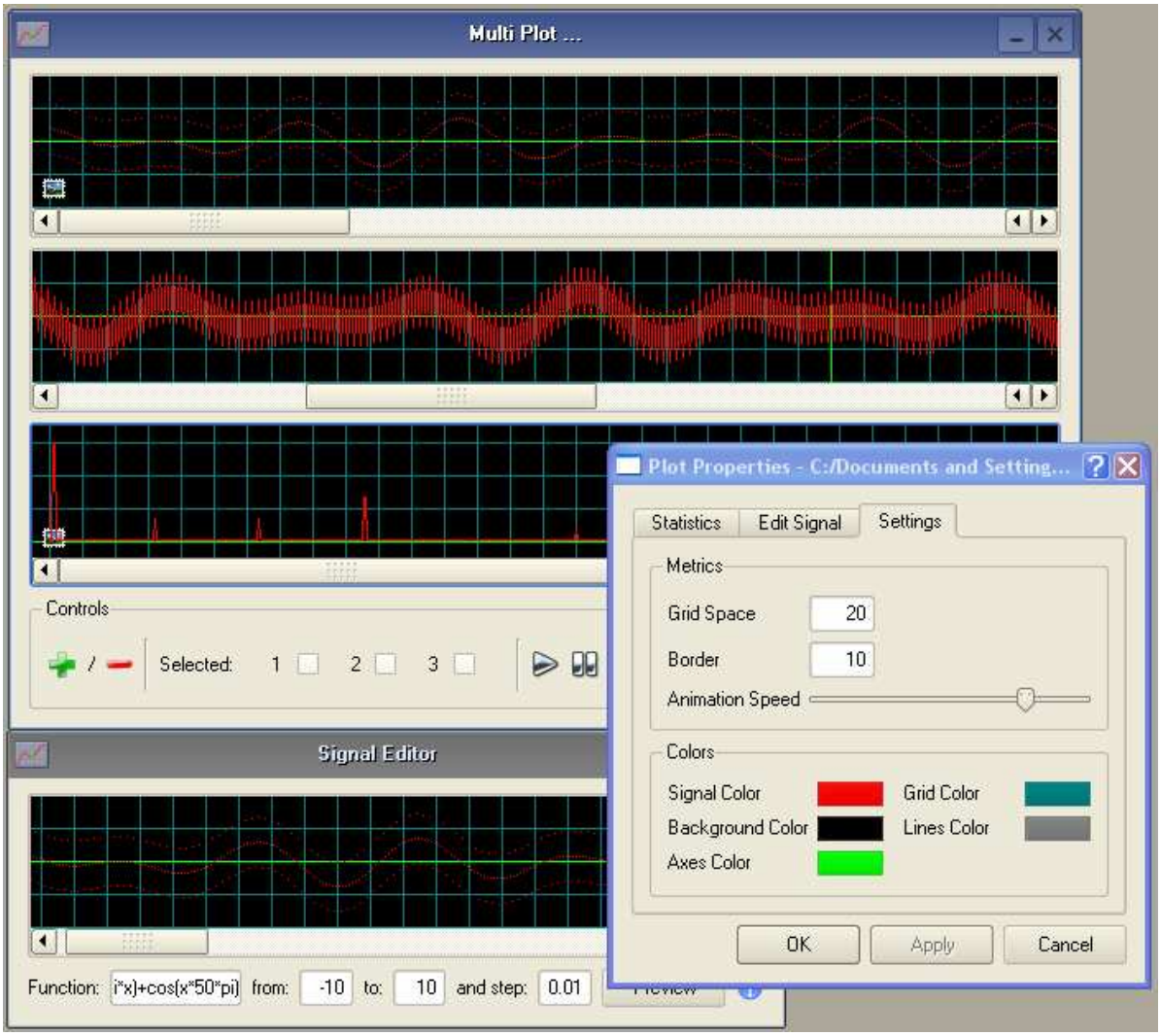
ΑΤΕΙ ΘΕΣΣΑΛΟΝΙΚΗΣ - ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Ανάπτυξη Εκπαιδευτικού Λογισμικού Ψηφιακής Επεξεργασίας Σήματος



ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ανδρέας Μποντόζογλου

Επιβλέπων καθηγητής: Αθανάσιος Ι. Μάργαρης

ΘΕΣΣΑΛΟΝΙΚΗ 2008

Περιεχόμενα

1. Περίληψη

Σκοπός της εργασίας αυτής είναι η ανάπτυξη μιας εκπαιδευτικής εφαρμογής για την συμβολή στην παρουσίαση του μαθήματος «Επεξεργασία Σήματος και Εικόνας». Η εφαρμογή επικεντρώνεται στα σήματα, δίνεται η δυνατότητα ανάγνωσης αποθηκευμένων σημάτων και παρουσιάζεται η γραφική τους αναπαράσταση. Ωστόσο η ανάγνωση των σημάτων από αρχείο δεν είναι υποχρεωτική αφού μπορούν να δημιουργηθούν καινούρια σήματα, από την εφαρμογή, βάσει της μαθηματικής τους αναπαράστασης. Η αναπαράσταση των σημάτων μπορεί να είναι είτε αναλογική είτε ψηφιακή, και στις δύο περιπτώσεις υπάρχει η δυνατότητα animation. Διάφοροι μετασχηματισμοί και πράξεις μπορούν να εφαρμοστούν σε κάθε σήμα ξεχωριστά ή μεταξύ σημάτων. Τέλος ο χρήστης μπορεί να αλλάξει τα χρώματα των διαγραμμάτων και να τα αποθηκεύσει σε μορφή εικόνας, ώστε να μπορούν εύκολα να ενσωματωθούν σε κείμενα και ιστοσελίδες.

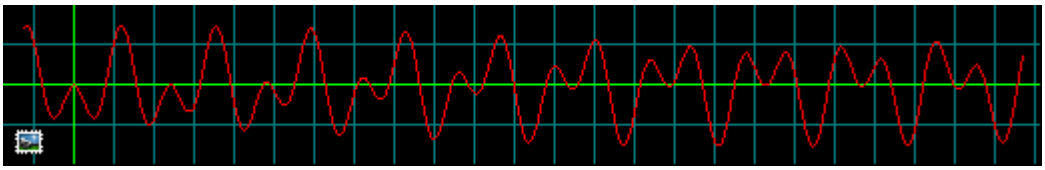
Για την υλοποίηση της εφαρμογής επιλέχθηκε η γλώσσα προγραμματισμού c++ λόγο της δυνατότητας ανάπτυξης αντικειμενοστρεφών εφαρμογών καθώς και της αρκετά καλής ταχύτητας εκτέλεσης υπολογισμών. Όσο αφορά την διεπιφάνεια χρήστη (το παραθυρικό μέρος της εφαρμογής) επιλέξαμε το Qt framework της Trolltech, το οποίο παρέχει τη δυνατότητα μεταγλώτισσης και εκτέλεσης της εφαρμογής σε πολλές πλατφόρμες διαφόρων αρχιτεκτονικών, σε αντίθεση με άλλα περιβάλλοντα σχεδιασμένα για ένα μόνο λειτουργικό σύστημα.

2. Βασικές Έννοιες – Σήματα [περιεχόμενα]

Σε αυτή την ενότητα παρουσιάζονται και αναλύονται οι βασικότερες έννοιες γύρω από τη θεωρία των σημάτων, οι οποίες είναι απαραίτητες για την κατανόηση και χρήση τις εφαρμογής. Θα αναφερθούμε στους τύπους σημάτων, στις θεμελιώδεις πράξεις μεταξύ σημάτων καθώς και στις ιδιότητές τους.

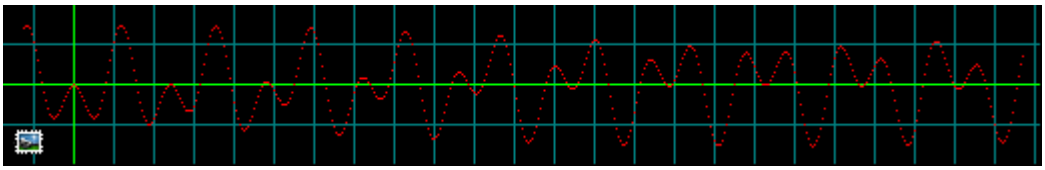
2.1 Αναλογικά και ψηφιακά σήματα [περιεχόμενα]

Τα σήματα χωρίζονται σε δύο βασικές κατηγορίες, σε αναλογικά και σε ψηφιακά. Τα αναλογικά σήματα είναι συναρτήσεις συνεχούς μεταβλητής και παίρνουν τιμές που ανήκουν σε μια συνεχή περιοχή τιμών.



Εικόνα 1: Αναλογικό σήμα

Τα ψηφιακά σήματα είναι συναρτήσεις διακριτής μεταβλητής και παίρνουν τιμές οι οποίες ανήκουν σε μια ομάδα από διακριτές στάθμες. Για παράδειγμα, η ανεξάρτητη μεταβλητή μπορεί να παίρνει τις τιμές 1, 2, 3, 4, 5... και το σήμα να παίρνει κάθε φορά τιμές από τις διακριτές στάθμες -1, -0.5, 0, 0.5 1. Παρακάτω παρουσιάζεται το αντίστοιχο ψηφιακό σήμα τη εικόνας 1.



Εικόνα 2: Ψηφιακό σήμα

2.2 Σήματα διακριτού χρόνου [περιεχόμενα]

Τα σήματα διακριτού χρόνου ορίζονται ως συναρτήσεις μίας ανεξάρτητης ακέραιας μεταβλητής. Η μεταβλητή αυτή δεν είναι υποχρεωτικό να αναπαριστά χρόνο αλλά ένα οποιοδήποτε φυσικό μέγεθος. Η διαφορά των σημάτων διακριτού χρόνου με τα ψηφιακά σήματα είναι ότι τα διακριτά σήματα παίρνουν οποιεσδήποτε τιμές, οι οποίες δεν ανήκουν υποχρεωτικά σε ένα σύνολο από προκαθορισμένες στάθμες. Ένα σήμα διακριτού χρόνου, πέρα από την γραφική αναπαράσταση, μπορεί να αναπαρασταθεί με τους παρακάτω τρεις τρόπους:

- **Συναρτησιακή αναπαράσταση**, στην οποία καθορίζουμε την τιμή του σήματος για κάθε περιοχή τιμών της ανεξάρτητης μεταβλητής n.

$$x(n)=\begin{cases}5, & n=-2,-1\\ 3, & n=0\\ -2, & n=1,2,3\end{cases}$$

- **Αναπαράσταση σε μορφή πίνακα**, στην οποία καταγράφουμε την τιμή του σήματος για κάθε τιμή της ανεξάρτητης μεταβλητής n.

n	...	-3	-2	-1	0	1	...
x(n)	...	15	-9	-2	2	-6	...

- **Ακολουθιακή αναπαράσταση**, στην οποία καταγράφουμε τις τιμές του σήματος σε μορφή ακολουθίας χρησιμοποιώντας το σύμβολο (↑) για να δείξουμε την τιμή του σήματος για n=0.

$$x(n) = \{...0,\ 3,\ -1,\ -5,\ 5,\ 3\ ,\ 0,\ 0,\ -2,\ 8\}$$
$$\uparrow$$

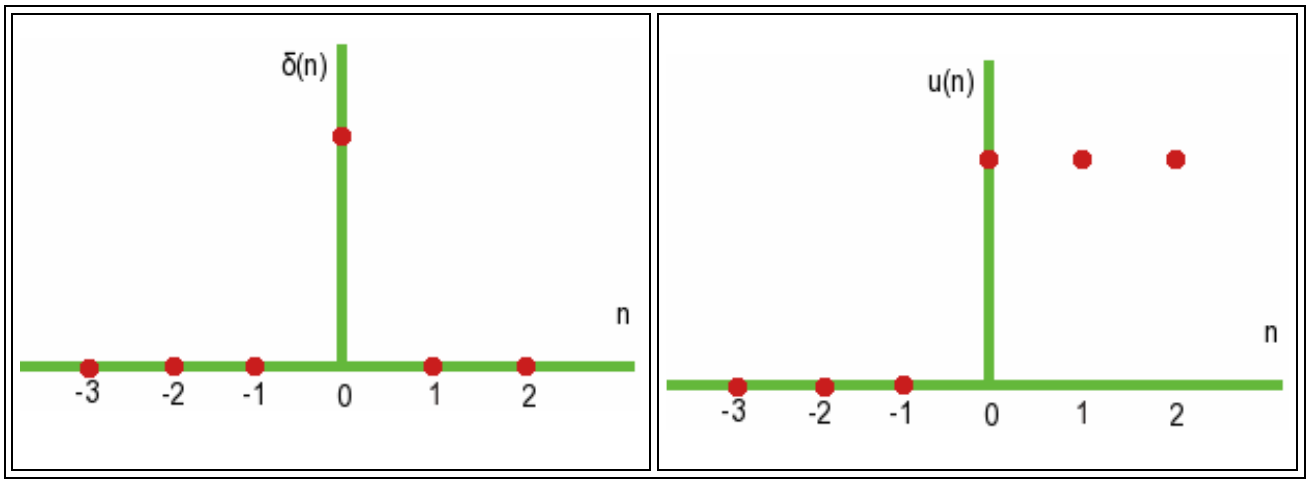
Σημαντικό ρόλο στην ανάλυση σημάτων διακριτού χρόνου, παίζουν οι συναρτήσεις της μοναδιαίας διακριτής ώσης (ή διακριτής συνάρτησης δ) και της μοναδιαίας συνάρτησης βήματος.

