



ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ  
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

## Υπολογιστική Εργασία

👤 Μοσχογιάννης Πασχάλης

✉️ [pmoschogiannis@uth.gr](mailto:pmoschogiannis@uth.gr)

👤 Δημητρίου Βασίλειος

✉️ [vasildimitriou@uth.gr](mailto:vasildimitriou@uth.gr)

**Μάθημα:** ΜΔΕ671 Προχωρημένα Θέματα Επεξεργασίας Φωνής και Λόγου

**Ακαδημαϊκό Έτος:** 2023-24

**Καθηγητής:** Ποταμιάνος Γεράσιμος

**Ημερομηνία υποβολής:** 25 Φεβρουαρίου 2024

### ΜΕΡΟΣ Α | Άσκηση Α-1

Ηχογραφήστε το ονοματεπώνυμό σας με συχνότητα δειγματοληψίας τα 16 kHz ή 22,050 kHz.

#### Επίλυση:

Η ηχογράφηση επιτεύχθηκε με τη χρήση διαφόρων εργαλείων. Μεταξύ αυτών είναι το εργαλείο SoX [1], ή "Sound eXchange," που αποτελεί ένα ισχυρό και ευέλικτο εργαλείο στον τομέα της επεξεργασίας ήχου. Επίσης έγινε χρήση του εργαλείου Praat το οποίο είναι ένα λογισμικό ανάλυσης και σύνθεσης ήχου που χρησιμοποιείται κυρίως για ερευνητικούς σκοπούς στον τομέα της φωνητικής, της φωνολογίας και γενικότερα στη γλωσσολογία. Αναπτύχθηκε από τον Paul Boersma και τον David Weenink στο Πανεπιστήμιο του Άμστερνταμ [2]. Τέλος, με τη χρήση της συνάρτησης audio recorder στο MATLAB.

Για την ηχογράφηση του ονοματεπώνυμου μας με χρήση του εργαλείου SoX και συχνότητα δειγματοληψίας τα 16kHz:

```
sox -t alsa default -c 1 -r 16000 -b 16 -e signed-integer -t wav  
recorded_audio.wav remix 1 trim 0 3
```

ενώ με συχνότητα δειγματοληψίας τα 22,050 kHz:

```
sox -t alsa default -c 1 -r 22050 -b 16 -e signed-integer -t wav  
recorded_audio.wav remix 1 trim 0 3
```

Για την ηχογράφηση μέσω ενός bash script αρχείου και επιλογή της συχνότητας δειγματοληψίας δίνοντας την αντίστοιχη παράμετρο:

```
#!/bin/bash

# Check if the sample rate is provided as an argument
if [ $# -eq 0 ]; then
echo "Usage: $0 <sample_rate>"
exit 1
fi

# Get the sample rate from the command-line argument
sample_rate=$1

# Validate sample rate (optional)
if [ "$sample_rate" -ne 16000 ] && [ "$sample_rate" -ne 22050 ];
then
echo "Invalid sample rate. Please use 16000 or 22050."
exit 1
fi

# Perform audio recording for 3 seconds using SoX
sox -t alsa default -c 1 -r $sample_rate -b 16 -e signed-integer -t
wav recorded_audio.wav remix 1 trim 0 3

echo "Recording completed at $sample_rate Hz."
```

δίνοντας δικαιώματα εκτέλεσης στο αρχείο bash script:

```
chmod +x rec_script.sh
```

εκτέλεση με συχνότητα δειγματοληψίας 16 kHz:

```
./rec_script.sh 16000
```

εκτέλεση με συχνότητα δειγματοληψίας 22,050 kHz:

```
./rec_script.sh 22050
```

Παρακάτω παρατίθεται ο κώδικας στο περιβάλλον του MATLAB. Γίνεται χρήση των εντολών για επιλογή συχνότητας δειγματοληψίας τα 16 kHz, βάθος του δείγματος τα "16 bits per sample" και χρόνος καταγραφής τα 3 δευτερόλεπτα:

```
% Set the sample rate
fs = 16000; % or 22050 for 22.05 kHz

% Set the recording duration in seconds
recordingDuration = 3; % 5 seconds for both names

% Create an audio recorder object
recorder = audiorecorder(fs, 16, 1); % 16 bits per sample, 1 channel
% (monophonic)

% Record the audio
disp('Start speaking...');

recordblocking(recorder, recordingDuration);
```

```

disp('End of recording');

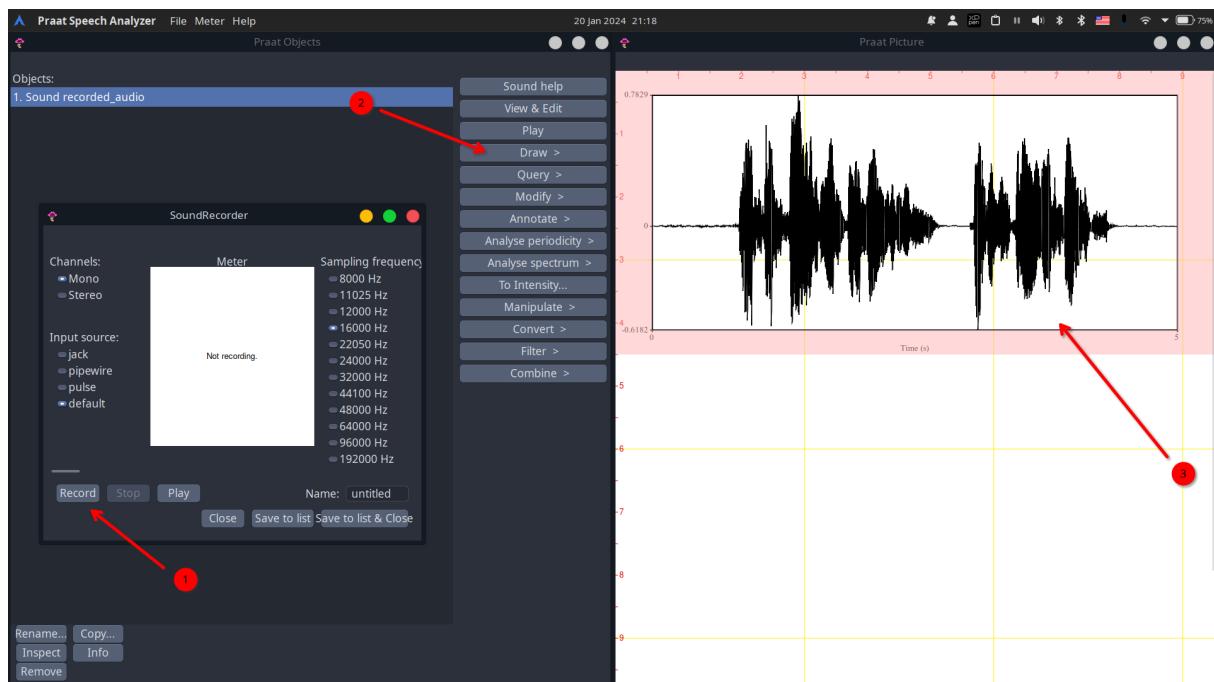
% Get the recorded data
audioData = getaudiodata(recorder);

% Play the recorded audio
sound(audioData, fs);

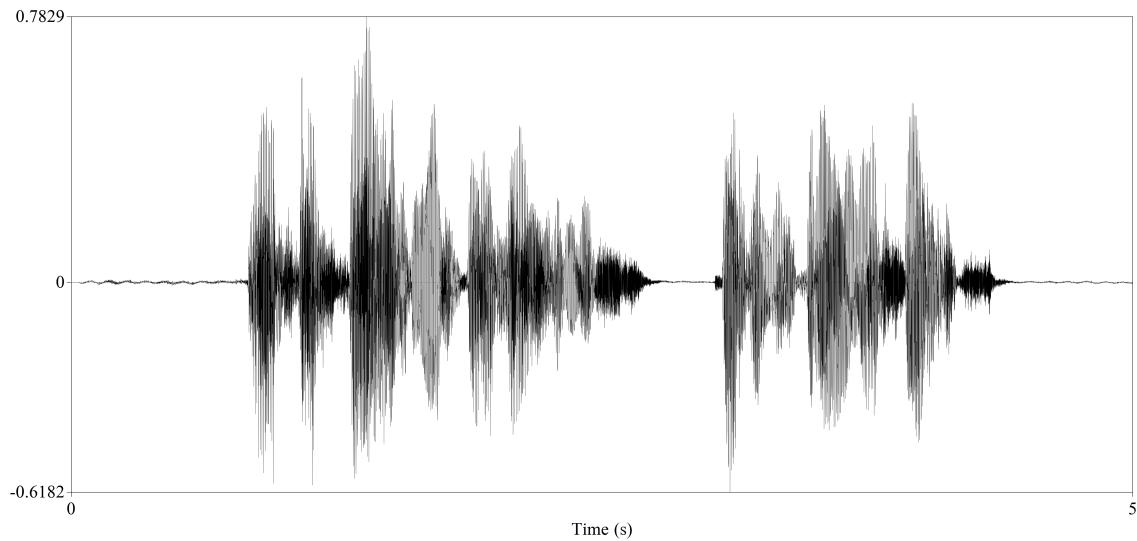
% Save the recorded audio to a file (optional)
audiowrite('recorded_audio.wav', audioData, fs);

```

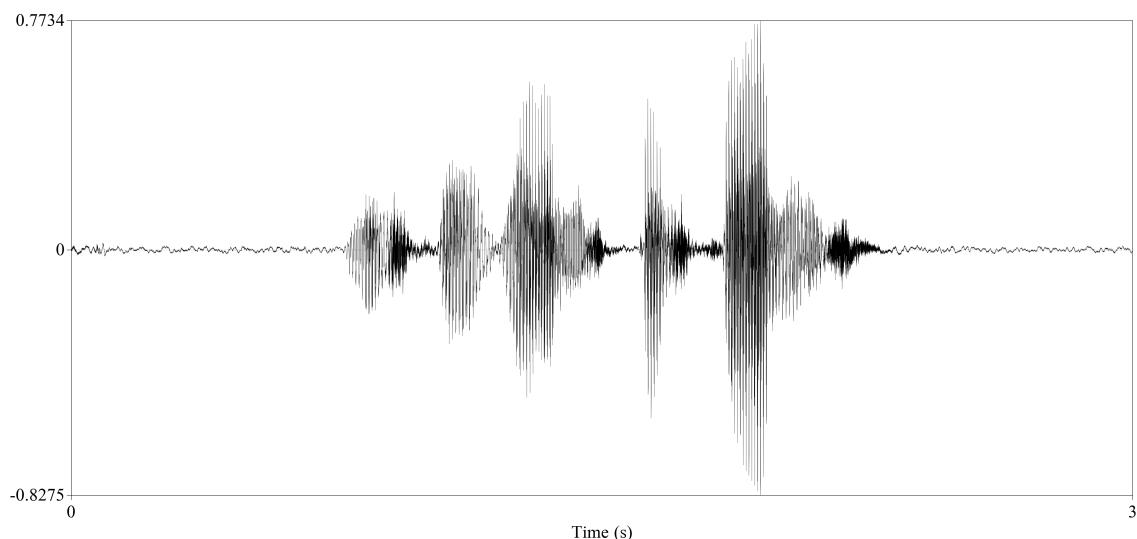
Κάνοντας χρήση του εργαλείου Praat [2] όπως φαίνεται στην Εικόνα 1 για καταγραφή και αποτύπωση του ήχου:



**Εικόνα 1:** Ηχογράφηση στο Praat [2].



**Εικόνα 2:** Αποτύπωση του ηχητικού σήματος της φράσης ”Μοσχογιάννης Πασχάλης και Δημητρίου Βασίλης” στο Praat [2].



**Εικόνα 3:** Αποτύπωση του ηχητικού σήματος του ενός ονοματεπώνυμου στο Praat [2].

## ΜΕΡΟΣ Α | Άσκηση Α-2

Σχεδιάστε το φασματόγραμμα (spectrogram) του σήματος φωνής που ηχογραφήσατε, χρησιμοποιώντας παράθυρο Hamming μήκους περί τα 10 msec και παράθυρο Hamming μήκους περί τα 100 msec, με μετατόπιση (και στις 2 περιπτώσεις) περί τα 5 msec. Σχολιάστε τις διαφορές στα φασματογράμματα.

### Επίλυση:

Το λογισμικό ανάλυσης και σύνθεσης ήχου Praat [2] όπως φαίνεται στην [Εικόνα 4](#) με την κατάλληλη αλληλουχία ενεργειών (επιλογή του αντικειμένου recorded\_audio > Analyse spectrum > To spectrogram) εξάγει τα φασματογράμματα που αναλύουν τις υφιστάμενες συχνότητες στα ορισμένα χρονικά παράθυρα.

Εφόσον η συχνότητα δειγματοληγίας του ηχογραφημένου σήματος είναι 16,000 Hz, το μονόπλευρο διάγραμμα πλάτους Fourier, θα πρέπει να καλύπτει συχνότητες έως τα 8,000 Hz, δηλαδή το ήμισυ της συχνότητας δειγματοληγίας εξαιτίας της συμμετρικότητας που το διακατέχει. Στο ιστόγραμμα στην [Εικόνα 6](#), οι συχνότητες επεκτείνονται έως τα 8 kHz με τον μετασχηματισμό Fourier να αποσυνθέτει το σήμα στις διάφορες συχνότητες που το αποτελούν. Το πρόβλημα με τον μετασηματισμό Fourier είναι η αδυναμία απεικόνισης των επικρατέστερων συχνοτήτων ανά χρονικό παράθυρο.

Η δημιουργία ενός φασματογράμματος είναι μια διαδικασία που επιτρέπει την ανάλυση και την οπτικοποίηση της συχνοτικής περιεκτικότητας ενός ηχητικού σήματος σε σχέση με τον χρόνο. Στην ουσία η εφαρμογή του μετασχηματισμού Fourier γίνεται σε κάθε παράθυρο του σήματος, με την ολίσθηση του παραθύρου σε ολόκληρη τη διάρκεια της εγγραφής. Αυτή η διαδικασία παράγει έναν πίνακα των συχνοτήτων που εμφανίζονται στο σήμα σε κάθε χρονική στιγμή, με τον άξονα X να αντιπροσωπεύει τον χρόνο και τον άξονα Y τις συχνότητες. Η ένταση κάθε συχνότητας σε κάθε χρονική στιγμή απεικονίζεται με διαφορετικές αποχρώσεις ή χρώματα. Όσον αφορά το πλάτος του παραθύρου Hamming, αυτό επηρεάζει την ανάλυση συχνότητας και χρόνου του φασματογράμματος. Ένα μικρότερο παράθυρο προσφέρει μεγαλύτερη χρονική ανάλυση αλλά χαμηλότερη συχνοτική ανάλυση, επιτρέποντας την ανίχνευση γρήγορων αλλαγών στο σήμα αλλά με λιγότερη λεπτομέρεια στις συχνότητες. Αντίθετα, ένα μεγαλύτερο παράθυρο βελτιώνει τη συχνοτική ανάλυση αλλά μειώνει τη χρονική ανάλυση, καθιστώντας δυνατή την ανίχνευση λεπτομερειών στις συχνότητες αλλά με χειρότερη ανίχνευση των χρονικών αλλαγών. Έτσι, στο φασματόγραμμα στην [Εικόνα 9](#) γίνεται σκιαγράφηση των κομματιών που αφορούν τα ηχητικά "Μοσχογιάννης Πασχάλης", "και" και τελικά το "Δημητρίου Βασίλης". Αντίστοιχα στην [Εικόνα 10](#) διακρίνονται τα φασματογράμματα σε παράθυρα Hamming 10msec και 100msec του ενός ονοματεπωνύμου.

