

ΨΗΦΙΑΚΗ ΕΠΕΞΕΡΓΑΣΙΑ ΣΗΜΑΤΩΝ

Project

Φεβρουάριος 20, 2022

Σαπουντζή Αθανασία Δέσποινα 2624

Για το project χρησιμοποίησα το online matlab (R2021)

ΜΕΡΟΣ Α: ΣΧΕΔΙΑΣΗ ΦΙΛΤΡΩΝ

A1. (FIR κάνοντας χρήση παραθύρου Hamming)

Απο τη μεθοδολογία για το FIR φίλτρο, το φίλτρο είναι 5ου βαθμού. Η συχνότητα αποκοπής υπολογίζεται από τον τύπο $(ws + wp) / 2$. Άρα από τους υπολογισμούς το φίλτρο είναι βαθμού $N = 40$.

Κωδικας

```
%methodology for construction of FIR filter
fsample = 16000;
As = 40;
Rp = 1;
fs = 0.15;
fp = 0.05;
ws=0.3*pi;
wp=0.1*pi;
%minimum filter length M from hamming's equation:
N = ceil( 4/(fs-fp) );
%discrete time fir-hamming-lowpass coefficients
coefficients
= fir1(N, (wp+ws)/2, 'low', hamming(N+1));
```

A2. (IIR φίλτρο Butterworth με bilinear transform)

Απο τη μεθοδολογία για το IIR φίλτρο, το φίλτρο είναι $N = 5$

Κωδικας

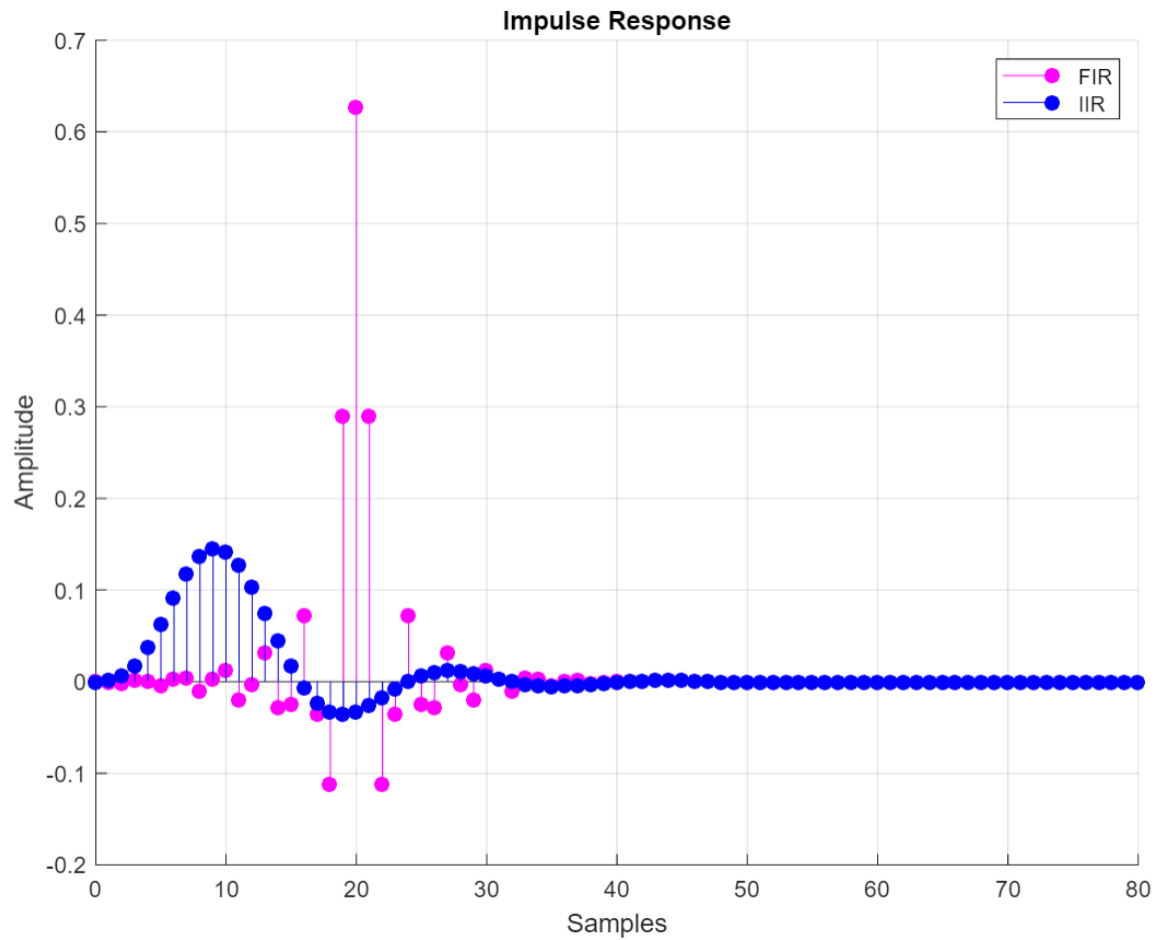
```
%methodology for construction of IIR filter Butterworth via bilinear
transform
%First convert to analog
WP = 2*fsample*tan(wp/2);
WS = 2*fsample*tan(ws/2);
%Butterworth
[N,Wn] = buttord(WP, WS, Rp, As, 's');
[z, p] = butter(N, Wn, 'low', 's');
%bilinear transform
[num, den] = bilinear(z, p, fsample);
```

Σε κάθε περίπτωση:

- Κρουστικές αποκρίσεις των φίλτρων

Κωδικας

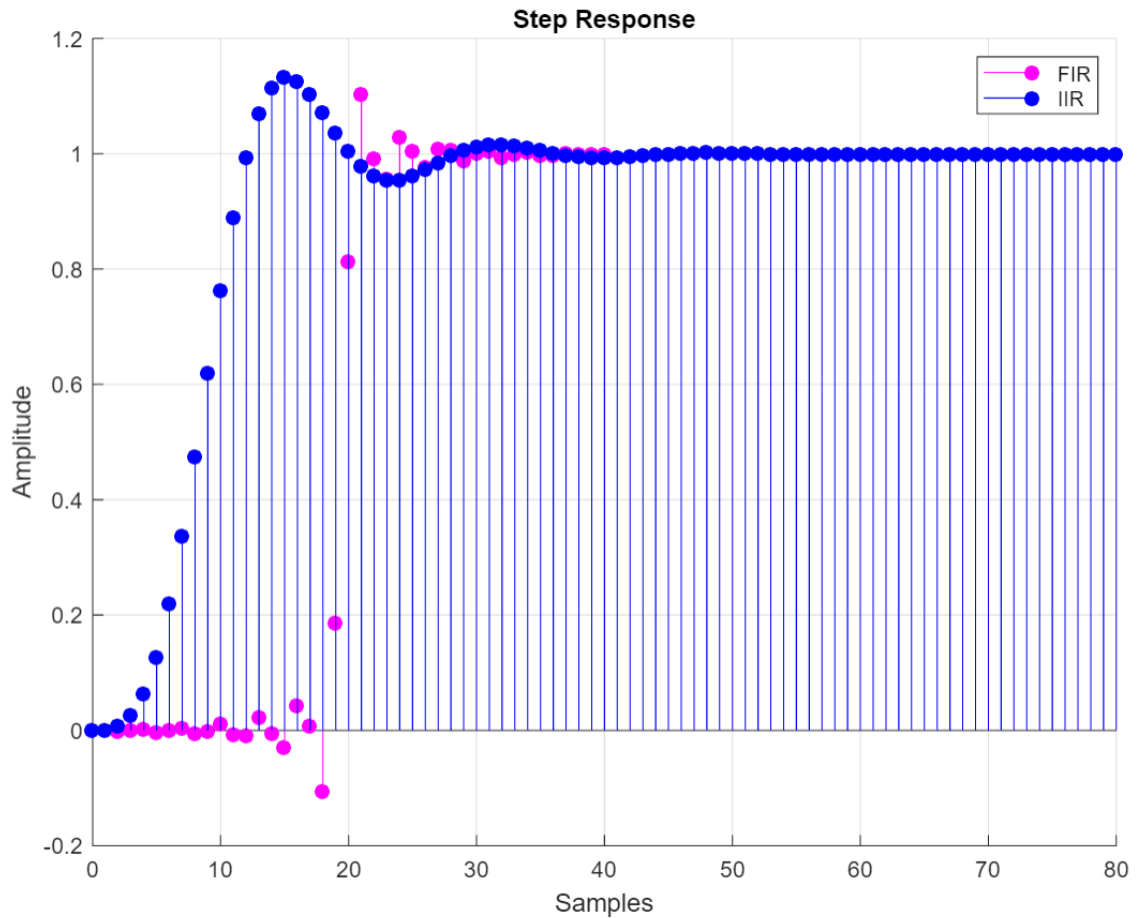
```
hold on;
figure (1);
title('Impulse Response');
grid('ON');
ylabel('Amplitude');
xlabel('Samples');
%for FIR filter
[firY, firX] = impz(coefficients);
stem(firX, firY, 'm', 'filled');
hold on;
%for IIR filter
[iirY, iirX] = impz(num, den);
stem(iirX, iirY, 'b', 'filled');
hold on;
legend('FIR', 'IIR');
hold off;
```



- Βηματικές αποκρίσεις των φίλτρων

Κωδικας

```
figure (2);
hold on;
title('Step Response');
grid('ON');
xlabel('Samples');
ylabel('Amplitude');
%for FIR filter Step response
[FIR_y, FIR_x] = stepz(coefficients);
stem(FIR_x, FIR_y, 'm', 'filled');
hold on;
%for IIR filter Step response
[IIR_y, IIR_x] = stepz(num, den);
stem(IIR_x, IIR_y, 'b', 'filled');
hold on;
legend('FIR', 'IIR');
hold off;
```



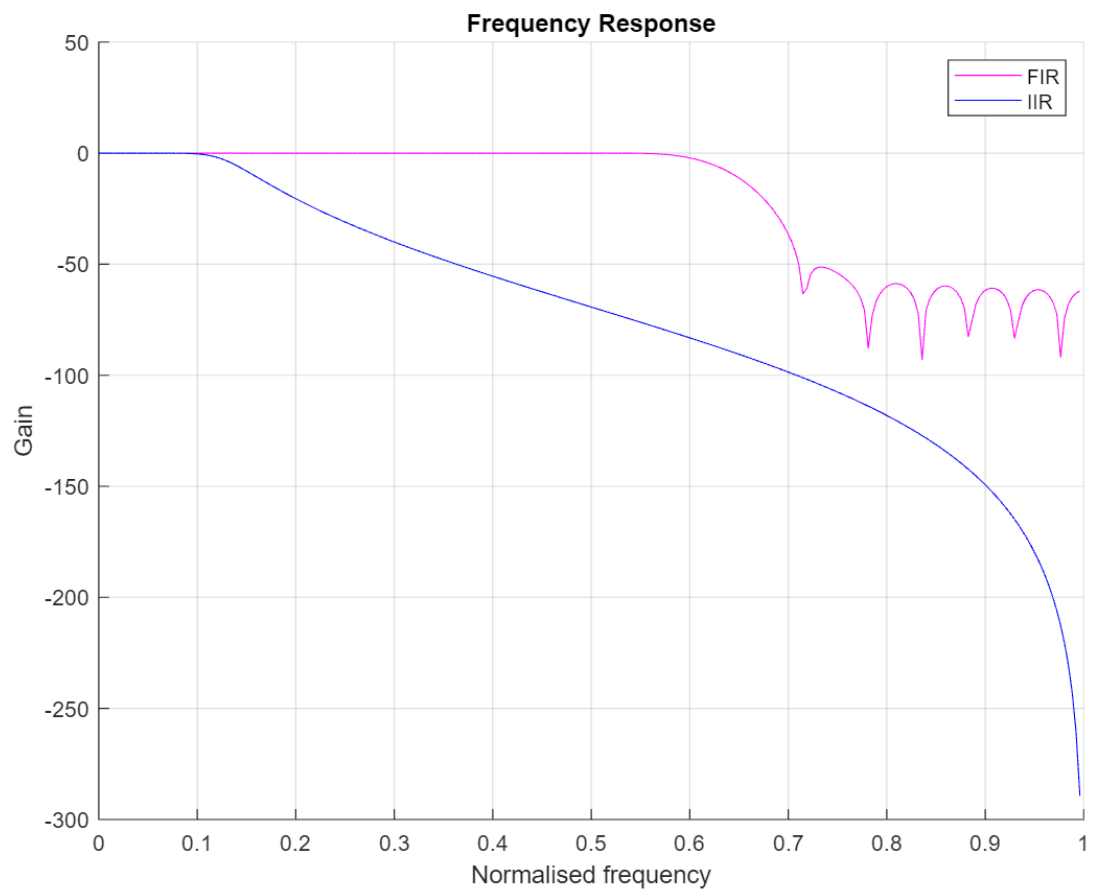
- Μέτρο απόκρισης συχνότητας σε λογαριθμική κλίμακα (σε dB)

