ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΨΗΦΙΑΚΕΣ ΕΠΙΚΟΙΝΩΝΙΕΣ ΙΙ

1η ΣΕΙΡΑ ΑΣΚΗΣΕΩΝ

Ειρήνη Δόντη

ΑΜ: 03119839

9ο εξάμηνο

Αθήνα 2023-2024

Παρακάτω, παρατίθενται ο συμπληρωμένος κώδικας του δοσμένου αρχείου MATLAB:

```
%%%%%%     Simulation of a DMT Tx-RX System over an AWGN  line     %%%%%%
%  -------------------------     winter semester 2023      ---------------
-----------------%%

clear all; close all;

%% System parameters
K=255;       % number of subcarriers
Tu=1/4000;   % DMT symbol duration - 4000 frames per second
Fi=4312.5;   % subcarrier spacing
d=0;         % guard interval portion -- zero in this simulation
delta=d*Tu;  % guard interval duration
Ts=Tu+delta; % actual symbol duration
Ks=2*(K+1);  % FFT size
Fs=Ks*Fi;    % Nyquist frequency
T=1/Fs;      % time-sampling period
%
% Gain vector
% It contains the gain for each subcarrier (zero means "don't use")
G=[0 ones(1,K-1) 0];
for i=1:31 G(i)=0; end;    % voice and uplink band -- not used
G(64)=0;                   % sub-carrier 64 will be used as pilot
for i=180:200 G(i)=0; end; % very noisy band -- not used
for i=201:230 G(i)=1.5; end; % higher gain band
for i=231:256 G(i)=0; end; % very noisy band -- not used
used_carriers=sum(G~=zeros(1,256));
% other parameters
M = 16;      % Size of signal constellation per subcarrier - same for all
mb = log2(M); % Number of bits per subcarrier
Ns = 500;      % Number of DMT symbols to process.
% Use a small value of Ns (e.g. 10) for demonstration
% and a large value (e.g. 500) for BER estimation
n_bits=Ns*mb*used_carriers;    % number of bits to generate
Rate=mb*used_carriers/Tu;  %actual transmission rate, in bps
nsamp = 16;  % Oversampling rate, for analog signal simulation
Tn=T/nsamp;  % oversampling period
SNR=5;      % signal-to-noise-ratio, in db
%% Input bit stream
% Create a binary data stream as a vector.
x = randi(2,1,n_bits)-1; %x = randint(1,n_bits); % Random binary data
stream

%% QAM constellation encoding
% A. Define a vector for mapping bits to symbols using
% orthogonal full-grid constellation and Gray coding.
% The vector corresponds to 16-QAM constellation.
mapping = [0 1 3 2 4 5 7 6 12 13 15 14 8 9 11 10].';

% B. Do ordinary binary-to-decimal mapping.
xsym=bi2de(reshape(x,mb,length(x)/mb).','left-msb');

% C. Mapping to QAM constellation
xsym = mapping(xsym+1);
```

```
y=qammod(xsym,M)';
% show a scatterplot of the QAM-encoded data
scatterplot(y);
%-------------------------------------------------------
%  Exercise # 1
% Display the 16-QAM constellation with the associated
% 4-bit codes shown next to ech point of the constellation
%-------------------------------------------------------
% .....

text(real(y) - 0.0,imag(y) + 0.3, dec2base(xsym,2,4),'Color',[0 1 0]);
text(real(y) - 0.5,imag(y) + 0.3, num2str(xsym),'Color',[0 1 0]);

title('16-QAM Symbol Mapping')
axis([-4 4 -4 4])

% .....
% D. Compute weights over thye entire spectrum
power=y*y'/length(y); amp=sqrt(power);
k=1;
y_mat=[];
for n=1:Ns
    z=zeros(2*(K+1),1);
    for i=1:length(G)
        if G(i)~=0
           z(i)=G(i)*y(k);
           z(2*(K+1)-i+1)=z(i)';
           k=k+1;
        end;
    end;
    z(64)=1.5*amp; z(2*(K+1)-63)=z(64);  % pilot
    y_mat=[y_mat z];
end;
% Stem Plot of a DMT symbol (amplitudes of complex values)
figure;
stem(abs(y_mat(:,1)));
title('carrier modulation within a DMT symbol');
xlabel('carier index'); ylabel('amplitude');

% Compute IFFT to get DMT time symbols
s=2*(K+1)*ifft(y_mat, 'symmetric');
figure; pwelch(reshape(s,Ns*2*(K+1),1),[],[],[],Fs);

%%
% Upsampling by using the MATLAB interp function.
% This is needed for the simulation of the analog signal
for n=1:length(s(1,:))
    s_up(:,n)=interp(s(:,n),nsamp);
end;

% plot spectrum of upsampled signal and explain the high freq spectral
images
figure; pwelch(reshape(s_up,Ns*2*(K+1)*nsamp,1),[],[],[],Fs);
Fs1=Fs*nsamp; Ks1=Ks*nsamp; % the sampling frequency is changed accordingly
%
%%  add noise
sr=awgn(s_up,SNR,'measured');
figure; pwelch(reshape(sr,Ns*2*(K+1)*nsamp,1),[],[],[],Fs);
clear s_up;
```