Στο πλαίσιο του κώδικα που ακολουθεί φαίνεται το περιεχόμενο του αρχείου spectrograms.m για την εφαρμογή αυτών που προαναφέρθηκαν του ενός ονοματεπωνύμου "Μοσχογιάννης Πασχάλης".

```
% Load the recorded audio file
[audioData, fs] = audioread('recorded_audio.wav');

% Store the recording duration in seconds
recordingDuration = 3;

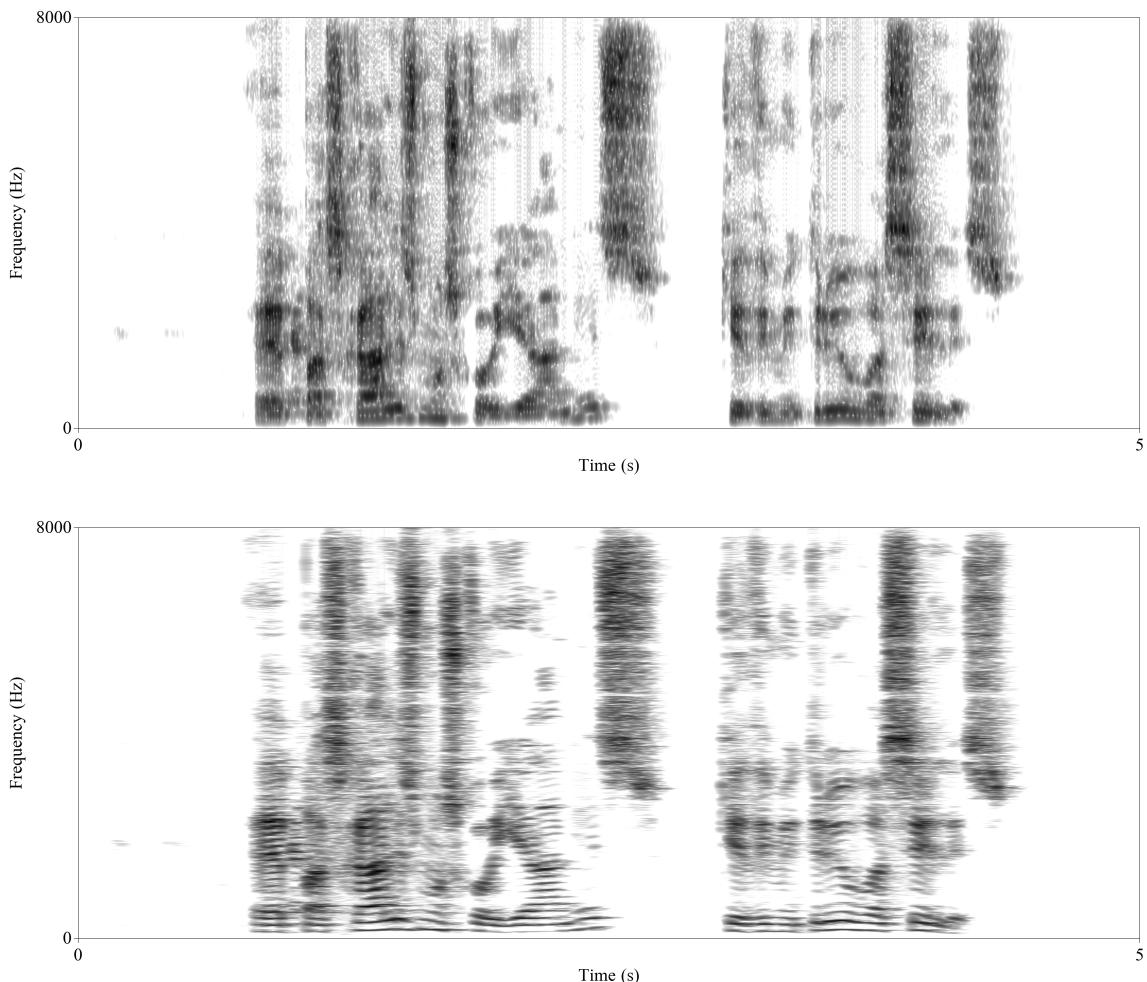
% Calculate the length of the signal
%L = recordingDuration*fs;
L = size(audioData,1);
```

```
% Fast Fourier Transform before spectrogram creation
figure(1);
Y = fft(audioData);
plot(fs/L*(0:L-1),abs(Y),'LineWidth',3)
title('Complex Magnitude of fft Spectrum')
xlabel('f (Hz)')
ylabel('|fft(Audio)|')

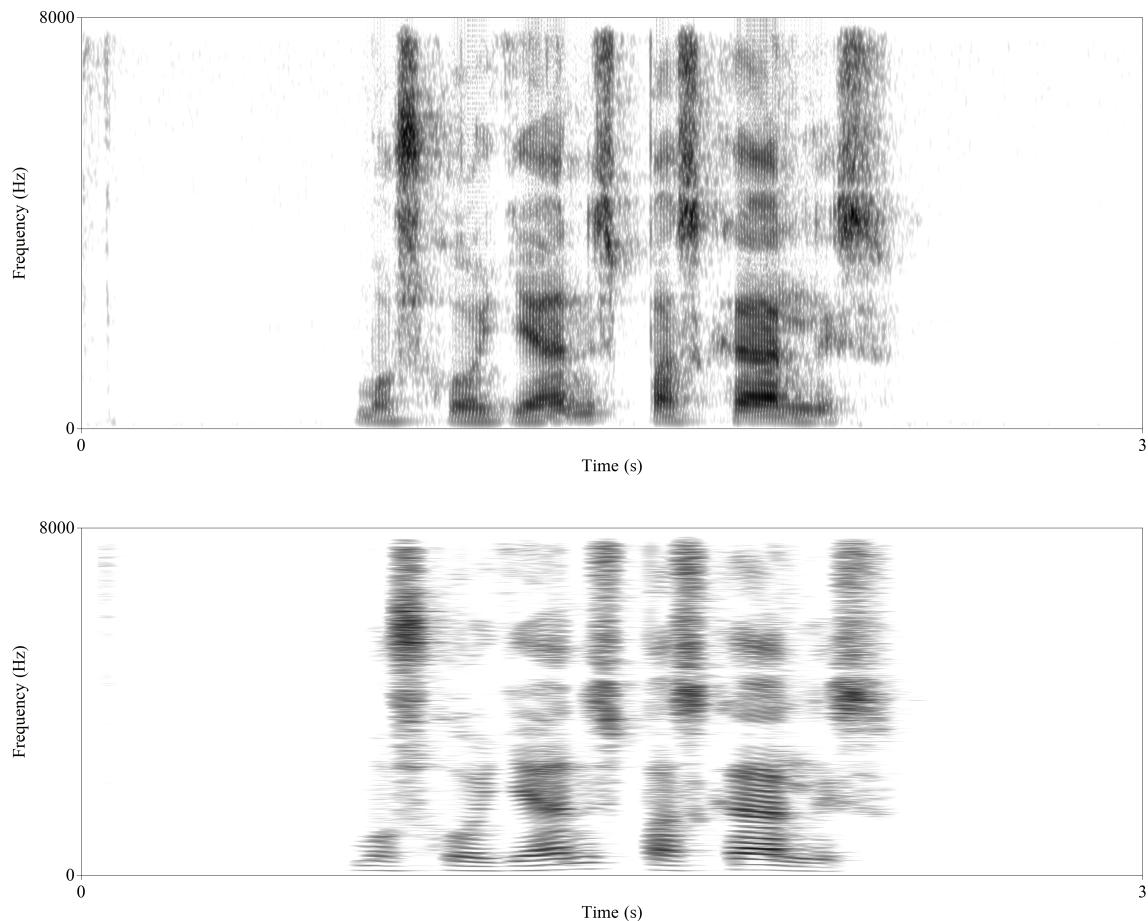
figure(2);
P2 = abs(Y/L);
P1 = P2(1:L/2+1);
P1(2:end-1) = 2*P1(2:end-1);
f = fs/L*(0:(L/2));
plot(f,P1,'LineWidth',3)
title('Single-Sided Amplitude Spectrum of audioData(t)')
xlabel('f (Hz)')
ylabel('|P1(f)|')

% Parameters for the spectrogram
window10ms = hamming(round(fs * 0.01),'symmetric'); % 10 msec Hamming
    window
window100ms = hamming(round(fs * 0.1),'symmetric'); % 100 msec
    Hamming window
overlap = round(fs * 0.005); % 5 msec overlap

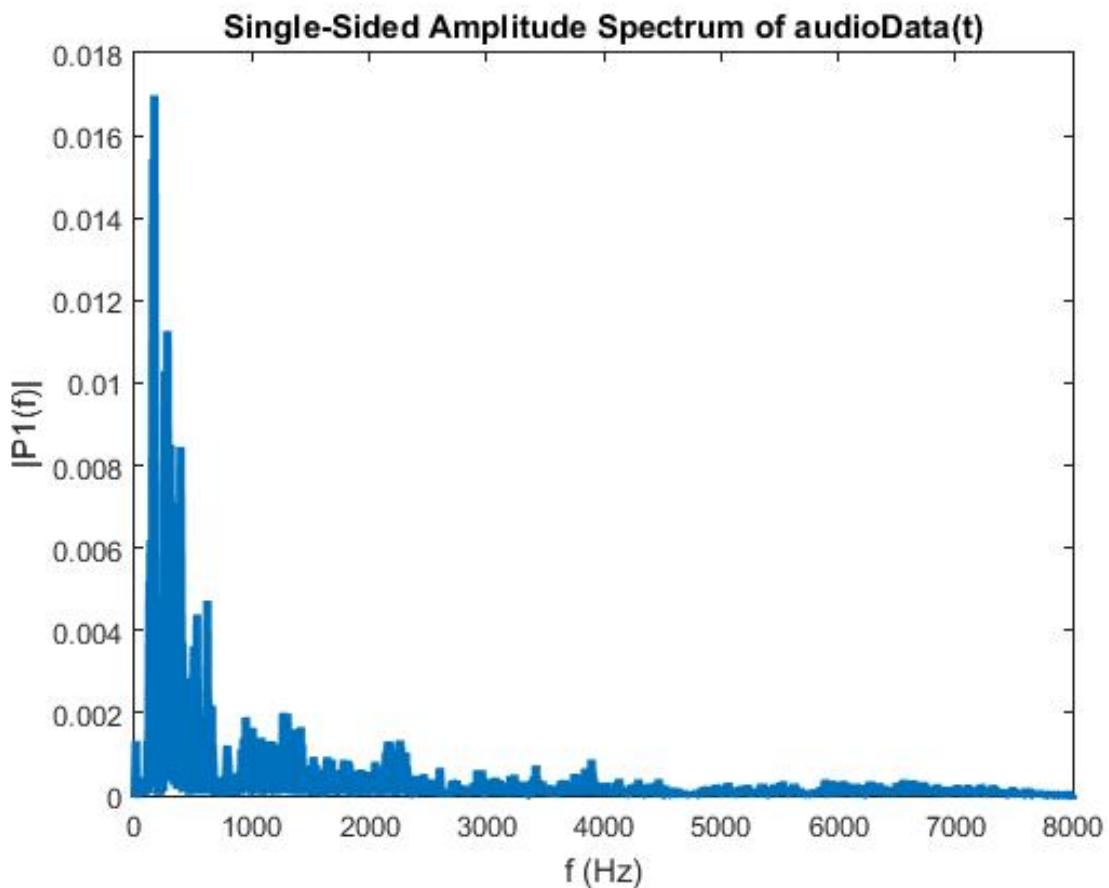
% Spectrogram with 10 msec Hamming window and 5 msec overlap
figure(3);
subplot(2,1,1);
spectrogram(audioData, window10ms, overlap, [], fs, 'yaxis');
title('Spectrogram with 10ms Hamming Window and 5ms Overlap');
xlabel('Time (s)');
ylabel('Frequency (kHz)');
subplot(2,1,2);
% Spectrogram with 100 msec Hamming window and 5 msec overlap
spectrogram(audioData, window100ms, overlap, [], fs, 'yaxis');
title('Spectrogram with 100ms Hamming Window and 5ms Overlap');
xlabel('Time (s)');
ylabel('Frequency (kHz)');
```



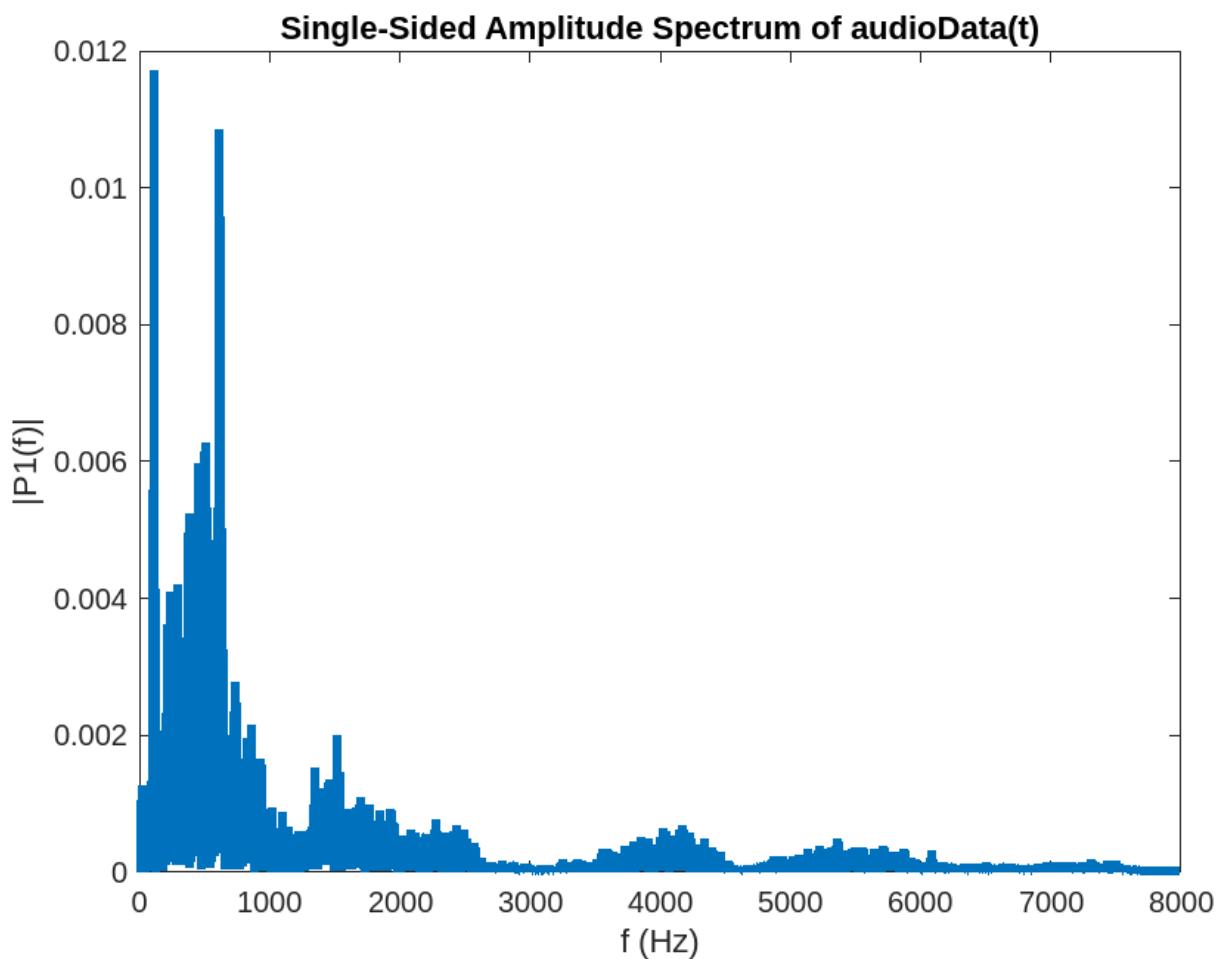
**Εικόνα 4:** Φασματογράμματα για 10msec και 100msec κατά αντιστοιχία στο Praat [2] για τη φράση "Μοσχογιάννης Πασχάλης και Δημητρίου Βασίλης".



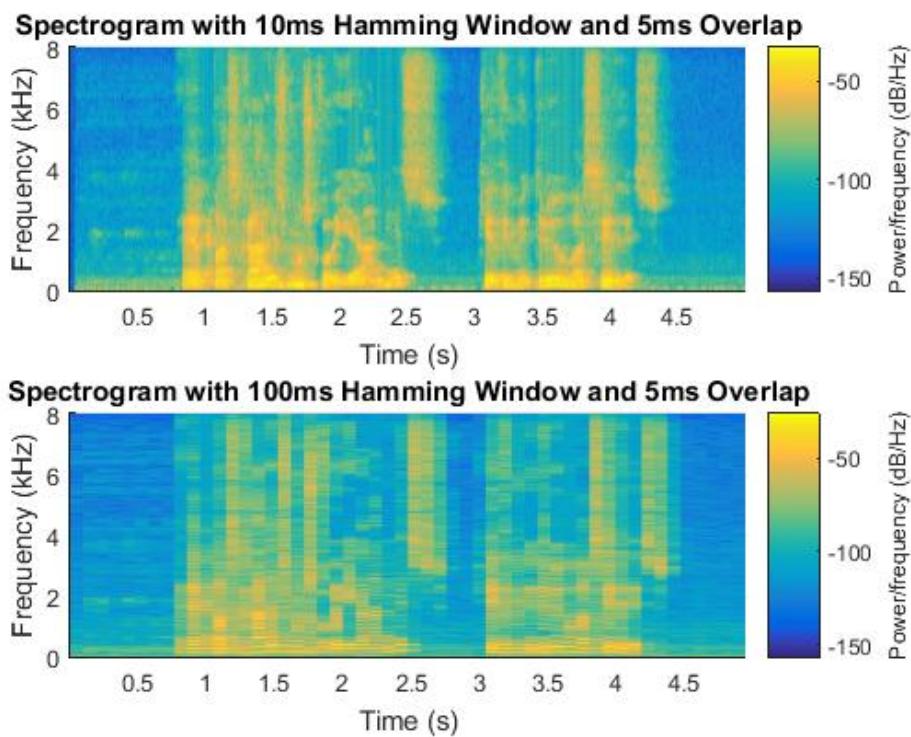
**Εικόνα 5:** Φασματογράμματα για 10msec και 100msec κατά αντιστοιχία στο Praat [2] του ενός ονοματεπωνύμου.



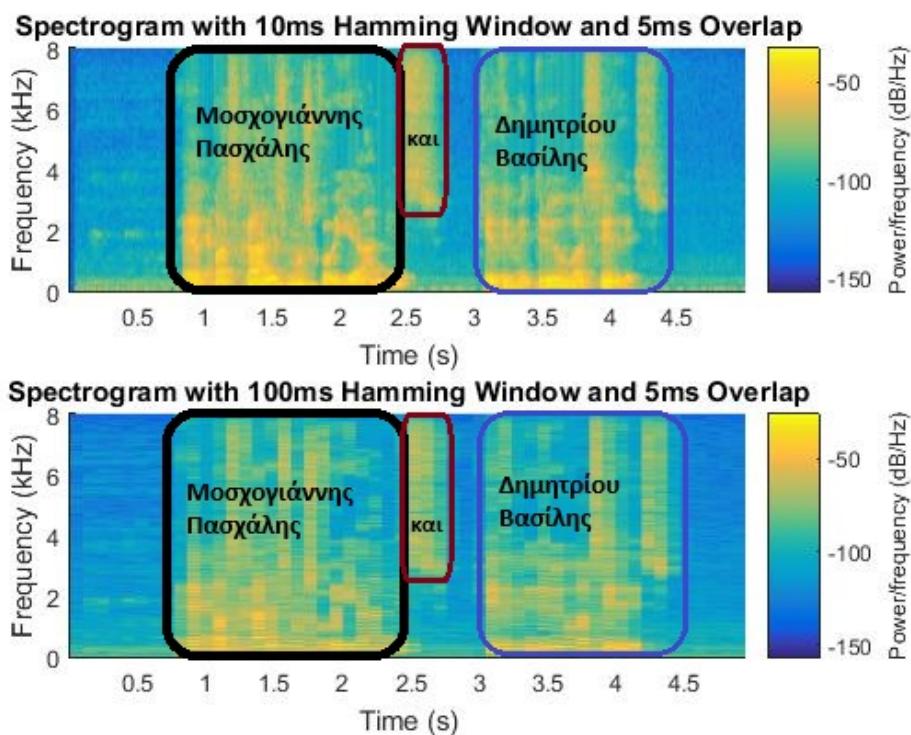
**Εικόνα 6:** Διακριτός Μετασχηματισμός Fourier (Αλγόριθμος Fast Fourier Transform) για τη φράση "Μοσχογιάννης Πασχάλης και Δημητρίου Βασίλης".



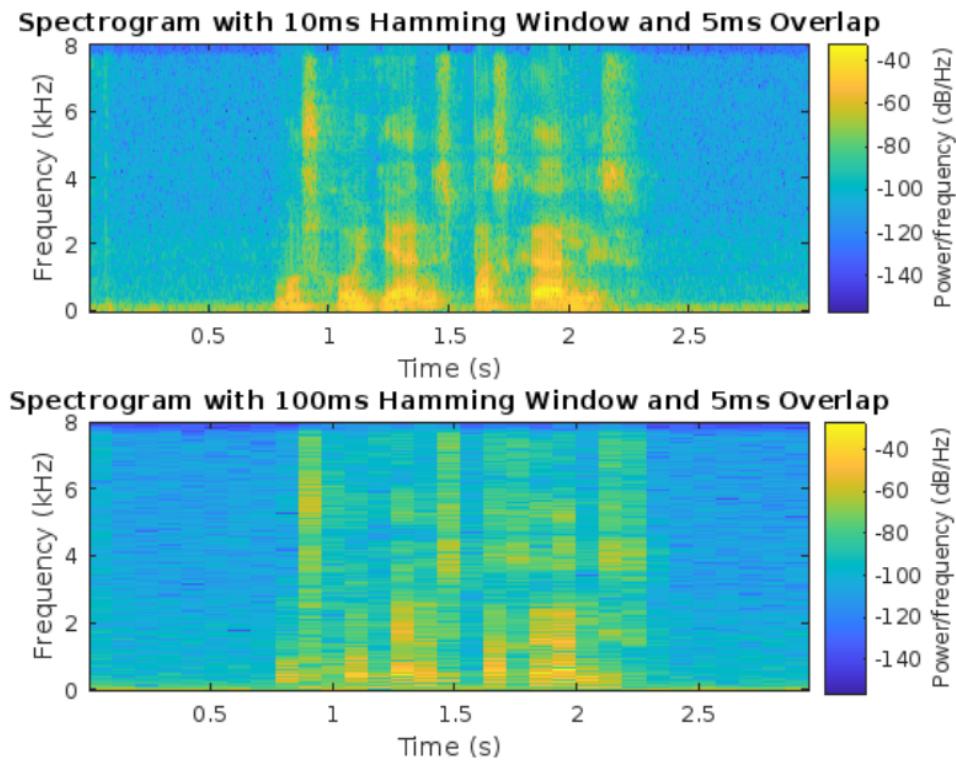
**Εικόνα 7:** Διακριτός Μετασχηματισμός Fourier (Αλγόριθμος Fast Fourier Transform) του ενός ονοματεπωνύμου.



**Εικόνα 8:** Φασματογράμματα σε παράθυρα Hamming 10msec και 100msec για τη φράση "Μοσχογιάννης Πασχάλης και Δημητρίου Βασίλης".