Κωδικας

```
%frequency domain FIR
[FIR_h,FIR_w]=freqz(coefficients,1,256);
%amplitude-gain FIR
FIR_h_inDB = 20*log10(abs(FIR_h));
%frequency domain IIR
[IIR_h,IIR_w]=freqz(num,den,256);
%amplitude-gain IIR
IIR_h_inDB = 20*log10(abs(IIR_h));
```

```
figure(3);
hold on;
title('Frequency Response');
grid('ON');
ylabel('Gain');
xlabel('Normalized frequency');
%normalized frequency(Nyquist) FIR
plot(FIR_w/(pi), FIR_h_inDB, 'm');
hold on;
%Normalized frequency(Nyquist) IIR
plot(IIR_w/(pi), IIR_h_inDB, 'b');
legend('FIR', 'IIR');
```

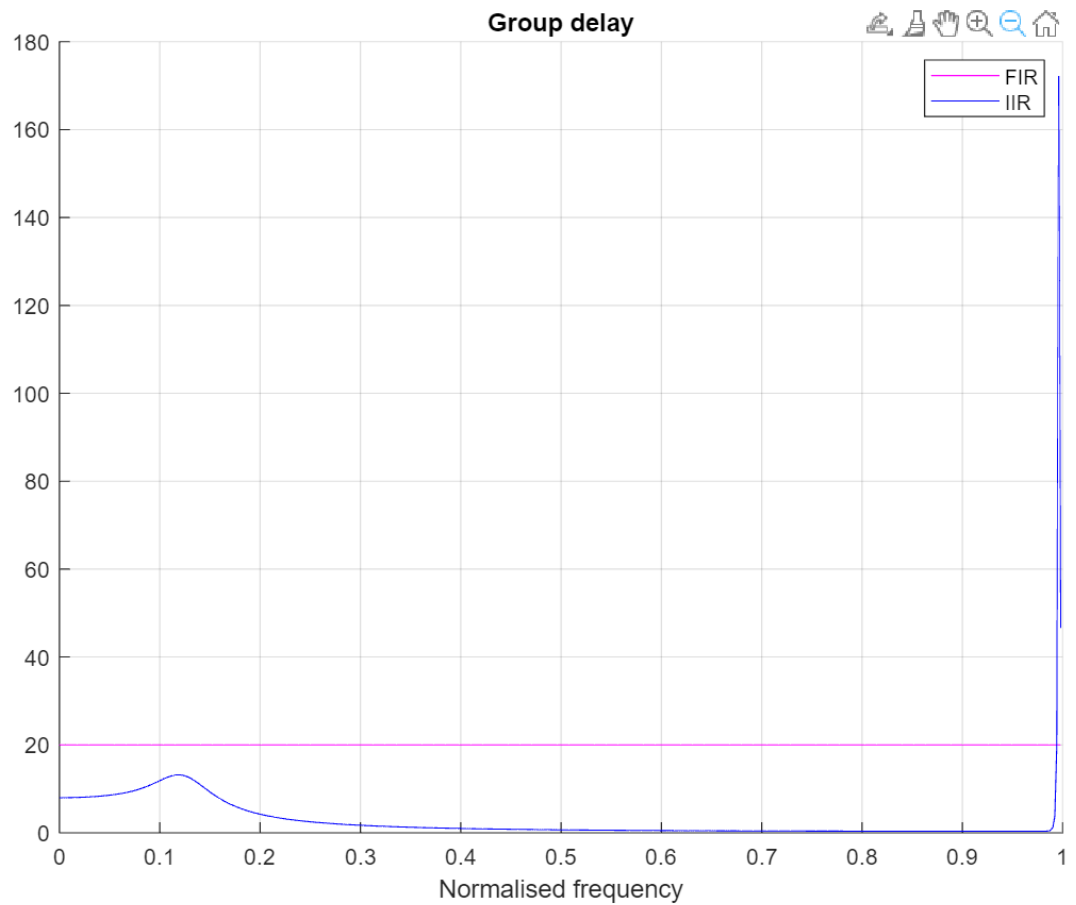
```
hold off;
```



- Καθυστέρηση ομάδας.

Κωδικας

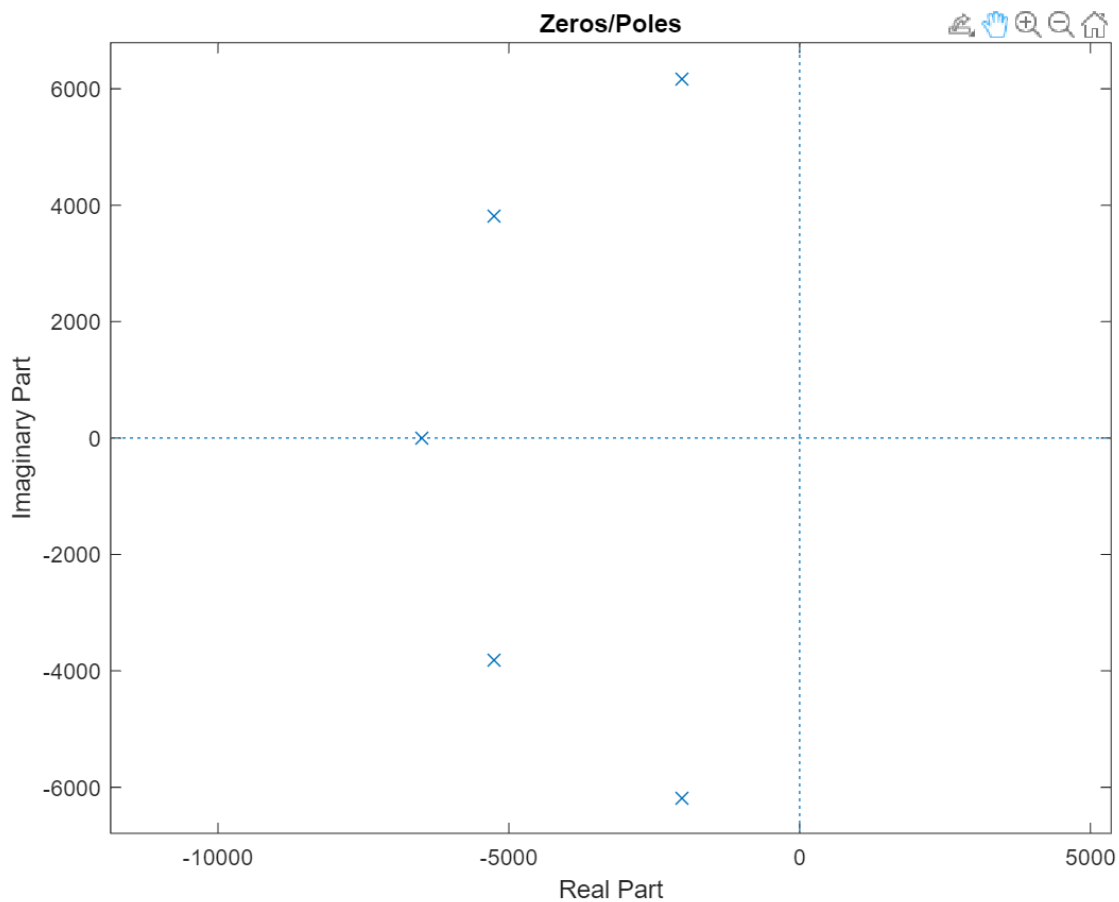
```
./deviceQuery Starting...  
%by default there are 8192 samples  
[FIR_gd, FIR_w] = grpdelay(coefficients);  
[IIR_gd, IIR_w] = grpdelay(num, den);  
figure(4);  
hold on;  
title('Group delay');  
grid('ON');  
xlabel('Normalized frequency');  
plot(FIR_w/pi, FIR_gd, 'm');  
hold on;  
plot(IIR_w/pi, IIR_gd, 'b');  
legend('FIR', 'IIR');  
hold off;
```



- Διάγραμμα μηδενικών και πόλων (για το IIR φίλτρο μόνο)

Κωδικας

```
figure(5);
%only for IIR filter this time, because FIR does not have poles
[b, a] = zplane(z, p);
hold on;
title('Zeros/Poles');
hold off;
```



ΜΕΡΟΣ Β: ΑΝΑΛΥΣΗ ΣΗΜΑΤΩΝ ΜΕ DFT

B1.1

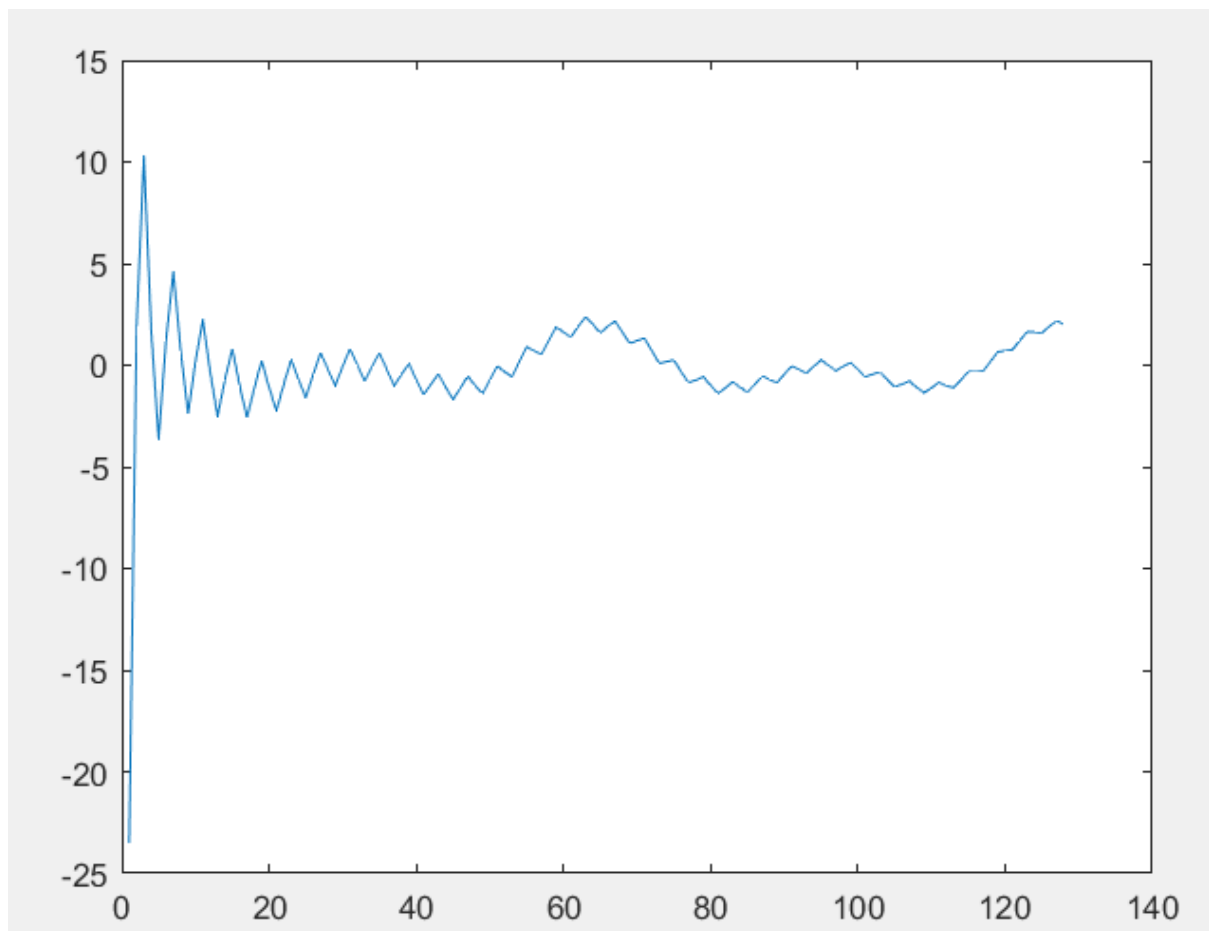
Αρχικά εφάρμοσα παραθύρωση με ορθογώνιο παράθυρο στο σήμα

```
%signal x[n] = 80δ[n] - 80sin(pi*n/2)/(pi*n) + cos(pi*n/32) + cos(pi*n/16)
N = 10000;
n= 0:N-1;
Nbig = 10000;
Nbins = 8000;
signal = 80*dirac(n) - 80*sin(pi*n/2)/(pi*n) + cos(pi*n/32) + cos(pi*n/16);
L =8;
%rectangular window
X = rectwin(L);
wrect = rectwin(L);
%apply windowing technique to signal
```