Συνάρτηση της μοναδιαίας διακριτής ώσης:

$$\delta(n)=\begin{cases}1, & n=0\\ 0, & \text{οπουδήποτεαλλού}\end{cases} \quad = u(n) - u(n-1)$$

Μοναδιαία συνάρτηση βήματος:

$$u(n)=\begin{cases}1, & n\geq 0\\ 0, & \text{οπουδήποτεαλλού}\end{cases} \quad = \sum_{k=-\infty}^{\infty}\delta(k)$$



Εικόνα 3: Οι συναρτήσεις δ και u.

Αποδεικνύεται ότι κάθε διακριτό σήμα x(n) μπορεί να γραφεί ως ένα άθροισμα απείρων μετατοπισμένων μοναδιαίων ώσεων οι οποίες είναι πολλαπλασιασμένες με έναν συντελεστή βάρους που αντιστοιχεί στην εκάστοτε τιμή του σήματος. Η ιδιότητα αυτή αποδίδεται με την παρακάτω εξίσωση:

$$x(n)=\ldots+x(-1)*\delta(n+1)+x(0)*\delta(n)+x(1)*\delta(n-1)+\ldots \quad = \sum_{k=-\infty}^{\infty}x(k)*\delta(n-k)$$

2.3 Θεμελιώδεις πράξεις με διακριτά σήματα [περιεχόμενα]

Πάνω σε διακριτά σήματα μπορούμε να εφαρμόσουμε αρκετούς μετασχηματισμούς και πράξεις με βασικότερες την χρονική μετατόπιση, την κλιμάκωση του πλάτους του σήματος, την πρόσθεση και τον πολλαπλασιασμό σημάτων. Παρακάτω παρουσιάζονται οι πράξεις αυτές αναλυτικά:

- **Μεταβολή της ανεξάρτητης μεταβλητής – μετατόπιση στο χρόνο**

Ένα διακριτό σήμα x(n) μπορεί να μετατοπιστεί στον χρόνο, αν αντικαταστήσουμε την ανεξάρτητη μεταβλητή n με την τιμή (n-k), όπου k είναι μια ακέραια τιμή. Αν η τιμή k είναι θετική τότε το σήμα μετατοπίζεται κατά k μονάδες προς τα πίσω ενώ σε αντίθετη περίπτωση το σήμα μετατοπίζεται κατά |k| μονάδες προς τα μπροστά.

- **Κλιμάκωση πλάτους**

Η κλιμάκωση του πλάτους κατά μία σταθερά Α γίνεται πολλαπλασιάζοντας την κάθε τιμή του σήματος με την σταθερά αυτή. Το νέο σήμα που θα προκύψει θα δίδεται από την σχέση

$$x(n) = Ax(n), \quad -\infty < n < \infty$$

- **Πρόσθεση σημάτων**

Το άθροισμα δύο διακριτών σημάτων x1(n) και x2(n) ορίζεται ως ένα νέο σήμα y(n) το οποίο έχει κάθε φορά τιμή, το άθροισμα των τιμών των x1 και x2. Στις περιοχές που το ένα από τα δύο x δεν ορίζεται τότε το y θα έχει την τιμή του άλλου. Επομένως το σήμα y(n) θα ικανοποιεί την σχέση

$$y(n) = x1(n) + x2(n), \quad -\infty < n < \infty$$

- **Πολλαπλασιασμός σημάτων**

Το γινόμενο δύο διακριτών σημάτων x1(n) και x2(n) ορίζεται ως ένα νέο σήμα y(n) το οποίο, για κάθε n έχει τιμή το γινόμενο των τιμών των σημάτων x1 και x2. Στις περιοχές που κάποιο από τα x1,x2 δεν ορίζεται, το y μηδενίζεται. Επομένως το νέο σήμα θα δίδεται από τη σχέση

$$y(n) = x1(n) * x2(n), \quad -\infty < n < \infty$$

2.4 Στατιστικές ιδιότητες διακριτών σημάτων [περιεχόμενα]

Εάν υποθέσουμε ότι έχουμε ένα μονοδιάστατο διακριτό σήμα με τιμές x_1, x_2, ... x_n τότε μπορούμε να ορίσουμε τις παρακάτω ιδιότητες της ακολουθίας αυτής.

Όταν ένα σήμα έχει τιμές που συσσωρεύονται γύρω από μία περιοχή, είναι χρήσιμος ο χαρακτηρισμός του με βάση έναν μικρό αριθμό ροπών, δηλαδή αθροισμάτων ακέραιων δυνάμεων των τιμών του. Περισσότερο χρησιμοποιούμενη είναι η μέση τιμή

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

η οποία προβλέπει την τιμή γύρω από την οποία υπάρχει συσσώρευση των δεδομένων. Σημειώστε ότι η μέση τιμή δεν είναι ο μόνος εκτιμητής του σημείου συσσώρευσης, ούτε κατανάγκην ο

καλύτερος. Εναλλακτικός εκτιμητής είναι η μεσαία τιμή, δηλαδή η τιμή για την οποία μικρότερες και μεγαλύτερες τιμές της μεταβλητής είναι εξίσου πιθανές.

Έχοντας χαρακτηρίσει τη μέση τιμή, επιθυμούμε συνήθως στη συνέχεια να εκφράσουμε την διασπορά γύρω από αυτή την τιμή. Και εδώ οι επιλογές είναι περισσότερες της μίας. Συνηθέστερα απαντώνται η μεταβλητότητα (variance)

$$Var(x_1, x_2, \ldots, x_n) = \frac{1}{N-1}\sum_{i=1}^{N}(x_i+\bar{x})^2$$

και η τυπική απόκλιση (standard deviation)

$$\sigma(x_1, x_2, \ldots, x_n) = \sqrt{Var} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i+\bar{x})^2}$$

Εάν το σήμα έχει πολύ έντονες διακυμάνσεις και ασυνεχή σημεία μακρυά από τη μέση τιμή, είναι ενδεχόμενο οι παραπάνω εκτιμήσεις της διασποράς ακόμη και να μην συγκλίνουν. Μία περισσότερο στιβαρή εκτίμηση προσφέρεται από την μέση απόλυτη απόκλιση, που ορίζεται ως

$$ADev(x_1, x_2, \ldots, x_n) = \frac{1}{N}\sum_{i=1}^{N}|x_i+\bar{x}|$$

Οι ανώτερες ροπές χαρακτηρίζουν την μορφή κατανομής των τιμών του σήματος. Ειδικότερα, η τρίτη ροπή (skewness) εκφράζει τον βαθμό ασυμετρίας της κατανομής.

$$Scew(x_1, x_2, \ldots, x_n) = \frac{1}{N}\sum_{i=1}^{N}\left[\frac{x_i-\bar{x}}{\sigma}\right]^3$$

Αν η ροπή είναι θετική, η κατανομή εκτείνεται ασύμμετρα προς τις θετικότερες τιμές της μεταβλητής. Αντίστοιχα, η τέταρτη ροπή (kurtosis) δείχνει πόσο οξεία (θετική ροπή) ή πεπλατυσμένη (αρνητική ροπή) είναι η κατανομή και ορίζεται ως

$$Kurt(x_1, x_2, \ldots, x_n) = \left\{\frac{1}{N}\sum_{i=1}^{N}\left[\frac{x_i-\bar{x}}{\sigma}\right]^4\right\}-3$$

Το μέτρο σύγκρισης είναι η κανονική κατανομή με την ίδια μέση τιμή και τυπική απόκλιση, η οποία έχει την τέταρτη ροπή ίση με 3.

2.5 Μετατροπή αναλογικού σήματος σε ψηφιακό [περιεχόμενα]

Η μετατροπή ενός αναλογικού σήματος σε ψηφιακό είναι απαραίτητη διαδικασία εάν θέλουμε να επεξεργαστούμε το σήμα αυτό. Η μετατροπή αυτή πραγματοποιείται από τις διατάξεις ADCs (Analog to Digital Converters) και αποτελείται από τα τρία παρακάτω βήματα:

- **Δειγματοληψία (Sampling)**

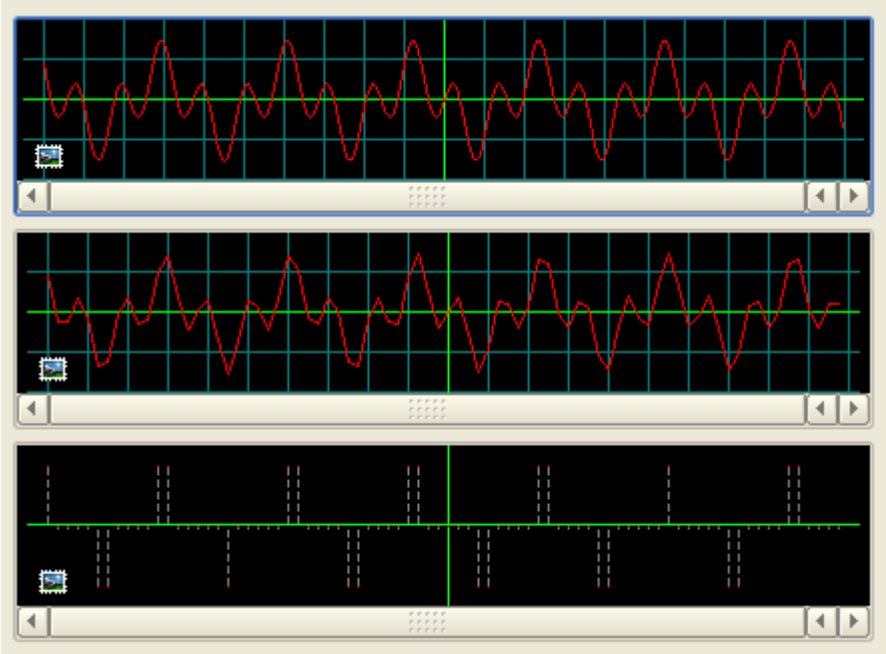
Στο βήμα αυτό γίνεται η μετατροπή του αναλογικού σήματος σε σήμα διακριτού χρόνου. Για να το κάνουμε αυτό μετράμε τις τιμές του πλάτους του σήματος σε διακριτές χρονικές στιγμές. Συνήθως οι χρονικές αυτές στιγμές ισαπέχουν μεταξύ τους. Η μεταξύ τους απόσταση λέγεται περίοδος δειγματοληψίας. Όσο μεγαλύτερη είναι η περίοδος δειγματοληψίας τόσο μικρότερο θα είναι το σήμα που θα παραχθεί (πιο διάσπαρτα δείγματα) και τόσο μικρότερη είναι η ακρίβεια του σήματος αυτού σε σχέση με το αρχικό.

- **Κβαντισμός (Quantization)**

Το σήμα που έχουμε σαν αποτέλεσμα της προηγούμενης διαδικασίας είναι ένα διακριτό σήμα συνεχών τιμών. Μπορεί δηλαδή να έχουμε μια ακολουθία τιμών οι οποίες ισαπέχουν μεταξύ τους αλλά είναι συνεχείς και μπορούν να πάρουν οποιαδήποτε τιμή. Οπότε στο στάδιο του κβαντισμού οι τιμές του σήματος προσαρμόζονται έτσι ώστε οι νέες τιμές που θα προκύψουν να ανήκουν σε ένα σύνολο προκαθορισμένων τιμών (στάθμες). Η διαφορά του σήματος που είχαμε μετά τη δειγματοληψία και του σήματος μετά τον κβαντισμό ονομάζεται σφάλμα κβαντισμού και εξαρτάται από το πλήθος των σταθμών που έχουμε επιλέξει. Όσο μικρότερο είναι το πλήθος των σταθμών τόσο μεγαλύτερες είναι οι στρογγυλοποιήσεις-προσαρμογές που πρέπει να γίνουν στις τιμές των δειγμάτων.

