Ψηφιακή Επεξεργασία Σημάτων – 2η Εργαστηριακή Άσκηση

Όνομα: Βλαντισλάβ
Επώνυμο: Λέντελ
ΑΜ: 03114054

Μέρος 1

Βήμα 1.0 : part1_0.m

Επειδή το σήμα είναι δικαναλικό παίρνουμε το μέσο όρο των 2 στηλών του πίνακα, αφότου το διαβάσουμε με τη συνάρτηση audioread, και το κανονικοποιούμε. Σπάμε το σήμα σε πλαίσια των 512 δειγμάτων με τη βοήθεια της συνάρτησης buffer, και στη συνέχεια εφαρμόζουμε παράθυρο Hanning σε κάθε ένα παράθυρο, αποθηκεύοντας το αποτέλεσμα σε έναν 2 διαστάσεων πίνακα, τη μεγάλη διάσταση του οποίου είχε υπολογίσει η συνάρτηση buffer. Ταυτόχρονα, υπολογίζουμε το Absolute Threshold of Hearing και αποθηκεύουμε το αποτέλεσμα στο διάνυσμα Tq.

Βήμα 1.1: part1_1.m

Για διευκόλυνση υπολογίζουμε το ζητούμενο φάσμα με τη χρήση του fft, που είναι το άθροισμα του δοσμένου τύπου. Επίσης, υπολογίζουμε στο διάνυσμα bark την αντίστοιχη κλίμακα.

Βήμα 1.2: part1_2.m

Αρχικά υπολογίζουμε τα Δ_κ για κάθε κ από 1 μέχρι 256, αποθηκεύοντας το αποτέλεσμα σε δομή τύπου cell array του Matlab, εφόσον θέλουμε να αποθηκεύσουμε σε κάθε θέση του πίνακα ένα διάνυσμα τιμών μεταβλητού μεγέθους, κάτι που δεν επιτρέπει ένας απλός πίνακας του Matlab. Ύστερα, με βάση αυτή τη δομή υλοποιούμε την λογική συνάρτηση S_T, αποθηκεύοντας τις λογικές τιμές στο διάνυσμα St. Αυτές τις υπολογίζουμε εντός δομής επανάληψης, για όλα τα κ αλλά και για όλες τις τιμές εντός των Δ_κ, ελέγχοντας έτσι αν ικανοποιείται η συνθήκη του τύπου P(k)>P(k+-1) && P(k)>P(k+-Dk)+7. Τέλος, υπολογίζουμε τις τιμές Ptm(k), αλλά και Pnm με τη συνάρτηση findNoiseMaskers, όπως υποδεικνύεται από την εκφώνηση.

Βήμα 1.3: part1_3.m

Απλά χρησιμοποιούμε την συνάρτηση checkMasters για την μείωση των μασκών.

Βήμα 1.4: part1_4.m

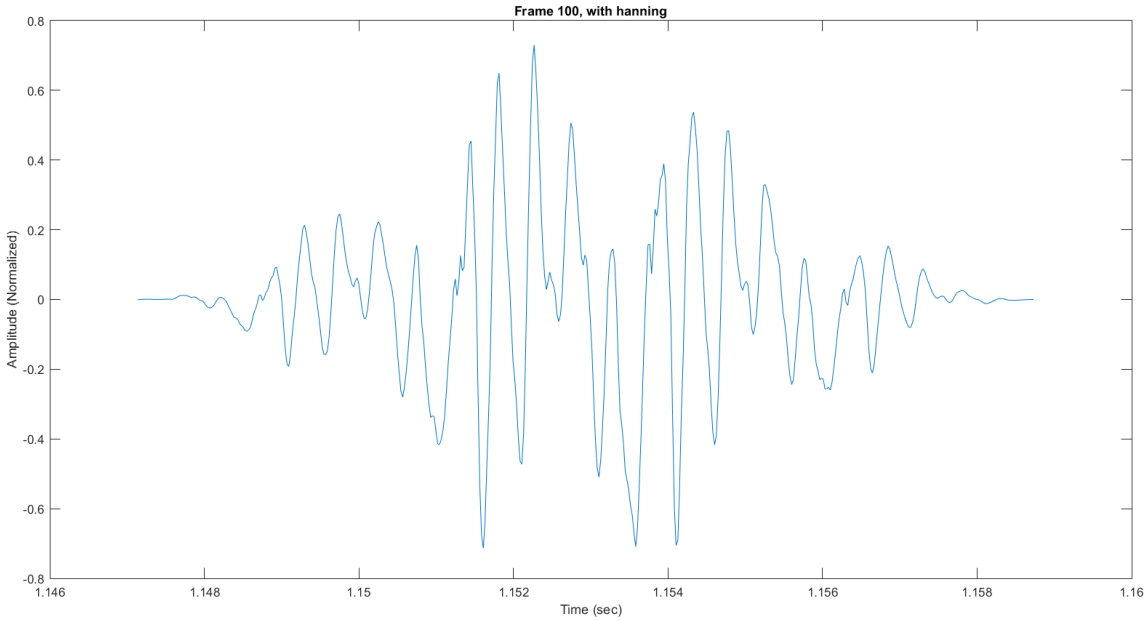
Επειδή τα j για τα οποία θα ισχύει Ptm(j)>0, ή αντίστοιχα Pnm(j)>0, θα είναι ενδεχομένως διαφορετικού πλήθους για κάθε παράθυρο τα αποθηκεύουμε σε cell array. Ύστερα, για τον υπολογισμό τους δημιουργούμε έναν πίνακα Ttm διαστάσεων 256 x (πλήθος των j δεικτών για το συγκεκριμένο παράθυρο) και ελέγχουμε εντός ποιων ορίων είναι το Δb. Αν δεν ανήκει σε κάποιο από τα διαστήματα προχωράμε στο επόμενο j, αφήνοντας την θέση κ του Ttm πίνακα (ή Tnm αντίστοιχα) μηδενική. Διαφορετικά, υπολογίζουμε κατάλληλα το SF ώστε να το προσθέσουμε κατάλληλα στο τελικό πίνακα.

