ΙΩΆΝΝΗΣ ΠΟΛΥΧΡΟΝΌΠΟΥΛΟΣ ΑΜ:03121089

Άσκηση 1: Ανάλυση Σήματος Φωνής στα Πεδία Χρόνου και Συχνότητας

Α) Αρχικά, δημιούργησα ένα ξεχωριστό αρχείο *Voice_recorder.m* με κώδικα:

```
-----------------------------------------------------------------------
clc
clear all
close all
warning off
Frequency = 8000; % Sampling frequency in hertz
ch = 1; % Number of channels -- 2 options -- 1 (mono) or 2 (stereo)
datatype = 'uint8';
number_of_bits = 16; %8,16,or 24
duration = 2; % 2 seconds duration
recorder = audiorecorder(Frequency ,number_of_bits ,ch);
disp('Start speaking...')
% Record audio to audiorecorder object,...
...hold control until recording completes.
recordblocking(recorder, duration);
disp('End of Recording...');
% Store recorded audio signal in numeric array
audiodata = getaudiodata(recorder, datatype);
% Write audio file
audiowrite('Ioannis Polychronopoulos.wav', audiodata, Frequency);
-----------------------------------------------------------------
```

με το οποίο ηχογράφησα τον εαυτό μου με ρυθμό δειγματοληψίας $8000Hz$.

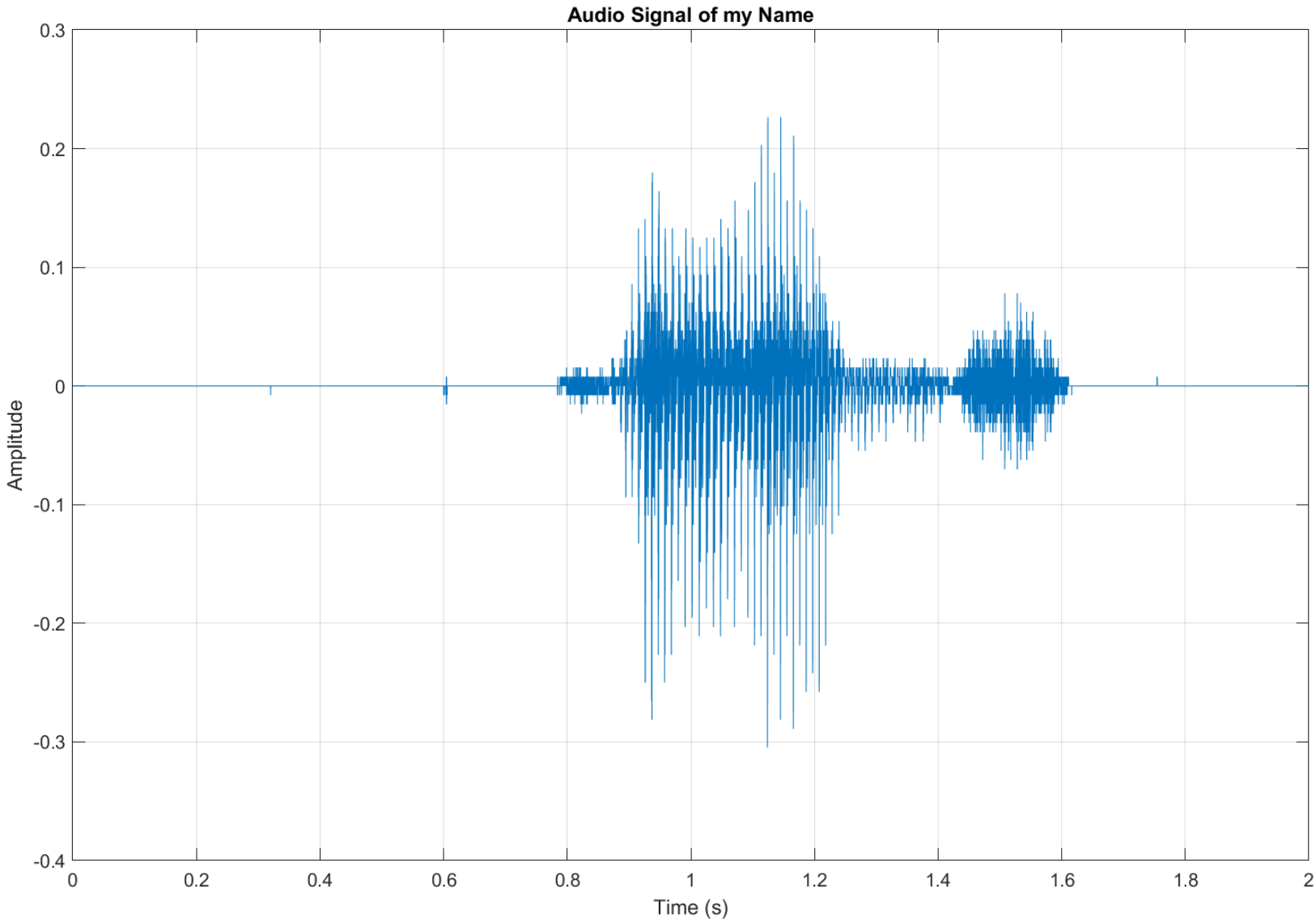
Έπειτα, αξιοποιώντας την εντολή $\boldsymbol{audioread}()$ έσωσα το αρχείου ήχου και την συχνότητα δειγματοληψίας ως εξής:

```
% Reading and saving the audio file to the vector audio
[audio,fs_audio] = audioread('Ioannis Polychronopoulos.wav');
```

Β) Για να σχεδιάσουμε το αρχείο ήχου που μόλις καταγράψαμε και έχουμε αποθηκευμένο στη μεταβλητή `audio` χρησιμοποιούμε την εντολή $\boldsymbol{plot}()$**.**

```
-----------------------------------------------------------------------
% Initializing a vector which is going to represent time from 0-2 seconds
time = linspace(0,2,length(audio));

% Plotting the audio signal
figure
plot(time,audio);
grid on
xlabel('Time (s)');
ylabel('Amplitude');
title('Audio Signal of my Name');

-----------------------------------------------------------------
```

Το γράφημα του αρχείου ήχου φαίνεται στη παρακάτω εικόνα.

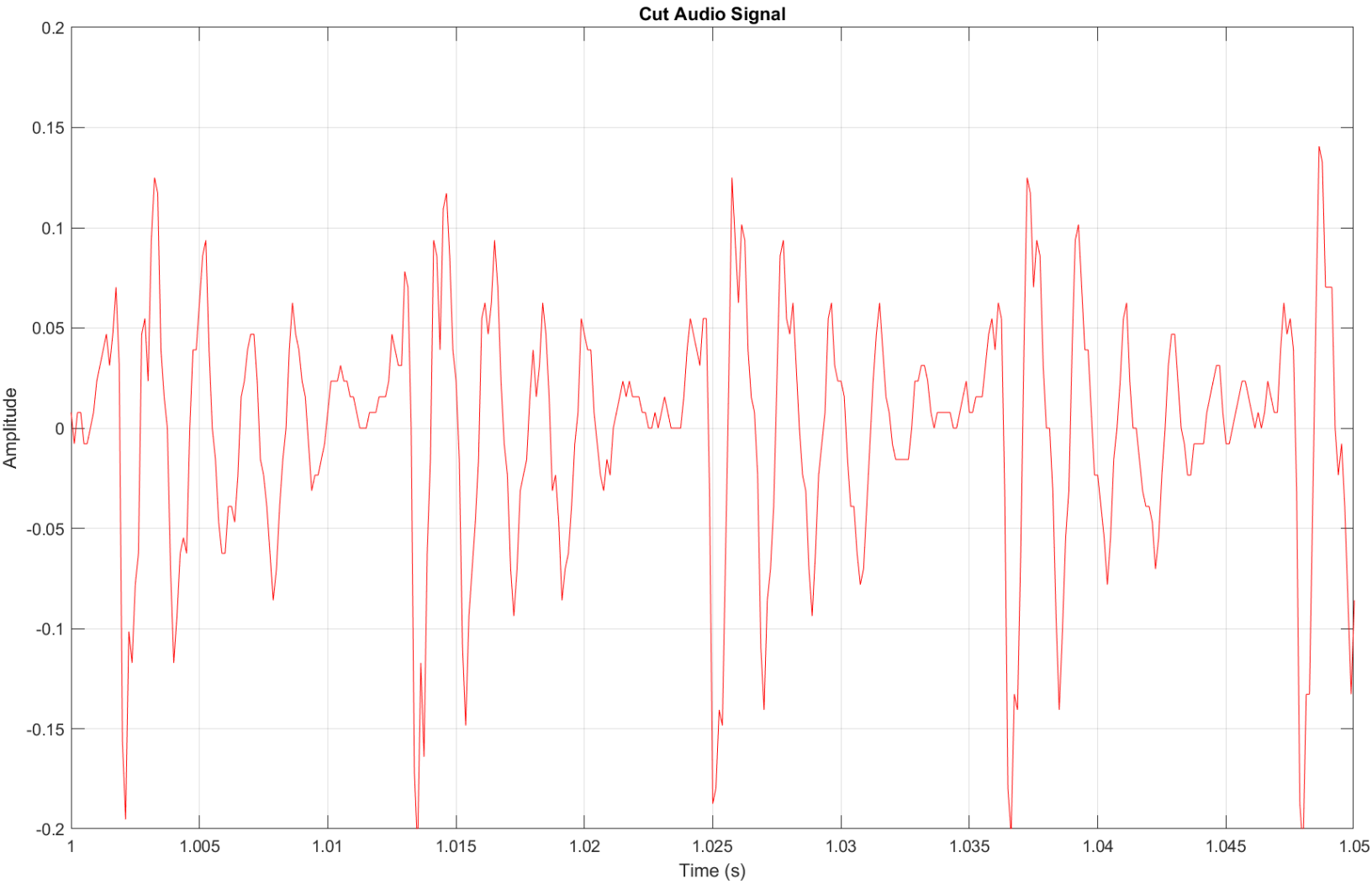


Στη συνέχεια, διαλέγω το ακόλουθο παράθυρο $50\mu s \rightarrow [1,1.05]$ και θέτω στο vector `cut_signal` τις τιμές της μεταβλητής `audio` που αντιστοιχούν στο παραπάνω χρονικό διάστημα.

```
-------------------------------------------------------------------------

time_start = 1;
time_finish = 1.05;
start = fs_audio*time_start;
finish = fs_audio*time_finish;
cut_signal = audio(start:finish);
-------------------------------------------------------------------------
```

Επιβεβαιώνω πως στο παράθυρο που επέλεξα το σήμα είναι περιοδικό κάνοντας την γραφική παράσταση του κομμένου σήματος.

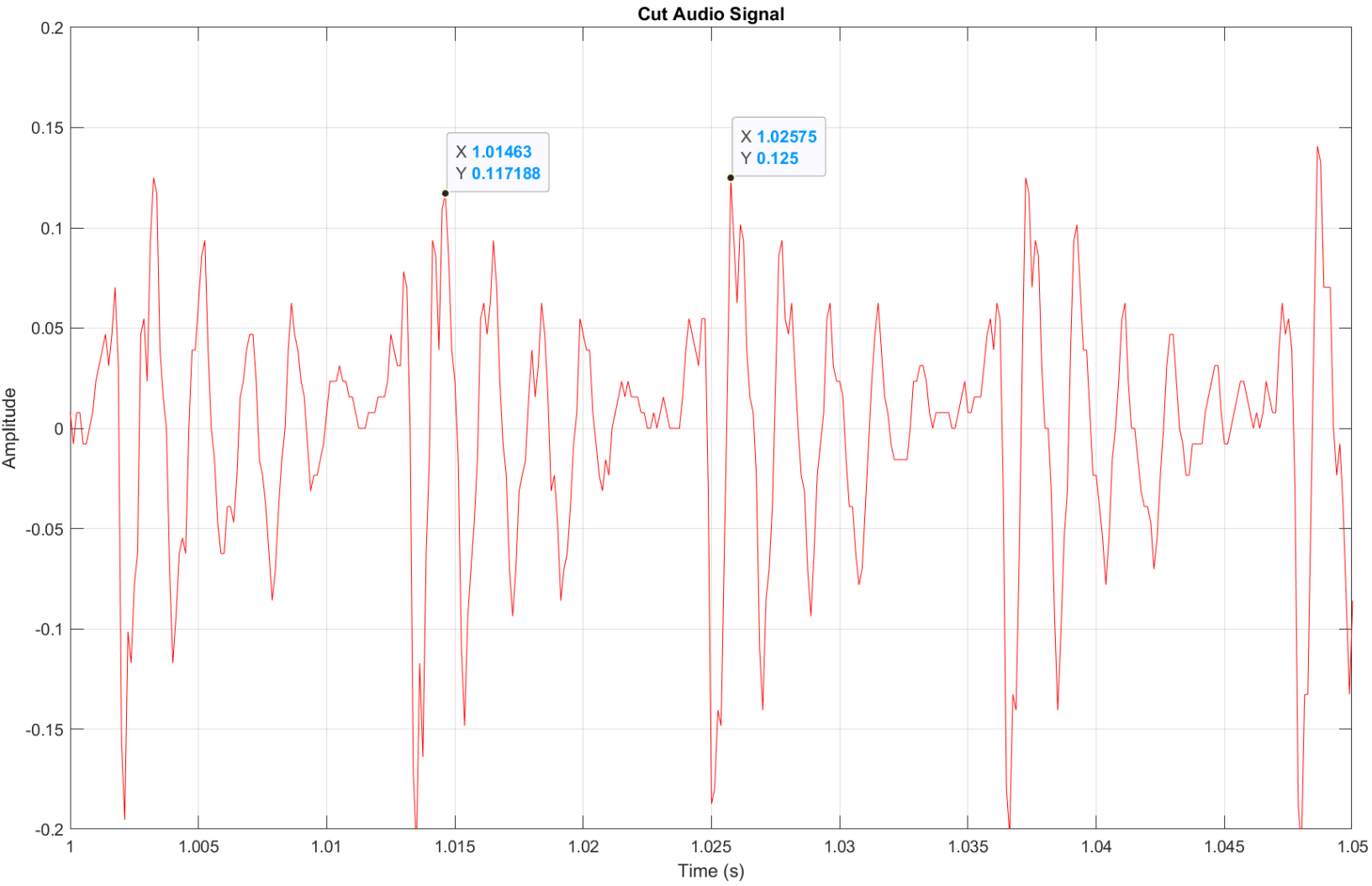
```
-------------------------------------------------------------------------
% Time interval for cut signal
interval = linspace(time_start,time_finish,length(cut_signal));

% Plotting the cut version of our signal
figure
plot(interval,cut_signal,'-r');
grid on
axis([time_start time_finish -0.2 0.2]);
xlabel('Time (s)');
ylabel('Amplitude');
title('Cut Audio Signal');
-------------------------------------------------------------------------
```



Η απόσταση μεταξύ δύο κορυφών στο παραπάνω γράφημα είναι

$$\Delta t = 0.01112 sec = 11.12\, msec = T$$

Που αντιστοιχεί με την περίοδο του αποσπάσματος.



Τέλος, ακούμε με την εντολή ***sound***() πως το τμήμα αυτό αντιστοιχεί στο φώνημα ‘ω’ από το ‘Ιωάννης’.

```
-------------------------------------------------------------------------
% Listening to the cut audio signal

sound(cut_signal,fs_audio);

-------------------------------------------------------------------------
```

Γ) Αφού κανονικοποιήσουμε το αρχικό σήμα στο διάστημα $[-1,1]$ με την συνάρτηση $\boldsymbol{normalize}()$ υπολογίζουμε την ενέργεια του επικαλυπτόμενο από το παράθυρο Hamming.

```
-------------------------------------------------------------------------
% Normalizing the audio signal

normalized_signal = normalize(audio,'range',[-1 1]);
-------------------------------------------------------------------------
```

Το παράθυρο Hamming με παράμετρο $M = 200$ γράφεται στο MATLAB ως εξής: $\boldsymbol{hamming(200)}$.

Τώρα για τον υπολογισμό της ενέργειας:

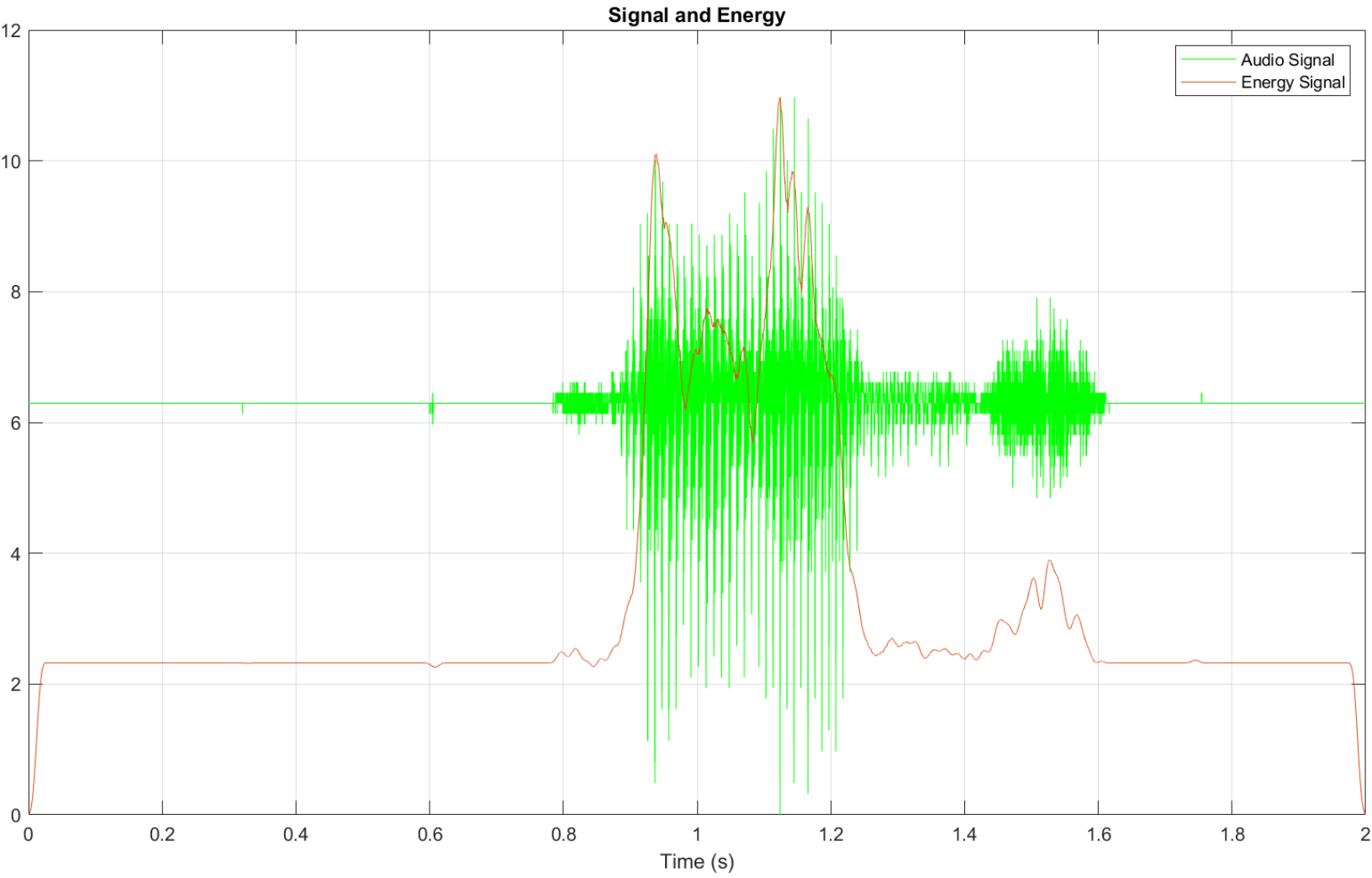
$$E[n] = \sum_{m=1}^{M} x^2[m]\, w[n-m] = x^2[n] * w[n], \qquad με\ w[n] = hamming(200)$$

Την συνέλιξη αυτή μπορούμε να την υπολογίσουμε με την εντολή $\boldsymbol{conv}()$.

```
-------------------------------------------------------------------------
% Finding the energy of the signal with convolution
energy = conv(normalized_signal.^2,hamming(200));
-------------------------------------------------------------------------
```

Κατόπιν, κάνουμε την γραφική παράσταση του αρχικού σήματος κανονικοποιημένο στο `[min(energy),max(energy)]` και την ενέργεια του σήματος στο ίδιο παράθυρο (figure).

```
-------------------------------------------------------------------------
% Plotting the energy of the signal and the signal itself
figure

% Normalizing the audio signal to change the scales properly
normalized_signal = normalize(audio,'range',[min(energy),max(energy)]);
plot(time,normalized_signal,'-g');
grid on
hold on
energy_time = linspace(0,2,length(energy));
plot(energy_time, energy);
hold off
legend('Audio Signal','Energy Signal');
xlabel('Time (s)');
title('Signal and Energy');
-------------------------------------------------------------------------
```



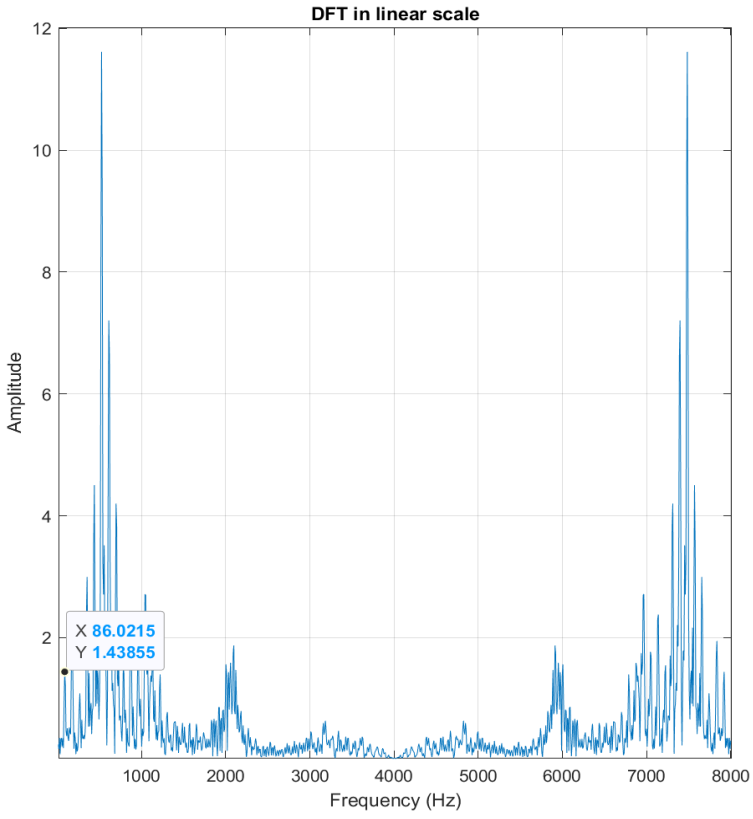
Παρατηρήσεις: Ο υπολογισμός της ενέργειας του σήματος επικαλυπτόμενο από παράθυρο hamming ανάγεται σε υπολογισμό συνέλιξης πράξη που εξομαλύνει το σήμα μας από τις απότομες διακυμάνσεις που περιείχε. Παράλληλα, παρατηρούμε στο σχήμα πως η ενέργεια είναι θετική, καθώς έχουμε συνέλιξη δύο θετικών συναρτήσεων, και πως αυξάνεται στις περιοχές όπου αυξάνεται και η ένταση του σήματός μας.

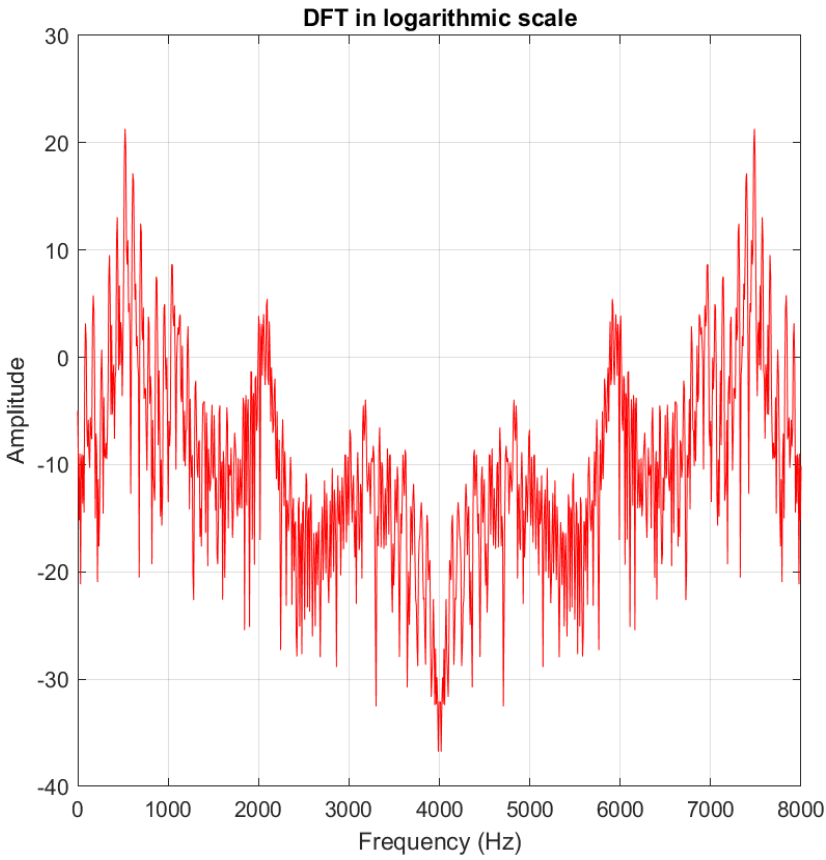
Δ) Εφαρμόζουμε Discrete Fourier Transform (DFT) στο απομονωμένο σήμα του ερωτήματος Β υπολογισμένο σε $N = 1024$ δείγματα, αξιοποιώντας την εντολή $\boldsymbol{fft}()$.

```
-------------------------------------------------------------------
% Applying the Discrete Fourier Transform (DFT) at our cut signal
DFT_cut_signal = fft(cut_signal,1024);
-------------------------------------------------------------------
```

Στη συνέχεια, υλοποιούμε την γραφική παράσταση του $|X[\kappa]|$ $\kappa\alpha\iota$ $20\log_{10}|X[k]|$ στο ίδιο figure χρησιμοποιώντας την εντολή ***subplot***().

```
-------------------------------------------------------------------------
% Initializing the frequency vector
fft_frequency= linspace(0,fs_audio,length(DFT_cut_signal));

% Plotting the DFT in linear scale
figure
subplot(1,2,1);
plot(fft_frequency,abs(DFT_cut_signal));
grid on
xlabel('Frequency (Hz)');
ylabel('Amplitude');
title('DFT in linear scale');

% Plotting the DFT in logarithmic scale
subplot(1,2,2);
plot(fft_frequency, 20*log10(abs(DFT_cut_signal)),'r');
grid on
xlabel('Frequency (Hz)');
ylabel('Amplitude');
title('DFT in logarithmic scale');
-------------------------------------------------------------------------
```





Ε) Εποπτικά, μπορούμε να δούμε πως η κυρίαρχη συχνότητα που επικρατεί στο σήμα μας (και η θεμελιώδης) είναι $86.0215Hz$. Η θεμελιώδης περίοδος του σήματος θα είναι

$$T_o = \frac{1}{86.0215}\ sec = 11.16msec \cong 11.12\ msec = T$$

Με αυτό τον τρόπο επιβεβαιώνουμε την σχέση θεμελιώδους συχνότητας περιόδου.