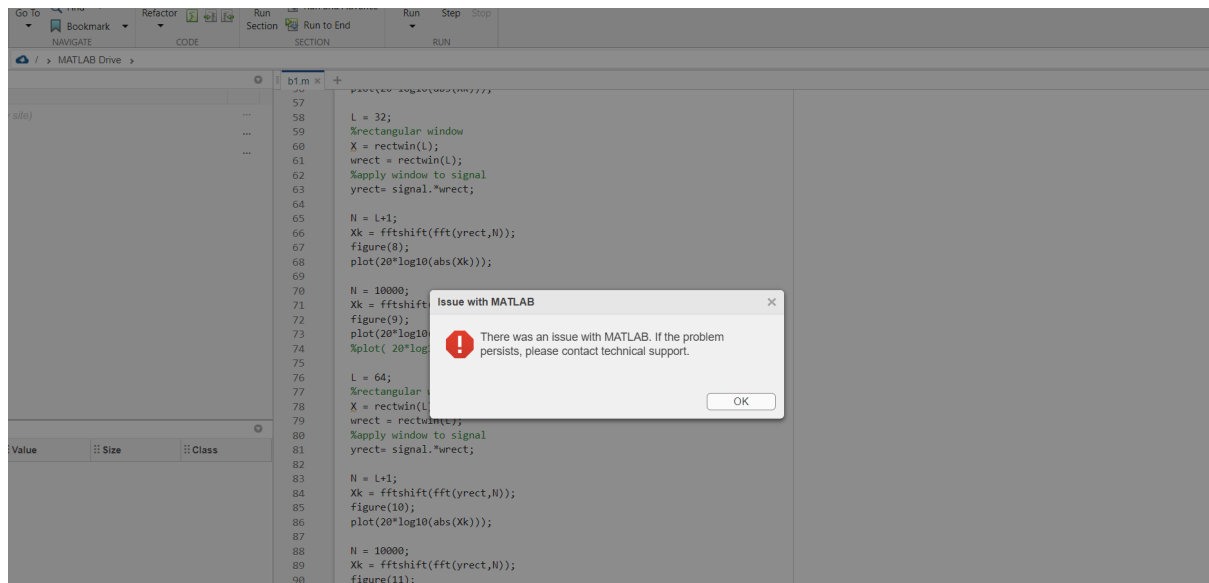
```

wrect = rectwin(L);
%apply windowing technique to signal
for n = 1:8
    yrect(n) = (80*dirac(n) - 80*sin(pi*n/2)/(pi*n) + cos(pi*n/32) +
cos(pi*n/16))*wrect(n);
end
figure(1);
plot(yrect);

```



Δυστυχώς το online matlab διέκοπτε τη σύνδεση όποτε προσπαθούσα να κανω πλοτ



The screenshot shows the MATLAB online IDE interface. The code editor displays the following MATLAB code:

```
57  
58 L = 32;  
59 %rectangular window  
60 X = rectwin(L);  
61 yrect = rectwin(L);  
62 %apply window to signal  
63 yrect = signal.*yrect;  
64  
65 N = L+1;  
66 Xk = fftshift(fft(yrect,N));  
67 figure(8);  
68 plot(20*log10(abs(Xk)));  
69  
70 N = 10000;  
71 Xk = fftshift(fft(yrect,N));  
72 figure(9);  
73 plot(20*log10(abs(Xk)));  
74 %plot( 20*log10(abs(Xk)));  
75  
76 L = 64;  
77 %rectangular window  
78 X = rectwin(L);  
79 yrect = rectwin(L);  
80 %apply window to signal  
81 yrect = signal.*yrect;  
82  
83 N = L+1;  
84 Xk = fftshift(fft(yrect,N));  
85 figure(10);  
86 plot(20*log10(abs(Xk)));  
87  
88 N = 10000;  
89 Xk = fftshift(fft(yrect,N));  
90 figure(11);
```

An error dialog box titled "Issue with MATLAB" is displayed over the code, containing the message: "There was an issue with MATLAB. If the problem persists, please contact technical support." with an "OK" button.

Κατέβασα και το free trial matlab αλλά πάγωνε όποτε το έτρεχα

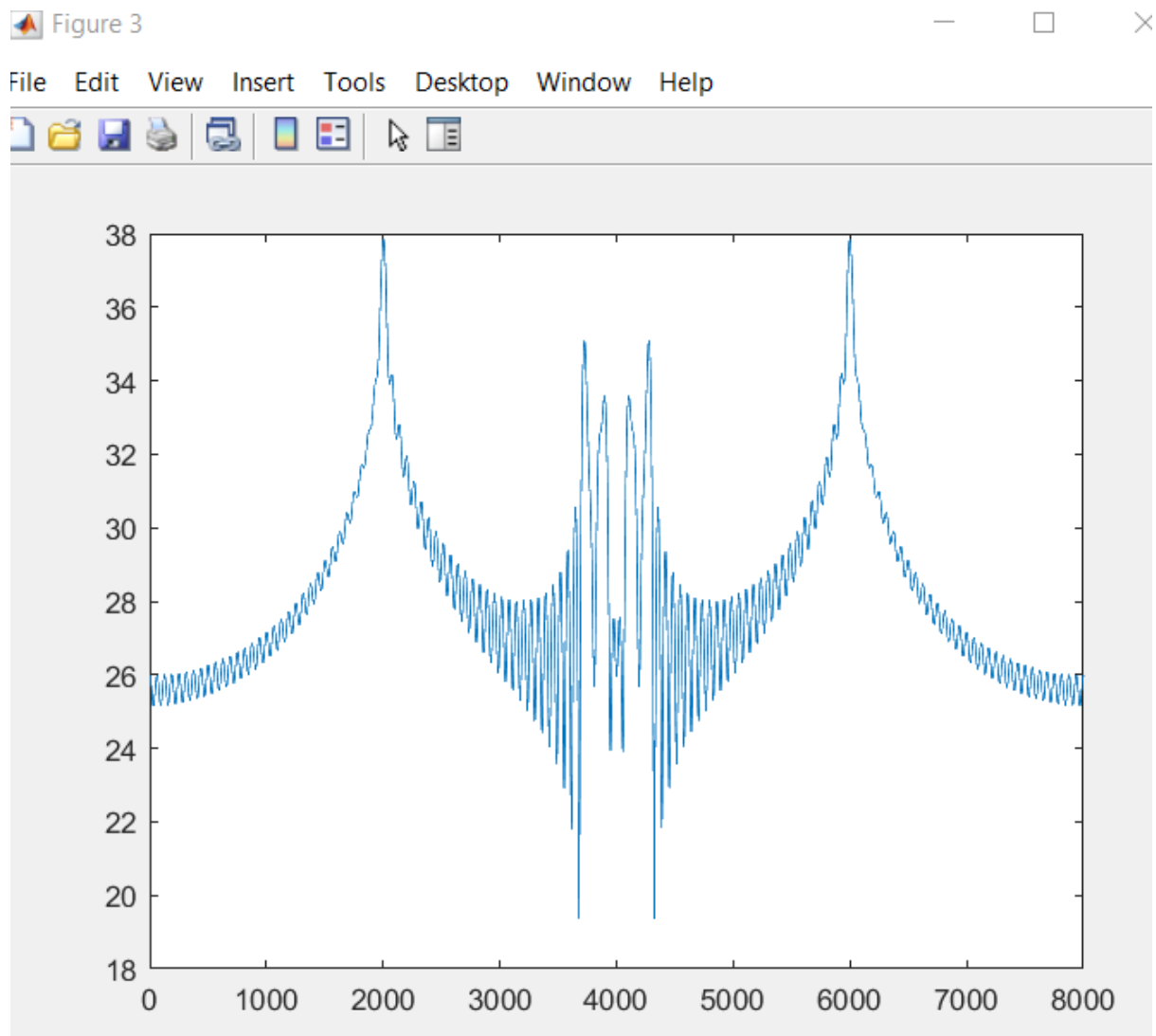
Τελικα αυτο συνεβη οποτε εκανα signal = signal.* yrect επομένως εφάρμοσα άλλο τρόπο με for loop

Σχεδίαση του μέτρου του DFT του παραθυρωμένου σήματος

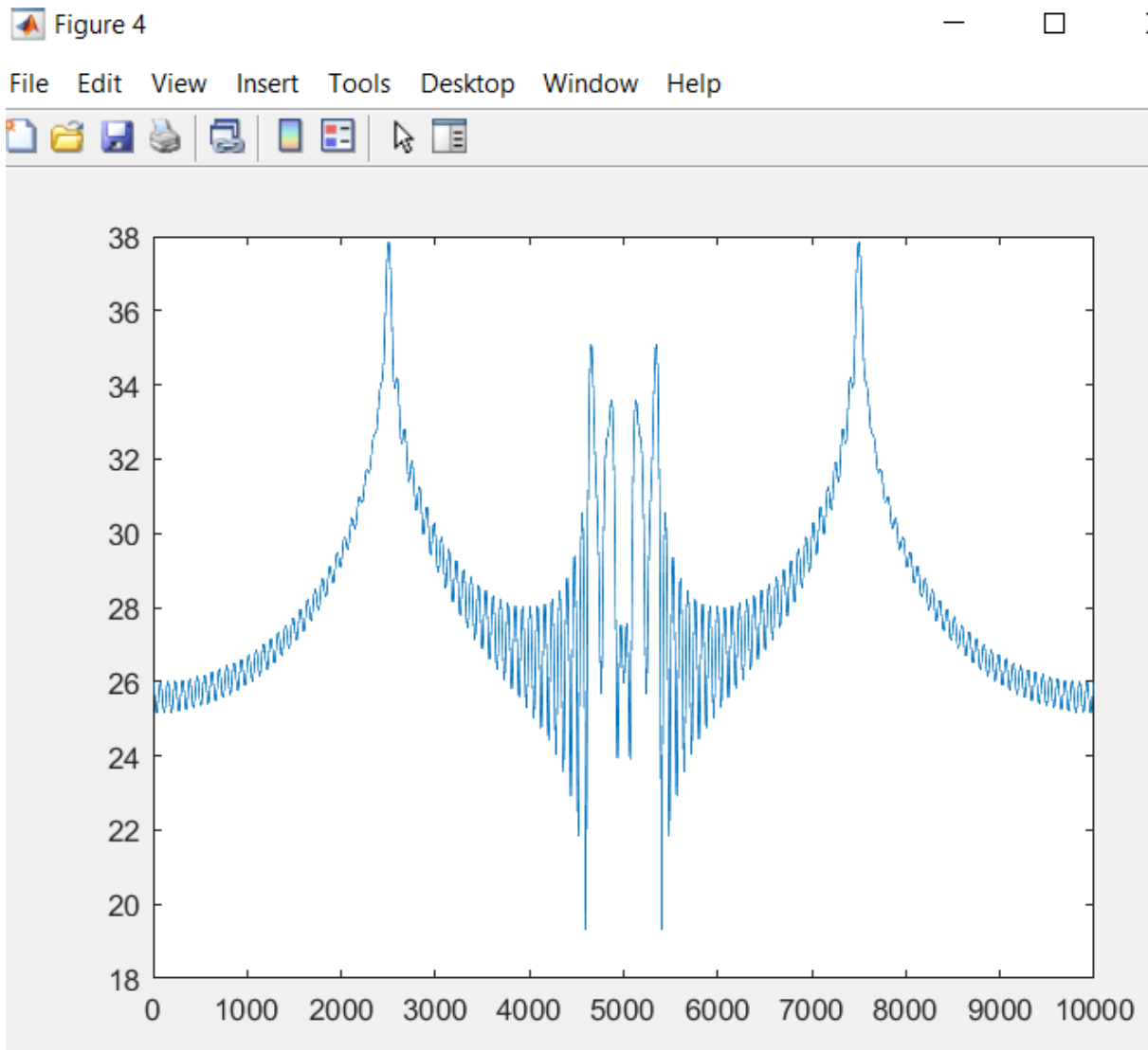
B1.2

Σχεδίαση του μέτρου του DFT του παραθυρωμένου σήματος

```
N = L+1;  
Xk = fftshift(fft(yrect,N));  
figure(3);  
plot(20*log10(abs(Xk)));
```



```
N = 10000;  
Xk = fftshift(fft(yrect,N));  
figure(4);  
plot(20*log10(abs(Xk)));
```



