ΑΤΕΙ ΚΡΗΤΗΣ

ΠΑΡΑΡΤΗΜΑ ΡΕΘΥΜΝΟΥ

ΤΜΗΜΑ ΜΟΥΣΙΚΗΣ ΤΕΧΝΟΛΟΓΙΑΣ ΚΑΙ ΑΚΟΥΣΤΙΚΗΣ

ΠΤΥΧΙΑΚΗ

ΔΗΜΙΟΥΡΓΙΑ ΨΗΦΙΑΚΩΝ ΦΙΛΤΡΩΝ IIR ΜΕ ΧΡΗΣΗ ΓΛΩΣΣΑΣ ΠΡΟΓΡΑΜΑΤΙΣΜΟΥ C ΣΕ ΠΕΡΙΒΑΛΛΟΝ ΕΡΓΑΣΙΑΣ LINUX/UNIX

ΣΠΟΥΔΑΣΤΗΣ: ΜΑΥΡΟΓΙΑΝΝΗΣ ΔΗΜΗΤΡΙΟΣ ΑΜ:317

ΕΙΣΗΓΗΤΗΣ: Dr. ΠΟΤΑΜΙΤΗΣ ΗΛΙΑΣ

ΑΘΗΝΑ 2008

ΠΡΟΛΟΓΟΣ

Κατα την διάρκεια της πρακτικής μου εξάσκησης η οποία πραγματοποιήθηκε στην Ελληνική Αεροπορική Βιομηχανία (ΕΑΒ) στον τομέα των δορυφορικών προγραμμάτων και εφαρμογών βρέθηκα αντιμέτωπος με μία πρόκληση. Το αντικείμενο της πρακτικής άσκησης μου εμπεριείχε εφαρμογές αλλά και αντιμετώπιση προβλημάτων τα οποία δεν μου ήταν εφάμιλλα και ανήκαν στον ευρύτερο τεχνολογικό τομέα. Η γνώσεις απο τις σπουδές μου στο τμήμα μουσκής τεχνολογίας και ακουστικής με βοήθησαν έτσι ουτως ώστε να μπορέσω να ανταπεξέλθω αλλά και να βοηθήσω την παραγωγική διαδικασία όταν μου ζητήθηκε. Κατα την διάρκεια της πρακτικής μου γνώρισα και τπ λειτουργικό περιβάλλον Linux/Unix. Πάνω σε αυτό εφάρμοσα τις γνώσεις μου. Ομολογώ πως ήταν η πρώτη φορά που χρησιμοποίησα το εν λόγο λειτουργικό σύστημα. Σαν ένας άνθρωπος που παρακολουθεί τις εξελίξεις γνώριζα για αυτό αλλά ποτε δεν μου είχε δοθεί η ευκαιρία να δουλέψω και να πειραματιστώ περαιτέρω.

Πολύ μεγάλη εντύπωση μου έκανε ότι εφαρμογές υψηλής σημασίας αλλά και ολόκληρα δίκτυα δούλευαν αποκλειστικά με αυτό το Λ.Σ. Τότε γεννήθηκε η ιδέα να χρησιμοποιήσω αυτο το λειτουργικό για την εργασία μου. Η εργασία λοιπόν ουσιαστικά έχει τους παρακάτω σκοπούς: Να κάνει μία πλήρη παρουσίαση του λειτουργικού συστήματος Linux/Unix τέτοια ώστε να μπορέσει ο μέσος χρήστης να καταλάβει περι τίνος πρόκειται, για το πώς δουλεύει και γενικά γιατί αξίζει η χρησιμοποίηση του. Το δεύτερο σκέλος αφορά την ψηφιακή επεξεργασία σήματος. Ένα ισχυρό θεωρητικό υπόβαθρο για την ψηφιακή επεξεργασία σήματος η οποία είναι μείζον σημασίας ειδικά για την μουσική τεχνολογία αλλά και για την τεχνολογία γενικότερα.

Στο πρακτικό μέρος της πτυχιακής ακολουθούν τέσσερα φίλτρα προγραμματισμένα σε γλώσσα προγραμματισμού C και το πώς μπορούν να εφαρμοσθούν πάνω σε audio (wav) αρχεία μέσα απο ένα λειτουργικό σύστημα Linux/Unix. Επίσης ο σχεδιασμός φίλτρων με την βοήθεια του MATLAB και του εργαλείου σχεδίασης FDATool και πως γίνεται οποιοδήποτε IIR φίλτρο εφαρμογή σε audio αρχεία με την βοήθεια του προγράμματος που προγραμματίσθηκε και ποιες είναι οι διαδικασίες που πρέπει να ακολουθήσει κάποιος έτσι ούτως ώστε να επιτύχει αυτό το σκοπό. Κάτι που αξίζει να σημειωθεί είναι ότι όλη η διαδικασία της εργασίας πραγματοποιήθηκε με freeware προγράμματα εκτός από την χρησιμοποίηση του MATLAB.

Με βάση λοιπόν ένα freeware λειτουργικό σύστημα και χρησιμοποιώντας freeware προγράμματα κατάφερα να εφαρμόσω επιτυχώς τα προγραμματισμένα μου φίλτρα σε audio αρχεία και να έχω τα αποτελέσματα που προσδοκούσα.

Καλή ανάγνωση.

ΠΕΡΙΕΧΟΜΕΝΑ

Κεφάλαιο 1 – Εισαγωγή Στο Linux

ΚΕΦΑΛΑΙΟ 2 – Ηχητικά Σήματα

ΚΕΦΑΛΑΙΟ 3 – Εισαγωγή Στην Ψηφιακή Επεξεργασία Σήματος

ΚΕΦΑΛΑΙΟ 4 – IIR Φίλτρα

ΠΡΑΚΤΙΚΟ ΜΕΡΟΣ

ΚΕΦΑΛΑΙΟ 5 - Δομή Αρχείων WAV

ΚΕΦΑΛΑΙΟ 6 – Ο Προγραμματισμός Των Φίλτρων

ΚΕΦΑΛΑΙΟ 7 - Σχεδίαση φίλτρων με την βοήθεια του Filter Design and Analysis Tool στο Matlab

Κεφάλαιο 1

Εισαγωγή στο Linux.

1.1 Τι είναι Λειτουργικό σύστημα (Operating System)

Κάθε Ηλεκτρονικός Υπολογιστής (Η/Υ) αποτελείται από δύο συνθετικά: Το **Υλικό** (Hardware) και το **Λογισμικό** (Software) του. Το Υλικό αποτελούν τα ηλεκτρικά, ηλεκτρονικά και μηχανικά μέρη του Η/Υ ενώ το Λογισμικό αποτελούν τα προγράμματα, δηλαδή οι οδηγίες για το "τι πρέπει να κάνει ο Η/Υ". Το βασικότερο μέρος του Λογισμικού αποτελεί το **Λειτουργικό Σύστημα** (Operating System) το οποίο αποτελείται από τα προγράμματα τα οποία είναι απαραίτητα για την αξιοποίηση του Υλικού (Hardware) και τη λειτουργία του συστήματος του Η/Υ.

Αναλυτικότερα οι βασικές "αρμοδιότητες" του λειτουργικού συστήματος είναι:

• Να λειτουργεί ως ενδιάμεσος (Interface) ανάμεσα στον άνθρωπο και στη μηχανή.

• Να διαχειρίζεται τις δυνατότητες και τους πόρους (resources) του συστήματος υπολογιστή έτσι ώστε να παράγεται χρήσιμο έργο (Resource Allocation).

Με αυτόν τον τρόπο το λειτουργικό σύστημα:

• Μεταφέρει εντολές ή απαιτήσεις του χρήστη στον Η/Υ.

• Δίνει χρήσιμες πληροφορίες στον χρήστη για την κατάσταση του συστήματος.

• Ενεργοποιεί και δίνει οδηγίες στην Κεντρική Μονάδα Επεξεργασίας (CentralProcess Unit) κατανέμοντας τον χρόνο λειτουργίας της στους διάφορους χρήστες.

• Διαχειρίζεται την Κεντρική Μνήμη (RAM) του συστήματος καθώς και τις συσκευές εξόδου και εισόδου, ελέγχοντας την ροή των δεδομένων (είσοδος) και την έξοδο των πληροφοριών (έξοδος).

• Ελέγχει την εκτέλεση των προγραμμάτων των διαφόρων χρηστών.

• Οργανώνει και διαχειρίζεται τα αρχεία του συστήματος.

• Εφαρμόζει μηχανισμούς οι οποίοι βελτιώνουν την Ασφάλεια του υπολογιστή από διάφορους κινδύνους.

1.2 Η διεπαφή χρήστη

1.2.1. Είναι δύσκολο το Linux;

Εάν το Linux είναι δύσκολο στη μάθηση εξαρτάται από το πρόσωπο που ρωτάτε. Έμπειροι χρήστες UNIX θα πουν όχι, επειδή το Linux είναι ένα ιδανικό λειτουργικό σύστημα για δυνατούς χρήστες και προγραμματιστές, και έχει αναπτυχθεί και αναπτύσσεται από τέτοιους ανθρώπους.

Όλα όσα ένας καλός προγραμματιστής επιθυμεί είναι διαθέσιμα: μεταγλωττιστές, βιβλιοθήκες, εργαλεία ανάπτυξης και αποσφαλμάτωσης. Αυτά τα πακέτα υπάρχουν σε κάθε πρότυπη διανομή Linux. Ο μεταγλωττιστής-C περιλαμβάνεται δωρεάν – πράγμα που αντιτίθεται σε πολλές διανομές UNIX οι οποίες απαιτούν δαπάνες αδειοδότησης για αυτό το εργαλείο. Όλη η τεκμηρίωση και τα εγχειρίδια υπάρχουν, και συχνά περιλαμβάνονται παραδείγματα για να σας βοηθήσουν να ξεκινήσετε σε χρόνο μηδέν. Μοιάζει σαν το UNIX και η εναλλαγή μεταξύ του UNIX και του Linux να είναι φυσικό επακόλουθο.

Στον πρώτο καιρό του Linux, το να είναι κάποιος έμπειρος χρήστης έμοιαζε να ήταν προαπαιτούμενο για να ξεκινήσει να εργάζεται με το σύστημα. Αυτοί που κατείχαν το χειρισμό του Linux ένιωθαν καλύτερα από τους υπόλοιπους "αποτυχημένους" που δεν είχαν αφυπνιστεί ακόμα. Ήταν κοινή τακτική να πεις σε έναν αρχάριο χρήστη να "RTFM" (read the f...g. manuals-διάβασε τα γ....α εγχειρίδια). Ενώ τα εγχειρίδια υπήρχαν σε κάθε σύστημα, ήταν δύσκολο να βρεθεί η τεκμηρίωση, και ακόμα και αν κάποιος την έβρισκε, οι εξηγήσεις ήταν σε τόσο τεχνικούς όρους που ο νέος χρήστης αποθαρρυνόταν εύκολα από την εκμάθηση του συστήματος.

Η επιτροπή χρηστών-Linux άρχιζε να αντιλαμβάνεται ότι για να γίνει το Linux κάποτε σημαντικός ανταγωνιστής στην αγορά λειτουργικών συστημάτων, έπρεπε να γίνουν σημαντικές αλλαγές στην προσβασιμότητα του συστήματος.

1.2.2 Το Linux για μη-έμπειρους χρήστες

Εταιρίες όπως η RedHat, η SuSE, η Mandriva (πιο πρόσφατα ή Canonical) ιδρύθηκαν, παρέχοντας συσκευασμένες διανομές Linux κατάλληλες για μαζική κατανάλωση. Αυτές ολοκλήρωσαν πολλές από τις γραφικές διεπαφές χρήστη (GUI), που είχαν αναπτυχθεί από την κοινότητα, για να διευκολύνουν τη διαχείριση προγραμμάτων και υπηρεσιών. Ως χρήστης Linux σήμερα έχετε τα μέσα για να γνωρίσετε το σύστημα σας από μέσα προς τα έξω, αλλά δεν είναι πλέον απαραίτητο να έχετε αυτή τη γνώση για να κάνετε το σύστημα σας να συμμορφώνεται με τα αιτήματα σας.

Στη σημερινή εποχή μπορείτε να συνδεθείτε γραφικά και να ξεκινήσετε όλες τις απαιτούμενες εφαρμογές χωρίς καν να χρειάζεται να πληκτρολογήσετε ούτε έναν χαρακτήρα, ενώ έχετε ακόμη την δυνατότητα να έχετε πρόσβαση στον πυρήνα του συστήματος εάν αυτό χρειάζεται. Εξαιτίας της δομής του, το Linux επιτρέπει στον

χρήστη να αναπτυχθεί μέσα στο σύστημα: το Linux ταιριάζει το ίδιο σε παλιούς αλλά και νέους χρήστες. Οι νέοι χρήστες δεν εξαναγκάζονται να κάνουν δύσκολα πράγματα, ενώ οι έμπειροι χρήστες δεν εξαναγκάζονται να δουλεύουν με τον ίδιο τρόπο όπως δούλευαν όταν αρχικά ξεκίνησαν να μαθαίνουν Linux.

Όσο η ανάπτυξη στον τομέα των υπηρεσιών συνεχίζεται, σπουδαία πράγματα γίνονται για τους χρήστες υπολογιστών γραφείου, που θεωρούνται γενικά σαν την ομάδα που είναι λιγότερο πιθανό να κατανοήσει πως δουλεύει ένα σύστημα. Όσοι αναπτύσσουν εφαρμογές υπολογιστών γραφείου καταβάλουν απίστευτες προσπάθειες να δημιουργήσουν τους πιο όμορφους υπολογιστές γραφείου που έχετε δει, ή να κάνουν την Linux μηχανή σας να μοιάζει σαν τον προηγούμενό σας MS Windows ή MacIntosh σταθμό εργασίας. Οι τελευταίες υλοποιήσεις επίσης περιλαμβάνουν υποστήριξη επιτάχυνσης 3D και υποστήριξη για συσκευές USB, ενημερώσεις του συστήματος και εγκατάσταση / ενημέρωση πακέτων λογισμικού με ένα κλικ, κ.α.. Το Linux τα έχει αυτά, και προσπαθεί να παρουσιάσει όλες τις διαθέσιμες υπηρεσίες σε λογική μορφή που μπορούν να κατανοήσουν οι συνηθισμένοι άνθρωποι. Παρακάτω βρίσκεται μια σύντομη λίστα που περιέχει μερικά σπουδαία παραδείγματα· αυτοί οι ιστοτόποι περιέχουν πολλές εικόνες που θα σας δώσουν μια ιδέα από το πως μπορεί να μοιάζει μια επιφάνεια εργασίας Linux:

- http://www.gnome.org
- http://kde.org/screenshots/
- http://www.openoffice.org
- http://www.mozilla.org

1.3 Εκδόσεις του Linux

1.3.1. Linux και GNU

Το Linux μπορεί να φαίνεται διαφορετικό ανάλογα με τη διανομή, το υλικό και το προσωπικό σας γούστο, αλλά οι βασικές αρχές στις οποίες χτίζονται όλες οι γραφικές και οι άλλες διασυνδέσεις, παραμένουν οι ίδιες. Το σύστημα Linux βασίζεται στα GNU εργαλεία (το Gnu Δεν είναι UNIX), που παρέχουν ένα σύνολο από καθορισμένους τρόπους χειρισμού και χρήσης του συστήματος. Όλα τα εργαλεία GNU είναι Ανοιχτού Κώδικα, οπότε μπορούν να εγκατασταθούν σε κάθε σύστημα. Οι περισσότερες διανομές προσφέρουν προ-μεταγλωττισμένα πακέτα από τα πιο συχνά εργαλεία, όπως πακέτα RPM σε RedHat και πακέτα Debian (αποκαλούμενα επίσης deb ή dpkg) σε Debian, οπότε δεν χρειάζεται να είστε προγραμματιστές για να εγκαταστήσετε ένα πακέτο στο σύστημα σας. Παρόλαυτα, αν κάνετε και σας αρέσει να κάνετε πράγματα μόνοι σας, θα απολαύσετε το Linux ακόμα περισσότερο, εφόσον οι περισσότερες διανομές περιέχουν ένα πλήρες σετ από εργαλεία ανάπτυξης, επιτρέποντας την εγκατάσταση νέου λογισμικού καθαρά από τον πηγαίο κώδικα. Το στήσιμο αυτό σας επιτρέπει επίσης να εγκαταστήσετε λογισμικό ακόμα και αν δεν υπάρχει σε προ-πακεταρισμένη μορφή κατάλληλη για το σύστημα σας.

Μία λίστα συνηθισμένου GNU λογισμικού:

- Bash: Ο φλοιός GNU
- GCC: Ο GNU Μεταγλωττιστής C
- GDB: Ο Αποσφαλματωτής GNU
- Coreutils: ένα σετ βασικών βοηθημάτων του UNIX στυλ, όπως **ls**, **cat** και **chmod**
- Findutils: για αναζήτηση και εύρεση αρχείων
- Fontutils: για μετατροπή γραμματοσειρών από μια μορφοποίηση σε άλλη ή για δημιουργία νέων γραμματοσειρών
- Το Gimp: GNU Image Manipulation Program (Πρόγραμμα Χειρισμού Εικόνων GNU)
- Gnome: Το περιβάλλον επιφάνειας εργασίας GNU
- Emacs: ένας πολύ δυνατός επεξεργαστής κειμένου
- Ghostscript και Ghostview: διερμηνευτής και γραφικό frontend για αρχεία PostScript.
- GNU Photo: λογισμικό για αλληλεπίδραση με ψηφιακές κάμερες
- Octave: μια γλώσσα προγραμματισμού, με κύριο σκοπό την πραγματοποίηση αριθμητικών υπολογισμών και την επεξεργασία εικόνων.
- GNU SQL: συσχετιστικό σύστημα βάσεων δεδομένων
- Radius: ένας απομακρυσμένος server πιστοποίησης και υπολογισμών

Πολλές εμπορικές εφαρμογές είναι διαθέσιμες για Linux, και για περισσότερες πληροφορίες σχετικά με αυτά τα πακέτα αναφερόμαστε στην ειδική τους τεκμηρίωση.

Για να εγκαταστήσετε παραλειπόμενα ή νέα πακέτα, θα χρειαστείτε κάποιο τύπο διαχείρισης λογισμικού. Οι πιο κοινές υλοποιήσεις περιλαμβάνουν τα RPM και dpkg. Το RPM είναι ο διαχειριστής πακέτων του RegHat (RedHat Package Manager), που χρησιμοποιείται σε πληθώρα συστημάτων Linux, ακόμα και αν το όνομα του δεν το

υποδηλώνει αυτό. Το Dpkg είναι το σύστημα διαχείρισης πακέτων του Debian, το οποίο χρησιμοποιεί μια διασύνδεση που ονομάζεται **apt-get**, η οποία μπορεί επίσης να διαχειριστεί και τα πακέτα RPM. Το Novell Ximian Red Carpet είναι μια έμμεσα συσχετιζόμενη υλοποίηση του RPM με γραφικό front-end. Άλλοι πωλητές σχετικού λογισμικού μπορεί να έχουν τις δικές τους διαδικασίες εγκατάστασης, που μερικές φορές παρομοιάζονται με το InstallShield και άλλα, όπως είναι γνωστό για το MS Windows και άλλες πλατφόρμες (πχ ο Synaptic Package Manager του Ubuntu). Όσο προχωράμε με το Linux, πιθανόν θα έρθετε σε επαφή με ένα ή περισσότερα από αυτά τα προγράμματα.

1.4 ΛΙΣΤΑ ΔΙΑΝΟΜΩΝ

Το Linux είναι ένα. Οι διανομές διαφέρουν κυρίως στα προγράμματα εγκατάστασης. Αφού εγκαταστήσετε το Linux θα μάθετε να δουλεύετε το λειτουργικό ακριβώς με τον ίδιο τρόπο ανεξάρτητα από το ποιά διανομή εγκαταστήσατε.

Για νέους χρήστες

Fedora (www.fedoraproject.org)

Mandriva (www.mandriva.com)

MEPIS (www.mepis.org)

OpenSuse (www.opensuse.org)

PCLinuxOS (www.pclinuxos.com)

Ubuntu (www.ubuntu.com)

Για προχωρημένους χρήστες

Arch (www.archlinux.org)

Debian (www.debian.org)

Gentoo (www.gentoo.org)

Slackware (www.slackware.com)

CentOs (www.centos.org)

Ελληνικές

Antix (www.antix.mepis.org)

Iloog (www.ilug.gr/iloog)

Knoppel (www.knoppel.org)

Slackel (www.slackel.gr)

Live

Damn Small Linux (www.damnsmalllinux.org)

Knoppix (www.knoppix.org)

Puppy (www.puppylinux.org)

Slax (www.slax.com)

Εξειδικευμένες

Dynebolic (www.dynebolic.org)

Edubuntu (www.edubuntu.org)

Scientific (www.scientificlinux.org)

1.5 Linux και Sound Engineering

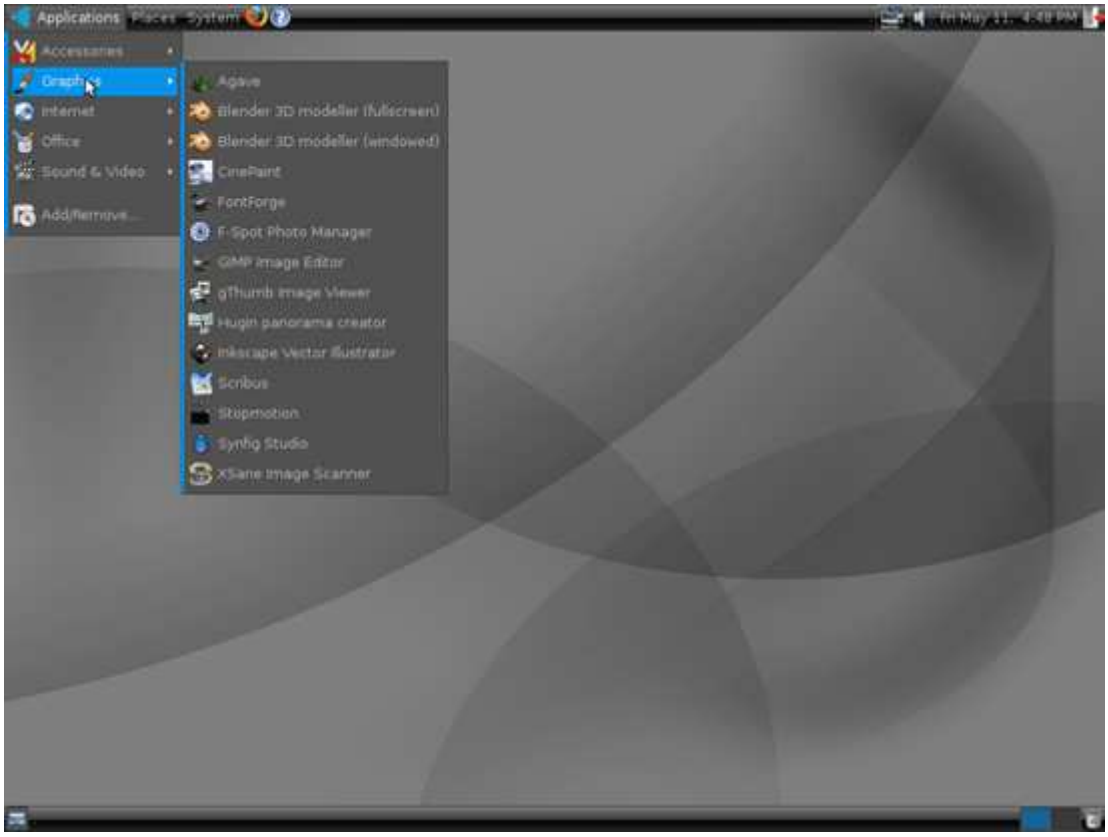
Παρακάτω θα μιλήσουμε για τις διανομές Linux οι οποίες είναι σχεδιασμένες για τον επαγγελματία μηχανικό ήχου. Σχεδόν όλες διατίθενται σαν ελεύθερο λογισμικό.

1.5.1Ubuntu Studio

Ιδιαίτερη μνεία πρέπει να γίνει για την έκδοση **Ubuntu Studio** η οποία προορίζεται για τον Επαγγελματία των πολυμέσων.

Δεν εστιάζει μόνο στο audio κομμάτι αλλά στο video και το animation . Η δύναμη αυτής της διανομής είναι ότι βασίζεται στην έκδοση Ubuntu η οποία είναι πολύ δημοφιλείς και υπάρχουν άπειρα tutorials για να “πειράξουμε” και να μάθουμε την παραπάνω έκδοση σε συνδυασμό με την ευκολία της χρήσης της.

Για περισσότερε πληροφορίες από την ιστοσελίδα http://ubuntustudio.org/ από την οποία μπορούμε να προμηθευτούμε και την διανομή.



Εικ 1.5.1 Ubuntu Studio

1.5.2 64Studio

Επόμενη διανομή είναι η 64Studio. Αυτή η διανομή βασίζεται σε μια παλιότερη διανομή του Linux την Debian (Η Ubuntu επίσης βασίζεται στην Debian.). Η παρούσα έκδοση έχει σχεδιαστεί για 64-bit συστήματα και η εταιρία η οποί διανέμει την παρούσα έκδοση παρέχει τεχνική υποστήριξη για την έκδοση. Η παρούσα έκδοση είναι απευθύνεται κυρίως στους επαγγελματίες του ήχου αλλά εμπεριέχει εφαρμογές και για πολυμέσα.

Το επίσημο site της διανομής : http://www.64studio.com/

Εικ 1.5.2 64studio workspace

1.5.3 Studio To Go

Η τελευταία έκδοση είναι μια έκδοση “Live” Αυτό σημαίνει ότι μπορούμε να bootαρουμε την έκδοση από το CD/DVD και να έχουμε ένα πλήρως λειτουργικό σύστημα χωρίς να πειράξουμε τις ρυθμίσεις του σκληρού μας δίσκου. Εάν όμως χρειαστεί η παρούσα έκδοση μπορεί να γίνει κανονικά εγκατάσταση στο σκληρό μας δίσκο όπως ακριβώς συμβαίνει με πολλές Live εκδόσεις. Η συγκεκριμένη έκδοση προορίζεται για μηχανικούς ήχου και μπορούμε να την αποκτήσουμε με 100 ευρώ που στην οποία θα πληρώσουμε την υποστήριξη. Εμπεριέχει εργαλεία για επαγγελματικές ηχογραφήσεις, mastering, sequencing και notation. Στο παρακάτω screenshot φαίνεται και η υποστήριξη σε VST plugins και η έκδοση έρχεται και με μια πλειάδα Soft Synths τα οποία μπορούμε να χρησιμοποιήσουμε.

Το site για την συγκεκριμένη έκδοση: http://www.studio-to-go.com/

Εικ 1.5.3 Studio To Go

1.6 Audio Software

1.6.1 Ardour

Στο λειτουργικό μας σύστημα θα χρειαστούμε ένα DAW (Digital audio workstation). Το κορυφαίο πρόγραμμα που υπάρχει αυτή την στιγμή για Linux είναι το Ardour.

Είναι ένα πρόγραμμα που προσπαθεί να προσομοιώσει την αίσθηση του δημοφιλέστερου DAW και επαγγελματικού standar Pro –Tools. Υποστηρίζει πλήρες αυτοματισμό, loop recording, υποστήριξη εξωτερικών συσκευών έλεγχου, LADSPA, plugin support, μπορεί να συγχρονίσει σε MIDI time code και έχει ένα επαγγελματικό περιβάλλον editing.

Ένα από τα ποιο σημαντικά χαρακτηριστικά του Ardour είναι η δυνατότητα του Crash Recovery. Μέσα σε λιγοστό χρόνο το πρόγραμμα επανεκκινείτε κρατώντας όλες τις ρυθμίσεις ακριβώς πριν το όποιο crash. Το μόνο ίσως μειονέκτημα του προγράμματος είναι η υποστήριξη MIDI η οποία όμως εξελίσσεται συνεχώς.

Η επίσημη σελίδα που μπορούμε να προμηθευτούμε το πρόγραμμα : http://ardour.org/

Να σημειώσουμε ότι υπάρχει και έκδοση για τους κατόχους Mac και λειτουργικά MacOsX.

Εικ 1.6.1 Επιφάνεια Εργασίας του Ardour.

1.7 Audio Editors

Υπάρχουν 3 πάρα πολλοί καλοί audio editors για Linux. Το **audacity** (το οποίο και χρησιμοποιήσαμε) , το **Sweep** και το **Rezound**. Και τα 3 προγράμματα εξελίσσονται συνεχώς και έχουν παρόμοιες λειτουργίες. Το audacity επίσης υπάρχει και σε έκδοση για λειτουργικό Windows Xp και Windows Vista, όπως επίσης και σε έκδοση Mac Os X.

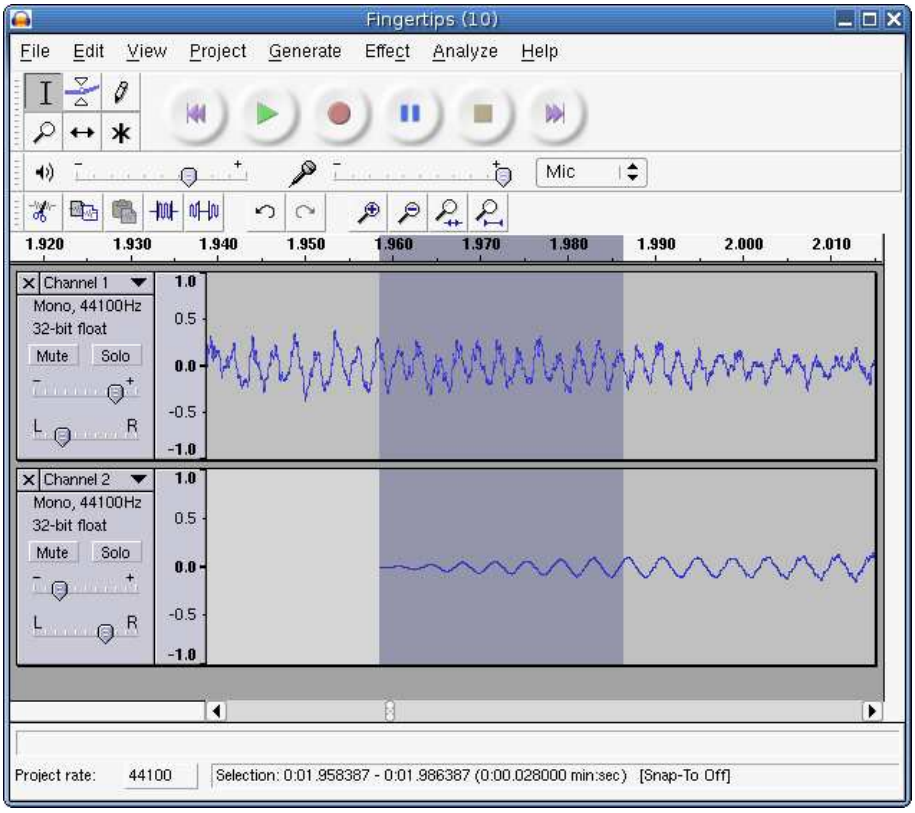
Υπάρχουν όλα τα features που περιμένει κανείς σε έναν sound editor όπως το να μπορείς να κάνεις zoom με την ροδέλα του mouse, υποστηρίζουν LADSP, plugins, Scrubbing, 32-bit αρχεία και ζωντανή ηχογράφηση audio.

http://audacity.sourceforge.net/

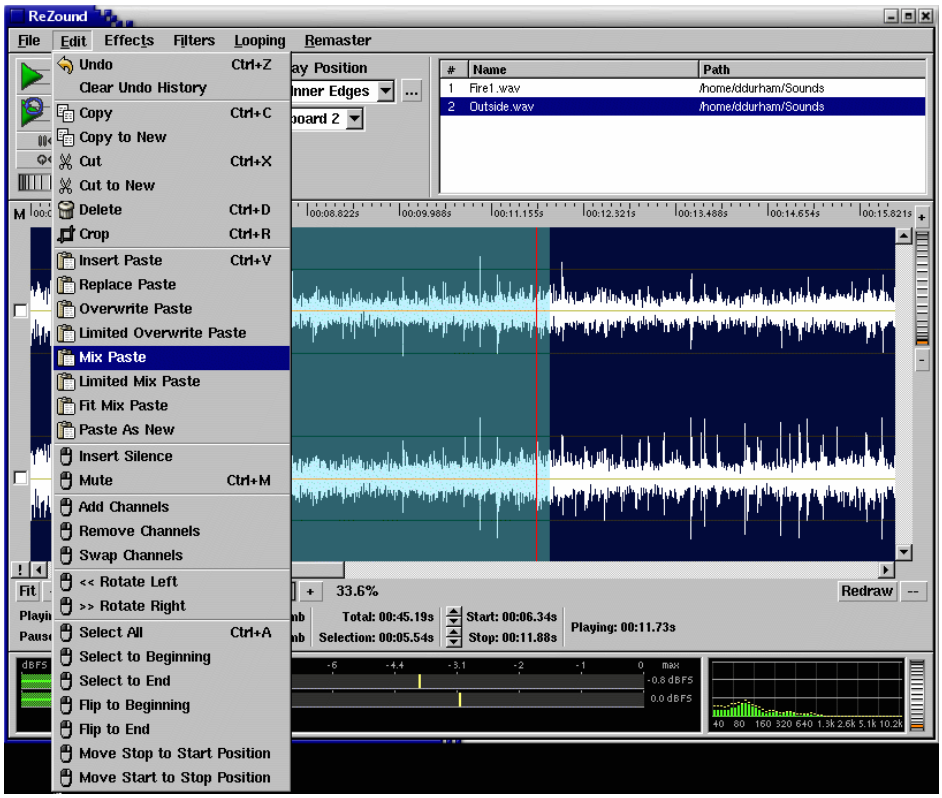
http://rezound.sourceforge.net/

http://www.metadecks.org/software/sweep/

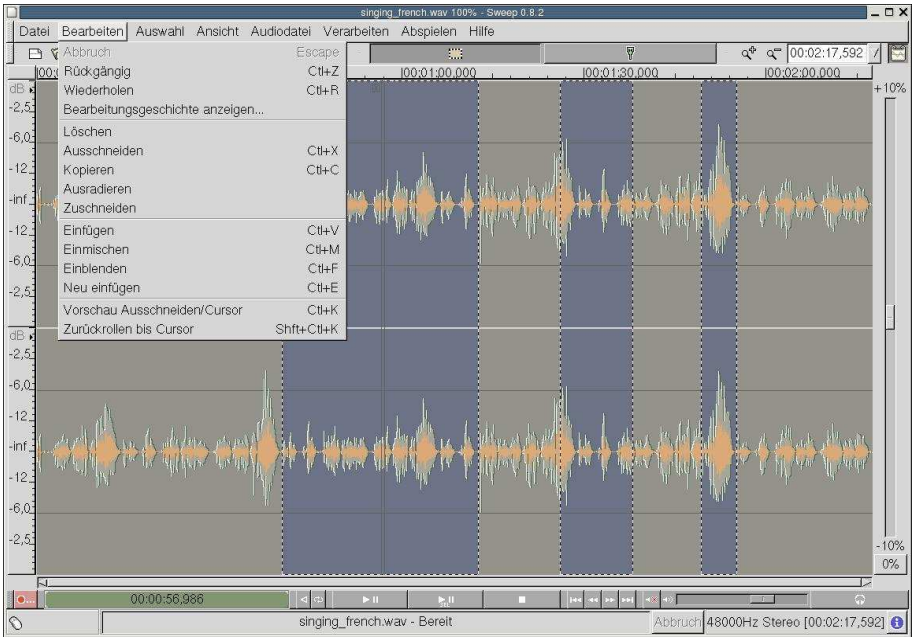
Παρακάτω βλέπουμε 3 χαρακτηριστικά screenshots από κάθε πρόγραμμα.



Εικ 1.7 Το Audacity σε περιβάλλον Linux.



Εικ 1.7.1 Rezound Workspace



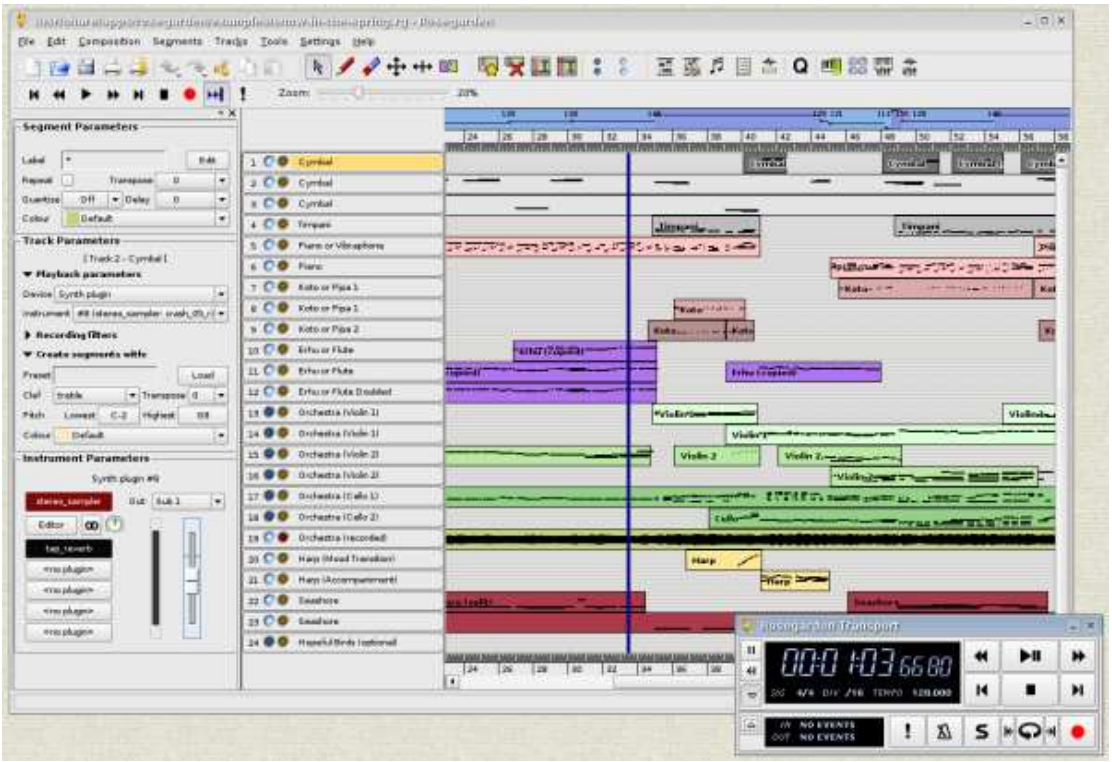
Εικ 1.7.2 Sweep Workspace

1.8 Midi Sequencing

Υπάρχουν δύο πολύ χρήσιμα προγράμματα για MIDI sequencing, το **Rosergarden** και το **Hydrogen**.

1.8.1 Rosergarden

Το Rosergarden είναι σχεδόν ένα ολοκληρωμένο DAW, αλλά εστιάζει στο MIDI sequencing. Μπορέι να δείχνει λίγο παλιομοδίτικο αλλά πραγματικά κάνει την δουλειά του και την κάνει καλά. Σε μερικές περιοχές είναι κορυφαίο. Έχει μια πραγματικά αξιοζήλευτη υποστήριξη εξωτερικών synth όπως επίσης και πάρα πολύ καλά notation features. Υποστηρίζει LADSPA όπως επίσης JACK και έτσι μπορεί να συνεργαστεί σχεδόν με όλα τα audio προγράμματα για Linux.



Εικ 1.8.1 Rosergarden Workspace.

1.8.2Hydrogen

Το hydrogen είναι σαν το module του Reason Redrum μόνο που είναι ποιο δυνατό. Είναι ένα pattern- based sequencer που μπορούμε να φορτώσουμε τα δικά μας .wav files και να τα πειράξουμε αμέσως. Είναι ιδανικό για μουσικούς της Hip –Hop ή Ηλεκτρονικής μουσικής.

Είναι πολύ απλό στη χρήση και ο χρήστης εκτός των άλλων μπορεί να κάνει sequence μια MIDI συσκευή με ένα pattern editor και να εξάγει το αποτέλεσμα σε .wav αρχείο ή midi αρχείο. Έχει έναν ολοκληρωμένο mixer και υποστηρίζει πρωτόκολλο JACK και έτσι μπορούμε να τρέξουμε το audio κατευθείαν από το Ardour για παράδειγμα. Επίσης υπάρχουν εκδόσεις για Windows αλλά και OSX.

http://www.hydrogen-music.org/

http://www.rosegardenmusic.com/

Εικ 1.8.2 Hydrogen Workspace

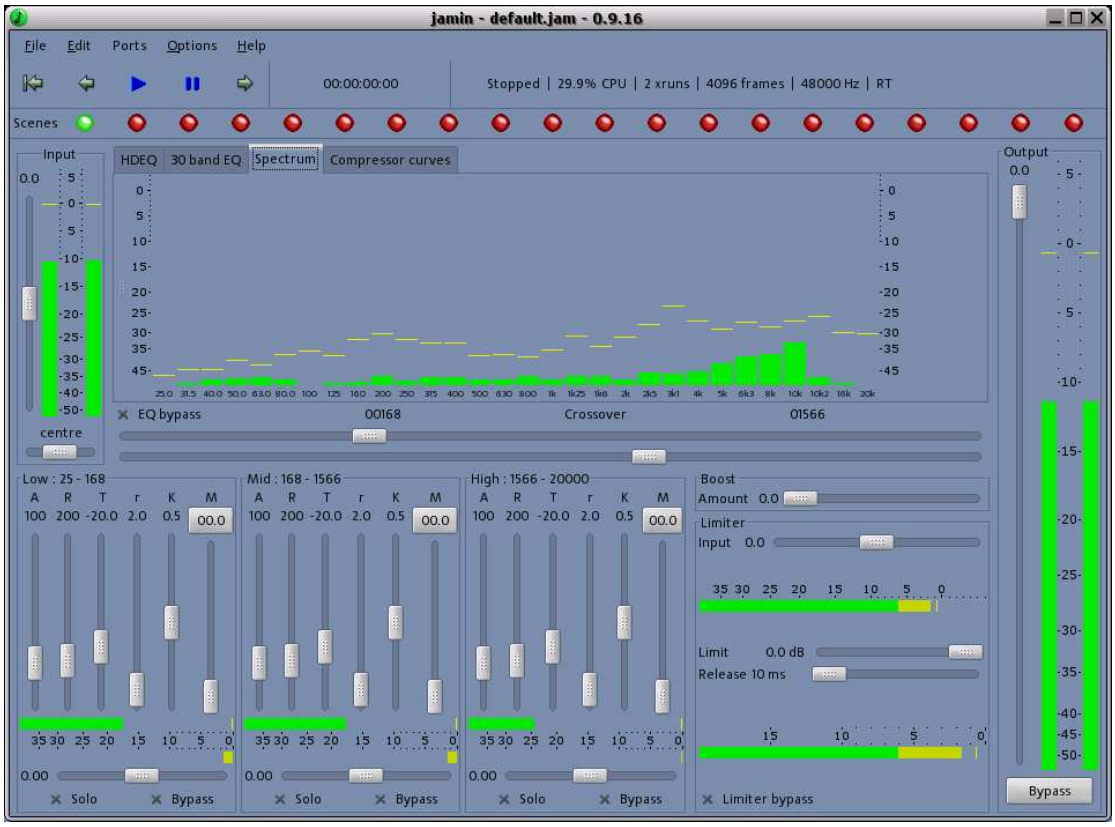
1.9 Mastering

1.9.1 JAMin.

Δεν θα μπορούσε να λείπει και ένα πρόγραμμα mastering από το Linux. Το JAMin είναι μία ολοκληρωμένη σουίτα audio mastering για παράδειγμα όπως Το T-Racks. Έχει ότι περιμένει κανείς από ένα πρόγραμμα για mastering. Spectrum analyzer, loudness maximizer, limiters κοκ. Υποστηρίζει πρωτόκολλο JACK και έτσι μπορούμε να τρέξουμε το audio output κατευθείαν από το Ardour χωρίς να κάνουμε mixdown σε stereo .wav αρχείο.

Για πληροφορίες : http://jamin.sourceforge.net/en/about.html

Εικ .1.9.1 Το JAMin σε συνεργασία με το Ardour.



Εικ 1..2 Το spectrum analyzer του JAMin

1.10 Συμπεράσματα.

Το λειτουργικό σύστημα Linux χρόνο με τον χρόνο αποκτά προγράμματα τα οποία είναι πλήρως επαγγελματικά και εύκολα στην χρήση. Η κοινότητα μεγαλώνει συνεχώς και υπάρχουν ραγδαίες εξελίξεις χρόνο με τον χρόνο. Τα παραπάνω προγράμματα είναι ενδεικτικά ενός συστήματος που μπορεί να κάνει ολοκληρωμένη επεξεργασία ήχου με opensource προγράμματα . Σαφώς και υπάρχουν πάρα πολλά προγράμματα για audio επεξεργασία σε Linux και καλό θα ήταν κάποιος που θέλει να ασχοληθεί να επισκεφθεί τις παρακάτω διευθύνσεις.

Σημαντικές διευθύνσεις :

http://linux-sound.org/one-page.html

http://linux-sound.org/

http://www.linuxproav.com

http://dynebolic.org/

http://www.linuxaudio.org/

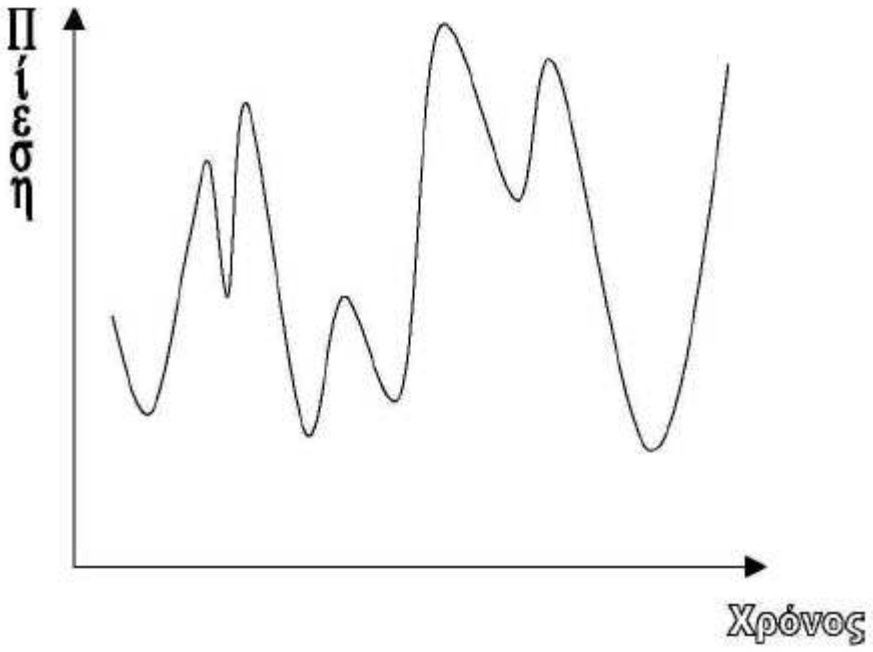
ΚΕΦΑΛΑΙΟ 2

ΗΧΗΤΙΚΑ ΣΗΜΑΤΑ

2.1 Τι είναι ήχος;

Από φυσική άποψη ένας ήχος παράγεται από μεταβολές της πίεσης που μεταδίδονται σε ένα μέσο που μπορεί να συμπιεστεί. Παρόλο που ο ακριβής μηχανισμός διαφέρει από περίπτωση σε περίπτωση οι βασικές αρχές της μετάδοσης και παραγωγής ήχων παραμένουν οι ίδιες. Πιο συγκεκριμένα, τα μόρια όλων των φυσικών σωμάτων προτιμούν να κρατούν σταθερές αποστάσεις από όλα τα γειτονικά τους μόρια. Κατά συνέπεια όταν για οποιονδήποτε λόγο τα μόρια σε μία περιοχή συμπιεστούν και επομένως οι αποστάσεις μεταξύ τους μικρύνουν, τα μόρια της περιοχής επιδιώκουν να επανέλθουν στην αρχική τους κατάσταση συμπιέζοντας με τη σειρά τους τα μόρια των γειτονικών τους περιοχών κ.ο.κ. Με αυτό τον τρόπο δημιουργούνται σε ένα μέσο μεταβολές πίεσης που μεταδίδονται με μία ορισμένη ταχύτητα. Οι συγκεκριμένες μεταβολές αποτελούν ένα ηχητικό κύμα. Οταν οι μεταβολές αυτές φτάσουν στο αυτί μας θέτουν ένα ολόκληρο μηχανισμο από μεμβράνες και οστά σε ταλάντωση και διεγείρουν κατάλληλα ορισμένα νευρικά κύτταρα με αποτέλεσμα το αίσθημα της ακοής. Συνοψίζοντας, θεωρούμε ότι σώματα σε ταλάντωση και μεταβολές πίεσης που μεταδίδονται με μία ορισμένη ταχύτητα αποτελούν την προέλευση των ήχων. Για παράδειγμα, αν χτυπήσουμε τη χορδή μίας κιθάρας, τότε η χορδή θα αρχίσει να ταλαντώνεται. Κάθε φορά που η χορδή κινείται προς τα πάνω συμπιέζει τα μόρια του αέρα που βρίσκονται πάνω της και τα παραγώμενα ηχητικά κύματα μεταδιδονται μέσω του αέρα μέχρι να συναντήσουν το τύμπανο του αυτιού μας. Οι αρχές λειτουργίας της χορδής είναι κοινές και για άλλες πηγές ήχων. Μια παράδειγμα όταν ένας σαξοφωνίστας φυσάει στο σαξόφωνο του, θέτει σε παλινδρομική κίνηση μία στήλη αέρα μέσα στο όργανο του. Ο σαξοφωνίστας μπορεί να μεταβάλλει την παλινδρόμηση του άερα ανοίγοντας και κλείνοντας τρύπες που βρίσκονται κατά μήκος του σαξοφώνου.

Η ταλάντωση που εκτελούν τα μόρια ενός σώματος κατά την παραγωγή ενός ήχου μπορεί να είναι αρκετά πολύπλοκη. Για παράδειγμα, η ανθρώπινη φωνή παράγεται από ένα συνδυασμό ταλαντώσεων των φωνητικών χορδών που προκαλούν την ταλάντωση του αέρα που βρίσκεται στους πνεύμονες, στο λαιμό, στο στόμα και στα ιγμόρια. Ο ήχος μίας κιθάρας ή ενός βιολιού προέρχεται από την ταλάντωση μίας χορδής του οργάνου που θέτει σε παλινδρομική κίνηση τον αέρα που βρίσκεται στο αντηχείο του οργάνου. Η χροιά του ήχου που παράγουν τα περισσότερα μουσικά όργανα εξαρτάται από το σχήμα και τις υπόλοιπες φυσικές ιδιότητες του αντηχείου τους.

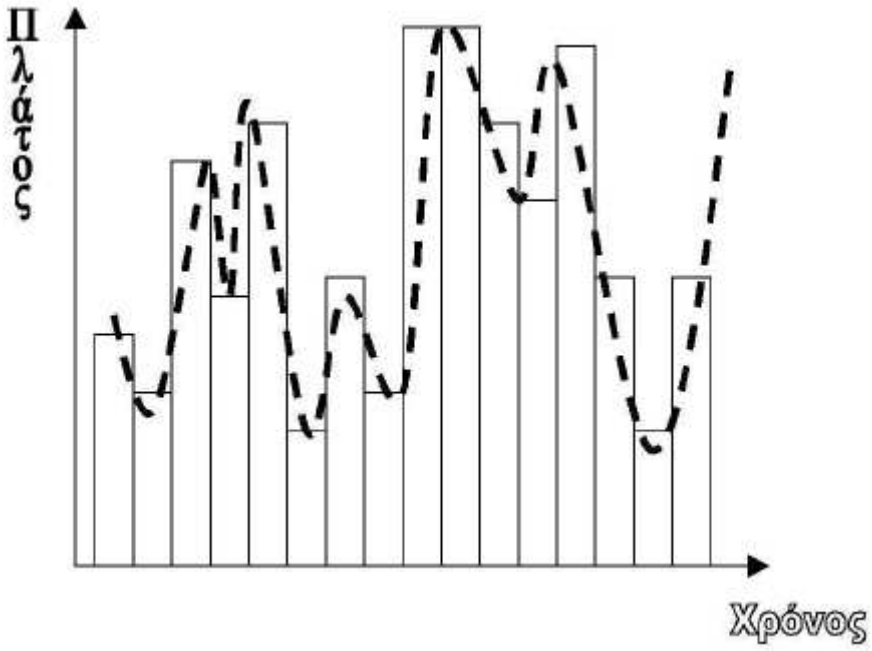


2.1 Μεταβολές πίεσης στο τύμπανο

Κάθε ήχος λοιπόν κάποια στιγμή αποτελείται από ένα σύνολο μεταβολών πίεσης. Αν είχαμε κάποιο τρόπο σχεδίασης των συγκεκριμένων μεταβολών της πίεσης που διεγείορυν το τύμπανο του ακροατή θα καταλήγαμε σε ένα διάγραμμα παρόμοιο με αυτό της Εικόνας 2.1. Ενας αρκετά απλός τρόπος για να δημιουργήσουμε ένα τέτοιο διάγραμμα θα ήταν να προσαρμόσουμε ένα κομμάτι μεταλλικό σύρμα σε ένα λεπτό φύλλο χαρτί και να τοποθετήσουμε ένα μαγνήτη σχετικά κοντά. Οταν αρχίσουμε να μιλάμε μπροστά στο χαρτί τότε οι μεταβολές της πίεσης που οφείλονται στη φωνή μας θα αναγκάζουν το χαρτί να κινείται μπρος ή πίσω, μεταβάλλοντας παράλληλα και τη θέση του μεταλλικού σύρματος ως προς το μαγνήτη. Η κίνηση αυτή του σύρματος θα δημιουργήσει ένα μεταβαλλόμενο ηλεκτρικό ρεύμα το οποίο θα είναι ανάλογο με τις μεταβολές της πίεσης του αέρα που προκαλεί η φωνή μας. Ο απλός αυτός μικροφωνικός μηχανισμός δουλεύει και αντίστροφα. Ενα μεταβαλλόμενο ηλεκτρικό ρεύμα μπορεί να μεταβάλλει το μαγνητικό πεδίο του μαγνήτη μεταβάλλοντας και τη θέση του σύρματος ως προς το μαγνήτη και προκαλώντας κατά συνέπεια την κίνηση του χαρτιού. Οι αρχές λειτουργίας της συσκευής που περιγράψαμε αντιστοιχούν στον τρόπο με τον οποίο ηχητικά σήματα μπορούν να μετατραπούν σε ηλεκτρικά σήματα μέχρι τις μέρες μας.

2.2 Ψηφιακός Ηχος

Όπως είναι δυνατό να μετατρέψουμε τις μεταβολές πίεσης που προέρχονται από έναν ήχο σε αναλογικό ηλεκτρικό σήμα με ανάλογο τρόπο είμαστε σε θέση να μεταρέψουμε ένα μεταβαλλόμενο ηλεκτρικό σήμα σε μία σειρά από διακριτές τιμές και αντίστροφα. Επειδή ομως η αναλογική και διακριτή μορφή ενός ήχου είναι διαφορετικές, πάντα χάνουμε πληροφορία κατά την εφαρμογής μίας τέτοιας μετατροπής. Ολη η προσπάθεια σε μία τέτοια περίπτωση είναι να καταλάβουμε τι είδους πληροφορία χάνουμε ώστε να επιλέξουμε τι πληροφορία θα πρέπει να κρατήσουμε κατά την εφαρμογή μίας τέτοιας μετατροπής.



Εικ 2 .2 Ψηφιακή και αναλογική αναπαράσταση ηχητικού σήματος

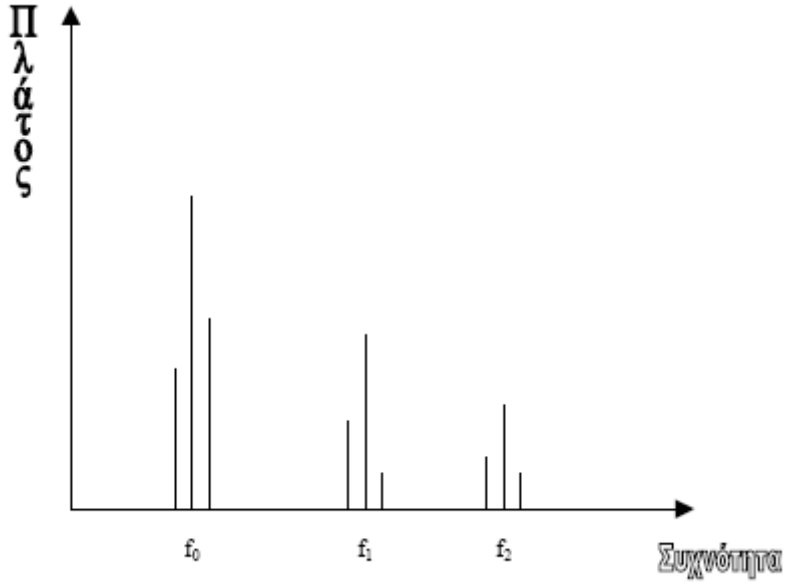
Σε ψηφιακή μορφή ένα ηχητικό σήμα παριστάνεται από μια σειρά από νούμερα (τα οποία ονομάζουμε δείγματα) τα οποία αντιστοιχούν στην πίεση του αέρα ή στην ηλεκτρική τάση σε διαδοχικές χρονικές στιγμές. Η δειγματοληψία ενος αναλογικού σήματος αντιμετωπίζει δύο κύρια προβλήματα που περιγράφονται στην Εικόνα 3.2. Η εικόνα περιγράφει δυο αναπαραστάσεις ενός ηχητικού σήματος. Η διακεκομμένη μαύρη γραμμή περιγράφει την αναλογική μορφή του σήματος. Η ψηφιακή μορφή του σήματος παριστάνεται σαν μία ακολουθία από ορθογώνια. Το πρώτο πρόβλημα προέρχεται από το γεγονός ότι κάθε τιμή του σήματος στη ψηφιακή του μορφή έχει μια ορισμένη χρονική διάρκεια (κάθε ορθογώνιο έχει ένα ορισμένο πλάτος). Το δεύτερο πρόβλημα είναι ότι οι τιμές του σήματος στη διακριτή του μορφή είναι ορισμένες, με άλλα λόγια υπάρχει ένα πεπερασμένο σύνολο από ύψη που μπορεί να έχει ένα ορθογώνιο. Κατά συνέπεια το ύψος του κάθε ορθογωνίου δεν ταυτίζεται απόλυτα με το ύψος του αναλογικού σήματος κάθε χρονική στιγμή. Τα δύο αυτά προβλήματα είναι οι βασικές πηγές απωλειών κατά τη μετατροπή ενός ήχου από

αναλογική σε ψηφιακή μορφή. Τα προβλήματα αυτά μπορούν να αντιμετωπιστούν λιγότερο ή περισσότερο αποτελεσματικά με τη μεταβολή της ψηφιακής αναπαράστασης ενός ήχου αλλά δεν μπορούν ποτέ να εξαλειφθούν. Το πρόβλημα δεν είναι λοιπόν πως μπορούμε να εξαλείψουμε τις απώλειες που αναφέραμε, αλλά δεδομένης της τεχνολογίας που διαθέτουμε πως μπορούμε να κάνουμε ανεκτές τις συγκεκριμένες απώλειες.

Το κύριο πλεονέκτημα της ψηφιακής επεξεργασίας ήχων προέρχεται από το γεγονός ότι η αντιγραφή ψηφιακών σημάτων δεν προσθέτει θόρυβο στο σήμα μας. Αντίθετα η αναλογική επεξεργασία ήχων προσθέτει θόρυβο κατά τη διάρκεια της διαδικασίας αντιγραφής ενός σήματος από το ένα μέσο στο άλλο.

2.3 Συχνότητα & Υψος

Αν θέλουμε να περιγράψουμε τον ήχο που παράγουν δύο πνευστά μουσικά όργανα όπως η τρομπέτα και η τούμπα θα παρατηρήσουμε ότι παρόλο που και τα δύο είναι παρόμοια όργανα, η τρομπέτα παράγει πιο ψηλούς ήχος από την τούμπα. Το ύψος του ήχου είναι ένα υποκειμενικό χαρακτηριστικό που σχετίζεται με ένα αντικειμενικό χαρακτηριστικό, τη συχνότητα. Οπως έχουμε ήδη αναφέρει η συχνότητα ενός συνημιτονικού ηχητικού σήματος αντιστοιχεί στον αριθμό των κύκλων που εκτελεί η συνάρτηση ανά δευτερόλεπτο και μετριέται σε κύκλους ανά δευτερόλεπτο ή Hertz (Hz). Ο συγκεκριμένος ορισμός της συχνότητας στηρίζεται στο γεγονός ότι το συνημίτονο είναι μία περιοδική συνάρτηση. Στην πράξη πολλοί λίγοι ήχοι στη φύση είναι περιοδικοί επομένως ο ορισμός της συχνότητας που δώσαμε δε θα μπορούσε να εφαρμοστεί για την πλειονότητα των ηχητικών σημάτων αν δεν ίσχυε ότι κάθε ήχος μπορεί να αναλυθεί σε ένα άρθροισμα κατάλληλων συνημιτονικών σημάτων. Το γεγονός ότι υπάρχει μία τέτοια ανάλυση μας επιτρέπει να αναφερόμαστε στο φάσμα συχνοτήτων (frequency spectrum) ενός ηχητικού σήματος που αποτελεί ένα διάγραμμα του πλάτος που έχει κάθε συχνότητα που περιέχεται στο σήμα μας. Για παράδειγμα, η Εικόνα 2.3 περιγράφει τη μορφή που έχει το φάσμα συχνοτήτων για ένα ηχητικό σήμα. Για να γίνει αντιληπτός ένα ήχος από το ανθρώπινο αυτί θα πρέπει να έχει συχνότητες μεταξύ 20 και 20.000 Hz.



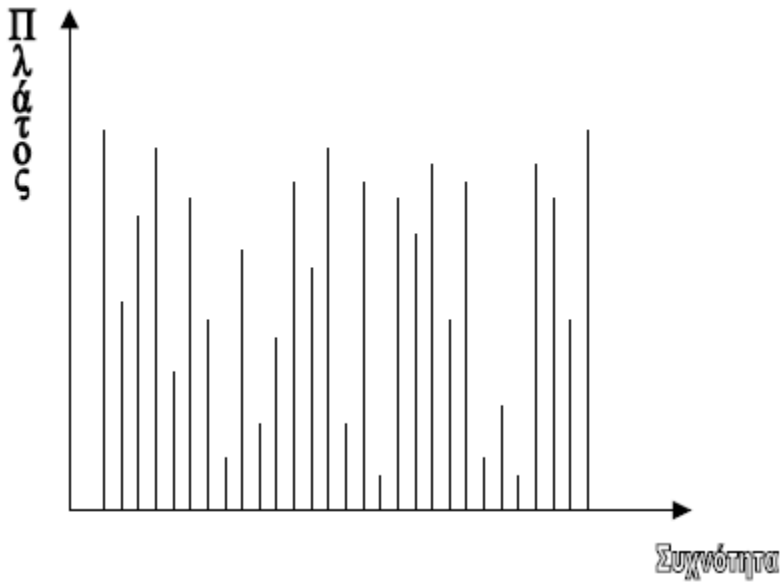
Εικ 2.3 Παράδειγμα φάσματος συχνοτήτων για ένα ηχητικό σήμα.

Κάθε περιοδικός ήχος μπορεί να αναλυθεί σε ένα άρθροισμα συνημιτονικών συναρτήσεων με διάφορα πλάτη και φάσεις και με συχνότητες οι οποίες είναι ακέραια πολλαπλάσια μίας θεμελιώδους συχνότητας (fundamental frequency). Οι συχνότητες αυτές ονομάζονται αρμονικές (harmonic frequencies) και το πλήθος και το σχετικό τους πλάτος είναι σε μεγάλο βαθμό υπεύθυνες για το υποκειμενικό αίσθημα της χροιάς ενός ήχου. Σε αρκετές περιπτώσεις η αρμονική συχνότητα με το μεγαλύτερο πλάτος καθορίζει και το ύψος του ήχου και συνήθως πάλι η συχνότητα αυτή είναι η θεμελιώδης συχνότητα.

