Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Σύγκριση σημάτων μουσικής με πρότυπα συμβολικής διαμόρφωσης**
Ονοματεπώνυμο Φοιτητή	**Αντώνης Κύρκος**
Πατρώνυμο	**Χρήστος**
Αριθμός Μητρώου	**ΜΠΠΛ/ 09061**
Επιβλέπων	**Άγγελος Πικράκης, Λέκτορας**

Ημερομηνία Παράδοσης **Ιούλιος 2013**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Άγγελος Πικράκης Λέκτορας	Γεώργιος Τσιχριτζής Καθηγητής	Χαράλαμπος Κωσταντόπουλος Επίκουρος Καθηγητής

Περίληψη

Η μουσική είναι αναμφίβολα για τους περισσότερους ανθρώπους, ένα κομμάτι της ψυχοσύνθεσης του καθενός μας. Η πολυπλοκότητά της μας δίνει άπειρους συνδυασμούς ακόμα και σε όμοια μουσικά κομμάτια.

Ο τρόπος έκφρασης και το ιδιαίτερο παίξιμο του κάθε μουσικού μας μεταφέρει και διαφορετικά ακουστικά ερεθίσματα.

Φυσικά πρέπει να ορίσουμε ένα μέτρο στο να μπορέσουμε να στοιχειοθετήσουμε το πότε ένα μουσικό κομμάτι είναι σωστά παιγμένο και πότε δεν είναι. Πότε αντιστοιχεί το παίξιμο στην ανάλογη παρτιτούρα και πόσο αποκλίνει από αυτή.

Αυτό είναι και το γενικό ζητούμενο στην παρούσα διπλωματική εργασία. Έχουμε ένα σύνολο στοιχείων που αντιστοιχούν σε δύο απλοϊκά μουσικά παιξίματα. Το πρώτο θεωρείται πρότυπο και αντιστοιχεί σε παρτιτούρα και το δεύτερο αντιστοιχεί σε μια προσπάθεια εκτέλεσης της παρτιτούρας αυτής.

Συγκρίνουμε τα στοιχεία και προσπαθούμε να εξάγουμε σχετικά αριθμητικά συμπεράσματα για το πόσο καλά αντιπροσωπεύει το παίξιμο την παρτιτούρα.

Abstract

Music is certainly for most people, a part of everyone's psyche. Its complexity gives us infinite combinations even at similar tracks.

The way of expression and distinctive playing of each musician brings us different auditory stimuli.

Of course we need to define a measure to be able to substantiate when a track is properly played and when it is not. When the playing is corresponding at score and how much it deviates from this.

This is the general aim of this paper. We have a set of data corresponding to two simple musical pieces. The first is standard and corresponds to the score and the second corresponds to a try to play the score.

We compare the data and we try to have conclusions about figures on how well the playing represents the score.

1. Εισαγωγή

Με την παρούσα εργασία θα προσπαθήσουμε να προσεγγίσουμε το πρόβλημα συγκρίσεως σειρών βάση πρότυπου αναφοράς για σειρές που προέρχονται από μια μουσική αντιστοιχία, αλλιώς θα μπορούσαμε να το χαρακτηρίσουμε σαν audio to midi alignment.

Εισάγουμε βάση του midi πρωτόκολλου μια σειρά δεδομένων όπου τα χρησιμοποιούμε σαν πρότυπο στην ταξινόμησή μας.

Στη συνέχεια εισάγουμε ένα αρχείο με ψηφιακή κωδικοποίηση σήματος ή μπορούμε και να εγγράψουμε ζωντανά και να το χρησιμοποιήσουμε για το δοκιμαστικό μας για την σειρά δεδομένων προς σύγκριση.

Χρησιμοποιούμε επεξεργασία σήματος με σκοπό να φιλτράρουμε το εισαγόμενο σήμα με σκοπό να μπορέσουμε να μελετήσουμε τις αλλαγές του σήματος και να εξαλείψουμε ένα ποσοστό θορύβου και λαθών που παράγονται από την εγγραφή και την μετατροπή του σήματος μέχρι την τελική σύγκριση.

Μετατρέπουμε τα στοιχεία μας από το πεδίο του χρόνου στη συχνότητα και με αλγόριθμους δυναμικού προγραμματισμού θα συγκρίνουμε της παραγόμενες σειρές στοιχείων με σκοπό την εξαγωγή συμπερασμάτων που να αντιστοιχούν στην ομοιότητα-ανομοιότητα του πρότυπου με το προ σύγκριση δοκιμαστικό.

Η εκτέλεση της εργασίας θα είναι σε περιβάλλον matlab και μέσα από ένα gui εξελιγμένο εργαλείο που μα επιτρέπει να πειραματιστούμε σε διαφορετικές απλοϊκές μουσικές σειρές και να τις επεξεργαστούμε ανάλογα.

Το παρόν αποτελεί ένα πολύ καλό εκπαιδευτικό εργαλείο για την γενική παρατήρηση μουσικών εγγραφών και μπορεί να αποτελέσει την βάση για ανάπτυξη σε μεγαλύτερο βάθος της επεξεργασίας μουσικών σημάτων μαζί με την συμβολή γνώσης από την επεξεργασία φωνής που ήταν και η βασική αναφορά για την εκτέλεση της παρούσας εργασίας.

1.1 Περιγραφή Προβλήματος

Έχουμε ένα πρότυπο αρχείο και ένα δοκιμαστικό, όπου και τα δύο αντιστοιχούν σε όμοια μουσικά παιξίματα.

Θέλουμε να συγκρίνουμε τα δύο αυτά αρχεία και να μπορέσουμε να καταλάβουμε πόσο όμοια ή ανόμοια είναι τα δύο αυτά μουσικά παιξίματα

Μετά από αυτή την επεξεργασία πρέπει να εξάγουμε δεδομένα που να αντιστοιχούν σε συμπεράσματα για την ποιότητα του αντίστοιχου παιξίματος.

2. MIDI Πρωτόκολλο

Το MIDI είναι ένα πρωτόκολλο που επιτρέπει τη σύνδεση, το συγχρονισμό και την επικοινωνία ηλεκτρονικών μουσικών οργάνων και ηλεκτρονικών υπολογιστών. Η λέξη MIDI είναι ακρωνύμιο του Musical Instrument Digital Interface, το οποίο σε ελληνική απόδοση σημαίνει Ψηφιακή διασύνδεση Μουσικών Οργάνων.

Το πρωτόκολλο MIDI δεν μεταδίδει ηχητικό σήμα, αλλά μηνύματα που περιέχουν πληροφορίες σχετικά με το **τονικό** ύψος και την ένταση μιας νότας, καθώς επίσης και σήμα χρονισμού που προσδιορίζει την ταχύτητα - το tempo - ενός κομματιού.

2.1 Ορισμός του MIDI

Το πρωτόκολλο MIDI (Musical Instrument Digital Interface) αποτελεί την ραχοκοκαλιά του συστήματος επικοινωνίας όλων των synthesizers μεταξύ τους αλλά και με τους υπολογιστές που μπορεί να χρησιμοποιούμε (εάν υποστηρίζουν υλοποίηση MIDI πρωτοκόλλου βέβαια). Πρόκειται για ένα σύνολο προδιαγραφών που καλύπτουν την μεταφορά δεδομένων σε καλώδια που συνδέουν μεταξύ τους τα synthesizers ή/και τους υπολογιστές και φροντίζει ώστε να υπάρχει αλληλοσυννενόηση μεταξύ όλων αυτών των (ετερογενών) μηχανημάτων, όπως στους ηλεκτρονικούς υπολογιστές με λογισμικό μουσικής εγγραφής (Sequencer), στις ρυθμομηχανές (drum-machines), στους δειγματολήπτες (samplers), στους συνθετητές με δυνατότητα μουσικής εγγραφής (Workstation Synthesizer), στις συσκευές συγχρονισμού κ.ά., ανεξαρτήτως κατασκευαστή.

Τα δεδομένα που μεταβιβάζονται μέσω της σύνδεσης MIDI δεν περιγράφουν κάποιο ηχητικό κύμα αλλά περιέχουν πληροφορίες, σε ψηφιακή μορφή, για την περιγραφή και χειρισμό **μουσικών γεγονότων (events)** σε πραγματικό χρόνο. Τα μουσικά γεγονότα μπορεί να αφορούν το πάτημα ή την αποδέσμευση ενός συγκεκριμένου πλήκτρου, την ταχύτητα (ένταση) με την οποία πιέζεται ένα πλήκτρο, καθώς και δεδομένα που αφορούν την κίνηση κάποιου πεντάλ ή άλλης μονάδας ελέγχου ενός ηλεκτρονικού μουσικού οργάνου.

Ο τρόπος λειτουργίας του MIDI έχει ως εξής: Όταν πιέσουμε, για παράδειγμα, το πλήκτρο του μεσαίου Ντο κάποιου ηλεκτρονικού μουσικού οργάνου που υποστηρίζει το πρωτόκολλο MIDI, τότε το όργανο θα εκτελέσει τη νότα Ντο. Ταυτόχρονα θα παραγάγει ένα κωδικοποιημένο σήμα στην MIDI έξοδό του το οποίο είναι αναγνωρίσιμο από κάθε συσκευή MIDI. Το σήμα αυτό θα περιλαμβάνει τις εξής παραμέτρους:

α) Note - On (Έναρξη εκτέλεσης νότας)

β) Τονικό ύψος (Pitch) – Μεσαίο Ντο,

γ) Ταχύτητα (Velocity) – Μια τιμή από το 0-127 σύμφωνα με την οποία θα καθορίζεται η ένταση της νότας.

Αν το σήμα αυτό μεταδοθεί σε ένα άλλο όργανο MIDI τότε αυτό είναι δυνατόν να εκτελέσει ταυτόχρονα την ίδια νότα με την ίδια ένταση και με κάποιο δικό του ήχο.

Το πρωτόκολλο MIDI προβλέπει ότι το κάθε όργανο MIDI αποτελείται από 16 MIDI-κανάλια στα οποία είναι δυνατόν να στέλνονται ταυτόχρονα και ανεξάρτητα δεδομένα με την χρήση ενός μόνο καλωδίου.

Μια από τις πιο συνηθισμένες χρήσεις των δεδομένων MIDI είναι η αποθήκευση τοποθετημένων με βάση το χρόνο, ακολουθιών μουσικών γεγονότων σε ψηφιακή μνήμη. Με την εγγραφή και στη συνέχεια αναπαραγωγή μιας τέτοιας ακολουθίας μουσικών γεγονότων, μπορούμε να πετύχουμε την αναπαράσταση της εκτέλεσης ενός μουσικού έργου με

αποτέλεσμα παρόμοιο με μια συνηθισμένη ηχογράφηση. Η λειτουργία αυτή εκτελείται από το sequencer. Ένα sequencer μπορεί είτε να είναι εξειδικευμένη συσκευή που πολλές φορές είναι ενσωματωμένη σε ηλεκτρονικά μουσικά όργανα είτε εξειδικευμένο μουσικό λογισμικό σε ηλεκτρονικό υπολογιστή. Το μουσικό λογισμικό Cubase, μεταξύ άλλων, λειτουργεί και ως ένα MIDI sequencer τεραστίων δυνατοτήτων.

General MIDI (GM)

Το πρότυπο General MIDI αφορά στην τοποθέτηση των διαφόρων οργάνων και ήχων με μια συγκεκριμένη σειρά (σε 128 θέσεις – patches) ώστε όλες οι συσκευές MIDI που χρησιμοποιούν αυτό το πρότυπο να μπορούν εύκολα να εκτελέσουν ακολουθίες MIDI με τους σωστούς ήχους όταν γίνεται μεταφορά από μια συσκευή στην άλλη. Στα 15 από τα 16 MIDI-κανάλια μπορούν να τοποθετηθεί αντίστοιχος αριθμός ήχων κυρίως από μελωδικά όργανα ενώ το MIDI-κανάλι με αριθμό 10 είναι κρατημένο αποκλειστικά για κρουστούς ήχους.

2.2 Ιστορία του MIDI

Η ιστορία του MIDI είναι σχετικά πρόσφατη. Μέχρι το 1970 τα πιο πολλά συνθεσάιζερ ήταν αναλογικά και η διασύνδεσή τους απαιτούσε πολύ καλές γνώσεις ηλεκτρονικών. Αργότερα, εμφανίστηκαν τα πρώτα ψηφιακά μουσικά όργανα, αλλά χωρίς να απευθύνονται στο μεγάλο πλήθος των ερασιτεχνών μουσικών. Ήταν πιο πολύ κλειστά συστήματα που δεν διέθεταν τρόπους διασύνδεσης, ή αν είχαν κάποιες δυνατότητες ενδοεπικοινωνίας, αυτές προϋπήρχαν μόνο για συσκευές και κονσόλες της ίδιας εταιρίας. Επιπλέον, και οι προσωπικοί υπολογιστές ήταν ακριβοί και οι κάρτες ήχου ακόμα ακριβότερες. Το πιο σημαντικό πρόβλημα, παρόλα αυτά, δεν ήταν το ακριβό υλικό, αλλά η έλλειψη προτύπων που θα καθόριζαν την ενδοεπικοινωνία μεταξύ ηλεκτρονικών μουσικών οργάνων

Κάθε ένα μοντέλο από τα πρώτα συνθεσάιζερ είχε μία ιδιαίτερη «προσωπικότητα». Μερικά είχαν υπέροχους «παχιούς» ήχους χάλκινων πνευστών. Άλλα ήταν πιο πιστά στα ξύλινα ή τα έγχορδα ή στο πιάνο. Αυτό λοιπόν που ήταν αναγκαίο ήταν ένας τρόπος με τον οποίο ένας μουσικός να μπορεί να συνδυάσει τους καλύτερους ήχους του κάθε οργάνου σε ένα και μόνο εύχρηστο μουσικό σύστημα. Το 1979, κάποια καινούργια κλαβιέ από τις εταιρίες Oberheim, Rhodes και Roland κατασκευάσθηκαν με βύσματα διασύνδεση {interface plugs} στο πίσω μέρος τους, μέσω του οποίου μπορούσαν να συνδεθούν μεταξύ τους. Αυτή η σύνδεση γινόταν μόνο μεταξύ οργάνων της ίδιας εταιρίας. Όταν λοιπόν παίζαμε στο ένα κλαβιέ, ακουγόταν, εκτός από αυτό και όλα τα άλλα κλαβιέ που είχαμε συνδέσει με αυτό. Αυτή η λύση έλυνε κατά ένα μέρος το πρόβλημα αλλά δεν έδινε απάντηση στο μεγάλο ερώτημα του πως να συνδεθούν μεταξύ τους διάφορα όργανα από διαφορετικές εταιρίες

Δύο φορές το χρόνο, η NAMM (National Association of Music Merchandisers) ή Εθνική Ένωση Μουσικών Εμπόρων, οργανώνει ένα διεθνές συνέδριο για την επίδειξη καινούργιων μουσικών προϊόντων και την εύρεση νέων μεθόδων για τη προώθηση των μουσικών οργάνων και των εξαρτημάτων. Στη διάρκεια ενός τέτοιου συνεδρίου το 1982, έγινε μια συνάντηση από μία μικρή ομάδα κατασκευαστών συνθεσάιζερ με την καθοδήγηση του Dave Smith, προέδρου της Sequential Circuits, μιας γνωστής εταιρίας συνθεσάιζερ αυτής της εποχής. Κατά τη συνάντηση αυτή συζητήθηκε μία πρόταση για την αποδοχή, από όλους τους κατασκευαστές συνθεσάιζερ και εν γένει ηλεκτρονικών μουσικών οργάνων, ενός παγκοσμίου πρωτόκολλου για την μετάδοση και λήψη πληροφοριών σχετικών με τη μουσική εκτέλεση, μεταξύ όλων των εικών των ηλεκτρονικών μουσικών οργάνων. Η αρχική αυτή πρόταση ονομάσθηκε UMI (Universal Musical Interface) ή Παγκόσμια Μουσική Διασύνδεση.

Στη συνέχεια, αυτή η πρόταση πέρασε από ένα σημαντικό αριθμό αναθεωρήσεων πριν να γίνει γνωστή με το όρο MIDI ή MIDI Standard {Πρωτόκολλο MIDI}.Τελικά, το 1983 η Sequential Circuits από την Αμερική και η Roland από τη Ιαπωνία παρουσίασαν τα πρώτα κλαβιέ με MIDI και σύντομα ακολούθησαν ουσιαστικά όλες οι άλλες εταιρίες συνθεσάιζερ. . Έτσι δημιουργήθηκε η τυποποίηση 1.0 του MIDI. Το πρότυπο αυτό δεν είναι το καλύτερο δυνατό. Έχει κάποιες ατέλειες και αρκετοί κατασκευαστές το παραλλάσσουν και δημιουργούν δικά τους πρότυπα. Παρόλα αυτά, όλοι οι κατασκευαστές υποστηρίζουν το γενικό πρότυπο του MIDI.

Το διασυνδετικό MIDI δημιουργήθηκε το 1982 με κύριο αίτιο την ανάγκη εύρεσης ενός κοινά αποδεκτού τρόπου επικοινωνίας ανάμεσα στις συσκευές των διάφορων κατασκευαστών, ώστε να επιτευχθεί ένα μεγαλύτερο άνοιγμα στην αγορά που είχε βρεθεί σε αδιέξοδο. Μέχρι τότε, κάθε κατασκευάστρια εταιρεία έκανε χρήση επικοινωνιακών μεθόδων περιορισμένων και ασύμβατων με μηχανήματα άλλων κατασκευαστών, και δεν φαινόταν με ποιο τρόπο θα μπορούσαν να εκμεταλευτούν την επερχόμενη άνοδο των προσωπικών υπολογιστών, ενώ η απουσία κοινής γλώσσας επικοινωνίας των μηχανημάτων διαφορετικών εταιρειών οδηγούσε σε μια κλειστή αγορά.

Έτσι, ειδικοί προερχόμενοι από τους τότε μεγάλους κατασκευαστές ηλεκτρονικών οργάνων, συνεργάστηκαν ώστε να πετύχουν την επικοινωνία των μικροεπεξεργαστών που όλα σχεδόν τα μουσικά όργανα διέθεταν την εποχή εκείνη. Αυτό οδήγησε σε μια μεγάλη επιτυχία όπου δύο διαφορετικοί συνθετητές συνδεδεμένοι μπόρεσαν να αναγνωρίσουν τα μεταδιδόμενα μηνύματα ο ένας του άλλου ώστε, να μπορούν να παιχτούν νότες και εντάσεις στο κλαβιέ (πλήκτρα) του ενός και να αναπαράγονται ταυτόχρονα από το άλλο, ακριβώς σαν να παίζονταν στα δικά του πλήκτρα.

Πλέον τα όργανα μπορούσαν να επικοινωνούν με μια κοινή γλώσσα και για πρώτη φορά, τα χαρακτηριστικά μιας μουσικής εκτέλεσης, όπως π.χ. το ύψος των ήχων που παίζονται, η έντασή τους κ.ά., μεταδίδονται υπό μορφή πληροφόρησης από και προς διαφορετικές συσκευές ή προσωπικούς υπολογιστές.

Από το 1985 και μετά σχεδόν κάθε νέα συσκευή που παρουσιαζόταν διέθετε ενσωματωμένο το διασυνδετικό MIDI, πέρα από την κλασική υποδοχή για σύνδεση της ηχητικής του εξόδου (audio out) σε ενισχυτή με ηχεία. Από εκεί και πέρα το MIDI εξελίσσεται από τις εταιρείες και το συναντάμε σε διάφορες ορολογίες: MIDI Language (γλώσσα MIDI), MIDI Code (MIDI κώδικας), MIDI Protocol (πρωτόκολλο MIDI), MIDI Specification (προδιαγραφή MIDI) και MIDI Standard (πρότυπο MIDI).

Όσον αφορά τον τρόπο μετάδοσης των πληροφοριών του πρωτοκόλλου MIDI, επιλέχθηκε ο *σειριακός* και όχι ο *παράλληλος*, αν και η σειριακή μετάδοση είναι αισθητά πιο αργή από την παράλληλη.

Παράλληλη μετάδοση πληροφόρησης σημαίνει ότι τα 8 bits που περιέχει κάθε byte δεδομένων μεταφέρονται ταυτόχρονα, κατά μήκος οκτώ διαφορετικών παράλληλων γραμμών από ένα καλώδιο

Σειριακή μετάδοση πληροφόρησης σημαίνει ότι τα 8 bits κάθε byte δεδομένων μεταφέρονται κατά μήκος μιας γραμμής μέσα σ' ένα καλώδιο. Αυτή η μορφή επικοινωνίας είναι πιο αργή αλλά και πιο οικονομική, κι επομένως προσιτή στον μέσο καταναλωτή. Πράγματι, η επιλογή αυτή δικαίωσε τους τεχνικούς καθώς η οικονομικότερη λύση βοήθησε στην γρηγορότερη διάδοση της τεχνολογίας. Επίσης η ταχύτητα μετάδοσης των 31,250 bit/δευτερόλεπτο κρίθηκε ικανοποιητική ακόμη και για τους πιο απαιτητικούς χρήστες.

Μέχρι σήμερα το πρωτόκολλο MIDI δεν έχει υποκατασταθεί από ένα άλλο σύστημα διασύνδεσης ηλεκτρονικών μουσικών οργάνων σύστημα, πάνω στο οποίο θα ήταν δυνατή η

δόμηση ενός στούντιο – κάτι σαν αυτό που σήμερα συνηθίζουμε να ονομάζουμε MIDI Home Studio. Και όπως και οι υπολογιστές, έτσι και το MIDI χρησιμοποιείται σήμερα από εκατομμύρια μουσικών είτε αυτοί είναι επαγγελματίες είτε ερασιτέχνες για πολλές εφαρμογές που έχουν να κάνουν με τη σύνθεση, τον αυτοσχεδιασμό, την εκπαίδευση, την επεξεργασία παρτιτούρας κλπ. Ακόμα το Πρωτόκολλο MIDI θα το συναντήσουμε και σε χώρους εκτός της μουσικής, όπως ο φωτισμός θεάτρων, τα ηλεκτρονικά παιγνίδια μέσω υπολογιστών, στις επικοινωνίες, τα στούντιο ηχοληψίας, κλπ.

2.3 Βασικές έννοιες του MIDI

Απ' ότι φαίνεται λοιπόν, το πρότυπο MIDI κωδικοποιεί βασικές λειτουργίες των ηλεκτρονικών μουσικών οργάνων.

Μερικές από τις βασικές αυτές λειτουργίες περιγράφονται ευθύς αμέσως.

Master Device

Είναι η βασική μονάδα ελέγχου του συγκροτήματος από συνδεδεμένα μουσικά όργανα. Η συνηθέστερη μονάδα αυτού του τύπου είναι ο ίδιος ο υπολογιστής.

Keyboard Controllers

Πρόκειται για μουσικά όργανα που μπορούν να ανταλλάξουν MIDI σήματα, αλλά συνήθως δεν παράγουν τόνους ή άλλους ήχους. Χρησιμοποιούνται σαν ελεγκτές ομάδων (cluster controllers) άλλων MIDI οργάνων.

Keyboard Synthesizers

Αυτά είναι μουσικά όργανα που παίζουν την μουσική που έχει κωδικοποιηθεί στο πρότυπο MIDI. Διαθέτουν δυνατότητες παραγωγής ήχου και μπορούν να χρησιμοποιηθούν και σαν κύριες συσκευές σε σύνδεση κύριου-υποτελή (master-slave).

Slave Devices

Αυτές είναι οι συσκευές που ελέγχονται από τα συνθεσάιζερ της προηγούμενης κατηγορίας. Συνήθως είναι όργανα που ειδικεύονται στην παραγωγή ειδικών εφφέ ή παράγουν ήχους τύμπανων ή λοιπών κρουστών οργάνων ή είναι οι λεγόμενοι expanders.

Expanders

Τα όργανα αυτά είναι συνθεσάιζερ χωρίς πληκτρολόγια και γι' αυτό είναι υποτελείς στις master-slave συνδέσεις. Προτιμώνται γιατί είναι φθηνότερα και ελαφρύτερα από τα πλήρη συνθεσάιζερ και χρησιμεύουν για ενδιάμεσες ή τελικές βαθμίδες σε μεγάλες αλυσίδες MIDI συστημάτων.

Πολλές φορές η ίδια κάρτα ήχου χρησιμοποιείται ως expander. Δηλαδή μπορεί να παίξει κομμάτια MIDI μουσικής χωρίς να υπάρχει σύνδεση με συνθεσάιζερ χρησιμοποιώντας τις DSP δυνατότητες που έχει.

Sequencers

Οι sequencers είναι συσκευές ή προϊόντα λογισμικού όπου μπορεί να καταγραφεί με όρους MIDI μια μουσική σύνθεση. Σε πολλούς sequencers μπορεί να καταγράψει κανείς και μουσικές φράσεις που παίζει σε συνδεδεμένο συνθεσάιζερ.

Drum Machines

Αποκαλούνται επίσης και drum computers. Πρόκειται για συσκευές που χρησιμεύουν για την παραγωγή των χτυπημάτων του ρυθμού και γενικά για την προσομοιωμένη παραγωγή ήχου

κρουστών οργάνων (percussion). Σαν μελωδικά ηλεκτρονικά όργανα επικοινωνούν με το πρότυπο MIDI.

Velocity

Με την έννοια της ταχύτητας προσπαθούμε να καλύψουμε το γεγονός ότι έχει σημασία για την παραγωγή του ήχου ο τόπος και η ταχύτητα που ο εκτελεστής μουσικής φράσης χτυπάει τα πλήκτρα του οργάνου ή του κρουστού ή με ποια ταχύτητα χειρίζεται τις χορδές του έγχορδου οργάνου. Τα πιο πολλά πληκτρολόγια MIDI αναγνωρίζουν την παράμετρο αυτή και συνεπώς μπορούμε να κωδικοποιήσουμε την ένταση της ταχύτητας εκτέλεσης των νότων. Αυτό τελικά μεταφράζεται σε ένταση του ηχητικού σήματος για τις παραγόμενες νότες. Το σύστημα MIDI αναγνωρίζει 128 διαφορετικές στάθμες ταχύτητας. Αυτή η κβάντωση γενικά θεωρείται περιορισμένη σε σχέση με τις λεπτές διαφοροποιήσεις που μπορούν να πετύχουν εμπνευσμένοι μουσικοί με ακουστικά όργανα.

Aftertouch

Το γνώρισμα αυτό αναφέρεται στο βαθμό πίεσης που ασκείται σε ένα πλήκτρο την ώρα που παίζει μια νότα μέχρι ο μουσικός εκτελεστής να το απελευθερώσει. Και πάλι το πρότυπο MIDI διακρίνει 128 διαφορετικές στάθμες κβάντωσης του aftertouch.

Έχουμε δύο είδη aftertouch: το μονοφωνικό και το πολυφωνικό. Στο μονοφωνικό το όργανο αισθάνεται την πίεση μόνο του ισχυρότερα πιεζόμενου πλήκτρου της συγχορδίας ενώ στο πολυφωνικό το όργανο ανιχνεύει την πίεση κάθε πλήκτρου της συγχορδίας και τα αποδίδει το καθένα ξεχωριστά.

Pitch bend

Ένα καλό MIDI οφείλει να μην αναγνωρίζει τις νότες ως δεδομένες και άκαμπτες συχνότητες. Καλό θα ήταν να μπορούσε να λυγίσει τις νότες που παίζει, όπως για παράδειγμα μία κιθάρα γλιστράει λίγο πιο πάνω ή λίγο πιο κάτω από τη συχνότητα δεδομένης νότας.

2.4 Τα κανάλια του MIDI

Οι προδιαγραφές του πρωτόκολλου έχουν προβλέψει ώστε οι MIDI πληροφορίες να μπορούν να μεταδοθούν σε μέχρι και 16 διαφορετικά κανάλια - Οι σύγχρονοι συνθετητές είναι εξοπλισμένα με τη δυνατότητα αυτή.

Τα MIDI κανάλια προσφέρουν την πολύ σημαντική υπηρεσία της ταυτόχρονης αναπαραγωγής 16 διαφορετικών ήχων από ένα και μόνο synthesizer αλλά και άλλες που θα δούμε παρακάτω.

Για παράδειγμα, ας πούμε ότι ένας χρήστης που συνθέτει με τη βοήθεια Η/Υ τη μουσική του, κατέχει τον παρακάτω εξοπλισμό:

Συνθετητή εξοπλισμένο με υποδοχές MIDI και συνδεδεμένο με τα κατάλληλα καλώδια με έναν Η/Υ εξοπλισμένο με κάρτα ήχου που έχει δυνατότητα MIDI και ειδικό software που μπορεί να καταγράφει MIDI πληροφορίες και λέγεται *sequencer*.

Τότε, αφού συνδέσει σωστά τα σχετικά καλώδια και ελέγξει την ορθή λήψη και αποστολή των MIDI μηνυμάτων, εκκινεί το πρόγραμμα εγγραφής (Sequencer) το οποίο μπορεί να είναι ένα από τα εμπορικά Cakewalk και Cubase, ή κάποιο άλλο με παρόμοιες λειτουργίες.

Επισημαίνουμε ότι στα προγράμματα εγγραφής, το κάθε "κανάλι" όπως λέγεται, είναι οπτικοποιημένο και αντιπροσωπεύεται από μία χρωματιστή λωρίδα η οποία στην ειδική ορολογία ονομάζεται *"channel strip"* (δηλ. *η λωρίδα του καναλιού*). Πατώντας επάνω στην κάθε λωρίδα μπορούμε να της αναθέσουμε το κανάλι που θα αντιπροσωπεύει, το όργανο που θα αναπαράγει, την ένταση και πολλά άλλα χαρακτηριστικά. Από τη στιγμή που ο συνθετητής είναι

συνδεδεμένος με τον Η/Υ, τα πατήματα και οι ρυθμίσεις που γίνονται στην κάθε λωρίδα-κανάλι, ταυτόχρονα γίνονται αντιληπτά και από τον συνθετητή και αντίστροφα.

Ο χρήστης, προκειμένου να συνθέσει τη μουσική του ας πούμε ότι θα χρησιμοποιήσει τα εξής :

Το κανάλι 1 όπου θα ορίσει να καταγράφει και να αναπαράγει το Πιάνο.

Το κανάλι 2 όπου θα ορίσει να καταγράφει και να αναπαράγει το Μπάσο.

Το κανάλι 3 όπου θα ορίσει να καταγράφει και να αναπαράγει την Τρομπέτα.

Το κανάλι 4 όπου θα ορίσει να καταγράφει και να αναπαράγει το Σαξόφωνο.

Το κανάλι 10 όπου θα ορίσει να καταγράφει και να αναπαράγει τη Ντραμς ή τα Κρουστά.

Είναι μία από τις προδιαγραφές του MIDI να θεωρείται το κανάλι 10 ως το κανάλι που συνήθως καταγράφεται και αναπαράγεται ο ήχος ντραμς και κρουστών γενικά.

Κατά τη διαδικασία εγγραφής, ο χρήστης θα ακολουθήσει την εξής σειρά :

Θα επιλέξει τη λωρίδα που αντιπροσωπεύει το κανάλι 10, και πατώντας με το ποντίκι το σχετικό πλήκτρο εκκίνησης εγγραφής του Sequencer (που συνήθως είναι στρογγυλό και κόκκινου χρώματος) καταγράφει τον ρυθμό με την βοήθεια των πλήκτρων του συνθετητή (που "μιμείται" τους ήχους της ντραμς). Μόλις τελειώσει, σταματά την εγγραφή.

Κατόπιν, επιλέγει τη λωρίδα που αντιπροσωπεύει το κανάλι 2 και πατώντας με το ποντίκι το σχετικό πλήκτρο εκκίνησης εγγραφής, παίζει στα πλήκτρα του synthesizer τις νότες που αντιστοιχούν στο μπάσο (με ήχο μπάσου φυσικά). Παρατηρούμε όμως ότι οι ήχοι Ντραμς που λίγο πριν καταγράφτηκαν, ακούγονται επίσης, και έτσι μας βοηθούν στο να καταγραφεί με σωστό χρόνο το μπάσο.

Αφού τελειώσει η εγγραφή το μπάσου, ακολουθεί την ίδια διαδικασία ώστε να παίξει τις νότες του πιάνου. Και πάλι, την ώρα της εγγραφής του πιάνου, ακούγονται και η ντραμς αλλά και το μπάσο που λίγο πριν κατέγραψε κ.ο.κ..

Έτσι, με αυτόν τον σταδιακό τρόπο, ο χρήστης μπορεί να καταγράψει ένα-ένα μέχρι και 16 όργανα (αν και υπάρχουν τρόποι ώστε στο ίδιο κανάλι να ακουστούν περισσότερα του ενός όργανα αρκεί να μην τα θέλουμε ταυτόχρονα) και κατόπιν να επιλέξει την αναπαραγωγή τους ώστε να ακουστεί ολοκληρωμένο το μουσικό έργο.

Είναι σημαντικό να σημειώσουμε ότι, τα MIDI μηνύματα που αποστέλνονται με το πάτημα των πλήκτρων του συνθετητή και καταγράφονται στο Sequencer, δεν περιέχουν τον ήχο που ακούμε, αλλά είναι μόνο ψηφιακές πληροφορίες όπως: *"ο χρήστης πάτησε τη νότα Λα της τρίτης οκτάβας"*. *"Ο χρήστης πάτησε τη νότα Ντο της 5ης οκτάβας"*. *"Η ένταση της νότας ήταν **75**(στην κλίμακα των 128)"*. *"Χρησιμοποιήθηκε το κανάλι 7"* κ.λ.π.

Αυτό που τελικά θα ακούσουμε όταν πατήσουμε το κουμπί αναπαραγωγής του Sequencer, δεν είναι τίποτε άλλο από τον ήχο του συνθετητή μας, καθώς θα λαμβάνει πίσω όλες μαζί τις πληροφορίες δηλ. όλα τα καταγεγραμμένα MIDI μηνύματα, που λίγο πριν έστειλε προς τον Η/Υ. Έτσι, κατά την αναπαραγωγή, το Sequencer απλώς στέλνει πίσω στο συνθετητή όλες τις καταγεγραμμένες εντολές και ελέγχει πλέον το synthesizer "δίνοντας εντολή" ώστε να παίξει την τάδε νότα, με τη συγκεκριμένη ένταση, στο τάδε κανάλι, με τον τάδε ήχο κ.ο.κ.

Σε άλλη χρήση των καναλιών MIDI είναι δυνατόν ο χρήστης αντί για Η/Υ να συνδέσει περισσότερα του ενός όργανα μέσω MIDI, τα οποία να ελέγχει μέσω του πρώτου στη σειρά.

Είναι λοιπόν δυνατό το synthesizer *master*, να μεταδίδει τις MIDI πληροφορίες του σε δύο ακόμη όργανα ως εξής :

Το συνθεσάιζερ Α (*master*) στέλνει πληροφορίες MIDI και το συνθεσάιζερ Β τις δέχεται στο κανάλι 2 ενώ το συνθεσάιζερ Γ στο κανάλι 3. Με τον τρόπο αυτό μπορεί ο χρήστης να επιλέξει ώστε το **Α** να αναπαράγει ήχο πιάνου, ενώ το **Β** ήχο από βιολιά και το Γ ήχο από βιολοντσέλα.

Με τον τρόπο αυτό ο χρήστης παράγει έναν ήχο, γεμάτο, πλήρη και ισχυρό, παίζοντας σε ένα και μόνο κλαβιέ (πλήκτρα) αλλά ελέγχοντας ταυτόχρονα 3 διαφορετικά όργανα συνδεδεμένα μέσω MIDI.

Το πρώτο μέρος κάθε αρχείου MIDI ονομάζεται Header Chunk. Σε αυτό, πέραν του προσδιορισμού της ανάλυσης του αρχείου με βάση τις προδιαγραφές του προγράμματος που το δημιούργησε, περιέχονται διάφορες χρήσιμες πληροφορίες. Ένα MIDI FILE μπορεί να περιέχει οποιαδήποτε από τα μηνύματα που περιέχει το πρωτόκολλο MIDI, περιλαμβανομένων και των μηνυμάτων System Exclusive. Κάθε ένα γεγονός του MIDI αρχείου κωδικοποιείται με τρόπο που περιγράφει τη χρονική του απόσταση από το προηγούμενο γεγονός, είτε με χτύπους και υποδιαιρέσεις, είτε με δευτερόλεπτα και υποδιαιρέσεις τους.

2.5 Τρόπος Λειτουργίας του MIDI Πρωτοκόλλου

Το διασυνδετικό MIDI αποτελείται από το πρωτόκολλο επικοινωνίας (software) και το υλικό του μέρος (hardware). Το πρώτο, περιέχει τις εντολές (*MIDI Language*) που χρειάζονται για την "περιγραφή" μιας μουσικής σύνθεσης, ενώ το δεύτερο αναφέρεται στα χαρακτηριστικά των κυκλωμάτων που παράγουν και ερμηνεύουν την πληροφόρηση MIDI και προσδιορίζει τον τύπο των συνδέσεων και το είδος των καλωδίων που πρέπει να χρησιμοποιηθούν.

Τώρα, όσον αφορά τον τρόπο λειτουργίας του, το MIDI μεταφέρει πληροφορίες μεταξύ τερματικών συσκευών που μπορούν να καταλαβαίνουν το Πρωτόκολλο MIDI. Για την κατανόηση αυτής της έννοιας μπορούμε να θεωρήσουμε ότι δύο συνομιλητές είναι δύο τερματικές συσκευές. Ο ένας από αυτούς είναι ο αποστολέας μιας πληροφορίας και ο άλλος είναι ο δέκτης ή και το αντίστροφο. Το MIDI δηλαδή ένας τρόπος μετάδοσης και αποδοχής πληροφοριών, με άλλα λόγια ένας τρόπος επικοινωνίας με τον εξωτερικό κόσμο, είτε αυτός είναι ο χρήστης είτε άλλες συσκευές. Ένα όργανο που είναι εξοπλισμένο με το MIDI Πρωτόκολλο μπορεί να επικοινωνήσει με οποιοδήποτε άλλο όργανο που είναι επίσης εξοπλισμένο με το MIDI Πρωτόκολλο, ανεξαρτήτως εταιρίας ή μοντέλο. Το MIDI Πρωτόκολλο είναι κοινό σε όλα τα όργανα.

Το MIDI είναι μια ψηφιακή διασύνδεση {digital interface} ή σύνδεση {connection}. Αυτή η διασύνδεση στο σύνολό της αποτελείται από μία συσκευή από μηχανικό εξοπλισμό {Hardware} και από λογισμικό {Software} το οποίο μεταφέρει τις πληροφορίες μεταξύ δύο τερματικών. Η MIDI διασύνδεση είναι ένα διπλής κατεύθυνσης σειριακός ασυγχρόνιστος σύνδεσμος {bidirectional serial asynchronous link} που μεταφέρει τα δεδομένα {data} σε ρυθμό των 31,250 bits ανά δευτερόλεπτο {bits per second ή bps}.

Με τον όρο ‘διπλή κατεύθυνση’ εννοούμε πως όταν συνδέουμε δύο συνθεσάιζερ χρησιμοποιούμε δύο ξεχωριστά καλώδια. Έτσι το κάθε όργανο μπορεί να στέλνει και να λαμβάνει πληροφορίες το ένα από το άλλο. Η μετάδοση αυτών των πληροφοριών γίνεται με τη χρήση bit. Επιπλέον, η έννοια της σειριακής διασύνδεσης δηλώνει ότι τα bit μεταδίδονται το ένα μετά το άλλο, μέσα από ένα και μοναδικό καλώδιο. Σε ρυθμό των 31,250 bps, χρειάζονται 320 κλάσματα του δευτερολέπτου {microseconds} για τη μετάδοση ενός byte (ή 10 bit) ή με άλλα λόγια μπορούμε να στείλουμε περισσότερα από 3000 byte το δευτερόλεπτο.

2.6 Οι Θύρες και οι Τρόποι Σύνδεσής τους

Το MIDI είναι φτιαγμένο έτσι ώστε να επιτρέπει την κυκλοφορία των πληροφοριών και στις δύο αντίστροφες κατευθύνσεις, μεταξύ δύο οργάνων και, αν χρειαστεί, να περάσει τις ίδιες πληροφορίες σε απεριόριστο αριθμό συνθεσάιζερ. Οι MIDI θύρες για τη MIDI διασύνδεση {MIDI Interface Ports} υπάρχουν στο πίσω μέρος όλων των MIDI οργάνων. Κάθε συνθεσάιζερ ή/και υπολογιστής μπορεί να έχει κάποια από τις παρακάτω τυποποιημένες MIDI θύρες :

1) Θύρα MIDI IN: Εκεί λαμβάνονται τα δεδομένα που θα επεξεργαστεί το συγκεκριμένο μηχάνημα. Οι εισερχόμενες πληροφορίες λαμβάνονται από τη θύρα MIDI IN και έπειτα στέλνονται στον επεξεργαστή του οργάνου. Εκεί αναλύονται και επεξεργάζονται, δίνοντας το ίδιο αποτέλεσμα σαν η εκτέλεση και η επεξεργασία να γινόταν πάνω στο ίδιο αυτό όργανο.

2) Θύρα MIDI OUT: Εκεί αποστέλλονται τα δεδομένα που θα στείλει το συγκεκριμένο μηχάνημα . Το MIDI δεν μεταφέρει ήχους μέσω των καλωδίων, αλλά ψηφιακού κώδικες, οι οποίοι αναπαριστούν το τι παίζεται στο όργανο. Αυτές οι πληροφορίες στέλνονται από τη θύρα MIDI OUT του οργάνου που εκτελεί το ρόλο του Ελεγκτή {Master}, σε ένα άλλο συνθεσάιζερ που εκτελεί το ρόλο του Αποδέκτη {Slave} της εντολής το οποίο αναπαράγει την εκτέλεση.

3) Θύρα MIDI THRU: Έχει νόημα μόνο για σύνδεση synthesizers μεταξύ τους . Επιτρέπει στα εισερχόμενα δεδομένα να περάσουν σε κάποιο άλλο όργανο. Όποια πληροφορία φτάνει στη θύρα MIDI IN ενός οργάνου μπορεί μέσω της θύρας MIDI THRU να διοχετευθεί σε ένα άλλο όργανο. Στην ουσία αυτή η θύρα επαναλαμβάνει τις πληροφορίες που έφτασαν στη συσκευή προκειμένου αυτές να είναι διαθέσιμες για αποστολή σε κάποια άλλη συσκευή MIDI. Αυτό μας επιτρέπει μέσω ενός οργάνου να μπορούμε να ελέγχουμε δύο ή παραπάνω όργανα. Είναι σημαντικό να καταλάβουμε ότι, ότι εκτελείται σε ένα κλαβιέ βγαίνει από το MIDI OUT και όχι από το MIDI THRU. Όταν οι υπολογιστές συνδέονται μεταξύ τους για να μοιραστούν πληροφορίες, δημιουργούν ένα δίκτυο. Το MIDI είναι ένα δίκτυο για μουσικά όργανα.

Για να χρησιμοποιήσουμε υπολογιστή για σύνδεση με synthesizers πρέπει να διαθέτουμε το κατάλληλο υλικό (hardware) που να μας παρέχει στον υπολογιστή midi ports (τουλάχιστον την midi-in και την midi-out θύρα για σύνδεση με ένα synthesizer), αλλά, φυσικά, και το κατάλληλο λογισμικό που θα εκμεταλλευτεί την ύπαρξη του MIDI πρωτοκόλλου και θα έχει ενεργή αλληλεπίδραση με τα synthesizers με τα οποία είναι συνδεδεμένο μέσω MIDI.

Εννοείται ότι το πρωτόκολλο MIDI δεν ασχολείται απλά με την αποστολή των νοτών που πρέπει να παιχτούν και των παραμέτρων τους, αλλά υλοποιεί με διάφορες ειδικές τεχνικές και κάθε δυνατότητα που διαθέτουν τα σημερινά synthesizers. Κλασικό παράδειγμα είναι τα μηνύματα που στέλνονται όταν ο χειριστής του synthesizer χρησιμοποιεί τον τροχό αλλαγής συχνότητας (Pitch Bend Wheel) του synthesizer (προκαλεί ένα εφφέ που αλλάζει ελαφρά την συχνότητα της νότας - θυμίζει αυτό που κάνουν συχνά δεξιοτέχνες σαξοφωνίστες), τα μηνύματα που στέλνουν ροή "αποκλειστικής πληροφορίας" (system exclusive) προς κάποιο synthesizer (το πώς θα χρησιμοποιηθεί αυτή η πληροφορία είναι εντελώς διαφορετικό πράγμα σε κάθε μοντέλο synthesizer) καθώς και πολλά άλλα παρεμφερή μηνύματα ειδικών χρήσεων. Η ροή

βέβαια είναι πάντα δύο δρόμων : είτε κάποιο μηχάνημα στέλνει (μέσω του midi-out), είτε κάποιο μηχάνημα λαμβάνει (μέσω του midi-in).