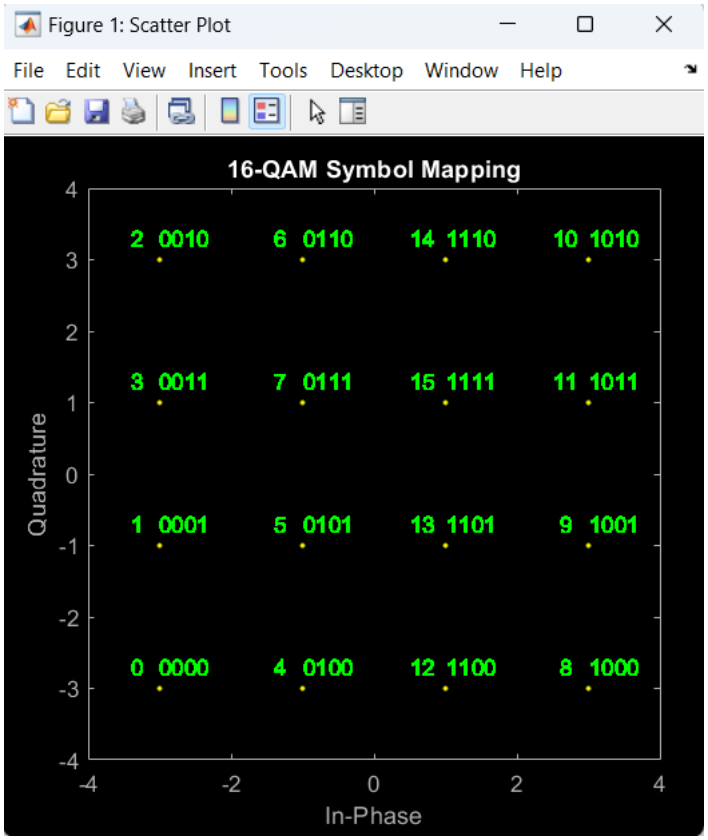
```
%
%%  Filter out the out-of-band noise and downsample the signal
% filter-out the bandpass components
% LP (Parks-McClellan) filtering
order=16*nsamp;
f1=1.1*Fi*(K+1)/Fs1; f2=1.3*f1;
fpts=[0 2*[f1 f2] 1];
mag=[1 1 0 0]; wt=[1 1];
b = firpm(order,fpts,mag,wt);
a=1;
for i=1:length(sr(1,:))
    dummy=[sr(:,i);zeros(order,1)];
    dummy1=filter(b,a,dummy);
    delay=order/2;
    sr_up(:,i)=dummy1(delay+(1:length(sr(:,1))));
end
figure; pwelch(reshape(sr_up,Ns*2*(K+1)*nsamp,1),[],[],[],Fs);
sr_down=downsample(sr_up,nsamp);
% figure; pwelch(reshape(sr_down,Ns*2*(K+1),1),[],[],[],Fs);
% sr_down=downsample(sr,nsamp);  %without filtering
clear sr sr_up;

%% Delineate DMT symbols and perform FFT to get the QAM symbols
yr_mat=1/Ks*fft(sr_down, Ks);
% keep only the payload carriers
yr=[];
for n=1:length(yr_mat(1,:))
    for i=1:length(G)
        if G(i)~=0
           yr=[yr yr_mat(i,n)/G(i)];
        end;
    end;
end;
scatterplot(yr);

%% QAM decoding and error counting
%-----------------------------------------------------------------
%  Exercise # 2
% Complete the code to implement QAM decoding and error counting.
% For each element of the received vector yr (complex value)
% find the nearest point on the 16-QAM constellation
% Compare the results with the transmitted vector y,
% count the errors and compute the BER
%-----------------------------------------------------------------
% ANSWER
% --------
% ...

dataSymbolsOutG = qamdemod(sr_down,M);    % Gray-coded data symbols
yreal=real(yr);  yimag=imag(yr);          % in phase & quadrature
components
l=[-3:2:3];
for n=1:length(yreal)
   [m,j]=min(abs(l-yreal(n)));
   yreal(n)=l(j);
   [m,j]=min(abs(l-yimag(n)));
   yimag(n)=l(j);
end
%scatterplot(yreal+1i*yimag);
```

```
err=not(y==(yreal+1i*yimag));
errors=sum(err);
BER=errors/(length(y))/mb;
fprintf("BER = %s \n", BER);

% ...
```