**Εικόνα 9:** Υπόδειξη του περιεχομένου (λέξεις) των φασματογραμμάτων.



**Εικόνα 10:** Φασματογράμματα σε παράθυρα Hamming 10msec και 100msec του ενός ονοματεπωνύμου.

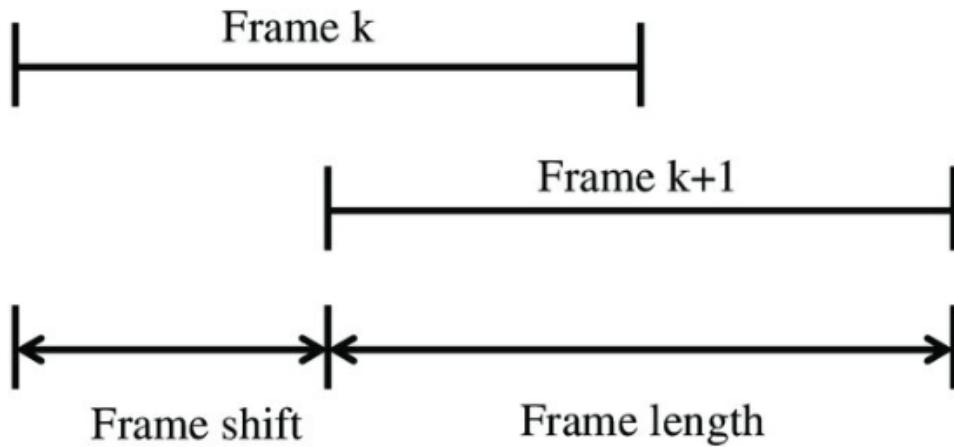
### ΜΕΡΟΣ Α | Ασκηση Α-3

Σχεδιάστε έναν αλγόριθμο που διαχωρίζει τα τμήματα του σήματος που περιέχουν έμφωνους ήχους (voiced) από αυτά που περιέχουν άφωνους ήχους (unvoiced), όπως και σιωπή (silence), που πιθανώς να υπάρχει στην ηχογράφηση σε διάφορα σημεία της, π.χ. ίσως μεταξύ του μικρού ονόματος και επιθέτου, όπως και στην αρχή και στο τέλος της. Χρησιμοποιήστε για τον σκοπό αυτόν την ενέργεια (energy) και ρυθμό διαβάσεων μηδενικής τιμής (zero-crossing rate), φυσικά υπολογισμένα σε παράθυρα του σήματος (κατάλληλου μήκους και επικάλυψης). Σχεδιάστε τις τιμές των δύο αυτών μεγεθών energy και zero-crossing rate σε ένα παράλληλο σχήμα με αυτό της κυματομορφής του σήματος φωνής. Σχεδιάστε επίσης την έξοδο (απόφαση) του αλγορίθμου σας σε τμήματα voiced, unvoiced, silence (Κεφ. 10.3).

#### Επίλυση:

Στον τομέα της φωνητικής ανάλυσης και επεξεργασίας ομιλίας, οι ήχοι κατηγοριοποιούνται σε τρεις κύριες κατηγορίες: φωνητικοί, άφωνοι και σιωπηλοί ήχοι. Οι φωνητικοί ήχοι χαρακτηρίζονται από τη δόνηση των φωνητικών χορδών, παράγοντας περιοδική και μελωδική ηχητική έκφραση. Παραδείγματα περιλαμβάνουν τα φωνήντα και ορισμένα σύμφωνα. Αντίθετα, οι άφωνοι ήχοι παράγονται χωρίς δόνηση των χορδών, είναι απροσδιόριστοι και σχετίζονται με συγκεκριμένα σύμφωνα, προσομοιώνοντας λευκό θόρυβο. Τέλος, οι σιωπηλοί ήχοι αντιπροσωπεύουν περίοδους απόλυτης σιωπής κατά την ομιλία, όπως οι παύσεις ανάμεσα σε λέξεις ή φράσεις. Η κατηγοριοποίηση αυτή είναι κρίσιμη για την ακριβή αναγνώριση και κατανόηση της ομιλίας σε ποικίλες εφαρμογές, όπως συστήματα αναγνώρισης ομιλίας και τεχνολογίες σύνθεσης φωνής.

Ο παρακάτω κώδικας είναι μια προσέγγιση για την ανάλυση ενός ηχητικού σήματος προκει-



**Εικόνα 11:** Απεικόνιση της μετατόπισης και της διάρκειας του frame.

μένου να διακρίνει μεταξύ έμφωνων, άφωνων και σιωπηλών τμημάτων. Αρχικά, φορτώνει το ηχητικό αρχείο 'recorded\_audio.wav'. Για την ανάλυση των τμημάτων ομιλίας, το ηχητικό σήμα διαιρείται σε μικρότερα κομμάτια (ή πλαίσια), το καθένα διάρκειας 30 milliseconds (ms). Αυτό σημαίνει ότι αν η συχνότητα δειγματοληψίας είναι 16 kHz, κάθε πλαίσιο περιέχει 480 δειγματα 10 ms επικάλυψη. Κάθε πλαίσιο δηλαδή ξεκινά 10 ms μετά το προηγούμενο, πράγμα που σημαίνει ότι υπάρχει μια επικάλυψη μεταξύ των πλαισίων. Αυτό βοηθάει να μην χάνονται πληροφορίες στις μεταβάσεις μεταξύ των πλαισίων. Επίσης, υπολογίζει την ενέργεια που είναι το άθροισμα των τετραγώνων των τιμών του σήματος και τον ρυθμό διαβάσεων μηδενικής τιμής (ZCR) που είναι ο αριθμός των φορών που το σήμα περνά από το μηδέν (αλλάζει πρόσημο) για κάθε πλαίσιο, εφαρμόζοντας ένα παράθυρο Hamming για την ελαχιστοποίηση των παραμορφώσεων στις άκρες των πλαισίων. Μεγάλες τιμές ενέργειας συνήθως δηλώνουν έντονη δραστηριότητα στο σήμα, όπως η ομιλία. Υψηλές τιμές ZCR συχνά δείχνουν άφωνους ήχους ή σιωπή [3].

Οι οριακές τιμές ITU (Initial Threshold for log Energy Upper limit) και ITR (Initial Threshold for log Energy Range) χρησιμοποιούνται για τον καθορισμό των κατώτατων ορίων για την ανίχνευση της ομιλίας. Το ITU, μια σταθερή τιμή μεταξύ -10 και -20 dB [3], καθορίζει το ανώτατο όριο της ενέργειας για την ανίχνευση ομιλίας, ενώ το ITR προσαρμόζεται βάσει της μέσης τιμής και της τυπικής απόκλισης της ενέργειας των πρώτων πλαισίων, λειτουργώντας ως κατώτατο όριο για την ανίχνευση [3].

Ο κώδικας συνεχίζει με την εφαρμογή ενός αλγορίθμου αναζήτησης για τον εντοπισμό της αρχής και του τέλους της ομιλίας, χρησιμοποιώντας τις οριακές τιμές ITU και ITR. Αυτή η διαδικασία περιλαμβάνει την αναζήτηση προς τα εμπρός και προς τα πίσω για τον εντοπισμό των πλαισίων που υπερβαίνουν το ITR και επιβεβαιώνονται με την υπέρβαση του ITU, καθορίζοντας έτσι τα όρια της ομιλίας.

Τέλος, ο κώδικας προσδιορίζει τα τμήματα του σήματος που θεωρούνται έμφωνα, άφωνα, και σιωπηλά, βάσει του ZCR και της ενέργειας. Τα έμφωνα τμήματα χαρακτηρίζονται από χαμηλό ZCR και υψηλή ενέργεια, ενώ τα άφωνα από υψηλό ZCR και σχετικά χαμηλή ενέργεια. Τα σιωπηλά τμήματα εντοπίζονται εκεί όπου και οι δύο παράμετροι είναι σχετικά χαμηλές [3]. Η διαδικασία αυτή επιτρέπει την λεπτομερή ανάλυση και διαχωρισμό του ηχητικού σήματος για περαιτέρω επεξεργασία ή ανάλυση.

```
% Load the recorded audio file
[audioData, fs] = audioread('recorded_audio.wav');
recordingDuration=3;

desired_fs = 16000;
% Set the target sampling rate
targetFs = desired_fs; % Replace with your desired target sampling
rate

% Plot the original and resampled waveforms
timeOriginal = (0:length(audioData)-1) / fs;

% Choose the waveform to be used
desiredAudio = audioData;

L = 640; % 40 msec frame duration
R = 160; % 10 msec frame shift

% Parameters for short-time analysis
frameSize = L; % 300 Samples at Fs=10kHz
frameOverlap = L - R; % 100 Samples at Fs=10kHz

% A3
% Use buffer for overlapping frames
bufferedAudio = buffer(desiredAudio, L, frameOverlap);

% Get the number of frames
numFrames = size(bufferedAudio, 2);
%Number of Frames = (Signal Duration - Frame Duration) / (Frame Shift
% - Frame Overlap) + 1
numFramesCrossCheck = floor(abs(((length(desiredAudio) - L) / (R -
frameOverlap)) + 1));

% Initialize arrays for short-time log energy and zero crossing rate
logEnergy = zeros(1, numFrames);
Energy = zeros(1, numFrames);
zeroCrossingRate = zeros(1, numFrames);

% Compute short-time log energy and zero crossing rate for each frame
for i = 1:numFrames
currentFrame = bufferedAudio(:, i);
% Apply Hamming window to the frame
hammingWindow = hamming(frameSize,'symmetric');
%hammingWindow = hamming(frameSize,'periodic');
windowedFrame = currentFrame .* hammingWindow;
% Short-time log energy
logEnergy(i) = 10 * log10(sum(windowedFrame.^2));
Energy(i) = sum(windowedFrame.^2);
% Short-time zero crossing rate
zeroCrossingRate(i) = R * sum(abs(diff(sign(windowedFrame)))) / (2 *
```

```

    frameSize);
end
% Normalized energy at 0
normalizedLogEnergy = logEnergy - max(logEnergy);

% Plot the short-time log energy and zero crossing rate
frameTime = (0:numFrames-1) * frameOverlap / targetFs;

%IZCT = max(35, mean(zeroCrossingRate(1:3)) + 3 * std(
%    zeroCrossingRate(1:3)));
IZCT = 35;
ITU = -12; % A constant between -10 and -20 ~-12
ITR = -max(-(ITU-10), (mean(logEnergy(1:3)) + 3 * std(logEnergy(1:3)))
    );
%ITR = ITU-10;

figure(8);
subplot(4,1,1);
plot(timeOriginal, desiredAudio);
title('Speech Signal');
xlabel('Time (s)');
ylabel('Amplitude');

subplot(4,1,2);
plot(1:numFrames, normalizedLogEnergy);
title('Short-Time Log Energy with Hamming Window');
xlabel('Frame Number');
ylabel('Log Energy');
legend('Normalized Log Energy', 'ITU' , 'ITR')

subplot(4,1,3);
plot(1:numFrames, zeroCrossingRate);
title('Short-Time Zero Crossing Rate with Hamming Window');
xlabel('Frame Number');
ylabel('Zero Crossing Rate Zc');
legend('Short-Time Zero Crossing Rate', 'IZCT' )

%2 iterations process of E and B values calculation because the wav
file

initialFrame = 1;
framesHorizon = numFrames;
check=1;

% Step 1: Forward search to find B1
while initialFrame <= framesHorizon
% Find the first frame where log energy exceeds lower threshold (ITR)

```

```

if normalizedLogEnergy(initialFrame) > ITR
% Check the region around the detected frame
regionCheck = normalizedLogEnergy(initialFrame-1:initialFrame+1) >
    ITU;
if all(regionCheck)
% Stable initial frame found
B1 = initialFrame;
break;
end
end
initialFrame = initialFrame + 1;
end

% Step 2: Backward search to find E1
initialFrame = framesHorizon;
while initialFrame >= 1
% Find the first frame where log energy exceeds lower threshold (ITR)
if normalizedLogEnergy(initialFrame) > ITR
% Check the region around the detected frame
regionCheck = normalizedLogEnergy(initialFrame-1:initialFrame+1) >
    ITU;
if all(regionCheck)
% Stable initial frame found
E1 = initialFrame;
break;
end
end
initialFrame = initialFrame - 1;
end

% Step 3: Search backward from B1 to B1-25 for zero-crossing count
if sum(zeroCrossingRate(B1-25:B1) > IZCT) >= 4
for i=B1-25:B1
if zeroCrossingRate(i) > IZCT
B2 = i;
break;
end
end
else
B2 = B1;
end

% Step 4: Search forward from E1 to E1+25 for zero-crossing count
if sum(zeroCrossingRate(E1:E1+25) > IZCT) >= 4
for i=E1:E1+25
if zeroCrossingRate(i) > IZCT
E2 = i;
break;
end
end

```

```

else
E2 = E1;
end

% Step 5: Final check around [B2, E2] for log energy threshold (ITR)
% Modify beginning and/or ending frame to match the extended region
if check==1
for i = B2-25:B2
if normalizedLogEnergy(i) > ITR
temporaryExceedingFrames=find(normalizedLogEnergy > ITR);
B2 = temporaryExceedingFrames(1)-1;
break;
end
end
for i = E2:E2+25
if normalizedLogEnergy(i) > ITR
temporaryExceedingFrames=find(normalizedLogEnergy > ITR);
E2 = temporaryExceedingFrames(end)+1;
break;
end
end
end

% Plot lines for beginning and ending frames on the log energy plot
subplot(4,1,2);
hold on;
line(get(gca, 'XLim'), [ITU, ITU], 'Color', 'k', 'LineStyle', '--', ...
      'LineWidth', 1.5);
line(get(gca, 'XLim'), [ITR, ITR], 'Color', 'c', 'LineStyle', '--', ...
      'LineWidth', 1.5);
line([B1, B1], get(gca, 'YLim'), 'Color', 'g', 'LineStyle', '--', ...
      'LineWidth', 1.5);
line([E1, E1], get(gca, 'YLim'), 'Color', 'r', 'LineStyle', '--', ...
      'LineWidth', 1.5);
line([B2, B2], get(gca, 'YLim'), 'Color', 'm', 'LineStyle', '--', ...
      'LineWidth', 1.5);
line([E2, E2], get(gca, 'YLim'), 'Color', 'b', 'LineStyle', '--', ...
      'LineWidth', 1.5);
hold off;
title('Short-Time Log Energy with Hamming Window');
xlabel('Frame Number');
ylabel('Log Energy');
legend('Normalized Log Energy','ITU' , 'ITR');
subplot(4,1,3);
plot(1:numFrames, zeroCrossingRate);
hold on;
line(get(gca, 'XLim'), [IZCT, IZCT], 'Color', 'k', 'LineStyle', '--', ...
      'LineWidth', 1.5);

```

```

line([B1, B1], get(gca, 'YLim'), 'Color', 'g', 'LineStyle', '--', ''
      'LineWidth', 1.5);
line([E1, E1], get(gca, 'YLim'), 'Color', 'r', 'LineStyle', '--', ''
      'LineWidth', 1.5);
line([B2, B2], get(gca, 'YLim'), 'Color', 'm', 'LineStyle', '--', ''
      'LineWidth', 1.5);
line([E2, E2], get(gca, 'YLim'), 'Color', 'b', 'LineStyle', '--', ''
      'LineWidth', 1.5);
hold off;
title('Short-Time Zero Crossing Rate with Hamming Window');
xlabel('Frame Number');
ylabel('Zero Crossing Rate Zc');
legend('Short-Time Zero Crossing Rate','IZCT' , 'B1' , 'E1' , 'B2' ,
       'E2')

% Define the section of our interest
% Focus on the voiced unvoiced section
notSilencedzeroCrossingRate = zeroCrossingRate(B2:E2);
unvoicedAudionSectionIndeces = find(notSilencedzeroCrossingRate>IZCT)
;
% Find voicedAudionSectionFrames and unvoicedAudionSectionIndeces
unvoicedAudionSectionFrames = unvoicedAudionSectionIndeces + (B2-1);
voicedAudionSectionFrames = zeros(1,length(
    notSilencedzeroCrossingRate)-length(unvoicedAudionSectionFrames));
voicedIndex=1;
for i=1:length(notSilencedzeroCrossingRate)
check=0;
for j=1:length(unvoicedAudionSectionIndeces)
if unvoicedAudionSectionIndeces(j) == i
check = check+1;
end
end
if check == 0
voicedAudionSectionFrames (voicedIndex)=i+(B2-1);
voicedIndex = voicedIndex+1;
end
end

subplot(4,1,2);
hold on;
% Highlight voiced frames in green
scatter(voicedAudionSectionFrames, -95*ones(1,length(
    voicedAudionSectionFrames)), 'g', 'filled');
% Highlight unvoiced frames in cyan
scatter(unvoicedAudionSectionFrames, -95*ones(1,length(
    unvoicedAudionSectionFrames)), 'c', 'filled');
% Highlight unvoiced frames in red

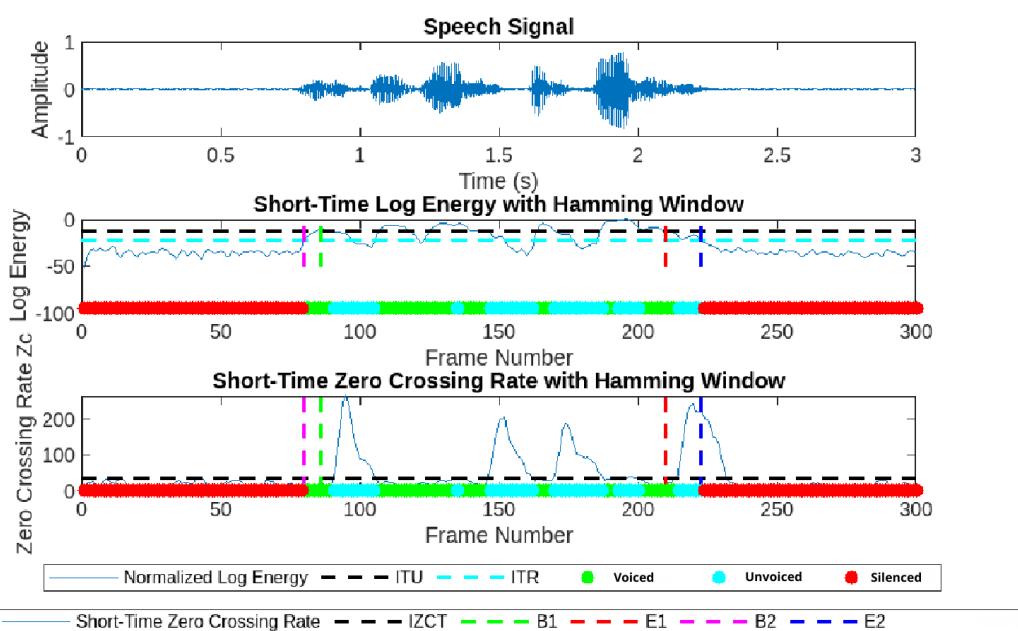
```

```

scatter([1:B2-1,E2+1:numFrames], -95*ones(1,length([1:B2-1,E2+1:
    numFrames])), 'r', 'filled');
hold off;

subplot(4,1,3);
hold on;
% Highlight voiced frames in green
scatter(voicedAudionSectionFrames, 0*ones(1,length(
    voicedAudionSectionFrames)), 'g', 'filled');
% Highlight unvoiced frames in cyan
scatter(unvoicedAudionSectionFrames, 0*ones(1,length(
    unvoicedAudionSectionFrames)), 'c', 'filled');
% Highlight unvoiced frames in red
scatter([1:B2-1,E2+1:numFrames], 0*ones(1,length([1:B2-1,E2+1:
    numFrames])), 'r', 'filled');
hold off;

```



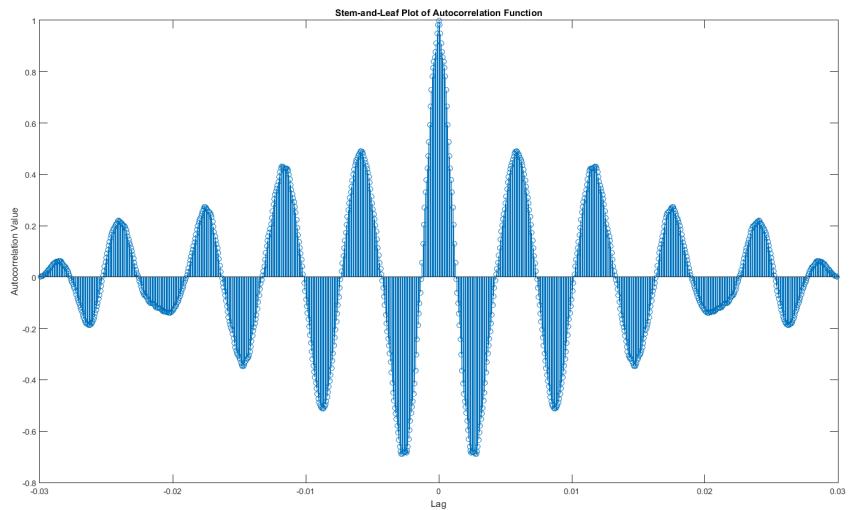
**Εικόνα 12:** Ανάλυση της ενέργειας και του ρυθμού διαβάσεων μηδενικής τιμής (Zero-Crossing Rate, ZCR).