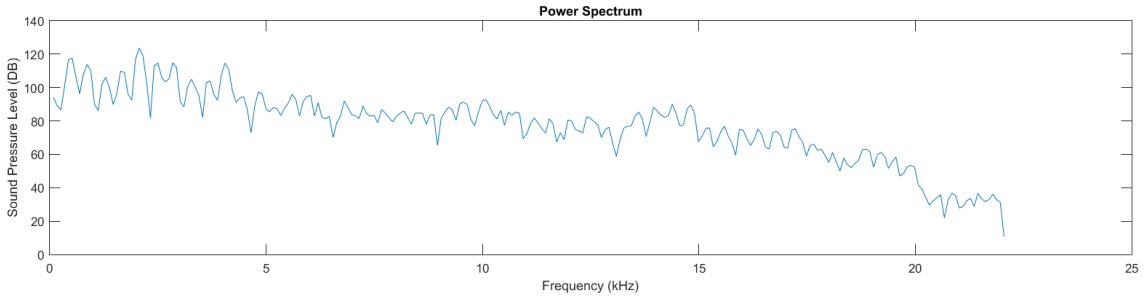
Βήμα 1.5: part1_5.m

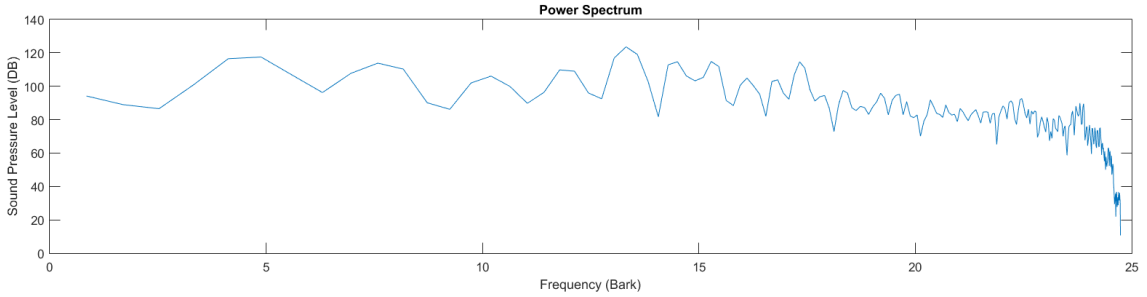
Για να υπολογίσουμε τα αθροίσματα για το συγκεκριμένο παράθυρο λαμβάνουμε τα διανύσματα μεταβλητού μήκους από τα cell arrays Ttm και Tnm , και αθροίζουμε κατάλληλα πάνω σε αυτά.

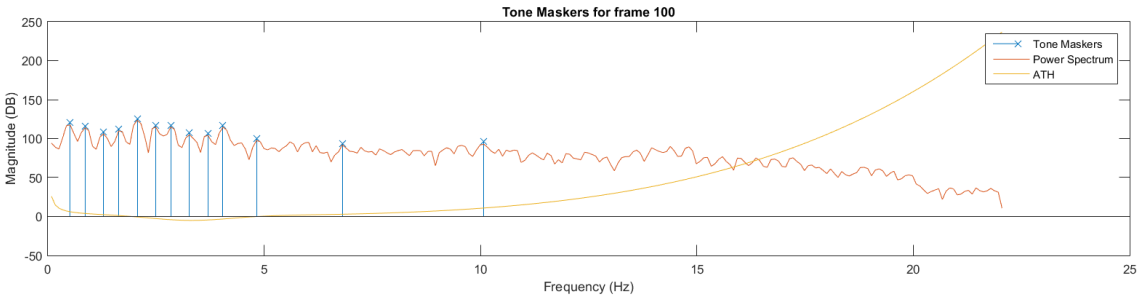
Απεικόνιση αποτελεσμάτων: plot_part1.m

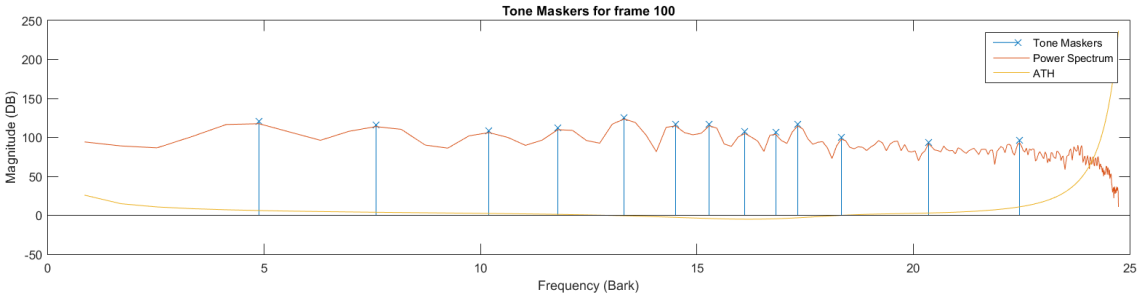
Τα διαγράμματα αναφέρονται στο πλαίσιο με νούμερο 100, και είναι:

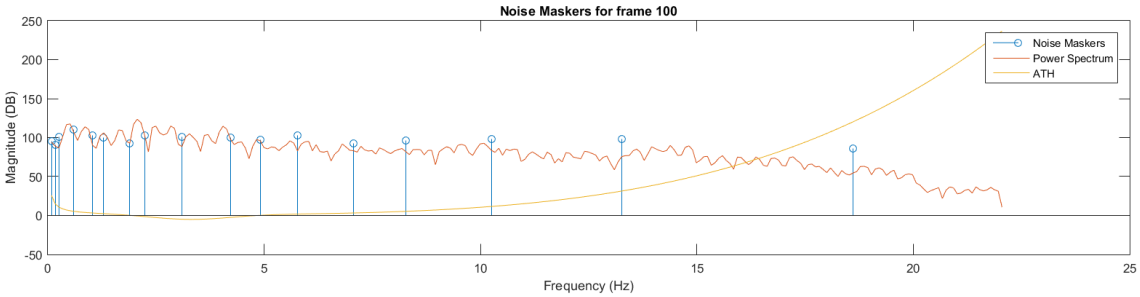


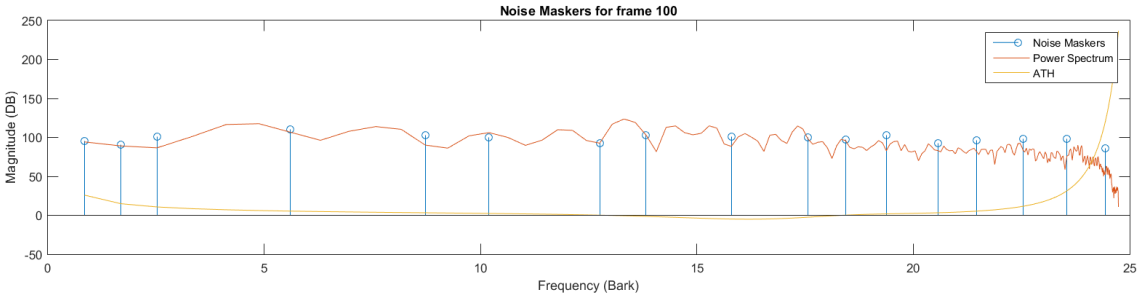


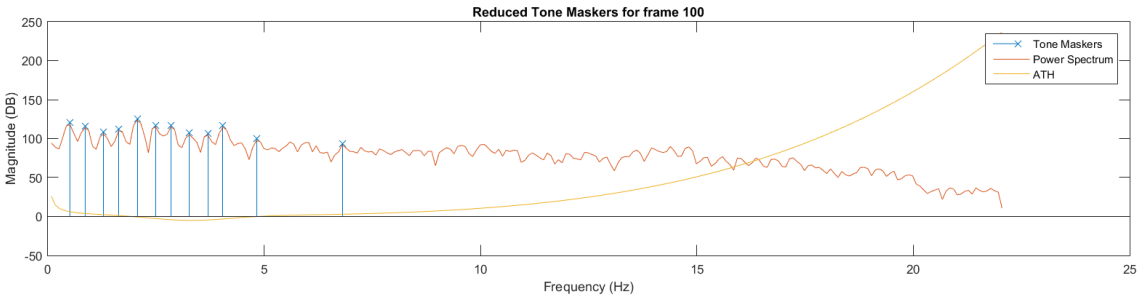


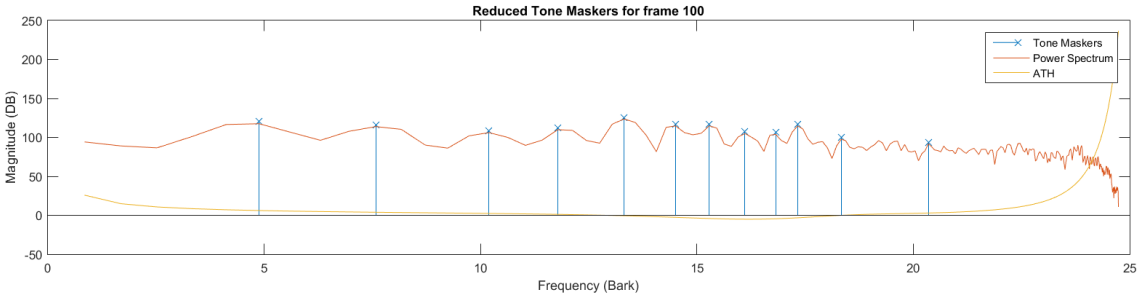

Tone Maskers for frame 100
Tone Maskers
Power Spectrum
ATH
Magnitude (DB)
Frequency (Hz)

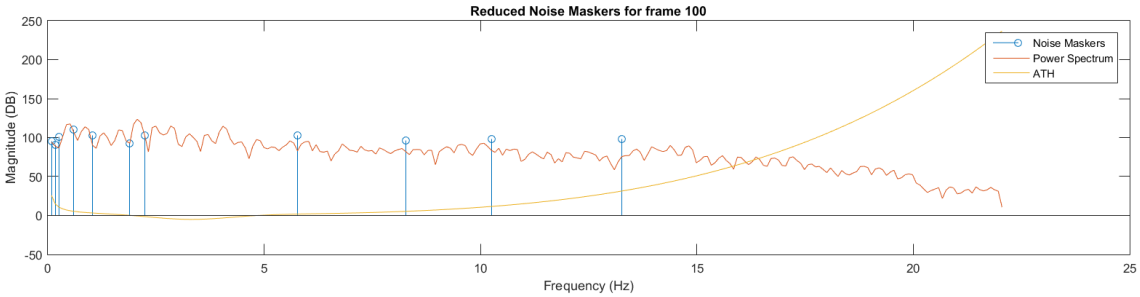

Tone Maskers for frame 100
Tone Maskers
Power Spectrum
ATH
Magnitude (DB)
Frequency (Bark)