Άσκηση 2: Ανακατασκευή Περιγράμματος Σχήματος (Shape Tracing) μέσω Σειρών Fourier

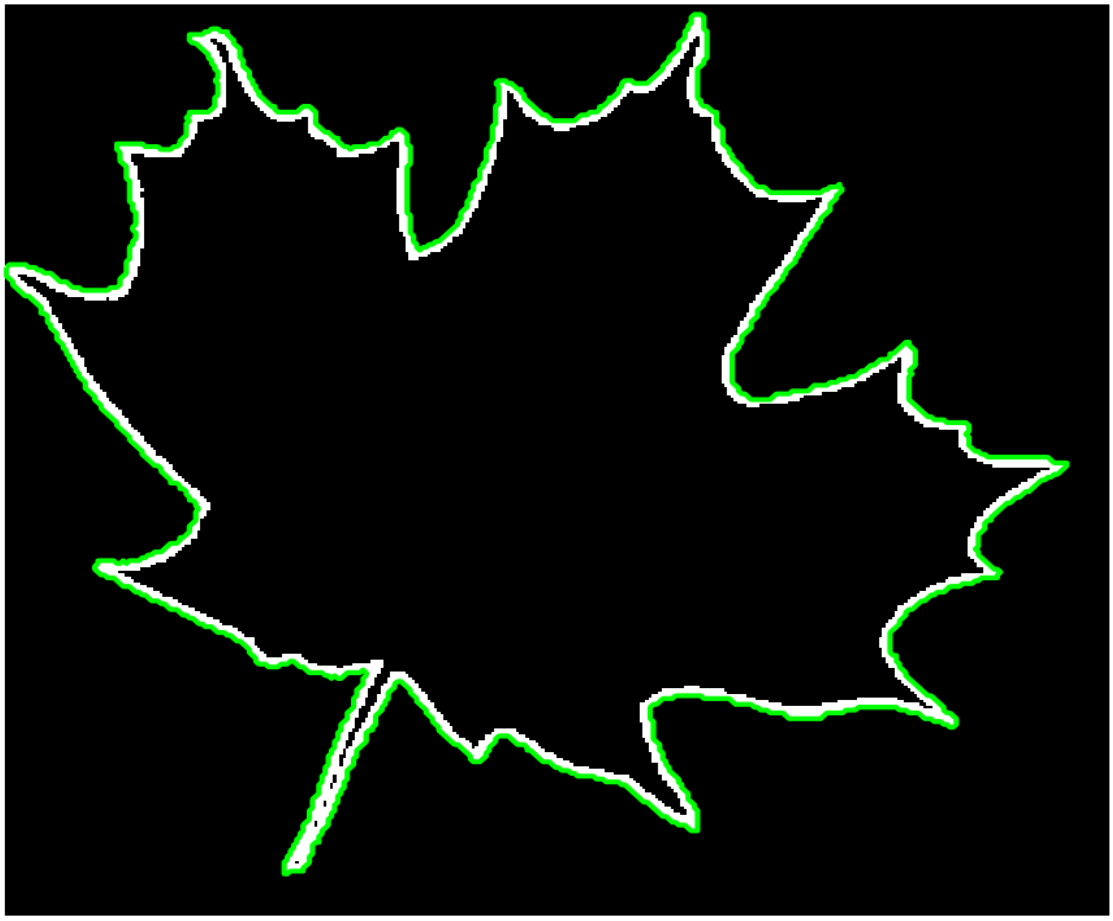
Α) Φορτώνουμε την εικόνα *leaf.png* και την απεικονίζουμε με τις εντολές $\boldsymbol{imread}()$ και $\boldsymbol{imshow}()$ αντίστοιχα.

```
-------------------------------------------------------------------------
% Reading and saving the image
leaf = imread('leaf3.png');

% Displaying the image
figure
imshow(leaf);
-------------------------------------------------------------------------
```

Β) Η εικόνα μας αποτελεί ένα 2-D array με τιμές 0 ή 1, όπου η τιμή 0 αντιπροσωπεύει μαύρο pixel ενώ η τιμή 1 άσπρο. Για να βρούμε το περίγραμμα της εικόνας πρέπει να βρούμε μια θέση όπου να αρχίζει η εύρεση του περιγράμματος, το οποίο επιτυγχάνεται με την εντολή $\boldsymbol{find}()$, και στη συνέχεια με την εντολή $\boldsymbol{bwtraceboundary}()$ βρίσκουμε τα μονοδιάστατα σήματα που ζητούνται και συνεπώς το περίγραμμα της εικόνας.

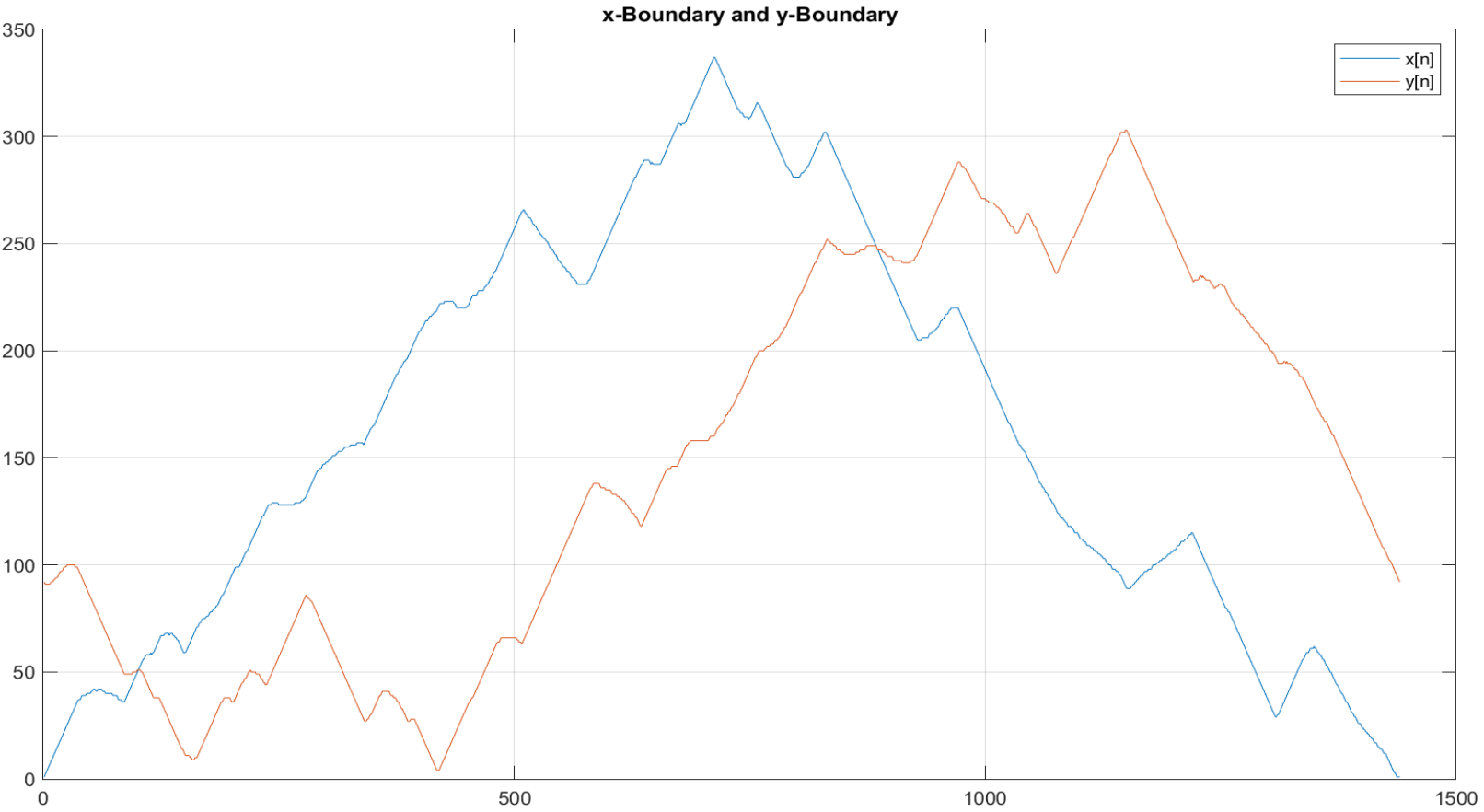
```
-------------------------------------------------------------------------
% With the find function find all the positions of the 1's
[row, column] = find(leaf);

% Using the bwtraceboundary function to trace the boundary
contour = bwtraceboundary(leaf,[row(1),column(1)],'N');
plot(contour(:,2),contour(:,1),'g','Linewidth',2);
-------------------------------------------------------------------------
```

Όπου, `contour(:,2)` αντιπροσωπεύει την x-συνιστώσα ενώ το `contour(:,1)` αντιπροσωπεύει την y-συνιστώσα



Έπειτα, σχεδιάζουμε τα σήματα σε κοινό διάγραμμα.

```
-------------------------------------------------------------------------
% Ploting the x and y values
figure
x = contour(:,2);
y = contour(:,1);
plot(x);
grid on
hold on
plot(y);
hold off
title('x-Boundary and y-Boundary');
legend('x[n]','y[n]')
pause(2);
-------------------------------------------------------------------------
```



Γ) Ορίζουμε το μιγαδικό σήμα $z[n] = x[n] + j \cdot y[n]$

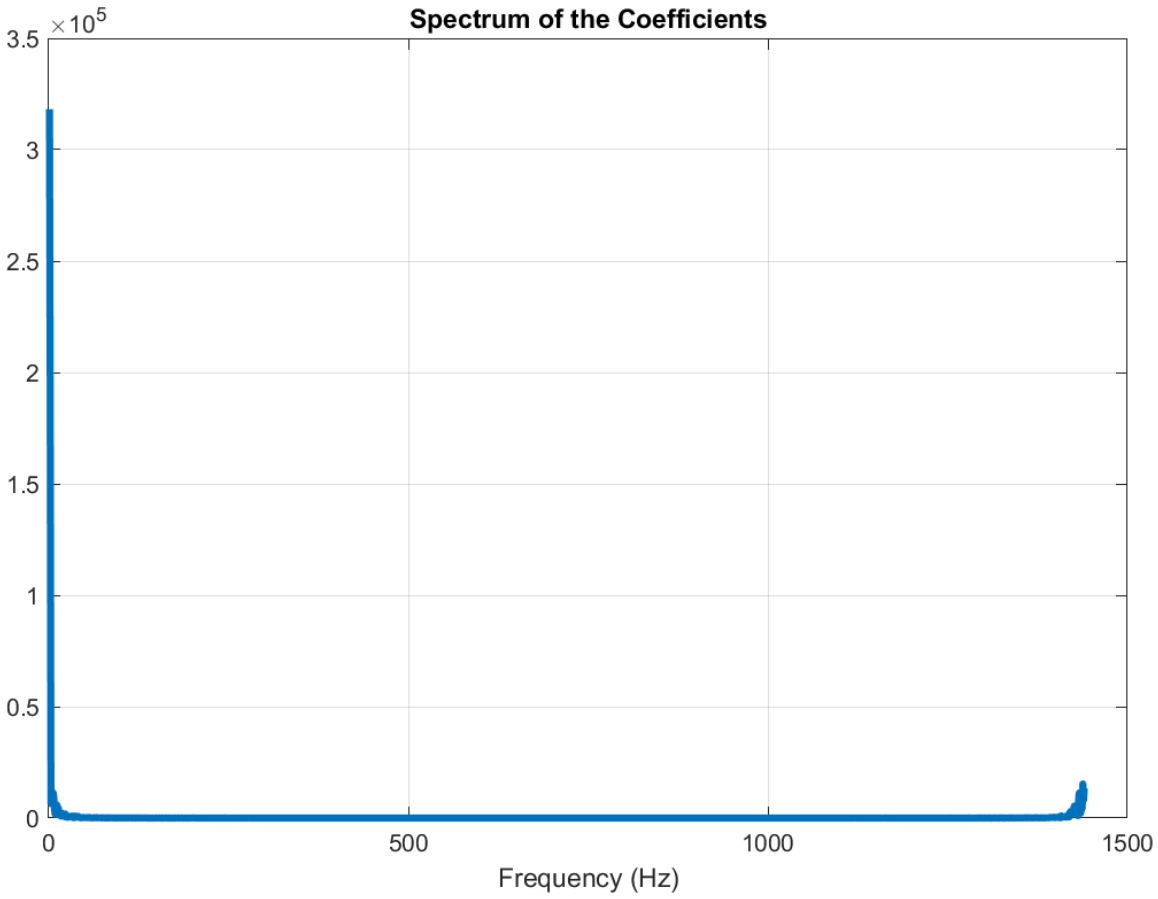
```
z = x + 1j*y;
```

παίρνουμε το DFT

```
Z = fft(z);
```

και σχεδιάζουμε το μέτρο των συντελεστών.

```
-----------------------------------------------------------------------
figure
plot(abs(Z),'Linewidth',3);
grid on
title('Spectrum of the Coefficients');
xlabel('Frequency (Hz)');
-----------------------------------------------------------------------
```



Δ) Αρχικά, με το πρώτο for loop ανακατασκευάζουμε το σήμα για τις τιμές του $M = 10, 50, 200$. Κατόπιν, μέσα στο for loop έχουμε 2 nested-for loops με το πρώτο να χρησιμοποιείται για τον υπολογισμό του αθροίσματος (βάζουμε Z[κ+1] διότι στο MATLAB η δεικτοδότηση αρχίζει από το 1 και όχι από το 0) και το δεύτερο για την ανακατασκευή της εικόνας.
Άθροισμα:

```
Zm = zeros(1,N);
    for k = 0:M
        Zm = Zm + (Z(k+1)*exp((2j*pi*k*n)/N))/N;
    end
```

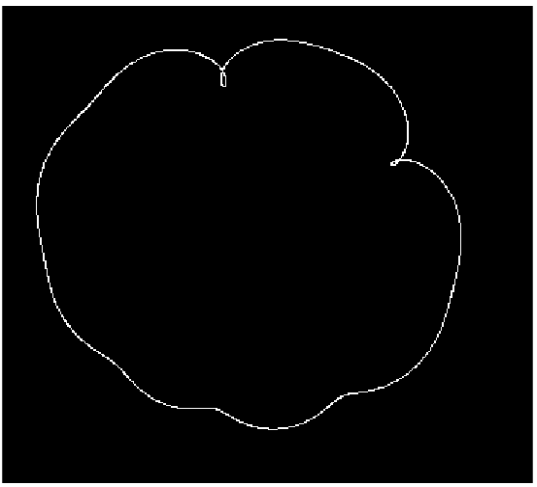
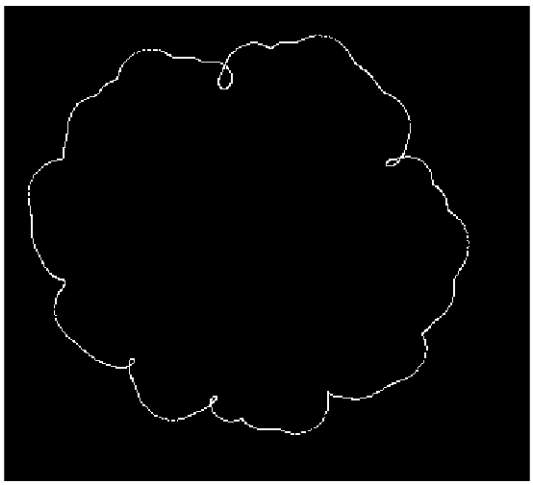
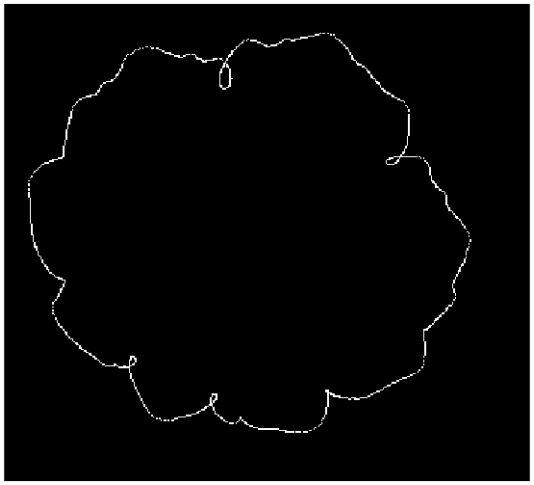
Ανακατασκευή εικόνας:

```
% Reconstructing the 317x350 image
    reconstructed_image = zeros(317,350);
    for index = 1:length(Xz)
        reconstructed_image(Xz(index)+1,Yz(index)+1) = 1;
    end
```

Απεικόνιση εικόνας:

```
% Showing the images
    drawnow;
    figure(9)
    imshow(reconstructed_image);
    hold on
    pause(2);
```

- $M = 10$ $M = 50$ $M = 200$

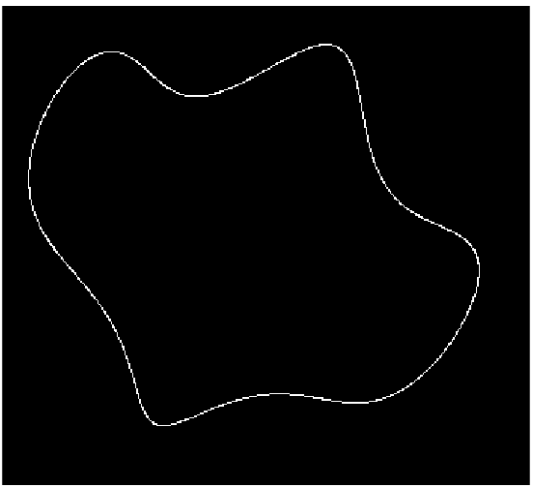
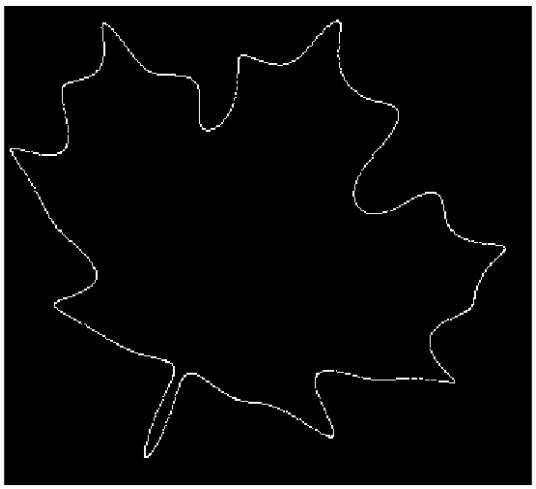
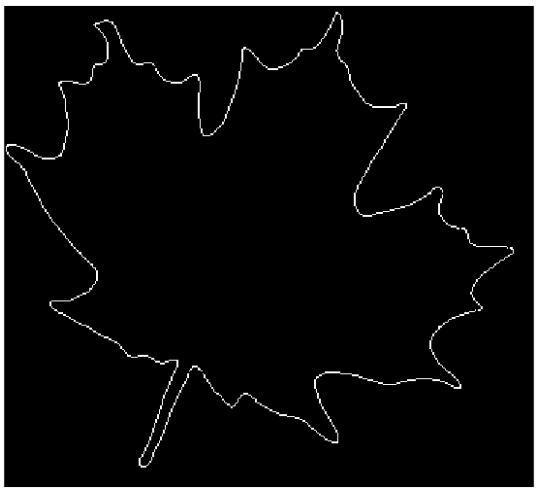
Παρατηρησεις:

- Στο άθροισμα επειδή στο matlab οι δείκτες ξεκινούν από το 1 και όχι το 0 βάζουμε Z(κ+1) αντί για Z(κ).
- Παρατηρούμε πως χρησιμοποιώντας απλα τους συντελεστές Fourier για τις διάφορες τιμές του M δεν έχουμε αρκετή πληροφορία για την ορθή ανακατασκευή του περιγράμματος. Αυτό εξηγείται με το φάσμα του αρχικού διανύσματος z. Συγκεκριμένα, βλέπουμε πως παίρνει πολύ μεγάλες τιμές στις πολύ μικρές και πολύ μεγάλες συχνότητες, ενώ για τις συχνότητες ενδιάμεσα παίρνει μηδενικές τιμές. Επομένως, επειδή δεν αντλούμε πληροφορίες από τις μεγαλύτερες συχνότητες δεν μπορούμε να ανακατασκευάσουμε σωστά το περίγραμμα της εικόνας.

Ε) Επαναλαμβάνουμε την παραπάνω διαδικασία αλλάζοντας το άθροισμα, με σκοπό να αξιοποιήσουμε τους συμμετρικούς συντελεστές Fourier.

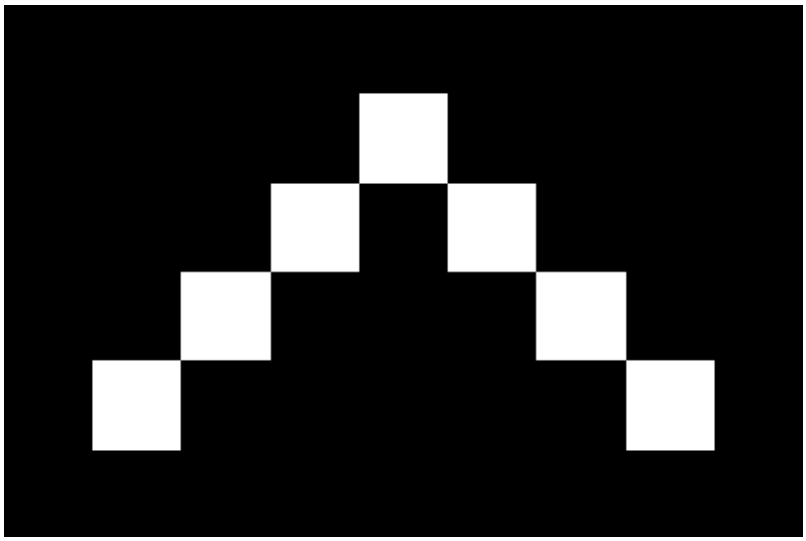
```
Zm = zeros(1,N);
    for k = 0:M/2
        Zm = Zm + (Z(k+1)*exp((2j*pi*(k)*n)/N))/N;
    end

    for k = N-M/2: N-1
        Zm = Zm + (Z(k+1)*exp((2j*pi*k*n)/N))/N;
    end
```

- $M = 10$ $M = 50$ $M = 200$

Παρατηρήσεις: Τώρα που παίρνουμε τις αρχικές και τις τελικές συχνότητες χρησιμοποιώντας τους συμμετρικούς συντελεστές Fourier λαμβάνουμε όλη την πληροφορία της εικόνας και γι'αυτό τον λόγο από τους πρώτους 50 συντελεστές έχουμε ήδη μια ικανοποιητική ανακατασκευή του αρχικού περιγράμματος. Στους 200 συντελεστές έχουμε ουσιαστικά την ίδια εικόνα.

ΣΤ) Η δυαδική εικόνα που επιλέγω είναι η εξής:

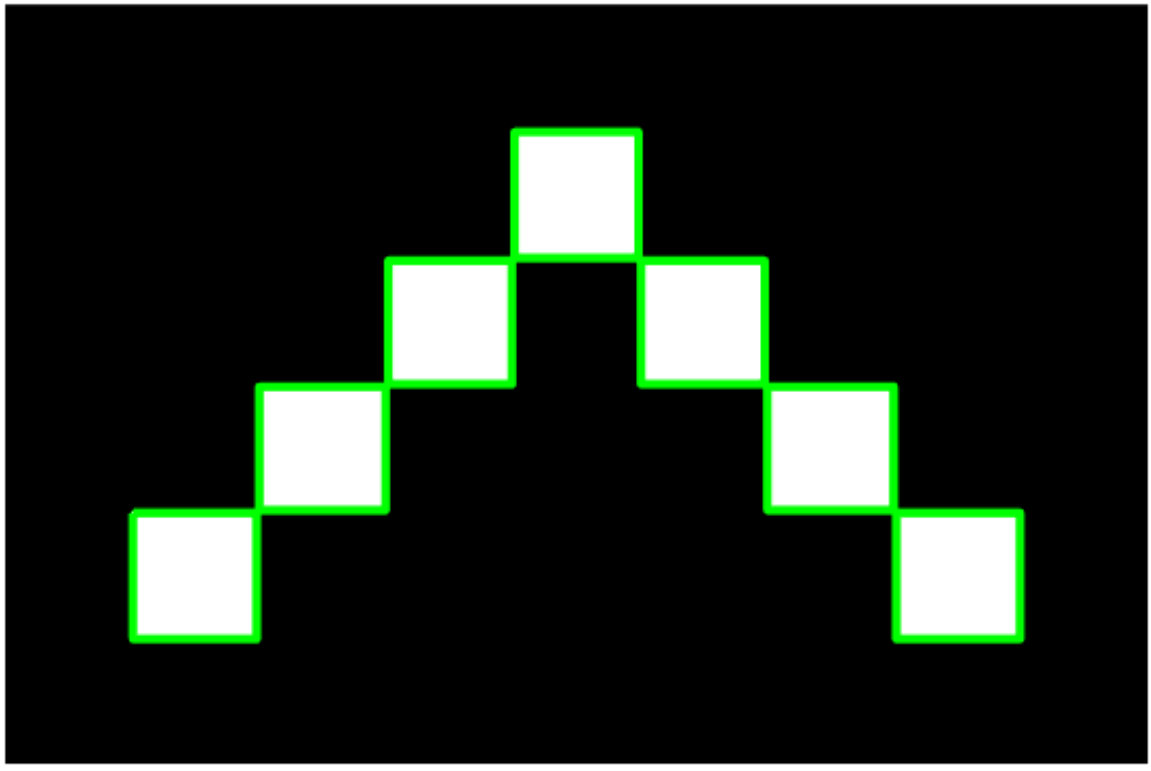


Αρχικά, αποθηκεύω την παραπάνω εικόνα και την μετατρέπω στην επιθυμητή δυαδική μορφή με την χρήση των εντολών ***imread***() και ***im2bw***() αντίστοιχα.

```
-------------------------------------------------------------------------
% Reading and saving the image
rect = imread('rect.png');
rect = im2bw(rect);
-------------------------------------------------------------------------
```

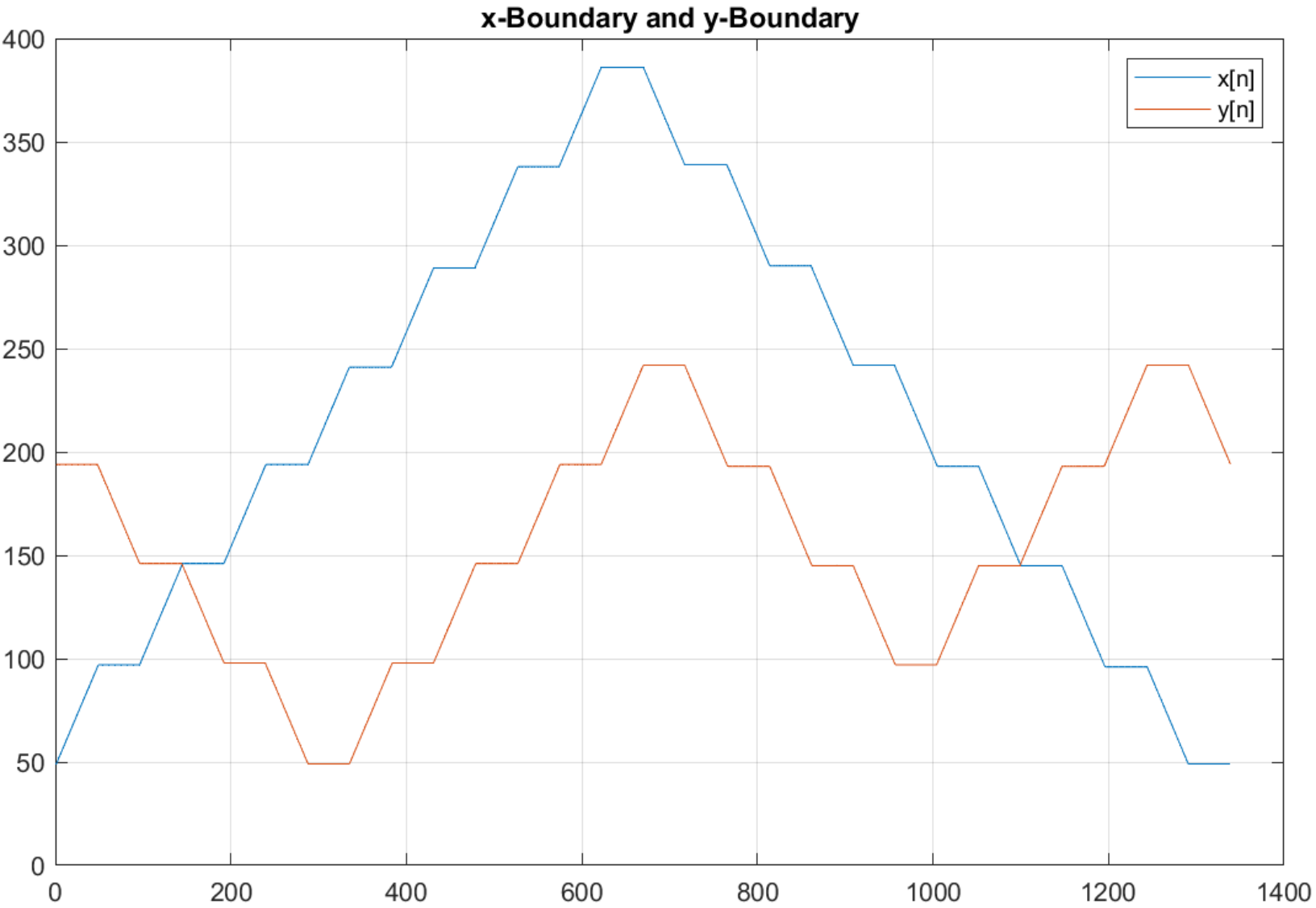
Έπειτα, απεικονίζω την εικόνα και το περίγραμμά της.

```
-------------------------------------------------------------------------
% Displaying the image
figure
imshow(rect);
pause(2);

hold on

% With the find function find all the positions of the 1's
[row, column] = find(rect);

% Using the bwtraceboundary function to trace the boundary
contour = bwtraceboundary(rect,[row(1),column(1)],'N');
plot(contour(:,2),contour(:,1),'g','Linewidth',2);
pause(2);
-------------------------------------------------------------------------
```