Παρακάτω, παρατίθενται τα βασικά σχήματα και σχετικά σχόλια για το συμπληρωμένο αρχείο MATLAB:

Αρχικά, στο αρχείο περιλαμβάνεται η αρχικοποίηση των παραμέτρων συστήματος και δημιουργείται ένα διάνυσμα δυαδικών δεδομένων, το οποίο θα χρησιμοποιηθεί για την απεικόνιση 16QAM. Χρησιμοποιούμε το διάνυσμα mapping για την κωδικοποίηση Gray. Χωρίζουμε τα σημεία σε τετράδες και τα αναδιατάσσουμε ώστε, με την εντολή qammod(), να δημιουργήσουμε τα μιγαδικά σύμβολα QAM.

Απεικονίζουμε τη 16QAM αναπαράσταση με τους σχετιζόμενους 4-bit αριθμούς, (κάνοντας χρήση της εντολής scatterplot()) δίπλα από κάθε σημείο (σε δεκαδική και δυαδική μορφή), όπως φαίνεται παρακάτω:



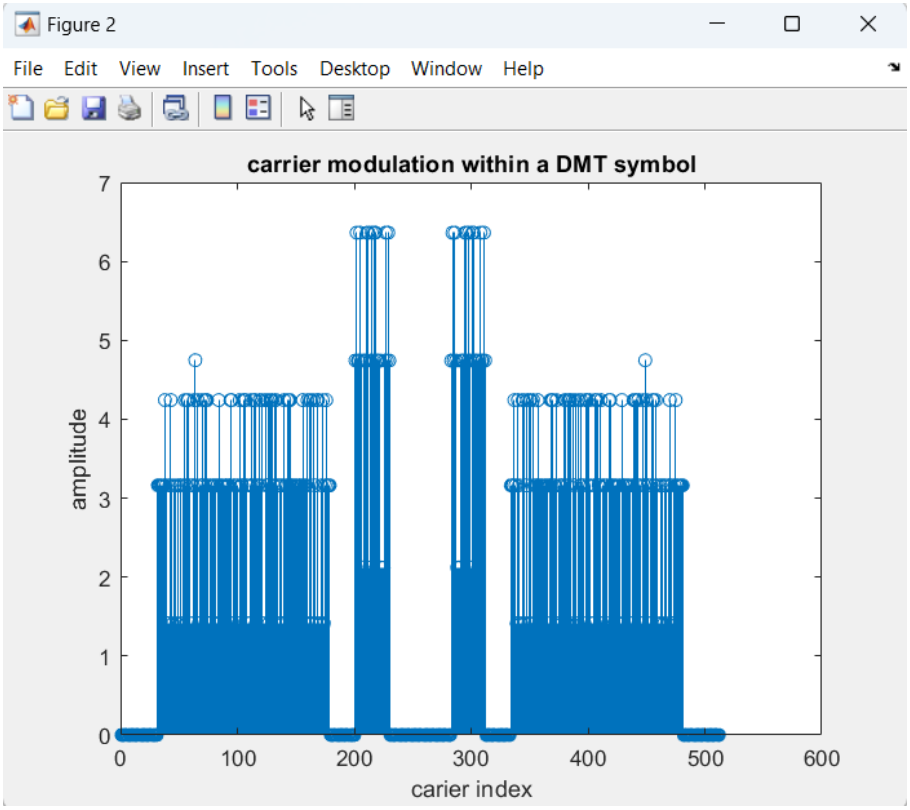
Παρατηρούμε ότι τα γειτονικά bit για κάθε σημείο της αναπαράστασης, διαφέρουν μεταξύ τους κατά ένα bit.

