise")
```

```
%% Baseband audio signal

[Sc1,f1] = Analiz_spec(channel1, Fs, "Spectru semnal analogic");
```

```
%% Player audio
player = audioplayer(y,Fs);
```





2. Cuantizarea semnalului audio

```
%% Quantizare semnal

[S,L] = bounds(channel1);                    %capetele intervalului
V = max(abs(S), abs(L));                     %maximul dintre capete
semnal = channel1(951 : 960);
out = compand(channel1, 255, V);             %compresare u-law

nb = 8; %numar de bits de cuantizare trebuie ales la fel ca si numarul de biti pentru compandare

delta = 2*V / 2^nb;                  %pas de cuantizare

partition = -V + delta : delta : V - delta;      %partition
codebook=-V + delta/2 : delta : V - delta/2;             %codebook

[indexes,quants,distor] = quantiz(out,partition,codebook);
```

3. Convertire nivele in binar si afisarea bitilor

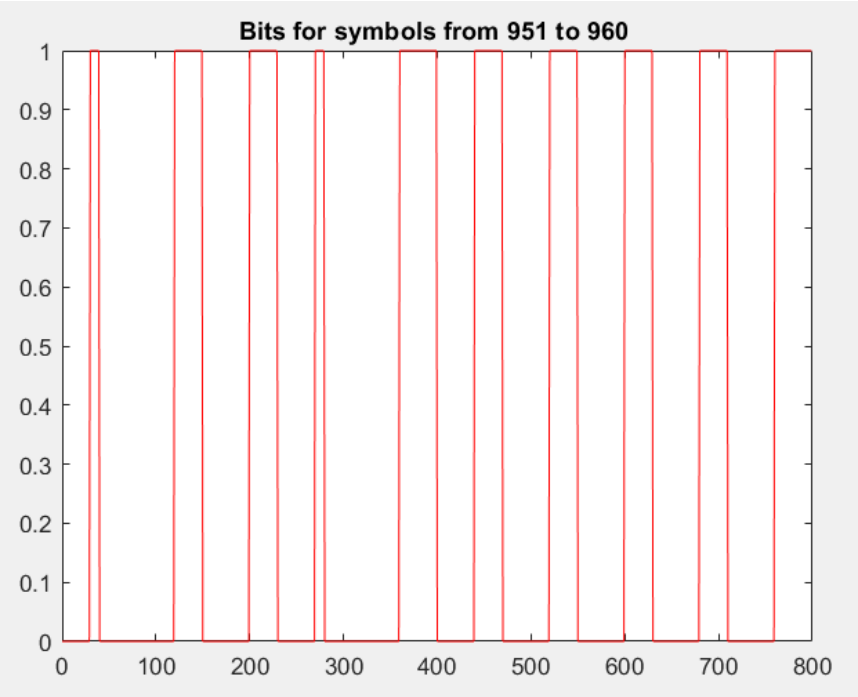
```
%% Convert levels in binary

indexes_binary = int2bit(indexes,8);
```

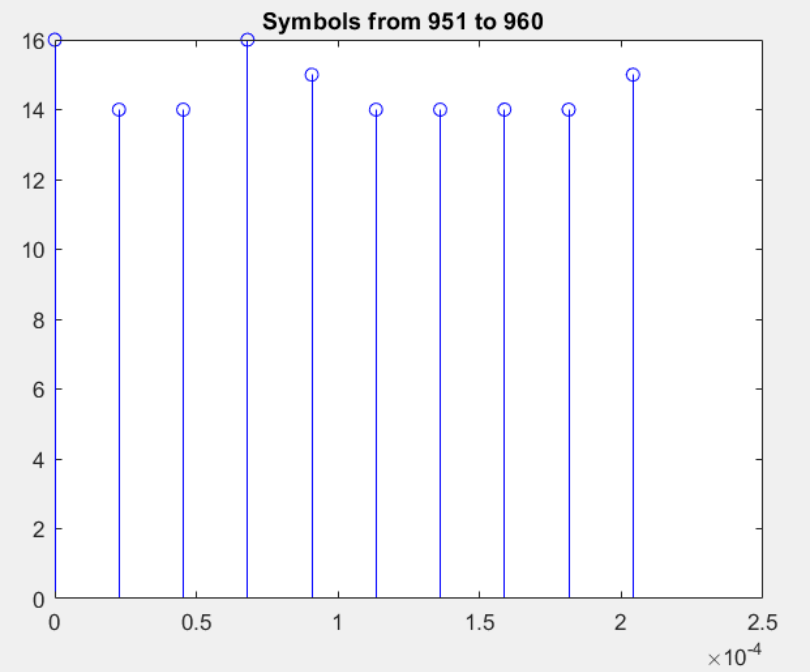
```
%% Send just first 10 symbols