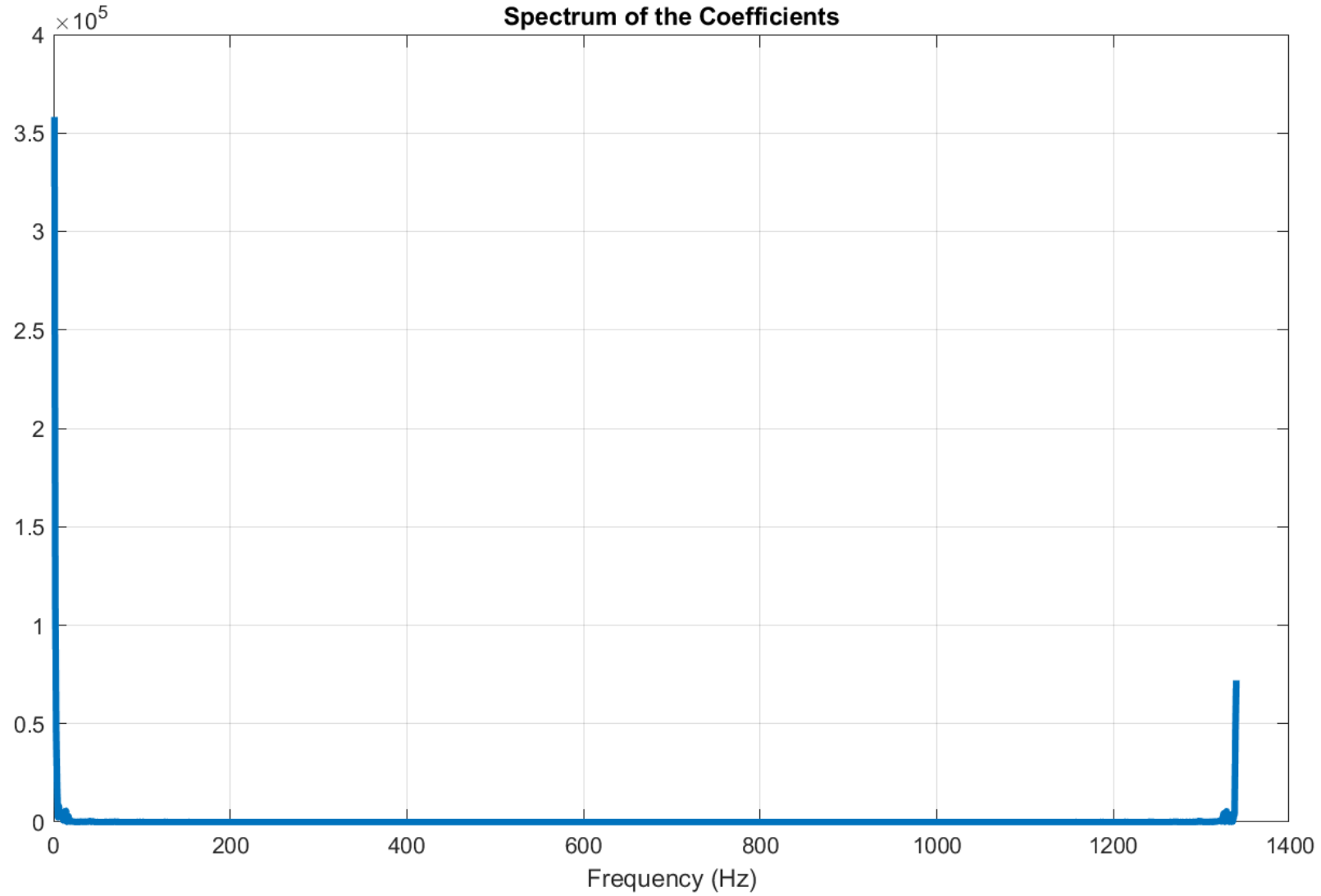
Στη συνέχεια, κάνω την γραφική παράσταση των σημάτων $x[n]\ and\ y[n]$.

```
-------------------------------------------------------------------------
% Ploting the x and y values
figure
x = contour(:,2);
y = contour(:,1);
plot(x);
grid on
hold on
plot(y);
hold off
title('x-Boundary and y-Boundary');
legend('x[n]','y[n]')
pause(2);
-------------------------------------------------------------------------
```



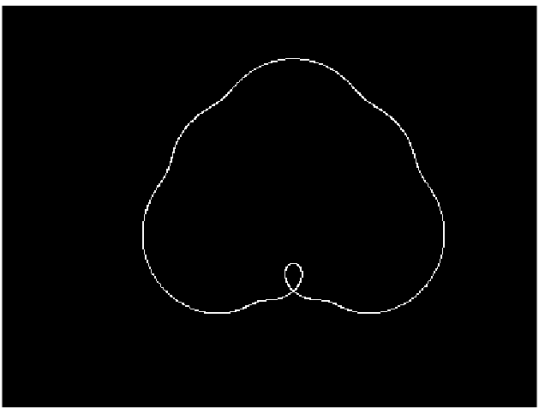
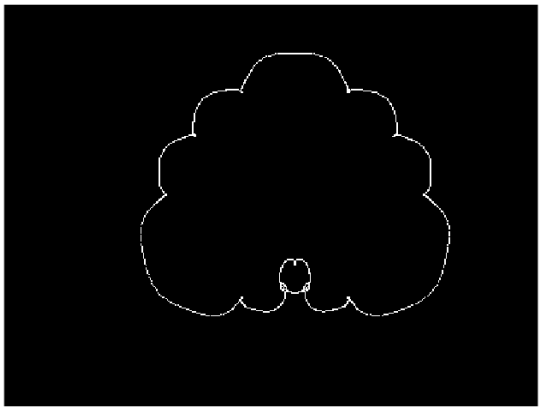
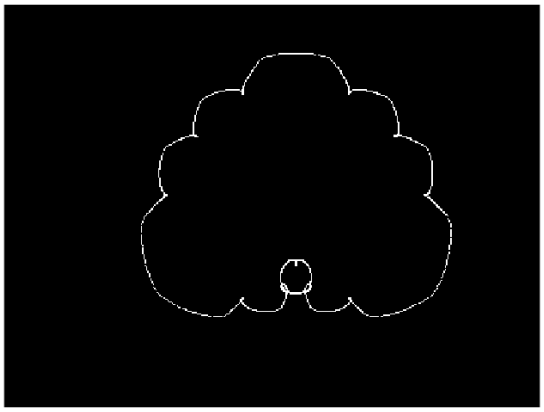
Κατόπιν, υπολογίζουμε το μιγαδικό σήμα $z[n] = x[n] + j \cdot y[n]$, παίρνουμε τον διακριτό μετασχηματισμό Fourier και μετά κάνουμε την γραφική παράσταση του $|Z[\kappa]|$.

```
-------------------------------------------------------------------------
z = x + 1j*y;

% Calculating the coefficients of the discrete fourier series
with fft()
Z = fft(z);
figure
plot(abs(Z),'Linewidth',3);
grid on
title('Spectrum of the Coefficients');
xlabel('Frequency (Hz)');
pause(2);
-------------------------------------------------------------------------
```



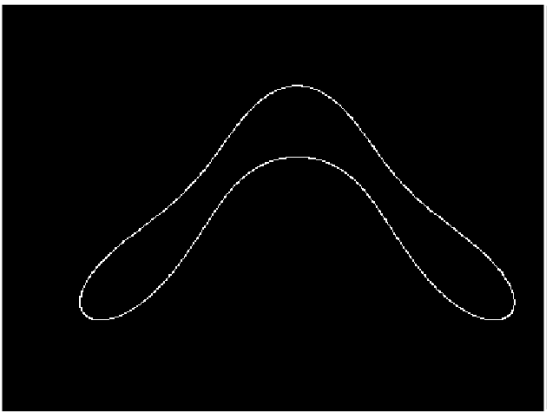
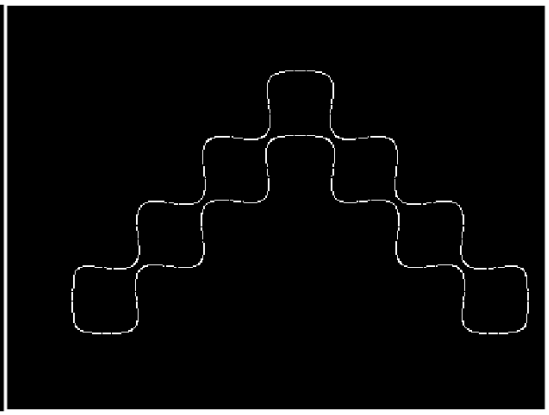
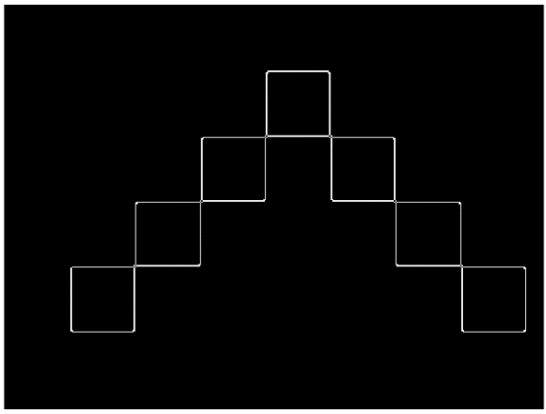
Για την ανακατασκευή της εικόνας με την χρήση των συντελεστών Fourier χρησιμοποιώ την ακριβώς ίδια διαδικασία με το ερώτημα (Δ) και το αποτέλεσμα είναι το εξής:

- $M = 10$ $M = 50$ $M = 200$

Χρησιμοποιώ τώρα τους συμμετρικούς συντελεστές Fourier για την ανακατασκευή.

- $M = 10$ $M = 50$ $M = 200$

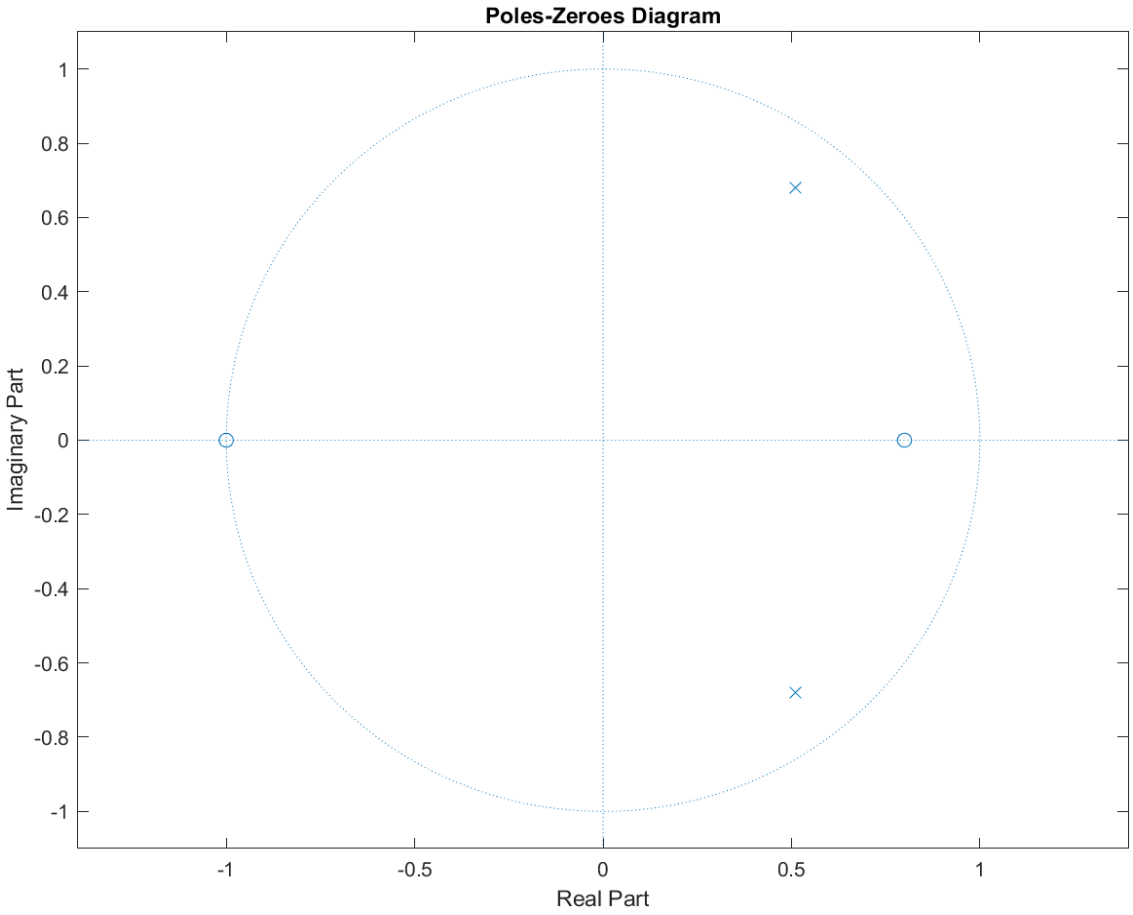
Άσκηση 3: Σχεδίαση Φίλτρων και Εφαρμογή σε Σήμα Μουσικής

Άσκηση 3.1: Σχεδίαση Ζωνοπερατών Φίλτρων

Α) Μας δίνονται οι εξής θέσεις πόλων και μηδενικών αντίστοιχα, $\{0.51 \pm 0.68 \cdot i\}, \{0.8, -1\}$, τα οποία αποτυπώνουμε στο ακόλουθο διάγραμμα χρησιμοποιώντας την εντολή $\boldsymbol{zplane}()$:

```
-------------------------------------------------------------------------
% Storing the poles and the zeroes into vectors
poles = [0.51 + 1j*0.68, 0.51 - 1j*0.68];
zeroes = [0.8, -1];
```

```
% Plotting the diagram poles-zeroes
figure
% zplane requires column vectors so we use the transpose function
zplane(transpose(zeroes),transpose(poles));
title('Poles-Zeroes Diagram');
pause(2);
```



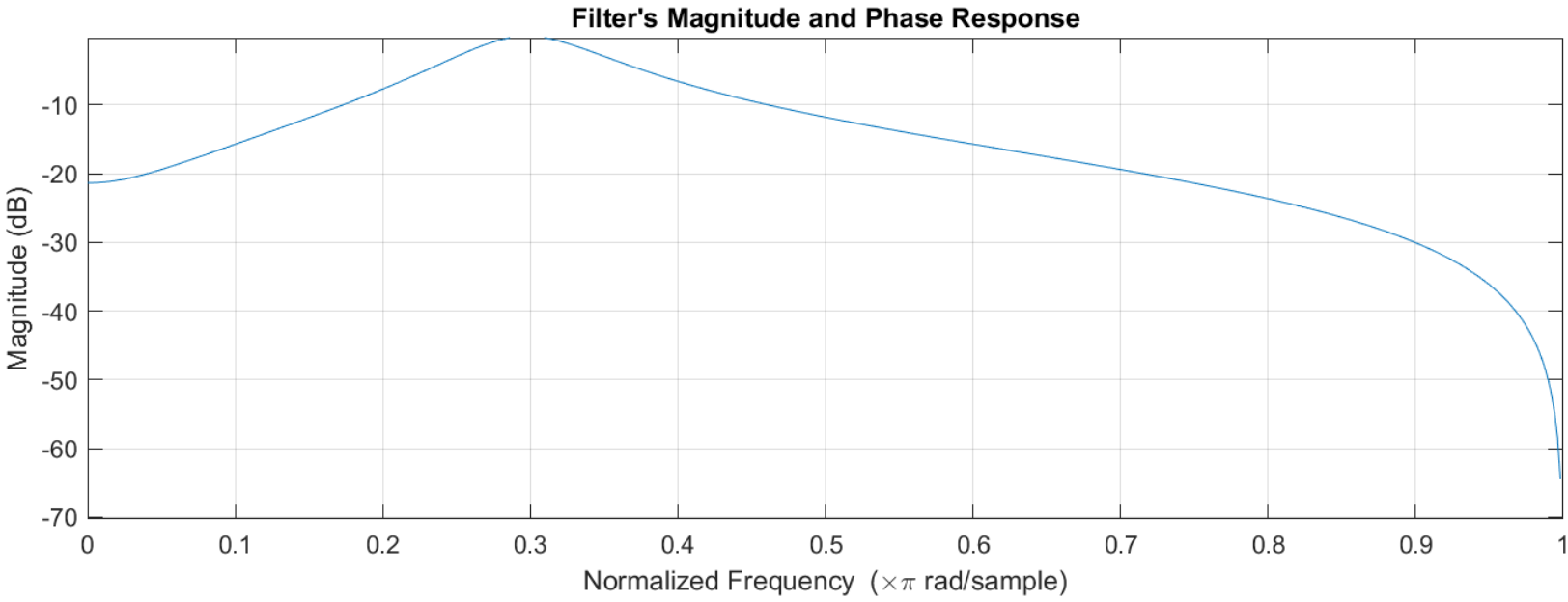
Στη συνέχεια, γνωρίζοντας τους πόλους και τα μηδενικά του φίλτρου με την βοήθεια την εντολής ***zp2tf*()** βρίσκουμε τους συντελεστές του αριθμητή και του παρανομαστή της συνάρτησης μεταφοράς.

```
% Calculating the vector coefficients a and b
K = 0.15;
[num, den] = zp2tf(transpose(zeroes),transpose(poles),K);
% Numerator: Coefficients of the output variable Y(z)
% Denominator: Coefficients of the input variable X(z)
```

Β) Έπειτα, γνωρίζοντας τους συντελεστές της συνάρτησης μεταφοράς, σχεδιάζουμε την απόκριση πλάτους και με χρήση της συνάρτησης ***freqz*()**.

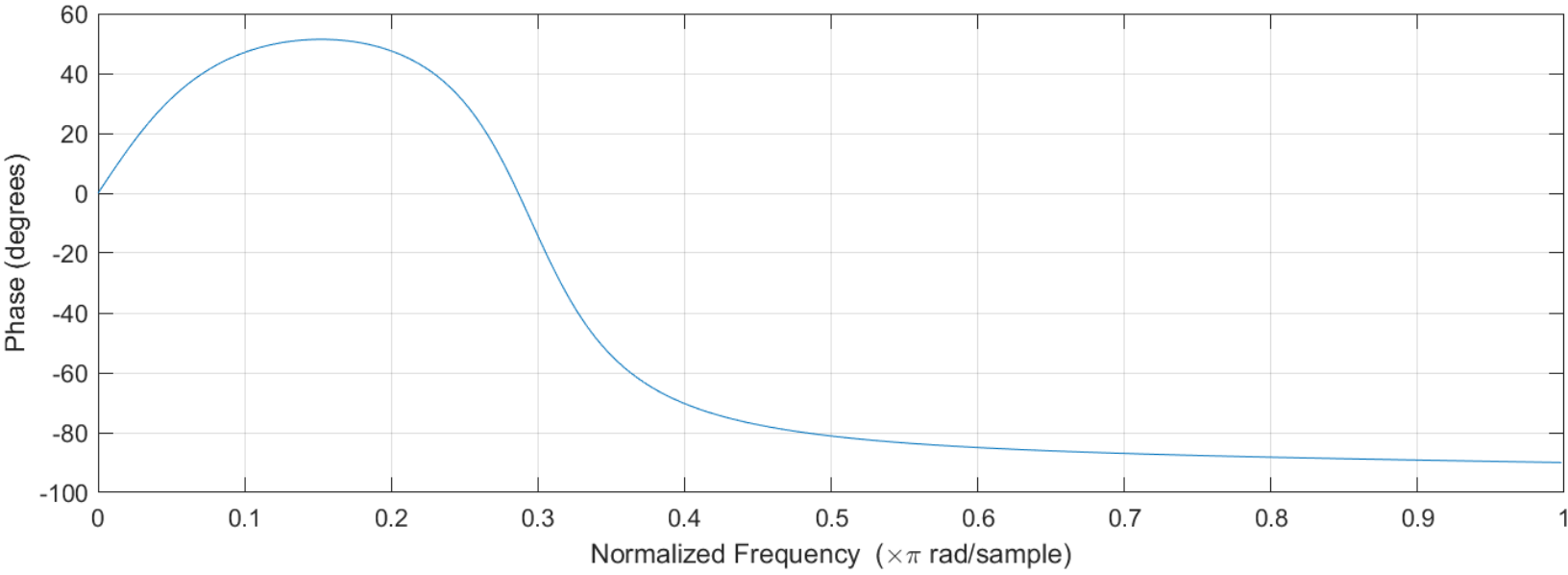
```
-------------------------------------------------------------------------
% Plotting the Amplitude and Phase Response of the filter
figure
freqz(num,den);
title('Filter''s Magnitude and Phase Response')
pause(2);
-------------------------------------------------------------------------
```

- *Απόκριση πλάτους:*



Σχολιασμός: Παρατηρούμε πως για ζωνοπερατό φίλτρο με κεντρική συχνότητα $0.3 \times \pi \frac{rad}{sample}$ και η γωνία των πόλων $\arctan\left(\frac{0.68}{0.51}\right) \cong 0.3\pi$ είναι περίπου ίσες. Παράλληλα, παρατηρούμε πως στη κεντρική συχνότητα του φίλτρου η απόκριση πλάτους αποκτά την μέγιστη τιμή.

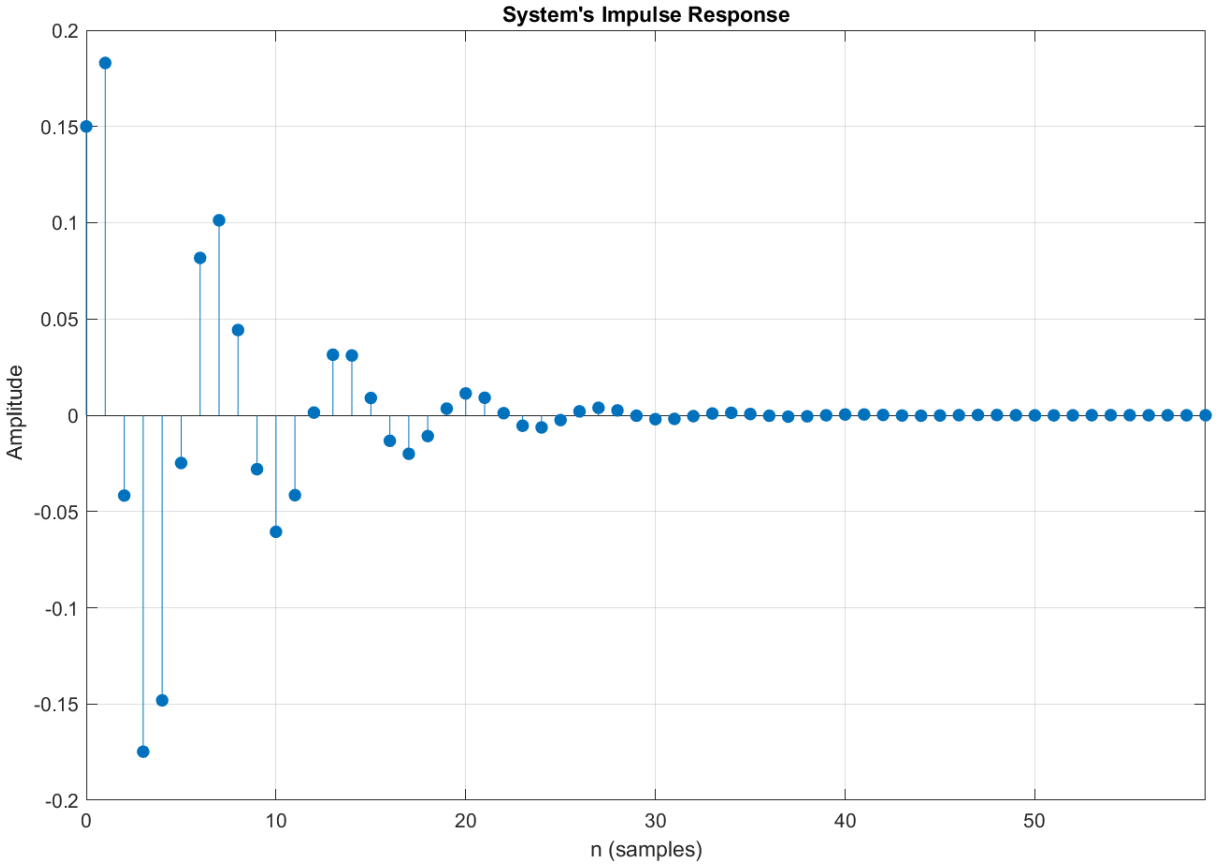
- *Απόκριση φάσης:*

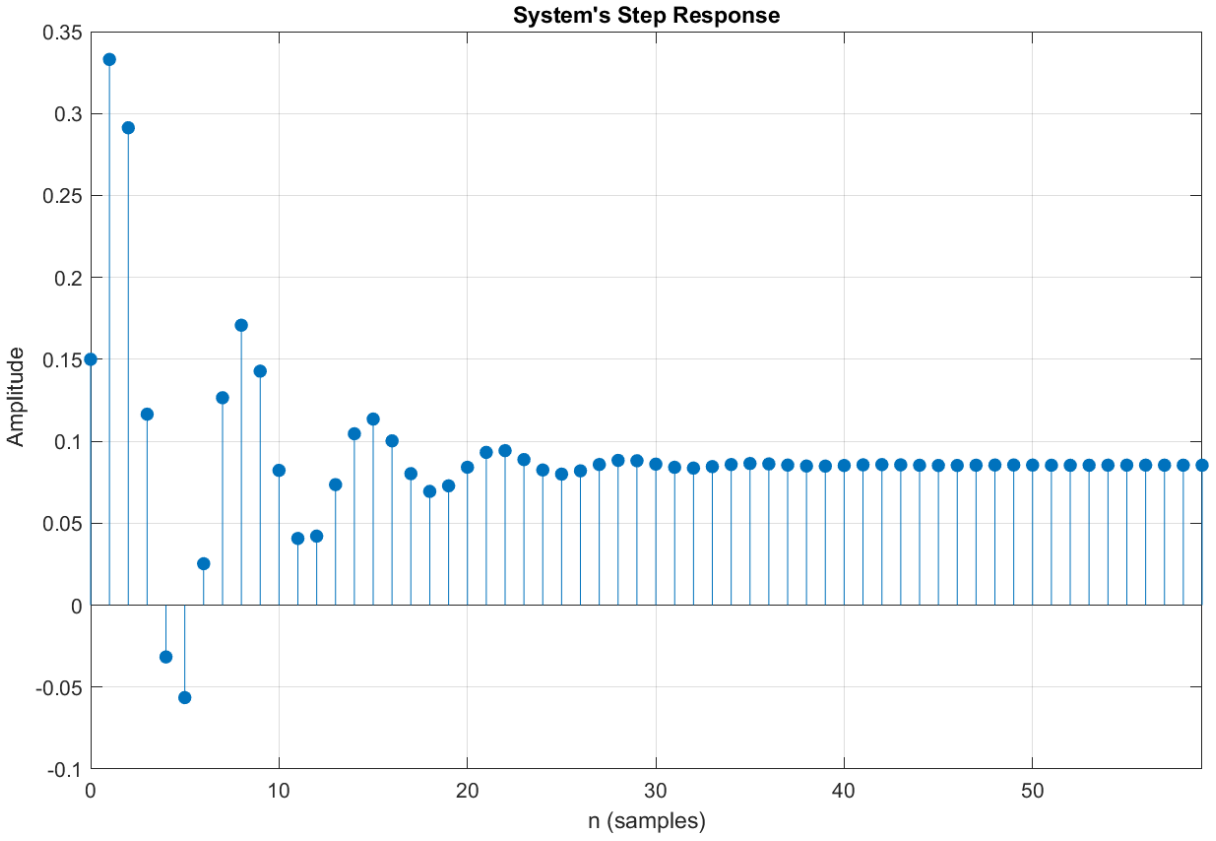


Σχολιασμός: Παρατηρούμε πως η γραφική παράσταση της απόκρισης φάσης αλλάζει κυρτότητα στην τιμή της κεντρικής συχνότητας.

Γ) Σχεδιάζουμε την κρουστική απόκριση του συστήματος χρησιμοποιώντας την συνάρτηση ***impz***() και την βηματική απόκριση του συστήματος χρησιμοποιώντας την συνάρτηση ***stepz***().

