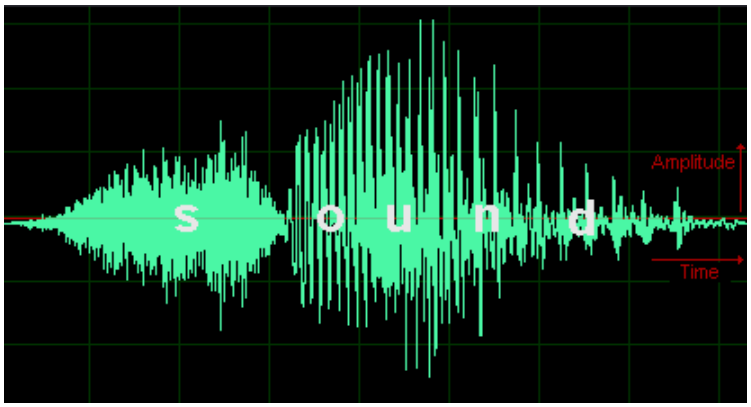
Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Μάθημα: Ψηφιακή Επεξεργασία Σημάτων

2η Εργαστηριακή Άσκηση

Θέμα: Γραμμική Πρόβλεψη (LPC) και Ομοιομορφική (Cepstrum) Επεξεργασία με Matlab και Εφαρμογές στη Σύνθεση και Συμπίεση Φωνής



Ονοματεπώνυμο: Βαβουλιώτης Γεώργιος ΑΜ: **03112083**

Ονοματεπώνυμο: Αθανασίου Νικόλαος ΑΜ: **03112074**

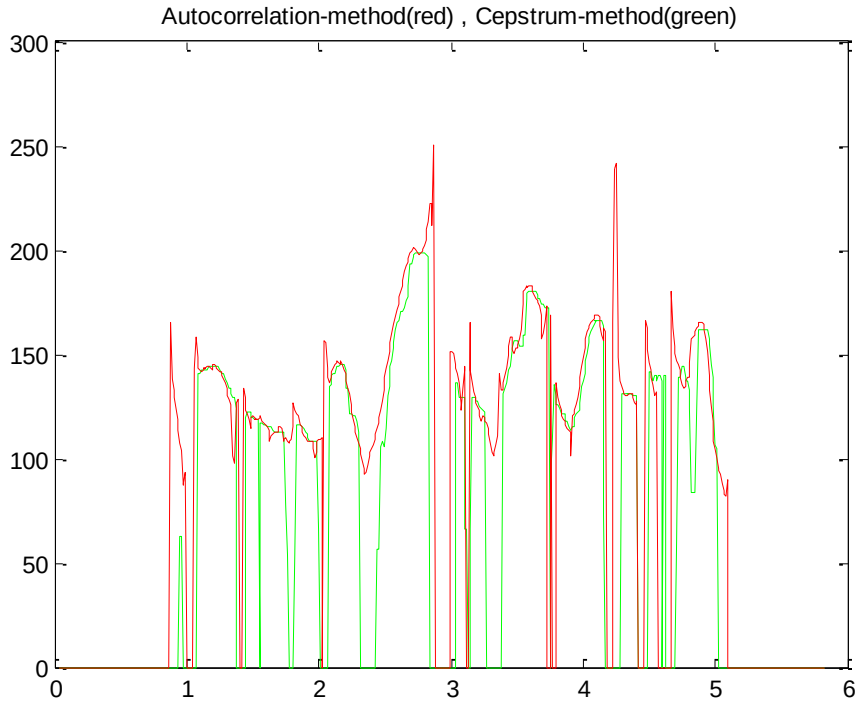
Ημερομηνία Παράδοσης: 21/5/2015

ΜΕΡΟΣ 1^{Ο}: Εξαγωγή pitch φωνής με χρήση Cepstrum

Στόχος της άσκησης αυτής είναι η εξαγωγή της θεμελιωδούς συχνότητας της φωνής που υπάρχει στο αρχείο **speech.wav** το οποίο μας δίνεται, με χρήση ομομορφικής επεξεργασίας σημάτων. Σ' αυτό μας βοήθησε ιδιαίτερα το βιβλίο των Raber & Schafer και συγκεκριμένα η ενότητα 7.3 αυτού.
Μας δίνεται ένα αρχείο επίσης με όνομα **pitch.mat** το οποίο περιέχει την ζητούμενη θεμελιώδη συχνότητα, η οποία όμως εχει εξαχθεί με κάποιο αλγόριθμο που βασίζεται στην αυτοσυσχέτιση.
Η διαδικασία την οποία ακολουθήσαμε για να παράξουμε το τελικό αποτέλεσμα αναλύεται στα παρακάτω επιμέρους στάδια :

- Αρχικά με χρήση της συνάρτησης ***wavread()*** του Matlab εισάγουμε το σήμα φωνής που μας ενδιαφέρει να επεξεργαστούμε (speech.wav) και παρατηρούμε ότι εχει διάρκεια περίπου 5.83 sec και είναι δειγματοληπτημένο στα 48kHz.. Η εντολή που το έκανε αυτό είναι η `[x, fs] = wavread ('speech.wav') ;`.
- Στη συνέχεια με χρήση της συνάρτησης ***buffer()*** δημιουργώ κατάλληλα επικαλυπτόμενα πλαίσια του σήματος εισόδου για να μπορέσω να το επεξεργαστώ. Επίσης με χρήση της συνάρτησης ***hamming()*** δημιουργώ ένα hamming window για την παραθύρωση των επιμέρους πλαισίων.
- Τρέχοντας ένα επαναληπτικό βρόχο (πλήθος επαναλήψεων = 580) παραθυρώνω κάθε πλαίσιο και δημιουργώ το cepstrum του με χρήση της συνάρτησης rceps(). Ωστόσο επειδή το αρχείο αυτό περιέχει φωνή αντρική (50 ως 250 Hz) θα πρέπει να κόψω μερικά δείγματα για να πάρω σωστό αποτέλεσμα στο τέλος. Έτσι κρατάω το cepstrum ανάμεσα στα δυο όρια τα οποία είναι υπολογισμένα στο source αρχείο με όνομα **exercise1** και κρατάω για κάθε πλαίσιο σε ένα πίνακα την μέγιστη κορυφή(πλάτος) καθώς και τον δείκτη στον οποίο εμφανίζεται το μέγιστο αυτό.
- Μόλις ολοκληρωθέι ο βρόχος αυτός είμαι σε θέση πλέον να δημιουργήσω τον ζητούμενο πίνακα συχνοτήτων. Τρέχω ένα επαναληπτικό βρόχο 580 επαναλήψεων και πάλι, και ανάλογα με την τιμή του μεγίστου που υπάρχει στην αντίστοιχη θέση του πίνακα μου συμπεραίνω αν ο αντίστοιχος ήχος είναι έμφωνος ή άφωνος. Αν είναι άφωνος βάζω στον πίνακα συχνοτήτων τον αριθμό 0 ενώ αν είναι έμφωνος διαιρώ την συχνότητα δειγματοληψίας με τον αντίστοιχο δείκτη προσαυξημένο κατά 191 όπου 191 είναι τα δείγματα που έκοψα. Αν δεν λάμβανα υπόψην μου τα δείγματα αυτά το απότέλεσμα μου θα ήταν πολύ μακριά από το σωστό.
- Αφού δημιούργησα τον πίνακα των συχνοτήτων είμαι σε θέση να αναπαραστήσω σε κοινό διάγραμμα τις δυο χρονοσειρές συχνοτήτων για να

εξάγω συμπέρασμα σχετικά με την ποότητα του αποτελέσματος της μεθόδου που υλοποίησα. Για να γίνει αυτό φορτώνω το αρχείο με όνομα **pitch.mat** με χρήση της συνάρτησης ***load()*** του Matlab το οποίο περιέχει τον πίνακα συχνοτήτων όπως αυτός προέκυψε από την μέθοδο autocorrelation. Το κοινό γράφημα των δυο χρονοσειρών φαίνεται παρακάτω (το πράσινο γράφημα αντιστοιχεί στην μέθοδο που ανέπτυξα παραπάνω) :



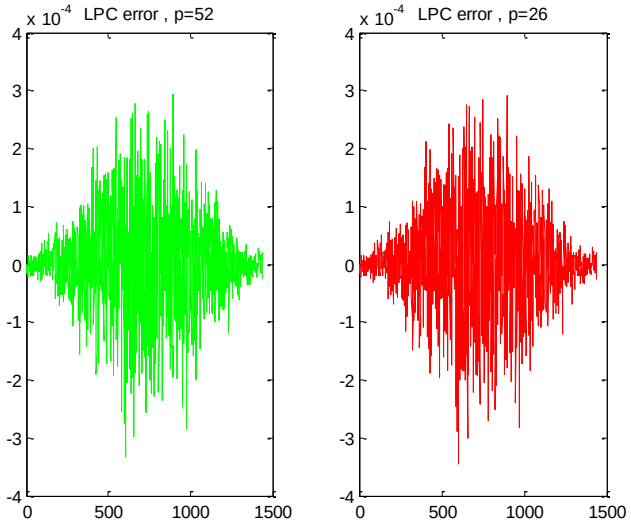
<u>Σχόλιο</u> : Από το παραπάνω γράφημα παρατηρούμε ότι τα δυο αποτελέσματα είναι πάρα πολύ κοντά, άρα η μέθοδος που χρησιμοποιήσαμε από άποψη ποιότητας αποτελέσματος είναι αρκετά ικανοποιητική.