indexes_binary_80 = indexes_binary(7601 : 7680);
quants_10 = quants(951 : 960);
```

```
%% Symbols to modulate

symbols_80= indexes(951:960);
```

```
%% Plot the digital signal
Fbit = Fs * 8;
sps = 10;
rep_sign = repmat(indexes_binary_80, 1, sps);
oversampled_binary_indexes = reshape(transpose(rep_sign), 1, numel(rep_sign));
Fsample = sps * Fbit;
ts=0 : 1 : 800-1;
figure('Name', 'Bits');
plot(ts,oversampled_binary_indexes,'r');
title("Bits for symbols from 951 to 960 ");
```

```
%% Plot the symbols

t=0 : 1/Fs : (10-1)/Fs;
figure('Name', 'Symbols');
stem(t,indexes(951:960),'b');
title("Symbols from 951 to 960  ");
```

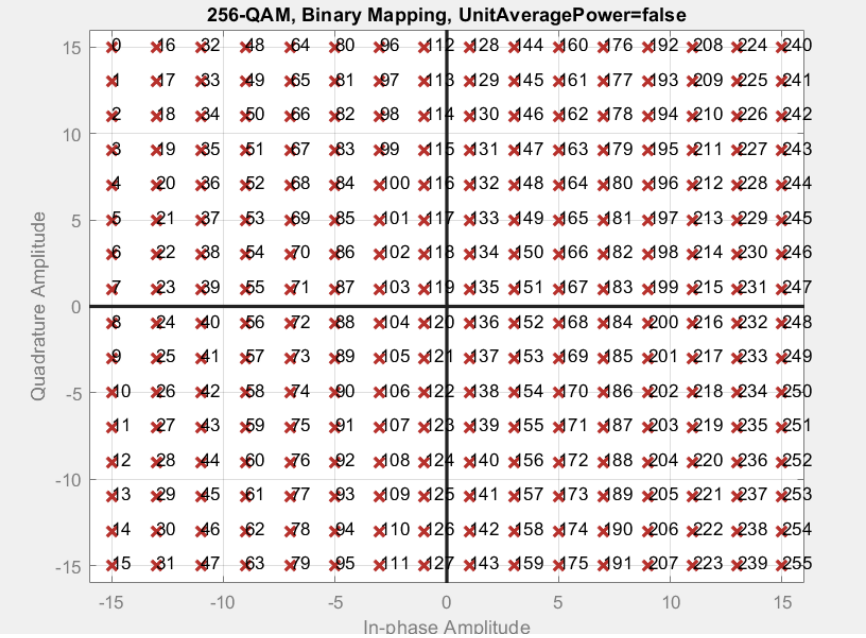




4. Modularea QAM si separea componentelor

```
%% Modulare qam

M = 256;            %transmitem 8 biti

Y_bin = qammod(indexes, M, 'bin', PlotConstellation=true);      %QAM-256
Y_gray = qammod(indexes, M, PlotConstellation=true);            %QAM-256
```

```
%% Separare componente

Ibin = real(Y_bin);
Qbin = imag(Y_bin);