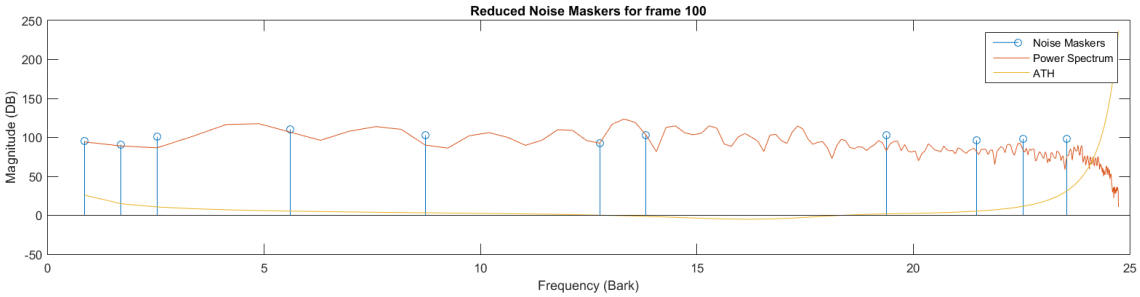

Noise Maskers for frame 100
Noise Maskers
Power Spectrum
ATH
Magnitude (DB)
Frequency (Hz)

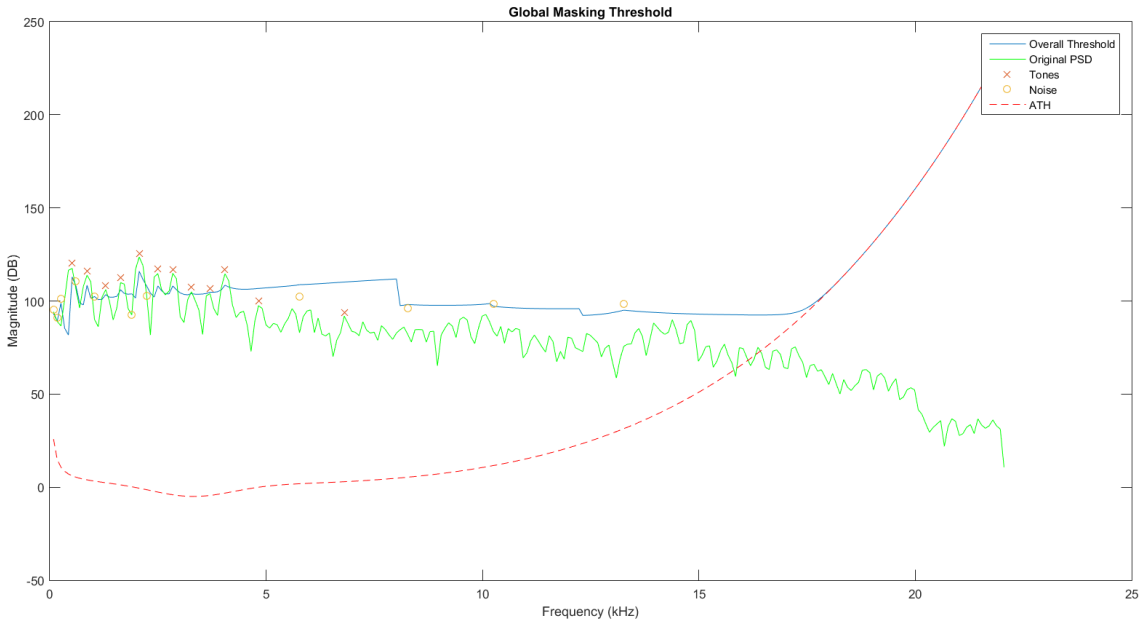

Noise Maskers for frame 100
Noise Maskers
Power Spectrum
ATH
Magnitude (DB)
Frequency (Bark)

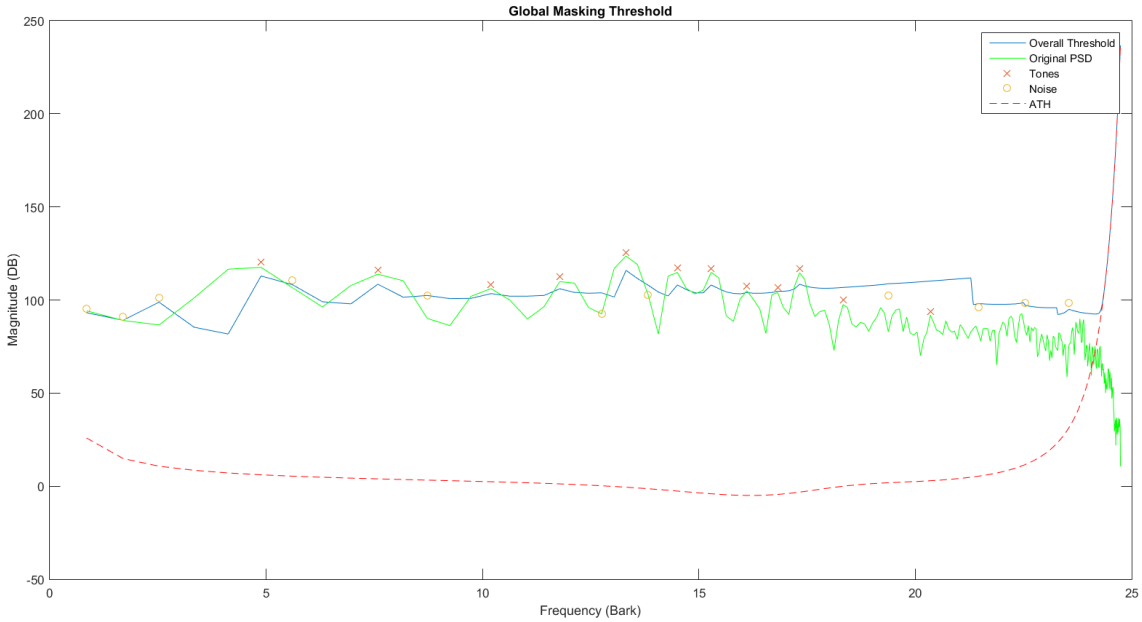

Reduced Tone Maskers for frame 100
Tone Maskers
Power Spectrum
ATH
Magnitude (DB)
Frequency (Hz)