ΜΕΡΟΣ 2^{Ο}: Ψηφιακή Σύνθεση Φωνής με Γραμμική Πρόβλεψη (LPC Vocoder)

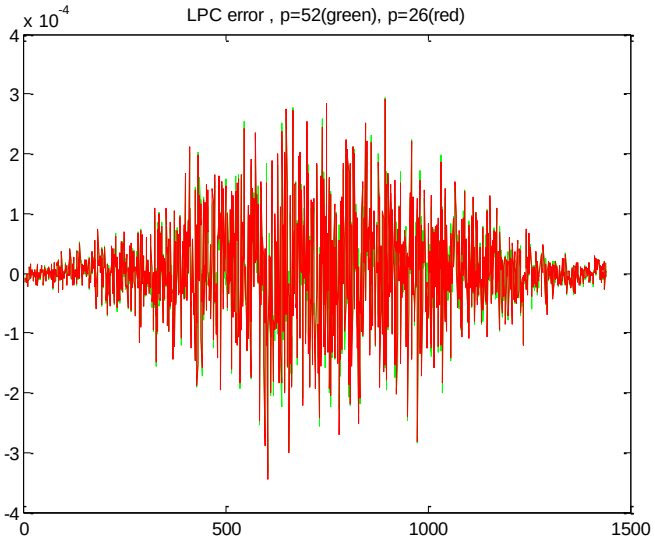
2.1: Ανάλυση Φωνής με Γραμμική Πρόβλεψη

Στο μέρος αυτό, στην ουσία υλοποιούμε ένα σύστημα LPC μοντελοποίησης με χρήση του autocorrelation. Για παράθυρο χρησιμοποιούμε hamming window (όπως είδαμε και στην πρώτη εργαστηριακή αναφορά το hamming window έχει ορισμένα πλεονεκτήματα σε σχέση με το rectangular window) και η τάξη που προβλέπτη που χρησιμοποιούμε είναι p = Fs(kHz) +4 η οποία κρίνεται επαρκής για ανάλυση φωνής και μοντελοποίηση της φασματικής πολυπλοκότητας. Συγκεκριμένα αυτό που έχουμε να κάνουμε περιγράφεται παρακάτω :

- Αρχικά με χρήση της συνάρτησης ***wavread()*** του Matlab εισάγουμε το σήμα φωνής που μας ενδιαφέρει να επεξεργαστούμε (speech.wav) και ταυτόχρονα παίρνουμε και τη συχνότητα δειγματοληψίας, η οποία είναι Fs=48000 Hz .
- Το σύστημα ανάλυσης πρέπει να δέχεται πλαίσια ανάλυσης του αρχικού σήματος, δηλαδή θα έπρεπε να δημιουργήσω ένα hamming window για να μπορέσω να παραθυροποιήσω το σήμα μου, ωστόσο αυτό δεν το κάνω στον κώδικα διότι αυτό γίνεται μέσα στη συνάρτηση ***lpc_analysis()*** που μας δίνεται έτοιμη (η εντολή που δημιουργεί το ζητούμενο hamming window είναι η `w=hamming(0.030*48000);`).
- Έπειτα επιλέγω τυχαία ένα από τα πλαίσια ανάλυσης του αρχικού σήματος μου για να επιδείξω τις ζητούμενες γραφικές παραστάσεις, αφού δεν έχει ιδιαίτερη αξία να γίνει αυτό για καθένα από τα πλαίσια λόγω του μεγάλου τους αριθμού. Στέλνοντας λοιπόν το τυχαίο πλαίσιο ανάλυσης στη συνάρτηση lpc_analysis(), η οποία μας δίνεται έτοιμη, καταφέρνω να πάρω τους συντελεστές γραμμικής πρόβλεψης $\alpha_κ$, το κέρδος G και το σήμα λάθους, τα οποία θέλω, όπως επίσης και το reconstructed σήμα, το οποίο όμως στο μέρος αυτό δεν χρειάζομαι. Η παραπάνω διαδικασία γίνεται για 2 διαφορετικές τιμές της τάξης του προβλέπτη (p=52 και p=26). Πλέον είμαι σε θέση να αναπαραστήσω γραφικά το λάθος της γραμμικής πρόβλεψης και για τις 2 τιμές του προβλέπτη. Τα γραφήματα φαίνονται παρακάτω :



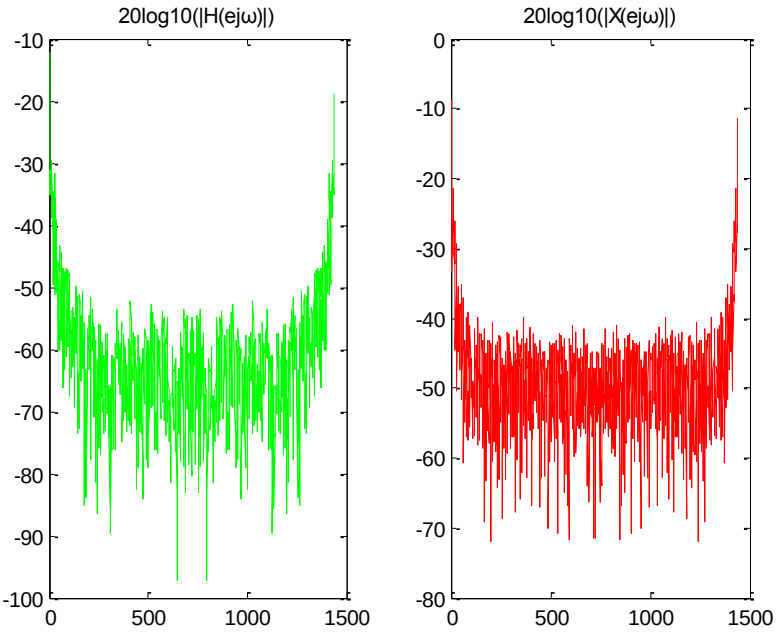
Σε κοινό διάγραμμα έχω το εξής αποτέλεσμα :



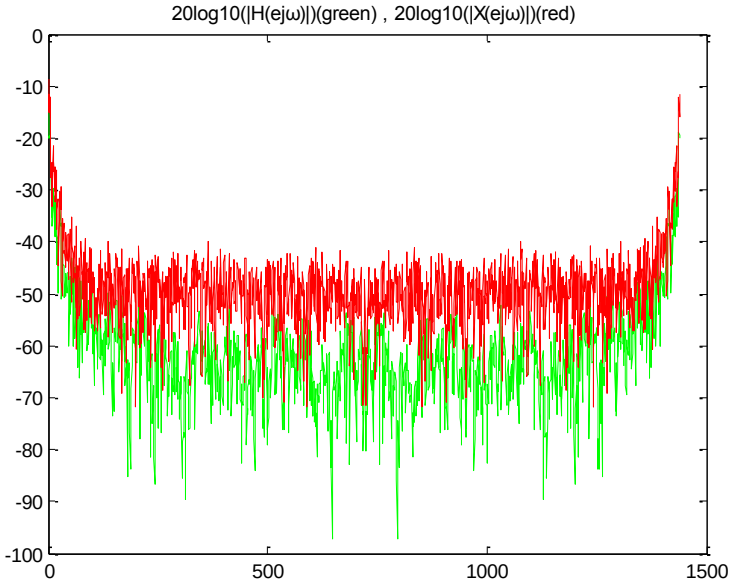
Παρατήρηση : Στο κοινό διάγραμμα παρατηρώ ότι η διαφορές είναι ελάχιστες. Για να παρατηρήσουμε τις διαφορές πρέπει να γίνει κατάλληλη μεγέθυνση . Για τον λόγο αυτό παρέδωσα και τα δυο γραφήματα για να γίνει καλύτερα αντιληπτή η μορφή τους.

