ΗΜΜΥ ΕΜΠ

6° Εξάμηνο

Ψηφιακές Επικοινωνίες Ι

Lab 2

Μέρος 1

Ερώτηση 1: X=Z. Η `ifftshift` είναι αντίστροφη της `fftshift` οπότε η μία αναιρεί την άλλη, γι'αυτό είναι X=Z.

Ερώτηση 2: X=Z και X=Y. Εδώ N=4 (άρτιο) οπότε η `fftshift` και η `ifftshift` δίνουν το ίδιο αποτέλεσμα, επομένως ισχύει και X=Y. Στην πρώτη περίπτωση όπου N=5 (περιττό) αυτό δεν ισχύει.

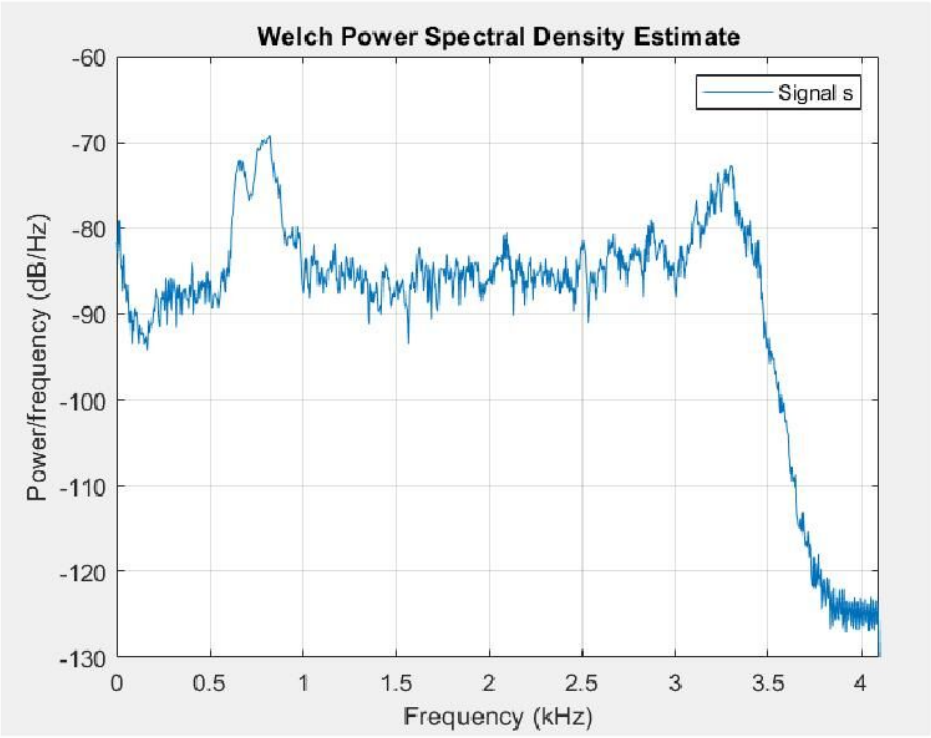
Ερώτηση 3: Γράφουμε κατευθείαν το x όπως προκύπτει από το `ifftshift(xb)`

```
clear all; close all; clc;
x=[5 4 3 2 1 1 2 3 4];
X=fft(x);
Xb=fftshift(X);
subplot (2,1,2); plot('-4:4],Xb);ylabel('Xb');
```

Μέρος 2

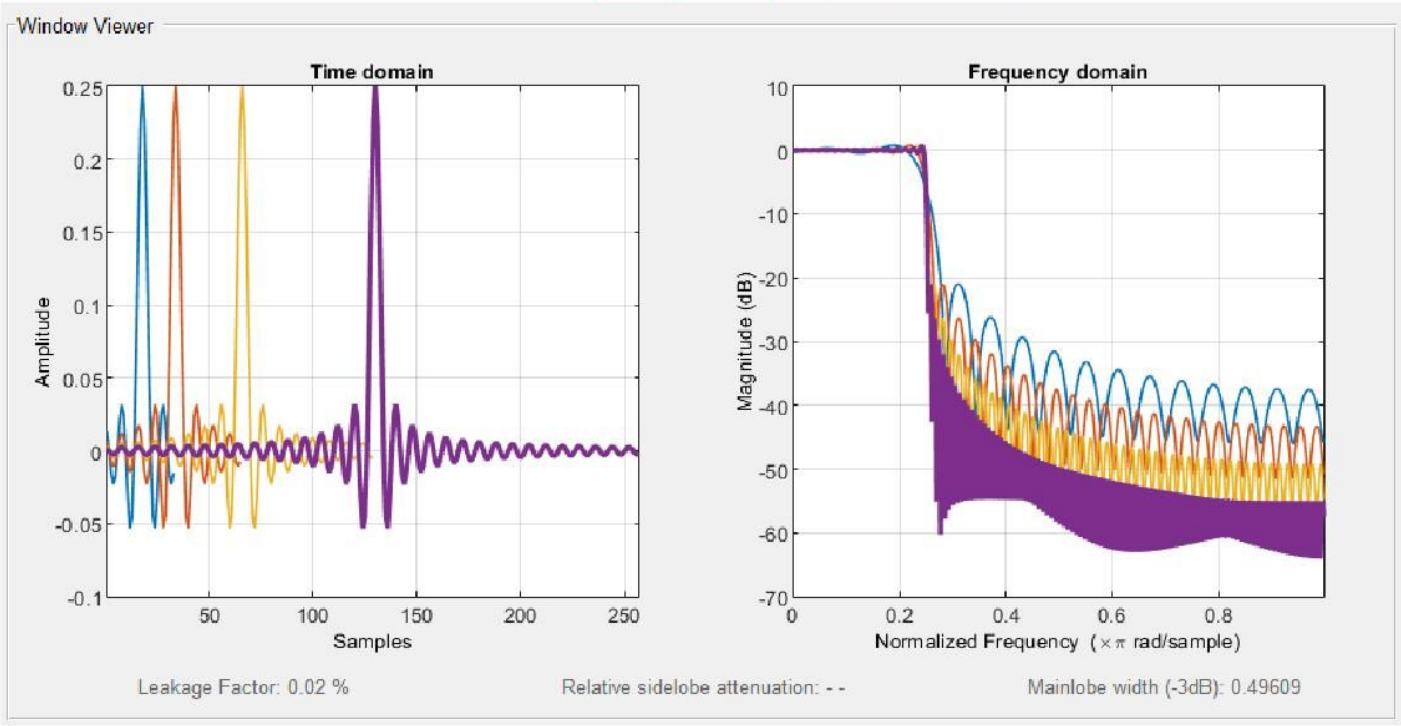
Κώδικας και γραφικές παραστάσεις:

```
clear all; close all;
% File sima.mat has the signal s
% and sampling frequency Fs.
% The spectrum reaches frequencies up to 4kHz
% but everything above 1kHz is noise and must be filtered out.
load sima;

% % Uncomment the following lines to get sine waves
% Fs=8192; Ts=1/Fs; T=1;
% t=0:Ts:T-Ts;
% A=1;
%
s=A*sin(2*pi*800*t)+A*sin(2*pi*1000*t)+A*sin(2*pi*2000*t)+A*sin(2*pi*3000*
t);


figure; pwelch(s,[],[],[],Fs); legend('Signal s');
```

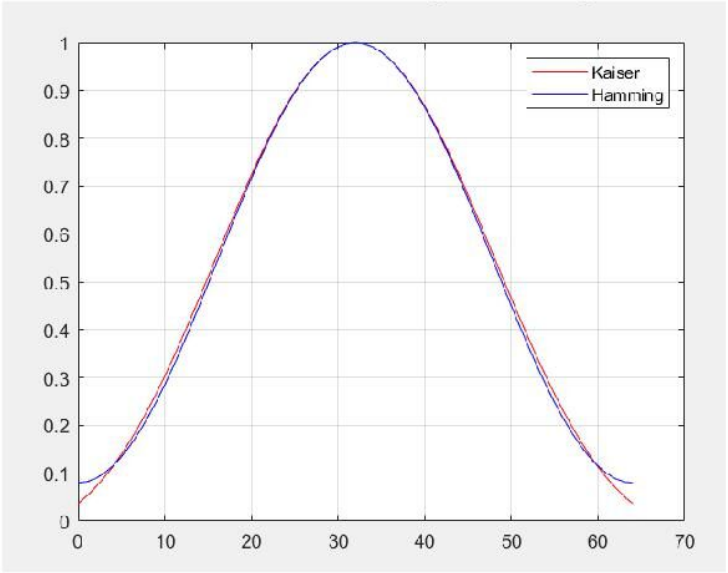



```
% Ideal low pass function H with cutoff frequency Fs/8
H=[ones(1,Fs/8) zeros(1,Fs-Fs/4) ones(1,Fs/8)];
% Impulse response with inverse Fourier transform
% Alternatively, the analytic function Sa(x) can be used
h=ifft(H,'symmetric');
```