Reduced Tone Maskers for frame 100
Tone Maskers
Power Spectrum
ATH
Magnitude (DB)
Frequency (Bark)


Reduced Noise Maskers for frame 100
Noise Maskers
Power Spectrum
ATH
Magnitude (DB)
Frequency (Hz)


Reduced Noise Maskers for frame 100
Noise Maskers
Power Spectrum
ATH
Magnitude (DB)
Frequency (Bark)





Όλα τα διαγράμματα για το Μέρος 1 της άσκησης βρίσκονται στο φάκελο plots_part1 του αρχείου .zip που υποβλήθηκε, και υπολογίζονται εκ νέου με την εκτέλεση του κώδικα plot_part1.m , ύστερα από την δημιουργία των κατάλληλων πινάκων των προηγούμενων ερωτημάτων.

Μέρος 2

Βήμα 2.0: part2_0.m

Τα παράθυρα του σήματος για αυτό το μέρος φροντίζουμε να είναι εκείνα στα οποία δεν έχει εφαρμοστεί παράθυρο Hanning, διαφορετικά έχουμε αλλοίωση του αποτελέσματος. Επίσης, λαμβάνουμε υπόψιν ότι η αρίθμηση στο Matlab αρχίζει από το 1, οπότε για να υπολογίσουμε σωστά τα ζητούμενα φίλτρα εκτελούμε επανάληψη για n και k από 0 εώς L-1 και 0 εώς M-1 αντίστοιχα, υπολογίζοντας την τιμή ακριβώς με βάση τον τύπο, αλλά αποθηκεύοντας τις τιμές στις θέσεις n+1, κ+1 για το φίλτρο h, και στις θέσεις L-n, κ+1 για το φίλτρο g, για την διευκόλυνσή μας.

Βήμα 2.1: part2_1.m

Εφαρμόζουμε συνέλιξη των παραθύρων με τα φίλτρα, και καταλήγουμε σε 3 διαστάσεων πίνακα, στον οποίο η $1^η$ διάσταση αναφέρεται στα πλαίσια, η $2^η$ σε ένα από τα 32 φίλτρα και η $3^η$ στα δείγματα που προέκυψαν. Ύστερα, κάνουμε υποδειγματοληψία με τη συνάρτηση downsample.

Βήμα 2.2: part2_2.m

Εδώ για να υπολογίσουμε τα επίπεδα κβάντισης έχουμε: R=2^16 διότι δίνεται από την εκφώνηση ότι το αρχικό σήμα έχει κωδικοποιηθεί με 16 bits, επομένως τα δικά του επίπεδα θα είναι 2^16, προκειμένου να κάνουμε τους υπόλοιπους υπολογισμούς. Τα διαστήματα των Tg για το κάθε παράθυρο ανάλυσης, αλλά για το κάθε φίλτρο κ, το υπολογίζουμε κατευθείαν, λαμβάνοντας υπόψιν τις κεντρικές συχνότητες του κάθε φίλτρου, ως: [8*(κ-1)+1,8*κ]. Χωρίσαμε, δηλαδή, το διάστημα των διακριτών συχνοτήτων [1:256], στα οποία ορίζεται το Tg, σε 32 ισομήκη υποδιαστήματα, όσα είναι και τα φίλτρα, και εύρους 256/M=256/32=8. Σε καθένα από αυτά παίρνουμε το ελάχιστο Tg για να υπολογίσουμε το πλήθος των απαιτούμενων bit Bκ για την κωδικοποίηση. Για την κβαντοποίηση παίρνουμε το πλήθος σε Δ του σήματος, ξεκινώντας την αρίθμηση των επιπέδων από την ελάχιστη τιμή του σήματος. Ο τύπος που χρησιμοποιήσαμε αναφέρεται στον υπολογισμό επιπέδων ενός ομοιόμορφου κβαντιστή. Θεωρούμε πως στέλνουμε κωδικοποιημένη ακολουθία 2^Bκ επιπέδων, οπότε αποθηκεύουμε στο αποτέλεσμα ακεραίους. Για το κάθε φίλτρο του κάθε παραθύρου στέλνουμε και το βήμα Δ, καθώς και το πρώτο επίπεδο, προκειμένου να γίνει αποκωδικοποίηση.

Βήμα 2.3: part2_3.m

Αποκωδικοποιούμε το σήμα με βάση τις παραμέτρους που έχουμε αποθηκεύσει σε κατάλληλο πίνακα 2 διαστάσεων, δηλαδή τιμή για κάθε παράθυρο και για κάθε φίλτρο. Στη συνέχεια, αφού κάνουμε υπερδειγματοληψία, εφαρμόζουμε τα φίλτρα σύνθεσης σε κάθε κομμάτι k της λαμβανόμενης ακολουθίας και προσθέτουμε τα αποτελέσματα. Για την ανακατασκευή, δεδομένου ότι η χρήση των φίλτρων έχει ως αποτέλεσμα παραπανήσια δείγματα, εφαρμόζουμε overlap add με βήμα 512 δείγματα (όσα στο αρχικό παράθυρο) και μήκος, όμως, ίσο με 639, δηλαδή όσα δείγματα έχει το νέο παράθυρο. Εν τέλει, γράφουμε στο αρχείο με όνομα PerceptualAudio.wav το αποτέλεσμα, με τη βοήθεια της συνάρτησης audiowrite.