Ο κώδικας για την 16QAM αναπαράσταση των σημείων, παρουσιάζεται παρακάτω:

```
%  Exercise # 1
% Display the 16-QAM constellation with the associated
% 4-bit codes shown next to ech point of the constellation
%-------------------------------------------------------
% .....

text(real(y) - 0.0,imag(y) + 0.3, dec2base(xsym,2,4),'Color',[0 1 0]);
text(real(y) - 0.5,imag(y) + 0.3, num2str(xsym),'Color',[0 1 0]);

title('16-QAM Symbol Mapping')
axis([-4 4 -4 4])

% .....
```

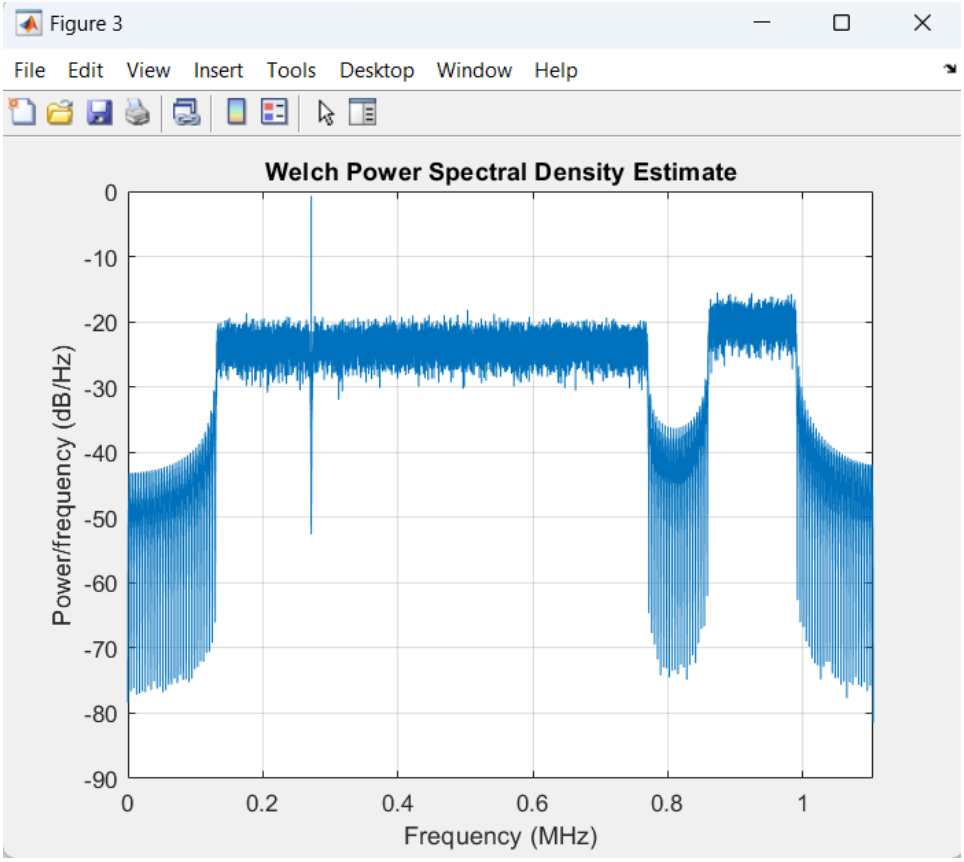
Στον παραπάνω κώδικα, χρησιμοποιήθηκε η εντολή text() η οποία καταγράφει πάνω στη γραφική με πράσινο χρώμα, κοντά στα μιγαδικά σημεία, την δυαδική τους αντιστοίχιση (με τη βοήθεια της εντολής dec2base()) καθώς και τη αντίστοιχη δεκαδική αντιστοίχιση (με τη βοήθεια της εντολής num2str()).