- **Κωδικοποίηση (Coding)**

Στο στάδιο αυτό, οι κβαντισμένες τιμές που έχουν προκύψει από την προηγούμενη διαδικασία μετατρέπονται σε μία ακολουθία δυαδικών αριθμών με μέγεθος n bits.



Εικόνα 4: Δειγματοληψία - Κβαντισμός

2.6 Συνέλιξη [περιεχόμενα]

Ονομάζουμε συνέλιξη των δύο συναρτήσεων (convolution sum) την συνάρτηση που ορίζεται από τον τύπο

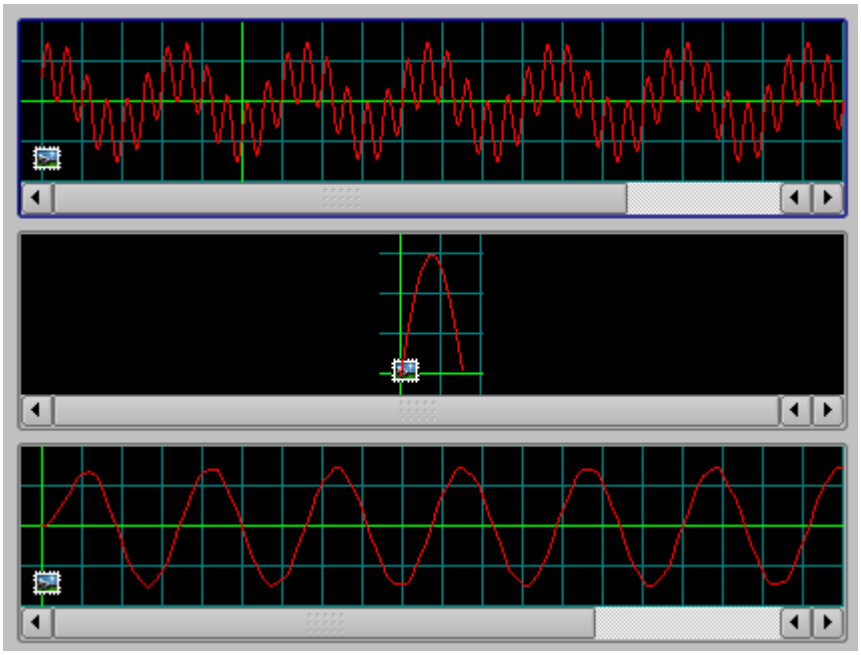
$$g * h = \int_{-\infty}^{\infty} g(\tau) h(t-\tau) d\tau$$

Η συνέλιξη δύο συναρτήσεων r(t), s(t), που συμβολίζεται ως r*s, είναι μαθηματικά ισοδύναμη με την συνέλιξη σε αντίστροφη σειρά, s*r. Εντούτοις, στις περισσότερες εφαρμογές οι δύο συναρτήσεις έχουν διαφορετική μορφή και σημασία. Η μία συνάρτηση, έστω η s, είναι ένα σήμα που συνεχίζεται απεριόριστα στον χρόνο. Η άλλη ονομάζεται κρουστική απόκριση, και έχει συνήθως σχήμα κορυφής/καμπάνας, που εκφυλίζεται στο μηδέν καθώς απομακρυνόμαστε από την κορυφή. Το αποτέλεσμα της συνέλιξης είναι να εξομαλύνεται το σήμα s(t) σύμφωνα με την μορφή της r(t) (εικόνα 5). Για παράδειγμα, αν η συνάρτηση απόκρισης έχει ένα ορισμένο εύρος, E, τότε η συνέλιξη εξομαλύνει τις διακυμάνσεις υψηλής συχνότητας του σήματος, και ειδικότερα εκείνες που έχουν χαρακτηριστικό χρόνο T<E.

Στην διακριτή περίπτωση, το σήμα s(t) αποτελείται από το σύνολο των τιμών sj, και η συνάρτηση απόκρισης είναι επίσης ένα σύνολο διακριτών τιμών, rk. Τότε η συνέλιξη έχει την εξής ερμηνεία: Η τιμή r0 δείχνει ποιο πολλαπλάσιο του σήματος στη θέση j μεταφέρεται στην ίδια θέση j, η τιμή r1 δείχνει ποιο πολλαπλάσιο του σήματος στη θέση j-1 προστίθεται στη θέση j, κλπ. Έτσι, η συνάρτηση απόκρισης r0=1, rk=0 είναι το φίλτρο ταυτότητας, δηλαδή μεταφέρει με τη συνέλιξη κάθε σήμα στον εαυτό του. Η συνάρτηση r14=1,5, rk=0 παράγει με τη συνέλιξη ένα σήμα που είναι το σήμα εισόδου πολλαπλασιασμένο επί 1,5 και καθυστερημένο κατά 14 χρονικά βήματα. Τα παραπάνω παραδείγματα αντιστοιχούν στον εξής ορισμό τη διακριτής συνέλιξης,

$$(r * s)_j = \sum_{k=-M/2+1}^{M/2} s_{j-k} r_k$$

όπου η συνάρτηση r έχει μη-μηδενικές τιμές μόνον στο διάστημα –M/2 < k < M/2, ή αλλιώς η διάρκεια της r είναι ίση με M.



*Εικόνα 5: Συνέλιξη των sin(x*1000)+cos(x) και sin(x) στο διάστημα [0, 3.2)*

2.7 Μετασχηματισμός Fourier [περιεχόμενα]

Σύμφωνα με τη θεωρία περιοδικών συναρτήσεων, κάθε σήμα μπορεί να γραφεί ως άθροισμα ημιτονοειδών συναρτήσεων κάθε μια από τις οποίες χαρακτηρίζεται από το πλάτος (συντελεστής βαρύτητας) και από τη συχνότητά της. Συνεπώς ένα σήμα εκφράζεται ισοδύναμα με δύο τρόπους: ο ένας είναι η ίδια η χρονοσειρά, δηλαδή οι τιμές του σήματος την κάθε χρονική στιγμή, και ο άλλος είναι η τιμές του πλάτους για κάθε συχνότητα. Αν ξέρουμε τον τύπο h(t) που περιγράφει μαθηματικά ένα σήμα, τότε μπορούμε να βρούμε το πλάτος της συχνότητας f που αντιστοιχεί σε κάθε επιμέρους ημιτονοειδή συνάρτηση, με την παρακάτω σχέση.

$$H(f) = \int_{-\infty}^{\infty} h(t) e^{2\pi i f t} dt$$

Αυτός είναι ο ορισμός του μετασχηματισμού Fourier. Αντίστροφα, αν γνωρίζουμε το πλάτος κάθε συχνότητας του σήματος μπορούμε να ανακατασκευάσουμε το σήμα σύμφωνα με τον αντίστροφο μετασχηματισμό Fourier που περιγράφεται από τη σχέση

$$h(t) = \int_{-\infty}^{\infty} H(f) e^{-2\pi i f t} df$$

Στις περισσότερες περιπτώσεις, η συνάρτηση *h(t)* προέρχεται από δειγματοληψία σε χρονικές τιμές που ισαπέχουν. Αν συμβολίσουμε με *Δ* το χρονικό διάστημα μεταξύ διαδοχικών δειγματοληψιών, τότε οι γνωστές τιμές του σήματος είναι

$$h_n = h(n\Delta) \quad n=...,-3,-2,-1,0,1,2,3,...$$

Το αντίστροφο του *Δ* είναι ο ρυθμός δειγματοληψίας.

Υποθέτουμε ότι διαθέτουμε *N* διαδοχικές τιμές μίας συνάρτησης *h(t)* στις χρονικές στιγμές $t_k = k\Delta$, όπου *k=0,1,2,...,N-1*. Για απλότητα, θα θεωρήσουμε ότι ο αριθμός *N* είναι άρτιος. Με *N* μόνον τιμές σήματος δεν μπορούμε προφανώς να βρούμε περισσότερες από *N* ανεξάρτητες τιμές του μετασχηματισμού. Έτσι, αντί να βρούμε τον μετασχηματισμό Fourier για οποιαδήποτε τιμή στο διάστημα $-f_c < f < f_c$, αναζητούμε την μετασχηματισμένη συνάρτηση στις διακριτές συχνότητες

$$f_n = \frac{n}{N\Delta}, \quad n = -\frac{N}{2},...,\frac{N}{2}$$

Οι ακραίες τιμές της παραμέτρου *n* δίνουν το άνω και κάτω όριο Nyquist.

Ο διακριτός μετασχηματισμός βρίσκεται προσεγγίζοντας το αντίστοιχο ολοκλήρωμα ως εξής:

$$H(f_n) = \int_{-\infty}^{\infty} h(t) e^{2\pi i f_n t} dt \approx \sum_{k=0}^{N-1} h_k e^{2\pi i f_n t_k} \Delta = \Delta \sum_{k=0}^{N-1} h_k e^{2\pi i k n / N}$$

Έτσι, ορίζουμε τον διακριτό μετασχηματισμό Fourier ώστε να είναι ανεξάρτητος οποιασδήποτε διαστατικής παραμέτρου, όπως το χρονικό βήμα Δ, σύμφωνα με τη σχέση.

$$H_n = \sum_{k=0}^{N-1} h_k e^{2\pi i k n / N}$$

Συνεπώς, η σχέση μεταξύ του διακριτού μετασχηματισμού Fourier ενός πλήθους τιμών και του συνεχούς μετασχηματισμού όταν οι τιμές θεωρηθούν ως δείγμα μίας συνεχούς συνάρτησης, δίνεται από τον τύπο

$$H\left(f_n\right) \approx \Delta H_n$$

Ο διακριτός αντίστροφος μετασχηματισμός Fourier υπολογίζει τις τιμές h_k αν γνωρίζουμε τις H_n, και δίνεται από τη σχέση

$$h_k = \frac{1}{N} \sum_{n=0}^{N-1} H_n e^{-2\pi i k n / N}$$

Γρήγορος μετασχηματισμός Fourier (FFT)

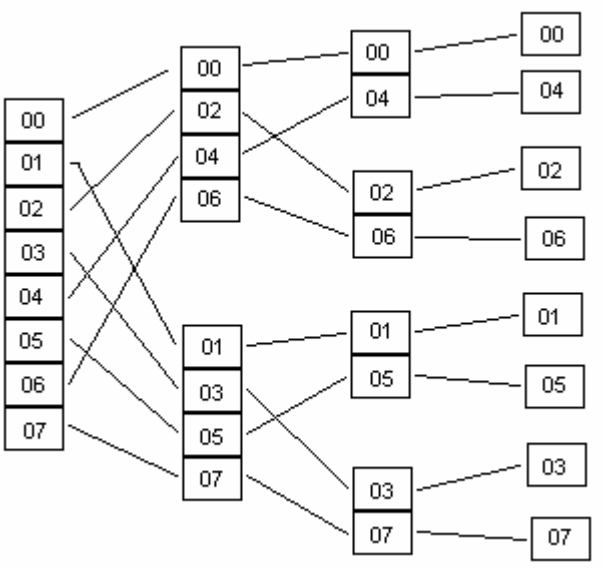
Ένα σημαντικό ερώτημα αφορά τον αριθμό των πράξεων για τον υπολογισμό του διακριτού μετασχηματισμού ενός σήματος. Αντικαθιστώντας τον μιγαδικό αριθμό $W = e^{2\pi i / N}$ λαμβάνουμε το αποτέλεσμα

$$H_n = \sum_{k=0}^{N-1} W^{nk} h_k$$

Το προφανές συμπέρασμα είναι ότι χρειάζονται N πολλαπλασιασμοί μιγαδικών αριθμών για κάθε τιμή H_n, δηλαδή N^2 συνολικά πράξεις. Στην πραγματικότητα, υπάρχει κατάλληλος αλγόριθμος ο οποίος απαιτεί πολύ λιγότερες πράξεις ($N \log_2 N$). Ο αλγόριθμος αυτός έγινε γνωστός στα μέσα της δεκαετίας του 1960 από την εργασία των Cooley και Tukey, και αποτελεί μία από τις σημαντικότερες προόδους στον τομέα της αριθμητικής ανάλυσης.