Κβαντοποίηση με μη προσαρμοζόμενο κβαντιστή: part2_NonPerceptualAudioCoding.m

Σε αυτό το κομμάτι κώδικα εφαρμόζουμε την ίδια διαδικασία με τα προηγούμενα βήματα, έχοντας όμως σταθερό αριθμό σε bits B=8, και βήμα Δ=2/2^8, καθώς οι τιμές θεωρούμε ότι κυμαίνονται από το -1 έως το +1. Γράφουμε το αποτέλεσμα στο αρχείο με όνομα NonPerceptualAudio.wav .

Τα 2 αρχεία ήχου βρίσκονται στον φάκελο wavFiles του zip που υποβάλλαμε.

part2_calc.m , part2_calc_method2.m:

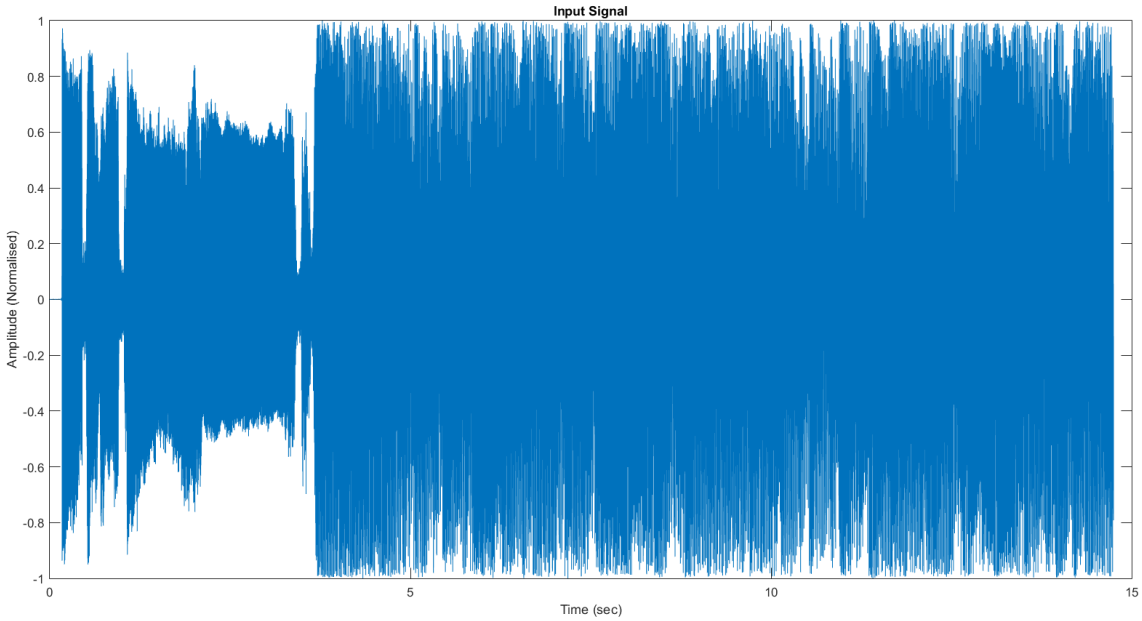
Σε αυτά τα κομμάτια κώδικα υπολογίζουμε τα ποσοστά συμπίεσης ως προς το αρχικό σήμα και απεικονίζουμε το αρχικό σήμα, το παραγόμενο αλλά και το σήμα λάθους. Για το ποσοστό συμπίεσης υπολογίζουμε το πλήθος των bit που μεταδίδουμε συνολικά, δηλαδή του συμπιεσμένου σήματος, εντός βρόχου επανάληψης για την καθεμία από τις 32 ακολουθίες του κάθε πλαισίου που μεταδίδουμε. Στον αθροιστή προσθέτουμε κάθε φορά την ποσότητα length(yQ(frame,k))*B(frame,k), δηλαδή το πλήθος των bits ανά μετάδοση, αλλά και τα 16bit που αντιστοιχούν στο βήμα και το πρώτο επίπεδο που μεταδίδουμε ταυτόχρονα. Αντίστοιχη διαδικασία εφαρμόζουμε και για την δεύτερη μέθοδο συμπίεσης, διατηρώντας σταθερό τον αριθμό B. Επειδή το αρχικό σήμα έχει κωδικοποιηθεί με 16bits, υπολογίζουμε το πλήθος ασυμπίεστων bits με τον τύπο length(signal)*16. Το ποσοστό το

βρίσκουμε με την διαίρεση του πρώτου με το δεύτερο μέγεθος. Ύστερα, λαμβάνουμε υπόψιν την καθυστέρηση που εισάγεται στο τελικό σήμα λόγω των φίλτρων, την οποία και βρίσκουμε με την συνάρτηση του Matlab immse, και ύστερα την αφαιρούμε. Τέλος, υπολογίζουμε το μέσο τετραγωνικό σφάλμα και αναπαριστούμε το σήμα λάθους.