Έπειτα, υπολογίζονται τα βάρη σε όλο το φάσμα ώστε να απεικονιστεί το DMT σύμβολο, όπως φαίνεται στην παρακάτω απεικόνιση:

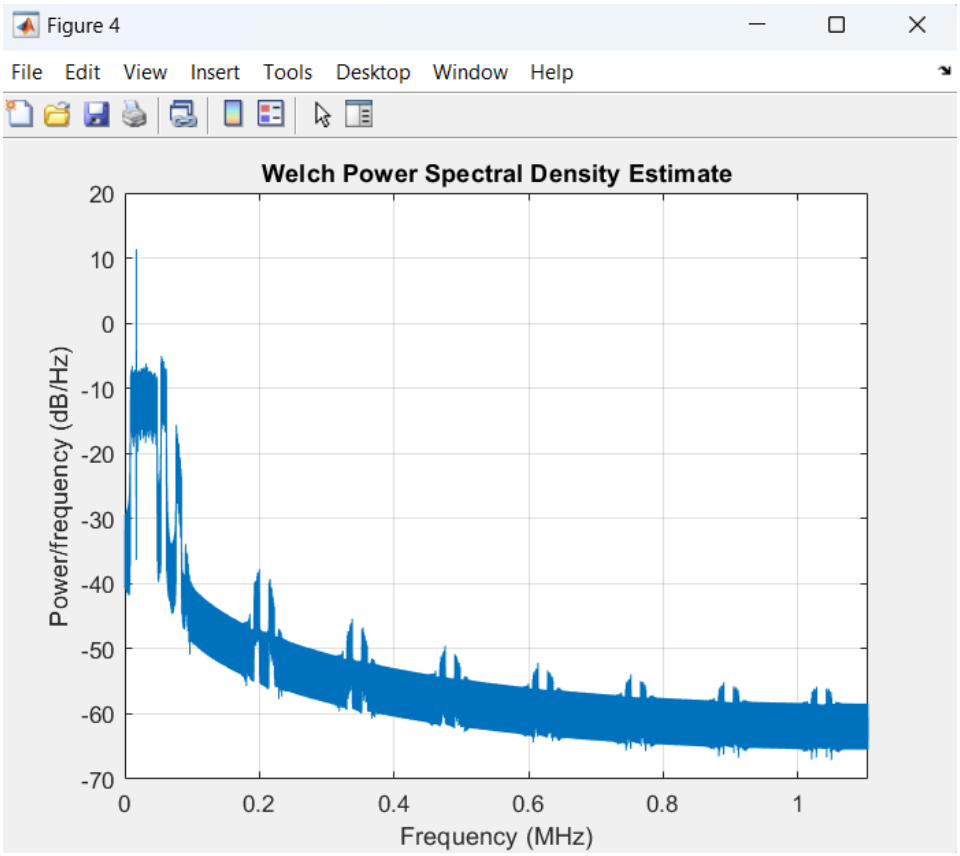


Παρατηρούμε ότι το παραπάνω διάγραμμα είναι συμμετρικό ως προς την τιμή K = 255 (αριθμός των υποδιαύλων), το οποίο είναι λογικό, καθώς το μέγεθος του διαύλου ορίστηκε ως 2(K+1).