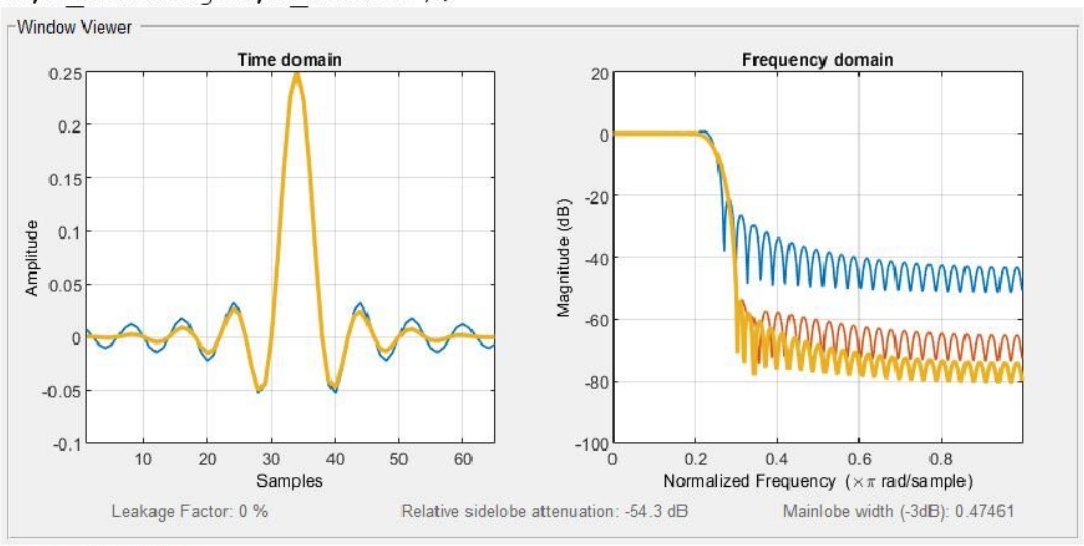
```
middle=length(h)/2;
h=[h(middle:end) h(1:middle-1)];
h32=h(middle+1-16:middle+17);
h64=h(middle+1-32:middle+33);
h128=h(middle+1-64:middle+65);
h256=h(middle+1-128:middle+129);
% figure; stem([0:length(h64)-1],h64); grid;
% figure; freqz(h64,1); % frequency response of h64
wvtool(h32,h64,h128,h256);% frequency responses of cut h
```



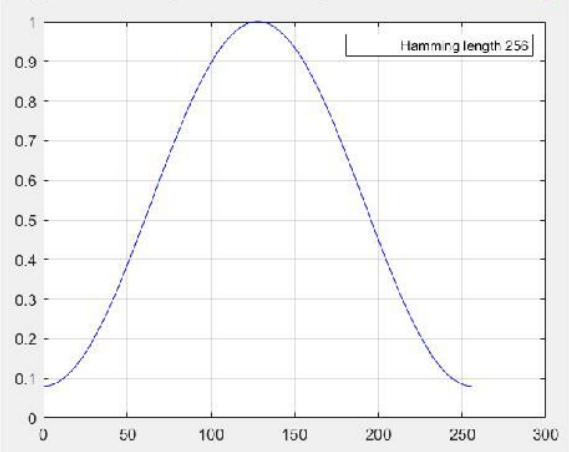

```
% side peaks are high!
% Multiply the cut impulse response with windows
% We use h64 with hamming and kaiser windows
wh64=hamming(length(h64));
wk=kaiser(length(h64),5);
figure; plot(0:64,wk,'r',0:64,wh64,'b'); grid; legend('Kaiser','Hamming');
```



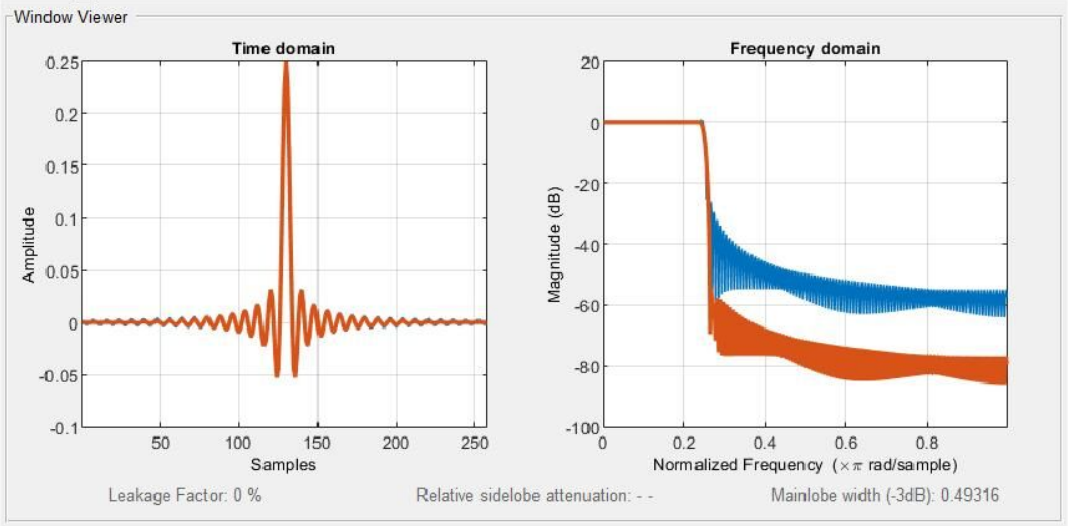
```
h_hamming64=h64.*wh64';
h_kaiser=h64.*wk';
wvtool(h64,h_hamming64,h_kaiser);
```



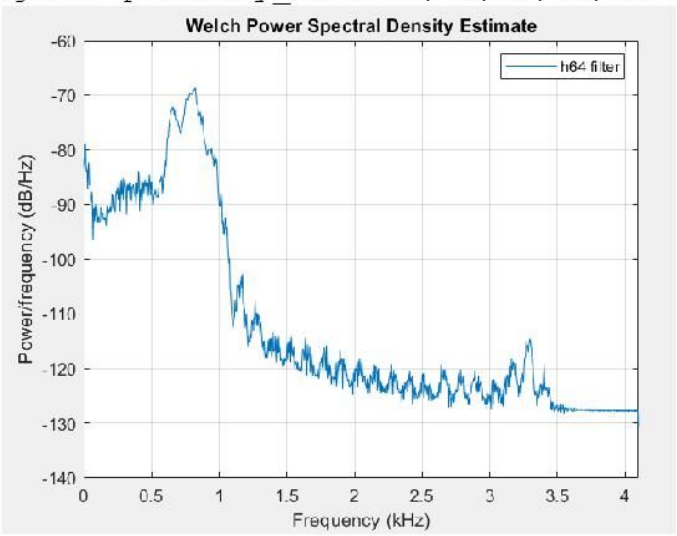

```
% Hamming with h256
wh256=hamming(length(h256));
figure; plot(0:256,wh256,'b'); grid; legend('Hamming length 256');
```

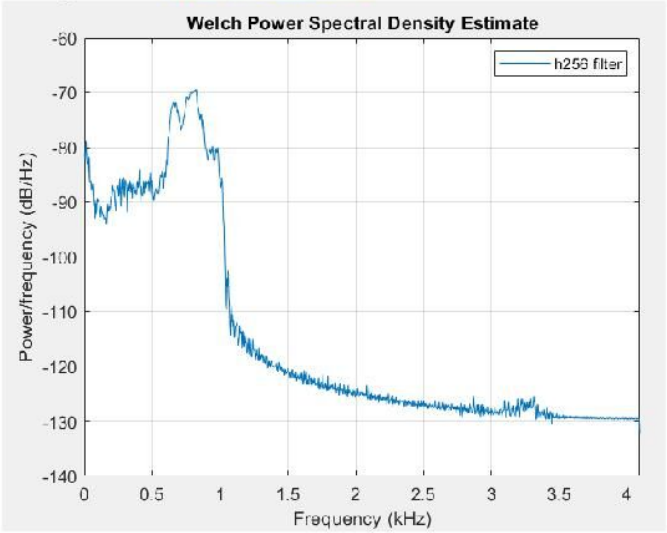



```
h_hamming256=h256.*wh256';
wvtool(h256,h_hamming256);
```

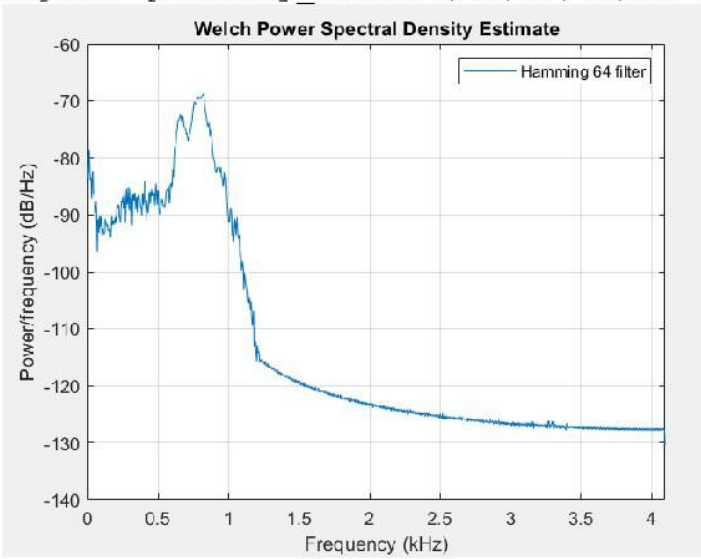


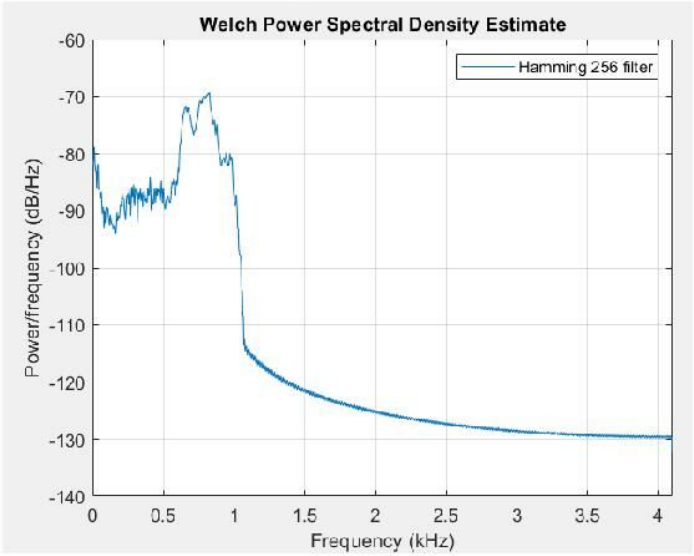
```
% figure; stem([0:length(h64)-1],h_hamming); grid;
% figure; freqz(h_hamming,1);

% Filter the signal with each filter
y_rect64=conv(s,h64);
figure; pwelch(y_rect64,[],[],[],Fs); legend('h64 filter');
y_rect256=conv(s,h256);
figure; pwelch(y_rect256,[],[],[],Fs); legend('h256 filter');
```