Η αντιστοιχία μεταξύ ύψους και συχνότητας δεν ισχύει πάντα. Για παράδειγμα το ύψος αρκετά δυνατών ήχων συνήθως είναι χαμηλότερο από ότι το ύψος ασθενών ήχων με την ίδια συχνότητα.

2.4 Θόρυβος

Αν ένα ηχητικό σήμα έχει ένα φάσμα συχνοτήτων παρόμοιο με αυτό της Εικόνας 3.3 τότε μπορούμε να διακρίνουμε τις συχνότητες με το μεγαλύτερο πλάτος και να τις χρησιμοποιήσουμε για να χαρακτηρίσουμε το ηχητικό σήμα. Τι συμβαίνει όμως στην περίπτωση που το φάσμα συχνοτήτων έχει τη μορφή της Εικόνας 3.4; Στην περίπτωση αυτή δεν υπάρχει κάποια συχνότητα ή σύνολο συχνοτήτων που μπορούν να χρησιμοποιηθούν για το χαρακτηρισμό του ηχητικού σήματος. Ενα ηχητικό σήμα που διαθέτει ένα πλήρες σύνολο συχνοτήτων ονομάζεται θόρυβος (noise). Η τεχνική χρήση του όρου θόρυβος είναι αρκετά διαφορετική από την καθημερινή χρήση του όρου. Ενας δυνατός οξύς ήχος που στην καθημερινή ζωή θα τον χαρακτηρίζαμε ως θόρυβο έχει ένα αρκετά καθορισμένο φάσμα συχνοτήτων και επομένως δεν αποτελεί θόρυβο με την τεχνική σημασία του όρου.



Εικόνα 2.4: Παράδειγμα φάσματος συχνοτήτων για σήμα θορύβου.

Οσο και αν φαίνεται περίεργο ο θόρυβος βρίσκει αρκετές εφαρμογές στη μουσική. Για παράδειγμα, τα περισσότερα κρουστά όργανα παράγουν ήχους με φάσματα συχνοτήτων παρόμοια με αυτό της Εικόνας 2.4.

2.5 Ένταση

Η υποκειμενική αίσθηση της έντασης ενός ήχου είναι συνδεδεμένη με την ισχύ του ηχητικού σήματος που διεγείρει το αυτί μας. Η κύρια αιτία για την οποία ήχοι έχουν διαφορετικές εντάσεις είναι ότι πιέζουν με διαφορετική δύναμη το τύμπανο του αυτιού μας. Σε μία τέτοια περίπτωση, ένας φυσικός θα έλεγε ότι τα ηχητικά κύματα έχουν διαφορετική ισχύ. Οσο πιο μεγάλη ισχύ έχουν τα ηχητικά κύματα τόσο μεγαλύτερη πίεση εξασκούν στο μηχανισμό του αυτιού μας.
Τα ηλεκτρικά σήματα έχουν και αυτά ισχύ, η στιγμιαία τιμή της οποίας είναι ανάλογη του τεραγώνου των διαφορών τάσης που προκαλούν στο μέσο στο οποίο διαδίδονται. Η ολική ισχύ ενός ηλεκτρικού σήματος προέρχεται από την άρθροιση όλων των στιγμιαίων τιμών της ισχύος στη μονάδα του χρόνου. Η ολική ισχύς ενός ηχητικού σήματος σε ψηφιακή μορφή είναι ανάλογη του αρθροίσματος των τετραγώνων των διακριτών τιμών από τις οποίες συντίθεται. Η μέση τιμή της ισχύος στην περίπτωση αυτή είναι ανάλογη της μέσης τιμής του αρθροίσματος των τετραγώνων των διακριτών τιμών από τις οποίες συντίθεται το σήμα μας.

Στις εφαρμογές πολυμέσων δε μας ενδιαφέρει τόσο η απόλυτη τιμή της ισχύος ενός ηχητικού σήματος όσο η σχετική τιμή της ισχύος μεταξύ δύο ηχητικών σημάτων. Η

σχετική ισχύς μετριέται σε bels ή πιο συχνά σε decibels (dB) (1 dB = 0.1bel). Για να συγκρίνουμε την ισχύ δύο ηχητικών σημάτων υπολογίζουμε το λόγο των ισχύων τους. Ο λογάριθμος ως προς 10 του λόγου αυτού είναι η διαφορά των δύο ισχύων σε bels. Αν πολλαπλασιάσουμε το συγκεκριμένο λογάριθμο επί 10 παίρνουμε την διαφορά σε dB. Μαθηματικά έχουμε ότι:

$$(\text{Διαφορά Έντασης})\text{dB} = 10 \times \log10\ (\text{Ισχύς1}/\text{Ισχύς2})$$

Κατά σύμβαση για τη μέτρηση της έντασης ενός ήχου χρησιμοποιείται η διαφορά της έντασης του σε dB από ένα ηχητικό σήμα αναφοράς. Το σήμα αναφοράς είναι προσεγγιστικά ο πιο ασθενής ήχος συχνότητας 1000 Hz που μπορεί να γίνει αντιληπτός από το ανθρώπινο αυτί. Με βάση αυτή την κλίμακα ο πιο δυνατός ήχος που μπορούμε να ακούσουμε είναι περίπου 120 dB δυνατότερος (δηλ. ένα εκατομμύριο επί ένα εκατομμύριο φορές πιο δυνατός σε απόλυτη ένταση από τον ήχο αναφοράς) και αντιστοιχεί στον ήχο του κινητήρα ενός αεροπλάνου. Η λογαριθμική βάση υπολογισμού των dB έχει σαν αποτέλεσμα δύο ήχοι καθένας από τους οποίους έχει π.χ. ένταση 60 dB να έχουν ένταση μόλις 63 dB όταν ακούγονται συγχρόνως. Μια αύξηση της έντασης ενός ήχου κατά 1.000.000 φορές αντιστοιχεί σε μία αύξηση της έντασης του κατά 60 dB.

Πέρα από την μέτρηση της έντασης ενός ήχου η κλιμακα σε dB χρησιμοποιείται στη μέτρηση της απώλειας έντασης που έχουν ορισμένα ηλεκτρονικά κυκλώματα ή αλγόριθμοι επεξεργασίας ψηφιακού ήχου. Για παράδειγμα διαφορετικοί ήχοι με την ίδια ισχύ μπορεί να προκύψουν με σχετική διαφορά 10 dB στην ισχύ τους μετά από μια τέτοια επεξεργασία. Επιπλέον η κλίμακα σε dB χρησιμοποιείται για τη μέτρηση του θορύβου ή της παραμόρφωσης που εισάγεται σε ένα σήμα.

Για παράδειγμα ένας μουσικός οπτικός δίσκος αποθηκεύει τα δείγματα του ήχου σε ακεραίους μήκους 16-bit. Το εύρος των αριθμών που μπορούν να αποθηκευτούν σε ακέραιους αυτής της μορφής είναι από –32.768 ως +32.767. Κατά τη διάρκεια της αποθήκευσης οι τιμές του πρωτότυπου σήματος έχουν στρογγυλευτεί στον πλησιέστερο ακέραιο άρα το μέγιστο λάθος που έχει γίνει κατά την αποθήκευση είναι 0.5, το οποίο είναι 2^{-16} φορές μεγαλύτερο από τη μέγιστη τιμή που μπορεί να λάβει το σήμα μας. Επειδή η ισχύς του σήματος είναι ανάλογη του τετραγώνου του πλάτους του, το λάθος στην περίπτωση μας θα έχει 2^{-32} φορές μεγαλύτερη ισχύ από την ισχύ του μεγαλύτερου σήματος μας. Ο λόγος της ισχύος μεταξύ του δείγματος με τη μέγιστη τιμή και του λάθους (θορύβου) στην περίπτωση μας είναι 2^{32} προς 1, ή$10 \times \log10(2^{32}) \approx 96.3$ dB.

Υπάρχουν δύο κύριοι λόγοι για τους οποίους η μέτρηση σε dB αντιστοιχεί στην ανθρώπινη αντίληψη της έντασης ενός ήχου (δηλ. την υποκειμενική ένταση (loudness)). Ο πρώτος λόγος είναι ότι η ανθρώπινη ακοή είναι κατά προσέγγιση λογαριθμική δηλ. η διαφορά μεταξύ των υποκειμενικών εντάσεων δύο ήχων δεν εξαρτάται από τη απόλυτη τιμή της διαφοράς αυτής αλλά από το λόγο των ισχύων τους. Παρόλο που δεν είναι απόλυτα σωστό μπορούμε να θεωρήσουμε ότι το 1 dB αντιστοιχεί στην ελάχιστη διαφορά έντασης που μπορεί να γίνει αντιληπτή μεταξύ

δύο ήχων.

Ο δεύτερος λόγος προέρχεται από το γεγονός ότι η υποκειμενική ένταση ενός σύνθετου ήχου εξαρτάται από τη σχετική τιμή των συνιστωσών του. Πιο συγκεκριμένα, η ανθρώπινη ακοή εμφανίζει μια συγκεκριμένη μορφή ψευδαίσθησης που είναι γνωστή ως συγκάλυψη (masking). Αν ένας ήχος αποτελείται από δύο συνιστώσες και ισχύει ότι η ένταση της μίας από αυτές είναι αρκετά μεγαλύτερη από την άλλη, τότε σε αρκετές περιπτώσεις η ασθενέστερη συνιστώσα δε γίνεται αντιληπτή. Ουσιαστικά, η ανθρώπινη ακοή προσαρμόζεται στο δυνατότερο ήχο ενώ ο ασθενέστερος ήχος γίνεται αντιληπτός σαν περισσότερο ασθενής από ότι πραγματικά είναι. Το φαινόμενο γίνεται αρκετά πιο έντονο όταν οι δύο συνιστώσες έχουν κοντινά ύψη. Η συγκάλυψη βρίσκει αρκετές εφαρμογές στη συμπίεση ηχητικών σημάτων. Πιο συγκεκριμένα, αρκετοί μηχανισμοί συμπίεσης ανιχνεύουν και εξαλείφουν στο συμπιεσμένο σήμα τις συνιστώσες εκείνες που θα συγκαλυφθούν κατά την ακρόαση του ήχου. Το σήμα που προκύπτει είναι απλούστερο και περισσότερο εύκολο να συμπιεστεί.

Η υποκειμενική αντίληψη της έντασης ενός ήχου εξαρτάται και από άλλους παράγοντες όπως λ.χ. το ύψος του ήχου. Πιο συγκεκριμένα, το αυτί μας είναι πιο ευαίσθητο σε ορισμένα εύρη συχνοτήτων από ότι σε άλλα. Η πιο ευαίσθητη περιοχή για το αυτί μας βρίσκεται μεταξύ 2700-3200 Hz με την ευαισθησία να μειώνεται βαθμιαία σε κάθε πλευρά αυτής της περιοχής. Κατά συνέπεια ένας συνημιτονικός ήχητικό σήμα με συχνότητα 3000 Hz και με μία ορισμένη ένταση θα ακούγεται πιο ισχυρός από ένα σήμα της ίδιας έντασης αλλά με συχνότητα 200 ή 8000 Hz.

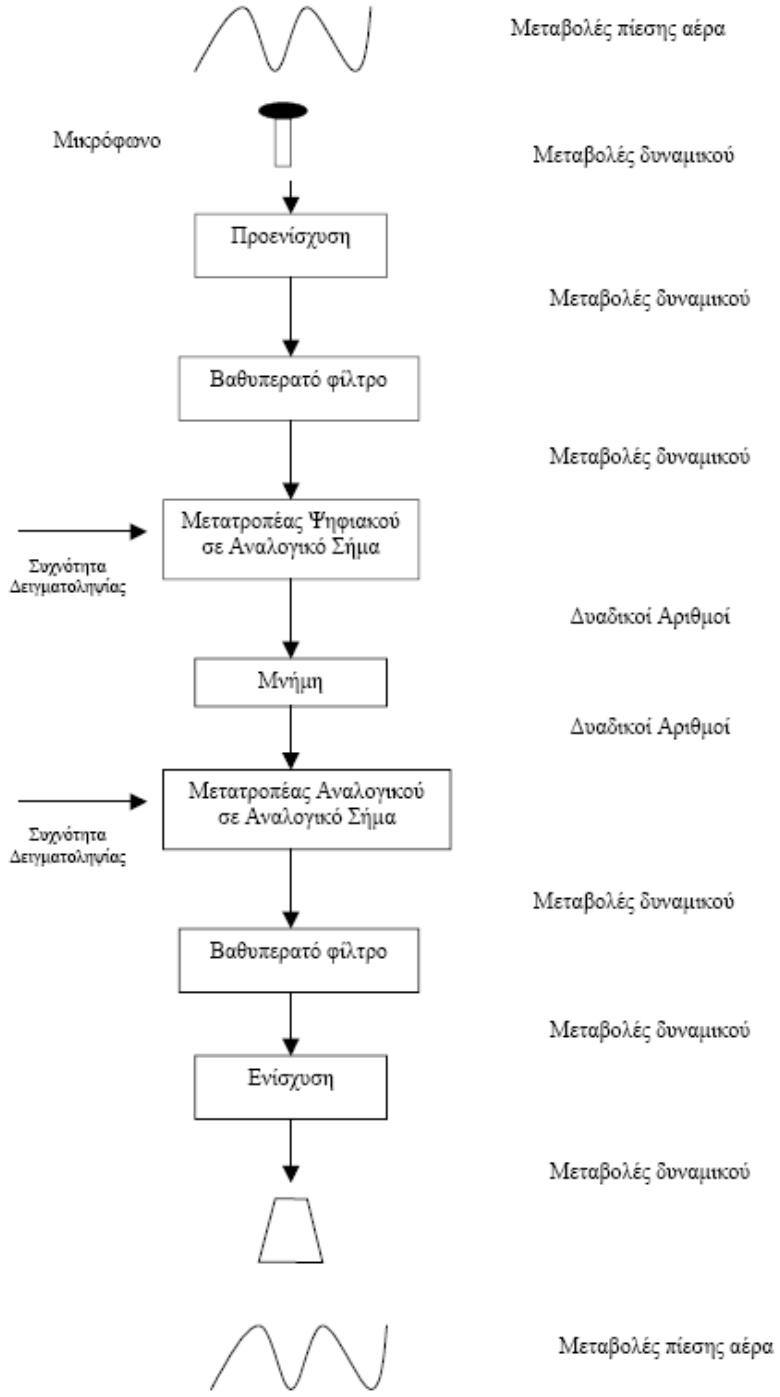
Επιπλέον το αυτί μας είναι λιγότερο ευαίσθητο σε σύνθετους παρά σε απλούς τόνους. Πιο συγκεκριμένα, δεν αντιλαμβανόμαστε αρκετά εύκολα θορύβους που περιέχουν αρκετές υψηλές συχνότητες. Ορισμένες ψηφιακές τεχνικές, όπως το dithering, μετασχηματίζει ορισμένους τύπους λαθών κατά την επεξεργασία ενός ηχητικού σήματος σε θόρυβο υψηλής συχνότητας που γίνεται δύσκολα αντιληπτός.

2.6 Στάδια Επεξεργασίας Ψηφιακού Ήχου

Ενα σύστημα ψηφιακού ήχου για να ενσωματωθεί σε ένα σύστημα πολυμέσων θα πρέπει να προσφέρει τις δυνατότητες της καταγραφής, της αποθήκευσης και της απόδοσης (playback) ηχητικών σημάτων.

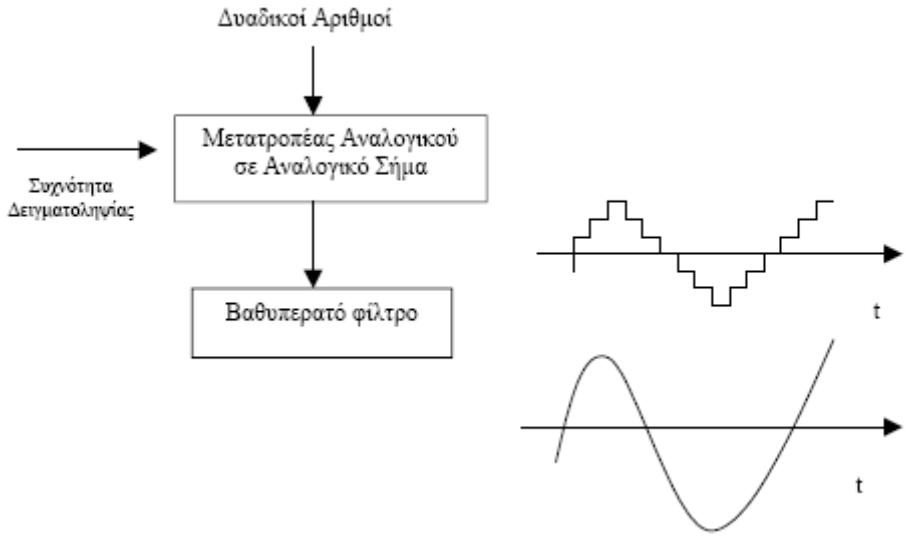
Η καταγραφή ενός ηχητικού σήμαρτος γίνεται με τη βοήθεια ενός μικροφώνου που μετατρέπει τα ηχητικά σήματα σε διαφορές δυναμικού μέσω του μηχανισμού που περιγράφηκε στην § 2.1. Επειδή οι διαφορές αυτές έχουν μικρό εύρος για τη διευκόλυνση της επεξεργασίας συνήθως παρεμβάλλεται ένα στάδιο προενίσχυσης του σήματος έτσι ώστε να προκύψει μία περισσότερο ενισχυμένη μορφή του ηλεκτρικού σήματος. Το ηλεκτρικό σήμα μετά το στάδιο της προενίσχυσης τροφοδοτείται σε ένα βαθυπερατό φίλτρο. Σκοπός του φίλτρου αυτού είναι η αποκοπή των συχνοτήτων που υπάρχουν στο σήμα μας και που είναι μεγαλύτερες του μισού της συχνότητας δειγματοληψίας έτσι ώστε να αποφύγουμε φαινόμενα αναδίπλωσης κατά τη δειγματοληψία του ηλεκτρικού σήματος. Στη συνέχεια το

στάδιο της δειγματοληψίας μετατρέπει το αναλογικό ηλεκρικό σήμα στην είσοδο του σε μια ακολουθία δυαδικών αριθμών η οποία και αποθηκεύεται στη μνήμη του υπολογιστικού συστήματος. Στο σημείο αυτό το ηχητικό σήμα έχει μετασχηματιστεί από αναλογική σε ψηφιακή μορφή.



Εικόνα 2.6: Στάδια Επεξεργασίας Συστήματος Ψηφιακού Ήχου.

Η απόδοση ενός ψηφιακού ηχητικού σήματος ακολουθεί μία αντίστροφη ακολουθία βημάτων από αυτή της καταγραφής. Πιο συγκεκριμένα, το σήμα τροφοδοτείται από τη μνήμη του υπολογιστικού συστήματο σε ένα μετατροπέα ψηφιακού σε αναλογικό σήματος. Ανάλογα με τον τρόπο λειτουργίας του μετατροπέα και τη συχνότητα δειγματοληψίας που χρησιμοποιείται ο μετατροπέας παράγει ένα αναλογικό ηλεκτρικό σήμα που αποτελεί μια προσεγγιστική μορφή του σήματος εισόδου. Η εικόνα 2.6.1 μας δίνει μια ιδέα της μορφής που έχει το σήμα στην έξοδο του μετατροπέα. Το βαθυπερατό φίλτρο που ακολουθεί σκοπό έχει να εξομαλύνει τη μορφή του ηλεκτρικού σήματος το οποίο ενισχύεται και στη συνέχεια τροφοδοτείται στην είσοδο ενός ηχείου το οποίο και αποδίδει το σήμα.



Εικόνα 2.6.1: Εξομάλυνση αναλογικού σήματος κατά το στάδιο της απόδοσης.

Τα προβλήματα που αντιμετωπίζει η ψηφιακή επεξεργασία ήχων εντοπίζονται στην επιλογή μίας κατάλληλης συχνότητας δειγματοληψίας τόσο κατά την καταγραφή όσο και κατά την απόδοση ενός ήχου και στην εισαγωγή θορύβου λόγω της πεπερασμένης διακριτικότητα της ψηφιακής αναπαράστασης.

2.6.1 Προβλήματα Δειγματοληψίας

Η χρησιμοποίηση ενός βαθυπερατού φίλτρου πριν από τη δειγματοληψία του αναλογικού σήματος σκοπό έχει να αποκόψει πλήρως συχνότητες που μπορούν να

προκαλέσουν αναδίπλωση κατά τη δειγματοληψία. Ξέρουμε ότι τέτοιες συχνότητες είναι όσες έχουν τιμή μεγαλύτερη του ½ της συχνότητας Nyquist. Η χρήση τέτοιων φίλτρων εισάγει δύο νέα προβλήματα. Το πρώτο πρόβλημα προέρχεται από το γεργονός ότι δεν υπάρχει ένα τέλειο βαθυπερατό φίλτρο. Με άλλα λόγια δεν υπάρχει ένα φίλτρο που να μπορεί να αποκόψει τέλεια συχνότητες πάνω από ένα επιθυμητό όριο.

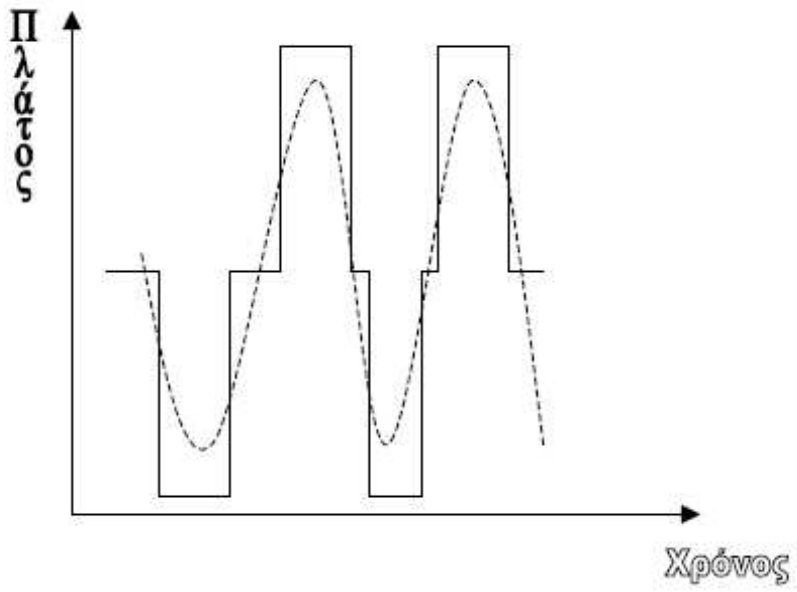
Το δεύτερο πρόβλημα προέρχεται από το γεγονός ότι η χρήση βαθυπερατών φίλτρων εισάγει παραμορφώσεις φάσης στο σήμα στο οποίο εφαρμόζεται. Γενικά η επίδραση που έχει η φάση στον τρόπο με τον οποίο αντιλαμβανόμαστε ένα ήχο είναι αρκετά σύνθετη. Το αυτί μας δε μπορεί να αντιληφθεί διαφορές μεταξύ ηχητικών σημάτων που έχουν μια απόλυτη διαφορά φάσης. Παρόλα αυτά οι μεταβολές στη φάση ενός σήματος αποκτούν ιδιαίτερη σημασία σε περίπτωση φιλτραρίσματος ενός ήχου. Πιο συγκεκριμένα, κάθε φίλτρο συνδυάζει σήματα που βρίσκονται εκτός φάσης για να μεταβάλλει τα χαρακτηριστικά τους. Ενα φίλτρο δημιουργεί τέτοιες διαφορές καθυστερώντας το σήμα στην είσοδο του για ένα ορισμένο χρόνο και συνδυάζοντας το σήμα που προκύπτει με την παρούσα είσοδο του για να δημιουργήσει αλληλοαναιρέσεις σε ορισμένες συχνότητες οι οποίες μεταβάλλουν το φάσμα του σήματος εισόδου. Οταν οι συχνότητες οι οποίες φιλτράρονται μεταβάλλονται με το χρόνο τότε το αποτέλεσμα περιγράφεται από το ηχητικό εφφέ που είναι γνωστό σα flanging ή phasing. Επιπλέον έχει παρατηρηθεί ότι συστήματα αναπαραγωγής ήχων τα οποία εισάγουν παραμορφώσεις φάσης στο σήμα εισόδου τους έχουν μια θολή ηχητική εικόνα (imaging) δυσκολεύοντας την (νοητική) τοποθέτηση των ηχητικών πηγών στο χώρο.

2.6.2 Προβλήματα Διακριτικότητας

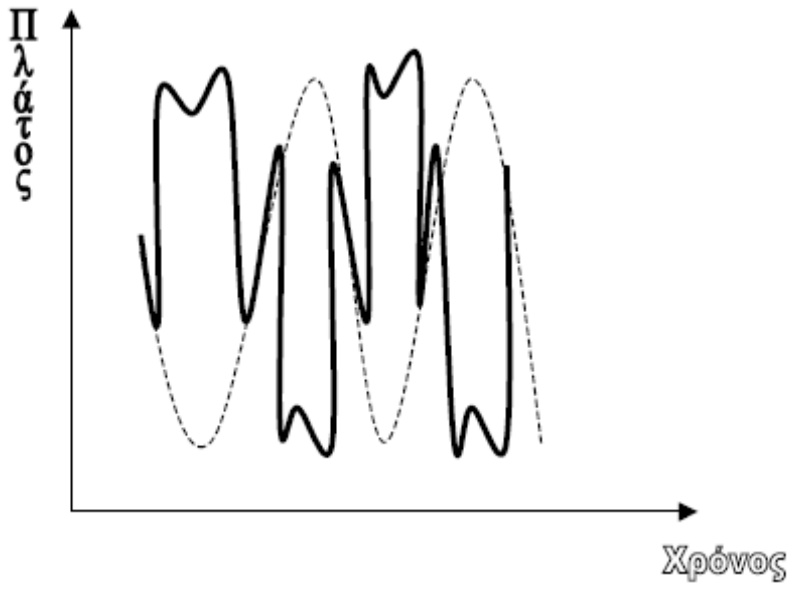
Ενα ακόμα σημαντικό πρόβλημα κατά τη διαδικασία της ηχογράφησης και αναπαραγωγής ψηφιακών ήχων είναι ο θόρυβος που προέρχεται από την πεπερασμένη διακριτικότητα της ψηφιακής αναπαράστασης (δηλ.από τον αριθμό των bits που χρησιμοποιούνται για την αναπαράσταση ενός σήματος). Ο θόρυβος αυτός είναι γνωστός και σα θόρυβος κβαντοποίησης (quantization error ή quantization noise). Πιο συγκεκριμένα, κατά τη σχεδίαση ενός συστήματος ψηφιακού ήχου θα πρέπει να αποφασιστεί ο τρόπος με τον οποίο θα αποθηκεύονται τα δείγματα του ήχου καθώς κάθε υπολογιστικό σύστημα μπορεί να χειριστεί ένα πεπερασμένο σύνολο από αριθμητικές τιμές. Το μήκος της λέξης κλαθε υπολογιστή καθορίζει και το εύρος των τιμών που μπορεί να χειριστεί. Για παράδειγμα, ένα σύστημα που αποθηκεύει ακέραιους σε λέξεις των 8 ή16 bits μπορεί να χειριστεί είτε 256 είτε 65.536 διαφορετικούς ακεραίους. Ο θόρυβος κβαντοποίησης προέρχεται από το γεγονός ότι τα δείγματα του αναλογικού σήματος που θα χρησιμοποιηθούν δεν είναι κατ’ ανάγκη ακέραιοι και συνεπώς η μετατροπή τους σε ψηφιακά δείγματα συνεπάγεται τη στρογγύλευση τους στην πλησιέστερη ακέραια μορφή. Στην περίπτωση αυτή μπορούμε να θεωρήσουμε ότι το ψηφιοποιημένο σήμα αποτελείται από το άρθροισμα του αναλογικού σήματος μας και ενός σήματος θορύβου.

Για να μπορέσουμε να αντιμετωπίσουμε το θόρυβο κβαντοποίησης θα πρέπει να καταλάβουμε πόσο δυνατός είναι σε σύγκριση με το υπόλοιπο σήμα. Ο λόγος σήματος πρός θόρυβο (signal-to-noise ratio ή SNR) μετρά τη σχετική ισχύ του ηχητικού σήματος ως προς το θόρυβο που περιέχεται σε αυτό. Προφανώς ένα σύστημα ψηφιακής επεξεργασίας ήχου θέλουμε να έχει όσο το δυνατόν μεγαλύτερο λόγο σήματος προς θόρυβο. Για παράδειγμα στην §2.5 υπολογίσαμε το μέγιστο λόγο σήματος προς θόρυβο για ένα 16-bit σύστημα επεξεργασίας ίσο με περίπου 96 dB. Σε συστήματα με μικρότερο μήκος λέξης η μέγιστη τιμή του λόγου είναι μικρότερη.

Αν το σήμα μας δεν έχει τη μέγιστη ισχύ τότε ο λόγος σήματος προς θόρυβο θα έχει μικρότερη τιμή από τη μέγιστη τιμή του. Η χειρότερη περίπτωση συμβαίνει όταν το σήμα μας έχει τόσο μικρή ισχύ ώστε να αναγκάζει τις τιμές που προκύπτουν κατά τη δειγματοληψία να αλλάζουν τιμή κατά μία η δύο στάθμες. Για παράδειγμα η Εικόνα 2.6.2 απεικονίζει ένα συνημιτονικό παλμό το πλάτος του οποίου είναι μικρότερο από 1. Στην περίπτωση αυτή οι τιμές του ψηφιακού σήματος θα εναλλάσσονται μεταξύ των τιμών –1, 0 και 1.



Εικόνα 2.6.2: Δειγματοληψία ενός αρκετά ασθενούς σήματος. Με διακεκομμένη και συνεχή γραμμή περιγράφονται το αναλογικό και το ψηφιακό σήμα αντίστοιχα.



Εικόνα 2.6.2.1: Δειγματοληψία ενός αρκετά ασθενούς σήματος. Με διακεκομμένη και συνεχή γραμμή περιγράφονται το αναλογικό και το σήμα θορύβου που παράγεται κατά τη δειγματοληψία της Εικόνας 2.6.2 αντίστοιχα.

Επειδή το αυτί μας είναι λιγότερο ευαίσθητο στην παρουσία τυχαίου θορύβου παρά θορύβου παραμορφώσεων, μια διαδικασία με την οποία αντιμετωπίζεται η παρουσία ορισμένου τύπου παραμορφώσεων είναι με τη μετατροπή των παραμορφώσεων αυτών σε θόρυβο υψηλών συχνοτήτων. Η μέθοδος αυτή που ονομάζεται dithering συνίσταται στην εισαγωγή μιάς μικρής ποσότητας αναλογικού θορύβου πριν από τη δειγματοληψία. Ο θόρυβος αυτός λέγεται dither και προκαλεί τυχαίες μεταβολές σε συχνότητες χαμηλής έντασης οι οποίες περιορίζουν την εμφάνιση των τετραγωνικών αυτών παλμών (και των αρμονικών τους).

Ενα άλλο πρόβλημα που οφείλεται στην περιορισμένη διακριικότητα της ψηφιακής αναπαράστασης αφορά τις μέγιστες εντάσεις ήχων που μπορούν να αποδοθούν σωστά από το σύστημα. Πιο συγκεκριμένα, το εύρος των εντάσεων που μπορεί να αναπαραχθεί από ένα ψηφιακό σύστημα ορίζεται σαν το δυναμικό εύρος (dynamic range) του συστήματος. Το δυναμικό εύρος ενός τέτοιου συστήματος μας δίνεται προσεγγιστικά από τον ακόλουθο τύπο:

$$(\text{δυναμικό εύρος})\text{dB} = \text{αριθμός bits λέξης} \times 6.11$$

Οταν το ψηφιακό σύστημα είναι αναγκασμένο να αποδώσει ένα ήχο εντάσεως μεγαλύτερης από το δυναμικό του εύρος, τότε, λόγω υπερχείλισης των δεδομένων, η θετική τιμή του σήματος μας μετατρέπεται σε μια μεγάλη αρνητική τιμή με ολέθρια αποτελέσματα στην πιστότητα της απόδοσης του ήχου.

2.7 Μέθοδοι Επεξεργασίας Ηχητικών Σημάτων

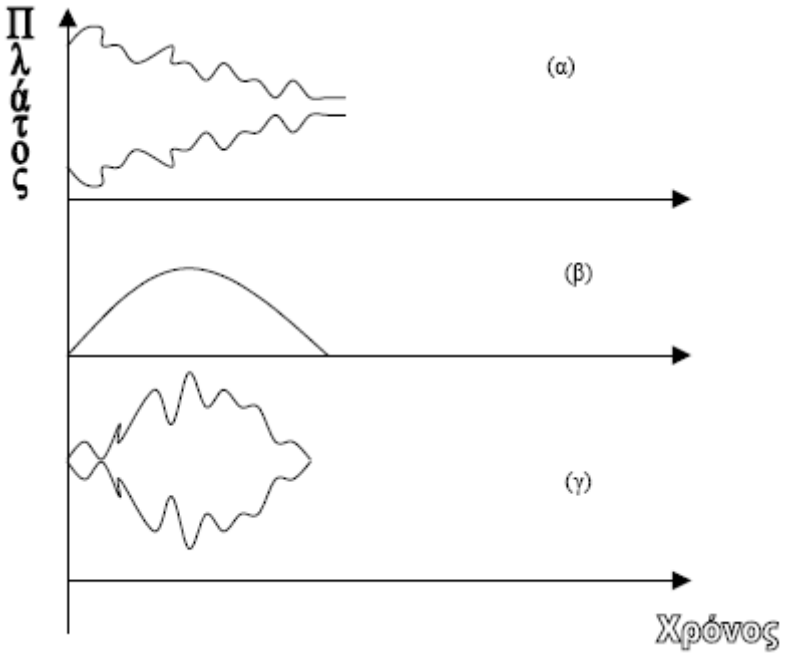
Στην τρέχουσα ενότητα θα παρουσιάσουμε τις κυριότερες μεθόδους επεξεργασίας σήματος που χρησιμοποιούνται σε ηχητικά σήματα. Οι μέθοδοι αυτές περιλαμβάνουν τη μεταβολή τπυ δυναμικού εύρους, τα διάφορα είδη φίλτρων, τη συνέλιξη και τους διάφορους τρόπους χρονική υστέρησης ενός ηχητικού σήματος.

2.7.1 Δυναμική Μεταβολή Εύρους

Οι τεχνικές δυναμικής μεταβολής του εύρους των ηχητικών σημάτων (dynamic range processing) μετασχηματίζουν το πλάτος (ένταση) των ηχητικών σημάτων. Οι τεχνικές αυτές χρησιμοποιούνται σε μια σειρά από συσκευές όπως μορφοποιητές περιγράμματος (envelope shapers), πύλες θορύβου (noise gates), συμπιεστές (compressors), μεγεθυντές (expanders), κλπ. Οι συγκεκριμένες μέθοδοι εφαρμόζονται σε μια ποικιλία εργασιών που εκτείνονται από αρκετά πρακτικές εφαρμογές όπως λ.χ. η απάλειψη του θορύβου από παλιές ηχογραφήσεις μέχρι τη μορφοποίηση μουσικών σημάτων ή ακόμα της ανθρώπινης φωνής κατά τη μουσική σύνθεση.

2.7.1.1 Μορφοποίηση Περιγράμματος

Τα περισσότερα από τα συστήματα επεξεργασίας ήχων παρέχουν τη δυνατότητα της μορφοποίησης του περιγράμματος των σημάτων που διαχειρίζονται. Η μορφοποίηση μπορεί να περιλαμβάνει μια μικρή μεταβολή της έντασης του σήματος είτε μια επανασχεδίαση της μορφής του σήματος. Για παράδειγμα, η Εικόνα 2.7.1.1 περιγράφει τη μορφοποίηση με γραφικό τρόπο ενός ηχητικού σήματος.



Εικόνα 2.7.1.1: Μορφοποίηση περιγράμματος: (α) Αρχικό σήμα (β) Επιθυμητό περίγραμμα (γ) Μετασχηματισμός του αρχικού σήματος με βάση το επιθυμητό περίγραμμα.

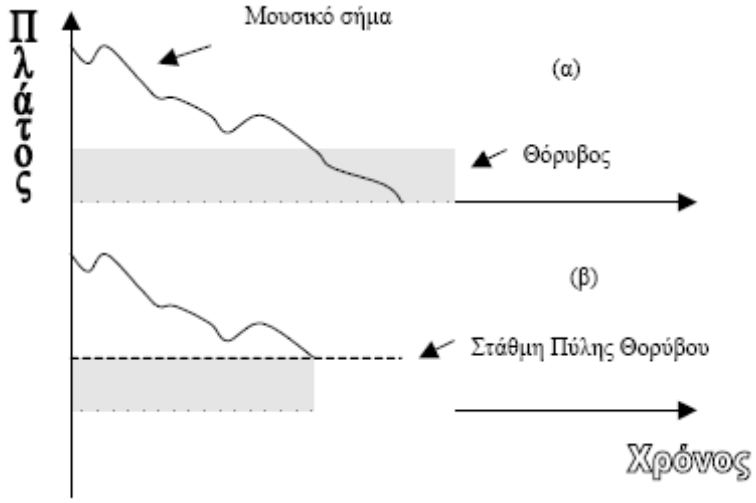
2.7.1.2 Πύλες Θορύβου

Οι πύλες θορύβου προσφέρουν ένα φτηνό τρόπο καθαρισμού μουσικών σημάτων που περιέχουν θόρυβο σταθερής έντασης (π.χ. hiss ή hum). Η πύλη θορύβου συμπεριφέρεται σαν ένας διακόπτης ο οποίος είναι ανοιχτός όταν δέχεται ως είσοδο ένα μουσικό σήμα υψηλής έντασης και κλείνει όταν η ένταση του σήματος πέφτει κάτω από ένα καθορισμένο όριο αποκόπτωντας έτσι οποιονδήποτε εναπομείναντα θόρυβο (βλ. Εικόνα 3.10).

2.7.1.3 Συμπιεστές

Οι συμπιεστές είναι ενισχυτές στους οποίους το μέγεθος ενίσχυσης (gain) του σήματος μεταβάλλεται ανάλογα με την ένταση του σήματος εισόδου. Οι συμπιεστές χρησιμοποιούνται αρκετά συχνά για να διατηρείται η ένταση του σήματος εισόδου σχετικά σταθερή. Στην περίπτωση αυτή ο συμπιεστής μειώνει την ένταση του σήματος εισόδου όταν αυτή ξεπερνά κάποια στάθμη. Ένας αρκετά διαδεδομένος τρόπος περιγραφής της λειτουργίας ενός συμπιεστή αποτελεί η συνάρτηση μεταφοράς (transfer function) η οποία περιγράφει τον τρόπο με τον οποίο μια τιμή

της έντασης του σήματος εισόδου αντιστοιχίζεται σε μια τιμή της έντασης του σήματος εισόδου.



Εικόνα 2.7.1.3: Πύλη Θορύβου: (α) Αρχικό σήμα + Θόρυβος (β) Μετασχηματισμός του αρχικού σήματος με χρήση πύλης θορύβου.

Ο λόγος συμπίεσης (compression ratio) ορίζεται ως ο λόγος της μεταβολής της έντασης του σήματος εισόδου ως προς τη μεταβολή του σήματος εξόδου. Για παράδειγμα ένας λόγος συμπίεσης 4:1 καθορίζει ότι μια μεταβολή της έντασης εισόδου κατά 4 dB προκαλεί μια μεταβολή μόνο 1 dB στο σήμα εξόδου. Ακραίοι λόγοι συμπίεσης (π.χ. 10:1) περιγράφονται με τον όρο limiting και χρησιμοποιούνται κυρίως σε περιπτώσεις κατά τις οποίες πρέπει να διασφαλιστεί ότι καμμία από τις πηγές παραγωγής ήχων δεν φτάνει στα όρια του δυναμικού της εύρους (π.χ. ζωντανές ηχογραφήσεις).

2.7.1.4 Μεγεθυντές

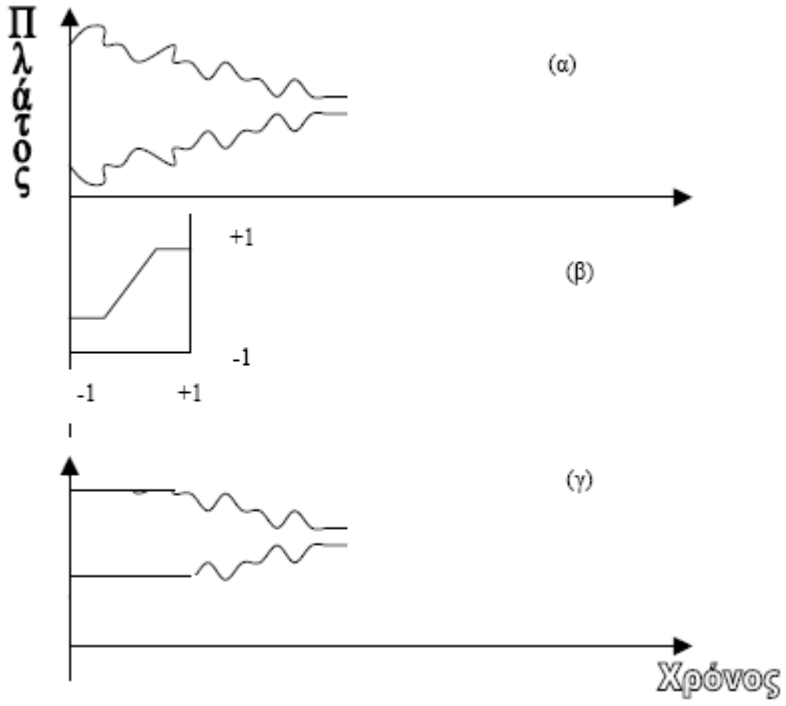
Οι μεγεθυντές (expanders) λειτουργούν αντίστροφα από τους συμπιεστές και μεγεθύνουν μικρές μεταβολές στο σήμα εισόδου. Ο λόγος μεγέθυνσης καθορίζει το βαθμό της μεταβολής του σήματος εισόδου. Για παράδειγμα, ένας λόγος μεγέθυνσης 1:5 δηλώνει ότι μία μεταβολή του σήματος εισόδου κατά 1 dB θα προκαλέσει μια μεταβολή κατά 5 dB του σήματος εξόδου.

Αρκετές από τις μεθόδους απάλειψης θορύβων σε ηχητικά συστήματα χρησιμοποιούν ένα μίγμα συμπιεστών και μεγεθυντών. Για παράδειγμα, το σύστημα Dolby εφαρμόζει ξεχωριστά ζεύγη συμπιεστών και μεγεθυντών για διαφορετικά εύρη συχνοτήτων στο σήμα εισόδου. Ο διαχωρισμός ανά συχνότητες των μεθόδων συμπίεσης διευκολύνει την αντιμετώπιση των προβλημάτων που εισάγει η μεταβολή

του δυναμικού εύρους ενός ηχητικού σήματος

2.7.1.5 Προβλήματα κατά τη Μεταβολή του Δυναμικού Εύρους

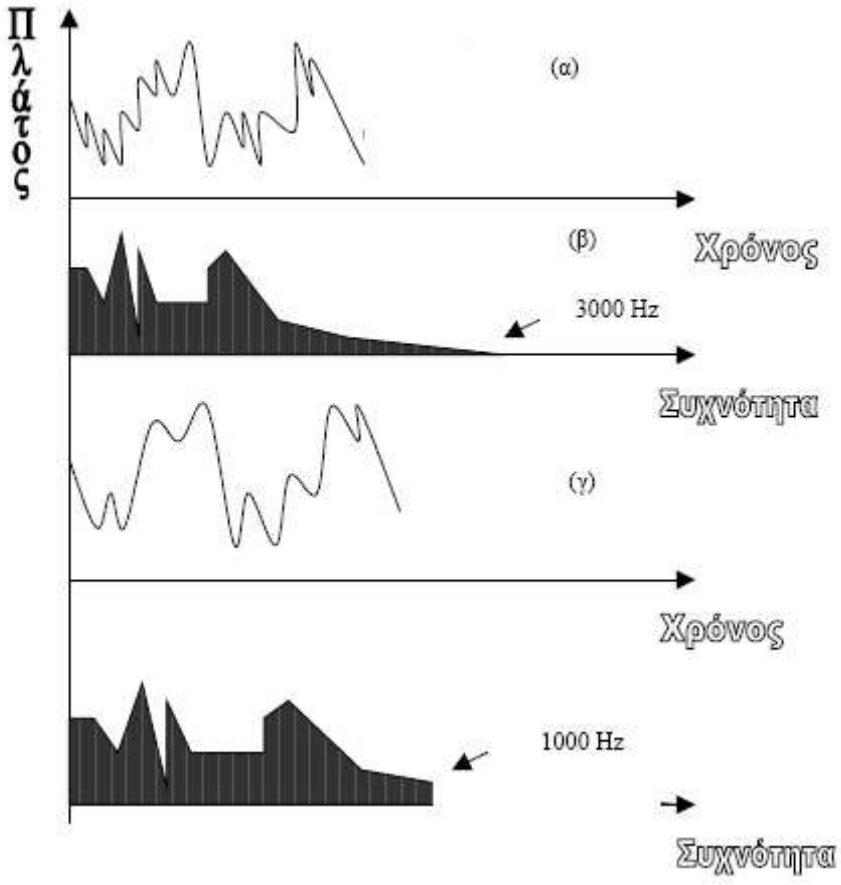
Το μεγαλύτερο πρόβλημα που προκύπτει από τη χρήση των μεθόδων μεταβολής του δυναμικού εύρους που εξετάσαμε προέρχεται από την παραμόρφωση που εισάγουν στις μεταβατικές (transient) περιοχές μίας κυματομορφής όπως π.χ. στις αρχικές απότομες μεταβολές της έντασης (attack) και στη βαθμιαία εξασθένιση (decay) του σήματος που παράγουν αρκετά μουσικά όργανα και που καθορίζουν σε σημαντικό βαθμό τη χροιά του ήχου τους.



Εικόνα 2.7.1.5: Συμπίεση: (α) Αρχικό σήμα (β) Συνάρτηση Μεταφοράς (γ) Μετασχηματισμός του αρχικού σήματος με βάση τη συνάρτηση μεταφοράς.

2.7.2 Ψηφιακά Φίλτρα

Ο όρος φίλτρα αναφέρεται σε συσκευές που ενισχύουν ή αποκόπτουν τμήματα του φάσματος συχνοτήτων που απαρτίζουν ένα ήχητικό σήμα. Υπάρχουν δύο τρόποι με τους οποίους μπορούμε να περιγράψουμε το αποτέλεσμα της εφαρμογής ενός φίλτρου σε ένα σήμα εισόδου: στο χώρο του χρόνου (time domain) και στο χώρο της συχνότητας (frequency domain). Για παράδειγμα η Εικόνα 3.12 περιγράφει το αποτέλεσμα της εφαρμογής ενός φίλτρου που αποκόπτει συχνότητες μεγαλύτερες των 1000 Hz σε ένα ηχητικό σήμα.

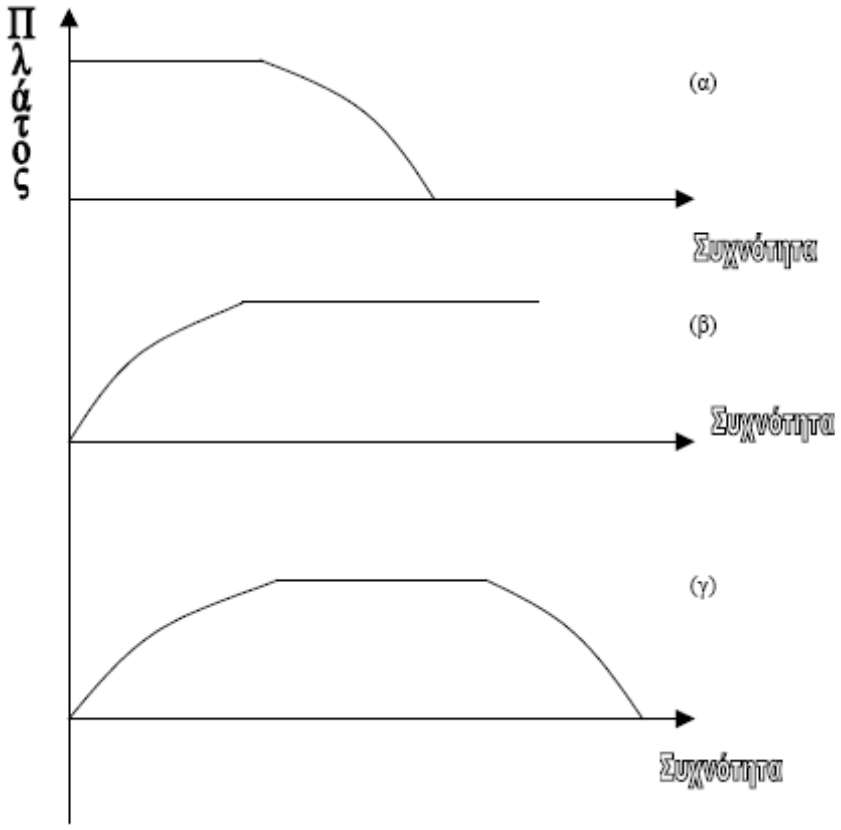


Εικόνα 2.7.2: Ψηφιακό φίλτρο: (α) Αρχικό σήμα στο πεδίο του χρόνου (β) Φάσμα συχνοτήτων αρχικού σήματος (γ) Φιλτραρισμένο σήμα στο πεδίο του χρόνου (δ) Φάσμα συχνοτήτων φιλτραρισμένου σήματος.

Για να έχουμε μια όσο το δυνατόν πληρέστερη περιγραφή των αποτελεσμάτων που έχει η εφαρμογή ενός σήματος σε ένα ηχητικό σήμα θα πρέπει να χρησιμοποιήσουμε ένα σήμα εισόδου που να περιέχει όλες τις συχνότητες. Τι μορφή θα έχει ένα τέτοιο σήμα:

Από τον προηγούμενο αιώνα είναι γνωστό μέσω της ανάλυσης κατά Fourier ότι υπάρχει μία αντίστροφη σχέση μεταξύ της διάρκειας ενός σήματος και των συχνοτήτων που περιέχει. Διαισθητικά μπορούμε να δούμε ότι ένα συνημιτονικό σήμα άπειρης διάρκειας θα αποτελείται από μία και μόνο συχνότητα. Οσο μικραίνει η διάρκεια ενός τέτοιου σήματος, η πολυπλοκότητα του φάσματος συχνοτήτων που αντιστοιχεί σε αυτό θα αυξάνει. Με άλλα όσο μικραίνει η διάρκεια ενός σήματος τόσο θα αυξάνει ο αριθμός των συνημιτονικών συνιστωσών άπειρης διάρκειας που θα πρέπει να χρησιμοποιήσουμε για να το περιγράψουμε. Σε ένα ψηφιακό σύστημα, το πιο σύντομο σήμα που υπάρχει αποτελείται από ένα και μόνο δείγμα. Το σήμα αυτό περιέχει το σύνολο των συχνοτήτων και αποτελεί τη ψηφιακή προσέγγιση ενός αναλογικού σήματος απειροελάχιστης διάρκειας που είναι γνωστό ως το δέλτα του Kronecker ήη κρουστική συνάρτηση (unit impulse). Αν τροφοδοτήσουμε το σήμα αυτό στη είσοδο ενός γραμμικού, αμετάβλητου στο χρόνο φίλτρου τότε το σήμα

εξόδου θα περιγράφει την επίδραση του φίλτρου σε όλο το φάσμα συχνοτήτων με άλλα λόγια θα μας δίνει μια ακριβέστατη περιγραφή της συμπεριφοράς του φίλτρου στο χώρο της συχνότητας,. Η περιγραφή του συγκεκριμένου σήματος εξόδου στο χώρο του χρόνου ονομάζεται κρουστική απόκριση (impulse response) του φίλτρου. Η περιγραφή του συγκεκριμένου σήματος εξόδου στο χώρο της συχνότητας ονομάζεται απόκριση συχνότητας (frequency response) του φίλτρου.



Εικόνα 2.7.2: Παραδείγματα απόκρισης συχνότητας φίλτρων: (α) Βαθυπερατό φίλτρο (β) Υψυπερατό φίλτρο (γ) Ζωνοπερατό φίλτρο.

Ανάλογα με τη μορφή που έχει η απόκριση συχνότητας τα φίλτρα διαχωρίζονται (κυρίως) σε βαθυπερατά (lowpass), υψιπερατά (highpass) και ζωνοπερατά (bandpass) Ενα βαθυπερατό φίλτρο εξασθενεί τις συχνότητες που είναι μεγαλύτερες από μια ορισμένη συχνότητα. Ενα υψιπερατό φίλτρο παρουσιάζει την αντίθετη συμπεριφορά. Τέλος ενα ζωνοπερατό (bandpass) φίλτρο εξασθενεί τις συχνότητες που είναι μικρότερες (μεγαλύτερες) από ένα κάτω (άνω) όριο γύρω από μία κεντρική συχνότητα.

Ενα σημαντικό χαρακτηριστικό των κατηγοριών φίλτρων που αναφέραμε είναι η συχνότητα αποκοπής (cutoff frequency). Η συχνότητα αυτή ορίζεται σαν η συχνότητα εκείνη στην οποία το πλάτος του σήματος πέφτει στο 0.707 της μέγιστης τιμής της. Επειδή η ισχύς του σήματος είναι ανάλογη του τετραγώνου του πλάτους του, στη συχνότητα αποκοπής η ισχύς του σήματος θα είναι η μισή της μέγιστης τιμής της ($0.707^2 = 0.5$). Οσο πιο μικρή είναι η διαφορά μεταξύ της συχνότητας στην οποία αρχίζει η εξασθένηση του σήματος και της συχνότητας αποκοπής τόσο πιο απότομο είναι το φίλτρο.

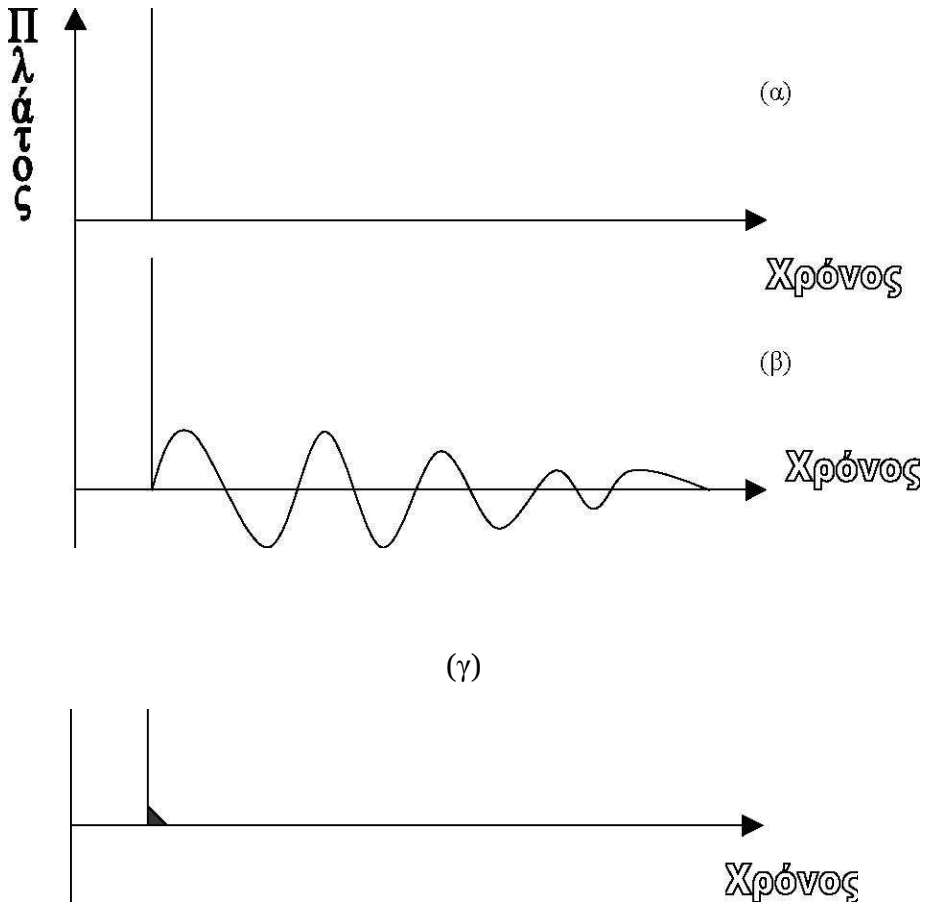
Οσο πιο απότομο είναι το φίλτρο τόσο πιο μεγάλη σε διάρκεια είναι η κρουστική του απόκριση και αντίστροφα.

2.7.2.1 Λογισμικό Υλοποίησης Ψηφιακών Φίλτρων

Υπάρχουν δύο τρόποι με τους οποίους υλοποιείται ένα ψηφιακό φίλτρο:
1 Με την χρονική υστέρηση ενός αντιγράφου του σήματος εισόδου και τον συνδυασμό του καθυστερημένου σήματος με το καινούργιο σήμα εισόδου. Στην περίπτωση αυτή μιλάμε για φίλτρα FIR (Finite Impulse Response).
2 Με την καθυστέρηση ενός αντιγράφου του σήματος εξόδου και τον συνδυασμό του με το σήμα εισόδου. Στην περίπτωση αυτή μιλάμε για φίλτρα IIR (Infinite Impulse Response).

Η υλοποίηση των ψηφιακών φίλτρων βασίζεται στη χρήση εξισώσεων που περιγράφουν την επίδραση του φίλτρου στα δείγματα του σήματος εισόδου. Οι εξισώσεις αυτές μπορούν να παρασταθούν σα διαγράμματα στα οποία τα βέλη περιγράφουν τη φορά του σήματος, οι απλές αριθμημένες γραμμές περιγράφουν τους συντελεστές που χρησιμοποιούνται στην υλοποίηση του φίλτρου, οι τελείες περιγράφουν σημεία διακλάδωσης του σήματος, τα σύμβολα «+» και «x» περιγράφουν τις πράξεις πρόσθεσης και πολλαπλασιασμού ενώ το γράμμαΔ περιγράφει ένα κύκλωμα καθυστέρησης του σήματος εισόδου κατά μια περίοδο. Για παράδειγμα, η Εικόνα 2.7.2.1 περιγράφει την υλοποίηση του βαθυπερατού φίλτρου που χαρακτηρίζεται από την εξίσωση:

$$y[n] = 0.5 \times x[n]+0.5 \times x[n-1]$$

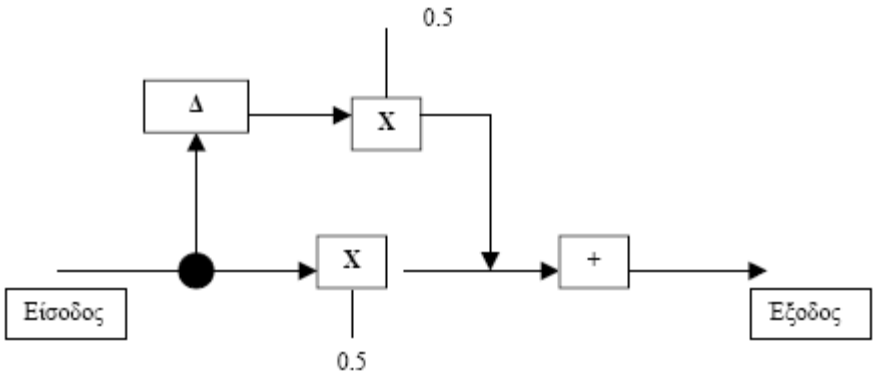


Εικόνα 2.7.2.1: Κρουστική απόκριση φίλτρων: (α) Κρουστική συνάρτηση εισόδου (β) Κρουστική απόκριση ζωνοπερατού φίλτρου (20 dB στα 300 Hz) με μικρό εύρος ζώνης (20 Hz) (γ) Κρουστική απόκριση βαθυπερατού φίλτρου (-15 dB στο 1 KHz).

2.7.2.2 Σύγκριση FIR και IIR φίλτρων

Η εφαρμογή ενός φίλτρου σε ένα σήμα μεταβάλλει πέρα από το πλάτος του και τη φάση του. Η εφαρμογή των FIR φίλτρων είναι γενικά προτιμότερη σε ηχητικά σήματα αφού είναι δυνατό να σχεδιαστούν τέτοια φίλτρα με γραμμική απόκριση φάσης. Μια τέτοια απόκριση προλαμβάνει πιθανές παραμορφώσεις φάσης που εισάγει το φιλτράρισμα ενός ήχου.

Επιπλέον, τα FIR φίλτρα είναι περισσότερο σταθερά από τα IIR μιας και δεν περιέχουν ανάδραση. Από την άλλη μεριά, η υλοποίηση των FIR φίλτρων απαιτεί περισσότερες αριθμητικές πράξεις και μνήμη από ότι η υλοποίηση FIR φίλτρων με αντίστοιχα χαρακτηριστικά.



Εικόνα 2.7.2.2: Παράδειγμα υλοποίησης βαθυπερατού φίλτρου.

2.7.3 Συνέλιξη

Η συνέλιξη αποτελεί μια από τις θεμελιώδεις λειτουργίες με τις οποίες υλοποιείται η επεξεργασία ενός ηχητικού σήματος. Βασικές μορφές επεξεργασίας ψηφιακού ήχου όπως είναι το φιλτράρισμα, η διαμόρφωση ή η αντήχηση στηρίζονται στη διαδικασία της συνέλιξης για να πετύχουν τα επιθυμητά αποτελέσματα. Για παράδειγμα, ένα γραμμικό, αμετάβλητο στο χρόνο φίλτρο συνελίσσει την κρουστική του απόκριση με το σήμα εισόδου για να πετύχει τη δημιουργία ενός φιλτραρισμένου σήματος στην έξοδο.

Η πράξη της συνέλιξης παριστάνεται με το σύμβολο «*». Ο μαθηματικός ορισμός της συνέλιξης δύο πεπερασμένων ψηφιακών σημάτων α, b περιγράφεται από τον τύπο:

$$h[k] = \sum_{n=0}^{N-1} a[n]b[k-n]$$

όπου N είναι ο αριθμός των δειγμάτων του α και η μεταβλητή k λαμβάνει τιμές από 0 έως και L. Η τιμή του L υπολογίζεται από τον τύπο:

$$L = N + M - 2$$

όπου M ο αριθμός των δειγμάτων του σήματος b.

Όπως αναφέραμε και προηγουμένως η συνέλιξη χρησιμοποιείται για την υλοποίηση αρκετών μορφών επεξεργασίας ήχων. Για παράδειγμα, ένας απλός τρόπος με τον οποίο μπορούμε να προσομοιώσουμε την ακουστική εντύπωση και πιο συγκεκριμένα τις αντηχήσεις (reverbarations) που δημιουργεί ένας χώρος (π.χ. ένα άδειο μεγάλο δωμάτιο ή ένα τούνελ) είναι να μετρήσουμε με κατάλληλα όργανα την κρουστική απόκριση του συγκεκριμένου χώρου. Στη συνέχεια μπορούμε να σχεδιάσουμε ένα

ψηφιακό φίλτρο το οποίο θα έχει την ίδια κρουστική απόκριση με το χώρο μας. Αν συνελίξουμε ένα οποιοδήποτε ηχητικό σήμα με την κρουστική απόκριση του φίλτρου που κατασκευάσαμε το σήμα εξόδου θα αποκτήσει τα ακουστικά χαρακτηριστικά του χώρου που μας ενδιαφέρει!!

2.7.4 Μορφές Χρονικής Καθυστέρησης

Η χρονική καθυστέρηση (delay) ενός ηχητικού σήματος συνίσταται στην κατακράτηση στη μνήμη της ροής του σήματος και στη συνέχεια στη ανάμιξη της με το σήμα που λαμβάνεται από το σύστημα την παρούσα χρονική στιγμή. Μια τέτοια επεξεργασία δημιουργεί μια ποικιλία από ηχητικά εφφέ.