Μέσω αυτών των θυρών γίνεται δυνατή η σύνδεση οργάνων μεταξύ τους με ποικίλους τρόπους. Αυτοί είναι:

• Σύνδεση μονής κατεύθυνσης {one way connection}: Η θύρα MIDI OUT του οργάνου Α, που ονομάζεται Ελεγκτής {Master}, μεταδίδει τα δεδομένα στο MIDI IN του οργάνου Β, που ονομάζεται Αποδέκτης {Slave}. Η αντίστροφη σύνδεση είναι επίσης πιθανή.

• Σύνδεση διπλής κατεύθυνσης {bidirectional connection} ή σύνδεση αμφίδρομης

επικοινωνίας {handshake connection}: Το MIDI OUT του οργάνου Α συνδέεται με το MIDI IN του οργάνου Β και αντίστροφα. Με αυτό τον τρόπο τα MIDI bytes μπορούν να ταξιδέψουν και στις δύο κατευθύνσεις.

• Σύνδεση τριών MIDI οργάνων: Το όργανο Α ενεργοποιεί το όργανο Β και το όργανο Β ενεργοποιεί το όργανο Γ, αλλά το Α δεν ενεργοποιεί το Γ. Το όργανο Γ θα μπορούσε να ενεργοποιεί το Α με μία ακόμη σύνδεση.

• Σύνδεση Daisy Chain (που στην κυριολεξία σημαίνει «γιρλάντα από μαργαρίτες»):

Τα δεδομένα που στέλνονται από το όργανο Α στο Β αντιγράφονται και στέλνονται αμέσως στο όργανο Γ, το οποίο με τη σειρά του τα μεταφέρει στο όργανο Δ, και πάει λέγοντας. (Το MIDI σήμα έχει μία έντονη τάση να διακόπτεται όταν περνάει από ένα όργανο σε ένα άλλο. Για αυτό συνιστάται να μη υπάρχουν πάνω από 4 όργανα σε μία σύνδεση daisy chain).

2.7 Standard MIDI files

Ένα Standard MIDI FILE μπορεί, εκτός της MIDI πληροφόρησης, να περιέχει και άλλες πληροφορίες (Meta Events) σχετικές με το όνομα της μουσικής ακολουθίας, το μουσικό μέτρο, τον οπλισμό, τα ονόματα των Tracks, τους δείκτες, τους στίχους κ.α. Η ελευθερία που παρέχει το πρωτόκολλο MIDI στον τρόπο αποθήκευσής τους, δημιουργεί συχνά προβλήματα στην αναγνώριση και σωστή εισαγωγή τους στα διαφορετικά προγράμματα. Όσο περνά ο καιρός και η επικοινωνία μεταξύ των κατασκευαστών γίνεται καλύτερη και αυτά τα προβλήματα αναμένεται να ξεπεραστούν. Υπό μορφή MIDI αρχείου μπορούμε να αποθηκεύσουμε και Tempo Maps που περιγράφουν τις αλλαγές Tempo μίας μουσικής ακολουθίας. Τα MIDI αρχεία που περιέχουν Tempo Maps χρησιμοποιούνται συχνότατα για το συγχρονισμό ήχου και εικόνας σε κινηματογραφικές ή βίντεο εφαρμογές.

Υπάρχουν τριών ειδών MIDI αρχεία. Το πρώτο ονομάζεται MIDI FILE Type 0 και κατά τη μετατροπή τοποθετεί όλη τη πληροφόρηση σε ένα Track, χωρίς να αλλάζει το MIDI κανάλι στο οποίο απευθύνεται κάθε γεγονός. Μπορεί αυτό το είδος αρχείου να περιορίζει την αναλυτική επεξεργασία της πληροφόρησης, είναι όμως χρήσιμο για την αναπαραγωγή ακολουθιών με τη χρήση φτηνών MIDI περιφερειακών που χρησιμοποιούνται σε εκπαιδευτικά ιδρύματα ή ζωντανές εμφανίσεις.

Το δεύτερο είδος MIDI αρχείου ονομάζεται MIDI FILE Type 1 και τοποθετεί την πληροφόρηση σε περισσότερα από ένα Tracks, κάθε ένα από τα οποία στέλνει την πληροφόρησή του σε ένα ή περισσότερα MIDI κανάλια. Η μετατροπή μίας μουσικής ακολουθίας σε αυτό το είδος αρχείου συνίσταται στην περίπτωση που θέλουμε να το μεταφέρουμε για επεξεργασία σε κάποιο άλλο πρόγραμμα.

Το τρίτο είδος MIDI αρχείου ονομάζεται MIDI FILE Type 2 και συναντάται εξαιρετικά σπάνια. Το αρχείο αυτό, όπως και το προηγούμενο, περιέχει ξεχωριστά Tracks κάθε ένα από τα οποία μπορεί να μεταδώσει την πληροφόρησή του σε ένα ή περισσότερα MIDI κανάλια. Σε αυτό το

είδος MIDI αρχείου κάθε Track μπορεί να αντιπροσωπεύει ένα Pattern ή μία υποακολουθία με διαφορετικό χρόνο έναρξης, MIDI κανάλια ή Tempo Map.

Οι προδιαγραφές του πρωτοκόλλου MIDI προβλέπουν τη μεταφορά αρχείων MIDI μεταξύ διαφορετικών υπολογιστών. Αν και ένας Macintosh δεν μπορεί χωρίς το κατάλληλο πρόγραμμα να διαβάσει μία δισκέτα που έχει εγκαινιαστεί σε ένα PC, εν τούτοις ένα MIDI αρχείο που έχει δημιουργηθεί στον Macintosh, μπορεί να μεταφερθεί σε ένα PC με τη χρήση ενός σειριακού καλωδίου ή ενός Modem.

Μία πρόσφατη προσθήκη στο πρωτόκολλο επικοινωνίας MIDI απλοποιεί τη μεταφορά MIDI αρχείων μεταξύ διαφορετικών υπολογιστών, καθιστώντας δυνατή τη μεταφορά τους μέσω MIDI καλωδίων.

Όπως για το πρωτόκολλο MIDI έτσι και για τα αρχεία MIDI επινοήθηκαν με τα χρόνια πολλές διαφορετικές χρήσεις. Ενώ το αρχικό σκεπτικό προέβλεπε την ανταλλαγή αρχείων μεταξύ των προγραμμάτων sequencing, στη συνέχεια αξιοποιήθηκε και σε άλλα προγράμματα. Τα προγράμματα αλγοριθμικής σύνθεσης για παράδειγμα, μπορούν να αποθηκεύσουν τα αποτελέσματα της επεξεργασίας τους σε μορφή MIDI αρχείου, το οποίο στη συνέχεια μπορεί να εισαχθεί σε ένα sequencing για μελέτη και περαιτέρω επεξεργασία και στη συνέχεια σε πρόγραμμα μουσικής σημειογραφίας για εκτύπωση. Τέλος, προγράμματα σχεδιασμένα για Audio Post Production ή προγράμματα που συνδυάζουν Audio και MIDI είναι σε θέση να εισάγουν ή να εξάγουν MIDI αρχεία.

Το πλήθος των εφαρμογών του πρωτοκόλλου MIDI βρήκε το δρόμο του και στη μουσική εκπαίδευση. Τα MIDI αρχεία επέτρεψαν την επικοινωνία μουσικών αρχείων ανεξαρτήτως προγράμματος ή υπολογιστή. Αυτό προέτρεψε πολλές εταιρίες να στραφούν στον τομέα της εκπαίδευσης. Η δημιουργία μουσικού εκπαιδευτικού υλικού με σκοπό τη μελέτη και εμβάθυνση σε θέματα θεωρίας, αρμονίας και μουσικής εκτέλεσης, ξεπέρασε κάθε προσδοκία. Σήμερα την αγορά κατακλύζουν MIDI αρχεία για κάθε ανάγκη και κάθε γούστο. Ανάλογα με τις ανάγκες σας, μπορείτε να επιλέξετε εκείνο που θέλετε να εισάγετε στο πρόγραμμά σας, να το ακούσετε, να το μελετήσετε, να το επεξεργαστείτε ή να το τυπώσετε. Τα αρχεία MIDI σε συνδυασμό με τις προδιαγραφές General MIDI έθεσαν, από πλευράς μουσικής, τις βάσεις για την αλματώδη ανάπτυξη των πολυμέσων (Multimedia).

3. Ψηφιακή ηχογράφηση αρχείου ήχου

Ψηφιακή ηχογράφηση αρχείου ήχου ονομάζουμε την αποθήκευση ηχητικού κύματος σε ψηφιακή μνήμη, συνήθως σε σκληρό δίσκο. Η λειτουργία της τεχνολογίας αυτή βασίζεται στην μετατροπή του ηχητικού κύματος σε ψηφιακή πληροφορία μέσα από μια διαδικασία δειγματοληψίας και στην συνέχεια αποθήκευση των πληροφοριών αυτών υπό μορφή αρχείων ήχου. Η μετατροπή αυτή στον ηλεκτρονικό υπολογιστή πραγματοποιείται από την κάρτα ήχου του, η οποία περιέχει το μετατροπέα Analog-to-Digital (A/D converter).

Η ποιότητα μιας ψηφιακής ηχογράφησης εξαρτάται κυρίως από τη συχνότητα δειγματοληψίας και την ανάλυση των δειγμάτων. Το πιο διαδεδομένο μέσο εγγραφής ψηφιακού ήχου, το CD, είναι γραμμένο με συχνότητα δειγματοληψίας 44,1KHz και ανάλυση 16bit. Τα σύγχρονα συστήματα ηχογράφησης με ηλεκτρονικό υπολογιστή προσφέρουν ακόμα καλύτερη ποιότητα στην ψηφιοποίηση του ηχητικού κύματος. Έτσι ένα σύγχρονο ψηφιακό σύστημα μπορεί να προσφέρει επιλογή ηχογράφησης μέχρι και 192KHz / 24bit.

4. Θεωρία του ήχου

4.1 Εισαγωγή

Ο ήχος είναι από τα πιο εντυπωσιακά στοιχεία των πολυμεσικών εφαρμογών με την έννοια ότι έχει να πει κάτι σε κάθε γλώσσα, σε κάθε ακροατήριο, με δυνατότητες έκφρασης από τον ψίθυρο ως την κραυγή. Μπορεί να παράσχει ακουστική απόλαυση ως μουσική φόρμα, μπορεί να εντυπωσιάσει ως ηχητικό εφφέ, μπορεί να ξεκουράσει ως υπόκρουση.

Η ιδιαιτερότητα κάθε ήχου οφείλεται σε ένα σύνολο από ηχητικά μεγέθη, άλλα από τα οποία είναι ανεξάρτητα από την προσωπική αντίληψη του ακροατή και αποκαλούνται αντικείμενα, ενώ άλλα είναι άμεσα συνδεδεμένα με τον ακροατή και λέγονται υποκειμενικά χαρακτηριστικά του ήχου. Τέτοια αντικειμενικά χαρακτηριστικά, είναι η συχνότητα και η ένταση και ως υποκειμενικά η ακουστικότητα, το ύψος και η χροιά.

Συχνότητα: 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 υψηλό ήχο από την τούμπα. Το ύψος του ήχου είναι ένα υποκειμενικό γνώρισμα που σχετίζεται αμοιβαία με ένα αντικειμενικό χαρακτηριστικό, τη συχνότητα. Η συχνότητα έχει να κάνει με το πόσο γρήγορα ή αργά πάλλεται το σώμα που δημιουργεί τον ήχο, για παράδειγμα, όταν η χορδή μιας κιθάρας πάλλεται 100 φορές το δευτερόλεπτο, τότε προκαλείται ο αντίστοιχος αριθμός πυκνωμάτων ούτως ώστε, να γίνεται αντιληπτός ο συγκεκριμένος ήχος. Συμπερασματικά, η συχνότητα ορίζει τον αριθμό τον κύκλων που εκτελεί η συνάρτηση ανά δευτερόλεπτο και μετριέται σε κύκλους ανά δευτερόλεπτο ή Hertz (Hz).

Κάθε ήχος για να γίνει αντιληπτός από τι ανθρώπινο αυτί, θα πρέπει να έχει συχνότητα μεταξύ 20 και 20.000Hz. Οι πιο χρήσιμες συχνότητες βρίσκονται κάτω από 10 kHz, (π.χ ομιλία, μουσική, διάφοροι θόρυβοι). Εντούτοις, υπάρχουν ήχοι με συχνότητα μεγαλύτερη των 20 kHz οι οποίοι ονομάζονται και υπέρηχοι, ενώ αυτοί με μικρότερη συχνότητα των 20 Hz λέγονται υπόηχοι. Αν και οι δύο κατηγορίες δεν έχουν επίδραση από το ανθρώπινο αυτί, μπορεί να έχουν επίδραση στη υγεία του.

Ένταση: Το δεύτερο βασικό γνώρισμα του ήχου είναι η ένταση, στενά συνδεδεμένη με την ισχύει του ηχητικού σήματος που διεγείρει το αυτί μας. Η κύρια αιτία που οι ήχοι έχουν διαφορετικές εντάσεις είναι ότι πιέζουν με διαφορετική δύναμη το τύμπανο του αυτιού μας,

δηλαδή το πόσο έντονες είναι οι αναταράξεις που προκαλεί το σώμα που παράγει τον ήχο. Όσο πιο μεγάλη ισχύ διαμορφώνουν τα ηχητικά κύματα ολοένα και περισσότερη δύναμη εξασκούν στο μηχανισμό του αυτιού μας. Για παράδειγμα, κτυπώντας απλά την χορδή μιας κιθάρας τότε αυτή πάλλετε με μια συχνότητα, δημιουργώντας αναταράξεις που όμως δεν είναι έντονες. Αντίθετα αν κτυπήσουμε την ίσια χορδή με δύναμη παρατηρείται πως πάλλεται με την ίδια συχνότητα αλλά με πολύ εντονότερες διαταραχές. Άρα, ένταση ορίζεται από το πλάτος της δόνησης με αποτέλεσμα όσο μεγαλύτερο είναι το πλάτος τόσο ισχυρότερος είναι ο ήχος.

Στην φυσική, ως ένταση ορίζεται το ποσό της ηχητικής ενέργειας στην μονάδα του χρόνου και εκφράζεται σε Watt/m^2. Οι τιμές των ηχητικών εντάσεων που γίνονται αντιληπτές από τον άνθρωπο καλύπτουν την περιοχή από 10-12 έως 10 Watt/m^2. Σύμφωνα με αυτό παρουσιάζει η χρήση της γραμμικής κλίμακας στην περιγραφή της ηχητικής έντασης, ενώ ο τρόπος που γίνονται συνειδητές οι ακουστικές εντάσεις είναι λογαριθμικός και όχι γραμμικός (δηλ. η διαφορά μεταξύ των υποκειμενικών εντάσεων δύο ήχων δεν εξαρτάται από την απόλυτη τιμή της διαφοράς αυτής αλλά από το λόγο των ισχύων του). Αυτό σημαίνει ότι η αύξηση της έντασης από 2μ Watt/m^2 σε 4 Watt/m^2, αντιστοιχεί όπως και στην περίπτωση 5μ Watt/m_2 σε 10μ Watt/m^2.

Κατά συνέπεια, για τους πιο πάνω λόγους, η σχετική τιμή μεταξύ 2 ηχητικών κυμάτων δεν μετριέται σε Watt/m^2 αλλά σε bels ή πιο συχνά σε decibels(dB=μονάδα μέτρηση της έντασης ενός ήχου και 1dB αντιστοιχεί σε 0,1 bel). Για να συγκρίνουμε το την ισχύ δύο ηχητικών κυμάτων υπολογίζουμε το λόγο των ισχύων τους. Η λογαριθμική κλίμακα dB παρουσιάζει της σχέση της ηχητικής ισχύος με μια συγκεκριμένη πηγή, σε σχέση με την χαμηλότερη ηχητική ισχύ που μπορεί να γίνει αντιληπτή από το ανθρώπινο αυτί (10^{-12} Watt/m^2).

Ένταση ήχου=10 log(P/Po) dB[1]

Η τιμή P= τιμή μέτρησης

Η τιμή Po=τιμή αναφοράς σε μονάδες Watt/m^2

Παράδειγμα: Για P=10^{-12}Watt/m^2 έχουμε

Ένταση =10 log(10-12/10-12)=10 log 1= **0 db**

Δηλαδή, ο ήχος έντασης 0 db αντιστοιχεί στο κατώφλι ακουστικότητας ενώ ο ήχος έντασης 140 db αντιστοιχεί στο όριο πόνου. Συμπερασματικά, μικρότερη ένταση σε db αντιπροσωπεύει μεγάλη αύξηση ηχητικής ισχύος ενώ θεωρητικά, η αύξηση κατά 3 db διαμορφώνει διπλασιασμός της ηχητικής ισχύος. Έτσι, όταν ένας ήχος είναι 10 φορές πιο ισχυρός από την σχεδόν απόλυτη ησυχία έχει ένταση 10db ,100 φορές πιο ισχυρός έχει ένταση 20db και 1000 φορές έχει ένταση 30dB

Ακουστικότητα: Η ακουστικότητα αποτελεί ένα από τα πιο υποκειμενικά γνωρίσματα του ήχου τα οποία κάθε ακροατής αντιλαμβάνεται με διαφορετικό τρόπο. Ήχος με ίδια ένταση θα έχει μικρότερη ακουστικότητα για ένα άτομο κε προβληματική ακοή. Με επακόλουθο η ακουστικότητα να έχει σχέση με την ένταση του ήχου. Σε σχέση με την ακουστικότητα οι ήχοι διακρίνονται σε ασθενείς με ισχυρούς, ενώ μονάδα μέτρησης της είναι το Phon, δηλαδή ήχος μόλις ακούγεται έχει ακουστικότητα 1 Phon ενώ με ακουστικότητα 130 Phon προκαλεί πόνο στο αυτί.

[1] Καλουμπιτσίδης Νικος, *Σήματα Συστήματα και Αλγόριθμοι,* εκδ. Δίαυλος , Αθήνα1994 (5η έκδοση).

Ύψος: Ένα άλλο υποκειμενικό χαρακτηριστικό αποτελεί το ύψος το οποίο συνδέεται άμεσα με τη συχνότητα το ήχου. Η σχέση αυτή διακρίνει τους ήχους σε δύο μορφές, πρώτον σε οξείς με μεγάλη συχνότητα, όπως τους ήχους ενός βιολιού και κατά δεύτερον σε βαρείς με μικρή συχνότητα, όπως τους ήχους που παράγει το μπάσο τύμπανο. Και στις δύο περιπτώσεις η συχνότητα εξαρτάται από το πάχος και το μήκος της χορδής, με αποτέλεσμα όσο πιο μεγάλη σε μήκος και διάμετρο είναι η χορδή ,τόσο μικρότερη συχνότητα παράγεται, ενώ όσο πιο πολύ μειώνεται το πάχος και το μήκος περισσότερο αυξάνεται η συχνότητα .

Χροιά: Ως τρίτο βασικό υποκειμενικό γνώρισμα είναι η χροιά, χάρη στην οποία ο ήχος ξεχωρίζει ακόμη και αν τα υπόλοιπα χαρακτηριστικά του είναι τα ίδια. Η χροιά αναφέρεται στους σύνθετους ήχους και εξαρτάται από τους απλούς ήχους που αποτελούν το σύνθετο. Κάθε περιοδικός ήχος μπορεί να αναλυθεί σε ένα άθροισμα συνημιτονικών συναρτήσεων με διάφορα πλάτη και φάσεις και με συχνότητες ακέραια πολλαπλάσιες μιας θεμελιώδους συχνότητας (fundamental frequency). Οι συχνότητες αυτές αποκαλούνται αρμονικές (harmonic frequency) 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 και το μεγαλύτερο πλάτος καθορίζει και το ύψος του ήχου.

4.2 Η ακουστική του ήχου

Ακουστική λέγεται η επιστήμη που ασχολείται με τον ήχο.

Στην σύντομη εισαγωγή στην ακουστική και στη θεωρία της μουσικής που επιχειρούμε ρίχνουμε φως στον κόσμο της μουσικής και στα φυσικά του χαρακτηριστικά. Θα μιλήσουμε επίσης ακόμα και για τις τεχνικές δειγματοληψίας.

4.3 Τι είναι ο ήχος;

Μια απλή απάντηση στο ερώτημα αυτό είναι η εξής: ήχος είναι κάθε τι που ακούμε. Ο ήχος παράγεται από μια πηγή και συλλαμβάνεται από το αυτί μας. Όμως για να φτάσει ο ήχος από την πηγή στο δέκτη, πρέπει να μεσολαβήσει ένα φέρον μέσο. Αυτό συνήθως είναι ο αέρας, αλλά μέσο διάδοσης του ήχου είναι και το νερό ή κάποιο στερεό σώμα. Χωρίς μέσο διάδοσης δεν είναι δυνατή η μεταφορά του ήχου. Έτσι για παράδειγμα δεν είναι δυνατή η συνομιλία στο φεγγάρι. Επίσης, το μέσο διάδοσης καθορίζει και τις ιδιότητες του ήχου. Αλλιώς π.χ. ακούγονται οι ήχοι στο νερό, αλλιώς φιλτράρονται και με άλλες ταχύτητες μεταδίδονται.

Εμάς θα μας απασχολήσουν οι ήχοι που συναντάμε στην καθημερινή μας ζωή, αυτοί που διαδίδονται στον αέρα. Όταν ο αέρας πάλλεται, δημιουργούνται κύματα. Αυτές οι κυμάνσεις συλλαμβάνονται ως ήχος. Το πλάτος και η συχνότητα τους διαμορφώνουν το ποιόν του ήχου που ακούγεται.

Άρα ο ήχος είναι ενέργεια.

Ο απλούστερος τύπος κύμανσης είναι το ημιτονικό σήμα. Αυτό αντιστοιχεί σε μία μόνο συχνότητα, και περισσότερο αποτελεί ιδανική σύλληψη παρά πραγματική κατάσταση.

Το ημιτονικό κύμα είναι περιοδικό σήμα. Αυτό σημαίνει πως ο πρώτος παλμός συνοδεύεται από πολλούς ίδιους παλμούς. Τα περιοδικά κύματα δημιουργούν ήχους που λέγονται τόνοι, όπως είναι οι τόνοι που παράγει μια κιθάρα, ένα πιάνο ή ένα διαπασών. Στην πράξη αυτοί οι ήχοι που παράγονται δεν περιέχουν μόνο μια συχνότητα, αλλά και μερικές ακόμα παραπλήσιες συχνότητες, αλλά μπορούμε να θεωρήσουμε με ικανοποιητική προσέγγιση πως έχουμε πράγματι παραγωγή τόνων.

Ο ήχος όμως των κυμάτων που με δύναμη πέφτουν στην ακρογιαλιά θεωρείται ως θόρυβος, παρά ως τόνος ή τόνοι, γιατί αποτελείται από πληθώρα μη-περιοδικών κυμάνσεων.

Το βασικό σημείο της κύμανσης, είναι η έντασή της, δηλαδή το πλάτος της. Όσο μεγαλύτερο το πλάτος, τόσο ισχυρότερα ακούγεται ο ήχος. Φυσική μονάδα μέτρησης του ήχου είναι το decibel (dB). Η κλίμακα μέτρησης των dB είναι λογαριθμική, γιατί το ανθρώπινο αυτί συλλαμβάνει την ένταση του ήχου λογαριθμικά, δηλαδή μπορεί και ακούει ήχους που ποικίλουν σημαντικά σε ένταση, το ίδιο καλά. Δε θα υπεισέλθουμε στις λεπτομέρειες των λογαριθμικών κλιμάκων. Απλώς θα πούμε ότι η κλίμακα αυτή είναι συγκριτική και σήματα που διαφέρουν 3 dB έχουν διπλάσια ένταση το ισχυρότερο σε σχέση με το ασθενέστερο. Επίσης, για τη μέτρηση ισχύος κάθε σήματος λαμβάνεται συγκριτική στάθμη βαθμονόμησης τα 10 ή 12 Watt που αντιστοιχεί σε ηχητικό σήμα μη-αντιληπτό από το αυτί μας. Ένας άλλος λόγος χρήσης λογαριθμικής κλίμακας είναι το γεγονός ότι το ανθρώπινο αυτί αντιλαμβάνεται τεράστιο εύρος έντασης ηχητικών σημάτων και μόνο με τη χρήση λογαριθμικής κλίμακας μπορούμε να έχουμε συνοπτική εικόνα των εντάσεων.

Ο ήχος, όπως τον αντιλαμβανόμαστε καθημερινά, δεν έχει ως άμεσα αντιληπτό μέγεθος μόνο την ένταση. Κάτι άλλο που αντιλαμβανόμαστε ευθύς αμέσως είναι η συχνοτική συμπεριφορά του σήματος που μας επιτρέπει να κατατάξουμε τις φωνές σε μπάσες ή οξείες και τους ήχους σε στριγγλούς ή μουντούς.

Το δεύτερο βασικό λοιπόν γνώρισμά του ηχητικού σήματος είναι η συχνότητά του.

Ο αριθμός των παλμικών δονήσεων το δευτερόλεπτο είναι η συχνότητα του σήματος και μετριέται σε Hertz (Hz).

Για τον μουσικό κόσμο η συχνότητα των 440 Hz είναι πολύ σημαντική. Η συχνότητα αυτή αντιστοιχεί στη νότα Λα1 (A1). Τα περισσότερα μουσικά όργανα κουρδίζονται με βάση αυτή τη συχνότητα.

Το ανθρώπινο αυτί μπορεί να αντιληφθεί ήχους από 20 Hz ως 20 KHz. Αυτό το συχνοτικό εύρος διαφέρει από άτομο σε άτομο, ειδικά στο πάνω κατώφλι του. Επίσης οι συνεχείς θόρυβοι που βομβίζουν τα αυτιά μας στις πόλεις και η πάροδος του χρόνου αμβλύνουν ακόμα περισσότερο το συχνοτικό εύρος της αντίληψης των ήχων από τα αυτιά μας.

Πάντως οι πιο χρήσιμες συχνότητες βρίσκονται κάτω από τα 10 KHz. Εκεί εντοπίζονται οι συχνότητες που έχουν να κάνουν με την ομιλία, την μουσική ακόμα και με τους διάφορους θορύβους.

Ένας τόνος με συχνότητα 220 Hz και ένας με συχνότητα 880 Hz είναι τόνοι που αντιστοιχούν στη νότα Λα. Αν και αντιστοιχούν στη νότα Λα και οι δύο, εντούτοις ο ένας είναι μια οκτάβα κάτω από την Λα1 και ο άλλος μία οκτάβα παραπάνω.

Αυτή η διάταξη των νότων δείχνει μια βασική μουσική αρχή: κάθε διπλασιαζόμενη συχνότητα αντιστοιχεί στην ίδια ακριβώς νότα, μόνο που αυτή είναι μια οκτάβα υψηλότερη. Συνεπώς ο λόγος συχνοτήτων των βασικών νότων (νότα Ντο) δύο διαδοχικών οκτάβων είναι 1:2.

Μερικές χαρακτηριστικές σχέσεις μεταξύ νότων και οι λόγοι των συχνοτήτων τους, δηλαδή τα διάφορα μουσικά διαστήματα, αναγράφονται στον παρακάτω πίνακα:

Διαστήματα	Νότες	παράδειγμα
Βασικές νότες	1:1	Ντο – ντο
Ελάχιστη (μινόρε) τρίτης	5:6	Ντο – μι δίεση
Μέγιστη (ματζόρε) τρίτης	4:5	Ντο – μι

Τετάρτης	3:4	Ντο – φα
Πέμπτης	2:3	Ντο – σολ
Ελάχιστη (μινόρε) έκτης	5:8	Ντο – λα ύφεση
Μέγιστη (ματζόρε) έκτης	3:5	Ντο – λα
Οκτάβα	1:2	Ντο – ντο’

Οι σύγχρονες μουσικές κλίμακες αποτελούνται από επτά βασικούς τόνους και πέντε ημιτόνια. Οι επτά βασικοί τόνοι σχηματίζουν ή ολόκληρα διαστήματα (ντο-ρε, ρε-μι, φα-σολ, σολ-λα, λα-σι) ή ημιτόνια (μι-φα, σι-ντο).

Κάθε μια από τις νότες αυτές μπορεί να ανέλθει συχνοτικά και να σχηματιστεί η ίδια νότα σε δίεση (#) ή να κατέλθει και να σχηματιστεί η ίδια νότα σε ύφεση (b). Στο πεντάγραμμο πρώτα είναι η νότα σε ύφεση, μετά η ίδια η νότα και μετά η δίεσή της. Θεωρητικά, μια νότα σε δίεση και η επόμενή της σε ύφεση δεν είναι το ίδιο πράγμα. Δηλαδή το Ρε δίεση (D#) και το Μι ύφεση (Eb) έχουν παραπλήσια συχνότητα, αλλά όχι ακριβώς την ίδια.

Για ευκολία στην μουσική εκτέλεση (ειδικά στα όργανα με πλήκτρα, όπως το πιάνο) χρησιμοποιούνται οι συγκερασμένες κλίμακες, όπου η δίεση μιας νότας και η ύφεση της επόμενής της αποδίδονται στην ίδια συχνότητα και συνεπώς χρειάζεται ένα πλήκτρο γι’ αυτές. Έτσι, τα όργανα αυτά έχουν 12 νότες ανά οκτάβα. Η απώλεια αυτού του συγκερασμού δεν είναι μεγάλη γιατί ακουστικά αυτός ο συμβιβασμός δεν είναι εύκολα αντιληπτός.

Σε μερικά όργανα όπως το βιολί που δεν έχει τάστα, μπορεί κανείς να αποδώσει για παράδειγμα το Ρε δίεση (D#) και το Μι ύφεση (Eb).

Όλα αυτά που είπαμε τα βλέπουμε συγκεντρωμένα στο σχήμα που φαίνεται η συγκερασμένη κλίμακα. Για να βρούμε την ακριβή συχνότητα κάθε νότας αρκεί να πολλαπλασιάσουμε τη συχνότητα της θεμελιώδους της οκτάβας (Ντο-C) με το νούμερο που αναγράφεται πάνω από κάθε νότα.

Για παράδειγμα το σολ δίεση έχει συχνότητα:

1.58740 * τη συχνότητα του ντο.

Όλα όσα αναφέρθηκαν μέχρι στιγμής είναι αρκετά για να χειρίζεται κανείς με άνεση εργαλεία προγραμμάτων ήχου που στηρίζουν τη λειτουργία τους σε πεντάγραμμα ή πλήκτρα σαν του πιάνου.

4.4 Το ηχόχρωμα

Η συχνότητα δεν είναι το μόνο χαρακτηριστικό γνώρισμα κάθε ήχου. Για παράδειγμα, μπορούμε να παίξουμε τη νότα Λα1, που αντιστοιχεί στα 440 Hz, με κιθάρα και με φλάουτο. Το άκουσμα της ίδιας νότας είναι εντελώς διαφορετικό σε κάθε μουσικό όργανο. Αυτό οφείλετε στο γεγονός ότι εκτός από τον τόνο των 440 Hz, που είναι καθαρά ημιτονικό σήμα, παράγονται και διαφορετικές αρμονικές αυτής της συχνότητας, διαφορετικές για κάθε όργανο. Σ’ αυτές οφείλετε και το διαφορετικό ηχόχρωμα του κάθε οργάνου.

Έτσι όταν ένα όργανο παράγει των θεμελιώδη τόνο των 440 Hz, παράγονται με μικρότερη ένταση και οι τόνοι των 880 Hz , 1320 Hz, 1760 Hz ...

Παρόλο που οι αρμονικές χαρακτηρίζουν τελικά τον ήχο, η θεμελιώδης συχνότητα καθορίζει το όνομα της νότας που παίζουμε.

Το τι μορφή θα έχει η υπέρθεση ενός τόνου με μία αρμονική το βλέπουμε στο επόμενο σχήμα, όπου ένα ημιτονικό σήμα 100 Hz συνδυάζεται με ένα άλλο διπλάσιας συχνότητας αλλά με πλάτος το μισό του πρώτου.

Στα πιο πολλά διαγράμματα επεξεργασίας ήχου έχουμε τη δυνατότητα να δημιουργήσουμε ήχους συνδυάζοντας διάφορους τόνους. Συνήθως αυτή η διεργασία ονομάζεται FM σύνθεση και περιέχει και τη δυνατότητα για διαμόρφωση της συχνότητας συν τοις άλλοις. Από εκεί άλλωστε προέρχεται και το όνομά της (frequency modulation).

5. Ο ήχος και Υπολογιστές

5.1 Ψηφιακά Αρχεία Ήχου

Ο ήχος είναι μια διακύμανση πιέσεων και το μικρόφωνο αποτελεί το όργανο εκείνο που μετατρέπει ένα ηχητικό σήμα σε διακύμανση ηλεκτρονικής τάσης, αυτή η ηλεκτρονική τάση είναι το αναλογικό ηλεκτρονικό σήμα. Για να γίνει δυνατή η επεξεργασία του από τον Η/Υ απαραίτητη προϋπόθεση η ψηφιοποίηση του, μετατρέποντας το αναλογικό (analogy) σε ψηφιακό (digital) με την χρήση των μετατροπέων, ADC (Analog-to-Digital-Converters) και DAC (Digital-to-Analog-Converters DACs) και αντίστροφα[2]. Έτσι, η διαδικασία αυτή συμπεριλαμβάνει τρείς βασικές λειτουργίες: την «δειγματοληψία» του αρχικού σήματος, τον «κβαντισμό» των τιμών του σήματος διακριτών χρόνου που προκύπτει από την δειγματοληψία και τέλος η «κωδικοποίηση».

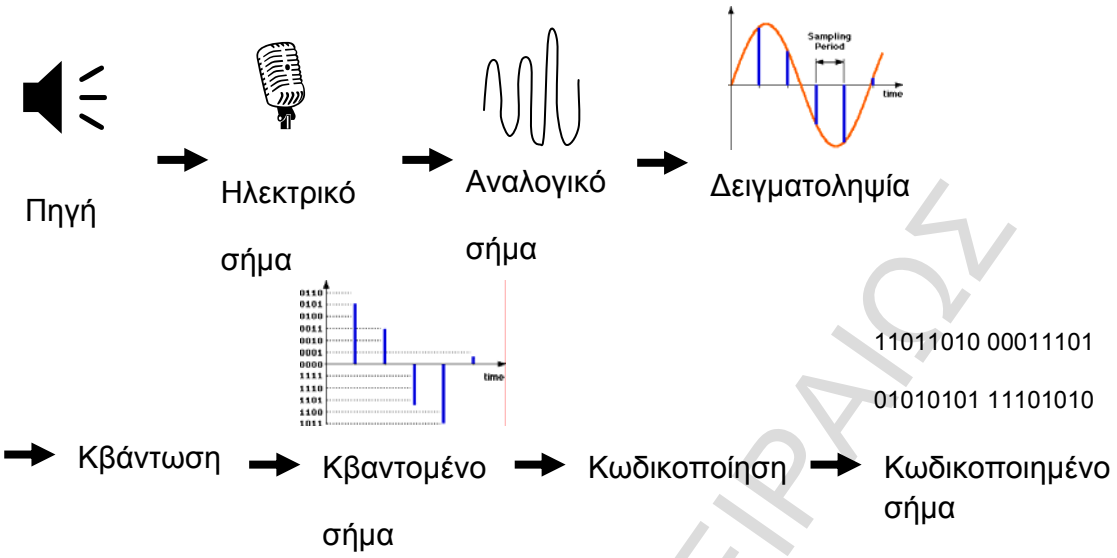
Πιο συγκεκριμένα, κατά την ***δειγματοληψία*** παίρνονται δείγματα του πλάτους του αναλογικού σήματος σε τακτά χρονικά διαστήματα με συνεπακόλουθο να δημιουργείται ένα στιγμιότυπο του σήματος της συγκεκριμένης τιμής, σύνολο το οποίου τα δείγματα διαμορφώνουν το ψηφιοποιημένο σήμα. Αντίθετα, στον ***κβαντισμο*** οι διαδοχικές τιμές της στάθμης (πλάτους) του σήματος διακριτού χρόνου sd(nTs) μετατρέπεται σε διακριτές (ψηφιακές) τιμές, πραγματοποιείται δηλαδή μια απεικόνιση της μορφής s(nTs) =Q{sd(nTs)} όπου Q{sd(nTs)}[3] είναι η κβαντισμένη τιμή. Κατά την παραπάνω μετατροπή το τελικό σήμα είναι διακριτό τόσο ως προς το χρόνο όσο και ως προς το πλάτος που καλείται ψηφιακό.

Όπως έχει ήδη τυπωθεί, το αναλογικό σήμα ακουστικής πίεσης μετατρέπεται σε ψηφιακό με την χρήση υποσυστημάτων εφαρμογής Α/Ψ (Analog-to-Digital-Converters) και το αντίστροφο, υλοποιείται με την χρήση Ψ/Α μετατροπών (Digital-to-Analog- Converters DACs) όπου οι κβαντισμένες τιμές μετατρέπονται σε αναλογικές[4]. Σε τέτοια υποσυστήματα οι διαδικασίες κβαντισμού και δειγματοληψίας πραγματοποιούνται ταυτόχρονα, ενώ η υλοποίηση των μετατροπέων και στις δύο φάσεις, παρουσιάζουν τόσο θεωρητικά όσο και πρακτικά προβλήματα, ενώ για την σταθερή μετατροπή καλό θα είναι να ικανοποιούνται συγκεκριμένες συνθήκες δειγματοληψίας και κβαντισμού στις οποίες αναφορά γίνεται πιο κάτω.

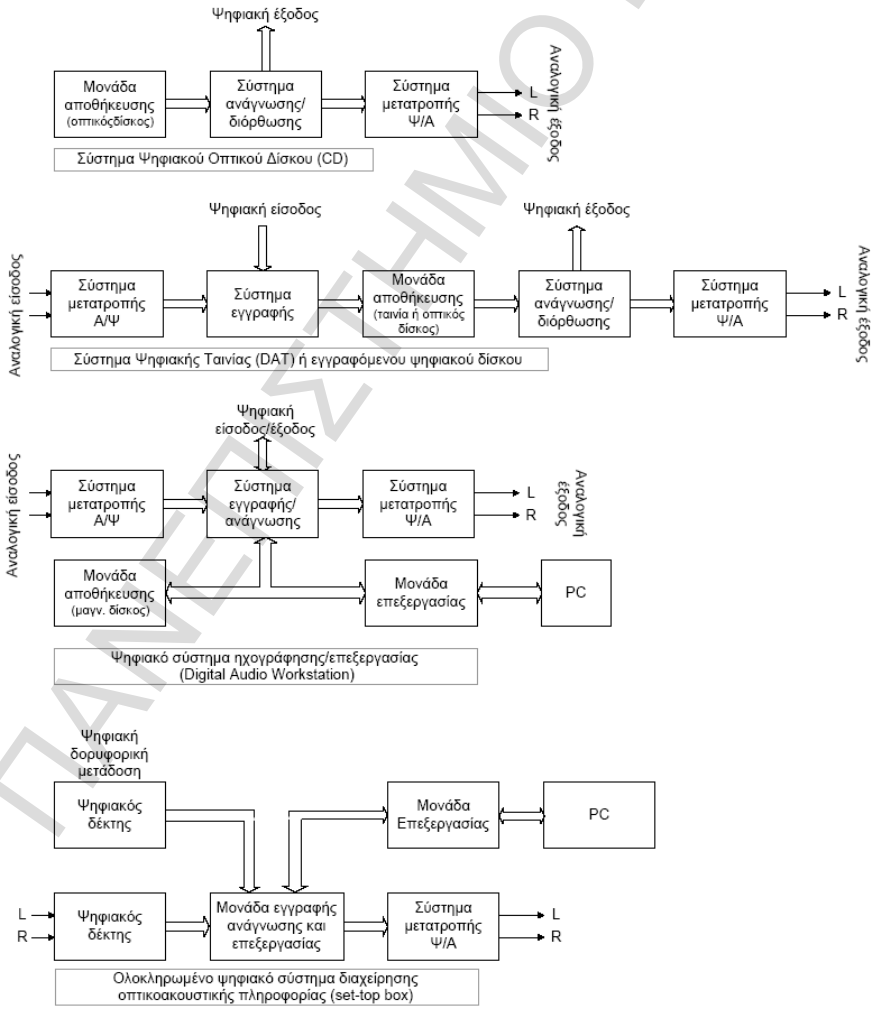
[2] British Library, *A manual of sound Archive Administration,* Alan Word, 1990

[3] Νίκος Καλουμπιτσίδης, *Σήματα Συστήματα και Αλγόριθμοι,* εκδ. Δίαυλος , Αθήνα1994 (5η έκδοση).

[4] British Library, *A manual of sound Archive Administration,* Alan Word, 1990



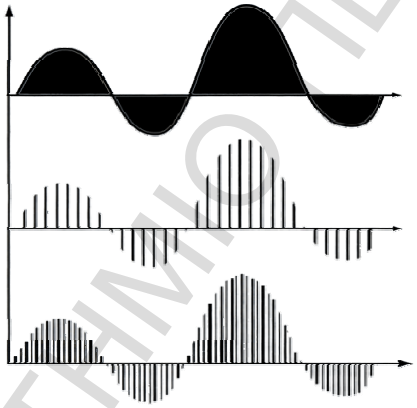
Διαδικασία Ψηφιοποίησης Ηχητικού σήματος.



5.2 Δειγματοληψία

Όπως προαναφέρθηκε, η δειγματοληψία είναι η διαδικασία που παράγει την σειρά των δειγμάτων από τα οποία δημιουργείται η ψηφιακή πληροφορία, δηλαδή η διαδικασία κατά την οποία ένα αναλογικό σήμα sc(t) (σήμα του οποίου η ανεξάρτητη χρονική μεταβλητή t, παίρνει συνεχείς τιμές) μετατρέπεται σε σήμα διακριτού χρόνου (σήμα το οποίο ορίζεται σε διακριτές χρονικές στιγμές), πραγματοποιείται δηλαδή μια απεικόνιση της μορφής s_c (t) sd(nTs) όπου n=1,2,…. είναι ο αριθμός του δείγματος και Ts(sec) η περίοδος δειγματοληψίας.[5]

Η επιλογή της περιόδου δειγματοληψίας, καθορίζει και το μέτρο της ποιότητας του ψηφιακού ήχου με επακόλουθο να αποτελεί καθοριστικό παράγοντα ορθής μετατροπής ενός σήματος συνεχούς χρόνου σε διακριτού χρόνου δείγματος. Η συχνότητα δειγματοληψίας (sampling frequency), δηλώνει τον αριθμό των δειγμάτων πλάτους του αναλογικού σήματος καθώς και το πόσες φορές το δευτερόλεπτο παίρνουμε δείγματα από το αναλογικό σήμα, το οποίο μετριέται σε χιλιάδες κύκλους ανά δευτερόλεπτο η kHz. Όσο μεγαλύτερη είναι η συχνότητα ψηφιοποίησης τόσο καλύτερη ποιότητα αναπαραγωγής έχουμε άλλα ταυτόχρονα, με ριζική αύξηση του χώρου αποθήκευση (σχ.2.2).[6]



(α) Αναλογικό σήμα, (β) Ψηφιακό με χαμηλή συχνότητα,
(γ) Ψηφιακό με υψηλή συχνότητα δειγματοληψίας.

Απαραίτητη διαδικασία για την μετατροπή ενός αναλογικού ηχητικού σήματος σε ψηφιακό είναι η δειγματοληψία, όπως αναδύεται στο **θεώρημα του Nyquist** από το οποίο απορρέει ότι συχνότητα δειγματοληψίας θα πρέπει να είναι μεγαλύτερη ή ίση του διπλάσιου της μέγιστης συχνότητας του αρχικού αναλογικού σήματος, **fs>2f ή Ts<=T/2.**

Για να ισχύει το παραπάνω πρέπει το αρχικό σήμα να αναζωπυρωθεί στο μισό της συχνότητας δειγματοληψίας περνώντας από ένα ιδανικό βαθυπερατού φίλτρου (βλ. παρακάτω).