B1.3

Για τα υπόλοιπα L

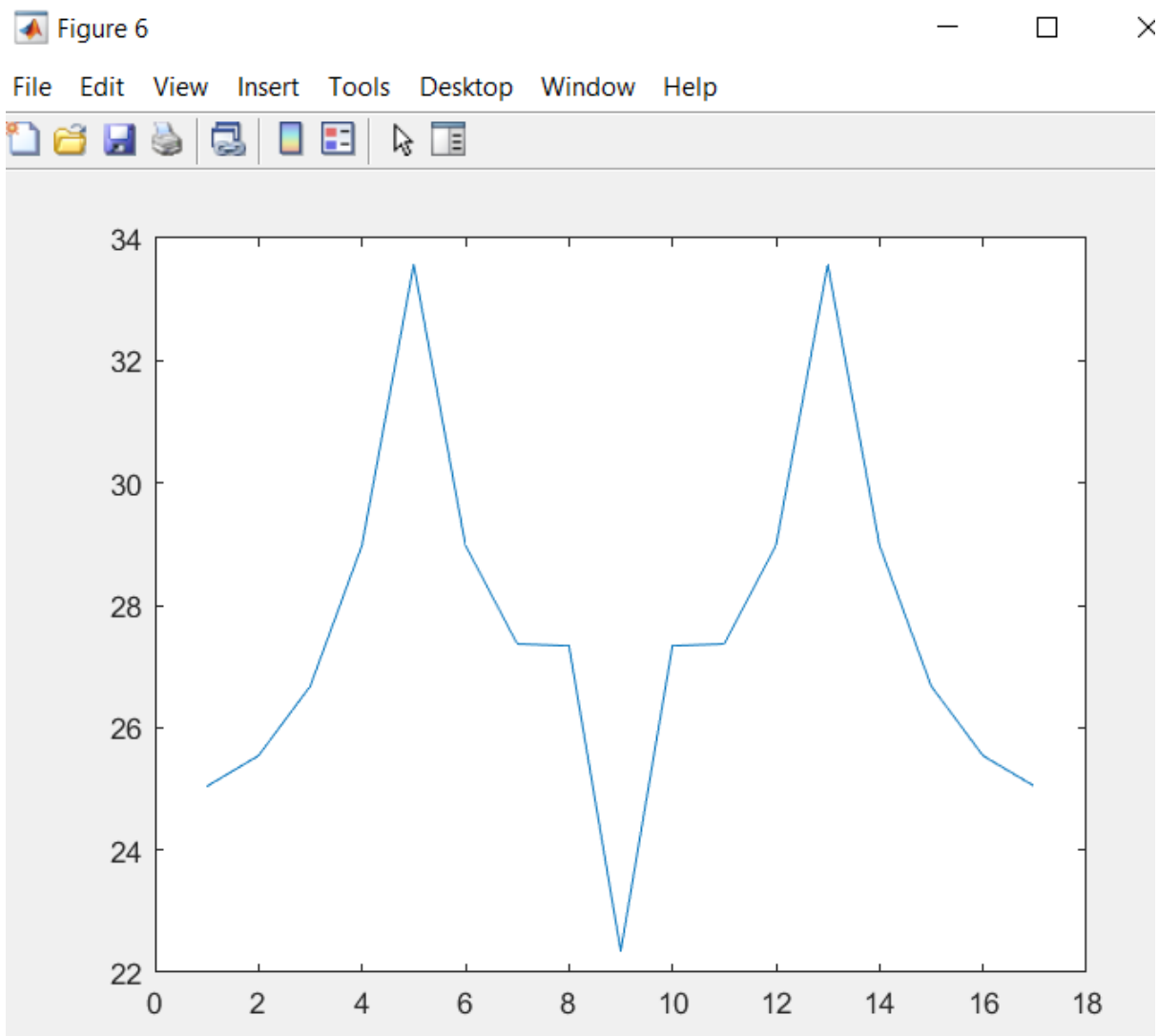
```

%signal x[n] = 80δ[n] - 80sin(pi*n/2)/(pi*n) + cos(pi*n/32) + cos(pi*n/16)

L =16;
%rectangular window
wrect = rectwin(L);
%apply window to signal
for n = 1:16
    yrect(n) = (80*dirac(n) - 80*sin(pi*n/2)/(pi*n) + cos(pi*n/32) +
cos(pi*n/16))*wrect(n);
end

N = L+1;
Xk = fftshift(fft(yrect,N));
figure(6);
plot(20*log10(abs(Xk)));

```

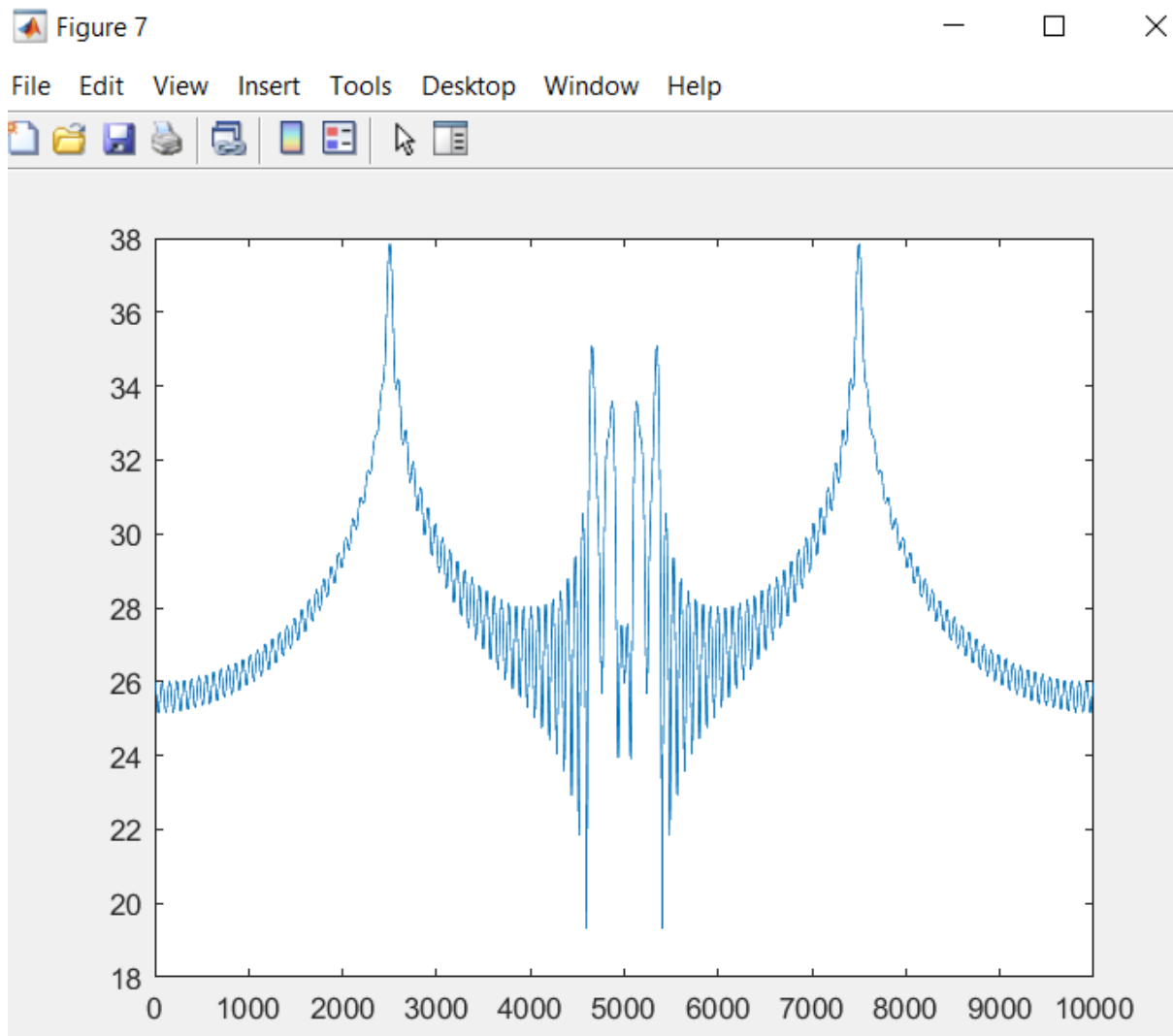


```

N = 10000;
Xk = fftshift(fft(yrect,N));

```

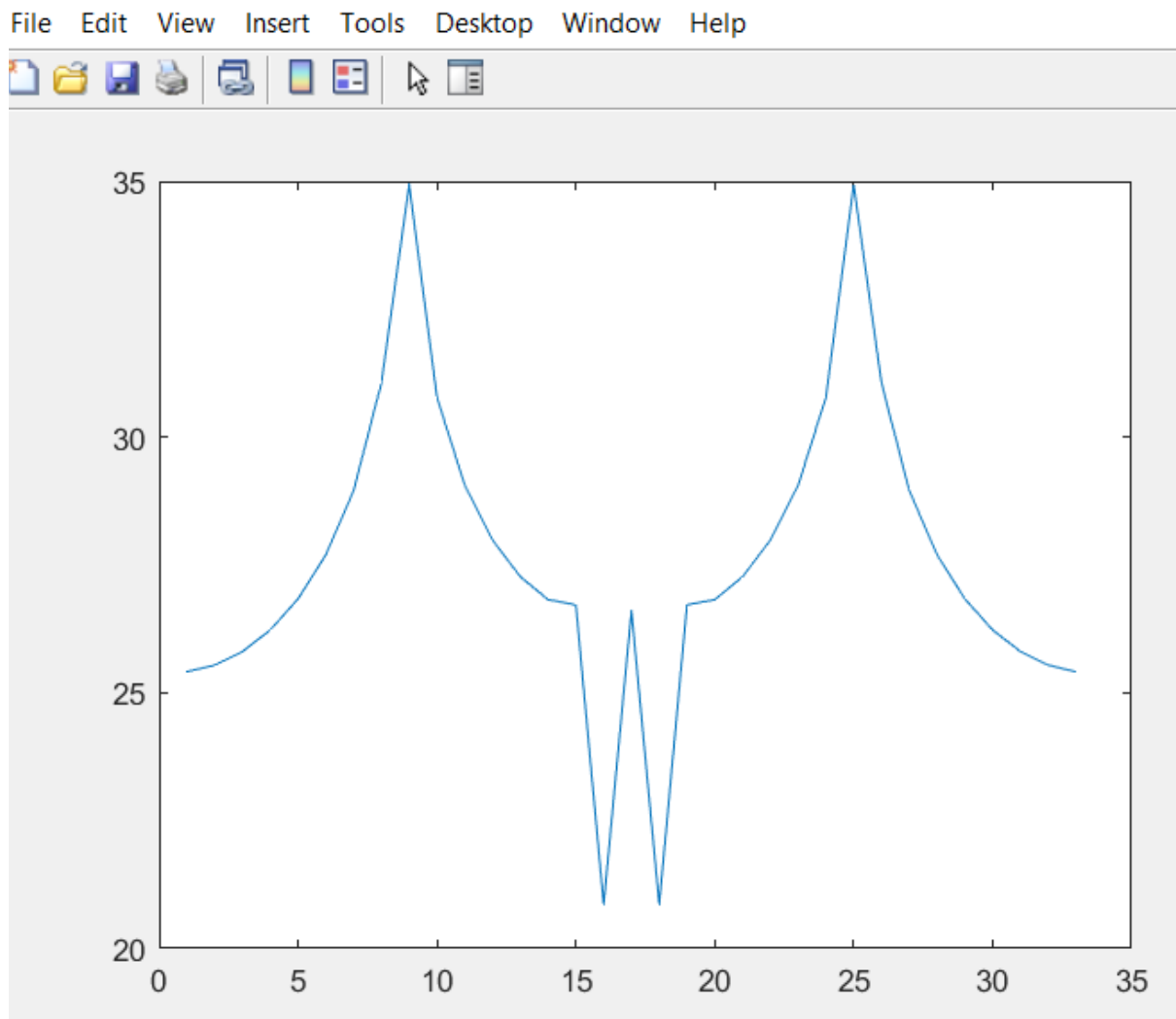
```
figure(7);
plot(20*log10(abs(Xk)));
```



```
L = 32;
%rectangular window
X = rectwin(L);
wrect = rectwin(L);
%apply window to signal
for n = 1:32
    yrect(n) = (80*dirac(n) - 80*sin(pi*n/2)/(pi*n) + cos(pi*n/32) +
cos(pi*n/16))*wrect(n);
end

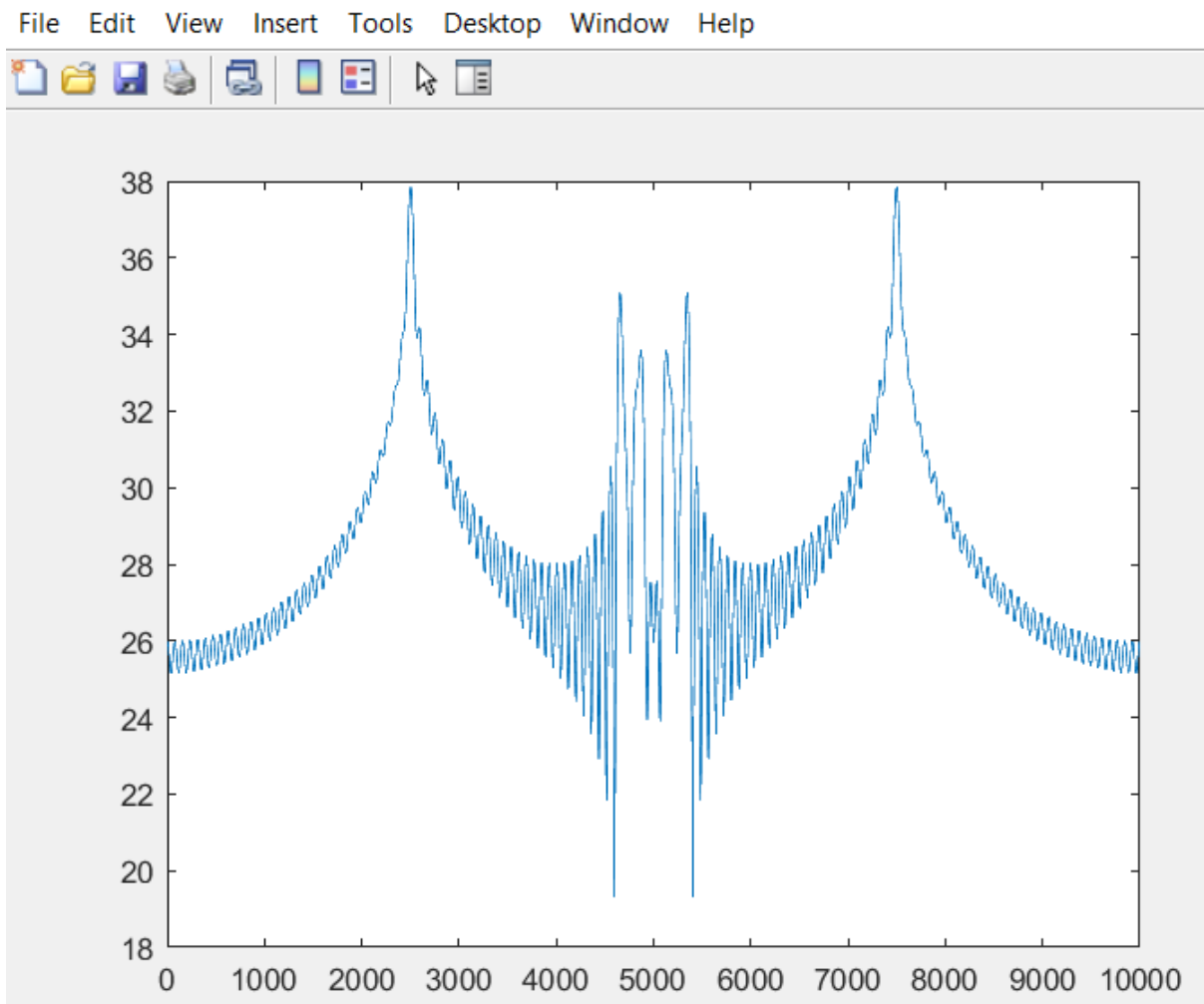
N = L+1;
Xk = fftshift(fft(yrect,N));
figure(8);
plot(20*log10(abs(Xk)));
```