Igray = real(Y_gray);
Qgray = imag(Y_gray);
```



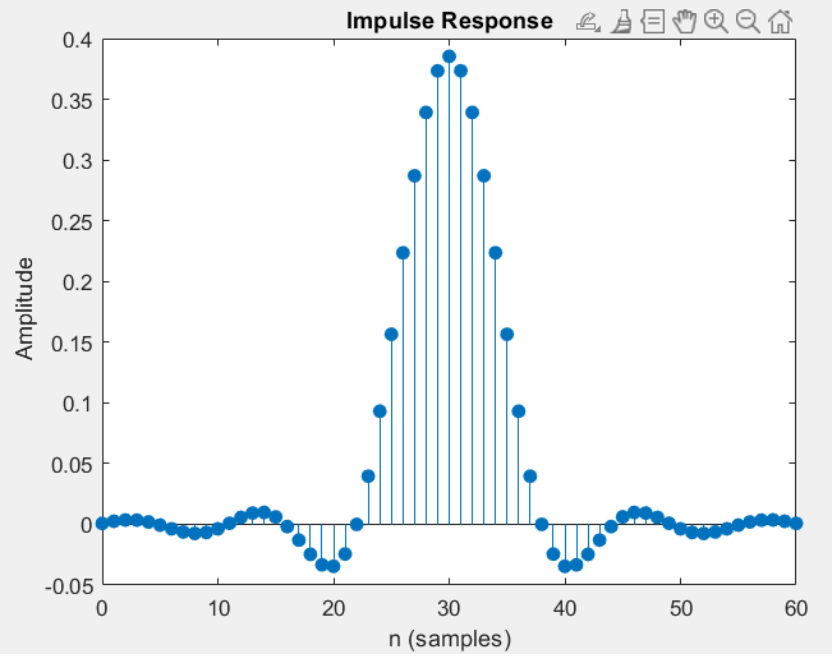
5. Aplicarea filtrului RRC si vizualizarea raspunsului la impuls

```
%% Filtru RRC

Span = 6;
Sps = 10;
rcosfilter_tx = comm.RaisedCosineTransmitFilter(Shape ="Square root", RolloffFactor=0.8, FilterSpa
% rcosfilter_tx.Gain=1/max(rcosfilter_tx.coeffs.Numerator);
```

```
%% Visualizare raspuns la impuls

impz(rcosfilter_tx.coeffs.Numerator);
```

```
%% Bin sequence

extended_Ibin = cat(1, Ibin, zeros(Span/2,1));
extended_Qbin=cat(1, Qbin, zeros(Span/2,1));
```

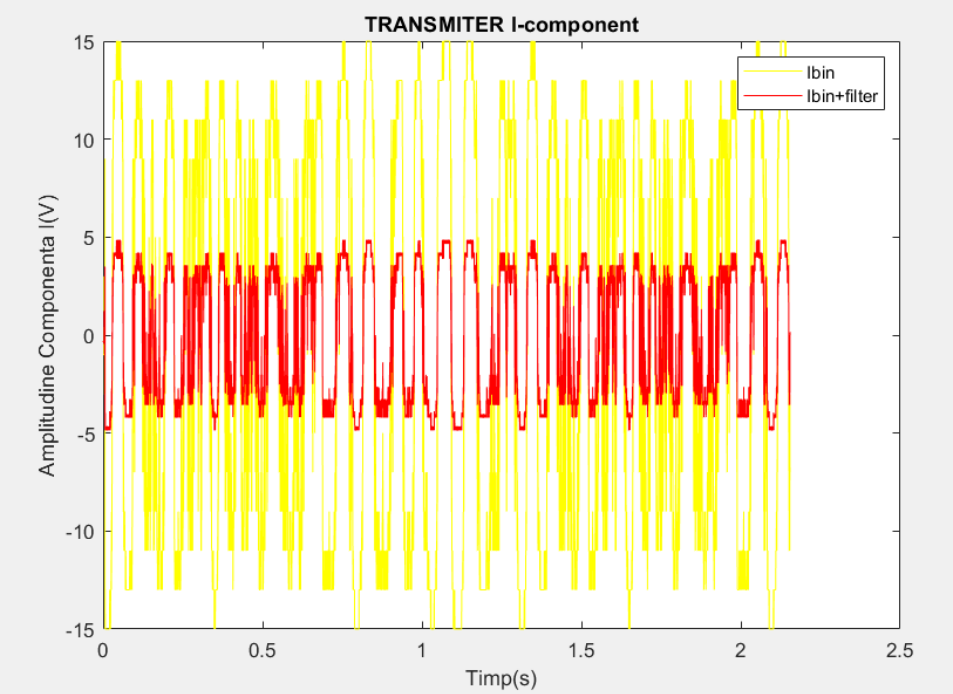
```
%% Gray sequence

extended_Igray = cat(1, Igray, zeros(Span/2,1));
extended_Qgray = cat(1, Qgray, zeros(Span/2,1));
```



6. Componentele I si Q pentru binary si gray