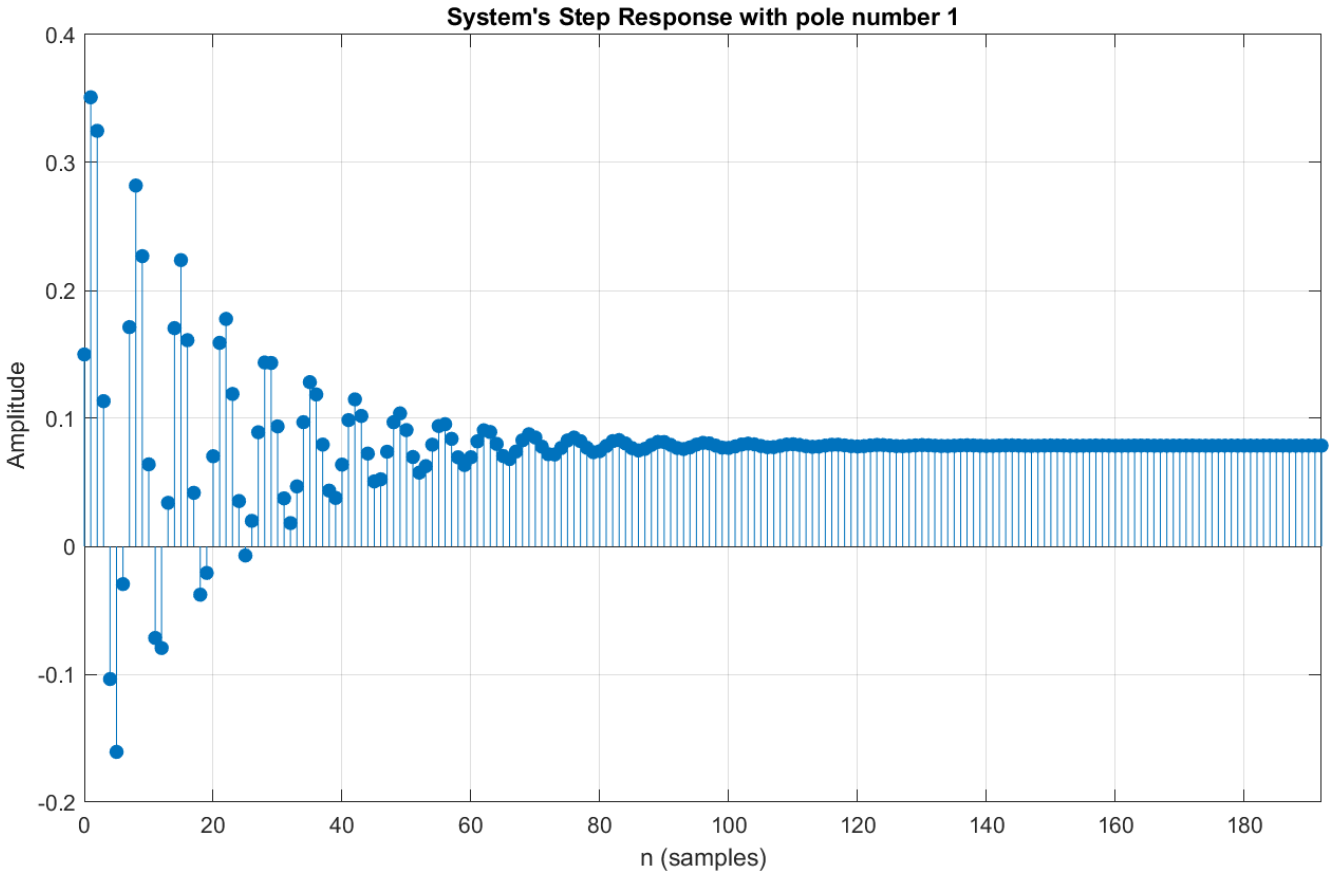
```
-------------------------------------------------------------------------
% Plotting the impulse response of the system
figure
impz(num,den);
grid on
title('System''s Impulse Response');
pause(2);

% Plotting the step response of the system
figure
stepz(num,den);
grid on
title('System''s Step Response');
pause(2);
-------------------------------------------------------------------------
```


System's Impulse Response
Amplitude
n (samples)
0.2
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
0
10
20
30
40
50

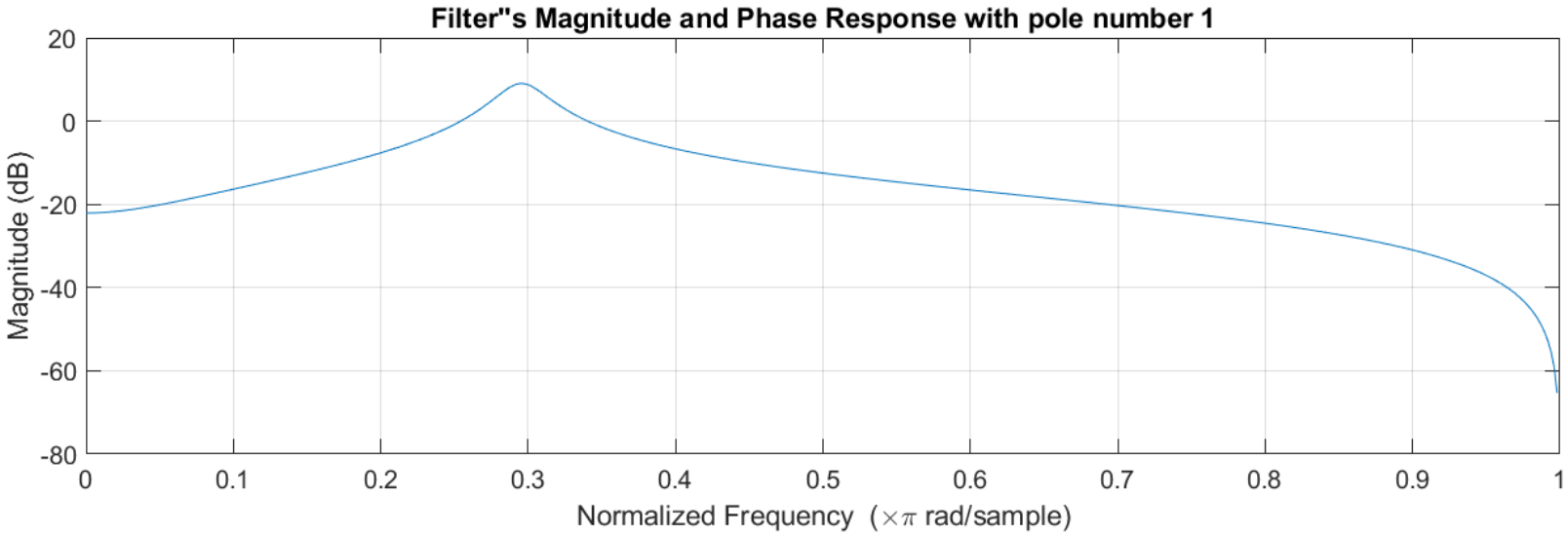

System's Step Response
Amplitude
n (samples)
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
-0.05
-0.1
0
10
20
30
40
50

Δ) Για πόλους $\{0.57 \pm 0.76i\}$:

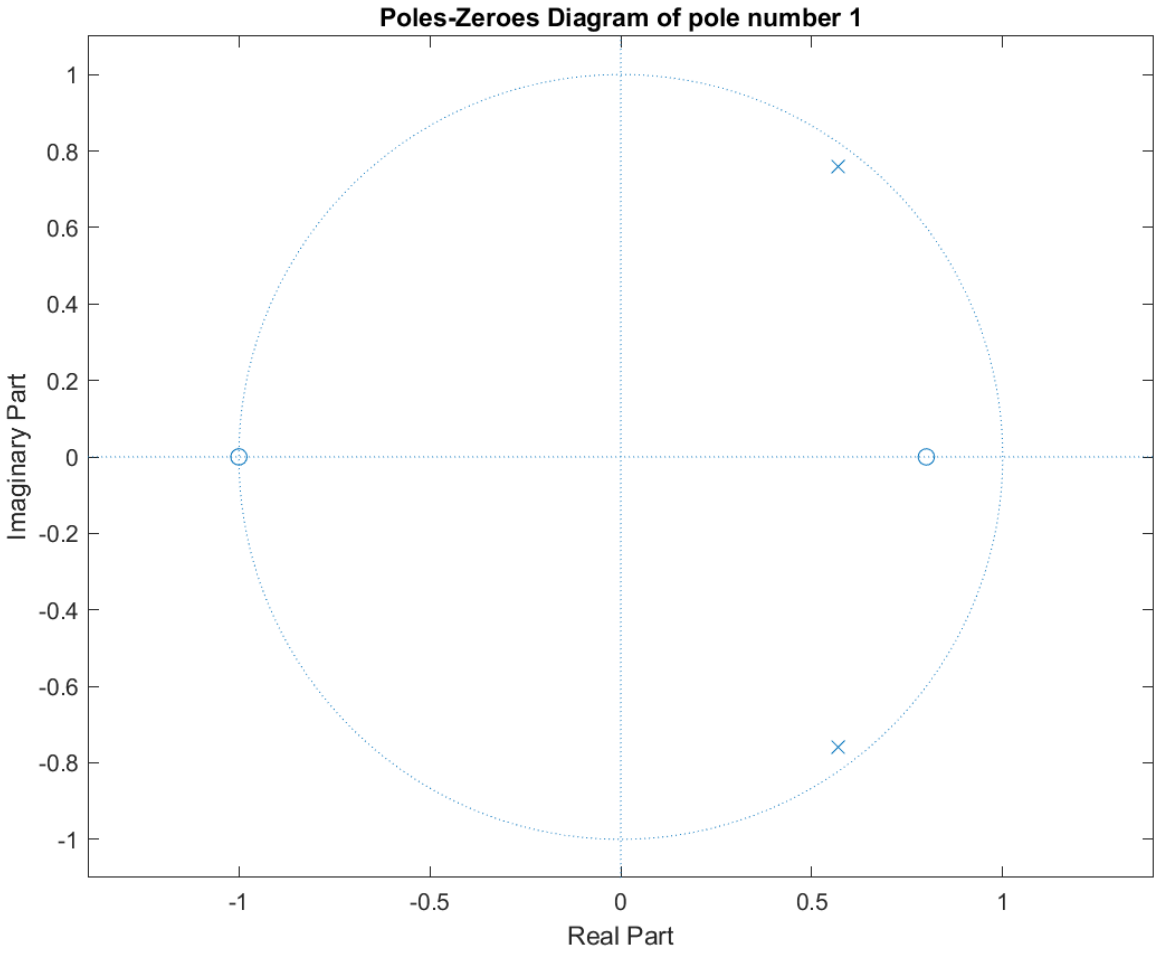
- Βηματική Απόκριση



- Απόκριση Πλάτους



- Διάγραμμα Πόλων-Μηδενικών



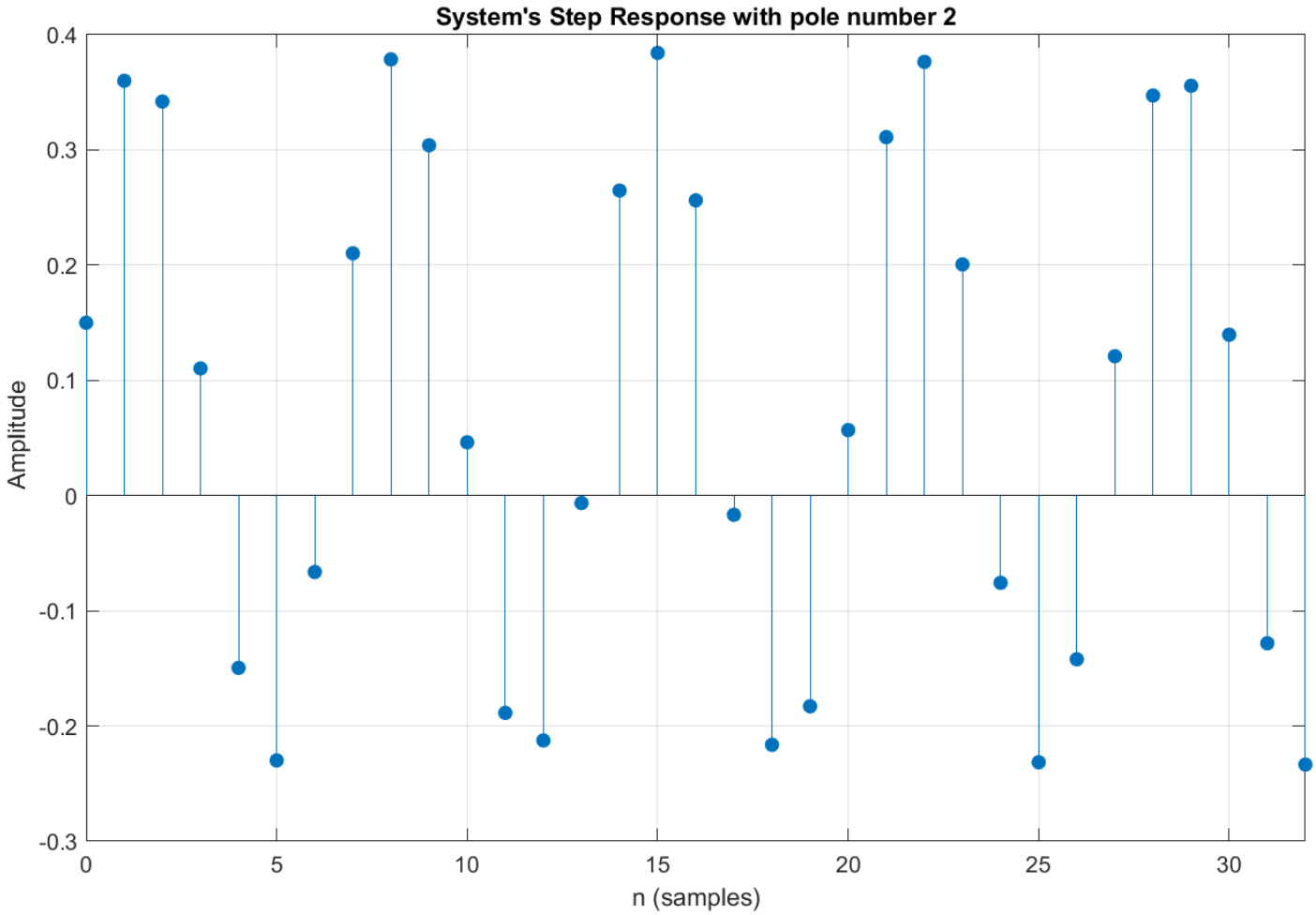
Σχολιασμός:

- Αφενός παρατηρούμε πως επειδή δεν άλλαξε η γωνία των πόλων στο μιγαδικό επίπεδο δεν άλλαξε και η κεντρική συχνότητα και αφετέρου παρατηρούμε πως παρόλο που η κεντρική συχνότητα παρέμεινε ίδια, επειδή αυξήθηκε το μέτρο των πόλων στο μιγαδικό επίπεδο (η απόσταση από την αρχή των αξόνων) βλέπουμε μια πιο απότομη αλλαγή στη κεντρική συχνότητα (πιο «μυτερό» σημείο) .

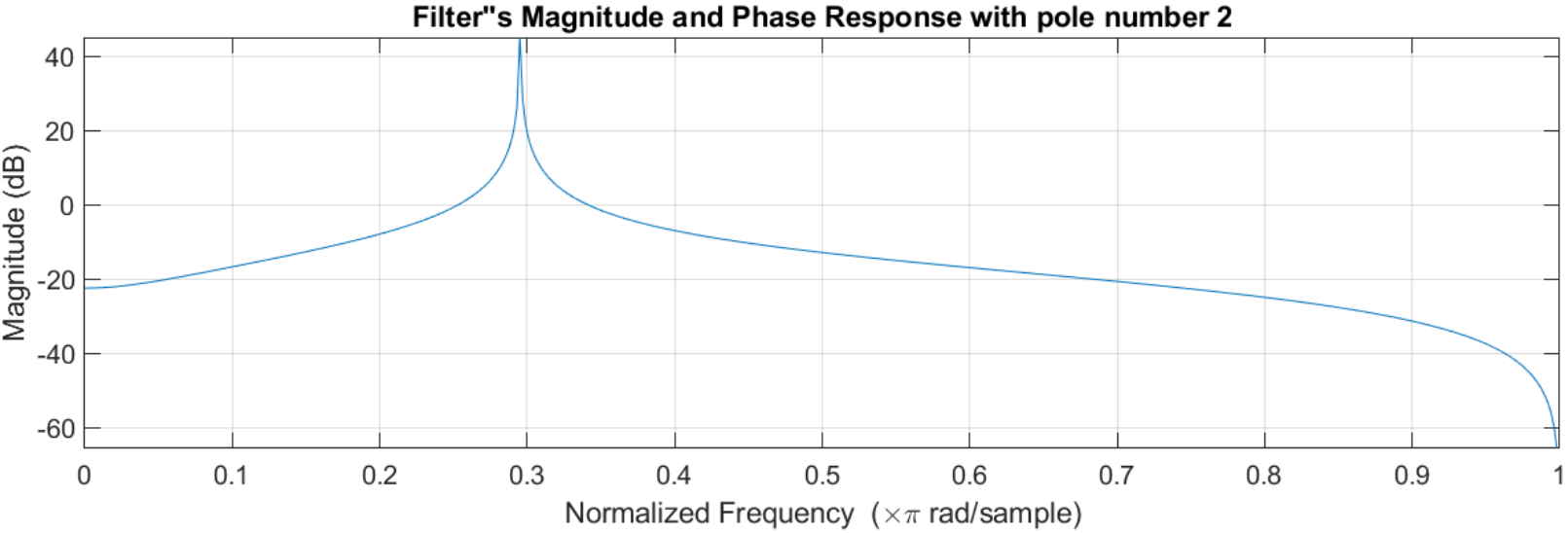
- Επίσης, παρατηρούμε πως επειδή οι πόλοι βρίσκονται μέσα στο μοναδιαίο κύκλο, η βηματική απόκριση είναι ασταθής στην αρχή (απότομες αυξομειώσεις) και στη συνέχεια σταθεροποιείται αρκετά γρήγορα.

Για πόλους $\{0.6 \pm 0.8i\}$:

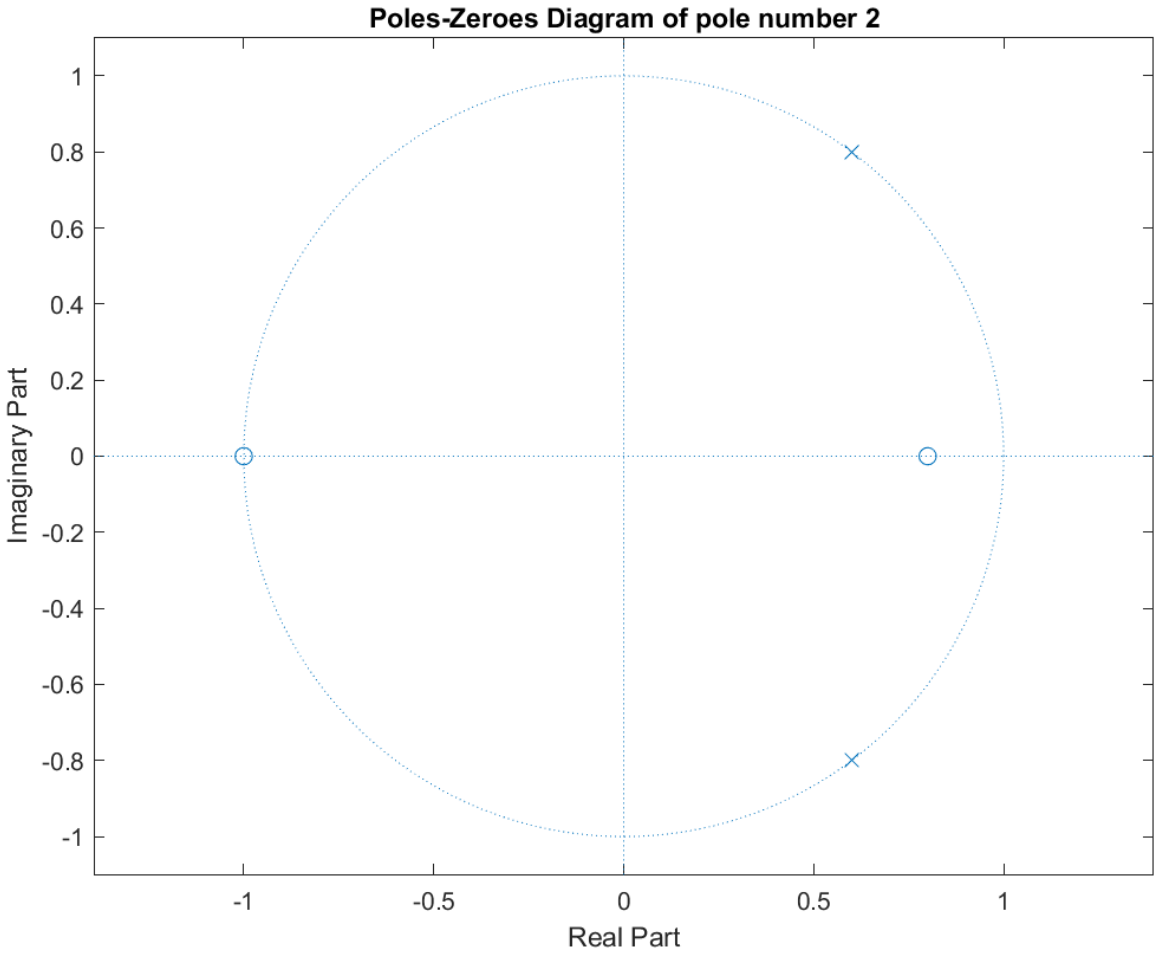
- Βηματική Απόκριση



- Απόκριση Πλάτους



- Διάγραμμα Πόλων-Μηδενικών



Σχολιασμός:

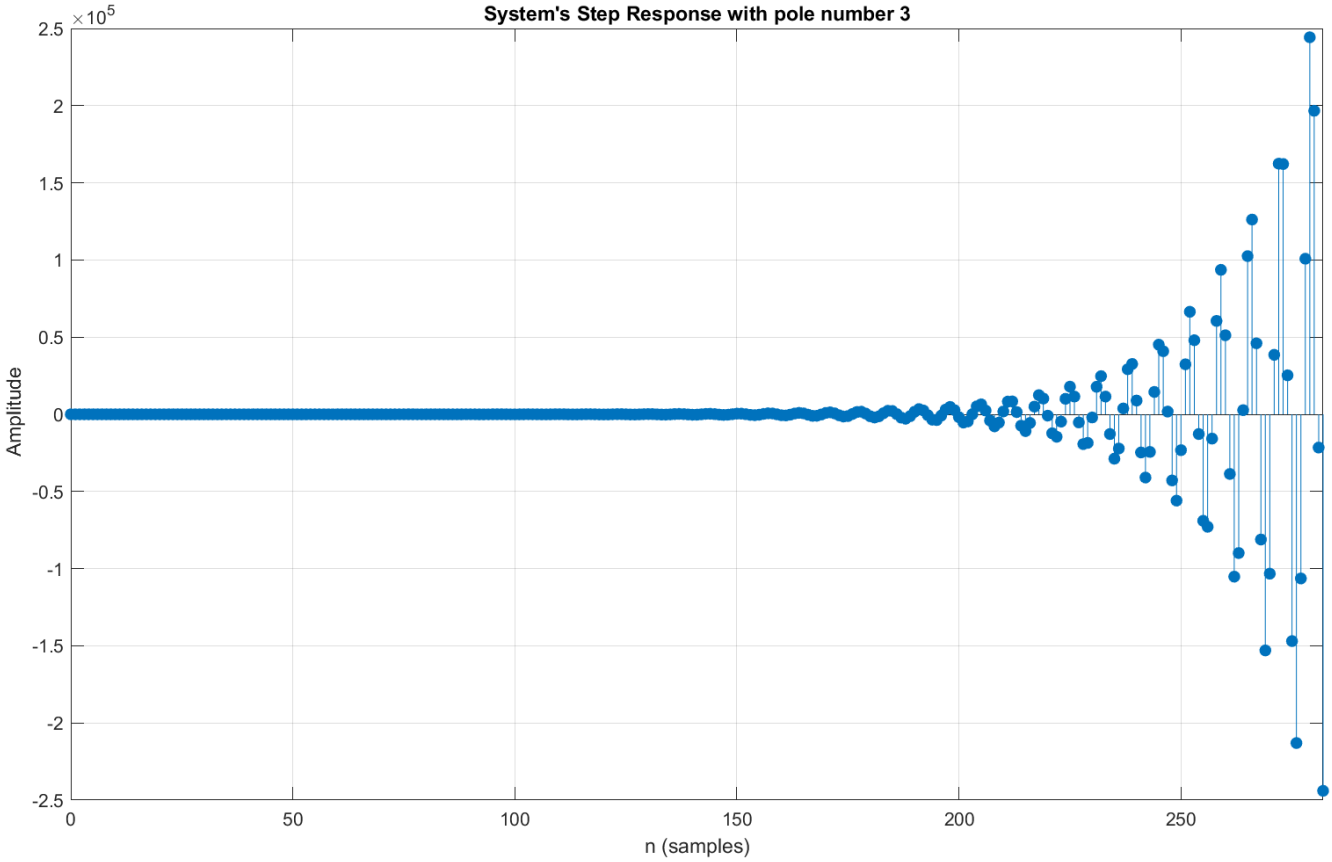
- Αρχικά παρατηρούμε πάλι πως αφού η γωνία των πόλων δεν μεταβλήθηκε, δεν μεταβλήθηκε και η κεντρική συχνότητα του φίλτρου.

- Παράλληλα, επειδή οι πόλοι βρίσκονται πάνω στο μοναδιαίο κύκλο, το σύστημά μας είναι οριακά ευσταθές πράγμα που μπορούμε να δούμε και στην απόκριση πλάτους καθώς η κεντρική συχνότητα αποτελεί γωνιακό σημείο του γράφου μας.

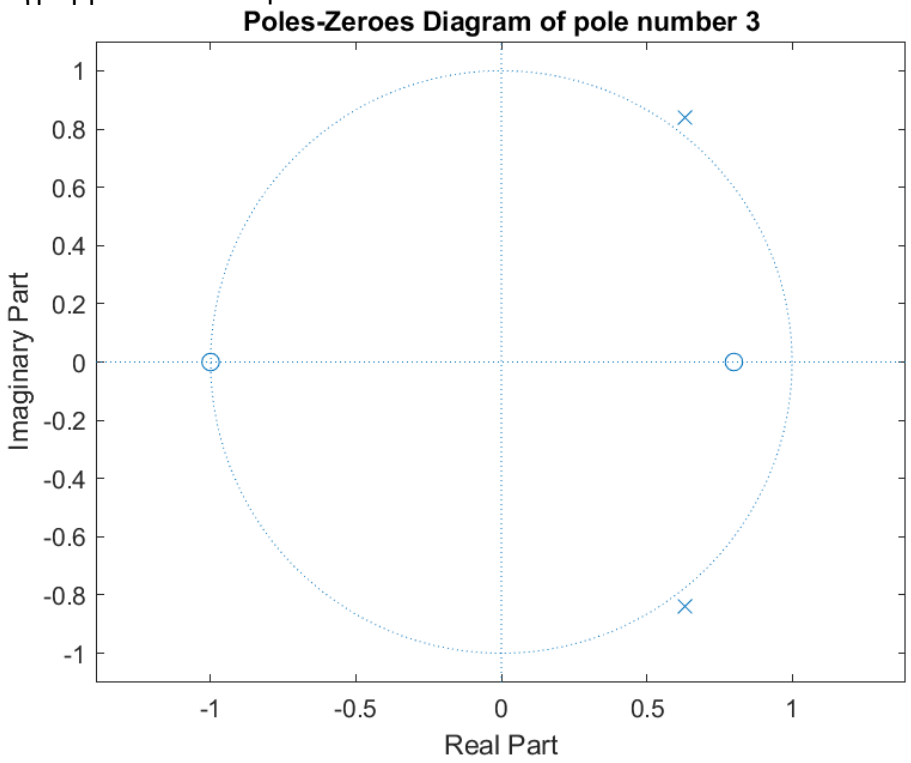
- Ακόμη μια υπόδειξη για την οριακή ευστάθεια έχουμε στην βηματική απόκριση του συστήματός μας, καθώς από την μία δεν τείνει να σταθεροποιηθεί σε μία συγκεκριμένη τιμή και από την άλλη δεν αποκλίνει.