Figure 8



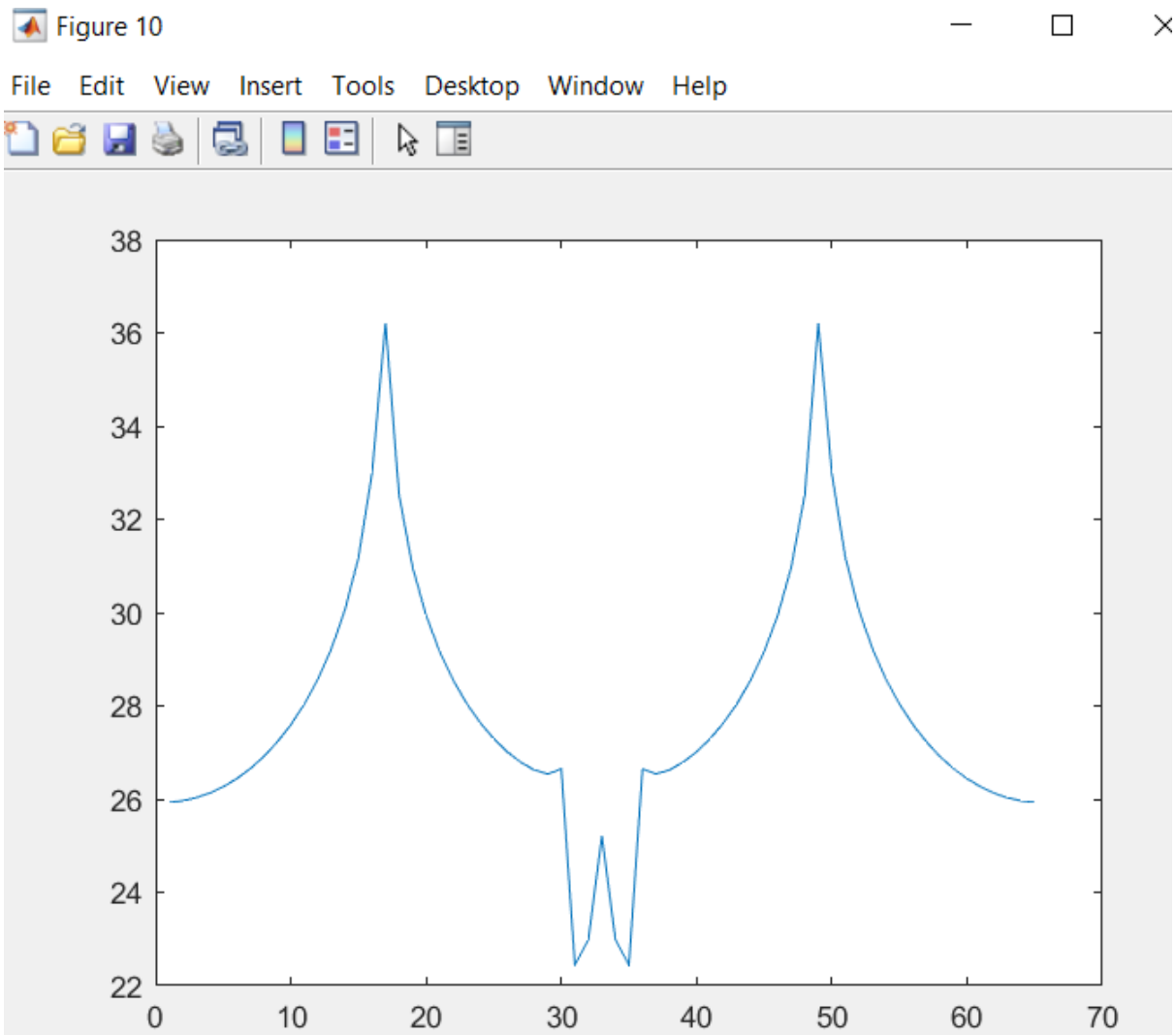
```
N = 10000;  
Xk = fftshift(fft(yrect,N));  
figure(9);  
plot(20*log10(abs(Xk)));
```

Figure 9

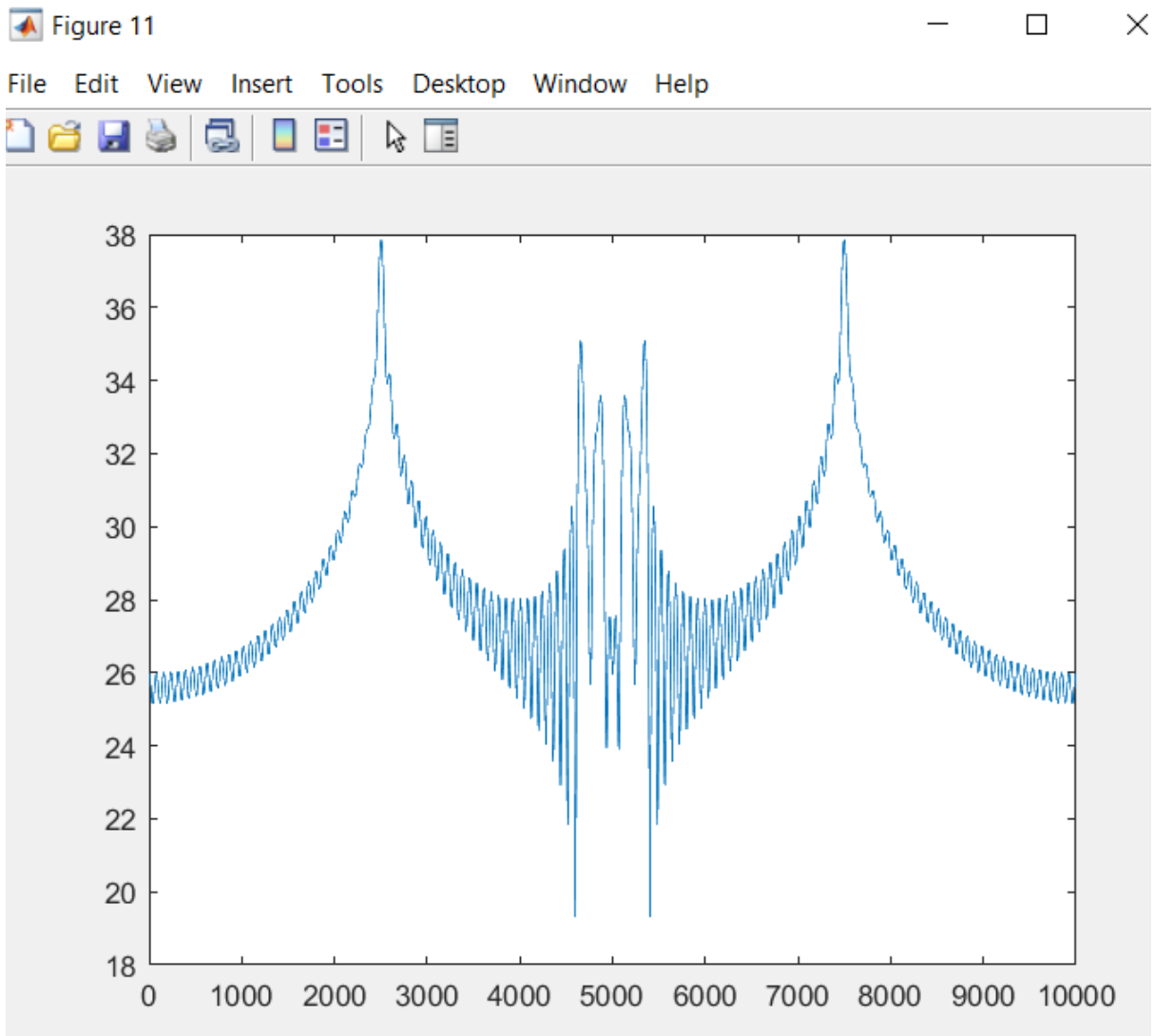


```
L = 64;
%rectangular window
X = rectwin(L);
wrect = rectwin(L);
%apply window to signal
for n = 1:64
    yrect(n) = (80*dirac(n) - 80*sin(pi*n/2)/(pi*n) + cos(pi*n/32) +
cos(pi*n/16))*wrect(n);
end

N = L+1;
Xk = fftshift(fft(yrect,N));
figure(10);
plot(20*log10(abs(Xk)));
```