#### ΜΕΡΟΣ Α | Άσκηση Α-4

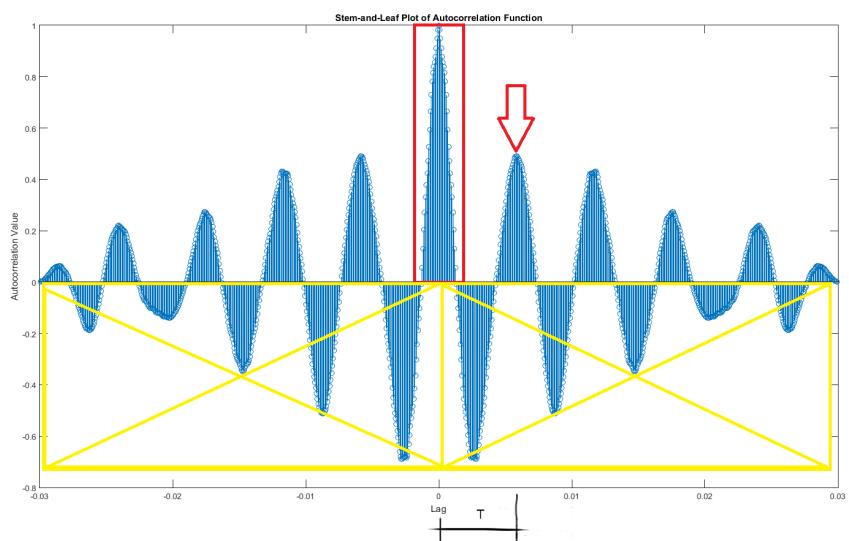
Υλοποιήστε μία μέθοδο για εύρεση της θεμελιώδους συχνότητας διέγερσης (pitch) έμφωνων ήχων (Κεφ. 10.5), και εφαρμόστε την στα έμφωνα τμήματα του ηχογραφημένου σήματος, όπως αυτά έχουν ανιχνευτεί από το προηγούμενο βήμα (Α-3). Σχεδιάστε την συχνότητα σε παράλληλο σχήμα με την ηχογραφημένη κυματομορφή, θεωρώντας ότι λαμβάνει μηδενική τιμή στα μη έμφωνα(δηλ. άφωνα) τμήματα. Για παράδειγμα, μπορείτε να χρησιμοποιήσετε αλγόριθμο με βάση την αυτοσυσχέτιση παραθυρωμένου σήματος (με ή χωρίς αποκοπή μικρομεσαίων τιμών σήματος), ή αλγόριθμο με χρήση φάσματος (άθροισμα scaled versions αυτού ή cepstrum). Προτείνεται να ομαλοποιήσετε το αποτέλεσμα στον χρόνο για εξομάλυνση ακραίων τιμών.

### Επίλυση:

Ο κώδικας αναπτύσσει μια μέθοδο για την εύρεση της θεμελιώδους συχνότητας (pitch) των έμφωνων ήχων σε ένα ηχογραφημένο σήμα, χρησιμοποιώντας τα έμφωνα τμήματα που εντοπίστηκαν από προηγούμενη ανάλυση. Η διαδικασία αρχίζει με τη διαίρεση του σήματος σε πλαίσια με επικάλυψη, εφαρμόζοντας σε κάθε πλαίσιο ένα παράθυρο Hamming για να μειωθεί το "spectral leakage". Ακολουθεί ο υπολογισμός της αυτοσυγχέτισης για κάθε πλαίσιο, και μέσω της εύρεσης του δεύτερου μεγαλύτερου peak στην αυτοσυγχέτιση, καθορίζεται η θεμελιώδης συχνότητα.



**Εικόνα 13:** Υπολογισμός της θεμελιώδους συχνότητας διέγερσης (pitch).



**Εικόνα 14:** Υπολογισμός της θεμελιώδους συχνότητας διέγερσης (pitch).

Τα αποτελέσματα της θεμελιώδους συχνότητας εφαρμόζονται μόνο στα έμφωνα τμήματα, ενώ για τα άφωνα και τα σιωπηλά τμήματα η συχνότητα θεωρείται ως μηδενική. Τα αποτελέσματα οπτικοποιούνται σε ένα διάγραμμα παράλληλα με την κυματομορφή του ηχογραφημένου σήματος, προσφέροντας μια οπτική απεικόνιση των θεμελιωδών συχνοτήτων σε σχέση με την

ομιλία. Αυτή η μεθοδολογία παρέχει μια ακριβή αναγνώριση της συχνότητας διέγερσης των έμφωνων ήχων, ενισχύοντας την ανάλυση και την κατανόηση των φωνητικών δομών του σήματος.

Στα πλαίσια αυτής της ενότητας ακολουθούμε την παρακάτω επαναληπτική διαδικασία για τον υπολογισμό των pitch των επιμέρους frames:

- **Βήμα 1: Εξαγωγή του τρέχοντος frame** Η πρώτη ενέργεια περιλαμβάνει τον διαχωρισμό του ήχου σε frames με επικάλυψη. Αυτό γίνεται για να επιτραπεί η ανάλυση του ήχου σε μικρά τμήματα. Αυτή η διαδικασία έχει ολοκληρωθεί στο προηγούμενο ερώτημα συνεπώς απλά κάνουμε χρήση της μεταβλητής του προηγούμενου ερωτήματος έτσι ώστε να προχωρήσουμε στα επόμενα βήματα.
- **Βήμα 2: Υπολογισμός της συνάρτησης αυτοσυσχέτισης** Σε αυτό το βήμα, υπολογίζεται η συνάρτηση αυτοσυσχέτισης για κάθε frame. Αυτό σημαίνει ότι υπολογίζεται η συσχέτιση του frame με καθυστερημένες εκδόσεις του ίδιου του frame σε διάφορες καθυστερήσεις lags. Αυτά υπολογίζονται κάνοντας χρήση της εντολής xcorr.
- **Βήμα 3: Εύρεση κορυφών ύρεση κορυφών** Στο τρίτο βήμα, εντοπίζονται οι κορυφές στη συνάρτηση αυτοσυσχέτισης όπως φαίνεται και στο διάγραμμα lag-autocorrelation. Οι κορυφές σε μη μηδενικές καθυστερήσεις αντιπροσωπεύουν επαναλαμβανόμενα πρότυπα ή περιοδικότητα στο σήμα. Αρχικά μηδενίζονται όλες οι αρνητικές κορυφές. Ο κώδικας μηδενίζει την κορυφή που αντιστοιχεί στο μηδενικό lag ενώ στην συνέχεια βρίσκει την ακριβώς επόμενη κορυφή. Η απόσταση μεταξύ αυτών των δυο διαδοχικών κορυφών εκφράζει την περίοδο pitch. Στο βήμα αυτό, ο διαδοχικός πλευρικός λόβος που αντιστοιχεί στη μέγιστη αυτοσυσχέτιση προσδιορίζεται ως pitch περιόδος. Αυτός ο λόγος αντιστοιχεί στην περίοδο του ήχου και χρησιμοποιείται για τον υπολογισμό της συχνότητας της βασικής τονικής σε αυτό το frame.
- **Βήμα 4: Υπολογισμός της θεμελιώδους συχνότητας διέγερσης (pitch)** Τέλος, η συχνότητα της θεμελιώδους συχνότητας υπολογίζεται ως η αντίστροφη της περιόδου του ήχου. Αυτή η συχνότητα αντιπροσωπεύει το βασικό ύψος ή την τονικότητα του ήχου στο συγκεκριμένο frame.
- **Βήμα 5: Φιλτράρισμα και Οπτικοποίηση Αποτελεσμάτων** Μετά τον υπολογισμό της θεμελιώδους συχνότητας για κάθε πλαίσιο, εφαρμόζεται φιλτράρισμα για να εξομαλύνθουν τα αποτελέσματα και να βελτιωθεί η συνολική ακρίβεια. Αυτό μπορεί να γίνει με την εφαρμογή μιας μεθόδου ομαλοποίησης ή μέσου όρου κινούμενου παραθύρου στις τιμές των συχνοτήτων, με σκοπό την εξάλειψη ακραίων τιμών και τη δημιουργία μιας πιο ομαλής καμπύλης των συχνοτήτων pitch. Στη συνέχεια, οι συχνότητες pitch που αντιστοιχούν στα έμφωνα τμήματα παραμένουν αμετάβλητες, ενώ για τα άφωνα και τα σιωπηλά τμήματα η συχνότητα ορίζεται ως μηδενική. Η οπτικοποίηση των επεξεργασμένων συχνοτήτων pitch πραγματοποιείται με την παράθεση ενός διαγράμματος που παρουσιάζει τις συχνότητες σε σχέση με την κυματομορφή του ηχογραφημένου σήματος, παρέχοντας μια ξεκάθαρη απεικόνιση της παρουσίας και της διακύμανσης του pitch στη διάρκεια του σήματος.

```
% Step1: Extract the current window -Divide the voice waveform into
% overlapping frames.
% Apply a window function to each frame to reduce spectral leakage.
% Frame segmentation - L changed because we need 30 msec
% Has been already executed - bufferedAudio

% Initialize pitch_periods array
pitch_periods = zeros(1, size(bufferedAudio, 2));
freq = pitch_periods;
autocorrValues = zeros(2*L-1, size(bufferedAudio, 2));
lags = zeros(2*L-1, size(bufferedAudio, 2));

% Iterate through frames
for i = 1:size(bufferedAudio, 2)
% Step2: Compute the short-time autocorrelation using the provided
% equation - Calculate the autocorrelation function for each frame.
% This involves computing the correlation of the frame with delayed
% versions of itself over different time lags.
% Define the analysis window (e.g., Hamming window)

% Step 3: Autocorrelation calculation
% Autocorrelation calculation for the purpose of calculating the
% fundamental frequencies - applying Hamming windows
hammingWindow = hamming(L, 'symmetric');
[autocorrValues(:,i),lags(:,i)] = xcorr(bufferedAudio(:,i).*%
    hammingWindow, 'coeff');
[autocorrValues(:,i),lags(:,i)] = xcorr(bufferedAudio(:,i), 'coeff');

% Process of locating the second peak using a temporary variable -
% making all the autocorralation values around first peak

% Step 4: Peak Picking
% Identify peaks in the autocorrelation function.
% Peaks at non-zero lags represent repeating patterns or
% periodicity in the signal.
rxx = autocorrValues(:,i);
findNegValsrxx = find(rxx < 0);
rxx(findNegValsrxx) = 0;
centerPeakWidth = find(rxx(L:end)==0,1);
rxx(L-centerPeakWidth+1 : L+centerPeakWidth) = min(rxx);

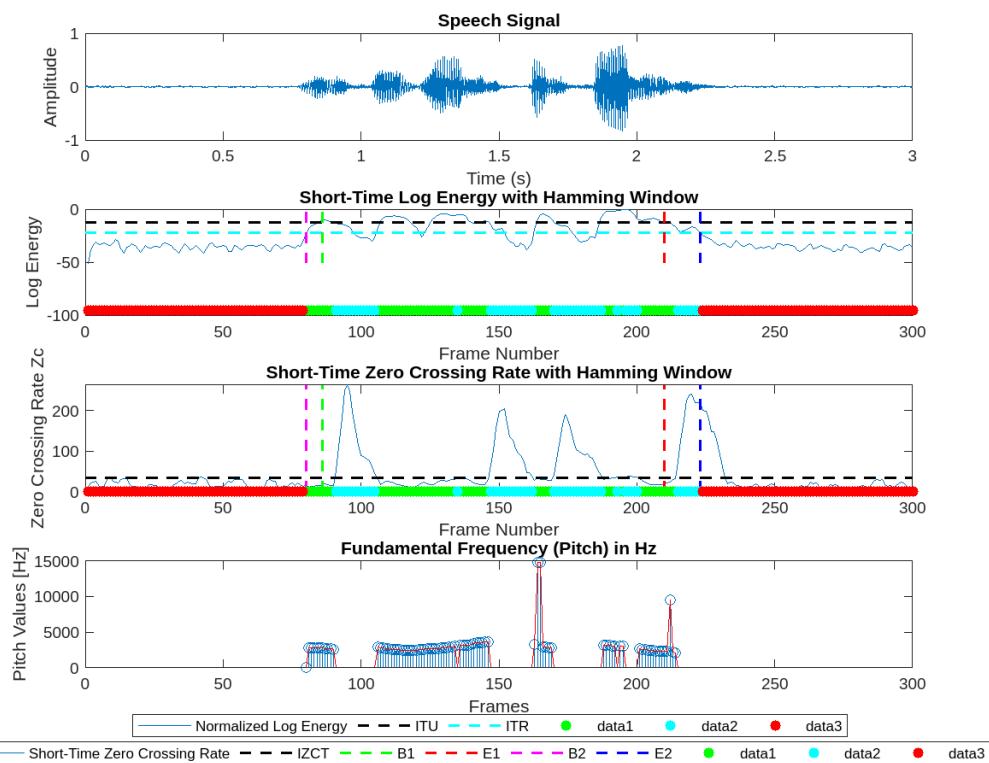
% Find the lag corresponding to the maximum autocorrelation
[max_value, max_index] = max(rxx);

% Step 5 : Pitch Period Extraction:
% Determine the lag corresponding to the highest peak in the
% autocorrelation function. This lag represents the pitch period of
% the frame. Convert lag to pitch period
pitch_periods(i) = (abs(max_index - L))*(L/fs);
freq(i) = fs / pitch_periods(i) ;
end
```

```
% Apply pitch to voiced segments
pitchPeriods = pitch_periods;
pitchPeriods(unvoicedAudionSectionFrames)=0;
pitchPeriods([1:B2,E2:numFrames])=0;
frequencies = freq;
frequencies(unvoicedAudionSectionFrames)=0;
frequencies([1:B2,E2:numFrames])=0;

figure(8);

subplot(4,1,4);
%plot(recordingDuration*(0:length(pitchValues) - 1)/(length(
%    pitchValues) - 1), pitchValues);
stem(voicedAudionSectionFrames,frequencies(voicedAudionSectionFrames))
);
hold on;
plot(frequencies,'r');
hold off;
title('Fundamental Frequency (Pitch) in Hz');
xlabel('Frames');
ylabel('Pitch Values [Hz]');
```



**Ευκόνα 15:** Υπολογισμός της θεμελιώδους συχνότητας διέγερσης (pitch).

## ΜΕΡΟΣ Α | Άσκηση Α-5

Απομονώστε τμήμα διάρκειας 30 msec που αντιστοιχεί σε κάποιο έμφωνο και σε κάποιο άφωνο τμήμα από το ηχογραφημένο σήμα. Υπολογίστε το διάνυσμα χαρακτηριστικών LPC για τα δύο κομμάτια για τάξη φίλτρου  $p = 8, 12, 16$ . Υπολογίστε το λάθος εκτίμησης κάθε φορά και σχεδιάστε το μέτρο της all-pole συνάρτησης μεταφοράς με βάση το μοντέλο LPC (Κεφ. 9).

### Επίλυση:

Τρέχουμε όλη την διαδικασία που προηγήθηκε διαζύγωντας την αλλαγή στο μέθεγος των frames. Αυτό που γίνεται είναι η αλλαγή των παραμέτρων του μεγέθους των frames αλλά και αυτής την μετατόπισης των frames. Πιο συγκεκριμένα:

```
L = 480; % 30 msec frame duration
R = 160; % 10 msec frame shift

% Parameters for short-time analysis
frameSize = L; % 300 Samples at Fs=10kHz
frameOverlap = L - R; % 100 Samples at Fs=10kHz
```

Με αυτόν τον τρόπο έχει αλλάξει το μέγεθος των frames και έχει περιοριστεί στα 30msec, ώστε να καλυφθούν οι ανάγκες αυτού του ερωτήματος.

Επίσης με τις ακόλουθες δυο εντολές, επιλέγουμε να κάνουμε χρήση του τελευταίου έμφωνου frame και του πρώτου άφωνου frame.

```
chosenVoiced = voicedAudionSectionFrames (end);
chosenUnvoiced = unvoicedAudionSectionFrames (1);
```

```
%A5

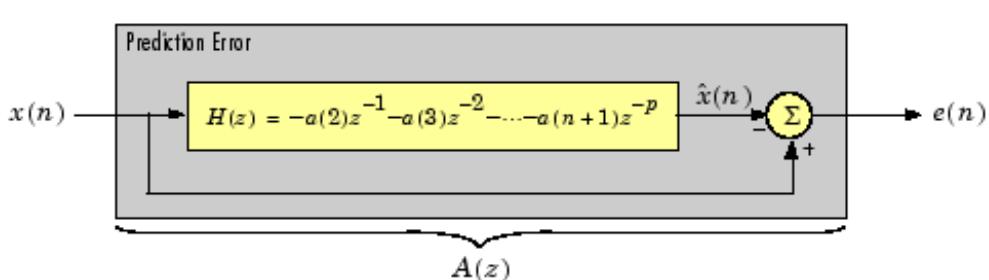
% Every frames has a duration of 30msec, the first unvoiced and the
% last
% voiced frames have been chosen
chosenVoiced = voicedAudionSectionFrames (end);
chosenUnvoiced = unvoicedAudionSectionFrames (1);
j=0;
Hfilter=0;
Hfilter8=0;
Hfilter12=0;
Hfilter16=0;
for p = [8,12,16]
Hfilter=0;
j = j+1;
windowSize = 5;
%b = (1/windowSize)*ones(1,windowSize);
a1 = lpc(bufferedAudio(:,chosenVoiced),p);
a2 = lpc(bufferedAudio(:,chosenUnvoiced),p);
est_xv = filter([0 -a1(2:end)],1,bufferedAudio(:,chosenVoiced));
est_xu = filter([0 -a2(2:end)],1,bufferedAudio(:,chosenUnvoiced));
figure(14);
subplot(3,1,j);
```

```

plot(1:L,bufferedAudio(:,chosenVoiced),1:L,est_xv,'--',1:L,abs(
    bufferedAudio(:,chosenVoiced)-est_xv),'+' )
grid
title(['Voiced Segment for p= ' num2str(p)])
xlabel('Sample Number')
ylabel('Amplitude')
legend('Original signal','LPC estimate','error')
figure(15);
subplot(3,1,j);
plot(1:L,bufferedAudio(:,chosenUnvoiced),1:L,est_xu,'--',1:L,abs(
    bufferedAudio(:,chosenUnvoiced)-est_xu),'+' )
grid
title(['Unvoiced Segment for p= ' num2str(p)])
xlabel('Sample Number')
ylabel('Amplitude')
legend('Original signal','LPC estimate','error')
for i=1:length(a1)-1
num2 = [a1(i) zeros(1,length(a1)-i) ];
den2 = [ 0 1 ]; % denominator coefficients
Hfilter = Hfilter + tf(den2,num2, 1/targetFs )^-i;
%Hfilter=Hfilter+tf([ a1(i) zeros(1,length(a1)-i) ],1, 1/targetFs)
;
end

%Hfilter=Hfilter*tf([1 zeros(1,length(a1))],[0 1],1/targetFs)^-1;
if p==8
Hfilter8=1/(1-Hfilter);
elseif p==12
Hfilter12=1/(1-Hfilter);
elseif p==16
Hfilter16=1/(1-Hfilter);
end
end

```



Εικόνα 16

χ αντιστοιχεί στο bufferedAudio του voiced ή του unvoiced σήματος

Στην Εικόνα 16 αυτή παρατηρούμε ότι κάνουμε χρήση ενός φίλτρου, αξιοποιώντας του συντελεστές χαρακτηριστικών LPC, που τα βρίσκουμε τρέχοντας την εντολή

```
a1 = lpc(bufferedAudio(:,chosenVoiced),p);
```

Στην συνέχεια κάνουμε χρήση της εντολής filter έτσι ώστε να κάνουμε μια εκτίμηση του σήματος , κάνοντας εισαγωγή του αρχικού σήματος για το συγκεκριμένο frame και τελικά κάνουμε απεικόνηση του αρχικού σήματος, του εκτιμώμενου σήματος , καθώς και του σφάλματος μεταξύ αυτών.