Οι συχνότητες δειγματοληψίας που χρησιμοποιούνται σήμερα για την ψηφιακή εγγραφή μουσικής, κυμαίνονται στα 22 με 44kHz επιτρέποντας να περάσει ένα φάσμα ακουστικών συχνοτήτων από 11 έως 22kHz. Κατά συνέπεια, ο ήχος που γίνεται αντιληπτός από το ανθρώπινο αυτί δεν ξεπερνά τα 18 με 22kHz και γι' αυτό θα πρέπει να δειγματολογούμε με συχνότητα περίπου 44kHz(CD) με επακόλουθο ένα πιστό ψηφιακό αντίγραφο του αναλογικού σήματος. Ο A/D παρέχει σε κάθε δείγμα μια τιμή ακέραιου αριθμού που ισοδυναμεί με το μέγεθος του δείγματος . Η ανάλυση τιμών εξαρτάται από την ανάλυση των δειγμάτων (sampling

[5] Καλουμπιτσίδης , Νίκος, *Σήματα Συστήματα και Αλγόριθμοι,* εκδ. Δίαυλος , Αθήνα1994 (5η έκδοση).

[6] Σ,Ν Δημητριάδης, Α.Σ Πομπόρτσης, Ε.Γ Τριανταφύλλου, *Τεχνολογία Πολυμέσων* ,εκδ. Τσίολα, Θεσσαλονίκη 2004

resolution), όπου εάν ο A/D είναι 8 bits, οι τιμές μπορεί να κυμαίνονται από 0 έως 255. Συνήθως κατά την διαδικασία της ηχογράφηση επιλέγουμε την μέγιστη ανάλυση δειγμάτων που υποστηρίζεται από το hardware. Ενώ μια καλή επιλογή είναι τα 8 bits και τα οποία χρησιμοποιούνται ευρέως . Αντίθετα, για ποιότητα CD ο ψηφιακός ήχος κωδικοποιείται ως 44,1kHz στηριζόμενος στο 16bit σύστημα δηλ. το αρχικό κύμα τεμαχίζεται 44.100 φορές το δευτερόλεπτο και ένα μέσο εύρος επίπεδο εφαρμόζεται σε κάθε δείγμα (16 bit σημαίνει ότι 65.536 διαφορετικές τιμές μπορούν να οριστούν ή να κβαντοποιηθούν σε κάθε δείγμα).

Κανάλια	Δείγμα	Δειγματοληψία	Χρόνος Δείγματος	Μέγεθος Αρχείου
Mono	8bits	11kHz	10 sec	110000 bytes
Stereo	8bits	11kHz	10 sec	220000 bytes
Mono	8bits	22kHz	10 sec	220000 bytes
Stereo	8bits	22kHz	10 sec	440000 bytes
Mono	16bits	44,1kHz	10 sec	882000 bytes
Stereo	16bits	44,1kHz	10 sec	1764000 bytes

Στον παραπάνω πίνακα παρατηρείται ο ρυθμός δειγματοληψίας (sampling rate) και ο αριθμός των δειγμάτων που συλλέγονται ανά sec. Όπως είναι φυσικό, το μέγεθος του αρχείου στο οποίο αποθηκεύεται το ψηφιοποιημένο σήμα ολοένα και αυξάνεται με το μέγεθος του δείγματος (sampling size) και το ρυθμό δειγματοληψίας (sampling rate) , με αποτέλεσμα η καλύτερη απόδοση να επιταχύνεται με δειγματοληψία 44,1 kHz με μέγεθος δείγματος 16bits σε 2 κανάλια ήχου (ποιότητα μουσικών CD).

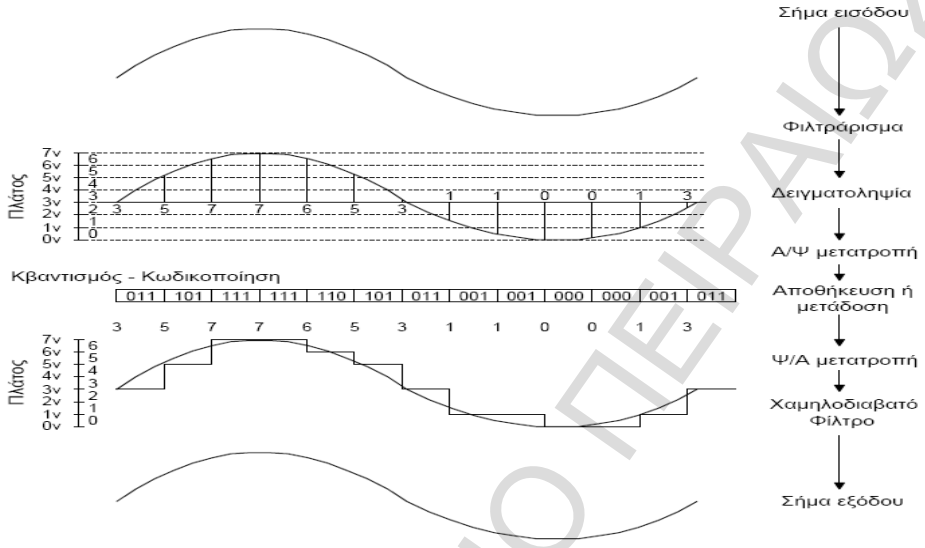
Συμπερασματικά, για να υπολογίσουμε το μέγεθος ενός ψηφιακού αρχείου το οποίο εξαρτάται από την χρονική διάρκεια και την ποιότητα του ήχου, πολλαπλασιάζουμε την συχνότητα δειγματοληψίας σε Hz με το μέγεθος του δείγματος σε bit και με την διάρκεια του ήχου σε δευτερόλεπτα :

Χωρητικότητα (bits) = Συχνότητα (Hz) . Μέγεθος δείγματος (bits) . Διάρκεια (sec)

5.3 Προβλήματα Δειγματοληψίας

Ο επιστήμονας H.Nyquist, πρώτος ανακάλυψε ότι η μέγιστη συχνότητα αναλογικού σήματος που μπορεί να αποδοθεί χωρίς αλλοίωση είναι το μισό της συχνότητας δειγματοληψίας. Συχνότητες μεγαλύτερες από τα μισά της συχνότητας ψηφιοποίησης εμφανίζονται λανθασμένες σαν χαμηλές συχνότητες, με επακόλουθο το φαινόμενο αυτού του ελλιπούς ρυθμού δειγματοληψίας να ονομάζεται αναδίπλωση (aliasing) .Έτσι για να μην εμφανιστούν συχνοτικά ειδώλια που θα παραμορφώσουν το σήμα μετά την δειγματοληψία οι ψηφιοποιητές φιλτράρουν όλες τις συχνότητες που είναι μεγαλύτερες από το μισό της συχνότητας δειγματοληψίας.

Κατά συνέπεια, το ηλεκτρονικό σήμα μετά το στάδιο της προενίσχυσης τροφοδοτείται σε ένα ιδανικό βαθυπερατό φίλτρο (low-pass) , με στόχο την αποκοπή των συχνοτήτων μεγαλύτερων από την μέγιστη συχνότητα δειγματοληψίας για αποφυγή του φαινομένου της αναδίπλωσης. Στη συνέχεια, κατά το στάδιο της δειγματοληψίας τροποποιείται το αναλογικό ηλεκτρικό σήμα της εισόδου σε δυαδικούς αριθμούς και αποθηκεύεται στη μνήμη του Η/Υ, με αποτέλεσμα να ολοκληρώνεται ο μετασχηματισμός από το αναλογικό (analog) στο ψηφιακό (digital) (σχ.2.3)[7].



Διαδικασία μετατροπής αναλογικού σε ψηφιακή μορφή και αντίστροφα.[8]

Η απόδοση ενός ψηφιακού ηχητικού σήματος ακολουθεί μια αντίστροφη διαδρομή βημάτων από αυτή της καταγραφής, δηλαδή η τροφοδότηση του σήματος υλοποιείται από τη μνήμη του Η/Υ σε ένα μετατροπέα ψηφιακού σε αναλογικού σήματος. Με βάση τον τρόπο λειτουργίας και την συχνότητα δειγματοληψίας, ο μετατροπέας παράγει ένα ηλεκτρικό σήμα που αποτελεί μια προσεγγιστική μορφή του ηλεκτρικού σήματος το οποίο ενισχύεται και στη συνέχεια τροφοδοτείται στη είσοδο του ηχείου, αποδίδοντας το σήμα.

Στο σημείο αυτό, ανακύπτει το ερώτημα αν η εισαγωγή τέτοιων φίλτρων (filter), διαμορφώνουν νέα προβλήματα κατά την ψηφιοποίηση του ήχου. Απάντηση είναι, πώς δεν υπάρχει ένα τέλειο βαθυπερατό φίλτρο που να μπορεί να αποκόψει τέλεια συχνότητες πάνω από ένα επιθυμητό όριο, ενώ η χρήση τέτοιων φίλτρων διαμορφώνει μικρές παραμορφώσεις στο σήμα στο οποίο και εκφράζεται. Ομοίως, όπως έχει ήδη τυπωθεί κατά την διαδικασία της Ψ/Α μετατροπής εξαιτίας των κβαντισμένων τιμών πλάτος του αναπαραγόμενου σήματος, διαμορφώνονται συχνότητες υψηλότερες της μέγιστης συχνότητας όπου και φιλτράρονται με ένα παρόμοιο φίλτρο εξομάλυνσης (smoothing filter)[9].

[7] Πηγή:Peter Elsea, Basic of Digital Recording, 23/3/06

[8] Πηγή: Φλώρος Αντρέας ,*Ψηφιακή Τεχνολογία Ήχου.*

[9] British Library, *A manual of sound Archive Administration,* Alan Word, 1990

5.4 Κβαντοποίηση

Το επόμενο εφαρμοσμένο βήμα με μεγάλη συμβολή στην ποιότητα του ήχου είναι αυτό της **κβάντωσης**, προκειμένου να αναπαρασταθεί κάθε δείγμα με την μορφή μιας δυαδικής σειράς από bits, καθώς πρέπει στο συνεχώς μεταβαλλόμενο πλάτος της τάσης του αναλογικού σήματος να τεθεί μια διακριτή τιμή. Σημαντικό είναι να τον τονίσουμε ότι η δειγματοληψία και η κβαντοποιήση είναι συμπληρωματικές διαδικασίες και ο συνδυασμός αυτών των δύο διαδικασιών, καλείται ψηφιοποίηση. Η κβαντοποιήση παίζει καθοριστικό ρόλο στο σχεδιασμό του συστήματος του ψηφιακού ήχου, ενώ ανάλογα με τις στάθμες κβάντισης διαμορφώνεται και το format του ψηφιακού ήχου.

Βασική παράμετρος της φάσης αυτής είναι το μέγεθος του δείγματος (sampling size), με επακόλουθο όσα περισσότερα bits χρησιμοποιούνται με μεγαλύτερη ακρίβεια περιγραφής. Για παράδειγμα, αν έχουμε 8 bit, τότε μπορούν χρησιμοποιηθούν 2^8=256 διαφορετικές τιμές, ενώ για 16bit μπορούν να χρησιμοποιηθούν 2^{16}=65.536 διαφορετικές τιμές, άρα μεγαλύτερη πιστότητα και ακρίβεια.

Κατά την πεπερασμένη διακριτικότητα της ψηφιακής αναπαράστασης, εισάγεται ο *θόρυβος* ένα σημαντικό πρόβλημα τόσο κατά την ηχογράφηση όσο και κατά την αναπαραγωγή, γνωστός και ως **θόρυβος κβάντοποίησης**. Το φαινόμενο αυτό προέρχεται από το γεγονός ότι τα δείγματα του αναλογικού σήματος που θα χρησιμοποιηθούν δεν είναι κατά ανάγκη ακέραιοι και συνεπώς η μετατροπή τους σε ψηφιακά δείγματα συνεπάγεται στην στρογγυλοποίηση του δείγματος. Πρέπει, όμως να τονισθεί ότι στη περίπτωση αυτή το ψηφιοποιημένο σήμα απαρτίζεται από το άθροισμα του αναλογικού σήματος καθώς και από ένα σήμα θορύβου το οποίο και μετριέται σε αρνητικά decibel (db= μονάδα μέτρησης έντασης του ήχου) .Το σφάλμα αυτό εκφράζεται από το λόγο σήματος προς θόρυβο, (Signal to Noise Ratio, SNR)[10] **S/E (dB)=6.02n+176** όπου το n είναι ο αριθμός των bits της λέξης στο format του ψηφιακού ήχου. Όσο μεγαλύτερος είναι ο αριθμός των db τόσο μικρότερος είναι ο θόρυβος, ενώ τα περισσότερα συστήματα σήμερα κυμαίνονται στα 90db SNR, όπου η παραμόρφωση μετριέται επί τις εκατό του αρχικού σήματος Η ευκρίνεια του κβαντισμού και του λόγου σήματος προς θόρυβο παρουσιάζεται στον παρακάτω πίνακα όπου ενδεικτικές τιμές SNR αντιστοιχούν σε διαφορετικές τάξεις μετατροπής.

Τάξη N(bits)	SNR (db)
8	49.8
12	73.8
16	97.8
18	109.8

Πίνακας Ευκρίνειας Κβαντισμού και SNR.

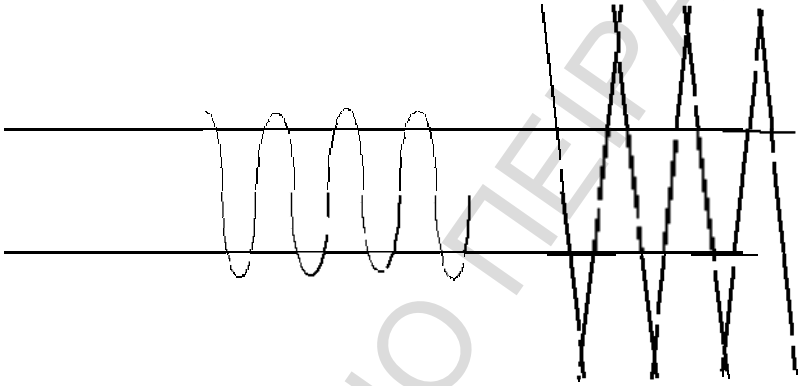
Συνοψίζοντας τα πιο πάνω, ανάγεται το συμπέρασμα ότι ο θόρυβος που παράγουν τα ψηφιακά ηχητικά συστήματα , εμφανίζεται κατά την διαδικασία της αναπαραγωγής του κβαντισμένου ηχητικού σήματος, έτσι ώστε να διαμορφώνεται ένας ανεπιθύμητος χαρακτήρας.

Στην περίπτωση των 16bit συστημάτων, επιβάλλεται η χρήση του **dither** κατά την Α/Ψ, μέθοδος κατά την οποία ενσωματώνεται μικρή ποσότητα αναλογικού θορύβου πριν από το στάδιο της

[10] British Library, *A manual of sound Archive Administration,* Alan Word,

δειγματοληψίας, εξασθενώντας πολλά από τα προβλήματα που συζητήθηκαν παραπάνω ούτος ώστε να εμφανίζεται ένα ικανοποιητικό αποτέλεσμα. Σε γενικές γραμμές, η προσθήκη του dither τροποποιεί την μορφή του σφάλματος και το καθιστά ανεξάρτητο από το σήμα εισόδου, αποκόπτει κάθε είδους αρμονικής παραμόρφωσης η οποία εμφανίζεται κατά την μετατροπή σήματος μικρού πλάτους και τέλος βελτιώνει την διακριτή ικανότητα του κβαντιστή αυξάνοντας κατά υποκειμενικό τρόπο την δυναμική του περιοχή.

Ένας άλλος σημαντικός παράγοντας που επηρεάζει την ποιότητα του ήχου, είναι ο **ψαλιδισμός** (clipping) του σήματος, όπου κατά την εγγραφή του σήματος η μέγιστη στάθμη του σήματος εισόδου υπερβαίνει την μέγιστη στάθμη κβαντισμού διαμορφώνοντας κάποια διαστρέβλωση. Στο παρακάτω σχήμα 2.4, απεικονίζεται ένα ψαλιδισμένο σήμα με κοψίματα στην κορυφή και στη βάση με επακόλουθο ο ψαλιδισμός να μειώνεται αφού μειωθεί το μέγεθος του σήματος.



Ψαλιδισμός (Clipping).

5.5 Κωδικοποίηση Ψηφιακού Σήματος

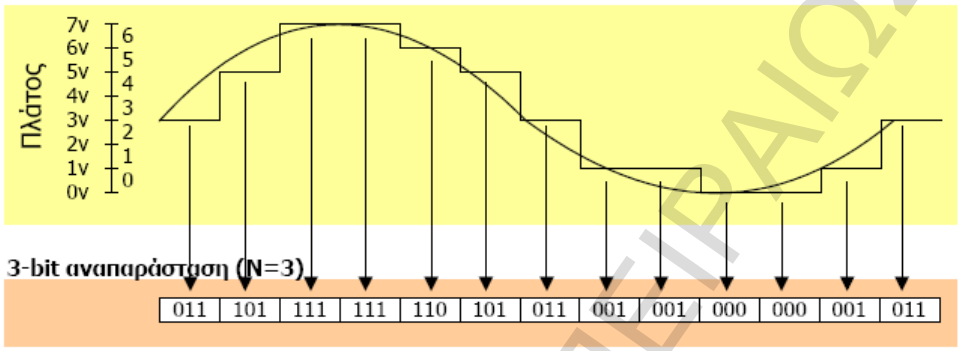
Ο συνδυασμός της δειγματοληψίας και της κβαντοποίησης ως συμπληρωματικές διαδικασίες αποκαλείται ψηφιοποίηση, ενώ για την εκμετάλλευση των πλεονεκτημάτων τους απαραίτητη προϋπόθεση η χρησιμοποίηση μιας διαδικασίας **κωδικοποίησης** (encoding process) για την μετατροπή του διακριτού συνόλου των τιμών των δειγμάτων σε μια πιο κατάλληλη μορφή. Κατά το στάδιο αυτό, η ένταση του κάθε δείγματος αντιστοιχεί σε ένα σταθερό δυαδικό αριθμό 0,1 και διατίθεται για την αποθήκευση κάθε δείγματος και καλείται εύρος δείγματος (τιμές που μπορεί να πάρει το κάθε δείγμα).

Για το ψηφιακό ήχο οι δυνατότητες ποικίλουν από 8bits (2^8=256 στάθμες) έως και 16bits (2^{16}=65536 στάθμες) ,με επακόλουθο η χρήση ενός δυαδικού συστήματος να αναδύει καλύτερα αποτελέσματα σε συνάρτηση με την επίδραση του θορύβου σε ένα μέσο μετάδοσης και αυτό γιατί ένα δυαδικό σύμβολο, αντέχει σε μια σχετική υψηλή στάθμη θορύβου καθώς εύκολα μπορεί να αναγεννηθεί.

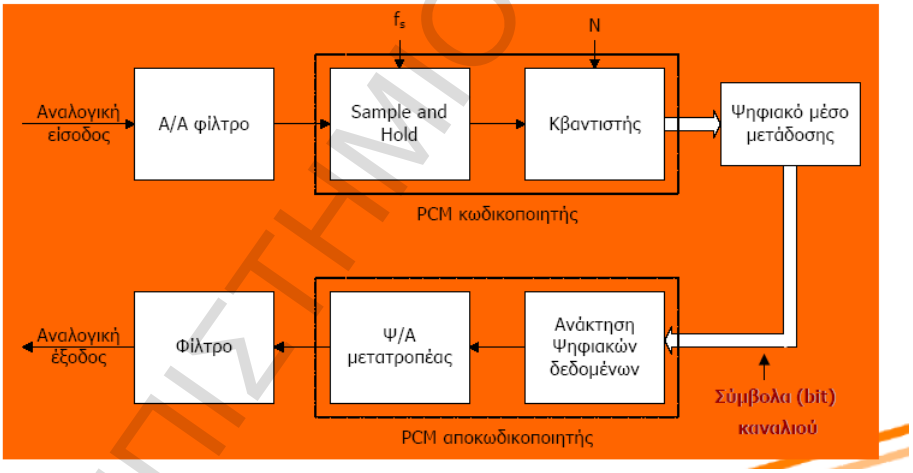
5.5.1 PCM (Pulse Code Modulation).

Μια από τις πιο απλές και ευρέως μεθόδους κωδικοποίησης ψηφιακού ήχου είναι η παλμοκωδική κωδικοποίηση PCM. Στην μέθοδο αυτή κάθε δείγμα αναπαριστάται με ένα σύνολο παλμών που αντιστοιχούν στον δυαδικό κώδικα και στην τιμή του δείγματος με επακόλουθο να αποθηκεύεται ένα προς ένα τα δείγματα σε ψηφιακή μορφή χρησιμοποιώντας γραμμική κωδικοποίηση. Όπως είναι αναμενόμενο, η πιστότητα του σήματος που προκύπτει είναι συνάρτηση του δυαδικού κώδικα, π.χ. ένα δυναμικό εύρος 128 τιμών θα έχει ως αποτέλεσμα την απάλειψη ήχων στο ψηφιοποιημένο σήμα με ένταση ίση ή μικρότερη από το 1/128 της έντασης του δυνατότερου ήχου που μπορεί να αναπαρασταθεί από το σήμα. Στο

σημείο αυτό, ανακύπτει το ερώτημα αν η πιστότητα σε ένταση σε ασθενή ηχητικά σήματα εισάγει πρόβλημα. Προφανώς εισάγει και για αυτό το λόγο έχουν αναπτυχθεί άλλες τεχνικές οι οποίες χρησιμοποιούν λογαριθμική κωδικοποίηση PCM όπου και αναλύονται στις επόμενες παραγράφους. Τέτοιες τεχνικές, χρησιμοποιούν περισσότερα δυαδικά ψηφία για την αναπαράσταση ασθενών ήχων και λιγότερα για τους ήχους μεγάλης έντασης στους οποίους η ευαισθησία της ανθρώπινης ακοής σε σχετικές διαφορές έντασης είναι μειωμένη (βλ. προβλήματα δειγματοληψίας).



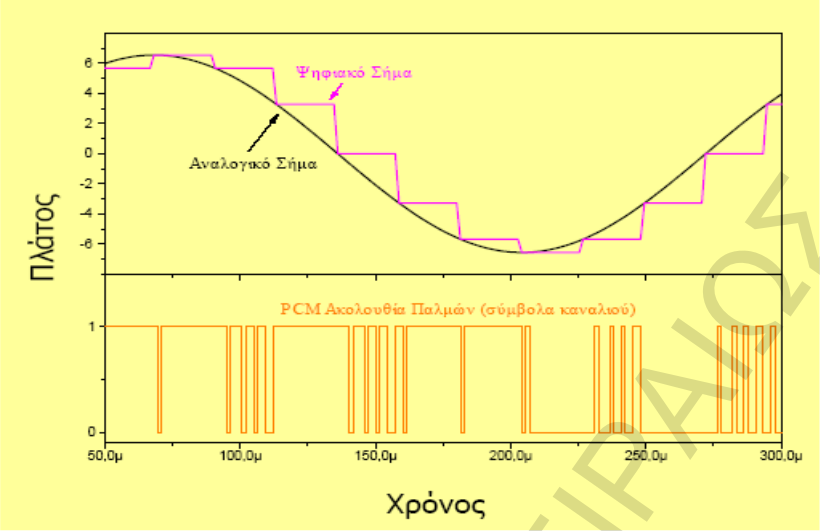
Κωδικοποίηση PCM[11]



Σχ. 3.2 : Χρήση PCM Κωδικοποίηση[12]

[11] Πηγή: Φλώρος Αντρέας ,*Ψηφιακή Τεχνολογία Ήχου.*

[12] Πηγή: Φλώρος Αντρέας ,*Ψηφιακή Τεχνολογία Ήχου*.



Παράδειγμα PCM Κωδικοποίηση.[13]

5.5.2 Mu-Law PCM και A-Law PCM.

Η διαφορά της λογαριθμικής κωδικοποίησης ήχου σε σχέση με την γραμμική έγκειται στο γεγονός ότι κατά την λογαριθμική αντιστοίχηση, αντιστοιχούνται ολοένα και περισσότερες στάθμες σε χαμηλές συχνότητες και λιγότερες στις υψηλές, ενώ ταυτόχρονα πραγματοποιείται τόσο καλύτερη αναπαράσταση όσο και καλύτερη συμπίεση του σήματος με αυτό της παλμοκωδικής, η οποία δεν πραγματοποιεί καμία συμπίεση ούτως ώστε να προκύπτει ένα αρχείο ήχου κωδικοποιημένο χωρίς καμία απώλεια. Πιο συγκεκριμένα, 8bits σε συνδυασμό με λογαριθμική κωδικοποίηση καλύπτει το ίδιο εύρο τιμών 14bits και παλμοκωδική κωδικοποίηση PCM, επομένως πρόκειται για μια συμπίεση της τάξης του 1,75 προς 1. Κατά συνέπεια, στην κατηγορία αυτή της λογαριθμικής κωδικοποίησης ήχου εντάσσονται οι Mu-Law PCM και A-Law PCM, δύο μέθοδοι που έχουν τυποποιηθεί από το ITU-T (International Telecommunication Union-Telecommunication Standardization Sector) στο πρότυπο G711, Pulse Code Modulation of noise Frequencies το οποίο και περιγράφει λεπτομερές τα δύο είδη κωδικοποίησης. Συγκεκριμένα, η Mu-Law υλοποιείται σε ISDN (Integrated Services Digital Network) δίκτυα της Ιαπωνίας και Β.Αμερικής ενώ η A-Law σε δίκτυα των υπόλοιπων χωρών.

5.5.3 DPCM (Differential Pulse Code Modulation).

Σε σχέση με την PCM κωδικοποίηση ψηφιακού σήματος, η διαφορική παλμοκωδική κωδικοποίηση δεν κωδικοποιεί το κάθε δείγμα ανεξάρτητα από τα υπόλοιπα αλλά σε συνάρτηση με τα γειτονικά δείγματα, αποθηκεύοντας τις διαφορές μεταξύ των διαδοχικών τιμών και όχι τις απόλυτες τιμές των δειγμάτων. Δηλαδή για την χρονική στιγμή t θα κωδικοποιηθεί η διαφορά των δειγμάτων $\delta t-\delta_{t-1}$ όπου ως προβλεπόμενη τιμή έχει χρησιμοποιηθεί η τιμή του δείγματος δ_{t-1} την χρονική στιγμή t-1. Η βασική έννοια DPCM που κωδικοποιεί μια διαφορά, στηρίζεται στο γεγονός ότι τα περισσότερα σήματα πηγής παρουσιάζουν σημαντικό συσχετισμό μεταξύ των διαδοχικών δειγμάτων που κωδικοποιούν, ούτως ώστε οι τιμές των δειγμάτων να διαμορφώνουν χαμηλό ποσοστό δυαδικών ψηφίων. Για την υλοποίηση της βασική έννοιας θα

[13] Πηγή: Φλώρος Αντρέας ,*Ψηφιακή Τεχνολογία Ήχου.*

πρέπει να προβλέψουμε την τρέχουσα αξία των δειγμάτων η οποία βασίζεται στα προηγούμενα δείγματα καθώς και να κωδικοποιήσουμε την αξία του δείγματος και την προσληφθείσα αξία.

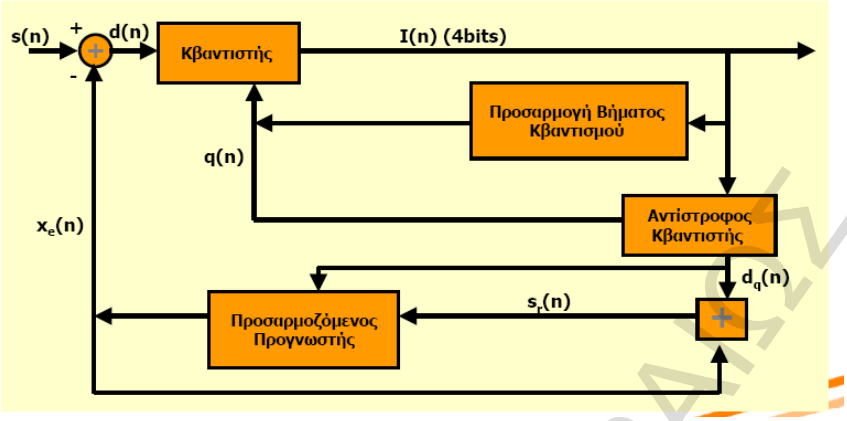
Μια ειδική μορφή της διαφορικής παλμοκωδικής κωδικοποίησης είναι η DM (Delta Modulation), ως απλουστευμένη DPCM κατά την οποία η διαφορά της προβλεπόμενης και της τρέχουσας τιμής του δείγματος κωδικοποιείται με ένα μόνο bit παίρνοντας τιμές σύμφωνα με την **αρχή DM**: Η παραγωγή DM είναι **0** εάν το τρέχον δείγμα έχει μικρότερο εύρος από το προηγούμενο και **1** εάν το τρέχον δείγμα έχει εύρος μεγαλύτερο από το αμέσως προηγούμενο, με επακόλουθο το DM να κωδικοποιεί την κατεύθυνση των διαφορών στο εύρος σήματος αντί την αξία της διαφοράς DPCM. Αυτό σημαίνει ότι κάθε δείγμα μπορεί να είναι είτε μεγαλύτερο είτε μικρότερο κατά ένα κβάντο από το προηγούμενο του με αποτέλεσμα ο περιορισμός να οδηγεί σε μεγάλη οικονομία αλλά αν το σήμα αλλάζει γρήγορα θα υπάρχει μεγάλη απώλεια πληροφορίας.

5.5.4 ADPCM (Adaptive Differential Pulse Code Modulation).

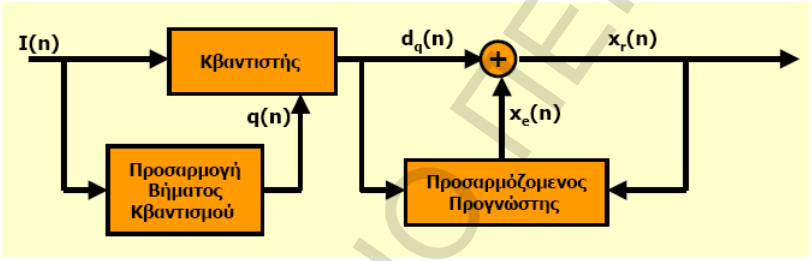
Η διαφορική παλμοκωδική κωδικοποίηση, αποτελεί μια επέκταση της DPCM μεθόδου κατά την οποία προβλέπεται η τιμή ενός δείγματος με βάση την τιμή του γειτονικού δείγματος και αυτό επειδή τα γειτονικά δείγματα πολύ πιθανόν να είναι όμοια αν όχι και ίδια. Συνεπώς, η ADPCM κωδικοποίησης, υπολογίζει την διαφορά μεταξύ της πραγματικής τιμής του δείγματος και της τιμής που είχε προβλεφθεί και κωδικοποιείται με συνεπακόλουθο, η διαφορά της τιμής να είναι σχετικά ελάχιστη (4bits) απαιτώντας λιγότερα bits. Στο σημείο αυτό, ένα σημαντικό μειονέκτημα αναδύεται ανάμεσα στις δύο μεθόδους κωδικοποίησης ADPCM και DPCM και αυτό εξαιτίας της εξάρτησης του μεγέθους των διαφορών μεταξύ διαδοχικών δειγμάτων ενός ηχητικού σήματος σε σχέση με την συχνότητα.

Όπως είναι αντιληπτό, οι διαφορές μεταξύ διαδοχικών δειγμάτων σε σχέση με ήχους χαμηλών συχνοτήτων, είναι σημαντικά μικρότερες από τις αντίστοιχες διαφορές σε ήχους υψηλών συχνοτήτων, με αποτέλεσμα η κωδικοποίηση αυτών των διαφορών να εντάσσεται στο συχνοτικό περιεχόμενο του σήματος της κάθε χρονικής στιγμής, γεγονός που δεν πραγματοποιείται στις παραπάνω μεθόδους. Αντίθετα, οι μέθοδοι *κωδικοποίηση υποζώνης* (subband coding) ,κατηγοριοποιούν το ηχητικό σήμα σε δύο ή περισσότερες ζώνες συχνοτήτων και συμπιέζουν κάθε μια από αυτές ξεχωριστά με σκοπό μια τέτοια κατηγοριοποίηση να εκμεταλλεύεται τα χαρακτηριστικά της ανθρώπινης ακοής που παρουσιάζει την μέγιστη ευαισθησία στο εύρος συχνοτήτων μεταξύ 2700-3200Hz με την ευαισθησία να ελαττώνεται όσο απομακρυνόμαστε από την ζώνη αυτή.

Συνοψίζοντας τα πιο πάνω, ανάγεται το συμπέρασμα, πως σε μια τέτοια μέθοδο κωδικοποίησης υποζώνης, τα αποτελέσματα της δεν γίνονται αντιληπτά από την ανθρώπινη ακοή και αυτό επιτυγχάνεται, γιατί έχει την ικανότητα τόσο να συμπιέζει ήχους με συχνοτικό περιεχόμενο που απέχουν από την παραπάνω ζώνη όσο και να εφαρμόζει την ελάχιστη συμπίεση που ανήκει στην ζώνη αυτή. Σε μια τέτοια κατηγορία, ανήκουν τα πρότυπα MPEG audio, Dolby AC-2 και AC-3 και το RealAudio για τα οποία εκτεταμένη αναφορά γίνεται στο παρακάτω κεφάλαιο.



Σχ.3.4: Κωδικοποιητής ADPCM[14]



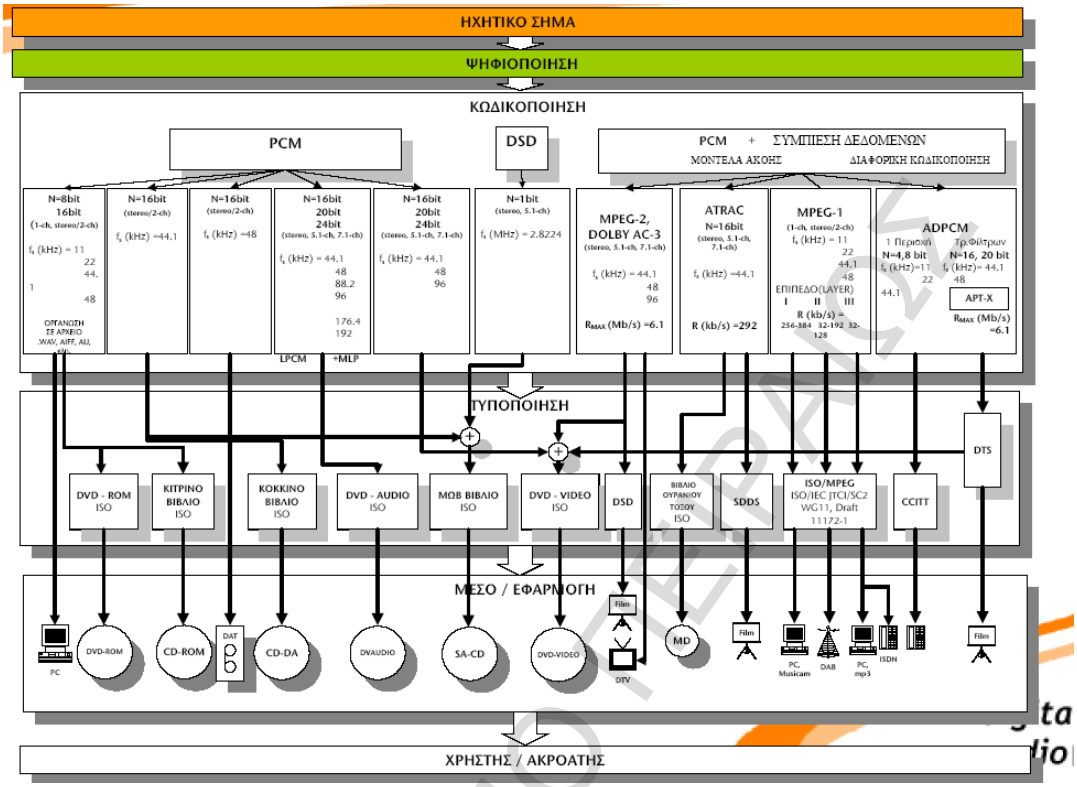
Αποκωδικοποιητής ADPCM[15]

5.5.5 LPC(Linear Predictive Coding)

Μια από τις νέες τεχνολογίες κωδικοποίησης ήχου που έχουν συνταχθεί για την ανθρώπινη ομιλία και πραγματοποιούν σημαντικούς βαθμούς συμπίεσης είναι η γραμμική προβλεπόμενη κωδικοποίηση LPC, με σκοπό ο κωδικοποιητής αυτός να συγκρίνει τα σήματα φωνής που παραλαμβάνει με βάση ένα αναλυτικό μοντέλο φωνής που έχει αποθηκευμένο. Τα χαρακτηριστικά που ταιριάζουν καλύτερα στο αναλυτικό μοντέλο μεταδίδονται, ενώ ο αποκωδικοποιητής χρησιμοποιεί τα χαρακτηριστικά αυτά για να ανασυνθέσει τα φωνητικά σήματα. Παρόλα αυτά, μειονέκτημα της μεθόδου αυτής εντοπίζονται στην αδυναμία της να επεξεργαστεί άλλο σήμα εκτός από την ομιλία, η οποία δημιουργήθηκε για να εξυπηρέτηση την μετάδοση ομιλίας στην κινητή τηλεφωνία.

[14] Πηγή: Φλώρος Αντρέας ,*Ψηφιακή Τεχνολογία Ήχου*

[15] Πηγή: Φλώρος Αντρέας ,*Ψηφιακή Τεχνολογία Ήχου*



Διάγραμμα Κωδικοποιήσεων Ψηφιακού Ήχου.[16]

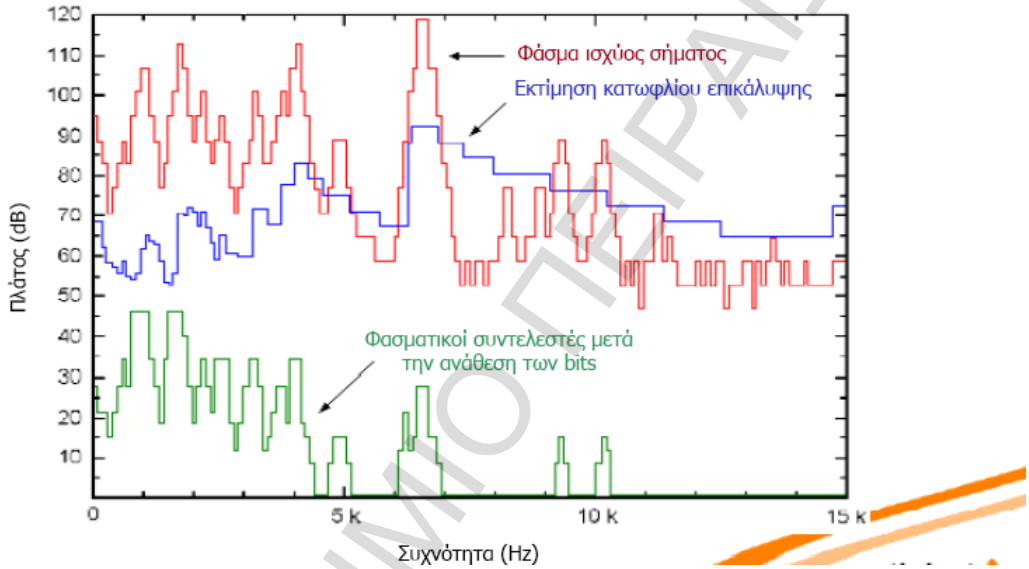
5.6 Συμπίεση Ψηφιακού Ήχου

Με δεδομένη την ολοένα και αυξανόμενη χρήση της ψηφιακής τεχνολογίας η οποία αντικαθιστά την αναλογική που παραδοσιακά χρησιμοποιείται, εδώ και δεκαετίες στις τηλεπικοινωνίες, την ηχογράφηση και αναπαραγωγή της μουσικής αλλά και σε άλλους τομείς έχει αρχίσει παράλληλα να γίνεται έρευνα προς την κατεύθυνση της συμπίεσης ψηφιακής πληροφορίας και στην περίπτωση μας του ψηφιακού ηχητικού σήματος με στόχο τη οικονομία εύρους φάσματος. Σύμφωνα με αυτά που έχουν αναφερθεί στα παραπάνω κεφάλαια, παρατηρείται ότι κατά την διαδικασία της ψηφιοποίησης ηχητικών δεδομένων, παράγονται αρχεία μεγάλων μεγεθών τα οποία δύσκολο να μεταφερθούν ή να υποστούν επεξεργασία αφού προϋποθέτουν πολύ χρόνο, μεγάλους αποθηκευτικούς χώρους και πολύ μνήμη.

Το τεχνικό πρόβλημα του χώρου και της ανεπάρκειας στην ταχύτητα μεταφοράς έρχονται να λύσουν οι τεχνολογίες συμπίεσης του ηχητικού σήματος. Οι τεχνολογίες αυτές, χρησιμοποιούν διάφορες τεχνικές μείωσης του όγκο και της ροής των δεδομένων που απαιτούνται για την κωδικοποίηση των ηχητικών σημάτων και διακρίνονται σε δύο κατηγορίες, τις μη απωλεστικές (lossless) και τις απωλεστικές (lossy). Οι μη απωλεστικές, έχουν το ιδιαίτερο χαρακτηριστικό ότι η διαδικασία συμπίεσης δεν αλλοιώνει καθόλου την πληροφορία δηλαδή μετά την αποσυμπίεση η πληροφορία επανέρχεται στην ακριβώς στην αρχική της μορφή. Συνήθως, οι μη απωλεστικοί αλγόριθμοι συμπίεσης ψηφιακού ήχου, εφαρμόζονται σε περιπτώσεις που δεν υπάρχει κανένα περιθώριο απωλειών. Αντίθετα, οι απωλεστικές τεχνικές αλλοιώνουν τα δεδομένα,

[16] Πηγή: Φλώρος Αντρέας ,*Ψηφιακή Τεχνολογία Ήχου*

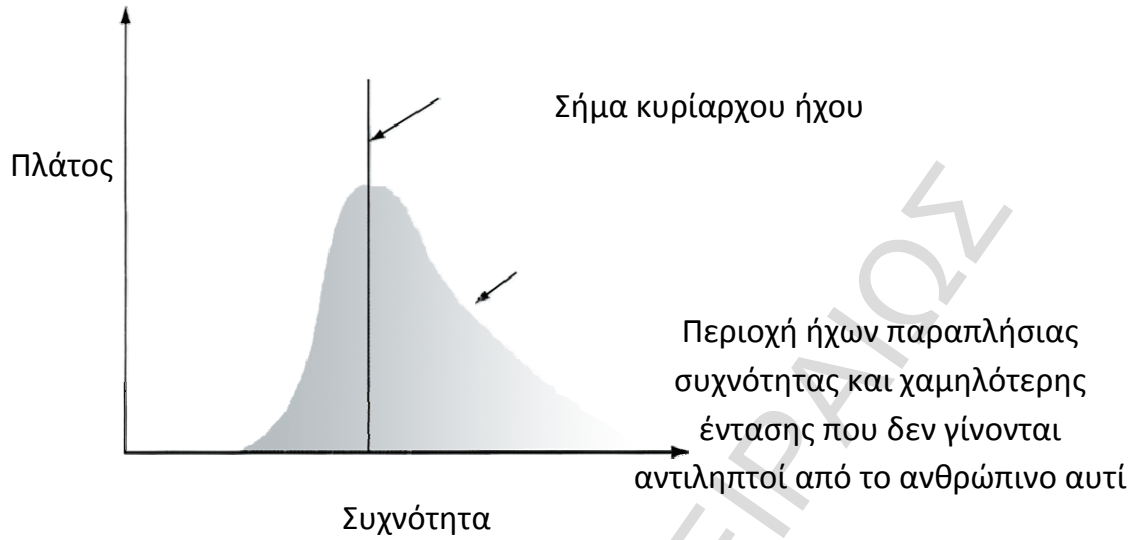
απορρίπτοντας τα μέρη εκείνα που δεν είναι χρήσιμα με βάση κάποιο συγκεκριμένο κριτήριο (irrelevancy) . Είναι φανερό, ότι σε τέτοιες περιπτώσεις το σημασιολογικό περιεχόμενο ουσιαστικά δεν μεταβάλλεται αλλά υπεισέρχεται η έννοια της μείωσης της ποιότητας καθώς οι απωλεστικές τεχνικές είναι περισσότερο αποτελεσματικές επιτυγχάνοντας συμπίεση που φθάνει το 24:1 με καλή ποιότητα ενώ οι μη απωλεστικές σπανίως ξεπερνούν το 4:1. Εντούτοις, στις σύγχρονες μεθόδους συμπίεσης εφαρμόζονται διάφορα ψυχοακουστικά μοντέλα αντίληψης βάση των οποίων απορρίπτονται οι ήχοι που έτσι και αλλιώς δεν μπορούν να γίνουν κατανοητοί από το ανθρώπινο αυτί.