- Τέλος, καλούμε να αναπαραστήσω γραφικά σε κοινό διάγραμμα για τις παραπάνω τιμές του προβλέπτη p (52 και 26) το φάσμα του LPC μοντέλου (20log10(|H(ejω)|)) και το φάσμα του παραθυρομένου σήματος x[n] (20log10(|X(ejω)|)) . Τα ζητούμενα γραφήματα παρατίθενται παρακάτω :

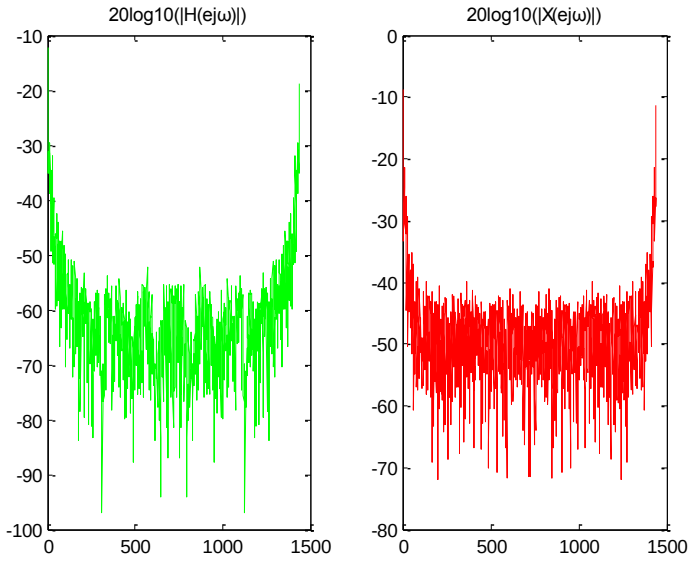
Για p=52 **:**



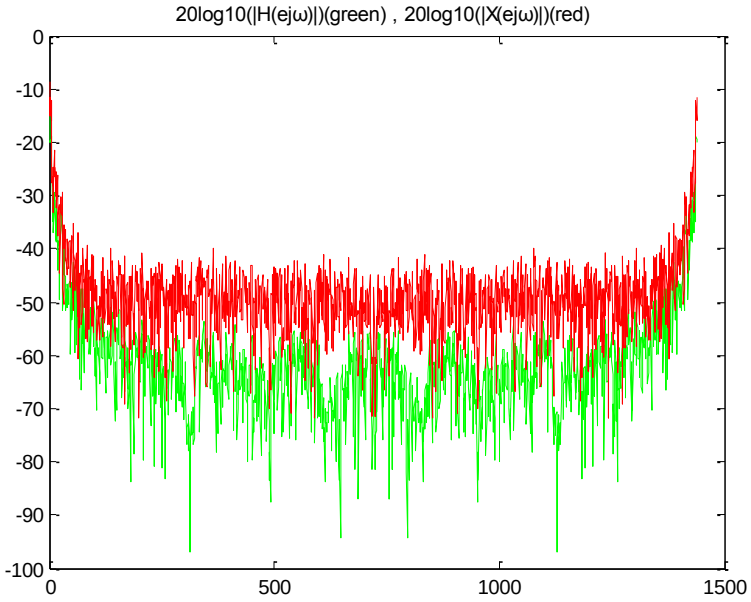
Σε κοινό διάγραμμα έχω το εξής αποτέλεσμα :



Για p=26 **:**



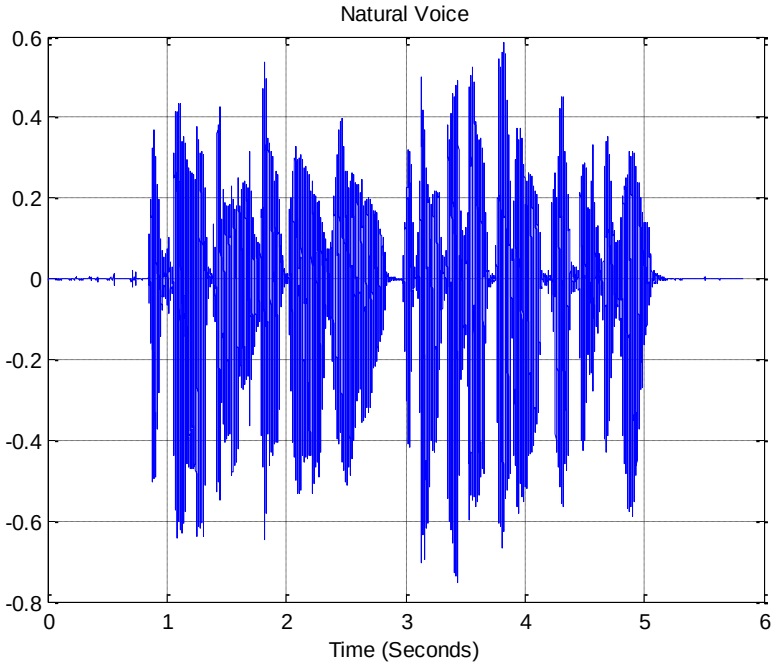
Σε κοινό διάγραμμα έχω το εξής αποτέλεσμα :

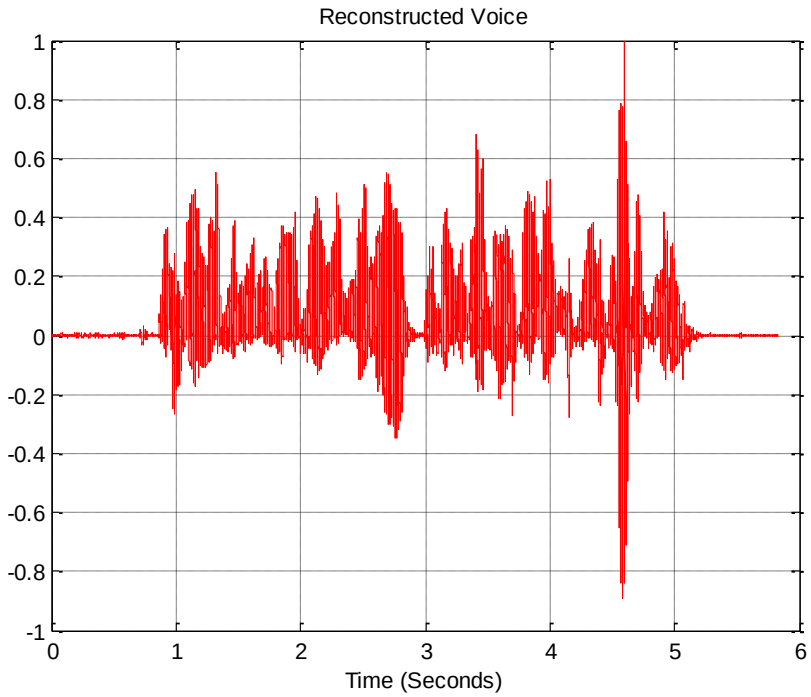


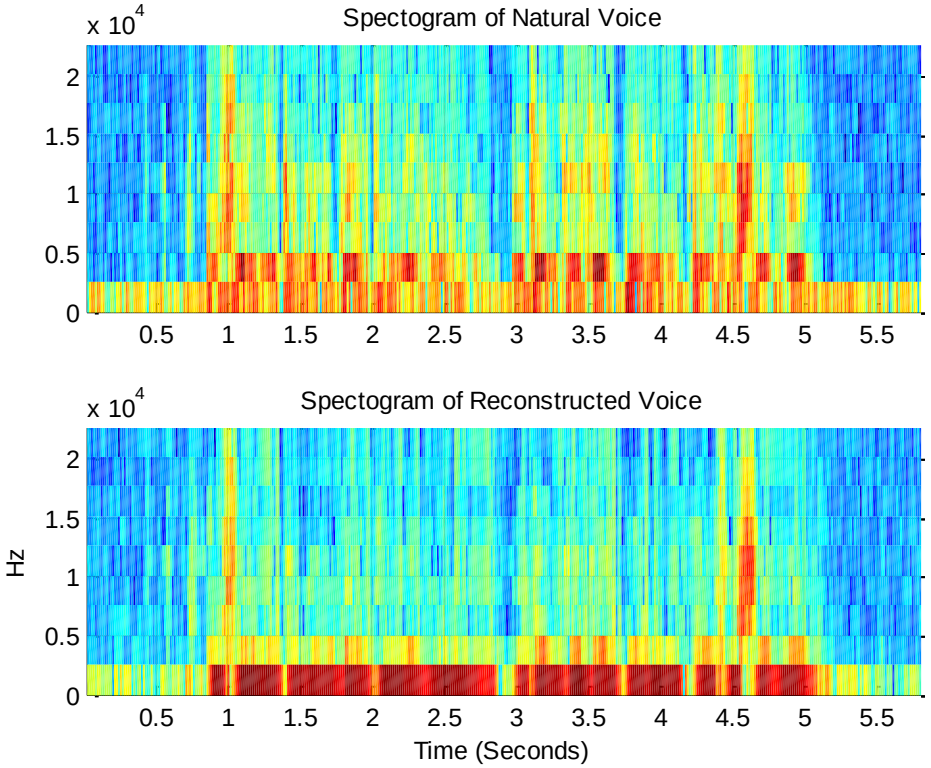
2.2 : <u>Σύνθεση Φωνής με Γραμμική Πρόβλεψη</u>