Η μαθηματική θεμελίωση του γρήγορου μετασχηματισμού Fourier βασίζεται σε ένα λήμμα που διατυπώθηκε από τους Danielson και Lanczos το 1942, σύμφωνα με το οποίο ο διακριτός μετασχηματισμός Fourier μιας ακολουθίας τιμών με μέγεθος N μπορεί να γραφεί ως το άθροισμα δύο διακριτών μετασχηματισμών, καθένας από τους οποίους έχει μήκος N/2. Από τους δύο αυτούς

μετασχηματισμούς, ο ένας υπολογίζεται από τα δείγματα που βρίσκονται στις άρτιες θέσεις της ακολουθίας εισόδου ενώ ο άλλος από τις περιττές. Η πιο σημαντική από τις ιδιότητες του λήμματος αυτού είναι η αναδρομικότητα: Για παράδειγμα, έχοντας διασπάσει τον μετασχηματισμό Χ(κ) στους Γ(κ) και Η(κ) (καθένας από τους οποίους έχει μέγεθος Ν/2), μπορούμε να εφαρμόσουμε την ίδια διαδικασία και στους δύο αυτούς μετασχηματισμούς και να έχουμε σαν αποτέλεσμα τέσσερις μετασχηματισμούς με μέγεθος Ν/4 ο καθένας. Η συνθήκη τερματισμού της αναδρομικής αυτής διαδικασίας είναι όταν έχουμε φτάσει στο σημείο να θέλουμε μετασχηματισμό Fourier μιας ακολουθίας που περιλαμβάνει ένα μόνο στοιχείο.



Εικόνα 6: Διάσπαση ακολουθίας για μετασχηματισμό Fourier

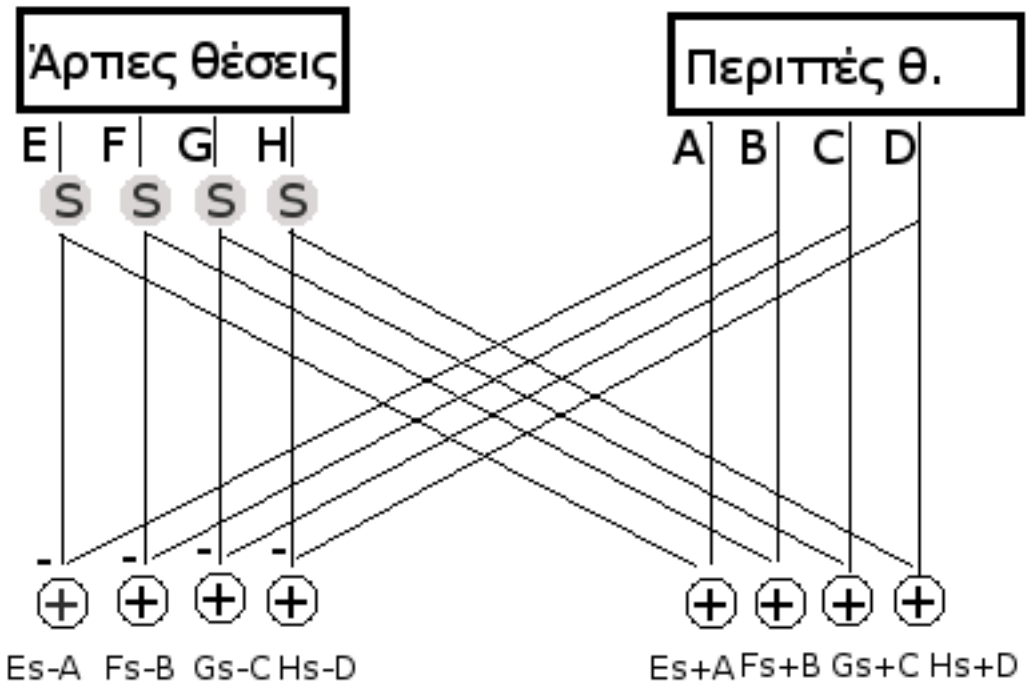
Επειδή σε κάθε εφαρμογή του αλγορίθμου το πλήθος των σημείων υποδιπλασιάζεται, είναι προφανές ότι το σύνολο των σημείων πρέπει να είναι δύναμη του 2, έτσι ώστε η ποσότητα $\log_2 N$ να είναι ακέραιος αριθμός. Σε αντίθετη περίπτωση ο γρήγορος μετασχηματισμός Fourier δεν μπορεί να εφαρμοστεί. Σε περιπτώσεις κατά τις οποίες το μέγεθος της ακολουθίας δεν είναι δύναμη του 2 μπορούμε είτε να προσθέσουμε μηδενικά στο τέλος της ακολουθίας, είτε να αγνοήσουμε τα τελευταία στοιχεία, ώστε το μέγεθος να γίνει δύναμη του 2. Αν και υπάρχουν πιο έξυπνες τεχνικές από τις παραπάνω, εμείς θα περιοριστούμε σε αυτές.

Παρατηρώντας προσεκτικά την υλοποίηση της παραπάνω διαδικασίας (Εικόνα 6), δεν είναι δύσκολο να διαπιστώσει κανείς ότι προκειμένου να είναι δυνατή η εφαρμογή του γρήγορου μετασχηματισμού Fourier επί της ακολουθίας των 8 σημείων, θα πρέπει αυτά να αναδιαταχθούν έτσι ώστε να εμφανίζονται με τη σειρά {00, 04, 02, 06, 01, 05, 03, 07}. Αποδεικνύεται πως αυτή η αναδιάταξη μπορεί να πραγματοποιηθεί εύκολα με τον ακόλουθο τρόπο: για κάθε δείγμα x(n), μετατρέπουμε το δείκτη του (n) στο δυαδικό σύστημα και στη συνέχεια αντιστρέφουμε τα bits του

αριθμού έτσι ώστε να σχηματίσουμε το κατοπτρικό του (bit reverse). Για το παράδειγμα της εικόνας 6 θα έχουμε:

Τιμή Δείκτη	Δυαδική Τιμή	Κατοπτρική Τιμή	Δεκαδική Τιμή
00	000	000	00
01	001	100	04
02	010	010	02
03	011	110	06
04	100	001	01
05	101	101	05
06	110	011	03
07	111	111	07

Ο υπολογισμός του μετασχηματισμού Fourier για κάθε ένα στοιχείο, όπως φαίνεται από τον τύπο, είναι το ίδιο το στοιχείο. Τώρα μένει να συνθέσουμε τα επιμέρους στοιχεία ώστε να πάρουμε το μετασχηματισμό όλης της ακολουθίας. Για να παρουσιάσουμε τη σύνθεση αυτή θα πάρουμε για παράδειγμα δύο ακολουθίες δεδομένων εισόδου(μονών και ζυγών στοιχείων) και τους μετασχηματισμούς τους. Έστω λοιπόν οι ακολουθίες {a,b,c,d} και {e,f,g,h} τον οποίων οι μετασχηματισμοί Fourier περιγράφονται από τις ακολουθίες {A,B,C,D} και {E,F,G,H} αντίστοιχα. Η σύνθεση γίνεται όπως φαίνεται στην παρακάτω εικόνα, όπου S είναι ο πολλαπλασιασμός του συντελεστή Fourier με τη συνάρτηση του συνημίτονου.



3 Εγχειρίδιο χρήσης [περιεχόμενα]

Σε αυτή την ενότητα αναφέρονται οι απαιτήσεις της εφαρμογής, βιβλιοθήκες και compilers που χρειάζονται για την εγκατάσταση και τη λειτουργία της. Επίσης παρουσιάζεται η ίδια η εφαρμογή και οι λειτουργίες της.

3.1 Προαπαιτούμενα [περιεχόμενα]

Η εφαρμογή μπορεί να εγκατασταθεί σε Windows, Linux/Unix και Mac και διάφορες άλλες πλατφόρμες. Για την εγκατάστασή της απαιτείται η ύπαρξη κάποιου compiler που να είναι συμβατός με την βιβλιοθήκη του Qt καθώς και η ίδια η βιβλιοθήκη του Qt. Στα περισσότερα Unix based συστήματα υπάρχει ο gcc, στα Windows χρησιμοποιούμε τον MiniGW ο οποίος είναι μια μικρή έκδοση του gcc. Για την ανάπτυξη της εφαρμογής χρησιμοποιήθηκε το qt-win-opensource-4.2.3-mingw. Επίσης δοκιμάστηκε η έκδοση 4.3 η οποία όμως δεν φάνηκε να δουλεύει σωστά, για τον λόγο αυτό προτείνεται να χρησιμοποιήσετε την έκδοση 4.2.3. Στην εικόνα παρουσιάζονται οι υποστηριζόμενες πλατφόρμες – compilers.

Supported Platforms

Platforms 32bit

	GCC	Provided by OS vendor	3rd party
Microsoft Windows	gcc (MinGW)	MS Visual Studio/Visual C++	Intel icc
Linux	gcc	-	-
Apple Mac OS X	gcc	-	-
IBM AIX	gcc	IBM xlC	-
HP HP-UX	gcc	HP aCC	-
SGI IRIX	gcc	SGI MipsPRO	-
Sun Solaris	gcc	Sun CC	-
FreeBSD	gcc	-	Intel icc

Platforms 64bit

	GCC	Provided by OS vendor	3rd party
Microsoft Windows	-	MS Visual Studio/Visual C++	Intel icc
IBM AIX	gcc	IBM xlC	-
HP-UX	-	HP acc	-
SGI IRIX	gcc	SGI MipsPRO	-
Sun Solaris	gcc	Sun cc	-
Linux	gcc	-	Intel icc

Εικόνα 7: Υποστηριζόμενες πλατφόρμες – compilers

Η παραπάνω εικόνα είναι από την σελίδα του Qt στο internet και παρουσιάζονται τα υποστηριζόμενα συστήματα μέχρι αυτή τη στιγμή. Για παραπάνω ή ανανεωμένες πληροφορίες ανατρέξτε στην διεύθυνση: http://trolltech.com/developer/notes/supported_platforms .

Την βιβλιοθήκη του Qt και τους compilers μπορείτε να τους βρείτε στις παρακάτω διευθύνσεις:

Qt: ftp://ftp.trolltech.com/qt/source/ (binaries και sources από όλες τις εκδόσεις)

MiniGW: http://www.mingw.org/download.shtml

GCC: http://gcc.gnu.org/install/binaries.html

3.2 Εγκατάσταση [περιεχόμενα]

Για να γίνει η εγκατάσταση της εφαρμογής πρέπει να γίνει πρώτα compile. Πρέπει να μπούμε στον κυρίως φάκελο της εφαρμογής, στον οποίο βρίσκεται το αρχείο ergasia.pro, σε command prompt.

Start -> Run... -> "**cmd**"

cd <διαδρομή φακέλου>

qmake

Η qmake θα διαβάσει το αρχείο ergasia.pro και θα παράγει τα Makefiles.

make

Εδώ γίνεται το η μεταγλώττιση της εφαρμογής. Σε αυτό το στάδιο δεν πρέπει να εμφανιστεί κανένα σφάλμα. Εάν εμφανιστούν σφάλματα ελέγξτε ότι η βιβλιοθήκη του Qt έχει εγκατασταθεί σωστά και ότι η μεταβλητή περιβάλλοντος QT_DIR έχει την σωστή τιμή.

Μετά από την παραπάνω διαδικασία το εκτελέσιμο αρχείο θα βρίσκεται στη διαδρομή <φάκελος του project>/release και θα έχει όνομα ergasia (.exe σε windows).