```
%% I and Q component for bin

Ibin_filtered = rcosfilter_tx(extended_Ibin);
Qbin_filtered = rcosfilter_tx(extended_Qbin);

Ibin_filtered = Ibin_filtered((Sps*Span)/2+1:end);
Qbin_filtered = Qbin_filtered((Sps*Span)/2+1:end);

to = (0 : 1 : length(Ibin) - 1) / Fs;
figure;
plot(to, Ibin, "y");
hold on;
tx = (0 : length(Ibin_filtered) - 1) / 10 /Fs;
plot(tx, Ibin_filtered, 'r'), xlabel("Timp(s)"), ylabel("Amplitudine Componenta I(V)")
legend('Ibin', 'Ibin+filter')
title("TRANSMITER I-component");

% hold on;
% plot(Qbin_filtered,'g');
```

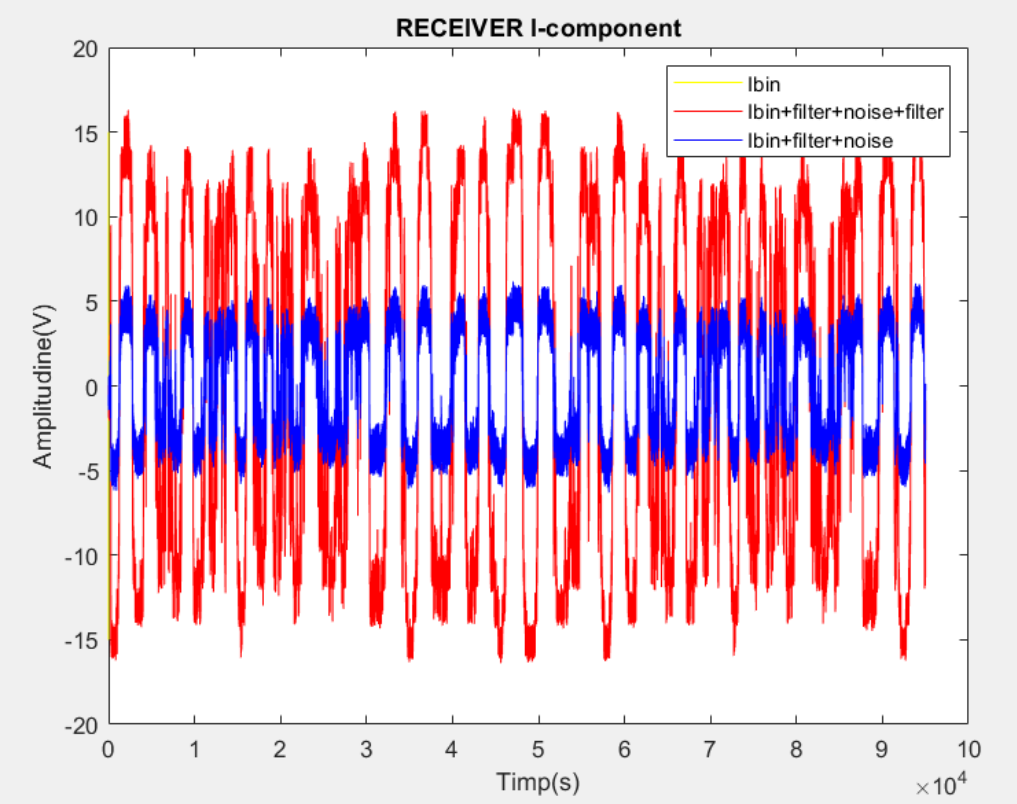
```
%% I and Q component for gray

Igray_filtered = rcosfilter_tx(extended_Igray);
Qgray_filtered = rcosfilter_tx(extended_Qgray);

Igray_filtered = Igray_filtered((Sps*Span)/2+1:end);
Qgray_filtered = Qgray_filtered((Sps*Span)/2+1:end);
```



7. Generare zgomot

```
%% SNR

snr = 20;

Ibin_filtered_noisy = awgn(Ibin_filtered,snr,'measured');
Qbin_filtered_noisy = awgn(Qbin_filtered,snr,'measured');

Igray_filtered_noisy = awgn(Igray_filtered,snr,'measured');
Qgray_filtered_noisy = awgn(Qgray_filtered,snr,'measured');

rcosfilter_rx = comm.RaisedCosineReceiveFilter(...
  'Shape',                  'Square root', ...
  'RolloffFactor',          0.8, ...
  'FilterSpanInSymbols',    Span, ...
  'InputSamplesPerSymbol',  Sps, ...
  'DecimationFactor',       1);
```

```
%% Filter at the receiver.