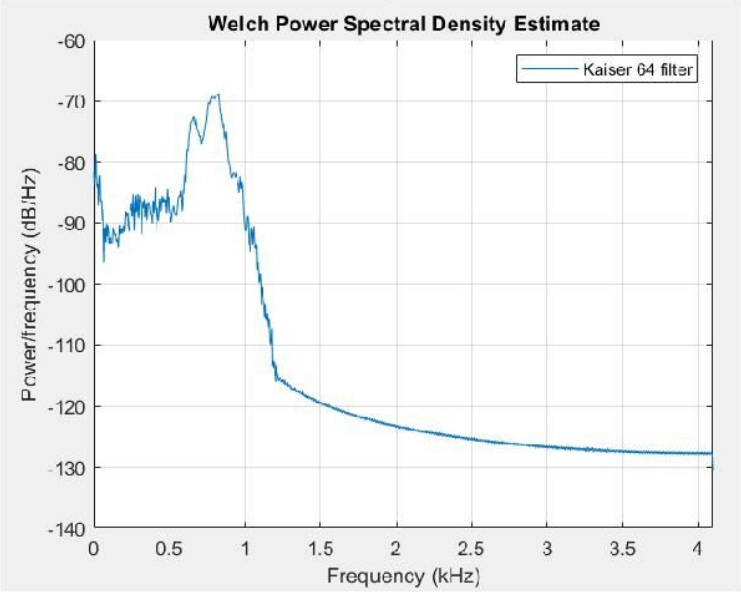



```
y_hamm64=conv(s,h_hamming64);
figure; pwelch(y_hamm64,[],[],[],Fs); legend('Hamming 64 filter');
y_hamm256=conv(s,h_hamming256);
figure; pwelch(y_hamm256,[],[],[],Fs); legend('Hamming 256 filter ');
```

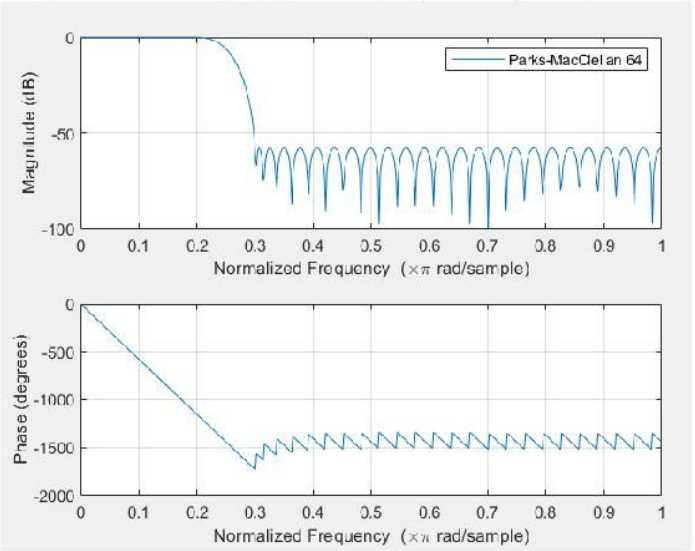


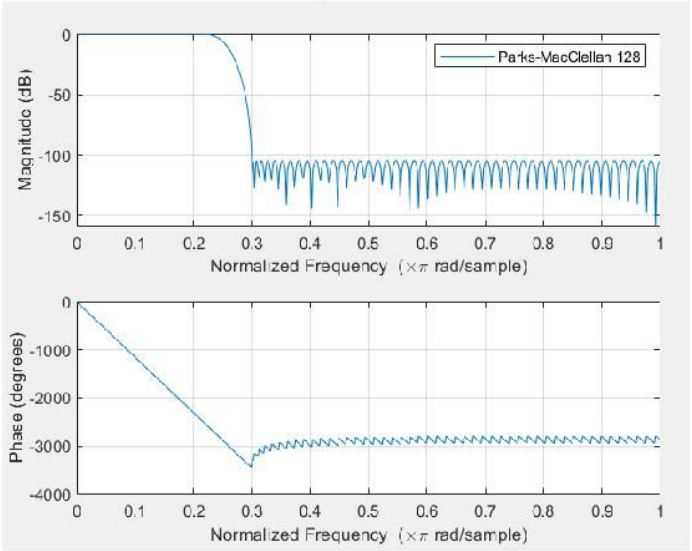


```
y_kais=conv(s,h_kaiser);
figure; pwelch(y_kais,[],[],[],Fs); legend('Kaiser 64 filter');
```

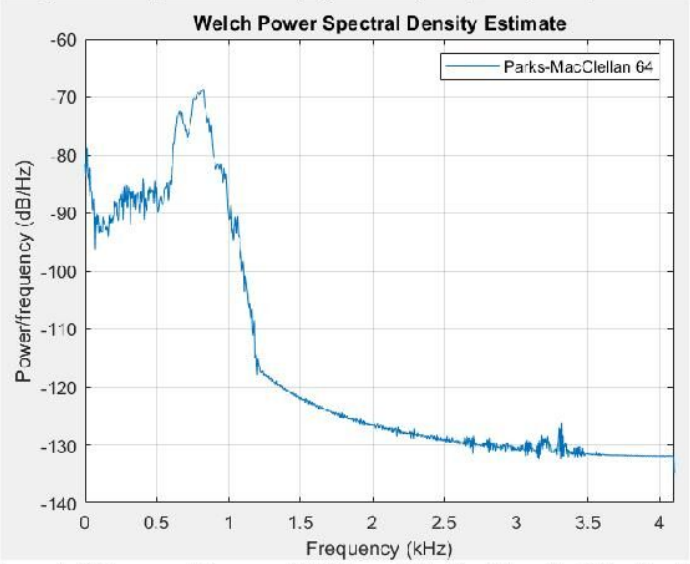


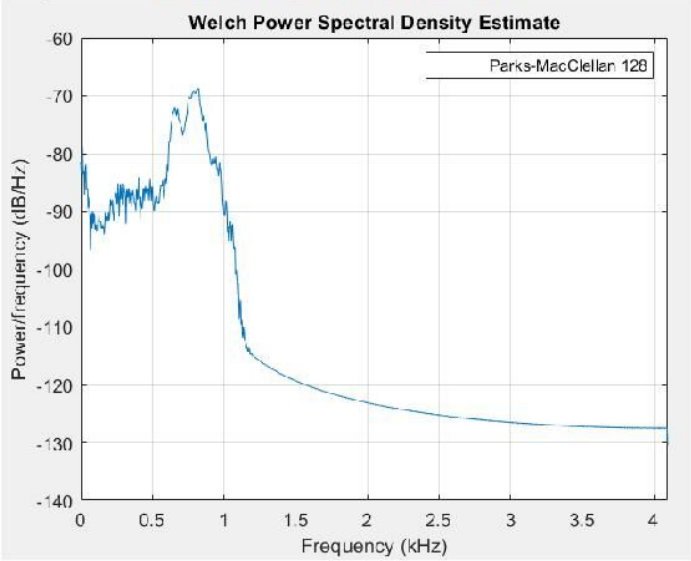

```
% Low pass Parks-MacClellan
hpm64=firpm(64, [0 0.10 0.15 0.5]*2, [1 1 0 0]);
figure; freqz(hpm64,1); legend('Parks-MacClellan 64');
hpm128=firpm(128, [0 0.10 0.15 0.5]*2, [1 1 0 0]);
figure; freqz(hpm128,1); legend('Parks-MacClellan 128');
```