Στο μέρος αυτό της άσκησης, καλούμαστε να κάνουμε ανάλυση φωνής με γραμμική πρόβλεψη. Για να το κάνουμε αυτό πρέπει η ανάλυση με γραμμική πρόβλεψη να γίνει σε μικρά επικαλυπτόμενα πλαίσια και να υπάρχει επικάλυψη ώστε να διασφαλίσουμε τη συνέχεια και την ομαλότητα της ανάλυσης. Συγκεκριμένα θεωρούμε ότι κάθε πλαίσιο ανάλυσης έχει μήκος 30msec και η επικάλυψη που έχει με το προηγούμενο πλαίσιο είναι ίση με 2L/3 όπως αυτό ζητείται στην εκφώνηση και υλοποιείται στο Matlab. Η διαδικασία που ακολουθείται στο μέρος αυτό περιγράφεται παρακάτω :

- Αρχικά με χρήση της συνάρτησης ***wavread()*** του Matlab εισάγουμε το σήμα φωνής που μας ενδιαφέρει να επεξεργαστούμε (speech.wav) και ταυτόχρονα παίρνουμε και τη συχνότητα δειγματοληψίας, η οποία είναι Fs=48000 Hz.
- Στη συνέχεια φορτώνουμε στο Matlab το αρχείο με όνομα pitch.mat με χρήση της συνάρτησης ***load().***
- Τέλος παιρνάμε τις παραμέτρους που μας υποδεικνύει η εκφώνηση στη δοσμένη συνάρτηση με όνομα ***lpc_vocoder()*** και παίρνουμε τα παρακάτω γραφήματα:







Το ζητούμενο αρχείο το οποίο περιέχει την συνθέτικη φωνή όπως αυτή προέκυψε από την συνάρτηση ***lpc_vocoder()*** υπάρχει στο συμπιεσμένο φάκελο με όνομα synthesis.wav .

ΜΕΡΟΣ 3^{Ο}: Βελτιωμένη Σύνθεση Φωνής με Χρήση Μακροπρόθεσμης Πρόβλεψης και Κατάλληλα Σχεδιασμένης Βάσης Δεδομένων (CELP)

3.1: Εισαγωγή Προβλέπτη μακράς καθυστέρησης

Στην άσκηση αυτή δημιουργούμε ένα προβλέπτη (προβλέπει την περιοδικότητα της φωνής στα σημεία που υπάρχει) και ακολουθεί τον πρώτο προβλ΄πετη κατά την ανάλυση. Είναι γνωστό ότι οι διαδοχικές περίοδοι έχουν μεγάλη ομοιότητα στη περίπτωση έμφωνων ήχων. Εμείς προσπαθούμε να απαλοίψουμε την περιοδικότητα που παρατηρείται στο λάθος του προβλέπτη (lpc). Για τον λόγο αυτό δημιουργούμε ένα νέο προβλέπτη 1^{ης} τάξης ο οποίος είναι και έχει τον παρακάτω τύπο :

$$P_e = \beta \cdot z^{-M}$$

Το σφάλμα της ανάλυσης του γραμμικού προβλέπτη, δηλαδή την παράμετρο lpc_analysis_error που μας επέστρεψε η συνάρτησης lpc_analysis το περνάμε από ένα φίλτρο της μορφής:

$$P_e = \frac{1}{1 - \beta \cdot z^{-M}}$$

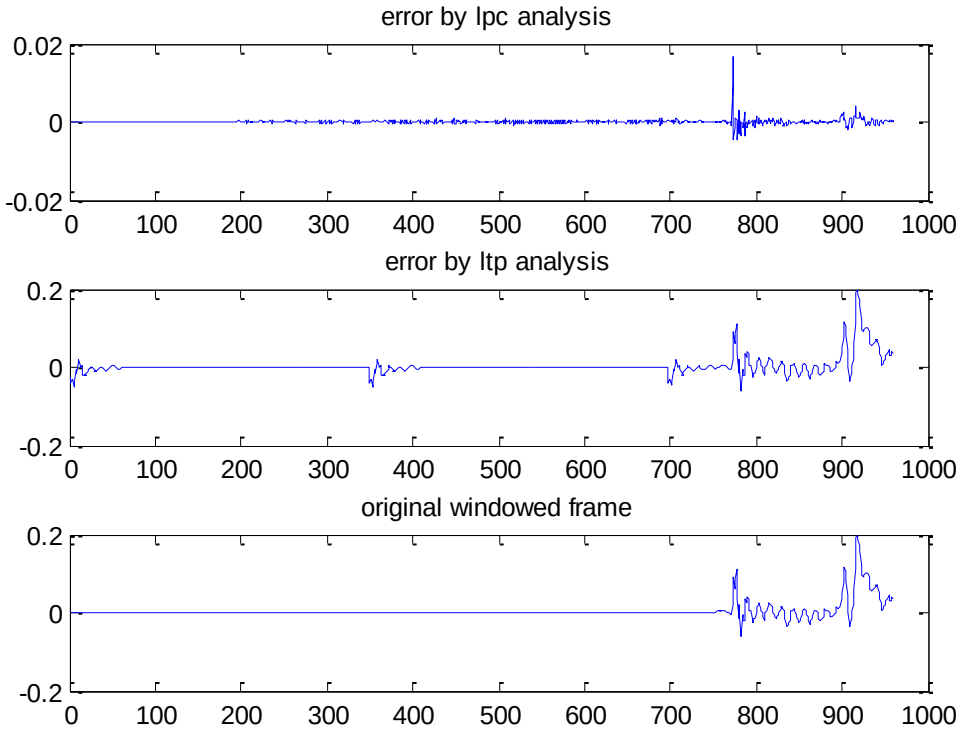
Ο συμβολισμός που χρησιμοποιείται αναλύεται παρακάτω :

- M : Μακρά καθυστέρηση (αντιστοιχεί σε ακέραια πολλαπλάσια της περιόδου του e[n]) .
- β : Συντελεστής γραμμικής πρόβλεψης (τον βρίσκω με το να ελαχιστοποιήσουμε το τετράγωνο του λάθους της νέας πρόβλεψης).

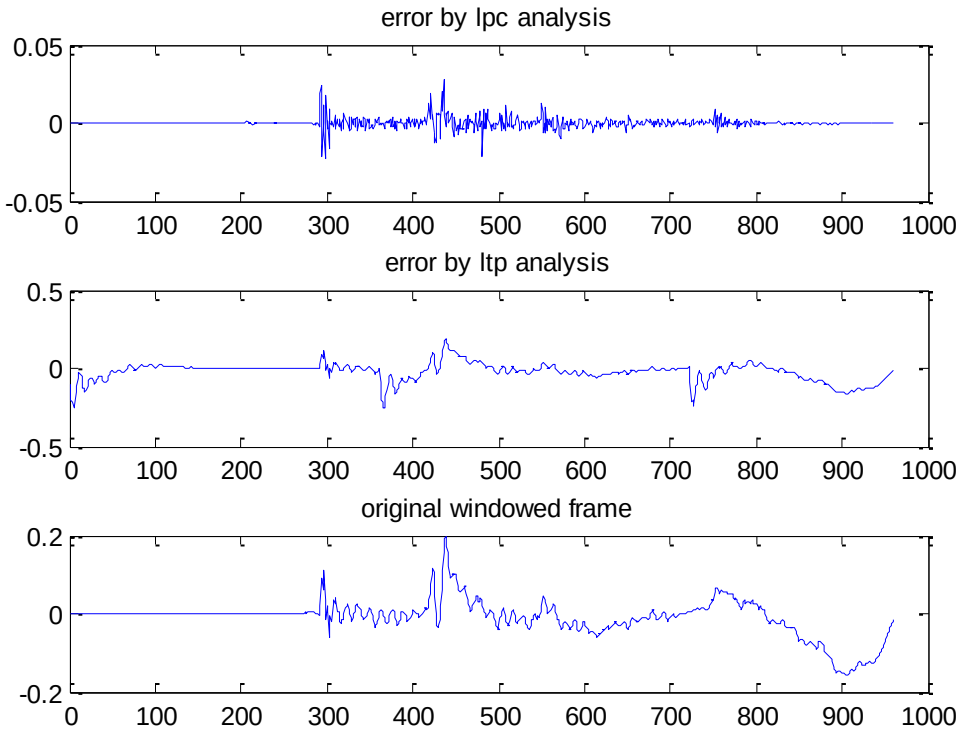
Η διαδικασία που ακολουθήσαμε φαίνεται παρακάτω :

Αρχικά πρέπει να υλοποιήσουμε το παραπάνω φίλτρο και να παραθέσουμε τα διαγράμματα των λαθών: $e[n], \widetilde{e[n]}$ σε αντιπαραβολή με το αρχικό σήμα για έναν *έμφωνο ήχο*. Για τον λόγο αυτό φτιάχνουμε ένα επαναληπτικό βρόχο και επιλέγουμε 4 διαδοχικά παράθυρα (86 έως 89) όπου παρατηρήθηκαν και έμφωνοι και άφωνοι ήχοι.
Για αυτά τα παράθυρα κάνουμε πρώτα lpc ανάλυση, μέσω χρήσης της δοσμένης συνάρτησης lpc analysis() και λαμβάνουμε το λάθος e[n].
Έπειτα το λάθος e[n] το περνάμε από το φίλτρο $P_e = \frac{1}{1-\beta \cdot z^{-M}}$. Την έξοδο αυτού του φίλτρου, την αφαιρούμε από το αρχικό παράθυρο και λαμβάνουμε το $\widetilde{e[n]}$. Στη συνέχεια παρατίθενται τα διαγράμματα των σφαλμάτων και οι τιμές του παραθυροποιημένου σήματος:

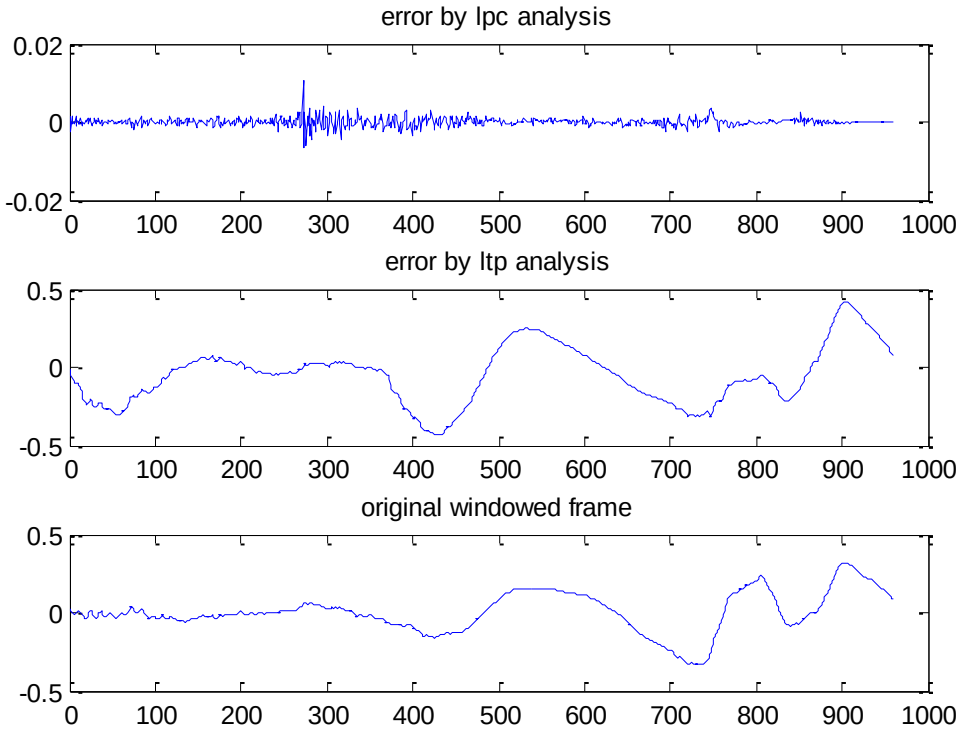
Οι τιμές των $e[n], \widetilde{e[n]}$ και σήματος για το 86ο παράθυρο:



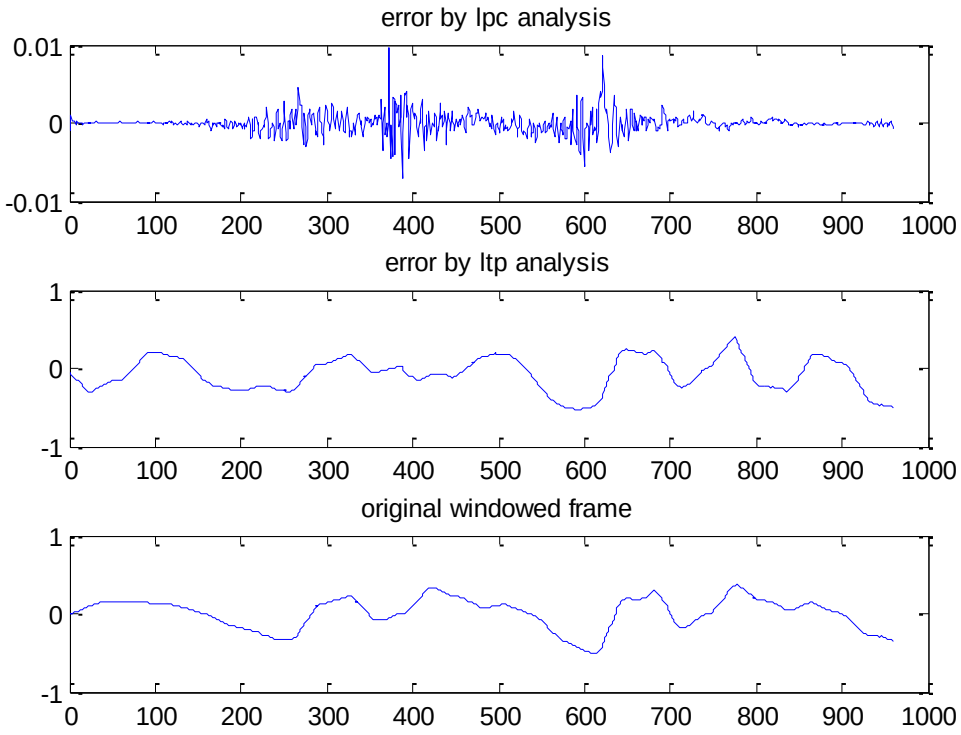
Οι τιμές των $e[n], \widetilde{e[n]}$ και σήματος για το 87ο παράθυρο:



Οι τιμές των $e[n], \widetilde{e[n]}$ και σήματος για το 88ο παράθυρο:



Οι τιμές των $e[n], \widetilde{e[n]}$ και σήματος για το 89ο παράθυρο:



Στο σημείο αυτό εισάγουμε το παρακάτω φίλτρο μακράς πρόβλεψης στον κώδικα της άσκησης 2.2:

$$P_e = \frac{1}{1 - \beta \cdot z^{-M}}$$

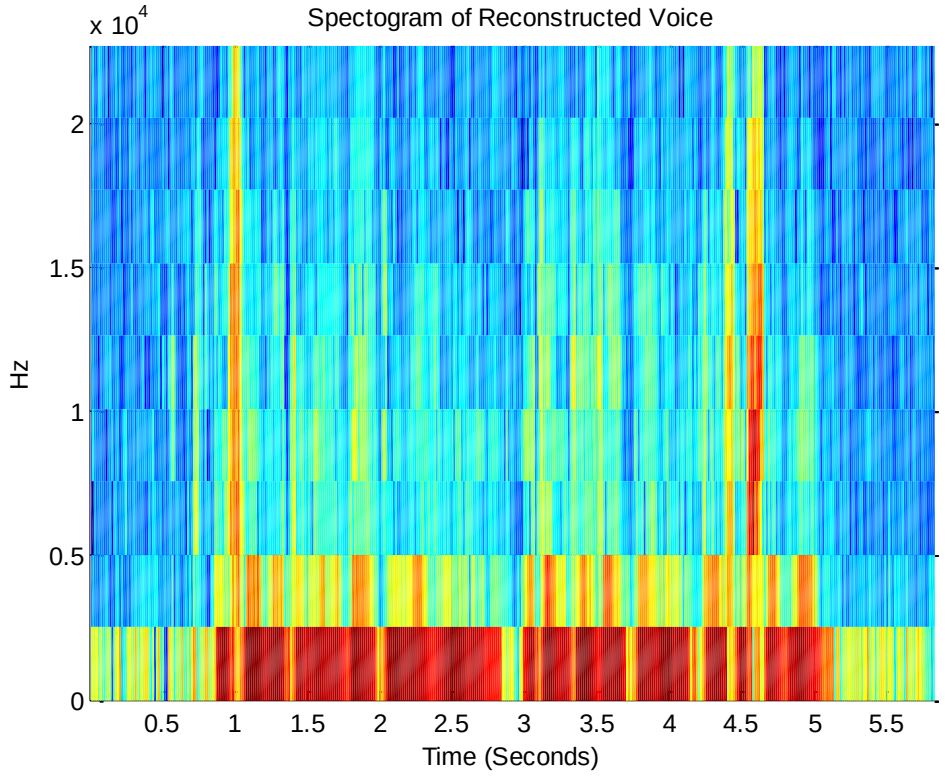
Στην ουσία λαμβάνουμε το error που δίνει η lpc ανάλυση. Για να βρούμε τα β και τα M κάνουμε το εξής:

Αρχικά παίρνουμε την αυτοσυσχέτιση του lpc_error με τη συνάρτηση xcorr η οποία μας προτάθηκε από τους βοηθούς του εργαστηρίου. Προφανώς η αυτοσυσχέτιση είναι συμμετρική, γεγονός το οποίο αν δεν λάβουμε υπόψην μας θα έχουμε πρόβλημα, επιλέγουμε να πάρουμε τις τιμές της από το peak της και μετά , καθώς αυτές πλέον θα αντιστοιχούν σίγουρα σε θετικές συχνότητες. Στη συνέχεια κόβουμε 100 δείγματα από την αρχή αφού η αυτοσυσχέτιση έχει 960 δείγματα και αναζητούμε το μέγιστο αυτής καθώς και το δείκτη που μας δίνει τη θέση αυτού του μεγίστου. Τα β, M υπολογίζονται με τον παρακάτω τρόπο:

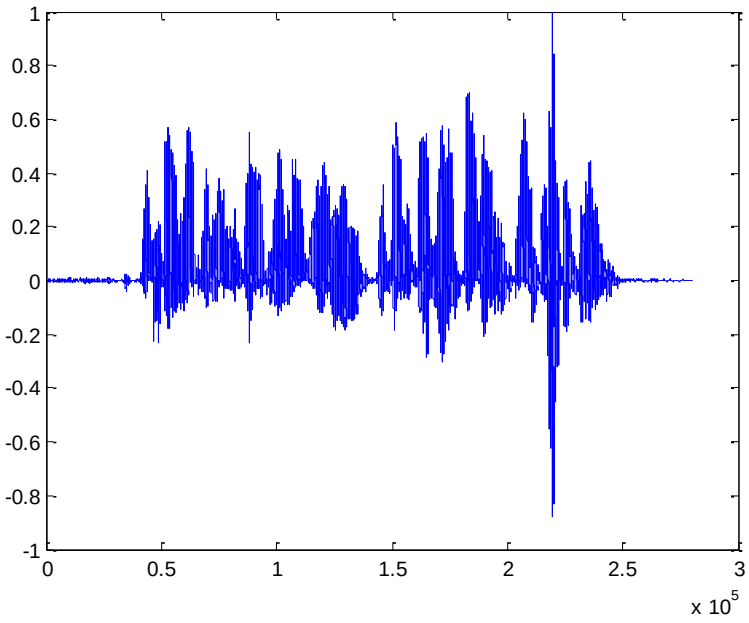
- M= ο δείκτης του μεγίστου αυτού που βρήκαμε
- Και $\beta = \frac{Τιμή\ της\ αυτοσυσχέτισης\ στη\ θέση\ M}{Μέγιστη\ τιμή\ της\ αυτοσυσχέτισης\ του\ σφάλματος\ (peak)}$

Τα αποτελέσματα αυτά φαίνονται στο παραδοτέο source αρχείο Matlab.

Μετά την έξοδο του φίλτρου, παρατηρούμε πως ο ήχος που ανασυνθέσαμε έχει καλύτερη ποιότητα, με μια μικρή προσθήκη θορύβου και ελάχιστη διαφορά σε κόστος bits. Το φασματογράφημα του ήχου που ανασυνθέσαμε είναι το ακόλουθο:

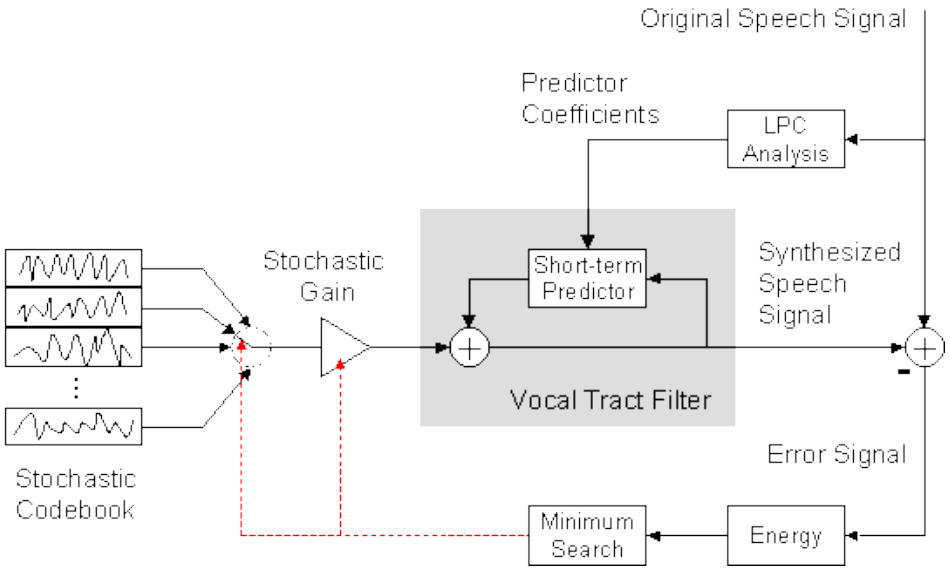


Για πληρότητα παραθέτω και το γράφημα του ήχου που ανασυνθέσαμε :



3.2.: <u>Δημιουργία Βάσης Τυχαίων Διεγέρσεων και Επιλογή της βέλτιστης για σύνθεση</u>

Εδώ εισάγουμε ένα codebook με βάση τυχαίων διεγέρσεων , από τις οποίες θα επιλέγουμε την καταλληλότερη κάθε φορά για την ανασύνθεση της φωνής. Αυτό γίνεται για να βελτιώσει την ποιότητα του ήχου μας.Το παρακάτω σχήμα αποτελεί μια ακριβή αναπαράσταση του αλγορίθμου που ακολουθήσαμε:



Ουσιαστικά στο ερώτημα αυτό ακολουθούμε την ίδια διαδικασία με το προηγούμενο, μόνο που αντί να χρησιμοποιούμε παλμούς της συνάρτησης ugenerator για την ανασύνθεση της φωνής με χρήση της συνάρτησης lpc_vocoder δημιουργούμε 1024 διαφορετικές διεγέρσεις της μορφής:

$$v_n = \sum_{k=0}^{N-1} c_k \cos(\pi kn/N + \phi_k), n = 0, 1, \ldots, 2N-1$$

όπου Ck και Φk είναι τυχαίες μεταβλητές.
Η Φk είναι ομοιόμορφα κατανεμημένη μεταξύ 0 και 2π ενώ η Ck είναι Rayleigh – κατανεμημένη με σ.π.π.:

$$p(c_k) = c_k \exp(-c_k^2/2), c_k > 0$$

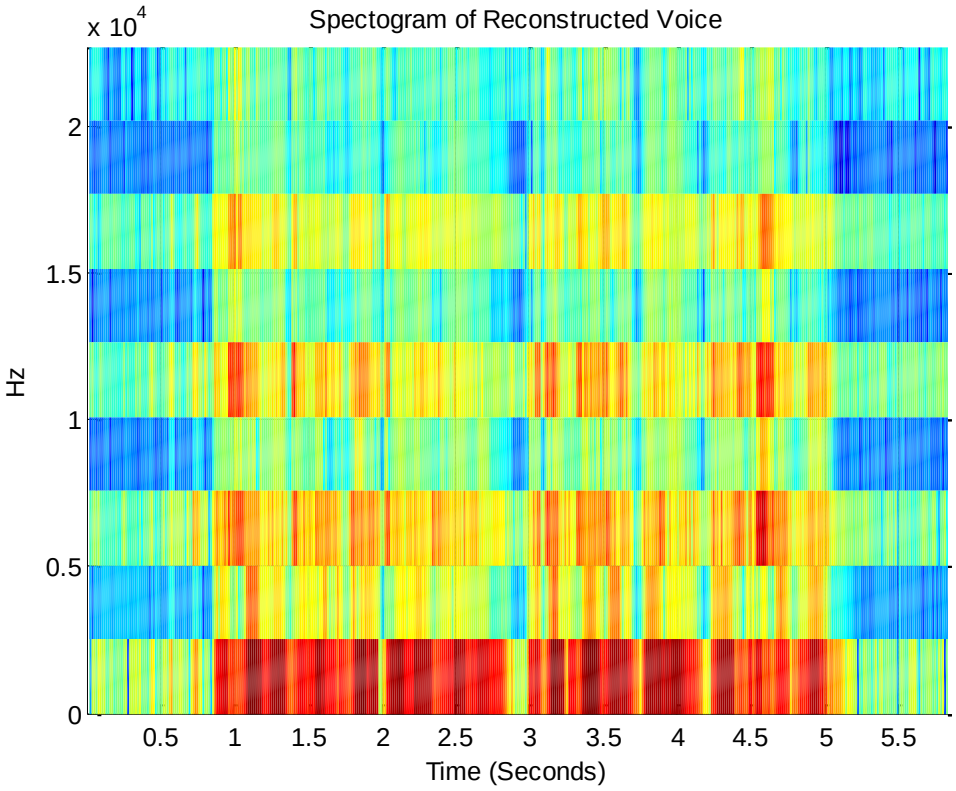
Η διαδικασία που ακολουθήσαμε για τη βελτιωμένη ανασύνθεση φωνής φαίνεται παρακάτω :

- Διαβάζουμε μέσω της συνάρτησης wavread() το σήμα μας.
- Δημιουργούμε παράθυρα 20 ms με overlap 10 ms και παράγουμε το codebook με βάση τις παραπάνω συναρτήσεις.
- Για κάθε παράθυρο του σήματός μας(πλαίσιο), κάνουμε lpc ανάλυση λαμβάνοντας τους συντελεστές a,G και το lpc_error. Με τη διαδικασία που περιγράφηκε και στο προηγούμενο ερώτημα υπολογίζουμε τις τιμές των β, M κάθε πλασίου. Για κάθε παράθυρο περνάμε όλες τις διεγέρσεις Vn από τα φίλτρα που φαίνονται παρακάτω:

  LTP Φίλτρο: $P_e = \frac{1}{1-\beta \cdot z^{-M}}$, με τα β, M κάθε παραθύρου

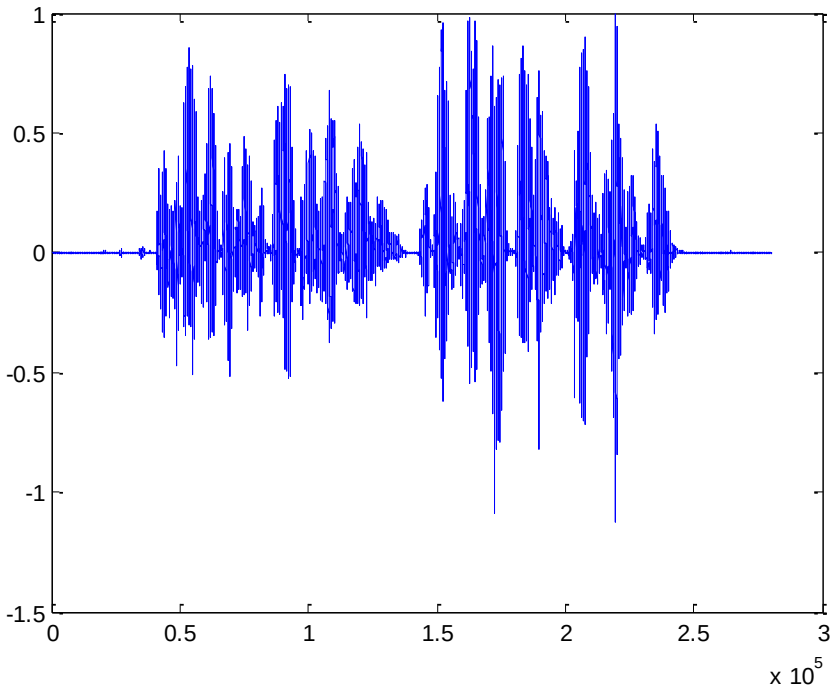
  LPC Φίλτρο: $H(z) = \frac{G}{A(z)}, όπου\ A(z) = 1 - \sum_{k=1}^{p} a_k z^{-k}$

- Αφού πάρουμε τις εξόδους των παραπάνω φίλτρων, υπολογίζουμε τη διαφορά τους από το παραθυροποιημένο σήμα και υπολογίζουμε την ενέργεια του σήματος που προκύπτει. Επιλέγουμε λοιπόν για κάθε παράθυρο τη διέγερση εκείνη που δίνει τη μικρότερη ενέργεια και βρίσκουμε τη καταλληλότερη διέγερση για κάθε πλαίσιο. Τώρα κάνουμε overlap and add και συνθέτουμε αυτές τις διεγέρσεις ώστε να πάρουμε το τελικό σήμα. Το ηχογραφημένο σήμα βρίσκεται στο zip αρχείο, όπως και το φασματογράφημά του. Το φασματογράφημα του φαίνεται και παρακάτω:



<u>Σχολιασμός</u> : Η ποιότητα του ήχου που ανασυνθέσαμε είναι καλύτερη, ωστόσο παρατηρούνται κάποια σημάδια θορύβου που μας την επηρεάζουν κάπως.

Για πληρότητα παραθέτω και το γράφημα του ήχου που ανασυνθέσαμε :



ΜΕΡΟΣ 4^{Ο}: Κωδικοποίηση/Συμπίεση

Εδώ εφαρμόζουμε κωδικοποίηση και συμπίεση του αρχείου ήχου μας. Για να το κάνουμε αυτό θα κβαντίσουμε τις παραμέτρους του LPC συνθέτη. Όμως αντί για τους συντελεστές της γραμμικής πρόβλεψης $\{\alpha_i\}$, θα χρησιμοποιήσουμε τα $g_i = log\frac{1-k_i}{1+k_i}$ (log area ratios), διότι οι συντελεστές γραμμικής πρόβλεψης μπορεί να παρουσιάσουν προβλήματα αστάθειας.Οι συντελεστές k_i που χρησιμοποιούμε παραπάνω είναι οι συντελεστές ανάκλασης (PARCOR) οι οποίοι προκύπτουν από τους συντελεστές $\{\alpha_i\}$ με τον αλγόριθμο Levinson-Durbin.

Δίνεται ότι η συχνότητα της lpc ανάλυσης είναι 100 frames per second.Για την κβάντιση των παραπάνω συντελεστώνχρησιμοποιούμε:

- *5 bits για τον κβαντισμό (σε γραμμική κλίμακα) κάθε παραμέτρου g_i*
- *6 bits για τον κβαντισμό (σε γραμμική κλίμακα) της θεμελιώδους περιόδου του pitch (στον κώδικα του matlab κάνουμε load το pitch που μας δίνεται)*
- *1 bit για το αν ο ήχος είναι έμφωνος ή άφωνος*
- *5 bits για τον κβαντισμό (σε λογαριθμική κλίμακα) του κέρδους G του μοντέλου γραμμικής πρόβλεψης.*

Υποθέτουμε 16 bits ανά δείγμα και βρίσκω τα παρακάτω :

Αρχικός ρυθμός πληροφορίας :

Έχουμε 16 Bits/sample και 279563 samples.

Η διάρκεια του σήματος φωνής είναι 5.87 sec, άρα:

Αρχικός ρυθμός πληροφορίας= $\frac{16*279563}{5,87} = 762011,5843\frac{bits}{sec} = 762\ kbits/sec$

Τελικός ρυθμός πληροφορίας :

Έχουμε:

- 5bits για κάθε ένα από τους 52 όρους g_i του κάθε frame
- 1bit για κάθε frame για το αν ο ήχος είναι έμφωνος ή άφωνος
- 5bits για τον συντελεστή G κάθε frame
- 6bits για τη θεμελιώδη περίοδο του Pitch σε κάθε frame

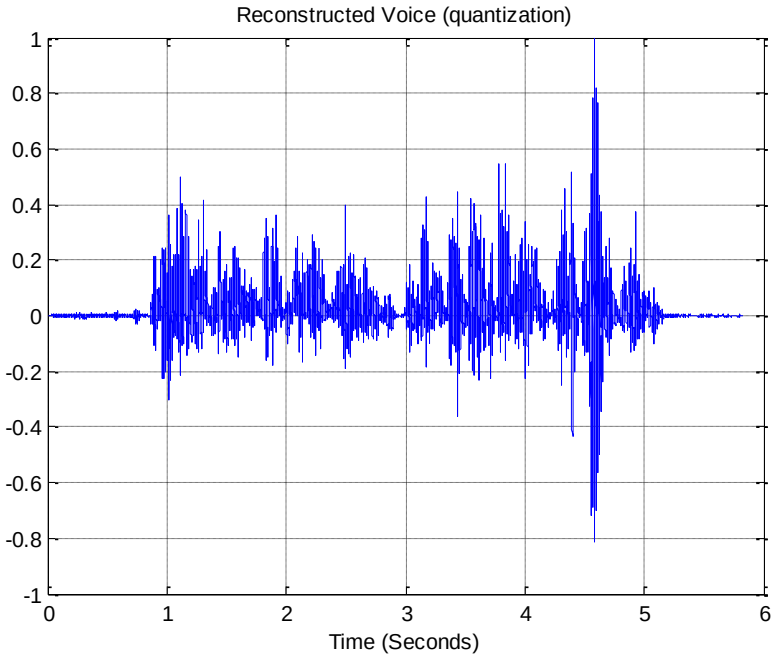
Άρα το σύνολο των frames είναι 584.