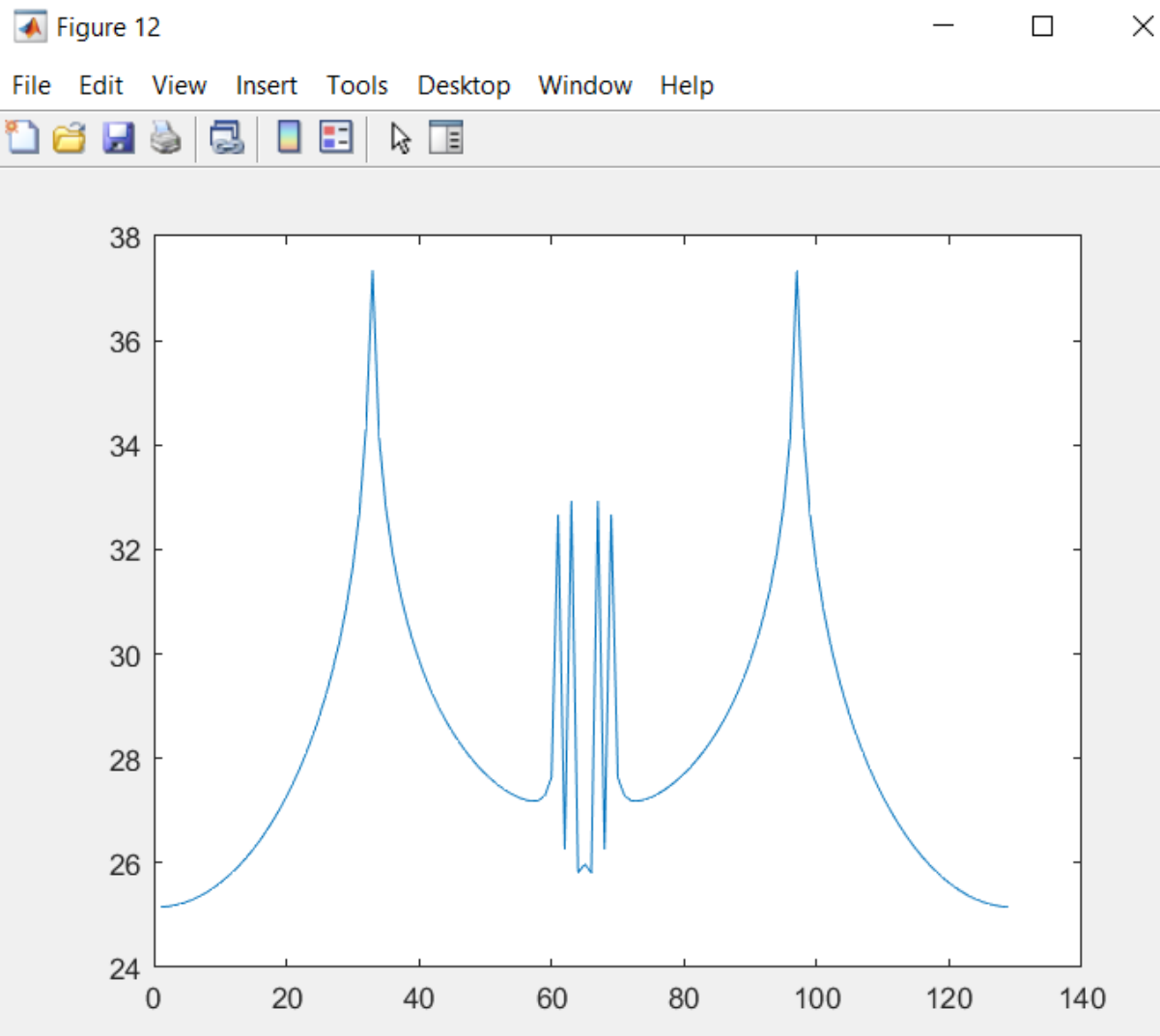


```
N = 10000;  
Xk = fftshift(fft(yrect,N));  
figure(11);  
plot(20*log10(abs(Xk)));  
%plot( 20*log10(abs(fftshift(fft(yrect, Nbins))))), 'r' );
```

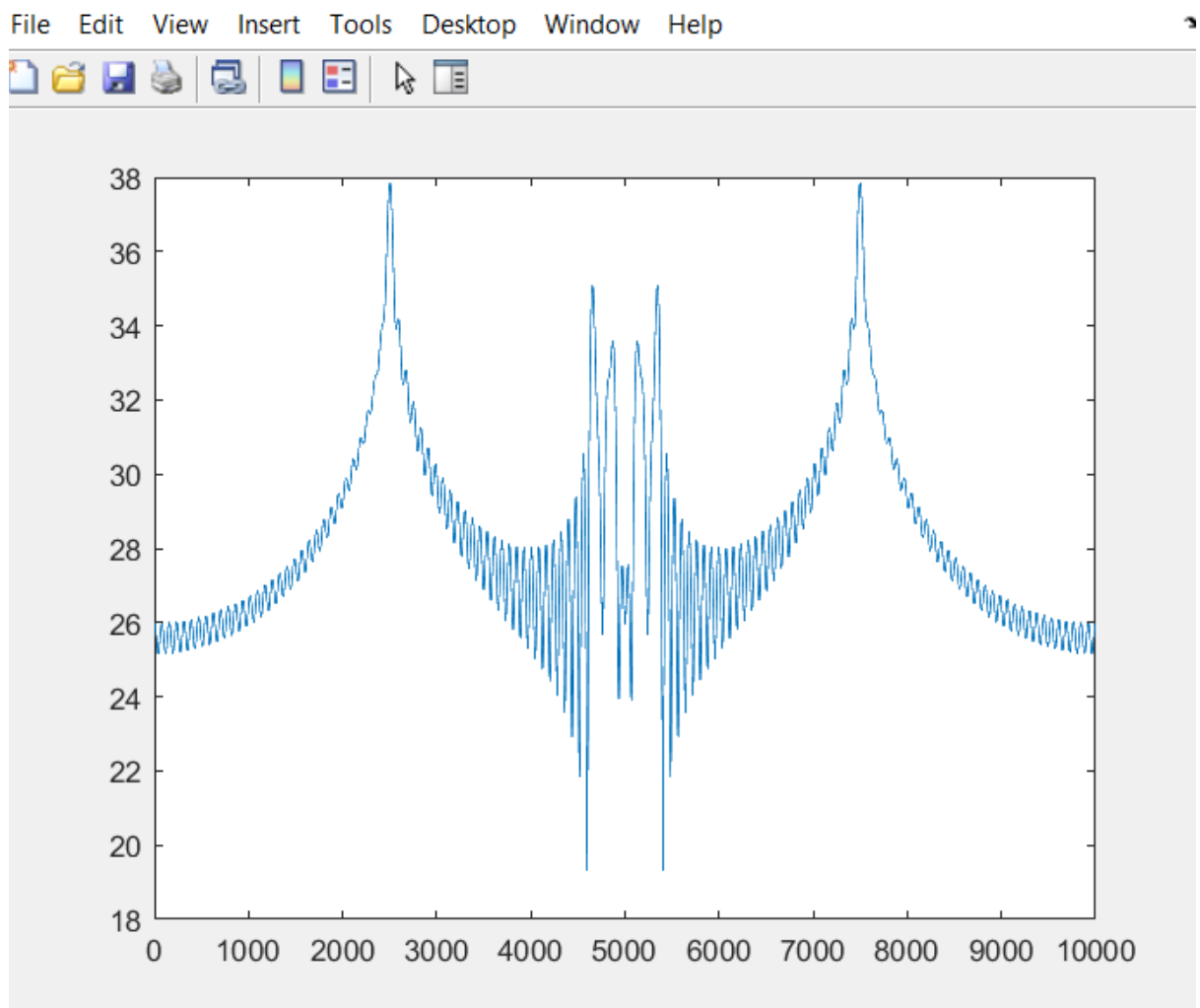
```
L = 128;
%rectangular window
X = rectwin(L);
wrect = rectwin(L);
%apply window to signal
for n = 1:128
    yrect(n) = (80*dirac(n) - 80*sin(pi*n/2)/(pi*n) + cos(pi*n/32) +
cos(pi*n/16))*wrect(n);
end

N = L+1;
Xk = fftshift(fft(yrect,N));
figure(12);
plot(20*log10(abs(Xk)));
```



```
N = 10000;  
Xk = fftshift(fft(yrect,N));  
figure(13);  
plot(20*log10(abs(Xk)));
```

Figure 13



B2

Αρχικά για την ηχογράφηση χρησιμοποίησα το μικρόφωνο του λάπτοπ μου και την εφαρμογή audacity μέσω της οποίας μπορώ να επιλέξω τα hz του δειγματος. Επέλεξα 16000 Hz (16KHz). Το αρχείο είναι stereo, 32 bit float.

→Χρησιμοποίησα audioread αντί για wavread καθώς το matlab δεν αναγνωρίζει τη συνάρτηση wavread.

→Παρατήρησα από το error της συνάρτησης spectrogram πως τα data από την audioread δεν ήταν διάνυσμα 1 στήλης αλλά 2. Αυτό οφείλεται στο ότι ηχογράφησα **stereo** ήχο που έχει 2 channels, επομένως ηχογράφησα το όνομά μου **mono** αυτή τη φορά (μέσω του audacity), οπότε η audioread επιστρέφει διάνυσμα που μπορώ να χρησιμοποιήσω στην spectrogram.

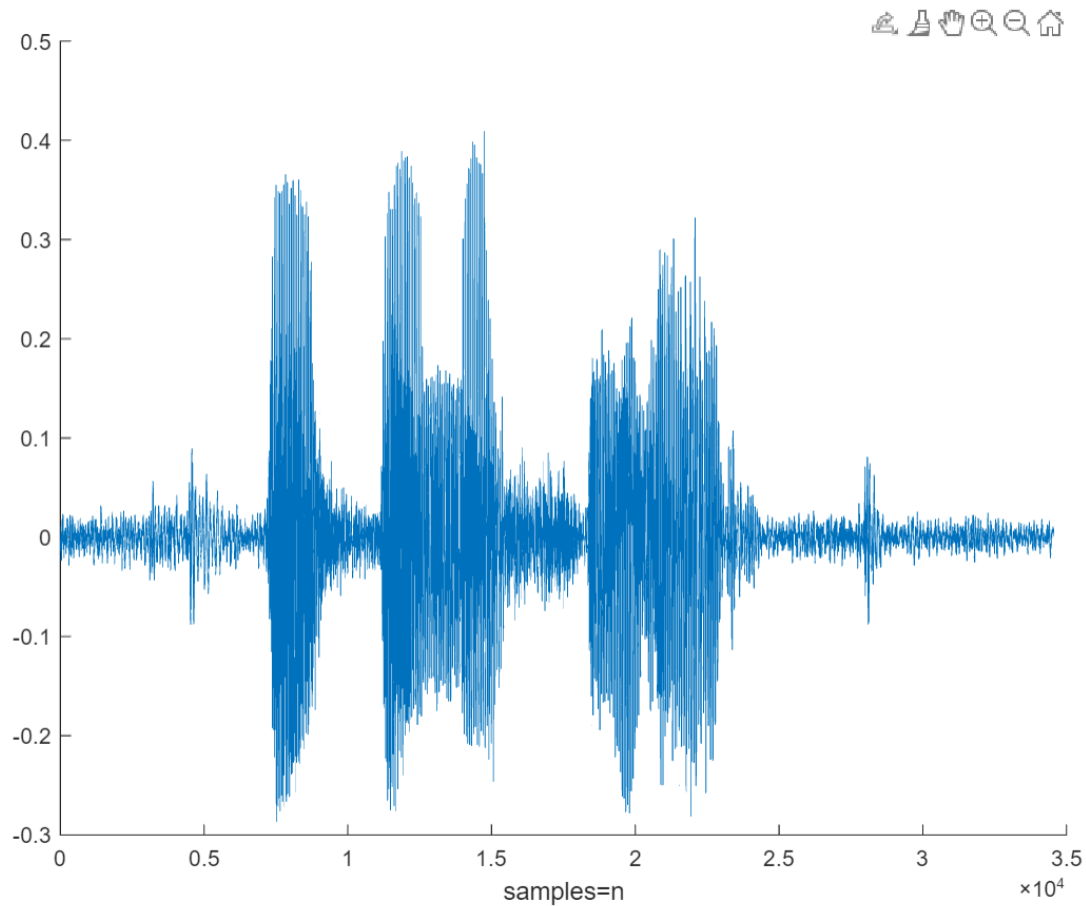
B2.1

Χρησιμοποίησα την imagesc για την απεικόνιση των data ως εικόνα που χρησιμοποιεί όλο το φάσμα των χρωμάτων στον χρωματικό χάρτη, με αποτέλεσμα να εμφανιστεί ένα καλό visualization των δεδομένων