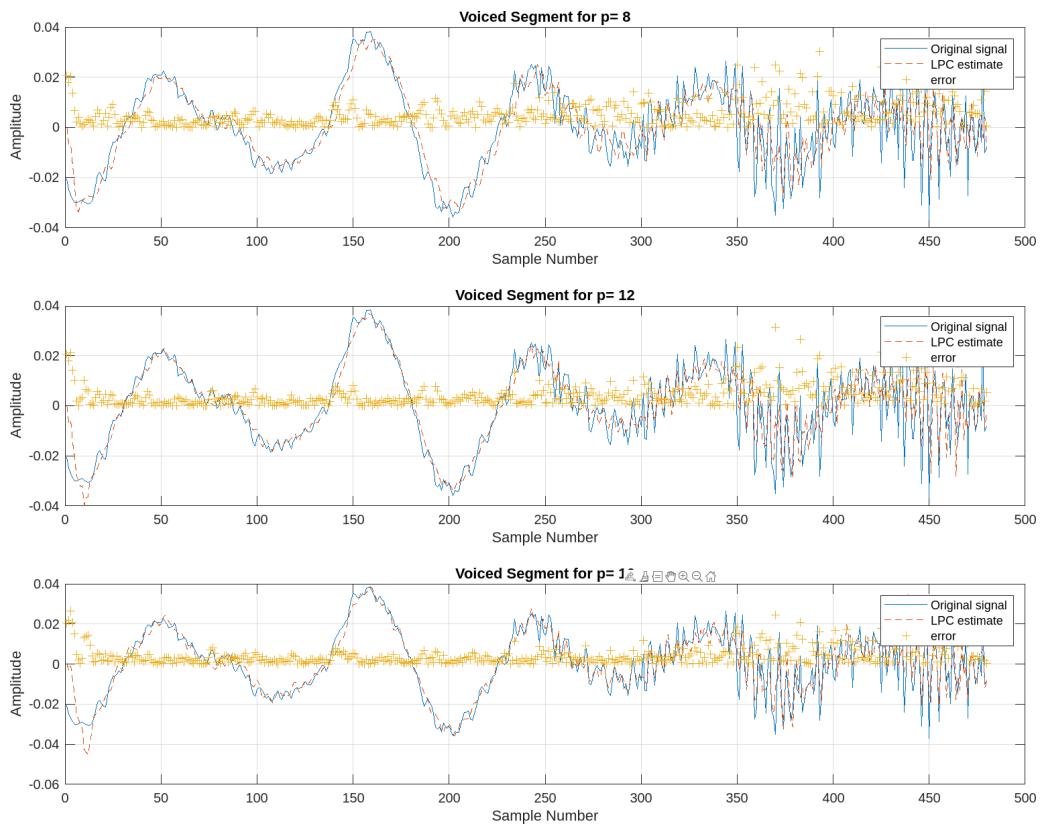
Το ίδιο φίλτρο θα μπορούσαμε να το δημιουργήσουμε δημιουργώντας τις ακόλουθες συναρτήσεις μεταφοράς:

```
>> Hfilter8
Hfilter8 =
-----  
z^9  
z^9 - z^8 + 0.3403 z^7 - 0.07885 z^6 + 0.3832 z^5 + 0.2961 z^4 + 0.312 z^3 - 0.04441 z^2 - 0.191 z  
Sample time: 6.25e-05 seconds
Discrete-time transfer function.
Model Properties
>> Hfilter12
Hfilter12 =
-----  
z^13  
  
z^13 - z^12 + 0.3402 z^11 - 0.1267 z^10 + 0.4199 z^9 + 0.1528 z^8 + 0.3031 z^7 - 0.02251 z^6 + 0.06252 z^5 - 0.05942 z^4 + 0.3922 z^3  
- 0.2442 z^2 + 0.01245 z  
Sample time: 6.25e-05 seconds
Discrete-time transfer function.
Model Properties
>> Hfilter16
Hfilter16 =
-----  
z^17  
  
z^17 - z^16 + 0.1569 z^15 - 0.1535 z^14 + 0.3681 z^13 + 0.2606 z^12 + 0.4079 z^11 - 0.01601 z^10 + 0.02931 z^9 + 0.02801 z^8 + 0.5156 z^7  
- 0.05458 z^6 + 0.02691 z^5 - 0.3023 z^4 - 0.2612 z^3 - 0.2807 z^2 + 0.03998 z  
Sample time: 6.25e-05 seconds
Discrete-time transfer function.
```

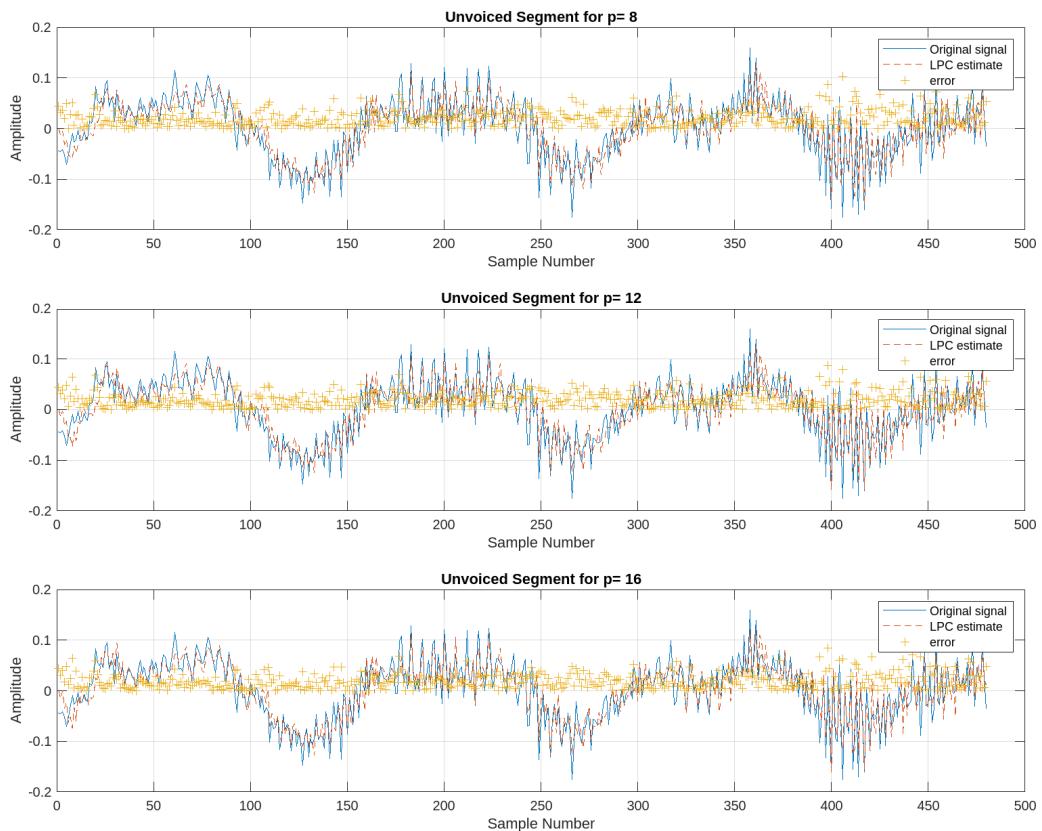
```
for i=1:length(a1)-1
num2 = [a1(i) zeros(1,length(a1)-i) ];
den2 = [ 0 1 ]; % denominator coefficients
Hfilter = Hfilter + tf(den2,num2, 1/targetFs )^-i;
%Hfilter=Hfilter+tf([ a1(i) zeros(1,length(a1)-i) ],1, 1/targetFs)
;
end

%Hfilter=Hfilter*tf([1 zeros(1,length(a1))],[0 1],1/targetFs)^-1;
if p==8
Hfilter8=1/(1-Hfilter);
elseif p==12
Hfilter12=1/(1-Hfilter);
elseif p==16
Hfilter16=1/(1-Hfilter);
end
```

με αυτόν τον τρόπο ορίζονται οι all-pole συναρτήσεις μεταφορά όπως και φαίνεται στην παρακάτω εικόντα για κάθε μια από τις περιπτώσεις p=8,12,16



Εικόνα 17



Εικόνα 18

## ΜΕΡΟΣ Β

Στο τμήμα αυτό της εργασίας θα δημιουργήσετε συνθετικό ήχο με συχνότητα δειγματοληψίας στα 10 kHz και με συνολική διάρκεια 3 sec, που θα αποτελείται από τη σειρά των εξής έξι φωνηέντων: /AO/, /IY/, /UH/, /EH/, /AH/, /IH/, διάρκειας 0.5 sec το καθένα. Για το σκοπό αυτό, θα χρησιμοποιήσετε το μοντέλο παραγωγής της φωνής, ακολουθώντας τα βήματα του Κεφ. 8.3 του βιβλίου και τη βοήθεια του Πίνακα 3.4 (Κεφ. 3.4).

### ΜΕΡΟΣ Β | Άσκηση Β-1

Θεωρήστε περίοδο pitch ίση με 8 msec, που για περίοδο δειγματοληψίας  $T = 10 - 4$  sec, μας δίνει  $N_p = 80$ . Ως διέγερση, θεωρούμε το ελαφρώς τροποποιημένο σήμα της εξίσωσης (8.43)  $p[n] = \sum_{k=0}^{\infty} \beta^k \delta[n - kN_p]$  που στο πεδίο του μετασχηματισμού αντιστοιχεί σε  $P[z] = \sum_{k=0}^{\infty} \beta^k z^{kN_p} = \frac{1}{1 - bz^{-N_p}}$  Σχεδιάστε τα  $p[n]$  (σε πεπερασμένο χρονικό διάστημα), το φάσμα  $P(e^{j\omega})$  (χρησιμοποιώντας DFT επαρκούς μήκους και δείχνοντας το αποτέλεσμα σε κλίμακα συχνοτήτων [0, 5 kHz]), όπως και το διάγραμμα πόλων-μηδενικών της  $P(z)$ .

#### Επίλυση:

Αρχικά ορίζουμε την περίοδο pitch που είναι 8msec και την περίοδο δειγματοληψίας που είναι  $10^{-4}$ . Έπειτα πρέπει να δημιουργήσουμε το σήμα  $p[n]$ , που είναι ένας μονόπλευρος σχεδόν περιοδικός σειριακός παλμός (one-sided quasi-periodic impulse train) και αυτό εκφράζεται από την παρακάτω εξίσωση :

```

num2 = [1, 0]; % numerator
coefficients
den2 = [1, zeros(1, 79), -0.9999]; % denominator
coefficients
sys2 = tf(num2, den2, 1/Fs, 'Variable', 'z^-1');
Yp=fft(pn,length(pn)); % DFT on input signal
freq = (0:length(pn)-1)*(Fs/length(pn)); % Frequency domain
% Amplitude P(e^(j?))
Pp = abs(Yp);
figure(1);
subplot(3,1,2)
% Results representation for the range 0 to 5 kHz
plot(freq, Pp);
title('Voiced excitation spectrum - Magnitude of Frequency Response w
      / Fundamental Frequency F_0 = 125 Hz');
xlabel('Frequency (Hz)');
ylabel('|P(e^{j\omega})|');
xlim([0 5000]);
grid on;

```

Στην συνέχεια διεξάγεται μετασχηματισμός Fourier πάνω στο ήδη αναπτηγμένο σήμα και παρατηρούμε στο διάγραμμα ότι η θεμελιώδης συχνότητα είναι 125 Hz, πιο συγκεκριμένα εστιάζοντας στην απόσταση μεταξύ των κορυφών μπορούμε να παρατηρήσουμε ότι μεσολαβεί απόσταση 125 Hz. Γίνεται λοιπόν αναπαράσταση του εύρους από 0 έως 5000.

```
Pz=sys2;
```

```
% Get the poles and zeros
[polesysp, zerosysp] = pzmap(Pz);
figure(1)
subplot(3,1,3)
% Plot zeros with a specific color, say, red 'r'
plot(real(zerosysp), imag(zerosysp), 'ro', 'MarkerSize', 10, 'LineWidth', 2);
hold on;
% Plot poles with a different color, say, blue 'b'
plot(real(polesysp), imag(polesysp), 'bx', 'MarkerSize', 10, 'LineWidth', 2);
title('Zeros and Poles Plot - Voiced Excitation P(z) ');
xlabel('Real Part');
ylabel('Imaginary Part');
grid on;
theta = 0:0.01:2*pi;
x = cos(theta);
y = sin(theta);
plot(x, y, 'r:');
hold off
legend('Zeros', 'Poles');
```

Ενώ στην συνέχεια προχωράμε στον μετασχηματισμό  $z$ . Αυτό ορίζεται στα πλαίσια της βιβλιογραφίας και για αυτόν τον σκοπό εισάγουμε χειροκίνητα την εξίσωση στο περιβάλλον του μάτλαμπ. Πιο συγκεκριμένα έχει ήδη οριστεί η συνάρτηση μεταφοράς στο προηγούμενο κομμάτι του αλγορίθμου που παρατέθηκε :

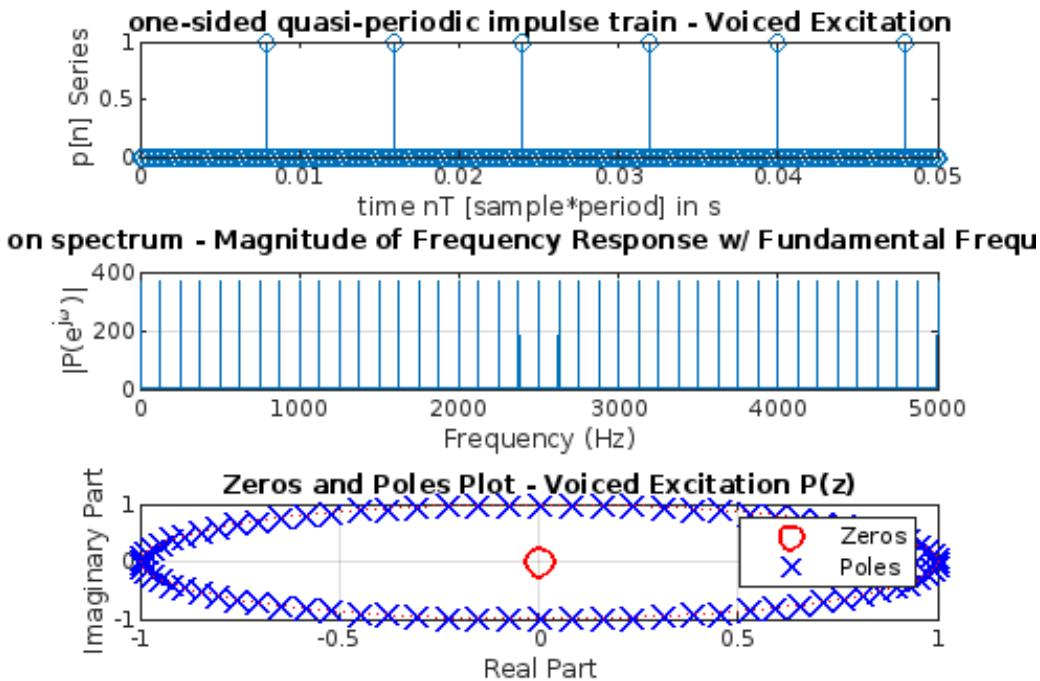
```
num2 = [1, 0]; % numerator
coefficients
den2 = [1, zeros(1, 79), -0.9999]; % denominator
coefficients
sys2 = tf(num2, den2, 1/Fs, 'Variable', 'z^-1');
```

και αυτό εναπόκειται στην εξίσωση 8.43 του συγκράμματος

Πάντως η θεμελιώσης συχνότητα επιβεβαιώνεται ότι είναι αναμενόμενη καθώς ισχύει  $10000/Np = 125Hz$

$Np = 80$

```
figure(1)
subplot(3,1,1)
stem(n/Fs, pn);
title('one-sided quasi-periodic impulse train - Voiced Excitation');
xlabel('time nT [sample*period] in s');
ylabel('p[n] Series');
xlim([0 0.05])
```



Εικόνα 19

## ΜΕΡΟΣ Β | Άσκηση Β-2

Θεωρήστε το μοντέλο του γλωτιδιακού παλμού (glottal pulse) της εξίσωσης (8.39), δηλ.:

$$g[n] = \begin{cases} 0.5 (1 - \cos(\pi(n+1)/25)), & \text{για } 0 \leq n \leq 24 \\ \cos(0.5\pi(n-24)/10), & \text{για } 25 \leq n \leq 33 \\ 0, & \text{αλλού.} \end{cases}$$

Σχεδιάστε το  $g[n]$ , το φάσμα του  $G(e^{j\omega})$  (όπως παραπάνω), και το διάγραμμα πόλων-μηδενικών της  $G(z)$ .

**Επίλυση:**

Αφού ολοκληρώσαμε με την διέγερση προχωράμε στην μοντελοποίηση του γλωτιδιακού παλμού. Ο κώδικας που σχετίζεται με την δημιουργία της ιστορία του είναι ο ακόλουθος:

```
% Part 2
% glottal pulse - Rossenberg model
N1 = 25;
N2 = 10;
% glottal pulse sections
horizonn = 6; % Parameter of the horizon to be included
ng = linspace(0,N1+N2-2+horizonn,N1+N2-1+horizonn);
gn = zeros(1,size(ng,2));
for i=ng
if i<=N1-1
gn(i+1) = 0.5*(1 - cos(pi*(i+1) / N1));
```

```

elseif i>=N1 && i<=N1+N2-2
gn(i+1) = cos(0.5*pi*(i-(N1-1))/N2);
else
gn(i+1) = 0;
end
end
%gn=[0.5*(1-cos(pi*(gn1+1)/gN1)), cos(0.5*pi*(gn2+1-gN1)/gN2)]
figure(2);
% Plot of glottal pulse
subplot(3,1,1)
stem(1000*ng/Fs, gn);
    %10^-4 because Fs and 10^3 because ms
title('Glottal Pulse');
xlabel('time nT [sample*period] in ms');
ylabel('Amplitude');

```

όπου σύμφωνα με την εξίσωση 8.38 του συγγράμματος επιδιώκουμε να εξάγουμε μια ιστορία για αυτήν την μεταβλητή. Βλέπουμε ότι έχουμε 3 διαφορετικές περιπτώσεις.