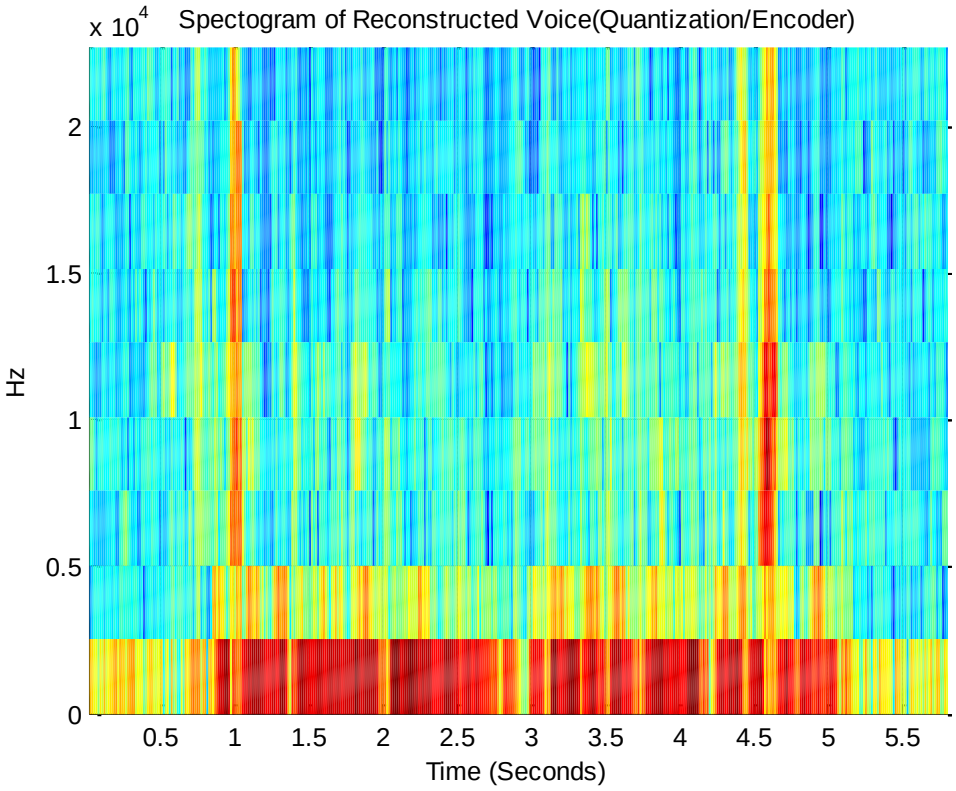
Τελικός ρυθμός πληροφορίας= $\frac{(5*52+1+5+6)}{5.87} * 584 \cong 27\frac{kbits}{sec}$

<u>Λόγος συμπίεσης</u> : $compress_{ratio} = \frac{762}{27} = 28.22$

Όσα αναφέρθηκαν παραπάνω υλοποιούνται με τις συναρτήσεις του Matlab .
Η διαδικασία που ακολουθήσαμε για να παρουμε τα ζητούμενα αποτελέσματα είναι η εξής:

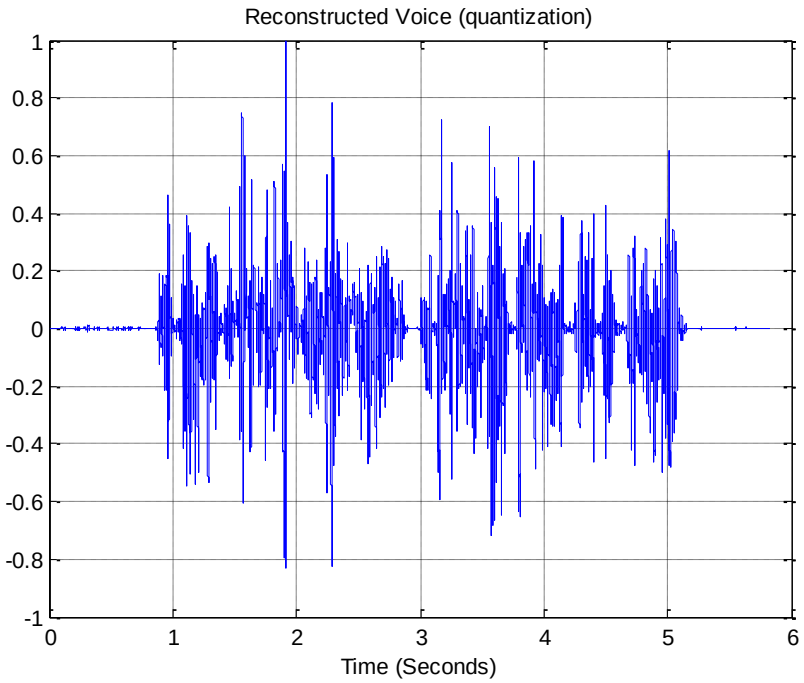
Παίρνουμε το σήμα και παραθυρώνουμε και χρησιμοποιούμε τη συνάρτηση Buffer για να πάρουμε παράθυρα μήκους 30 ms με overlap 20 ms.
Έπειτα περνάμε κάθε παράθυρο από τη συνάρτηση lpc analysis και κρατάμε τους συντελεστές γραμμικής πρόβλεψης καθώς και το Gain κάθε παραθύρου σε έναν πίνακα.
Με χρήση της συνάρτησης ***poly2rc()*** του Matlab υπολογίζουμε τους συντελεστές ki και τελικά βρίσκουμε τους συντελεστές gi. Τους αποθηκεύουμε σε πίνακες ώστε να τους έχουμε στη διάθεση μας συνολικά για όλα τα παράθυρα για να είμαστε σε θέση να τους κβαντίσουμε και για κάθε παράθυρο τσεκάρουμε την τιμή του pitch για να δούμε αν ο ήχος είναι έμφωνος ή άφωνος.
Υλοποιούμε μια συνάρτηση ***linearQuantizer()*** που κβαντίζει τις τιμές που θέλουμε και υπολογίζει τον αριθμό των διαστημάτων ανάλογα με τον αριθμό των Bits. Με αυτή τη συνάρτηση και τα κατάλληλα ορίσματα υλοποιούμε τόσο τις κβαντίσεις σε γραμμική όσο και σε λογαριθμική κλίμακα .
Έχουμε κβαντίσει όλα τα μεγέθη που επιθυμούμε και περνάμε τις κβαντισμένες παραμέτρους σε μια τροποποιημένη συνάρτηση του vocoder, την ***Quantized_Values_Vocoder()*** η οποία κάνει την ανασύνθεση του σήματος φωνής έχοντας ως βάση για τα φίλτρα της, όχι τις πραγματικές παραμέτρους που είχε, αλλά τις νέες κβαντισμένες τιμές. Για την απλή ανασύνθεση φωνής με χρήση του φίλτρου της απλής LPC Analysis παίρνουμε το ακόλουθο σήμα φωνής και το φασματογράφημά, τα οποία φαίνονται παρακάτω:

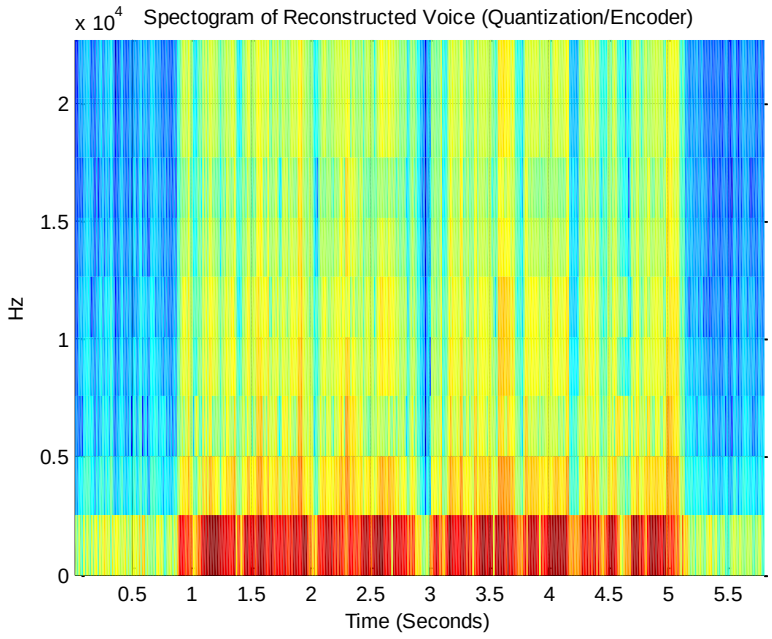




Το σήμα που εξέρχεται είναι το 'synthesis_encoded_lpc.wav' .

Τέλος, για κάθε παράθυρουπολογίζουμε τους συντελεστές β και M και τους κρατάμε και αυτούς σε πίνακες και κβαντίζουμε και αυτούς(5 bits για το β σε γραμμική κλίμακα και 5 bits για το M σε λογαριθμική κλίμακα) .
Καλούμε τώρα με τις κατάλληλες παραμέτρους τη συνάρτηση ***Quantized_Values_Celp_Vocoder()*** η οποία περνάει το σήμα μας από δύο φίλτρα προτού το ανασυνθέσει τα οποία είναι αυτά του ερωτήματος 3.2 με τις νέες κβαντισμένες παραμέτρους. Με τις εξόδους αυτών ανασυνθέτουμε με overlap and add το τελικό σήμα φωνής μας. Όμοια το γράφημα και το φασματογράφημά του σήματος είναι τα ακόλουθα:





Το σήμα που εξέρχεται είναι το 'synthesis_encoded_celp.wav' .

Παρατήρηση1 : Όλα τα γραφήματα υπάρχουν και στο zip αρχείο για πληρότητα.

Παρατήρηση2 :Για τη σωστή λειτουργία του κώδικα στην άσκηση 3 μπορεί να προστεθεί η εντολή clear all στην αρχή του κώδικα για να τρέξει παραπάνω από μια φορές χώρις να γίνεται clear workspace. Στην άσκηση 4 έχει εφαρμοστεί αυτό.