Σχ. 4.1: Ψυχοακουστικό Μοντέλο Ακοής .[17]

Οι σημαντικότεροι μέθοδοι συμπίεσης ψηφιακών ηχητικών αρχείων στηρίζονται στο φαινόμενο της ηχητικής σκίασης (auditory masking), που αποσκοπεί στην ακουστικότητα των ήχων. Πιο συγκεκριμένα, όταν υπάρχει ένας ήχος συγκεκριμένης συχνότητας και έντασης, άλλοι ήχοι σε κοντινές συχνότητες δεν γίνονται αντιληπτοί από το ανθρώπινό αυτί. Παράλληλα, κατά την συμπίεσης των ψηφιακών ηχητικών δεδομένων απορρίπτονται οι συχνότητες που το ανθρώπινο αυτί δεν μπορεί να ξεχωρίσει μέσα σε ένα καθορισμένο διάστημα συχνοτήτων. Αν και χάνονται ορισμένες ηχητικές πληροφορίες εντούτοις η ποιότητα του ήχου παραμένει σε υψηλά επίπεδα (Σχ.4.2).

[17] Πηγή: Ψηφιακή Τεχνολογία Ήχου, Φλώρος Αντρέας.



Φαινόμενο Ηχητικής Σκίασης.[18]

Το φαινόμενο της ηχητική σκίασης, είναι άμεσα συνδεδεμένο τόσο με το πρότυπο AC-3 Dolby Digital (Η.Π.Α) όσο και με το Mpeg των οποίων οι διαφορές τους στηρίζονται στο τρόπο υλοποίησης τους. Εντούτοις και τα δύο πρότυπα συμπίεσης ηχητικών δεδομένων, δουλεύουν χωρίζοντας το φάσμα των ακουστικών συχνοτήτων σε υπομπάντες χρησιμοποιώντας περίπλοκους αλγόριθμους και ψυχοακουστικά μέσα για να απορρίψουν τις μη ακουστικές από το ανθρώπινο αυτί συχνότητες. Ο ανταγωνισμός μεταξύ των προτύπων AC-3 και Mpeg, φαίνεται να ευνοεί το μοντέλο AC-3 το οποίο στην πορεία θα αναδειχθεί ως το διεθνές πρότυπο για την συμπίεσης ψηφιακών ηχητικών δεδομένων.

Συμπερασματικά, όλες οι διαδικασίες συμπίεσης και αποσυμπίεσης υλοποιούνται μέσω κατάλληλων προγραμμάτων, όπου ο ρυθμός μετάδοσης των ψηφιακών δεδομένων είναι πολύ σημαντικός καθορίζοντας την ποιότητα του ήχου καθώς εξαρτάται τόσο από την συχνότητα δειγματοληψίας όσο και το μέγεθος του δείγματος. Για παράδειγμα, ο ήχος του μουσικού CD χρησιμοποιεί συχνότητες δειγματοληψίας 44.1 kHz με μέγεθος δείγματος 16bits παράγοντας ένα ρυθμό δεδομένων 1.4 Mbit/sec, ενώ αν μειωθεί η συχνότητα δειγματοληψίας τότε θα χαθούν οι υψηλές συχνότητες του ήχου. Επομένως, οι περισσότεροι σύγχρονοι αλγόριθμοι συμπίεσης που χρησιμοποιούν ψυχοακουστικό μοντέλο, στηρίζονται στη μεταβολή των bit για να επιτύχουν την μεγαλύτερη συμπίεση διατηρώντας υψηλή την ποιότητα του αναπαραγόμενου ήχου.

5.6.1 Πρότυπο MPEG

Οι προσπάθειες για μετάδοση ψηφιακού DAB (Digital Audio Broadcasting), ξεκίνησε στην Ευρώπη από το 1987 με το πρόγραμμα Eureka, στο οποίο συμμετείχαν ερευνητές τόσο από το Ινστιτούτο Fraunhofer με επικεφαλής τους karlheinz Brandenburg όσο και τα μέλη της ομάδας Moving Picture Expent Group (MPEG). Η MPEG, ως μια συνεργασία Πανεπιστήμιων,

[18] Δρ. Οικονόμου, Δάφνη ,*Ψηφιοποίηση Συλλογών,* 22/11/05

ερευνητών ινστιτούτων και εταιριών λειτουργεί στα πλαίσια του Διεθνούς Οργανισμού Τυποποίησης γνωστή σαν ISO/IEC με στόχο την ανάπτυξη διεθνών πρότυπων για την συμπίεση και αποσυμπίεση, την επεξεργασία και την κωδικοποιημένη αντιπροσώπευση της κίνησης των εικόνων, του ήχου και τους συνδυασμούς τους.

Το όνομα MPEG, έχει επικρατήσει όμως, να αναφέρεται και στη οικογένεια των τυποποιήσεων (standards) που δημιουργήθηκαν από την ομάδα Mpeg και χρησιμοποιούνται για την μετάδοση οπτικών και ηχητικών δεδομένων σε ψηφιακή συμπιεσμένη μορφή. Στην οικογένεια Mpeg, εντάσσονται τα standards Mpeg-1 που αφορά την συμπίεση ήχου και εικόνας, το Mpeg-2 για την εφαρμογή στην ψηφιακή τηλεόραση και το Mpeg-4 ως standard για εφαρμογές επικοινωνίας πολυμέσων. Επίσης. υπάρχει στα σχέδια τους και το Mpeg-7, με στόχο την αναπαράσταση περιεχομένου (content representation) για την αναζήτηση πληροφοριών σε εφαρμογές. Στο σημείο αυτό, οφείλουμε να επισημάνουμε ότι τα δύο τελευταία στάδια του Mpeg βρίσκονται σε υπανάπτυξη και δεν έχουν γίνει ακόμη στάνταρ ενώ το Mpeg 1 και 2 έχουν τεθεί ήδη σε εκτεταμένη εφαρμογή. Ενδιάμεσα, για αρκετό χρονικό διάστημα σε εξέλιξη υπήρξε το Mpeg 3 αλλά εγκαταλείφθηκε και ενσωματώθηκε ένα μέρος του στο Mpeg 2.

Πιο συγκεκριμένα, το πρότυπο που έγκειται το ψηφιακό ήχο είναι το Mpeg-1 Audio, ως το πρώτο διεθνές πρότυπο για την ψηφιακή συμπίεση ήχου υψηλής πιστότητας που δεν αποτελεί ένα αλγόριθμο συμπίεσης αλλά μια οικογένεια τριών διαφορετικών τεχνικών κωδικοποίησης και συμπίεσης. Και τα τρία αυτά στάδια στηρίζομαι στην ίδια αρχή, δηλαδή η συμπίεση ολοκληρώνεται με το συνδυασμό ενός είδος κωδικοποίησης μετασχηματισμού και sub-band division ενώ οι διαφορές του αναδύονται στο τελικό στάδιο της κβαντοποιήσης. Παράλληλα, το πρότυπο Mpeg-1 Audio προβλέπει ένα ή δύο ηχητικά κανάλια χρησιμοποιώντας 16bits για την κωδικοποίησης των δειγμάτων, ενώ η συχνότητα δειγματοληψίας του ήχου μπορεί να είναι 32kHz, 44kHz ή 48 kHz.

Όπως προαναφέρθηκε, το πρότυπο Mpeg-1 Audio στηριζόμενο στην κωδικοποίηση ψηφιακού ήχου διακρίνεται σε Mpeg-1 Audio Layer I ,II και III (ή MP3). Συγκεκριμένα, το Mpeg-1 Audio Layer I χρησιμοποιήθηκε στο σύστημα συμπίεσης ψηφιακής κασέτας DCC της Philips προσφέροντας συμπίεση 4:1. Ως αποτέλεσμα, η ηχητική ποιότητα είναι μέτρια ενώ το bandwidth που απαιτείται είναι αυξημένο 192 ή 256 kbps ανά κανάλι. Το Mpeg-1 Audio Layer II (ή Mp2), χρησιμοποιήθηκε στο ψηφιακό ραδιόφωνο όπου ο αλγόριθμος αυτή της κατηγόριας έχει βελτιοποιηθεί για ένα εύρος ζώνης 96 ή 128 kbps ανά μονοφωνικό κανάλι, ενώ ως αποτέλεσμα η ποιότητα είναι εφάμιλλη του CD (6:1...8:1 με 256...142 kbps για στερεοφωνικό ήχο) . Αντίθετα, το Mpeg-1 Audio Layer III ή αλλιώς Mp3 (βλ.παρακάτω), έχοντας καλύτερη απόδοση από τα παραπάνω παρουσιάζει συμπίεση περίπου 12:1 με ποιότητα που πλησιάζει αυτή των CD και ρυθμό μετάδοσης δεδομένων στα 64 kbps. Σήμερα, με βάση αυτό το πρότυπο είναι κωδικοποιημένα τα αρχεία ήχου Mp3 και τα οποία παρουσιάζονται τόσο για την μεταφορά όσο και για την φόρτωση μέσω διαδικτύου όσο και για την αναπαραγωγή ή ανάκληση από το σκληρό δίσκο. Στον πιο κάτω πίνακα 4.2.1, παρουσιάζεται ο λόγος συμπίεσης σε συνάρτηση με το ρυθμό μετάδοσης των δεδομένων που υποστηρίζει το κάθε στρώμα του προτύπου Mpeg-1 Audio.

Στρώμα	Λόγος Συμπίεσης
Mpeg-1 Audio I	4:1 με 384 kbps για στερεοφωνικό ήχο
Mpeg-1 Audio II	6:1 με 256-192 kbps για στερεοφωνικό ήχο
Mpeg-1 Audio III	12:1 με 128-112 kbps για στερεοφωνικό ήχο

5.6.2 Λόγος Συμπίεσης

Όπως έχει αναφερθεί παραπάνω, το πρότυπο Mpeg 1 μπορεί να κωδικοποιήσει μόνο δυο κανάλια ήχου ενώ για το δίκτυο που δεν διαθέτει μεγάλο εύρος ζώνης χρησιμοποιούνται κυρίως τεχνολογίες Mpeg 2. Το Mpeg 2 για τον ήχο, επεκτείνει την κωδικοποίηση μονοφωνικού και στερεοφωνικού ήχου του Mpeg 1, χωρίζεται σε τρία επίπεδα ανάλογα με το bit rate (ρυθμό δειγματοληψίας) που πρόκειται να υπάρξει ενώ κάθε επίπεδο δεν είναι καλύτερο από το άλλο, απλώς πιο περίπλοκο. Όσο προχωράμε στα επίπεδα τόσο πιο περίπλοκος γίνεται ο κωδικοποιητής και τόσο καλύτερη εκμετάλλευση του ρυθμού δειγματοληψίας πραγματοποιείται. Ο παρακάτω πίνακας, αναδύει αποτελέσματα τεστ σύγκρισης των τριών επιπέδων με κλίμακα από το 1 ως το 5.

Επίπεδο	Bit rate	Συμπίεση	Ελάχιστη καθυστέρηση	Ποιότητα 64 kbit
I	192 kbit	4:1	19ms	---
II	128 kbit	6:1	35ms	2.1 ως 2.6
III	64 kbit	12:1	59ms	3.6 ως 3.8

Αυτή την στιγμή το αγαπημένο της βιομηχανίας είναι τι επίπεδο II αφού όταν σχεδίαζαν τους εξοπλισμούς τους δεν είχαν ακόμη οριστικοποιηθεί το επίπεδο III. Όμως με ολοένα και αυξανόμενους ρυθμούς το επίπεδο III κερδίζει θέση με πρωταρχικό χώρο το διαδίκτυο. Για δομημένη ποιότητα ήχου Mpeg Layer III απαιτεί μικρότερο bit rate ή αλλιώς για δοσμένο bit rate πετυχαίνει υψηλότερη ποιότητα ήχου. Συμπερασματικά, το πρότυπο προσφέρει δειγματοληψία ελαττωμένη κατά το ήμισυ (16 kHz, 22kHz και 24kHz), παρέχοντας βελτιωμένη ποιότητα για ρυθμούς μετάδοσης σε 64 Kbit/sec.

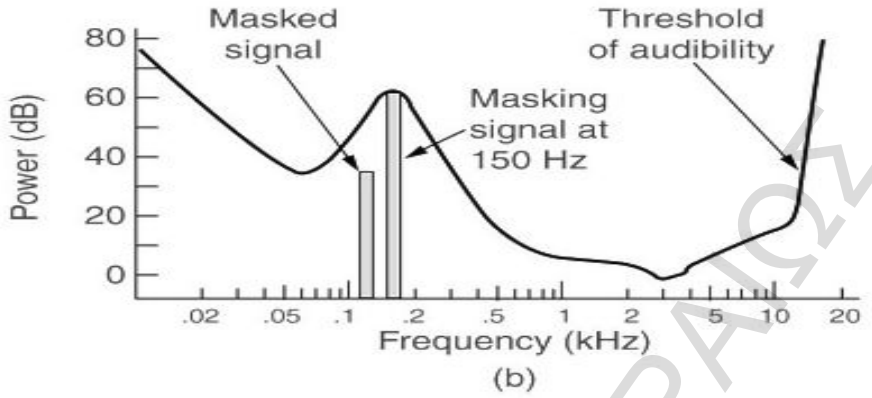
5.7 MP3

Το MP3 καταγράφηκε στην Ιστορία, ως μια από τις πιο θορυβώδες καινοτομίες την ψηφιακής μουσικής. Προκάλεσε χαρά στους μουσικόφιλους και πανικό στις δισκογραφικές εταιρίες, ταρακούνησε θεσμούς και αμφισβήτησε τα δεδομένα της εποχής τα οποία τελικά άλλαξαν άρδην. Η ψηφιακή μουσική είναι άρρηκτα συνδεδεμένη με το MP3, πράγμα που διαπιστώνει κανείς από το γεγονός ότι οι νέες συσκευές κατακλύζουν ριζικά και ραγδαία την αγορά.

Όπως αναφέρθηκε, το Mpeg Layer III ή MP3 όπως είναι παγκόσμια γνωστό έχει αναπτυχθεί από τους ερευνητές του Ινστιτούτου Fraughofer IIS με επικεφαλής, τους Karlheinz Branderburg και Dieter ως το πιο διαδεδομένο πρότυπο συμπίεσης ψηφιακού ήχου με στόχο την μεγαλύτερη συμπίεση και καλύτερη ποιότητα ήχου. Συγκεκριμένα, η συμπίεση των αρχείων ήχου μπορεί να φτάσει έως και 85%, χωρίς την ύπαρξη αισθητής διαφοράς στην ποιότητα απόδοσης ενώ αν η συμπίεση γίνει με ρυθμό μετάδοσης άνω των 128 Kbit/sec η διάφορά από το πρωτότυπο CD δεν είναι αντιληπτή. Αποτέλεσμα, η ύπαρξη μουσικών αρχείων 4-6 λεπτών που κανονικά θα καταλάμβαναν 40-70MB να καταλαμβάνουν χώρο μόνο 3-7MB.

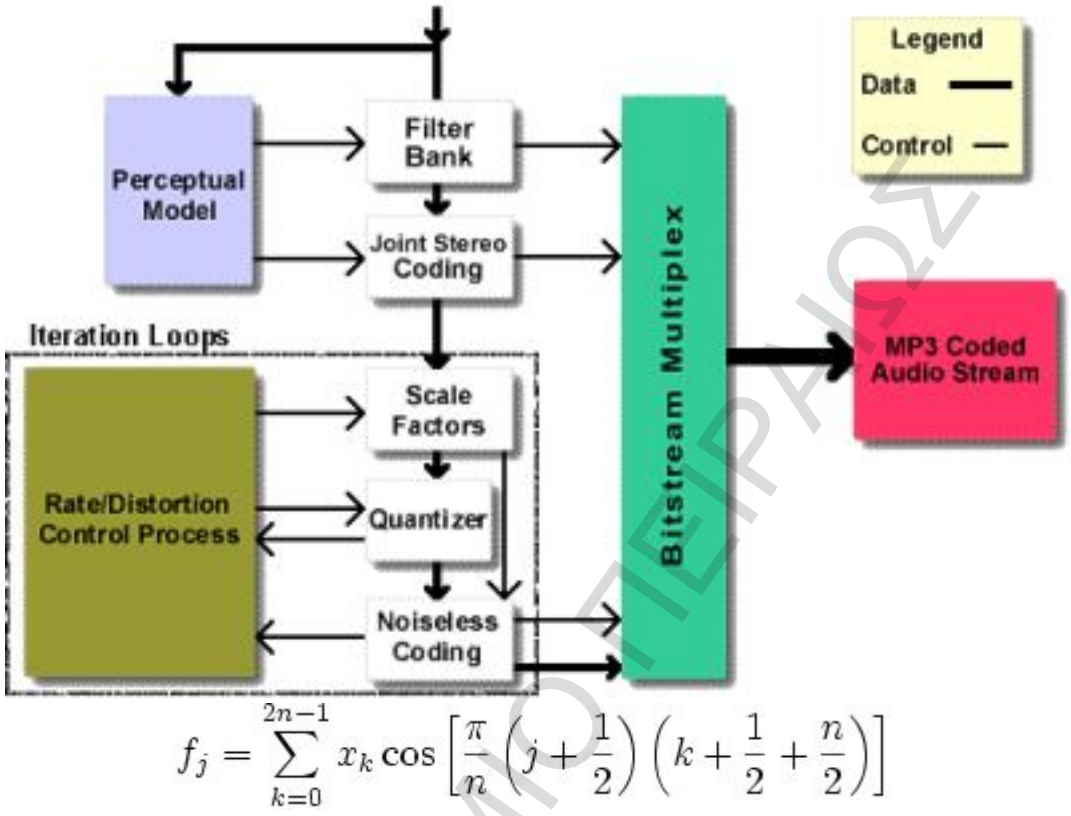
Οι codes MP3, χρησιμοποιούν ως βασικό μοντέλο ακοής αυτό που βασίζεται στις **καμπύλες κατωφλιού ακουστικότητας** (Minimal Audition Threshold) , δηλαδή την ελάχιστη ένταση που πρέπει να έχει ένας ήχος δια να τον ακούμε. Όσο μικρότερη είναι η ένταση, τόσο μικρότερη η ευαισθησία της ακοής και τόσο μεγαλύτερη στάθμη απαιτείται για να γίνει ακουστή μία συγκεκριμένη συχνότητα. Εμπειρικά αποτελέσματα, έχουν δείξει ότι το ανθρώπινο αυτί έχει μια περιορισμένη και εξαρτώμενη από την συχνότητα διακριτικότητας με επακόλουθο το κατώφλι ακουστικότητας να εξαρτάται από την ένταση του σήματος μέσα σε ένα περιορισμένο εύρος ζώνης γειτονικό αυτό της συχνότητας. Για κάθε συχνότητα του ακουστικού φάσματος το κατώφλι ακουστικότητας είναι διαφορετικό.

Ωστόσο, πολύ μεγαλύτερο ενδιαφέρον παρουσιάζει και το **φαινόμενο την επικάλυψης** (masking), με δεδομένους δυο γειτονικούς ήχους ο ισχυρότερος αλλοιώνει τοπικά την καμπύλη ακουστικότητας επικαλύπτοντας τον ασθενέστερο ο οποίος δεν γίνεται αντιληπτός από το ανθρώπινο αυτί. Έτσι το μοντέλο που χρησιμοποιείται στον codec MP3 υλοποιεί αυτό, δηλαδή υπολογίζει σε κάθε στιγμή το φασματικό περιεχόμενο του σήματος αποφασίζοντας ποιοι ήχοι επικαλύπτονται από ισχυρότερους με αποτέλεσμα να μην κωδικοποιηθούν. Σε κάθε χρονική στιγμή, ο codec MP3 έχει στη διάθεση του ένα αριθμό ψηφίων ο οποίος εξαρτάται από το βαθμό συμπίεσης που του έχει ζητηθεί, ενώ όσο εξελίσσεται η συμπίεση δεν χρησιμοποιούνται όλα τα ψηφία, ιδιαίτερα αν η στιγμιαία μορφή του σήματος είναι εύκολα συμπιεσμένη με συνεπακόλουθο την συντήρηση μιας δεξαμενής ψηφίων που απαιτούν μεγαλύτερη ακρίβεια στην κωδικοποίηση.



Φαινόμενο επικάλυψης. Ένας δυνατός ήχος, μεταβάλλεται τοπικά στην καμπύλη του κατωφλιού ακουστικότητας επικαλύπτοντας ένα γειτονικό αδύναμο ήχο.

Στο σημείο αυτό, οφείλουμε να αναφέρουμε ότι για καλύτερη συμπίεση δεδομένων το MP3 για την κωδικοποίηση των κβαντισμένων δειγμάτων χρησιμοποιεί την **εντροπική κωδικοποίηση (**entropy encoding**)**, η οποία κωδικοποιεί τα ψηφία που προκύπτουν από τα προηγούμενα στάδια. Η εντροπική κωδικοποίηση αποκαλείται και ως Huffan Coding. Συμπερασματικά, το ψυχοακουστικό μοντέλο για την συμπίεση ψηφιακού ήχου στην περίπτωση του MP3 τρέχει από το πεδίο του χρόνου σε αυτό της συχνότητας. Για υλοποίηση του περάσματος αυτού, χρησιμοποιείται μια πολύπλοκη μαθηματική διαδικασία κατά την οποία το υπό κωδικοποίησης μέρος του σήματος, υπόκεινται σε επεξεργασία από μια τράπεζα φίλτρων (filter bank) όπου και χωρίζει τι φάσμα σε 32 περιοχές και ακολουθεί ο μετασχηματισμός MDCT (Modified Discrete Cosine Transform), με στόχο την καλύτερη φασματική διακρικότητα. Η διαδικασία αυτή, στηρίζεται στην αρχή του μετασχηματισμού Fourier όπου και αναλύει ένα σήμα εξελισσόμενο στο χρόνο σε μια σειρά συνιστωσών στο πεδίο της συχνότητας.



$$f_j = \sum_{k=0}^{2n-1} x_k \cos\left[\frac{\pi}{n}\left(j+\frac{1}{2}\right)\left(k+\frac{1}{2}+\frac{n}{2}\right)\right]$$

Διάγραμμα βαθμίδων κωδικοποιητή MP3.

Μαθηματική περιγραφή του μετασχηματισμού MDCT.[19]

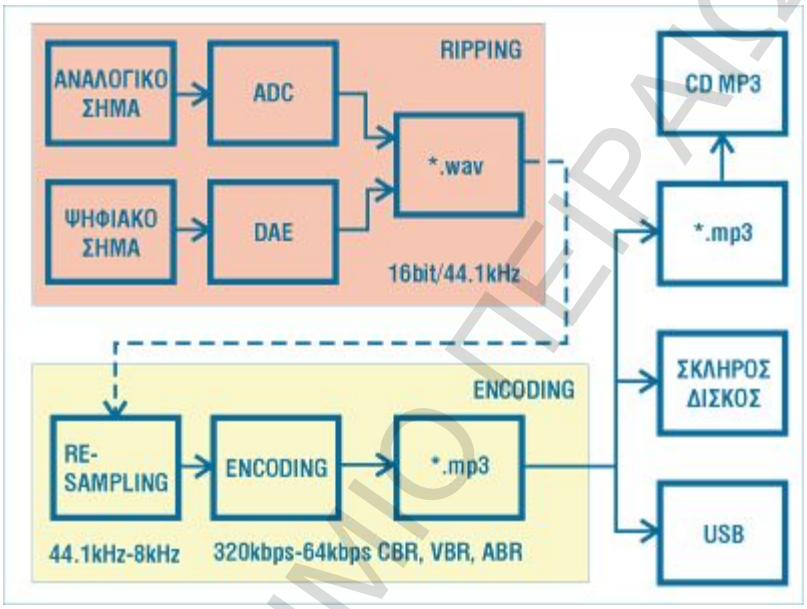
5.7.1 Δημιουργία και Αναπαραγωγή Αρχείου MP3 :

Πρώτου εστιάσουμε το ενδιαφέρον μας στην δημιουργία MP3, οφείλουμε να αναφέρουμε ότι σημαντικό χαρακτηριστικό τους είναι πως δεν έχουν όλα την ίδια συμπίεση, με αποτέλεσμα το ποσοστό συμπίεσης του κάθε αρχείου να καθορίζεται από εμάς. Όσο μεγαλύτερο είναι το ποσοστό συμπίεσης τόσο μικρότερο θα είναι το αρχείο, ενώ όσο περισσότερο συμπιέζεται ο ήχος τόσο περισσότερη πληροφορία χάνεται, γεγονός που επηρεάζει την ποιότητα του ήχου. Το μέγεθος και η ποιότητα του αρχείου ήχου ρυθμίζεται από το ροή μετάδοσης των δεδομένων.

Συγκεκριμένα, την δημιουργία ενός αρχείου MP3 υποδηλώνει η διαδικασία ripping, επιτρέποντας την δημιουργία ενός αρχείου wav που αποθηκεύεται στον υπολογιστή, από το πρωτογενές υλικό. Αν το υλικό αυτό είναι αναλογικό, θα πρέπει πρώτα να περάσει από έναν μετατροπέα A/D, αντίθετα αν το υλικό είναι αποθηκευμένο σε CD, τότε μπορούμε να χρησιμοποιήσουμε την διαδικασία Digital Audio Extraction (DAE) καθώς και να μεταφέρουμε το ψηφιακό περιεχόμενο του δίσκου απευθείας σε αρχείο wav. Το αρχείο wav που προκύπτει από τον ripper έχει προδιαγραφές που εξαρτώνται από την διαδικασία CD Quality, δηλαδή συχνότητα δειγματοληψίας 44.1kHz, και μήκος λέξης 16bit. (χωρίς να αποκλείονται και άλλες εκδοχές , όπως τα 48kHz ή τα 24bit). Αυτό το αρχείο, είναι η πηγή των δεδομένων που τοποθετείται στην είσοδο του encoder το οποίο πραγματοποιεί resampling (με βάση τις οδηγίες

[19] Πηγή: Fraunhofer IIS

που του δίνουμε), κωδικοποιώντας το σήμα με βάση το μοντέλο της απωλεστικής συμπίεσης με αποτέλεσμα την δημιουργία ενός αρχείο mp3 (Σχ.4.3.4)[20]. Το αρχείο αυτό, μπορεί να αποθηκευτεί τοπικά και να αναπαράγεται μέσω του σχετικού player, της κάρτας ήχου και των ηχείων του υπολογιστή, να μεταφερθεί σε κάποια εξωτερική συσκευή μέσω USB, να μετατραπεί και πάλι σε wav με απώτερο στόχο την εγγραφή του σε CD που είναι συμβατό με απλά CD players ή να εγγραφεί απ' ευθείας σε CD με στόχο να χρησιμοποιηθεί από συσκευές που είναι συμβατές με CD MP3.

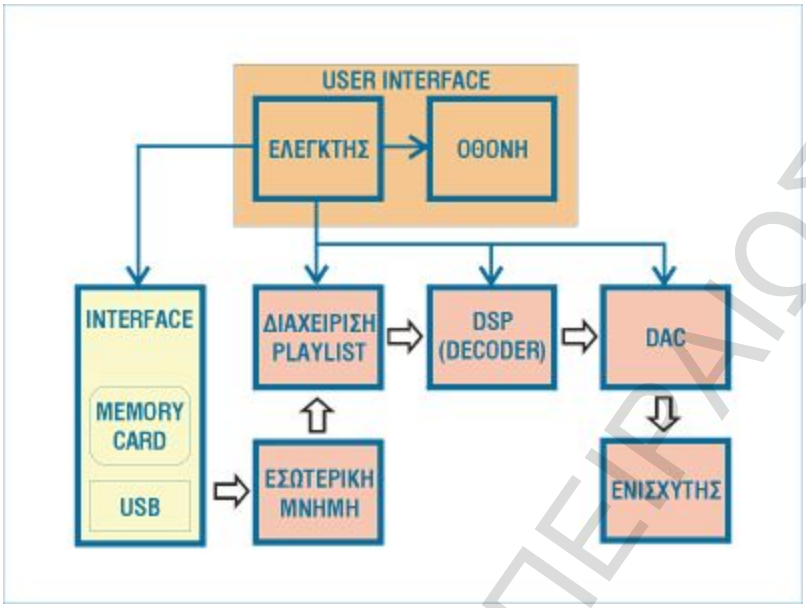


Σχ. 4.3.4: Διαδικασία Ripping και Encoding

Η διαδικασία αναπαραγωγή αρχείων MP3 παραγματοποείται όταν αυτά βρίσκονται αποθηκευμένα στον υπολογιστή. Καταρχήν τα αρχεία με κατάληξη mp3 μέσω των "File Associations", αναδύουν με διπλό κλικ το media player ούτος ώστε να μην χρειάζεται άλλη προσπάθεια από την πλευρά του χρήστη. Στην περίπτωση που τα αρχεία βρίσκονται στον υπολογιστή και που πρέπει να φορτωθούν σε κάποια εξωτερική συσκευή player, αυτό γίνεται χωρίς ειδικό λογισμικό αφού οι μνήμες και όλο και πιο συχνά τα players φαίνονται από την πλευρά του υπολογιστή ως "removable storage device" όταν συνδέονται στο USB. Αντίθετα, από την πλευρά του player τα πράγματα δεν είναι σύνθετα, ένα user interface παρέχει πρόσβαση τόσο στην δημιουργία όσο και διαχείριση ενός καταλόγου με τα περιεχόμενα του player είτε αυτά βρίσκονται σε εξωτερική είτε σε εσωτερική μνήμη. Στο σημείο αυτό, οφείλουμε να αναφέρουμε ότι η διαδικασία την διαχείριση είναι απλή ή σύνθετη ανάλογα με τις δυνατότητες της συσκευής, ενώ από την στιγμή που ο χρηστής επιλέξει το τρακ και στην συνέχεια το play, τα δεδομένα του αντίστοιχου αρχείου mp3 οδηγούνται, στον αποκωδικοποιητή, στον μετατροπέα d/a και στον ενισχυτή ακουστικών για περισσότερη επεξεργασία. Εντούτοις, πολλά players διαθέτουν και αναλογικές εισόδους έτσι ώστε ο ψηφιακός επεξεργαστής να διαθέτει μετατροπέα A/D και MP3 encoder. (Σχ.4.3.5)[21]

[20] Πηγή:iAudio

[21] Πηγή:iAudio



Διάγραμμα Βαθμίδων ενός MP3

Επίσης, καλό θα ήταν να τυπωθεί ότι, το πιο γνωστό και διαδεδομένο πρόγραμμα αναπαραγωγής μουσικών αρχείων MP3 είναι το Winamp της εταιρίας Nullsoft, το οποίο και διανέμεται δωρεά και είναι εγκατεστημένο σε εκατομμύρια υπολογιστές του κόσμου. Εκτός από το Winamp, μπορεί να βρει κανείς στο διαδίκτυο και άλλα προγράμματα αναπαραγωγής MP3 όπως τα Media Juke Box, Music Match, Sonigue, Ejay κ.α. Ο μεγαλύτερος όμως συναγωνιστής του Winamp είναι ο Media Player της Microsoft, ο οποίος διαχειρίζεται αποκωδικοποιητές MP3 του Ινστιτούτου Fraunhofer με καλύτερη ποιότητα ήχου, ενώ οι περισσότεροι mp3 player μπορούν να αναπαράγουν και άλλα πρότυπα ψηφιακής μουσικής όπως Wav, CD Audio, WMA κ.α , των οποίων ανάλυση γίνεται παρακάτω.

5.8 Άλλα πρότυπα

Παρά το γεγονός της η ριζική και ραγδαία εξέλιξη του MP3 το οποίο έχει γνωρίσει ευρεία αποδοχή μεταξύ χρηστών, εντούτοις, επικρατεί πληθώρα διαφορετικών τεχνολογιών που ολοένα και εξελίσσονται με πρωταρχικό στόχο την καλύτερη συμπίεση ψηφιακού ήχου. Τέτοιες προσπάθειες, έχουν υλοποιηθεί από την εταιρία Microsoft με το δικό της γνωστό και διαδεδομένο πρότυπο Windows Media Audio (WMA), το AAC (Advanced Audio Coding), το MP3 Pro, το OGG και το AC-3 Dolby Digital.

5.8.1 WMA (Windows Media Audio)

Η μεταφορά, η αποθήκευση και η χρήση ακουστικού υλικού με βάση την μορφή απωλεστικών συμπιεσμένων ηχητικών αρχείων μέσω υπολογιστή, ωθεί τον μεγαλύτερο κατασκευαστή λειτουργικών συστημάτων μα μην μείνει έξω από το παιχνίδι. Το πρότυπο Windows Media Audio (WMA) της εταιρία Microsoft, προσφέρει όμοιες δυνατότητες με το MP3, με άριστη ποιότητα τόσο αναπαραγωγής όσο και μεγαλύτερη συμπίεση (64 kbps). Πιο συγκεκριμένα, το WMA αποτελεί ένα σύστημα κωδικοποίησης/αποκωδικοποίησης ήχου, επιτρέποντας την συμπίεση ψηφιακών δεδομένων ήχου στο 1/20 του αρχικού τους όγκου και την εγγραφή τους σε ένα μόνο δίσκο CD με επακόλουθο τα τραγούδια που είναι προστατευμένα

να μην μπορούν να μεταδοθούν ελεύθερα. Συμπερασματικά, γι'αυτό ακριβώς το λόγω ο μεγαλύτερος αριθμός δισκογραφικών εταιριών χρησιμοποιεί στα πλαίσια υλοποίησης του έργου τους το πρότυπο αυτό.

5.8.2 MP3 Pro

Τον Ιανουάριο του 2001 στη CES, παρουσιάστηκε από την Coding Technologies η τεχνολογία Mp3 Pro, μια βελτιωμένη έκδοση του Mp3 με δυνατότητα να προσφέρει όμοια ποιότητα στο μισό μέγεθος των αρχείων, γεγονός που υλοποιείται με μεγαλύτερη συμπίεση δεδομένων. Συγκεκριμένα, η συμπίεση στα 64kbps και 96kbps, προσφέρει τη ίδια απόδοση ήχου με τα 128kbps και 192kbps του Mp3.

5.8.3 MP3 Surround

Το 2004 το Ινιστούτο Fraunhofer IIS παρουσίασε μία πολυκαναλική έκδοση του MP3, το MP3 Surround το οποίο βασίζεται στην τεχνολογία Binaural Cue Coding της Agere. Η τεχνολογία αυτή, επιτρέπει την μείξη σημάτων από πολλά κανάλια σε δύο, με στόχο την δημιουργία ενός σήματος συμβατού με τον απλό MP3 codec, ενώ κωδικοποιεί μία σειρά από παραμέτρους που περιγράφουν πλήρως το ηχητικό πεδίο surround. Τέτοιες παράμετροι είναι, οι χρονικές διαφορές μεταξύ των καναλιών, οι διαφορές στάθμης μεταξύ των καναλιών και η συσχέτιση μεταξύ των καναλιών.

5.8.4 AAC (Advance Audio Coding)

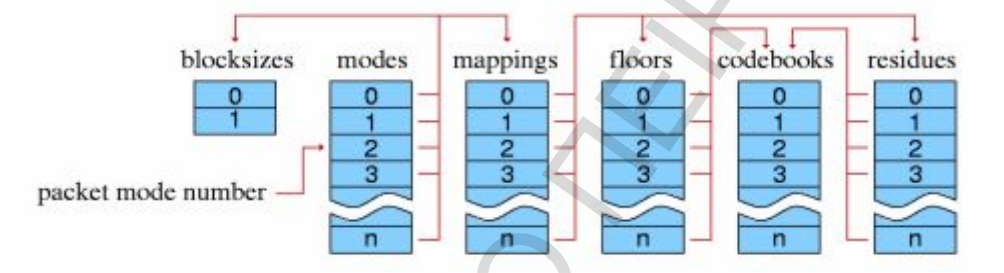
Το πρότυπο ACC αναπτύχθηκε για πρώτη φορά το 1997 από το Ινιστούτο Fraunhofer IIS και χρησιμοποιεί όπως και το MP3 το ψυχοακουστικό μοντέλο επικάλυψης, με σκοπό να καλύψει τα προβλήματα που υπήρχαν σχετικά με την ποιότητα των αρχείων MP3 στους μικρούς αριθμούς μετάδοσης των δεδομένων. Το πρότυπο ACC, ως ένα πραγματικό αριστούργημα κωδικοποίησης έχει την ικανότητα να αποδίδει εξαιρετικά υψηλή ποιότητα ήχου σε birate 64Kb/sec, επιτρέποντας τόσο την κωδικοποίηση 48 καναλιών ήχου και 16 καναλιών χαμηλής συχνότητας για εφέ όσο και την υποστήριξη πολλών γλωσσών ταυτόχρονα. Παράλληλα, το ACC διακατέχεται από τρεις διαφορετικές όψεις, την «κύρια», την «χαμηλής πολυπλοκότητα» και την «κλιμακούμενη συχνότητα δειγματοληψίας». Η «κύρια» όψη αναφέρεται σε εφαρμογές που η υπολογιστική ισχύει και εφαρμογές δεν είναι περιορισμένη, η «χαμηλής πολυπλοκότητα» σε εφαρμογές που η ισχύος και η μνήμη βρίσκονται σε μεγάλη ζήτηση, ενώ η τελευταία, είναι φτιαγμένη έτσι ώστε οι αποκωδικοποιητές να έχουν ελάχιστες απαιτήσεις σε μνήμη και ισχύ. Υποκειμενικά τεστ που έχουν πραγματοποιηθεί με καλά εκπαιδευμένους ακροατές, έδειξαν ότι η συγκεκριμένη κωδικοποίηση προσφέρει καλύτερη ποιότητα ήχου από οποιαδήποτε άλλη κωδικοποίηση ήχου με το μισό μόνο birate.Στο σημείο αυτό, οφείλουμε να αναφέρουμε ότι το πρότυπο ACC παρέχει καλύτερη απόδοση από το MP3, ενώ το 2003 η έκδοση του παρουσιάζεται συμβατή με τις προδιαγραφές του Mpeg ούτως ώστε το πρότυπο να αναφέρεται και ως Mpeg-4 ACC.

5.8.5 RA-Real Audio

Το Real Audio ως κλειστό πρότυπο, δημιουργήθηκε και υποστηρίχθηκε από την εταιρία Real Network με σκοπό την αναπαραγωγή ήχων στο Διαδίκτυο χωρίς να προηγείται κατέβασμα των ήχων στο σκληρό δίσκο του υπολογιστή. Το πρότυπο Real Audio, είναι αρκετά δημοφιλές και αυτό εξαιτίας της ελεύθερης διάθεσης του λογισμικού ανάγνωσης των αρχείων ήχου τέτοιου τύπου υποστηρίζοντας, μεγάλη συμπίεση και κατακανόνα χαμηλή ποιότητα ήχου.

5.8.6. OGG Vorbis

Ο codec Ogg Vorbis αναπτύχθηκε γύρω από το πρότυπο αρχείων Ogg και βασίζεται στη open source εφαρμογή απωλεστικής συμπίεσης με την ονομασία Vorbis. Ως προς τον τρόπο κωδικοποίησης, ο τρόπος μοιάζει με αυτό του Mp3 ενώ ταυτόχρονα ο Ogg Vorbis χρησιμοποιεί MDCT για τον μετασχηματισμό του σήματος από το πεδίο του χρόνου στο πεδίο της συχνότητας, καθώς και μία εναλλακτική μέθοδο επεξεργασίας του φάσματος, κατά την οποία κωδικοποιείται το φάσμα βάσης του οποίου η κατανομή είναι σχετικώς ομαλή και με περισσότερα ψηφία το απομένον φάσμα που η δομή και η χρονική εξέλιξη είναι πολύ πιο πολύπλοκη (σχ4.4.1)[22]. Συμπερασματικά, η τακτική αυτή σε συνδυασμό με την καλή ποιότητα ήχου που προσφέρει, ωθεί το πρότυπο Ogg σε ένα ανταγωνιστικό παιχνίδι ως προς το WMA και MP3.



Σχ. 4.4.1: Διάγραμμα βαθμίδων του κωδικοποιητή Ogg Vorbis.
Η κωδικοποίηση του φάσματος βάσης και του απομένοντος φάσματος.

5.8.7 AC3 Dolby Digital

Ένα από τα πιο διαδεδομένα πρότυπα για τον ψηφιακό πολυκάναλο ήχο είναι το AC3, που εκτός σημαντικού απροόπτου άρχισε να γίνεται το διεθνές πρότυπο για την συμπίεση ηχητικών δεδομένων. Στο ψηφιακό σύστημα ήχου AC3, ο ήχος κωδικοποιείται σε έξι συνολικά κανάλια στηριζόμενο στην μέθοδο 5.1. Συγκεκριμένα, υπάρχουν: (α) τρία κανάλια (αριστερό, κεντρικό, δεξί) που αποσκοπούν στο να φέρουν την βασική ηχητική πληροφορία, (β) δύο συνοδευτικά κανάλια περιβάλλοντος ήχου και (γ) ένα κανάλι για τις υπόλοιπες συχνότητες (σύστημα 3/2/.1). Ως προς τον τρόπο λειτουργίας τους, τα πέντε πρώτα κανάλια διαχειρίζονται συχνότητες ήχου στο διάστημα 3-20000Hz, ενώ το έκτο συχνότητες 3-120Hz. Συμπερασματικά, ο ρυθμός δειγματοληψία είναι 48KHz μεγαλύτερος από το ρυθμό των 44KHz που χρησιμοποιείται στα CDs, ενώ η συμπίεση των ηχητικών δεδομένων ανέρχεται στην αναλογία 10:1.

5.9 Τύποι Αρχείων Ψηφιακού Ήχου

Ανάλογα με την διαδικασία παραγωγής του, τα αρχεία ήχου διακρίνονται σε δύο κατηγορίες: τα αρχεία ήχων κυματομορφής και αρχεία MIDI. Τα αρχεία ήχων κυματομορφής, παράγονται με την διαδικασία της ψηφιοποίησης που σκοπό έχουν την παροχή ακριβής ψηφιακής εικόνας της κυματομορφής του ήχου. Πιο συγκεκριμένα, τα προγράμματα επεξεργασίας ήχων αυτής της κατηγορίας, αναπαριστάνουν τον ήχο ως κυματομορφή δίνοντας την ευκαιρία στο χρήστη να

[22] Πηγή: Τεχνολογίες Audio

κόψει, να αντιγράψει, να εισάγει και να τροποποιήσει είτε ένα μέρος είτε ολόκληρη την κυματομορφή. Οι βασικές δυνατότητες αυτών των εργαλείων είναι:

- εισαγωγή ήχου
- αντιγραφή, αποκοπή, επικόλληση ηχητικού τμήματος
- ρύθμιση έντασης και συχνότητας
- απαλοιφή θορύβου
- εισαγωγή διάφορων εφέ
- τροποποίηση επιπέδων κβάντωσης
- τροποποίηση χρονικής διάρκειας
- συμπίεση

Τα πιο γνωστά προγράμματα επεξεργασίας ήχων κυματομορφής είναι το Sound Forge Sonic της Foundry, το Sound Edit Macromedia, το Coll Edit Syntrillium, Wave Studio της Creative κ.α.

Η δεύτερη κατηγορία αρχείων ήχου σχετικά με την διαδικασία παραγωγής τους είναι το αρχείο MIDI, το οποίο δεν καταγράφει το ηχητικό σήμα αλλά τα στάδια που πραγματοποιούνται για να παραχθεί αυτό. Συγκεκριμένα, αντί να αποθηκεύσουμε τους ήχους μετατρέποντας το αναλογικό σήμα σε ψηφιακό αποθηκεύουμε τέσσερις αριθμούς για κάθε νότα που παίζεται, δηλαδή ο πρώτος μας δίνει ποια νότα είναι, ο δεύτερος μας λέει πιο όργανο την παίζει, ο τρίτος μας πληροφορεί πόσο δυνατά παίζεται η νότα και ο τέταρτος μας δίνει το χρόνο. Επειδή η διαδικασία αναπαράστασης μοιάζει με αυτή του κώδικα ASCII, πολλές φορές το MIDI αποκαλείται κα ως «κώδικα ASCII της μουσικής». Όσο αφορά τα προγράμματα επεξεργασίας αρχείων MIDI αυτά παρέχουν την δυνατότητα αναπαράστασης του ήχου ως παρτιτούρα καθώς και την δυνατότητα παροχής στο χρήστη, ώστε να τοποθετήσει νότες, παύσεις και άλλα σύμβολα της μουσικής σημειογραφίας πάνω στο πεντάγραμμο καθορίζοντας τον τρόπο εκτέλεσης της κάθε νότας.