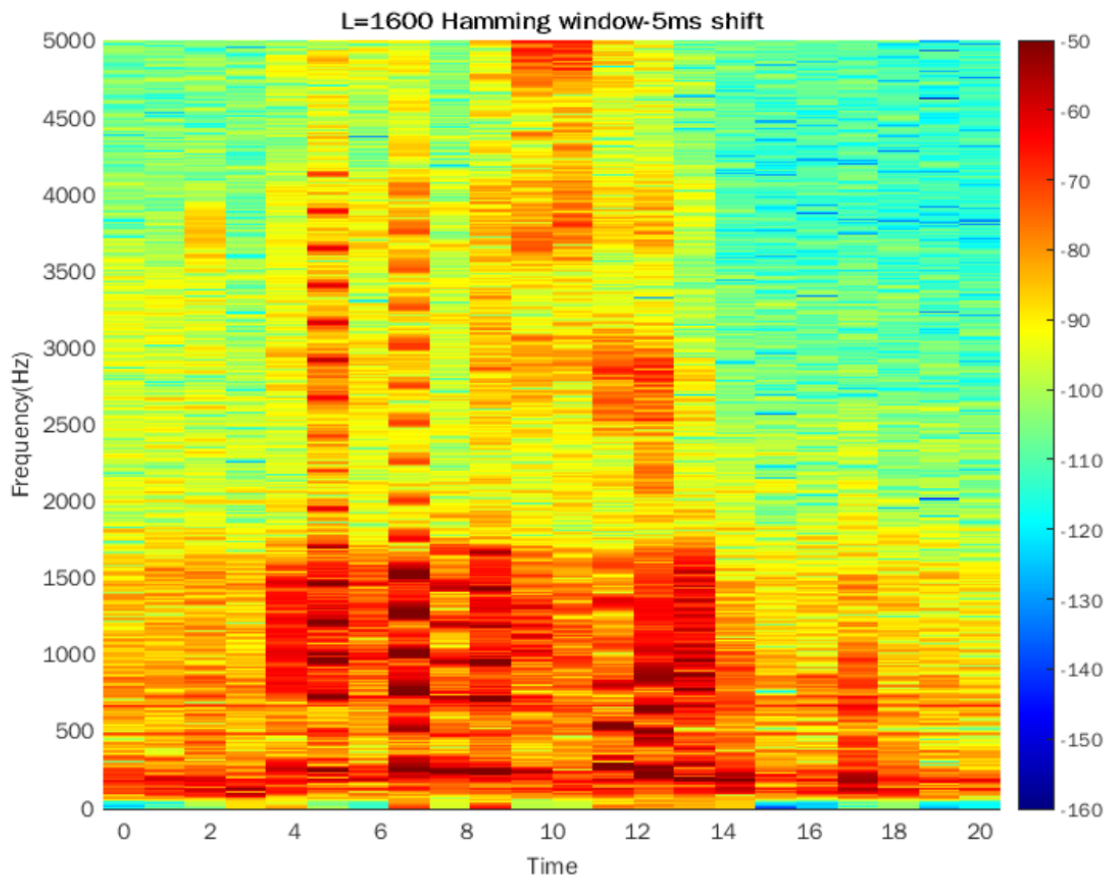
Κωδικας

```
%for sample = 16KHz
freqs = 0:10:5000;
% points per segment for 16KHz
window_1600 = hamming(1600);
window_160 = hamming(160);
% overlap with 80 points per segment (160/2=80)
%overlap is how many samples to include from the calculation of spectrum N-1
in spectrum N
overlap = 80;
%read the record
[name_sample, fsample] = audioread('name_mono.wav');
%fsample=16000 αφού η ηχογράφηση είναι στα 16KHz.
%check the name_sample (if it is a vector)
whos mySpeech

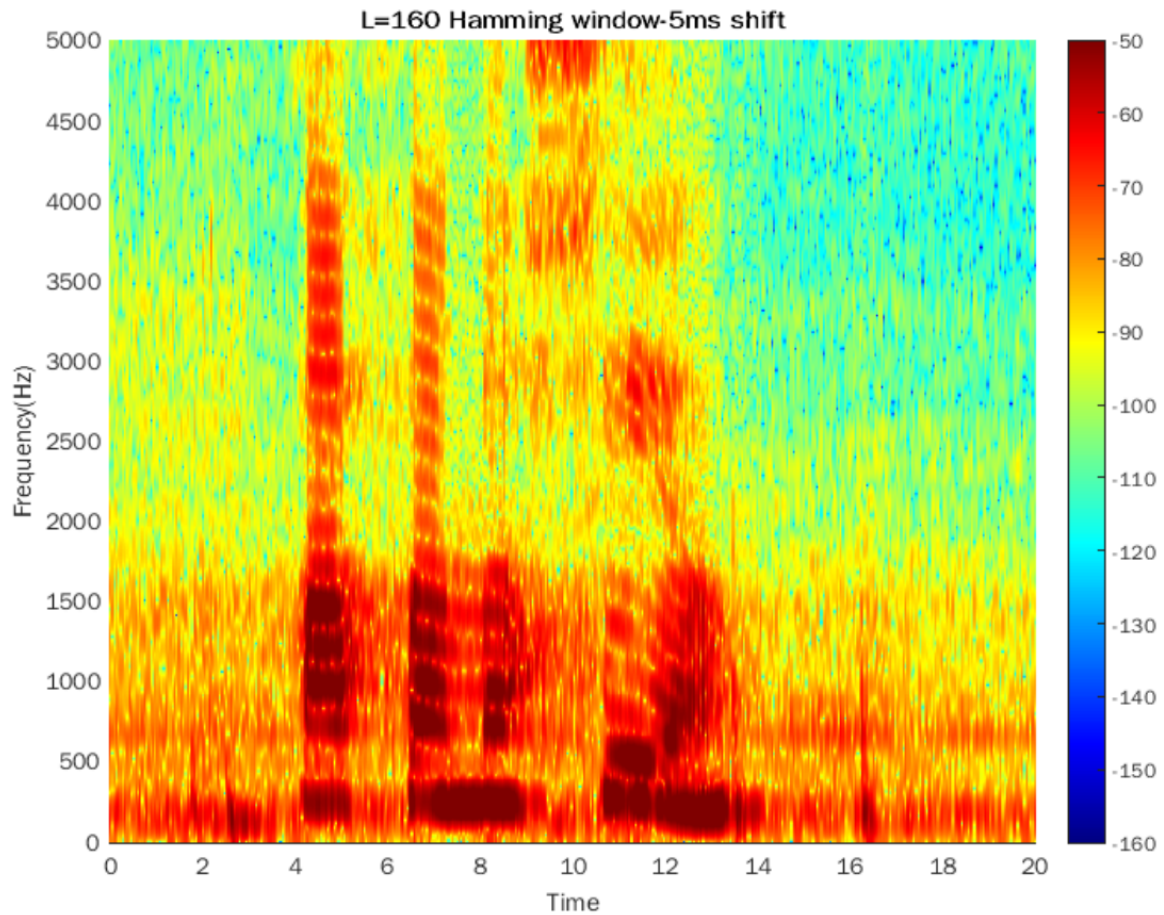
%plot the recording
figure(9);
hold on;
plot(name_sample);
xlabel('samples=n');
hold off;
```



```
% hamming window L=1600
figure(11);
hold on;
title('100ms Hamming window-5ms shift');
[S, F, T, P] = spectrogram(name_sample, window_1600, overlap, freqs, fsample,
'yaxis');
imagesc([0:0.1:20], F, 10*log10(abs(P)), [-160 -50] );
xlabel('Time');
ylabel('Frequency(Hz)');
title('L=1600');
axis xy; axis tight; colormap(jet); view(0,90);
colorbar();
hold off;
```



```
% hamming window L=160
figure (12);
hold on;
title('10ms Hamming window-5ms shift');
[S, F, T, P] = spectrogram(name_sample, window_160, overlap, freqs, fsample,
'yaxis');
imagesc([0:0.1:20], F, 10*log10(abs(P)), [-160 -50] );
xlabel('Time');
ylabel('Frequency(Hz) ');
title('L=160');
axis xy; axis tight; colormap(jet); view(0,90);
colorbar();
hold off;
```