Αφού λοιπόν απεικονίσουμε την ιστορία της εξίσωσης, κάνουμε χρήση αυτών των δεδομένων διεξάγοντας έναν μετασχηματισμό fourier πάνω στην υπάρχουσα ιστορία. Ο Κώδικας για αυτήν την διεργασία είναι ο ακόλουθος:

```

Yg=fft(gn,length(gn)); % DFT on
    input signal
freq = (0:length(gn)-1)*(Fs/length(gn)); % Frequency domain
% Amplitude P(e^(j?))
Pg = abs(Yg);
figure(2);
subplot(3,1,2)
% Results representation for the range 0 to 5 kHz
plot(freq, log(Pg));
title('Glottal pulse spectrum');
xlabel('Frequency (Hz)');
ylabel('log_{e}(|G(e^{j\omega})|)');
xlim([0 5000]);
grid on;

```

Και το επόμενο βήμα είναι ενας μετασχηματισμός z , κάνοντας χρήση της τυπικής διαδικασίας μετασχηματισμού z σύμφωνα δηλαδή με την παρακάτω εξίσωση

$$ztransform : F(z) = \sum_{n=0}^{\infty} \frac{f(n)}{z^n}$$

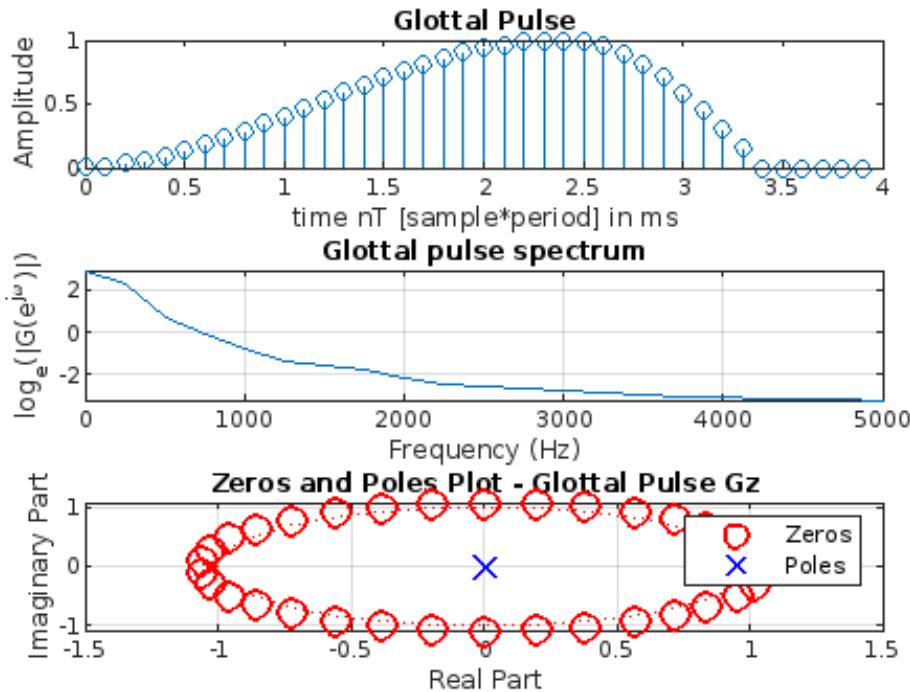
Και ο κώδικας για αυτό το κομμάτι είναι ο εξής :

```

for i=N+1
Gz=Gz+tf([ gn(i) zeros(1,N(end)-i) ],1, 1/Fs);
end
% Opposite direction sequence entails that we need to
% multiply by some tf z^-39
Gz=Gz*tf([1 zeros(1,N(end))],[0 1],1/Fs)^-1;

```

παρατηρούμε ότι όπως εδώ έτσι και στην προηγούμενη περίπτωση γίνεται χρήση συναρτήσεων μεταφοράς.



Εικόνα 20

## ΜΕΡΟΣ Β | Άσκηση Β-3

Στη συνέχεια θεωρήστε το σύστημα φωνητικής οδού (vocal tract) ως ένα φίλτρο της μορφής της εξίσωσης (8.41), δηλ.:

$$V(z) = \frac{1}{\prod_{\kappa=1}^K (1 - 2e^{-2\pi\sigma_\kappa T} \cos(2\pi F_\kappa T) z^{-1} + e^{-4\pi\sigma_\kappa T} z^{-2})},$$

όπου ο αριθμός των formants είναι  $K = 3$  (αντί για τα πέντε της (8.41)), οι συχνότητές τους (για το φωνήν /AO/, όπως δίνονται στον Πίνακα 3.4) είναι:  $F_1 = 570$  Hz,  $F_2 = 840$  Hz,  $F_3 = 2410$  Hz, και τα εύρη (bandwidths) αυτών ίσα με  $2\sigma_1 = 60$  Hz,  $2\sigma_2 = 100$  Hz, και  $2\sigma_3 = 120$  Hz. Σχεδιάστε το φάσμα  $V(e^{j\omega})$  (όπως παραπάνω), και το διάγραμμα πόλων-μηδενικών της  $V(z)$ .

## Επίλυση:

Στην συνέχεια δημιουργούμε ένα φίλτρο ως σύστημα φωνητικής οδού (vocal tract), κάνοντας χρήση της εξίσωση 8.40 του συγγράμματος. Ολόκληρη η διαδικασία εξαγωγής των αποτελεσμάτων αυτού του ερωτήματος είναι βασισμένη πάνω σε αυτήν την εξίσωση

$$V(z) = \frac{1}{\prod_{\kappa=1}^K (1 - 2e^{-2\pi\sigma_\kappa T} \cos(2\pi F_\kappa T) z^{-1} + e^{-4\pi\sigma_\kappa T} z^{-2})}$$

```
% 1 /AO/
Fk=[570,840,2410];
sigmak=[60,100,120]./2;
Vz_ao=1;
for i=1:length(Fk)
%Vzz=Vzz*(1-2*exp(-2*pi*sigmak(i)*(1/Fs))*cos(2*pi*Fk(i)*(1/Fs))*z^-1
%+ exp(-4*pi*sigmak(i)*(1/Fs))*z^-2);
Vz_ao=Vz_ao*tf([1 0 0],[1, -2*exp(-2*pi*sigmak(i)*(1/Fs))*cos(2*pi*Fk
(i)*(1/Fs)), exp(-4*pi*sigmak(i)*(1/Fs))],1/Fs);
```

```
end
```

Άρα αφού εξάγουμε μια συνάρτηση μεταφοράς κάνοντας χρήση του παραπάνω κομματιού του αλγορίθμου εξάγουμε την ιστορία κάνοντας εφαρμογή κρουστικής απόκρισης πάνω στην συνάρτηση μεταφοράς - φίλτρο vocal tract. Σε αυτό το σημείο θα ήταν δόκιμο το να γίνει αναφορά στην λειτουργία αυτής της εντολής. Η εντολή impulse στο MATLAB υπολογίζει την απόκριση κρουστικής απόκρισης ενός συστήματος. Όταν εφαρμόζεται σε ένα γραμμικό σταθερό χρονικά συστημα (LTI), υπολογίζει και επιστρέφει την έξοδο του συστήματος όταν η είσοδος είναι μια συνάρτηση κρουστικής, η οποία είναι μηδενική παντού εκτός από το χρόνο μηδέν, όπου έχει τιμή 1.

Ο κώδικας για την διεκπεραίωση αυτής της διαδικασίας είναι ο ακόλουθος:

```
figure(3)
subplot(3,1,1)
% Find the impulse response
[vnao, nvnao] = impulse(Vz_ao);
%[vn, nvn] = impulse(Vz, [ng(1)*1000/Fs ng(end)*1000/Fs]);
% Plot the impulse response
stem(nvnao*1000, vnao); %1/Fs because T and
                           1000 because ms
title('Vocal Tract Impulse Response v[n] /AO/');
ylabel('Amplitude v[n] series ');
xlabel('time nT [sample*period] in ms'),
```

Ενω στην συνέχεια καλούμαστε να κάνουμε μετασχηματισμό fourier πάνω σε αυτά τα δεδομένα ώστε να εξάγουμε το φάσμα  $V(e^j)$

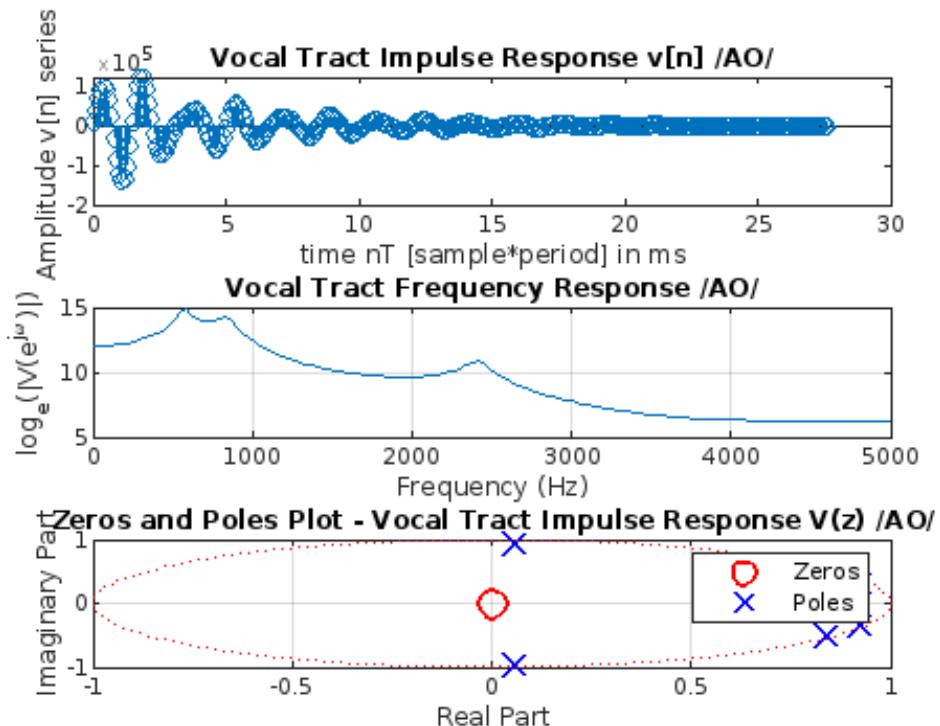
```
Yvao=fft(vnao,length(vnao)); % DFT on input signal
freqvao = (0:length(vnao)-1)*(Fs/length(vnao)); % Frequency domain
% Amplitude P(e^(j?))
Nvao = abs(Yvao);
figure(3);
subplot(3,1,2)
% Results representation for the range 0 to 5 kHz
plot(freqvao, log(Nvao));
title('Vocal Tract Frequency Response /AO/');
xlabel('Frequency (Hz)');
ylabel('log_{e}(|V(e^{j\omega})|)');
xlim([0 5000]);
grid on;
```

Και τελικά γίνεται αναπαράσταση των πόλων και των μηδενικών:

```
% Get the poles and zeros
[polesysao, zerosysao] = pzmap(Vz_ao);
figure(3)
subplot(3,1,3)
% Plot zeros with a specific color, say, red 'r'
plot(real(zerosysao), imag(zerosysao), 'ro', 'MarkerSize', 10,
      'LineWidth', 2);
hold on;
```

```
% Plot poles with a different color, say, blue 'b'
plot(real(polessyao), imag(polessyao), 'bx', 'MarkerSize', 10, ...
      'LineWidth', 2);
title('Zeros and Poles Plot - Vocal Tract Impulse Response V(z) /AO/');
    );
xlabel('Real Part');
ylabel('Imaginary Part');
grid on;
theta = 0:0.01:2*pi;
x = cos(theta);
y = sin(theta);
plot(x, y, 'r:');
hold off
legend('Zeros', 'Poles');
```

Να σημειωθεί σε αυτό το σημείο ότι γίνεται χρήση μονο των τριών πρώτων συχνοτήτων formants από τον πίνακα του συγγράμματος και για αυτόν τον λόγο παρατηρούμε ότι στο διάγραμμα πόλων και μηδενικών έχουμε ένα σύνολο με 6 σημεία στον χάρτη.



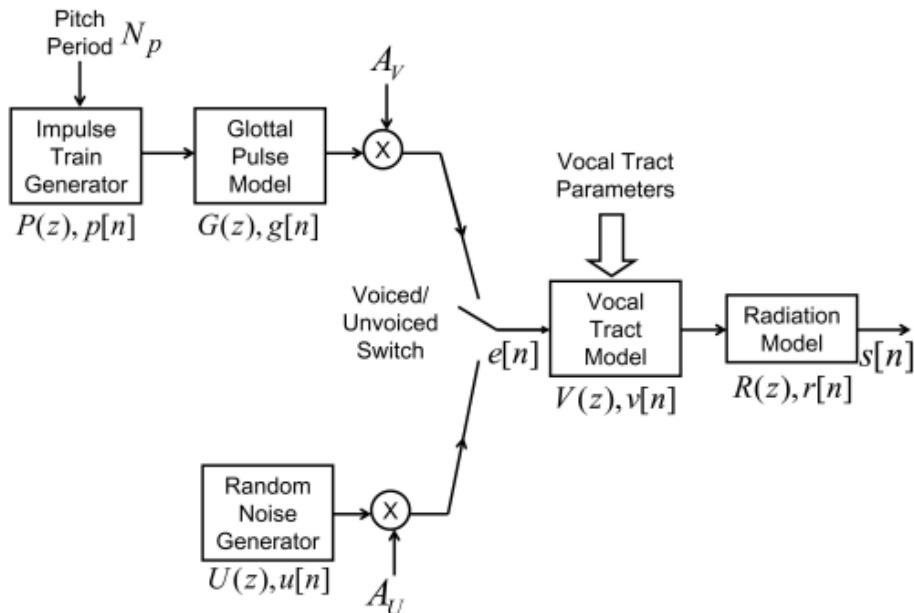
**Εικόνα 21**

## ΜΕΡΟΣ Β | Ασκηση Β-4

Το τελευταίο κομμάτι του μοντέλου του συστήματος αντιστοιχεί στο φορτίο ακτινοβολίας (radiation load), με  $r[n] = \delta[n] - 0.96\delta[n-1]$ . Σχεδιάστε τα  $r[n]$  (πεδίο χρόνου), φάσμα  $R(e^{j\omega})$  (όπως παραπάνω), και το διάγραμμα πόλων-μηδενικών της  $R(z)$ .

### Επίλυση:

Για την δημιουργία όλου λοιπόν του μοντέλου παραγωγής έμφωνων ήχων πρέπει να γίνει και η μοντελοποίηση του Rz. Αυτό το block diagram σχετίζεται με την πάνω οδό που περιλαμβάνει εκτός από τις προγούμενες συναρτήσεις μεταφοράς και την Rz.



**Εικόνα 22**

Αυτό το μοντέλο δημιουργείται από τον παρακάτω κώδικα:

```
% B4
Rz=tf( [1 -0.96] , [1 0] ,1/Fs);
nr=ng;
rn=zeros(1,length(nr));
for i = nr
if i == 1
rn(i+1) = -0.96;
else
rn(i+1)=0;
end
end
% Plot the series using a stem plot
figure(4)
subplot(3,1,1)
stem(1000*nr/Fs, rn);
title('Radiation Load');
xlabel('time nT [sample*period] in ms');
ylabel('r[n] Series');
```

Αυτό το παρατηρούμε στην εξίσωση 8.41 του συγγράμματος, όπου και ορίζεται η εξίσωση της συνάρτησης μεταφορά του φορτίου ακτινοβολίας (radiation load)

Και πάλι κάνουμε εφαρμογή της εντολής impulse πάνω στην συνάρτηση μεταφορά που ορίζεται από την παραπάνω εξίσωση και από εδώ εξάγουμε την ιστορία της απόκρισης. Κάνουμε

The model for the radiation load is the simple first difference system

$$R(z) = 1 - \gamma z^{-1}. \quad (8.41)$$

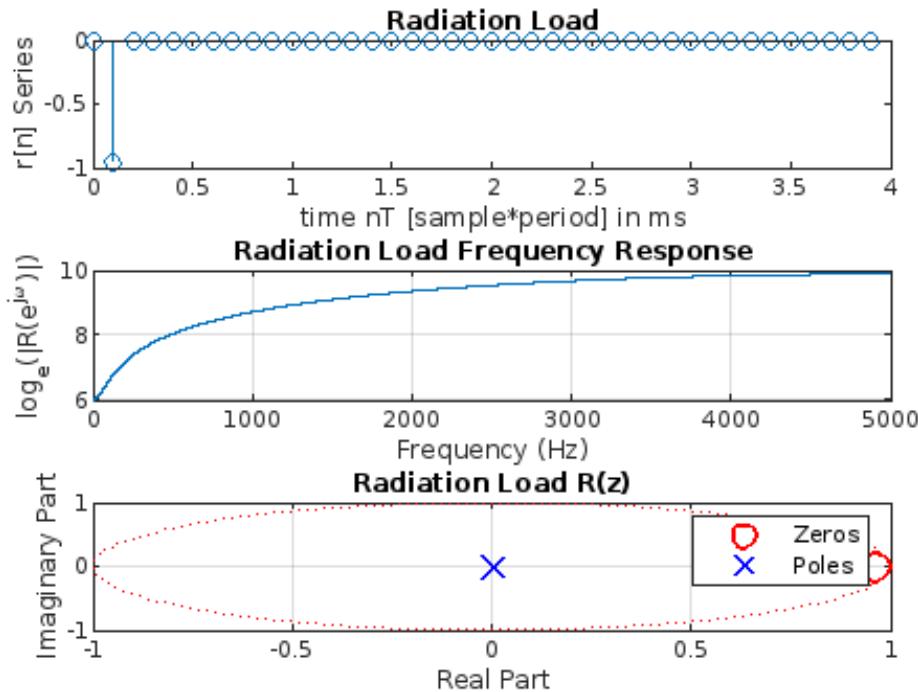
The corresponding impulse response  $r[n] = \delta[n] - \gamma\delta[n-1]$  is shown in Figure 8.13(c) and the single zero is shown in Figure 8.14(c) for the specific value of  $\gamma = 0.96$ .

## Εικόνα 23

χρήση αυτού του σήματος λοιπόν για την εξαγωγή μετασχηματικού Fourier για την απεικόνιση του φάσματος  $R(e^j)$

Και τελικά γίνεται γραφική αναπαράσταση των πόλων και των μηδενικών πάνω στην συνάρτηση μεταφοράς  $Rz$

```
% Get the poles and zeros
[polesysr, zerosysr] = pzmap(Rz);
figure(4)
subplot(3,1,3)
% Plot zeros with a specific color, say, red 'r'
plot(real(zerosysr), imag(zerosysr), 'ro', 'MarkerSize', 10, ...
    'LineWidth', 2);
hold on;
% Plot poles with a different color, say, blue 'b'
plot(real(polessysr), imag(polessysr), 'bx', 'MarkerSize', 10, ...
    'LineWidth', 2);
title('Radiation Load R(z)');
xlabel('Real Part');
ylabel('Imaginary Part');
grid on;
theta = 0:0.01:2*pi;
x = cos(theta);
y = sin(theta);
plot(x, y, 'r:');
hold off
legend('Zeros', 'Poles');
```



Εικόνα 24

## ΜΕΡΟΣ Β | Άσκηση Β-5

Στη συνέχεια, θεωρήστε το σήμα φωνής που προκύπτει από την συνέλιξη των παραπάνω, δηλ. το:  $s[n] = Ap[n] * g[n] * v[n] * r[n]$ , όπου το A είναι ένα σταθερό κέρδος, π.χ.,  $A = 5000$ . Σχεδιάστε το φάσμα  $S(e^{j\omega})$  του σήματος, όπως και το διάγραμμα πόλων-μηδενικών της  $S(z)$ .