Εν κατακλείδι, ανάλογα με την πλατφόρμα και το πρόγραμμα που χρησιμοποιείται για την παραγωγή των ψηφιακών ήχων, τα αρχεία αποθηκεύονται στον ηλεκτρονικό υπολογιστή με διαφορετική μορφή η οποία αναγνωρίζεται από τη έκταση του αρχείου .

.rif	RIFF	Αναπτύχθηκε από την Microsoft και υποστηρίζει αρχεία ψηφιακού ήχου WAV, MIDI.
.wav	Wave	Αποτελεί το πρότυπο αποθήκευσης ψηφιακού ήχου και είναι υποσύνολο του προτύπου RIFF.
.mid	MIDI	Διεθνές πρότυπο για την αποθήκευση μουσικών αρχείων MIDI.
.aif	AIFF	Δημιουργήθηκε από την Apple αλλά υποστηρίζεται και από άλλες πλατφόρμες, υποστηρίζει δειγματοληψία 32 bit.
.rmi	RMI	Αναπτύχθηκε από την Microsoft για την υποστήριξη αρχείων MIDI.
.mp3	MPEG-layer3	Πρότυπο συμπίεσης αρχείων ήχου που χρησιμοποιείται κυρίως για την διακίνηση αρχείων μουσικής στο Διαδίκτυο
.wma	WMA	Windows Media Audio από την Microsoft. Στόχο έχει την υποστήριξη απωλεστικής συμπίεσης
.ra	Real Audio	Προορίζεται για την άμεση αναπαραγωγή ήχων μέσω του Διαδικύου

Τύποι Αρχείων Ήχου.

5.10 Η διαδικασία της ψηφιοποίησης

Η μέχρι τώρα περιγραφή του ήχου που κάναμε, είναι η περιγραφή ενός αναλογικού σήματος. Όμως οι υπολογιστές δεν αναγνωρίζουν συνεχή σήματα, αλλά μόνο διακριτά. Πρέπει λοιπόν να βρεθεί κάποιος τρόπος ώστε το αναλογικό σήμα του ήχου να μετατραπεί σε διακριτό ψηφιακό σήμα. Η διαδικασία αυτή ονομάζεται δειγματοληψία και απαιτεί από πλευράς Η/Υ την παρουσία ειδικού υλικού (hardware) και κατάλληλου λογισμικού.

Το υλικό που πρέπει να υπάρχει οπωσδήποτε είναι ο μετατροπέας ADC (Analog to Digital Converter). Ο μετατροπέας αυτός συνήθως είναι τμήμα ενός ολοκληρωμένου κυκλώματος που επιτελεί και άλλες λειτουργίες σχετικά με την επεξεργασία ηχητικού σήματος και ονομάζεται DSP (Digital Signal Processor).

Η είσοδος του αναλογικού σήματος στον υπολογιστή γίνεται μέσω μικροφώνου ή αναλογικά ηχογραφημένου σήματος (line) στο ADC. Το ADC σαρώνει το εισαγόμενο σε αυτό αναλογικό τμήμα σε προκαθορισμένα χρονικά διαστήματα, μετράει το πλάτος του σήματος εκείνη τη στιγμή και το αποθηκεύει σε μορφή ψηφιακών δεδομένων. Η διαδικασία αυτή είναι η δειγματοληψία του σήματος και συνήθως γίνεται στις συχνότητες 8000 Hz, 11025 Hz, και 44100 Hz.

Αν το DSP ολοκληρωμένο είναι χωρητικότητας 8 bit αυτό σημαίνει πως μπορεί να παίρνει μέχρι 44100 δείγματα το δευτερόλεπτο και να τα αποθηκεύει σε πίνακες των 8 bits. Επειδή με 8 bits μεταβλητές μπορούν να περιγράφουν 255 διαφορετικές στάθμες, η ευκρίνεια του σήματος μας είναι οι παραπάνω 255 στάθμες κβάντωσης.

Οι συχνότητες δειγματοληψίας δεν έχουν τυχαίες τιμές. Κριτήριο για τον καθορισμό αυτών των τιμών αποτελεί το θεώρημα του Nyquist σύμφωνα με το οποίο:

Η μέγιστη συχνότητα αναλογικού σήματος που μπορεί να αποδοθεί χωρίς αλλοίωση ή παραποίηση (aliasing) πρέπει να είναι το μισό της συχνότητας δειγματοληψίας.

Αποδεικνύεται πως αν τα δείγματα μας έχουν παρθεί με μικρή συχνότητα δειγματοληψίας, θα υπάρχει πρόβλημα στην αναπαραγωγή των υψηλών συχνοτήτων και αδυναμία σωστού σχηματισμού του αναλογικού σήματος από τα δείγματα που έχουμε πάρει, άρα δεν θα μπορεί το σήμα να αποδοθεί επακριβώς.

Αν θεωρήσουμε πως το ανθρώπινο αυτί ακούει ήχους με μέγιστη φασματική διάταξη τα 20 KHz, τότε η συχνότητα δειγματοληψίας των 44100 Hz με δείγματα των 16 bits είναι αρκετή για να έχουμε ψηφιοποίηση στερεοφωνικής ποιότητας υψηλής πιστότητας. Αυτή η προδιαγραφή τυποποίησης είναι γνωστή ως ISO 10149 ή ως τυποποίηση του Κόκκινου Βιβλίου. Με αυτήν την τυποποίηση είναι ψηφιοποιημένοι οι ήχοι των μουσικών CD που κυκλοφορούν στο εμπόριο.

Ακόμα πιο ψηλά σε συχνότητα δειγματοληψίας βρίσκεται η τυποποίηση για τις ηχογραφήσεις ψηφιακών κασετοφώνων (Digital Audio Tape), που θέλει συχνότητα δειγματοληψίας 48000 Hz σε δείγματα των 16 bits.

Αυτές οι κωδικοποιήσεις είναι ακριβές σε αποθηκευτικό χώρο. Για παράδειγμα, για να αποθηκεύσουμε 1 λεπτό στερεοφωνικής μουσικής θα χρειαζόμασταν κατά το πρότυπο του Κόκκινου Βιβλίου:

(2 * fs * bits ανά δείγμα) / 8 * T = 2 * 44100 * 16 / 8 * 60 = 10,1 MB !

Για να περιορίσουμε το μέγεθος του ψηφιοποιημένου σήματος χρησιμοποιούμε όσο το δυνατόν μικρότερες συχνότητες δειγματοληψίας.

Για παράδειγμα για τη δειγματοληψία ανθρώπινης φωνής από το τηλεφωνικό δίκτυο, θα αρκούσε συχνότητα δειγματοληψίας των 8000 Hz σε δείγματα των 8 bits, μια και το τηλεφωνικό δίκτυο αποκόπτει τις συχνότητες άνω των 3400 Hz.

Όσο για την ψηφιοποίηση της ανθρώπινης φωνής, δειγματοληψία στα 22050 Hz θα κρινόταν ως ικανοποιητική μια και η οξύτερη γυναικεία φωνή δεν ξεπερνά τα 10000 Hz.

5.11 Η κωδικοποίηση του ήχου στα λειτουργικά περιβάλλοντα

Η διαδικασία ψηφιοποίησης αφορά κυρίως το υλικό της κάρτας ήχου ενός υπολογιστή. Η μορφή αποθήκευσης των δειγμάτων και η επεξεργασία τους αφορά το λειτουργικό σύστημα.

Υπάρχουν αρκετοί τρόποι μετάφρασης των ψηφιοποιημένων δειγμάτων σε αρχεία ήχου. Η πιο διαδεδομένη πλέον μορφή είναι η RIFF (Resource Interchange File Format). Σύμφωνα με αυτήν τα δείγματα αποθηκεύονται σε ομάδες (chunks) κατά τη δομή:

```
Typedef unsigned long DWORD;
Typedef unsigned char BYTE;
Typedef DWORD FOURCC;
Typedef struct {
FOURCC ckID;
DWORD ckSize; // Chunk size
BYTE ckData(ckSize); // Chunk data array
} CK;
```

Έτσι η Microsoft για παράδειγμα έχει μια δική της κωδικοποίηση κατά το πρότυπο RIFF που ονομάζεται WAVE μορφή και τα αρχεία της φέρουν την επέκταση wav.

Στην WAVE μορφή κωδικοποίησης χρησιμοποιούνται συχνότητες δειγματοληψίας 11025 Hz, 22050 Hz και 44100 Hz με δείγματα των 8 ή 16 bits.

5.12 Τα ηχητικά γεγονότα στα λειτουργικά περιβάλλοντα

Όλες οι πλατφόρμες των σύγχρονων λειτουργικών περιβαλλόντων έχουν ενσωματώσει τον ήχο ως στοιχειώδη δυνατότητα του τρόπου λειτουργίας τους. Αυτό σημαίνει πως η αντίδραση του λειτουργικού συστήματος σε κάποιο γεγονός (event) ή σε κάποια δυσλειτουργία, μπορεί να γίνει εκτός από τη συνήθη μέθοδο των οπτικών σημάτων (message boxes) με ήχους που εκφράζουν απορία, αποτροπή, επιδοκιμασία κ.α.

Στο περισσότερο διαδεδομένο λειτουργικό περιβάλλον, στα Windows έχουμε ήχους με ονομασίες όπως τα Chimes, Chord, Ding και Tada.

Στο Ms-Dos περιβάλλον δεν έχουμε υποστήριξη ήχου από το λειτουργικό περιβάλλον και συνεπώς κάθε κατασκευαστής καρτών ήχου έχει το δικό του τρόπο κωδικοποίησης. Αυτό που έχει όμως επικρατήσει ως πρότυπο (industry standard) είναι η συμβατότητα με την Soundblaster της εταιρίας Creative.

Ας σημειωθεί πως για επαγγελματικές εφαρμογές το περιβάλλον του Dos δεν ενδείκνυται, αλλά η μεγαλύτερη εγκατεστημένη βάση ηχητικών σημάτων σε υπολογιστές αφορά αυτό το περιβάλλον για έναν πολύ απλό λόγο: τα καλύτερα και περισσότερο διαδεδομένα παιχνίδια σε Η/Υ παραμένουν εγκατεστημένα σε λειτουργικό περιβάλλον του Dos.

Κάτι ανάλογο ισχύει και στις πλατφόρμες των πολυμέσων στο λειτουργικό σύστημα Unix. Οι διάφορες εταιρίες που πωλούν workstations με Unix υποστηρίζουν διαφορετικές μορφές αρχείων. Αν και υπάρχουν προγράμματα μετατροπής από το ένα είδος κωδικοποίησης ηχητικών σημάτων στο άλλο, δεν υπάρχει κάποια ενιαία μορφή κωδικοποίησης που να υποστηρίζεται από όλους. Πάντως ξεχωρίζει η πλατφόρμα της Silicon Graphics στα

workstations για την πολύ καλή υποστήριξη σε επεξεργασία ήχου για πολυμεσικές εφαρμογές σε επίπεδο λειτουργικού συστήματος.

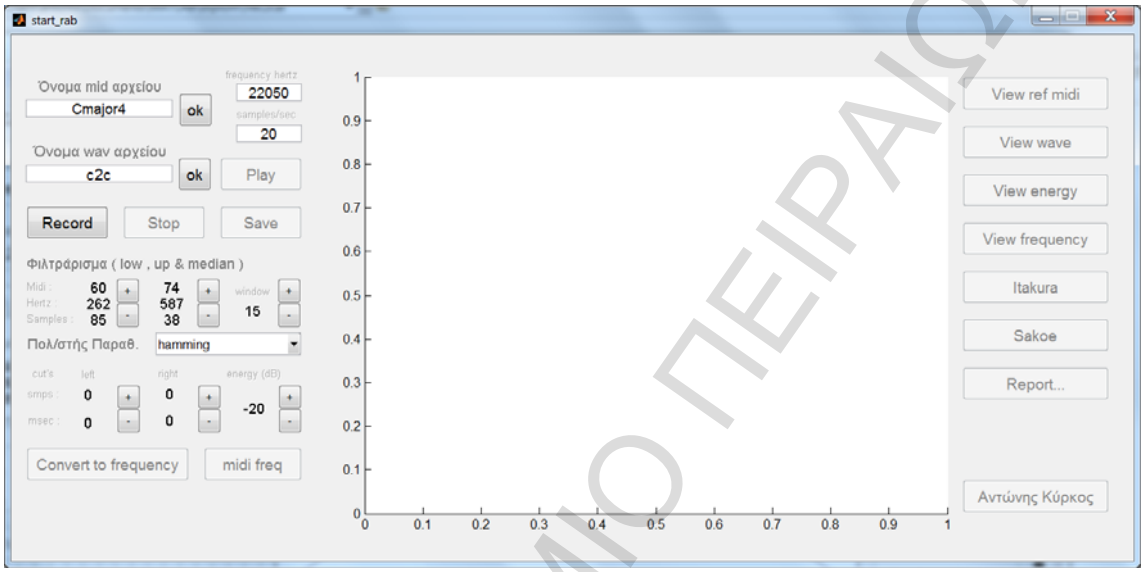
Από την άλλη μεριά στο MacOs των Macintosh έχουμε εδώ και αρκετά χρόνια υποστήριξη ηχητικών σημάτων και ρυθμίσεων. Οι macintosh άλλωστε πωλούνται με κάρτα ήχου στον βασικό εξοπλισμό τους. Γνωστά ηχητικά σήματα κυκλοφορούν με τα ονόματα Droplet, Indigo, Quack, Simple Beep, Sosumi, Wild Eep.

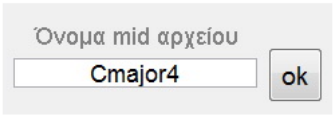
6. Εργασία

6.1 Το GUI στο MATLAB

Με την χρήση του matlab παράγουμε ένα gui για να μπορέσουμε να επεξεργαστούμε καλύτερα τα δεδομένα μας.

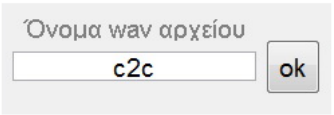
Ας δούμε τι δυνατότητες μας δίνει αυτό το εργαλείο και πως μπορούμε να το χειριστούμε.

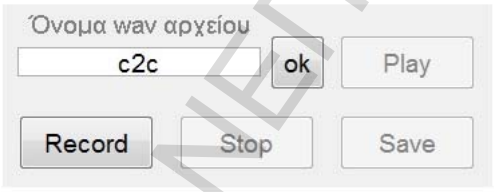




Εισαγωγή του αρχείου midi. Μετά την εισαγωγή η επικεφαλίδα αλλάζει αυτόματα από [Όνομα mid αρχείου] σε [play] και μπορούμε πατώντας με το ποντίκι πάνω στη λέξη play να ακούσουμε το midi αρχείο.

Εισαγωγή εγγραφής από ένα είδη υπάρχων αρχείο wav.



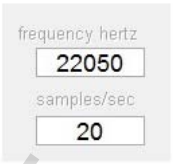


Εκτός από την εισαγωγή ενός υπάρχον αρχείου μπορούμε με την χρήση του μικροφώνου του υπολογιστή μας να κάνουμε απευθείας την δική μας ηχογράφηση.

Με το πάτημα του [Record] ενεργοποιείται το [Stop] και στην συνέχεια μόλις πατήσουμε για να τελειώσει η εγγραφή μας ενεργοποιούνται τα [Play] και [Save] κουμπιά με τις αντίστοιχες λειτουργίες.

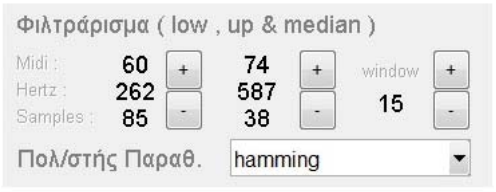
Κατά την λειτουργία της αποθήκευσης save, το αποθηκευμένο αρχείο παίρνει το όνομα που βρίσκεται στο textbox, όπως στην εικόνα ‘c2c.wav’

Μπορούμε να δούμε την συχνότητα δειγματοληψίας που χρησιμοποιούμε και τα πόσα παράθυρα επεξεργασίας έχουμε ανά δευτερόλεπτο. Μπορούμε να διορθώσουμε και να βάλουμε όποια συχνότητα ή δείγματα επιθυμούμε.



Παράδειγμα βλέπουμε στην εικόνα ότι έχουμε συχνότητα 22KHz και 50msec παράθυρο επεξεργασίας (50ms αντιστοιχεί με 20 δείγματα στο δευτερόλεπτο)

Καλό είναι τα πιο πάνω νούμερα να τα επιλέξουμε πριν το φόρτωμα αρχείων ή την εγγραφή για να έχουμε συμβατότητα μεταξύ των σειρών προς σύγκριση.



Στην εικόνα δίπλα έχουμε τις επιλογές αυτές που μπορούμε να περιορίσουμε την μετατροπή των δειγμάτων σε συχνότητα μέσα στα όρια συχνοτήτων που επιλέγουμε. Αυτό το κάνουμε για να περιορίσουμε τυχών λάθη μετατροπής. Η επιλογή γίνεται βάση της συχνότητας από της νότες από την συγκερασμένη κλίμακας της μουσικής. Για ευκολία μας βλέπουμε για κάθε νότα τον κωδικό σε midi, την συχνότητα σε Hz και τα πόσα δείγματα χρειάζονται για να έχουμε ένα πλήρες ημίτονο στην αντίστοιχη συχνότητα.

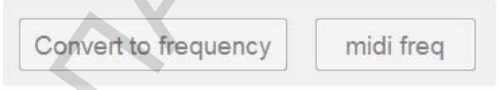
Στη συνέχεια επιλέγουμε τα πλαίσια για ένα median φίλτρο και τον τύπο του πολλαπλασιαστή παραθύρου (πχ hamming, hanning κ.α.)

Με τα εργαλεία της διπλανής εικόνας μπορούμε ψαλιδίσουμε την αρχή και το τέλος της ηχογράφησής μας με τρόπο τέτοιο ώστε να κρατήσουμε το επιθυμητό κομμάτι από την ηχογράφησή μας.

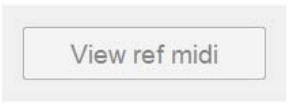
Μας δίνετε η δυνατότητα να ‘κόψουμε’ από αριστερά (αρχή) ή δεξιά (τέλος) ή ακόμα με βάση ενός υπόβαθρου ενέργειας. Οι επιλογές γίνονται πατώντας με το ποντίκι πάνω στις λέξεις και στα κουμπιά ανάλογα με το τι επιθυμούμε. Πατώντας πάνω στα {left} , {right} , {smps} , {msec} η αντίστοιχη λέξη γίνεται bold και μας δίνει τον έλεγχο να επηρεάσουμε αντίστοιχα νούμερα. Με το πάτημα στο {cut} κόβουμε και αντίστοιχα την εγγραφή.

Εδώ οι επιλογές που έχουμε είναι σε συνδυασμό και με τα κουμπιά επιλογής της εμφάνισης που θα δούμε πιο κάτω και με το πλαίσιο που παρατηρούμε τις κυματομορφές το οποίο είναι ενεργό και σε οποιοδήποτε πάτημα με το ποντίκι μας δίνει στοιχεία για το σημείο που πατήσαμε.

Παράδειγμα όταν έχουμε φορτώσει ένα wav και βλέπουμε την κυματομορφή, πατώντας πάνω στο {left} και στη συνέχεια πάνω σε κάποιο σημείο στην κυματομορφή, μας φέρνει τα νούμερα που αντιστοιχούν για τα δείγματα και αντίστοιχα msec, κάτω από την επιλογή {left}. Αν τώρα πατήσουμε το {cut} θα κόψει το σήμα από αριστερά μέχρι το σημείο που του υποδείξαμε κτλπ.



Οι επιλογές που βλέπουμε στην διπλανή εικόνα ενεργοποιούνται η εναλλάξ μία μετά την άλλη και μας δίνουν την δυνατότητα να μετατρέψουμε τα δείγματα της ηχογράφησης μας σε συχνότητα σε συνδυασμό με τις παραπάνω επιλογές. Στη συνέχεια την σειρά δεδομένων από συχνότητες να την κβαντίσουμε πάνω στις συγκερασμένες συχνότητες από τις νότες της μουσικής.



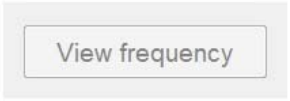
Επιλογή προβολής του midi αρχείου.

Το κουμπί ενεργοποιείται μετά το φόρτωμα midi.

Επιλογή προβολής του wav αρχείου ή της εγγραφής

Επιλογή προβολής της ενέργειας του wav αρχείου ή της εγγραφής

Τα κουμπιά [View wave] & [View energy] ενεργοποιούνται μετά το φόρτωμα wav αρχείου

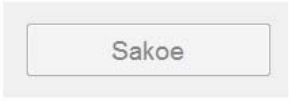


Επιλογή προβολής των συχνοτήτων από τα δείγματα

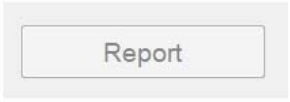
Το κουμπί ενεργοποιείται μετά την μετατροπή δειγμάτων σε συχνότητα

Όλες οι επιλογές προβολής επηρεάζουν το κεντρικό παράθυρο προβολής και το εργαλείο κοπής αρχής και τέλους

Επιλογή έναρξης των αλγόριθμων σύγκρισης του Itakura και Sakoe αντίστοιχα



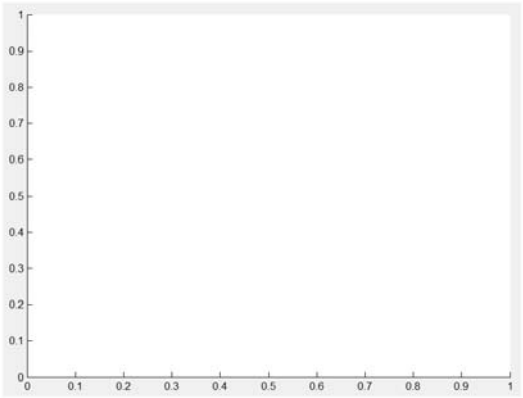
Τα κουμπιά ενεργοποιούνται μετά την μετατροπή σε midi συχνότητες



Η επιλογή για την έκδοση αποτελεσμάτων στο matlab.

Η επιλογή αυτή ενεργοποιείται μετά τον αλγόριθμο του Sakoe.

Τέλος το κεντρικό παράθυρο πολλαπλής προβολής δεδομένων.



7. Πρόγραμμα

7.1 Σειρά Εκτέλεσης

Η Σειρά ενεργειών που εκτελούμε κατά το χειρισμό του εργαλείου μας είναι η εξής

- Από ένα αρχείο midi, όπου το θεωρούμε ως πρότυπο, εξάγουμε ένα μονοδιάστατο πίνακα όπου κάθε στοιχείο του, αντιστοιχεί στην συχνότητα (σε Hz) της νότας και για την αντίστοιχη βάση χρόνου που κάνουμε την μετατροπή.

  Παράδειγμα αν στο midi έχουμε ένα μέσο λά που είναι παιγμένο για 1s και έχουμε παράθυρο 100ms η μετατροπή θα μας δώσει ένα πίνακα με 10 στοιχεία (10x100ms=1s) με τιμή στα 440 (λά → 440Hz)
- Από ένα δεύτερο αρχείο (midi ή ηχογραφημένο), που είναι το παίξιμο της αντίστοιχης με το midi-πρότυπο μελωδίας, διαβάζουμε τα στοιχεία τα φιλτράρουμε και τα εναρμονίσουμε με την βάση χρόνου που επιλέγουμε, με τρόπο τέτοιο ώστε να έχουμε ένα πίνακα συγκρίσιμο με αυτόν που προέρχεται από το πρότυπο.
- Χρησιμοποιούμε έναν αλγόριθμο του Rabinner για την εύρεση αρχής και τέλους της ηχογράφησης, με την δυνατότητα να επέμβουμε στην επεξεργασία του σήματος και να το διαμορφώσουμε κατά βούληση.
- Χρησιμοποιώντας δύο αλγόριθμους σύγκρισης στοιχείων βάση προτύπων αναφοράς των Itakura και Shakoe, έτσι ώστε να πάρουμε ένα αποτέλεσμα ανάλογο της ομοιότητας των δύο σειρών δεδομένων.
- Τέλος απεικονίζουμε τα αποτελέσματα και παράγουμε συμπεράσματα για την ομοιότητα του παιξίματος με το πρότυπο.

7.2 Εισαγωγή Στοιχείων - Εγγραφής

Με το εργαλείο μας στο matlab έχουμε την δυνατότητα να εισάγουμε τα προς σύγκριση αρχεία.

Αρχικά φορτώνουμε το πρότυπο από ένα midi αρχείο το οποίο επιλέγουμε και εισάγεται στο εργαλείο μας σε μορφή ενός δυσδιάστατου πίνακα.

Από το midi αυτό κρατάμε τη midi νότα, την έναρξη της και την διάρκεια της. Με τα δεδομένα αυτά μετατρέπουμε σε ένα πίνακα συχνοτήτων βάση του πιο κάτω τύπου και αναπαράγουμε τόσα δείγματα όσα η διάρκεια της αντίστοιχης νότας.

$$συχνότητα(Hz) = \left(\frac{440}{32}\right) 2^{\left(\frac{midi-9}{12}\right)}$$

Στη συνέχεια μπορούμε να ηχογραφήσουμε ένα δοκιμαστικό, όπου μπορούμε να το αποθηκεύσουμε στον υπολογιστή μας ή να το φορτώσουμε απευθείας από ένα υπάρχων wav αρχείο.

7.3. Εύρεση Αρχής και Τέλους ηχογράφησης

Κατά την διάρκεια του φορτώματος ενός δοκιμαστικού εκτελείτε ένας αλγόριθμος του Rabiner όπου μας αναγνωρίζει την αρχή και το τέλος της ηχογράφησης.

Ο αλγόριθμος αυτός εξετάζει την ενέργεια του σήματος και την εναλλαγή και πέρασμα από το μηδέν (zero crossing rate).

Η ενέργεια του σήματος υπολογίζεται βάση του τύπου

$$E_r = \sum_{m=0}^{L-1} (s[rR + m]w[m])^2$$

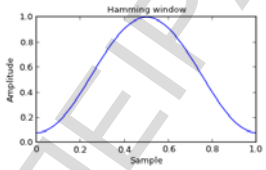
Όπου E_r η ενέργεια του ηχογραφημένου σήματος,

s[n] το ηχογραφημένο σήμα,

L μέγεθος πλαισίου,

R ολίσθηση πλαισίου,

w[n] παράθυρο Hamming, L σημείων.



Η ενέργεια του σήματος υπολογίζεται στη λογαριθμική κλίμακα και κανονικοποιείται στη μέγιστη τιμή των 0dB

$$\hat{E}_r = 10 \log_{10} E_r - \max(10 \log_{10} E_r)$$

Ο ρυθμός διέλευσης από το μηδέν υπολογίζεται

$$Z_r = \frac{R}{2L} \sum_{m=0}^{L-1} |sgn(s[rR + m]) - sgn(s[rR + m + 1])|$$

Όπου Z_r ο ρυθμός διέλευσης από το μηδέν

$sgn(x)$ $\begin{cases} -1 : x < 0 \\ 0 : x = 0 \\ 1 : x > 0 \end{cases}$

Ακόμα θα χρειαστούμε να υπολογίσουμε την μέση τιμή και την τυπική απόκλιση του λογαρίθμου της ενέργειας και του ρυθμού διέλευσης από το μηδέν.

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

Όπου $\bar{x}$ η μέση τιμή,

x_i η σειρά δεδομένων,

$$s' = \sqrt{s^2}$$

$$s^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2$$

Όπου s' η τυπική απόκλιση.

Ξεκινώντας την διαδικασία του αλγόριθμου εκτελούμε τις παρακάτω ενέργειες:

- Αρχικά το ηχογραφημένο σήμα το μετατρέπουμε σε 10KHz ρυθμό δειγματοληψίας για αυτόν τον αλγόριθμο.
- Εκτελούμε ένα ανωπερατό φίλτρο FIR, 101 σημείων, σταθερής κυμάτωσης (equiripple) για την εξάλειψη τυχών DC συνιστώσας (offset).
- Εξάγουμε τη ενέργεια και το ρυθμό διέλευσης από το μηδέν με επεξεργασία βραχέως χρόνου, χρησιμοποιώντας μέγεθος πλαισίου 40msec και ολίσθηση 10msec. Δηλαδή έχουμε L=400 δείγματα και R=100 δείγματα για ρυθμό πλαισίων 100 πλαίσια/sec, F_s=10000Hz .
- Κανονικοποιούμε την ενέργεια στα 0dB.
- Θεωρώντας ότι τα πρώτα 200ms (20 πλαίσια) δεν περιέχουν καθόλου μουσικό σήμα, υπολογίζουμε την μέση τιμή και την τυπική απόκλιση για της ενέργειας και του ρυθμού διέλευσης από το μηδέν για αυτά τα πρώτα πλαίσια, όπου μας δίνουν μια χονδρική στατιστική εκτίμηση για το υπόβαθρο θορύβου του σήματός μας.

  Όπου e_{avg} z_{cavg} μέσες τιμές και e_{sig} z_{csing} τυπικές αποκλίσεις
- Υπολογίζουμε το όριο του ρυθμού διέλευσης από το μηδέν

  Iz_{ct}=max (IF, z_{cavg} + 3 z_{csing})

  Όπου IF αποτελεί ένα γενικό κατώφλι για την ανίχνευση μη έμφωνων πλαισίων (για την εφαρμογή μας παίρνει την τιμή 10).
- Υπολογίζουμε το κατώφλι της ενέργειας

  Itu=Σταθερά στα -18dB

  Itr=max (Itu-10 , e_{avg} + 3 e_{sig})
- Ξεκινώντας από την αρχή αναζητάμε ένα πλαίσιο b1 όπου ο λογάριθμος της ενέργειας υπερβαίνει το κατώφλι Itr.
- Ελέγχουμε τα γειτονικά πλαίσια ώστε η ενέργεια να μην πέφτει κάτω από το όριο Itu. Αν η αναζήτηση αποτύχει ξανά αναζητάμε ένα νέο b1 από το πλαίσιο που βρισκόμαστε και προς το τέλος.
- Από το τελικό b1 αναζητάμε τα 25 προηγούμενα πλαίσια και μετράμε πόσα πλαίσια ξεπερνάνε το όριο Iz_{ct} . Εάν αυτά είναι περισσότερα από 4 τότε το αρχικό πλαίσιο μεταφέρεται στο πλαίσιο με τον χαμηλότερο δείκτη που ξεπερνά το όριο
- Αναζητάμε τώρα ένα πλαίσιο e1 με την ίδια λογική όπως τα τρία προηγούμενα βήματα αλλά αυτή την φορά η αναζήτηση είναι από το τέλος προς την αρχή και αντίστοιχα στα 25 συνεχόμενα πλαίσια για την τροποποίηση του δείκτη.

Με τον τρόπο αυτό διώχνουμε τα δείγματα του σήματος αρχής και τέλους που δεν περιέχουν πληροφορία και μας εισάγουν λάθη στην επεξεργασία του σήματος.

Ο αλγόριθμός αυτός βασίζεται στον αντίστοιχο αλγόριθμο του Rabiner για την αναγνώριση αρχής και τέλους μιας ομιλίας (Ψηφιακή Επεξεργασία Φωνής Θεωρία και εφαρμογές) όπου τα όρια έχουν τροποποιηθεί για το περιβάλλον ηχογράφησης της εφαρμογής μας.

7.4 Μετατροπή Δειγμάτων σε συχνότητα

Για την μετατροπή των δειγμάτων χρησιμοποιούμε την επεξεργασία βραχέος χρόνου για την εξαγωγή της θεμελιώδους συχνότητας όπου τα όρια μετατροπής τα εισάγουμε από το interface.

Για την εξαγωγή των της θεμελιώδους συχνότητας χρησιμοποιούμε αρχικά τον τύπο

$$Amdf(t) = \frac{1}{N-t-1}\sum_{n=0}^{N-t-1}|x(n)-x(n+t)|$$

Όπου $x(n)$ το ηχογραφημένο σήμα μετατοπισμένο κατά N,

t το offset της μετατόπισης,

και στην συνέχεια εξάγουμε τις περιόδους από τον τύπο

$$T_p = \underset{t}{arg}\ \underset{t=t_{min}}{\overset{t_{max}}{MIN}}\ (Amdf(t))$$

Η περίοδος που αντιστοιχεί στη θεμελιώδη συχνότητα είναι ο δείκτης του πίνακα Amdf όπου παρουσιάζει ελάχιστο στο περιεχόμενό του.

Οπότε στη συνέχεια υπολογίζουμε την συχνότητα

$$F_r = \frac{F_s}{T_p}$$

7.5 Αλγόριθμοι Σύγκρισης

Για να μπορέσουμε να συγκρίνουμε δύο σειρές στοιχείων χρησιμοποιούμε δύο αλγόριθμους δυναμικού προγραμματισμού, που αναπτύχθηκαν από τους Fumitada Itakura - Shuzo Saito και Sakoe Chiba.

Η διαδικασία των αλγόριθμων σύγκρισης είναι η εξής

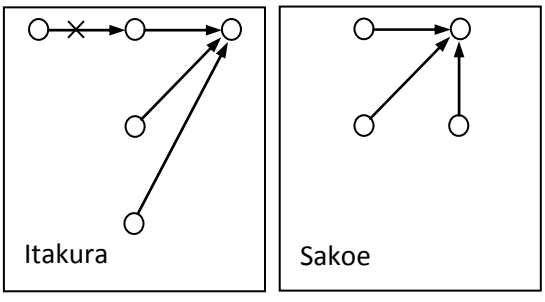
- Εισάγουμε δυο σειρές δεδομένων και δημιουργούμε ένα πίνακα τοπικού κόστους κόμβου με διαστάσεις x, y, όπου τοποθετούμε τις δύο ακολουθίες στις δυο διαστάσεις του πίνακα αντιστοίχως.
- Γεμίζουμε τον πίνακα κόστους βάση της Ευκλείδειας απόστασης όπου παίρνουμε ένα προς ένα κάθε στοιχείο της πρώτης σειράς με τα στοιχεία της δεύτερης κ.ο.κ.
- Αρχικοποιούμε ένα πίνακα όπου έχουμε το κόστος ενός κόμβου συναθροίζοντας και το συνολικό κόστος από τον κόμβο πρόγονο αναδρομικά.
- Αρχικοποιούμε ένα δεύτερο πίνακα όπου κρατάμε τις συντεταγμένες του πρόγονου κόμβου από τον τρέχοντα κόμβο.

  Η αρχικοποίηση αφορά το πρώτο στοιχείο του πίνακα και τις αρχικές μη επιτρεπόμενες μεταβάσεις βάση των περιορισμών του κάθε αλγόριθμου
- Ξεκινάμε την αναζήτηση σε συνέχεια από τους κόμβους αρχικοποίησης και ελέγχουμε για κάθε κόμβο το ελάχιστο κόστος μετάβασης από το σύνολο των επιτρεπόμενων κόμβων προγόνων και αποθηκεύουμε στους πίνακες τις αντίστοιχες τιμές μέχρι να ανατρέξουμε όλους τους κόμβους.
- Μόλις συμπληρωθεί όλος ο πίνακας συνολικού κόστους κάνουμε μία αντίστροφη αναζήτηση, από το τελευταίο στοιχείο προς το πρώτο (backtracking), έτσι ώστε να αναδείξουμε το μονοπάτι, μέσα από τα στοιχεία του πίνακα, με το μικρότερο δυνατό κόστος.
- Υπολογίζουμε το συνολικό κόστος και κανονικοποιούμε τα αποτελέσματα κατ' επιλογή.

Η διαφορά για τις δύο προσεγγίσεις έχει να κάνει με την μετάβαση από ένα κόμβο πρόγονο σε ένα τρέχοντα κόμβο και με ποιες είναι οι επιτρεπόμενές μεταβάσεις.

Αν υπολογίσουμε ότι στον οριζόντιο άξονα του πίνακα κόστους έχουμε το πρότυπο και στον κατακόρυφο έχουμε το δοκιμαστικό με την λογική των καρτεσιανών συντεταγμένων x y και αναπτύσσεται στο θετικό τέταρτο, οι επιτρεπόμενες μετακινήσεις από κόμβο σε κόμβο περιγράφονται στη διπλανό σχήμα για κάθε αλγόριθμο.

Παρατηρούμε ότι ο Itakura είναι πιο αυστηρός στους περιορισμούς του σε σύγκριση με τον Sakoe που λειτουργεί πιο ανεκτικά στις μεταβάσεις από κόμβο σε κόμβο.



Ο Fumitada Itakura[23], γεννήθηκε 6 Αυγούστου, 1940 είναι ένας Ιάπωνας επιστήμονας ο οποίος έκανε την πρωτοποριακή εργασία στη στατιστική επεξεργασία σήματος και την εφαρμογή της ανάλυσης και σύνθεσης φωνής.

[23] Πηγή:Wikipedia

Τα αποτελέσματα που παίρνουμε, από τον πίνακα κόστους και την απόσταση (best path) των δύο άκρων του, είναι ανάλογα της σύγκρισης των δύο ακολουθιών.

Ένα παράδειγμα είναι να συγκρίνουμε δύο ίδιες σειρές, όπως το [1 2 3 4 5]

5	10	6	3	1	0
4	6	3	1	0	1
3	3	1	0	1	3
2	1	0	1	3	6
1	0	1	3	6	10
	1	**2**	**3**	**4**	**5**

Παρατηρούμε ότι στη διαγώνιο δεν έχουμε κόστος μιας και τα στοιχεία ίδια. Ενώ όσο συγκρίνουμε διαφορετικά στοιχεία το κόστος αυξάνει.

Στη συνέχεια μπορούμε να συγκρίνουμε σειρές που αντιστοιχούν σε νότες.

Διαβάζουμε midi αρχεία και κάνουμε σύγκριση τις νότες που έχει το κάθε midi με διάφορες παραλλαγές.

7.6 Εξαγωγή Αποτελεσμάτων

Μετά από την σύγκριση εξετάζουμε το μονοπάτι του κόστους και μπορούμε να δούμε πόσα δείγματα από το πρότυπο αντιστοιχούν με το δοκιμαστικό και με μια κανονικοποίηση να πάρουμε ένα ποσοστό που εκφράζει το πώς παίχτηκε η αντίστοιχη νότα και να μπορέσουμε να εξάγουμε ποσοτικοποιημένα συμπεράσματα.

Με την αντιστοιχία των δειγμάτων δημιουργούμε ένα πλέγμα πάνω στο γράφημα στο εργαλείο μας όπου μπορούμε πατώντας σε όποιο κουτάκι του πλέγματος να ακούσουμε τη νότα του πρότυπου αρχικά και στη συνέχεια να ακούσουμε το κομμάτι από το δοκιμαστικό που αντιστοίχισε ο αλγόριθμος σύγκρισης με την πιο πάνω νότα του πρότυπου.

Στο πλέγμα τυπώνουμε και ένα χαρακτήρα που μας δείχνει πλασματικά αν η αναλογία στις διάρκειες είναι αποδεκτή.

8. Σύγκριση MIDI αρχείων Νότες – Κλίμακες

Συγκρίνουμε την κλίμακα μείζονα του Ντο με μια σχεδόν ίδια που περιέχει δύο λάθος νότες.

am=midiInfo(readmidi('Cmajor4.mid'))

```
am =
     2     0    60    64     0     1     2     4
     2     0    62    64     1     2     3     6
     2     0    64    64     2     3     5     8
     2     0    65    64     3     4     7    10
     2     0    67    64     4     5     9    12
     2     0    69    64     5     6    11    14
     2     0    71    64     6     7    13    16
     2     0    72    64     7     8    15    17
```

a=am(:,3)'

```
a =
    60    62    64    65    67    69    71    72
```

bm=midiInfo(readmidi('Cmajor4e2.mid'))

```
bm =
     2     0    60    64     0     1     2     4
     2     0    62    64     1     2     3     6
     2     0    64    64     2     3     5     8
     2     0    66    64     3     4     7    10
     2     0    67    64     4     5     9    12
     2     0    69    64     5     6    11    14
     2     0    71    64     6     7    13    16
     2     0    74    64     7     8    15    17
```

b=bm(:,3)'

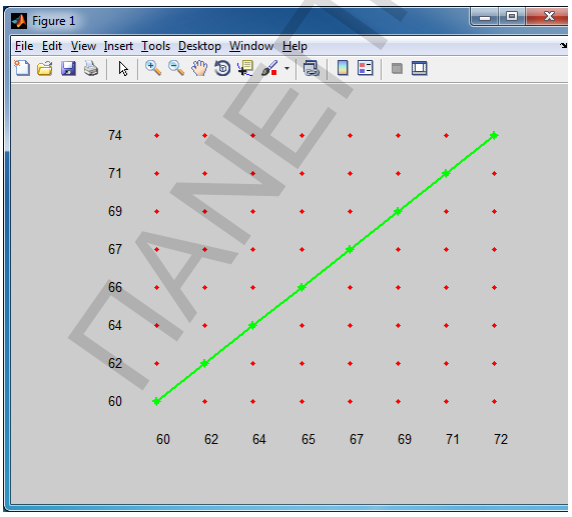
```
b =
    60    62    64    66    67    69    71    74
```

DTWItakura(a,b,1)

ans = 0.3750

DTWSakoe(a,b,1)

ans = 0.3750

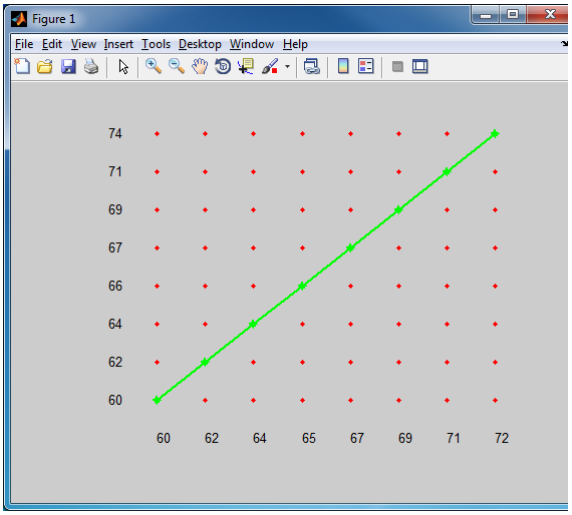


ίδιο γράφημα

```
bm=midiInfo(readmidi('Cmajor4e2.mid'))
b=bm(:,3)'
b  =
     60     62     64     66     67     69     71     72
```

DTWItakura(a,b,1)
ans = 0.1250

DTWSakoe(a,b,1)
ans = 0.1250



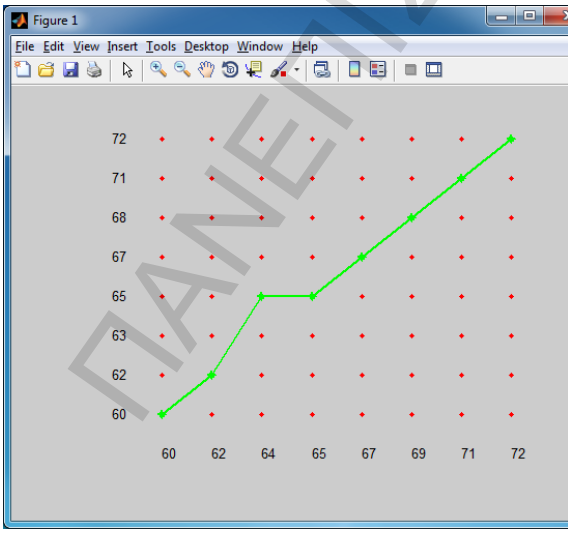
ίδιο γράφημα

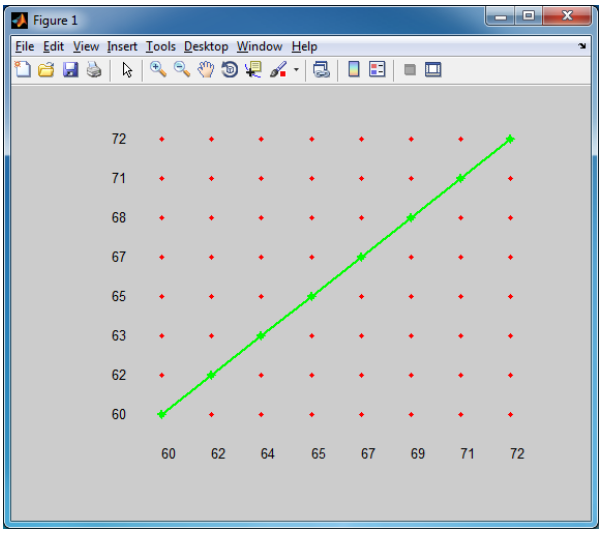
Συγκρίνουμε την κλίμακα μείζονα του Ντο με μια ελάσσονα του Ντο.

```
bm=midiInfo(readmidi('Cminor4.mid'))
b=bm(:,3)'
b  =
     60     62     63     65     67     68     71     72
```

DTWItakura(a,b,1)
ans = 0.2500

DTWSakoe(a,b,1)
ans = 0.250



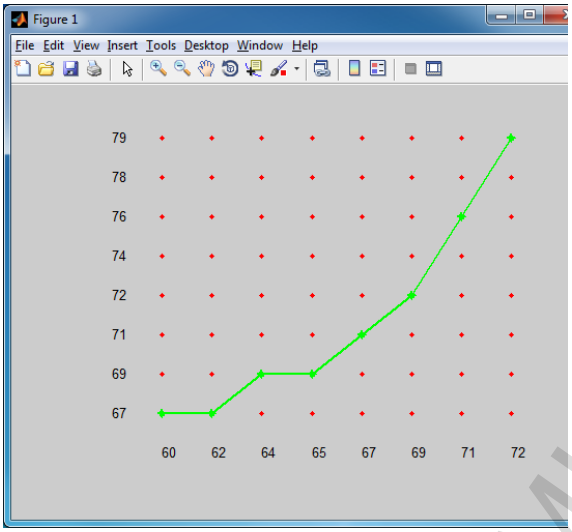


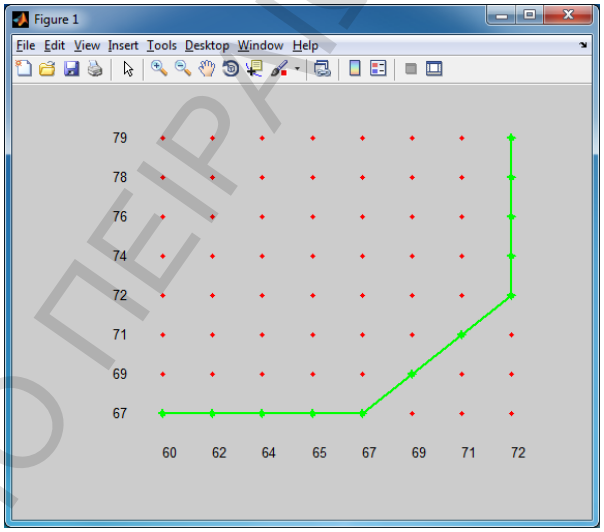
Συγκρίνουμε την κλίμακα μείζονα του Ντο με μια μείζονα του Σολ.

```
bm=midiInfo(readmidi('Gmajor4.mid'))
b=bm(:,3)'
b  =
     67     69     71     72     74     76     78     79
```

DTWItakura(a,b,1)
ans = 5

DTWSakoe(a,b,1)
ans = 3





min(a)
ans = 60

min(b)
ans = 67

a1=a-min(a)
a1 = 0 2 4 5 7 9 11 12

b1=b-min(b)
b1 = 0 2 4 5 7 9 11 12

Είναι ίσα!!!