Πιο συγκεκριμένα, η χρονική καθυστέρηση που εφαρμόζεται στο σήμα μπορεί να είναι σταθερή ή μεταβλητή. Στην πρώτη περίπτωση, όταν η διάρκεια της καθυστέρησης είναι μικρή (≤10 msec) δημιουργούνται αλλοιώσεις στο φάσμα συχνοτήτων του σήματος εισόδου ανάλογες με αυτές που προκαλούνται από την εφαρμογή ενός βαθυπερατού φίλτρου. Μια μέση καθυστέρηση (≥10 msec, ≤ 50 msec) προσθέτει «σώμα» στον ήχο δημιουργώντας την ψευδαίσθηση μίας υψηλότερης υποκειμενικής έντασης από αυτή που πραγματικά διαθέτει ένα ηχητικό σήμα. Τέλος μεγάλες καθυστερήσεις (≥ 50 msec) δημιουργούν το φαινόμενο της ηχούς στο σήμα εισόδου.

Στη φύση η ηχώ δημιουργείται όταν τα ηχητικά σήματα μεταδίδονται από την πηγή τους, ανακλώνται σε μια επιφάνεια και γίνονται αντιληπτά από τον ακροατή αρκετά μετά από τον ήχο που ακολούθησε την ευθεία οδό διάδοσης προς τον ακροατή. Επειδή η ταχύτητα διάδοσης του ήχου στον αέρα είναι περίπου 344 μέτρα το δευτερόλεπτο στους 20^{0} C, μια καθυστέρηση άφιξης του σήματος στον ακροατή ίση με1 msec αντιστοιχεί σε απόσταση διάδοσης 30 cm. Για να δημιουργηθεί το φαινόμενο της ηχούς θα πρέπει να υπάρξει καθυστέρηση άφιξης του σήματος κατά τουλάχιστον 50 msec. Η συγκεκριμένη καθυστέρηση αντιστοιχεί σε μια απόσταση περίπου 8 μέτρων από την επιφάνεια ανάκλασης του ηχητικού σήματος ή σε διαδρομή περίπου 16 μέτρων του ηχητικού σήματος από την πηγή του μέχρι τον ακροατή μέσω της επιφάνειας ανάκλασης. Οταν οι αποστάσεις που περιγράψαμε είναι μικρότερες τότε εμφανίζονται φανόμενα αντήχησης (reverberation) κατά την ακρόαση του ήχου.

Η μεταβλητή καθυστέρηση ενός ηχητικού σήματος χρησιμοποιείται για την υλοποίηση τριών εφφέ: flanging, phasing και chorus.

2.8 Ευαισθησία

Το αυτί μας είναι τρομερά ικανό να διακρίνει μεταξύ ενός συνόλου ηχητικών σημάτων που δέχεται συγχρόνως όπως και να αντιλαμβανεται πολυ μικρές διαφορές στη λήψη του ιδίου σήματος. Για παράδειγμα το αυτί μας είναι ικανό να παρακολουθεί δύο ή περισσότερες συζητήσεις συγχρόνως. Επίσης είναι ικανό να αντιλαμβάνεται διακοπές στην παραγωγή ενός ηχητικού σήματος που αρχίζουν από τα 40 msec. Τέλος το αυτί μας αντιλαμβάνεται παροδικές παραμορφώσεις

συχνότητας που αρχίζουν από το 10% της εκπεμπομένης συχνότητας. Η εκπληκτική διακριτική ικανότητα του αυτιού μας μπορεί να γίνει κατανοητή και από την ευκολία με την οποία αντιλαμβανόμαστε ανεπάισθητους και ασυνήθιστους ήχους σε οικείους χώρους.

2.9 Τριδιάστατος ήχος

Ο προσδιορισμός της κατεύθυνσης από την οποία εκπέμπεται ένας ήχος στηρίζεται στην εκτίμηση τριών μεγεθών που καθορίζουν τη θέση της ηχητικής πηγής ως προς τη θέση του ακροατή: του αζιμουθίου (azimuth) δηλ. της οριζόντιας γωνίας που σχηματίζει η ηχητική πηγή με τη θέση του ακροατή, της απόστασης ή της ταχύτητας για σταθερές ή κινούμενες πηγές αντίστοιχα και του ζενίθ (zenith) δηλ. του ύψους ή της κάθετης γωνίας που σχηματίζει η πηγή με τον ακροατή.

Η εκτίμηση του αζιμουθίου μίας ηχητικής πηγής βασίζεται στα ακόλουθα χαρακτηριστικά της ακρόασης:

- Στο διαφορετικό χρόνο άφιξης ενός ήχου σε καθένα από τα αυτία μας όταν ο ήχος έρχεται από μια πλευρά.
- Στη διαφορά στο πλάτος των υψηλών συχνοτήτων που γίνονται αντιληπτές σε καθένα από τα αυτιά μας. Η διαφορά οφείλεται στην απορρόφηση ορισμένων υψηλών συχνοτήτων από το κρανίο του ακροατή.
- Στις επιπτώσεις που έχει στο φάσμα συχνοτήτων του ήχου οι ασύμμετρες ανακλάσεις του ηχητικού σήματος στην εξωτερική πλευρά των αυτιών, των ώμων και του κορμού του ακροατή.

Η εκτίμηση της απόστασης στηρίζεται στα παρακάτω χαρακτηριστικά της ακροάσης:

- Στο λόγο της έντασης του άμεσου σήματος προς την ένταση του σήματος που προέρχεται από αντηχήσεις, όταν η ένταση του άμεσου σήματος μειώνεται ανάλογα με το τετράγωνο της απόστασης της πηγής από τον ακροατή.
- Στην απώλεια των συνιστωσών που περιέχουν υψηλές συχνότητες σε ένα ηχητικό σήμα λόγω της απόστασης της ηχητικής πηγής από τον ακροατή.
- Στην απώλεια των λεπτομερειών του ηχικού σήματος (δηλ. των συχνοτήτων χαμηλής έντασης) λόγω της απόστασης της ηχητικής πηγής από τον ακροατή.

Οταν η απόσταση μεταξύ της ηχητικής πηγής και του παρατηρητή μεταβάλλεται, η εκτίμηση της ταχύτητας και της κατεύθυνσης με την οποία γίνεται η μεταβολή στηρίζεται στη μεταβολή του ύψους του ηχητικού σήματος λόγω του φαινομένου Doppler.

Τέλος, η εκτίμηση του ζενίθ προκύπτει από τη μεταβολή του φάσματος συχνοτήτων του σήματος λόγω της ανάκλασης του στους ώμους και στο εξωτερικό τμήμα των αυτιών του ακροατή.

Συστήματα οικιακού κινηματογράφου (home cinema) ή εφαρμογές εικονικής

πραγματικότητας εκμεταλλεύονται τα χαρακτηριστικά της ανθρώπινης ακοής που περιγράψαμε για να δημιουργήσουν τη ψευδαίσθηση του τριδιάστατου ήχου. Η αληθοφάνεια μιας τέτοιας διεργασίας βελτιώνεται και με την εξομοίωση των ακουστικών χαρακτηριστικών του εικονικού χώρου από τον οποίο εκπέμπεται ο ήχος όπως λ.χ. με τον υπολογισμό των διαφόρων ανακλάσεων που θα έχει ένα ηχητικό σήμα στα διάφορα αντικείμενα σε έναν εικονικό κόσμο.

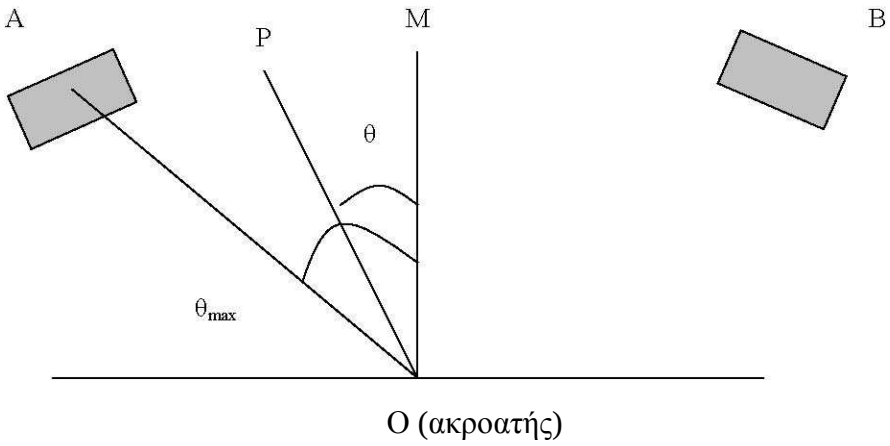
2.9.1 Προσομοίωση Αζιμουθίου

Η προσομοίωση του αζιμουθίου μίας ηχητικής πηγής μπορεί να γίνει με τη χρήση τουλάχιστον δύο ηχείων Α, Β και τη μεταβολή της έντασης του ήχου που παράγει καθένα από αυτά. Για παράδειγμα ο ακροατής έχει την αίσθηση ότι ένας ήχος προέρχεται από τη θέση στην οποία βρίσκεται το Α όταν η ένταση του Α έχει μη μηδενική τιμή και η ένταση του Β είναι ίση με το μηδέν. Η μεταβολή της οριζόντιας θέσης του ήχου γίνεται δυνατή με την εφαρμογή της μεθόδου της οριζόντιας μετατόπισης (panning).

Πιο συγκεκριμένα, αν υποθέσουμε ότι ο ακροατής και τα δύο ηχεία Α, Β έχουν τις θέσεις που περιγράφονται στην Εικόνα 3.16 τότε για να τοποθετηθεί νοητικά μια ηχητική πηγή σε ένα σημείο Ρ μεταξύ των ηχείων Α, Β θα πρέπει να υπολογιστεί η γωνία που σχηματίζει η ευθεία ΟΡ που συνδέει τον ακροατή με το Ρ με τη μεσοκάθετο ΟΜ στο ευθύγραμμοτμήμα ΑΒ. Στην περίπτωση αυτή η ένταση του ήχου σε καθένα από τα ηχεία Α, Β θα πρέπει να πολλαπλασιαστεί με τους συντελεστές ΕΑ, ΕΒ που προκύπτουν από τους τύπους:

$$E_A = \frac{\sqrt{2}}{2}(\cos(\partial) + \sin(\partial))$$

$$E_B = \frac{\sqrt{2}}{2}(\cos(\partial) - \sin(\partial))$$



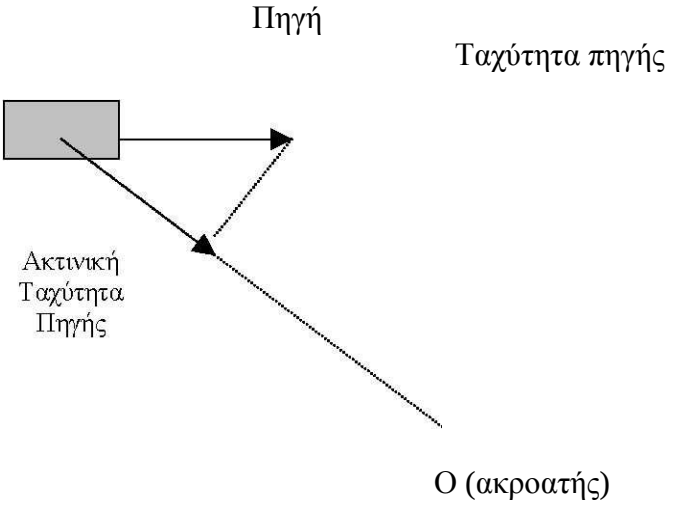
Εικόνα 2.9.1: Προσομοίωση αζιμουθίου με δύο ηχεία.

2.9.2 Φαινόμενο Doppler

Το φαινόμενο Doppler περιγράφει τη μεταβολή του ύψους του ήχου που παράγεται από μια ηχητική πηγή που κινείται σε σχέση με τη θέση του ακροατή. Για παράδειγμα, όταν στεκόμαστε στην άκρη του δρόμου, ο ήχος της μηχανής ενός αυτοκινήτου που κινείται προς έναν ακροατή με μεγάλη ταχύτητα γίνεται όλο και πιο οξύς όσο το αυτοκίνητο πλησιάζει τον ακροατή, ενώ το ύψος του συνεχώς ελαττώνεται όσο το αυτοκίνητο απομακρύνεται απο τον ακροατή. Το ύψος Y του ήχου που γίνεται αντιληπτό στην περίπτωση αυτή είναι ανάλογο της ακτινικής ταχύτητας της πηγής προς τον παρατηρητή και δίνεται από τον τύπο:

$$Y = f\frac{vs}{vs - vr}$$

όπου f η πραγματική συχνότητα του ηχητικού σήματος, vs η ταχύτητα του ήχου και vr η ακτινική ταχύτητα της πηγής σε σχέση με τον ακροατή. Η φυσική εξήγηση του φαινομένου στηρίζεται στην παρατήρηση ότι η απόσταση μεταξύ των ηχητικών κυμάτων που προέρχονται από την πηγή μειώνεται όσο η πηγή πλησιάζει στον παρατηρητή,επομένως η φαινόμενη συχνότητα του ήχου αυξάνεται. Το αντίθετο συμβαίνει όταν η πηγή απομακρύνεται από τον παρατηρητή.



Εικόνα 2.9.2: Φαινόμενο Doppler.

2.9.3 Προσομοίωση Ζενίθ

Η προσομοίωση του ζενίθ που αντιστοιχεί σε μια ηχητική πηγή επιτυγχάνεται με την εφαρμογή φίλτρων στο ηχητικό σήμα τα οποία μεταβάλλουν το φάσμα συχνοτήτων του σήματος κατά τρόπο ανάλογο των μεταβολών που επιφέρουν οι ανακλάσεις του ήχου στο κεφάλι και στους ώμους του ακροατή. Οι μεταβολές αυτές διαφέρουν από ακροατή σε ακροατή καθώς εξαρτώνται από τις διαστάσεις των κεφαλιών και των ώμων τους. Η απόκριση συχνότητας των συγκεκριμένων φίλτρων ονομάζεται συνάρτηση μεταφοράς κεφαλής (head-related transfer function ή HRTF) και για την επίτευξη ενός ικανοποιητικού βαθμού πιστότητας θα πρέπει να αντιστοιχεί στα χαρακτηριστικά του συγκεκριμένου ακροατή.

2.10 Μέθοδοι Συμπίεσης Ηχητικών Σημάτων

Οι περισσότερες μέθοδοι συμπίεσης ηχητικών σημάτων ανήκουν στις τεχνικές με απώλειες. Ο κύριος λόγος για τον οποίο οι συγκεκριμένες τεχνικές είναι τόσο δημοφιλείς οφείλεται στο γεγονός ότι σε ένα ηχητικό σήμα συνήθως δεν υπάρχουν ομοιότυπα (patterns) με ιδιαίτερα μεγάλη συχνότητα εμφάνισης. Πιο συγκεκριμένα, η ψηφιοποίηση οποιουδήποτε αναλογικού σήματος εισάγει παραμορφώσεις που εντοπίζονται κυρίως στα λιγότερο σημαντικά δυαδικά ψηφία του κάθε δείγματος (βλ. § 3.6.2). Ο τυχαίος χαρακτήρας των συγκεκριμένων παραμορφώσεων εμποδίζει την εμφάνιση ομοιοτύπων στη συγκεκριμένη κατηγορία σημάτων.

Ένας επιπλέον λόγος για την περιορισμένη εφαρμογή των τεχνικών χωρίς απώλειες οφείλεται στην ανομοιόμορφη συμπίεση που επιτυγχάνουν σε ένα ηχητικό σήμα. Ο περιορισμός αυτός εμποδίζει τη χρησιμοποίηση των συγκεκριμένων τεχνικών από

εφαρμογές που στηρίζονται στην επίτευξη ενός σταθερού ρυθμού μετάδοσης των δειγμάτων ενός ηχητικού σήματος (π.χ. ραδιόφωνο στο Διαδίκτυο). Για παράδειγμα, έστω ότι μια ραδιοφωνική εκπομπή λαμβάνεται από έναν ακροατή στο Διαδίκτυο μέσω ενός modem που μπορεί να μεταφέρει περίπου 3000 bits/sec. Στην περίπτωση αυτή η εφαρμογή θα πρέπει να διασφαλίσει ότι ένα δευτερόλεπτο ήχου θα πρέπει πάντα να χωρέσει σε 3000 δυαδικά ψηφία. Δυστυχώς οι τεχνικές χωρίς απώλειες δεν μπορούν να πετύχουν έναν τέτοιο σταθερό ρυθμό συμπίεσης καθόλη τη διάρκεια της μετάδοσης. Αντίθετα, η πλειονότητα των τεχνικών με απώλειες πετυχαίνει σταθερούς ρυθμούς συμπίεσης. Για παράδειγμαη μέθοδος IMA ADPCM που θα περιγράψουμε παρακάτω πετυχαίνει πάντα λόγο συμπίεσης 4:1 σε ψηφιοποιημένα ηχητικά σήματα με μήκος δείγματος 16-bit.

Οι κυριότερες μέθοδοι συμπίεσης σε ηχητικά σήματα εφαρμόζονται σε μια συγκεκριμένη αναπαράσταση του σήματος που είναι γνωστή ως παλμοκωδική διαμόρφωση (Pulse Code Modulation ή PCM). Στη συγκεκριμένη μέθοδο κάθε δείγμα αναπαριστάται με ένα σύνολο παλμών που αντιστοιχούν στο δυαδικό κώδικα που αντιστοιχεί στην τιμή του δείγματος. Η πιστότητα του σήματος που προκύπτει είναι συνάρτηση του δυναμικού εύρους του δυαδικού κώδικα. Για παράδειγμα, ένα δυναμικό εύρος 128 τιμών θα έχει ως αποτέλεσμα την απάλειψη ήχων στο ψηφιοποιημένο σήμα που έχουν ένταση ίση ή μικρότερη από το 1/128 της έντασης του δυνατότερου ήχου που μπορεί να αναπαρασταθεί από το σύστημα. Προφανώς μια τέτοια αναπαράσταση εισάγει προβλήματα πιστότητας σε ασθενή σε ένταση ηχητικά σήματα. Για την αντιμετώπιση του συγκεκριμένου προβλήματος είναι δυνατόν να χρησιμοποιηθούν μη γραμμικές μέθοδοι παλμοκωδικής διαμόρφωσης (non-linear PCM). Για παράδειγμα, αν το εύρος τιμών του κώδικα σε μια τέτοια μέθοδο είναι 128 τότε μια τιμή δείγματος ίση με1 μπορεί να μην αντιστοιχεί σε ήχο έντασης ίση με το 1/128 της έντασης του δυνατότερου ήχου αλλά σε ήχο αρκετά μικρότερης έντασης. Μια τέτοια μέθοδος είναι μη γραμμική καθώς χρησιμοποιεί περισσότερα δυαδικά ψηφία για την αναπαράσταση ασθενών ήχων στους οποίους όπως είδαμε υπάρχουν μεγαλύτερα προβλήματα διακριτικότητας και λιγότερα για τους ήχους μεγάλης έντασης στους οποίους η ευαισθησία της ανθρώπινης ακοής σε σχετικές διαφορές έντασης είναι σημαντικά μειωμένη.

Οι πλέον διαδεδομένες μέθοδοι συμπίεσης ηχητικών σημάτων είναι τεχνικές κωδικοποίησης πηγής και ειδικότερα τεχνικές διαφορικής κβαντοποίησης όπως οι οικογένειες μεθόδων DPCM (Differential Pulse Code Modulation), ADPCM (Adpative Differential Pulse Code Modulation) και οι τεχνικές κωδικοποίησης υποζώνης (subband coding). Πιο συγκεκριμένα, η οικογένενεια των DPCM τεχνικών αποθηκεύουν τις διαφορές μεταξύ διαδοχικών τιμών του σήματος και όχι τις απόλυτες τιμές των δειγμάτων. Το ίδιο συμβαίνει και στις τεχνικές ADPCM με τη διαφορά ότι στις τελευταίες η κλίμακα αναπαράστασης των διαφορών είναι μεταβλητή και προσαρμόζεται σε μία πρόβλεψη του απόλυτου μεγέθους των τιμών των δειγμάτων που συγκρίνονται με βάση τις προηγούμενες τιμές των δειγμάτων αυτών. Τα πρότυπα συμπίεσης WAVE της Microsoft και AIFF-C της Apple αποτελούν παραδείγματα ADPCM μεθόδων.

Ακριβέστερα IMA ADPCM δηλ. μια ειδική κατηγορία ADPCM μεθόδων.
στο συχνοτικό περιεχόμενο του σήματος κάθε χρονική στιγμή κάτι που δε γίνεται στις δύο τεχνικές που περιγράψαμε. Αντίθετα, οι τεχνικές κωδικοποίησης υποζώνης

υποδιαιρούν το ηχητικό σήμα σε δύο η περισσότερες ζώνες συχνοτήτων και συμπιέζουν καθεμία από αυτές ξεχωριστά. Η χρησιμοποίηση μίας τέτοιας υποδιαίρεσης εκμεταλλεύεται τα φυσιολογικά χαρακτηριστικά της ανθρώπινης ακοής η οποία παρουσιάζει τη μέγιστη ευαισθησία στο εύρος συχνοτήτων μεταξύ 27003200 Hz με την ευαισθησία να μειώνεται όσο απομακρυνόμαστε από τη ζώνη αυτή. Κατά συνέπεια μια μέθοδος κωδικοποίησης υποζώνης μπορεί να συμπιέζει δραστικά ήχους με συχνοτικό περιεχόμενο που απέχει αρκετά από τη ζώνη που αναφέραμε και να εφαρμόζει την ελάχιστη συμπίεση σε συχνότητες που ανήκουν στη ζώνη αυτή. Τα αποτελέσματα μίας τέτοιας διαδικασίας δε θα γίνονται εύκολα αντιληπτά από την ανθρώπινη ακοή. Στην κατηγορία των μεθόδων κωδικοποίησης υποζώνης ανήκουν τα πρότυπα συμπίεσης MPEG audio, Dolby AC-2 και AC-3, το σύστημα Sony MiniDisc και το RealAudio.

Ενα σημαντικό μειονέκτημα των τεχνικών DPCM και ADPCM οφείλεται στην εξάρτηση του μεγέθους των διαφορών μεταξύ διαδοχικών δειγμάτων ενός ηχητικού σήματος από τη συχνότητα του σήματος. Όπως είναι αναμενόμενο, οι διαφορές μεταξύ διαδοχικών δειγμάτων σε ήχους χαμηλών συχνοτήτων είναι σημαντικά μικρότερες από τις αντίστοιχες διαφορές σε ήχους υψηλών συχνοτήτων. Επομένως η διακριτική ικανότητα της κωδικοποίησης των διαφορών θα πρέπει να προσαρμόζεται.

ΚΕΦΑΛΑΙΟ 3

ΕΙΣΑΓΩΓΗ ΣΤΗΝ ΨΗΦΙΑΚΗ ΕΠΕΞΕΡΓΑΣΙΑ ΣΗΜΑΤΟΣ

3.1 Ιστορική Αναδρομή

Από τον Δεύτερο Παγκόσμιο Πόλεμο, ορισμένοι ηλεκτρονικοί μηχανικοί εξέταζαν την δυνατότητα εφαρμογής τεχνικών ψηφιακού λογισμικού σε τομείς όπου η επεξεργασία σημάτων έπαιζε σημαντικό ρόλο. Αρχικά τα συμπεράσματα δεν ήταν καθόλου ευνοϊκά για την χρησιμοποίηση τους.

• Κόστος, μέγεθος και αξιοπιστία ήταν οι παράγοντες οι οποίοι ευνοούσαν την χρησιμοποίηση αναλογικών φίλτρων και αναλογικών τεχνικών φασματικής ανάλυσης.

• Το 1960 ο I.J. Gold ανέπτυξε μεθόδους για τον υπολογισμό του διακριτού μετασχηματισμού Fourier (DFT) με την χρησιμοποίηση μετασχηματισμών πινάκων.

• Αυτή η ανακάλυψη παρέμεινε ανεκμετάλλευτη λόγω των περιορισμένων υπολογιστικών δυνατοτήτων της εποχής εκείνης.

• Η Ψηφιακή Επεξεργασία Σημάτων - Digital Signal Processing (DSP) διαδόθηκε την δεκαετία του 60 με την ανάπτυξη των ηλεκτρονικών υπολογιστών.

• Kaiser: σχεδιασμός και σύνθεση ψηφιακών φίλτρων

• Cooley-Tukey: γρήγορη μέθοδος υπολογισμού του διακριτού μετασχηματισμού Fourier (FFT). Πρώτος καθαρά ψηφιακός αλγόριθμος

• Αρχικά, η υλοποίηση των φίλτρων γινόταν με λογισμικό Υλοποίηση DSP

• Τα πρώτα DSP κυκλώματα εκτελούσαν προκαθορισμένες πράξεις

• ο χρήστης δεν μπορούσε να τα προγραμματίσει και έτσι δεν είχε τηνδυνατότητα να μεταβάλλει εξωτερικά τα χαρακτηριστικά τους.

• Ο μικροπρογραμματισμός έδωσε νέες προοπτικές όπως την ανάπτυξη πολλαπλών συναρτήσεων, σύνθεση νέων.

• Large Scale Integration (LSI) επέτρεψε την χρήση του προγραμματισμού στα DSP κύκλώματα

Η εντυπωσιακή ανάπτυξη της μικροηλεκτρονικής και των υπολογιστών είχε καθοριστική επίδραση στην ψηφιακή επεξεργασία σημάτων και εικόνων. Οι τεχνικές ψηφιακής επεξεργασίας σημάτων χρησιμοποιούνται σήμερα σε πολλές περιοχές της επιστήμης και της τεχνολογίας, όπως για παράδειγμα τις επικοινωνίες, την αεροναυτική, την σεισμολογία, την βιοϊατρική τεχνολογία.

Ως σήμα ορίζουμε το σύνολο των τιμών που λαμβάνει μια φυσική ποσότητα. Μαθηματικά αυτό εκφράζεται ως συνάρτηση ή ακολουθία μιας ή περισσοτέρων ανεξάρτητων μεταβλητών. Τα σήματα περιέχουν πληροφορία σχετικά με την συμπεριφορά ή τη φύση ενός φαινομένου.

Τα σήματα χωρίζονται σε κατηγορίες ανάλογα με τον τύπο και το πλήθος των μεταβλητών. Έτσι, ανάλογα με το αν η ανεξάρτητη μεταβλητή είναι συνεχής ή διακριτή, έχουμε σήματα συνεχούς ή διακριτού χρόνου. Ανάλογα με το πλήθος των ανεξάρτητων μεταβλητών έχουμε σήματα μιας μεταβλητής ή διάστασης (μονοδιάστατα, 1-D), δύο μεταβλητών ή διαστάσεων (διδιάστατα, 2-D) και πολλών μεταβλητών ή διαστάσεων (πολυδιάστατα, N-D). Παράδειγμα μονοδιάστατου σήματος είναι η ομιλία (speech), όπου η ανεξάρτητη μεταβλητή είναι ο χρόνος και η εξαρτημένη μεταβλητή η ακουστική πίεση. Μια εικόνα (φωτογραφία) αποτελεί χαρακτηριστικό παράδειγμα σήματος δύο διαστάσεων. Εδώ η εξαρτημένη μεταβλητή είναι η φωτεινότητα της εικόνας και οι δύο ανεξάρτητες μεταβλητές είναι οι δύο χωρικές συντεταγμένες. Τέλος, παράδειγμα σήματος τριών διαστάσεων είναι η ακολουθία εικόνων (video), όπου οι δύο ανεξάρτητες μεταβλητές είναι χωρικές και η τρίτη είναι ο χρόνος . Η εξαρτημένη μεταβλητή είναι και εδώ η φωτεινότητα της κάθε εικόνας.

Καθένα από τα παραπάνω σήματα είναι απαραίτητο να το επεξεργαστούμε με στόχο να βελτιώσουμε την ποιότητά του, να εξάγουμε την χρήσιμη πληροφορία και να εξαλείψουμε κατά το δυνατόν τον θόρυβο, να το αποθηκεύσουμε ή να το μεταδώσουμε

3.2 Ψηφιακή Επεξεργασία Σήματος

Η ψηφιακή επεξεργασία σήματος ασχολείται με την ψηφιακή αναπαράσταση των σημάτων και την ανάλυση, τροποποίηση και εξαγωγή πληροφοριών από αυτά, με την βοήθεια ψηφιακών επεξεργαστών. Περιπτώσεις κατά τις οποίες θέλουμε να αφαιρέσουμε τον θόρυβο από ένα σήμα ή να βρούμε τον μετασχηματισμό Fourier κάποιων δεδομένων ή να μετατρέψουμε ένα σήμα σε μια μορφή πιο κατάλληλη για επεξεργασία και ανάλυση της πληροφορίας που εμπεριέχει, αποτελούν παραδείγματα της ψηφιακής επεξεργασίας σήματος. Αυτή χρησιμοποιείται όλο και περισσότερο σε πολλές περιοχές εφαρμογών όπου παραδοσιακά χρησιμοποιούνταν αναλογικές μορφές επεξεργασίας, αλλά και σε νέες εφαρμογές στις οποίες οι αναλογικές μέθοδοι είναι δύσκολο ή και αδύνατο να χρησιμοποιηθούν. Το γεγονός αυτό οφείλεται στα πλεονεκτήματα που παρουσιάζει η ψηφιακή επεξεργασία σήματος.

Υπάρχουν πολλοί λόγοι για τους οποίους θα προτιμούσαμε την ψηφιακή επεξεργασία ενός σήματος έναντι της αναλογικής. Πρώτον, ένα ψηφιακό προγραμματιζόμενο σύστημα παρουσιάζει μεγάλη ευελιξία στην τροποποίηση των πράξεων ψηφιακής επεξεργασίας με μια απλή μετατροπή του προγράμματος. Μια τέτοια τροποποίηση ενός αναλογικού συστήματος συνεπάγεται την επανασχεδίαση του κυκλώματος και τον συνεπακόλουθο έλεγχο και επιβεβαίωση (testing and verification) της ορθής λειτουργίας του.

Η ακρίβεια (accuracy) παίζει επίσης πολύ σπουδαίο ρόλο. Οι ανοχές των στοιχείων των αναλογικών κυκλωμάτων καθιστούν δύσκολο τον προσδιορισμό της ακρίβειας ενός αναλογικού συστήματος επεξεργασίας. Στην περίπτωση ενός ψηφιακού συστήματος, ο έλεγχος των προδιαγραφών από άποψη ακρίβειας είναι πολύ πιο εύκολος.

Τα ψηφιακά σήματα αποθηκεύονται σε μαγνητικά ή οπτικά μέσα (π.χ. μαγνητικούς ή οπτικούς δίσκους, ταινίες, κά) χωρίς υποβάθμιση της πιστότητάς τους, πέραν αυτής που έχει υπεισέλθει εξαιτίας της διαδικασίας μετατροπής τους από αναλογικά σε ψηφιακά. Έτσι, δίνεται η δυνατότητα μεταφοράς και επεξεργασίας τέτοιων σημάτων σε μηπραγματικό χρόνο. Επιπλέον, δίνεται η δυνατότητα εφαρμογής πιο περίπλοκων αλγορίθμων επεξεργασίας σήματος. Συνήθως η υλοποίηση μαθηματικών πράξεων μεγάλης ακρίβειας είναι δύσκολο να γίνει σε σήματα τα οποία βρίσκονται σε αναλογική μορφή, πράγμα όμως που είναι συνηθισμένο και εύκολο για ένα ψηφιακό σήμα το οποίο επεξεργαζόμαστε με έναν υπολογιστή και με κατάλληλο λογισμικό.

Σε πολλές περιπτώσεις, η ψηφιακή επεξεργασία ενός σήματος είναι χαμηλότερου κόστους από την αντίστοιχη αναλογική. Αυτό μπορεί να οφείλεται είτε στο ότι το υλικό (hardware) σήμερα είναι φθηνότερο, είτε στην ευελιξία που παρέχεται λόγω της ψηφιακής υλοποίησης.

Αποτέλεσμα των πλεονεκτημάτων της ψηφιακής επεξεργασίας σήματος είναι η διαρκώς αυξανόμενη χρήση της σε όλο και περισσότερους τομείς εφαρμογών, όπως στην επεξεργασία ομιλίας, στη μετάδοση σήματος σε τηλεφωνικά κανάλια, στη σεισμολογία, στη γεωφυσική, στην ιατρική, στην εξερεύνηση του διαστήματος, στη μετεωρολογία, κα.

Φυσικά, η ψηφιακή επεξεργασία σήματος έχει και τα όριά της, τα οποία οφείλονται στους περιορισμούς που τίθενται στην ταχύτητα λειτουργίας των μετατροπέων αναλογικού σήματος σε ψηφιακό, καθώς και στους ίδιους τους ψηφιακούς επεξεργαστές σήματος. Έτσι, σήματα με εξαιρετικά μεγάλο εύρος συχνοτήτων, για παράδειγμα σήματα με εύρος συχνοτήτων της τάξεως των 100 MHz, υφίστανται επεξεργασία ακόμη και σήμερα με αναλογικές μεθόδους.

3.3 Τύποι σημάτων

Τα σήματα ταξινομούνται σε δύο μεγάλες κατηγορίες: στα σήματα συνεχούς χρόνου και στα σήματα διακριτού χρόνου. Συνήθως, ως ανεξάρτητη μεταβλητή χρησιμοποιείται ο χρόνος, χωρίς όμως να αποκλείεται η ανεξάρτητη μεταβλητή να είναι κάποιο άλλο φυσικό μέγεθος, όπως για παράδειγμα η απόσταση, η θερμοκρασία ή η πίεση. Παρ' όλα αυτά έχει επικρατήσει να μιλάμε για σήματα διακριτού χρόνου.

Στα σήματα **συνεχούς χρόνου** (continuous time) η ανεξάρτητη μεταβλητή είναι συνεχής, δηλαδή τα σήματα αυτά ορίζονται για οποιαδήποτε τιμή της ανεξάρτητης μεταβλητής. Η εξαρτημένη μεταβλητή, δηλαδή το πλάτος (amplitude) του σήματος, είναι και αυτή συνεχής. Γι' αυτό και τα σήματα αυτά αναφέρονται και ως σήματα συνεχούς χρόνου συνεχούς πλάτους ή αναλογικά σήματα. Παραδείγματα τέτοιων σημάτων είναι η ομιλία ως συνάρτηση του χρόνου ή η ατμοσφαιρική πίεση ως συνάρτηση του ύψους. Ένα αναλογικό σήμα περιγράφεται ως μια συνάρτηση x(t), όπου t πραγματικός αριθμός.

Στα σήματα **διακριτού χρόνου** (discrete time) η ανεξάρτητη μεταβλητή είναι διακριτή, δηλαδή τα σήματα αυτά ορίζονται μόνο για συγκεκριμένες τιμές της ανεξάρτητης μεταβλητής. Με άλλα λόγια, η ανεξάρτητη μεταβλητή παίρνει τιμές από ένα διακριτό σύνολο τιμών. Η εξαρτημένη μεταβλητή, δηλαδή το πλάτος του σήματος, είναι συνεχής. Γι' αυτό και τα σήματα αυτά αναφέρονται και ως σήματα διακριτού χρόνου συνεχούς πλάτους . Παραδείγματα τέτοιων σημάτων είναι ο δείκτης Dow-Jones ως συνάρτηση του χρόνου (π.χ. ανά ημέρα) ή το κατά κεφαλήν εισόδημα ως συνάρτηση του τόπου διαμονής. Στη περίπτωση που και η εξαρτημένη μεταβλητή παίρνει διακριτές τιμές, τότε μιλάμε για σήματα διακριτού χρόνου διακριτού πλάτους ή ψηφιακά σήματα. Ένα σήμα διακριτού χρόνου περιγράφεται ως x(n), όπου n ακέραιος. Πρόκειται για μία ακολουθία (sequence) αριθμών, γι' αυτό συχνά αναφερόμαστε στο σήμα αυτό και ως ακολουθία .

3.3.1. Η έννοια της συχνότητας στα σήματα

Η έννοια της συχνότητας είναι βασική και γνωστή σε όλους μας. Την έχουμε συναντήσει στον ραδιοφωνικό δέκτη που χρησιμοποιούμε ή στο στερεοφωνικό σύστημα που έχουμε ή στο φίλτρο που πρέπει να βάλουμε στην φωτογραφική μας μηχανή. Από την Φυσική γνωρίζουμε ότι η συχνότητα σχετίζεται με ένα τύπο περιοδικής κίνησης, η οποία ονομάζεται αρμονική ταλάντωση, και η οποία περιγράφεται από ημιτονοειδείς συναρτήσεις. Η έννοια της συχνότητας σχετίζεται άμεσα με την έννοια του χρόνου, αφού η διάσταση αυτής είναι το αντίστροφο του χρόνου. Κατά συνέπεια, η φύση του χρόνου (συνεχής ή διακριτή) αναμένεται να επηρεάζει την φύση της συχνότητας.

Ημιτονοειδή σήματα συνεχούς χρόνου

Μία απλή αρμονική ταλάντωση περιγράφεται μαθηματικά από το ημιτονοειδές σήμα συνεχούς χρόνου:

$$x_a(t)=A*\cos(\Omega t+\theta),\ -\infty<t<\infty \quad (3.1)$$

όπου A το πλάτος (amplitude) του ημιτονοειδούς, Ωη συχνότητα σε ακτίνια ανά δευτερόλεπτο (rad/s) και θ η φάση σε ακτίνια Η σχέση (3.1) μπορεί να γραφεί και ως:

$$x_a(t)=A*\cos(2\pi Ft+\theta),\ -\infty<t<\infty \quad (3.2)$$

όταν θέσουμε

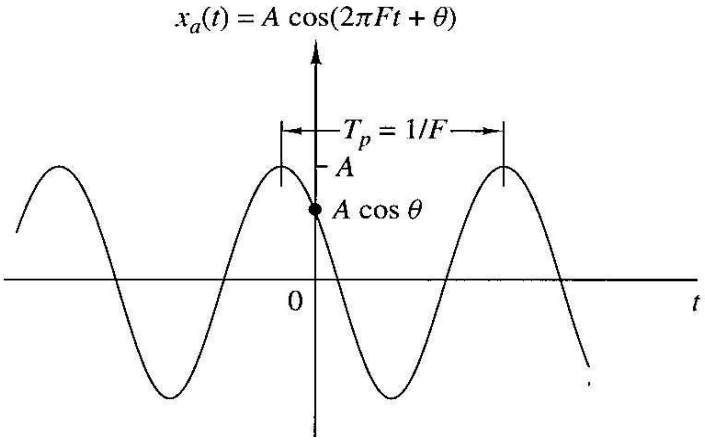
$$\Omega=2\pi F \quad (3.3)$$

όπου F η συχνότητα σε κύκλους ανά δευτερόλεπτο ή hertz (Hz). Το αναλογικό αυτό σήμα παρουσιάζει τις εξής ιδιότητες:

Είναι περιοδικό: Πράγματι, για οποιαδήποτε τιμή της συχνότητας F, η συνάρτηση $x_a(t)$ είναι περιοδική, δηλαδή $x_a(t+Tp)=x_a(t)$, όπου $Tp=1/F$ είναι η βασική περίοδος του ημιτονοειδούς σήματος.

Για διαφορετικές συχνότητες έχουμε διαφορετικά σήματα.

Αύξηση της συχνότητας F συνεπάγεται αντίστοιχη αύξηση του ρυθμού ταλάντωσης του σήματος, δηλαδή περισσότερες περίοδοι εμπεριέχονται σε ένα συγκεκριμένο χρονικό διάστημα.



Εικόνα 3.3.1 Ημιτονοειδές σήμα συνεχούς χρόνου

Σημαντική Παρατήρηση

Η συχνότητα είναι από τη φύση της μια θετική ποσότητα. Αυτό είναι προφανές, αφού η συχνότητα σε ένα περιοδικό σήμα εκφράζει τον αριθμό των κύκλων στην μονάδα του χρόνου. Σε ορισμένες περιπτώσεις όμως, για λόγους ευκολίας από μαθηματικής απόψεως, απαιτείται η εισαγωγή αρνητικών συχνοτήτων. Αυτό γίνεται κατανοητό αν θυμηθούμε ότι το ημιτονοειδές σήμα (σχέση 3.1) μπορεί να γραφεί και ως:

$$x_{\acute{a}}(t) = A\cos(\grave{U}t+\grave{e}) = \frac{A}{2}e^{j(\grave{U}t+\grave{e})} + \frac{A}{2}e^{-j(\grave{U}t+\grave{e})} \qquad (3.4)$$

βασιζόμενοι στην ταυτότητα του Euler $e^{\pm j\varphi}=\cos\varphi\pm j\sin\varphi$. Παρατηρούμε λοιπόν ότι το ημιτονοειδές σήμα μπορεί να προέλθει από την πρόσθεση δύο συζυγών μιγαδικών εκθετικών σημάτων ίσου πλάτους. Τα μιγαδικά εκθετικά σήματα συνηθίζουμε να τα παριστάνουμε ως διανύσματα πάνω στο μιγαδικό επίπεδο, τα οποία ονομάζουμε φάσορες (phasors). Οι φάσορες της σχέσης (3.4) περιστρέφονται με γωνιακές συχνότητες ±Ω rad/sec. Η θετική συχνότητα αντιστοιχεί σε ομοιόμορφη περιστροφή του φάσορα με φορά αντίθετη της κίνησης των δεικτών του ρολογιού (αριστερόστροφη περιστροφή). Κατά συνέπεια, η αρνητική συχνότητα αντιστοιχεί σε ομοιόμορφη περιστροφή του φάσορα κατά την φορά της κίνησης των δεικτών του ρολογιού (δεξιόστροφη περιστροφή).

Όπως λοιπόν αναφέραμε, για λόγους ευκολίας από μαθηματικής απόψεως θα χρησιμοποιούμε θετικές και αρνητικές συχνότητες.. Αυτό σημαίνει ότι η περιοχή συχνοτήτων των αναλογικών σημάτων θα είναι $-\infty<F<\infty$.

Ημιτονοειδή σήματα διακριτού χρόνου

Ένα ημιτονοειδές σήμα διακριτού χρόνου μπορεί να εκφραστεί ως:

$$x(n)=A\cos(\omega n+\theta),\ -\infty<n<\infty \qquad (3.5)$$

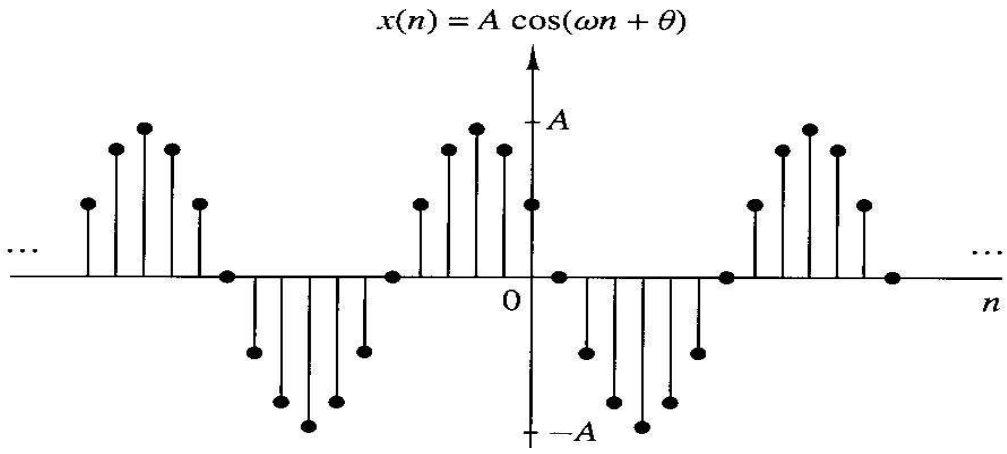
όπου n ακέραια μεταβλητή, η οποία αντιπροσωπεύει τον αριθμό (την θέση) του δείγματος, Α το πλάτος του σήματος, ω η συχνότητα του σήματος σε ακτίνια ανά δείγμα και θ η φάση σε ακτίνια . Η σχέση (3.5) μπορεί να γραφεί και ως:

$$x(n)=A\cos(2\pi fn+\theta),\ -\infty<n<\infty \qquad (3.6)$$

όταν θέσουμε

$$\omega=2\pi f \qquad (3.7)$$

όπου f η συχνότητα σε κύκλους ανά δείγμα.



Εικόνα 3.3.1 Ημιτονοειδή σήματα διακριτού χρόνου

Σε αντίθεση με ένα ημιτονοειδές σήμα συνεχούς χρόνου, ένα ημιτονοειδές διακριτού χρόνου παρουσιάζει τις ακόλουθες ιδιότητες:

Τα ημιτονοειδή σήματα διακριτού χρόνου των οποίων οι συχνότητες διαφέρουν κατά ακέραιο πολλαπλάσιο του 2π είναι ίδια (ταυτίζονται).

Ο μέγιστος ρυθμός ταλάντωσης ενός ημιτονοειδούς διακριτού χρόνου επιτυγχάνεται για :

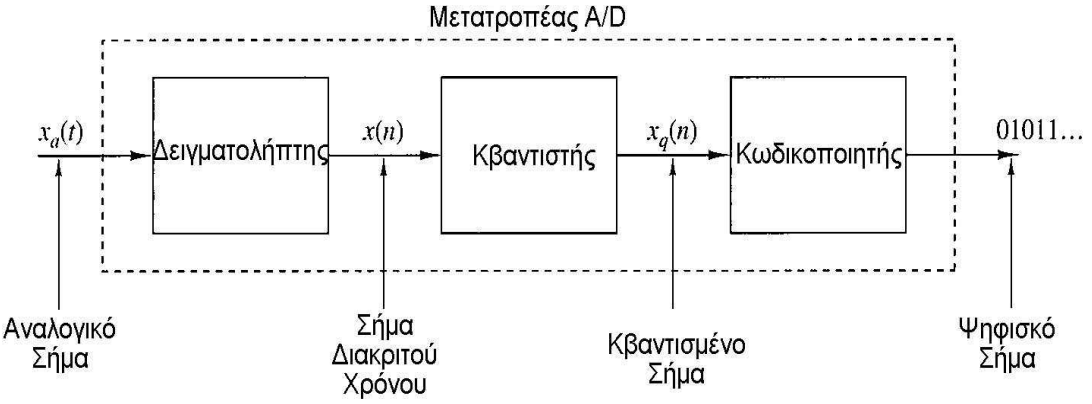
$\omega=\pi$ (ή $\omega=-\pi$)

ή ισοδύναμα για: $f=\frac{1}{2}$ (ή $f=-\frac{1}{2}$) (3.8)

3.3.2. Μετατροπή σήματος απο ψηφιακό σε αναλογικό και απο αναλογικό σε ψηφιακό

Τα περισσότερα σήματα που παρουσιάζουν πρακτικό ενδιαφέρον, όπως για παράδειγμα η ομιλία, τα βιολογικά σήματα, τα σεισμικά σήματα, κ.ά, είναι αναλογικά. Για να επεξεργαστούμε αναλογικά σήματα με ψηφιακά μέσα, απαιτείται η μετατροπή αυτών σε ψηφιακή μορφή, δηλαδή η μετατροπή τους σε μία ακολουθία αριθμών πεπερασμένης ακρίβειας. Η διαδικασία αυτή ονομάζεται μετατροπή αναλογικού-σε-ψηφιακό (analog-todigital conversion, A/D) και τα αντίστοιχα κυκλώματα ονομάζονται μετατροπείς αναλογικού-σε-ψηφιακό (analog-to-digital converters, ADCs). Η αντίστροφη διαδικασία της μετατροπής ενός ψηφιακού σήματος σε αναλογικό είναι γνωστή ως μετατροπή ψηφιακούσε-αναλογικό (digital-to-analog conversion, D/A) και γίνεται με την βοήθεια κυκλωμάτων τα οποία ονομάζονται μετατροπείς ψηφιακού-σε-αναλογικό (digital-to-analog converters, DACs).

Η διαδικασία της μετατροπής ενός αναλογικού σήματος σε ψηφιακό γίνεται σε τρία στάδια, όπως δείχνεται στο Σχήμα 3.3.2



3.3.2 Δειγματοληψία (sampling): Αυτή είναι η διαδικασία μετατροπής ενός σήματος συνεχούς χρόνου σε ένα σήμα διακριτού χρόνου, παίρνοντας δείγματα του σήματος συνεχούς χρόνου σε διακριτές στιγμές του χρόνου. Έτσι, αν xα(t) είναι η είσοδος στον δειγματολήπτη, τότε η έξοδος αυτού είναι xα(nT)≡x(n), όπου T η περίοδος δειγματοληψίας.

1 Κβάντιση (quantisation): Πρόκειται για την διαδικασία μετατροπής ενός σήματος διακριτού χρόνου συνεχών τιμών σε ένα σήμα διακριτού χρόνου διακριτών τιμών (ψηφιακό). Το κάθε δείγμα του σήματος αντιπροσωπεύεται από μία τιμή η

οποία επιλέγεται από ένα πεπερασμένο σύνολο πιθανών τιμών. Η διαφορά μεταξύ του αρχικού μη-κβαντισμένου δείγματος x(n) και της κβαντισμένης εξόδου xq(n) αποτελεί το λεγόμενο σφάλμα κβάντισης.
2 Κωδικοποίηση (coding): Κατά την διαδικασία της κωδικοποίησης, κάθε διακριτή τιμή xq(n) αντιπροσωπεύεται από έναν αριθμό αποτελούμενο από b-bits.

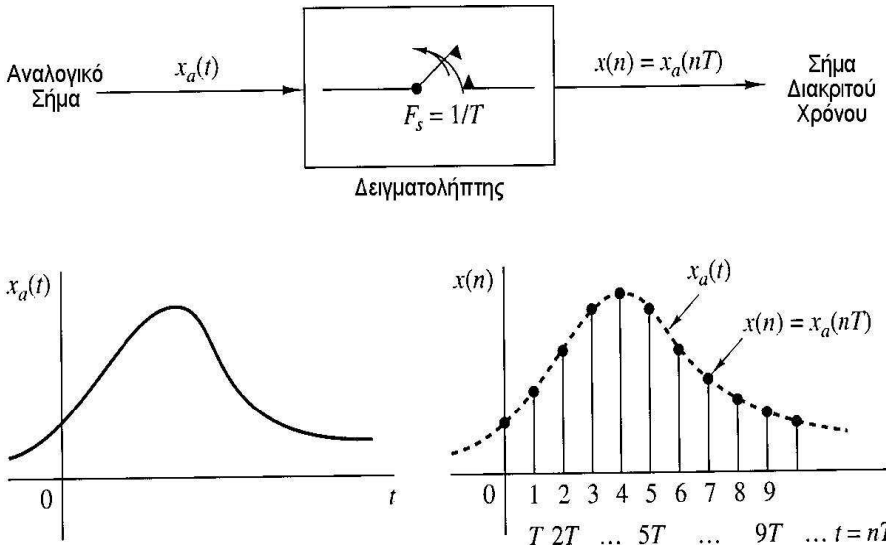
Ας εξετάσουμε ξεχωριστά καθένα από αυτά τα τρία στάδια:

Δειγματοληψία Αναλογικών Σημάτων

Η δειγματοληψία ενός αναλογικού σήματος xα(t) επιτυγχάνεται παίρνοντας δείγματα αυτού ανά T δευτερόλεπτα, όπως δείχνεται στο Σχήμα 3.3.3 Η διαδικασία αυτή περιγράφεται από την σχέση:

$$x(n) = x\alpha(nT), \quad -\infty<n<\infty \qquad (3.9)$$

όπου x(n) είναι το σήμα διακριτού χρόνου που προκύπτει. Το χρονικό διάστημα T μεταξύ των διαδοχικών δειγμάτων ονομάζεται περίοδος δειγματοληψίας και το αντίστροφο του $\frac{1}{T} = Fs$ αποτελεί τον ρυθμό δειγματοληψίας (sampling rate) σε δείγματα ανά δευτερόλεπτο ή αλλιώς την συχνότητα δειγματοληψίας (sampling frequency) σε Hz.



Εικ. 3.3.3 Οι μεταβλητές χρόνου t και n για τα σήματα συνεχούς χρόνου και διακριτού χρόνου αντίστοιχα, συνδέονται γραμμικά μέσω της περιόδου δειγματοληψίας T ή ισοδύναμα μέσω του ρυθμού δειγματοληψίας $Fs = \frac{1}{T}$ ώς εξής:

$$t = nT = \frac{n}{F_s} \qquad (3.10)$$

Επομένως, αναμένουμε να υπάρχει κάποια σχέση που να συνδέει την συχνότητα F (ή Ω) των αναλογικών σημάτων με την συχνότητα f (ήω) των σημάτων διακριτού χρόνου. Για να βρούμε αυτή την σχέση, ξεκινούμε από την (3.9) και αντικαθιστούμε το xα(t) με την συνάρτηση του ημιτονοειδούς σήματος της εξίσωσης (3.2). Έτσι έχουμε:

$$x(n)= x_a(nT)=A\cos(2\pi FnT+\theta)=A\cos(2\pi n\frac{F}{F_s}+\theta) \qquad (3.11)$$

Συγκρίνοντας την (1.11) με την αντίστοιχη σχέση (1.6) για το ημιτονοειδές σήμα διακριτού χρόνου, διαπιστώνουμε ότι:

$$f = \frac{F}{Fs} \qquad (3.12)$$

ή ισοδύναμα:

$$\omega=\Omega T \qquad (3.13)$$

Από την σχέση (3.12) παρατηρούμε ότι η συχνότητα f είναι μια κανονικοποιημένη ή σχετική συχνότητα (normalized or relative frequency). Κατά συνέπεια, για να προσδιορίσουμε την F Hz, όταν μας δίνεται η f, πρέπει απαραίτητα να γνωρίζουμε την συχνότητα δειγματοληψίας Fs.

Η περιοχή συχνοτήτων F ή Ω των ημιτονοειδών συνεχούς χρόνου είναι:

$$-\infty<F<\infty \text{ ή } -\infty<\Omega<\infty \qquad (3.14)$$

Για τα ημιτονοειδή διακριτού χρόνου είδαμε ότι μόνον οι συχνότητες f ή ω που βρίσκονται στο διάστημα:

$$-\frac{1}{2}\le f\le\frac{1}{2} \quad \text{ή} \quad -\pi\le\omega\le\pi \qquad (3.15)$$

είναι διαφορετικές. Αντικαθιστώντας στην τελευταία σχέση τις μεταβλητές f και ω με τις ισοδύναμές τους από τις σχέσεις (3.12) και (3.13), βρίσκουμε ότι η συχνότητα του ημιτονοειδούς συνεχούς χρόνου, όταν αυτό δειγματοληπτείται με ρυθμό Fs=1/T, θα

πρέπει να βρίσκεται στην περιοχή:

$$-\frac{1}{2T} = -\frac{F_s}{2} \le F \le \frac{F_s}{2} = \frac{1}{2T}$$

ή $$-\frac{\delta}{T} = -\delta F_s \le \dot{U} \le \delta F_s = \frac{\delta}{\hat{O}}$$ (3.16) - (3.17)

Παρατηρούμε επομένως ότι η βασική διαφορά μεταξύ των σημάτων συνεχούς χρόνου και διακριτού χρόνου βρίσκεται στην περιοχή τιμών των μεταβλητών συχνότητας F και f ή Ω και ω. Η περιοδική δειγματοληψία ενός σήματος συνεχούς χρόνου οδηγεί στην απεικόνιση της απείρου εύρους περιοχής συχνοτήτων F (ήΩ), στην πεπερασμένου εύρους περιοχή συχνοτήτων f (ή ω). Και αφού η μέγιστη συχνότητα σ' ένα σήμα διακριτού χρόνου είναι $f = \frac{1}{2}$ ή ω=π, έπεται ότι για ένα ρυθμό δειγματοληψίας Fs, η αντίστοιχη μέγιστη F ή Ω θα ισούται με:

$$F_{max} = \frac{F_s}{2} = \frac{1}{2T} \quad \text{ή} \quad \dot{U}_{max} = \delta F_s = \frac{\delta}{T}$$ (3.18)

Με άλλα λόγια, η δειγματοληψία εισάγει μια ασάφεια, αφού η μέγιστη συχνότητα ενός σήματος συνεχούς χρόνου, η οποία μπορεί να αναπαρασταθεί σωστά, είναι Fmax=Fs/2, όταν λαμβάνονται δείγματα του σήματος αυτού με ρυθμό Fs=1/T. Πριν όμως προχωρήσουμε σε κάποια παραδείγματα που θα μας δείξουν το τι συμβαίνει όταν οι συχνότητες του αναλογικού σήματος είναι μεγαλύτερες από Fs/2, ας δούμε το θεώρημα της δειγματοληψίας, το οποίο απαντά στο εξής ερώτημα: Ποιός ο ρυθμός δειγματοληψίας Fs για την σωστή αναπαράσταση ενός αναλογικού σήματος το οποίο μας δίνεται; Δηλαδή, πόσο συχνά πρέπει να παίρνουμε δείγματα ώστε να έχουμε ένα πιστό αντίγραφο του αναλογικού σήματος; Η απάντηση σ' αυτό το ερώτημα δόθηκε αρχικά από τον Nyquist (1928) και στη συνέχεια από τον Shannon (1949) και αποτελεί το λεγόμενο **θεώρημα δειγματοληψίας** ή **θεώρημα του Shannon,** διατυπώνεται δε ως εξής:

Η συχνότητα Fs, με την οποία λαμβάνονται τα δείγματα ενός σήματος, πρέπει να είναι τουλάχιστον διπλάσια από την υψηλότερη συχνότητα Fmax που περιέχεται στο σήμα,

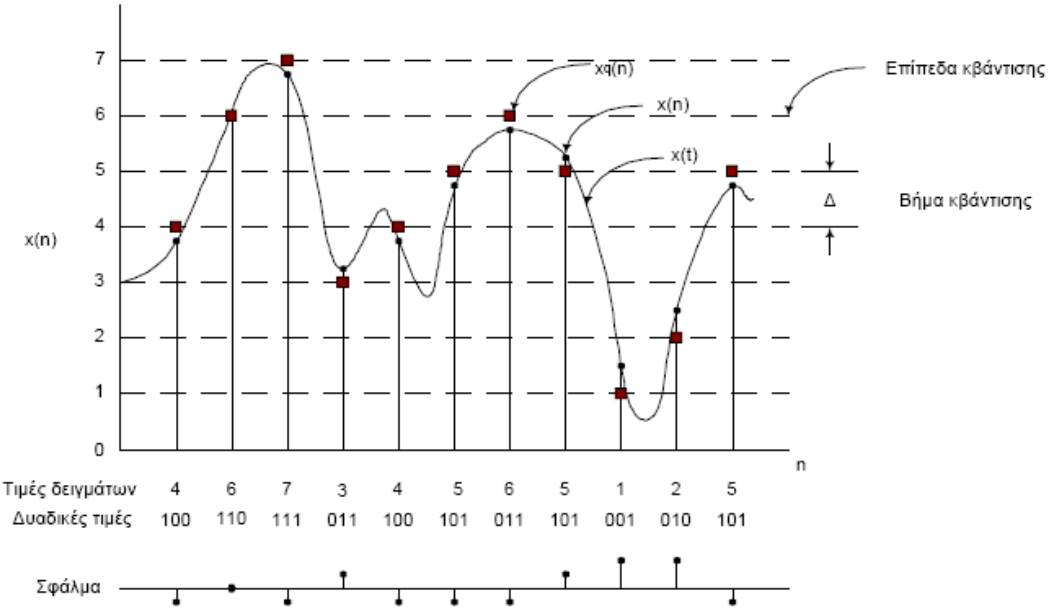
δηλαδή Fs ≥2Fmax (3.19)

Με άλλα λόγια, το θεώρημα δειγματοληψίας μας λέει πως για να μη χαθεί πληροφορία θα πρέπει να παίρνουμε τουλάχιστον δύο δείγματα ανά περίοδο (της υψηλότερης συχνότητας του σήματος). Για παράδειγμα, αν θελήσουμε να

ψηφιοποιήσουμε ένα σήμα ομιλίας και χρησιμοποιούμε μικρόφωνο το οποίο λειτουργεί για συχνότητες μεταξύ 300 Hz και 3 kHz, τότε η μικρότερη συχνότητα δειγματοληψίας που πρέπει να χρησιμοποιήσουμε είναι 6 kHz. Ας δούμε τώρα τι θα συμβεί αν το θεώρημα δειγματοληψίας δεν γίνει σεβαστό.

Κωδικοποίηση των Κβαντισμένων Δειγμάτων

Κατά την διαδικασία της κωδικοποίησης σ' ένα μετατροπέα αναλογικού-σε-ψηφιακό, ένας μοναδικός δυαδικός αριθμός εκχωρείται σε κάθε επίπεδο κβάντισης. Αν έχουμε L επίπεδα κβάντισης, χρειαζόμαστε τουλάχιστον L διαφορετικούς δυαδικούς αριθμούς. Με ένα μήκος λέξης b bits μπορούμε να έχουμε 2^b διαφορετικούς δυαδικούς αριθμούς. Άρα, πρέπει $2^b \geq L$ ή ισοδύναμα $b \geq \log 2L$. Στο παράδειγμα του Σχήματος 1.9 χρησιμοποιήσαμε έναν κωδικοποιητή με b=3 bits. Στο εμπόριο υπάρχουν διαθέσιμοι μετατροπείς A/D με ακρίβεια μέχρι και b=16 bits. Χαρακτηριστική είναι η περίπτωση των μουσικών CD's όπου χρησιμοποιούνται μετατροπείς A/D ακρίβειας 16 bits.



3.3.2 Κβάντιση και κωδικοποίηση σήματος

3.4 Βασικά σήματα διακριτού χρόνου

Τα σήματα που περιγράφονται στη συνέχεια θεωρούνται ως τα βασικά (στοιχειώδη) σήματα διακριτού χρόνου.

α) Μοναδιαίο δείγμα (unit sample) ή μοναδιαία κρουστική ακολουθία (unit impulse sequence): Είναι το πλέον βασικό σήμα διακριτού χρόνου το οποίο ορίζεται ως:

$$\ddot{a}(n)=\begin{cases} 1, & n=0 \\ 0, & n\neq 0 \end{cases} \qquad (3.20)$$

β) Μοναδιαία βηματική ακολουθία (unit step sequence): Ορίζεται ως:

$$u(n)=\begin{cases} 1, & n\geq 0 \\ 0, & n<0 \end{cases} \qquad (3.21)$$

γ) Σταθερή ακολουθία (constant sequence):

$$x(n) = A \qquad -\infty < n < \infty \qquad (3.22)$$

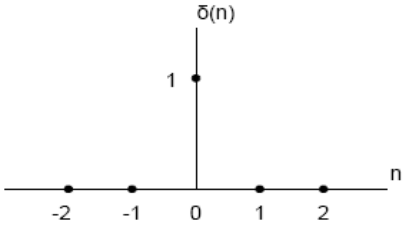
δ) Γραμμική ακολουθία (linear sequence):

$$x(n) = An \qquad -\infty < n < \infty \qquad (3.23)$$

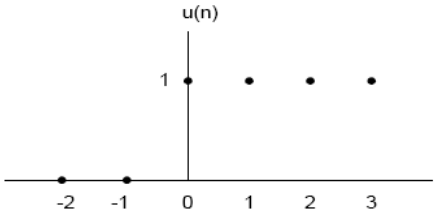
ε) Εκθετική ακολουθία (exponential sequence):

$$x(n) = a^n \qquad -\infty < n < \infty \quad (3.24)$$

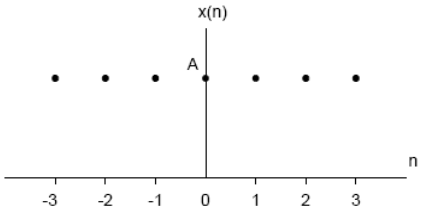
Η ακολουθία αυτή παρουσιάζει ιδιαίτερο ενδιαφέρον. Η μορφή της εξαρτάται από την τιμή του a. Έτσι, αν a πραγματικός αριθμός, τότε αυτή είναι φθίνουσα για $|a|<1$ (Σχήμα 1.14α,β) και αύξουσα για $|a|>1$. Αν a μιγαδικός αριθμός, δηλαδή $a=re^{j\omega}$, τότε $x(n)= r^n e^{j\omega n}$ ή $x(n)= r^n [\cos(\omega n) + j \sin(\omega n)]$. Για r=1 το πραγματικό και φανταστικό μέρος είναι αντίστοιχα μια συνημιτονική και μια ημιτονική ακολουθία σταθερού πλάτους της μορφής του Σχήματος 3.4α. Για r<1 έχουμε φθίνουσες ημιτονικές ακολουθίες της μορφής του Σχήματος 3.4β και για r>1 έχουμε αύξουσες ημιτονικές ακολουθίες της μορφής του Σχήματος 3.4γ.