Επιπλέον, υπολογίζοντας την IFFT των βαρών, βρίσκουμε τα DMT σύμβολα στο πεδίο του χρόνου και με τη βοήθεια της εντολής pwelch() αναπαρίσταται η εκτίμηση φασματικής πυκνότητας κατά Welch, όπως φαίνεται στην παρακάτω απεικόνιση:

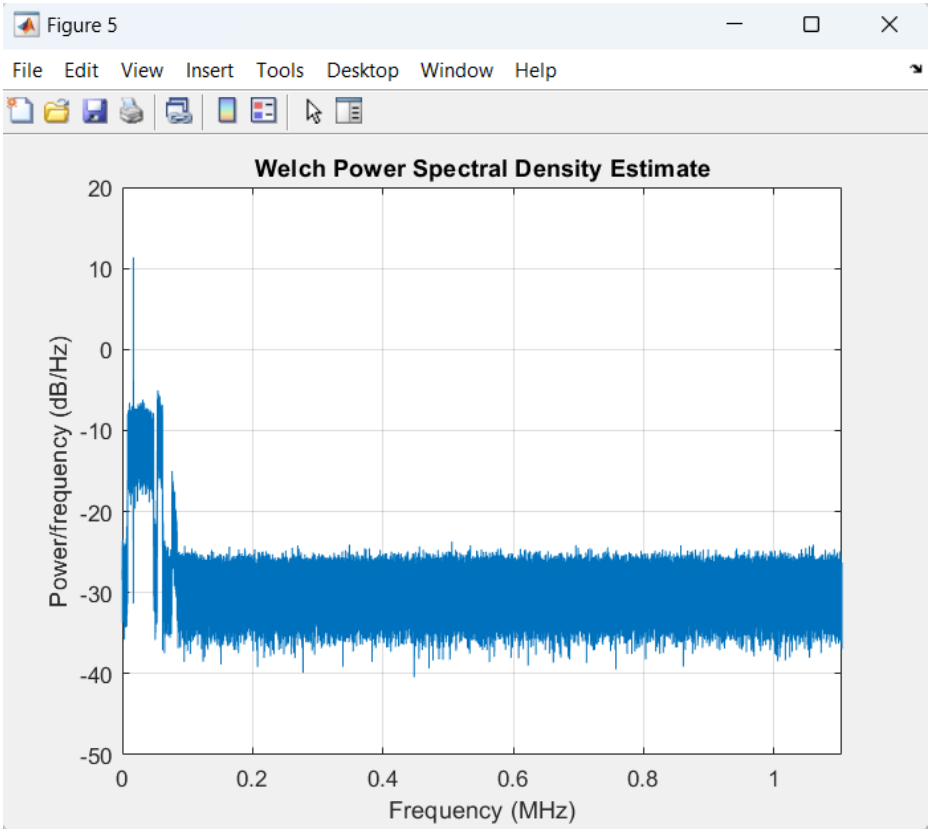


Επιπλέον, κάνοντας upsampling, βρίσκουμε τα DMT σύμβολα στο πεδίο του χρόνου και με τη βοήθεια της εντολής pwelch() αναπαρίσταται η εκτίμηση φασματικής πυκνότητας κατά Welch, όπως φαίνεται στην παρακάτω απεικόνιση:



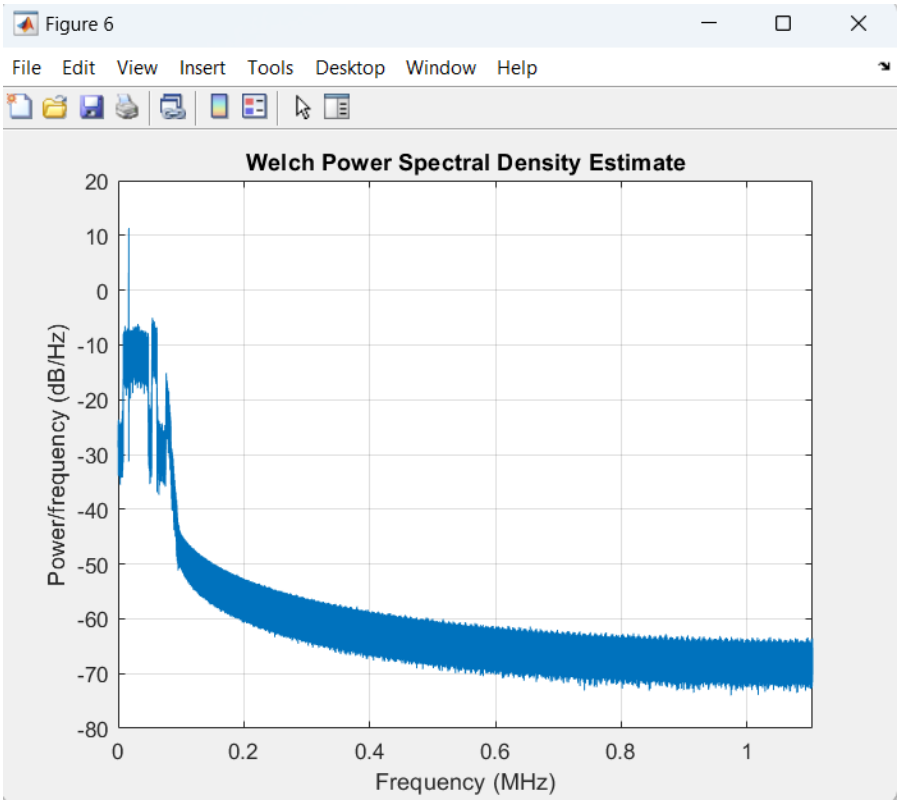
Παρατηρούμε ότι, από κάποια τιμή συχνότητας και μετά, η εκτίμηση φασματικής πυκνότητας ισχύος, από αρχικά σταθερές τιμές, μειώνεται με εκθετικό ρυθμό μέχρι να φτάσει σε σταθερά χαμηλότερη τιμή.

Επιπλέον, προσθέτοντας θόρυβο στο προηγούμενο σήμα, βρίσκουμε τα DMT σύμβολα στο πεδίο του χρόνου και με τη βοήθεια της εντολής pwelch() υπολογίζουμε την εκτίμηση φασματικής πυκνότητας κατά Welch, όπως φαίνεται στην παρακάτω απεικόνιση:



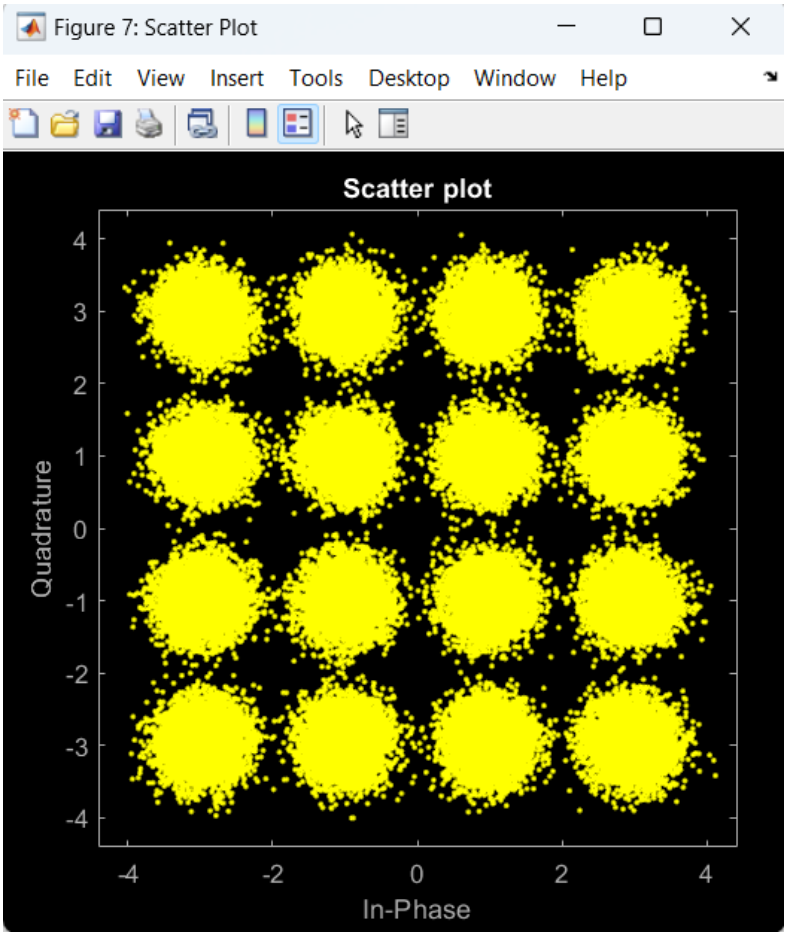
Παρατηρούμε ότι, από κάποια τιμή συχνότητας και μετά, η εκτίμηση φασματικής πυκνότητας ισχύος, από αρχικά σταθερές τιμές, μειώνεται απευθείας σε σταθερά χαμηλότερη τιμή.