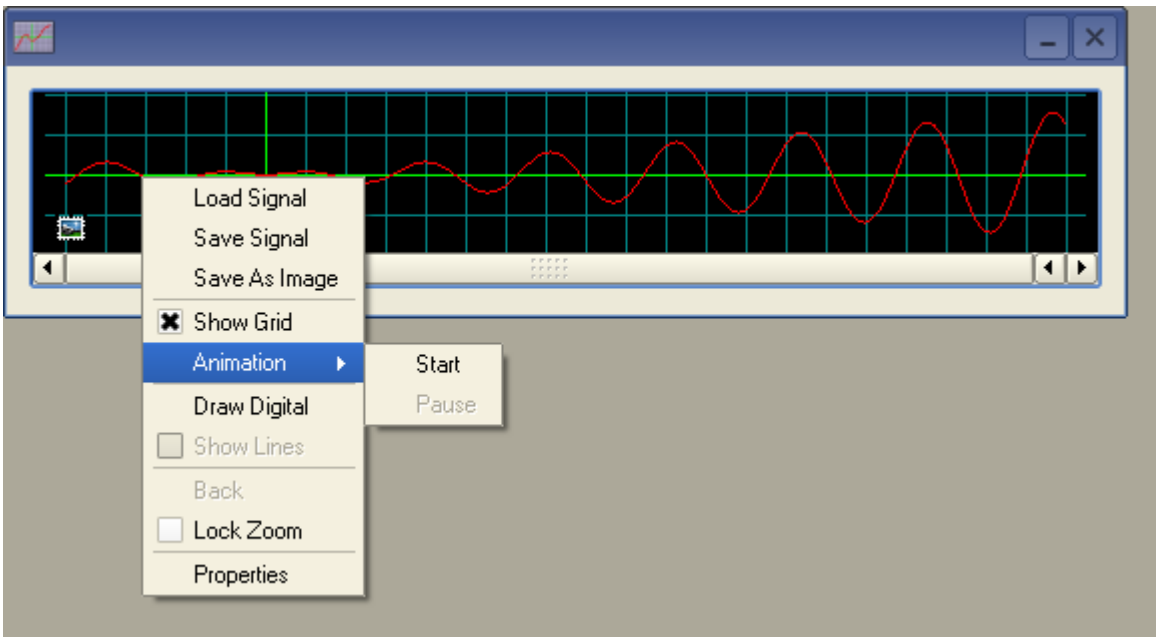
3.3 Λειτουργίες [περιεχόμενα]

Η εφαρμογή κάνει γραφικές παραστάσεις σημάτων τα οποία μπορούν μετά να επεξεργαστούν, να γίνουν πράξεις μεταξύ των σημάτων ή μετασχηματισμοί πάνω σε κάποιο σήμα καθώς και να παρουσιαστούν σαν κινούμενα. Τα σήματα μπορούν να είναι είτε αποθηκευμένα σε αρχεία με τη μορφή “x y” σε κάθε γραμμή τους και να φορτώνονται σε κάποιο διάγραμμα ή μπορούν να δημιουργηθούν από έναν editor που περιλαμβάνει η εφαρμογή. Στα σήματα που δημιουργήθηκαν ή προέκυψαν από μετασχηματισμούς/πράξεις υπάρχει η δυνατότητα αποθήκευσής τους σε αρχείο. Τέλος, τα διαγράμματα που έχουν δημιουργηθεί μπορούν να αποθηκευτούν σε εικόνα της μορφής PNG.

Με την εκκίνηση της εφαρμογής εμφανίζεται ένα κενό παράθυρο με το μενού του. Η περιοχή που είναι κενή είναι ο χώρος εργασίας, όπου μέσα μπορούμε να ανοίξουμε πολλά άλλα παράθυρα. Τα παράθυρα που μπορούν υπάρχουν είναι τεσσάρων ειδών και παρουσιάζονται παρακάτω:

Το απλό διάγραμμα:

Το απλό διάγραμμα είναι ένα παράθυρο με ένα μόνο διάγραμμα. Το διάγραμμα έχει μενού για τη διαχείρισή του, το οποίο εμφανίζεται με δεξί κλικ οπουδήποτε μέσα σε αυτό καθώς και μία εικόνα στην κάτω αριστερή γωνία του η οποία χρησιμοποιείται για drag 'n' drop του σήματος. Σέρνοντας την εικόνα αυτή από ένα διάγραμμα σε ένα άλλο, αντιγράφεται το σήμα του πρώτου διαγράμματος στο δεύτερο. Αν το δεύτερο είχε ήδη φορτωμένο σήμα τότε το χάνει και δεν μπορούμε να γυρίσουμε στην προηγούμενη κατάσταση. Το drag 'n' drop μπορεί να γίνει και μεταξύ διαγραμμάτων που ανήκουν σε διαφορετικά παράθυρα.



Εικόνα 8: Απλό διάγραμμα

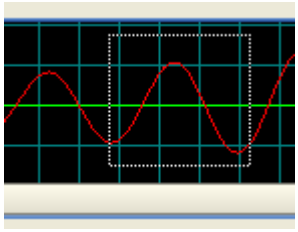
Στο μενού εμφανίζονται αρκετές επιλογές. Οι τρις πρώτες είναι για τη διαχείριση του σήματος, δηλαδή το φόρτωμα νέου σήματος στο διάγραμμα (Load Signal), την αποθήκευσή του σε αρχείο (.dat ή .sig) (Save Signal) καθώς και την αποθήκευση του διαγράμματος σαν εικόνα (Save As Image).

Οι δύο επόμενες επιλογές είναι για το αν θέλουμε να είναι φανερό το πλέγμα του διαγράμματος (Show Grid) και αν είναι φανερό τότε το κουτάκι από δίπλα είναι επιλεγμένο και για τον έλεγχο της κίνησης του σήματος στο διάγραμμα. Το υπο-μενού Animation έχει την επιλογή Start από την οποία ξεκινά η κίνηση του σήματος. Όταν πατηθεί το Start τότε την θέση του παίρνει το Stop και ενεργοποιείται η επιλογή Pause η οποία "παγώνει" την κίνηση του σήματος. Όταν πατηθεί Pause τότε την θέση του παίρνει το Resume το οποίο συνεχίζει την κίνηση του σήματος από τη στιγμή που πατήθηκε το Pause.

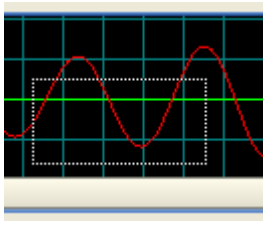
Οι αμέσως επόμενες δύο επιλογές ελέγχουν τον τρόπο με τον οποίο ζωγραφίζεται το σήμα στο διάγραμμα. Αν αυτό ζωγραφίζεται σαν να είναι αναλογικό, τότε από σημείο σε σημείο του σήματος ζωγραφίζεται μια γραμμή. Σε αντίθετη περίπτωση ζωγραφίζονται μόνο τα σημεία του σήματος. Η αλλαγή από αναλογικό σε ψηφιακό γίνεται από την επιλογή Draw Digital. Όταν αυτή πατηθεί τότε τη θέση της παίρνει η επιλογή Draw Analog. Εάν το σήμα ζωγραφίζεται σαν ψηφιακό τότε ενεργοποιείται η επιλογή Show Lines η οποία αν επιλεχθεί εμφανίζονται, κάθετες προς τον άξονα x'x, γραμμές από κάθε σημείο του σήματος.

Το διάγραμμα αυτού του τύπου υποστηρίζει μεγέθυνση σε κάποια περιοχή τιμών ώστε τα

σημεία του σήματος να εμφανίζονται με μεγαλύτερη λεπτομέρεια και σε άλλη κλίμακα. Η μεγέθυνση γίνεται με την επιλογή μιας περιοχής μέσα στο διάγραμμα με κρατημένο το αριστερό πλήκτρο του ποντικιού. Η περιοχή αυτή δεν μπορεί να είναι οποιαδήποτε. Θα πρέπει να υπάρχουν δύο ή παραπάνω σημεία του σήματος στην περιοχή αυτή και το πλάτος του σήματος δεν μπορεί να βγαίνει από την περιοχή και να ξανα-μπαίνει σε αυτή (να υπάρχει ασυνέχεια δηλαδή).



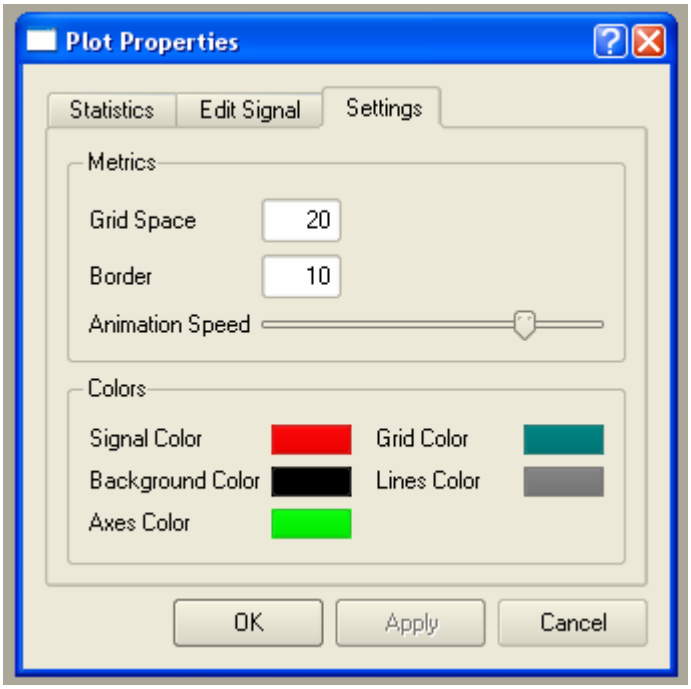
Εικόνα 9: Σωστό



Εικόνα 10: Λάθος

Η επιλογή Zoom Lock όταν είναι επιλεγμένη, απενεργοποιεί την παραπάνω λειτουργία ώστε να μην γίνεται μεγέθυνση κατά λάθος. Όταν το σήμα έχει μεγεθυνθεί τότε ενεργοποιείται η επιλογή Back η οποία μας φέρνει στην αμέσως προηγούμενη κατάσταση. Αν επιλεχθεί πολλές φορές Back, κάποια στιγμή θα φτάσουμε στο αρχικό μας σήμα. Τα μεγεθυμένα σήματα μπορούν να αποθηκευτούν σαν να είναι κανονικά και στο αρχείο θα αποθηκευτούν μόνο τα σημεία του αρχικού σήματος που ανήκουν στην επιλεγμένη περιοχή.

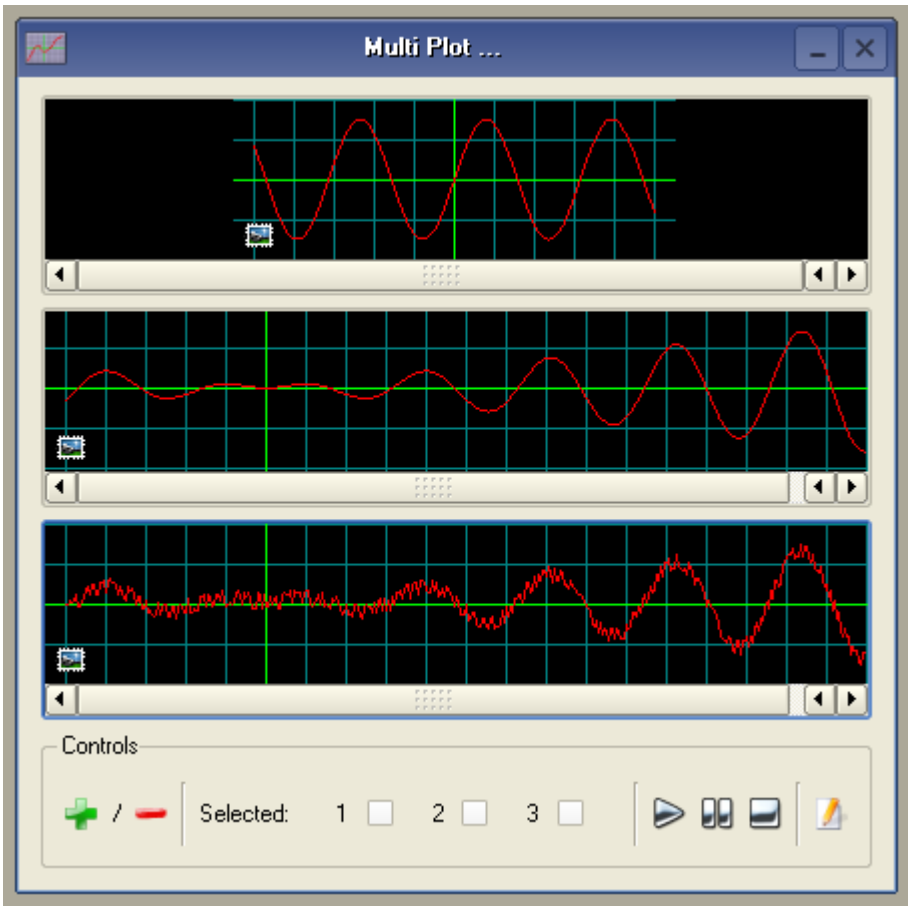
Τέλος, η υπάρχει η επιλογή Properties η οποία εμφανίζει έναν διάλογο που μας παρέχει διάφορες δυνατότητες. Ο διάλογος αυτός έχει τρις καρτέλες, στην πρώτη εμφανίζονται τα στατιστικά στοιχεία του σήματος, στη δεύτερη δίνεται η δυνατότητα να κάνουμε κάποιες μετατροπές στο σήμα και στην τρίτη μπορούμε να αλλάξουμε το διάγραμμα (όχι το σήμα). Στη δεύτερη καρτέλα (Edit Signal) μπορούμε να μετατοπίσουμε το σήμα δεξιά-αριστερά και πάνω-κάτω καθώς και να του προσθέσουμε τυχαίο θόρυβο. Στην τρίτη καρτέλα μπορούμε να αλλάξουμε τα χρώματα του διαγράμματος, την ταχύτητα κίνησης, την



απόσταση μεταξύ του πλέγματος καθώς και το περιθώριο που υπάρχει γύρο από το σήμα και μέχρι το τέλος του διαγράμματος. Το περιθώριο αυτό υπάρχει για να μην έρχεται το σήμα κολλητά με το διάγραμμα. Κάνοντας το μεγαλύτερο αλλάζει και η κλίμακα στην οποία σχεδιάζεται το σήμα.