Ibin_filtered_received = rcosfilter_rx([Ibin_filtered_noisy; zeros((Sps*Span)/2, 1)]);
Qbin_filtered_received = rcosfilter_rx([Qbin_filtered_noisy; zeros((Sps*Span)/2, 1)]);
Igray_filtered_received = rcosfilter_rx([Igray_filtered_noisy; zeros((Sps*Span)/2, 1)]);
Qgray_filtered_received = rcosfilter_rx([Qgray_filtered_noisy; zeros((Sps*Span)/2, 1)]);


Ibin_filtered_received2 = Ibin_filtered_received((Sps*Span)/2+1:end);
Qbin_filtered_received2 = Qbin_filtered_received((Sps*Span)/2+1:end);
Igray_filtered_received2 = Igray_filtered_received((Sps*Span)/2+1:end);
Qgray_filtered_received2 = Qgray_filtered_received((Sps*Span)/2+1:end);


 to = 44100 * (0 : length(Ibin)-1) / 44100 / Fs;
 figure;
 plot(to, Ibin, "y");
 hold on;

 tx = (0 : length(Ibin_filtered_received)-1)/10;
 plot(tx,Ibin_filtered_received,'r');

 ts = (0 : length(Ibin_filtered_received2) - 1) / 10;
 plot(ts, Ibin_filtered_noisy, 'b'), xlabel("Timp(s)"), ylabel("Amplitudine(V)")
 legend('Ibin', 'Ibin+filter+noise+filter', 'Ibin+filter+noise')
 title("RECEIVER I-component")
```



8. Downsampling

```
%% Constelation values

specific_values = -15 : 2 : 15;
```

```
%% Downsample

I_bin_received = downsample(Ibin_filtered_received2 ,10);
Q_bin_received = downsample(Qbin_filtered_received2, 10);
```

```
%% Round

I_bin_received2 = Roundtospecific(I_bin_received, specific_values);
Q_bin_received2 = Roundtospecific(Q_bin_received, specific_values);
```

```
%% Downsample

I_gray_received = downsample(Igray_filtered_received2, 10);
Q_gray_received = downsample(Qgray_filtered_received2, 10);
```

```
%% Round

I_gray_received2 = Roundtospecific(I_gray_received, specific_values);
Q_gray_received2 = Roundtospecific(Q_gray_received, specific_values);
```

```
QAM_bin = I_bin_received2 + 1i*Q_bin_received2;
QAM_gray = I_gray_received2 + 1i*Q_gray_received2;


cd1 = comm.ConstellationDiagram(ShowReferenceConstellation=false);
cd1(QAM_bin);

cd2 = comm.ConstellationDiagram(ShowReferenceConstellation=false);
cd2(QAM_gray);
```

Functia Roundtospecific:

```
function[rounded_values]=Roundtospecific(values,specific)
rounded_values=values;
    for d = 1: length(values)
        v=abs(specific-values(d));
        [~,index]=min(v);
        rounded_values(d)=specific(index);

    end
end
```

9. Demodulare si calcul BER si SER.