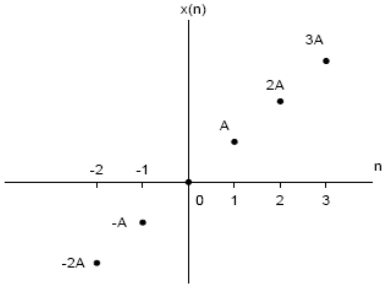
Κρουστική ακολουθία



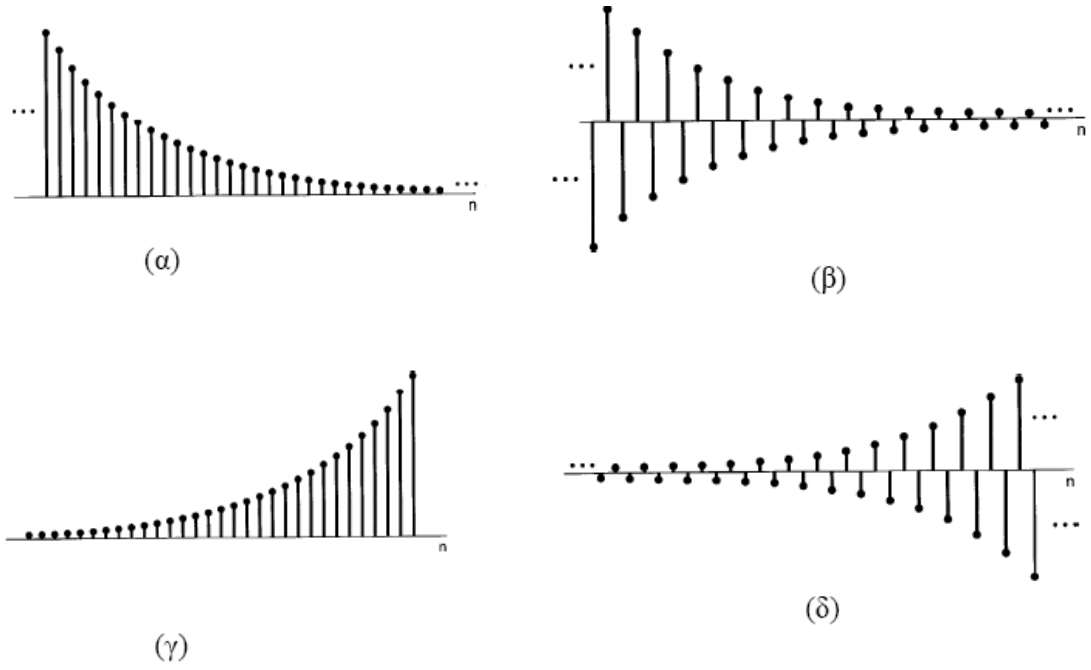
Βηματική ακολουθία



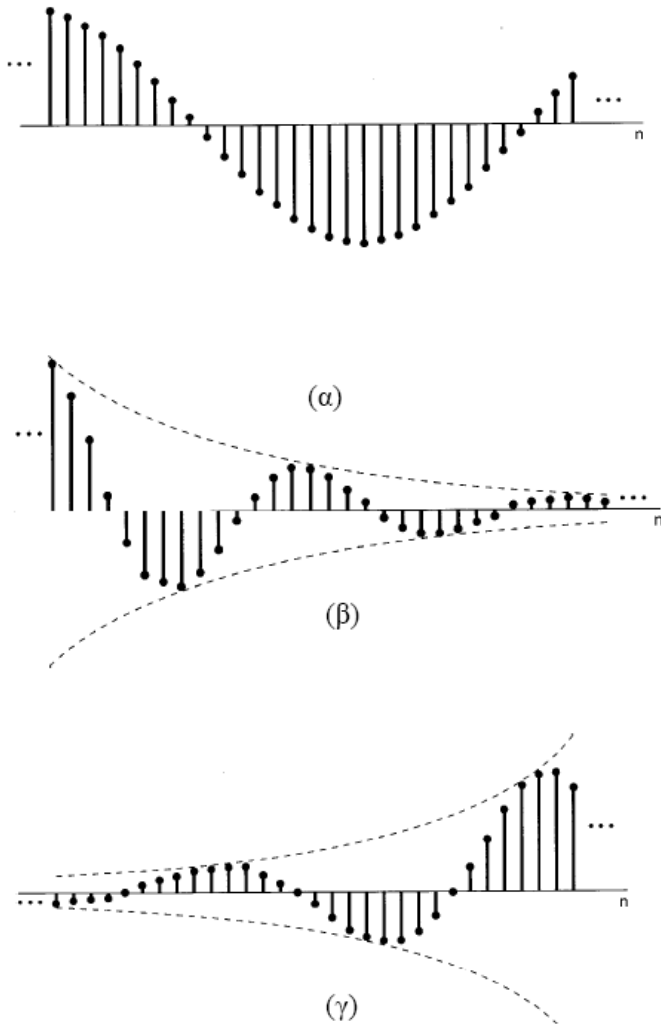
Σταθερή ακολουθία



Γραμμική ακολουθία



Σχήμα 3.4. Εκθετική ακολουθία $x(n) = a^n$ για a πραγματικό και (α) 0<a<1, (β) –1<a<0, (γ) a>1 και (δ) a<-1



3.4.1 Γραφική αναπαράσταση του πραγματικού ή φανταστικού μέρους της εκθετικής ακολουθίας x(n) = an για a μιγαδικό (a=rejω), όπου (α) r=1, (β) r<1 και (γ) r>1

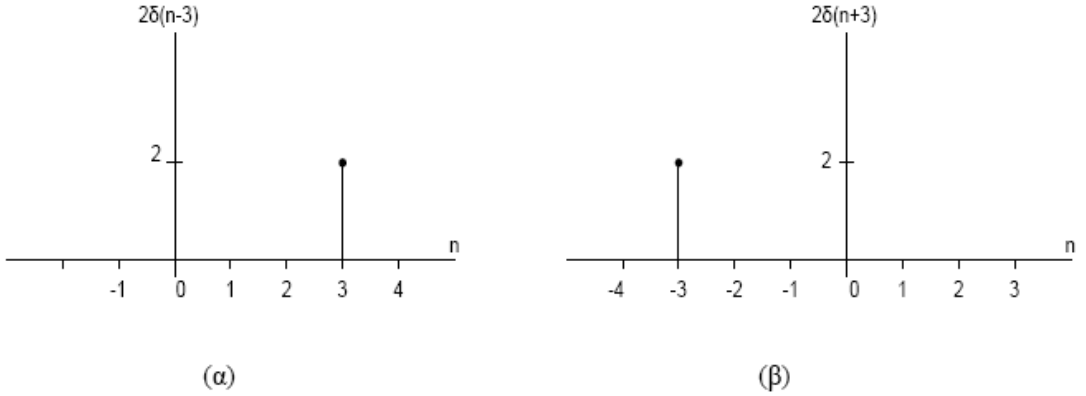
3.4.1. Στοιχειώδεις πράξεις

Ολίσθηση σημάτων διακριτού χρόνου

Η μαθηματική περιγραφή της ολίσθησης και η κατανόηση αυτής είναι βασικής σημασίας. Για παράδειγμα, η ολίσθηση της μοναδιαίας κρουστικής κατά no μονάδες (δείγματα) ορίζεται ως:

$$\delta(n-n_o)=\begin{cases}1, & n=n_o\\ 0, & n\neq n_o\end{cases} \qquad (3.25)$$

Στο Σχήμα 3.4.1 δείχνονται οι συναρτήσεις 2δ(n-3) και 2δ(n+3).

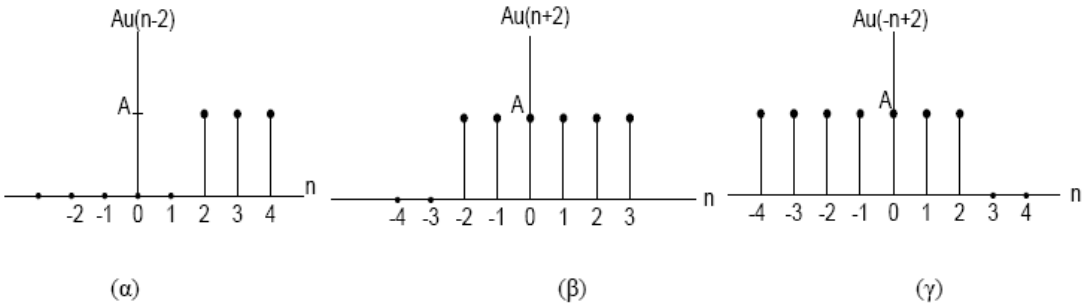


Σχήμα 3.4.1. Γραφικές παραστάσεις των μοναδιαίων ακολουθιών (α) 2δ(n-3) και (β) 2δ(n+3).

Με όμοιο τρόπο ορίζεται και η ολισθημένη κατά no μοναδιαία βηματική ακολουθία:

$$u(n-n_o)=\begin{cases}1, & n\geq n_o\\ 0, & n<n_o\end{cases} \qquad (3.26)$$

Στο Σχήμα 3.4.1.1 δείχνονται παραδείγματα ολίσθησης μιας βηματικής συνάρτησης κατά δύο δείγματα (no = 2)



Σχήμα 3.4.1.1. Γραφικές παραστάσεις των βηματικών ακολουθιών (α) Au(n-2), (β) Au(n+2) και (γ) Au(-n+2)

Παρατηρούμε ότι η μη μηδενική τιμή μιας κρουστικής βρίσκεται εκεί όπου το όρισμα της δ(*) γίνεται μηδέν. Όμοια, μία βηματική ακολουθία είναι μη μηδενική για εκείνες τις τιμές για τις οποίες το όρισμα της u(*) είναι μεγαλύτερο ή ίσο του μηδενός. Για παράδειγμα, μια κρουστική ακολουθία με πλάτος δείγματος 4 στη θέση n=3, εκφράζεται ως 4δ(n-3). Μια βηματική ακολουθία πλάτους –2 για όλες τις θετικές τιμές του n, καθώς και για n=0, εκφράζεται ως x(n)=-2u(n). Η κατοπτρική αυτής ως προς τον άξονα των τεταγμένων είναι η x(-n)=-2u(-n). Αυτή έχει πλάτος –2 για όλες τις αρνητικές τιμές του n, καθώς και για n=0. Η ολίσθηση αυτής κατά 4 θέσεις προς τα αριστερά θα μας δώσει την ακολουθία x(-n+4)=-2u(-n+4).

Είμαστε τώρα σε θέση να δούμε εύκολα ότι οι σχέσεις που συνδέουν την κρουστική και την βηματική ακολουθία είναι οι εξής

$$u(n) = \sum_{m=-\infty}^{n} \delta(m) \qquad (3.27)$$

$$\delta(n) = u(n) - u(n-1) \qquad (3.28)$$

Γενικά, η ακολουθία x(n-n0) είναι ένα αντίγραφο της x(n) το οποίο έχει υποστεί ολίσθηση. Για n0>0 έχουμε μια δεξιά ολίσθηση η οποία ισοδυναμεί με καθυστέρηση (delay) του σήματος, ενώ για n0<0 έχουμε μια αριστερή ολίσθηση η οποία ισοδυναμεί με προήγηση (advance) του σήματος.

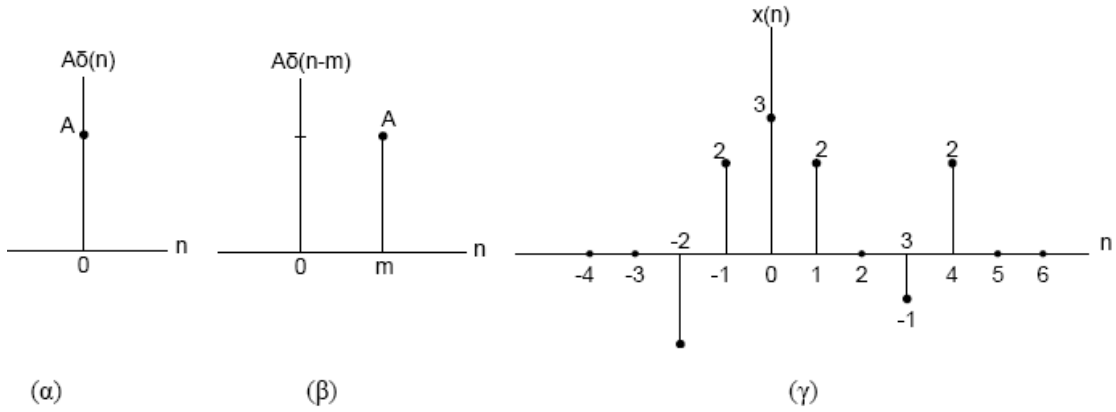
Γενική περιγραφή ακολουθίας

Οποιοδήποτε σήμα x(n) μπορεί να γραφεί ως άθροισμα ολισθημένων κρουστικών δειγμάτων πολλαπλασιασμένων με συντελεστές βάρους:

$$x(n)=\sum_{m=-\infty}^{+\infty} x(m)\ddot{a}(n-m) \qquad (3.29)$$

η ακολουθία x(n), με {x(n)}={... 0, 0, -2, 2, **3**, 2, 0, -1, 2, 0, 0, ...}, όπου με έντονη γραφή και υπογράμμιση σημειώνεται η χρονική στιγμή n=0 (στοιχείο 3), μπορεί να περιγραφεί ως:

x(n) = … -2δ(n+2) + 2δ(n+1) + 3δ(n) + 2δ(n-1) -δ(n-3) + 2δ(n-4) + … =
= … + x(-2)δ(n+2) + x(-1)δ(n+1) + x(0)δ(n) + x(1)δ(n-1) + x(3)δ(n-3) + x(4)δ(n-4) + …
και γενικά προκύπτει η σχέση (3.29)



Σχήμα 3.4.1.2 Η μοναδιαία κρουστική στην περιγραφή οποιουδήποτε σήματος διακριτού χρόνου
x(n):(α) Aδ(n), (β) Aδ(n-m), (γ) x(n)

3.5 Συστήματα διακριτού χρόνου

Ένα σύστημα διακριτού χρόνου είναι η οντότητα εκείνη που δέχεται μια είσοδο διακριτού χρόνου x(n) και παράγει μια έξοδο επίσης διακριτού χρόνου y(n) (Σχήμα 3.5). Τα συστήματα με τα οποία θα ασχοληθούμε στο βιβλίο αυτό έχουν δύο βασικά χαρακτηριστικά. Είναι γραμμικά (linear) και χρονικά αμετάβλητα (time-invariant). Θα αναφερόμαστε σ' αυτά με τον αγγλικό όρο LTI (Linear Time-Invariant) για λόγους συμβατότητας με την διεθνή βιβλιογραφία.

Σχήμα 3.5. Γενικό διάγραμμα συστήματος διακριτού χρόνου

Γραμμικό ονομάζεται ένα σύστημα το οποίο υπακούει στην **αρχ**ή **τη**ς **υπέρθεσης**. Συγκεκριμένα, εάν η είσοδος του συστήματος, το οποίο αρχικά βρισκόταν σε ηρεμία[1], αποτελείται από ένα γραμμικό συνδυασμό σημάτων, τότε η έξοδος του συστήματος (απόκριση) θα ισούται με το γραμμικό συνδυασμό των αποκρίσεων των επιμέρους σημάτων, σαν αυτά να είχαν εφαρμοσθεί το καθένα χωριστά. Μαθηματικά αυτό εκφράζεται ως εξής: αν y1(n) είναι η απόκριση του συστήματος στην είσοδο x1(n) και y2(n) είναι η απόκριση αυτού στην είσοδο x2(n), τότε η απόκριση του συστήματος στην είσοδο ax1(n) + bx2(n) θα είναι ay1(n) + by2(n), όπου a, b σταθερές.

[1] Αρχική ηρεμία σημαίνει ότι στο σύστημα δεν έχει εφαρμοστεί καμία διέγερση (είσοδος) πριν από την χρονική στιγμή n=n0, κατά την οποία εφαρμόστηκε η είσοδος x(n).

Ας εξετάσουμε την περίπτωση ενός γραμμικού και ενός μη γραμμικού συστήματος. Ένα παράδειγμα γραμμικού συστήματος είναι αυτό του οποίου η έξοδος ισούται με y(n)=x(n)-x(n-1). Για είσοδο x1(n), η έξοδος του συστήματος θα είναι y1(n)=x1(n)-x1(n1). Για είσοδο x2(n) η έξοδος του συστήματος θα είναι y2(n)=x2(n)-x2(n-1). Αν τώρα εφαρμόσουμε ως είσοδο x3(n) τον γραμμικό συνδυασμό των δύο προηγουμένων ακολουθιών εισόδου, δηλαδή x3(n)=a x1(n)+b x2(n), η έξοδος y3(n) του συστήματος θα ισούται με:

$$y3(n) = x3(n)-x3(n-1) = [a\ x1(n)+b\ x2(n)] – [a\ x1(n-1)+b\ x2(n-1)]$$

= a[x1(n) - x1(n-1)] + b[x2(n) - x2(n-1)] = a y1(n)+b y2(n)

άρα ισχύει η αρχή της υπέρθεσης.

Ένα παράδειγμα μη γραμμικού συστήματος είναι εκείνο το οποίο παράγει στην έξοδό του το τετράγωνο της εισόδου, δηλαδή y(n)=x(n)2
. Για είσοδο x1(n) η έξοδος του συστήματος θα είναι y1 (n)=x1 (n)2

Για είσοδο x2(n) η έξοδος του συστήματος θα είναι $y_2(n) = (x_2(n))^2$. Αν τώρα εφαρμοσθεί στην είσοδο το σήμα x3(n)=ax1(n) + bx2(n) η έξοδος θα είναι:

$$y_3(n) = (x_3(n))^2 = (ax_1(n) + bx_2(n))^2 = (ax_1(n))^2 + (bx_2(n))^2 + 2abx_1(n)x_2(n)$$
$$= a^2 y_1(n) + b^2 y_2(n) + 2abx_1(n)x_2(n) \neq ay_1(n) + by_2(n)$$

Χρονικά αμετάβλητο ονομάζεται ένα σύστημα του οποίου η συμπεριφορά και οι ιδιότητες δεν αλλάζουν με τον χρόνο. Αυτό σημαίνει ότι μια χρονική ολίσθηση της εισόδου θα αντιστοιχεί σε χρονική ολίσθηση της εξόδου. Με άλλα λόγια, εάν y(n) είναι η έξοδος ενός χρονικά αμετάβλητου συστήματος για είσοδο x(n), τότε y(n-n0) θα είναι η έξοδος αυτού για είσοδο x(n-n0).

Ευσταθές (stable) ονομάζεται ένα σύστημα εάν και μόνον εάν κάθε φραγμένη είσοδος παράγει μια φραγμένη έξοδο (Bounded Input Bounded Output, BIBO). Με άλλα λόγια, ένα τέτοιο σύστημα μας εξασφαλίζει ότι όσο η είσοδος παραμένει φραγμένη ($|x(n)| \leq Mx < \infty$), η έξοδος δεν θα απειρίζεται ($|y(n)| \leq My < \infty$) για όλα τα n, όπου Mx, My πεπερασμένοι αριθμοί. Σε διαφορετική περίπτωση το σύστημα ονομάζεται ασταθές (unstable).

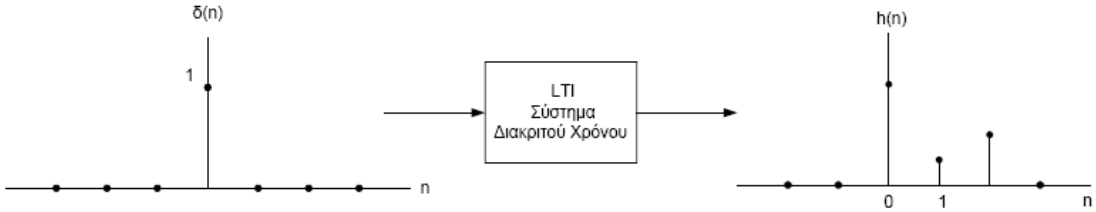
Αιτιατό (causal) σύστημα είναι εκείνο του οποίου η έξοδος, σε κάθε χρονική στιγμή, εξαρτάται μόνο από τις τιμές του σήματος εισόδου στην παρούσα χρονική στιγμή και σε προηγούμενες χρονικές στιγμές. Με άλλα λόγια, οι μεταβολές στην έξοδο (αποτέλεσμα) ενός τέτοιου συστήματος έπονται των μεταβολών της εισόδου.

Στη συνέχεια, θα εξετάσουμε τη σημασία της κρουστικής απόκρισης μονοδιάστατου συστήματος διακριτού χρόνου και θα δούμε ότι με τη βοήθειά της μπορούμε, μέσω της πράξης της συνέλιξης, να υπολογίσουμε την έξοδο ενός γραμμικού συστήματος διακριτού χρόνου για οποιαδήποτε είσοδο.

3.5.1. Κρουστική απόκριση συστήματος

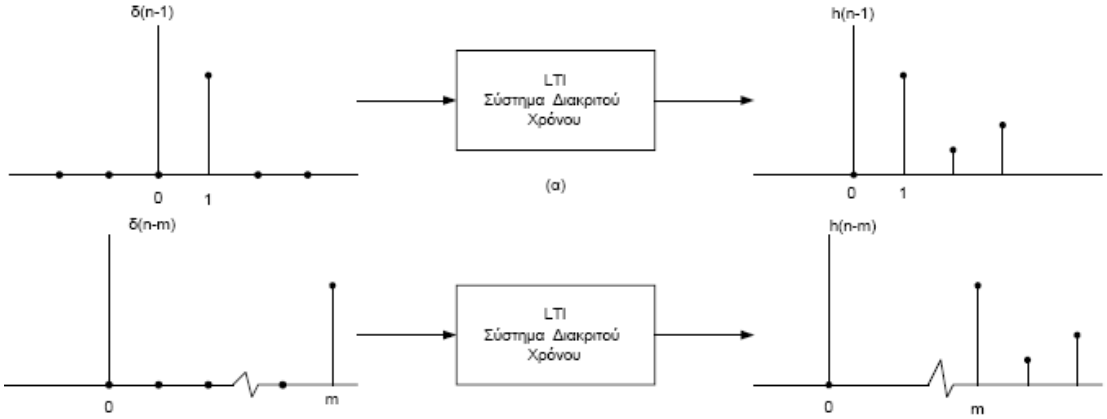
Ας θυμηθούμε τη μοναδιαία κρουστική ακολουθία δ(n). Αυτή έχει τιμή 1 για n=0 και τιμή 0 οπουδήποτε αλλού. Εφαρμόζουμε αυτό το σήμα στην είσοδο ενός LTI συστήματος διακριτού χρόνου, το οποίο αρχικά ηρεμεί, δηλαδή εφαρμόζουμε μια διέγερση την στιγμή n=0. Το σήμα εξόδου, το οποίο θα παρατηρηθεί μετά την στιγμή n=0, είναι χαρακτηριστικό του ίδιου του συστήματος. Αυτό το σήμα εξόδου αποτελεί την **κρουστική απόκριση,** h(n), του συστήματος. Η κρουστική απόκριση ονομάζεται και **φυσική απόκριση** του συστήματος. Ένα παράδειγμα κρουστικής απόκρισης

συστήματος διακριτού χρόνου δείχνεται στο Σχήμα 3.5.1



Σχήμα 3.5.1. Κρουστική απόκριση συστήματος διακριτού χρόνου

Αν εφαρμόζαμε την διέγερση την στιγμή n=1, τότε η απόκριση του συστήματος θα ήταν ίδια με την προηγούμενη, αλλά θα άρχιζε από την στιγμή n=1, όπως φαίνεται στο Σχήμα 3.5.1.2α. Και γενικά, αν εφαρμόζαμε την κρουστική είσοδο την χρονική στιγμή n=m, τότε το αποτέλεσμα θα ήταν η ίδια απόκριση αλλά με αρχή την στιγμή m (Σχήμα 3.5.1.2β). Όπως καταλαβαίνουμε αυτό οφείλεται στο γεγονός ότι το σύστημά μας είναι χρονικά



Σχήμα 3.5.1.2. Κρουστική απόκριση συστήματος διακριτού χρόνου για είσοδο (α) δ(n-1) και (β) δ(n-m)

3.5.2 Συνέλιξη

Τίθεται συνεπώς το ερώτημα: Ποιά θα είναι η απόκριση ενός συστήματος διακριτού χρόνου για είσοδο x(n), αν γνωρίζουμε την κρουστική του απόκριση h(n); Η απάντηση στο ερώτημα αυτό δίνεται μονολεκτικά από την λέξη συνέλιξη (convolution). Η έξοδος y(n) του συστήματος θα ισούται με την συνέλιξη της εισόδου x(n) και της κρουστικής h(n) του συστήματος, ή:

$$y(n) = x(n) * h(n) \qquad (3.30)$$

όπου * το σύμβολο της συνέλιξης. Όμως τι είναι η συνέλιξη και πώς υπολογίζεται;

Έστω, λοιπόν, ότι x(n) η είσοδος και h(n) η κρουστική απόκριση συστήματος διακριτού χρόνου. Πριν προχωρήσουμε, ας θυμηθούμε την σχέση η οποία μας λέει ότι ένα σήμα διακριτού χρόνου μπορεί να εκφρασθεί ως γραμμικός συνδυασμός ολισθημένων κρουστικών. Επίσης, ας μην ξεχνάμε ότι το σύστημα που εξετάζουμε

είναι γραμμικό (άρα ισχύει η αρχή της υπέρθεσης) και χρονικά αμετάβλητο. Έχοντας αυτά κατά νου, μπορούμε να εκφράσουμε την είσοδο x(n) ως:

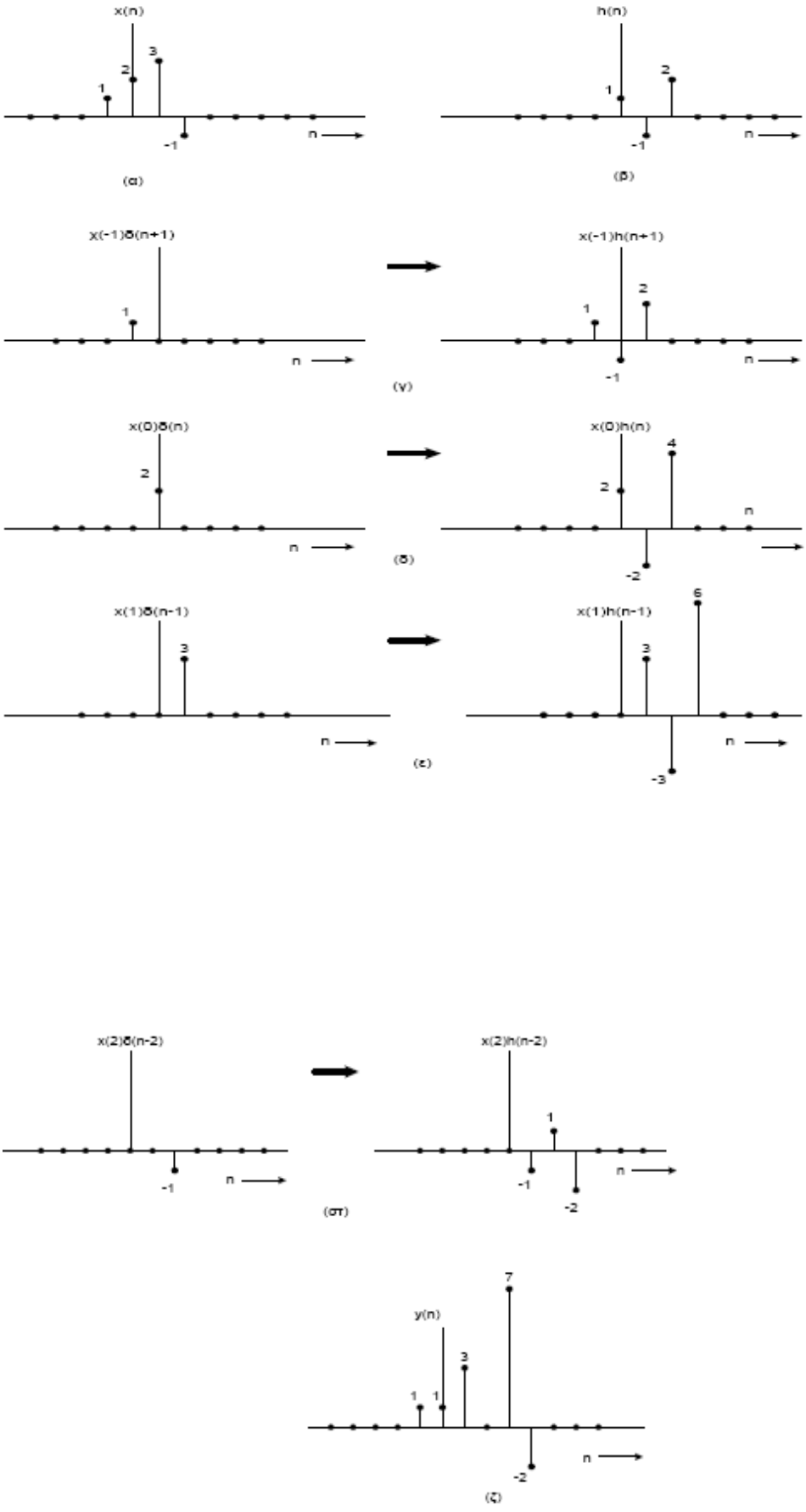
$$x(n) = \sum_{m=-\infty}^{\infty} x(m)\delta(n-m) = \ldots + x(-1)\delta(n+1) + x(0)\delta(n) + x(1)\delta(n-1) + x(2)\delta(n-2) + \ldots \quad (3.31)$$

Για κάθε μία από τις εισόδους x(m)δ(n-m), είδαμε ότι η έξοδος θα ισούται με x(m)h(n-m). Λόγω της γραμμικότητας του συστήματος, η τελική έξοδος y(n) θα είναι ίση με το άθροισμα των επιμέρους αποκρίσεων, δηλαδή:

$$y(n) = \sum_{m=-\infty}^{\infty} x(m)h(n-m) \quad (3.32)$$

Αυτή είναι η σχέση της γραμμικής συνέλιξης

Η διαδικασία που μόλις περιγράψαμε δείχνεται παραστατικά με την βοήθεια του παραδείγματος του Σχήματος 1.23, για την περίπτωση κατά την οποία {x(n)}={1,**2**,3,-1} και {h()}={**1**,-1,2}. Στα Σχήματα 3.5.2α,β φαίνονται οι ακολουθίες x(n), h(n). Στο αριστερό μέρος των Σχημάτων 3.5.2γ έως 3.5.2στ δείχνονται οι κρουστικές x(m)δ(n-m), ενώ στο δεξιό μέρος των ίδιων σχημάτων φαίνονται οι αντίστοιχες αποκρίσεις τους.



Σχήμα 3.5.2. Γραμμική συνέλιξη

Παρατηρούμε ότι το μήκος της απόκρισης είναι 6 δείγματα. Γενικά, αν N1 είναι το μήκος της μιας ακολουθίας και N2 το μήκος της άλλης ακολουθίας, τότε η γραμμική συνέλιξη αυτών δίνει μια νέα ακολουθία με μήκος N1+N2-1. Ο υπολογισμός της συνέλιξης δύο σημάτων διακριτού χρόνου με χαρτί και μολύβι γίνεται συνήθως με δύο τρόπους. Είτε γραφικά είτε με την μέθοδο της ολισθαίνουσας ράβδου

ΚΕΦΑΛΑΙΟ 4

IIR ΦΙΛΤΡΑ

4.1 IIR ΦΙΛΤΡΑ

Υπάρχουν δύο είδη ψηφιακών φίλτρων που διακρίνονται από το το μήκος των δειγμάτων της κρουστικής απόκρισης τους. Τα φίλτρα που έχουν πεπερασμένη χρονική διάρκεια κρουστικής απόκρισης, αναφέρονται ως φίλτρα FIR (Finite Impulse Response), ενώ αυτά με κρουστική απόκριση που εκτείνεται στο άπειρο λέγονται IIR (Infinite Impulse Response)

Η συνάρτηση μεταφοράς των φίλτρων IIR είναι

$$H(z)=h(0)+h(1)z^{-1}+h(2)z^{-2}+h(3)z^{-3}+ \ldots \quad (4.1)$$

ή σε αναδρομική (recursive) μορφή είναι ,

$$H(z) = \frac{b_0 + b_1 z^{-1} + \ldots + b_n z^{-n}}{1 + a_1 z^{-1} + \ldots + a_n z^{-n}} \quad (4.2)$$

Παρατηρούμε ότι η περιγραφή του IIR σε αναδρομική μορφή επιτυγχάνονεται με πεπερασμένο αριθμό συντελεστών, ενώ αντίθετα στην (4.1) ο αριθμός συντελεστών τείνει στο απείρο. Προφανώς η μορφή της (4.2) είναι προτιμητέα και για τον λόγο τα φίλτρα IIR αναφέρονται γενικά και ως αναδρομικά φίτρα.

Τα κυριότερα πλεονεκτήματα των FIR είναι η ευκολία υλοποιησής τους με γραμμική φάση και η από κατασκευή ευστάθεια των σε οποιεσδήποτε συνθήκες.

Τα IIR φίλτρα πλεονεκτούν των FIR γιατί απαιτούν χαμηλότερη τάξη φίλτρου από τα FIR για τις ίδιες βέβαια προδιαγραφές

Η συνάρτηση μεταφοράς ενός τέτοιου φίλτρου δίνεται από τη σχέση:

$$H(z) = \frac{\sum_{i=0}^{M} a_i z^{-i}}{1 + \sum_{i=1}^{N} b_i z^{-i}} \qquad (4.3)$$

όπου ai, bi οι συντελεστές του φίλτρου και τουλάχιστον ένας από τους συντελεστές bi είναι διάφορος του μηδενός. Ο αριθμός N προσδιορίζει την τάξη του φίλτρου, το οποίο για να είναι αιτιατό προϋποθέτει ότι Ν ≥Μ. Για να είναι ένα τέτοιο φίλτρο ευσταθές θα πρέπει όλοι οι πόλοι του να βρίσκονται στο εσωτερικού του μοναδιαίου κύκλου στο επίπεδο–z. Η αντίστοιχη εξίσωση διαφορών του φίλτρου είναι:

$$y(n) = \sum_{i=0}^{M} a_i x(n-i) - \sum_{i=1}^{N} b_i y(n-i) \qquad (4.4)$$

όπου x(n) και y(n) οι ακολουθίες εισόδου και εξόδου, αντίστοιχα.

Είδαμε ότι για την υλοποίηση ενός φίλτρου χρειαζόμαστε πολλαπλασιαστές, αθροιστές και στοιχεία καθυστέρησης

$$X(re^{j\omega}) = \sum_{n=-\infty}^{+\infty} x(n)(re^{j\omega})^{-n} = \sum_{n=-\infty}^{+\infty} \left[x(n)r^{-n}\right] e^{-j\omega n} = F\left\{x(n)r^{-n}\right\} \qquad (4.5)$$

όπου με F{g(n)} συμβολίζουμε το μετασχηματισμό Fourier της ακολουθίας g(n).

Η σχέση (4.5) μας δείχνει ότι ο M.Z. της ακολουθίας x(n) στο σημείο z=rejω ισούται με το μετασχηματισμό Fourier διακριτού χρόνου (DTFT) της τροποποιημένης ακολουθίας x(n)r−n για την γωνιακή συχνότητα ω. Αν η παράμετρος r επιλεγεί ίση με τη μονάδα, τότε η (4.5) μας δίνει τον DTFT της x(n). Επιπλέον, όταν η παράμετρος ω μεταβάλλεται,τότε όλες οι τιμές της μιγαδικής μεταβλητής z θα βρίσκονται πάνω στο μ*οναδιαίο κύκλο*(unit circle) του μιγαδικού επιπέδου, δηλαδή στον κύκλο που έχει ως κέντρο την αρχή των αξόνων και ακτίνα ίση με τη μονάδα. Η σχέση μεταξύ του DTFT και του M.Z. είναι πλέον προφανής. *Ο DTFT μιας ακολουθίας ισούται με το M.Z. αυτής για τις τιμές του z που βρίσκονται πάνω στο μοναδιαίο κύκλο του μιγαδικού επιπέδου-z* (z-plane) Δηλαδή για r=|z|=1 η σχέση (4.5) καταλήγει σ' αυτή του διακριτού χρόνου μετασχηματισμού Fourier:

$$X(z)\Big|_{z=e^{j\omega}} = X(e^{j\omega}) = F\{x(n)\} \quad (4.6)$$

Από τη σχέση (4.2) γίνεται φανερό ότι τα IIR φίλτρα είναι επαναληπτικά ή αναδρομικά (recursive), με την έννοια ότι δείγματα της εξόδου χρησιμοποιούνται από το σύστημα για τον υπολογισμό των νέων τιμών της εξόδου σε επόμενες χρονικές στιγμές. Το γεγονός αυτό παρουσιάζει μεγάλο πλεονέκτημα από άποψη υπολογιστικής πολυπλοκότητας, σε σχέση με τα FIR φίλτρα. Με άλλα λόγια, για να επιτύχουμε μια επιθυμητή απόκριση χρειαζόμαστε σημαντικά λιγότερους συντελεστές για ένα IIR φίλτρο σε σχέση με το αντίστοιχο FIR φίλτρο. Από την άλλη πλευρά όμως, υπάρχουν δύο σοβαρά μειονεκτήματα: (α) τα IIR φίλτρα είναι ασταθή, αν οι συντελεστές bi δεν έχουν επιλεγεί σωστά, δηλαδή αν οι πόλοι της συνάρτησης μεταφοράς βρίσκονται εκτός του μοναδιαίου κύκλου (και (β) τα IIR φίλτρα δεν έχουν γραμμική απόκριση φάσης στη ζώνη διέλευσης, όπως έχουν τα μη επαναληπτικά FIR φίλτρα με συμμετρική ή αντισυμμετρική κρουστική απόκριση.

4.2 Αναλογικά φίλτρα

Τα IIR ψηφιακά φίλτρα μπορούν εύκολα να σχεδιαστούν αρχίζοντας από ένα αναλογικό φίλτρο και κατόπιν χρησιμοποιώντας κατάλληλη απεικόνιση του επιπέδου–*s* στο επίπεδο–*z*. Έτσι, η σχεδίαση ενός ψηφιακού φίλτρου ανάγεται ουσιαστικά στη σχεδίαση του κατάλληλου αναλογικού φίλτρου και στην μετέπειτα μετάβαση από το $H(s)$ στο $H(z)$, έτσι ώστε τα επιθυμητά χαρακτηριστικά του αναλογικού φίλτρου να διατηρούνται κατά τον καλύτερο δυνατό τρόπο.

Η σχεδίαση αναλογικών φίλτρων είναι ένας τομέας για τον οποίο υπάρχει πλούσια βιβλιογραφία. Στην παρούσα ενότητα θα περιγράψουμε σε συντομία τα χαρακτηριστικά των πλέον γνωστών βαθυπερατών (lowpass) αναλογικών φίλτρων. Ακολούθως, θα αναφέρουμε τους μετασχηματισμούς συχνότητας με τους οποίους ένα αναλογικό βαθυπερατό φίλτρο μετατρέπεται σε υψηπερατό (highpass), ζωνοδιαβατό (bandpass) ή φίλτρο απόρριψης ζώνης (bandstop).

Πριν όμως προχωρήσουμε, ας δούμε τη γενική μορφή της απόκρισης συχνότητας ενός αναλογικού βαθυπερατού φίλτρου, ώστε να εξοικειωθούμε με τις έννοιες και τους συμβολισμούς. Η απόκριση συχνότητας ενός αναλογικού φίλτρου (αναλογικού συστήματος) προκύπτει από τη συνάρτηση μεταφοράς $H(s)$ αυτού για τιμές του s πάνω στον άξονα των φανταστικών $j\Omega$, δηλαδή:

$$H(j\Omega) = H(s)\Big|_{s=j\Omega} \quad (4.7)$$

Η απόκριση αυτή ονομάζεται «κανονικοποιημένη», επειδή η μέγιστη τιμή του μέτρου στη ζώνη διέλευσης είναι ίση με τη μονάδα (ή ίση με 0 dB). Η κυμάτωση (ripple) στη ζώνη διέλευσης συμβολίζεται με $\frac{1}{\sqrt{1+\varepsilon^2}}$ και αντιπροσωπεύει την ελάχιστη τιμή του μέτρου στη ζώνη διέλευσης. Η μέγιστη τιμή της κυμάτωσης στη ζώνη αποκοπής συμβολίζεται με $1/A$. Η τιμή αυτή σε dB ισούται με $-20\log(1/A)$. Οι συχνότητες Ωp και Ωs αποτελούν τις συχνότητες στο *όριο της ζώνης διέλευσης* (passband edge frequency) και στο *όριο της ζώνης αποκοπής* (stopband edge frequency), αντίστοιχα. Ο λόγος των δύο αυτών συχνοτήτων ονομάζεται *λόγος μετάβασης* (transition ratio) ή *παράμετρος επιλεκτικότητας* (selectivity parameter) και συμβολίζεται συνήθως ως k, δηλαδή $k = \frac{\Omega\rho}{\Omega s}$. (4.8)

Για ένα βαθυπερατό φίλτρο ισχύει $k < 1$. Τέλος, ο λόγος $\frac{\varepsilon}{\sqrt{A^2-1}}$ (4.9) ονομάζεται *παράμετρος διακριτότητας* (discrimination parameter) και συμβολίζεται ως $k1$, δηλαδή $k1 = \frac{\varepsilon}{\sqrt{A^2-1}}$. (4.10) Συνήθως ισχύει $k1 << 1$.

4.2.1 Γνωστα αναλογικά βαθυπερατά φίλτρα

Σε όλες τις περιπτώσεις των φίλτρων που θα αναφέρουμε αμέσως τώρα, ο στόχος μας θα είναι ο εξής: να προσεγγίσουμε κατά τον καλύτερο δυνατό τρόπο την απόκριση ενός ιδανικού φίλτρου το οποίο έχει κέρδος ίσο με 1 σε όλη τη ζώνη διέλευσης, κέρδος ίσο με 0 σε όλη τη ζώνη αποκοπής, και η ζώνη μετάβασης είναι μηδενικού εύρους, δηλαδή δεν υπάρχει ζώνη μετάβασης. Τα πιο γνωστά βαθυπερατά αναλογικά φίλτρα είναι τα ακόλουθα:

4.2.1.1Φίλτρα Butterworth. Τα βαθυπερατά φίλτρα Butterworth έχουν μόνο πόλους και το μέτρο της απόκρισης συχνότητας δίνεται από τη σχέση:

$$\left|H(j\Omega)\right| = \frac{1}{\left(1+\left(\Omega/\Omega_c\right)^{2N}\right)^{1/2}} = \frac{1}{\left(1+\varepsilon^2\left(\Omega/\Omega_p\right)^{2N}\right)^{1/2}} \quad (4.11)$$

όπου N η τάξη του φίλτρου, Ωc η συχνότητα αποκοπής (cut–off frequency), Ωp η συχνότητα στο όριο της ζώνης διέλευσης και $\frac{1}{\sqrt{1+\varepsilon}}$ (4.12) το μέτρο της απόκρισης συχνότητας στο όριο της ζώνης διέλευσης. Παρατηρούμε ότι: α. Για $\Omega = \Omega c$ το μέτρο της απόκρισης συχνότητας ισούται με $\frac{1}{\sqrt{2}}$, δηλαδή $\left|H(j\Omega)\right| = \frac{1}{\sqrt{2}}$ ανεξάρτητα από την τιμή του N.

Αυτό σημαίνει ότι η ισχύς
του σήματος υποδιπλασιάζεται για τη συγκεκριμένη συχνότητα. Η τιμή αυτή εκφρασμένη σε decibels (dB) ισούται με –3, δηλαδή:

$$\left|H(j\Omega)\right|_{dB} = 20\log_{10}\left|H(j\Omega)\right| = 20\log_{10}\left(\frac{1}{\sqrt{2}}\right) = -3.0103 \cong -3\text{dB} \quad (4.12)$$

β. Για $\Omega = 0$ το μέτρο της απόκρισης ισούται με 1, δηλαδή με 0 dB, ανεξάρτητα από την τιμή του N.

γ. Επειδή η παράγωγος του μέτρου της απόκρισης είναι πάντοτε αρνητική για θετικές τιμές του Ω, συνεπάγεται ότι η απόκριση συχνότητας μειώνεται μονοτονικά καθώς το Ω αυξάνεται. Με άλλα λόγια :

$$\left|H(j\Omega_2)\right| < \left|H(j\Omega_1)\right| \quad \text{για } 0 \le \Omega_1 < \Omega_2. \quad (4.13)$$

Για συχνότητα αποκοπής Ωc ίση με 1 rad/sec (Ωc = 1) παίρνουμε από τη σχέση (4.4) το ονομαζόμενο «πρωτότυπο βαθυπερατό φίλτρο» (prototype lowpass filter). Οι γραφικές παραστάσεις του μέτρου της απόκρισης συχνότητας του κανονικοποιημένου Butterworth βαθυπερατού φίλτρου με Ωc = 1 (πρωτότυπου) για διαφορετικές τιμές του N, φαίνεται στο Σχήμα 4.2α. Παρατηρούμε ότι αύξηση της τάξης N του φίλτρου οδηγεί σε βελτίωση των ζωνών διέλευσης και αποκοπής με ταυτόχρονη μείωση της ζώνης μετάβασης.
Οι δύο παράμετροι οι οποίες χαρακτηρίζουν πλήρως ένα φίλτρο Butterworth είναι η συχνότητα αποκοπής Ωc στα –3 dB και η τάξη του φίλτρου N (βλ. σχέση 4.4). Αυτές οι παράμετροι μπορούν να προσδιοριστούν από τις προδιαγραφές του φίλτρου, όπως αυτές φαίνονται στο Σχήμα 5.1, δηλαδή το όριο της ζώνης διέλευσης Ωp, την ελάχιστη τιμή του μέτρου στη ζώνη διέλευσης $1/\sqrt{1^2+\varepsilon}$ το όριο της ζώνης αποκοπής Ωs και τη μέγιστη τιμή του μέτρου στη ζώνη αποκοπής $1/A$. Έτσι, από την εξίσωση (4.4) έχουμε:

$$\left|H(j\Omega_p)\right|^2 = \frac{1}{1+\left(\Omega_p/\Omega_c\right)^{2N}} = \frac{1}{1+\varepsilon^2} \quad (4.14)$$

$$\left|H(j\Omega_s)\right|^2 = \frac{1}{1+\left(\Omega_s/\Omega_c\right)^{2N}} = \frac{1}{A^2} \quad (4.15)$$

Λύνοντας το σύστημα αυτό των εξισώσεων ως προς την τάξη *N* του φίλτρου καταλήγουμε στη σχέση:

$$N = \frac{1}{2}\frac{\log_{10}\left[(A^2-1)/\varepsilon^2\right]}{\log_{10}\left(\Omega_s/\Omega_p\right)} = \frac{\log_{10}(1/k_1)}{\log_{10}(1/k)} \quad (4.16)$$

Εξυπακούεται ότι η τιμή του *N* πρέπει να είναι ακέραιος αριθμός. Στην περίπτωση όπου η τιμή που υπολογίζεται από την παραπάνω σχέση δεν είναι ακέραιος, τότε αυτή στρογγυλεύεται στον αμέσως μεγαλύτερο ακέραιο.
Τέλος, όπως αναφέραμε και στην αρχή της παραγράφου, τα φίλτρα Butterworth έχουν μόνο πόλους. Αυτό φαίνεται από τη σχέση (4.4). Οι θέσεις των πόλων πάνω στο μιγαδικό επίπεδο μπορούν να υπολογιστούν ως εξής:

Αφού

$$H(s)H(-s) = \left|H(j\Omega)\right|^2 \quad (4.17)$$

για

$$s = j\Omega \quad (4.18)$$

η εξίσωση (5.4) γράφεται ως:

$$H(s)H(-s) = \frac{1}{1+\left(-s^2/\Omega_c^2\right)^N} \quad (4.19)$$

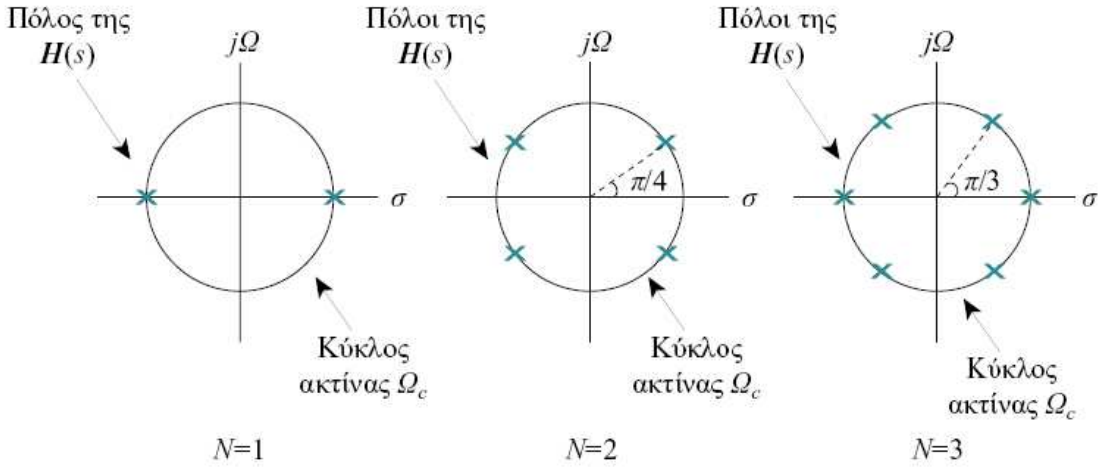
Οι πόλοι της $H(s)H(-s)$ βρίσκονται πάνω σε ένα κύκλο ακτίνας Ωc και σε σημεία που ισαπέχουν το ένα από το άλλο. Αυτό προκύπτει από την εξίσωση (5.7) με μηδενισμό του παρανομαστή της, δηλαδή:

$$\frac{-s^2}{\Omega_c^2} = (-1)^{1/N} = e^{j(2q+1)\pi/N}, \qquad q = 0,1,....N-1 \quad (4.20)$$

και συνεπώς

$$s_q = \Omega_c e^{j\pi/2} e^{j(2q+1)\pi/2N}, \qquad q = 0,1,....N-1 \quad (4.21)$$

Στο Σχήμα 4.2.1 δείχνονται οι θέσεις των πόλων διαφόρων φίλτρων Butterworth τάξηςN = 1, N = 2 και N = 3. Οι πόλοι που βρίσκονται στο αριστερό ημιεπίπεδο είναι αυτοί που αντιστοιχούν στην $H(s)$, ενώ εκείνοι του δεξιού ημιεπιπέδου αντιστοιχούν στην $H(–s)$. Υπενθυμίζεται ότι για να είναι ένα αναλογικό σύστημα ευσταθές, πρέπει οι πόλοι του να βρίσκονται στο αριστερό ημιεπίπεδο.



Σχήμα 4.2.1

Είναι εύκολο να δούμε ότι η συνάρτηση μεταφοράς των πρωτότυπων (Ωc = 1) βαθυπερατών φίλτρων Butterworth πρώτης, δεύτερης και τρίτης τάξης είναι αντίστοιχα:

$$H(s)=\frac{1}{s+1},\ H(s)=\frac{1}{s^2+\sqrt{2}s+1}\ \text{και}\ H(s)=\frac{1}{s^3+2s^2+2s+1} \quad (4.22)$$

4.2.1.2 Φίλτρα Chebyshev

Υπάρχουν δύο τύποι φίλτρων Chebyshev. Τα φίλτρα τύπου I είναι φίλτρα μόνο με πόλους τα οποία παρουσιάζουν ομοιόμορφη κυμάτωση στη ζώνη διέλευσης και μονοτονική συμπεριφορά στη ζώνη αποκοπής (Σχήμα 4.2β). Το μέτρο της απόκρισης συχνότητας των βαθυπερατών φίλτρων Chebyshev τύπου I δίνεται από τη σχέση:

$$\left|H(j\Omega)\right|=\frac{1}{\left(1+\varepsilon^2 T_N^2\left(\Omega/\Omega_{\mathrm{p}}\right)\right)^{1/2}} \quad (4.23)$$

όπου ε η παράμετρος του φίλτρου που σχετίζεται με την κυμάτωση στη ζώνη διέλευσης, Ωp η συχνότητα στο όριο της ζώνης διέλευσης και TN(x) πολυώνυμο Chebyshev τάξης N, το οποίο ορίζεται ως:

$$T_N(x)=\begin{cases}\cos(N\cos^{-1}x), & |x|<1\\ \cosh(N\cosh^{-1}x), & |x|>1\end{cases} \quad (4.24)$$

Τα φίλτρα Chebyshev τύπου II είναι φίλτρα με πόλους και μηδενικά, παρουσιάζουν μονοτονική συμπεριφορά στη ζώνη διέλευσης και ομοιόμορφη κυμάτωση στη ζώνη αποκοπής (Σχήμα 4.2γ). Τα μηδενικά στην περίπτωση αυτών των φίλτρων βρίσκονται πάνω στο φανταστικό άξονα του επιπέδου–s. Το μέτρο της απόκρισης συχνότητας δίνεται από τη σχέση:

$$\left|H(j\Omega)\right|=\frac{1}{\left(1+\varepsilon^2\left[T_N^2\left(\Omega_{\mathrm{s}}/\Omega_{\mathrm{p}}\right)/T_N^2\left(\Omega_{\mathrm{s}}/\Omega\right)\right]\right)^{1/2}} \quad (4.25)$$

όπου *T*N(*x*) είναι και πάλι το πολυώνυμο Chebyshev τάξης *N*, και *Ωs* η συχνότητα στο όριο της ζώνης αποκοπής.

4.2.1.3 Ελλειπτικά φίλτρα.

Τα ελλειπτικά φίλτρα ή φίλτρα Cauer παρουσιάζουν ομοιόμορφη κυμάτωση τόσο στη ζώνη διέλευσης, όσο και στη ζώνη αποκοπής (Σχήμα 5.2δ). Τα φίλτρα αυτά έχουν πόλους και μηδενικά και το μέτρο της απόκρισης στη συχνότητα δίνεται από τη σχέση:

$$\left|H(j\Omega)\right| = \frac{1}{\left(1+\varepsilon^2 U_N\left(\Omega/\Omega_{\mathrm{p}}\right)\right)^{1/2}} \qquad (5.26)$$

όπου *UN*(*x*) η Ιακωβιανή (Jacobian) ελλειπτική συνάρτηση τάξης *N*. Τα μηδενικά βρίσκονται πάνω στον άξονα των φανταστικών του επιπέδου–*s*.
Τα ελλειπτικά φίλτρα είναι καλύτερα των φίλτρων Butterworth και Chebyshev, ως προς το γεγονός ότι απαιτούνται φίλτρα μικρότερης τάξης, δηλαδή λιγότεροι συντελεστές, για να ικανοποιήσουν τις ίδιες προδιαγραφές. Με άλλα λόγια, για δεδομένη τάξη του φίλτρου και δεδομένες προδιαγραφές, ένα ελλειπτικό φίλτρο θα παρουσιάζει την στενότερη ζώνη μετάβασης. Από την άλλη πλευρά όμως, η απόκριση φάσης των ελλειπτικών φίλτρων παρουσιάζει περισσότερες μη γραμμικότητες στη ζώνη διέλευσης σε σχέση με τα αντίστοιχα φίλτρα Butterworth ή Chebyshev.

4.3 Σχεδίαση IIR ψηφιακών φίλτρων

Όπως και στην περίπτωση των FIR φίλτρων, έτσι και εδώ, υπάρχουν πολλές μέθοδοι σχεδίασης IIR φίλτρων. Η πρώτη και πιο απλή από αυτές, και είναι αυτή της απευθείας τοποθέτησης των πόλων και μηδενικών του φίλτρου πάνω στο μιγαδικό επίπεδο–z, προσδιορίζοντας άμεσα τη συνάρτηση μεταφοράς και κατά συνέπεια, τους συντελεστές του φίλτρου. Η μέθοδος αυτή έχει περισσότερο θεωρητική και εκπαιδευτική σημασία παρά πρακτική αξία. Υπάρχουν και άλλες τεχνικές, κατά τις οποίες η σχεδίαση του φίλτρου γίνεται απευθείας στο πεδίο–z, όπως η μέθοδος των προσεγγίσεων Pade ή η μέθοδος των ελαχίστων τετραγώνων, τις οποίες όμως δε θα πραγματευτούμε.

Περισσότερο συνηθισμένο είναι να χρησιμοποιούμε τεχνικές οι οποίες μετατρέπουν ένα αναλογικό φίλτρο σε ψηφιακό. Και εδώ υπάρχουν διαφορετικές μέθοδοι μετατροπής ενός αναλογικού φίλτρου σε ψηφιακό. Εμείς θα αναφερθούμε στις δύο πιο γνωστές από αυτές, τη μέθοδο της αμετάβλητης κρουστικής και τη μέθοδο του διγραμμικού μετασχηματισμού.
Τέλος, θα πρέπει να επαναλάβουμε ότι υλοποιήσιμα και ευσταθή IIR φίλτρα δε μπορεί να έχουν γραμμική απόκριση φάσης. Έτσι, οι προδιαγραφές των φίλτρων θα αναφέρονται μόνο στο μέτρο της απόκρισης στη συχνότητα. Αυτό δεν σημαίνει ότι η φάση δεν είναι ουσιώδους σημασίας. Σημαίνει απλώς πως, αφού μέτρο και φάση είναι αλληλένδετα, δεχόμαστε τη φάση όπως αυτή προκύπτει από τη διαδικασία σχεδίασης. Αν κάποια εφαρμογή απαιτεί γραμμική φάση, τότε θα πρέπει να σχεδιάσουμε ένα FIR φίλτρο με συμμετρική ή αντισυμμετρική κρουστική απόκριση.

4.3.1 Γραφική μέθοδος

Η μέθοδος αυτή αναφέρεται και ως *γεωμετρική* ή ως *μέθοδος σχεδίασης βασισμένη στους πόλους και στα μηδενικά*. Σύμφωνα με τη μέθοδο αυτή, η σχεδίαση ενός IIR ψηφιακού φίλτρου γίνεται με κατάλληλη τοποθέτηση των πόλων και των μηδενικών πάνω στο μοναδιαίο κύκλο στο επίπεδο–z. Η τοποθέτηση ενός πόλου κοντά στο μοναδιαίο κύκλο οδηγεί σε μεγάλες τιμές για τη συνάρτηση μεταφοράς σε συχνότητες που βρίσκονται κοντά στον πόλο. Αντίστοιχα, η ύπαρξη ενός μηδενικού κοντά ή και πάνω στον μοναδιαίο κύκλο οδηγεί σε πολύ μικρές τιμές ή και μηδενισμό της συνάρτησης μεταφοράς και κατ' επέκταση της εξόδου του φίλτρου (συστήματος). Για να είναι ευσταθές το φίλτρο που θα σχεδιαστεί, θα πρέπει οι πόλοι του να βρίσκονται στο εσωτερικό του μοναδιαίου κύκλου. Τα μηδενικά μπορεί να βρίσκονται οπουδήποτε στο επίπεδο–z. Τέλος, οι πόλοι και τα μηδενικά μπορεί να είναι πραγματικοί ή μιγαδικοί αριθμοί. Στην περίπτωση όμως που κάποιοι από τους πόλους ή τα μηδενικά είναι μιγαδικοί, τότε αυτοί θα πρέπει να εμφανίζονται ως ζεύγη συζυγών.

4.3.2 Μέθοδος αμετάβλητης κρουστικής

Σκοπός μας σύμφωνα με τη *μέθοδο της αμετάβλητης κρουστικής* (impulse invariance method) είναι να σχεδιάσουμε ένα IIR φίλτρο του οποίου η μοναδιαία κρουστική απόκριση $h(n)$ να ισοδυναμεί με τη δειγματοληπτημένη κρουστική απόκριση ενός κατάλληλα επιλεγμένου αναλογικού φίλτρου, δηλαδή:

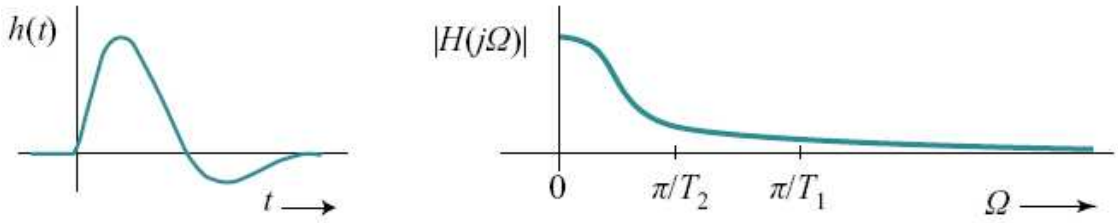
$$h(n) \equiv h(nT) = h(t)\Big|_{t=nT} \qquad \text{n} = 0,1,2,.... \tag{4.27}$$

όπου T η περίοδος δειγματοληψίας (βλ. Σχήμα 4.5α, β).

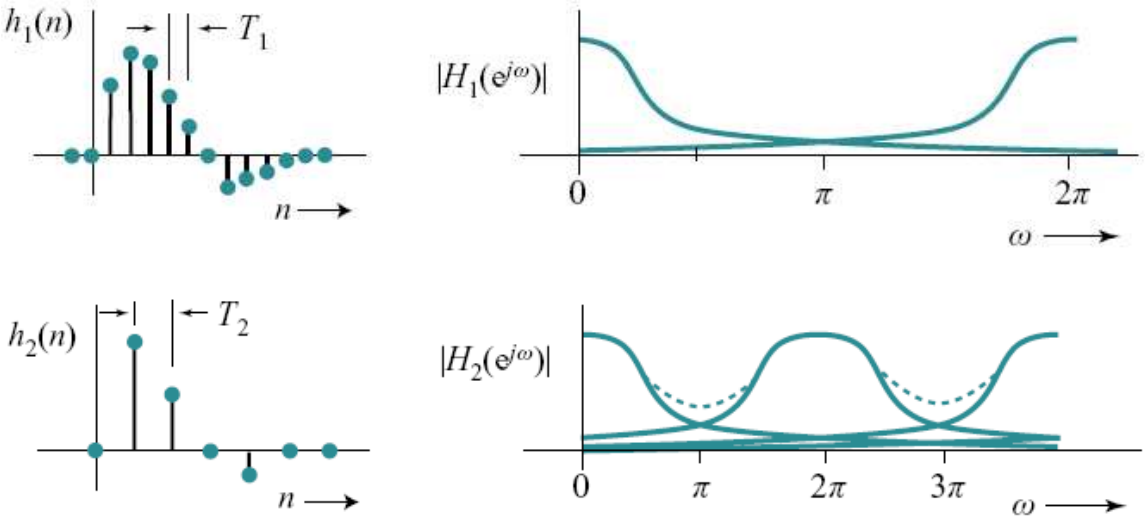
Η συνάρτηση μεταφοράς $H(z)$ ισούται με τον μετασχηματισμό–z της δειγματοληπτημένης μοναδιαίας κρουστικής απόκρισης $h(nT)$, δηλαδή:

$$H(z) = Z\{h(nT)\} \quad (4.28)$$

Η αναλογική συνάρτηση μεταφοράς $H(s)$ ισούται με τον μετασχηματισμό Laplace της κρουστικής απόκρισης $h(t)$.