Πολλαπλά διαγράμματα:

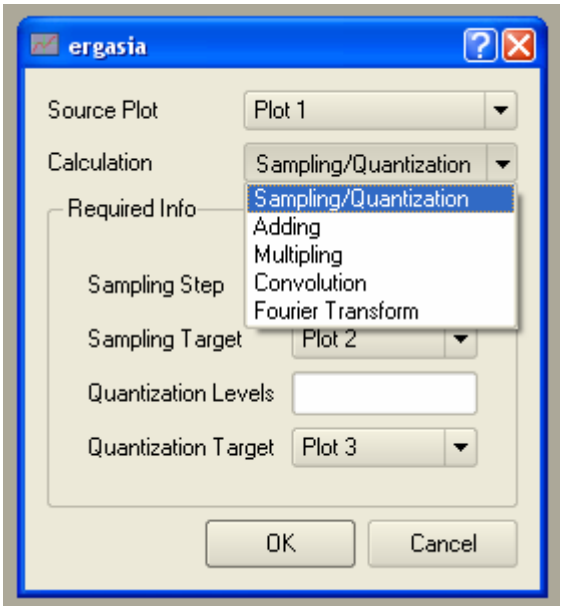
Σε αυτό το παράθυρο περιέχονται πολλά διαγράμματα του παραπάνω τύπου, που καθένα έχει το δικό του μενού και δίνεται η δυνατότητα διαχείρισής τους ταυτόχρονα όσο αφορά την κίνηση. Επίσης μπορούμε να προσθέσουμε διαγράμματα (στο σύνολο μέχρι 5) και να αφαιρέσουμε διαγράμματα από το παράθυρο. Ένα διάγραμμα θα περιέχεται πάντα και δεν μπορεί να αφαιρεθεί.



Εικόνα 11: Πολλαπλά διαγράμματα

Με το (+) και το (-) προσθέτουμε και αφαιρούμε διαγράμματα. Ακριβώς από δίπλα υπάρχει ένα check box για κάθε ένα διάγραμμα. Όλα τα επιλεγμένα διαγράμματα θα ακολουθούν τις εντολές Play / Pause / Stop, ενώ τα μη επιλεγμένα δεν επηρεάζονται (αν κινούνται, θα συνεχίσουν να κινούνται). Τέλος από το παράθυρο αυτό μας δίνει την δυνατότητα να κάνουμε πράξεις και

μετασχηματισμούς στα διαγράμματα που έχει φορτωμένα. Αυτό γίνεται με το τελευταίο στη σειρά κουμπί (Edit). Για να μπορούμε να κάνουμε πράξεις πρέπει να υπάρχουν τουλάχιστον τρία διαγράμματα και ένα από αυτά να έχει ένα σήμα φορτωμένο. Πατώντας το Edit εμφανίζεται ένας διάλογος από τον οποίο μπορούμε να επιλέξουμε την πράξη και τα διαγράμματα που θέλουμε.



Εικόνα 12: Διάλογος πράξεων

Το Source Plot είναι το βασικό διάγραμμα, του οποίου το σήμα θα χρησιμοποιηθεί για κάποιο μετασχηματισμό ή σαν πρώτη παράμετρος στην πράξη που θέλουμε να κάνουμε. Την πράξη την επιλέγουμε ακριβώς από κάτω. Η εφαρμογή υποστηρίζει Δειγματοληψία, Κβαντισμό, Πρόσθεση δύο σημάτων, Πολλαπλασιασμό δύο σημάτων, Συνέλιξη και μετασχηματισμό Fourier. Για κάθε μια από τις πράξεις αυτές χρειάζονται και άλλες παράμετροι, οι οποίες εμφανίζονται μέσα στο πλαίσιο Required Info. Επίσης κάθε πράξη έχει τους περιορισμούς της, δηλαδή:

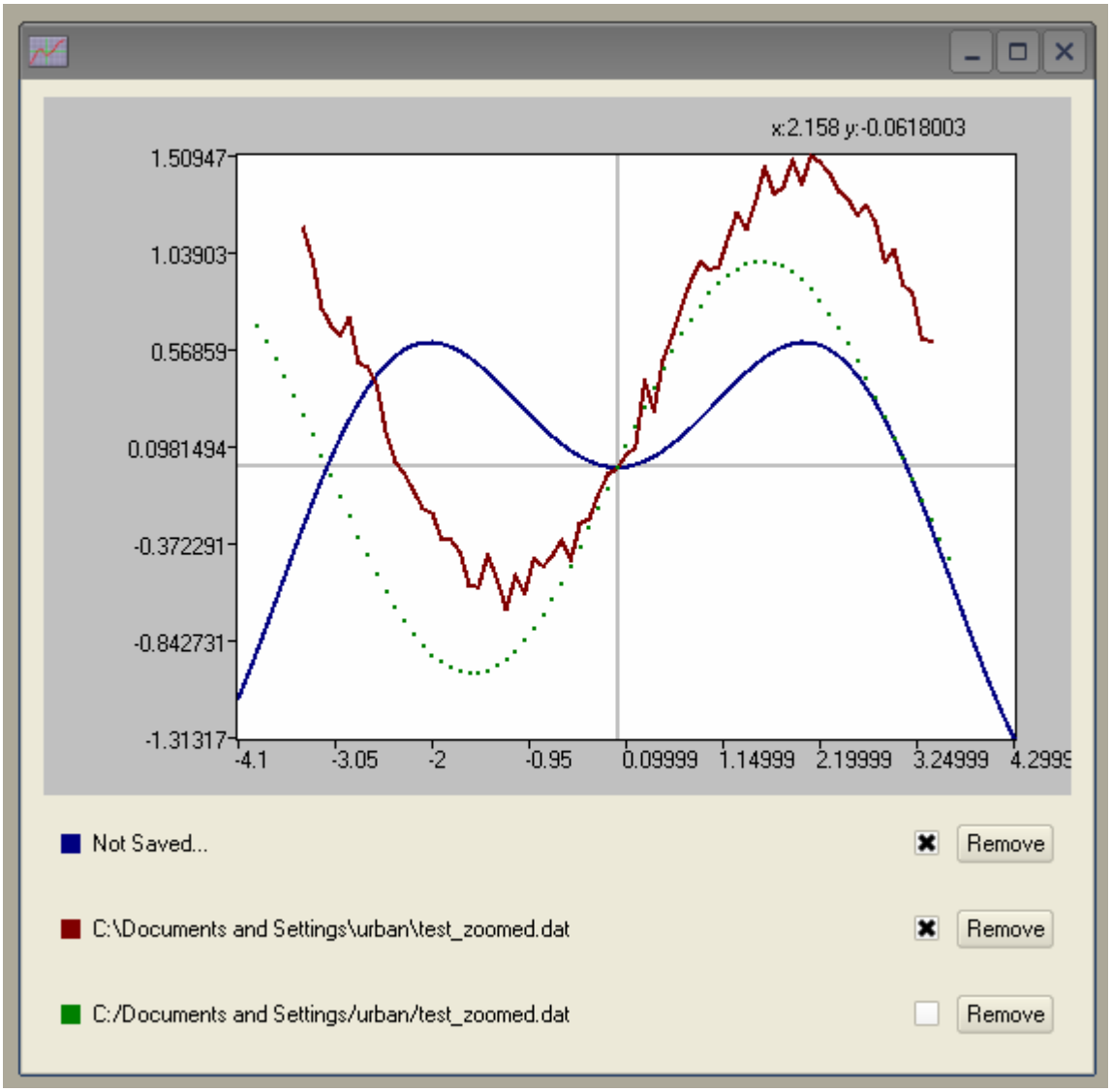
Στην δειγματοληψία δεν μπορούμε να έχουμε βήμα (Sampling Step) μηδέν και στον κβαντισμό δεν μπορούμε να έχουμε λιγότερες από δύο στάθμες (Quantization Levels). Εάν μία μόνο από τις δύο παραμέτρους έχει συμπληρωθεί, τότε θα γίνει μόνο εκείνο το μέρος, για παράδειγμα αν έχει συμπληρωθεί μόνο ο αριθμός των σταθμών τότε θα γίνει μόνο κβαντισμός στο σήμα του Source Plot.

Στην πρόσθεση, τον πολλαπλασιασμό και την συνέλιξη τα δύο σήματα που θα περάσουμε σαν παραμέτρους πρέπει να έχουν τουλάχιστον μία περιοχή που να ορίζονται και τα δύο και να έχουν το ίδιο βήμα. Λέγοντας βήμα εννοούμε την απόσταση μεταξύ δύο διαδοχικών σημείων στο σήμα, η οποία πρέπει να είναι και σταθερή για να έχουμε σωστό αποτέλεσμα.

Σε κάθε περίπτωση εάν ο πλήθος των διαγραμμάτων ή των σημάτων δεν είναι αρκετό για να γίνει η πράξη και να εμφανιστούν τα αποτελέσματα, εμφανίζεται μήνυμα σφάλματος στην θέση που κανονικά θα ήταν οι παράμετροι της πράξης και απενεργοποιείται το κουμπί OK.

Το αναλυτικό διάγραμμα:

Το αναλυτικό διάγραμμα ανοίγει σε ένα παράθυρο και το οποίο περιέχει ένα μόνο διάγραμμα. Αυτό είναι τελείως διαφορετικό από τα προηγούμενα διαγράμματα, στην ουσία τα συμπληρώνει. Προσφέρει τη δυνατότητα για αναλυτική μελέτη ενός σήματος, εμφανίζοντας χάρακες με κλίμακα στις δύο διαστάσεις χ και ψ. Το διάγραμμα αυτό δεν είναι σχεδιασμένο για μεγάλα σε μήκος σήματα αλλά για να αναλύονται συγκεκριμένες περιοχές οπότε δεν διαθέτει scrollbars δηλαδή η περιοχή που πλοτάρει είναι σταθερή. Βασικό πλεονέκτημα του, εκτός από την αρίθμηση στους άξονες, είναι ότι μπορεί να έχει παραπάνω από ένα σήματα ταυτόχρονα σχεδιασμένα σε κοινή κλίμακα.