```
%% Demodulare

recovered_signal_bin = qamdemod(QAM_bin,M,'bin');     % Binary-encoded data symbols
recovered_signal_G = qamdemod(QAM_gray,M);             % Gray-coded data symbols
```

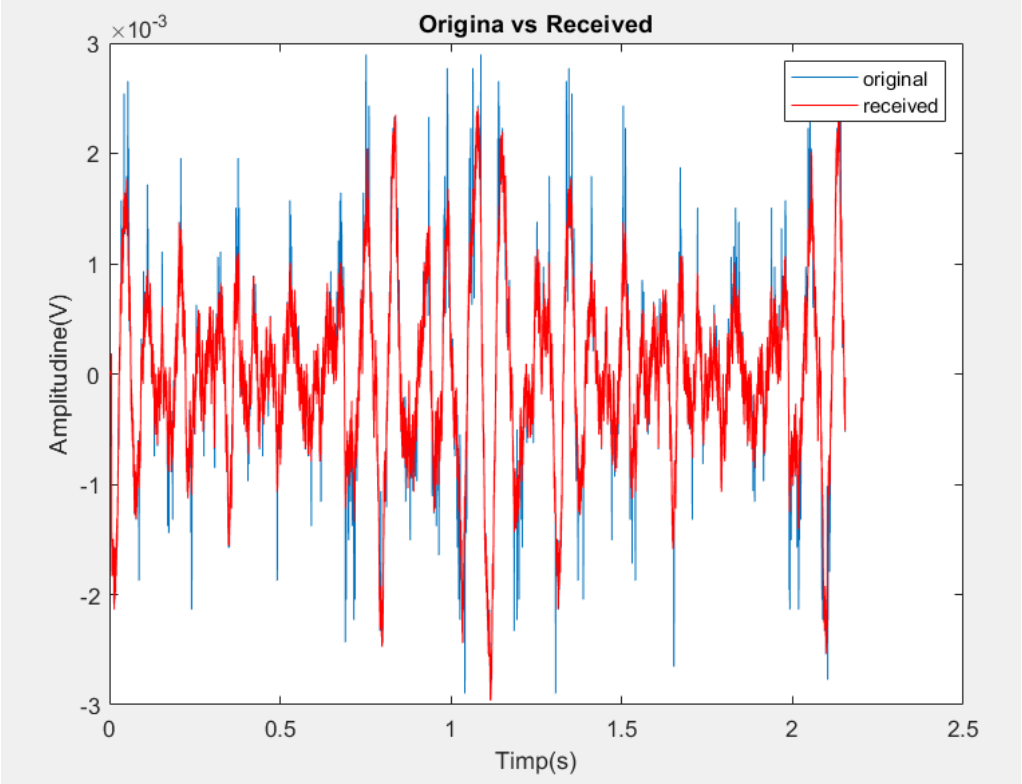
```
%% SER

count_bin_errors=0;
count_G_errors=0;

for i = 1 : numel(recovered_signal_bin)

    if recovered_signal_bin(i) ~= indexes(i)
        count_bin_errors = count_bin_errors+1;
    end


    if recovered_signal_G(i) ~= indexes(i)
        count_G_errors = count_G_errors+1;
    end

end

count_bin_error_rate = count_bin_errors / numel(recovered_signal_bin);
count_G_error_rate = count_G_errors / numel(recovered_signal_bin);
```

```
fprintf('\nThe Symbol error rate is %5.2e, based on %d errors for binary notation.\n', ...
    count_bin_error_rate, count_bin_errors);


fprintf('\nThe Symbol error rate is %5.2e, based on %d errors for Gray notation.\n', ...
    count_G_error_rate,count_G_errors);
```

```
%% BER
dataOut = int2bit(recovered_signal_bin, 8);
dataOutG = int2bit(recovered_signal_G, 8);

[numErrors,ber] = biterr(indexes_binary, dataOut);
fprintf('\nThe binary coding bit error rate is %5.2e, based on %d errors for binary notation.\n', ...
    ber,numErrors)


[numErrors,ber] = biterr(indexes_binary, dataOutG);
fprintf('\nThe binary coding bit error rate is %5.2e, based on %d errors for Gray notation.\n', ...
    ber,numErrors)
```

10. Expandare

```
%% Expandare

quantele = codebook(recovered_signal_bin + 1);
expanded = compand(quantele, 255, V, 'mu/expander');
figure;
plot(to, expanded);
hold on;
plot(to, channel1, 'r'), xlabel("Timp(s)"), ylabel("Amplitudine(V)"),legend('original','received');
title("Original vs Received")
```


Origina vs Received
×10-3
3
2
1
0
-1
-2
-3
Amplitudine(V)
0
0.5
1
1.5
2
2.5
Timp(s)
original
received