Για πόλους $\{0.63 \pm 0.84i\}$:

- Βηματική Απόκριση:



- Διάγραμμα Πόλων-Μηδενικών



Σχολιασμός: Παρατηρούμε πως οι πόλοι βρίσκονται εκτός του μοναδιαίου κύκλου στο μιγαδικό επίπεδο και συνεπώς το σύστημά μας είναι ασταθές. Επομένως, η βηματική απόκριση θα αρχίζει από την τιμή 0 και στη συνέχεια θα αποκλίνει με την αύξηση του n.

Συμπερασματικά, καταλαβαίνουμε πως σε ένα ζωνοπερατό φίλτρο η γωνία των πόλων στο μιγαδικό επίπεδο προσδιορίζει την κεντρική συχνότητα του φίλτρου μας ενώ το μέτρο τους καθορίζει αφενός αν το σύστημά μας είναι ευσταθές ή ασταθές και αφετέρου, αν είναι ευσταθές , τη ζώνη διέλευσης του φίλτρου.

Ε) Δημιουργούμε το σήμα $x[n] = sin(0.3\pi n) + sin(0.7\pi n)$ με την βοήθεια της συνάρτησης $\boldsymbol{gensig}()$ αφού πρώτα βρούμε την περίοδο των δύο ημιτονοειδών σημάτων καθώς απαιτούνται ως παράμετροι στη συνάρτηση.

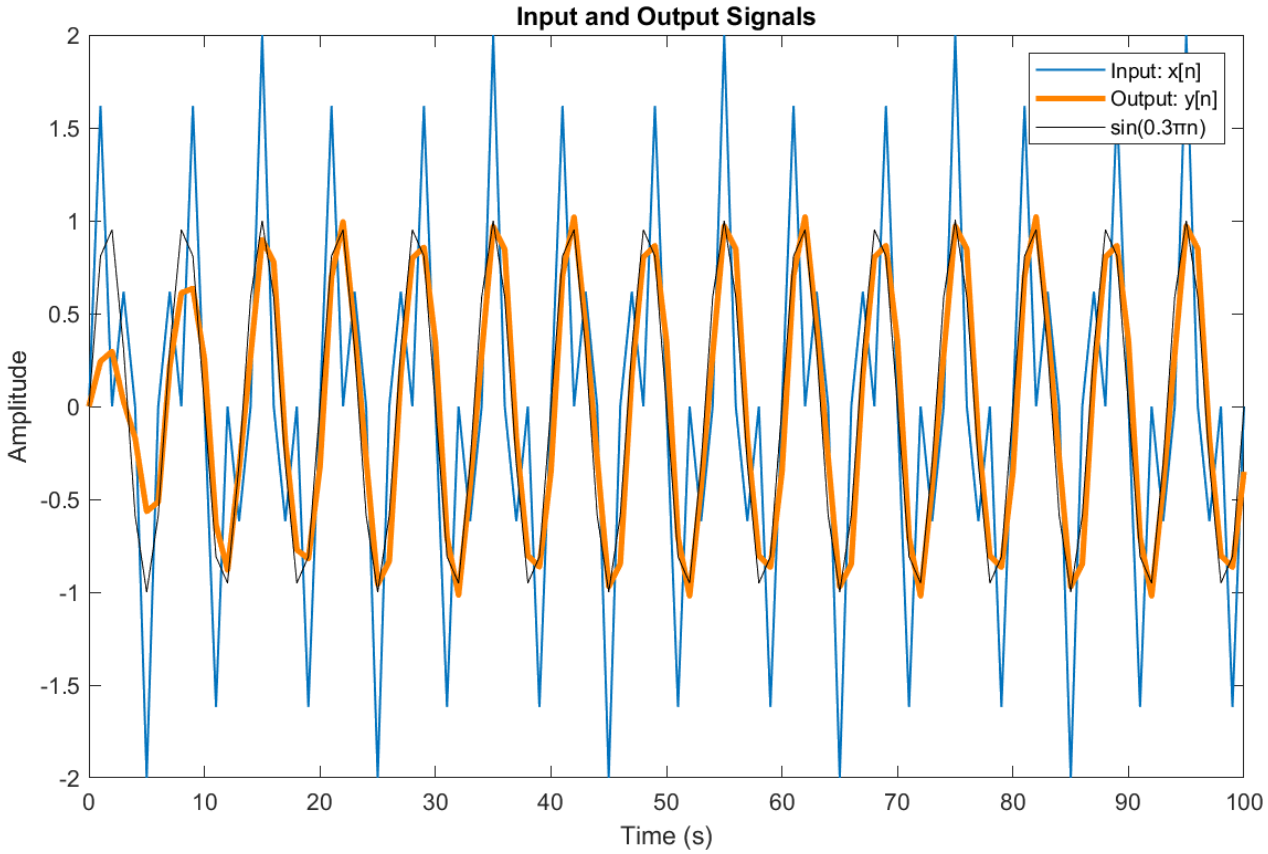
$$T_1 = \frac{2\pi}{\omega_1} = \frac{2\pi}{0.3\pi} = \frac{20}{3} sec \qquad T_2 = \frac{2\pi}{\omega_2} = \frac{2\pi}{0.7\pi} = \frac{20}{7} sec$$

```
% gensig syntax:
%(type of signal, period of signal, duration of signal, distance
% of successive steps)
x = gensig("sine",20/3,100,1)+ gensig("sine",20/7,100,1);
```

Έχουμε το σύστημα του ερωτήματος (Α) δηλαδή έχουμε τους πόλους $\{0.51 \pm 0.68 \cdot i\}$ και τα μηδενικά $\{0.8, -1\}$.

Για να βρούμε την απόκριση του συστήματος-φίλτρου θα χρησιμοποιήσουμε την συνάρτηση $\boldsymbol{filter}()$, η οποία δέχεται ως παράμετρο την είσοδο και τους συντελεστές της συνάρτησης μεταφοράς και επιστρέφει την απόκριση του φίλτρου.

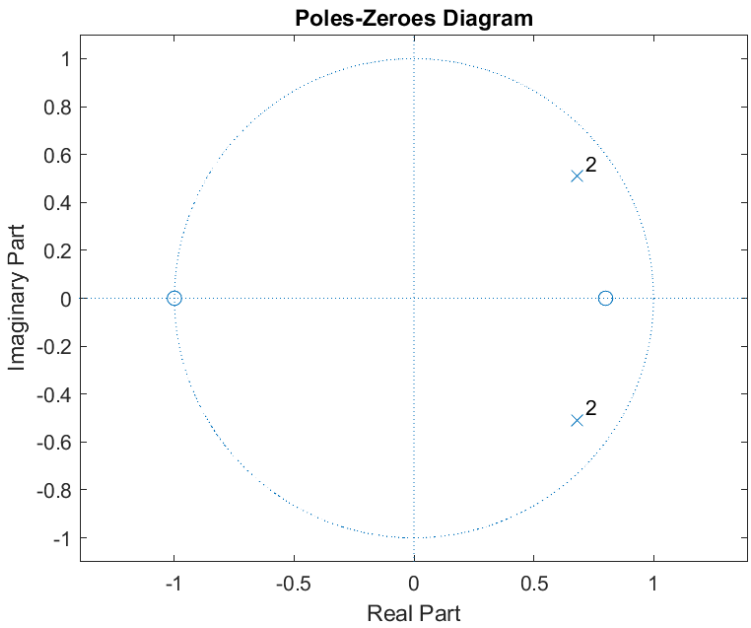
```
% Filtered Output Signal
y = filter(num,den,x);

% Plotting the input and output signals
figure
plot(0:1:100,x);
hold on
plot(0:1:100,y,'LineWidth',2,'Color',[1 0.513 0]);
plot(0:1:100,gensig("sine",20/3,100,1),"Color",[0 0 0])
hold off
xlabel('Time (s)');
ylabel('Amplitude');
legend('Input: x[n]','Output: y[n]', 'sin(0.3πn)');
title("Input and Output Signals");
```



Παρατηρήσεις:

Σχεδιάζουμε και το σήμα $\sin(0.3\pi n)$ καθώς η απόκριση του συστήματος ταυτίζεται σχεδόν με αυτό το σήμα. Αυτό είναι κατανοητό διότι το φίλτρο του ερωτήματος (Α) έχει κεντρική συχνότητα 0.3π που ισούται με την γωνιακή συχνότητα του σήματος $\sin(0.3\pi n)$. Ωστόσο, η συνιστώσα με συχνότητα $\omega = 0.7\pi$ αποκόπτεται και επομένως από το σήμα εισόδου παραμένει μονό το $\sin(0.3\pi n)$.

ΣΤ) Για διπλούς πόλους $\{0.68 \pm 0.51\}$ και μηδενικά $\{0.8, -1\}$ σχεδιάζουμε το διάγραμμα πόλων-μηδενικών.

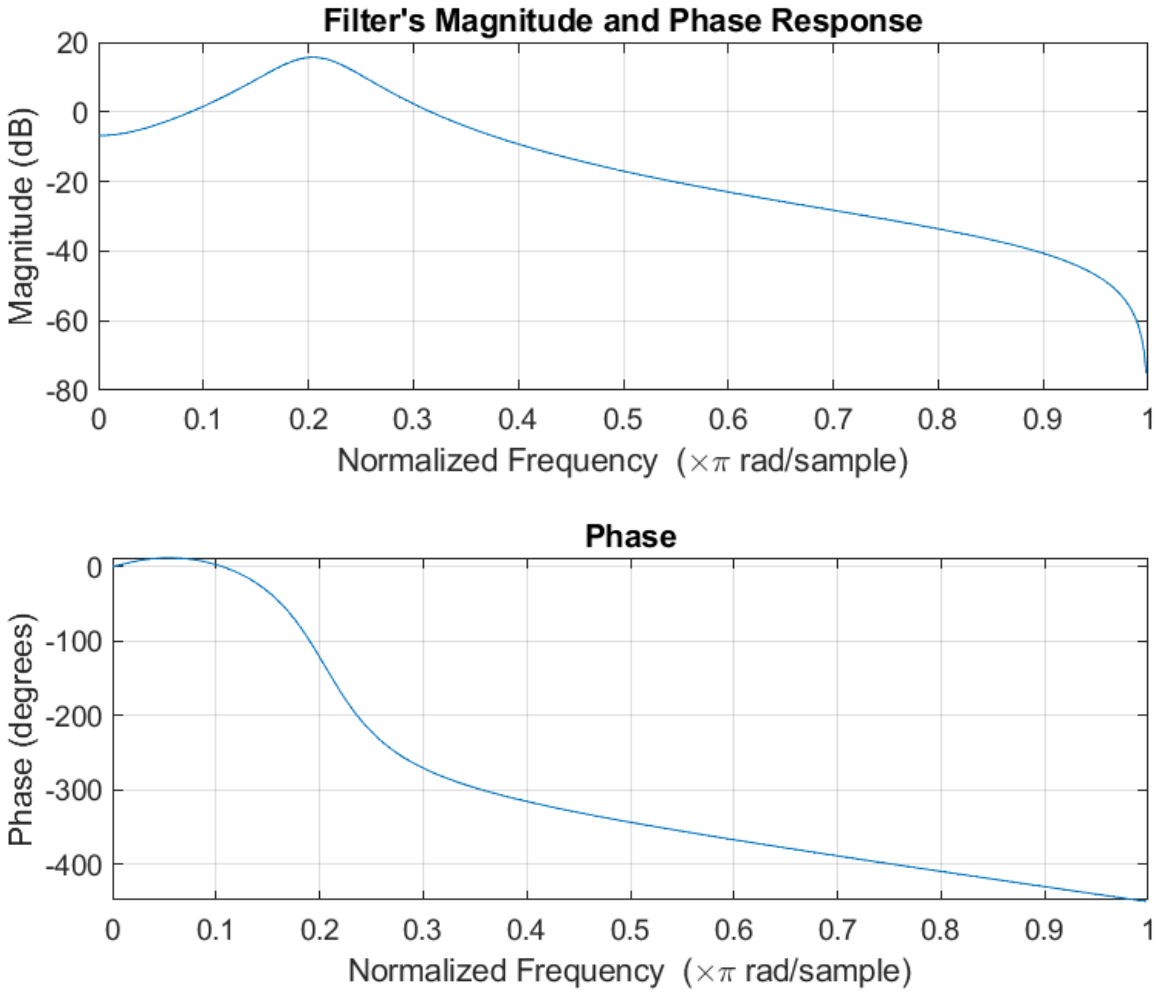


Υπολογίζουμε τους συντελεστές a_i και b_i.

```
% Calculating the vector coefficients a and b
[num, den] = zp2tf(zeroes,poles,K);
```

Σχεδιάζουμε την απόκριση πλάτους και φάσης του φίλτρου.

```
% Plotting the Amplitute and Phase Response of the filter
figure
freqz(num,den);
title('Filter''s Magnitude and Phase Response')
pause(2);
```

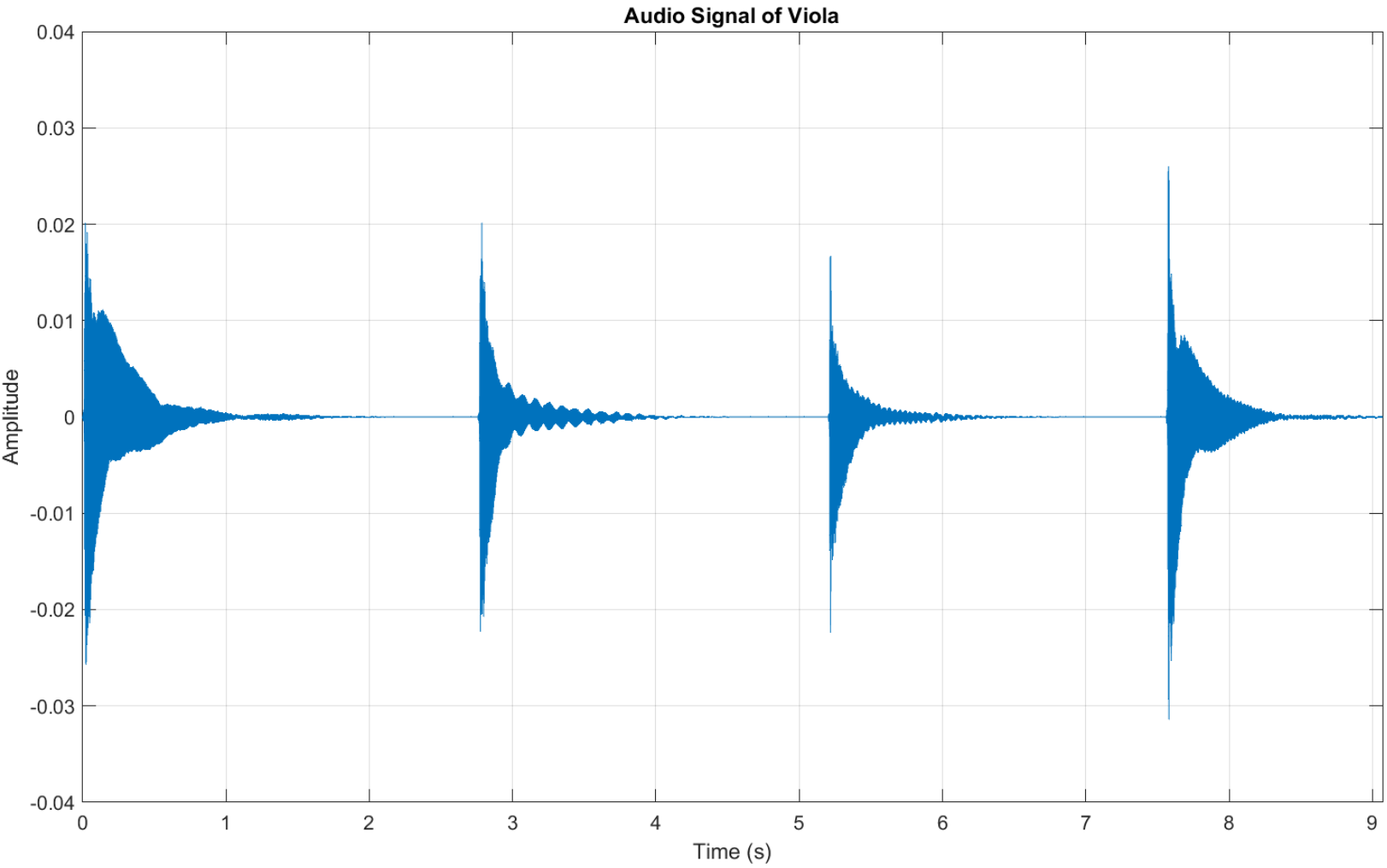


Παρατηρήσεις: Παρατηρούμε πως η αύξηση της πολλαπλότητας των πόλων στο μιγαδικό επίπεδο στενεύει την ζώνη διέλευσης του φίλτρου καθώς βλέπουμε την γραφική παράσταση της απόκρισης πλάτους να γίνεται πιο μυτερή στο σημείο της κεντρικής συχνότητας.

Άσκηση 3.2: Εφαρμογή Φίλτρων σε Μουσικά Σήματα

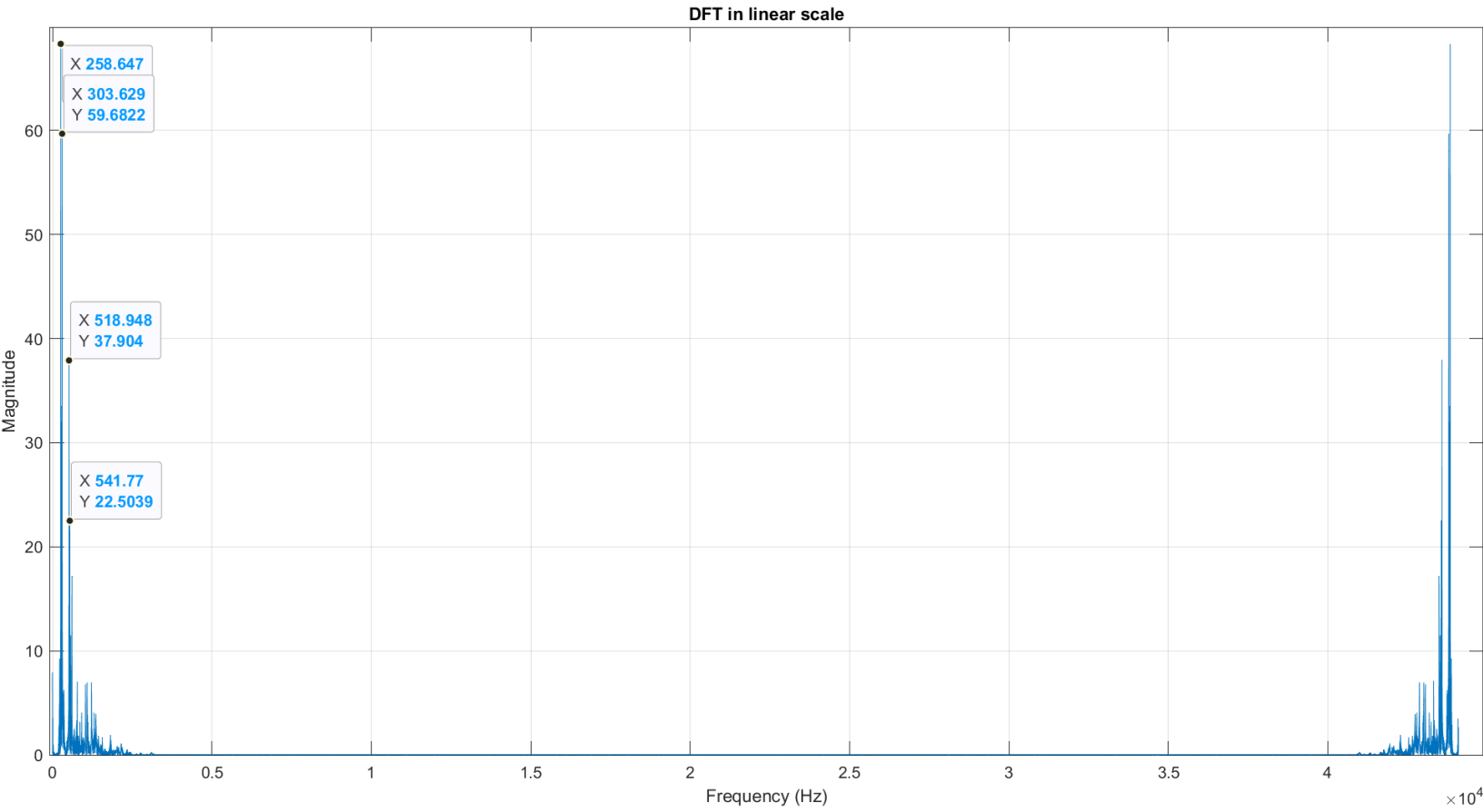
(Α) Αρχικά, φορτώνουμε το αρχείο *viola_series.wav* με την εντολή $\boldsymbol{audioread}()$**.** Στη συνέχεια, σχεδιάζουμε το σήμα μας με την εντολή $\boldsymbol{plot}()$ και ακούμε το σήμα με την εντολή $\boldsymbol{sound}()$**.** Ωστόσο, δεν γνωρίζουμε την διάρκεια του σήματος ώστε να ορίσουμε το διάνυσμα χρόνου. Με σκοπό την επίλυση του προβλήματος αυτού χρησιμοποιούμε την συνάρτηση $\boldsymbol{audioinfo}()$ και αποθηκεύουμε όλες τις πληροφορίες του σήματος στη μεταβλητή `info_series`. Από όλες τις πληροφορίες που έχουμε για το σήμα εμείς εξάγουμε την διάρκεια του σήματος ως εξής: `info_series.Duration`.

```
--------------------------------------------------------------------------
% Reading and saving the audio file to the variable viola_series
[viola_series, fs_viola_series] = audioread('viola_series.wav');
info_series = audioinfo('viola_series.wav');

% Initializing a vector which is going to represent time from 0-size
seconds
time_viola_series = linspace(0,info_series.Duration,
length(viola_series));

% plotting the audio signal
figure
plot(time_viola_series,viola_series);
grid on;
xlabel('Time (s)');
ylabel('Amplitude');
axis([0 info_series.Duration -0.04 0.04]);
title('Audio Signal of Viola');
pause(2);

% Listening to the audio
sound(viola_series,fs_viola_series);
pause(9);
--------------------------------------------------------------------------
```



(B) Εφαρμόζουμε DFT στο σήμα μας και στη συνέχεια σχεδιάζουμε την γραφική παράσταση του μέτρου του φάσματος σε γραμμική κλίμακα.

```
---------------------------------------------------------------------------
% Applying the Discrete Fourier Transform (DFT) at our cut signal
DFT_viola = fft(viola_series);
Freq = linspace(0,fs_viola_series,length(DFT_viola));

% Plotting the DFT in linear scale
figure
plot(Freq,abs(DFT_viola));
grid on
xlabel('Frequency (Hz)');
ylabel('Magnitude');
title('DFT in linear scale');
---------------------------------------------------------------------------
```



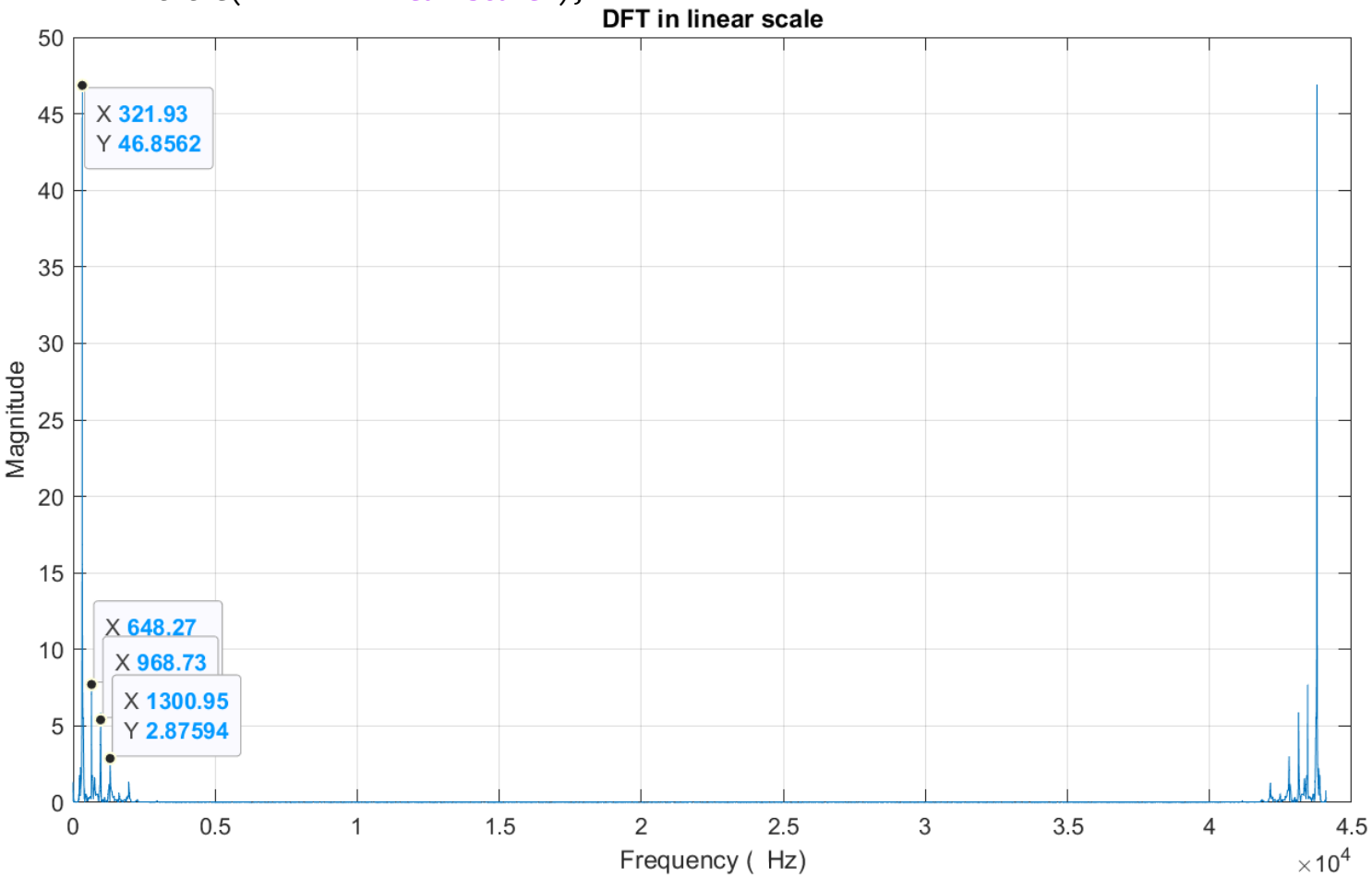
Σχολιασμός:

- Οι παραπάνω συχνότητες που επικρατούν αποτελούν τις συχνότητες των τεσσάρων νοτών που υπάρχουν στο σήμα μας.
- Όσο αυξάνεται η συχνότητα το πλάτος φθίνει στο γράφημά μας.

(Γ) Φορτώνουμε το αρχείο *viola_note.wav* και σχεδιάζουμε το μέτρο φάσματος κατά τα γνωστά.

```
% Load the signal
[viola_note, fs_viola_note] = audioread('viola_note.wav');

DFT_viola_note = fft(viola_note);
viola_Frequency = linspace(0,fs_viola_note,length(viola_note));

% Plotting the DFT
figure
plot(viola_Frequency,abs(DFT_viola_note));
grid on
xlabel('Frequency (kHz)');
ylabel('Magnitude');
```

```
title('DFT in linear scale');
```



Βλέπουμε πως το μέτρο φάσματος του σήματός μας έχει spikes στις αρμονικές. Συγκεκριμένα, παρατηρούμε πως όσο προχωράνε οι αρμονικές τόσο μικρότερα είναι τα spikes του φάσματος. Συγκριτικά με το προηγούμενο γράφημα μέτρου φάσματος παρατηρούμε πως έχουν παρόμοια μορφή και σχήμα.