Εικόνα 13: Αναλυτικό διάγραμμα

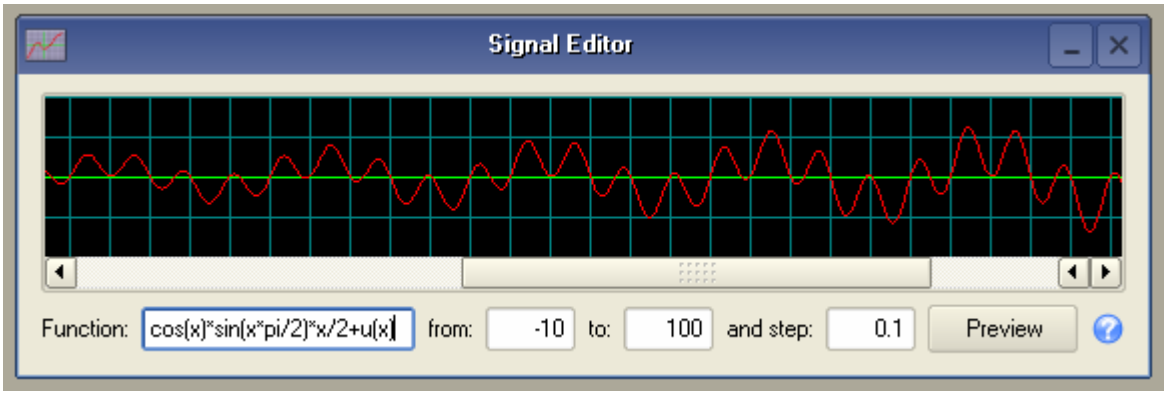
Για κάθε ένα σήμα που έχει φορτωμένο υπάρχει και μία γραμμή ελέγχου, κάτω από την περιοχή του διαγράμματος, στην οποία εμφανίζεται το χρώμα του σήματος και το όνομα του αρχείου στο οποίο είναι αποθηκευμένο. Από εκεί μπορούμε να επιλέξουμε, με το check box, εάν θέλουμε να σχεδιάζεται αναλογικά ή ψηφιακά το σήμα. Από το κουμπί Remove μπορούμε να αφαιρέσουμε κάποιο σήμα από το διάγραμμα. Το αναλυτικό διάγραμμα δέχεται drag 'n' drop από άλλα διαγράμματα. Τα σήματα που σέρνουμε μέσα στο διάγραμμα μπορεί να είναι ήδη μεγεθυμένα από το προηγούμενο διάγραμμα, σε αυτή την περίπτωση θα μεταφερθεί μόνο η επιλεγμένη για μεγέθυνση περιοχή και έτσι μπορούμε να μελετήσουμε συγκεκριμένες περιοχές ενός μεγάλου σε μήκος σήματος

Τέλος το διάγραμμα αυτό, όπως και τα προηγούμενα, έχει δικό του μενού (με δεξί κλικ) από το οποίο μπορούμε να φορτώσουμε κάποιο σήμα το οποίο είναι αποθηκευμένο σε αρχείο καθώς και

να το αποθηκεύσουμε σαν εικόνα.

Ο επεξεργαστής σημάτων:

Στον επεξεργαστή σημάτων έχουμε τη δυνατότητα να δημιουργήσουμε σήματα δίνοντας του σαν όρισμα μία συνάρτηση, την περιοχή που θέλουμε και το βήμα. Στο παράθυρο εμφανίζεται ένα απλό διάγραμμα στο οποίο πλοτάρεται το σήμα που δημιουργούμε.



Εικόνα 14: Επεξεργαστής σημάτων

Πατώντας το κουμπί με το ερωτηματικό εμφανίζεται ένα παράθυρο με τις συναρτήσεις που υποστηρίζονται για το πεδίο Function. Η ανεξάρτητη μεταβλητή της συνάρτησης πρέπει πάντα να είναι η (x) και δεν υποστηρίζεται δεύτερη μεταβλητή. Η σύνταξη που απαιτείται μοιάζει πολύ με αυτή της γλώσσας c. Οι τελεστές που υποστηρίζονται είναι:

() Έκφραση σε παρένθεση. Οι εκφράσεις αυτές θα εκτελεστούν πρώτες.

A^B Εκθετικό. Δηλαδή το A υψωμένο στην B.

-A Αρνητικό ενός αριθμού A.

!A Το λογικό not. Δίνει αποτέλεσμα 1 όταν το A είναι μηδέν.

A*B A/B A%B Πολλαπλασιασμός, Διαίρεση, Ακέραιο υπόλοιπο.

A+B A-B Πρόσθεση, Αφαίρεση.

A=B A!=B A<B Τελεστές σύγκρισης. Το αποτέλεσμα θα είναι είτε 0 είτε 1.
A<=B A>B A>=B

A&B Λογικό AND. Το αποτέλεσμα θα είναι είτε 0 είτε 1.

A|B Λογικό OR. Το αποτέλεσμα θα είναι είτε 0 είτε 1.

Μια ενδεικτική λίστα των συναρτήσεων που υποστηρίζονται είναι cos, sin, tan, sqrt, max, min, ceil, floor, log, exp, log10, abs. Για αναλυτική λίστα των συναρτήσεων πατήστε το Help του επεξεργαστή σημάτων.

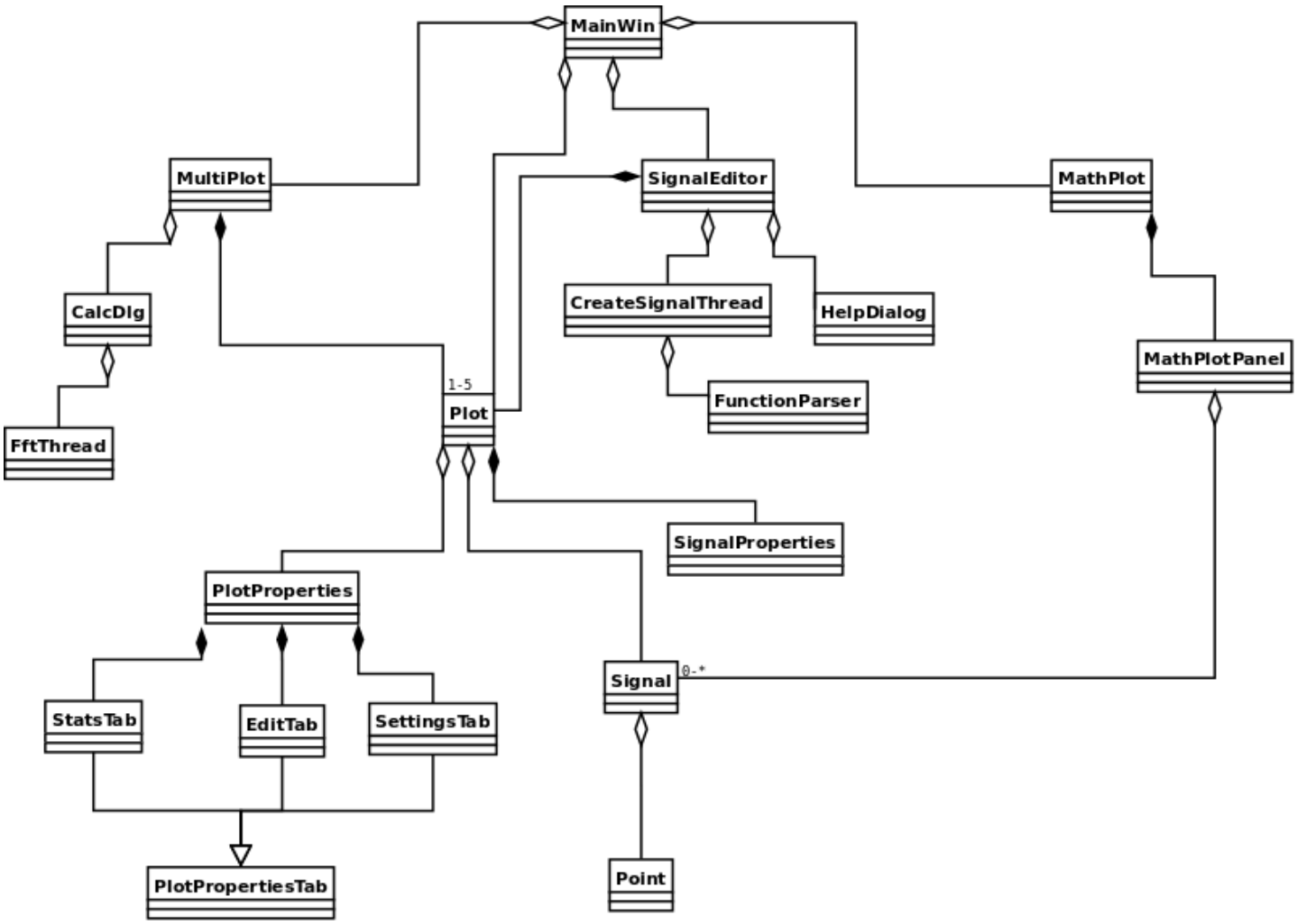
Στην σημείο αυτό γίνεται έλεγχος για σφάλματα τόσο όσο αφορά την μορφή της συνάρτησης καθώς και κατά την εκτέλεσή της. Αν υπάρχει λάθος στην συνάρτηση τότε θα εμφανιστεί μήνυμα σφάλματος στο οποίο θα αναφέρεται η θέση του χαρακτήρα που έγινε το πρώτο σφάλμα (με αρίθμηση από το 0). Αν παρουσιαστεί λάθος κατά την εκτέλεση (π.χ. διαίρεση με το μηδέν) εμφανίζεται πάλι το κατάλληλο μήνυμα

4 Τεκμηρίωση Πηγαίου κώδικα [περιεχόμενα]

Σε αυτή την ενότητα θα ασχοληθούμε με τη δομή της εφαρμογής από προγραμματιστικής απόψεως. Παρακάτω αναλύονται οι κλάσεις που δημιουργήθηκαν/χρησιμοποιήθηκαν, οι μεταξύ τους σχέσεις καθώς και οι μέθοδοί τους. Επίσης παρουσιάζονται και κάποια επιπλέον αρχεία πηγαίου κώδικα που χρησιμοποιούνται σαν βιβλιοθήκες για συγκεκριμένες ρουτίνες καθώς και αρχεία απαραίτητα για την μεταγλώττιση και εκτέλεση του κώδικα.

4.1 Διάγραμμα Οντοτήτων – Κλάσεων [περιεχόμενα]

Παρακάτω παρουσιάζεται ένα συνοπτικό διάγραμμα κλάσεων. Στο διάγραμμα δεν εμφανίζονται μέλη κλάσεων αλλά ούτε και μέθοδοι Αυτά θα περιγραφούν παρακάτω με περισσότερη λεπτομέρεια



Εικόνα 15: Διάγραμμα κλάσεων

Οι σχέσεις των παραπάνω κλάσεων με την βιβλιοθήκη του Qt είναι (':' = επεκτείνει):

MainWin : QMainWindow
MultiPlot : QWidget
SignalEditor : QWidget
CreateSignalThread : QThread
Plot : QGraphicsView
CalcDlg : Qdialog
FftThread : QThread
HelpDialog : QDialog
PlotPropTab : QWidget
PlotProperties : Qdialog

4.2 Αναλυτική περιγραφή των κλάσεων [περιεχόμενα]

Σε αυτήν την ενότητα θα αναλύσουμε τα μέλη και τις μεθόδους των κλάσεων.

Γρήγορο ευρετήριο κλάσεων:

Plot [top]

Η Plot είναι μια από τις βασικότερες κλάσεις τις εφαρμογής. Είναι υπεύθυνη για όλες τις λειτουργίες ενός διαγράμματος καθώς και για τη διαχείριση των σημάτων τα οποία απεικονίζονται σε αυτό. Σε αυτήν γίνεται η σχεδίαση του σήματος στην οθόνη, το animation του σήματος, zoom in και zoom out στο σήμα, η αποθήκευσή του σε αρχείο ή με τη μορφή εικόνας (PNG), λειτουργίες drag 'n' drop μεταξύ Plots, εμφάνιση ιδιοτήτων και αλλαγές χρωμάτων.

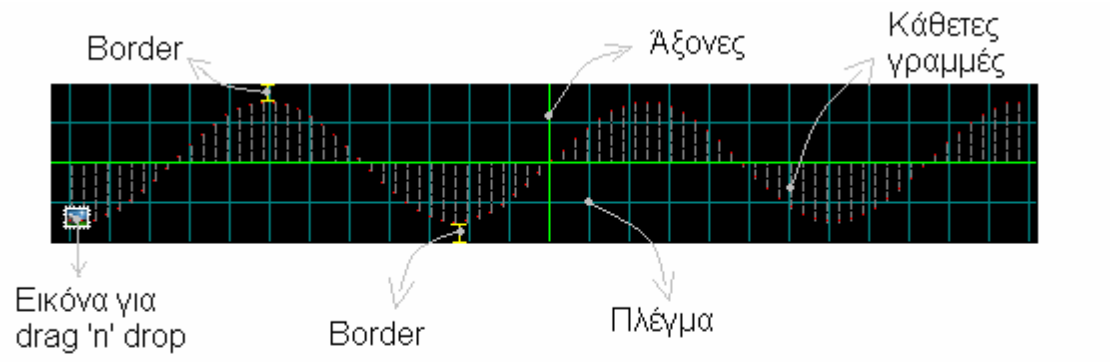
Σταθερές που έχουν οριστεί στην κλάση

Όνομα	Προκαθορισμένη Τιμή	Περιγραφή
GRID_Z	1.0	Η τιμή Z* που χρησιμοποιείται για να ζωγραφιστεί το πλέγμα στο διάγραμμα
LINE_Z	2.0	Η τιμή Z που χρησιμοποιείται για να ζωγραφιστούν οι κάθετες γραμμές προς τον άξονα X.
AXES_Z	3.0	Η τιμή Z που χρησιμοποιείται για να ζωγραφιστούν οι άξονες.