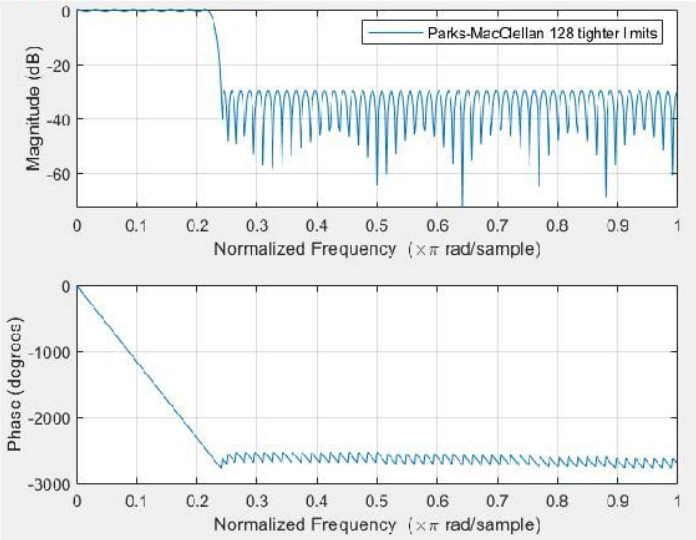


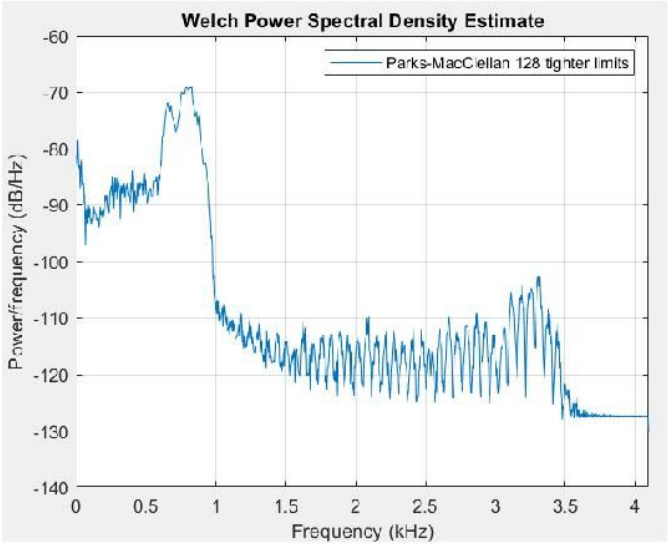
```
s_pm64=conv(s,hpm64);
figure; pwelch(s_pm64,[],[],[],Fs); legend('Parks-MacClellan 64');
s_pm128=conv(s,hpm128);
figure; pwelch(s_pm128,[],[],[],Fs); legend('Parks-MacClellan 128');
```






```
hpm128new=firpm(128, [0 0.11 0.12 0.5]*2, [1 1 0 0]);
figure; freqz(hpm128new,1); legend('Parks-MacClellan 128 tighter limits');
s_pm128new=conv(s,hpm128new);
figure; pwelch(s_pm128new,[],[],[],Fs); legend('Parks-MacClellan 128
tighter limits');
```






```
sound(20*s); % listen to the initial signal s
%sound(20*s_lp); % listen to the filtered signal s_lp
% sound(20*s_pm64);
% sound(20*s_pm128);
% sound(20*s_pm128new);
```

Ερωτήσεις:

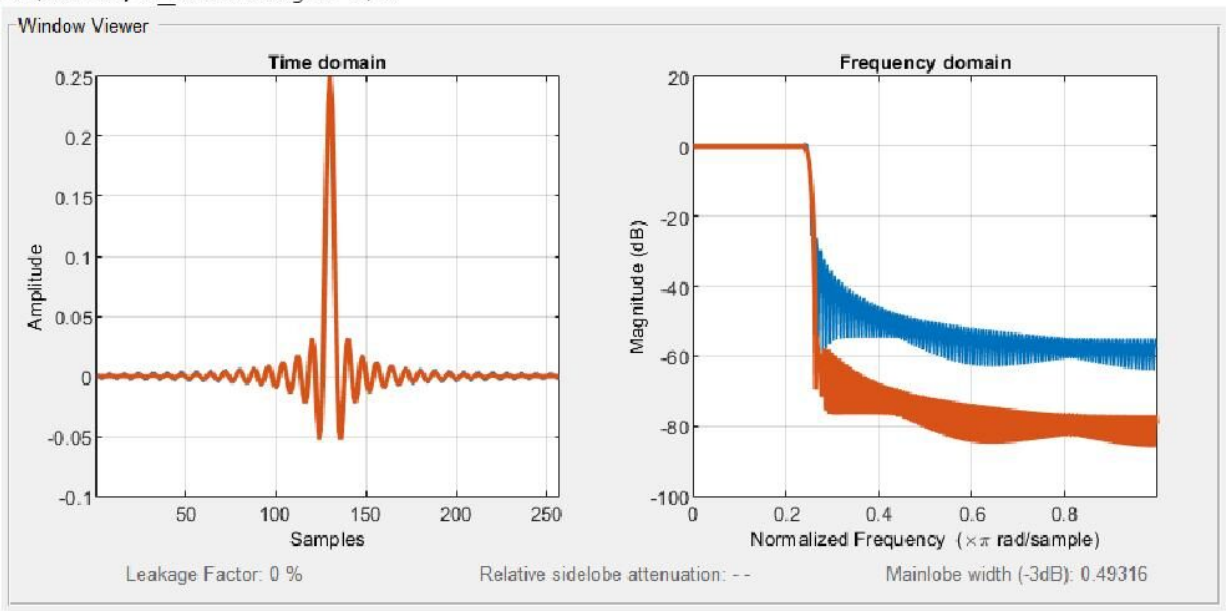
1. Πριν: `h=[h(middle+1:end) h(1:middle)];`

Μετά: `h=[h(middle:end) h(1:middle-1)];`

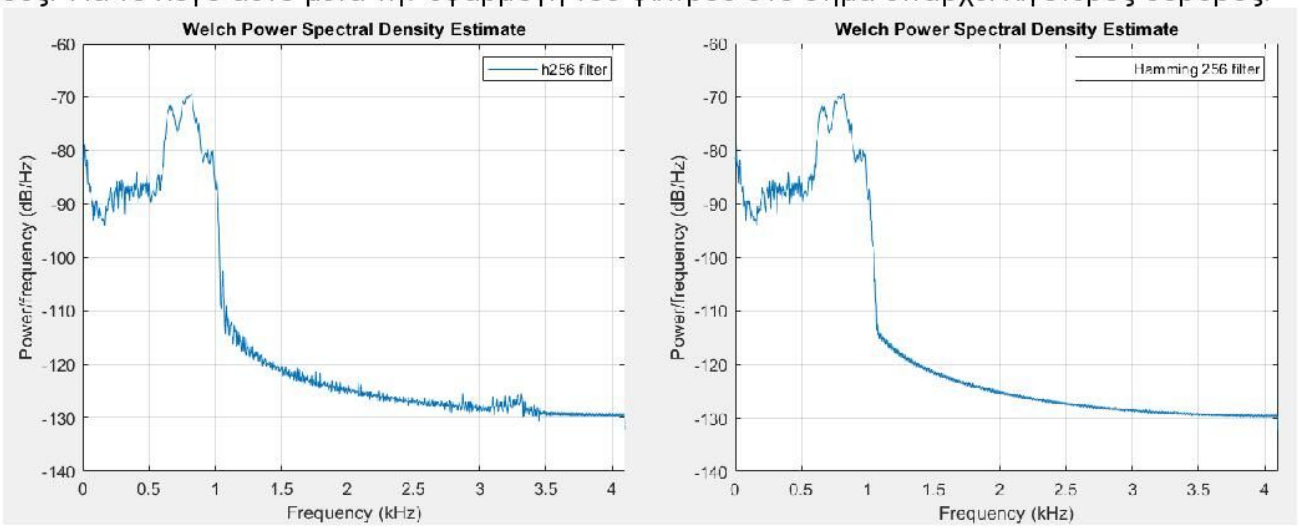
Για να πετύχουμε το ίδιο αποτέλεσμα με `fftshift` και `fftshift` πρέπει το μήκος της h να είναι άρτιο. Με την παραπάνω τροποποίηση το μήκος αλλάζει από 8193 σε 8192.

2. Οι εξής γραμμές κώδικα κάνουν το ζητούμενο:

```
h256=h(middle+1-128:middle+129);
wh256=hamming(length(h256));
h_hamming256=h256.*wh256';
wvtool(h256,h_hamming256);
```

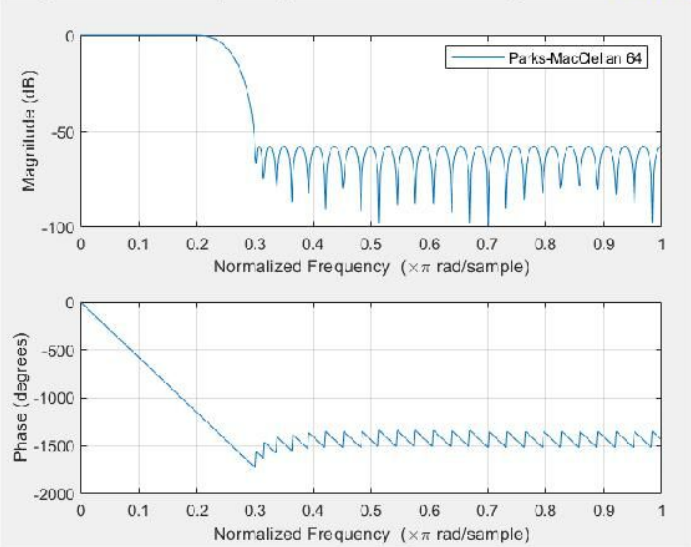


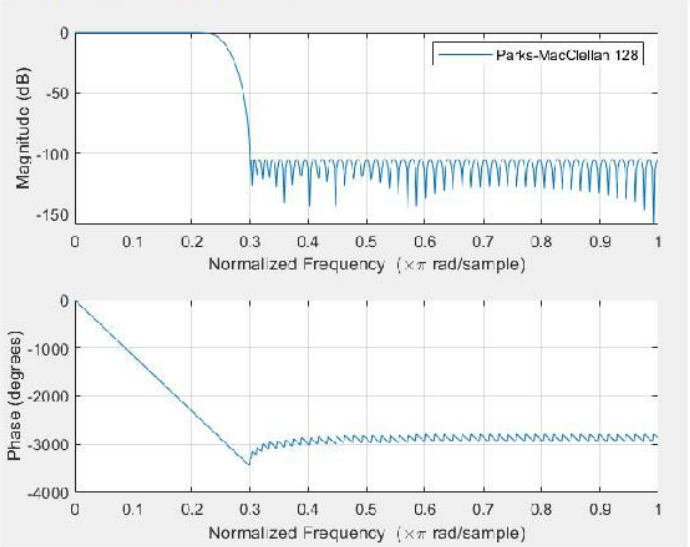
Όπως και στην περίπτωση για μήκος 64, η απόκριση συχνότητας του παραθύρου hamming πέφτει λιγότερο απότομα αλλά οι λωβοί είναι πολύ πιο χαμηλά από το αντίστοιχο ορθογωνικό παράθυρο ίδιου μήκους. Για το λόγο αυτό μετά την εφαρμογή του φίλτρου στο σήμα υπάρχει λιγότερος θόρυβος:



3. Οι εξής γραμμές κώδικα κάνουν το ζητούμενο:

```
hpm64=firpm(64, [0 0.10 0.15 0.5]*2, [1 1 0 0]);
figure; freqz(hpm64,1); legend('Parks-MacClellan 64');
hpm128=firpm(128, [0 0.10 0.15 0.5]*2, [1 1 0 0]);
figure; freqz(hpm128,1); legend('Parks-MacClellan 128');
```

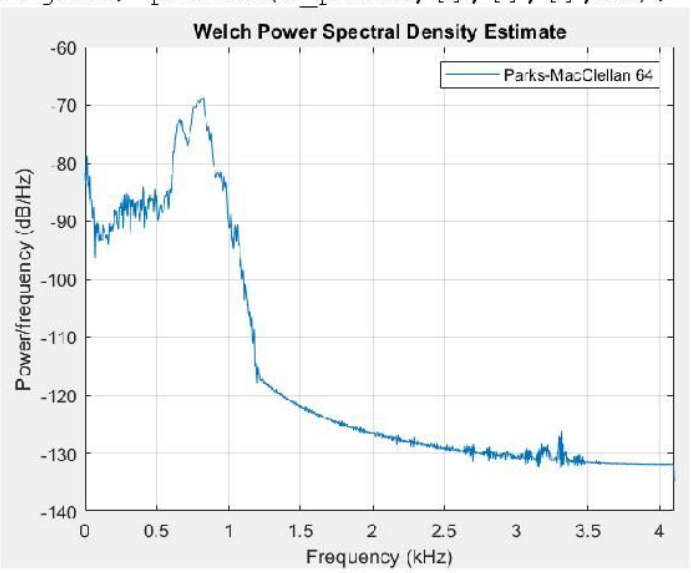


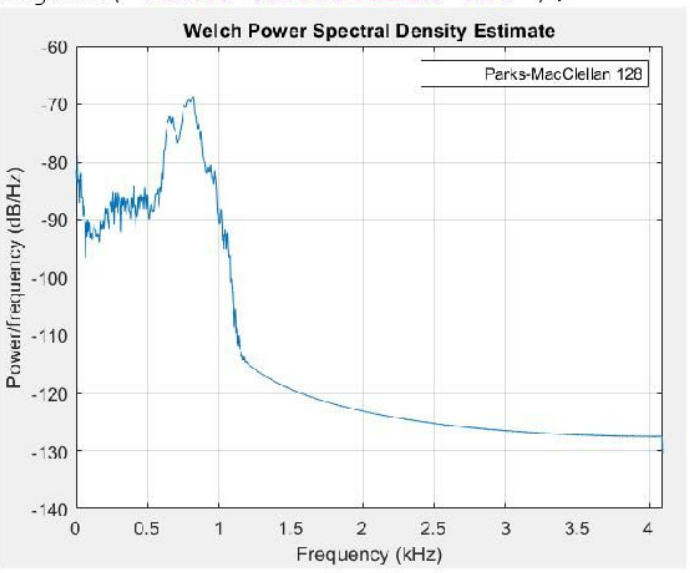


Οι λωβοί είναι πιο στενοί και πιο χαμηλά στο φίλτρο μήκους 128 αλλά αργεί περισσότερο να πέσει ο κύριος λωβός.

Αυτό φαίνεται και στην εφαρμογή του φίλτρου, όπως βλέπουμε το σήμα μετά την εφαρμογή του φίλτρου μεγαλύτερου μήκους έχει λιγότερα σκαμπανεβάσματα θορύβου στις υψηλότερες συχνότητες:

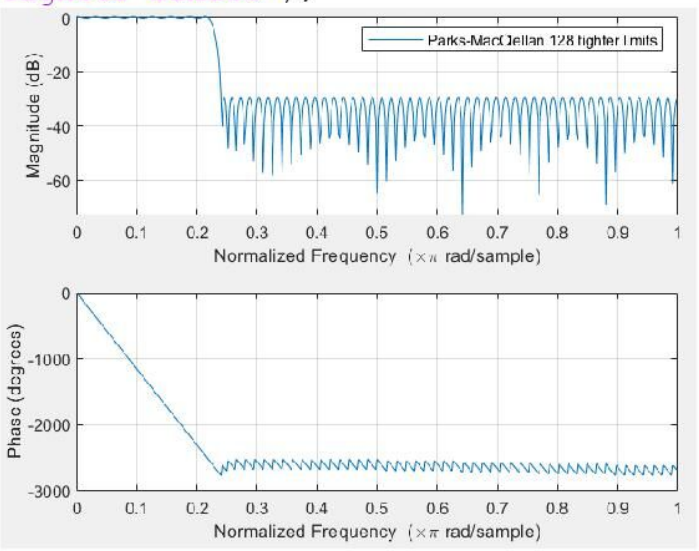
```
s_pm64=conv(s,hpm64);
figure; pwelch(s_pm64,[],[],[],Fs); legend('Parks-MacClellan 64');
s_pm128=conv(s,hpm128);
figure; pwelch(s_pm128,[],[],[],Fs); legend('Parks-MacClellan 128');
```

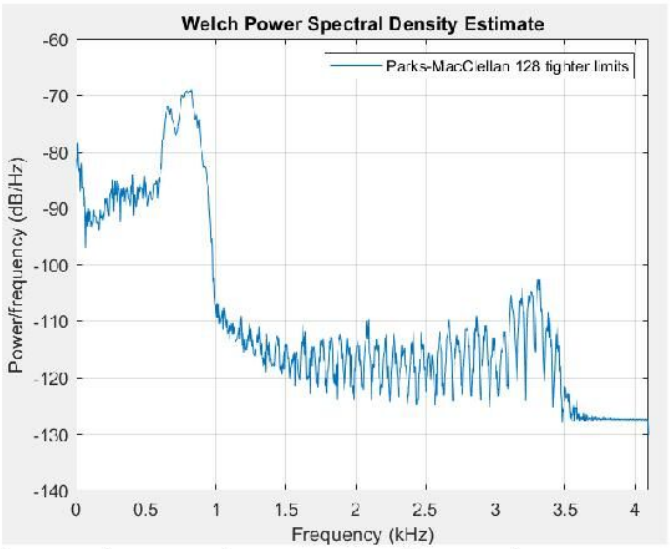




4. Οι εξής γραμμές κώδικα κάνουν το ζητούμενο:

```
hpm128new=firpm(128, [0 0.11 0.12 0.5]*2, [1 1 0 0]);
figure; freqz(hpm128new,1); legend('Parks-MacClellan 128 tighter limits');
s_pm128new=conv(s,hpm128new);
figure; pwelch(s_pm128new,[],[],[],Fs); legend('Parks-MacClellan 128
tighter limits');
```

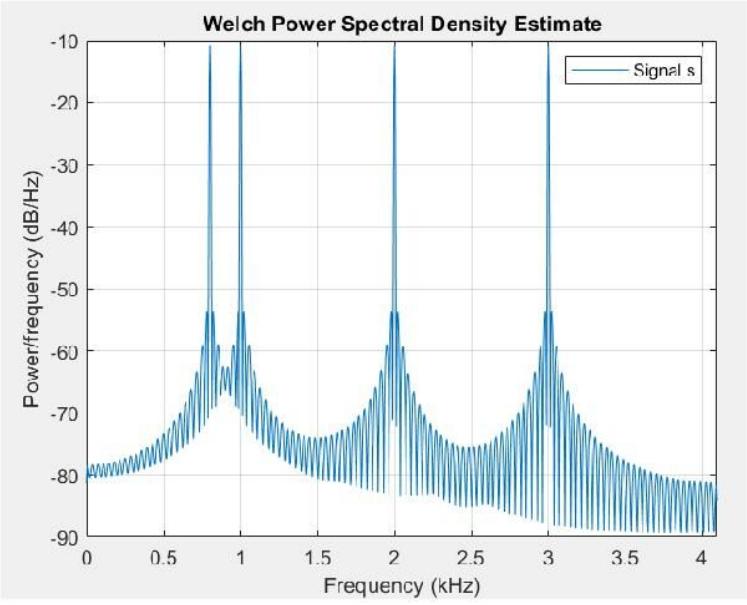




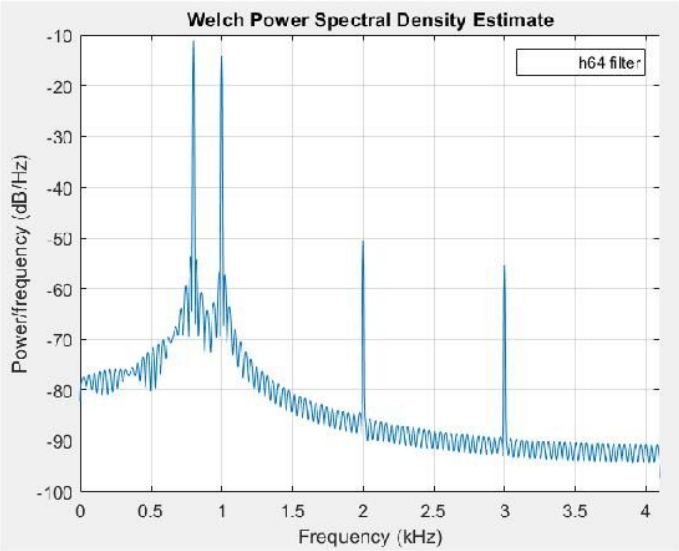
Με στενότερα όρια μειώνεται η αυστηρότητα του φίλτρου, όπως φαίνεται παραπάνω από την απόκριση συχνότητας. Οι λωβοί είναι τώρα στα -30dB αντί στα -110dB που ήταν πριν. Για το λόγο αυτό έχει τόσο πολύ θόρυβο το σήμα μετά το φιλτράρισμα.