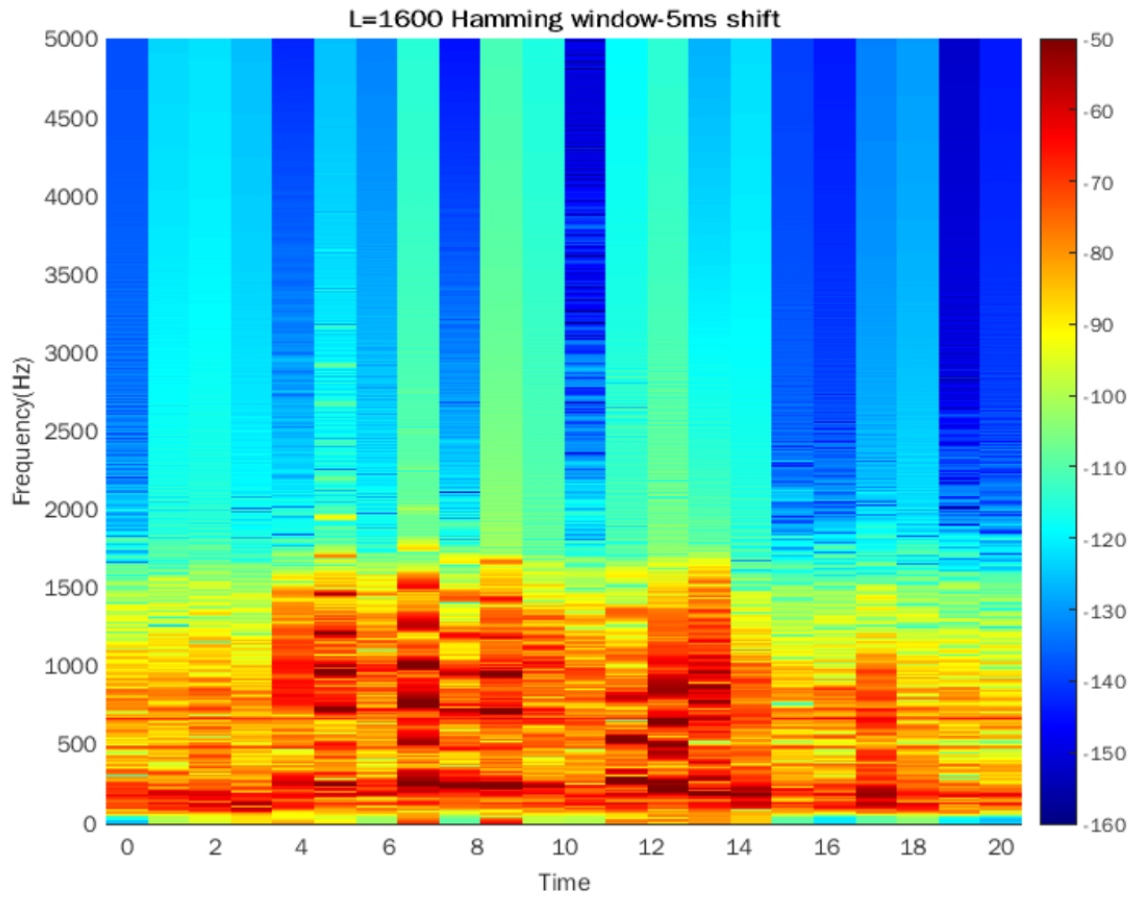


B2.2

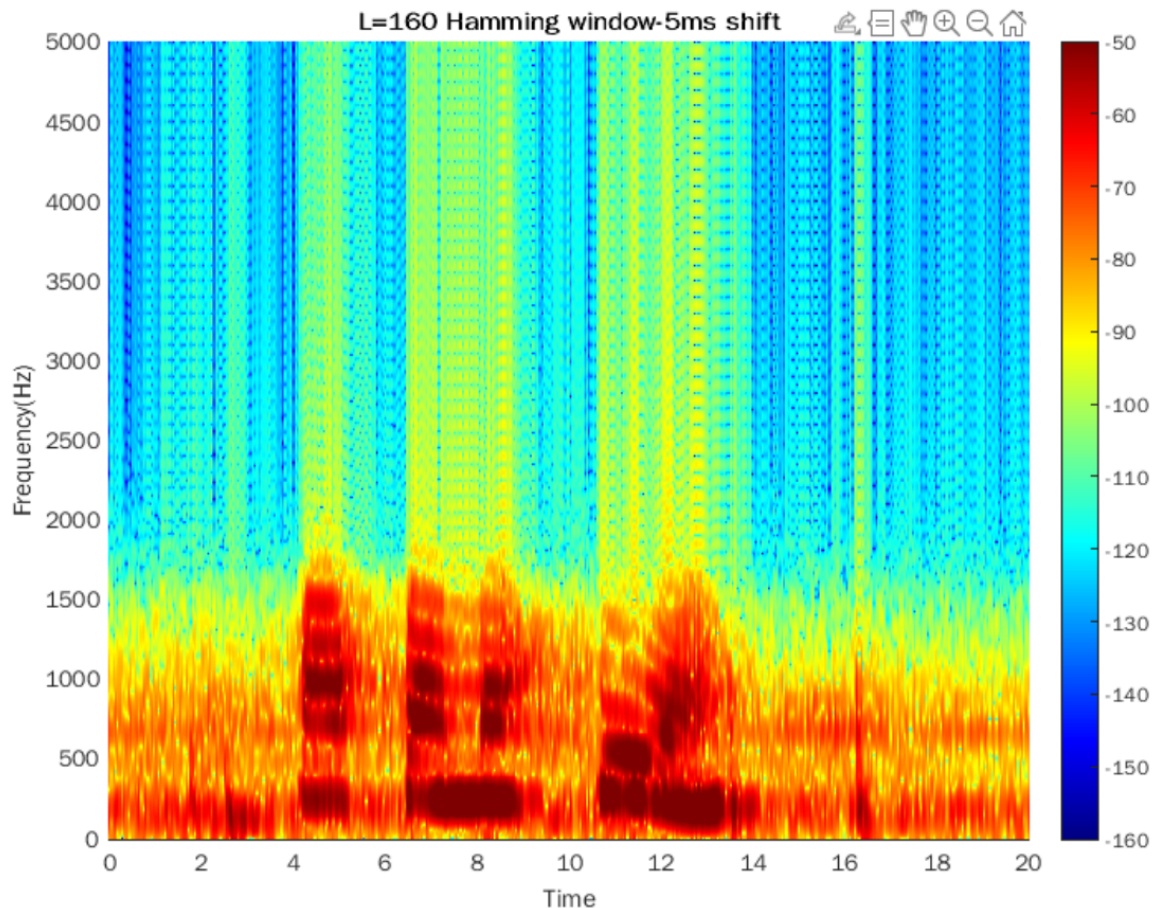
```
%apply filter from part A
filtered_record = filter(num, den, name_sample);

audiowrite('filtered_name_16KHz.wav', filtered_record, fsample);

%Hamming window L=1600
figure (13);
hold on;
[S, F, T, P] = spectrogram(filtered_record, window_1600, overlap, freqs,
fsample, 'yaxis');
imagesc([0:0.1:20], F, 10*log10(abs(P)), [-160 -50] );
xlabel('Time');
ylabel('Frequency(Hz)');
title('L=1600 Hamming window-5ms shift');
axis xy; axis tight; colormap(jet); view(0,90);
colorbar();
hold off;
```



```
%Hamming window L=160
figure (14);
hold on;
[S, F, T, P] = spectrogram(filtered_record, window_160, overlap, freqs,
fsample, 'yaxis');
imagesc([0:0.1:20], F, 10*log10(abs(P)), [-160 -50] );
xlabel('Time');
ylabel('Frequency(Hz)');
title('L=160 Hamming window-5ms shift');
axis xy; axis tight; colormap(jet); view(0,90);
colorbar();
hold off;
```

Συμπέρασμα

→ Από τα παραπάνω figures παρατηρώ ότι μέσω του κατωπερατού φίλτρου, κόβονται οι υψηλές συχνότητες (όπως ήταν αναμενόμενο). Επίσης, μειώνεται το πλάτος του dft, με αποτέλεσμα να μειωθεί η ένταση.

→ Η μείωση αυτή συμβαίνει εξαιτίας του M/S fourier. Πιο συγκεκριμένα αν εφαρμοστεί κατωπερατό φίλτρο N βαθμού, όσο $N \uparrow \rightarrow$ πλάτος \downarrow

Παραρτημα (οι κώδικες χωρίς πλοτς ανάμεσα)

PART A AND B2

```
%Sapountzi Athanasia Despoina 2624
%Part A and Part B2
%% A1. FIR Filter
%methodology for construction of FIR filter
fsample = 16000;
As = 40;
Rp = 1;
fs = 0.15;
fp = 0.05;
ws=0.3*pi;
wp=0.1*pi;
%minimum filter length M from hamming's equation:
```

```

N = ceil( 4/(fs-fp) );
%discrete time fir-hamming-lowpass coefficients
coefficients = fir1(N, (wp+ws)/2, 'low', hamming(N+1));
%% A2. IIR Filter
%methodology for construction of IIR filter Butterworth via bilinear transform
%First convert to analog
WP = 2*fsample*tan(wp/2);
WS = 2*fsample*tan(ws/2);
%Butterworth
[N,Wn] = buttord(WP, WS, Rp, As, 's');
[z, p] = butter(N, Wn, 'low', 's');
%bilinear transform
[num, den] = bilinear(z, p, fsample);
%% Impulse response
hold on;
figure (1);
title('Impulse Response');
grid('ON');
ylabel('Amplitude');
xlabel('Samples');
%for FIR filter
[firY, firX] = impz(coefficients);
stem(firX, firY, 'm', 'filled');
hold on;
%for IIR filter
[iirY, IIR_x] = impz(num, den);
stem(IIR_x, iirY, 'b', 'filled');
hold on;
legend('FIR', 'IIR');
hold off;
%% Step response
figure (2);
hold on;
title('Step Response');
grid('ON');
xlabel('Samples');
ylabel('Amplitude');
%for FIR filter Step response
[FIR_y, FIR_x] = stepz(coefficients);
stem(FIR_x, FIR_y, 'm', 'filled');
hold on;
%for IIR filter Step response
[IIR_y, IIR_x] = stepz(num, den);
stem(IIR_x, IIR_y, 'b', 'filled');
hold on;
legend('FIR', 'IIR');
hold off;
%% Frequency response  $H(e^{j\omega})$ 
%frequency domain FIR
[FIR_h, FIR_w]=freqz(coefficients,1,256);
%amplitude-gain FIR

```

```

FIR_h_inDB = 20*log10(abs(FIR_h));
%frequency domain IIR
[IIR_h,IIR_w]=freqz(num,den,256);
%amplitude-gain IIR
IIR_h_inDB = 20*log10(abs(IIR_h));
figure(3);
hold on;
title('Frequency Response');
grid('ON');
ylabel('Gain');
xlabel('Normalised frequency');
%normalized frequency(Nyquist) FIR
plot(FIR_w/(pi), FIR_h_inDB, 'm');
hold on;
%Normalized frequency(Nyquist) IIR
plot(IIR_w/(pi), IIR_h_inDB, 'b');
legend('FIR', 'IIR');
hold off;
%% Group delay
%  $-\frac{d}{dw}\{-\tan^{-1}(\frac{\text{Im}(H(e^{jw}))}{\text{Re}(H(e^{jw}))})\}$ 
%by default there are 8192 samples
[FIR_gd, FIR_w] = grpdelay(coefficients);
[IIR_gd, IIR_w] = grpdelay(num, den);
figure(4);
hold on;
title('Group delay');
grid('ON');
xlabel('Normalised frequency');
plot(FIR_w/pi,FIR_gd, 'm');
hold on;
plot(IIR_w/pi, IIR_gd, 'b');
legend('FIR', 'IIR');
hold off;
%% Zeroes/Poles
figure(5);
%only for IIR filter this time, because FIR does not have poles
[b, a] = zplane(z, p);
hold on;
title('Zeros/Poles');
hold off;
%% B2.1
%for sample = 16KHz
freqs = 0:10:5000;
% points per segment for 16KHz
window_1600 = hamming(1600);
window_160 = hamming(160);
% overlap with 80 points per segment (160/2=80)
%overlap is how many samples to include from the calculation of spectrum N-1
in spectrum N
overlap = 80;
%read the record

```

```

[name_sample, fsample] = audioread('name_mono.wav');
%fsample=16000 afou h hxografhsh einai sta 16KHz.
%check the name_sample (if it is a vector)
whos mySpeech
%plot the recording
figure(10);
hold on;
plot(name_sample);
xlabel('samples=n');
hold off;
% hamming window L=1600
figure(11);
hold on;

[S, F, T, P] = spectrogram(name_sample, window_1600, overlap, freqs, fsample,
'yaxis');
imagesc([0:0.1:20], F, 10*log10(abs(P)), [-160 -50] );
xlabel('Time');
ylabel('Frequency(Hz)');
title('L=1600 Hamming window-5ms shift');
axis xy; axis tight; colormap(jet); view(0,90);
colorbar();
hold off;
% hamming window L=160
figure (12);
hold on;
[S, F, T, P] = spectrogram(name_sample, window_160, overlap, freqs, fsample,
'yaxis');
imagesc([0:0.1:20], F, 10*log10(abs(P)), [-160 -50] );
xlabel('Time');
ylabel('Frequency(Hz)');
title('L=160 Hamming window-5ms shift');
axis xy; axis tight; colormap(jet); view(0,90);
colorbar();
hold off;
%% B2.2 Sound spectrograms with lowpass(IIR)
%apply filter from part A
filtered_record = filter(num, den, name_sample);
audiowrite('filtered_name_16KHz.wav', filtered_record, fsample);
%Hamming window L=1600
figure (13);
hold on;
[S, F, T, P] = spectrogram(filtered_record, window_1600, overlap, freqs,
fsample, 'yaxis');
imagesc([0:0.1:20], F, 10*log10(abs(P)), [-160 -50] );
xlabel('Time');
ylabel('Frequency(Hz)');
title('L=1600 Hamming window-5ms shift');
axis xy; axis tight; colormap(jet); view(0,90);
colorbar();
hold off;

```

```

%Hamming window L=160
figure (14);
hold on;
[S, F, T, P] = spectrogram(filtered_record, window_160, overlap, freqs,
fsample, 'yaxis');
imagesc([0:0.1:20], F, 10*log10(abs(P)), [-160 -50] );
xlabel('Time');
ylabel('Frequency(Hz)');
title('L=160 Hamming window-5ms shift');
axis xy; axis tight; colormap(jet); view(0,90);
colorbar();
hold off;

```

PART B1

```

%Author: Sapountzi Athanasia Despoina 02624
%%MEROS B1
%% B1.1
%signal  $x[n] = 80\delta[n] - 80\sin(\pi n/2)/(\pi n) + \cos(\pi n/32) + \cos(\pi n/16)$ 
%delta -> dirac function
N = 10000;
n= 0:N-1;
Nbig = 10000;
Nbins = 8000;
signal = 80*dirac(n) - 80*sin(pi*n/2)/(pi*n) + cos(pi*n/32) + cos(pi*n/16);
L =8;
%rectangular window
wrect = rectwin(L);
%apply windowing technique to signal
for n = 1:8
    yrect(n) = (80*dirac(n) - 80*sin(pi*n/2)/(pi*n) + cos(pi*n/32) +
cos(pi*n/16))*wrect(n);
end
figure(1);
plot(yrect);
%% B1.2
N = L+1;
Xk = fftshift(fft(yrect,N));
figure(3);
plot(20*log10(abs(Xk)));
N = 10000;
Xk = fftshift(fft(yrect,N));
figure(4);
plot(20*log10(abs(Xk)));
%% B1.3
L =16;
%rectangular window
wrect = rectwin(L);
%apply window to signal
for n = 1:16
    yrect(n) = (80*dirac(n) - 80*sin(pi*n/2)/(pi*n) + cos(pi*n/32) +
cos(pi*n/16))*wrect(n);

```

```

end
%plot( 20*log10(abs(fftshift(fft(yrect, Nbins)))), 'm' );
N = L+1;
Xk = fftshift(fft(yrect,N));
figure(6);
plot(20*log10(abs(Xk)));
N = 10000;
Xk = fftshift(fft(yrect,N));
figure(7);
plot(20*log10(abs(Xk)));
L = 32;
%rectangular window
X = rectwin(L);
wrect = rectwin(L);
%apply window to signal
for n = 1:32
    yrect(n) = (80*dirac(n) - 80*sin(pi*n/2)/(pi*n) + cos(pi*n/32) +
cos(pi*n/16))*wrect(n);
end
N = L+1;
Xk = fftshift(fft(yrect,N));
figure(8);
plot(20*log10(abs(Xk)));
N = 10000;
Xk = fftshift(fft(yrect,N));
figure(9);
plot(20*log10(abs(Xk)));
L = 64;
%rectangular window
X = rectwin(L);
wrect = rectwin(L);
%apply window to signal
for n = 1:64
    yrect(n) = (80*dirac(n) - 80*sin(pi*n/2)/(pi*n) + cos(pi*n/32) +
cos(pi*n/16))*wrect(n);
end
N = L+1;
Xk = fftshift(fft(yrect,N));
figure(10);
plot(20*log10(abs(Xk)));
N = 10000;
Xk = fftshift(fft(yrect,N));
figure(11);
plot(20*log10(abs(Xk)));
L = 128;
%rectangular window
X = rectwin(L);
wrect = rectwin(L);
%apply window to signal
for n = 1:128

```

```

        yrect(n) = (80*dirac(n) - 80*sin(pi*n/2)/(pi*n) + cos(pi*n/32) +
cos(pi*n/16))*wrect(n);
end
N = L+1;
Xk = fftshift(fft(yrect,N));
figure(12);
plot(20*log10(abs(Xk)));
N = 10000;
Xk = fftshift(fft(yrect,N));
figure(13);
plot(20*log10(abs(Xk)));

```