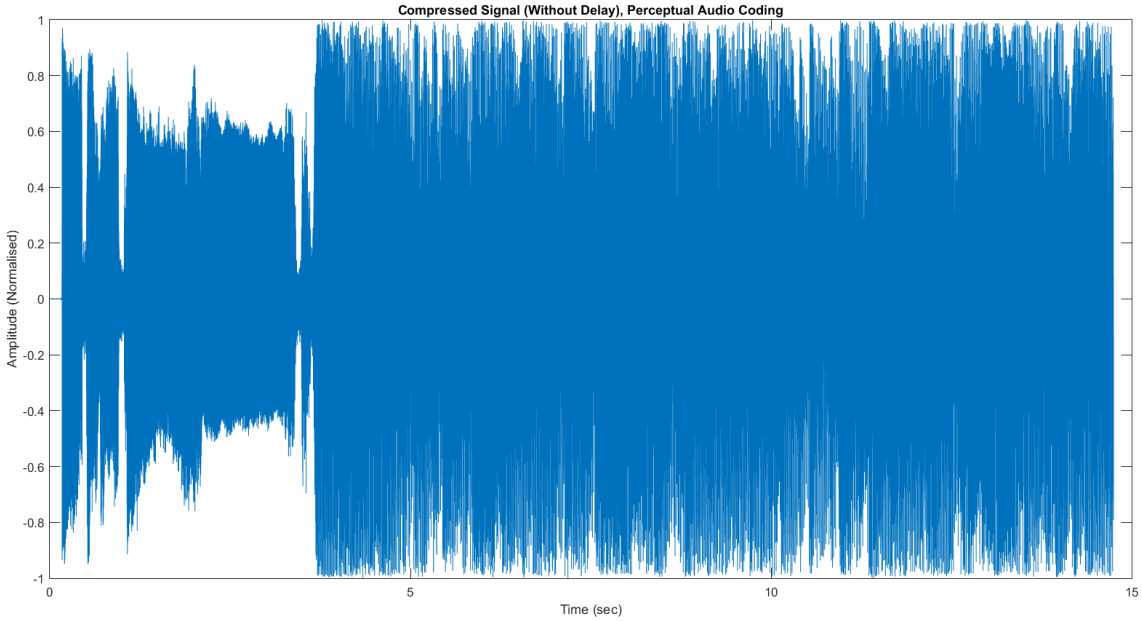
Από την πρώτη μέθοδο κωδικοποίησης λαμβάνουμε συμπίεση με ποσοστό ~132% και από τη δεύτερη ~160%. Παρατηρούμε ότι με τη $2^{η}$ μέθοδο επιτυγχάνουμε μεγαλύτερη εξοικονόμηση σε bits, ωστόσο ακούγοντας τα 2 σήματα παρατηρούμε ότι η πρώτη μέθοδος δίνει καλύτερο αποτέλεσμα. Το μέσο τετραγωνικό σφάλμα που προέκυψε είναι ~0,0049 και για τις 2 μεθόδους, χωρίς να σημαίνει αυτό ότι πρόκειται για 2 ισοδύναμες μεθόδους, διότι, όπως αναφέραμε, το αποτέλεσμα από την κωδικοποίηση βάσει ψυχοακουστικού μοντέλου έχει καλύτερο άκουσμα για τον άνθρωπο. Παρόλα αυτά, η κωδικοποίηση με μη προσαρμοζόμενο κβαντιστή έχει ένα, επίσης, ικανοποιητικό αποτέλεσμα, στο οποίο, όμως, παρατηρήσαμε στην αρχή του ήχου λίγο θόρυβο. Σε γενικές γραμμές, η διαφορά του με το προηγούμενο δεν είναι πολύ μεγάλη.

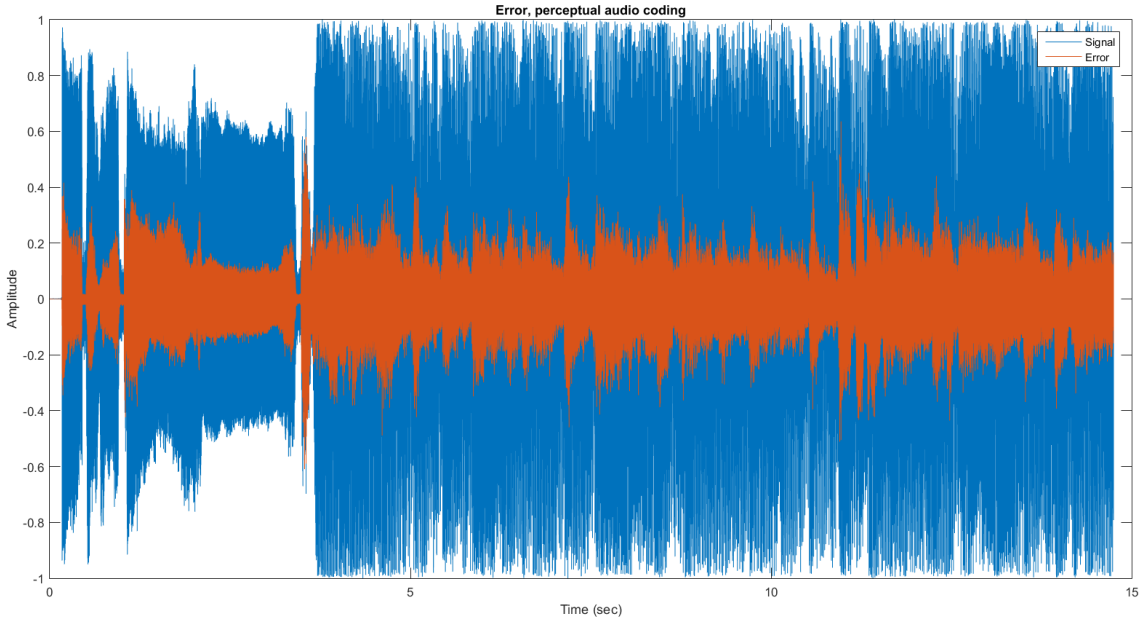
Τα διαγράμματα που προέκυψαν για το Μέρος 2 της άσκησης είναι:

Το αρχικό σήμα εισόδου:



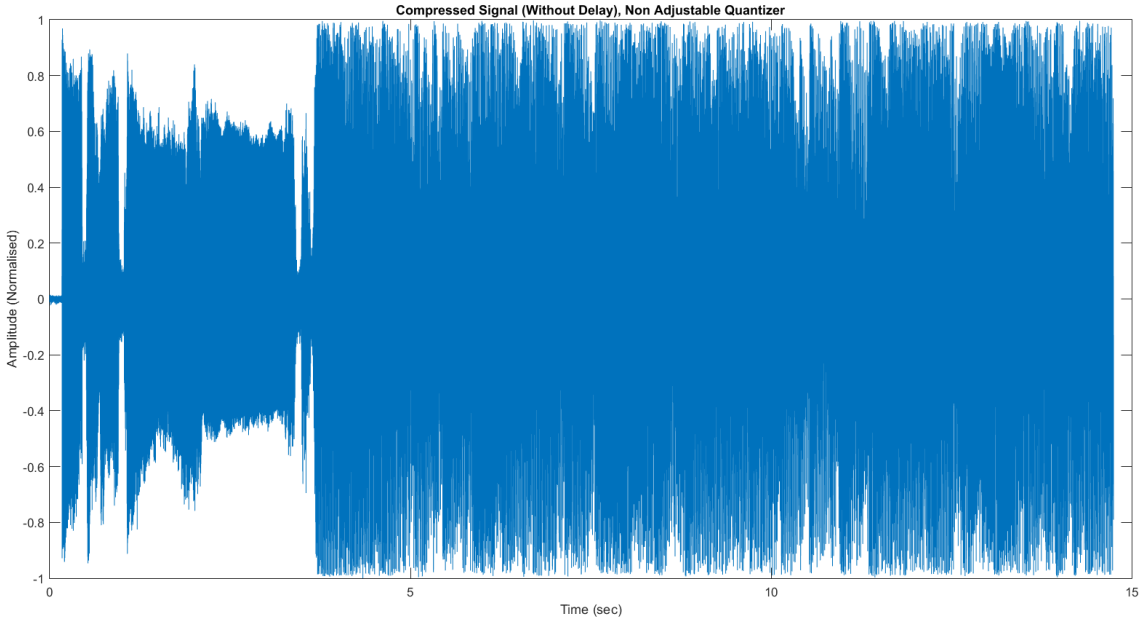
Με προσαρμοζόμενο κβαντιστή, βάσει ψυχοακουστικού μοντέλου:

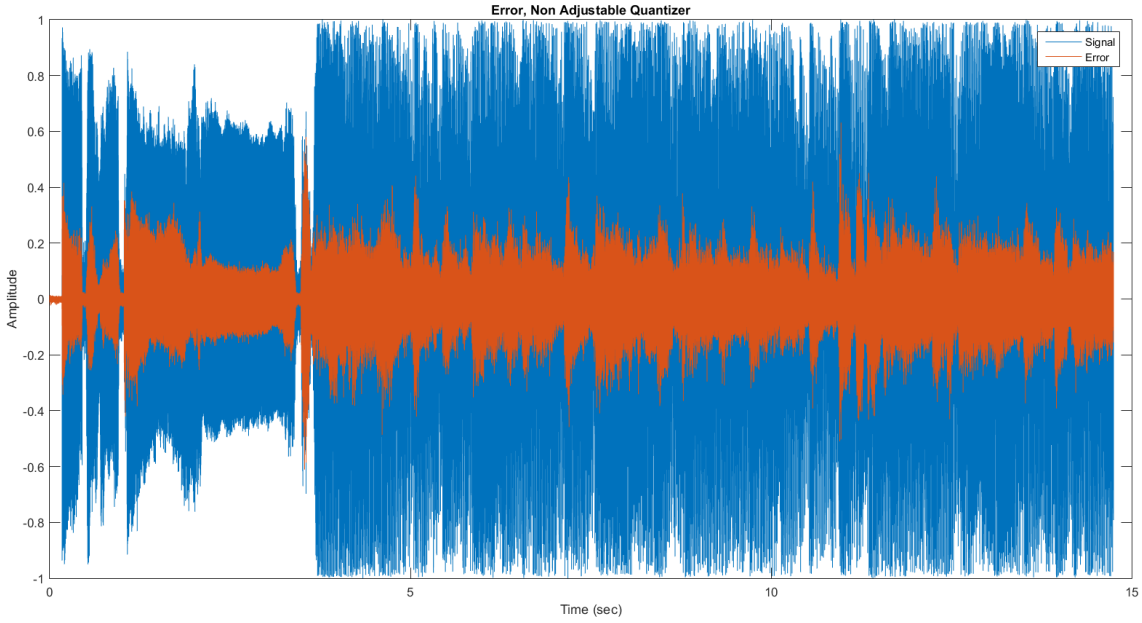




(Αρχικό σήμα και σήμα λάθους)

Χωρίς προσαρμοζόμενο κβαντιστή:





(Αρχικό σήμα και σήμα λάθους)

Όλα τα διαγράμματα για το Μέρος 2 της άσκησης βρίσκονται στον φάκελο plots_part2 του zip που υποβλήθηκε.

Παρατηρούμε και από τις γραφικές παραστάσεις ότι ύστερα από την κωδικοποίηση το σήμα απέχει ελάχιστα από το πραγματικό, αφού και το ίδιο το σήμα λάθους δείχνει να ακολουθεί τη μορφή του αρχικού, χωρίς να παρουσιάζει μεγάλες τιμές σχετικά με αυτό. Καταλήγουμε στο συμπέρασμα ότι με το perceptual audio coding μπορούμε ταυτόχρονα να εξοικονομήσουμε μνήμη αλλά και να λάβουμε ήχο που δύσκολα διακρίνεται από το αρχικό. Επίσης, με τη δεύτερη και απλούστερη μέθοδο εξοικονομούμε ακόμα περισσότερη μνήμη, με αποτέλεσμα, όμως, ελαφρώς χειρότερο από αυτό της πρώτης μεθόδου.