Εικόνα 4.3.2 Κρουστική απόκριση και μέτρο φάσματος αναλογικού φίλτρου



Εικόνα 4.3.2.1 Δειγματολειψία της κρουστικής του αναλογικού με περίοδο T1 < T2

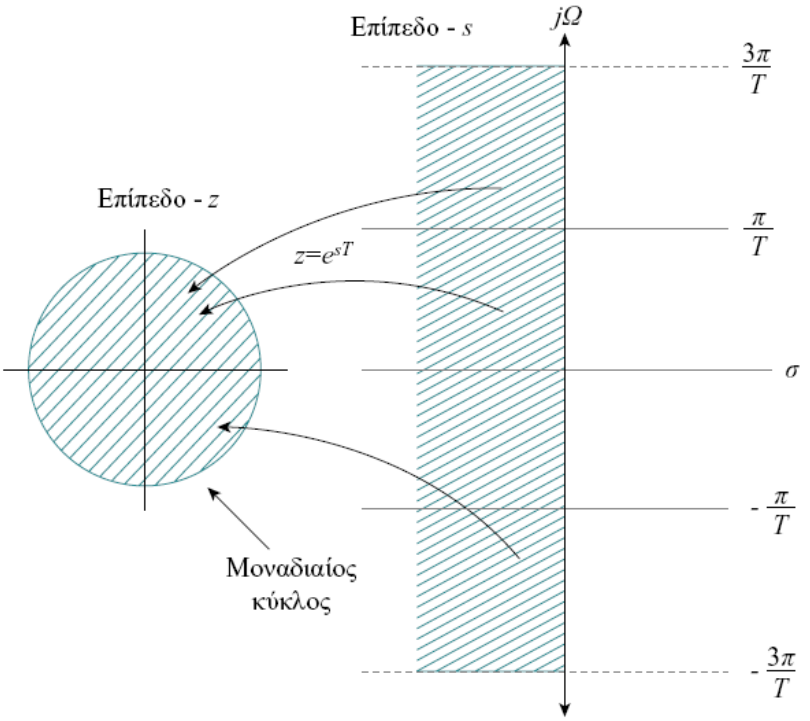
Στο σημείο αυτό θα πρέπει να εξετάσουμε το αποτέλεσμα της δειγματοληψίας της αναλογικής κρουστικής απόκρισης, δηλαδή το αποτέλεσμα της σχέσης (4.22). Ας θυμηθούμε ότι, όταν λαμβάνουμε από ένα αναλογικό σήμα δείγματα με ρυθμό $Fs = 1/T$ δειγμάτων ανά δευτερόλεπτο, το φάσμα του δειγματοληπτημένου σήματος που προκύπτει είναι μια περιοδική επανάληψη του αναλογικού φάσματος, με περίοδο Fs. Συνεπώς, αν ο ρυθμός δειγματοληψίας είναι μικρότερος από το διπλάσιο της μέγιστης συχνότητας που υπάρχει στο αναλογικό φάσμα, τότε θα παρουσιαστεί το φαινόμενο του ανεπαρκούς ρυθμού δειγματοληψίας (aliasing), όπως αυτό δείχνεται στο Σχήμα 4.5γ. Γίνεται φανερό ότι η περίοδος δειγματοληψίας T θα πρέπει να είναι αρκετά μικρή, για να αποφύγουμε ή τουλάχιστον να ελαχιστοποιήσουμε το φαινόμενο αυτό. Επίσης, καταλαβαίνουμε ότι η μέθοδος της αμετάβλητης κρουστικής δεν είναι κατάλληλη για τη σχεδίαση υψηπερατών φίλτρων, εξαιτίας του φαινομένου aliasing που παρουσιάζεται στο φάσμα λόγω της δειγματοληψίας. Για τους λόγους αυτούς, η μέθοδος χρησιμοποιείται κυρίως για την σχεδίαση βαθυπερατών και ζωνοδιαβατών ψηφιακών φίλτρων.
Η απεικόνιση των σημείων από το επίπεδο–s στο επίπεδο–z με την αμετάβλητη κρουστική απόκριση γίνεται μέσω της σχέσης:

$$z = e^{sT} \quad (4.29)$$

Για $s = \sigma + j\Omega$ και $z = re^{j\omega}$ βρίσκουμε ότι:

$$r = e^{\sigma T}, \quad \omega = \Omega T \quad (4.30)$$

Επομένως, για $\sigma < 0$ έχουμε $0 < r < 1$, ενώ για $\sigma > 1$ έχουμε $r > 1$. Όταν $\sigma = 0$, τότε $r = 1$. Αυτό σημαίνει ότι το αριστερό ημιεπίπεδο του s απεικονίζεται στο εσωτερικό του μοναδιαίου κύκλου στο z, ενώ το δεξί ημιεπίπεδο του s απεικονίζεται στο εξωτερικό του μοναδιαίου κύκλου στο z. Επίσης, ο άξονας $j\Omega$ απεικονίζεται πάνω στο μοναδιαίο κύκλο του επιπέδου–z. Η απεικόνιση αυτή όμως δεν είναι απεικόνιση ένα προς ένα. Αφού το ω είναι μοναδικό στην περιοχή $(-\pi, \pi)$, η απεικόνιση $\omega = \Omega T$ συνεπάγεται ότι η περιοχή των συχνοτήτων $-\pi/T \le \Omega \le \pi/T$ απεικονίζεται στις αντίστοιχες τιμές $-\pi \le \omega \le \pi$. Επιπλέον, όμως, και η περιοχή συχνοτήτων $\pi/T \le \Omega \le 3\pi/T$ απεικονίζεται επίσης στην ίδια περιοχή $-\pi \le \omega \le \pi$, και γενικά το ίδιο συμβαίνει για τις συχνότητες $(2k-1)\pi/T \le \Omega \le (2k+1)\pi/T$, όπου k ακέραιος. Άρα, η απεικόνιση της αναλογικής συχνότητας Ω στη συχνότητα ω του ψηφιακού χώρου είναι μία πολλά–σε–ένα απεικόνιση, γεγονός που αντικατοπτρίζει το φαινόμενο του ανεπαρκούς ρυθμού δειγματοληψίας. Η όλη απεικόνιση του επιπέδου–s στο επίπεδο–z φαίνεται στο Σχήμα 4.3.2.2



Εικόνα 4.3.2.2 Η όλη απεικόνιση του επιπέδου–s στο επίπεδο–z

4.3.3 Μέθοδος δυγραμμικού μετασχηματισμού

Η μέθοδος της αμετάβλητης κρουστικής απόκρισης για τη σχεδίαση IIR ψηφιακών φίλτρων είναι κατάλληλη για βαθυπερατά και ορισμένα ζωνοδιαβατά φίλτρα. Θα ασχοληθούμε τώρα με τη μέθοδο του διγραμμικού μετασχηματισμού (bilinear transformation method), η οποία τουλάχιστον δεν θέτει τέτοιους περιορισμούς. Πρόκειται για έναν μαθηματικό μετασχηματισμό του πεδίου–*s* στο πεδίο–*z*. Σύμφωνα με τον μετασχηματισμό αυτό, όταν μας δίνεται η συνάρτηση μεταφοράς $H(s)$ του αναλογικού φίλτρου, μπορούμε να υπολογίσουμε τη συνάρτηση μεταφοράς $H(z)$ του αντίστοιχου ψηφιακού φίλτρου, αντικαθιστώντας το s με το $(z-1)/(z+1)$, δηλαδή:

$$H(z) = H(s)\Big|_{s=\frac{z-1}{z+1}} \qquad (4.31)$$

Με άλλα λόγια, η απεικόνιση από το επίπεδο *s* στο επίπεδο *z* γίνεται με βάση τη σχέση:

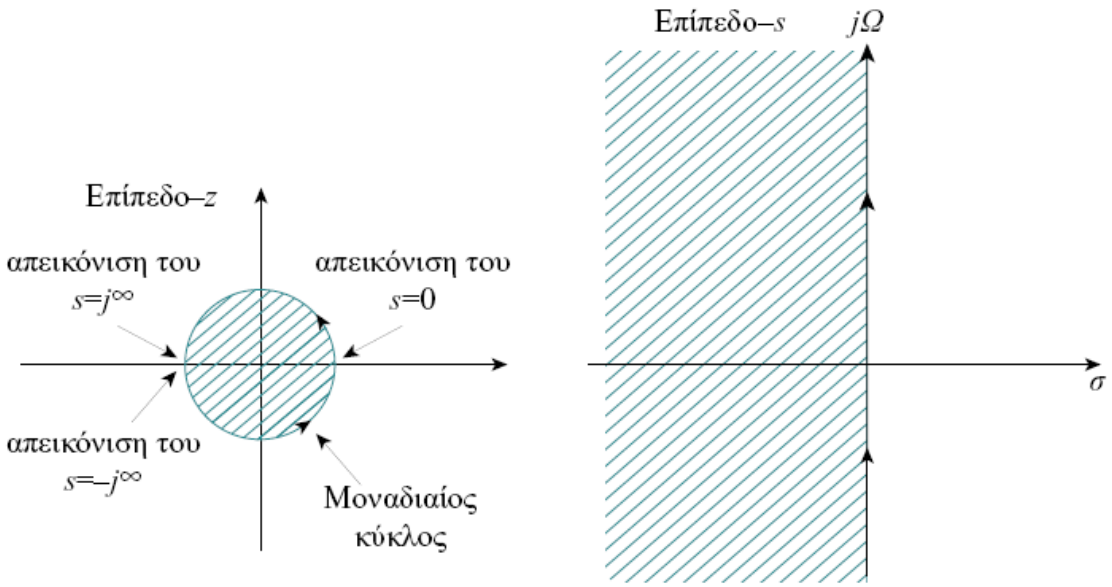
$$s = \frac{z-1}{z+1} \quad (4.32)$$

Για να διερευνήσουμε τα χαρακτηριστικά του διγραμμικού μετασχηματισμού, ας θεω-
ρήσουμε ότι $z = re^{j\omega}$ και $s = \sigma + j\Omega$. Τότε, η σχέση (4.32) μπορεί να εκφραστεί ως:

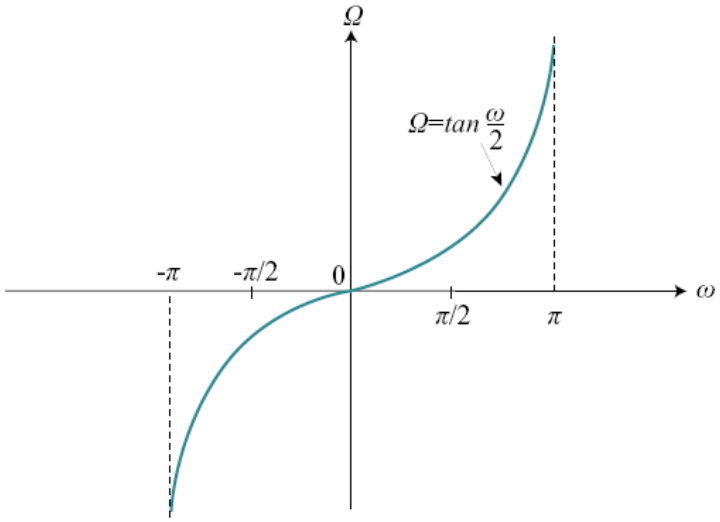
$$s = \frac{re^{j\omega}-1}{re^{j\omega}+1} = \frac{r^2-1}{\underbrace{1+r^2+2r\cos\omega}_{\Omega}} + j\frac{2r\sin\omega}{\underbrace{1+r^2+2r\cos\omega}_{\Omega}} \quad (4.33)$$

Από τη σχέση αυτή παρατηρούμε ότι, εάν $r < 1$, τότε $\sigma < 0$ και εάν $r > 1$, τότε $\sigma > 0$. Αυτό σημαίνει ότι το αριστερό ημιεπίπεδο του s απεικονίζεται στο εσωτερικό του μοναδιαίου κύκλου στο επίπεδο–z, ενώ το δεξί ημιεπίπεδο του s απεικονίζεται στο εξωτερικό του μοναδιαίου κύκλου (βλ. Σχήμα 4.7). Άρα, από ένα ευσταθές αναλογικό φίλτρο παίρνουμε ένα ευσταθές ψηφιακό φίλτρο. Όταν $r = 1$, τότε $\sigma = 0$ και:

$$\Omega = \tan\frac{\omega}{2} \quad (4.34)$$



Εικόνα 4.3.2.3 Απεικόνιση του επιπέδου – s στο επίπεδο – z μέσω του διγραμμικου μετασχηματισμου



Εικόνα 4.3.2.4 Μη γραμμική σχέση μεταξύ των αναλογικών και ψηφιακών συχνοτήτων λόγω του διγραμμικού μετασχηματισμού

Η σχέση μεταξύ των συχνοτήτων στα δύο πεδία s και z φαίνεται στο Σχήμα 4.8. Γίνεται φανερό ότι όλη η περιοχή των συχνοτήτων Ω απεικονίζεται μόνο μία φορά στην περιοχή $-\pi \le \omega \le \pi$. Πρόκειται επομένως, για μια ένα–προς–ένα απεικόνιση. Η απεικόνιση όμως, αυτή είναι μη γραμμική. Παρατηρείται, δηλαδή, μια παραμόρφωση ή

στρέβλωση των συχνοτήτων (frequency warping), εξαιτίας της μη γραμμικότητας της συνάρτησης της εφαπτομένης. Αυτή η στρέβλωση των συχνοτήτων θα πρέπει, όπως θα δούμε στη συνέχεια, να λαμβάνεται υπόψη κατά την σχεδίαση ενός IIR φίλτρου.

4.4 Υλοποιηση IIR Ψηφιακών Φίλτρων

Η υλοποίηση των IIR φίλτρων, όπως και αυτή των FIR φίλτρων, μπορεί να γίνει είτε σε κυκλωματική μορφή (hardware), είτε με προγραμματισμό (software) ενός υπολογιστικού συστήματος γενικού ή ειδικού σκοπού.

Γενικά, η εξίσωση διαφορών ενός φίλτρου είναι μια υπολογιστική διαδικασία (ένας αλγόριθμος) για τον υπολογισμό της ακολουθίας εξόδου $y(n)$ του συστήματος, από την ακολουθία εισόδου x οι υπολογισμοί αυτοί μπορούν να γίνουν με ένα διαφορετικό, αλλά ισοδύναμο, σύνολο εξισώσεων διαφορών. Κάθε ένα τέτοιο σύνολο εξισώσεων ορίζει μια υπολογιστική δια-δικασία ή έναν αλγόριθμο για την υλοποίηση του συστήματος.
Για κάθε ένα σύνολο εξισώσεων μπορούμε να σχεδιάσουμε ένα διάγραμμα βαθμίδων το οποίο να αποτελείται από στοιχεία καθυστέρησης, πολλαπλασιαστές και αθροιστές. Αυτό το διά γραμμα βαθμίδων το χαρακτηρίσαμε ως *δομή πραγματοποίησης* (realisation structure) του συστήματος. Στην περίπτωση κατά την οποία το σύστημα (φίλτρο) υλοποιηθεί προγραμματιστικά (software), το διάγραμμα βαθμίδων ή ισοδύναμα το σύνολο των εξισώσεων διαφορών, μετατρέπεται σε ένα σύνολο εντολών (πρόγραμμα) το οποίο «τρέχει» στο συγκεκριμένο υπολογιστικό σύστημα.
Το ερώτημα, βέβαια, που μπορεί να γεννηθεί στον αναγνώστη είναι το γιατί θα πρέπει να έχουμε διαφορετικές δομές πραγματοποίησης ενός φίλτρου, για παράδειγμα, άμεση μορφή τύπου 1 ή 2 σε σειρά ή παράλληλη από τη στιγμή που έχουμε σχεδιάσει το φίλτρο, γνωρίζουμε τους συντελεστές του και κατά συνέπεια μπορούμε να υλοποιήσουμε την εξίσωση διαφορών που έχει προκύψει ως έχει.
Η απάντηση στο ερώτημα αυτό είναι ότι οι διάφορες δομές πραγματοποίησης παρουσιάζουν διαφορετικά χαρακτηριστικά ως προς την απαιτούμενη *υπολογιστική πολυπλοκότητα* (computational complexity), τη *μνήμη* (memory) και τα προβλήματα λόγω του *πεπερασμένου μήκους λέξης* (finite wordlength effects).
Όταν λέμε *υπολογιστική πολυπλοκότητα* εννοούμε το πλήθος των αριθμητικών πράξεων (πολλαπλασιασμών, διαιρέσεων και προσθέσεων) οι οποίες απαιτούνται για τον υπολογισμό κάθε δείγματος εξόδου $y(n)$. Πριν από λίγα χρόνια, αυτές οι τρεις πράξεις ήταν οι μοναδικές που χρησιμοποιούσαμε για να μετρήσουμε την υπολογιστική πολυπλοκότητα. Στις μέρες μας όμως, με την εντυπωσιακή πρόοδο που έχει συντελεστείστο σχεδιασμό και την κατασκευή πολύπλοκων και αποτελεσματικών προγραμματιζόμενων ολοκληρωμένων κυκλωμάτων ψηφιακής επεξεργασίας σήματος (digital signal processors, DSPs), άλλοι παράγοντες παίζουν επίσης ρόλο στον προσδιορισμό της υπολογιστικής πολυπλοκότητας, όπως για παράδειγμα, το πλήθος των προσπελάσεων της μνήμης ή το πλήθος των συγκρίσεων δύο αριθμών.

Όταν λέμε *μνήμη* εννοούμε το πλήθος των θέσεων μνήμης που απαιτούνται για την αποθήκευση των δειγμάτων εισόδου και εξόδου, των παραμέτρων του συστήματος, καθώς και των ενδιάμεσων αποτελεσμάτων που τυχόν προκύπτουν.
Τέλος, τα προβλήματα λόγω του *πεπερασμένου μήκους λέξης*, ή αλλιώς λόγω της πεπερασμένης ακριβείας, αναφέρονται στα φαινόμενα κβάντισης τα οποία ενυπάρχουν σε οποιαδήποτε ψηφιακή υλοποίηση ενός συστήματος είτε κυκλωματική, είτε προγραμματιστική. Οι παράμετροι του συστήματος πρέπει απαραίτητα να αναπαρασταθούν με πεπερασμένη ακρίβεια. Τα αποτελέσματα που προκύπτουν στη διαδικασία υπολογισμού κάθε δείγματος εξόδου του συστήματος, πρέπει να στρογγυλευθούν (rounded) ή να αποκοπούν (truncated) στο κατάλληλο μήκος λέξης (κατάλληλη ακρίβεια) που επιβάλλει ο υπολογιστής ή το κύκλωμα που χρησιμοποιούμε για την υλοποίηση. Όλα αυτά είναι προβλήματα τα οποία επηρεάζουν σημαντικά την τελική μαζαπόφαση σχετικά με το ποια δομή πραγματοποίησης θα πρέπει να επιλέξουμε. Αποδεικνύεται ότι οι διαφορετικές δομές πραγματοποίησης ενός συστήματος, ενώ είναι ισοδύναμες στην περίπτωση που χρησιμοποιούμε άπειρη ακρίβεια, παρουσιάζουν πολύ διαφορετική συμπεριφορά μεταξύ τους για πεπερασμένη ακρίβεια

ΠΡΑΚΤΙΚΟ ΜΕΡΟΣ

ΚΕΦΑΛΑΙΟ 5

ΔΟΜΗ ΑΡΧΕΙΩΝ WAV

5.1 Γενικά

Το format αρχείων είναι το native είδος αρχείου για την αποθήκευση audio data στο λειτουργικό σύστημα των Windows. Έχει γίνει ένα απο τα πιο διαδεδομένα format ψηφιακού ήχου κυρίως για την αποδοχή των Windows αλλά και για την τεράστια γκάμα προγραμμάτων που έχουν γραφτεί για αυτήν την πλάτφόρμα. Σχεδόν κάθε μοντέρνο πρόγραμμα που μπορεί να διαχειριστεί ψηφιακό ήχο υποστηρίζει αρχεία wav. Τα παρακάτω specifications δίνουν μια λεπτομερή περιγραφή για την δομή και την λειτουργία του format wav.

5.2 Data Formats

Απο την στιγμή που το format των wave είναι γραμμένο για windows και κατ' επέκταση για Intel επεξεργαστές όλα τα data values αποθηκεύονται με την μέθοδο Little-Endian (LSB – το λιγότερο σημαντικό byte πρώτο)

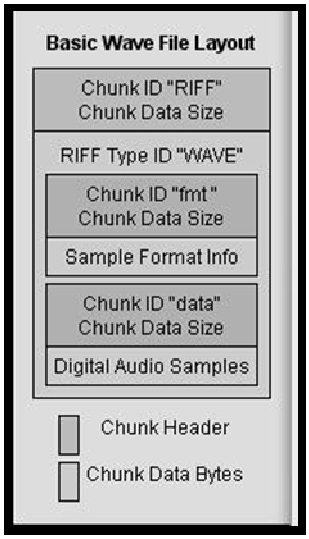
Τα αρχεία Wave μπορεί να εμπεριέχουν γραμμές κειμένου (Strings) ώστε να δίνουν πληροφορίες για point labels, νότες κλπ. Τα Strings αποθηκεύονται σε έαν format όπου το πρώτο byte υποδεικνείει των αριθμό των ASCII byte κειμένου που ακολουθούν. Τα bytes που ακολουθούν ελίναι οι χαρακτήρες bytes ASCII τα οποία δημιουργούν τα text strings. Οι προγραμματιστές Pascal μπορούν να παρτατηρήσουν ότι η ίδια διαδικασία χρησιμοποιείται και για τα Strings στην Pascal.

Εικόνα 5.2 Παράδειγμα ενός Wave String Format

5.3 Δομή αρχείου wav

Τα wave αρχεία χρησιμοποιούν την τυπίκη δομή RIFF στην οποία ομαδοποιεί τα περιεχόμενα του αρχείου (sample format, digital audio samples κλπ κλπ) σε διαφορετικά chunks, απο τα οποία το καθένα περιέχει την δική του επικεφαλίδα (header) και τα data bytes. Η επικεφαλίδα chunk υποδεικνείει τον τύπο και το μέγεθος του εκάστοτε chunk. Η μέθοδος οργάνωσης επιτρέπει σε προγράμματα που δεν χρησιμοποιούν ή δεν αναγνωρίζουν συγκεκριμένου τύπου chunks να τα προσπερνούν και να συνεχίζουν να επεξεργάζονται chunks που αναγνωρίζουν. Ορισμένου τύπου chunks μπορεί να εμπεριέχουν sub-chunks. Για παράδειγμα στο παρακάτω διάγραμμα μπορούμε να δούμε ότι τα “fmt” και “data” chunks είναι στην πραγματικότητα sub-chunks του “RIFF” chunk.

Ένα πράγμα που χρειάζετε προσοχή με τα RIFF file chunks είναι ότι πρέπει να είναι word aligned. Αυτό σημαίνει ότι το συνολικό μέγεθος τους πρέπει να είναι πολλαπλάσσιο των 2 bytes (πχ. 2,4,6,8 κλπ.). Εάν ένα chunk εμπεριέχει ένα παράξενο αριθμό data bytes ο οποίος δημιουργεί πρόβλημα στο word aligned,ένα extra byte με τιμή 0 πρέπει να επακολουθήσει μετά το τελευταίο data byte. Αυτό το τελευταίο byte που θα προστεθεί δεν θα υπολογιστεί στο μέγεθος του chunk.



Εικόνα 5.3 Δομή αρχείου Wave

5.3.1 Wave File Header - RIFF Type Chunk

Η επικεφαλίδα (header) ακολουθεί την τυπική δομή ενός αρχείου RIFF. Τα πρώτα 8 bytes του αρχείου είναι μια τυπική επικεφάλιδα RIFF chunk η οποία έχει ένα chunk ID ενός RIFF και ένα μέγεθος chunk ίσο με το μέγεθος του αρχείου μείον τα 8 bytes που χρησιμοποιεί η επικεφαλίδα (header). Τα πρώτα 4 data bytes στο RIFF chunk υποδεικνείει τον τύπο της πηγής που βρίσκουμε στο RIFF chunk. Τα αρχεία Wave πάντα χρησιμοποιούν "WAVE". Μετά απο το τύπου RIFF ακολουθούν όλα τα αρχεία Wave chunks που ορίζουν την audio κυματομορφή.

Offset	Size	Description	Value
0x00	4	Chunk ID	"RIFF" (0x52494646)
0x04	4	Chunk Data Size	(file size) - 8
0x08	4	RIFF Type	"WAVE" (0x57415645)
0x10	Wave chunks		

Εικόνα 5.3.1 Τιμές RIFF chunk τύπου.

5.3.2 Wave File Chunks

Υπάρχουν πολλών ειδών chunks που ορίζουν τα wave files. Πολλά wave files εμπεριέχουν μόνο δύο chunks συγκεκριμένα το Format Chunk και το Data Chunk. Αυτά τα δύο chunks χρειάζονται για για να περιγράψουν το format των δειγμάτων ψηφιακού ήχου (digital audio samples). Αν και δεν χρειάζεται απο τις επίσημες προφιαγραφές καλό θα είναι το Format Chunk να βρίσκεται πρίν το Data Chunk. Πολλά προγράμματα περιμένουν τα chunks να είναι αποθηκευμένα με αυτή την σειρά και βολεύει επίσης περισσότερο όταν κάνουμε streaming ψηφιακού ήχου απο μία αργή γραμμική πηγή όπως το Internet. Αν το format υπήρχε μετά το data όλο το data και μετα το format θα έπρεπε να γίνουν streaming πριν η αναπαραγωγή ξεκινήσει σωστά. Όλα τα RIFF chunks και σε επακόλουθο τα Wave Chunks αποθηκεύονται με αυτό τον τρόπο.

Offset	Size	Description
0x00	4	Chunk ID
0x04	4	Chunk Data Size
0x08	Chunk Data Bytes	

Εικόνα 5.3.2 RIFF chunk Format

Παρακάτω θα αναφέρουμε τα διαφορετικά Chunks των Wave αρχείων.

5.3.3 Format Chunk - "fmt "

Το format chunk εμπεριέχει πληροφορίες για το πως είναι αποθηκευμένη η κυμματομορφή και πως θα έπρεπε να αναπαραχθεί εμπεριέχοντας και τον τύπο της τυχόν συμπίεσης, τον αριθμό των καναλιών το sample rate, το bits per sample κλπ.

Offset	Size	Description	Value
0x00	4	Chunk ID	"fmt " (0x666D7420)
0x04	4	Chunk Data Size	16 + extra format bytes
0x08	2	Compression code	1 - 65,535
0x0a	2	Number of channels	1 - 65,535
0x0c	4	Sample rate	1 - 0xFFFFFFFF
0x10	4	Average bytes per second	1 - 0xFFFFFFFF
0x14	2	Block align	1 - 65,535
0x16	2	Significant bits per sample	2 - 65,535
0x18	2	Extra format bytes	0 - 65,535
0x1a	Extra format bytes *		

Εικόνα 5.3.3 Τιμές Wave Format Chunk

Code	Description
0 (0x0000)	Unknown
1 (0x0001)	PCM/uncompressed
2 (0x0002)	Microsoft ADPCM
6 (0x0006)	ITU G.711 a-law
7 (0x0007)	ITU G.711 Âµ-law
17 (0x0011)	IMA ADPCM
20 (0x0016)	ITU G.723 ADPCM (Yamaha)
49 (0x0031)	GSM 6.10
64 (0x0040)	ITU G.721 ADPCM
80 (0x0050)	MPEG
65,536 (0xFFFF)	Experimental

Εικόνα 6.3.3.1 Common Wave Compression Codes

5.3.4 Data Chunk - "data"

Το Wave Data Chunk εμπεριέχει την πληροφορία ψηφιακού ήχου η οποία μπορεί να αποοκωδικοποιηθεί χρησιμοποιώντας το format και την μέθοδο συμπίεσης που προσδιορίζεται απο το Wave Format Chunk. Αν ο κωδικός συμπίεσης είναι 1 (uncompressed PCM), τότε το Wave Data εμπεριέχει raw sample values. Τα αρχεία Wave συνήθως έχουν ένα data chunk αλλά μπορεί να έχουν περισσότερα αν εμπεριέχονται μέσα σε μία Wave List Chunk.

Offset	Length	Type	Description	Value
0x00	4	char[4]	chunk ID	"data" (0x64617461)
0x04	4	dword	chunk size	depends on sample length and compression
0x08	sample data			

Εικόνα 5.3.4 Data Chunk Format

5.3.5 Fact Chunk - "fact"

Ένα fact chunk αποθηκεύει τον κώδικα συμπίεσης των περιεχομένων ενός Wave αρχείου. Είναι αναγκαίο για κάθε συμπιεσμένο Wave αρχείο αλλά δεν είναι αναγκαίο για ασυμπίεστα Wave αρχεία. (compression code 1) τα οποία τα στοιχεία της κυμματομορφής μέσα σε ένα data chunk.

Offset	Size	Description	Value
0x00	4	Chunk ID	"fact" (0x66616374)
0x04	4	Chunk Data Size	depends on format
0x08	Format Dependant Data		

Εικόνα 5.3.5 Fact Chunk Format

5.3.6 Wave List Chunk - "wavl"

Ένα Wave List chunk χρησιμοποείται για να ορίσει εναλλακτικά “slnt” και “data” chunks. Αυτά τα chunks μπορεί να βοηθήσουν στην μείωση του μεγέθους ενός αρχείου wave αλλά δεν ενδικνειται καθως περιπλέκουν την δομή ενός αρχείου Wave με αυτου του είδους την συμπίεση.

Offset	Size	Description	Value
0x00	4	Chunk ID	"slnt" (0x736C6E74)
0x04	4	Chunk Data Size	depends on size of data and slnt chunks
0x08	List of Alternating "slnt" and "data" Chunks		

Εικόνα 5.3.6 Wave List Chunk Format

5.3.7 Silent Chunk - "slnt"

Ένα Silent Chunk χρησιμοποιείται για να ορίσει μέρος “σιωπής’’ που θα διαρκέσει κάποια διάρκεια samples. Πάντα εμπεριέχεται σε ένα wave list chunk. Αν και αυτό το το chunk εμπεριέχει σιωπη δεν ορίζει κατ’αναγκη μηδενικό εύρος ήχου.

Offset	Size	Description	Value
0x00	4	Chunk ID	"slnt" (0x736C6E74)
0x04	4	Chunk Data Size	4
0x08	4	Number of Silent Samples	0 - 0xFFFFFFFF

Εικόνα 5.3.7 Silent Chunk Format

5.3.8 Cue Chunk - "cue "

Ένα Cue Chunk ορίζει εαν ένα ή περισσότερα offset samples έχουν χρησιμοποιηθεί ώστε να μαρκάρουν συγκεκριμένες περιοχές σε μία κυμματομορφή. Για παράδειγμα στην αρχή και στο τέλος ενός refrain ενός τραγουδιού μπορεί να υπάρχουν cue points ώστε να γίνεται πιο ευκολα ο εντοπισμός του. Το cue chunk είναι προαιρετικό και αν περιέχεται μόνο ένα cue chunk πρέπει να εμπεριέχει τις πληροφορίες για όλα τα cue points ενός wave αρχείου.

Offset	Size	Description	Value
0x00	4	Chunk ID	"cue " (0x63756520)
0x04	4	Chunk Data Size	depends on Num Cue Points
0x08	4	Num Cue Points	number of cue points in list
0x0c	List of Cue Points		

Εικόνα 5.3.8 Cue Chunk Format

Offset	Size	Description	Value
0x00	4	ID	unique identification value
0x04	4	Position	play order position
0x08	4	Data Chunk ID	RIFF ID of corresponding data chunk
0x0c	4	Chunk Start	Byte Offset of Data Chunk *
0x10	4	Block Start	Byte Offset to sample of First Channel
0x14	4	Sample Offset	Byte Offset to sample byte of First Channel

Εικόνα 5.3.8.1 Cue Point Format

5.3.9 Playlist Chunk - "plst"

Ένα playlist chunk ορίζει την σειρα που θα αναπαραχθούν τα cue points. Τα cue points όπως πρωείπαμε ορίζονται στο cue chunk κάπου αλλού μέσα στο αρχείο. Ένα playlist chunk εμπεριέχει τις πληροφορίες που χρειάζεται το αρχείο για το σημείο που θα ξεκινήσει η αναπαραγωγή, για την διάρκεια της αναπαραγωγής (σε samples), για το πόσες φορές θα αναπαραχθεί ένας τομεας πριν προχωρήσει στον επόμενο κλπ.

Offset	Size	Description	Value
0x00	4	Chunk ID	"plst" (0x736C6E74)
0x04	4	Chunk Data Size	num segments * 12
0x08	4	Number of Segments	1 - 0xFFFFFFFF
0x0a	List of Segments		

Εικόνα 5.3.9 Playlist Chunk Format

5.3.10 Associated Data List Chunk - "list"

Ένα associated List Chunk ορίζει ετικέτες κειμένου (text labels) οι οποίες σχετίζονται με τα cue points έτσι ώστε να δώσει μια θέση σε καθεμία απο αυτές.

Offset	Size	Description	Value
0x00	4	Chunk ID	"list" (0x6C696E74)
0x04	4	Chunk Data Size	depends on contained text
0x08	4	Type ID	"adtl" (0x6164746C)
0x0c	List of Text Labels and Names		

Εικόνα 5.3.10 Associated Data List Chunk Format

5.3.11 Label Chunk - "labl"

Ένα Label Chunk εμπεριέχεται πάντα μέσα σε ένα associated data list chunk. Χρησιμοποιείται έτσι ώστε να συσχετίσει ένα text label με ένα cue point. Αυτου του είδους η πληροφορία συχνά εμμφανίζεται δίπλα σε markers η flags σε προγράμματα ψηφιακής επεξεργασίας ήχου.

Offset	Size	Description	Value
0x00	4	Chunk ID	"labl" (0x6C61626C)
0x04	4	Chunk Data Size	depends on contained text
0x08	4	Cue Point ID	0 - 0xFFFFFFFF
0x0c	Text		

Εικόνα 5.3.11 Label Chunk Format

5.3.12 Note Chunk - "note"

Ένα note chunk εμπεριέχεται πάντα μέσα σε ένα associated data list chunk. Χρησιμοποιείται έτσι ώστε να συσχετίσει ένα text comment με ένα cue point. Αυτου του είδους η πληροφορία αποθηκεύεται με παρόμοιο τρόπο όπως ένα label chunk.

Offset	Size	Description	Value
0x00	4	Chunk ID	"note" (0x6E6F7465)
0x04	4	Chunk Data Size	depends on contained text
0x08	4	Cue Point ID	0 - 0xFFFFFFFF
0x0C	Text		

Εικόνα 5.3.12 Note Chunk Format

5.3.13 Labeled Text Chunk - "ltxt"

Ένα note chunk εμπεριέχεται πάντα μέσα σε ένα associated data list chunk. Χρησιμοποιείται ώστε να συνδιάσει ένα text label με μια περιοχή ή έναν τομέα της κυμματομορφής. Αυτού του είδους η πληροφορία εμφανίζεται σε προγράμματα επεξεργασίας ήχου σαν μαρκαρισμένη περιοχή μέσα σε μια κυμματομορφή.

Offset	Size	Description	Value
0x00	4	Chunk ID	"ltxt" (0x6C747874)
0x04	4	Chunk Data Size	depends on contained text
0x08	4	Cue Point ID	0 - 0xFFFFFFFF
0x0c	4	Sample Length	0 - 0xFFFFFFFF
0x10	4	Purpose ID	0 - 0xFFFFFFFF
0x12	2	Country	0 - 0xFFFF
0x14	2	Language	0 - 0xFFFF
0x16	2	Dialect	0 - 0xFFFF
0x18	2	Code Page	0 - 0xFFFF
0x1A	Text		

Εικόνα 5.3.13 Labeled Text Chunk Format

5.3.14 Sampler Chunk - "smpl"

Το Sampler Chunk εμπεριέχει πληροφορίες πολύ σημαντικές του wave αρχείου. Το πεδίο που ορίζει τον Manufacturer ορίζει τον MIDI Manufacturer Association (MMA) κωδικό για το sampler που θα δεχθεί την κυμματομορφή του αρχείου. Κάθε κατασκευαστής ενός MIDI προιόντος έχει ένα μοναδικό ID το οποίο ορίζει την εταιρεία προέλευσης. Αν κανένας συγκεκριμένος κατασκευαστής δεν έχει οριστεί τότε πρέπει να οριστεί η τιμή 0. Η τιμή αποθηκεύεται με μερικές άλλες πληροφορίες ώστε να μπορεί να γλινει η μετάφραση σε ένα Midi περιβάλλον. Το high byte δείχνει τον αριθμό των low order bytes (1 ή 3) που είναι έγκυροι κωδικοί κατασκευαστών. Για παράδειγμα η τιμή για την Digidesign θα είναι 0x01000013 (0x13) ενώ για την Microsoft be 0x30000041 (0x00, 0x00, 0x41). Στον παρακάτω πίνακα βλέπουμε την λίστα των κατασκευαστων με τους κωδικούς τους.

ID	Manufacturer
0 (0x00)	Unknown
1 (0x01)	Sequential Circuits
2 (0x02)	Big Briar
3 (0x03)	Octave / Plateau
4 (0x04)	Moog
5 (0x05)	Passport Designs
6 (0x06)	Lexicon
7 (0x07)	Kurzweil
8 (0x08)	Fender
9 (0x09)	Gulbransen
10 (0x0A)	Delta Labs
11 (0x0B)	Sound Comp.
12 (0x0C)	General Electro
13 (0x0D)	Techmar
14 (0x0E)	Matthews Research
16 (0x10)	Oberheim
17 (0x11)	PAIA
18 (0x12)	Simmons
19 (0x13)	DigiDesign
20 (0x14)	Fairlight
21 (0x15)	JL Cooper
22 (0x16)	Lowery
23 (0x17)	Lin
24 (0x18)	Emu
27 (0x1B)	Peavey
32 (0x20)	Bon Tempi
33 (0x21)	S.I.E.L.
35 (0x23)	SyntheAxe
36 (0x24)	Hohner
37 (0x25)	Crumar
38 (0x26)	Solton
39 (0x27)	Jellinghaus Ms
40 (0x28)	CTS
41 (0x29)	PPG
47 (0x2F)	Elka
54 (0x36)	Cheetah

62 (0x3E)	Waldorf
64 (0x40)	Kawai
65 (0x41)	Roland
66 (0x42)	Korg
67 (0x43)	Yamaha
68 (0x44)	Casio
70 (0x46)	Kamiya Studio
71 (0x47)	Akai
72 (0x48)	Victor
75 (0x4B)	Fujitsu
76 (0x4C)	Sony
78 (0x4E)	Teac
80 (0x50)	Matsushita
81 (0x51)	Fostex
82 (0x52)	Zoom
84 (0x54)	Matsushita
85 (0x55)	Suzuki
86 (0x56)	Fuji Sound
87 (0x57)	Acoustic Technical Laboratory

Εικόνα 5.3.14 Λίστα κωδικών των κατασκευαστων

Επίσης εμπεριέχονται οι πληροφορίες για Product, Sample Period, MIDI Unity Note, MIDI Pitch Fraction, SMPTE Format, SMPTE Offset, Sample Loops, Sampler Data, List of Sample Loops, Cue Point ID, Type, Start, End, Fraction και Play Count.

Value	SMPTE Format
0	no SMPTE offset
24	24 frames per second
25	25 frames per second
29	30 frames per second with frame dropping (30 drop)
30	30 frames per second

Εικόνα 5.3.14.1 SMPTE Format Values

Offset	Size	Description	Value
0x00	4	Cue Point ID	0 - 0xFFFFFFFF
0x04	4	Type	0 - 0xFFFFFFFF
0x08	4	Start	0 - 0xFFFFFFFF
0x0C	4	End	0 - 0xFFFFFFFF
0x10	4	Fraction	0 - 0xFFFFFFFF
0x14	4	Play Count	0 - 0xFFFFFFFF

Εικόνα 5.3.14.2 Sample Loop Format

Value	Loop Type
0	Loop forward (normal)
1	Alternating loop (forward/backward, also known as Ping Pong)
2	Loop backward (reverse)
3 - 31	Reserved for future standard types
32 - 0xFFFFFFFF	Sampler specific types (defined by manufacturer)

Εικόνα 5.3.14.3 Loop Type Values

5.3.15 Instrument Chunk - "inst"

Ένα instrument chunk χρησιμοποιείται για να περιγράψει πως θα αναπαραχθεί μια κυμματομορφή σαν όργανο. Αυτή η πληροφορία είναι πολύ σημαντική για ανταλλαγή μουσικής πληροφορίας σε sample-based μουσικά προγράμματα, όπως οι trackers και τα software wavetables. Αυτό το chunk είναι προαιρετικό και δεν μπορεί να υπάρχει πάνω απο 1 σε ένα αρχείο Wave.

Offset	Size	Description	Value
0x00	4	Chunk ID	"lbt" (0x6C747874)
0x04	4	Chunk Data Size	7
0x08	1	Unshifted Note	0 - 127
0x09	1	Fine Tune (dB)	-50 - +50
0x0A	1	Gain	-64 - +64
0x0B	1	Low Note	0 - 127
0x0C	1	High Note	0 - 127
0x0D	1	Low Velocity	1 - 127
0x0E	1	High Velocity	1 - 127

Εικόνα 5.3.15 Instrument Chunk Format

ΚΕΦΑΛΑΙΟ 6

Ο ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΤΩΝ ΦΙΛΤΡΩΝ

6.1 Γενικά

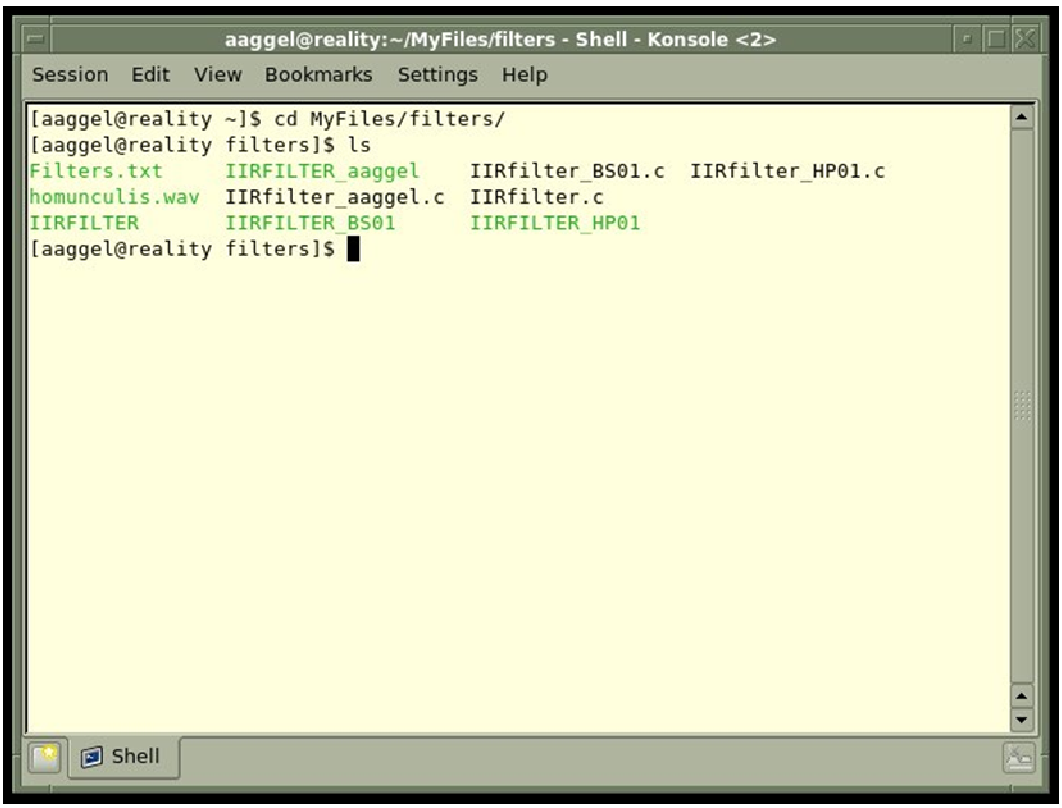
Για τον προγραμματισμό των φίλτρων και γενικα για την όλη επεξεργασία χρησημοποιήθηκε το OS Linux CentOs 5.Παρακάτω βλέπουμε την λίστα όλων των προγραμμάτων που χρησιμοποιήθηκαν για την περάτωση της εργασίας στο συγκεκριμένο λειτουργικό περιβάλλον.

- Operating System (Linux) **CentOs 5** (Red Hat)
- Terminal/Console για να δουλέψουμε με τις εντολές και τις βιβλιοθήκες του λειτουργικού συστήματος.
- Text Editor **Kwrite** για την εγγραφή του πηγαίου κώδικα.
- Gcc compiler ώστε να να κάνουμε compile το αρχείο και να είναι εκτελέσιμο αρχειο
  .exe
- Το πρόγραμμα Audacity το οποίο είναι ένα freeware πρόγραμμα για Linux ένας editor ήχου για την συχνοτική ανάλυση των audio αρχείων και γενικά για την αναπαραγωγή audio αρχείων.
- Την βιβλιοθήκη προγράμματων Sox για την απαλοιφή του Header απο το αρχείο Wav. Ώστε να διαβάζει κατευθείαν τα data δεδομένα του αρχείου.

6.2 Τα βήματα που ακολουθήσαμε.

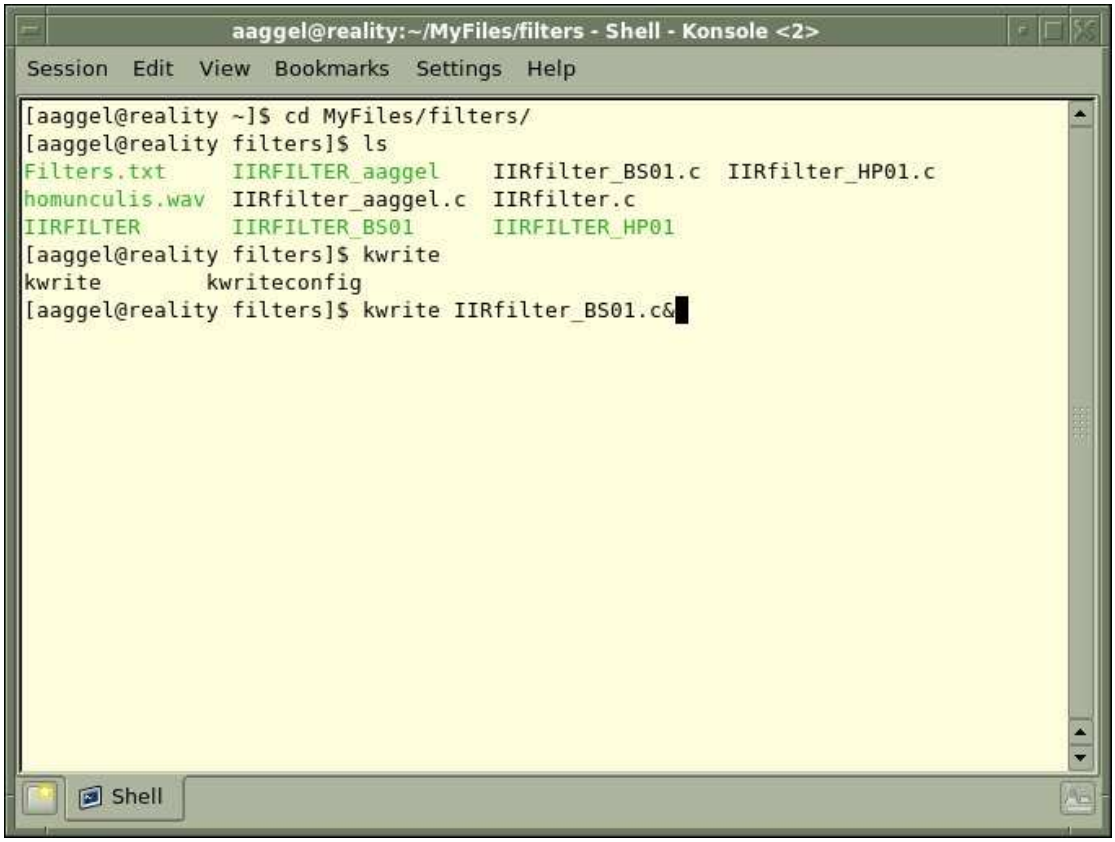
- Ανοίγουμε ένα terminal στο λειτουργικό μας σύστημα CentOs 5 ώστε να δουλέψουμε με τις εντολές μας.

- Πηγαίνουμε στον κατάλογο που έχουμε αποθυκεύσει τα δεδομένα μας (φίλτρα και αρχεία Wav) με την εντολή **cd MyFiles/Filters** ο οποίος είναι ο κατάλογος που έχουμε αποθηκεύσει τα δεδομένα μας.

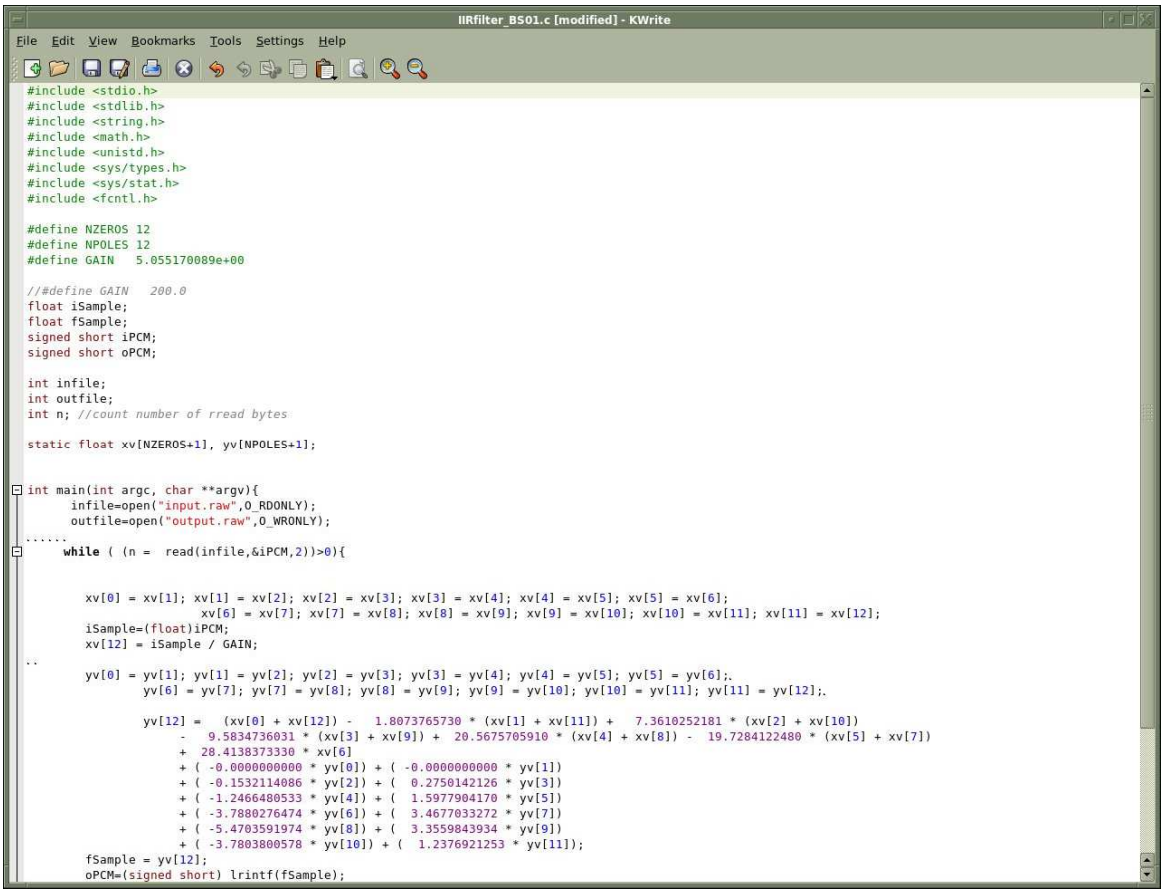


Βλέπουμε σε αυτόν τον φάκελο τα αρχεία που είναι αποθηκευμένα μέσα σε αυτόν.

- Με την εντολή **kwrite IIRfilter_BS01.c &** φορτώνουμε τον κώδικα που έχουμε γράψει στον Kwrite editor. (όπου **IIRfilter_BS01.c** το όνομα του αποθηκευμένου αρχείου.



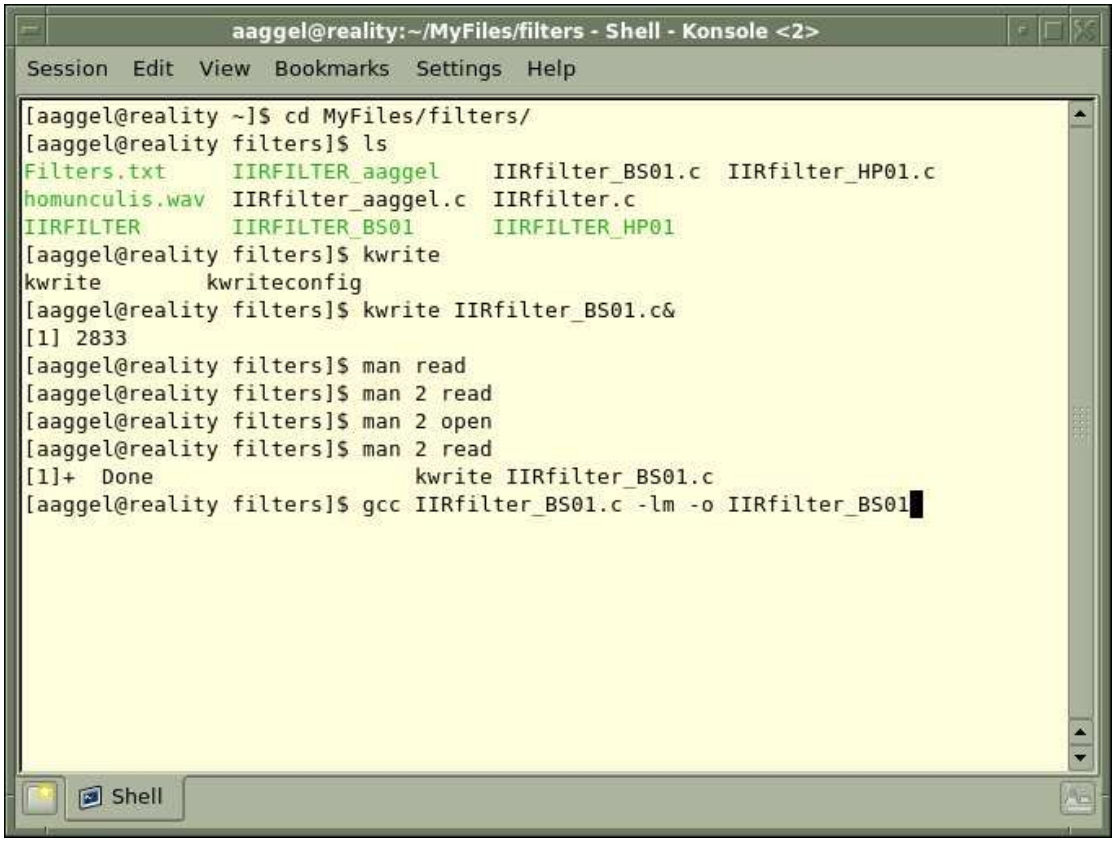
- Για την εγγραφή του πηγαίου κώδικα χρησιμοποιούμε τον editor Kwrite. Στην παρακάτω εικόνα βλέπουμε ένα screenshot απο τον Kwrite με το Band Stop φίλτρο που έχουμε προγραμματίσει σε γλώσσα C.



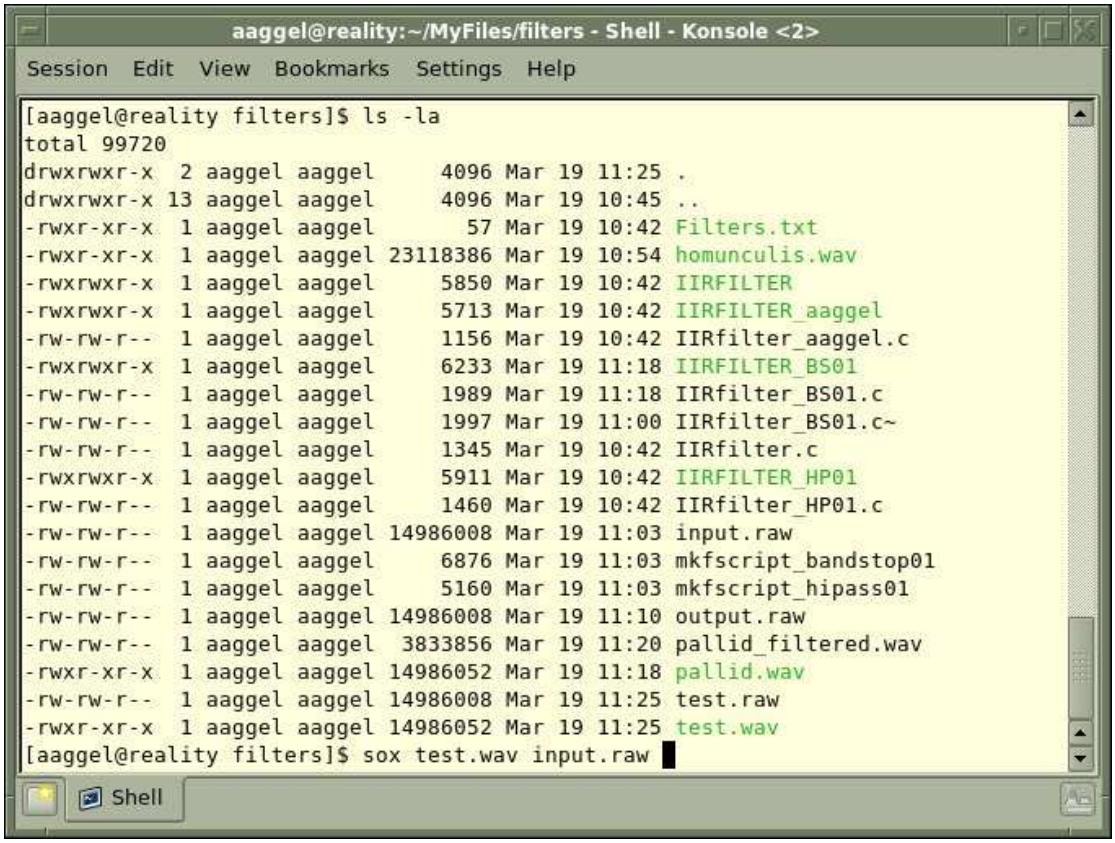
- Στο επόμενο μας βήμα χρησιμοποιούμε τον gcc compiler για να κάνουμε compile το φίλτρο μας χρησιμοποιώντας την εντολή **gcc IIRfilter_BS01.c – lm – o IIRfilter_BS01**

Σημείωση: Με την εντολή **-o** ορίζουμε τι όνομα θα έχει το αρχείο στην έξοδο μας.

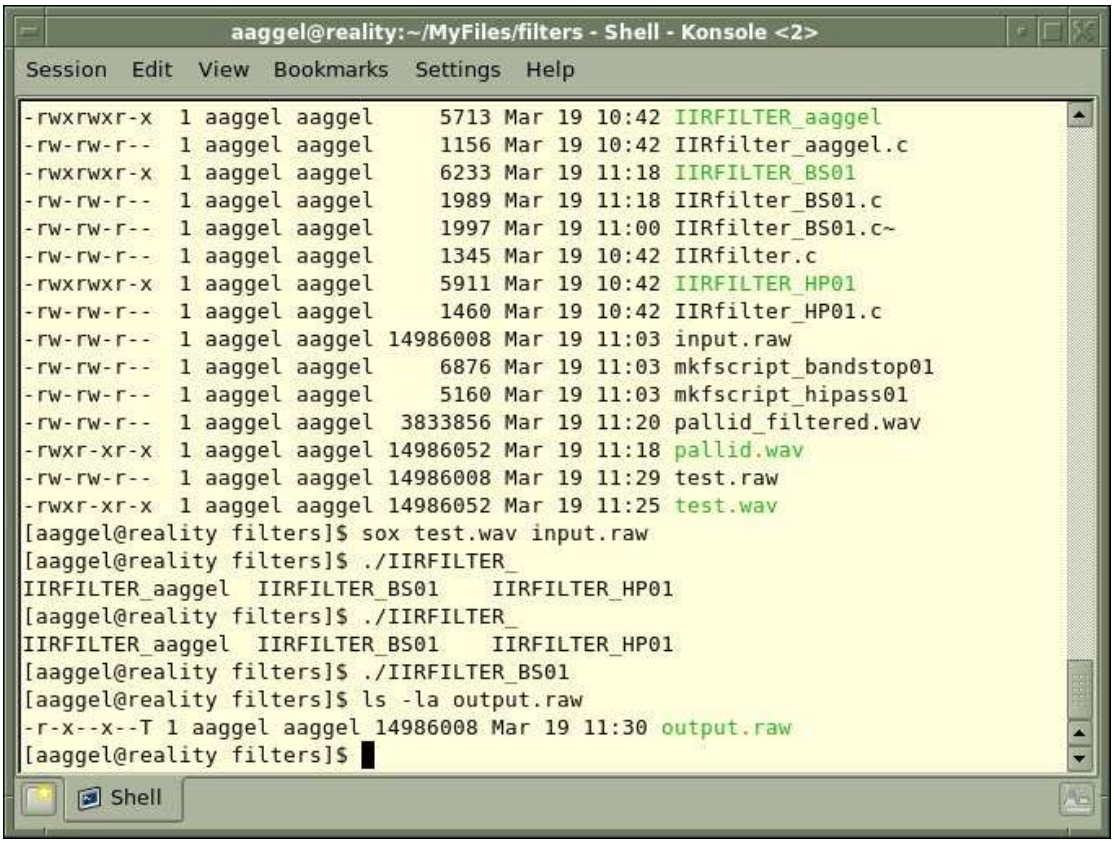
Με την εντολή **man read** και **man 2 open** ανοίγουμε το manual που υπάρχει για να δούμε τυχών εντολές αν δεν τις θυμόμαστε.



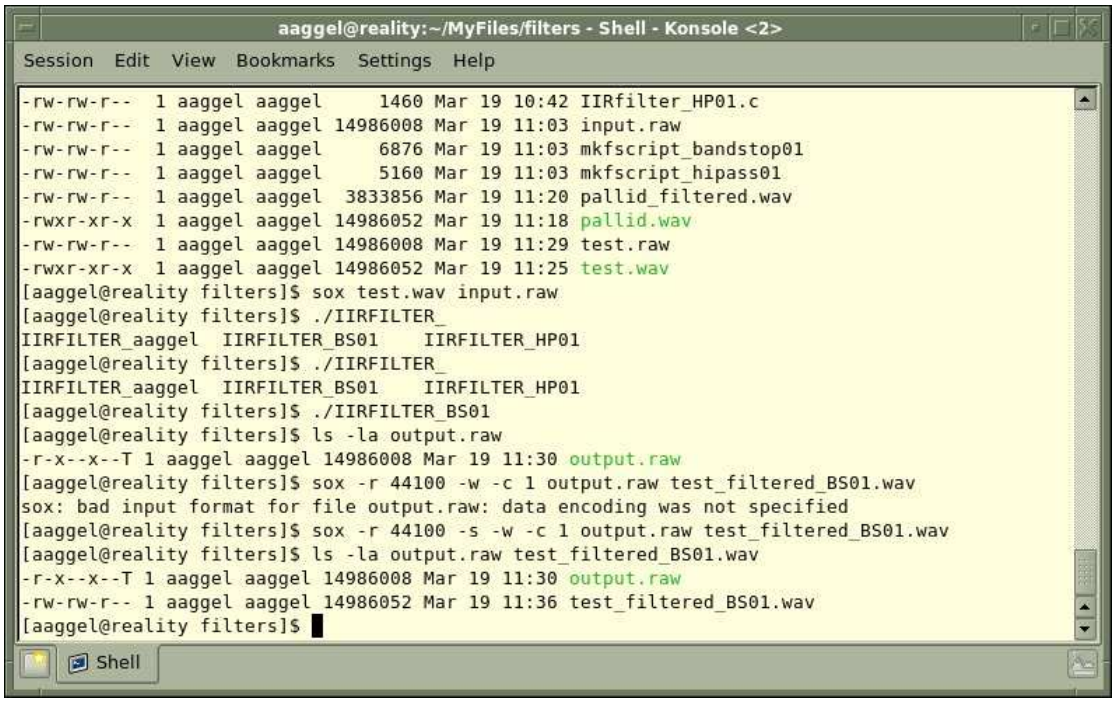
- Αφού έχουμε ήδη κάνει compile το αρχείο με τον gcc compiler θα πρέπει να αφαιρεθεί απο το wav αρχείο τον header του. Ο λόγος που το κάνουμε αυτό είναι για να διαβάσει το πρόγραμμα κατευθείαν raw data χωρίς τον data header. Για αυτόν τον λόγο θα χρησιμοποιήσουμε την εφαρμογή (αυτόνομο πρόγραμμα) Sox. Δημιουργούμε ένα backup αρχείο wav το οποίο το ονομάζουμε test.wav για να δουλέψουμε με αυτό. Με την εντολή **sox test.wav input.raw** αφαιρείται το header απο το test.wav και δημιουργείται ένα αρχείο input.raw δηλαδη το test.wav χωρίς τον header.



- Τρέχουμε το αρχείο ./IIRfilter που είχαμε κάνει compile και με την εντολή –ls –la δημιουργούμε ένα αρχείο output.raw το οποίο είναι ένα raw αρχείο στο οποίο έχει γίνει φιλτράρισμα στην προκειμένη περίπτωση Band Stop.