9. Σύγκριση MIDI αρχείων με Δειγματοληψία

Μετατροπή από τις νότες midi σε δείγματα συχνότητας (Hz) και τόσα δείγματα, ανάλογα με της διάρκειας της νότας έτσι ώστε να συγκρίνουμε και το κράτημα της νότας.

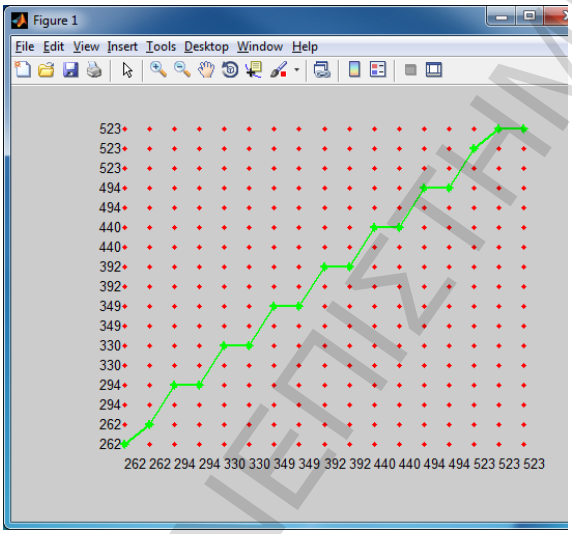
```
function as = midi2samle(am,f)
a=am(:,3)';
l=length(a);
for i=1:l
  af = (440/32)*2.^((a(i)-9)/12);
  js=(am(i,5)*f);
  je=(am(i,6)*f);
  for j=js:je
    as(j+1)=af;
  end
end
```

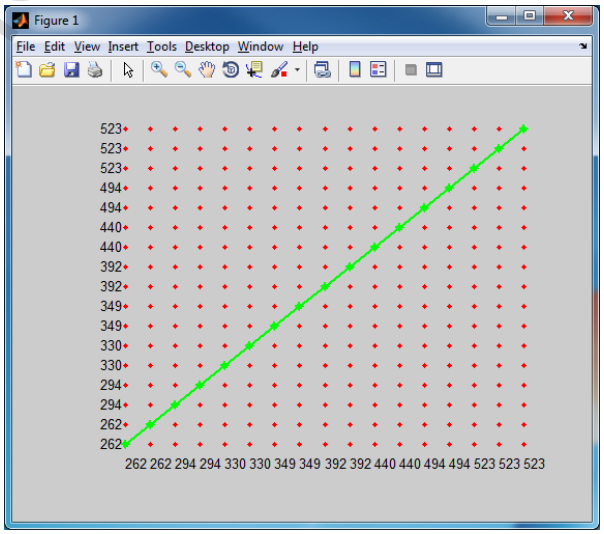
```
am=midiInfo(readmidi('Cmajor4.mid'))
as=midi2sample(am,2)
```

Συγκρίνουμε αρχικά τα ίδια δείγματα με δειγματοληψία 500ms (2 δείγματα το δευτερόλεπτο)

```
DTWItakura(as,as,2)
ans =  0
```

```
DTWSakoe(as,as,2)
ans =  0
```

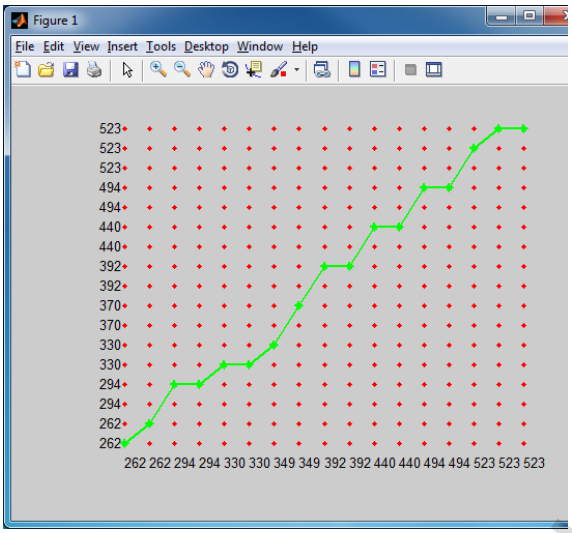




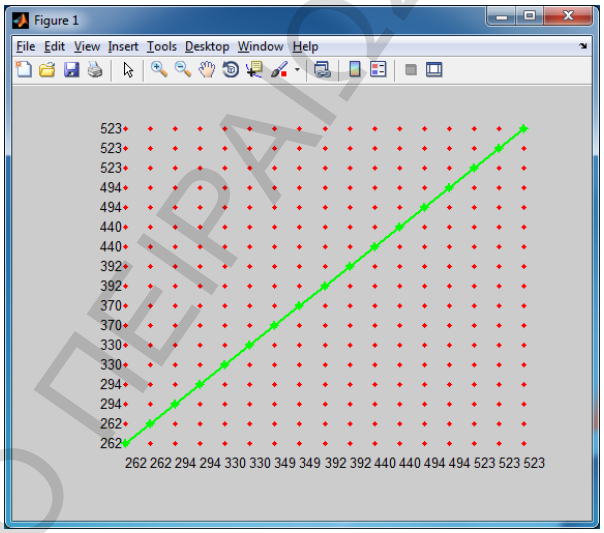
Περίπτωση ενός λάθους

bm=midiInfo(readmidi('Cmajor4e1.mid'))
bs=midi2sample(bm,2)

DTWItakura(as,bs,2)
ans = 2.3529



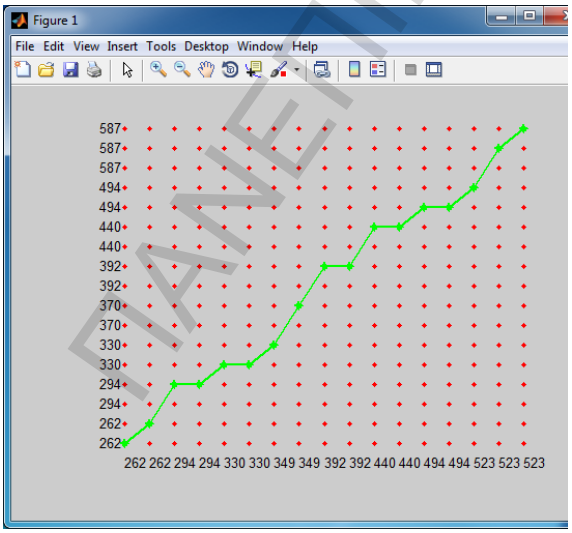
DTWSakoe(as,bs,2)
ans = 2.4706



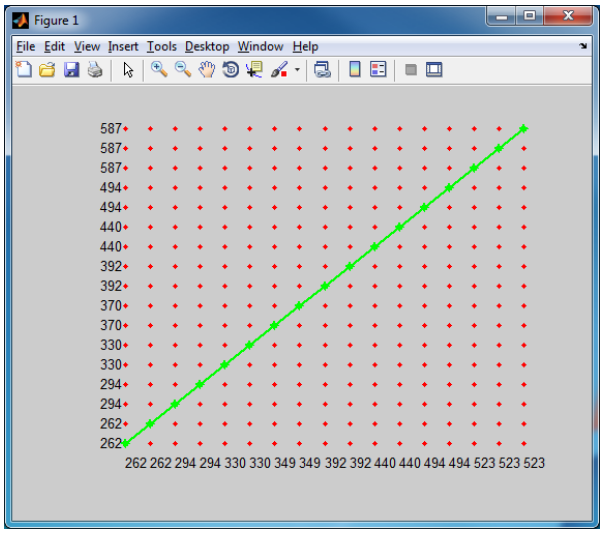
Περίπτωση δυο λάθων

bm=midiInfo(readmidi('Cmajor4e2.mid'))
bs=midi2sample(bm,2)

DTWItakura(as,bs,2)
ans = 11.5882



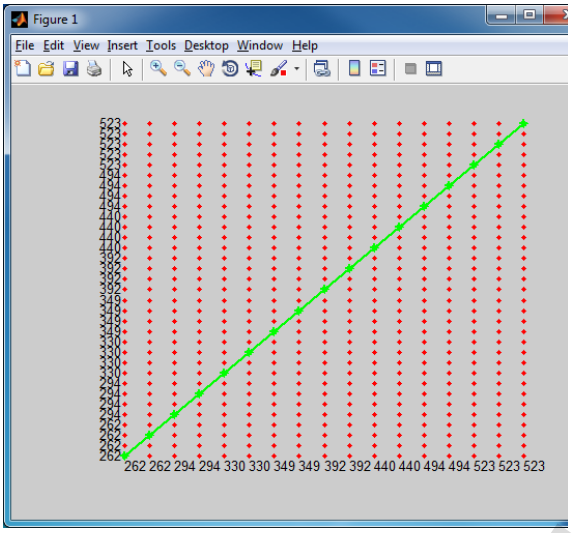
DTWSakoe(as,bs,2)
ans = 13.7647



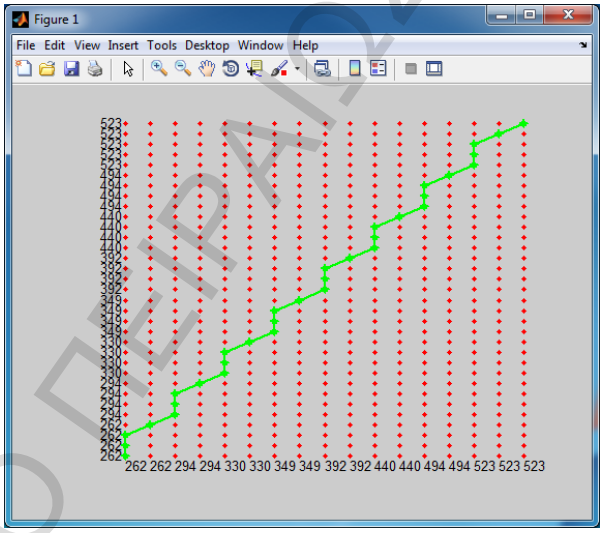
Περίπτωση ίδιας κλίμακας διαφορετικής διάρκειας

bm=midiInfo(readmidi('Cmajor2.mid'))
bs=midi2sample(bm,2)

DTWItakura(as,bs,2)
ans = 0



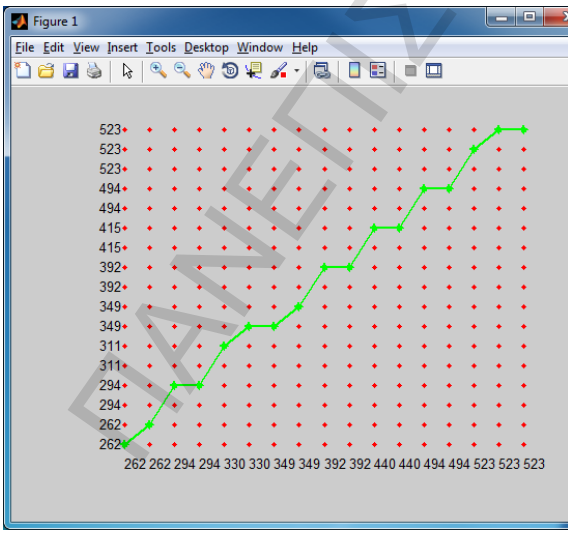
DTWSakoe(as,bs,2)
ans = 0



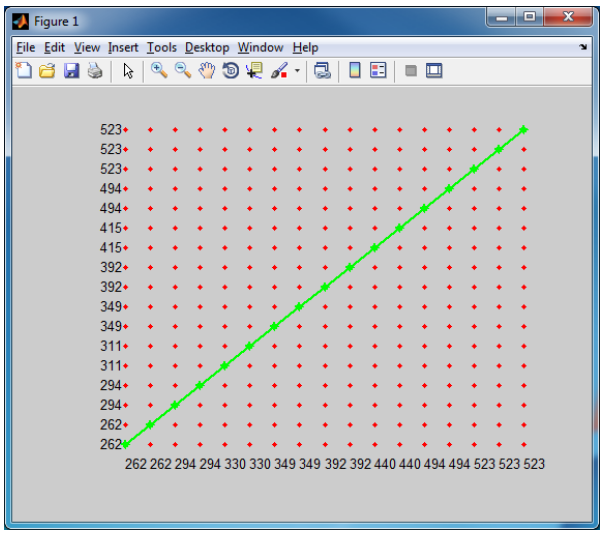
Περίπτωση διαφορετικής κλίμακας (μείζονα vs ελάσσονα)

bm=midiInfo(readmidi('Cminor4.mid'))
bs=midi2sample(bm,2)

DTWItakura(as,bs,2)
ans = 5.1765



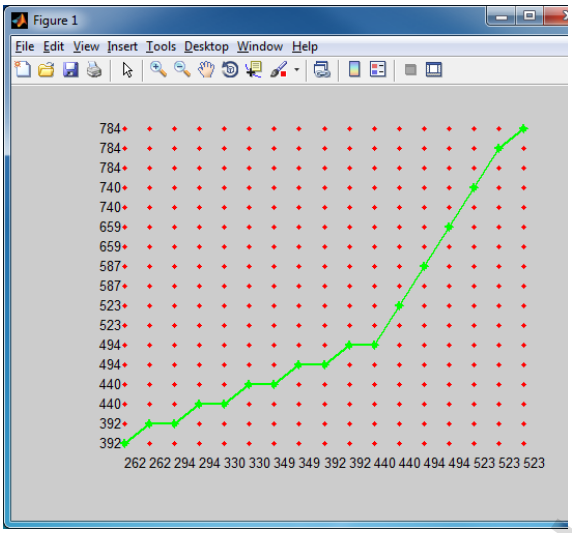
DTWSakoe(as,bs,2)
ans = 5.1765



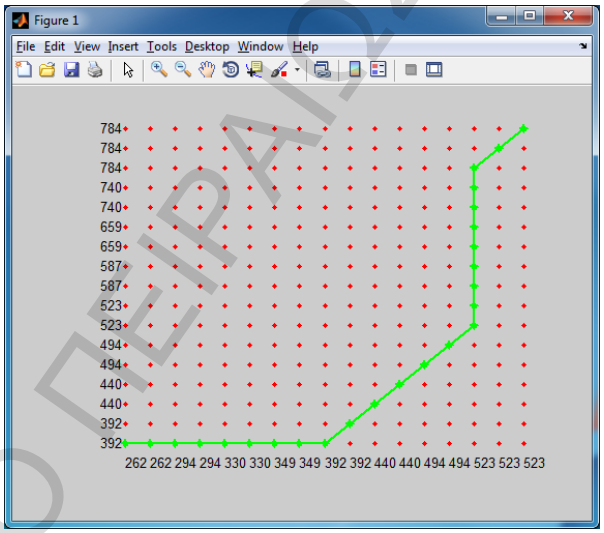
Περίπτωση διαφορετικής κλίμακας

```
bm=midiInfo(readmidi('Cmajor2.mid'))
bs=midi2sample(bm,2)
```

DTWItakura(as,bs,2)
ans = 135.1765

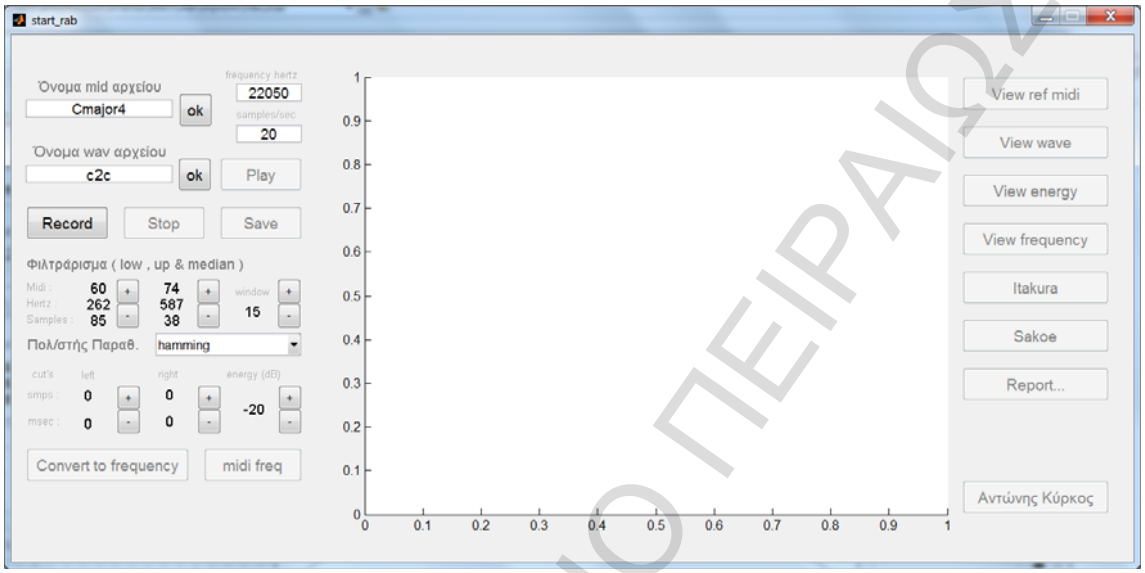


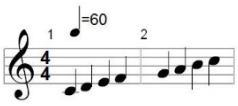
DTWSakoe(as,bs,2)
ans = 91.3200



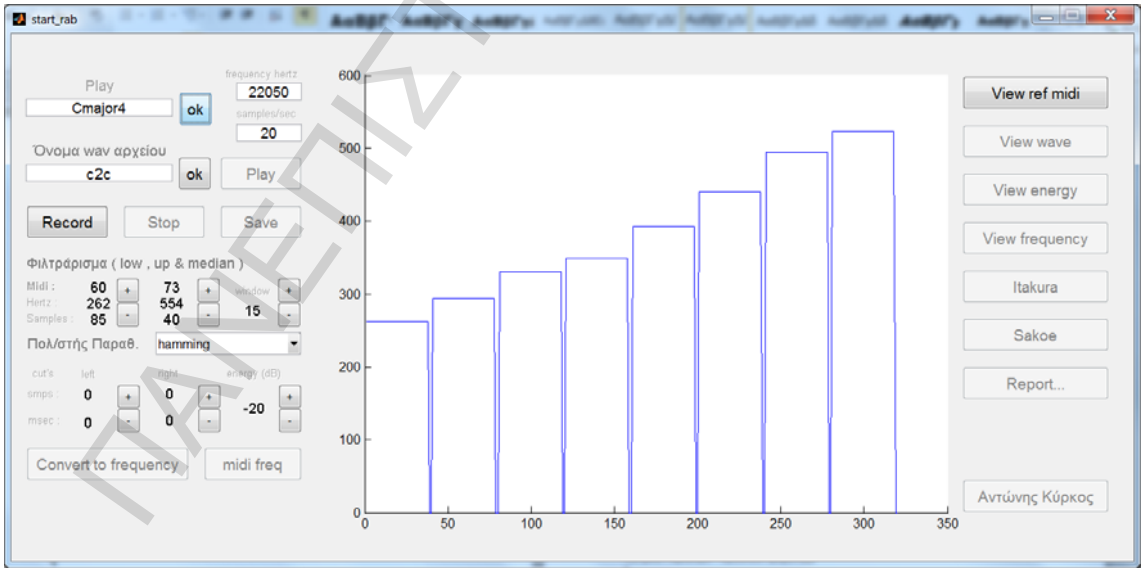
10. Σύγκριση MIDI με αρχείο ήχου

Δημιουργούμε ένα gui στο matlab τέτοιο ώστε να μας διευκολύνει στο να επεξεργαστούμε τα αρχεία να τα φιλτράρουμε και να εξάγουμε συμπεράσματα.

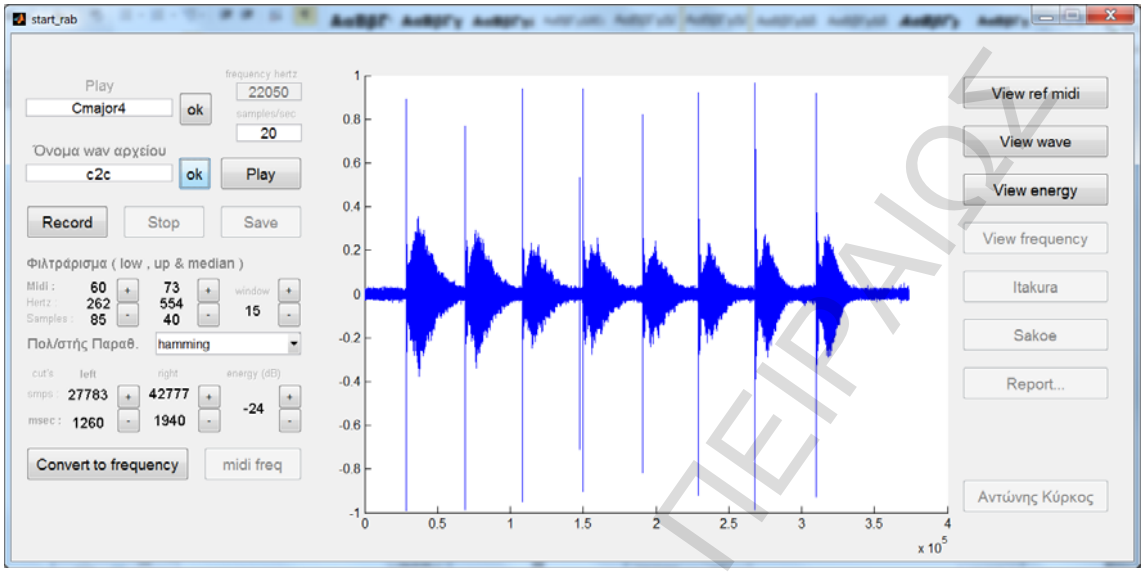




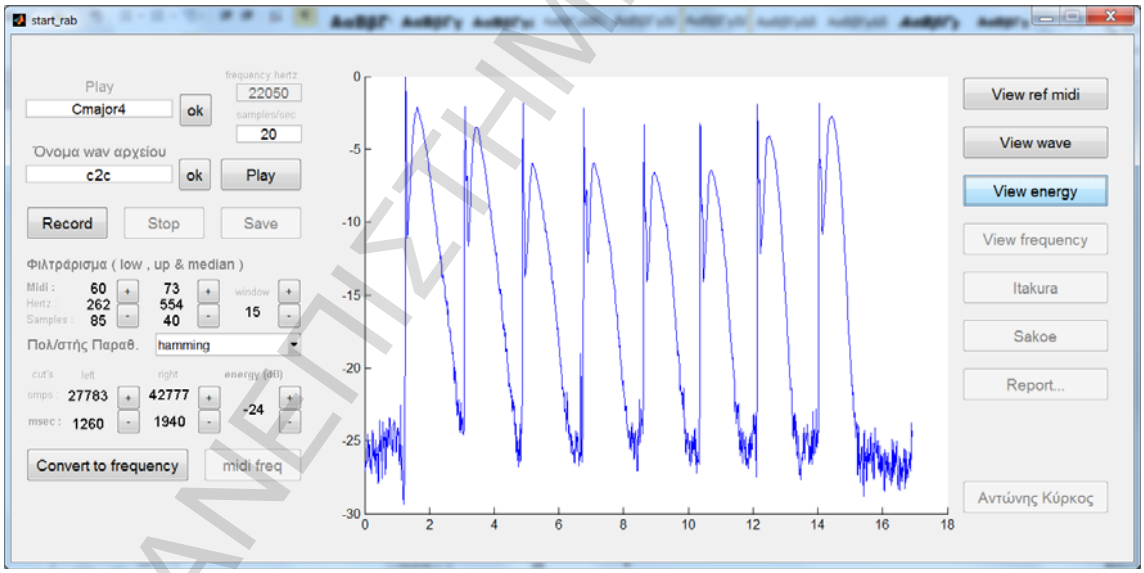
Εισαγωγή αρχείου midi



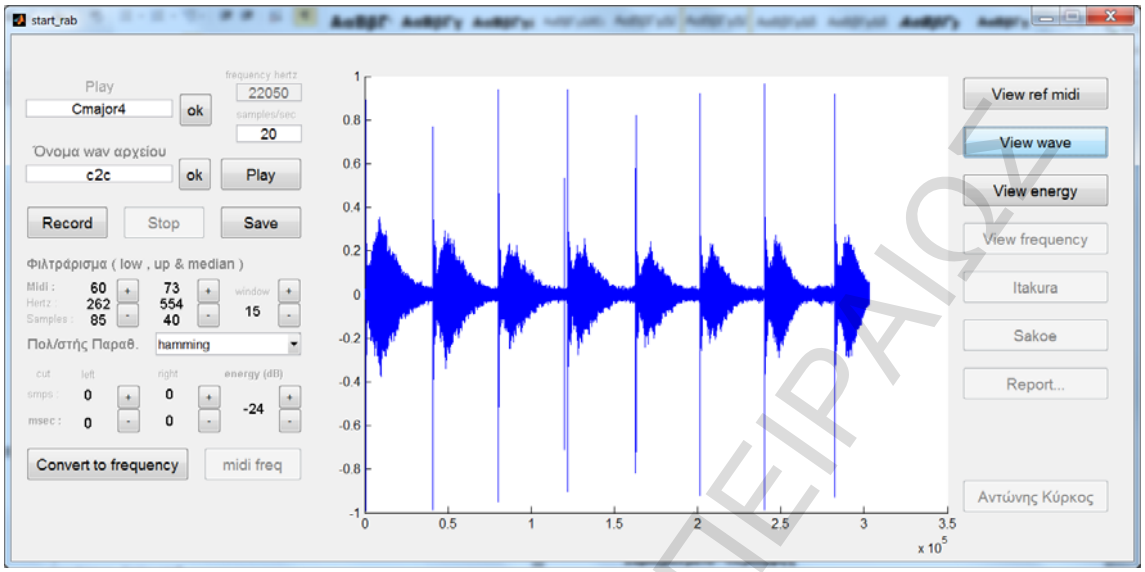
Εισαγωγή αρχείου wav



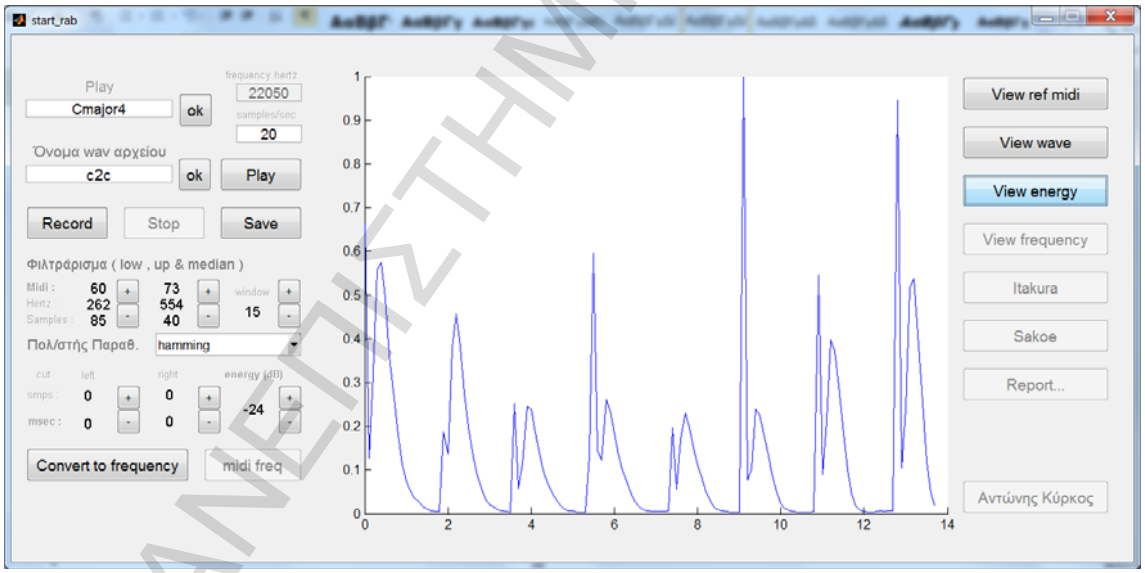
Κεντρικό Παράθυρο – Εργαλεία Προβολής



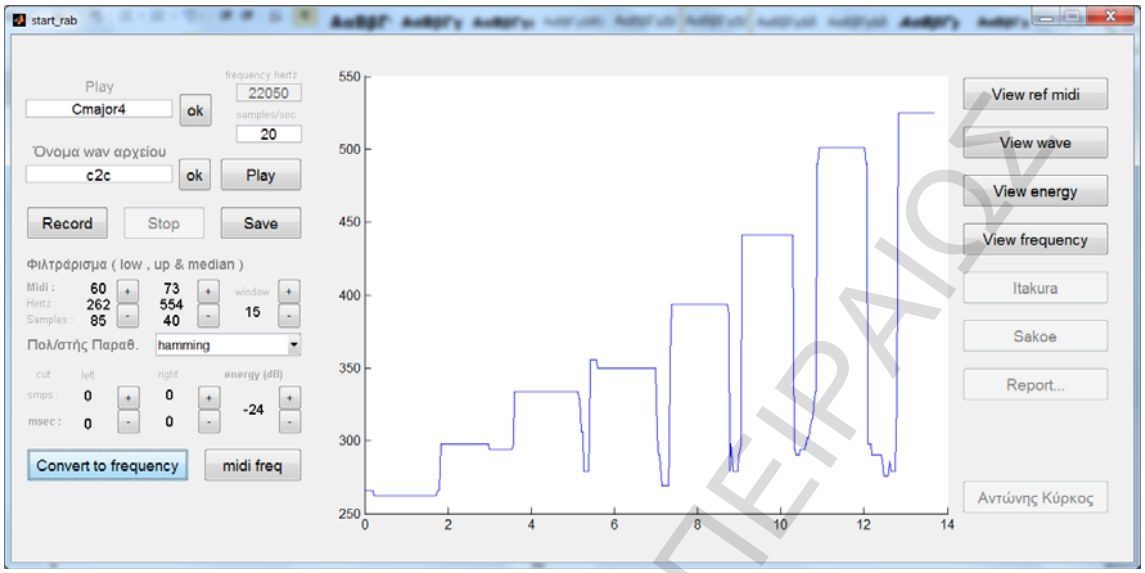
Όρια Φίλτρων – Επεξεργασία



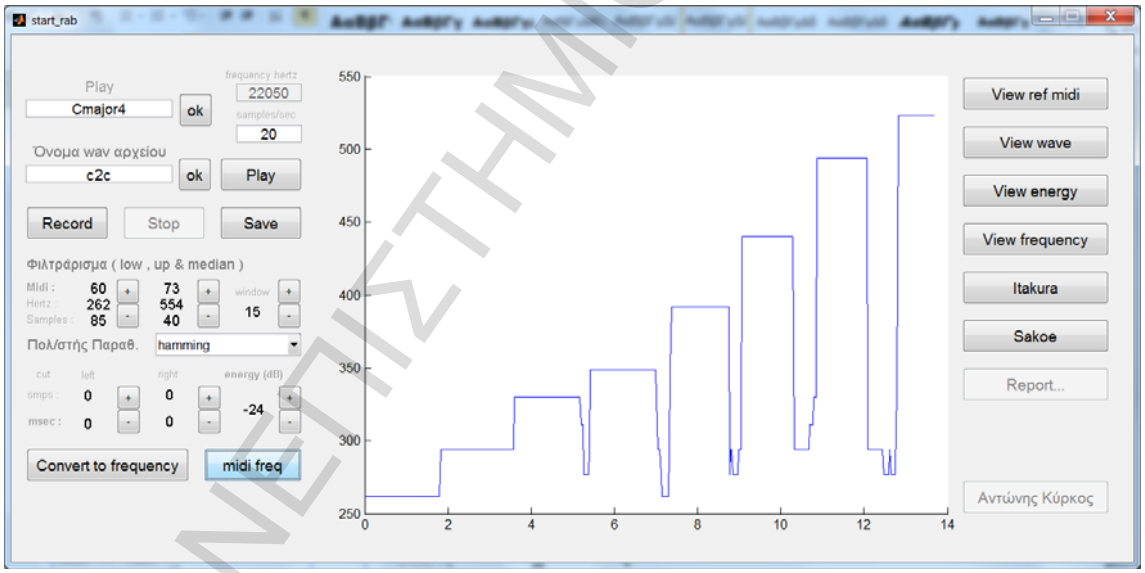
Ψαλιδισμός Δειγμάτων



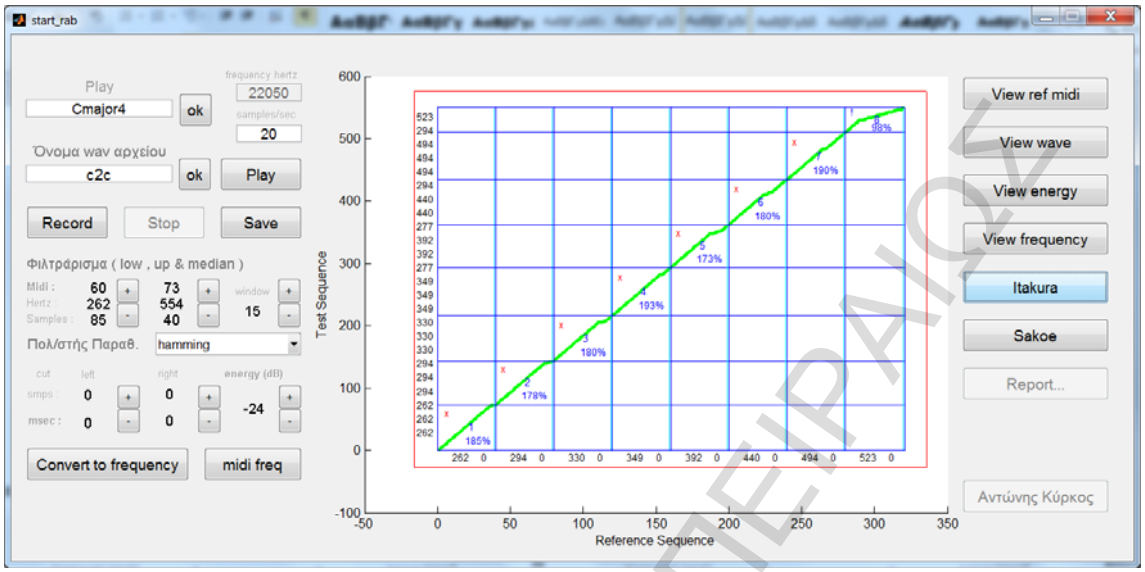
Μετατροπή Δειγμάτων σε Συχνότητα

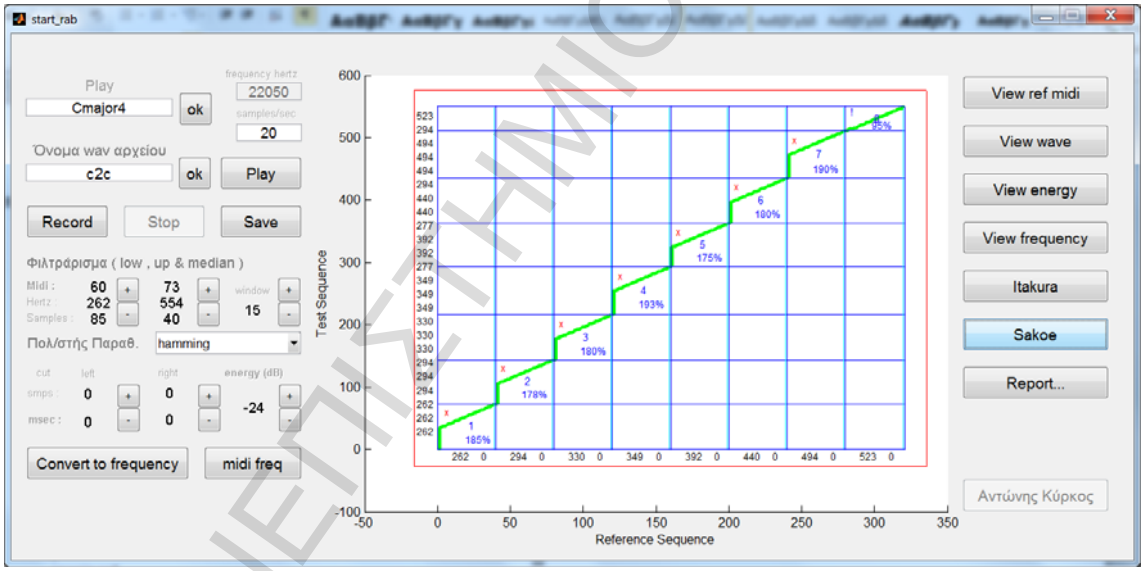


Μετατροπή Δειγμάτων σε συγκερασμένες συχνότητες



Αλγόριθμοι Σύγκρισης





Αποτελέσματα

Νότα 1η (262hz) C

διάρκεια frames, 38 στο πρότυπο και 73 στο δοκιμαστικό, με παύσεις 1 και 0 αντίστοιχα.

Συμπέρασμα : αναλογία 192% , κοιμήθηκες στο πλήκτρο!!!

Νότα 2η (294hz) D

διάρκεια frames, 38 στο πρότυπο και 71 στο δοκιμαστικό, με παύσεις 2 και 0 αντίστοιχα.

Συμπέρασμα : αναλογία 187% , κοιμήθηκες στο πλήκτρο!!!

Νότα 3η (330hz) E

διάρκεια frames, 38 στο πρότυπο και 64 στο δοκιμαστικό, με παύσεις 2 και 0 αντίστοιχα.

Συμπέρασμα : αναλογία 168% , μεγάλη διάρκεια παιξίματος

Νότα 4η (349hz) F

διάρκεια frames, 38 στο πρότυπο και 64 στο δοκιμαστικό, με παύσεις 2 και 0 αντίστοιχα.

Συμπέρασμα : αναλογία 168% , μεγάλη διάρκεια παιξίματος

Νότα 5η (392hz)

διάρκεια frames, 38 στο πρότυπο και 56 στο δοκιμαστικό, με παύσεις 2 και 0 αντίστοιχα.

Συμπέρασμα : αναλογία 147% , μεγάλη διάρκεια παιξίματος

Νότα 6η (440hz) A

διάρκεια frames, 38 στο πρότυπο και 50 στο δοκιμαστικό, με παύσεις 2 και 0 αντίστοιχα.

Συμπέρασμα : αναλογία 132% , μεγάλη διάρκεια παιξίματος

Νότα 7η (494hz) B

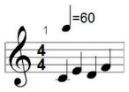
διάρκεια frames, 38 στο πρότυπο και 49 στο δοκιμαστικό, με παύσεις 2 και 0 αντίστοιχα.

Συμπέρασμα : αναλογία 129% , μεγάλη διάρκεια παιξίματος

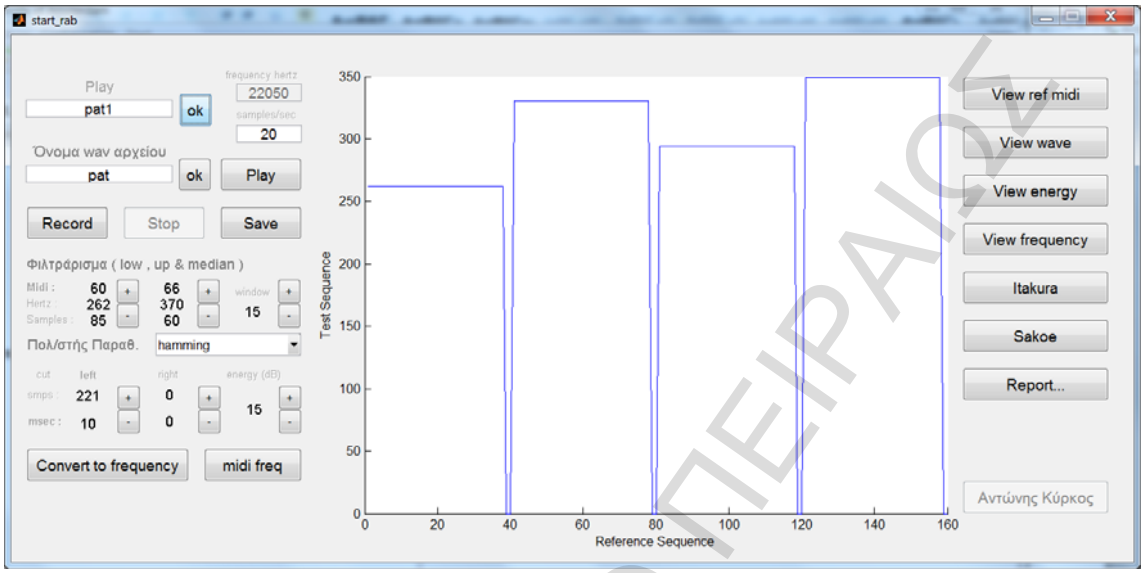
Νότα 8η (523hz) C

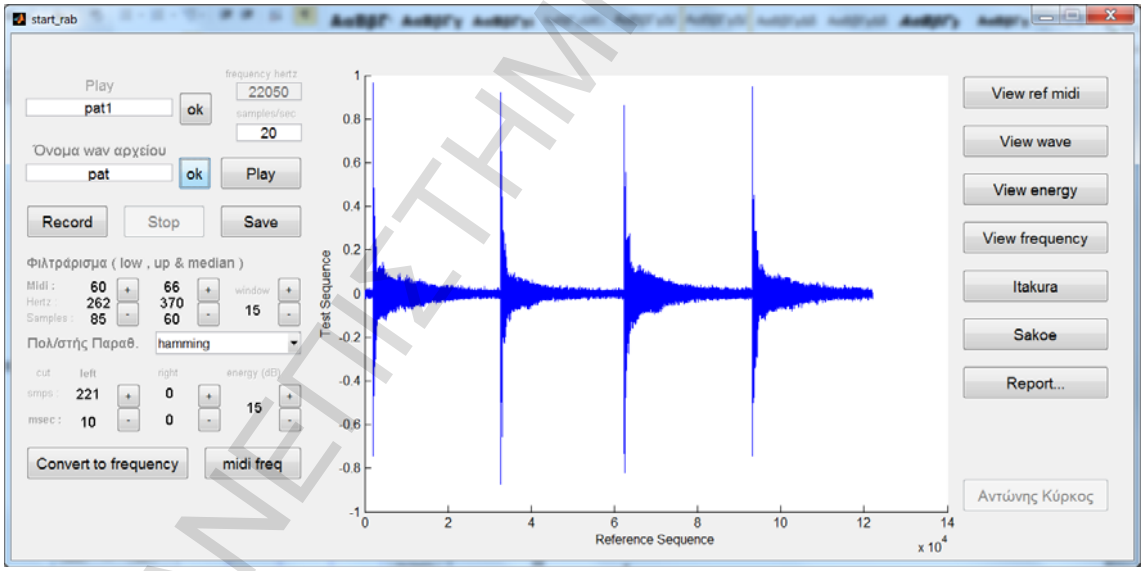
διάρκεια frames, 38 στο πρότυπο και 36 στο δοκιμαστικό, με παύσεις 2 και 0 αντίστοιχα.