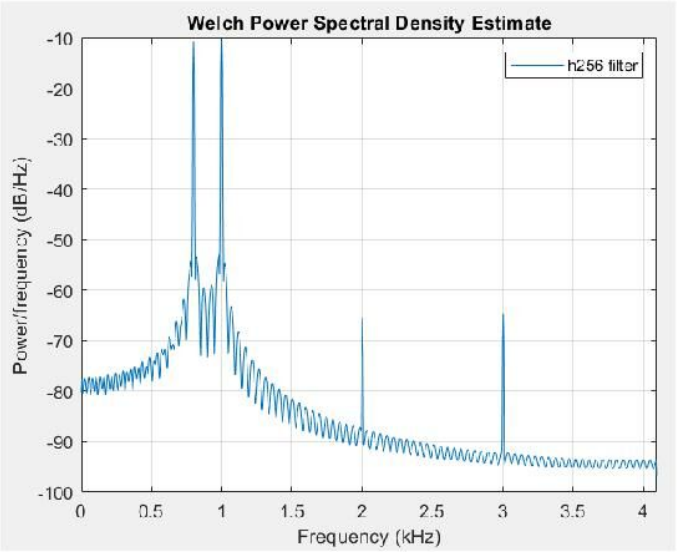
5. Αντί για `load sima` εκτελούμε τον ακόλουθο κώδικα στην αρχή:

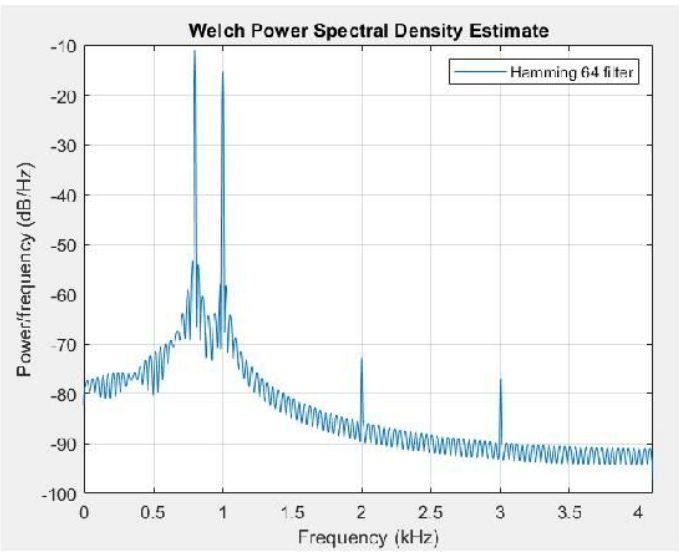
```
Fs=8192; Ts=1/Fs; T=1;
t=0:Ts:T-Ts;
A=1;
s=A*sin(2*pi*800*t)+A*sin(2*pi*1000*t)+A*sin(2*pi*2000*t)+A*sin(2*pi*3000*
t);
```

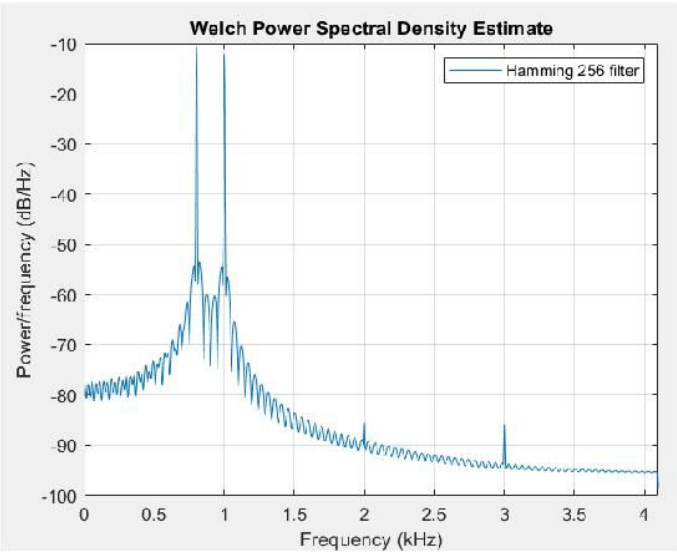


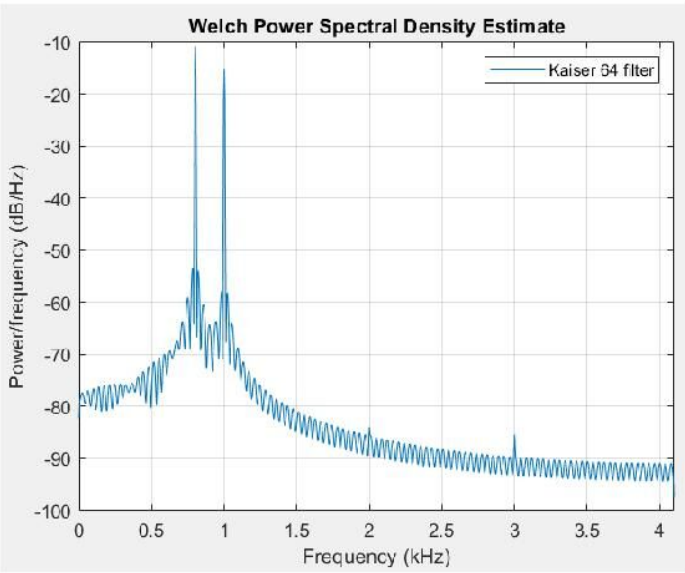
Η εφαρμογή των φίλτρων βγάζει τα εξής αποτελέσματα:

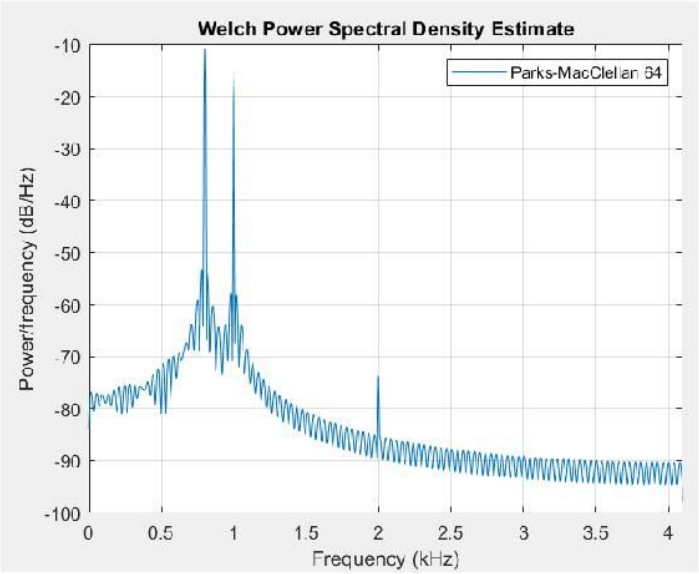


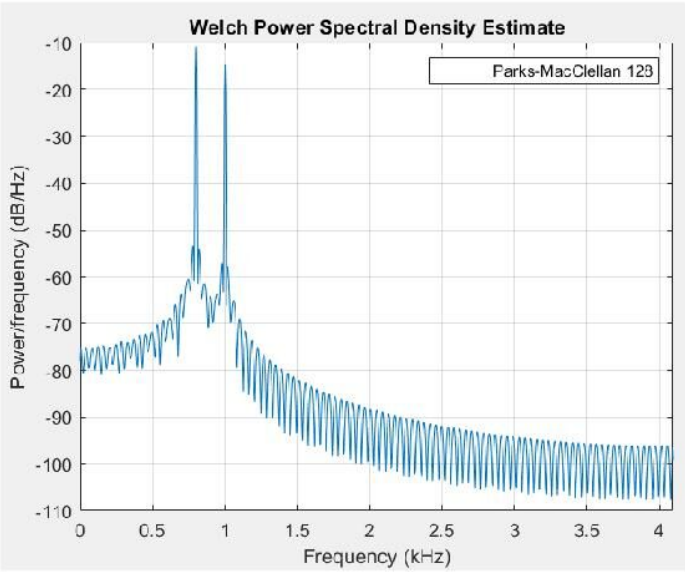


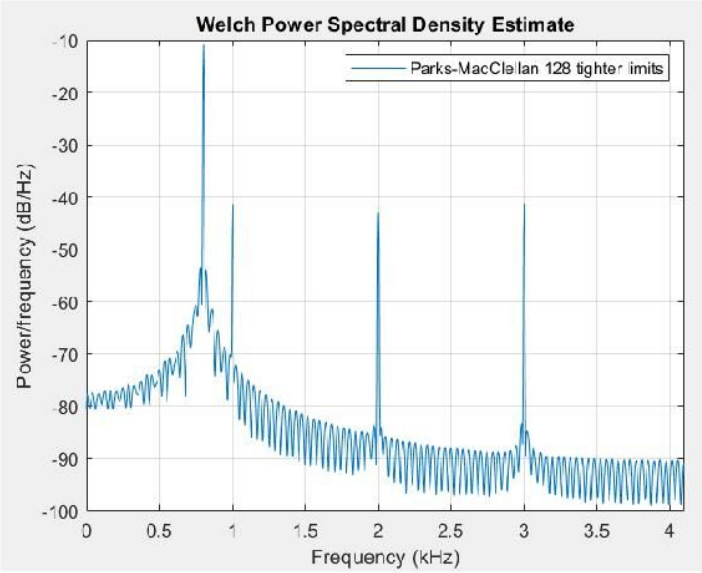






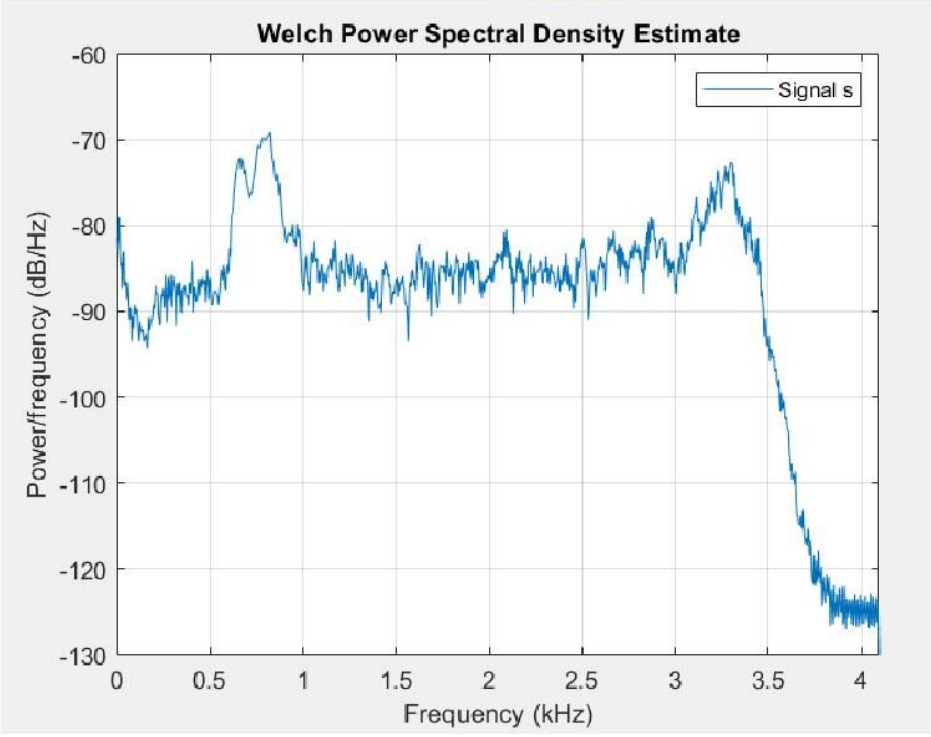




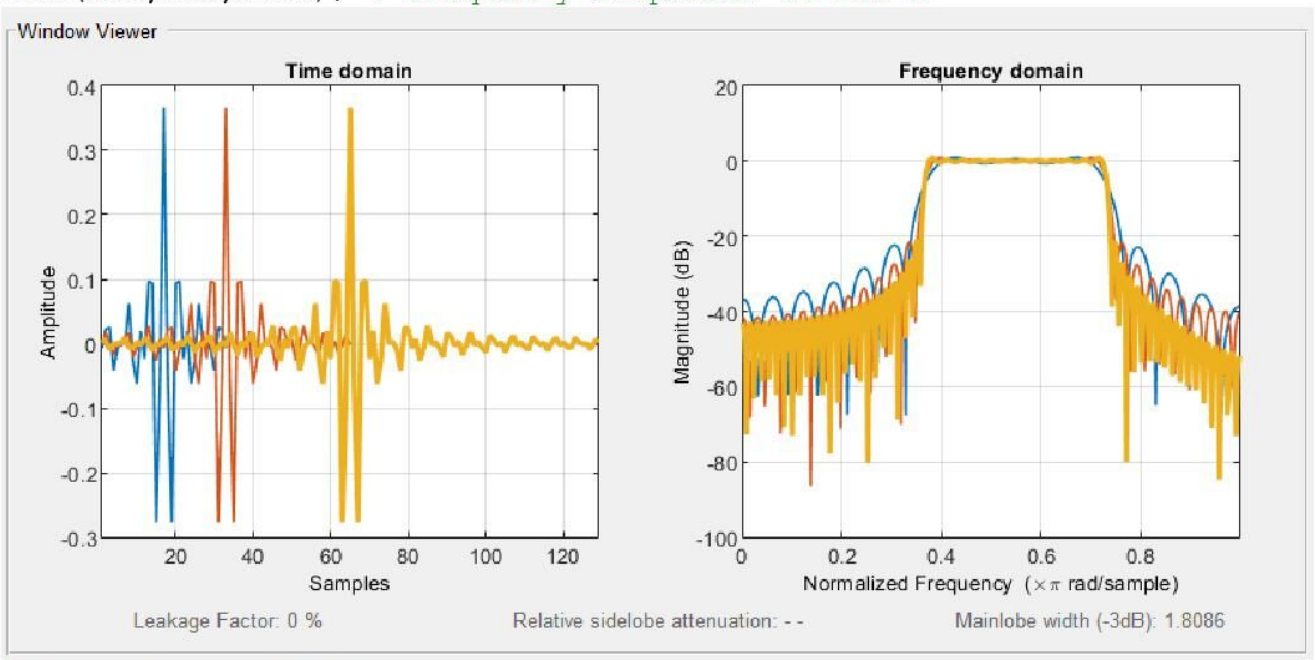


Μέρος 3

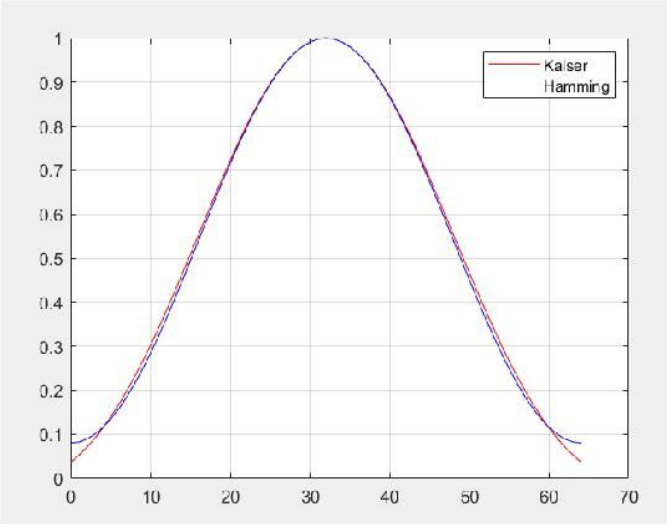
```
clear all; close all;
% File sima.mat has the signal s
% and sampling frequency Fs.
% The spectrum reaches frequencies up to 4kHz
% but everything above 1kHz is noise and must be filtered out.
load sima;
figure; pwelch(s,[],[],[],Fs); legend('Signal s');
```



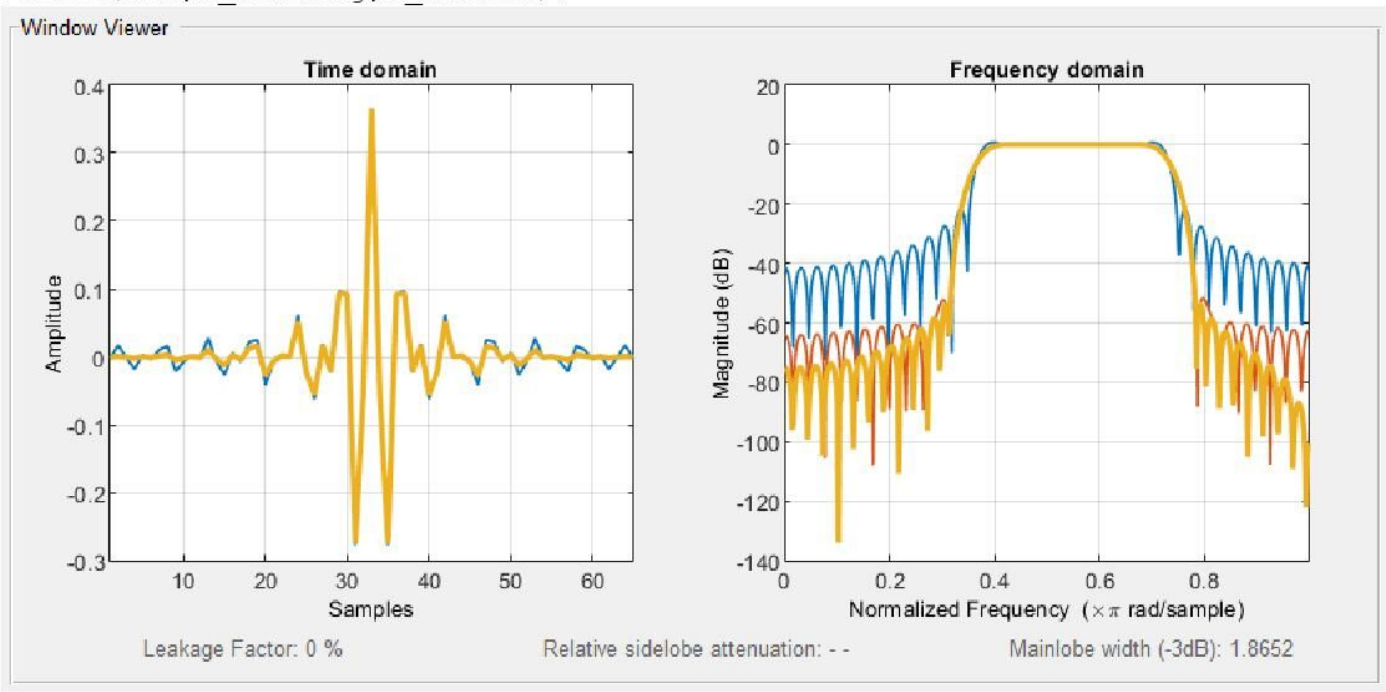
```
% Ideal band pass function H with band 1500kHz-3000kHz
H=[zeros(1,1500) ones(1,1500) zeros(1,2192) ones(1,1500) zeros(1,1500)];
% Impulse response with inverse Fourier transform
% Alternatively, the analytic function Sa(x) can be used
h=ifft(H,'symmetric');
middle=length(h)/2;
h=[h(middle+1:end) h(1:middle)];
h32=h(middle+1-16:middle+17);
h64=h(middle+1-32:middle+33);
h128=h(middle+1-64:middle+65);
% figure; stem([0:length(h64)-1],h64); grid;
% figure; freqz(h64,1); % frequency response of h64
wvtool(h32,h64,h128); % frequency responses of cut h
```