- Το output.raw που είναι ήδη φιλτραρισμένο πρέπει να ξαναγίνει wav αρχείο. Θα χρησιμοποιήσουμε πάλι το sox και με την εντολη **sox –r 44100 - s –w -c 1 output.raw test_filtered_BS01.wav** δημιουργούμε ένα αρχείο wav (το output.raw με τον header που είχαμε αφαιρέσει) με ονομασία test_filtered_BS01.wav . Το wav αρχείο αποθηκεύεται με αυτλην την ονομασία και είναι το αρχείο wav στο οποίο έχει εφαρμοσθεί το φίλτρο που έχουμε επιλέξει.



Η παραπάνω διαδικασία είναι η διαδικασία η οποία πρέπει να γίνει ώστε να εφαρμοσθεί ένα φίλτρο προγραμματισμένο σε γλώσσα C σε περιβάλλον εργασίας Linux/Unix απο τον χρήστη χωρίς την χρησιμοποίηση προγραμμάτων αλλά με τους δικούς του κώδικες που έχει φτιάξει ο χρήστης. Και φυσικά με freeware προγράμματα.

6.3 ΤΑ ΦΙΛΤΡΑ ΠΟΥ ΠΡΟΓΡΑΜΜΑΤΙΣΑΜΕ

6.3.1 Band Stop Filter

Το πρώτο φίλτρο που προγραμματίσαμε είναι ένα IIR φίλτρο **Butterworth** τύπου **Bandstop.**

Το φίλτρο έιναι **5ης τάξης** με **sample rate 44100** συχνότητα **1ης γωνίας** στα **8000 Hz** και **2ης γωνίας** στα **12000 Hz** με ένα **επιπλεον 0** στα **10000 Hz**

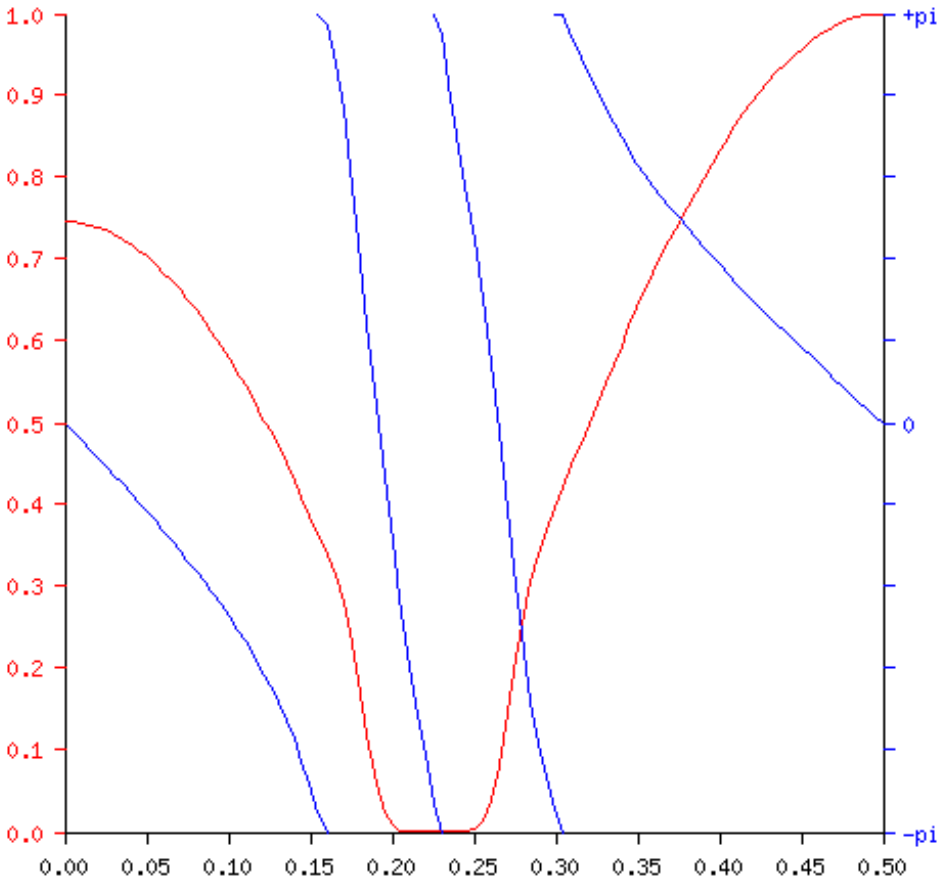
Παρακάτω είναι η δομή του φίλτρου σε γλώσσα C

```
#define NZEROS 12

#define NPOLES 12

#define GAIN   5.055170089e+00


static float xv[NZEROS+1], yv[NPOLES+1];


static void filterloop()

  { for (;;)

     { xv[0] = xv[1]; xv[1] = xv[2]; xv[2] = xv[3]; xv[3] = xv[4]; xv[4] = xv[5]; xv[5] = xv[6]; xv[6] = xv[7]; xv[7] = xv[8]; xv[8] = xv[9]; xv[9] = xv[10]; xv[10] = xv[11]; xv[11] = xv[12];

       xv[12] = next input value / GAIN;

       yv[0] = yv[1]; yv[1] = yv[2]; yv[2] = yv[3]; yv[3] = yv[4]; yv[4] = yv[5]; yv[5] = yv[6]; yv[6] = yv[7]; yv[7] = yv[8]; yv[8] = yv[9]; yv[9] = yv[10]; yv[10] = yv[11]; yv[11] = yv[12];

       yv[12] =   (xv[0] + xv[12]) -   1.8073765730 * (xv[1] + xv[11]) + 7.3610252181 * (xv[2] + xv[10])

                -   9.5834736031 * (xv[3] + xv[9]) +  20.5675705910 * (xv[4] + xv[8]) - 19.7284122480 * (xv[5] + xv[7])

                +  28.4138373330 * xv[6]
```

```
                + ( -0.0000000000 * yv[0]) + ( -0.0000000000 * yv[1])
                + ( -0.1532114086 * yv[2]) + (  0.2750142126 * yv[3])
                + ( -1.2466480533 * yv[4]) + (  1.5977904170 * yv[5])
                + ( -3.7880276474 * yv[6]) + (  3.4677033272 * yv[7])
                + ( -5.4703591974 * yv[8]) + (  3.3559843934 * yv[9])
                + ( -3.7803800578 * yv[10]) + (  1.2376921253 * yv[11]);
      next output value = yv[12];
    }
}
```

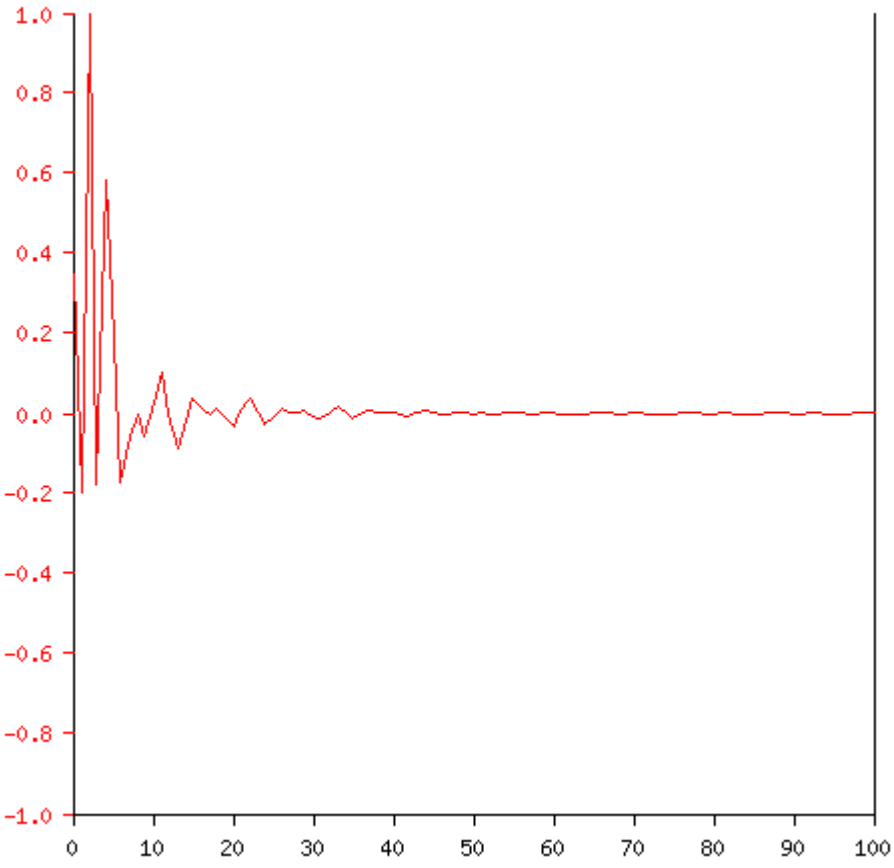
Μέγεθος και Φάση ανά Συχνότητα.

- Στον άξονα x είναι η συχνότητα σαν κλάσμα του sampling rate (πχ το 0.5 αντιπροσοπεύει την συχνότητα του Nyquist, η οποία είναι 22050 Hz)
- Στον άξονα y (κόκκινο) είναι το magnitude (γραμμικό, normalized)
- Στον άξονα y (μπλέ) είναι η φάση.



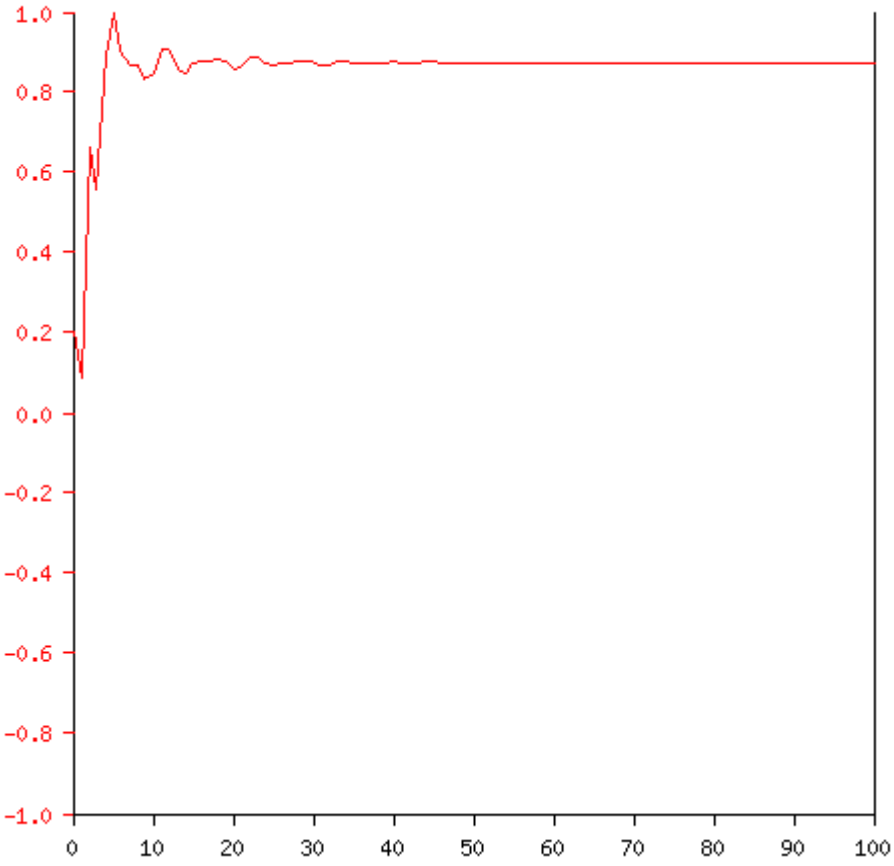
Impulse response

- Στον άξονα x βρίσκεται ο χρόνος σε samples (πχ 44100 αντιστοιχούν σε 1 second)
- Στον άξονα y (κόκκινο) το filter response (γραμμικό, normalized)



Step response

- Στόν άξονα x ο χρόνος σε Samples (πχ 44100 αντιστοιχούν σε 1 second)
- Στον άξονα y (κόκκινο) το filter response (γραμμικό, normalized)



6.3.2 Το προγραμματισμένο φίλτρο για την εφαρμογή

Παρακάτω δίνουμε το φίλτρο που προγραμματίσαμε ώστε να εφαρμοσθεί πάνω σε ένα αρχείο wav με βαση την δομη του φίλτρου και τα στοιχεία που δώσαμε παραπάνω. Ο κώδικας είναι σε γλώσσα προγραμματισμού C.

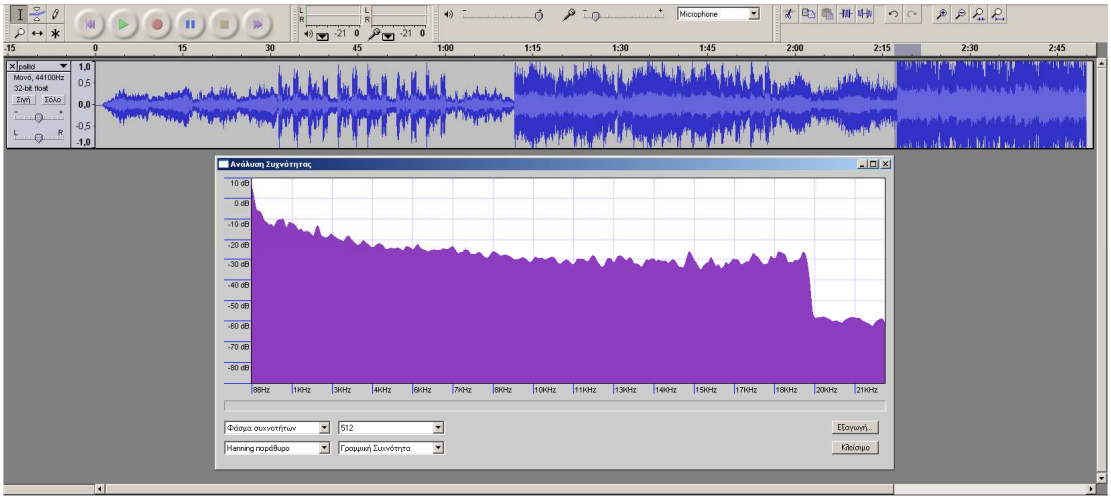
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

#define NZEROS 12
#define NPOLES 12
#define GAIN   5.055170089e+00

//#define GAIN   200.0
float iSample;
float fSample;
signed short iPCM;
signed short oPCM;

int infile;
int outfile;
int n; //count number of rread bytes

static float xv[NZEROS+1], yv[NPOLES+1];


int main(int argc, char **argv){
      infile=open("input.raw",O_RDONLY);
      outfile=open("output.raw",O_WRONLY);

     while ( (n =  read(infile,&iPCM,2))>0){


        xv[0] = xv[1]; xv[1] = xv[2]; xv[2] = xv[3];
xv[3] = xv[4]; xv[4] = xv[5]; xv[5] = xv[6];
                         xv[6] = xv[7]; xv[7] = xv[8];
xv[8] = xv[9]; xv[9] = xv[10]; xv[10] = xv[11]; xv[11] =
xv[12];
        iSample=(float)iPCM;
        xv[12] = iSample / GAIN;
```

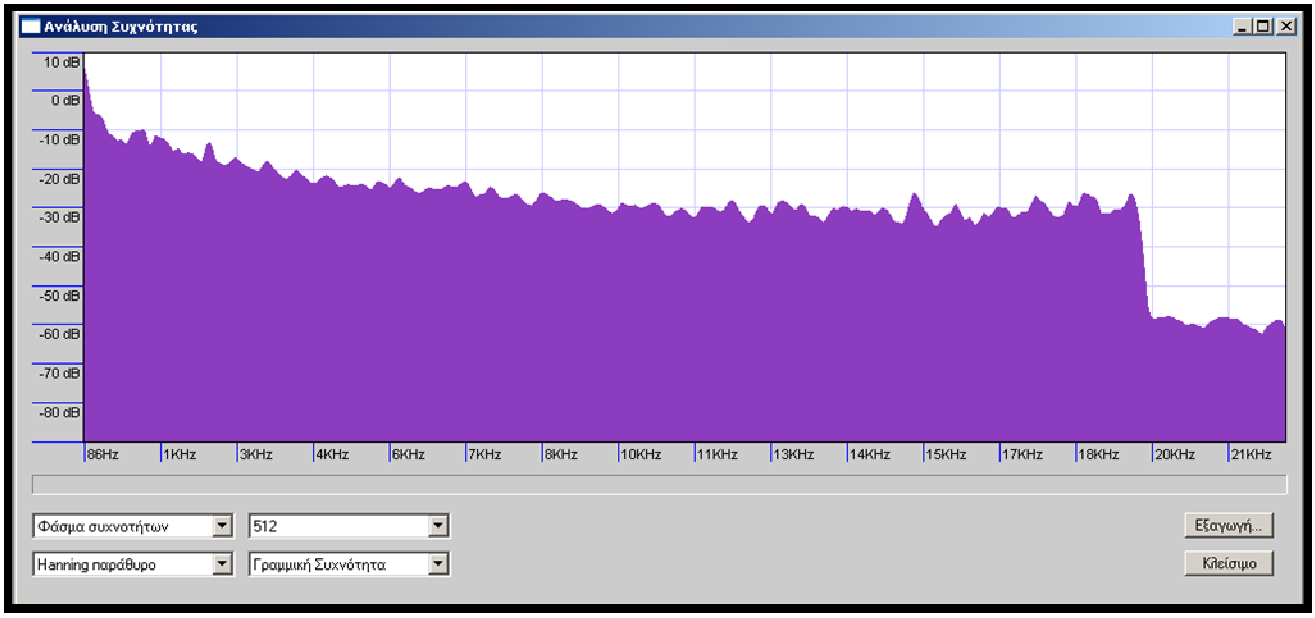
```
        yv[0] = yv[1]; yv[1] = yv[2]; yv[2] = yv[3];
yv[3] = yv[4]; yv[4] = yv[5]; yv[5] = yv[6];
                yv[6] = yv[7]; yv[7] = yv[8]; yv[8] =
yv[9]; yv[9] = yv[10]; yv[10] = yv[11]; yv[11] = yv[12];

                yv[12] =   (xv[0] + xv[12]) -
1.8073765730 * (xv[1] + xv[11]) +   7.3610252181 * (xv[2]
+ xv[10])
                    -   9.5834736031 * (xv[3] + xv[9]) +
20.5675705910 * (xv[4] + xv[8]) -  19.7284122480 * (xv[5]
+ xv[7])
                    +  28.4138373330 * xv[6]
                    + ( -0.0000000000 * yv[0]) + ( -
0.0000000000 * yv[1])
                    + ( -0.1532114086 * yv[2]) + (
0.2750142126 * yv[3])
                    + ( -1.2466480533 * yv[4]) + (
1.5977904170 * yv[5])
                    + ( -3.7880276474 * yv[6]) + (
3.4677033272 * yv[7])
                    + ( -5.4703591974 * yv[8]) + (
3.3559843934 * yv[9])
                    + ( -3.7803800578 * yv[10]) + (
1.2376921253 * yv[11]);
        fSample = yv[12];
        oPCM=(signed short) lrintf(fSample);


      write(outfile,&oPCM,2);
      }
     close(infile);
     close(outfile);
}
```

Ο παραπάνω είναι ο κώδικας είναι το band stop φίλτρο γραμμένο σε C ο οποίος γίνεται compile απο τον gcc compiler του Linux και εφαρμόζεται πάνω σε οποιοδήποτε αρχείο wav.

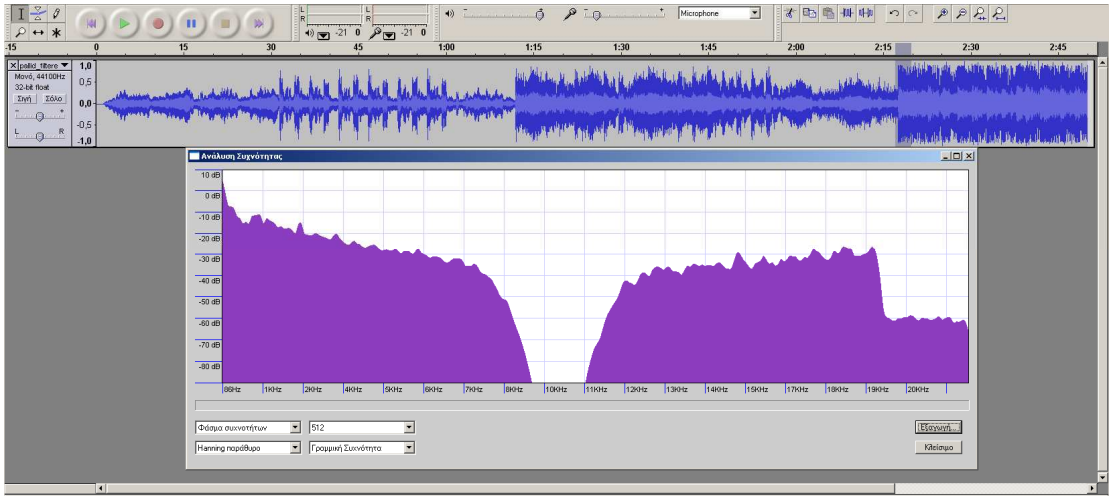
Το band stop φίλτρο μας εφαρμόσθηκε πάνω σε ένα αρχείο wav. Με το πρόγραμμα Audacity για Linux θα δούμε μια συχνοτική ανάλυση του αρχείου μας πρίν την εφαρμογή αλλά και μετα την εφαρμογή του φίλτρου.

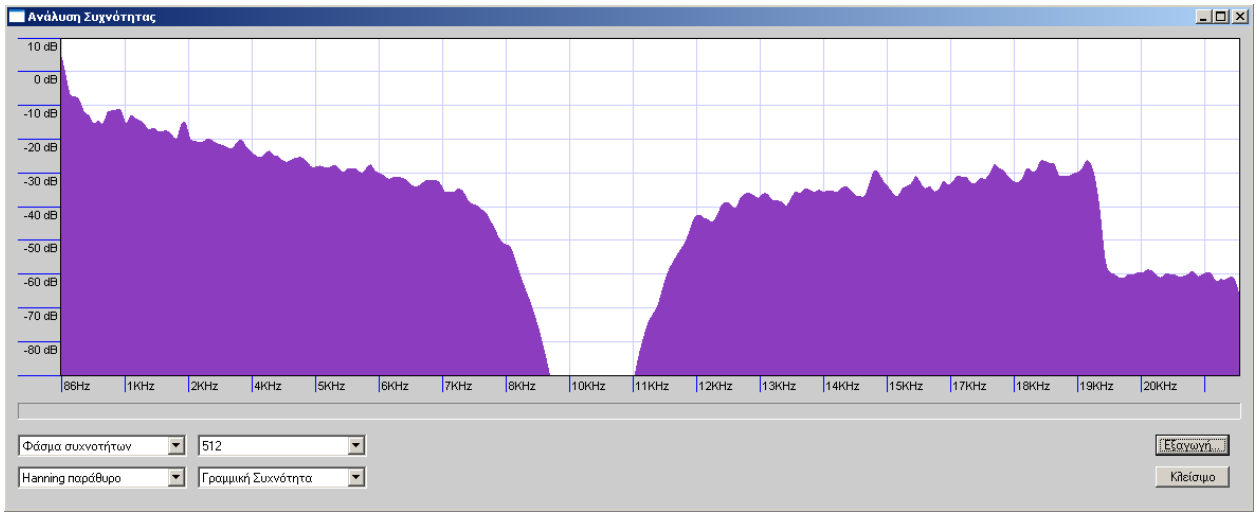
Παρακάτω βλέπουμε την συχνοτική ανάλυση σε μια περιοχή του αρχείου wav πριν την εφαρμογή του φίλτρου.





Έπειτα βλέπουμε το τελικό μας αρχείο wav μετα την εφαρμογή του φίλτρου.





6.3.3 IIR (Butterworth) High Pass Filter

Το δεύτερο φίλτο που προγραμματίσαμε είναι ένα IIR φίλτρο **Butterworth** τύπου **High Pass.**

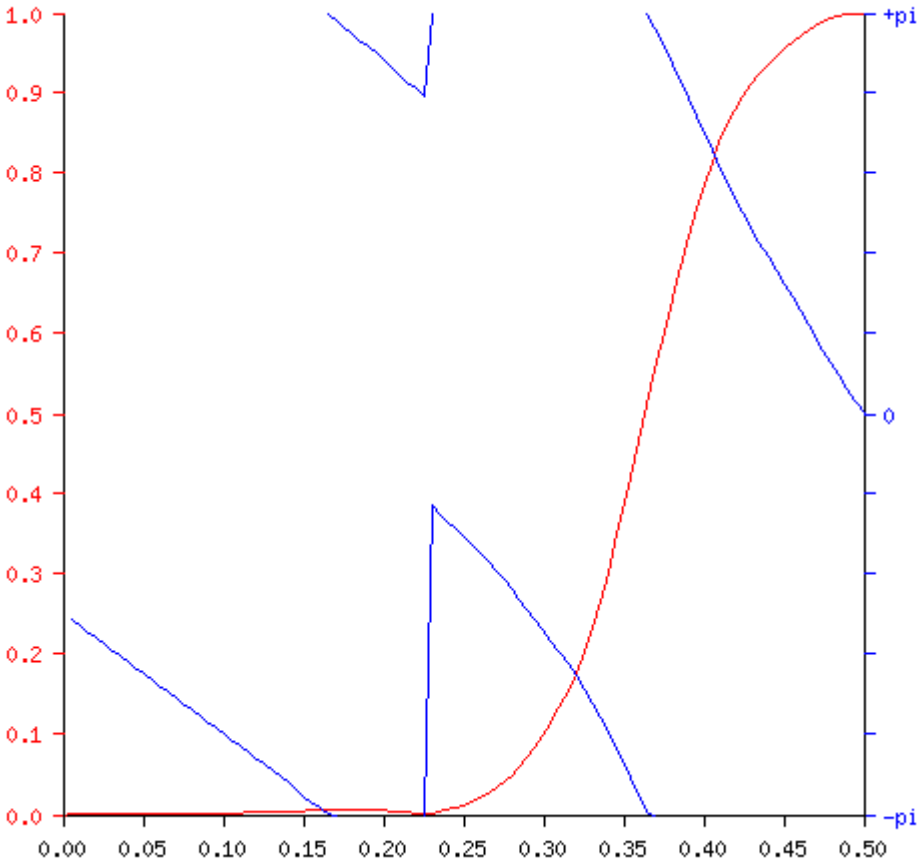
Το φίλτρο είναι **3ης τάξης** με **sample rate 44100**, συχνότητα **1ης γωνίας 16000 Hz** και με ένα **επιπλέον 0** στα **10000 Hz**

Παρακάτω είναι η δομή του φίλτρου σε γλώσσα C

```
#define NZEROS 5

#define NPOLES 5

#define GAIN   5.749266068e+01


static float xv[NZEROS+1], yv[NPOLES+1];


static void filterloop()

  { for (;;)

     { xv[0] = xv[1]; xv[1] = xv[2]; xv[2] = xv[3]; xv[3] = xv[4]; xv[4] = xv[5];

       xv[5] = next input value / GAIN;

       yv[0] = yv[1]; yv[1] = yv[2]; yv[2] = yv[3]; yv[3] = yv[4]; yv[4] = yv[5];

       yv[5] =   (xv[5] - xv[0]) +   3.2910384304 * (xv[1] - xv[4]) +   4.8731152912 *
(xv[3] - xv[2])

                  + ( -0.0000000000 * yv[0]) + ( -0.0000000000 * yv[1])

                  + ( -0.1664152505 * yv[2]) + ( -0.7989247266 * yv[3])

                  + ( -1.3137156123 * yv[4]);

       next output value = yv[5];

     }

}
```

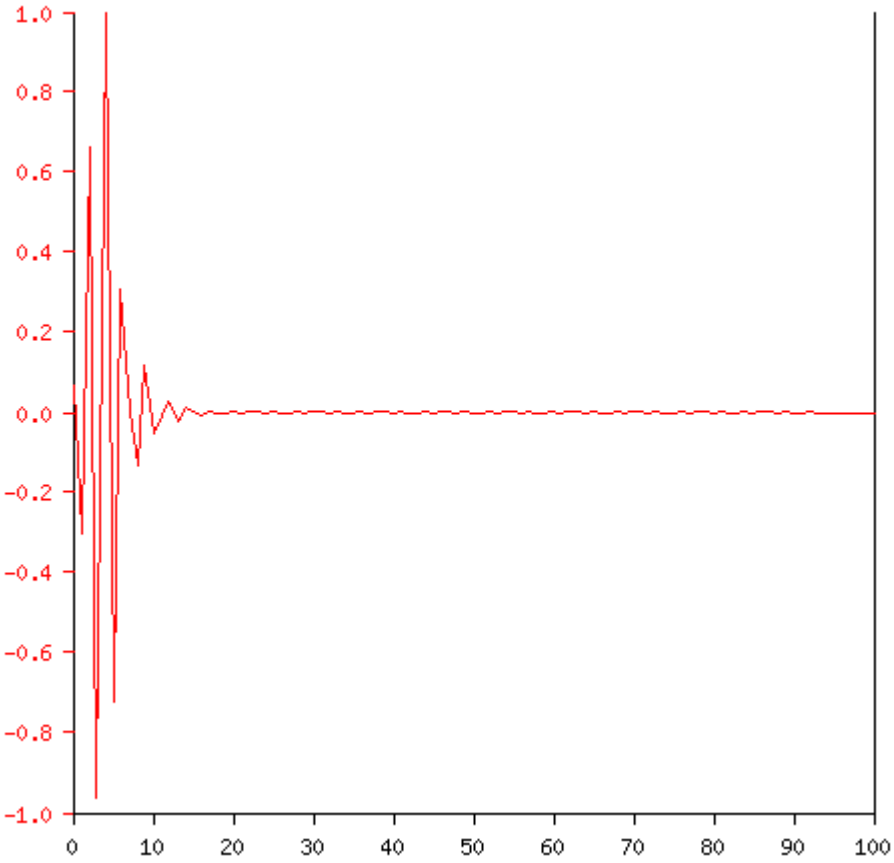
Μέγεθος και Φάση ανά Συχνότητα.

- Στον άξονα x είναι η συχνότητα σαν κλάσμα του sampling rate (πχ το 0.5 αντιπροσοπεύει την συχνότητα του Nyquist, η οποία είναι 22050 Hz)
- Στον άξονα y (κόκκινο) είναι το magnitude (γραμμικό, normalized)
- Στον άξονα y (μπλέ) είναι η φάση.



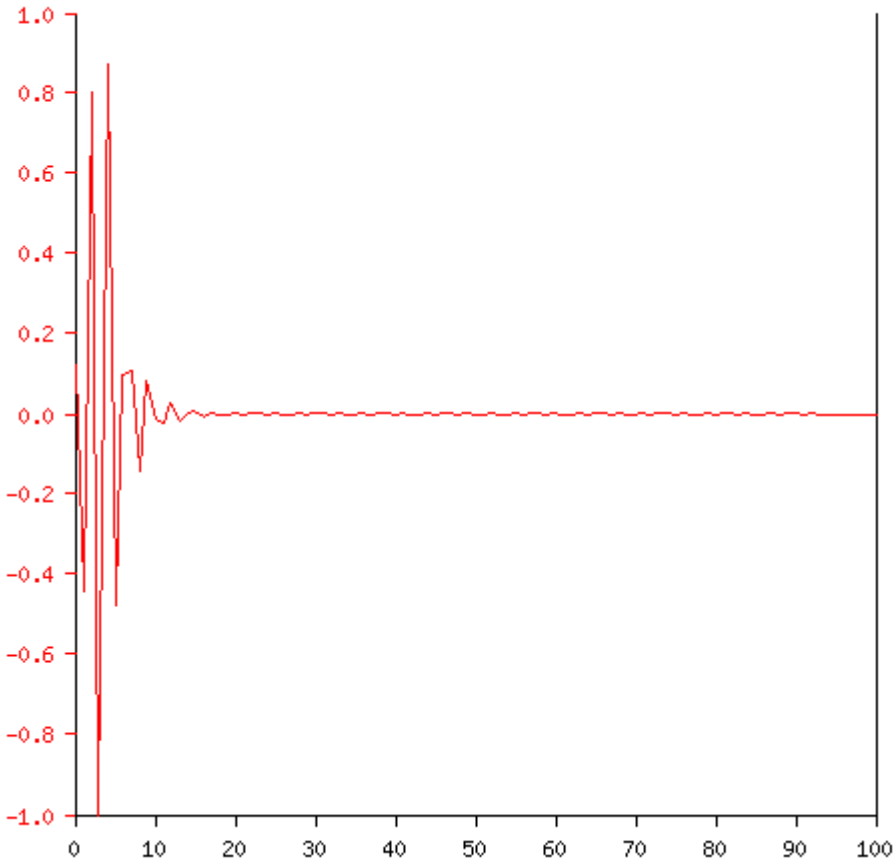
Impulse response

- Στον άξονα x βρίσκεται ο χρόνος σε samples (πχ 44100 αντιστοιχούν σε 1 second)
- Στον άξονα y (κόκκινο) το filter response (γραμμικό, normalized)



Step response

- Στόν άξονα x ο χρόνος σε Samples (πχ 44100 αντιστοιχούν σε 1 second)
- Στον άξονα y (κόκκινο) το filter response (γραμμικό, normalized)



6.3.4 Το προγραμματισμένο φίλτρο για την εφαρμογή

Παρακάτω δίνουμε το φίλτρο που προγραμματίσαμε ώστε να εφαρμοσθεί πάνω σε ένα αρχείο wav με βαση την δομη του φίλτρου και τα στοιχεία που δώσαμε παραπάνω. Ο κώδικας είναι σε γλώσσα προγραμματισμού C.

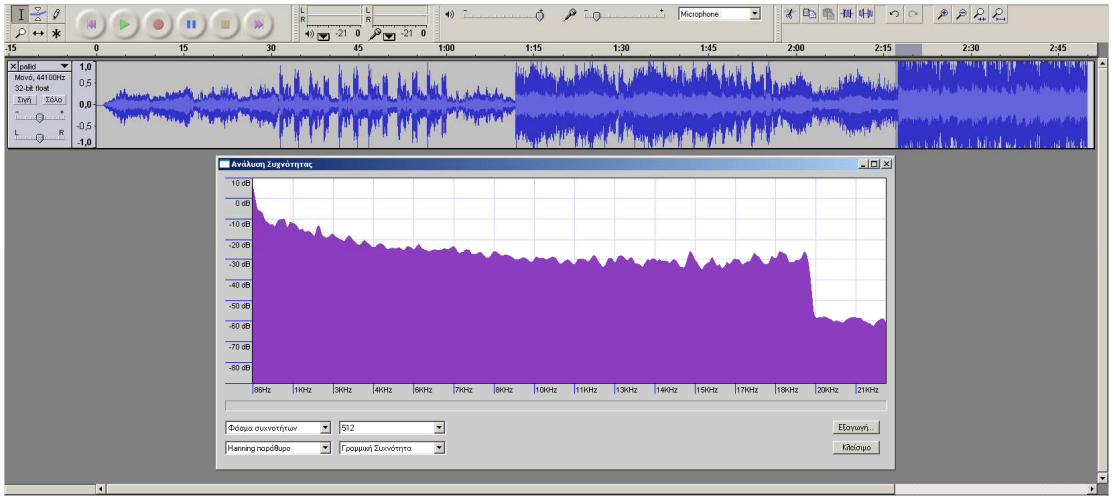
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

#define NZEROS 5
#define NPOLES 5
#define GAIN   5.749266068e+01


//#define GAIN   200.0
float iSample;
float fSample;
signed short iPCM;
signed short oPCM;

int infile;
int outfile;
int n; //count number of rread bytes

static float xv[NZEROS+1], yv[NPOLES+1];


int main(int argc, char **argv){
      infile=open("input.raw",O_RDONLY);
      outfile=open("output.raw",O_WRONLY);

     while ( (n =  read(infile,&iPCM,2))>0){


        xv[0] = xv[1]; xv[1] = xv[2]; xv[2] = xv[3];
xv[3] = xv[4]; xv[4] = xv[5];
```

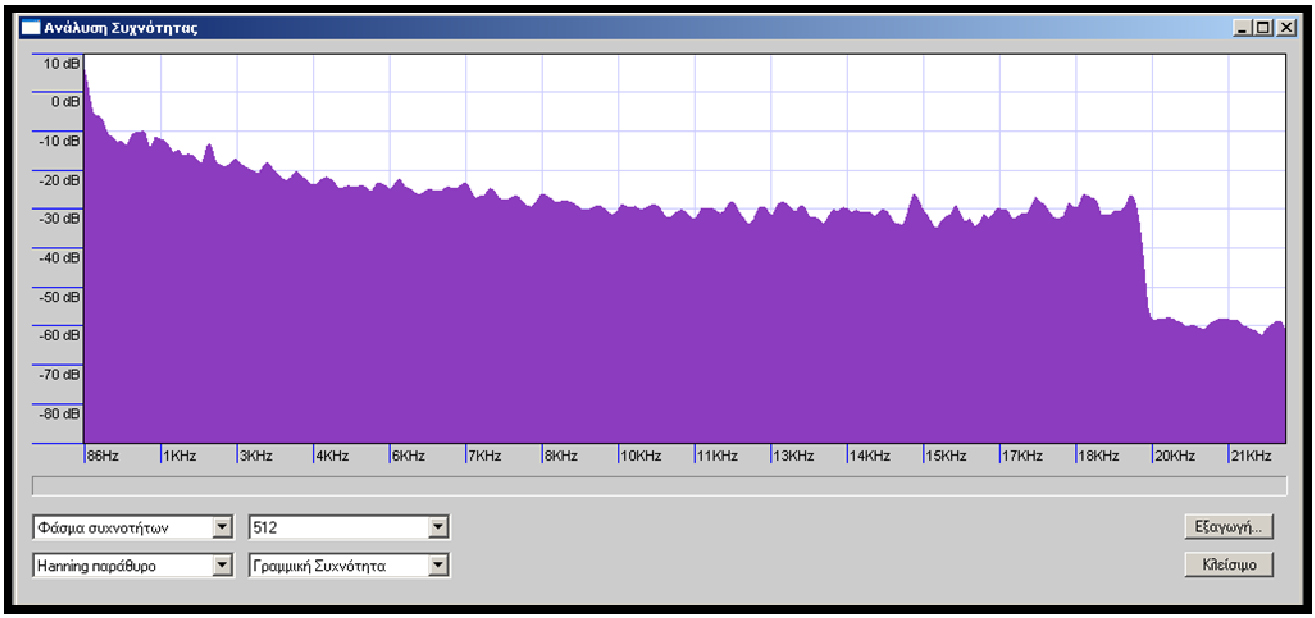
```
        iSample=(float)iPCM;
        xv[5] = iSample / GAIN;
        yv[0] = yv[1]; yv[1] = yv[2]; yv[2] = yv[3];
yv[3] = yv[4]; yv[4] = yv[5];
        yv[5] =   (xv[5] - xv[0]) +   3.2910384304 *
(xv[1] - xv[4]) +   4.8731152912 * (xv[3] - xv[2])
                     + ( -0.0000000000 * yv[0]) + ( -
0.0000000000 * yv[1])
                     + ( -0.1664152505 * yv[2]) + ( -
0.7989247266 * yv[3])
                     + ( -1.3137156123 * yv[4]);
        fSample = yv[5];
        oPCM=(signed short) lrintf(fSample);


      write(outfile,&oPCM,2);
      }
     close(infile);
     close(outfile);
}
```

Ο παραπάνω είναι ο κώδικας είναι το high pass φίλτρο γραμμένο σε C ο οποίος γίνεται compile απο τον gcc compiler του Linux και εφαρμόζεται πάνω σε οποιοδήποτε αρχείο wav.

Το high pass φίλτρο μας εφαρμόσθηκε πάνω σε ένα αρχείο wav. Με το πρόγραμμα Audacity για Linux θα δούμε μια συχνοτική ανάλυση του αρχείου μας πρίν την εφαρμογή αλλά και μετα την εφαρμογή του φίλτρου.

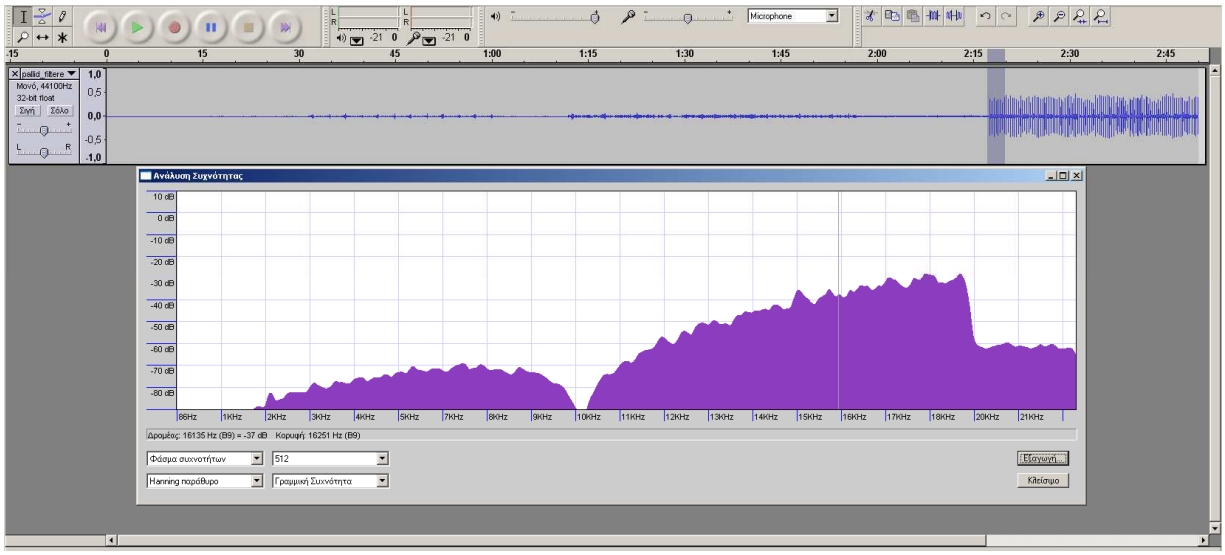
Παρακάτω βλέπουμε την συχνοτική ανάλυση σε μια περιοχή του αρχείου wav πριν την εφαρμογή του φίλτρου.

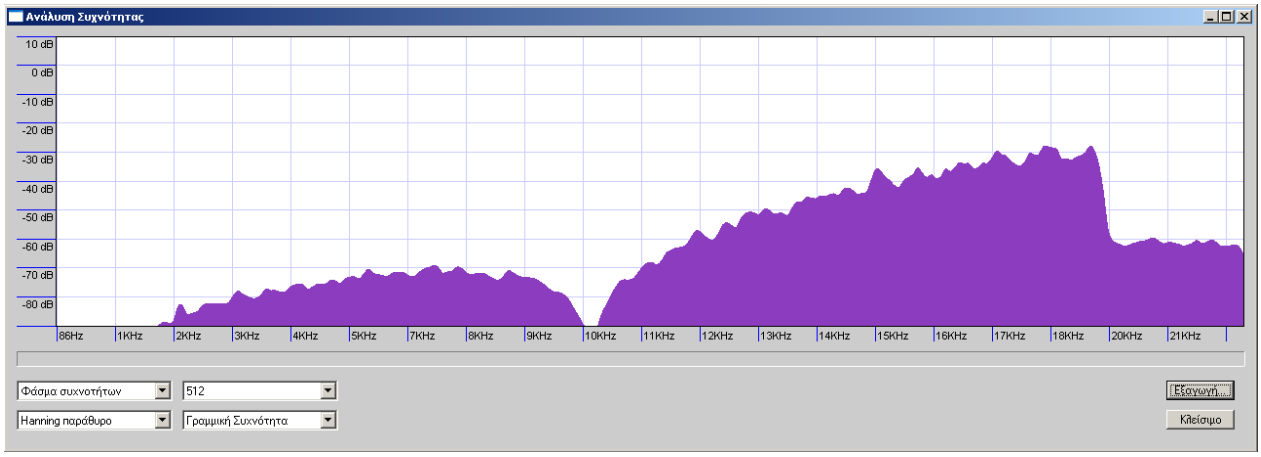




Και επίσης το φάσμα συχνοτήτων

Έπειτα βλέπουμε το τελικό μας αρχείο wav μετα την εφαρμογή του φίλτρου.





6.3.5 IIR (Butterworth) Band Pass Filter

Το επόμενο φίλτρο που προγραμματίσαμε είναι ένα φίλτρο IIR φίλτρο Butterworth τύπου Bandpass.

Το φίλτρο έιναι **6ης τάξης** με **sample rate 44100** συχνότητα **1ης γωνίας** στα **4000 Hz** και **2ης γωνίας** στα **8000 Hz .**

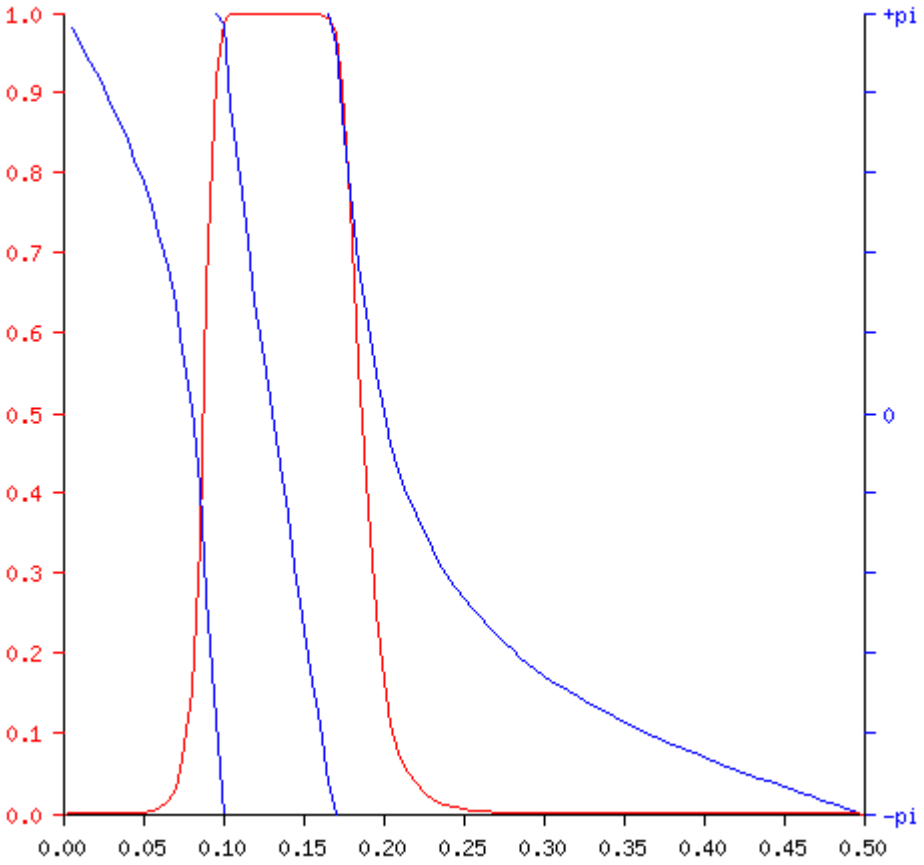
Παρακάτω βλέπουμε την ρουτίνα του φίλτρου με τα παραπάνω χαρακτηριστικά σε γλώσσα C.

```
#define NZEROS 12

#define NPOLES 12

#define GAIN   4.853283544e+03


static float xv[NZEROS+1], yv[NPOLES+1];


static void filterloop()

  { for (;;)

     { xv[0] = xv[1]; xv[1] = xv[2]; xv[2] = xv[3]; xv[3] = xv[4]; xv[4] = xv[5]; xv[5] = xv[6]; xv[6] = xv[7]; xv[7] = xv[8]; xv[8] = xv[9]; xv[9] = xv[10]; xv[10] = xv[11]; xv[11] = xv[12];

       xv[12] = next input value / GAIN;

       yv[0] = yv[1]; yv[1] = yv[2]; yv[2] = yv[3]; yv[3] = yv[4]; yv[4] = yv[5]; yv[5] = yv[6]; yv[6] = yv[7]; yv[7] = yv[8]; yv[8] = yv[9]; yv[9] = yv[10]; yv[10] = yv[11]; yv[11] = yv[12];

       yv[12] =   (xv[0] + xv[12]) - 6 * (xv[2] + xv[10]) + 15 * (xv[4] + xv[8])

                  - 20 * xv[6]

                  + ( -0.1063859002 * yv[0]) + (  1.0329127707 * yv[1])

                  + ( -5.0769073922 * yv[2]) + ( 16.3256693250 * yv[3])

                  + (-37.9384193630 * yv[4]) + ( 66.7827640450 * yv[5])

                  + (-91.1118721670 * yv[6]) + ( 97.0029505050 * yv[7])
```

```
            + (-80.0718209110 * yv[8]) + ( 50.0980857960 * yv[9])
            + (-22.6624265590 * yv[10]) + (  6.7046280606 * yv[11]);
    next output value = yv[12];
  }
}
```

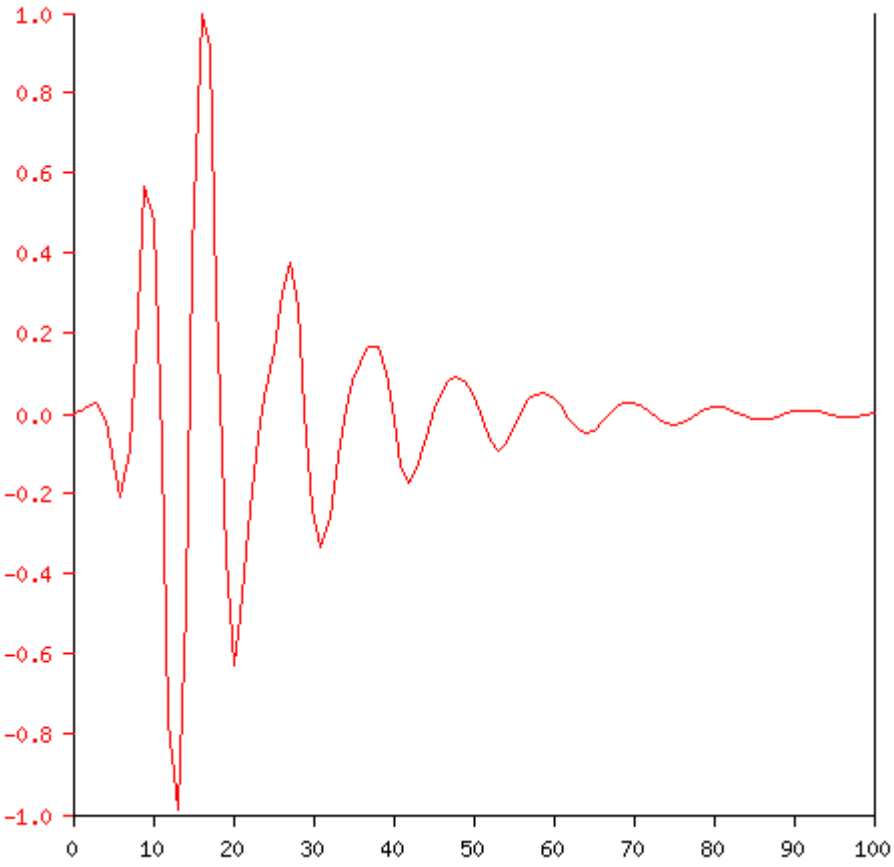
Μέγεθος και Φάση ανά Συχνότητα.

- Στον άξονα x είναι η συχνότητα σαν κλάσμα του sampling rate (πχ το 0.5 αντιπροσωπεύει την συχνότητα του Nyquist, η οποία είναι 22050 Hz)
- Στον άξονα y (κόκκινο) είναι το magnitude (γραμμικό, normalized)
- Στον άξονα y (μπλε) είναι η φάση.



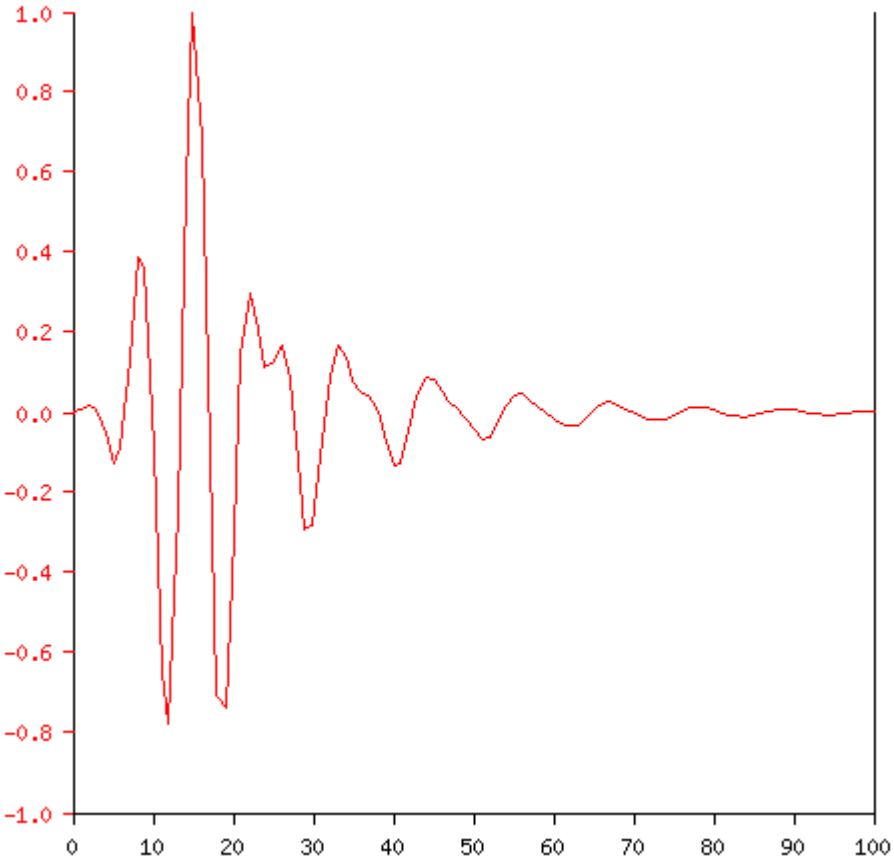
Impulse response

- Στον άξονα x βρίσκεται ο χρόνος σε samples (πχ 44100 αντιστοιχούν σε 1 second)
- Στον άξονα y (κόκκινο) το filter response (γραμμικό, normalized)



Step response

- Στόν άξονα x ο χρόνος σε Samples (πχ 44100 αντιστοιχούν σε 1 second)
- Στον άξονα y (κόκκινο) το filter response (γραμμικό, normalized)



6.3.6 Το προγραμματισμένο φίλτρο για την εφαρμογή

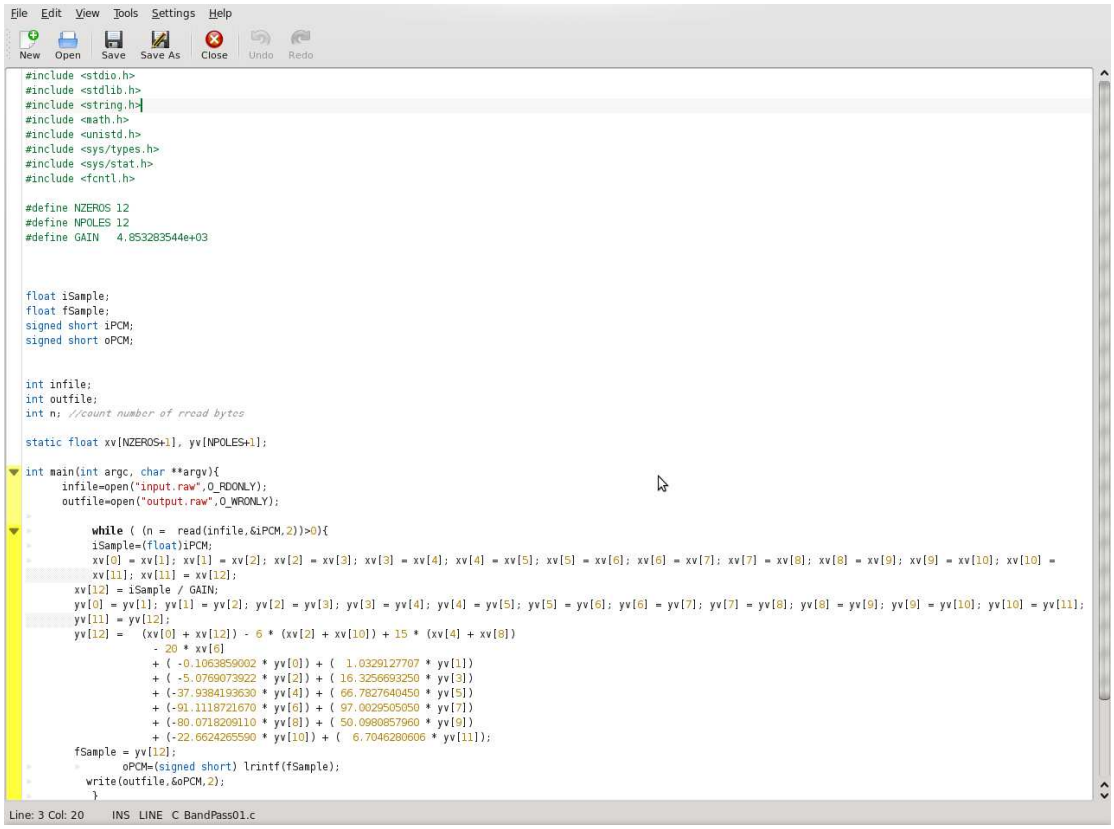
Παρακάτω δίνουμε το φίλτρο που προγραμματίσαμε ώστε να εφαρμοσθεί πάνω σε ένα αρχείο wav με βαση την δομη του φίλτρου και τα στοιχεία που δώσαμε παραπάνω. Ο κώδικας είναι σε γλώσσα προγραμματισμού C.