Για τον υπολογισμό της θεμελιώδης συχνότητας του σήματος θα βρούμε για ποια συχνότητα το μέτρο φάσματος γίνεται μέγιστο. Για τον σκοπό αυτό θα αξιοποιήσουμε την εντολή $\boldsymbol{max}()$ του MATLAB. Συγκεκριμένα, θα βρούμε την θέση του πίνακα `viola_Frequency` όπου το μέτρο φάσματος γίνεται μέγιστο και κατόπιν θα επικαλεστούμε αυτή την τιμή ως την θεμελιώδη συχνότητα του σήματός μας.

```
% Computing the index that corresponds to the fundamental
% frequency in the viola_Frequency vector
[~,I] = max(abs(DFT_viola_note));

% Find the fundamental frequency while knowing the index
fundamental_frequency = viola_Frequency(I);
```

(Δ) Για την υλοποίηση ενός ζωνοπερατού φίλτρου όπου απομονώνει την 2η αρμονική του σήματος θα χρησιμοποιήσω την συνάρτηση $\boldsymbol{butter}()$, η οποία χρησιμοποιείται για την σχεδίαση ψηφιακών φίλτρων Butterworth. Φίλτρα Butterworth είναι ένας τύπος φίλτρου άπειρης παλμικής απόκρισης (IIR) και μπορούν να σχεδιάσουν ως φίλτρο χαμηλής διέλευσης, φίλτρο υψηλής διέλευσης ή ζωνοπερατά φίλτρα. Η σύνταξη της συνάρτησης $\boldsymbol{butter}()$ για την σχεδίαση ζωνοπερατού φίλτρου είναι η ακόλουθη.

```
[z,p,~] = butter(n,Wn,'bandpass');
```

- Wn: οι κανονικοποιημένες συχνότητες αποκοπής. Συγκεκριμένα, αποτελεί ένα διάνυσμα με 2 στοιχεία, το πρώτο αντιπροσωπεύει την κάτω άκρη της ζώνης συχνοτήτων και το δεύτερο αντιπροσωπεύει την πάνω άκρη της ζώνης συχνοτήτων.
- n: Η τάξη του φίλτρου, η οποία καθορίζει πόσο απότομα αποκόπτει τις συχνότητες εκτός του πεδίου.
- z: Τα μηδενικά του φίλτρου
- p: Οι πόλοι του φίλτρου

Στη περίπτωσή μας επιλέγουμε την τάξη του φίλτρου να είναι 3 (σχετικά απότομη αποκοπή) και το $Wn = [2 \cdot f_0 - \frac{bandwidth}{2}, 2 \cdot f_0 + \frac{bandwidth}{2}]$. Ορίζοντας το $bandwidth = 10$ έχουμε:

```
% width of the passband around the second harmonic
bandwidth = 20;

% define the range of the second harmonic
fmin = harmonics - bandwidth/2; % lower cutoff frequency
fmax = harmonics + bandwidth/2; % upper cutoff frequency

% Design the poles and zeroes of the filter
Wn = [fmin fmax]/(fs_viola_note/2);
% Normalized cutoff frequencies
[z,p,~] = butter(3,Wn,'bandpass');
```

Με `harmonics = 2*fundamental_Frequency;`

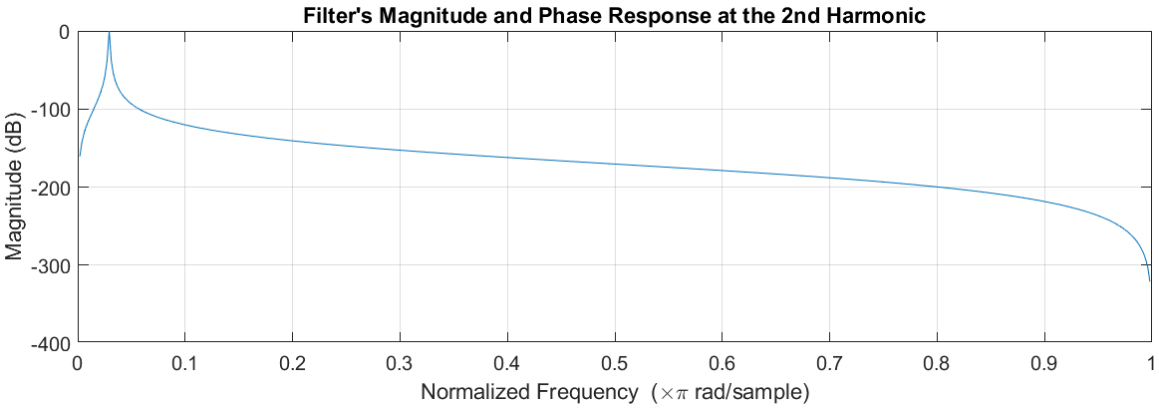
Στη συνέχεια, έχοντας βρει τους πόλους και τα μηδενικά του φίλτρου μας χρησιμοποιούμε την εντολή $\boldsymbol{zp2tf}()$ για να βρούμε τους συντελεστές της συνάρτησης μεταφοράς.

```
% Convert the poles and zeroes to the numerator and denominator
% polynomials coefficients accordingly
[num,den] = zp2tf(z,p,K(index));
```

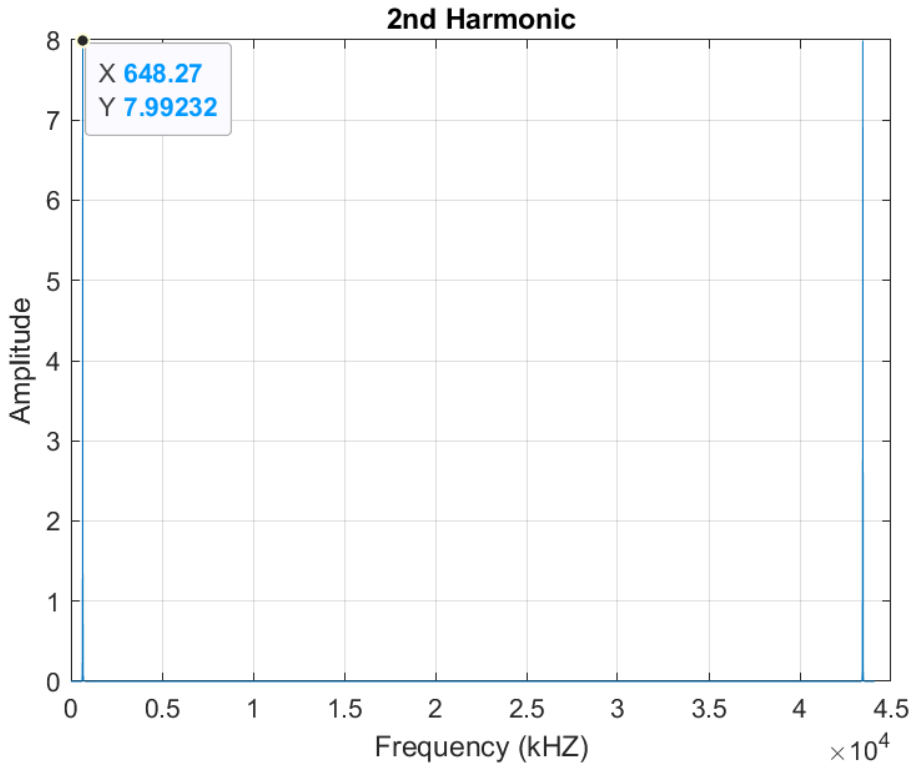
Για την παράμετρο K παρατηρούμε με trial and error πως όταν παίρνει την τιμή $K = 3 \cdot 10^{-9}$ τότε το κέρδος του φίλτρου είναι πολύ κοντά στα 0dB, όπως φαίνεται και από το παρακάτω διάγραμμα απόκρισης πλάτους.

Έπειτα, έχοντας φτιάξει το φίλτρο μας το εφαρμόζουμε στο αρχικό μας σήμα, με σκοπό να απομονώσομε την δεύτερη αρμονική.

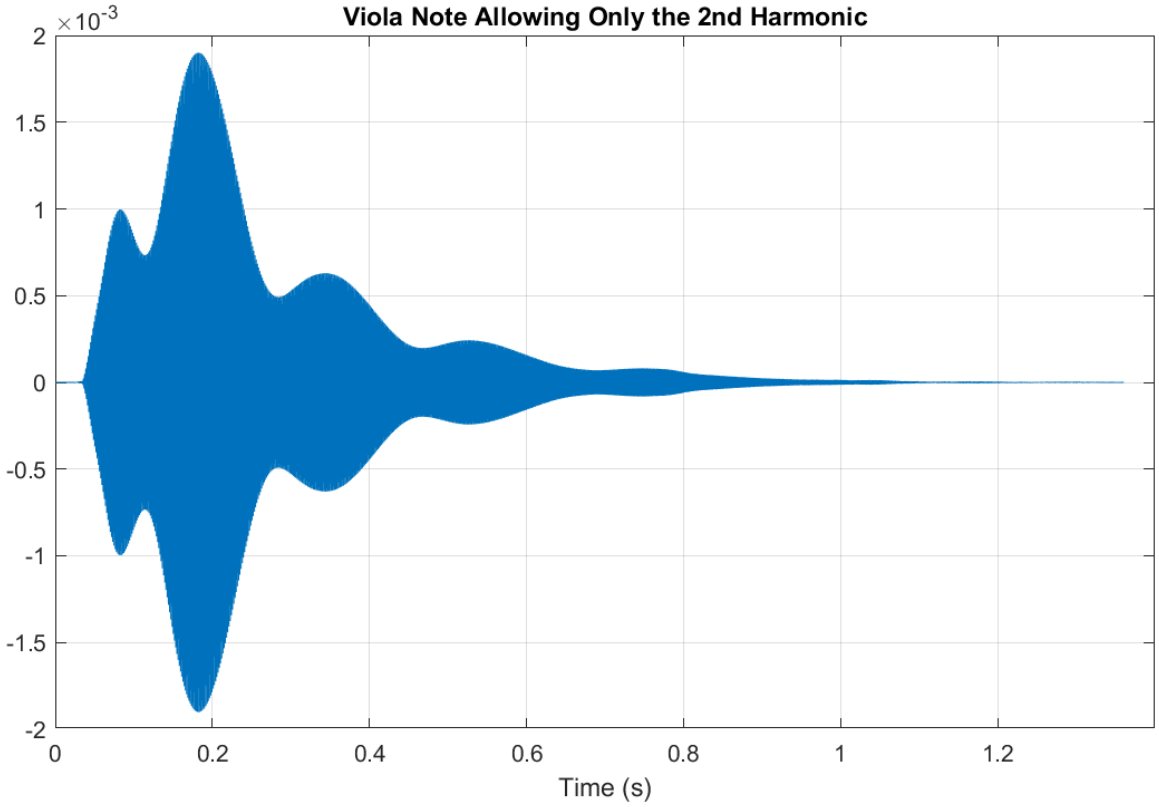
```
% Apply the filter to the signal
y_harmonic = filter(num,den,viola_note);
```



Παίρνουμε τώρα τον DFT του φιλτραρισμένου σήματος και παρατηρούμε πως η μόνη συχνότητα που κυριαρχεί είναι η δεύτερη αρμονική του σήματος.

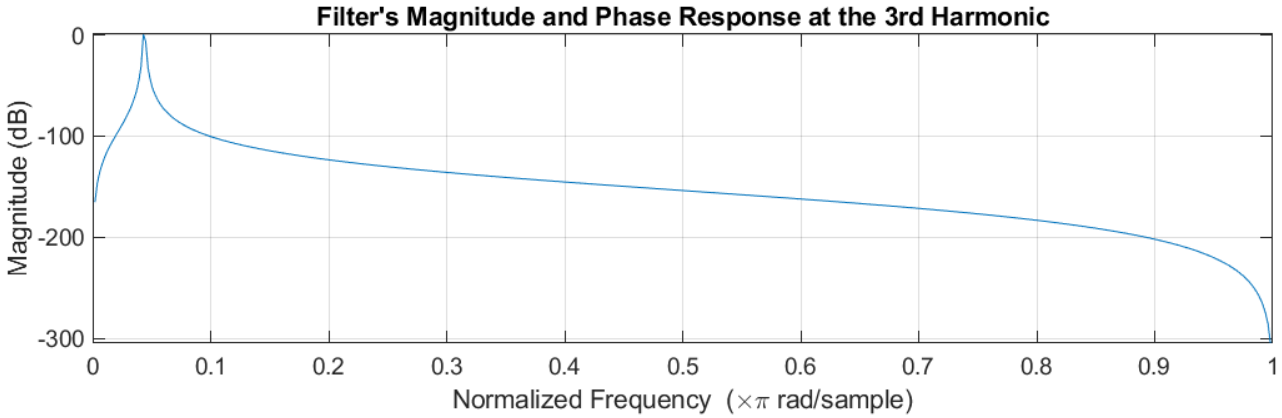


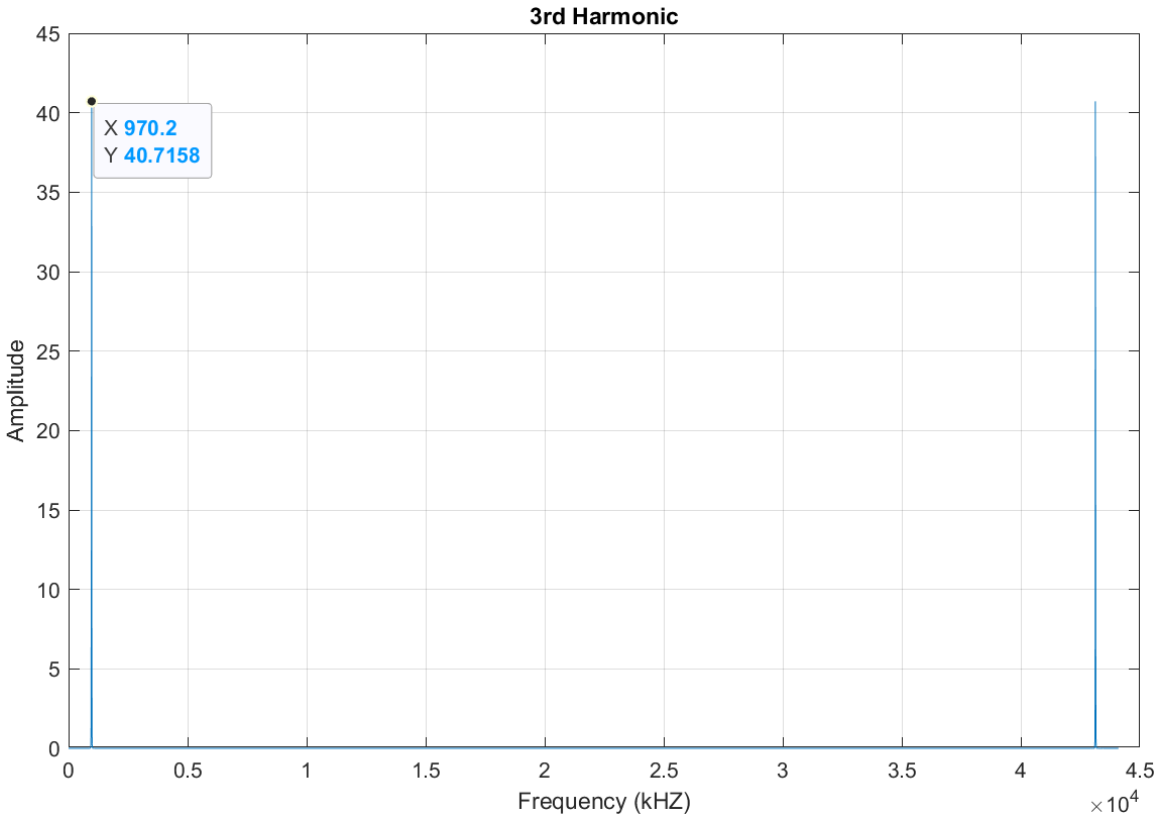
Τέλος, σχεδιάζουμε την γραφική παράσταση του φιλτραρισμένου σήματος.

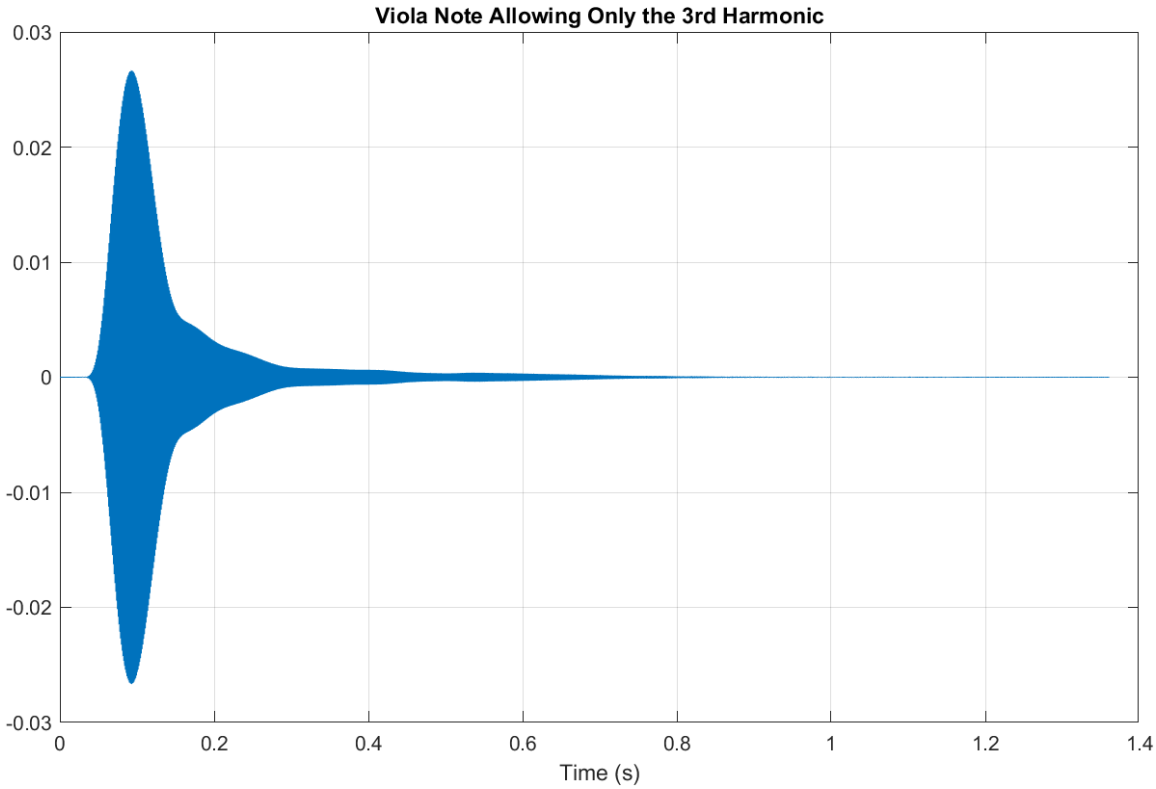


Υποσημείωση: Γνωρίζω πως η μεθοδολογία στην οποία μας κατευθύνει η υπόδειξη του ερωτήματος είναι να χρησιμοποιήσουμε τις παρατηρήσεις μας από την άσκηση 3.1, δηλαδή πως η κεντρική γωνιακή συχνότητα αντιστοιχεί στην γωνία τον πόλων στο μιγαδικό επίπεδο. Η κεντρική συχνότητα στο ζωνοπερατό φίλτρο που θέλουμε να κατασκευάσουμε αντιστοιχεί με την δεύτερη αρμονική του σήματος κανονικοποιημένο στο διάστημα [0,2π], δηλαδή $\frac{2\pi(2\cdot f_0)}{fs}$ (βγαίνει περίπου ίση με 0.0293π) και επιλέγουμε το μέτρο των πόλων να είναι περίπου ίσο με 1 και φυσικά μικρότερο ή ίσο του 1 ώστε το φίλτρο να είναι ευσταθές (γενικά όσο πιο κοντά είναι οι πόλοι στο μοναδιαίο κύκλο τόσο καλύτερα αποκόπτουμε ανεπιθύμητες συχνότητες). Παράλληλα, μπορούμε να πάρουμε ως μηδενικά το τριπλό ζεύγος [-1 ,1] (κοντά στο μοναδιαίο κύκλο και με μεγάλη πολλαπλότητα για να ελαττώσουμε τις άλλες συχνότητες) και το κόστος K θα το ρυθμίζαμε κατάλληλα στη συνέχεια (δηλαδή θα ελέγχαμε διάφορες τιμές και θα βλέπαμε στην απόκριση πλάτους αν το κέρδος του φίλτρου είναι 0dB). Ωστόσο, η διαδικασία αυτή μου φάνηκε αρκετά χρονοβόρα και επίπονη, ενώ ταυτόχρονα είχα μελετήσει τα φίλτρα Butterworth. Συνεπώς, μου φάνηκε πολύ πιο απλή η μεθοδολογία αυτή. Ελπίζω να μην υπάρξει επίπτωση στη βαθμολογία για κάτι τέτοιο και ευχαριστώ πολύ για την κατανόηση.

Τα ίδια ακριβώς βήματα ακολουθούμε και για την απομόνωση της τρίτης αρμονικής. Αναλυτικότερα, βρίσκουμε πως η παράμετρος κέρδους K πρέπει να είναι $K = 2 \cdot 10^{-8}$ για να έχουμε κέρδος 0dB, ενώ παράλληλα το μέτρο του φάσματος και η γραφική παράσταση της τρίτης αρμονική είναι τα ακόλουθα:







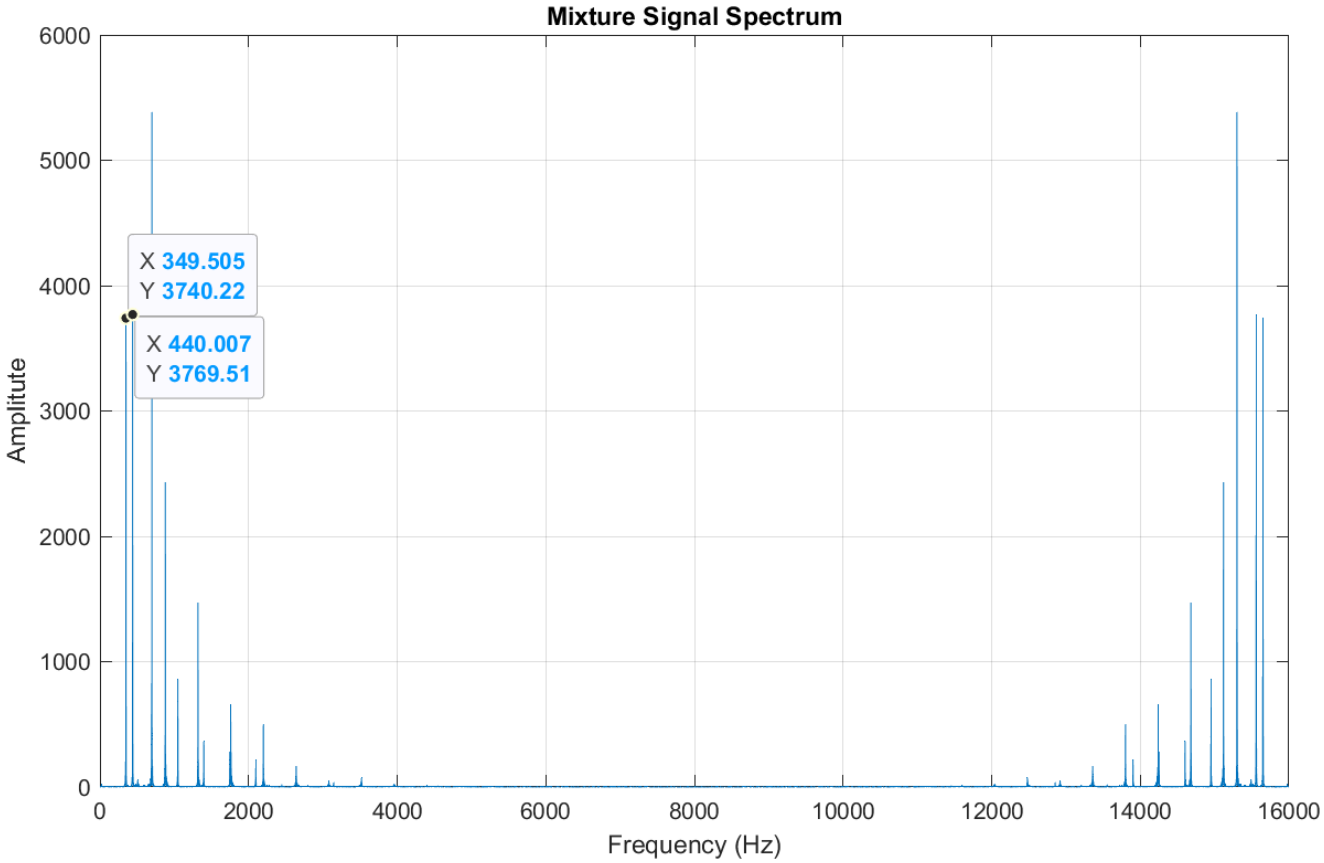
Άσκηση 4: Διαχωρισμός Μουσικών Νοτών

(Α) Αρχικά, φορτώνουμε το σήμα *mixture.wav* και στη συνέχεια ακούμε το σήμα κατά τα γνωστά.

```
% Load the audio file
[mixture, fs_mixture] = audioread('mixture.wav');
info_mixture = audioinfo("mixture.wav");

% Listening to the audio
sound(mixture);
```

Κατόπιν σχεδιάζουμε το μέτρο φάσματος του σήματος με την χρήση της εντολής $\boldsymbol{fft}()$.

```
% Taking the DFT of the signal
mixture_DFT = fft(mixture);
mixture_frequency = linspace(0,fs_mixture,length(mixture));

% Plotting the fft
figure
plot(mixture_frequency,abs(mixture_DFT));
xlabel('Frequency (Hz)')
ylabel('Amplitute');
title("Mixture Signal Spectrum");
```



Παρατηρούμε τις δύο θεμελιώδεις συχνότητες από τις οποίες αποτελείται το σήμα μας στο διάγραμμα μέτρου φάσματος. Έπειτα, εμφανείς είναι και οι αρμονικές των αντίστοιχων θεμελιωδών συχνοτήτων.