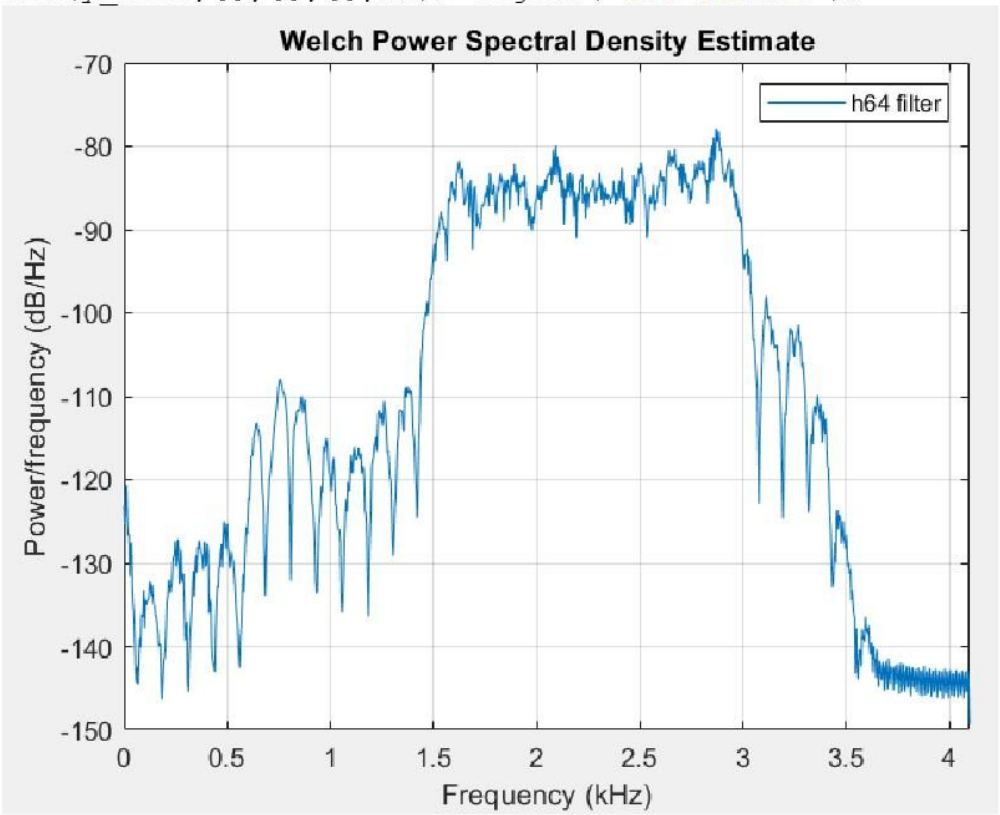

```
% side peaks are high!
% Multiply the cut impulse response with windows
% We use h64 with hamming and kaiser windows
wh=hamming(length(h64));
wk=kaiser(length(h64),5);
figure; plot(0:64,wk,'r',0:64,wh,'b'); grid; legend('Kaiser','Hamming');
```



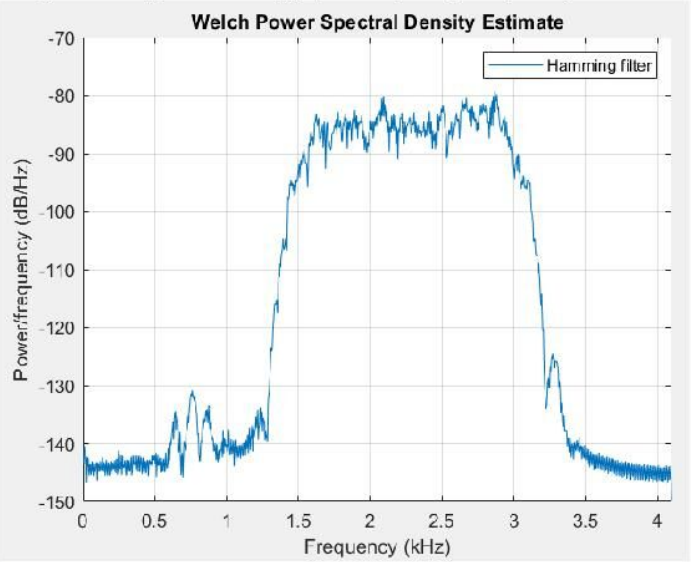
```
h_hamming=h64.*wh';
% figure; stem([0:length(h64)-1],h_hamming); grid;
% figure; freqz(h_hamming,1);
h_kaiser=h64.*wk';
wvtool(h64,h_hamming,h_kaiser);
```

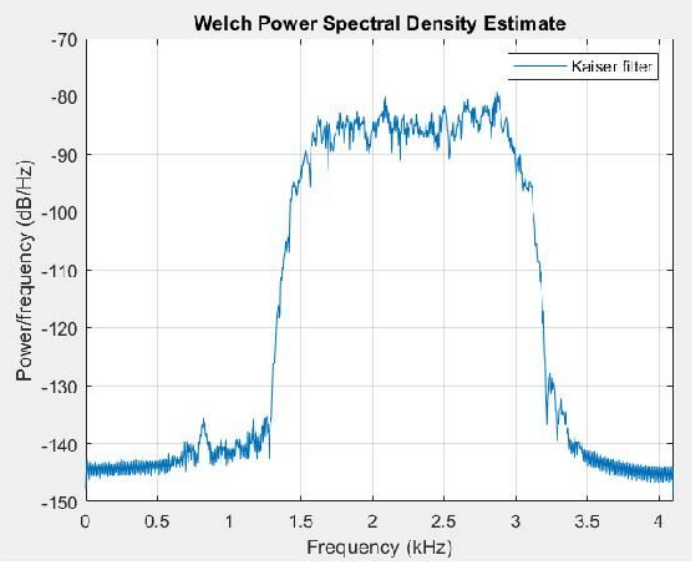



```
% Filter the signal with each filter
y_rect=conv(s,h64);
figure; pwelch(y_rect,[],[],[],Fs); legend('h64 filter');
```

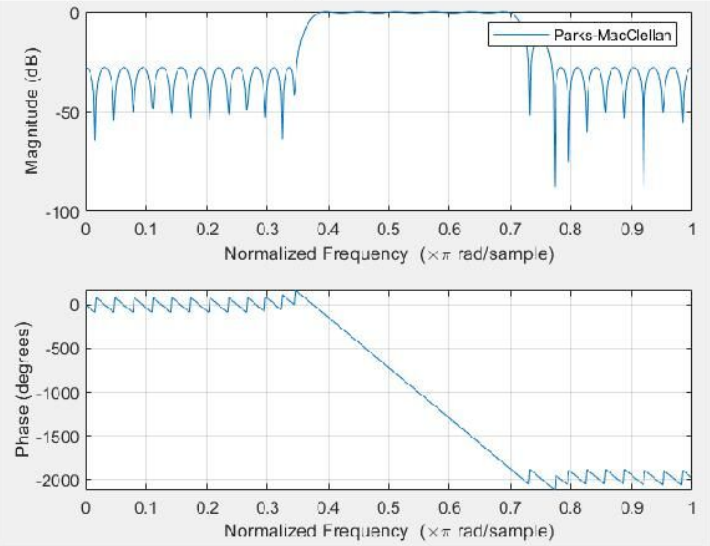


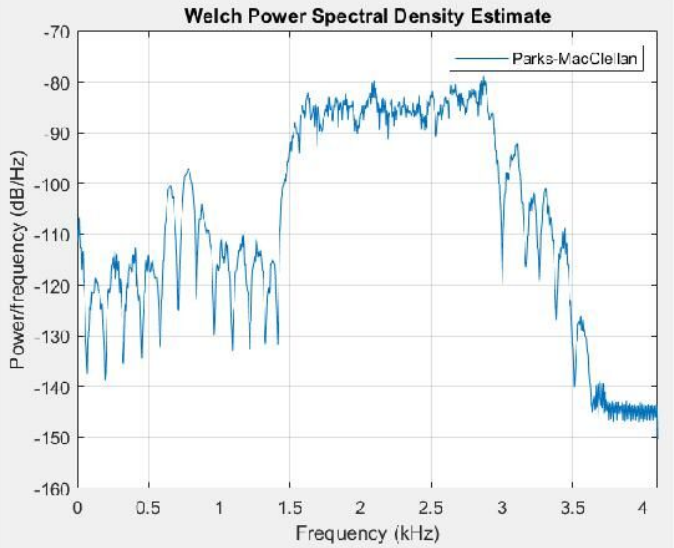
```
y_hamm=conv(s,h_hamming); legend('Hamming filter');
figure; pwelch(y_hamm,[],[],[],Fs);
y_kais=conv(s,h_kaiser); legend('Kaiser filter');
figure; pwelch(y_kais,[],[],[],Fs);
```






```
%
% Low pass Parks-MacClellan
f1=1500; f2=3000;
f=2*[0 f1*0.95 f1*1.05 f2*0.95 f2*1.05 Fs/2]/Fs;
hpm=firpm(64, f, [0 0 1 1 0 0]);
figure; freqz(hpm,1); legend('Parks-MacClellan');
s_pm=conv(s,hpm);
figure; pwelch(s_pm,[],[],[],Fs); legend('Parks-MacClellan');
```






```
sound(20*s); % listen to the initial signal s
%sound(20*s_lp); % listen to the filtered signal s_lp
%sound(20*s_pm);
```