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <math.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>


#define NZEROS 12

#define NPOLES 12

#define GAIN   4.853283544e+03




float iSample;

float fSample;

signed short iPCM;

signed short oPCM;



int infile;
```

```
int outfile;
int n; //count number of rread bytes

static float xv[NZEROS+1], yv[NPOLES+1];

int main(int argc, char **argv){
    infile=open("input.raw",O_RDONLY);
    outfile=open("output.raw",O_WRONLY);

        while ( (n =  read(infile,&iPCM,2))>0){
        iSample=(float)iPCM;
        xv[0] = xv[1]; xv[1] = xv[2]; xv[2] = xv[3]; xv[3] = xv[4]; xv[4] = xv[5]; xv[5] = xv[6]; xv[6] = xv[7]; xv[7] = xv[8]; xv[8] = xv[9]; xv[9] = xv[10]; xv[10] = xv[11]; xv[11] = xv[12];
      xv[12] = iSample / GAIN;
      yv[0] = yv[1]; yv[1] = yv[2]; yv[2] = yv[3]; yv[3] = yv[4]; yv[4] = yv[5]; yv[5] = yv[6]; yv[6] = yv[7]; yv[7] = yv[8]; yv[8] = yv[9]; yv[9] = yv[10]; yv[10] = yv[11]; yv[11] = yv[12];
      yv[12] =   (xv[0] + xv[12]) - 6 * (xv[2] + xv[10]) + 15 * (xv[4] + xv[8])
            - 20 * xv[6]
            + ( -0.1063859002 * yv[0]) + (  1.0329127707 * yv[1])
            + ( -5.0769073922 * yv[2]) + ( 16.3256693250 * yv[3])
            + (-37.9384193630 * yv[4]) + ( 66.7827640450 * yv[5])
            + (-91.1118721670 * yv[6]) + ( 97.0029505050 * yv[7])
            + (-80.0718209110 * yv[8]) + ( 50.0980857960 * yv[9])
            + (-22.6624265590 * yv[10]) + (  6.7046280606 * yv[11]);
      fSample = yv[12];
              oPCM=(signed short) lrintf(fSample);
```

```
        write(outfile,&oPCM,2);
         }


       close(infile);
  close(outfile);
       }
```

Ο παραπάνω είναι ο κώδικας είναι το band pass φίλτρο γραμμένο σε C ο οποίος γίνεται compile απο τον gcc compiler του Linux και εφαρμόζεται πάνω σε οποιοδήποτε αρχείο wav. Παρακάτω βλέπουμε τον κώδικα στον Kwrite editor τον οποίο χρησιμοποιήσαμε για να γράψουμε τον κώδικα.

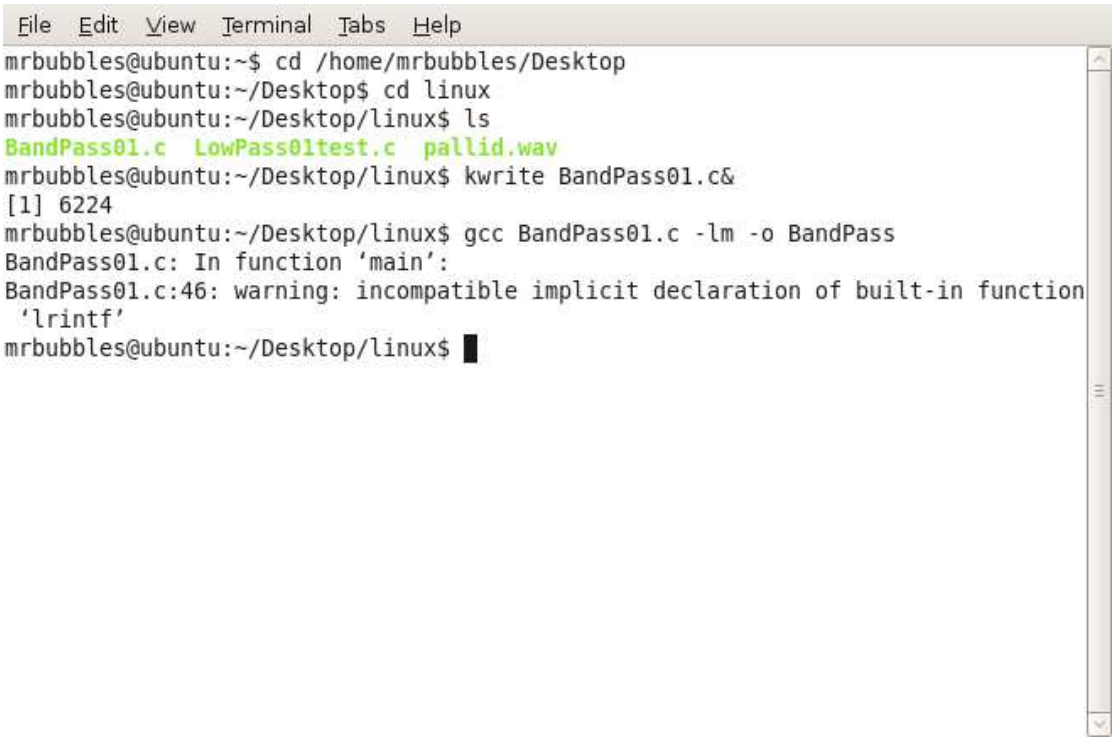


6.3.7 Τα Βήματα που ακολουθήσαμε

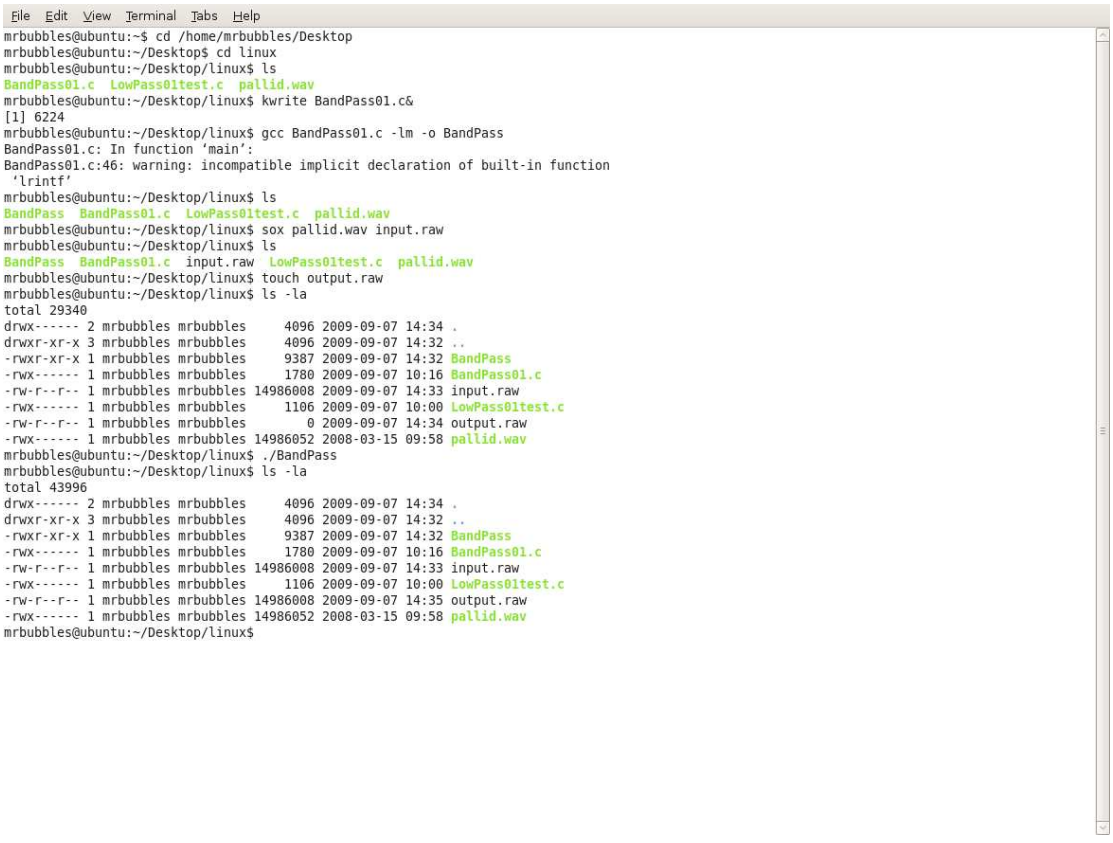
Στο παραπάνω φίλτρο χρησιμοποιήσαμε μια άλλη έκδοση Linux την Ubuntu. Τα βήματα τα οποία ακολουθήσαμε είναι τα παρακάτω:

- Ανοίγουμε ένα terminal στο λειτουργικό μας και με την **εντολή cd /home/mrbubbles/Desktop/Filtra** ανοίγουμε το Directory και βλέπουμε που έχουμε αποθηκεύσει τα φίλτρα μας . Με την εντολή **ls** βλέπουμε τα περιεχόμενα του φακέλου.

- Με την εντολή **gcc Bandpass01.c - lm –o Bandpass** κάνουμε compile το φίλτρο και δημιουργούμε ένα αρχειο exe με την ονομασία Bandpass



- Αφού έχουμε ήδη κάνει compile το αρχείο με τον gcc compiler θα πρέπει να αφαιρεθεί απο το wav αρχείο τον header του. Ο λόγος που το κάνουμε αυτό είναι για να διαβάσει το πρόγραμμα κατευθείαν raw data χωρίς τον data header. Για αυτόν τον λόγο θα χρησιμοποιήσουμε την βιβλιοθήκη Sox. Με την εντολή **sox pallid.wav input.raw** αφαιρείται το header απο το pallid.wav και δημιουργείται ένα αρχείο input.raw δηλαδη το pallid.wav χωρίς τον header. Επίσης με την εντολή **touch output.raw** δημιουργούμε ένα raw αρχείο οποίο θα επανατοποθετήσουμε το header μετά την εφαρμογή του φίλτρου.

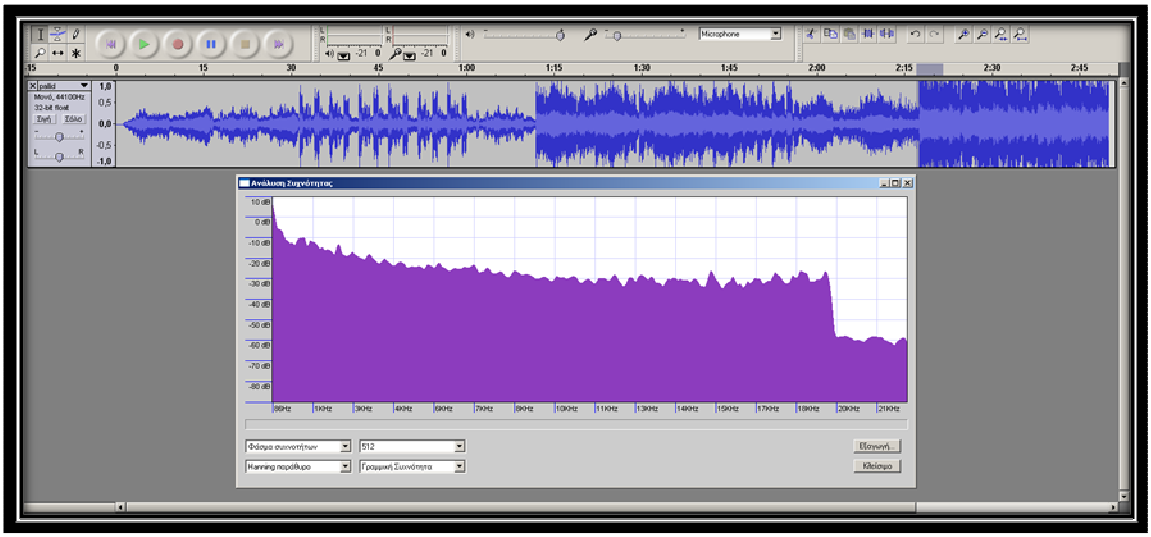


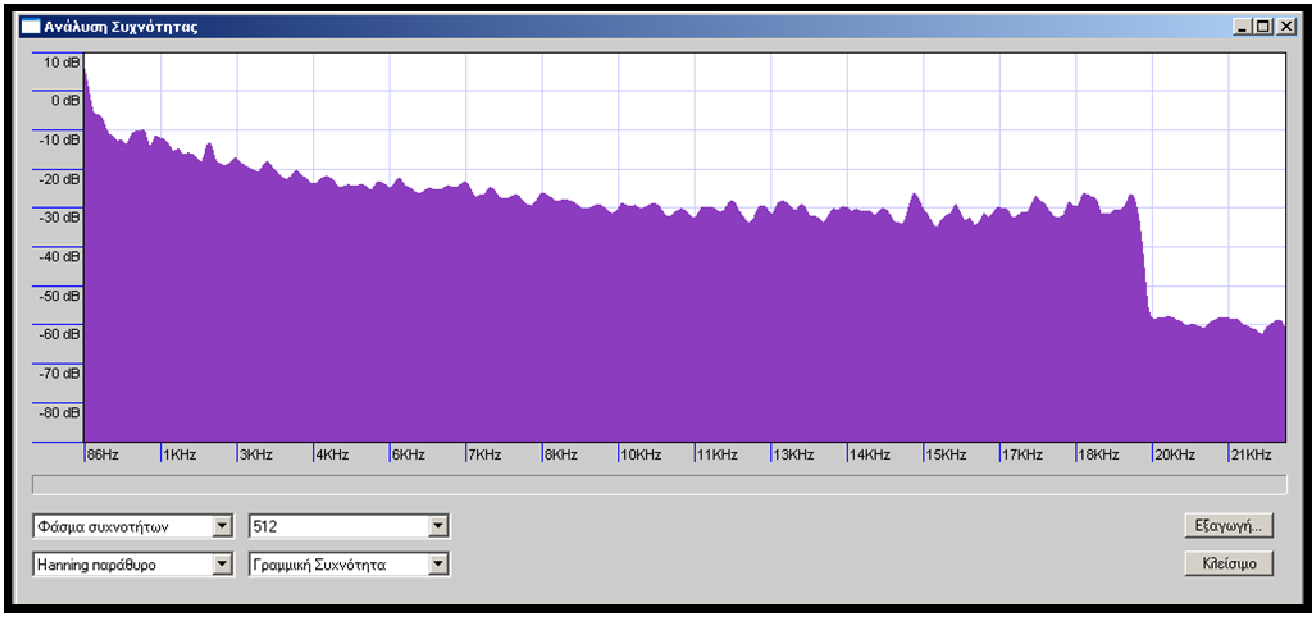
- Το output.raw που είναι ήδη φιλτραρισμένο πρέπει να ξαναγίνει wav αρχείο. Θα χρησιμοποιήσουμε πάλι το sox και με την εντολη **sox –r 44100 - s –w -c 1 output.raw BandPass.wav** δημιουργούμε ένα αρχείο wav (το output.raw με τον header που είχαμε αφαιρέσει) με ονομασία BandPass.wav . Το wav αρχείο αποθηκεύεται με αυτην την ονομασία και είναι το αρχείο wav στο οποίο έχει εφαρμοσθεί το φίλτρο που έχουμε επιλέξει.

```
File  Edit  View  Terminal  Tabs  Help
mrbubbles@ubuntu:~$ cd /home/mrbubbles/Desktop
mrbubbles@ubuntu:~/Desktop$ cd linux
mrbubbles@ubuntu:~/Desktop/linux$ ls
BandPass01.c  LowPass01test.c  pallid.wav
mrbubbles@ubuntu:~/Desktop/linux$ kwrite BandPass01.c&
[1] 6224
mrbubbles@ubuntu:~/Desktop/linux$ gcc BandPass01.c -lm -o BandPass
BandPass01.c: In function ‘main’:
BandPass01.c:46: warning: incompatible implicit declaration of built-in function
 ‘lrintf’
mrbubbles@ubuntu:~/Desktop/linux$ ls
BandPass  BandPass01.c  LowPass01test.c  pallid.wav
mrbubbles@ubuntu:~/Desktop/linux$ sox pallid.wav input.raw
mrbubbles@ubuntu:~/Desktop/linux$ ls
BandPass  BandPass01.c  input.raw  LowPass01test.c  pallid.wav
mrbubbles@ubuntu:~/Desktop/linux$ touch output.raw
mrbubbles@ubuntu:~/Desktop/linux$ ls -la
total 29340
drwx------ 2 mrbubbles mrbubbles     4096 2009-09-07 14:34 .
drwxr-xr-x 3 mrbubbles mrbubbles     4096 2009-09-07 14:32 ..
-rwxr-xr-x 1 mrbubbles mrbubbles     9387 2009-09-07 14:32 BandPass
-rwx------ 1 mrbubbles mrbubbles     1780 2009-09-07 10:16 BandPass01.c
-rw-r--r-- 1 mrbubbles mrbubbles 14986008 2009-09-07 14:33 input.raw
-rwx------ 1 mrbubbles mrbubbles     1106 2009-09-07 10:00 LowPass01test.c
-rw-r--r-- 1 mrbubbles mrbubbles        0 2009-09-07 14:34 output.raw
-rwx------ 1 mrbubbles mrbubbles 14986052 2008-03-15 09:58 pallid.wav
mrbubbles@ubuntu:~/Desktop/linux$ ./BandPass
mrbubbles@ubuntu:~/Desktop/linux$ ls -la
total 43996
drwx------ 2 mrbubbles mrbubbles     4096 2009-09-07 14:34 .
drwxr-xr-x 3 mrbubbles mrbubbles     4096 2009-09-07 14:32 ..
-rwxr-xr-x 1 mrbubbles mrbubbles     9387 2009-09-07 14:32 BandPass
-rwx------ 1 mrbubbles mrbubbles     1780 2009-09-07 10:16 BandPass01.c
-rw-r--r-- 1 mrbubbles mrbubbles 14986008 2009-09-07 14:33 input.raw
-rwx------ 1 mrbubbles mrbubbles     1106 2009-09-07 10:00 LowPass01test.c
-rw-r--r-- 1 mrbubbles mrbubbles 14986008 2009-09-07 14:35 output.raw
-rwx------ 1 mrbubbles mrbubbles 14986052 2008-03-15 09:58 pallid.wav
mrbubbles@ubuntu:~/Desktop/linux$ sox -r 44100 -s -w -c 1 output.raw BandPass.wav
mrbubbles@ubuntu:~/Desktop/linux$
```

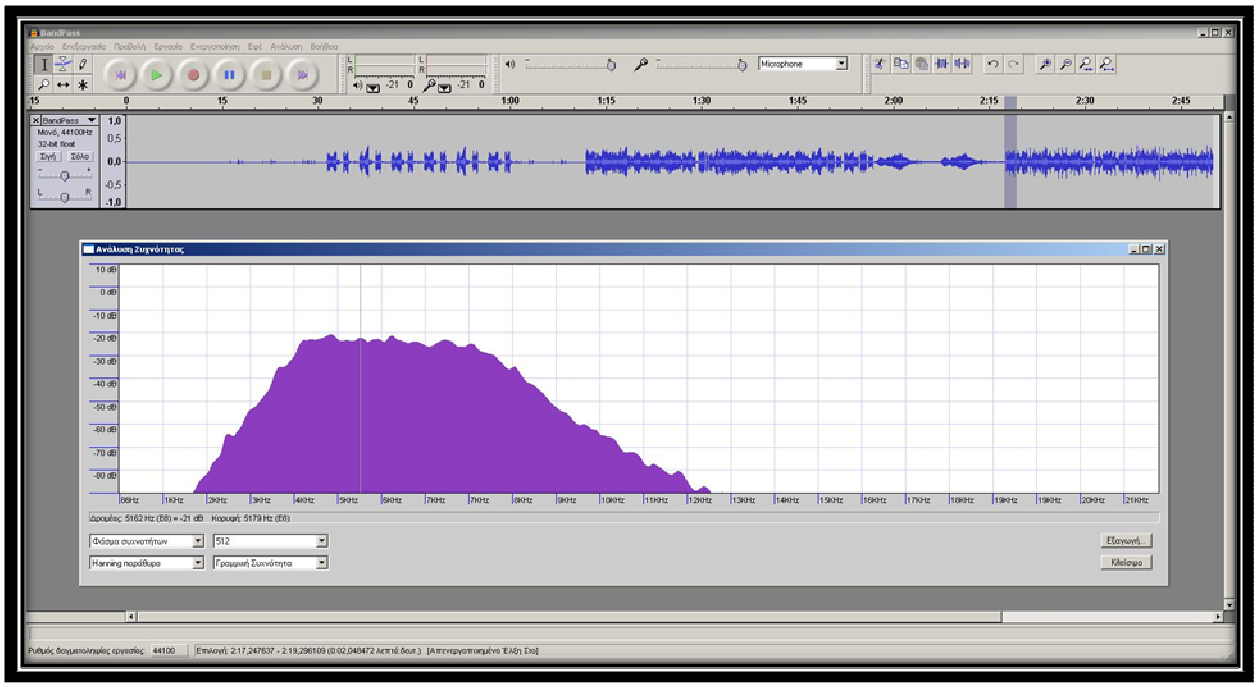
Με την βοήθεια του προγράμματος Audacity θα δούμε το συχνοτικό φάσμα του αρχείου πρίν και μετά την εφαρμογή του φίλτρου.

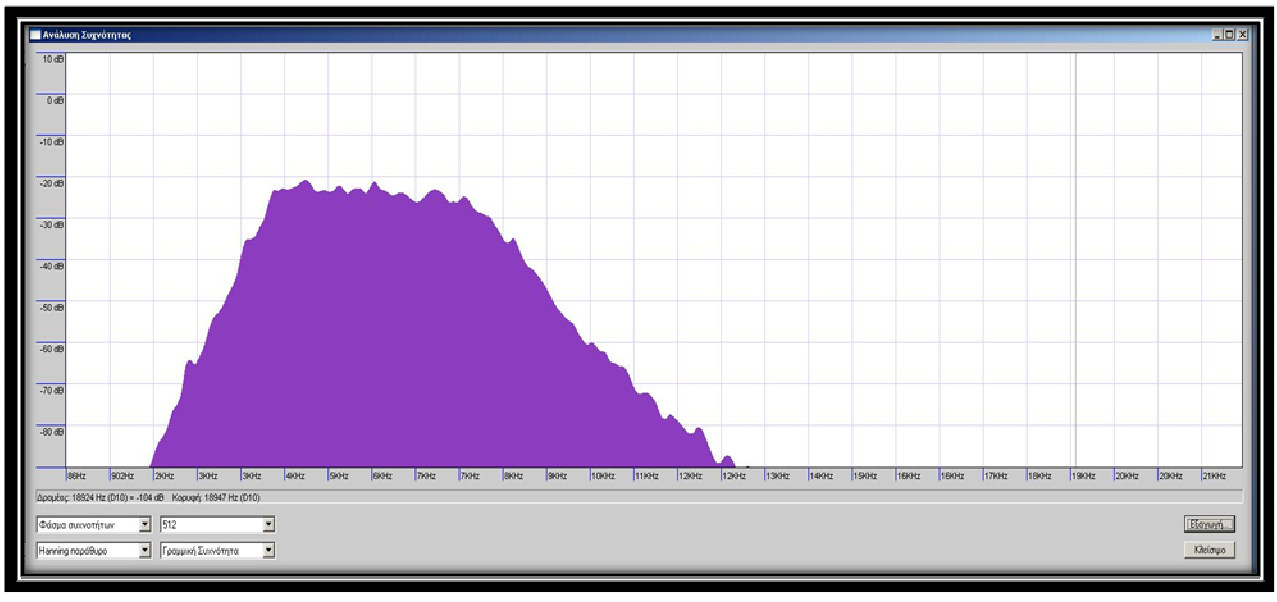
Παρακάτω βλέπουμε την συχνοτική ανάλυση σε μια περιοχή του αρχείου wav πριν την εφαρμογή του φίλτρου.





Έπειτα βλέπουμε το τελικό μας αρχείο wav μετα την εφαρμογή του φίλτρου.





6.3.8 IIR (Butterworth) Low Pass Filter

Το επόμενο φίλτρο που προγραμματίσαμε είναι ένα φίλτρο IIR φίλτρο Butterworth τύπου Lowpass.

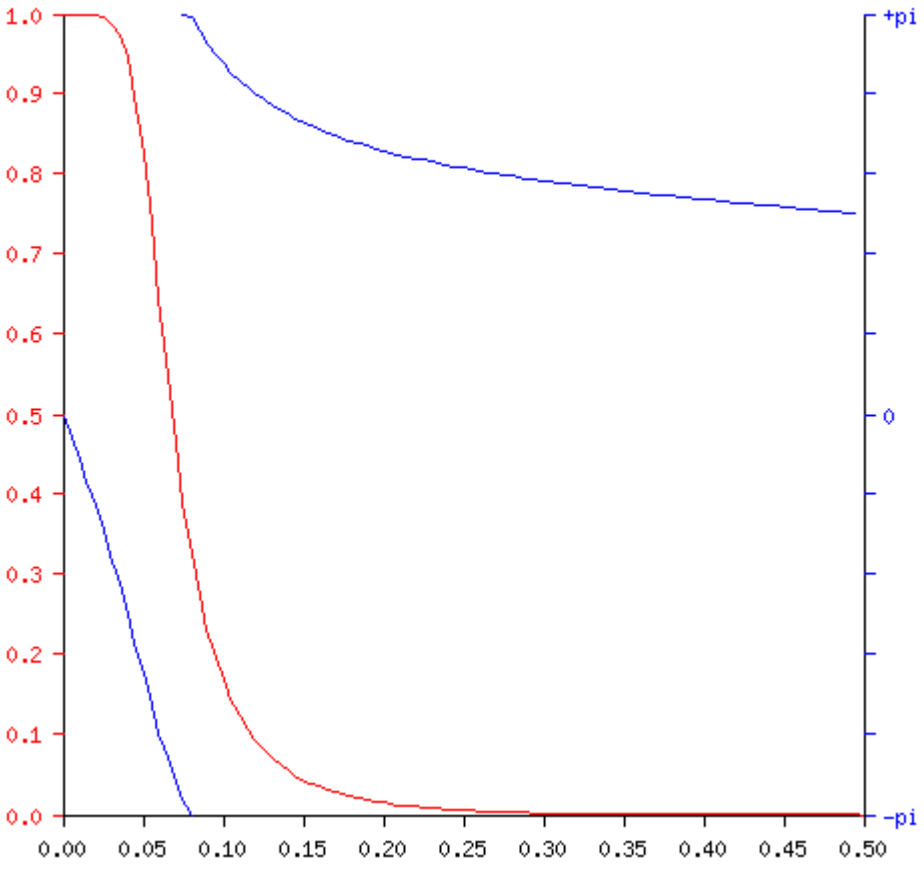
Το φίλτρο είναι **3ης τάξης** με **sample rate 44100** συχνότητα **1ης γωνίας** στα **2500Hz** και **2ης γωνίας** στα **3000 Hz .**

Παρακάτω βλέπουμε την ρουτίνα του φίλτρου με τα παραπάνω χαρακτηριστικά σε γλώσσα C.

```
#define NZEROS 3

#define NPOLES 3

#define GAIN   2.453002600e+02


static float xv[NZEROS+1], yv[NPOLES+1];


static void filterloop()

 { for (;;)

    { xv[0] = xv[1]; xv[1] = xv[2]; xv[2] = xv[3];

      xv[3] = next input value / GAIN;

      yv[0] = yv[1]; yv[1] = yv[2]; yv[2] = yv[3];

      yv[3] =   (xv[0] + xv[3]) + 3 * (xv[1] + xv[2])

                + (  0.4885691400 * yv[0]) + ( -1.8122629375 * yv[1])

                + (  2.2910807054 * yv[2]);

      next output value = yv[3];

    }

 }
```

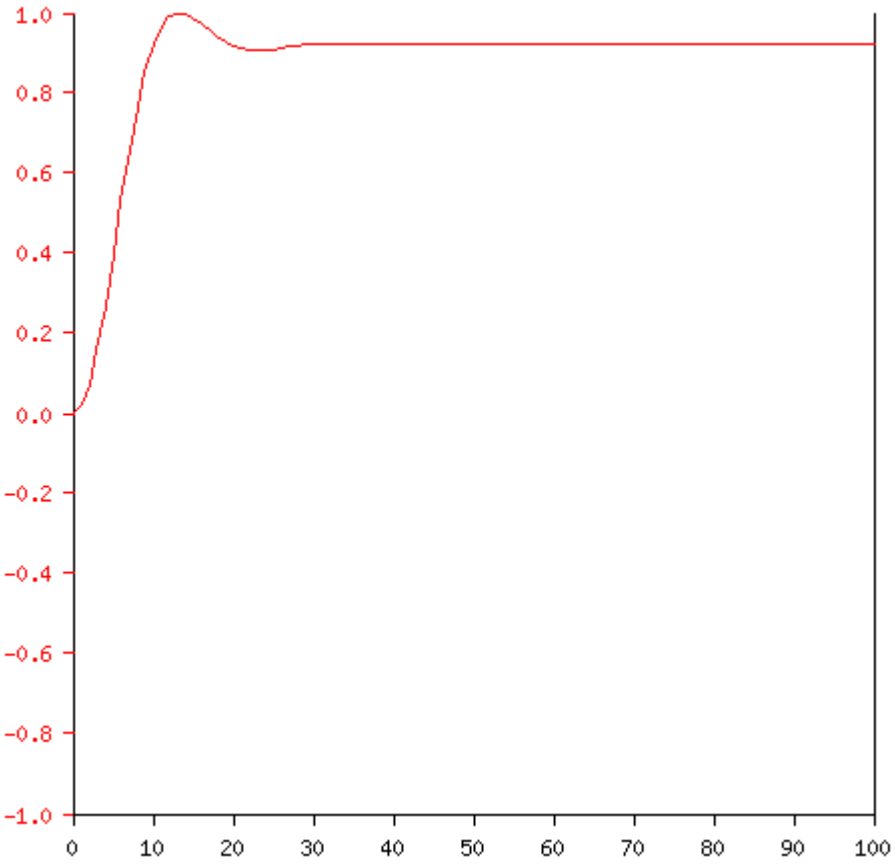
Μέγεθος και Φάση ανά Συχνότητα.

- Στον άξονα x είναι η συχνότητα σαν κλάσμα του sampling rate (πχ το 0.5 αντιπροσωπεύει την συχνότητα του Nyquist, η οποία είναι 22050 Hz)
- Στον άξονα y (κόκκινο) είναι το magnitude (γραμμικό, normalized)
- Στον άξονα y (μπλε) είναι η φάση.



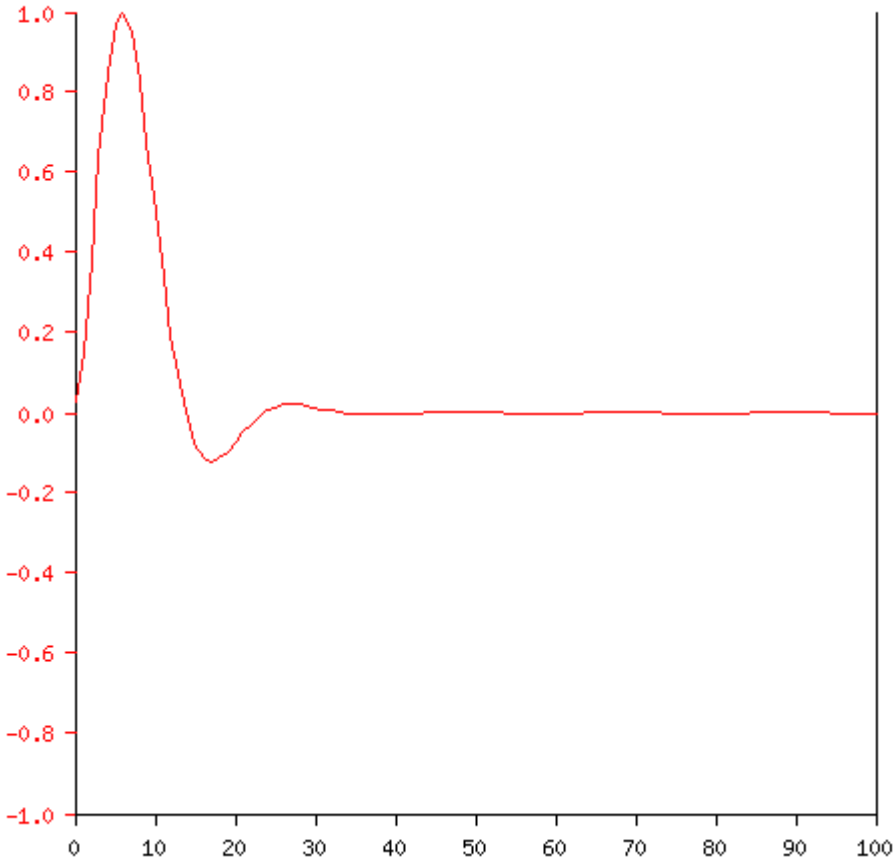
Impulse response

- Στον άξονα x βρίσκεται ο χρόνος σε samples (πχ 44100 αντιστοιχούν σε 1 second)
- Στον άξονα y (κόκκινο) το filter response (γραμμικό, normalized)



Step response

- Στόν άξονα x ο χρόνος σε Samples (πχ 44100 αντιστοιχούν σε 1 second)
- Στον άξονα y (κόκκινο) το filter response (γραμμικό, normalized)



6.3.9 Το προγραμματισμένο φίλτρο για την εφαρμογή

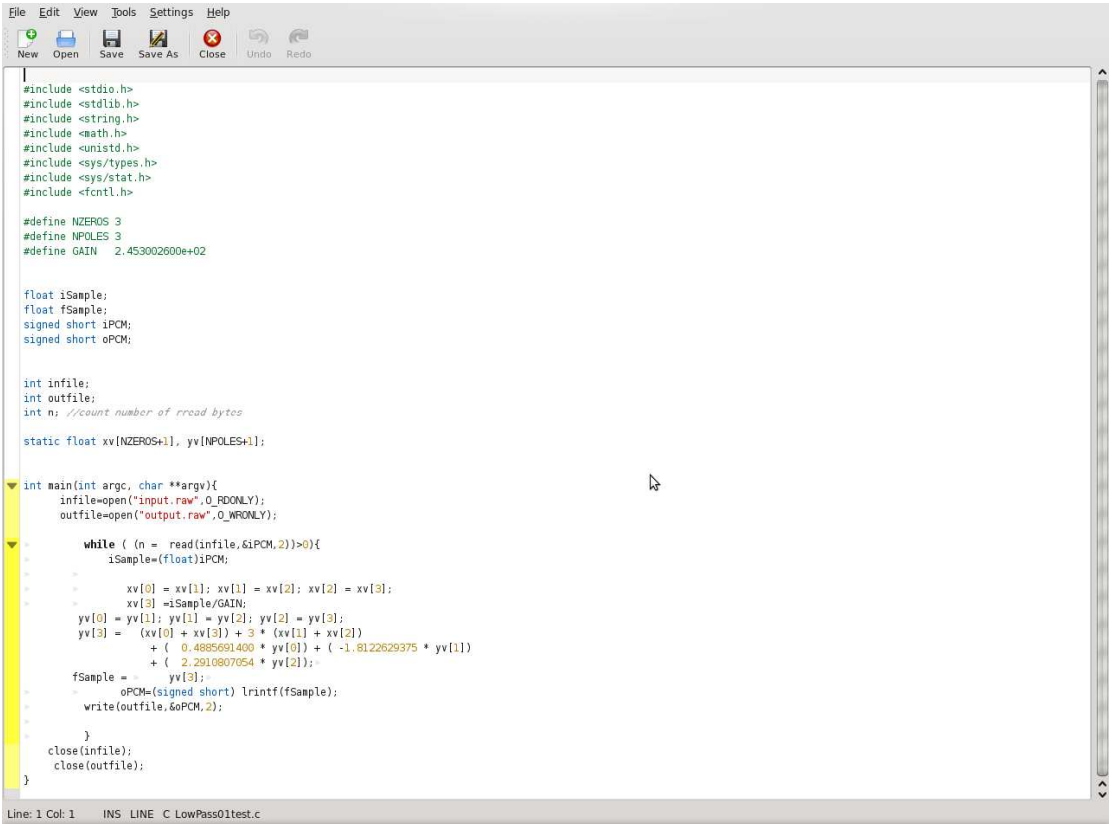
Παρακάτω δίνουμε το φίλτρο που προγραμματίσαμε ώστε να εφαρμοσθεί πάνω σε ένα αρχείο wav με βαση την δομη του φίλτρου και τα στοιχεία που δώσαμε παραπάνω. Ο κώδικας είναι σε γλώσσα προγραμματισμού C.

```
#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <math.h>

#include <unistd.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>


#define NZEROS 3

#define NPOLES 3

#define GAIN   2.453002600e+02



float iSample;

float fSample;

signed short iPCM;

signed short oPCM;



int infile;
```

```
int outfile;
int n; //count number of rread bytes

static float xv[NZEROS+1], yv[NPOLES+1];



int main(int argc, char **argv){
     infile=open("input.raw",O_RDONLY);
     outfile=open("output.raw",O_WRONLY);

              while ( (n =  read(infile,&iPCM,2))>0){
                 iSample=(float)iPCM;

                          xv[0] = xv[1]; xv[1] = xv[2]; xv[2] = xv[3];
                          xv[3] =iSample/GAIN;
        yv[0] = yv[1]; yv[1] = yv[2]; yv[2] = yv[3];
        yv[3] =   (xv[0] + xv[3]) + 3 * (xv[1] + xv[2])
                 + (  0.4885691400 * yv[0]) + ( -1.8122629375 * yv[1])
                 + (  2.2910807054 * yv[2]);
       fSample =   yv[3];
                          oPCM=(signed short) lrintf(fSample);
              write(outfile,&oPCM,2);

              }
   close(infile);
    close(outfile);
}
```

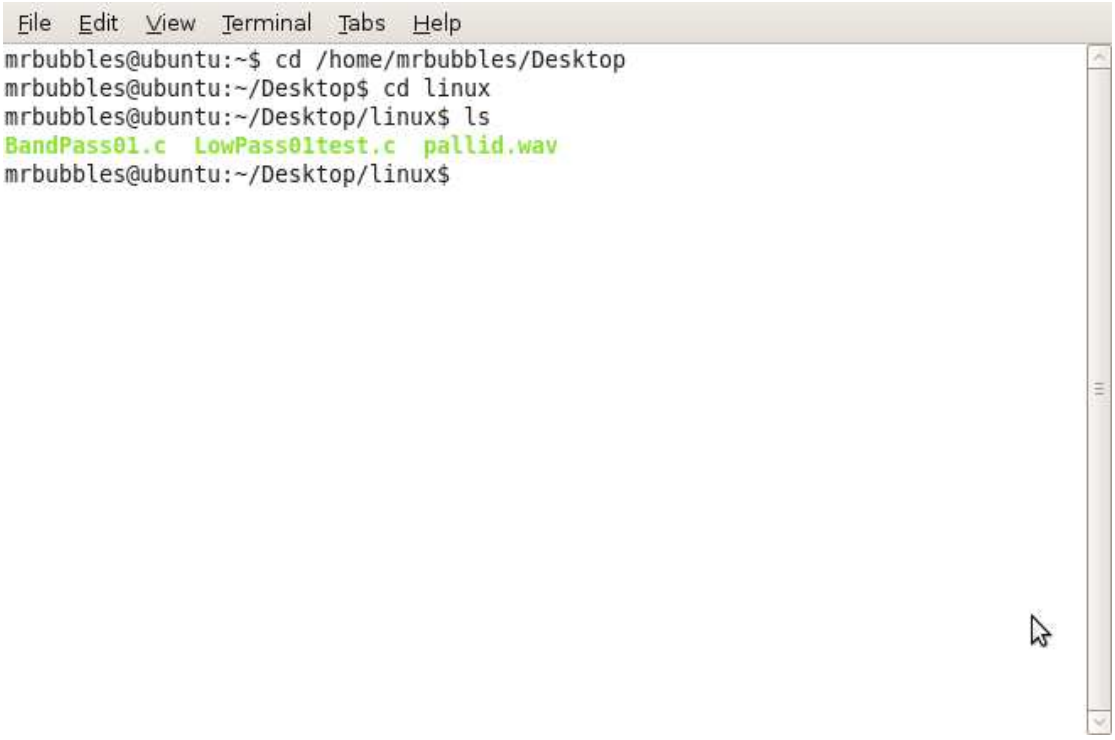
6.3.10 Τα βήματα που ακολουθήσαμε

Ο παραπάνω είναι ο κώδικας είναι το band pass φίλτρο γραμμένο σε C ο οποίος γίνεται compile απο τον gcc compiler του Linux και εφαρμόζεται πάνω σε οποιοδήποτε αρχείο wav. Παρακάτω βλέπουμε τον κώδικα στον Kwrite editor τον οποίο χρησιμοποιήσαμε για να γράψουμε τον κώδικα.

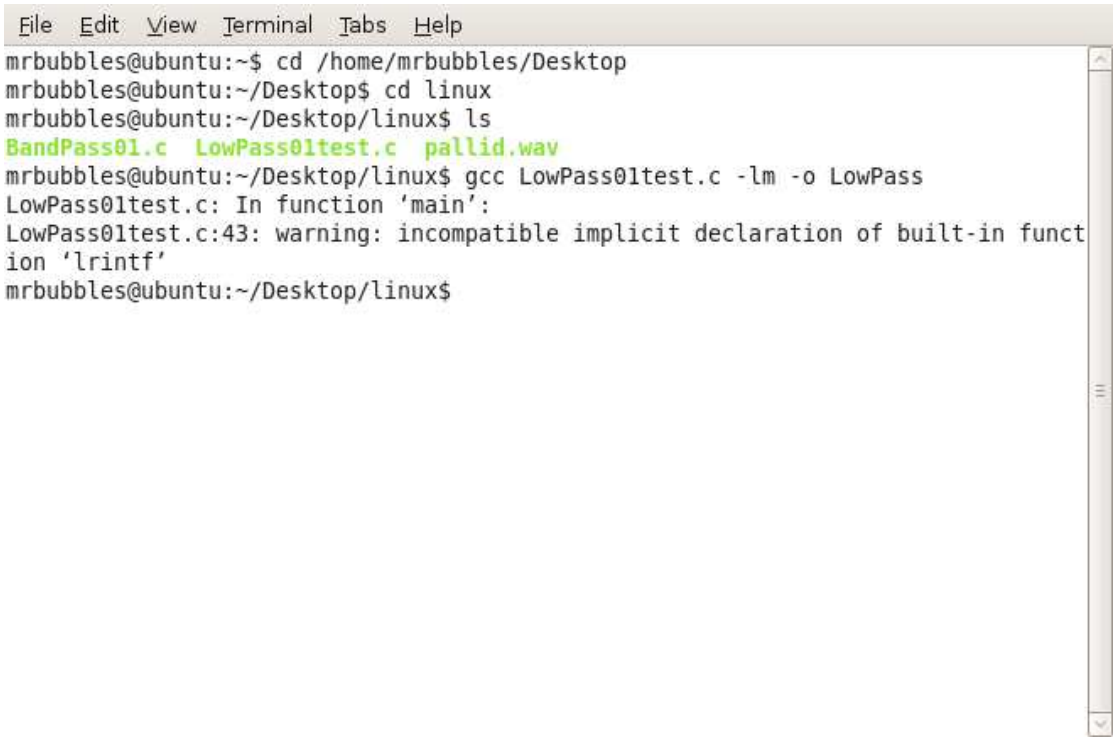


Στο παραπάνω φίλτρο χρησιμοποιήσαμε επίσης την έκδοση Linux Ubuntu. Τα βήματα τα οποία ακολουθήσαμε είναι τα παρακάτω:

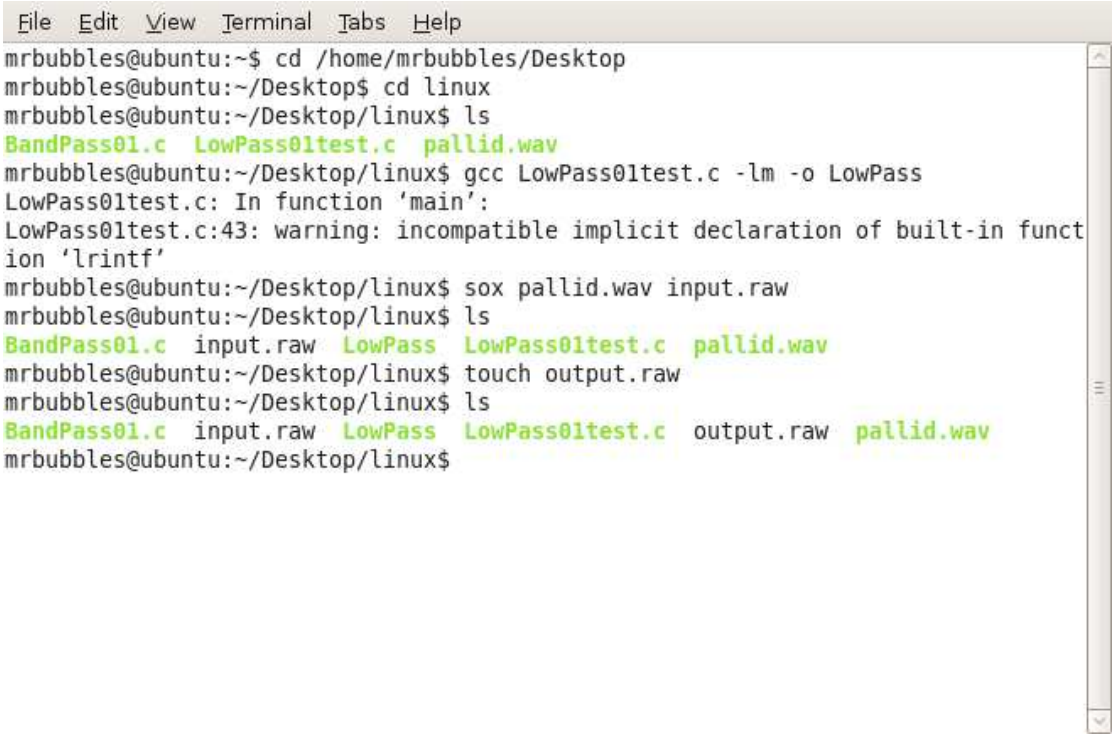
- Ανοίγουμε ένα terminal στο λειτουργικό μας και με την **εντολή cd /home/mrbubbles/Desktop/Filtra** ανοίγουμε το Directory και βλέπουμε που έχουμε αποθηκεύσει τα φίλτρα μας . Με την εντολή **ls** βλέπουμε τα περιεχόμενα του φακέλου.



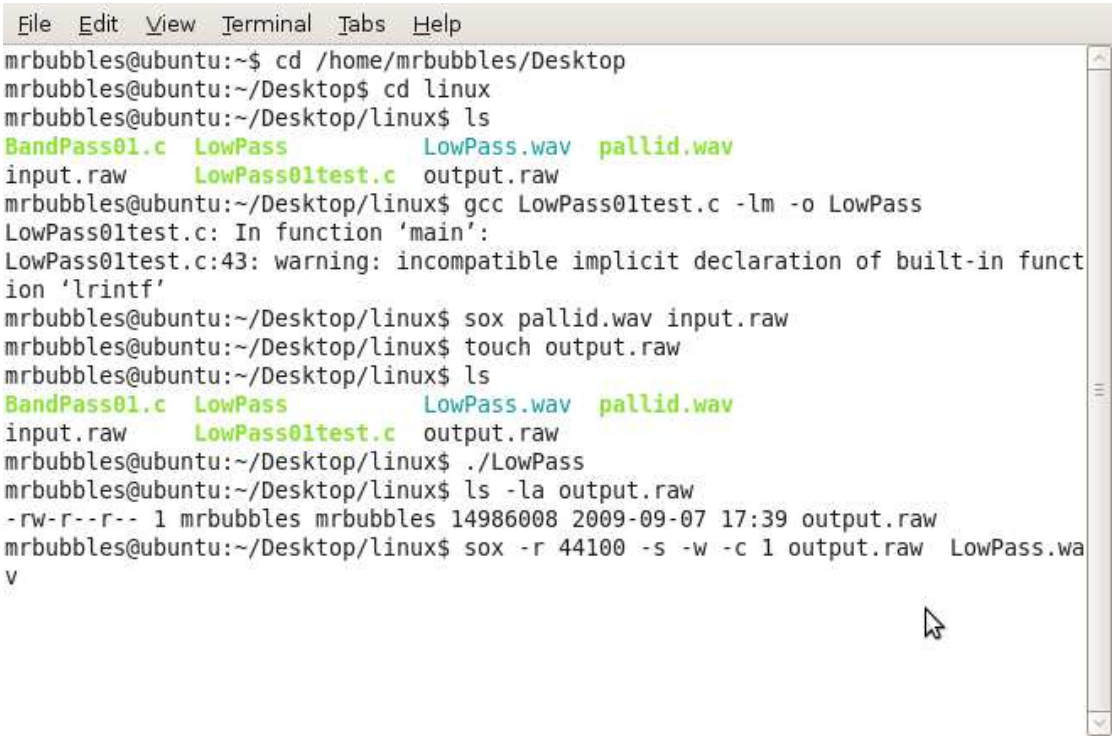
- Με την εντολή **gcc Lowpass01.c - lm –o Lowpass** κάνουμε compile το φίλτρο και δημιουργούμε ένα αρχειο exe με την ονομασία Lowpass



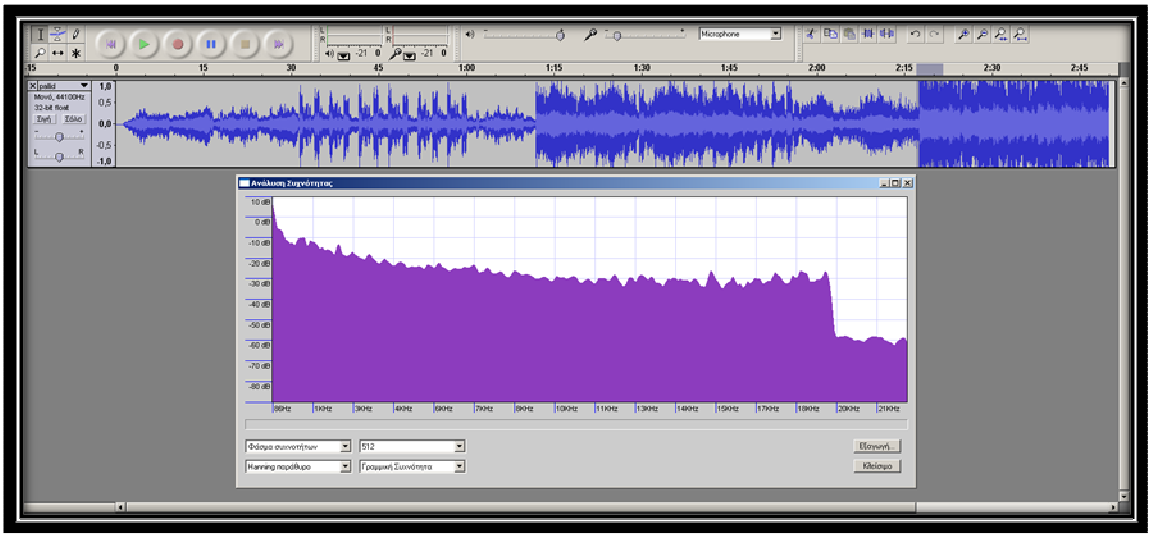
- Αφού έχουμε ήδη κάνει compile το αρχείο με τον gcc compiler θα πρέπει να αφαιρεθεί απο το wav αρχείο τον header του. Ο λόγος που το κάνουμε αυτό είναι για να διαβάσει το πρόγραμμα κατευθείαν raw data χωρίς τον data header. Για αυτόν τον λόγο θα χρησιμοποιήσουμε την βιβλιοθήκη Sox. Με την εντολή **sox pallid.wav input.raw** αφαιρείται το header απο το pallid.wav και δημιουργείται ένα αρχείο input.raw δηλαδη το pallid.wav χωρίς τον header. Επίσης με την εντολή **touch output.raw** δημιουργούμε ένα raw αρχείο οποίο θα επανατοποθετήσουμε το header μετά την εφαρμογή του φίλτρου.

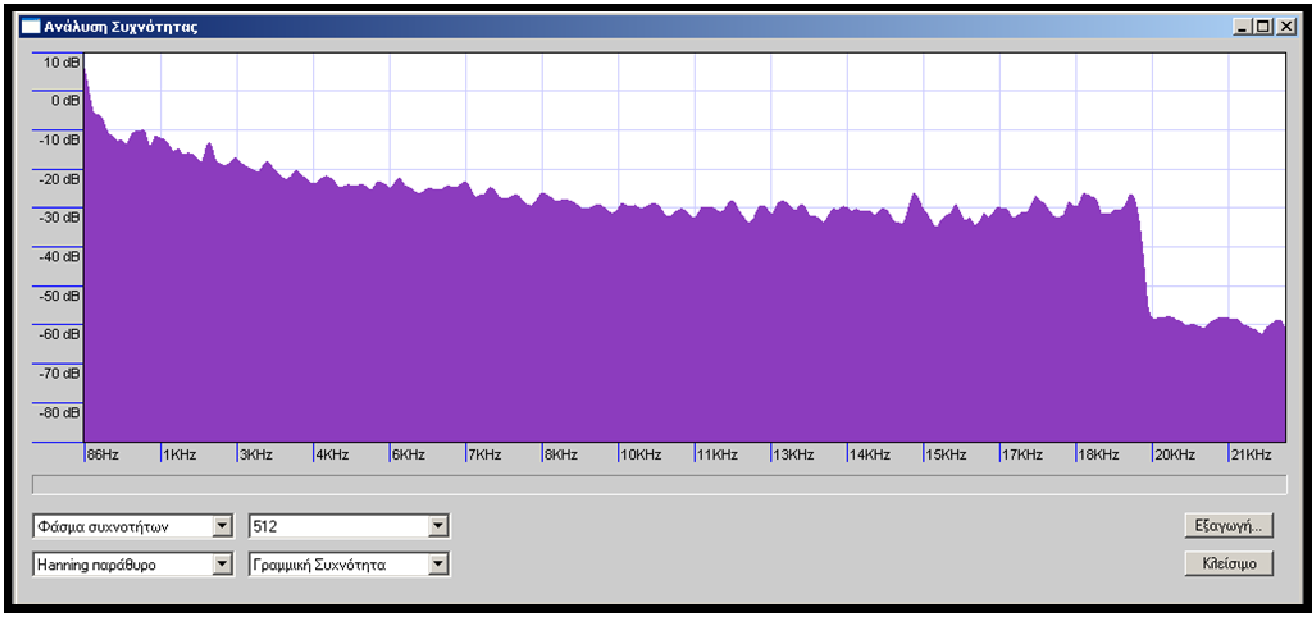


- Το output.raw που είναι ήδη φιλτραρισμένο πρέπει να ξαναγίνει wav αρχείο. Θα χρησιμοποιήσουμε πάλι το sox και με την εντολη **sox –r 44100 - s –w -c 1 output.raw LowPass.wav** δημιουργούμε ένα αρχείο wav (το output.raw με τον header που είχαμε αφαιρέσει) με ονομασία LowPass.wav . Το wav αρχείο αποθηκεύεται με αυτην την ονομασία και είναι το αρχείο wav στο οποίο έχει εφαρμοσθεί το φίλτρο που έχουμε επιλέξει.

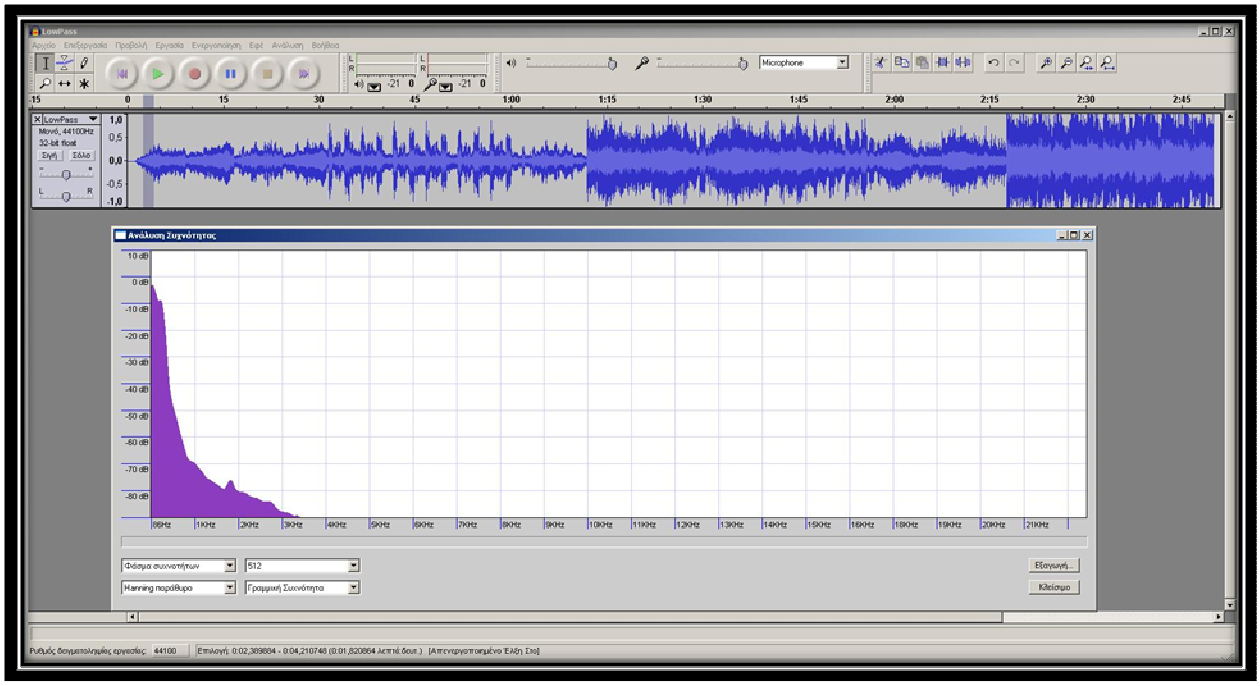


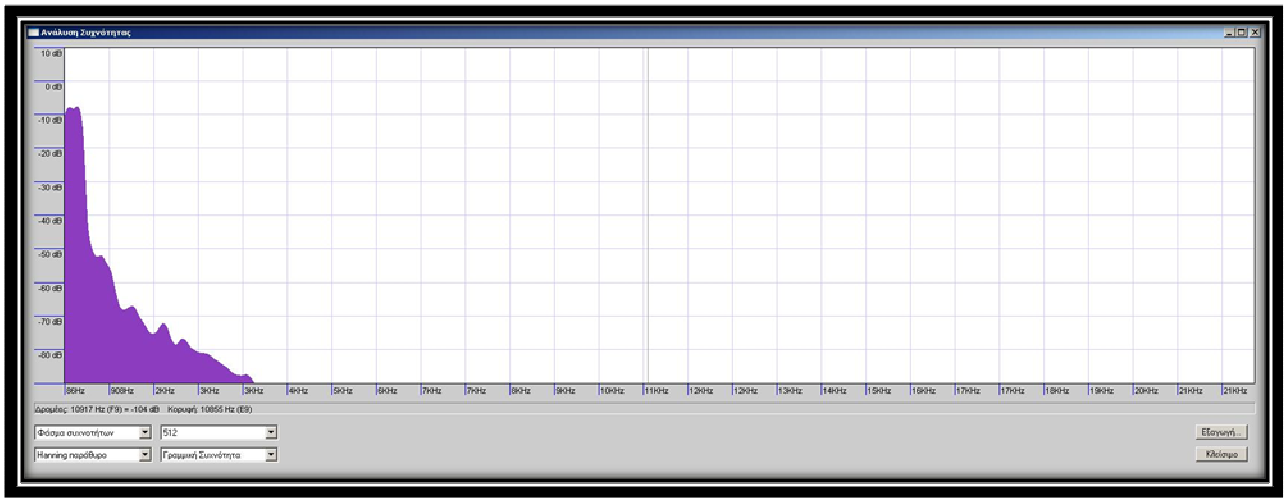
Παρακάτω βλέπουμε την συχνοτική ανάλυση σε μια περιοχή του αρχείου wav πριν την εφαρμογή του φίλτρου.





Έπειτα βλέπουμε το τελικό μας αρχείο wav μετα την εφαρμογή του φίλτρου.





ΚΕΦΑΛΑΙΟ 7

Σχεδίαση φίλτρων με την βοήθεια του Filter Design and Analysis Tool στο Matlab

7.1 Εισαγωγή στο Filter Design and Analysis Tool (FDATool)

Το Filter Design Analysis Tool (FDATool) είναι ένα δυνατό γραφικό περιβάλλον (GUI) στο Signal Processing Toolbox για σχεδιασμό και ανάλυση φίλτρων.

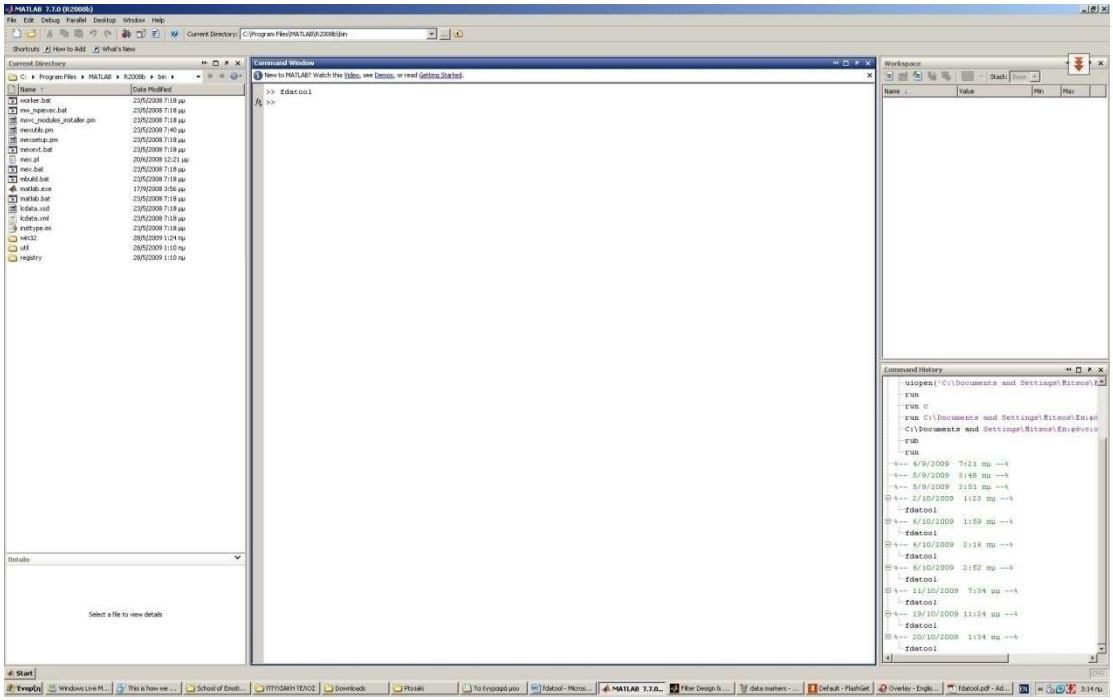
Το FDATool μας βοηθά να σχεδιάσουμε γρήγορα και αποτελεσματικά ψηφιακά φίλτρα FIR ή IIR διαλέγοντας τις παραμέτρους, εισάγοντας φίλτρα από το MATLAB ή προσθέτοντας, αφαιρώντας η μετακινώντας πόλους και μηδενικά. Το FDATool επίσης έχει εργαλεία για ανάλυση φίλτρων όπως το εύρος και την φάση.

Μπορούμε να χρησιμοποιήσουμε το FDATool σαν μια εναλλακτική μέθοδο για σχεδίαση φίλτρων.

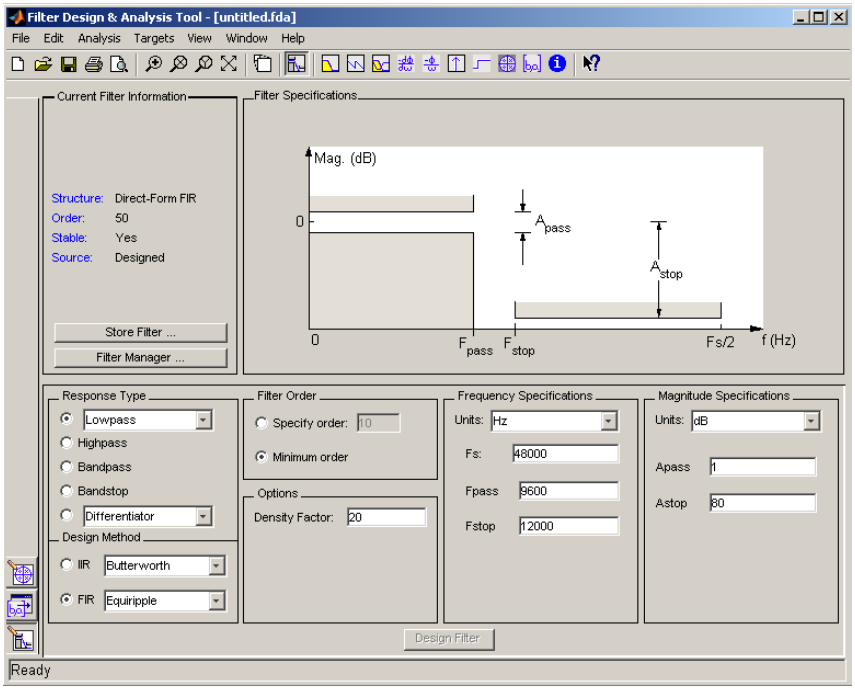
Ξεκινώντας.

Πληκτρολογούμε την εντολή fdatool στην γραμμή εντολών του MATLAB

>>fdatool



Θα δούμε το παράθυρο Tip Of The Day το οποίο μας δίνει προτάσεις για την χρησιμοποίηση του FDATool . Μετά εμφανίζεται το γραφικό περιβάλλον το οποίο έχει ένα default φίλτρο.



Το γραφικό περιβάλλον βλέπουμε ότι έχει τρείς κύριες περιοχές:

- Τις πληροφορίες του υπάρχοντος φίλτρου.
- Την περιοχή που εμφανίζεται το φίλτρο.
- Και το panel του σχεδιασμού του φίλτρου.

Το πάνω μισό του γραφικού περιβάλλοντος μας δείχνει τις πληροφορίες για τα specifications και responses για το φίλτρο μας. Στην περιοχή που εμφανίζονται οι πληροφορίες για το υπάρχων φίλτρο, πάνω αριστερά βλέπουμε τις πληροφορίες του φίλτρου, την δομή του, την τάξη του, τον αριθμό των τομέα που χρησιμοποιούνται και το αν το φίλτρο είναι σταθερό ή όχι. Επίσης μας δίνει την δυνατότητα να χρησιμοποιήσουμε το Filter Manager για να δουλέψουμε με πολλαπλά φίλτρα.

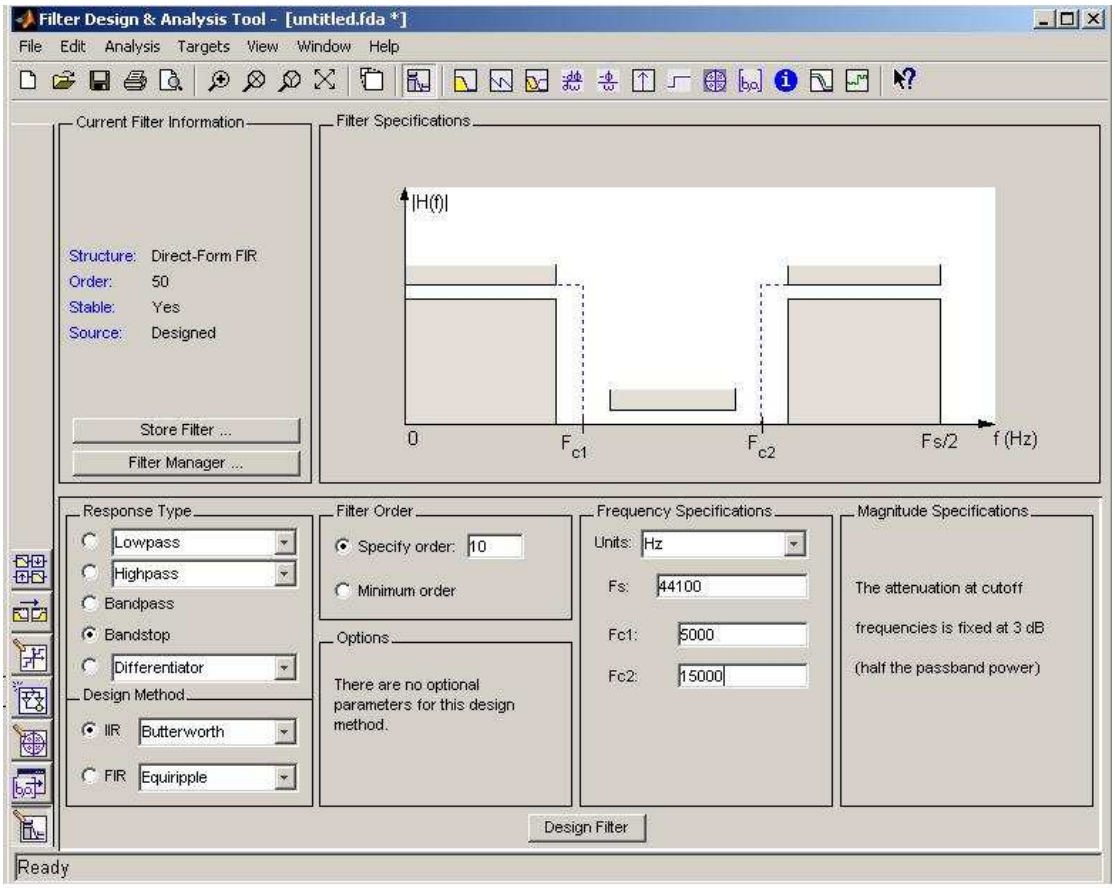
Η περιοχή της απεικόνισης του φίλτρου, πάνω δεξιά μας παρέχει πληροφορίες για διάφορα filter responses όπως το magnitude response, group delay και τα filter coefficients.

Το κάτω μισό του γραφικού περιβάλλοντος είναι το interactive περιβάλλον του FDATool. Το panel σχεδιασμού που βρίσκεται στο κάτω μισό του γραφικού περιβάλλοντος είναι εκεί που ορίζουμε τα specification του φίλτρου μας. Ελέγχει το τι θα εμφανίζεται στα άλλα panels στις πάνω περιοχές του γραφικού περιβάλλοντος.

Το εργαλείο διαθέτει Context- Sensitive βοήθεια. Μπορούμε με δεξί κλίκ στο What's This? Κουμπί για να αποκτήσουμε πληροφορίες για τα διαφορετικά μέρη του εργαλείου.

7.1.1 Σχεδιάζοντας το Φίλτρο.

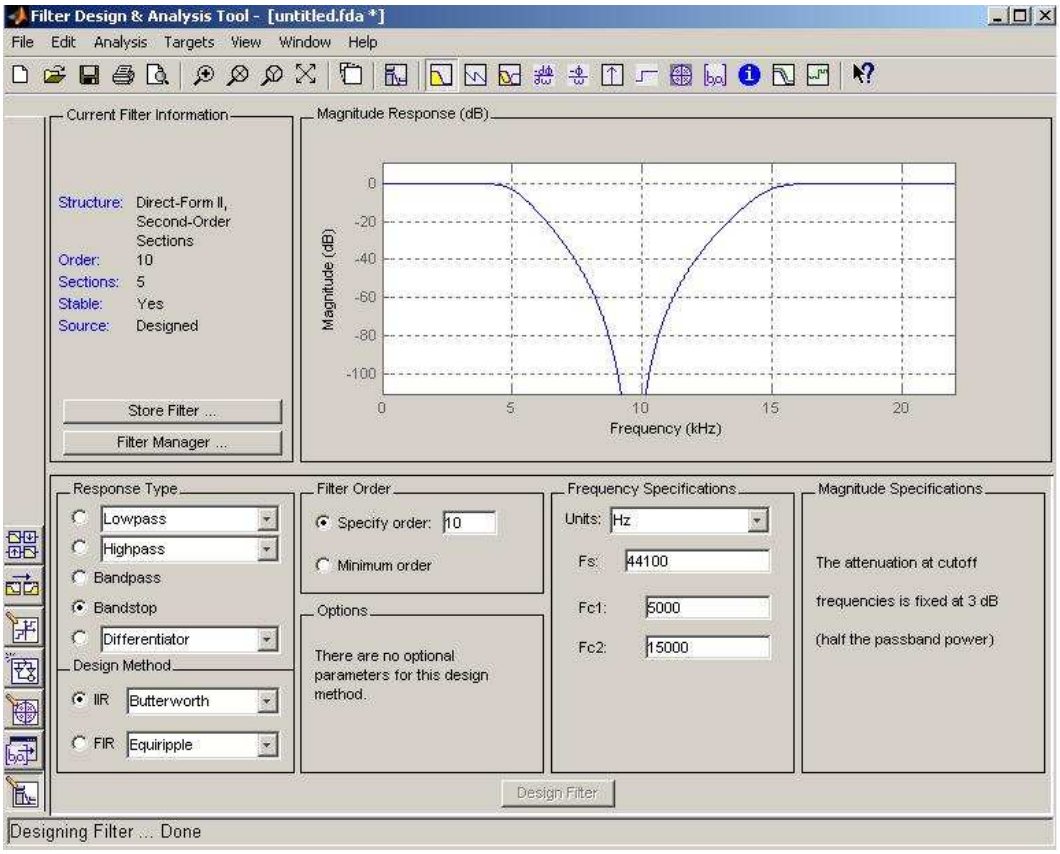
Μπορούμε να σχεδιάσουμε οποιοδήποτε φίλτρο θέλουμε, IIR ή FIR οποιοδήποτε τύπου (Butterworth, Chebyshev κλπ.). Επίσης μπορούμε να επιλέξουμε στο δεύτερο παράθυρο την τάξη του φίλτρου (filter order) και στο τρίτο παράθυρο τα Frequency Specifications και στο τέταρτο παράθυρο το Magnitude Specifications. Για να σχεδιάσουμε για παράδειγμα ένα IIR Butterworth Bandstop $10^{ης}$ τάξης με Fs 44100 και Fc1 5000 Hz και Fc2 15000Hz θα ακολουθήσουμε τα παρακάτω βήματα.



1. Από το παράθυρο design method θα επιλέξουμε την μέθοδο IIR και Butterworth.
2. Στο Response Type παράθυρο θα επιλέξουμε Bandstop
3. Στο παράθυρο filter order τσεκάρουμε την επιλογή Specify Order και στο κουτάκι θα βάλουμε την τάξη του φίλτρου που θέλουμε στην συγκεκριμένη περίπτωση 10.
4. Στο παράθυρο Frequency Specifications επιλέγουμε την μονάδα που χρησιμοποιούμε να είναι Hz και στη συνέχεια στις επιλογές Fs, Fc1 και Fc2

διαλέγουμε τις συχνότητες που θέλουμε που στην προκειμένη περίπτωση είναι 44100, 5000, 15000

5. Πατάμε το κουμπί Design Filter για να δημιουργηθεί το φίλτρο.



Το magnitude response του φίλτρου εμφανίζεται στην περιοχή ανάλυσης φίλτρου με το που υπολογιστούν οι μεταβλητές.

7.1.2 Βλέποντας άλλες αναλύσεις

Μόλις σχεδιάσουμε το φίλτρο, μπορούμε να δούμε τις παρακάτω αναλύσεις του φίλτρου στο display window πατώντας σε καθένα από τα κουμπιά στο toolbar:

Σε σειρά από αριστερά προς τα δεξιά τα κουμπιά είναι :

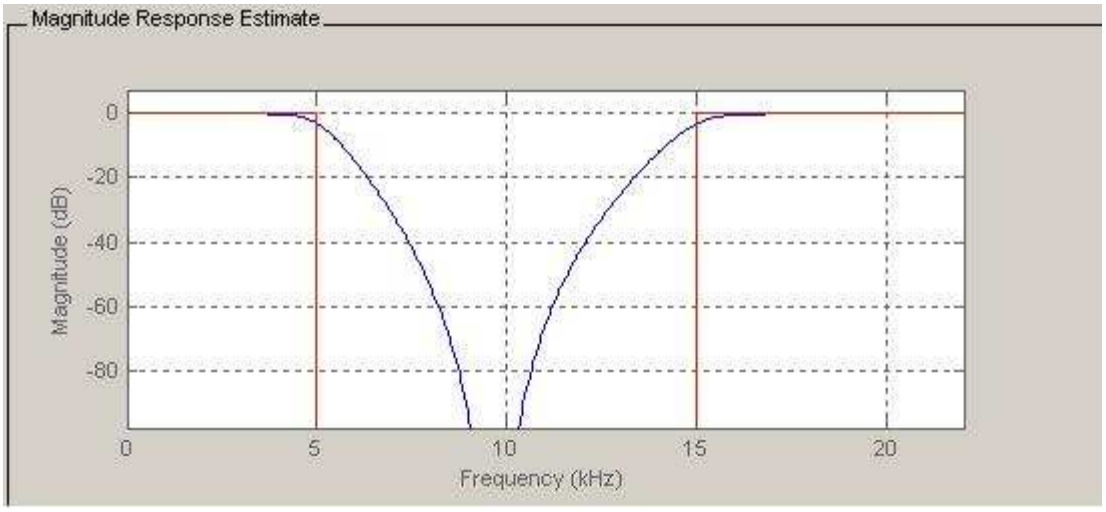
- Magnitude response
- Phase response
- Magnitude and Phase responses
- Group delay response
- Phase delay response
- Impulse response
- Step response
- Pole-zero plot

- Filter Coefficients
- Filter Information

7.1.3 Συγκρίνοντας το Design to Filter Specifications

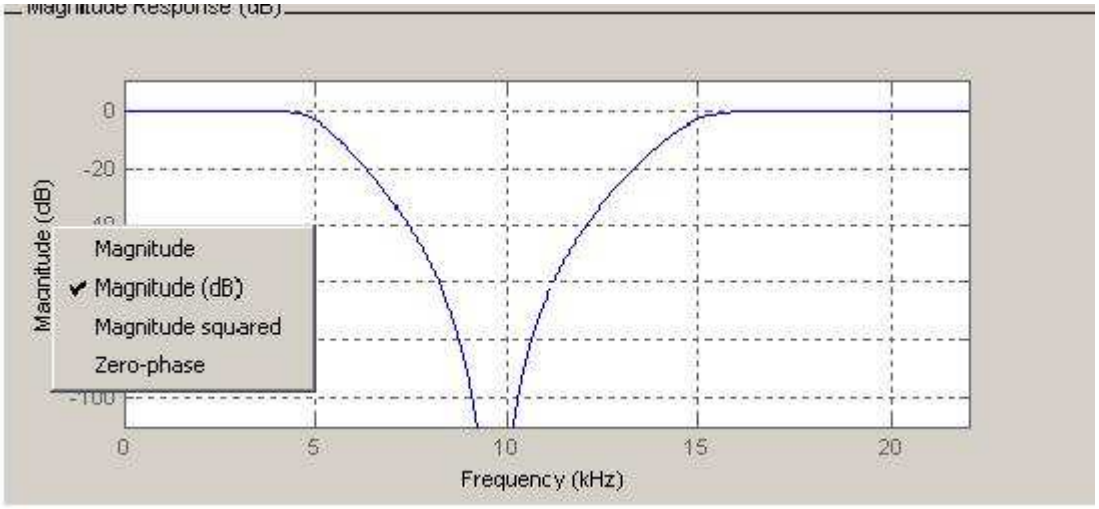
Το FDATool μας επιτρέπει να μετρήσουμε πόσο κοντά είναι το φίλτρο που έχουμε δημιουργήσει πληροί τις προδιαγραφές χρησιμοποιώντας το Specification Masks, το οποίο επικαλύπτει τις προδιαγραφές του φίλτρου στο response plot. Στο Display Region του παραθύρου όταν εμφανίζουμε το Magnitude Plot διαλέγουμε το Specification Mask από το View Menu για να επικαλύψει τις προδιαγραφές του φίλτρου στο response plot.

Το magnitude response του φίλτρου με το Specification Mask φαίνεται στην παρακάτω εικόνα:



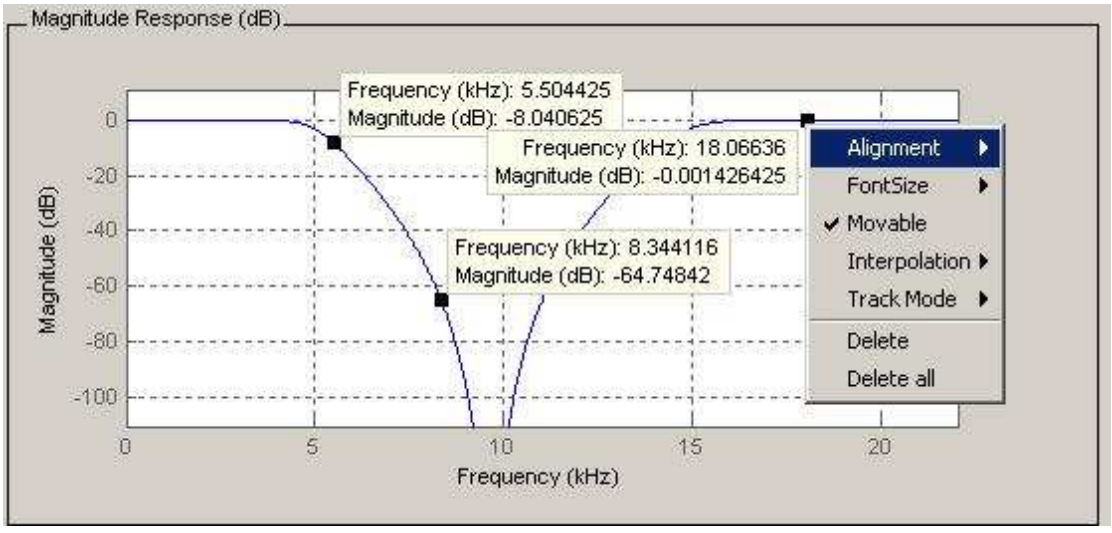
7.1.4 Αλλάζοντας τα Axes Unites.

Μπορούμε να αλλάξουμε τα τους άξονες x- ή y- κάνοντας δεξί κλίκ στο axis label και επιλέγοντας τις μονάδες που επιθυμούμε, εμφανίζοντας ένα checkmark δίπλα από την επιλογή.



7.1.5 Μαρκάροντας τα Data Points

Στο display region , μπορούμε να κλικάρουμε σε οποιοδήποτε σημείο του plot για να προσθέσουμε ένα data marker ο οποίος θα μας εμφανίσει τα δεδομένα σε εκείνο το σημείο. Με δεξί κλικ στο data marker εμφανίζεται ένα μενού στο οποίο μπορούμε να μετακινήσουμε, να διαγράψουμε η να ρυθμίσουμε την εμφάνιση των data markers.



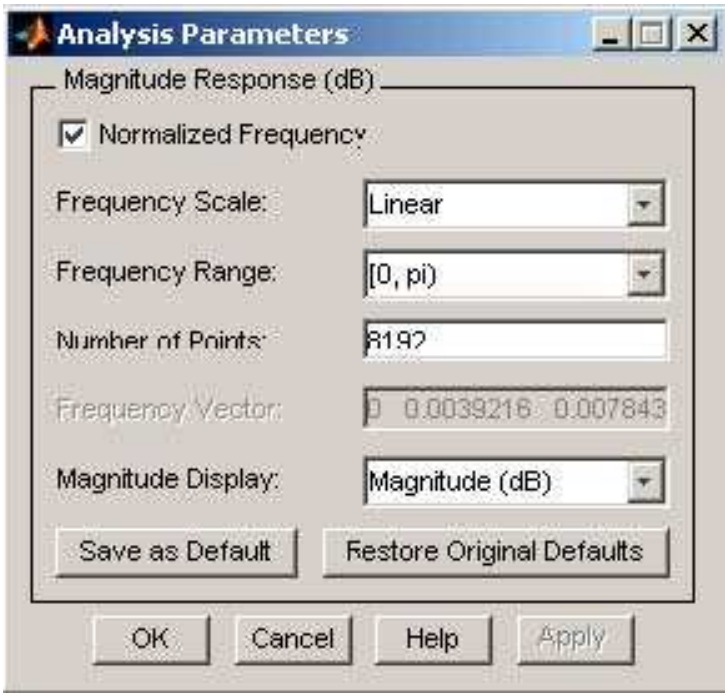
7.1.6 Χρησιμοποιώντας μια διαφορετική δομή φίλτρου.

Μπορούμε να χρησιμοποιήσουμε το Convert Structure dialog από το Edit μενού για να αλλάξουμε την δομή του φίλτρου σε μια καινούρια δομή. Τα φίλτρα μπορούν να μετατραπούν στις παρακάτω μορφές:

- State-Space
- Direct-Form FIR
- Direct-Form FIR Transposed
- Direct-Form Symmetric FIR

7.1.7 Αλλάζοντας τις παραμέτρους ανάλυσης.

Κάνοντας δεξί κλικ στο plot και επιλέγοντας το Analysis Parameters εμφανίζουμε ένα dialog box για να αλλάξουμε συγκεκριμένες παραμέτρους της ανάλυσης. (Μπορούμε επίσης να διαλέξουμε το Analysis Parameters από το Analysis menu.)



Για να σώσουμε τις παραμέτρους σαν τα default values πατάμε το κουμπί Save as Default. Για να επιστρέψουμε στα original default του MATLAB πατάμε το κουμπί Restore Original Defaults.

7.1.8 Εξάγοντας το φίλτρο.

Μόλις έχουμε κατασταλάξει στο σχεδιασμό του φίλτρου μπορούμε να το εξάγουμε στις παρακάτω διαδρομές:

- MATLAB workspace
- MAT-file
- Text-file

Επιλέγουμε Export από το File menu

Αν κάνουμε export σε MATLAB workspace, μπορούμε να κάνουμε export σαν συντελεστές ή σαν object απο το pulldown menu.

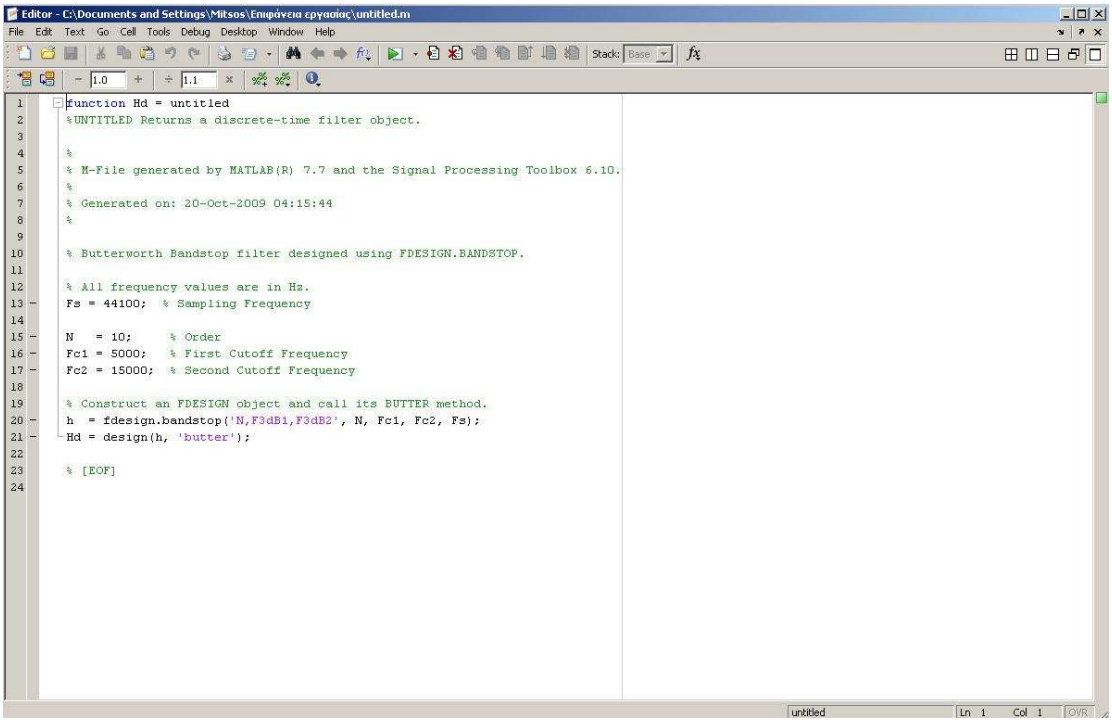
Αν θέλουμε να κάνουμε export σαν object τα properties του object έχουν τον έλεγχο τις εμφάνισης και τις συμπεριφοράς. Μπορούμε να χρησιμοποιήσουμε τις εντολές GET και SET στην γραμμή εντολών του MATLAB ώστε να έχουμε πρόσβαση στις ιδιότητες του object.

7.1.9 Δημιουργώντας ένα M-File

Το FDATool μας επιτρέπει να δημιουργήσουμε ένα M-code, για να ξαναδημιουργήσουμε το φίλτρο. Αυτό μας επιτρέπει να κάνουμε embed το φίλτρο μας σε έναν υπάρχων κώδικα η να αυτοματοποιήσουμε την δημιουργία των φίλτρων μας σε ένα script.

Επιλέγοντας generate M-file από το File menu και ορίζοντας το filename στο παράθυρο Generate M-file.

Ο κώδικας που βλέπουμε παρακάτω δημιουργήθηκε με βάση το IIR φίλτρο που σχεδιάσαμε παραπάνω:



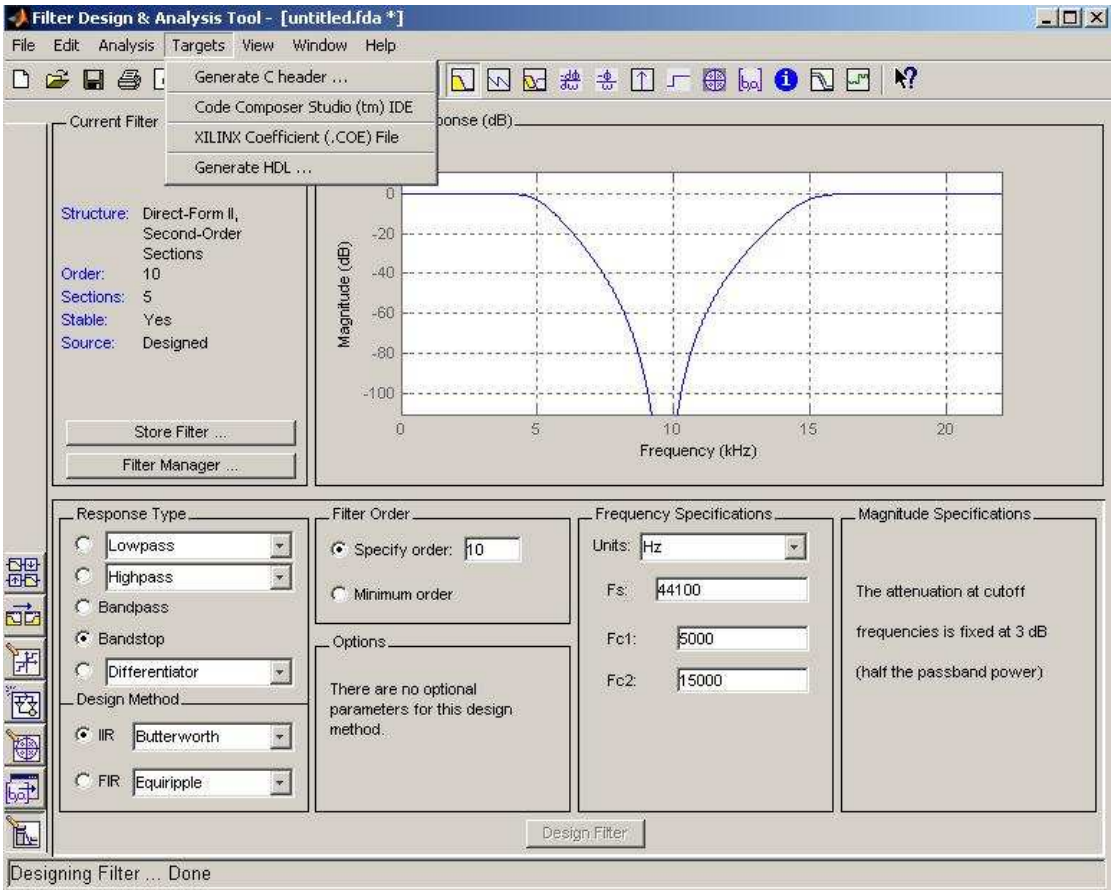
7.1.10 Κβαντοποιώντας ένα Φίλτρο

Αν έχουμε εγκατεστημένο στο μηχάνημα μας το Filter Design Toolbox το panel Set quantization parameters είναι διαθέσιμος στο sidebar:

Μπορούμε να χρησιμοποιήσουμε αυτό το panel για να κβαντοποιήσουμε και να αναλύσουμε φίλτρα double-precision. Με το Filter Design Toolbox μπορούμε να κβαντοποιήσουμε από double–precision σε single-precision. Αν έχουμε το Fixed Point Toolbox μπορούμε να κβαντοποιήσουμε φίλτρα σε fixed-point precision. Να σημειώσουμε ότι δεν μπορούμε να συνδυάσουμε floating-point και fixed-point arithmetic στο φίλτρο μας.

7.1.11 Targets

Το target menu του FDATool μας επιτρέπει να δημιουργήσουμε διάφορους τύπους κώδικα που αντιπροσωπεύουν το φίλτρο μας. Για παράδειγμα μπορούμε να δημιουργήσουμε C header files, XILINX μεταβλητές, (COE) files (με το Filter Design Toolbox) και VHDL, Verilog μαζί με test benches (με το Filter Design HDL Coder)



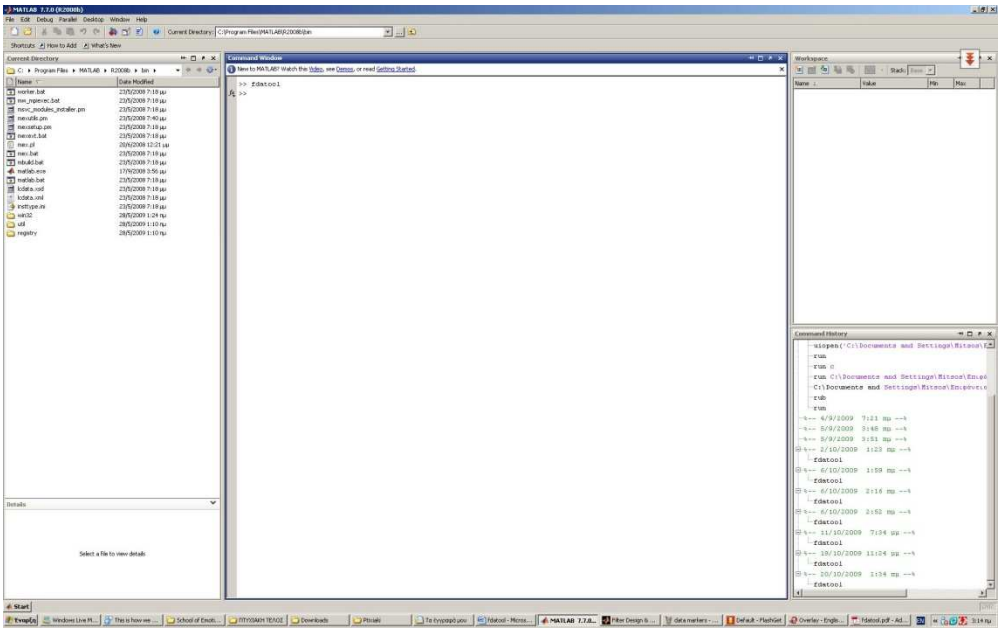
7.2 Διαδικασία προγραμματισμού φίλτρων με την βοήθεια του MATLAB.

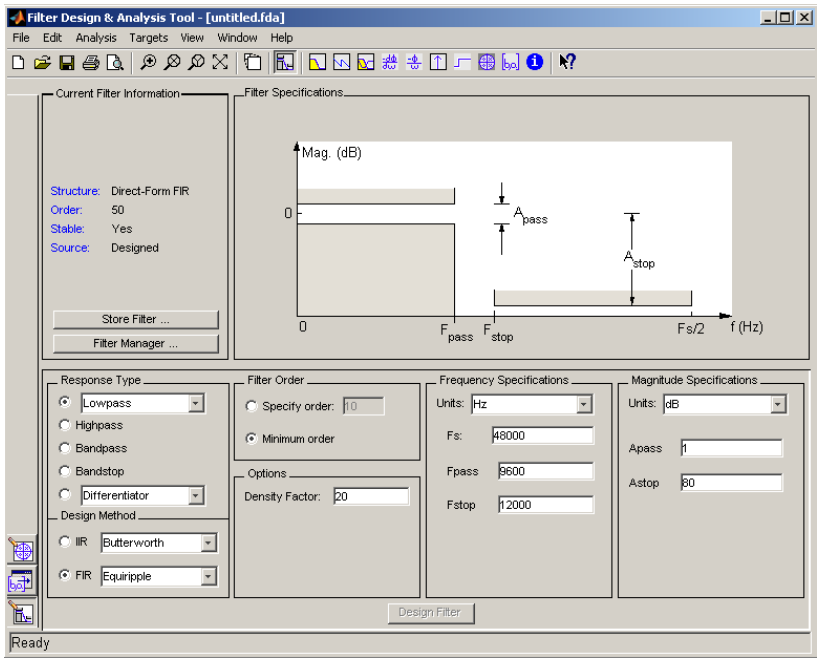
Όπως είδαμε και στο παραπάνω κεφάλαιο μπορούμε να προγραμματίσουμε στο MATLAB οποιοδήποτε φίλτρο θέλουμε με το FDATool. Με λίγα λόγια μπορούμε να σχεδιάσουμε ένα οποιοδήποτε φίλτρο IIR και με την διαδικασία που θα αναλυθεί παρακάτω να κάνουμε εφαρμογή το φίλτρο σε οποιοδήποτε wav αρχείο στο Linux.

7.2.1 Διαδικασία σχεδιασμού φίλτρου στο MATLAB και εφαρμογή σε audio αρχείο στο Linux.

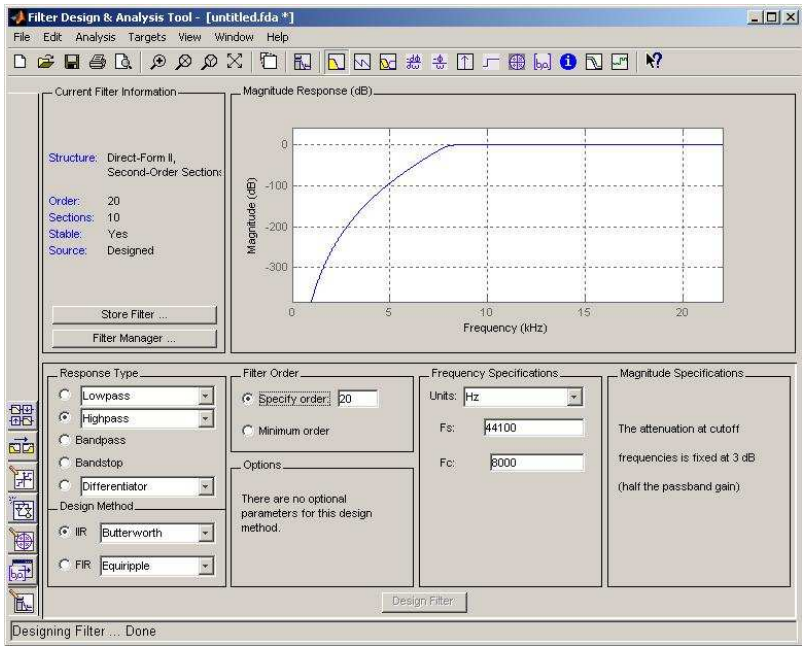
Η διαδικασία που ακολουθούμε είναι η εξής:

1. Ανοίγουμε το MATLAB και με την εντολή fdatool ανοίγουμε το γραφικό περιβάλλον σχεδίασης φίλτρου.

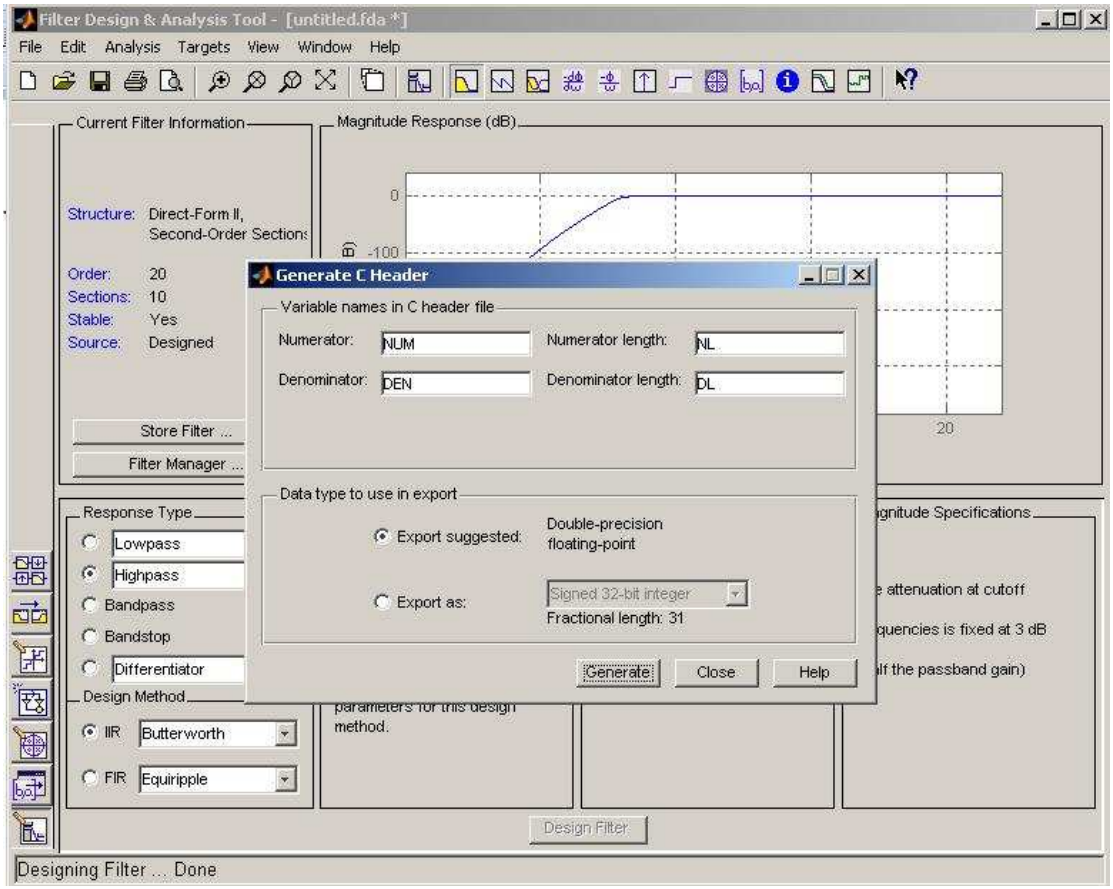




2. Για παράδειγμα επιλέγουμε να σχεδιάσουμε ένα φίλτρο IIR Butterworth High pass, 20^{ης} τάξης με Fs 44100 και Fc 8000 Hz



3. Εφόσον σχεδιάσουμε το φίλτρο θέλουμε να δημιουργήσουμε τον C header ο οποίος θα γίνει είσοδος στο πρόγραμμα που έχουμε προγραμματίσει. Επιλέγοντας Targets > Generate C header > Generate θα δημιουργηθεί ένα αρχείο με την ονομασία fdacoefs.h



Το αρχείο fdacoefs.h θα έχει την παρακάτω μορφή:

```
/*
 * Filter Coefficients (C Source) generated by the Filter Design and Analysis Tool
 *
 * Generated by MATLAB(R) 7.4 and the Signal Processing Toolbox 6.7.
 *
 * Generated on: 25-Oct-2009 21:44:28
 *
 */

/*
 * Discrete-Time IIR Filter (real)
 * -------------------------------
 * Filter Structure    : Direct-Form II, Second-Order Sections
 * Number of Sections  : 10
 * Stable              : Yes
 * Linear Phase        : No
 */

/* General type conversion for MATLAB generated C-code  */
```

```
#include "tmwtypes.h"
/*
 * Expected path to tmwtypes.h
 * C:\Program Files\MATLAB\R2007a\extern\include\tmwtypes.h
 */
#define MWSPT_NSEC 21
const int NL[MWSPT_NSEC] = { 1,3,1,3,1,3,1,3,1,3,1,3,1,3,1,3,1,3,1,3,1 };
const real64_T NUM[MWSPT_NSEC][3] = {
  {
    0.3719387173836,                0,                0
  },
  {
                  1,               -2,                1
  },
  {
    0.5370190330983,                0,                0
  },
  {
                  1,               -2,                1
  },
  {
     0.395918370066,                0,                0
  },
  {
                  1,               -2,                1
  },
  {
    0.4458233956945,                0,                0
  },
  {
                  1,               -2,                1
  },
  {
     0.316889180591,                0,                0
  },
  {
                  1,               -2,                1
  },
  {
    0.4849011311397,                0,                0
  },
  {
                  1,               -2,                1
  },
  {
    0.6788821768825,                0,                0
  },
  {
                  1,               -2,                1
  },
  {
    0.4371128255947,                0,                0
  },
  {
                  1,               -2,                1
  },
  {
    0.4192430060339,                0,                0
  },
  {
                  1,               -2,                1
  },
  {
    0.5848415811091,                0,                0
  },
  {
```

```
        1,          -2,          1
 },
 {
        1,          0,           0
 }
};
const int DL[MWSPT_NSEC] = { 1,3,1,3,1,3,1,3,1,3,1,3,1,3,1,3,1,3,1,3,1 };
const real64_T DEN[MWSPT_NSEC][3] = {
 {
        1,          0,           0
 },
 {
        1,  -0.6199802267799,    0.484022883537
 },
 {
        1,          0,           0
 },
 {
        1,  -0.4708146457561,   0.1269709549212
 },
 {
        1,          0,           0
 },
 {
        1,  -0.4436221995089,  0.06188143956714
 },
 {
        1,          0,           0
 },
 {
        1,  -0.5254777224691,    0.257815864521
 },
 {
        1,          0,           0
 },
 {
        1,  -0.7799423872374,   0.8669181701365
 },
 {
        1,          0,           0
 },
 {
        1,  -0.5665762803143,   0.3561919057045
 },
 {
        1,          0,           0
 },
 {
        1,  -0.4542475801634,  0.08731500560119
 },
 {
        1,          0,           0
 },
 {
        1,  -0.4384305807709,  0.04945446096859
 },
 {
        1,          0,           0
 },
 {
        1,  -0.4941482682574,    0.182823751917
 },
 {
        1,          0,           0
 },
 {
```

```
        1,   -0.689334001811,   0.6500323540978
 },
 {
        1,          0,          0
 }
};
```

Θα πρέπει να κάνουμε δύο αλλαγές προκειμένου να δουλέψει το αρχείο σαν είσοδο στο πρόγραμμα που έχει προγραμματιστεί ώστε να γίνει compile σε audio αρχείο.

Οι αλλαγές που κάνουμε είναι οι εξής:

1. Σβήνουμε την γραμμή **#include "tmwtypes.h"**
2. Αντικαθιστούμε το **const real64_T** με **double** και στο Numerator αλλά και στον Denumerator (NUM, DEN)

Με αυτές τις αλλαγές το αρχείο είναι έτοιμο να μπορεί να διαβαστεί από το πρόγραμμα σε c που έχει προγραμματιστεί.

7.2.2 Το πρόγραμμα που προγραμματίσαμε σε C

Ακολουθεί το πρόγραμμα σε c που έχει προγραμματιστεί με σκοπό να διαβάζει το αρχείο fdacoefs.h με τα δεδομένα του εκάστοτε σχεδιασμένου φίλτρου και να γίνεται σε compile σε Linux εφαρμόζοντας το φίλτρο σε audio αρχεία.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include "fdacoefs.h"
#define FILTER_SECTIONS (MWSPT_NSEC-1)/2

double iSample;
double fSample;
double *iSampleArray;
double *fSampleArray;
double *start_iSampleArray;
double *start_fSampleArray;
signed short iPCM;
signed short oPCM;
signed short *iPCMArray;
signed short *oPCMArray;
signed short *start_iPCMArray;
```

```
signed short *start_oPCMArray;


int infile;
int outfile;
int n;                                          //count number of rread bytes
int i, j, k;
int Number_of_sections;

double current_section_nominator_coef[3];
double current_section_denominator_coef[3];
double current_section_gain;
double section_nominator_coef[FILTER_SECTIONS][3];
double section_denominator_coef[FILTER_SECTIONS][3];
double section_gain[FILTER_SECTIONS];
double section_Delays[FILTER_SECTIONS][2];
double DLY[FILTER_SECTIONS][2];



long int number_of_samples_in_file;




///filter specific variables

double out, w, floatsample;







char infilename[1024];
char outfilename[1024];


int
main (int argc, char **argv)
{
  struct stat st;
  long int fsize;
//  outfile = open ("output.raw", O_WRONLY);

  Number_of_sections = (MWSPT_NSEC - 1) / 2;
  printf ("the number of sections are %d\n", Number_of_sections);

  for (i = 0; i < Number_of_sections; ++i)
    {
      section_gain[i] = NUM[(i * 2)][0];
      section_nominator_coef[i][0] = NUM[(i * 2) + 1][0];
      section_nominator_coef[i][1] = NUM[(i * 2) + 1][1];
      section_nominator_coef[i][2] = NUM[(i * 2) + 1][2];
      section_denominator_coef[i][0] = DEN[(i * 2) + 1][0];
      section_denominator_coef[i][1] = DEN[(i * 2) + 1][1];
      section_denominator_coef[i][2] = DEN[(i * 2) + 1][2];
      printf
              ("section %d  gain nom(3) den (3 is %lf |%lf %lf %lf |%lf %lf %lf \n",
               i, section_gain[i], section_nominator_coef[i][0],
               section_nominator_coef[i][1], section_nominator_coef[i][2],
               section_denominator_coef[i][0], section_denominator_coef[i][1],
               section_denominator_coef[i][2]);

    }
```

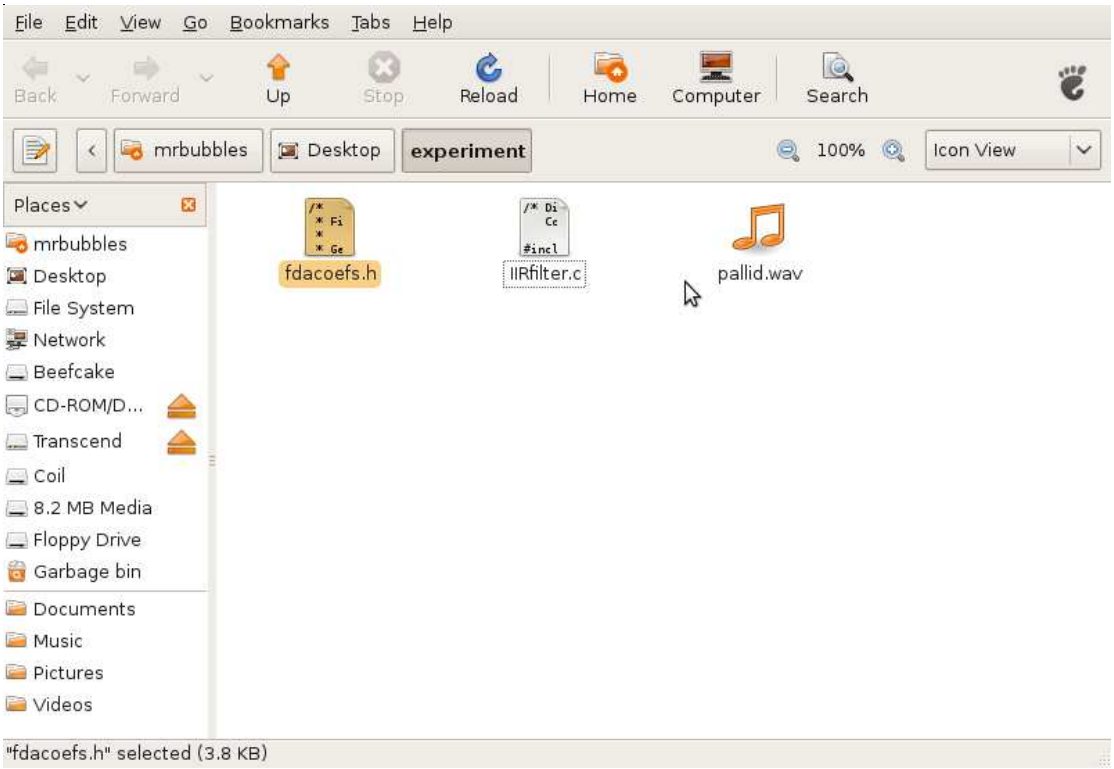
```
//OK NOW WE HAVE LOADED THE ARRAYS OF STAGES
//AND WE ARE GOING TO READ THE FILE INTO MEMORY


  stat ("input.raw", &st);
  fsize = st.st_size;
  printf ("filesize is %d\n", fsize);
  number_of_samples_in_file = fsize / 2;
  printf ("number of samples on this MONO wav is : %d\n",
                number_of_samples_in_file);

  //allocate memory to load the samples
  printf ("allocating memory for input data....");
  iPCMArray =
    (signed short *) malloc (number_of_samples_in_file *
                                          sizeof (signed short));
  printf ("finished!!\n");

  //now we read the file into memory
  printf ("reading PCM data from File input.raw....");

  sprintf (infilename, "%s", argv[1]);
  infile = open (infilename, O_RDONLY);
  i = 0;
  while ((n = read (infile, &iPCMArray[i], 2)) > 0)
  {
     ++i;
    }
  close (infile);
  printf ("Read %d PCM data into memory\n", i);


//alocating memory for input sample (converted to double) and output (double)
  printf ("allocating memory for input/ouput Double Data....");
  iSampleArray =
    (double *) malloc (number_of_samples_in_file * sizeof (double));
  fSampleArray =
    (double *) malloc (number_of_samples_in_file * sizeof (double));
  printf ("Done!!\n");


  //convert pcm data to float
  printf ("Converting PCM data to Double Precision...");
  for (i = 0; i < number_of_samples_in_file; ++i)
    {
      iSampleArray[i] = (double) iPCMArray[i];
//      printf("for sample nr %d the values are %d  %f\n",i, iPCMArray[i],iSampleArray[i]);
    }
  printf ("Done!!\n");

// and now lets start filtering
  printf("Start Filtering .....");
  for (j = 0; j < number_of_samples_in_file; ++j)
    {
      floatsample = iSampleArray[j];
      //printf("input sample %d is %f\n",j,floatsample);
      for (i = 0; i < Number_of_sections; ++i)
                {
                  w = floatsample - section_denominator_coef[i][1] * DLY[i][0] - section_denominator_coef[i][2] *
DLY[i][1];
                  out = w + section_nominator_coef[i][1] * DLY[i][0] + section_nominator_coef[i][2] * DLY[i][1];
                  DLY[i][1] = DLY[i][0];
                  DLY[i][0] = w;
                  out*=section_gain[i];
                  floatsample = out;
```

```
            }
    fSampleArray[j] = out;
    //printf("output float sample nr %d is %f \n",j,fSampleArray[j]);
   }

 printf("DONE!!\n");




 //go to the start of array and write to output file to check if we have
 //read the file correctly
 sprintf (outfilename, "%s", argv[2]);
 printf("Writing Filtered Data to %s .....",outfilename);
 mode_t mode = S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH;
 outfile = open (outfilename, O_WRONLY | O_CREAT | O_TRUNC, mode);

 for (i = 0; i < number_of_samples_in_file; ++i)
  {
   //write (outfile, &iPCMArray[i], 2);
   oPCM = (signed short) lrintf (fSampleArray[i]);
   write (outfile, &oPCM, 2);
  }
 free (iPCMArray);
 free (iSampleArray);
 free (fSampleArray);
 close (outfile);
 printf("Done !! Exiting Program.\n");




//  while ((n = read (infile, &iPCM, 2)) > 0)
 //  {
 //we cast the input pcm sample to a double value
 //  iSample = (double) iPCM;
 //  for (i = 0; i < Number_of_sections; i = i + 2)
//     {
//       current_section_gain = NUM[(i * 2)][0];
//       printf ("Section is %d and the gain is %lf \n", i,
//               current_section_gain);
//     }



 // oPCM = (signed short) lrintf (fSample);


 //  write (outfile, &oPCM, 2);
 // }
 close (infile);
 //close (outfile);
}
```

7.2.3 Διαδικασία εφαρμογής φίλτρου σε audio αρχεία.

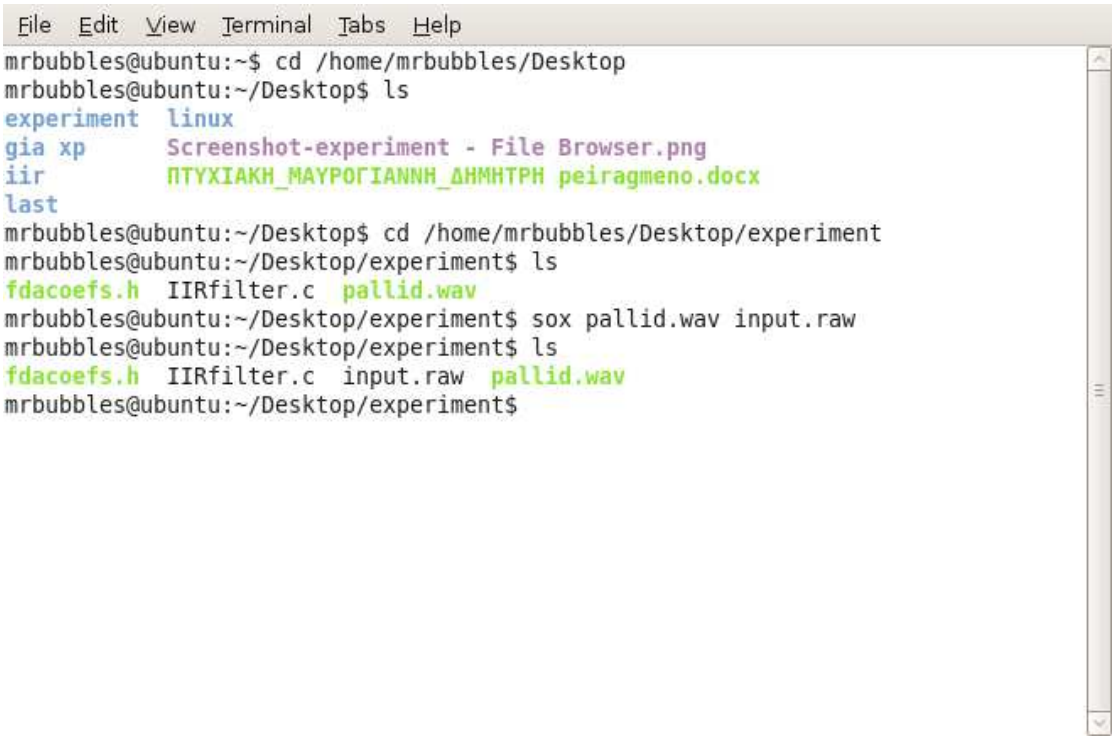
Έχουμε λοιπόν δυο αρχεία : Το fdacoefs.h ο C header από το σχεδιασμένο μας φίλτρο από το MATLAB και το IIRfilter.c το οποίο έχει τον κώδικα σε c. Η διαδικασία που θα ακολουθήσουμε στο Linux είναι η παρακάτω:

1.Στον ίδιο φάκελο έχουμε τα αρχεία fdacoefs.h , IIRfilter.c και επίσης το wav αρχείο στο οποίο θέλουμε να κάνουμε εφαρμογή το φίλτρο. Στην προκειμένη περίπτωση το αρχείο pallid.wav



2.Ανοίγουμε ένα terminal στο Linux και βρίσκουμε τον φάκελο που έχουμε αποθηκευμένα τα αρχεία μας. Με την εντολή **cd /home/mrbubbles/Desktop/experiment** βρίσκουμε το directory και με την εντολή **ls** βλέπουμε τα περιεχόμενα του φακέλου.

3. Στη συνέχεια πληκτρολογούμε την εντολή **sox pallid.wav input.raw** (οπου pallid.wav το όνομα του wav αρχείου που θέλουμε να γίνει η εφαρμογή του φίλτρου) για να αφαιρέσουμε τον header από το αρχείο wav και να έχουμε raw data. Δημιουργείτε το αρχείο input.raw.



4. Πληκτρολογούμε την εντολή **gcc IIRfilter.c –lm –o IIRfilter** έτσι ώστε να γίνει compile το αρχείο fdacoefs.h σε εκτελέσιμο αρχείο έτοιμο ώστε να γίνει εφαρμογή στο raw data του audio αρχείου. Δημιουργείτε το εκτελέσιμο αρχείο **IIRfilter**

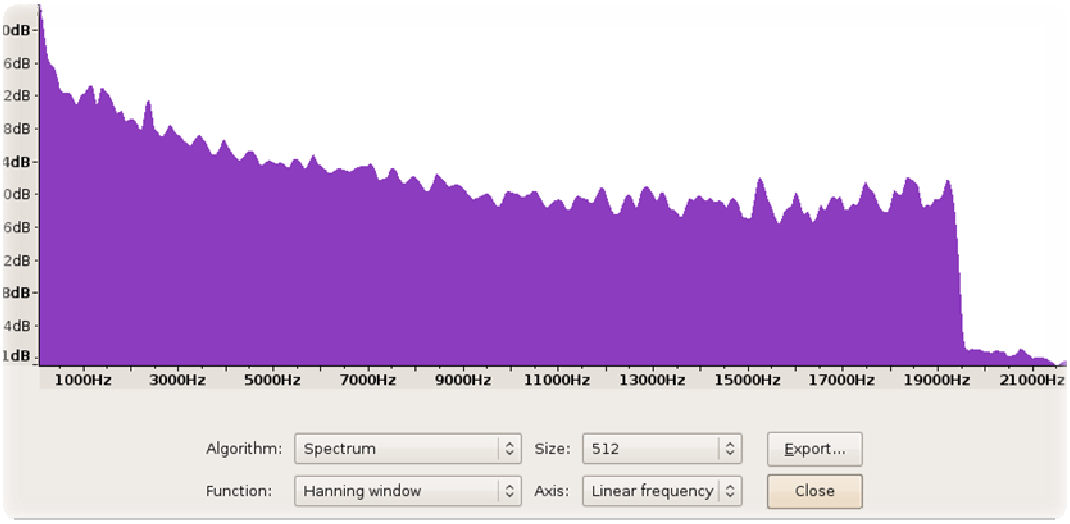
5. Πληκτρολογούμε την εντολή **./IIRfilter input.raw output.raw** όπου θα γίνει εφαρμογή του εκτελέσιμου αρχείου IIRfilter στα raw data του input.raw και θα μας δώσει το output.raw το οποίο είναι το audio αρχείο φιλτραρισμένο χωρίς τον header.

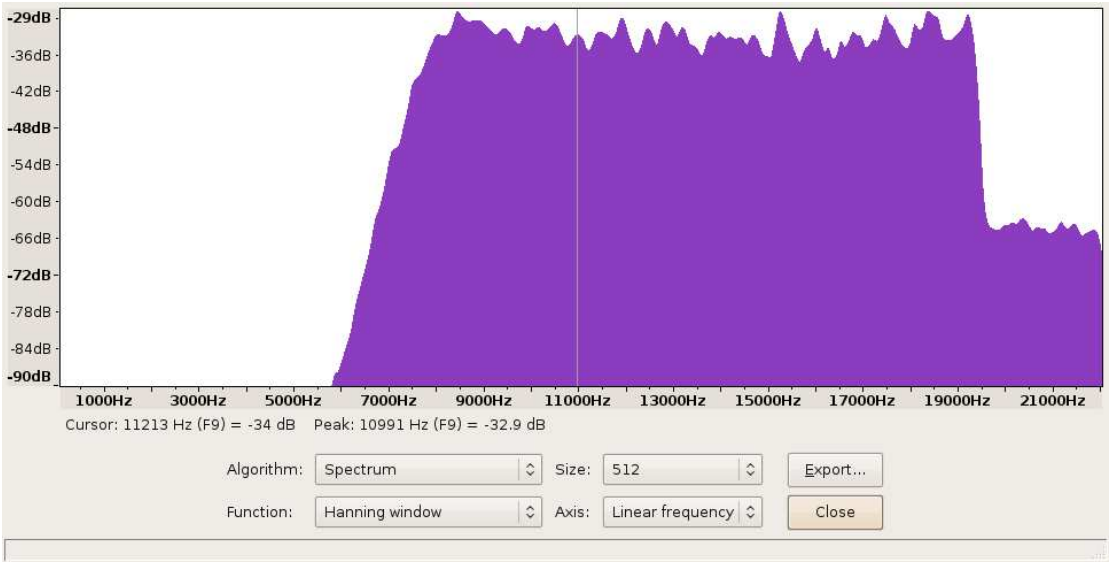
```
File  Edit  View  Terminal  Tabs  Help
 -0.443622 0.061881
section 3  gain nom(3) den (3 is 0.445823 |1.000000 -2.000000 1.000000 |1.000000
 -0.525478 0.257816
section 4  gain nom(3) den (3 is 0.316889 |1.000000 -2.000000 1.000000 |1.000000
 -0.779942 0.866918
section 5  gain nom(3) den (3 is 0.484901 |1.000000 -2.000000 1.000000 |1.000000
 -0.566576 0.356192
section 6  gain nom(3) den (3 is 0.678882 |1.000000 -2.000000 1.000000 |1.000000
 -0.454248 0.087315
section 7  gain nom(3) den (3 is 0.437113 |1.000000 -2.000000 1.000000 |1.000000
 -0.438431 0.049454
section 8  gain nom(3) den (3 is 0.419243 |1.000000 -2.000000 1.000000 |1.000000
 -0.494148 0.182824
section 9  gain nom(3) den (3 is 0.584842 |1.000000 -2.000000 1.000000 |1.000000
 -0.689334 0.650032
filesize is 14986008
number of samples on this MONO wav is : 7493004
allocating memory for input data....finished!!
reading PCM data from File input.raw....Read 7493004 PCM data into memory
allocating memory for input/ouput Double Data....Done!!
Converting PCM data to Double Precision...Done!!
Start Filtering .....DONE!!
Writing Filtered Data to output.raw .....Done !! Exiting Program.
mrbubbles@ubuntu:~/Desktop/experiment$
```

6. Εφόσον έχει γίνει η εφαρμογή του φίλτρου στο raw data θέλουμε να επαναφέρουμε τον header ώστε να έχουμε πλέον το φιλτραρισμένο αρχείο wav. Πληκτρολογούμε την **εντολή sox –r 44100 -s –w –c 1 output.raw pallidfiltered.wav** όπου το pallidfiltered.wav η ονομασία που δίνουμε εμείς στο φιλτραρισμένο μας αρχείο.

```
File  Edit  View  Terminal  Tabs  Help
section 8  gain nom(3) den (3 is 0.419243 |1.000000 -2.000000 1.000000 |1.000000
 -0.494148 0.182824
section 9  gain nom(3) den (3 is 0.584842 |1.000000 -2.000000 1.000000 |1.000000
 -0.689334 0.650032
filesize is 14986008
number of samples on this MONO wav is : 7493004
allocating memory for input data....finished!!
reading PCM data from File input.raw....Read 7493004 PCM data into memory
allocating memory for input/ouput Double Data....Done!!
Converting PCM data to Double Precision...Done!!
Start Filtering .....DONE!!
Writing Filtered Data to output.raw .....Done !! Exiting Program.
mrbubbles@ubuntu:~/Desktop/experiment$ ls
fdacoefs.h   IIRfilter    input.raw   pallid.wav
fdacoefs.h~  IIRfilter.c  output.raw
mrbubbles@ubuntu:~/Desktop/experiment$ sox -r 44100 -s -w -c 1 output.raw pallid
 HPfilterd.wav
sox soxio: Can't open input file `pallid': No such file or directory
mrbubbles@ubuntu:~/Desktop/experiment$ sox -r 44100 -s -w -c 1 output.raw pallid
filtered.wav
mrbubbles@ubuntu:~/Desktop/experiment$ ls
fdacoefs.h   IIRfilter    input.raw   pallidfiltered.wav
fdacoefs.h~  IIRfilter.c  output.raw  pallid.wav
mrbubbles@ubuntu:~/Desktop/experiment$
```

Έτσι λοιπόν το High Pass φίλτρο που σχεδιάσαμε στο MATLAB με την παραπάνω διαδικασία γίνεται εφαρμογή σε audio αρχείο στο Linux. Μπορούμε να σχεδιάσουμε οποιοδήποτε IIR φίλτρο και να το κάνουμε εφαρμογή σε wav αρχείο με την παραπάνω διαδικασία προσέχοντας πάντα το αρχείο c header να έχει την ονομασία fdacoefs.h. Παρακάτω με την βοήθεια του Audacity βλέπουμε την συχνοτική ανάλυση πριν την εφαρμογή αλλά και μετά την εφαρμογή του φίλτρου:




-29dB
-36dB
-42dB
-48dB
-54dB
-60dB
-66dB
-72dB
-78dB
-84dB
-90dB
1000Hz
3000Hz
5000Hz
7000Hz
9000Hz
11000Hz
13000Hz
15000Hz
17000Hz
19000Hz
21000Hz
Cursor: 11213 Hz (F9) = -34 dB
Peak: 10991 Hz (F9) = -32.9 dB
Algorithm:
Spectrum
Size:
512
Export...
Function:
Hanning window
Axis:
Linear frequency
Close

Συμπεράσματα

Βλέπουμε λοιπόν πως με ένα λειτουργικό σύστημα Linux/Unix και χωρίς κανένα κόστος μπορούμε να επιτύχουμε την δημιουργία ψηφιακών φίλτρων. Στο διαδύκτιο υπάρχουν πολλές ελεύθερες βιβλιοθήκες για να πάρουμε ιδεές ή και δομές φίλτρων. Το πώς θα τα χρησιμοποιήσουμε και κατα βάση πώς θα τα προγραμματίσουμε είναι αυτό που θα κάνει το κάθε φίλτρο ξεχωριστό. Με βασικές γνώσης της γλώσσας C πραγματοποιήσαμε 4 φίλτρα και είδαμε στην πράξη πως εφαρμόζονται πάνω σε wav αρχεία. Επίσης με τη βοήθεια του προγράμματος MATLAB και το εργαλείο FDATool μπορούμε πολύ εύκολα να σχεδιάσουμε οποιοδήποτε φίλτρο IIR και με την διαδικασία που περιγράφεται να εφαρμόσουμε με λίγα βήματα σε audio αρχεία το εκάστοτε φίλτρο που χρειαζόμαστε . Επίσης το λειτουργικό σύστημα Linux κάθε μέρα αποκτάει ακόμα και περισσότερους χρήστες και έχει μια τεράστια κοινότητα αφοσιωμένων χρηστών. Με βασικό όπλο του το μηδενικό κόστος του σε σχέση με τις υπηρεσίες που προσφέρει σε λίγο καιρό θα είναι παντού και θα χρησιμοποιείται για όλες τις εφαρμογές. Ήδη για τις τηλεποικοινωνίες αλλά και τις δικτυακές εφαρμογές το Linux είναι επαγγελματικό Standart. Όσο περνάει ο καιρός ολοένα και περισσότερες εταιρείες δραστηριοποιούνται για την ανάπτυξη επαγγελματικού software και δεν θα αργήσει η μέρα που και για την μουσική τεχνολογία αλλά και για τις εφαρμογές ακουστικής το λειτουργικό συστημα Linux θα είναι ανάμεσα στις πρώτες επαγγελματικές επιλογές μας. Οπότε η ψηφιακή επεξεργασία σήματος και ο προγραμμάτισμος φίλτρων σε συνάρτηση με το Linux που επετεύχθει με αυτήν την εργασία είναι ένα μικρό βήμα για τις εξελίξεις που πρόκειται να επακολουθήσουν.

ΒΙΒΛΙΟΓΡΑΦΙΑ

- Α. Σκόδρας& Β. Αναστόπουλος: 'ΨΗΦΙΑΚΗ ΕΠΕΞΕΡΓΑΣΙΑ ΣΗΜΑΤΟΣ ΚΑΙ ΕΙΚΟΝΑΣ'
- Linux For Dummies 8th Edition by Dee-Ann LeBlanc and Richard Blum
- Understanding Digital Signal Processing by Richard G. Lyons
- Digital Signal Processing: Principles, Algorithms, and Applications by J. G. Proakis and D. G. Manolakis
- Theory and Application of Digital Signal Processing by Rabiner and Gold. A comprehensive, industrial-strength DSP reference book.
- Digital Signal Processing by Alan V. Oppenheim and Ronald W. Schafer.
- Digital Signal Processing by William D. Stanley.
- The Scientist and Engineer's and Guide to Digital Signal Processing by Steven W. Smith.
- Mixed Signal and DSP Design Techniques edited by Walt Kester
- Συστήματα πολυμέσων , Νικήτας Σγουρός
- Don Davis & Carolyn Davis (1997), "SOUND SYSTEM ENGINEERING" second edition, Focal Press, USA
- Gary Davis &Ralph Jones (1989), "SOUND REINFORCEMENT HANDBOOK" second edition, Yamaha, USA
- Digital Signal Processing (Dsp) Handbook - Madisetti & Williams - Crc Press
- Digital Filter Designer's Handbook – C. Britton Rorabaugh

ΔΙΑΔΥΚΤΙΟ

- www.linux.gr (17/12/07)
- www.hellug.gr (18/12/07)
- http://linux.techteam.gr (18/12/07)
- www.centos.org (13/12/07)
- http://www.ibiblio.org/pub/Linux/ (22/12/07)
- http://www.oreilly.com/pub/topic/linux (22/12/07)
- http://www.linuxquestions.org/ (24/1/08)
- http://www.bores.com/courses/intro/iir/index.htm (12/2/08)
- http://www.dsptutor.freeuk.com/IIRFilterDesign/IIRFiltDes102.html (14/2/08)
- http://ptolemy.berkeley.edu/java/Filter.html (15/2/08)
- http://www.etro.vub.ac.be/Research/DSSP/Education/DSP1/files/iir_les5_Eng.pdf (17/12/07)

ΕΥΧΑΡΙΣΤΙΕΣ

Τέλος θα ήθελα να ευχαριστήσω τους καθηγητές μου κ.Μπακαρέζο Ευθύμιο για τις συμβουλές του, κ.Ποταμίτη Ηλία που ανέλαβε την επίβλεψη της πτυχιακής, τον προιστάμενο του τμήματος μας κ. Παπαδογιάννη Νεκτάριο και τον κ.Αγγελή απο την Ελληνική Αεροπορική Βιομηχανία που με βοήθησε κατα την διάρκεια και της πρακτικής αλλά και της πτυχιακής μου εργασίας.

Αυτή η πτυχιακή είναι αφιερωμένη στην οικογένεια μου και την Ελένη Μπουγά για την συμπαράσταση τους σε όλες μου τις επιλογές όλα αυτά τα χρόνια.