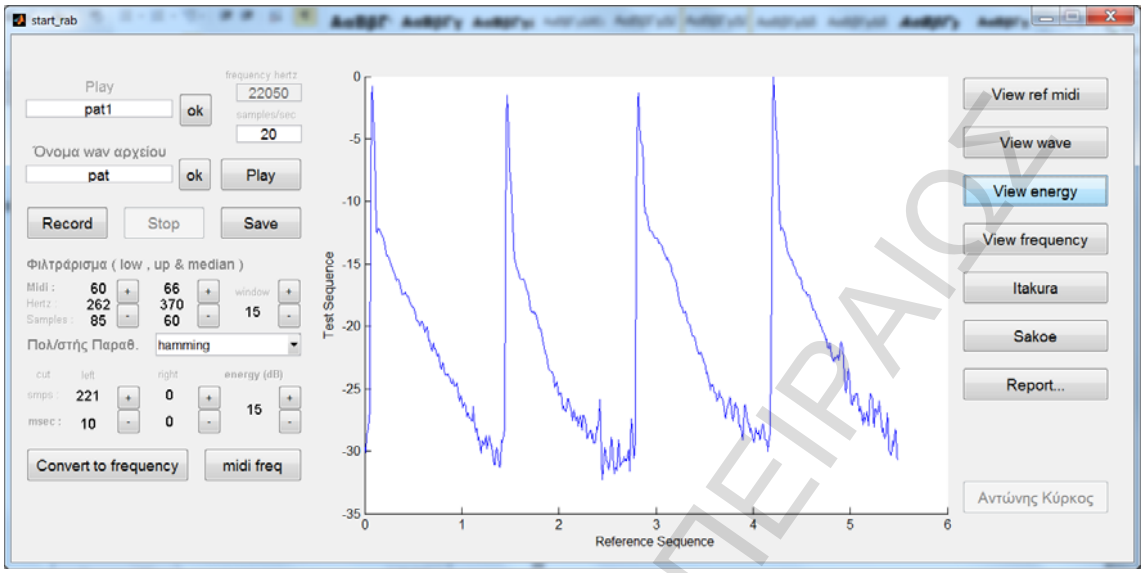
Συμπέρασμα : αναλογία 95% , τέλεια διάρκεια παιξίματος

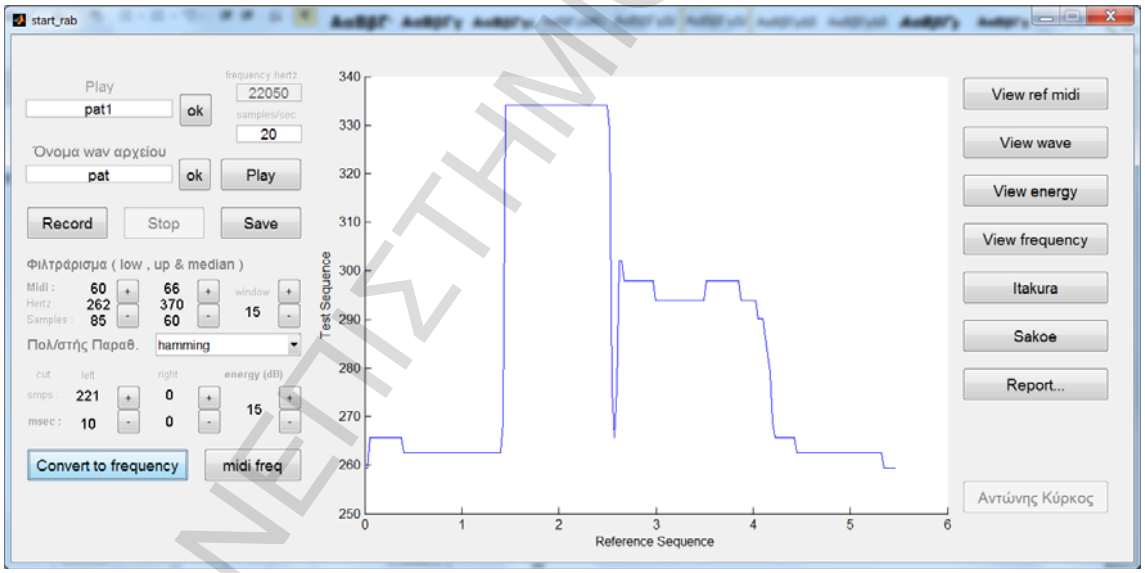


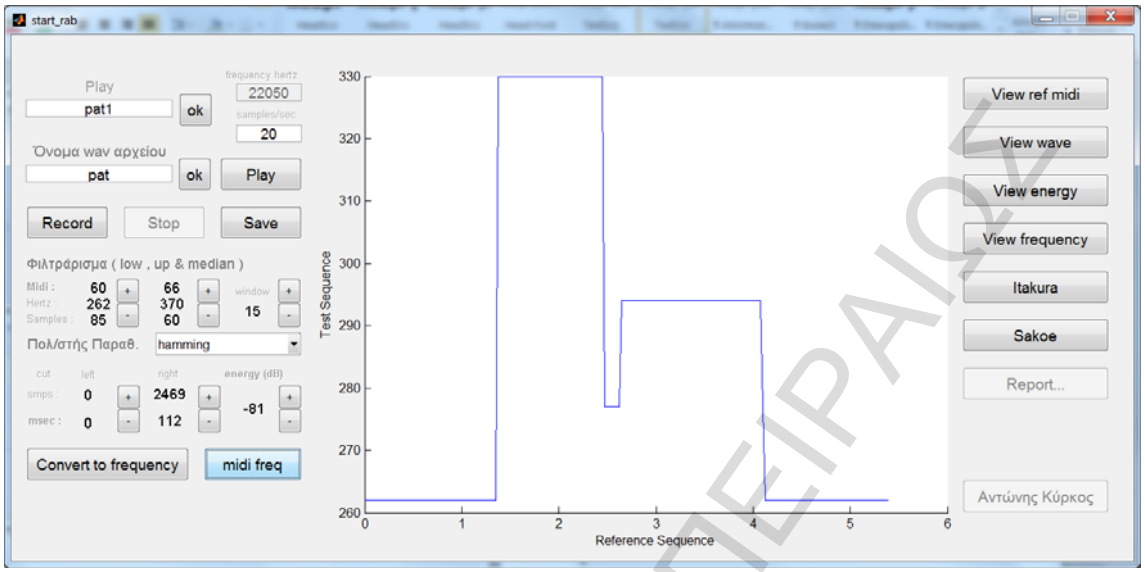
Εισαγωγή αρχείου midi

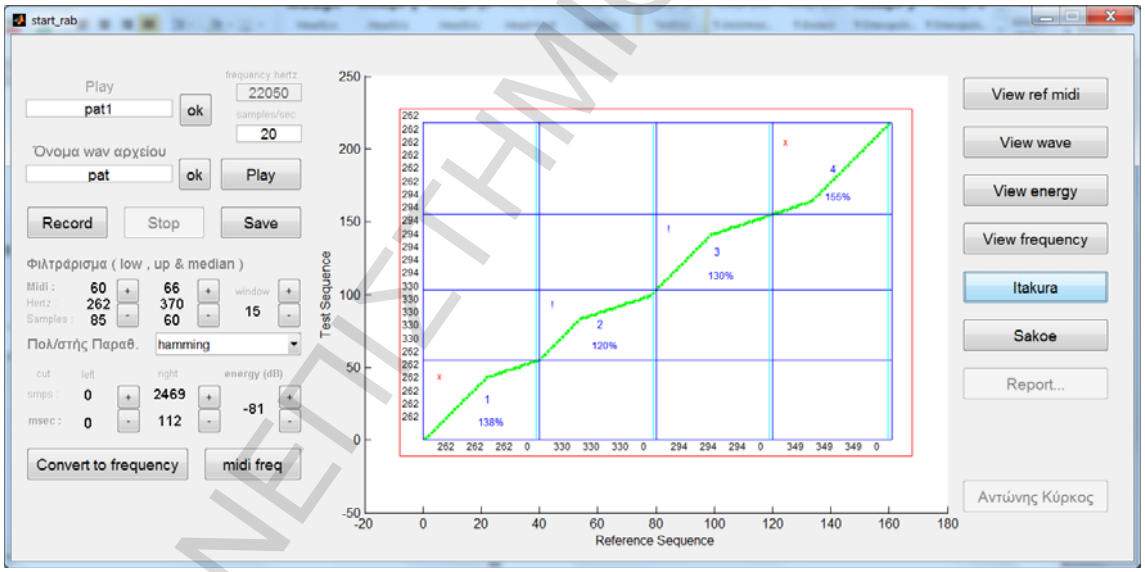


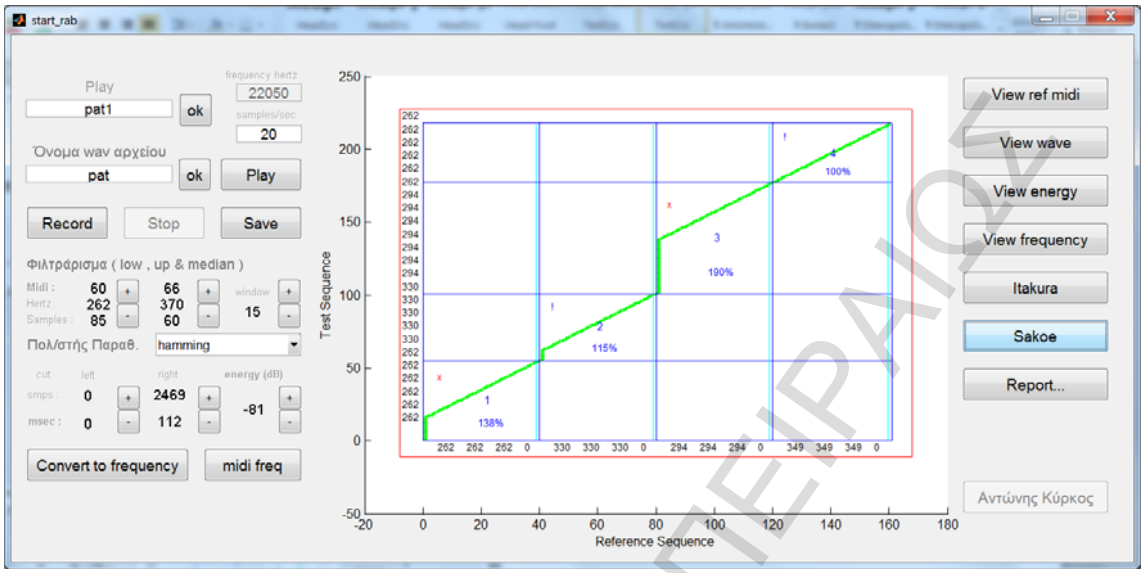












Νότα 1η (262hz) C

διάρκεια frames, 38 στο πρότυπο και 55 στο δοκιμαστικό, με παύσεις 1 και 0 αντίστοιχα.

Συμπέρασμα : αναλογία 145% , μεγάλη διάρκεια παιξίματος

Νότα 2η (330hz) E

διάρκεια frames, 38 στο πρότυπο και 44 στο δοκιμαστικό, με παύσεις 2 και 0 αντίστοιχα.

Συμπέρασμα : αναλογία 116% , καλή διάρκεια παιξίματος

Νότα 3η (294hz) D

διάρκεια frames, 38 στο πρότυπο και 58 στο δοκιμαστικό, με παύσεις 2 και 0 αντίστοιχα.

Συμπέρασμα : αναλογία 153% , μεγάλη διάρκεια παιξίματος

Νότα 4η (349hz) F

διάρκεια frames, 38 στο πρότυπο και 0 στο δοκιμαστικό, με παύσεις 2 και 0 αντίστοιχα.

Συμπέρασμα : αναλογία 0% , Λείπει η νότα στο παιξήμο

11. Συμπεράσματα

Παρατηρούμε ότι μπορούμε να συγκρίνουμε μουσικά κομμάτια και να εξάγουμε συμπεράσματα για το πώς είναι παιγμένο ένα μουσικό κομμάτι.

Φυσικά το ότι χρησιμοποιούμε στην παρούσα εργασία ηχογραφημένο σήμα μας εισάγει θόρυβο και γενικά μας αυξάνει τα σφάλματα που δυστυχώς δεν μπορούνε να εξαλειφθούν τελείως.

Για αυτό και περιοριζόμαστε σε ένα παίξιμο μίας νότας τη φορά και όχι σε πολυφωνία, όπως και σε απλές μελωδίες και όχι σε πολύ γρήγορες εναλλαγές ή πολύ σύντομες νότες που κατά την επεξεργασία μας θα χάνονταν τα δείγματα.

Η εργασία θα μπορούσε να συνεχιστεί σε επόμενο επίπεδο και να εμπλουτιστεί με επιπλέων δυνατότητες όπως η εισαγωγή περισσοτέρων μορφών κωδικοποίησης του ήχου (mp3 κτλπ)

Ακόμα και με την δυνατότητα σε απευθείας επικοινωνία ένος κλαβιέ με τον υπολογιστή μέσο του midi προτόκολου.

Γενικά το εργαλείο αναπτύσσεται με λογική εκπαιδευτικού χαραχτήρα και έχει ένα μεγάλο φάσμα ανάπτυξης και βελτίωσης.

12. Κώδικας

Ο κώδικας του GUI Matlab

```
function varargout = start_rab(varargin)
% START_RAB M-file for start_rab.fig
%      START_RAB, by itself, creates a new START_RAB or raises the existing
%      singleton*.
%
%      H = START_RAB returns the handle to a new START_RAB or the handle to
%      the existing singleton*.
%
%      START_RAB('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in START_RAB.M with the given input arguments.
%
%      START_RAB('Property','Value',...) creates a new START_RAB or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before start_rab_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to start_rab_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help start_rab

% Last Modified by GUIDE v2.5 16-Jun-2013 12:49:57

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @start_rab_OpeningFcn, ...
                   'gui_OutputFcn',  @start_rab_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before start_rab is made visible.
function start_rab_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to start_rab (see VARARGIN)

% Choose default command line output for start_rab
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes start_rab wait for user response (see UIRESUME)
% uiwait(handles.figure1);
```

```
% --- Outputs from this function are returned to the command line.
function varargout = start_rab_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

function edit_mid_file_Callback(hObject, eventdata, handles)
% hObject    handle to edit_mid_file (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_mid_file as text
%        str2double(get(hObject,'String')) returns contents of edit_mid_file as a double

% --- Executes during object creation, after setting all properties.
function edit_mid_file_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_mid_file (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton_mid_ok.
function pushbutton_mid_ok_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton_mid_ok (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global ref_mid ref_notes ref_frq ref_wav fs
% clear ref_mid ref_notes ref_frq
ref_file=strcat('mid\',get(handles.edit_mid_file,'String'),'.mid');
ref_mid=midiInfo(readmidi(ref_file));
ref_notes=ref_mid(:,3)'
% How many samples in second we calculate
s=str2double(get(handles.edit_samples,'String'))
%
ref_frq=[]
for i=1:length(ref_notes)
    audiofeq=(440/32)*2.^((ref_notes(i)-9)/12);
    js=ref_mid(i,5)*2*s;
    je=ref_mid(i,6)*2*s;
    for j=1+round(js):round(je)
        ref_frq(j)=round(audiofeq);
    end
    ref_frq(j-1)=0;
    ref_frq(j)=0;
end
fs=str2double(get(handles.edit_frequency,'String'));
% auto up down midi limit
midi=min(ref_notes);
hertz=round((440/32)*2^((midi-9)/12));
samples=ceil(fs/hertz);
set(handles.text9,'String',int2str(midi));
set(handles.text10,'String',int2str(hertz));
set(handles.text11,'String',int2str(samples));
midi=1+max(ref_notes);
hertz=round((440/32)*2^((midi-9)/12));
samples=ceil(fs/hertz);
%

ts=1/fs;
```

```
%
l_ref=length(ref_notes);
RnTs=0:ts:((ref_mid(l_ref,5)+ref_mid(l_ref,6)));
Rx=zeros(1,length(RnTs));
ref_wav=[]
for i=1:l_ref
    fo = (440/32)*2.^((ref_notes(i)-9)/12);
    js=round(ref_mid(i,5)*fs);
    if js==0
        js=1;
    end
    je=round(ref_mid(i,6)*fs);
    ref_wav(js:je)=Rx(js:je)+(.7*cos(2*pi*fo*RnTs(js:je)));
end
%
set(handles.text12,'String',int2str(midi));
set(handles.text13,'String',int2str(hertz));
set(handles.text14,'String',int2str(samples));
%
set(handles.text6,'FontWeight','bold')
hold
cla
plot(handles.axes1,ref_frq)
hold off
set(handles.pushb_view_midi,'Enable','on');
set(handles.text1,'String','Play','Enable','off');

function edit_frequency_Callback(hObject, eventdata, handles)
% hObject    handle to edit_frequency (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_frequency as text
%        str2double(get(hObject,'String')) returns contents of edit_frequency as a double

% --- Executes during object creation, after setting all properties.
function edit_frequency_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_frequency (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function edit_samples_Callback(hObject, eventdata, handles)
% hObject    handle to edit_samples (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_samples as text
%        str2double(get(hObject,'String')) returns contents of edit_samples as a double

% --- Executes during object creation, after setting all properties.
function edit_samples_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_samples (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function edit_wav_file_Callback(hObject, eventdata, handles)
% hObject    handle to edit_wav_file (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_wav_file as text
%        str2double(get(hObject,'String')) returns contents of edit_wav_file as a double

% --- Executes during object creation, after setting all properties.
function edit_wav_file_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_wav_file (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton_wav_ok.
function pushbutton_wav_ok_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton_wav_ok (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global x fs bits nrj Zc Tnrj flag_cut flag_axes view_axes
clc
col = get(hObject,'backg');
set(hObject,'str','run','backg',[1 .5 .5])
pause(.01)
file=strcat('wav\',get(handles.edit_wav_file,'String'),'.wav');
[x,fs,bits]=wavread(file);
flag_cut=1;
set(handles.text26,'FontWeight','normal')
set(handles.text27,'FontWeight','bold')
flag_axes=1;
set(handles.text18,'FontWeight','bold')
set(handles.text19,'FontWeight','normal')
set(handles.text20,'FontWeight','normal')
set(handles.edit_frequency,'Enable','off','String',int2str(fs));
hold
cla
plot(handles.axes1,x)
hold off
view_axes=0;
% calculate nrj=stEnergy (x,fs,round(.05*fs),round(.025*fs));
s=str2double(get(handles.edit_samples,'String'));
[nrj,Zc,Tnrj]=ZrlogEr (x,fs,400,100);
% nrj=nrj/max(nrj);

% Find Begin and End of wav
eavg=mean(nrj(1:20));
esig=std(nrj(1:20));
itu=-18;
itr=max(itu-10, eavg+3*esig);
set(handles.text_nrj,'String',int2str(itr));
zcavg=mean(Zc(1:20));
zcsig=std(Zc(1:20));
izct=max(10, zcavg+3*zcsig);
i=1;
len=length(nrj);
b=0;
e=0;
be=1;
en=len;
while ((b==0 | e==0) & i<len)
    if nrj(i)>itr && min(nrj(i+1:i+11))>itu && b~=1
        p=0;
```

```
        be=i;
        for j=i-1:-1:max(1,i-25)
            if Zc(j)>izct
                p=p+1;
                if p==4
                    be=j;
                end
            end
        end
        b=1;
    end
    if nrj(len-i)>itr && min(nrj(len-i-11:len-i-1))>itu && e~=1
        p=0;
        en=len-i;
        for j=en+1:min(len,en+25)
            if Zc(j)>izct
                p=p+1;
                if p==4
                    en=j;
                end
            end
        end
        e=1;
    end
    i=i+1;
end
msec=be*10;
msamp=(msec*fs)/1000;
set(handles.text21,'String',int2str(msamp));
set(handles.text22,'String',int2str(msec));
msec=(len-en)*10;
msamp=(msec*fs)/1000;
set(handles.text23,'String',int2str(msamp));
set(handles.text24,'String',int2str(msec));

set(handles.pushbutton_play,'Enable','on');
set(handles.pushbutton_fft,'Enable','on');
set(handles.pushb_view_wave,'Enable','on');
set(handles.pushb_view_nrj,'Enable','on');
set(hObject,'str','ok','backg',col);
pause(.01)

% --- Executes on button press in pushbutton_play.
function pushbutton_play_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton_play (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global x fs bits
sound (x,fs,bits);

% --- Executes on button press in pushbutton_record.
function pushbutton_record_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton_record (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global fs bits rec
bits=16;
fs=str2double(get(handles.edit_frequency,'String'));
set(handles.edit_frequency,'Enable','off');
rec=audiorecorder(fs,bits,1);
record(rec);
set(handles.pushbutton_stop,'Enable','on');

% --- Executes on button press in pushbutton_stop.
function pushbutton_stop_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton_stop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
global x fs rec nrj Tnrj Zc view_axes
stop(rec);
x=[];
x=getaudiodata(rec);
clear rec;
% calculate nrj=stEnergy (x,fs,round(.05*fs),round(.025*fs));
s=str2double(get(handles.edit_samples,'String'));
[nrj,Zc,Tnrj]=ZrlogEr (x,fs,400,100);
% nrj=nrj/max(nrj);
hold
cla
plot(handles.axes1,x)
hold off
view_axes=0;
set(handles.pushbutton_play,'Enable','on');
set(handles.pushbutton_save,'Enable','on');
set(handles.pushbutton_fft,'Enable','on');
set(handles.pushb_view_wave,'Enable','on');
set(handles.pushb_view_nrj,'Enable','on');

% --- Executes on button press in pushbutton_save.
function pushbutton_save_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton_save (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global x fs
file=strcat('wav\',get(handles.edit_wav_file,'String'),'.wav');
wavwrite (x,fs,file);

% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
fs=str2double(get(handles.edit_frequency,'String'));
midi=str2double(get(handles.text9,'String'));
midi=midi+1;
hertz=round((440/32)*2^((midi-9)/12));
samples=ceil(fs/hertz);
set(handles.text9,'String',int2str(midi));
set(handles.text10,'String',int2str(hertz));
set(handles.text11,'String',int2str(samples));

% --- Executes on button press in pushbutton8.
function pushbutton8_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
fs=str2double(get(handles.edit_frequency,'String'));
midi=str2double(get(handles.text9,'String'));
midi=midi-1;
hertz=round((440/32)*2^((midi-9)/12));
samples=ceil(fs/hertz);
set(handles.text9,'String',int2str(midi));
set(handles.text10,'String',int2str(hertz));
set(handles.text11,'String',int2str(samples));

% --- Executes on button press in pushbutton9.
function pushbutton9_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
fs=str2double(get(handles.edit_frequency,'String'));
midi=str2double(get(handles.text12,'String'));
midi=midi+1;
hertz=round((440/32)*2^((midi-9)/12));
samples=ceil(fs/hertz);
set(handles.text12,'String',int2str(midi));
```

```
set(handles.text13,'String',int2str(hertz));
set(handles.text14,'String',int2str(samples));

% --- Executes on button press in pushbutton10.
function pushbutton10_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
fs=str2double(get(handles.edit_frequency,'String'));
midi=str2double(get(handles.text12,'String'));
midi=midi-1;
hertz=round((440/32)*2^((midi-9)/12));
samples=ceil(fs/hertz);
set(handles.text12,'String',int2str(midi));
set(handles.text13,'String',int2str(hertz));
set(handles.text14,'String',int2str(samples));

% --- Executes on button press in pushbutton11.
function pushbutton11_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
median=str2double(get(handles.text_median,'String'));
median=median+1;
set(handles.text_median,'String',int2str(median));

% --- Executes on button press in pushbutton12.
function pushbutton12_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
median=str2double(get(handles.text_median,'String'));
median=median-1;
set(handles.text_median,'String',int2str(median));

% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = cellstr(get(hObject,'String')) returns popupmenu1 contents as cell array
%        contents{get(hObject,'Value')} returns selected item from popupmenu1

% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton13.
function pushbutton13_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global flag_cut
fs=str2double(get(handles.edit_frequency,'String'));
if flag_cut==0
    msamp=str2double(get(handles.text21,'String'));
    msamp=msamp+1;
    msec=(msamp/fs)*1000;
else
```

```
    msec=str2double(get(handles.text22,'String'));
    msec=msec+1;
    msamp=(msec*fs)/1000;
end
set(handles.text21,'String',int2str(msamp));
set(handles.text22,'String',int2str(msec));

% --- Executes on button press in pushbutton14.
function pushbutton14_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global flag_cut
fs=str2double(get(handles.edit_frequency,'String'));
if flag_cut==0
    msamp=str2double(get(handles.text21,'String'));
    msamp=msamp-1;
    msec=(msamp/fs)*1000;
else
    msec=str2double(get(handles.text22,'String'));
    msec=msec-1;
    msamp=(msec*fs)/1000;
end
set(handles.text21,'String',int2str(msamp));
set(handles.text22,'String',int2str(msec));

% --- Executes on button press in pushbutton15.
function pushbutton15_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global flag_cut
fs=str2double(get(handles.edit_frequency,'String'));
if flag_cut==0
    msamp=str2double(get(handles.text23,'String'));
    msamp=msamp+1;
    msec=(msamp/fs)*1000;
else
    msec=str2double(get(handles.text24,'String'));
    msec=msec+1;
    msamp=(msec*fs)/1000;
end
set(handles.text23,'String',int2str(msamp));
set(handles.text24,'String',int2str(msec));

% --- Executes on button press in pushbutton16.
function pushbutton16_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton16 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global flag_cut
fs=str2double(get(handles.edit_frequency,'String'));
if flag_cut==0
    msamp=str2double(get(handles.text23,'String'));
    msamp=msamp-1;
    msec=(msamp/fs)*1000;
else
    msec=str2double(get(handles.text24,'String'));
    msec=msec-1;
    msamp=(msec*fs)/1000;
end
set(handles.text23,'String',int2str(msamp));
set(handles.text24,'String',int2str(msec));

% --- Executes on button press in pushbutton17.
function pushbutton17_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton17 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
global fs nrj
s=str2double(get(handles.edit_samples,'String'));
n=str2double(get(handles.text_nrj,'String'));
n=n+1;
set(handles.text_nrj,'String',int2str(n));
nrjlgh=length(nrj)
i=1;
while nrj(i)<(n/100)
    i=i+1;
end
msamp=(i-2)*2*fs/s;
msec=(msamp/fs)*1000;
set(handles.text21,'String',int2str(msamp));
set(handles.text22,'String',int2str(msec));
i=nrjlgh;
while nrj(i)<(n/100)
    i=i-1;
end
msamp=(nrjlgh-i-3)*2*fs/s;
msec=(msamp/fs)*1000;
set(handles.text23,'String',int2str(msamp));
set(handles.text24,'String',int2str(msec));

% --- Executes on button press in pushbutton18.
function pushbutton18_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton18 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global fs nrj
s=str2double(get(handles.edit_samples,'String'));
n=str2double(get(handles.text_nrj,'String'));
n=n-1;
set(handles.text_nrj,'String',int2str(n));
nrjlgh=length(nrj)
i=1;
while nrj(i)<(n/100)
    i=i+1;
end
msamp=(i-2)*2*fs/s;
msec=(msamp/fs)*1000;
set(handles.text21,'String',int2str(msamp));
set(handles.text22,'String',int2str(msec));
i=nrjlgh;
while nrj(i)<(n/100)
    i=i-1;
end
msamp=(nrjlgh-i-3)*2*fs/s;
msec=(msamp/fs)*1000;
set(handles.text23,'String',int2str(msamp));
set(handles.text24,'String',int2str(msec));

% --- Executes on button press in pushbutton_fft.
function pushbutton_fft_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton_fft (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global x fs Frq Tfrq
txt = get(hObject,'String');
col = get(hObject,'ForegroundColor');
set(hObject,'String','prossessing...','FontWeight','bold','ForegroundColor',[1 .5 .5])
pause(.01)

% [Fr,T]=stFundAMDF(x,Fs,winlength,winstep,windowMultiplier,Tmin,Tmax)
s=str2double(get(handles.edit_samples,'String'));
str=get(handles.popupmenu1,'String');
val=get(handles.popupmenu1,'Value');
```

```
[Frq,Tfrq]=stFundAMDF(x,fs,round((1/s)*fs),round((1/s)*fs/2),str{val},get(handles.text14,'String
'),get(handles.text11,'String'));
% Median
median=str2double(get(handles.text_median,'String'));
Frq=medfilt1(Frq,round(median));
%
set(handles.pushbutton_midi_frq,'Enable','on');
set(handles.pushb_view_freq,'Enable','on');
hold
cla
plot(handles.axes1,Tfrq,Frq)
hold off
set(hObject,'str',txt,'FontWeight','normal','ForegroundColor',col);
pause(.01)

% --- Executes on button press in pushbutton_midi_frq.
function pushbutton_midi_frq_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton_midi_frq (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Frq Tfrq Frm Mn nrj
[Frm,Mn]=f2mf(Frq);
sql=(str2double(get(handles.text_nrj,'String')))/100
set(handles.pushbutton_Itakura,'Enable','on');
set(handles.pushbutton_Sakoe,'Enable','on');

for i=1:length(nrj)
    if nrj(i)<sql
        Frm(i)=0;
    end
end
hold
cla
plot(handles.axes1,Tfrq,Frm)
hold off

% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over text2.
function text2_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to text2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.edit_frequency,'Enable','on')

% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over text26.
function text26_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to text26 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global flag_cut
flag_cut=0;
set(hObject,'FontWeight','bold')
set(handles.text27,'FontWeight','normal')

% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over text27.
function text27_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to text27 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global flag_cut
flag_cut=1;
set(hObject,'FontWeight','bold')
set(handles.text26,'FontWeight','normal')

% --- Executes on button press in pushb_view_midi.
function pushb_view_midi_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to pushb_view_midi (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global ref_frq
hold
cla
plot(handles.axes1,ref_frq)
hold off

% --- Executes on button press in pushb_view_wave.
function pushb_view_wave_Callback(hObject, eventdata, handles)
% hObject    handle to pushb_view_wave (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global x view_axes
hold
cla
plot(handles.axes1,x)
hold off
view_axes=0;

% --- Executes on button press in pushb_view_nrj.
function pushb_view_nrj_Callback(hObject, eventdata, handles)
% hObject    handle to pushb_view_nrj (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global nrj Tnrj flag_axes view_axes
flag_axes=0;
set(handles.text18,'FontWeight','normal')
set(handles.text19,'FontWeight','normal')
set(handles.text20,'FontWeight','bold')
hold
cla
plot(handles.axes1,Tnrj,nrj)
hold off
view_axes=1;

% --- Executes on button press in pushb_view_freq.
function pushb_view_freq_Callback(hObject, eventdata, handles)
% hObject    handle to pushb_view_freq (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global Frq Tfrq flag_axes view_axes
hold
cla
plot(handles.axes1,Tfrq,Frq)
hold off

% --- Executes on button press in pushbutton_Itakura.
function pushbutton_Itakura_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton_Itakura (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global ref_frq ref_mid Frm BestPath flag_axes cross div
s=str2double(get(handles.edit_samples,'String'));
flag_axes=3
[MatchingCost,BestPath,D,Pred]=DTWItakura(ref_frq,Frm,1);
Itakura=MatchingCost
% grid plot
[cross,div]=gridplot(ref_mid,BestPath,s);
% figure
% hold
% cla
% plot(handles.axes1,BestPath)
% hold off

% --- Executes on button press in pushbutton_Sakoe.
function pushbutton_Sakoe_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to pushbutton_Sakoe (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global ref_frq ref_mid Frm BestPath flag_axes cross div
s=str2double(get(handles.edit_samples,'String'));
flag_axes=3
[MatchingCost,BestPath,D,Pred]=DTWSakoe(ref_frq,Frm,1);
Sakoe=MatchingCost
% grid plot
[cross,div]=gridplot(ref_mid,BestPath,s);
set(handles.pushbutton28,'Enable','on');

% figure
% hold
% plot(handles.axes1,BestPath)
% cla
% stem(handles.axes1,BestPath)
% hold off

% --- Executes on mouse press over axes background.
function axes1_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to axes1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global ref_wav x nrj fs flag_axes view_axes BestPath cross
p=get(hObject,'currentpoint')
s=str2double(get(handles.edit_samples,'String'))

switch flag_axes
   case 0
       l=length(x);
       n=p(3)*100
       set(handles.text_nrj,'String',int2str(n));
       disp('flag is for energy')
       nrjlgh=length(nrj)
       i=1;
       while nrj(i)<p(3)
           i=i+1;
       end
       msamp=(i-2)*2*fs/s;
       msec=(msamp/fs)*1000;
       set(handles.text21,'String',int2str(msamp));
       set(handles.text22,'String',int2str(msec));
       i=nrjlgh;
       while nrj(i)<p(3)
           i=i-1;
       end
       msamp=(nrjlgh-i-3)*2*fs/s;
       msec=(msamp/fs)*1000;
       set(handles.text23,'String',int2str(msamp));
       set(handles.text24,'String',int2str(msec));

    case 1
       disp('flag is one for left')
       if view_axes==1
           msamp=p(1)*fs;
       else
           msamp=p(1);
       end
       msec=(msamp/fs)*1000;
       set(handles.text21,'String',int2str(msamp));
       set(handles.text22,'String',int2str(msec));
       p(1)

    case 2
       disp('flag is two for right')
       l=length(x);
       if view_axes==1
```

```
            msamp=l-p(1)*fs;
        else
            msamp=l-p(1);
        end
        msec=(msamp/fs)*1000;
        set(handles.text23,'String',int2str(msamp));
        set(handles.text24,'String',int2str(msec));
        p(1)

    case 3
        disp('flag is two for play sound')
        point=round(p(1))
        lc=length(cross)
        start=cross(1)
        stop=cross(lc)
        for i=1:lc
            if point<imag(cross(i))
                start=cross(i-1)
                stop=cross(i)
                break
            end
        end
        start=start*(.025*fs)
        stop=stop*(.025*fs)

        str=round(imag(start))
        stp=round(imag(stop))

        if i==lc
            stp=length(ref_wav)
        end
        sound(ref_wav(str:stp),fs)
        pause (.5)
        str=round(real(start))
        stp=round(real(stop))
        if i==lc
            stp=length(x)
        end
        sound(x(str:stp),fs)

    otherwise
      disp('Unknown flag....')
end

% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over text18.
function text18_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to text18 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global flag_axes
if flag_axes==1
    set(handles.text21,'String','0')
    set(handles.text22,'String','0')
else
    flag_axes=1;
    set(hObject,'FontWeight','bold')
    set(handles.text19,'FontWeight','normal')
    set(handles.text20,'FontWeight','normal')
end

% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over text22.
function text22_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to text22 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over text20.
function text20_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to text20 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global nrj Tnrj flag_axes
if flag_axes==0
    sm_all=sum(nrj);
    npc=100*sm_all/length(nrj);
    set(handles.text_nrj,'String',int2str(npc));
end

% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over text29.
function text29_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to text29 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% txt = get(hObject,'String');
% col = get(hObject,'ForegroundColor');
global x fs nrj Tnrj flag_axes view_axes
set(hObject,'String','run','FontWeight','bold')
pause(.01)
l=length(x);
lowlim=max(1,str2double(get(handles.text21,'String')));
uplim=str2double(get(handles.text23,'String'));
switch flag_axes
   case 0
       disp('flag is for energy')
       uplim=l-uplim;
       x=x(lowlim:uplim);
       set(handles.text21,'String','0');
       set(handles.text22,'String','0');
       set(handles.text23,'String','0');
       set(handles.text24,'String','0');
       % set(handles.text_nrj,'String','0');

    case 1
       disp('cut left')
       x=x(lowlim:l);
       set(handles.text21,'String','0');
       set(handles.text22,'String','0');

   case 2
       disp('cut right')
       uplim=l-uplim;
       x=x(1:uplim);
       set(handles.text23,'String','0');
       set(handles.text24,'String','0');

   otherwise
      disp('Unknown flag....')
end
s=str2double(get(handles.edit_samples,'String'))
[nrj,Tnrj]=stEnergy (x,fs,round((1/s)*fs),round((2/s)*fs));
nrjm=max(nrj);
nrj=nrj/nrjm;
hold
cla
if view_axes==1
    plot(handles.axes1,nrj)
else
    plot(handles.axes1,x)
end
hold off
set(handles.pushbutton_save,'Enable','on');
```

```
% set(hObject,'String',txt,'FontWeight','normal','ForegroundColor',col);
set(hObject,'String','cut','FontWeight','normal');
pause(.01)

% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over text30.
function text30_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to text30 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over text19.
function text19_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to text19 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global flag_axes
if flag_axes==2
    set(handles.text23,'String','0')
    set(handles.text24,'String','0')
else
    flag_axes=2;
    set(hObject,'FontWeight','bold')
    set(handles.text18,'FontWeight','normal')
    set(handles.text20,'FontWeight','normal')
end

% --- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over text1.
function text1_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to text1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global ref_wav fs
%ts=1/fs;
%
%l_ref=length(ref_notes);
%RnTs=0:ts:((ref_mid(l_ref,5)+ref_mid(l_ref,6)));
%Rx=zeros(1,length(RnTs));
%for i=1:l_ref
%    fo = (440/32)*2.^((ref_notes(i)-9)/12);
%    js=ref_mid(i,5)*fs;
%    if js==0
%        js=1;
%    end
%    je=ref_mid(i,6)*fs;
%    Rx(js:je)=Rx(js:je)+(.7*cos(2*pi*fo*RnTs(js:je)));
%end
sound(ref_wav,fs)

% --- Executes on mouse press over axes background.
function axes2_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to axes2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes on button press in pushbutton28.
function pushbutton28_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton28 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global cross div ref_frq Frm
clc

for i=1:length(cross)-1
    r_cnt=0;
    t_cnt=0;
```

```
zr_cnt=0;
zt_cnt=0;
ind=imag(cross(i));
note=ref_frq(ind);
if note==0
    note=ref_frq(ind+1);
end
fprintf('Νότα %dη (%dhz) ', i, note);
switch note
    case {262,523}
        fprintf('C ');
    case {}
        fprintf('C#');
    case {294}
        fprintf('D ');
    case {}
        fprintf('D#');
    case {330}
        fprintf('E ');
    case {349}
        fprintf('F ');
    case {}
        fprintf('F#');
    case {342}
        fprintf('G ');
    case {}
        fprintf('G#');
    case {440}
        fprintf('A ');
    case {}
        fprintf('A#');
    case {494}
        fprintf('B ');
end
fprintf('\n');
for j=imag(cross(i)):imag(cross(i+1))-1
    switch ref_frq(j)
        case note
            r_cnt=r_cnt+1;
        case 0
            zr_cnt=zr_cnt+1;
    end
end
for j=real(cross(i)):real(cross(i+1))
    switch Frm(j)
        case note
            t_cnt=t_cnt+1;
        case 0
            zt_cnt=zt_cnt+1;
    end
end
ana=round(100*(t_cnt/r_cnt));
zan=round(100*(zt_cnt/zr_cnt));
fprintf('διάρκεια frames, %d στο πρότυπο ', r_cnt);
fprintf('και %d στο δοκιμαστικό, ', t_cnt);
fprintf('με παύσεις %d ', zr_cnt);
fprintf('και %d αντίστοιχα.\n', zt_cnt);
fprintf('Συμπέρασμα : αναλογία %d%s',  ana, '% , ');
sc=round(ana/10);
switch sc
    case {1,2}
        fprintf('σχεδόν ανύπαρκτη νότα');
    case {3,4,5,6,7}
       fprintf('μικρή διάρκεια παιξίματος');
    case {8,9,11,12}
       fprintf('καλή διάρκεια παιξίματος');
    case {10}
       fprintf('τέλεια διάρκεια παιξίματος');
```

```
        case {13,14,15,16,17}
            fprintf('μεγάλη διάρκεια παιξίματος');
        case {18,19,20,21,22,23,24,25}
            fprintf('κοιμήθηκες στο πλήκτρο!!!');
        otherwise
            fprintf('Λείπει η νότα στο παιξήμο');
    end
    zsc=round(zan/10);
    switch zsc
        case {1,2}
            fprintf(', σχεδόν ανύπαρκτες παύσεις');
        case {3,4,5,6,7}
           fprintf(', μικρή διάρκεια παύσεων');
        case {8,9,11,12}
           fprintf(', καλές παύσεις');
        case {10}
           fprintf(', τέλεια διάρκεια παύσεων');
        case {13,14,15,16,17}
            fprintf(', μεγάλη διάρκεια παύσεων');
        case {18,19,20,21,22,23,24,25}
            fprintf(', πολύ μεγάλη διάρκεια παύσεων');
        otherwise
            fprintf(' ');
    end

    fprintf('\n\n');
end

% --- Executes on button press in pushbutton29.
function pushbutton29_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton29 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global cross div ref_frq Frm
j=1;
for i=1:length(ref_frq)

end

% --- Executes on button press in pushbutton30.
function pushbutton30_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton30 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

function [E,Z,T]=ZrlogEr(x,Fs,L,R)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FUNCTION
%   [E,Z,T]=ZrlogEr(x,Fs,L,R)
%   Computes the energy and Zero Crossing
% INPUT ARGUMENTS:
%   x  : signal                    (sequence of samples).
%   Fs : sampling frequency        (10kHz).
%   L  : length of the moving window (400 number of samples).
%   R  : step of the moving window   (100 number of samples).
% OUTPUT ARGUMENTS:
%   E    : Energy values in dB.
%   Z    : Zero Crossing Rate
%   T    : time equivalent of the first sample of each window.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if nargin<4
    return;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if Fs>10000
        x=resample(x,10000,Fs);
        Fs=10000;
End
```

```
x = filter(Hpf,x);
i=1;
m=1;
N=length(x);
h=hamming(L);
while (m+L-1<=N)
    y=x(m:m+L-1);
    E(i)=sum((y.*h).^2);
    T(i)=(m-1)*(1/Fs);
    Z(i)=0;
    for k=2:L
        if y(k)>=0 & y(k-1)<0
            Z(i)=Z(i)+1;
        elseif y(k)<0 & y(k-1)>=0
            Z(i)=Z(i)+1;
        end
    end
    Z(i)=R*Z(i)/(2*L);
    i=i+1;
    m=m+R;
end
E=10*log10(E);
E=E-max(E);
```