(Β) Εάν έχουμε ένα σήμα με θεμελιώδη συχνότητα f_0 τότε η n^{th} αρμονική του σήματος είναι $f_n = n * f_0$. Επομένως, για την πρώτη και την δεύτερη θεμελιώδη συχνότητα του σήματός μας οι αντίστοιχες αρμονικές θα είναι:

```
First five harmonics for the first note:
        350         700        1050        1400        1750

First five harmonics for the second note:
        440         880        1320        1760        2200
```

Αν η συχνότητα δειγματοληψίας είναι f_s και θέλουμε να κανονικοποιήσουμε την n^{th} αρμονική στο διάστημα $[0,2\pi]$ τότε: $f_{normalized} = \frac{2\pi \cdot f_n}{f_s}$

Άρα έχουμε:

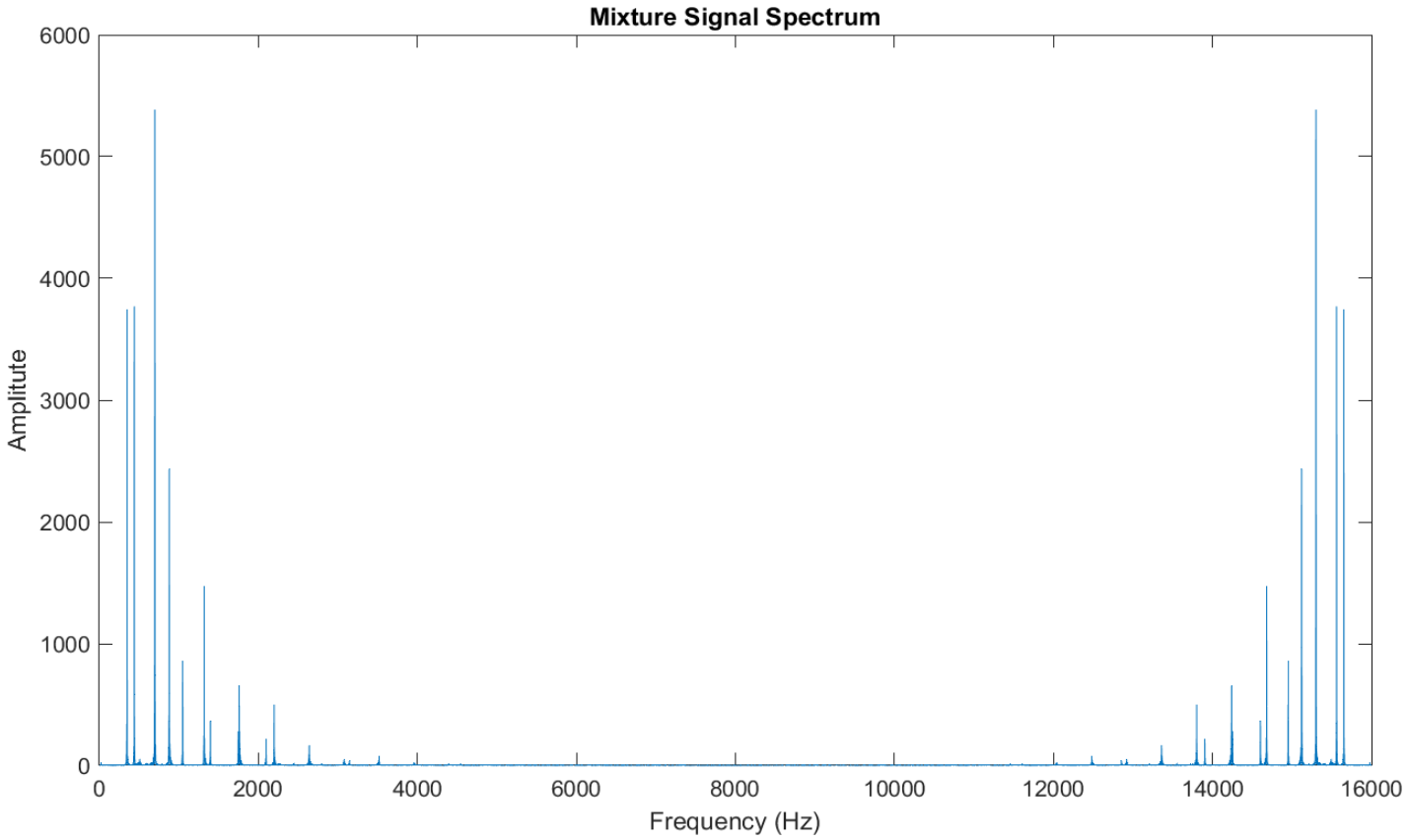
```
Normalized frequencies for the first note:
    0.1374    0.2749    0.4123    0.5498    0.6872

Normalized frequencies for the first note:
    0.1728    0.3456    0.5184    0.6912    0.8639
```

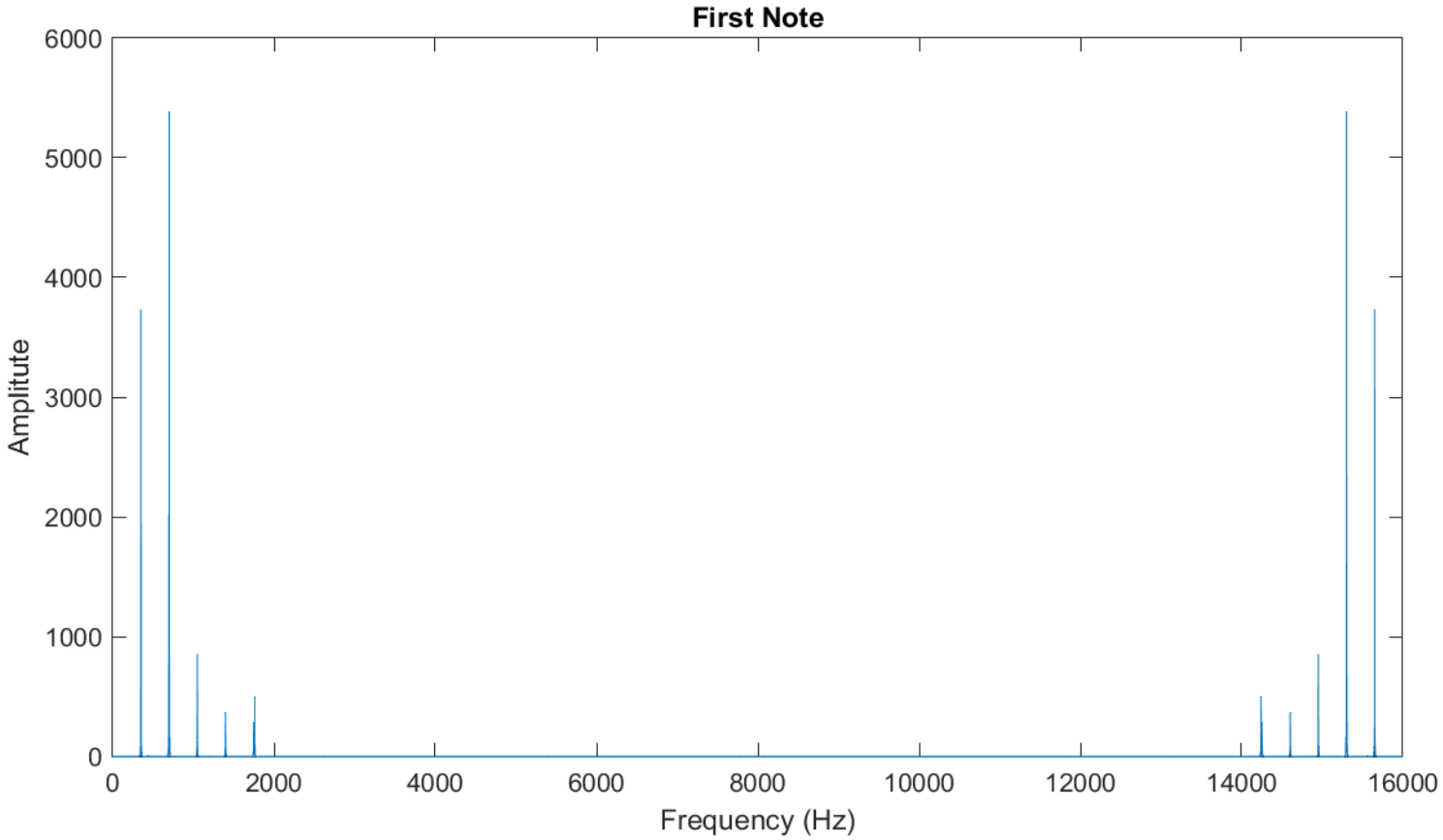
(Γ) Για την υλοποίηση ζωνοπερατού φίλτρου με κέρδος ζώνης διέλευσης ίσο με 0dB με σκοπό την απομόνωση αρμονικών συχνοτήτων θα ακολουθήσουμε την ίδια μεθοδολογία που ακολουθήσαμε και στην άσκηση (3.2Δ) με τα φίλτρα Butterworth. Για το πρώτο σήμα με θεμελιώδη συχνότητα 350Hz, έχοντας απομονώσει τις πρώτες 5 αρμονικές, τις προσθέτουμε για να κατασκευάσουμε το πρώτο μας σήμα. Την ίδια διαδικασία ακολουθούμε και για την δεύτερη θεμελιώδη συχνότητα 440Hz.

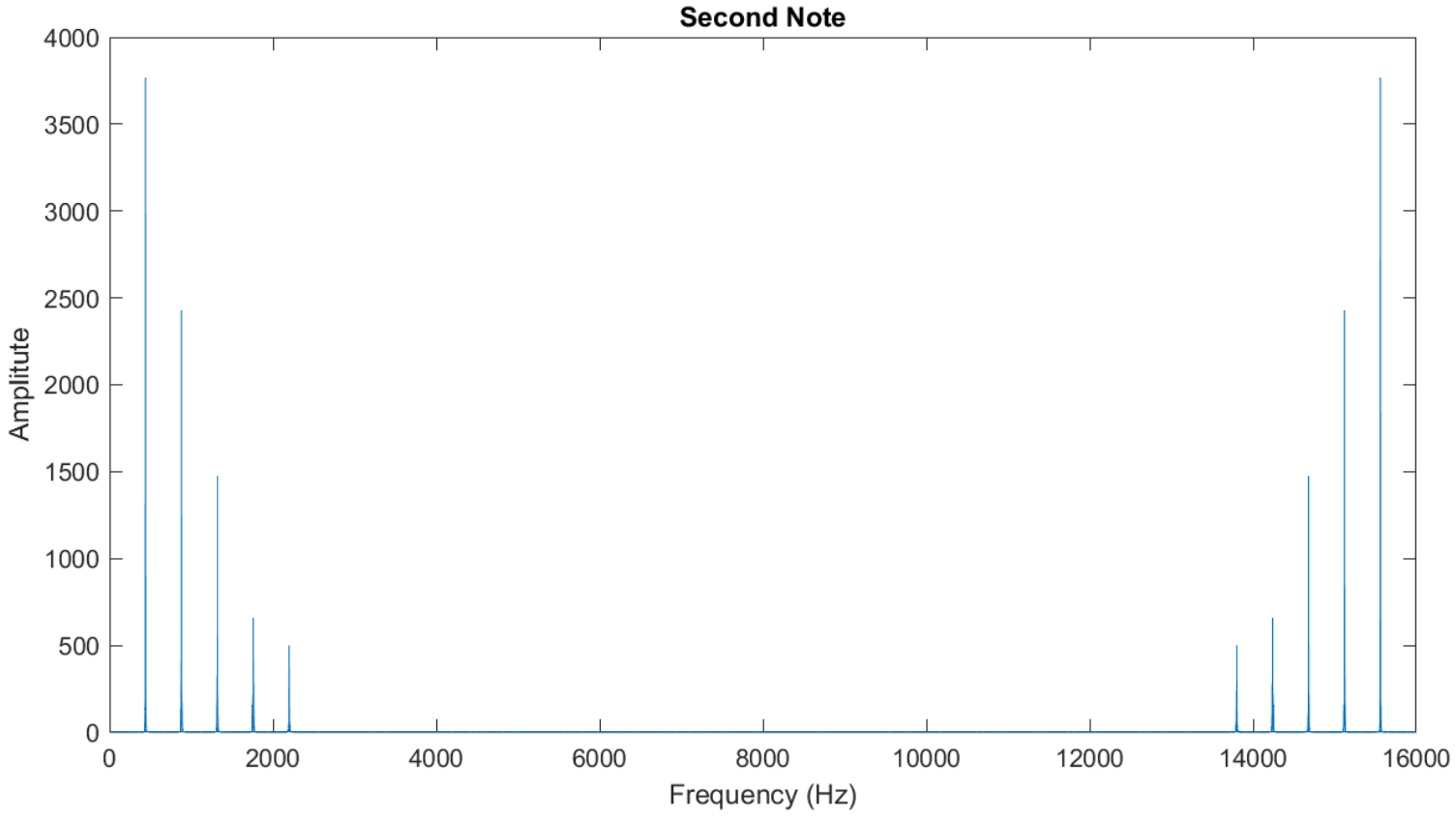
Σημείωση: Παρατηρούμε πως για $K = 6 \cdot 10^{-8}$ το κέρδος ζώνης διέλευσης είναι ίσο με 0dB για κάθε φίλτρο.

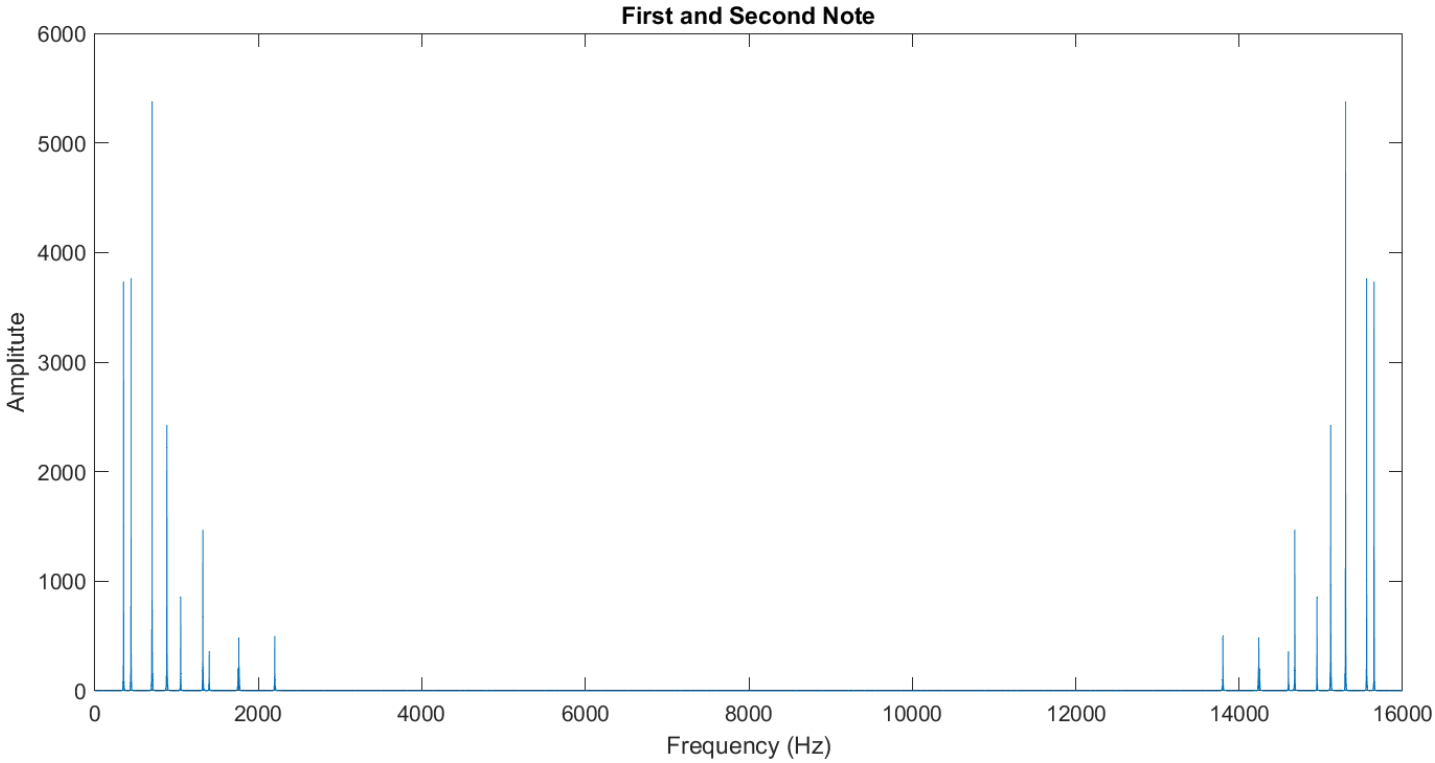
(Δ) ***Εποπτική αξιολόγηση:***

Αρχικά παίρνουμε τον DFT των δύο σημάτων και στη συνέχεια τα προσθέτω. Παρατηρούμε πως το μέτρο φάσματος του αρχικού σήματος σχεδόν ταυτίζεται με το μέτρο φάσματος του αθροίσματος του πρώτου και του δεύτερου μέτρου φάσματος (ειδικά στις αρχικές συχνότητες).



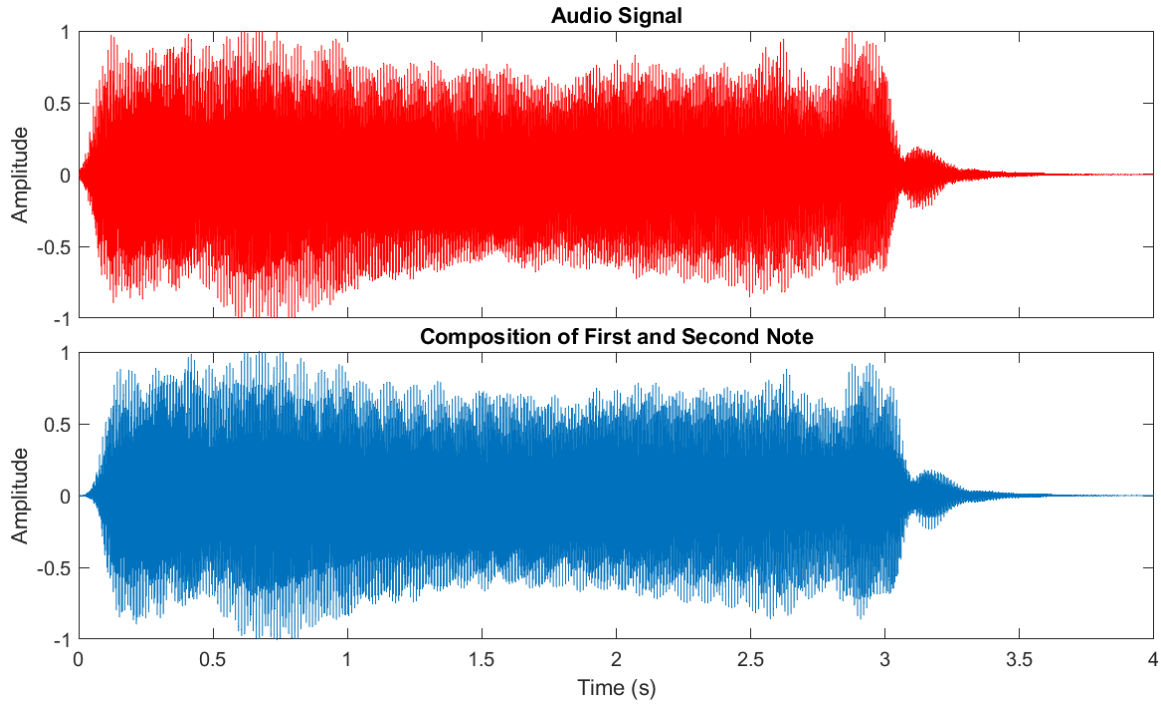

First Note
6000
5000
4000
3000
2000
1000
0
Amplitute
0
2000
4000
6000
8000
10000
12000
14000
16000
Frequency (Hz)


Second Note
4000
3500
3000
2500
2000
1500
1000
500
0
Amplitute
0
2000
4000
6000
8000
10000
12000
14000
16000
Frequency (Hz)

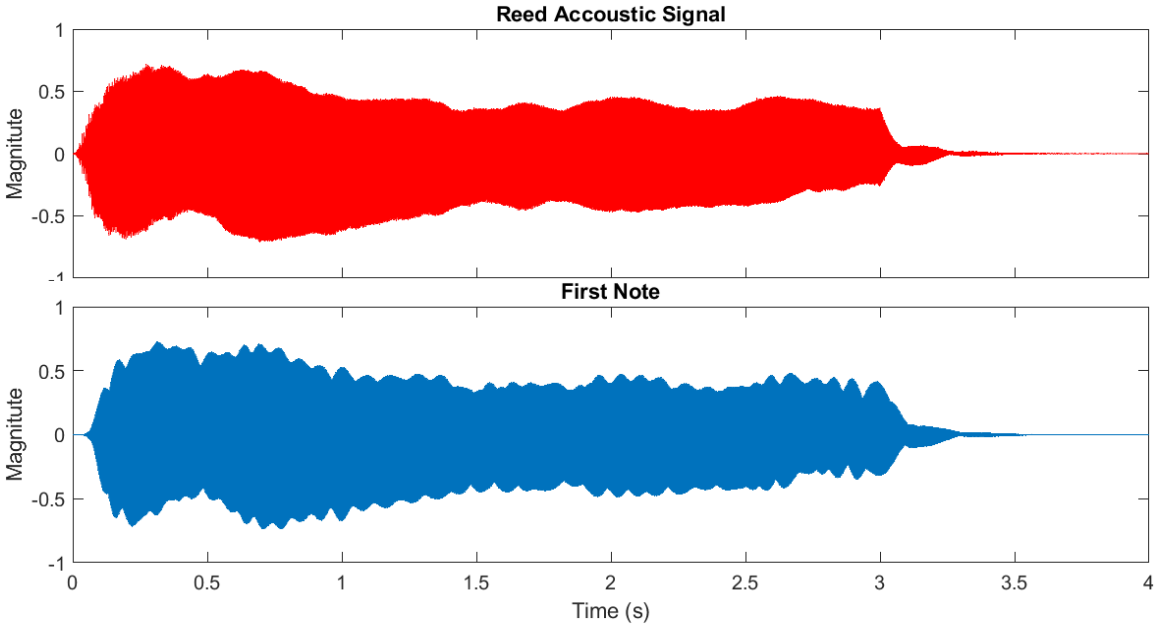


Παρατηρούμε πως το φάσμα της σύνθεσης των δύο απομονωμένων σημάτων μας δίνει σχεδόν το ίδιο φάσμα με το αρχικό μας σήμα. Ωστόσο, η διαφορά μεταξύ τους είναι πως στη σύνθεση δεν είναι εμφανείς οι αρμονικές τάξης μεγαλύτερης του 5.

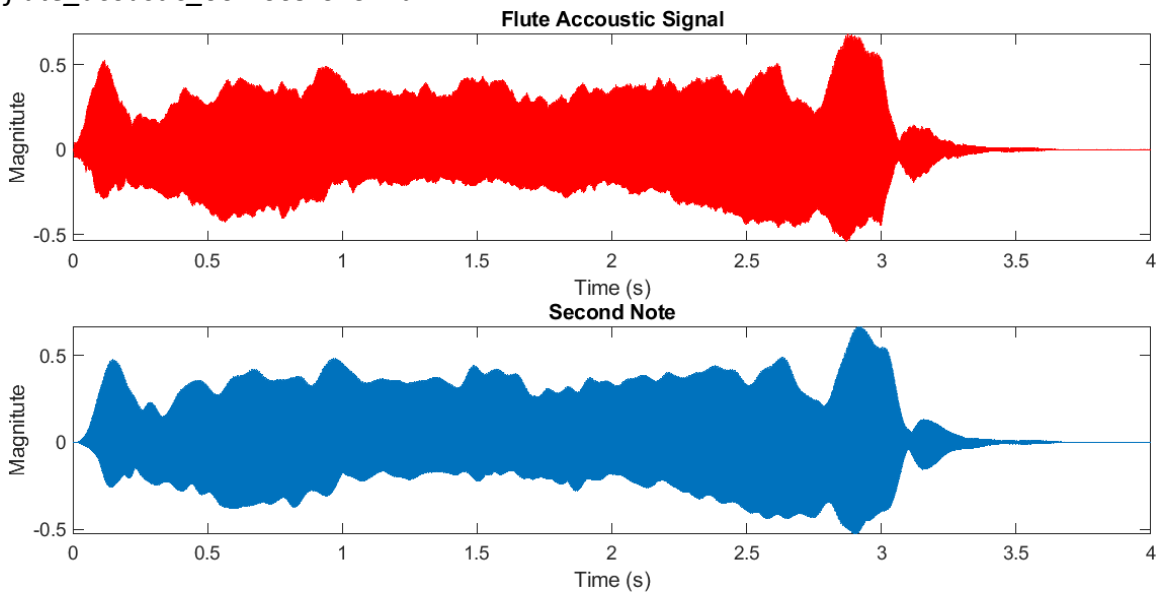
Στη συνέχεια, πλοτάρουμε το ανακατασκευασμένο σήμα με το αρχικό και τα συγκρίνουμε.



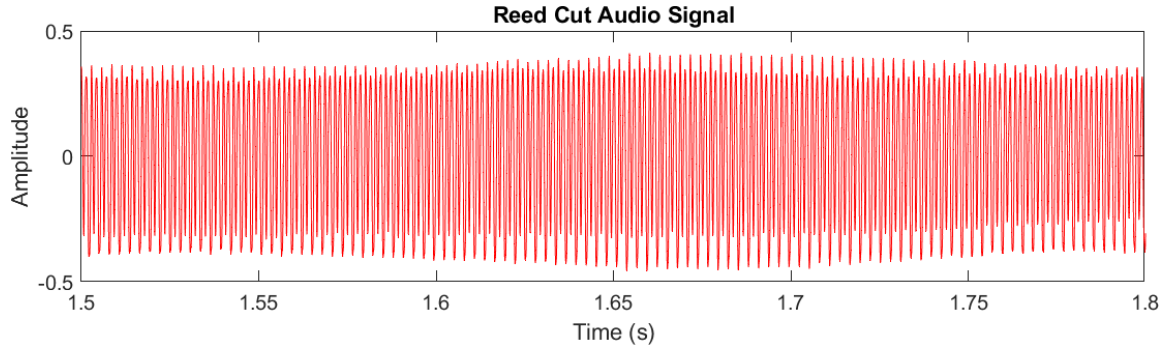
Παρατηρούμε πως το αρχικό μας σήμα και το σήμα που ανακατασκευάσαμε μέσα από τις 5 πρώτες αρμονικές των δύο θεμελιωδών περιόδων του αρχικού σήματος είναι σχεδόν πανομοιότυπες.
Έπειτα, συγκρίνουμε το σήμα με θεμελιώδη περίοδο 350Hz με το πηγαίο σήμα *reed_acoustic_037-057-127.wav.*

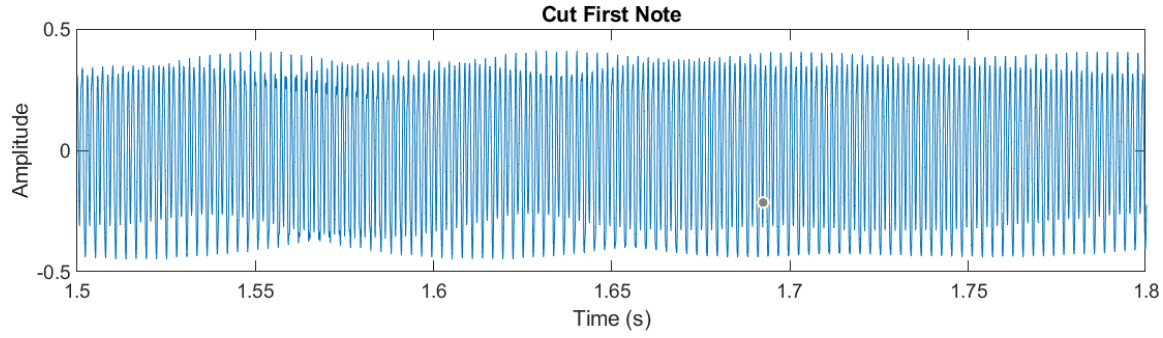


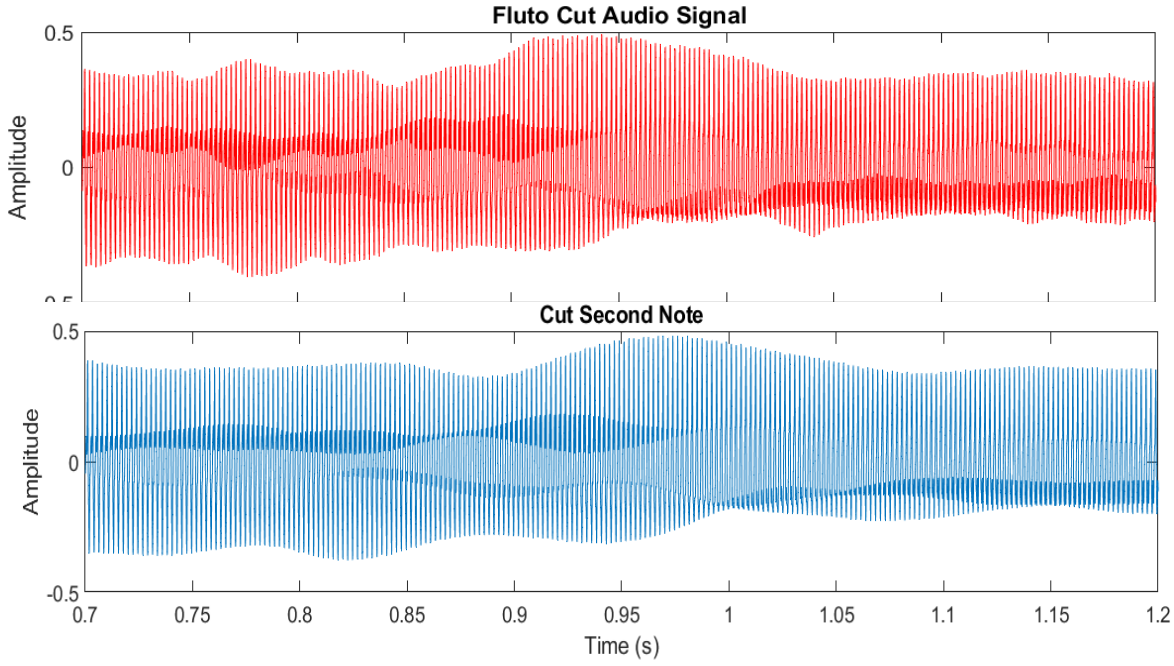
Παράλληλα συγκρίνουμε το σήμα με θεμελιώδη περίοδο 440Hz με το αρχείο ήχου *flute_acoustic_002-069-025.wav.*



Είναι προφανές πως και στις δυο περιπτώσεις τα σήματα είναι αρκετά όμοια. Ας επιβεβαιβώσουμε την ομοιότητα αυτή ακόμα περισσότερο συγκρίνοντας αποσπάσματα μεταξύ των σημάτων που φιλτράραμε και τα αντίστοιχα αρχεία ήχου.