## Επίλυση:

Τελικά το ολικό μοντέλο αναπαράσταση έμφωνων, που όπως αναφέρθηκε και σε προηγούμενο σχήμα είναι σειριακά συσχετισμένα μεταξύ τους δημιουργούνται πολλαπλασιάζοντας τις επιμέρους συναρτήσεις μεταφορά μεταξύ τους:

```
% B5
%Again is defined in the beginning of the script
Sz_ao=Gz*Vz_ao*Rz*Again*Pz;
Hz_ao=Gz*Vz_ao*Rz*Again;
%[ssn,ttn]=impulse(Sz,0.039);
%[hn,~]=impulse(Hz,0.039);
[ssnao,ttnao]=impulse(Sz_ao);
[hnao,~]=impulse(Hz_ao);
figure(5)
subplot(3,1,1)
plot(ttnao*1000, ssnao);
title('Output of speech model system s[n] /AO/');
xlabel('time nT [sample*period] in ms');
ylabel('s[n] Series');
xlim([0 40])
```

Από αυτό το κομμάτι του κώδικα προκύπτει η ιστορία της απόκρισης αυτού του σήματος για το φωνίεν ΑΟ

Στην συνέχεια κάνουμε χρήση και πάλι του σήματος για να εφαρμόσουμε μετασχηματισμό Fourier πάνω του. Το κομμάτι του αλγορίθμου που αντιστοιχεί σε αυτό είναι:

```

Ysao=fft(ssnao,length(ssnao));
Yhao=fft(hnao,length(hnao));
freqsao = (0:length(ssnao)-1)*(Fs/length(ssnao));
freqhao = (0:length(hnao)-1)*(Fs/length(hnao));
% Amplitude P(e^(j?))
Psao = abs(Ysao);
Phao = abs(Yhao);
figure(5);
subplot(3,1,2)
% Results representation for the range 0 to 5 kHz
plot(freqsao, log(Psao));
hold on;
plot(freqhao, log(Phao), '--r');
hold off;
title('Output of speech model system /AO/');
xlabel('Frequency (Hz)');
ylabel('log_{e}(|S(e^{j\omega})|)');
xlim([0 5000]);
grid on;

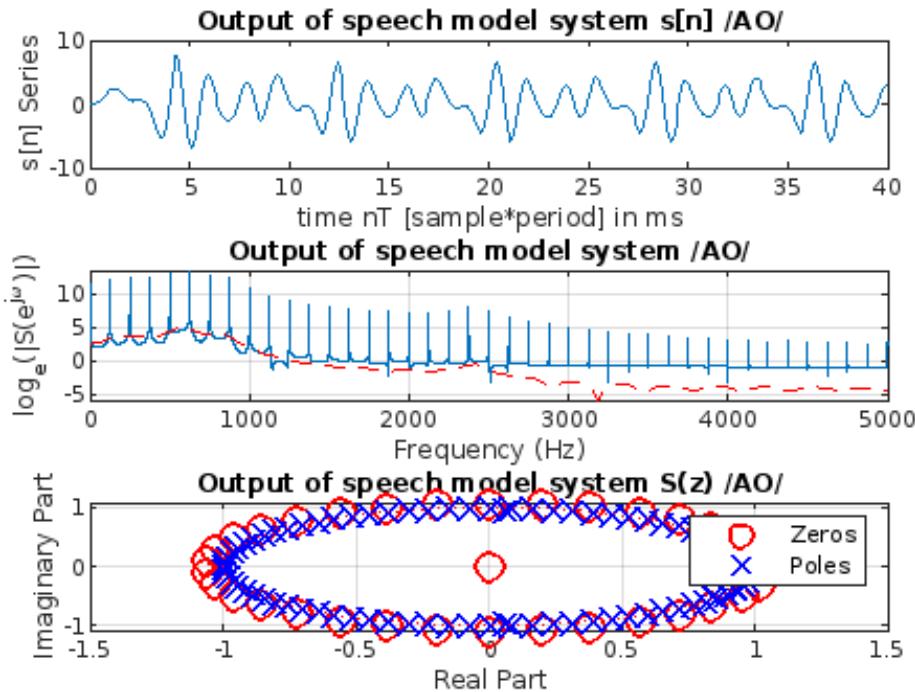
```

Και τελικά εξάγονται οι πόλοι και τα μηδενικά αυτής της συνάρητησης μεταφοράς κάνοντας χρήση του παρακάτω κώδικα :

```

% Get the poles and zeros
[polesyssao, zerosyssao] = pzmap(Sz_ao);
figure(5)
subplot(3,1,3)
% Plot zeros with a specific color, say, red 'r'
plot(real(zerosyssao), imag(zerosyssao), 'ro', 'MarkerSize', 10, 'LineWidth', 2);
hold on;
% Plot poles with a different color, say, blue 'b'
plot(real(polesyssao), imag(polesyssao), 'bx', 'MarkerSize', 10, 'LineWidth', 2);
title('Output of speech model system S(z) /AO/');
xlabel('Real Part');
ylabel('Imaginary Part');
grid on;
theta = 0:0.01:2*pi;
x = cos(theta);
y = sin(theta);
plot(x, y, 'r:');
hold off;
legend('Zeros', 'Poles');

```



Εικόνα 25

**ΜΕΡΟΣ Β | Άσκηση Β-6**

Ακούστε το σήμα  $s[n]$ , διάρκειας 0.5 sec, επιβεβαιώνοντας ότι μοιάζει με το φωνήν /AO/.

**Επίλυση:**

sound(ssnah); sound(ssneh); sound(ssnih); sound(ssniy); sound(ssnuh); sound(ssnao);

**ΜΕΡΟΣ Β | Άσκηση Β-7**

Επαναλάβετε την παραπάνω διαδικασία ώστε να συνθέσετε τα υπόλοιπα πέντε σήματα  $s[n]$ , διάρκειας 0.5 sec το καθένα, που να αντιστοιχούν στα φωνήντα /ΙΥ/, /UH/, /EH/, /AH/, /IH/, διαλέγοντας κάθε φορά τα κατάλληλα F1, F2, και F3, από τον Πίνακα 3.4 του βιβλίου. Όσον αφορά διαγράμματα, σχεδιάστε τα αντίστοιχα  $V(e^{j\omega})$  και  $S(e^{j\omega})$  για κάθε ένα από τα πέντε νέα φωνήντα, όπως και τα διαγράμματα πόλων-μηδενικών των  $V(z)$  και  $S(z)$ .

**Επίλυση:**

Η ίδια διαδικασία επαναλαμβάνεται σε αυτό το χωρίο και για τα υπόλοιπα έμφωνα το μόνο που αλλάζει είναι οι συχνότητες που διέπουν το Vz.

```

Για
/ΙΥ/
Fk=[270,2290,3010];
sigmak=[60,100,120]./2;
Vz_iy=1;
for i=1:length(Fk)

```

```

Vz_iy=Vz_iy*tf([1 0 0],[1, -2*exp(-2*pi*sigmak(i)*(1/Fs))*cos(2*pi*Fk
    (i)*(1/Fs)),exp(-4*pi*sigmak(i)*(1/Fs))],1/Fs);
endFor $\alpha$ 

/uh/
Fk=[440,1020,2240];
sigmak=[60,100,120]/.2;
Vz_uh=1;
for i=1:length(Fk)
Vz_uh=Vz_uh*tf([1 0 0],[1, -2*exp(-2*pi*sigmak(i)*(1/Fs))*cos(2*pi*Fk
    (i)*(1/Fs)),exp(-4*pi*sigmak(i)*(1/Fs))],1/Fs);
endFor $\alpha$ 

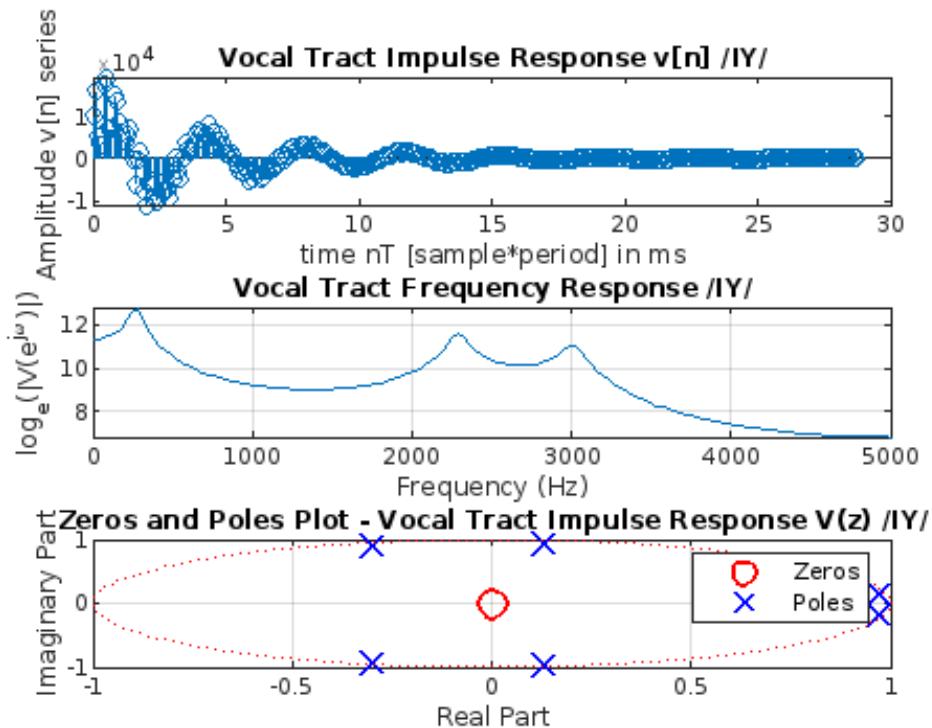
/eh/
Fk=[530,1840,2480];
sigmak=[60,100,120]/.2;
Vz_eh=1;
for i=1:length(Fk)
Vz_eh=Vz_eh*tf([1 0 0],[1, -2*exp(-2*pi*sigmak(i)*(1/Fs))*cos(2*pi*Fk
    (i)*(1/Fs)),exp(-4*pi*sigmak(i)*(1/Fs))],1/Fs);
endFor $\alpha$ 

/ah/
Fk=[520,1190,2390];
sigmak=[60,100,120]/.2;
Vz_ah=1;
for i=1:length(Fk)
Vz_ah=Vz_ah*tf([1 0 0],[1, -2*exp(-2*pi*sigmak(i)*(1/Fs))*cos(2*pi*Fk
    (i)*(1/Fs)),exp(-4*pi*sigmak(i)*(1/Fs))],1/Fs);
endFor $\alpha$ 

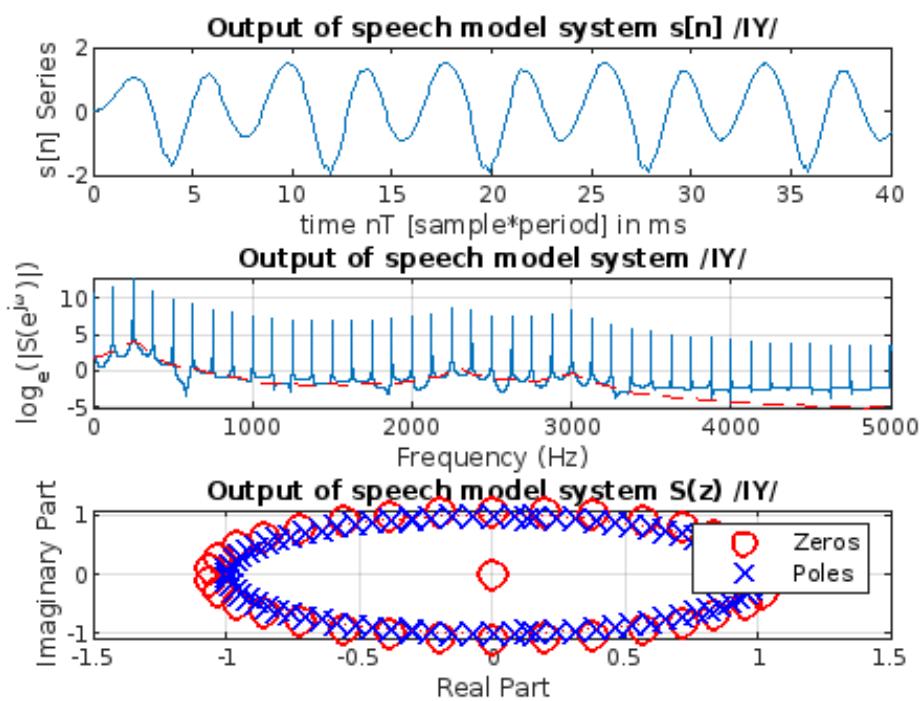
/ih/
Fk=[390,1990,2550];
sigmak=[60,100,120]/.2;
Vz_ih=1;
for i=1:length(Fk)
Vz_ih=Vz_ih*tf([1 0 0],[1, -2*exp(-2*pi*sigmak(i)*(1/Fs))*cos(2*pi*Fk
    (i)*(1/Fs)),exp(-4*pi*sigmak(i)*(1/Fs))],1/Fs);
end

```

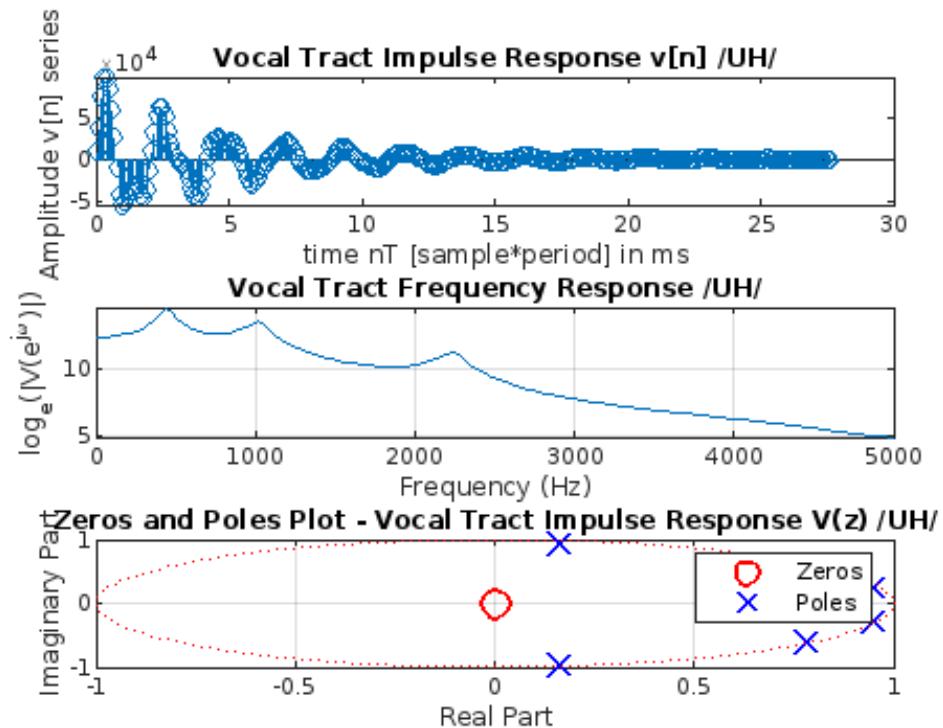
Για την κάθε μια από τις ζητούμενες περιπτώσεις του ερωτήματος ορίζουμε μια νέα συνάρτηση μεταφοράς.