Επιπλέον, φιλτράροντας τα στοιχεία διέλευσης ζώνης, βρίσκουμε τα DMT σύμβολα και με τη βοήθεια της εντολής pwelch() υπολογίζουμε την εκτίμηση φασματικής πυκνότητας κατά Welch, όπως φαίνεται στην παρακάτω απεικόνιση:



Παρατηρούμε ότι η γραφική παράσταση δεν εμφανίζει πολλές κατακόρυφες γραμμές σε σχέση με τη γραφική στην οποία πραγματοποιήθηκε upsampling. Το παραπάνω είναι λογικό, καθώς έγινε φιλτράρισμα του θορύβου εκτός ζώνης.

Οριοθετούμε τα DMT σύμβολα και εκτελούμε FFT ώστε να απεικονίσουμε τη 16QAM αναπαράσταση (κάνοντας χρήση της εντολής scatterplot()) όπως φαίνεται παρακάτω:



Παρατηρούμε ότι τα σημεία διασκορπίζονται γύρω από τα μιγαδικά σημεία y και συνεπώς θα υπάρχουν υπολογιστικά σφάλματα.

Επίσης, εφαρμόζουμε αποκωδικοποίηση QAM με την εντολή qamdemod() και έπειτα για το πραγματικό και φανταστικό μέρος του διανύσματος yr βρίσκουμε την κοντινότερη απόσταση από την 16-QAM απεικόνιση. Έπειτα, συγκρίνουμε το αποτέλεσμα με το λαμβανόμενο διάνυσμα y. Αν το διάνυσμα y δεν είναι ίδιο με κοντινότερο διάνυσμα της απεικόνισης, τότε αθροίζεται στο διάνυσμα λαθών. Η τιμή BER υπολογίζεται ως το πηλίκο των παραπάνω υπολογιζόμενων λαθών με το πηλίκο του μήκους διανύσματος y με τον αριθμό bits των υποδιαύλων.

Οπότε, βάσει των παραπάνω, το υπολογιζόμενο BER υπολογίζεται και τυπώνεται με την εκτέλεση του προγράμματος ως:

```
>> Lab1_DontiEirini
BER = 1.525424e-04
```

Ο κώδικας για την 16QAM αποκωδικοποίηση και τον υπολογισμό BER, παρουσιάζεται παρακάτω:

```
%% QAM decoding and error counting
%---------------------------------------------------------------
%  Exercise # 2
% Complete the code to implement QAM decoding and error counting.
% For each element of the received vector yr (complex value)
% find the nearest point on the 16-QAM constellation
% Compare the results with the transmitted vector y,
% count the errors and compute the BER
%---------------------------------------------------------------
% ANSWER
% --------
% ...

dataSymbolsOutG = qamdemod(sr_down,M);    % Gray-coded data symbols
yreal=real(yr);  yimag=imag(yr);          % in phase & quadrature
components
l=[-3:2:3];
for n=1:length(yreal)
   [m,j]=min(abs(l-yreal(n)));
   yreal(n)=l(j);
   [m,j]=min(abs(l-yimag(n)));
   yimag(n)=l(j);
end
%scatterplot(yreal+1i*yimag);
err=not(y==(yreal+1i*yimag));
errors=sum(err);
BER=errors/(length(y))/mb;
fprintf("BER = %s \n", BER);

% ...
```