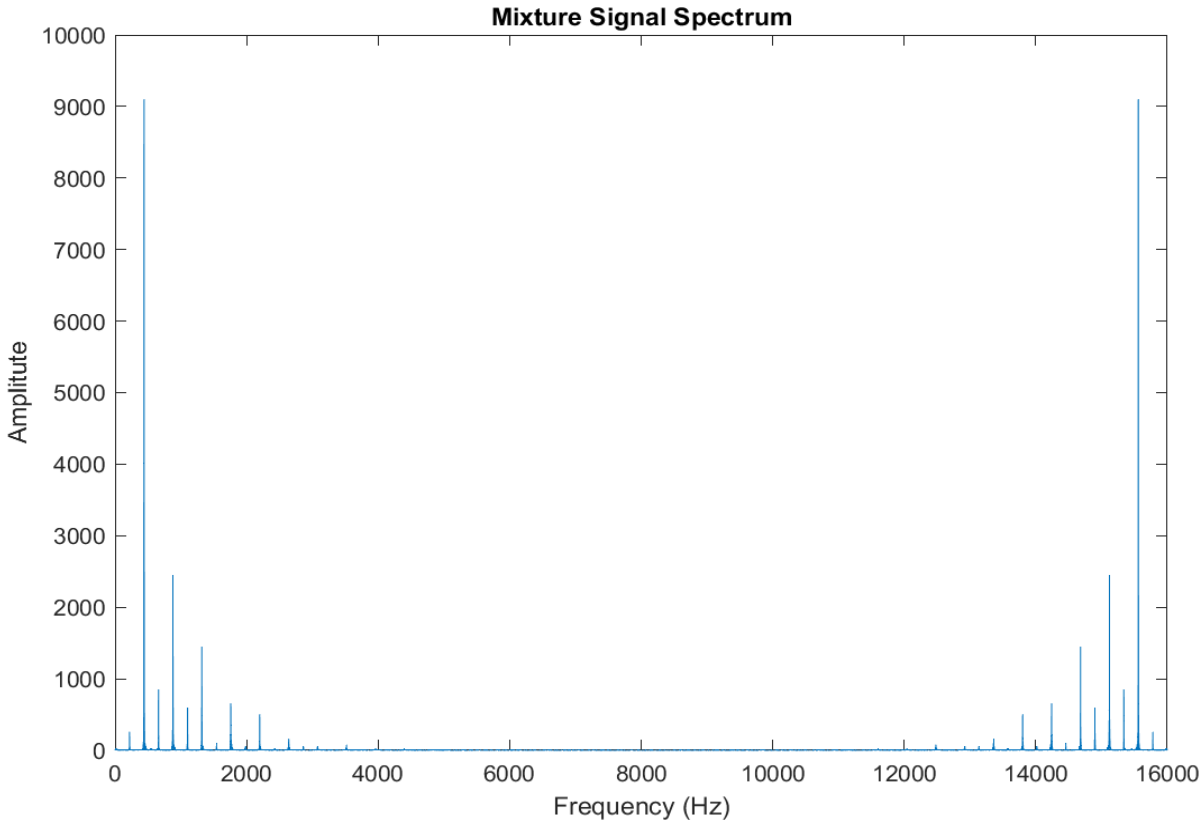




Ακουστική Επιβεβαίωση:

Μπορούμε να ακούσουμε το ανακατασκευασμένο σήμα με την εντολή ***sound*** και έτσι επιβεβαιώνουμε ακουστικά πως τα δύο αυτά σήματα μοιάζουν σε μεγάλο βαθμό.

(E) Έχοντας φορτώσει το αρχείο mixture2.wav ακούμε το σήμα με την εντολή ***sound*** με σκοπό να το συγκρίνουμε με το σήμα που θα ανακατασκευάσουμε. Στη συνέχεια παίρνουμε τον DFT του σήματος και πλοτάρουμε το μέτρο φάσματος.

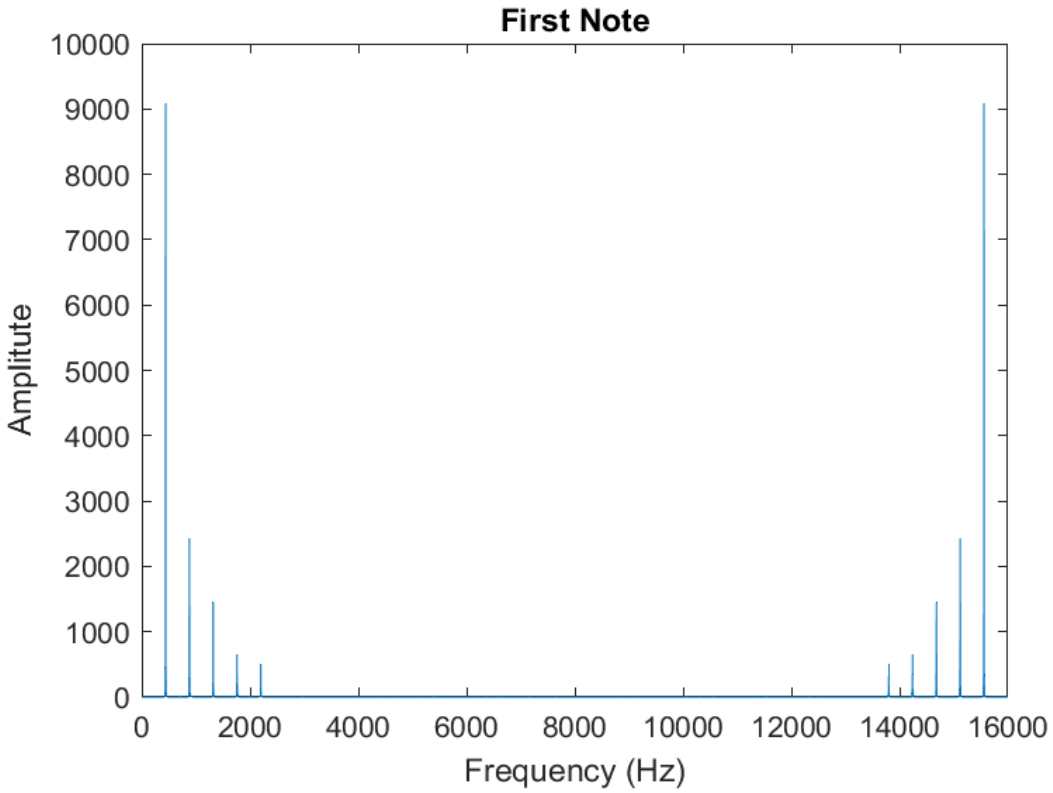


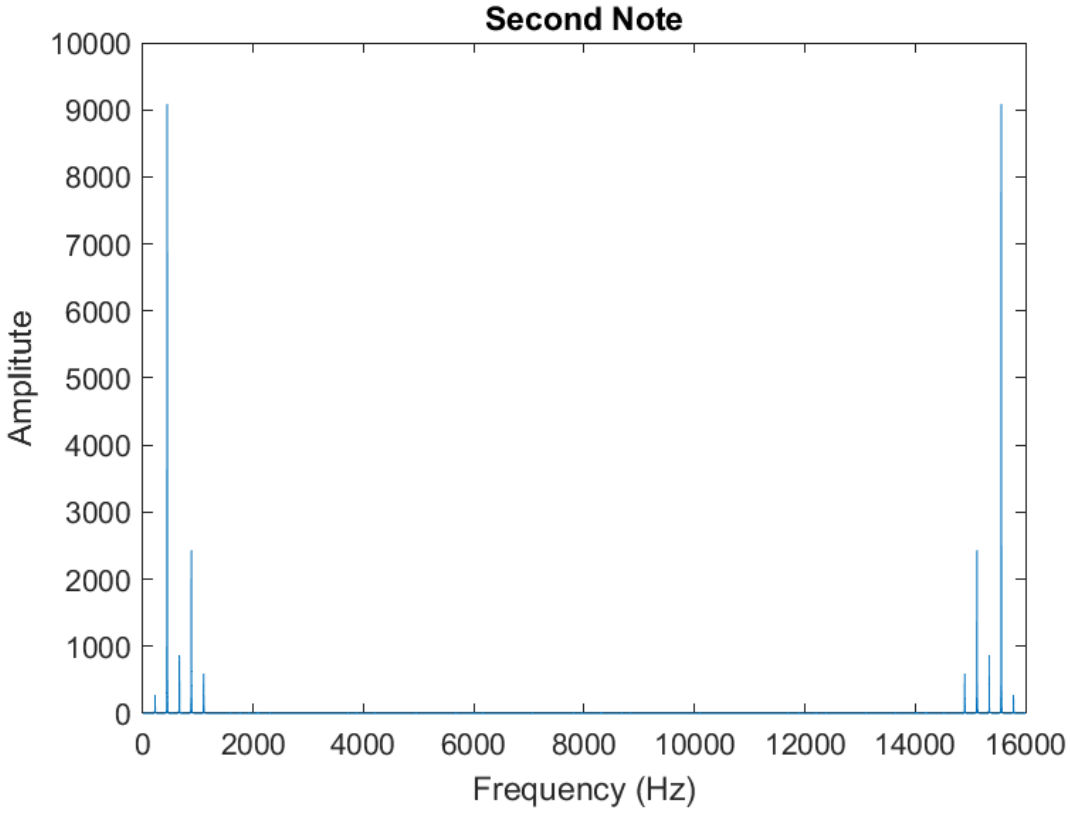
Κατόπιν, έχοντας βρει τις πρώτες πέντε αρμονικές από τις θεμελιώδεις συχνότητες του σήματος (220Hz και 440Hz αντίστοιχα) αρχίζουμε να εφαρμόζουμε την γνωστή μεθοδολογία για την κατασκευή ζωνοπερατών φίλτρων με σκοπό την απομόνωση μιας αρμονική ενός σήματος.

```
First five harmonics for the first note:
         440         880        1320        1760        2200

First five harmonics for the second note:
         220         440         660         880        1100
```

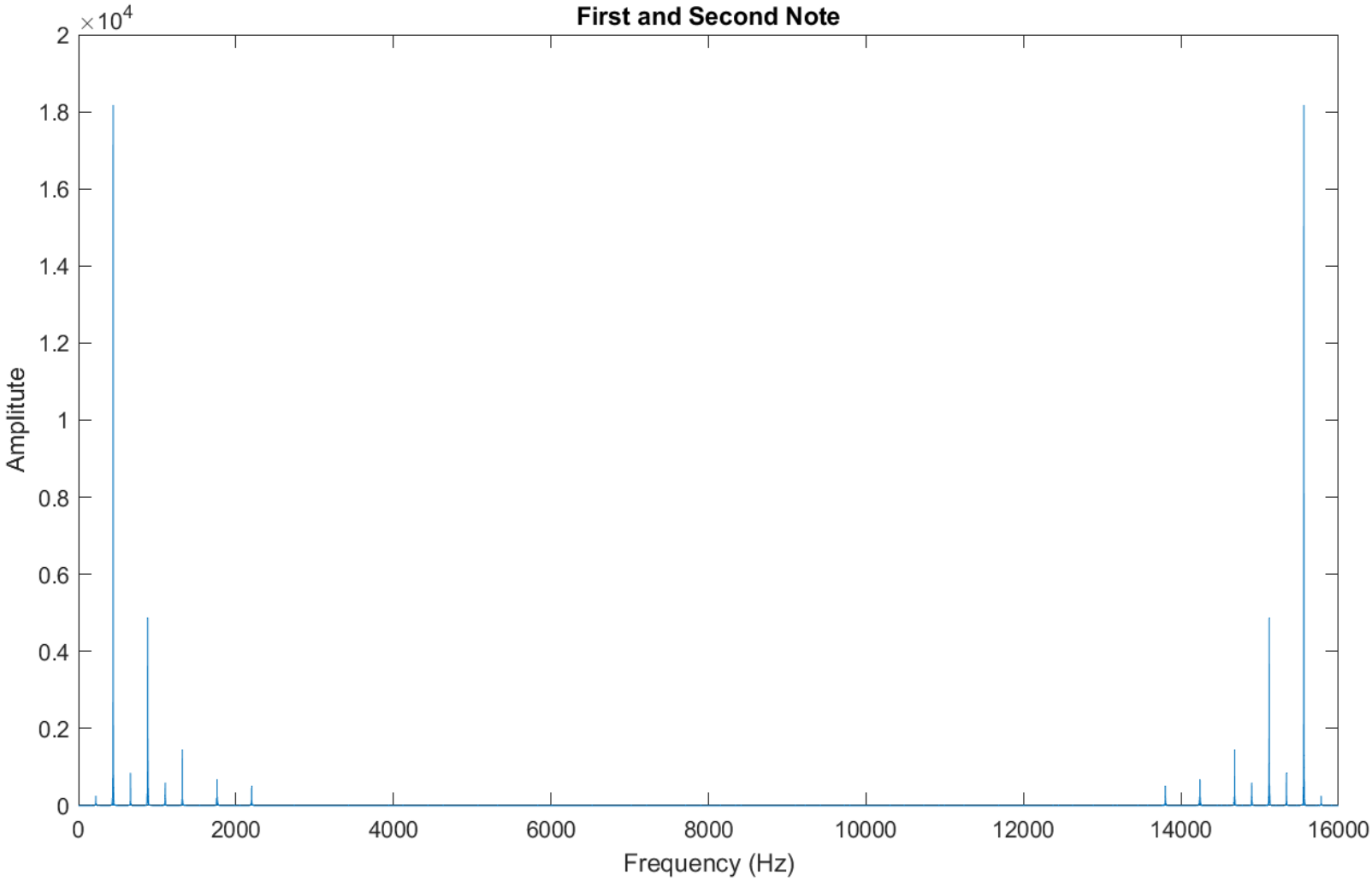
Στη συνέχεια, δημιουργούμε τα δυο σήματα που προκύπτουν από το άθροισμα των δυο συνόλων αρμονικών, όπως και στο προηγούμενο ερώτημα.

Παίρνουμε τον DFT του πρώτου και του δεύτερου σήματος.


First Note
Amplitute
Frequency (Hz)
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
0
2000
4000
6000
8000
10000
12000
14000
16000

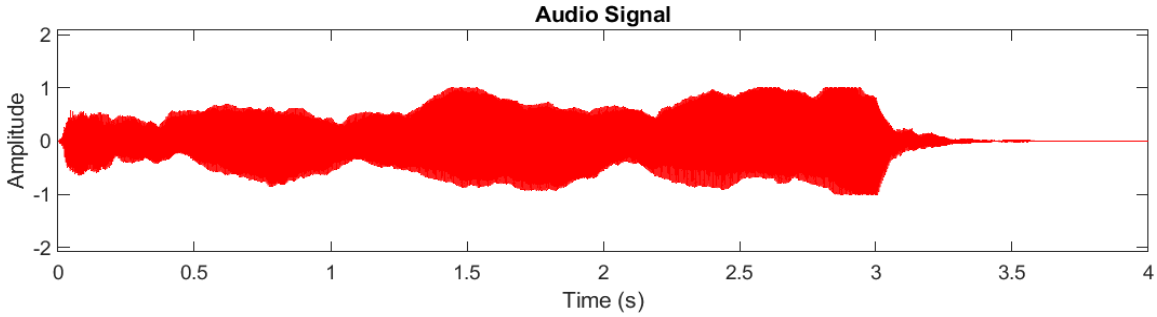

Second Note
Amplitute
Frequency (Hz)
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
0
2000
4000
6000
8000
10000
12000
14000
16000

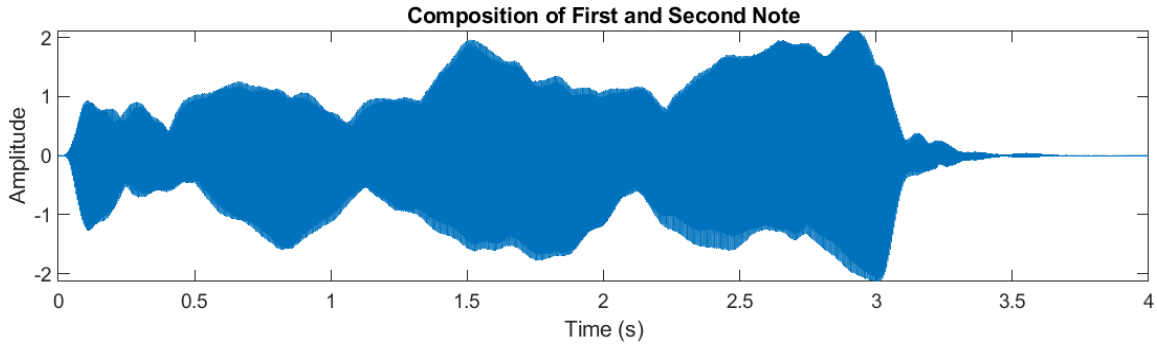
Προσθέτουμε τα μέτρα φάσματος των δυο σημάτων και παίρνουμε το εξής γράφημα.



Παρατηρούμε πως στο μέτρο φάσματος της σύνθεσης των δυο σημάτων όταν η συχνότητα ισούται με τις αρμονικές της θεμελιώδους συχνότητας 440Hz το φάσμα έχει το διπλάσιο μέτρο σε σύγκριση με το μέτρο φάσματος του αρχικού σήματος. Αυτό συμβαίνει καθώς οι αρμονικές την συχνότητας 440Hz αποτελούν αρμονικές και της συχνότητας 220Hz με άρτιο συντελεστή $(2\kappa \cdot f_0)$. Έτσι, προσθέτουμε δύο φορές αυτές τις αρμονικές με αποτέλεσμα να γίνονται πιο κυρίαρχες στο σήμα μας.

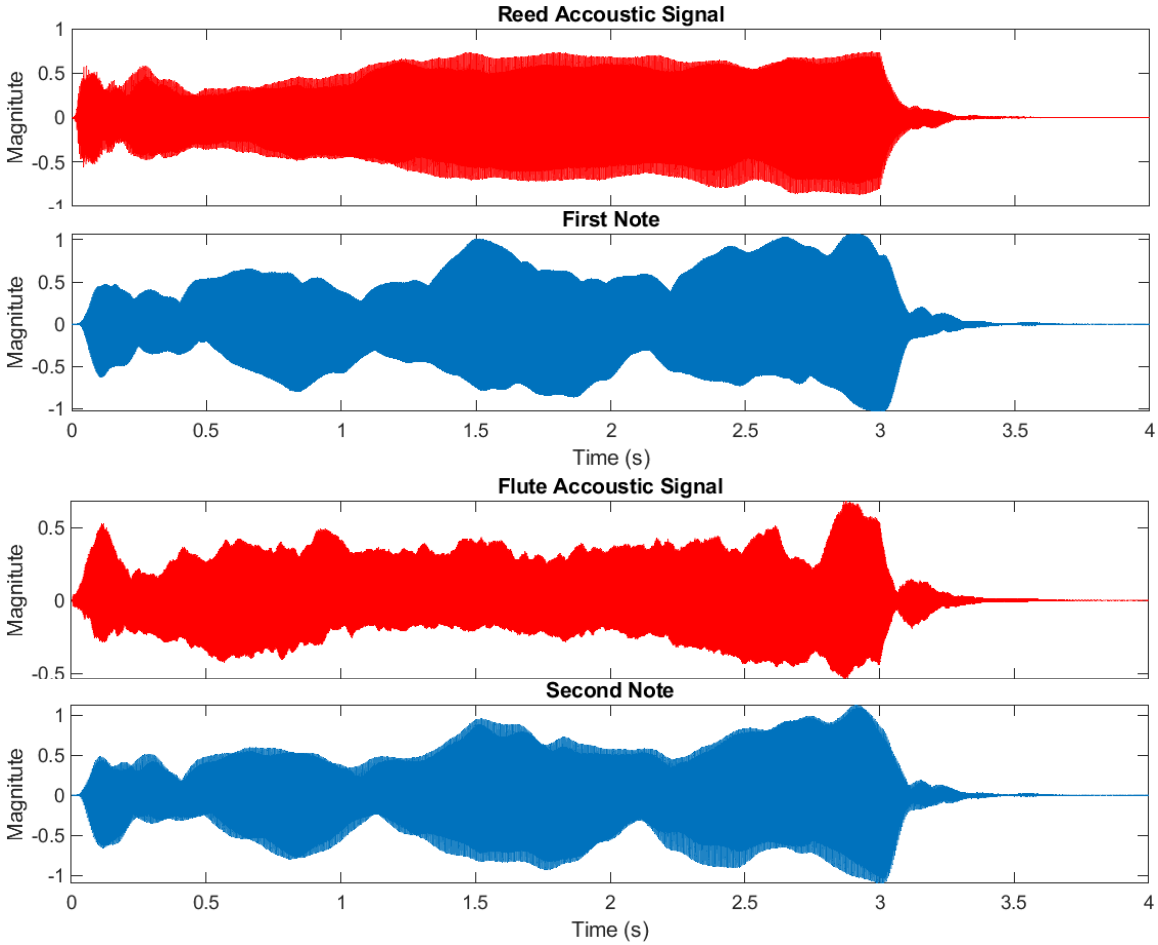
Έπειτα, συγκρίνουμε το σύνθετο σήμα μας με το αρχικό:



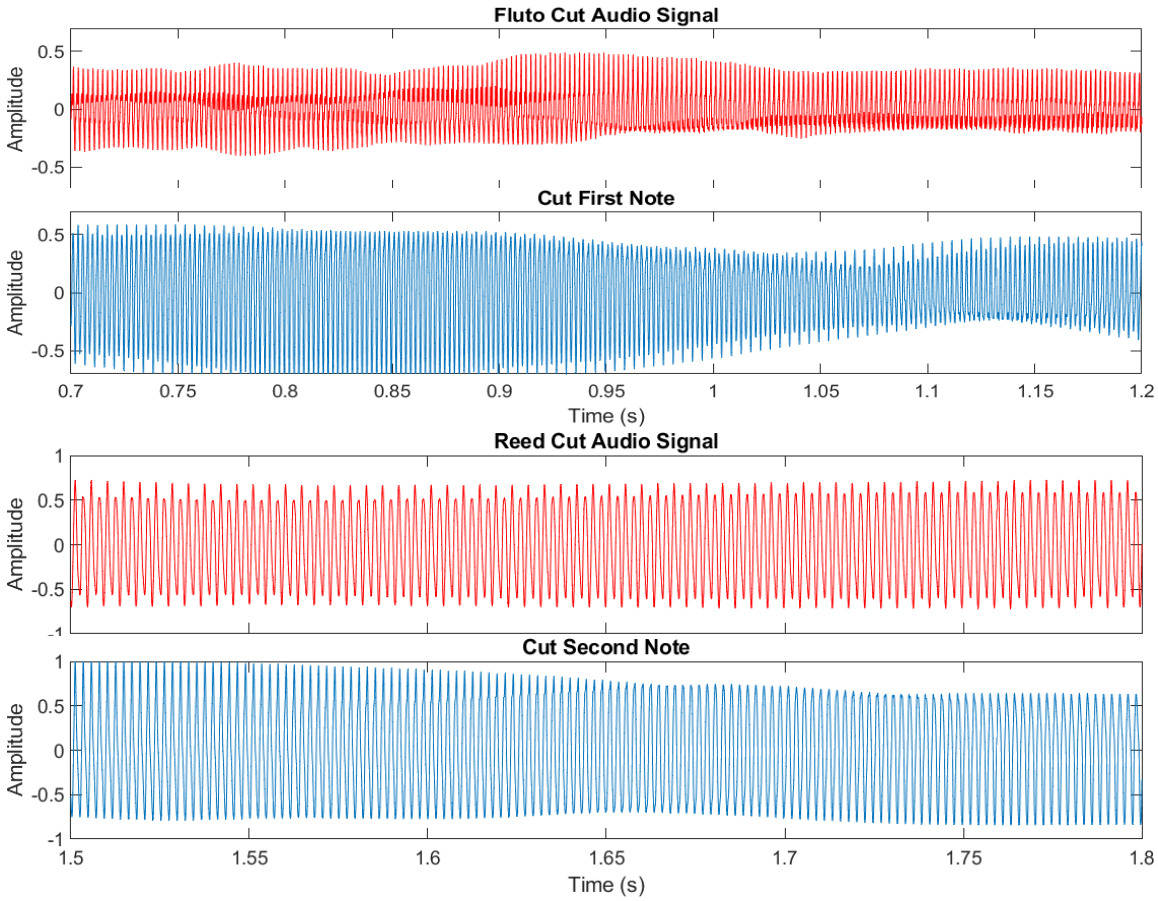


Παρατηρούμε πως το πλάτος του σύνθετου σήματος είναι το διπλάσιο από αυτό του αρχικού σήματος.

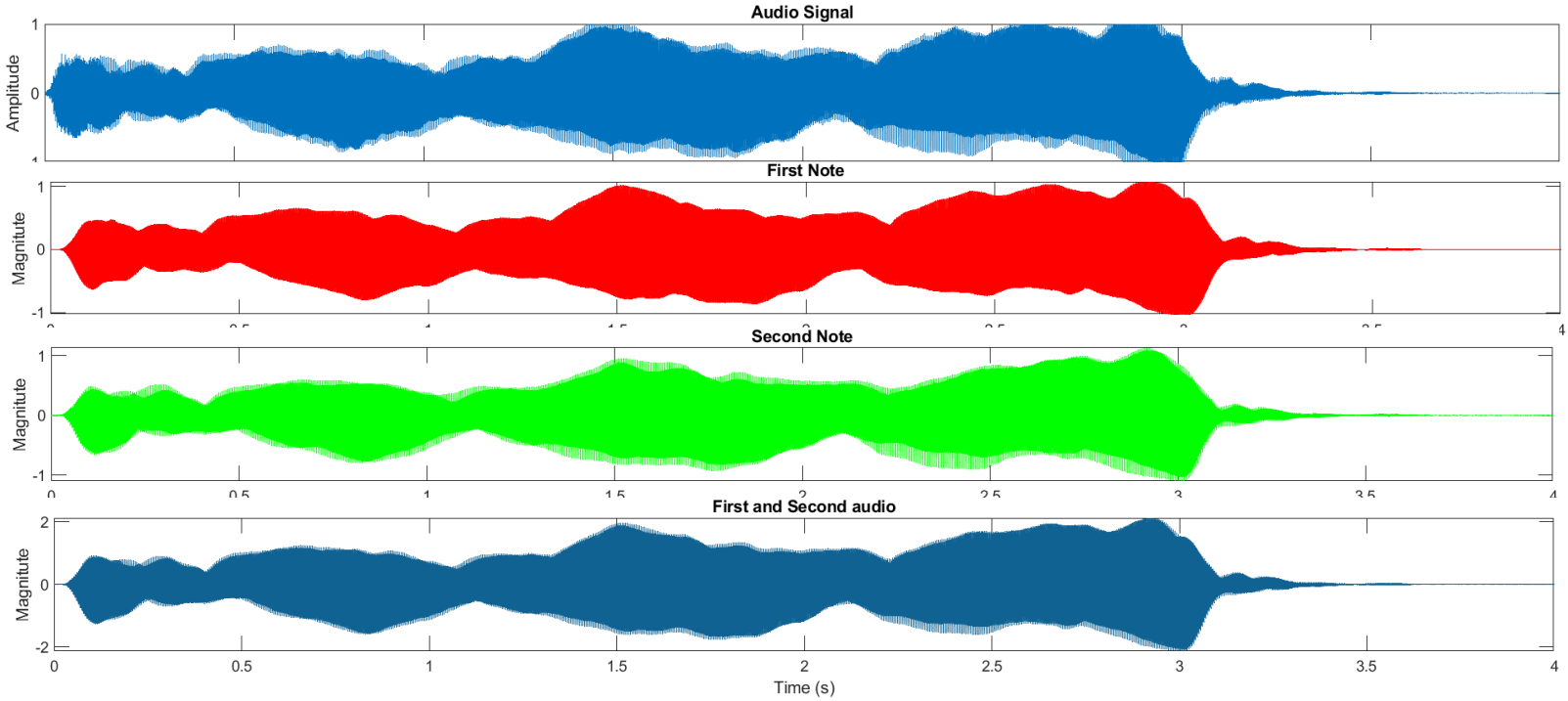
Έπειτα, όπως και στο προηγούμενο ερώτημα, θα συγκρίνουμε τα δυο σήματα με τα αντίστοιχα αρχεία *flute_acoustic_002-069-025.wav και reed_acoustic_037-057-127.wav3.*



Βλέπουμε πως τα δύο σήματα που ανακατασκευάσαμε δεν μοιάζουν ιδιαίτερα με τα πηγαία σήματα flute_acoustic_002-069-025.wav και reed_acoustic_037-057-127.wav του σήματός μας. Οι παρατηρήσεις μας επιβεβαιώνονται συγκρίνοντας αντίστοιχα αποσπάσματα.



Για να έχουμε μια καλύτερη οπτικοποίηση για το τι συμβαίνει στα ζωνοπερατά φίλτρα που φτιάξαμε και γιατί η σύνθεση μας δίνει το αρχικό μας σήμα με διπλάσιο πλάτος θα συγκρίνουμε τις εξής γραφικές παραστάσεις:



Επομένως, από τα παραπάνω διαγράμματα παρατηρούμε πως από τις 5 πρώτες αρμονικές των δυο θεμελιωδών συχνοτήτων ανακατασκευάζουμε το αρχικό μας σήμα και στις δυο περιπτώσεις. Αυτό οφείλεται στο γεγονός πως αρμονικές της μιας συχνότητας αποτελούν αρμονικές με άρτιο δείκτη την άλλης θεμελιώδους συχνότητας.

Οι παρατηρήσεις μας επιβεβαιώνονται ακουστικά καθώς ακούμε το σήμα με αυξημένη ένταση.