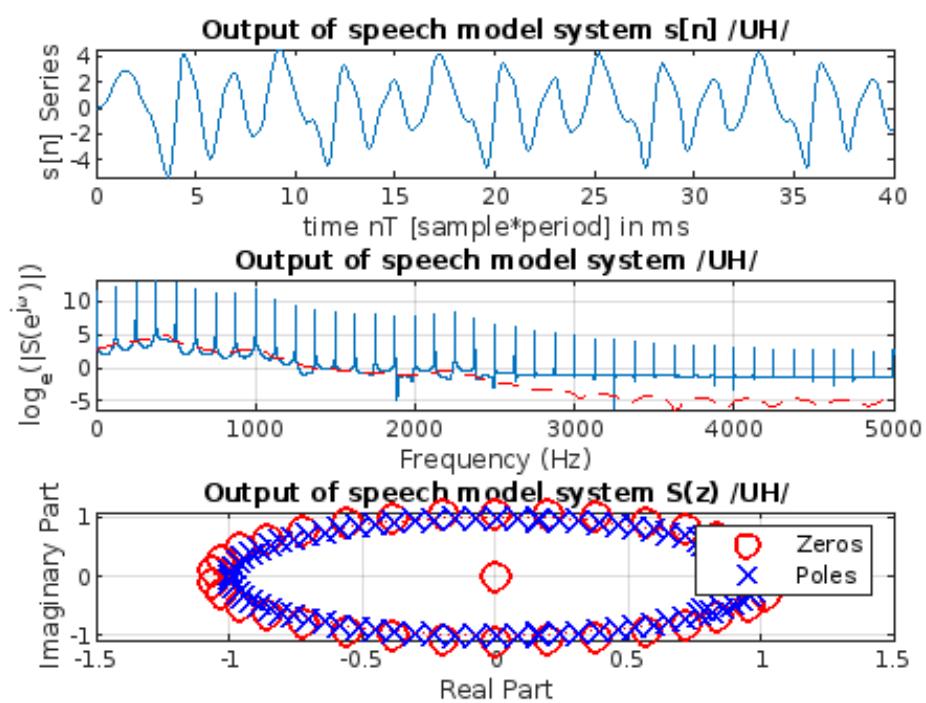
Εικόνα 26



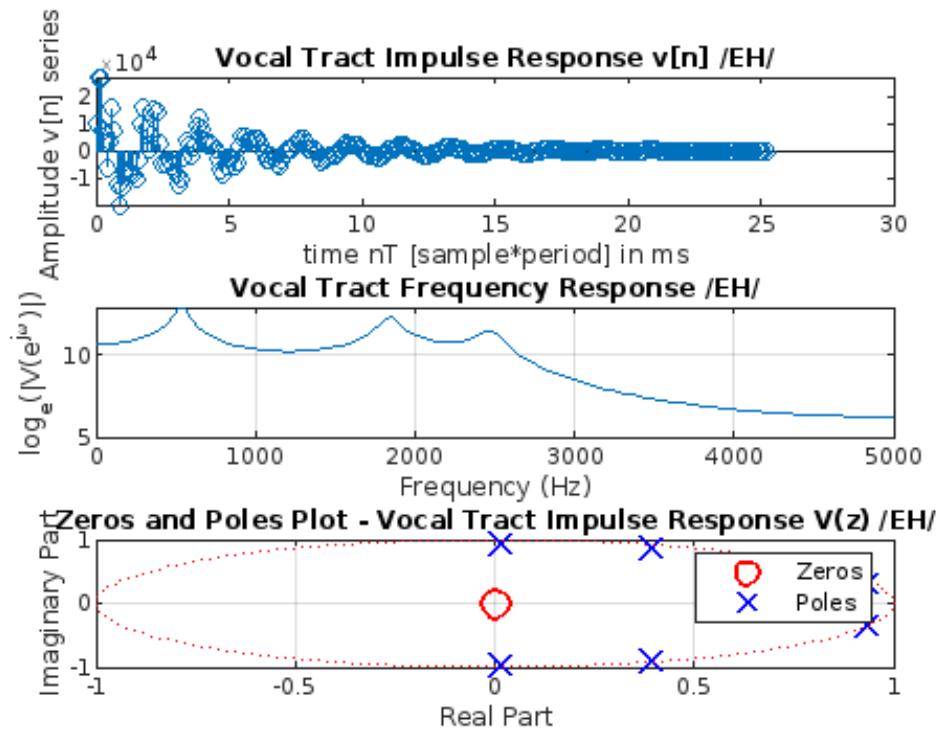
Εικόνα 27



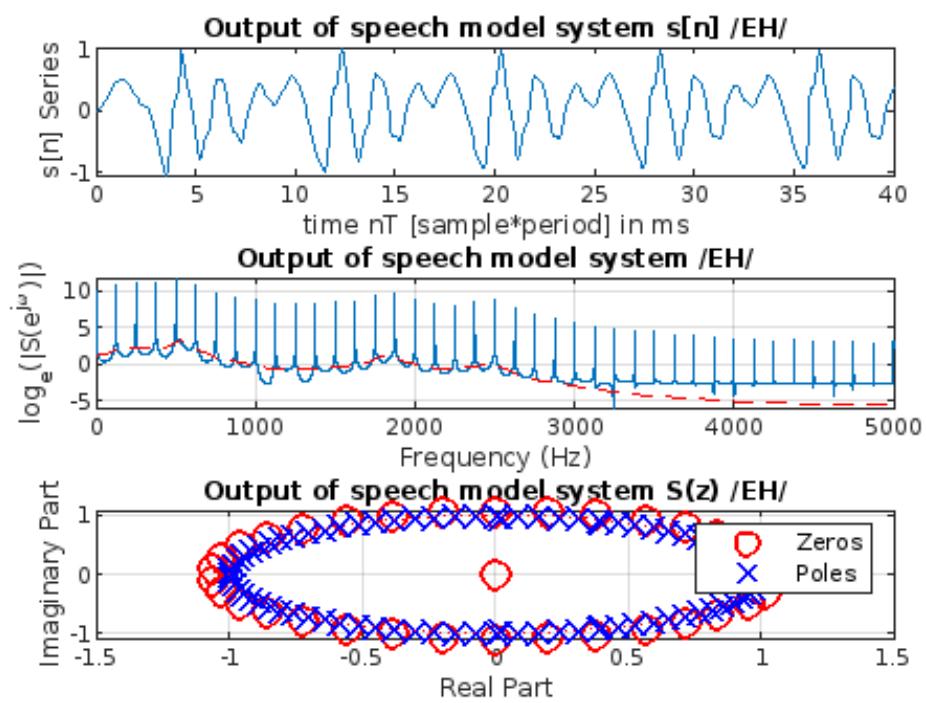
Εικόνα 28



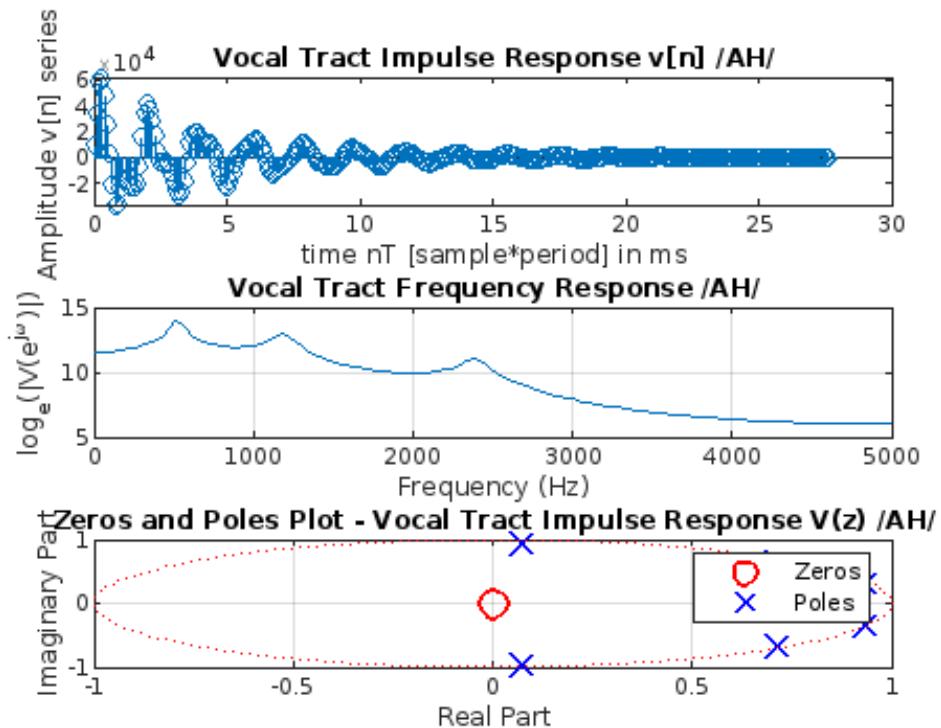
Εικόνα 29



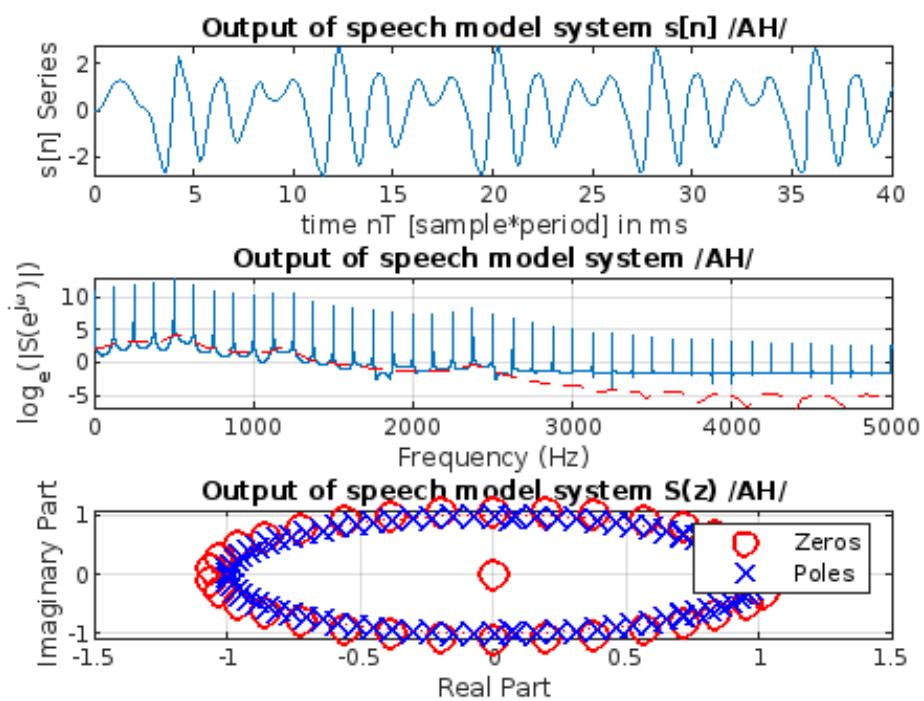
Εικόνα 30



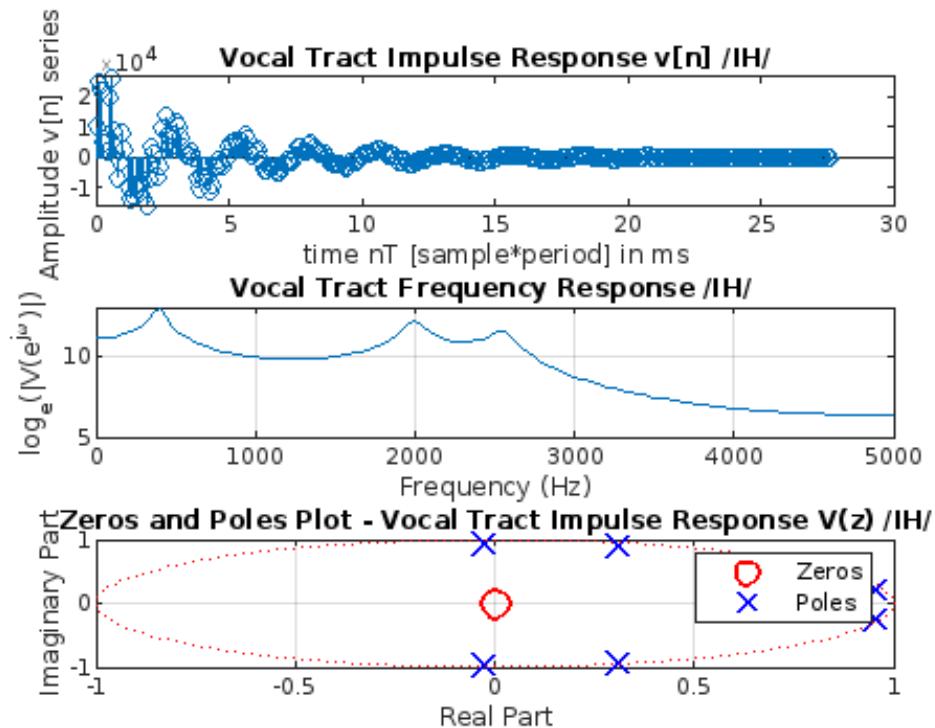
Εικόνα 31



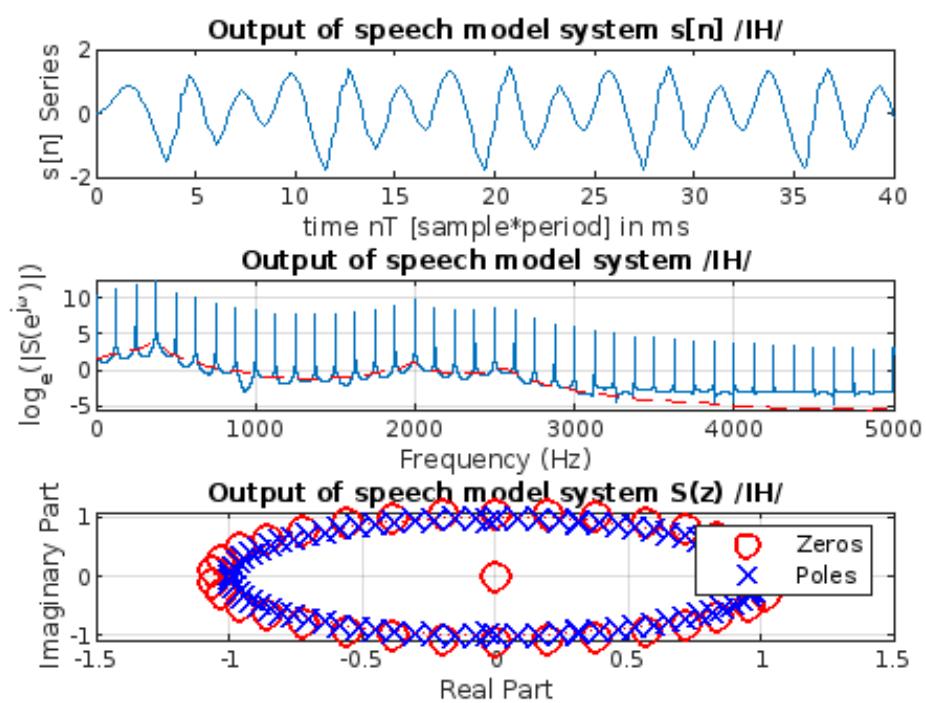
Εικόνα 32



Εικόνα 33



Εικόνα 34



Εικόνα 35

## ΜΕΡΟΣ Β | Άσκηση Β-8

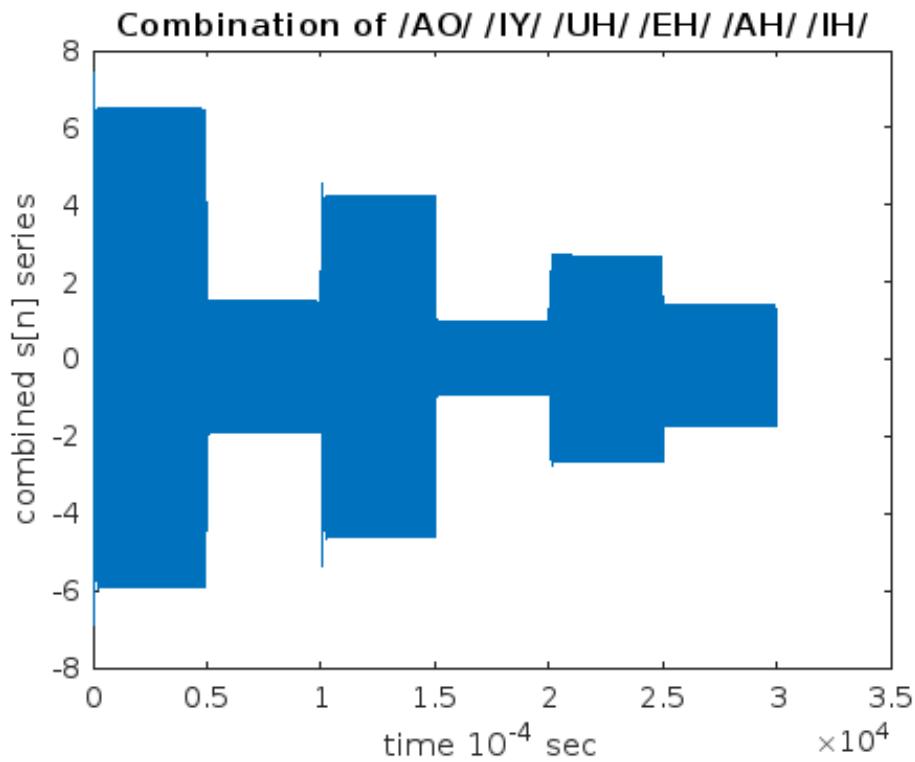
Σχεδιάστε τα συνολικά έξι  $s[n]$  που συνθέσατε (για πεπερασμένο αριθμό δειγμάτων, π.χ. 1000, το καθένα), σχολιάζοντας τη μορφή τους και πιθανές διαφορές τους.

**Επίλυση:**

Για τα ερωτήματα Β8 και Β9 κάνουμε σύνθεση των επιμέρους σημάτων κάνοντας χρήση του πάρακάτω κομματιού του αλγορίθμου. Παρατηρούμε ότι το πλάτος των σημάτων διαφέρει αρκετά.

Η απόκριση για το κάθε έμφωνο παραγόμενο σήμα (vowel) από τα ζητούμενα, παρατηρούμε ότι έχει διαφορετικό πλάτος, και αυτό το φαινόμενο μπορεί να εξαρτάται από την θεμελιώδη συχνότητα διέγερσης και από τον τόνο της φωνής. Στην προκειμένη περίπτωση ο συνδυασμός των επιμέρους συχνοτήτων που χρησιμοποιούνται για την παραγωγή του vocal tract μοντέλου αποδίδει τέτοια συνάρτηση μεταφοράς που η ιστορία του κάθε vowel έχει το δικό του χαρακτηριστικό πλάτος. Βλέπουμε εδώ τον κώδικα:

```
figure(16)
svowelsappend=[ssnao(1:find(ttnao==0.5))' , ssniy(1:find(ttniy==0.5))
' , ssnuh(1:find(ttnuh==0.5))' , ssneh(1:find(ttneh==0.5))'
, ssnah(1:find(ttnah==0.5))' , ssnih(1:find(ttnih==0.5))'];
plot(svowelsappend)
title('Combination of /AO/ /IY/ /UH/ /EH/ /AH/ /IH/');
xlabel('time 10^{-4} sec');
ylabel('combined s[n] series');
audiowrite('output.wav', svowelsappend, Fs);
sound(svowelsappend)
```



Εικόνα 36

### ΜΕΡΟΣ Β | Άσκηση Β-9

Ενώστε (append) τα έξι σήματα σε ένα σήμα διάρκειας 3 sec, ακούστε το, και αποθηκεύστε το σε ένα αρχείο ήχου, .wav, που θα εσωκλείσετε στα παραδοτέα της εργασίας.

#### Επίλυση:

Αυτό επιτυγχάνεται με τη χρήση του MATLAB, όπου τα επιμέρους σήματα συνδυάζονται σειριακά (append) για να δημιουργηθεί το τελικό σήμα. Στη συνέχεια, αυτό το συνθετικό σήμα απεικονίζεται σε ένα γράφημα, παρέχοντας οπτική αναπαράσταση της συνένωσης των φωνηέντων. Το τελικό σήμα εξάγεται ως αρχείο ήχου .wav με τη χρήση της συνάρτησης audiowrite του MATLAB και στη συνέχεια αναπαράγεται χρησιμοποιώντας την εντολή sound.

Το εξαγόμενο αρχείο ονομάζεται "output.wav" και περιλαμβάνεται στα παραδοτέα της εργασίας, παρέχοντας ένα ακουστικό δείγμα της συνθετικής αναπαράστασης των επιλεγμένων φωνηέντων.

### ΜΕΡΟΣ Β | Άσκηση Β-10

Σχεδιάστε το φασματόγραμμα (spectrogram) του σήματος φωνής που συνθέσατε (με παρόμοιες παραμέτρους με το Α-2). Ηχογραφήστε επίσης ένα αντίστοιχο φυσικό σήμα φωνής από εσάς που να περιέχει τα έξι συγκεκριμένα φωνήεντα και στη συνέχεια σχεδιάστε το φασματόγραμμά του και συγκρίνετε το με αυτό του συνθετικού σήματος.

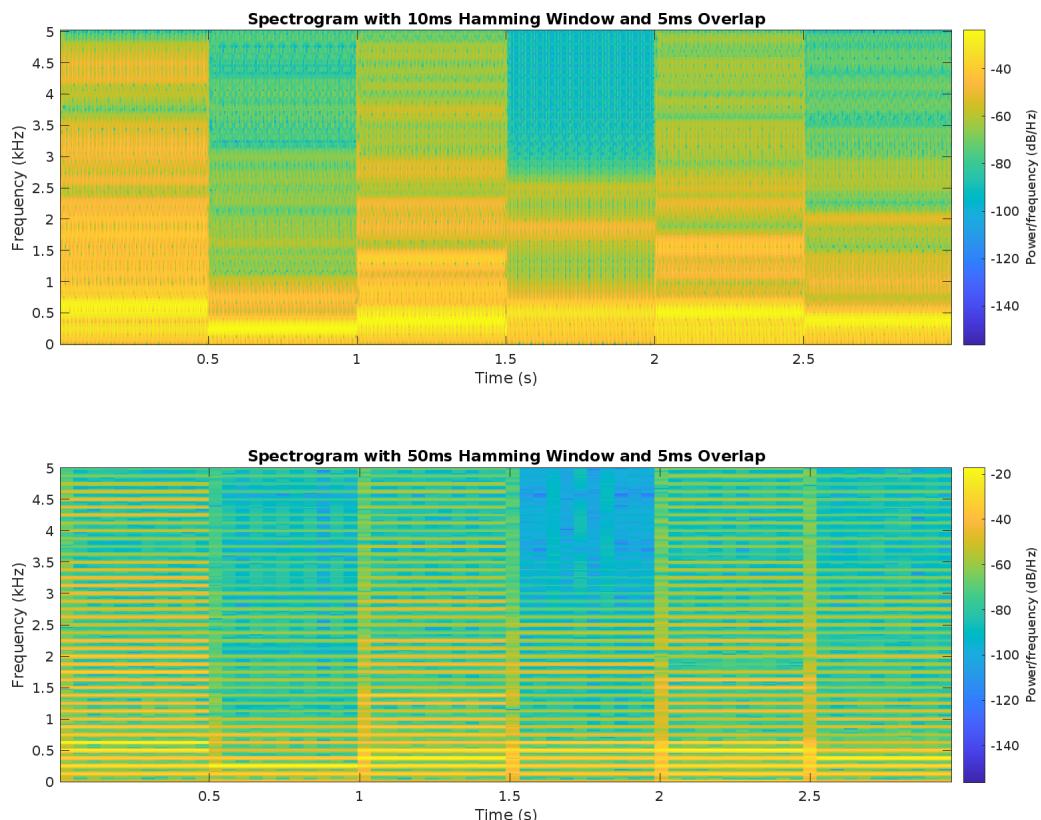
#### Επίλυση:

Τέλος, δημιουργούμε το φασματόγραμμα για όλα το συνολικό αυτό σήμα που αποτελεί σύν-

Θεση των επιμέρους έμφωνων σημάτων.

```
% B10
% Load the recorded audio file
[audioData, fs] = audioread('output.wav');
% Parameters for the spectrogram
window10ms = hamming(round(fs * 0.01), 'symmetric'); % 10 msec Hamming
% window
window100ms = hamming(round(fs * 0.05), 'symmetric'); % 100 msec
% Hamming window
overlap = round(fs * 0.005); % 5 msec overlap
% Spectrogram with 10 msec Hamming window and 5 msec overlap
figure(17);
subplot(2,1,1);
spectrogram(audioData, window10ms, overlap, [], fs, 'yaxis');
title('Spectrogram with 10ms Hamming Window and 5ms Overlap');
xlabel('Time (s)');
ylabel('Frequency (kHz)');
subplot(2,1,2);
% Spectrogram with 50 msec Hamming window and 5 msec overlap
spectrogram(audioData, window100ms, overlap, [], fs, 'yaxis');
title('Spectrogram with 50ms Hamming Window and 5ms Overlap');
xlabel('Time (s)');
ylabel('Frequency (kHz)');
```

Επίσης γίνεται ηχογράφηση αυτού του σήματος.



Εικόνα 37

**Σχετικές Βιβλιογραφικές Πηγές**

- [1] The SoX Project, Available online at <http://sox.sourceforge.net>, *SoX: Sound eXchange*, 2024.
- [2] P. Boersma and D. Weenink, “Praat: doing phonetics by computer.” [Computer program], 2024. Retrieved 6 January 2024 from <http://www.praat.org/>.
- [3] L. R. Rabiner and R. W. Schafer, *Introduction to Digital Speech Processing*. 2007.