Ο κώδικας του AMDF Matlab

```
function [Fr,T]=stFundAMDF(x,Fs,winlength,winstep,windowMultiplier,Tmin,Tmax)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FUNCTION
%   [Fr,T]=stFundAMDF(x,Fs,winlength,winstep,windowMultiplier,Tmin,Tmax)
%   Fundamental Frequency Tracking using the Average Magnitude Difference Function [Rabi78].
%
%   Reference [Rabi78]: Rabiner L.R., Schafer R.W., "Digital Processing of Speech Signals,"
Prentice Hall, 1978.
%
% INPUT ARGUMENTS:
%   x:                  signal (sequence of samples).
%   Fs:                 sampling frequency (Hz).
%   winlength:          length of the moving window (number of samples).
%   winstep:            step of the moving window (number of samples).
%   windowMultiplier:   use 'hamming', 'hanning', etc., i.e., any valid
%                       Matlab window multiplier (or [] for rectangular
%                       window).
%   Tmin:               minimum period length (in samples).
%   Tmax:               maximum period length (in samples).
%
% OUTPUT ARGUMENTS:
%   Fr:     sequence of fundamental frequencies (Hz).
%   T:      time equivalent of the first sample of each window.
%
% (c) 2008 A. Pikrakis, S. Theodoridis, K. Koutroumbas, D. Cavouras
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[Fr,T]=MovingWindow(x,Fs,winlength,winstep,windowMultiplier,'sfFundAMDF(y,Fs,Tmin,Tmax);',['Tmin
=' num2str(Tmin) ';' 'Tmax=' num2str(Tmax) ';']);



function [FeatureSeq,T]=MovingWindow(x,Fs,winlength,winstep,windowMultiplier,Feature,extraArgs)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FUNCTION
%   [FeatureSeq,T]=MovingWindow(x,Fs,winlength,winstep,windowMultiplier,Feature,extraArgs)
%   This is an auxiliary function which implements the moving-window technique for short-term
processing operations.
%
% INPUT ARGUMENTS:
%   x:                  signal frame (sequence of samples).
```

```
%   Fs:                 sampling frequency (Hz).
%   winslength:         moving window length (number of samples).
%   winstep:            moving window step (number of samples).
%   windowMultiplier:  'hamming', 'hanning', etc. Any valid Matlab window function. If is set to
[] a rectangular window is used.
%   Feature:            string, e.g., 'sfSpectralCentroid(y,Fs);', which is evaluated to compute
the desired feature on a frame basis.
%   extraArgs:          string which contains any extra arguments that need to be passed to a
function call that computes a feature on a frame basis.
%                       Can be set equal to [] if no extra arguments are needed.
%
% OUTPUT ARGUMENTS:
%   FeatureSeq:         the extracted feature sequence.
%   T:                  time equivalent of the first sample of each instance of the moving
window.
%
% (c) 2008 A. Pikrakis, S. Theodoridis, K. Koutroumbas, D. Cavouras
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if ~isempty(windowMultiplier)
    h=eval([windowMultiplier '(winlength);']);
else
    h=ones(winlength,1);
end

if ~isempty(extraArgs)
    eval(extraArgs);
end

i=1;
N=length(x);
m=1;
while (m+winlength-1<=N)
    y=x(m:m+winlength-1) .*h;
    FeatureSeq(:,i)=eval(Feature);
    T(i)=(m-1)*(1/Fs);
    i=i+1;
    m=m+winstep;
end



function [Fr]=sfFundAMDF(x,Fs,Tmin,Tmax)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FUNCTION
%   [Fr]=sfFundAMDF(x,Fs,Tmin,Tmax)
%   Computes the fundamental frequency of a single frame using the Average
%   Magnitude Difference Function for periodicity detection [Rabi78].
%
%   Reference [Rabi78]: Rabiner L.R., Schafer R.W., "Digital Processing of Speech Signals,"
Prentice Hall, 1978.
%
% INPUT ARGUMENTS:
%   x:      signal frame (sequence of samples).
%   Fs:     sampling frequency (Hz).
%   Tmin:   minimum period length (in samples).
%   Tmax:   maximum period length (in samples).
%
% OUTPUT ARGUMENTS:
%   Fr:     fundamental frequency (Hz).
%
% (c) 2008 A. Pikrakis, S. Theodoridis, K. Koutroumbas, D. Cavouras
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Initialization and argument consistency checking
N=length(x); % frame length
if Tmin<=1 | Tmin>Tmax | Tmax >=N
    return;
```

```
end

% Processing starts here
AMDF(1:Tmin-1)=inf;
for t=Tmin:Tmax
    x_orig=[x;zeros(t,1)];
    x_shifted=[zeros(t,1);x];
    AMDF(t)=(1/(N-t))*sum(abs(x_orig - x_shifted));
end
[minAMDF,T]=min(AMDF(Tmin:Tmax));
Fr=Fs/(T+Tmin-1);
```

Ο κώδικας του DTWItakura Matlab

```
function [MatchingCost,BestPath,D,Pred]=DTWItakura(ref,test,genPlot)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FUNCTION
%   [MatchingCost,BestPath,D,Pred]=DTWItakura(ref,test,genPlot)
% Computes the Dynamic Time Warping cost between two feature sequences. The first
% argument is the prototype, which is placed on the horizontal axis of the matching
% grid. The function employs the standard Itakura local constraints on
% the cost grid, where the Euclidean distance has been used as the
% distance metric. No end-points contstraints have been adopted. This function calls
% BackTracking.m to extract the best path.
%
% INPUT ARGUMENTS:
%   ref:            reference sequence. Its size is mxI, where m is
%                   the number of features and I the number of feature vectors.
%   test:           test sequence. Its size is mxJ, where m is
%                   the number of features and J the number of feature vectors.
%   genPlot:        if set to 1, a plot of the best path is generated.
%
% OUTPUT ARGUMENTS:
%   MatchingCost:   the matching cost normalized, i.e., it is divided by
%                   the length of the best path.
%   BestPath:       backtracking path. Each node of the best path is represented as a complex
%                   number, where the real part stands for the row index and the
%                   imaginary part stands for the column index of the node.
%   D:              cost grid. Its size is JxI.
%   Pred:           matrix of node predecessors. The real part of Pred(j,i) is the row index
%                   and the imaginary part of Pred(j,i) is the column index of the predecessor
%                   of node (j,i).
%
% (c) 2010 S. Theodoridis, A. Pikrakis, K. Koutroumbas, D. Cavouras
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[D1,I]=size(ref);
[D2,J]=size(test);

if D1~=D2
    msgbox('Mismatch of dimensions between feature sequences');
    MatchingCost=[]; BestPath=[]; D=[]; Pred=[];
    return;
end

for j=1:J
    for i=1:I
       %Euclidean distance
       NodeCost(j,i)=sqrt(sum((ref(:,i)-test(:,j)).^2));
    end
end

%Initialization
D(1,1)=NodeCost(1,1);
Pred(1,1)=0;
D(1,2)=NodeCost(1,1)+NodeCost(1,2);
Pred(1,2)=1+sqrt(-1)*1;
```

```
if I>2
    for i=3:I % forbidden transitions
        D(1,i)=inf;
        Pred(1,i)=inf;
    end
end


for j=2:J % forbidden transitions
    D(j,1)=inf;
    Pred(j,1)=inf;
end

%Main Loop
for j=2:J
    for i=2:I
        if (Pred(j,i-1)==j+sqrt(-1)*(i-2))
            if j>2
                [D(j,i),ind]=min([D(j-1,i-1) D(j-2,i-1)]+NodeCost(j,i));
            else
                [D(j,i),ind]=min([D(j-1,i-1)]+NodeCost(j,i));
            end
            if ind==1
                Pred(j,i)=j-1+sqrt(-1)*(i-1);
            else
                Pred(j,i)=(j-2)+sqrt(-1)*(i-1);
            end
        else
            if j>2
                [D(j,i),ind]=min([D(j,i-1) D(j-1,i-1) D(j-2,i-1)]+NodeCost(j,i));
            else
                [D(j,i),ind]=min([D(j,i-1) D(j-1,i-1)]+NodeCost(j,i));
            end
            if ind==1
                Pred(j,i)=(j)+sqrt(-1)*(i-1);
            elseif ind==2
                Pred(j,i)=(j-1)+sqrt(-1)*(i-1);
            else
                Pred(j,i)=(j-2)+sqrt(-1)*(i-1);
            end
        end
    end
end
%End of Main Loop

if D(J,I)~=inf
    BestPath=BackTracking(Pred,J,I,genPlot,ref,test);
    [mB,nB]=size(BestPath);
    MatchingCost=D(J,I)/mB;
    ylabel('Test Sequence');
    xlabel('Reference Sequence');
else
    BestPath=inf;
    MatchingCost=inf;
End
```

Ο κώδικας του DTWSakoe Matlab

```
function [MatchingCost,BestPath,D,Pred]=DTWSakoe(ref,test,genPlot)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FUNCTION
%   [MatchingCost,BestPath,D,Pred]=DTWSakoe(ref,test,genPlot)
% Computes the Dynamic Time Warping cost between two feature sequences. The first
% argument is the prototype, which is placed on the horizontal axis of the matching
% grid. The function employs the Sakoe-Chiba local constraints on
```

```
% a cost grid, where the Euclidean distance has been used as the
% distance metric. No end-point constraints have been adopted.
% This function calls BackTracking.m to extract the best path.
%
% INPUT ARGUMENTS:
%   ref:            reference sequence. Its size is mxI, where m is
%                   the number of features and I the number of feature vectors.
%   test:           test sequence. Its size is mxJ, where m is
%                   the number of features and J the number of feature vectors.
%   genPlot:        if set to 1, a plot of the best path is generated.
%
% OUTPUT ARGUMENTS:
%   MatchingCost:   the matching cost is normalized, i.e., it is divided with the length of the
best path.
%   BestPath:       backtracking path. Each node of the best path is represented as a complex
%                   number, where the real part stands for the row index and the
%                   imaginary part stands for the column index of the node.
%   D:              cost grid. Its size is JxI.
%   Pred:           matrix of node predecessors. The real part of Pred(j,i) is the row
%                   index of the predecessor of node (j,i) and the imaginary part of Pred(j,i)
%                   is the column index of the predecessor of node (j,i).
%
% (c) 2010 S. Theodoridis, A. Pikrakis, K. Koutroumbas, D. Cavouras
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Initialization
[D1,I]=size(ref);
[D2,J]=size(test);

if D1~=D2
    msgbox('Mismatch of dimensions between feature sequences');
    MatchingCost=[]; BestPath=[]; D=[]; Pred=[];
    return;
end


NodeCost=zeros(J,I);
Pred=zeros(J,I);
D=zeros(J,I);

for j=1:J
    for i=1:I
        %Euclidean distance
        NodeCost(j,i)=sqrt(sum((ref(:,i)-test(:,j)).^2));
    end
end

%Initialization
D(1,1)=NodeCost(1,1);
Pred(1,1)=0;

for j=2:J
    D(j,1)=D(j-1,1)+NodeCost(j,1);
    Pred(j,1)=j-1 + sqrt(-1)*1;
end

for i=2:I
    D(1,i)=D(1,i-1)+NodeCost(1,i);
    Pred(1,i)=1 + sqrt(-1)*(i-1);
end

%Main Loop
for j=2:J
    for i=2:I
        %Sakoe-Chiba local path constraints
        [D(j,i),ind]=min([D(j-1,i-1) D(j-1,i) D(j,i-1)]+NodeCost(j,i));
        if ind==1
            Pred(j,i)=(j-1)+sqrt(-1)*(i-1);
```

```
        elseif ind==2
            Pred(j,i)=(j-1)+sqrt(-1)*(i);
        else
            Pred(j,i)=(j)+sqrt(-1)*(i-1);
        end
    end %for i=2:I
end %for j=2:J
%End of Main Loop

% Pred

BestPath=BackTracking(Pred,J,I,genPlot,ref,test);
[mB,nB]=size(BestPath);
MatchingCost=D(J,I)/mB;
ylabel('Test Sequence');
xlabel('Reference Sequence');
```

Ο κώδικας του BackTracking Matlab

```
function [BestPath]=BackTracking(Pred,startNodek,startNodel,genPlot,P,T)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% FUNCTION
%   [BestPath]=BackTracking(Pred,startNodek,startNodel,genPlot,P,T)
% Performs backtracking on a matrix of node predecessors and returns the
% extracted best path starting from node (startNodek,startNodel).
% The best path can be optionally plotted.
%
% INPUT ARGUMENTS:
%   Pred:       matrix of node predecessors. The real part of Pred(i,j) is the row
%               index of the predecessor of node (i,j). The imaginary part of Pred(i,j) is
%               the column index of the predecessor of node (i,j).
%   startNodek: row index of node from which backtracking starts.
%   startNodel: column index of node from which backtracking starts.
%   genPlot:    if set to 1, a plot of the best path is generated.
%   P:          reference sequence (horizontal axis). Needed for axis
%               labeling.
%   T:          test sequence (vertical axis). Neede for axis labeling.
%
% OUTPUT ARGUMENTS:
%   BestPath:   backtracking path, i.e., vector of nodes. Each node is represented
%               as a complex number where the real part stands for the row
%               index of the node and the imaginary part stands for the
%               column index of the node.
%
% (c) 2010 S. Theodoridis, A. Pikrakis, K. Koutroumbas, D. Cavouras
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Perform Backtracking
Node=startNodek+sqrt(-1)*startNodel;
BestPath=[Node];
while (Pred(real(Node),imag(Node))~=0)
    Node=Pred(real(Node),imag(Node));
    BestPath=[Node;BestPath];
end

%Plot the grid and best path

if genPlot==1
    hold
    cla
    stepx=round(length(P)/16)
    stepy=round(length(T)/25)
    for i=stepx:stepx:(length(P))
        if ~ischar(P(i))
            text(i-round(.5*stepx),-(stepy/2), num2str(P(i)),'FontSize',7);
        else
```

```
            text(i,0, P(i));
        end
    end

    for i=stepy:stepy:length(T)
        if ~ischar(T(i))
            text(-round(.7*stepx),i+7, num2str(round(T(i))),'FontSize',7);
%           text(-7,i-2, num2str(round(T(i))),'FontSize',7);
        else
            text(0,i, T(i));
        end
    end

    plot(imag(BestPath),real(BestPath),'g','LineWidth',1);
    plot(imag(BestPath),real(BestPath),'g.','LineWidth',1);

%   x=ceil((5+length(P))/10)*10;
%   y=ceil((5+length(T))/10)*10;
%   plot(handles.axes1,imag(BestPath),real(BestPath),'g','LineWidth',2);
%   plot(handles.axes1,imag(BestPath),real(BestPath),'g*','LineWidth',2);
%   axis off
%   ylabel('Test Sequence');
%   xlabel('Reference Sequence');
%   xlim([-20 (x+10)]);
%   ylim([-20 (y+10)]);
%   xlim('auto')
%   ylim('auto')

    hold off
end


function [Fmd,Mi]=f2mf(Fx)
% Function of
% Frequency to midi Frequency
%

Frm=(440/32)*2.^([0:120]/12);
L1=length(Fx);

for i=1:L1
    [near,ind]=min(abs( Frm - Fx(i)));
    Fmd(i)=Frm(ind);
    Mi(i)=ind;
end

Fmd=round(Fmd);

End


function [Notes,endtime] = midiInfo(midi,outputFormat,tracklist)
% [Notes,endtime] = midiInfo(midi,outputFormat,tracklist)
%
% Takes a midi structre and generates info on notes and messages
% Can return a matrix of note parameters and/or output/display
%   formatted table of messages
%
% Inputs:
%  midi - Matlab structure (created by readmidi.m)
%  tracklist - which tracks to show ([] for all)
%  outputFormat
%   - if it's a string write the formated output to the file
%   - if 0, don't display or write formatted output
%   - if 1, just display (default)
%
% outputs:
%   Notes - a matrix containing a list of notes, ordered by start time
```

```
%     column values are:
%      1     2    3  4   5  6  7       8
%     [track chan nn vel t1 t2 msgNum1 msgNum2]
%   endtime - time of end of track message
%

% Copyright (c) 2009 Ken Schutte
% more info at: http://www.kenschutte.com/midi

if nargin<3
  tracklist=[];
  if nargin<2
    outputFormat=1;
  end
end
if (isempty(tracklist))
  tracklist = 1:length(midi.track);
end

current_tempo = 500000;  % default tempo


[tempos, tempos_time] = getTempoChanges(midi);

% What to do if no tempos are given?
%  This seems at leat get things to work (see eire01.mid)
if length(tempos) == 0
  tempos = [current_tempo];
  tempos_time = [0];
end


fid = -1;
if (ischar(outputFormat))
  fid = fopen(outputFormat,'w');
end

endtime = -1;

% each row:
%  1     2    3  4   5  6  7      8
% [track chan nn vel t1 t2 msgNum1 msgNum2]
Notes = zeros(0,8);

for i=1:length(tracklist)
  tracknum = tracklist(i);

  cumtime=0;
  seconds=0;

  Msg = cell(0);
  Msg{1,1} = 'chan';
  Msg{1,2} = 'deltatime';
  Msg{1,3} = 'time';
  Msg{1,4} = 'name';
  Msg{1,5} = 'data';

  for msgNum=1:length(midi.track(tracknum).messages)

    currMsg = midi.track(tracknum).messages(msgNum);

    midimeta  = currMsg.midimeta;
    deltatime = currMsg.deltatime;
    data      = currMsg.data;
    type      = currMsg.type;
    chan      = currMsg.chan;

    cumtime = cumtime + deltatime;
```

```
seconds = seconds + deltatime*1e-6*current_tempo/midi.ticks_per_quarter_note;

[mx ind] = max(find(cumtime >= tempos_time));
current_tempo = tempos(ind);

% find start/stop of notes:
%    if (strcmp(name,'Note on') && (data(2)>0))
% note on with vel>0:
if (midimeta==1 && type==144 && data(2)>0)
  % note on:
   Notes(end+1,:) = [tracknum chan data(1) data(2) seconds 0 msgNum -1];
   %    elseif ((strcmp(name,'Note on') && (data(2)==0)) || strcmp(name,'Note off'))
   % note on with vel==0 or note off:
elseif (midimeta==1 && ( (type==144 && data(2)==0) || type==128 ))

  % note off:
  %      % find index, wther tr,chan,and nn match, and not complete

  ind = find((...
      (Notes(:,1)==tracknum) + ...
      (Notes(:,2)==chan) + ...
      (Notes(:,3)==data(1)) + ...
      (Notes(:,8)==-1)...
      )==4);

  if (length(ind)==0)
    %% was an error before; change to warning and ignore the message.
    warning('ending non-open note?');

  elseif (length(ind)>1)
    %% ??? not sure about this...
    %%disp('warning: found mulitple matches in endNote, taking first...');
    ind = ind(1);

  else

    % set info on ending:
    Notes(ind,6) = seconds;
    Notes(ind,8) = msgNum;

  end


  % end of track:
elseif (midimeta==0 && type==47)
  if (endtime == -1)
    endtime = seconds;
  else
    disp('two "end of track" messages?');
    endtime(end+1) = seconds;
  end


end

% we could check to make sure it ends with
%  'end of track'


if (outputFormat ~= 0)
  % get some specific descriptions:
  name = num2str(type);
  dataStr = num2str(data);

  if (isempty(chan))
    Msg{msgNum,1} = '-';
  else
    Msg{msgNum,1} = num2str(chan);
```

```
      end

      Msg{msgNum,2} = num2str(deltatime);
      Msg{msgNum,3} = formatTime(seconds);

      if (midimeta==0)
        Msg{msgNum,4} = 'meta';
      else
        Msg{msgNum,4} = '';
      end

      [name,dataStr] = getMsgInfo(midimeta, type, data);
      Msg{msgNum,5} = name;
      Msg{msgNum,6} = dataStr;
    end


  end

  if (outputFormat ~= 0)
    printTrackInfo(Msg,tracknum,fid);
  end

end

% make this an option!!!
% - I'm not sure why it's needed...
% remove start silence:
first_t = min(Notes(:,5));
Notes(:,5) = Notes(:,5) - first_t;
Notes(:,6) = Notes(:,6) - first_t;

% sort Notes by start time:
[junk,ord] = sort(Notes(:,5));
Notes = Notes(ord,:);


if (fid ~= -1)
  fclose(fid);
end

function printTrackInfo(Msg,tracknum,fid)

% make cols same length instead of just using \t
for i=1:size(Msg,2)
  maxLen(i)=0;
  for j=1:size(Msg,1)
    if (length(Msg{j,i})>maxLen(i))
      maxLen(i) = length(Msg{j,i});
    end
  end
end


s='';
s=[s sprintf('-------------------------------------------------\n')];
s=[s sprintf('Track %d\n',tracknum)];
s=[s sprintf('-------------------------------------------------\n')];

if (fid == -1)
  disp(s)
else
  fprintf(fid,'%s',s);
end


for i=1:size(Msg,1)
  line='';
```

```
  for j=1:size(Msg,2)
    sp = repmat(' ',1,5+maxLen(j)-length(Msg{i,j}));
    m = Msg{i,j};
    m = m(:)';  % ensure column vector
%    line = [line Msg{i,j} sp];
    line = [line m sp];
  end

  if (fid == -1)
    disp(line)
  else
    fprintf(fid,'%s\n',line);
  end

end


function s=formatTime(seconds)

minutes = floor(seconds/60);
secs = seconds - 60*minutes;

s = sprintf('%d:%2.3f',minutes,secs);

function [name,dataStr]=getMsgInfo(midimeta, type, data);

% meta events:
if (midimeta==0)
  if     (type==0);  name = 'Sequence Number';          len=2;  dataStr = num2str(data);
  elseif (type==1);  name = 'Text Events';             len=-1; dataStr = char(data);
  elseif (type==2);  name = 'Copyright Notice';        len=-1; dataStr = char(data);
  elseif (type==3);  name = 'Sequence/Track Name';     len=-1; dataStr = char(data);
  elseif (type==4);  name = 'Instrument Name';         len=-1; dataStr = char(data);
  elseif (type==5);  name = 'Lyric';                   len=-1; dataStr = char(data);
  elseif (type==6);  name = 'Marker';                  len=-1; dataStr = char(data);
  elseif (type==7);  name = 'Cue Point';               len=-1; dataStr = char(data);
  elseif (type==32); name = 'MIDI Channel Prefix';     len=1;  dataStr = num2str(data);
  elseif (type==47); name = 'End of Track';            len=0;  dataStr = '';
  elseif (type==81); name = 'Set Tempo';               len=3;
    val = data(1)*16^4+data(2)*16^2+data(3); dataStr = ['microsec per quarter note: '
num2str(val)];
  elseif (type==84); name = 'SMPTE Offset';            len=5;
    dataStr = ['[hh mm ss fr ff]=' num2str(data)];
  elseif (type==88); name = 'Time Signature';          len=4;
    dataStr = [num2str(data(1)) '/' num2str(data(2)) ', clock ticks and notated 32nd notes='
num2str(data(3)) '/' num2str(data(4))];
  elseif (type==89); name = 'Key Signature';           len=2;
    % num sharps/flats (flats negative)
    if (data(1)>=0)
       %       1   2   3   4   5     6    7
      ss={'C','G','D', 'A', 'E','B',  'F#', 'C#'};
      dataStr = ss{data(1)+1};
    else
       %    1   2   3   4   5     6    7
       ss={'F','Bb','Eb','Ab','Db','Gb','Cb'};
       dataStr = ss{abs(data(1))};
    end
    if (data(2)==0)
      dataStr = [dataStr ' Major'];
    else
      dataStr = [dataStr ' Minor'];
    end

  elseif (type==89); name = 'Sequencer-Specific Meta-event';   len=-1;
    dataStr = char(data);
    % !! last two conflict...

  else
```

```
    name = ['UNKNOWN META EVENT: ' num2str(type)]; dataStr = num2str(data);
  end

% meta 0x21 = MIDI port number, length 1 (? perhaps)
else

  % channel voice messages:
  %  (from event byte with chan removed, eg 0x8n -> 0x80 = 128 for
  %  note off)
  if     (type==128);  name = 'Note off';                 len=2; dataStr = ['nn='
num2str(data(1)) '  vel=' num2str(data(2))];
  elseif (type==144);  name = 'Note on';                  len=2; dataStr = ['nn='
num2str(data(1)) '  vel=' num2str(data(2))];
  elseif (type==160); name = 'Polyphonic Key Pressure';   len=2; dataStr = ['nn='
num2str(data(1)) '  vel=' num2str(data(2))];
  elseif (type==176); name = 'Controller Change';         len=2; dataStr = ['ctrl='
controllers(data(1)) '  value=' num2str(data(2))];
  elseif (type==192); name = 'Program Change';            len=1; dataStr = ['instr='
num2str(data)];
  elseif (type==208); name = 'Channel Key Pressure';      len=1; dataStr = ['vel='
num2str(data)];
  elseif (type==224); name = 'Pitch Bend';                len=2;
    val = data(1)+data(2)*256;
    val = base2dec('2000',16) - val;
    dataStr = ['change=' num2str(val) '?'];

  % channel mode messages:
  %  ... unsure about data for these... (do some have a data byte and
  %  others not?)
  %
  % 0xC1 .. 0xC8
  elseif (type==193);  name = 'All Sounds Off';          dataStr = num2str(data);
  elseif (type==194);  name = 'Reset All Controllers';   dataStr = num2str(data);
  elseif (type==195); name = 'Local Control';            dataStr = num2str(data);
  elseif (type==196); name = 'All Notes Off';            dataStr = num2str(data);
  elseif (type==197); name = 'Omni Mode Off';            dataStr = num2str(data);
  elseif (type==198); name = 'Omni Mode On';             dataStr = num2str(data);
  elseif (type==199); name = 'Mono Mode On';             dataStr = num2str(data);
  elseif (type==200); name = 'Poly Mode On';             dataStr = num2str(data);

    % sysex, F0->F7
  elseif (type==240); name = 'Sysex 0xF0';             dataStr = num2str(data);
  elseif (type==241); name = 'Sysex 0xF1';             dataStr = num2str(data);
  elseif (type==242); name = 'Sysex 0xF2';             dataStr = num2str(data);
  elseif (type==243); name = 'Sysex 0xF3';             dataStr = num2str(data);
  elseif (type==244); name = 'Sysex 0xF4';             dataStr = num2str(data);
  elseif (type==245); name = 'Sysex 0xF5';             dataStr = num2str(data);
  elseif (type==246); name = 'Sysex 0xF6';             dataStr = num2str(data);
  elseif (type==247); name = 'Sysex 0xF7';             dataStr = num2str(data);

    % realtime
    % (i think have no data..?)
  elseif (type==248); name = 'Real-time 0xF8 - Timing clock';             dataStr =
num2str(data);
  elseif (type==249); name = 'Real-time 0xF9';              dataStr = num2str(data);
  elseif (type==250); name = 'Real-time 0xFA - Start a sequence';             dataStr =
num2str(data);
  elseif (type==251); name = 'Real-time 0xFB - Continue a sequence';              dataStr =
num2str(data);
  elseif (type==252); name = 'Real-time 0xFC - Stop a sequence';             dataStr =
num2str(data);
  elseif (type==253); name = 'Real-time 0xFD';              dataStr = num2str(data);
  elseif (type==254); name = 'Real-time 0xFE';              dataStr = num2str(data);
  elseif (type==255); name = 'Real-time 0xFF';              dataStr = num2str(data);


  else
    name = ['UNKNOWN MIDI EVENT: ' num2str(type)]; dataStr = num2str(data);
```

```
  end

end

function s=controllers(n)
if (n==1); s='Mod Wheel';
elseif (n==2); s='Breath Controllery';
elseif (n==4); s='Foot Controller';
elseif (n==5); s='Portamento Time';
elseif (n==6); s='Data Entry MSB';
elseif (n==7); s='Volume';
elseif (n==8); s='Balance';
elseif (n==10); s='Pan';
elseif (n==11); s='Expression Controller';
elseif (n==16); s='General Purpose 1';
elseif (n==17); s='General Purpose 2';
elseif (n==18); s='General Purpose 3';
elseif (n==19); s='General Purpose 4';
elseif (n==64); s='Sustain';
elseif (n==65); s='Portamento';
elseif (n==66); s='Sustenuto';
elseif (n==67); s='Soft Pedal';
elseif (n==69); s='Hold 2';
elseif (n==80); s='General Purpose 5';
elseif (n==81); s='Temp Change (General Purpose 6)';
elseif (n==82); s='General Purpose 7';
elseif (n==83); s='General Purpose 8';
elseif (n==91); s='Ext Effects Depth';
elseif (n==92); s='Tremelo Depthy';
elseif (n==93); s='Chorus Depth';
elseif (n==94); s='Detune Depth (Celeste Depth)';
elseif (n==95); s='Phaser Depth';
elseif (n==96); s='Data Increment (Data Entry +1)';
elseif (n==97); s='Data Decrement (Data Entry -1)';
elseif (n==98); s='Non-Registered Param LSB';
elseif (n==99); s='Non-Registered Param MSB';
elseif (n==100); s='Registered Param LSB';
elseif (n==101); s='Registered Param MSB';
else
  s='UNKNOWN CONTROLLER';
end

%Channel mode message values
%Reset All Controllers   79      121     Val ??
%Local Control   7A      122     Val 0 = off, 7F (127) = on
%All Notes Off   7B      123     Val must be 0
%Omni Mode Off   7C      124     Val must be 0
%Omni Mode On    7D      125     Val must be 0
%Mono Mode On    7E      126     Val = # of channels, or 0 if # channels equals # voices in
receiver
%Poly Mode On    7F      127     Val must be 0
```

Ο κώδικας readmidi Matlab

```
function midi = readmidi(filename, rawbytes)
% midi = readmidi(filename, rawbytes)
% midi = readmidi(filename)
%
% Read MIDI file and store in a Matlab structure
% (use midiInfo.m to see structure detail)
%
% Inputs:
%  filename - input MIDI file
%  rawbytes - 0 or 1: Include raw bytes in structure
%             This info is redundant, but can be
%             useful for debugging. default=0
```

```
%

% Copyright (c) 2009 Ken Schutte
% more info at: http://www.kenschutte.com/midi


if (nargin<2)
  rawbytes=0;
end

fid = fopen(filename);
[A count] = fread(fid,'uint8');
fclose(fid);

if (rawbytes) midi.rawbytes_all = A; end

% realtime events: status: [F8, FF].  no data bytes
%clock, undefined, start, continue, stop, undefined, active
%sensing, systerm reset

% file consists of "header chunk" and "track chunks"
%   4B  'MThd' (header) or 'MTrk' (track)
%   4B  32-bit unsigned int = number of bytes in chunk, not
%       counting these first 8

% HEADER CHUNK -----------------------------------------------------
% 4B 'Mthd'
% 4B length
% 2B file format
%    0=single track, 1=multitrack synchronous, 2=multitrack asynchronous
%    Synchronous formats start all tracks at the same time, while asynchronous formats can start
and end any track at any time during the score.
% 2B track cout (must be 1 for format 0)
% 2B num delta-time ticks per quarter note
%

if ~isequal(A(1:4)',[77 84 104 100])  % double('MThd')
    error('File does not begin with header ID (MThd)');
end

header_len = decode_int(A(5:8));
if (header_len == 6)
else
    error('Header length != 6 bytes.');
end

format = decode_int(A(9:10));
if (format==0 || format==1 || format==2)
     midi.format = format;
else
    error('Format does not equal 0,1,or 2');
end

num_tracks = decode_int(A(11:12));
if (format==0 && num_tracks~=1)
    error('File is format 0, but num_tracks != 1');
end

time_unit = decode_int(A(13:14));
if (bitand(time_unit,2^15)==0)
  midi.ticks_per_quarter_note = time_unit;
else
  error('Header: SMPTE time format found - not currently supported');
end

if (rawbytes)
  midi.rawbytes_header = A(1:14);
end
```

```
% end header parse ---------------------------------------------------

% BREAK INTO SEPARATE TRACKS -----------------------------------------
% midi.track(1).data = [byte byte byte ...];
% midi.track(2).date = ...
% ...
%
% Track Chunks---------
% 4B 'MTrk'
% 4B length (after first 8B)
%
ctr = 15;
for i=1:num_tracks

  if ~isequal(A(ctr:ctr+3)',[77 84 114 107])  % double('MTrk')
    error(['Track ' num2str(i) ' does not begin with track ID=MTrk']);
  end
  ctr = ctr+4;

  track_len = decode_int(A(ctr:ctr+3));
  ctr = ctr+4;

  % have track.rawbytes hold initial 8B also...
  track_rawbytes{i} = A((ctr-8):(ctr+track_len-1));

  if (rawbytes)
    midi.track(i).rawbytes_header = A(ctr-8:ctr-1);
  end

  ctr = ctr+track_len;
end
% ---------------------------------------------------------------------

% Events:
%  - meta events: start with 'FF'
%  - MIDI events: all others

% MIDI events:
%  optional command byte + 0,1,or 2 bytes of parameters
%  "running mode": command byte omitted.
%
% all midi command bytes have MSB=1
% all data for inside midi command have value <= 127 (ie MSB=0)
% -> so can determine running mode
%
% meta events' data may have any values (meta events have to set
% len)
%

% 'Fn' MIDI commands:
%  no chan. control the entire system
%F8 Timing Clock
%FA start a sequence
%FB continue a sequence
%FC stop a sequence

% Meta events:
%  1B 0xFF
%  1B event type
%  1B length of additional data
%  ?? additional data
%


% "channel mode messages"
% have same code as "control change": 0xBn
%  but uses reserved controller numbers 120-127
```

```
%
%Midi events consist of an optional command byte
% followed by zero, one or two bytes of parameters.
% In running mode, the command can be omitted, in
% which case the last MIDI command specified is
% assumed.  The first bit of a command byte is 1,
% while the first bit of a parameter is always 0.
%   In addition, the last 4 bits of a command
%   indicate the channel to which the event should
%   be sent; therefore, there are 6 possible
%   commands (really 7, but we will discuss the x'Fn'
%   commands later) that can be specified.  They are:

% parse tracks ----------------------------------------
for i=1:num_tracks

  track = track_rawbytes{i};

  if (rawbytes); midi.track(i).rawbytes = track; end

  msgCtr = 1;
  ctr=9;  % first 8B were MTrk and length
  while (ctr < length(track_rawbytes{i}))

    clear currMsg;
    currMsg.used_running_mode = 0;
    % note:
    %  .used_running_mode is necessary only to
    %  be able to reconstruct a file _exactly_ from
    %  the 'midi' structure.  this is helpful for
    %  debugging since write(read(filename)) can be
    %  tested for exact replication...
    %

    ctr_start_msg = ctr;

    [deltatime,ctr] = decode_var_length(track, ctr);

    % ?
    %if (rawbytes)
    %  currMsg.rawbytes_deltatime = track(ctr_start_msg:ctr-1);
    %end

    % deltaime must be 1-4 bytes long.
    % could check here...


    % CHECK FOR META EVENTS ------------------------
    % 'FF'
    if track(ctr)==255

      type = track(ctr+1);

      ctr = ctr+2;

      % get variable length 'length' field
      [len,ctr] = decode_var_length(track, ctr);

      % note: some meta events have pre-determined lengths...
      %  we could try verifiying they are correct here.

      thedata = track(ctr:ctr+len-1);
      chan = [];

      ctr = ctr + len;
      midimeta = 0;
    else
      midimeta = 1;
```

```
% MIDI EVENT ----------------------------

% check for running mode:
if (track(ctr)<128)

  % make it re-do last command:
  %ctr = ctr - 1;
  %track(ctr) = last_byte;
  currMsg.used_running_mode = 1;

  B = last_byte;
  nB = track(ctr); % ?

else

  B  = track(ctr);
  nB = track(ctr+1);

  ctr = ctr + 1;

end

% nibbles:
%B  = track(ctr);
%nB = track(ctr+1);


Hn = bitshift(B,-4);
Ln = bitand(B,15);

chan = [];

msg_type = midi_msg_type(B,nB);

% DEBUG:
if (i==2)
    if (msgCtr==1)
      disp(msg_type);
    end
end


switch msg_type

 case 'channel_mode'

  % UNSURE: if all channel mode messages have 2 data byes (?)
  type = bitshift(Hn,4) + (nB-120+1);
  thedata = track(ctr:ctr+1);
  chan = Ln;

  ctr = ctr + 2;

  % ---- channel voice messages:
 case 'channel_voice'

  type = bitshift(Hn,4);
  len = channel_voice_msg_len(type); % var length data:
  thedata = track(ctr:ctr+len-1);
  chan = Ln;

  % DEBUG:
  if (i==2)
    if (msgCtr==1)
      disp([999  Hn type])
    end
  end
```

```
        ctr = ctr + len;

       case 'sysex'

        % UNSURE: do sysex events (F0-F7) have
        %  variable length 'length' field?

        [len,ctr] = decode_var_length(track, ctr);

        type = B;
        thedata = track(ctr:ctr+len-1);
        chan = [];

        ctr = ctr + len;

       case 'sys_realtime'

        % UNSURE: I think these are all just one byte
        type = B;
        thedata = [];
        chan = [];

      end

      last_byte = Ln + bitshift(Hn,4);

    end % end midi event 'if'


    currMsg.deltatime = deltatime;
    currMsg.midimeta = midimeta;
    currMsg.type = type;
    currMsg.data = thedata;
    currMsg.chan = chan;

    if (rawbytes)
      currMsg.rawbytes = track(ctr_start_msg:ctr-1);
    end

    midi.track(i).messages(msgCtr) = currMsg;
    msgCtr = msgCtr + 1;


  end % end loop over rawbytes
end % end loop over tracks


function val=decode_int(A)

val = 0;
for i=1:length(A)
  val = val + bitshift(A(length(A)-i+1), 8*(i-1));
end

function len=channel_voice_msg_len(type)

if     (type==128); len=2;
elseif (type==144); len=2;
elseif (type==160); len=2;
elseif (type==176); len=2;
elseif (type==192); len=1;
elseif (type==208); len=1;
elseif (type==224); len=2;
else
  disp(type); error('bad channel voice message type');
end

%
```

```
% decode variable length field (often deltatime)
%
%  return value and new position of pointer into 'bytes'
%
function [val,ptr] = decode_var_length(bytes, ptr)

keepgoing=1;
binarystring = '';
while (keepgoing)
  % check MSB:
  %  if MSB=1, then delta-time continues into next byte...
  if(~bitand(bytes(ptr),128))
    keepgoing=0;
  end
  % keep appending last 7 bits from each byte in the deltatime:
  binbyte = ['00000000' dec2base(bytes(ptr),2)];
  binarystring = [binarystring binbyte(end-6:end)];
  ptr=ptr+1;
end
val = base2dec(binarystring,2);


%
% Read first 2 bytes of msg and
%  determine the type
%  (most require only 1st byte)
%
% str is one of:
%  'channel_mode'
%  'channel_voice'
%  'sysex'
%  'sys_realtime'
%
function str=midi_msg_type(B,nB)

Hn = bitshift(B,-4);
Ln = bitand(B,7);

% ---- channel mode messages:
%if (Hn==11 && nB>=120 && nB<=127)
if (Hn==11 && nB>=122 && nB<=127)
  str = 'channel_mode';

  % ---- channel voice messages:
elseif (Hn>=8 && Hn<=14)
  str = 'channel_voice';

  %  ---- sysex events:
elseif (Hn==15 && Ln>=0 && Ln<=7)
  str = 'sysex';

  % system real-time messages
elseif (Hn==15 && Ln>=8 && Ln<=15)
  % UNSURE: how can you tell between 0xFF system real-time
  %   message and 0xFF meta event?
  %   (now, it will always be processed by meta)
  str = 'sys_realtime';

else
  % don't think it can get here...
  error('bad midi message');
end

function [cross,div]=gridplot(ref_mid,path,s)

ps=ref_mid(:,5)'
pe=ref_mid(:,6)'
ps=ps*2*s;
```

```
pe=pe*2*s;
len=length(path)
lx=imag(path(len))
ly=real(path(len))
hold

plot([0,lx+1],[0,0])
plot([0,0],[0,ly+1])
plot([0,lx+1],[ly+1,ly+1])
plot([lx+1,lx+1],[0,ly+1])

plot([-(.05*lx),(1.05*lx)],[-(.05*ly),-(.05*ly)],'Color','red')
plot([-(.05*lx),-(.05*lx)],[-(.05*ly),(1.05*ly)],'Color','red')
plot([-(.05*lx),(1.05*lx)],[(1.05*ly),(1.05*ly)],'Color','red')
plot([(1.05*lx),(1.05*lx)],[-(.05*ly),(1.05*ly)],'Color','red')

plot([ps(1),ps(1)],[0,ly])
plot([0,lx],[0,0])
cross=path(1);
for i=2:length(ps)

    if pe(i-1)==ps(i)
        plot([pe(i-1)-1,pe(i-1)-1],[1,ly],'color','c')
        plot([ps(i),ps(i)],[0,ly+1])
    else
        plot([pe(i-1),pe(i-1)],[1,ly],'color','c')
        plot([ps(i),ps(i)],[0,ly+1])
    end

    for j=ps(i):len
        if imag(path(j))==ps(i)
            line=real(path(j))
            plot([0,lx+1],[line,line])
            cross=[cross();path(j)];
            break
        end
    end
    x=imag(cross(i-1))+round((imag(cross(i))-imag(cross(i-1)))/2)
    y=real(cross(i-1))+round((real(cross(i))-real(cross(i-1)))/2)
    text(x,y, num2str(i-1),'FontSize',8,'color','b');
    ax=imag(cross(i))-imag(cross(i-1));
    ay=real(cross(i))-real(cross(i-1));
    div(i-1)=round(100*(ay/ax));
    str=num2str(div(i-1));
    text(x-round(.05*ax),y-round(.3*ay),[str,'%'],'FontSize',7,'color','b');
    if (div(i-1)>135 || div(i-1)<65)
        text(x-round(.4*ax),y+round(.3*ay),['x'],'FontSize',7,'color','r');
    else
        text(x-round(.4*ax),y+round(.3*ay),['!'],'FontSize',7,'color','b');
    end
end
cross=[cross();path(len)]
plot([pe(i),pe(i)],[1,ly],'color','c')
x=imag(cross(i))+round((imag(cross(i+1))-imag(cross(i)))/2)
y=real(cross(i))+round((real(cross(i+1))-real(cross(i)))/2)
text(x,y, num2str(i),'FontSize',8,'color','b');
ax=imag(cross(i+1))-imag(cross(i));
ay=real(cross(i+1))-real(cross(i));
div(i)=round(100*(ay/ax));
str=num2str(div(i));
text(x-round(.05*ax),y-round(.3*ay),[str,'%'],'FontSize',7,'color','b');
if (div(i)>135 || div(i)<65)
    text(x-round(.4*ax),y+round(.3*ay),'x','FontSize',7,'color','r');
else
    text(x-round(.4*ax),y+round(.3*ay),'!','FontSize',7,'color','b');
end

hold off
```

```
function Hd = hpe
%HPE Returns a discrete-time filter object.

%
% M-File generated by MATLAB(R) 7.9 and the Signal Processing Toolbox 6.12.
%
% Generated on: 21-May-2013 13:21:53
%

% Equiripple Highpass filter designed using the FIRPM function.

% All frequency values are in Hz.
Fs = 10000;  % Sampling Frequency

N     = 101;  % Order
Fstop = 110;  % Stopband Frequency
Fpass = 220;  % Passband Frequency
Wstop = 1;    % Stopband Weight
Wpass = 100;  % Passband Weight
dens  = 20;   % Density Factor

% Calculate the coefficients using the FIRPM function.
b  = firpm(N, [0 Fstop Fpass Fs/2]/(Fs/2), [0 0 1 1], [Wstop Wpass], ...
           {dens});
Hd = dfilt.dffir(b);
set(Hd, 'Arithmetic', 'double');

% [EOF]
```

Βιβλιογραφία

Introduction to Pattern Recognition: A MATLAB Approach
Aggelos Pikrakis - Sergios Theodoridis - Dionisis Cavouras - Konstantinos Koutroumbas
Apr 2010

Αναγνώριση Προτύπων \
Σ. Θεοδωρίδης – Κ. Κουτρουμπάς
2012

Theory and Applications of Digital Speech Processing
Lawrence Rabiner, Ronald Schafer
March 2010

Ψηφιακή επεξεργασία φωνής : Θεωρία και εφαρμογές
Πικράκης Άγγελος , Καρπούζης Κωνσταντίνος

A Level Building Dynamic Time Warping Algorithm for Connected Word Recognition
IEEE Transactions on Acoustics, Speech, and Signal Processing
Cory S. Myers, and Lawrence Rabiner
April 1981

Dynamic Time Warping
Information Retrieval for Music and Motion
Muller M.
2007

2009 Ken Schutte
http://www.kenschutte.com/midi