TEXT_Z	4.0	Η τιμή Z που θα χρησιμοποιούταν για να ζωγραφιστεί κείμενο στο διάγραμμα (δεν χρησιμοποιείται).
IMAGE_Z	9.0	Η τιμή Z που χρησιμοποιείται για να ζωγραφιστεί η εικόνα που κάνει drag 'n' drop.
SIGNAL_Z	10.0	Η τιμή Z που χρησιμοποιείται για να ζωγραφιστεί το σήμα.
RUNNING	0	Η κατάσταση του animation όταν “τρέχει”.
STOPED	1	Η κατάσταση του animation όταν είναι σταματημένο.
PAUSED	2	Η κατάσταση του animation όταν είναι “παγωμένο”.
DEFSCENE_W	300	Το default μήκος της σκηνής** του διαγράμματος (σε pixel).
DEFSCENE_H	80	Το default ύψος της σκηνής του διαγράμματος (σε pixel).

* Όσο μεγαλύτερη είναι η τιμή αυτή, τόσο πιο μπροστά από τα άλλα αντικείμενα θα ζωγραφίζεται το αντικείμενο με αυτό το Z.

** Σκηνή είναι ο χώρος που είναι διαθέσιμος για να ζωγραφιστούν αντικείμενα.



Εικόνα 16: Το διάγραμμα Plot

Public Μέθοδοι της Κλάσης

Plot(QWidget *parent =0)

Δομητής για την κατασκευή ενός άδειου Plot.

Plot(QString &filename, QWidget *parent =0)

Δομητής για την κατασκευή ενός Plot με φορτωμένο το σήμα από το αρχείο filename.

~Plot()

Εδώ γίνεται ότι χρειάζεται για τον σωστό τερματισμό-διαγραφή του διαγράμματος Πρέπει να καλεστεί η reject() για τον διάλογο PlotProperties. Εάν σταλθεί σήμα close() από το Plot τότε διαγράφεται σωστά, εάν όμως σταλθεί το σήμα close() από το MainWin τότε χωρίς την reject() ο διάλογος κλείνει αλλά δεν τερματίζεται η εφαρμογή (Κάτι περίεργο??).

QGraphicsScene * getScene()

Επιστρέφει δείκτη στην σκηνή του διαγράμματος

bool loadSignal(QString &filename)

Προσπαθεί να φορτώσει σήμα από το αρχείο filename. Αν αποτύχει επιστρέφει false αλλιώς επιστρέφει true. Μαζί με το σήμα διαβάζει και το .cfg αρχείο με τα Properties εάν αυτό υπάρχει στον ίδιο φάκελο και με το ίδιο filename. Καθαρίζει την υπάρχουσα σκηνή και δημιουργεί την καινούργια.

bool isEmpty()

Επιστρέφει true εάν το διάγραμμα απεικονίζει κάποιο σήμα και false εάν αυτό είναι άδειο.

void redrawScene()

Ξαναζωγραφίζει την σκηνή λαμβάνοντας υπ' όψιν τις μεταβλητές που καθορίζουν το πως θα ζωγραφιστεί το σήμα. Χρησιμοποιείται όταν αλλάζουν οι μεταβλητές αυτές και θέλουμε να κάνουμε εμφανή τα αποτελέσματα.

void setFgColor(QColor c)

Αλλάζει το χρώμα του σήματος. Η αλλαγή δεν είναι εμφανής μέχρι να καλεστεί η redrawScene().

void setBgColor(QColor c)

Αλλάζει το χρώμα του φόντου. Η αλλαγή δεν είναι εμφανής μέχρι να καλεστεί η redrawScene().

void setGridColor(QColor c)

Αλλάζει το χρώμα του πλέγματος. Η αλλαγή δεν είναι εμφανής μέχρι να καλεστεί η redrawScene().

void setAxesColor(QColor c)

Αλλάζει το χρώμα των αξόνων. Η αλλαγή δεν είναι εμφανής μέχρι να καλεστεί η redrawScene().

void setLinesColor(QColor c)

Αλλάζει το χρώμα των γραμμών που ζωγραφίζονται κάθετα στον άξονα X (μόνο στα ψηφιακά σήματα). Η αλλαγή δεν είναι εμφανής μέχρι να καλεστεί η redrawScene().

void setShowGrid(bool b)

Καθορίζει εάν θα φαίνεται το πλέγμα ή όχι, ανάλογα με το όρισμα που παίρνει (στο true φαίνεται). Η κλήση αυτής της μεθόδου προκαλεί ανανέωση της σκηνής και δεν χρειάζεται η κλήση της redrawScene.

void setDrawAnalog(bool b)

Καθορίζει τον τρόπο με τον οποίο θα συνδέονται τα σημεία του σήματος. Εάν το όρισμα είναι true τότε μεταξύ των σημείων θα ζωγραφίζεται γραμμή για να φαίνεται το σήμα σαν

αναλογικό. Αλλιώς ζωγραφίζονται μόνο τα σημεία. Η κλήση αυτής της μεθόδου προκαλεί ανανέωση της σκηνής και δεν χρειάζεται η κλήση της redrawScene.

void setShowLines(bool b)

Καθορίζει το εάν θα ζωγραφίζονται οι κάθετες προς τον X γραμμές στα αναλογικά μόνο σήματα. Η κλήση αυτής της μεθόδου προκαλεί ανανέωση της σκηνής και δεν χρειάζεται η κλήση της redrawScene.

void setGridSpace(int s)

Το πλέγμα είναι γραμμές που σχηματίζουν κουτάκια N επί N, όπου το N είναι σε pixel. Με την μέθοδο αυτή αλλάζουμε το N.

void setBorder(int b)

Η μέθοδος αυτή αλλάζει το περιθώριο που υπάρχει γύρο από το σήμα. Το b είναι σε pixel. Μπορεί να χρησιμοποιηθεί για να εμφανιστεί όλο το σήμα στο διάγραμμα χωρίς να χρειάζονται scrollbars.

void setAnimSleepTime(int time)

Η μέθοδος αυτή αλλάζει το sleep time του animation. Όσο μικρότερο είναι το όρισμα της μεθόδου τόσο πιο γρήγορα κινείται το σήμα.

void setSignal(Signal &sig)

Θέτει καινούριο σήμα στο διάγραμμα, καθαρίζει την σκηνή και το παλιό σήμα, και ζωγραφίζει το καινούριο.

void setSignalProperties(SignalProperties &sp)

Θέτει SignalProperties στο σήμα του διαγράμματος

QColor getFgColor()

Επιστρέφει το χρώμα του σήματος.

QColor getBgColor()

Επιστρέφει το χρώμα του φόντου.

QColor getGridColor()

Επιστρέφει το χρώμα του πλέγματος.

QColor getAxesColor()

Επιστρέφει το χρώμα των αξόνων.

QColor getLinesColor()

Επιστρέφει το χρώμα των κάθετων διακεκομμένων γραμμών .

Signal & getSignal()

Επιστρέφει το σήμα που ζωγραφίζεται την στιγμή αυτή.

bool getShowGrid()

Επιστρέφει true εάν εμφανίζεται το πλέγμα.

bool getDrawAnalog()

Επιστρέφει true εάν το σήμα ζωγραφίζεται αναλογικά.

bool getShowLines()
Επιστρέφει true εάν στα αναλογικά σήματα εμφανίζονται οι γραμμές.

int getGridSpace()
Επιστρέφει το περιθώριο μεταξύ των γραμμών του πλέγματος.

int getBorder()
Επιστρέφει το περιθώριο που υπάρχει γύρο από το σήμα (σε pixel).

int getAnimSleepTime()
Επιστρέφει το χρόνο που "κοιμάται"το animation.

int getAnimStatus()
Επιστρέφει την κατάσταση του animation, εάν είναι σταματημένο, εάν τρέχει ή εάν είναι παγωμένο. (βλ. Σταθερές που έχουν οριστεί στην κλάση)

SignalProperties & getSignalProperties()
Επιστρέφει το SignalProperties του σήματος που απεικονίζεται.

void startAnimation()
Ξεκινά το animation.

void pauseAnimation()
Παγώνει το animation.

void stopAnimation()
Σταματά το animation.

void resumeAnimation()
Συνεχίζει το animation όταν είναι παγωμένο.

Protected Μέθοδοι της Κλάσης

Οι παρακάτω μέθοδοι υπερβαίνουν τις μεθόδους του Qt ώστε να κάνουμε τις δουλείες που εμείς θέλουμε. Τα popup menu, drag 'n' drop και animation και zoom υλοποιούνται με αυτές τις μεθόδους.

void contextMenuEvent(QContextMenuEvent * event)
void mousePressEvent(QMouseEvent *event)
void mouseMoveEvent(QMouseEvent *event)
void mouseReleaseEvent(QMouseEvent *event)
void timerEvent(QTimerEvent * event)
void dragEnterEvent(QDragEnterEvent * event)
void dropEvent(QDropEvent *event)
void dragMoveEvent(QDragMoveEvent *event)
void dragLeaveEvent(QDragLeaveEvent * event)

Private Μέθοδοι της Κλάσης

void connectSlots()

Συνδέει όλα τα SIGNALS σε SLOTS για το Widget αυτό.

void createUI()

Στη μέθοδο αυτή δημιουργείται η γραφική αναπαράσταση του Widget και μπαίνουν τα επιμέρους widgets στην θέση τους.

void createMenus()

Δημιουργεί το popup menu και τα actions.

void setDefaults()

Αρχικοποιεί τις μεταβλητές σε προκαθορισμένες τιμές.

void applySignalProperties()

Θέτει τιμή στις μεταβλητές της κλάσης που επηρεάζονται από την SignalProperties. Χρησιμοποιείται όταν "διαβάζονται" καινούργια Properties, για να τα συγχρονιστούν μαζί τους οι μεταβλητές του διαγράμματος

void resizeRectCalcStep()

Υπολογίζει τις καινούργιες διαστάσεις της σκηνής βάσει του σήματος που είναι "φορτωμένο" στο διάγραμμα. Επίσεις υπολογίζει το καινούργιο βήμα που χρησιμοποιείται για να ζωγραφιστεί το σήμα. Καλείται από όλες τις μεθόδους που αλλάζουν το σήμα.

void drawGrid()

Ζωγραφίζει το πλέγμα.

void drawAxes()

Ζωγραφίζει τους άξονες.

void drawLines()

Ζωγραφίζει τις κάθετες γραμμές. Όταν δεν εμφανίζονται οι γραμμές, υπάρχουν στην οθόνη αλλά είναι κρυμμένες. Έτσι δεν χρειάζεται να ξαναζωγραφίζονται, απλώς καλείται η show() για κάθε γραμμή. Οι γραμμές αποθηκεύονται σε μια λίστα με pointers σε QGraphicsItem.

void clearScene()

Καθαρίζει την σκηνή από όλα τα αντικείμενα. Τα αντικείμενα διαγράφονται πλήρως.

void drawAnalogSignal()

Ζωγραφίζει αναλογικά το σήμα.

void drawDigitalSignal()

Ζωγραφίζει ψηφιακά το σήμα.

void animateAnalog()

Προχωράει ένα βήμα μπροστά το animation ενός αναλογικού σήματος. Η προηγούμενη θέση του σήματος κρατείται στην μεταβλητή animPos.

void animateDigital()

Προχωράει ένα βήμα μπροστά το animation ενός ψηφιακού σήματος. Η προηγούμενη θέση

του σήματος κρατείται στην μεταβλητή animPos.

void zoomIn()

Κάνει zoom in στην επιλεγμένη περιοχή. Χρησιμοποιεί τα σημεία selStart και selStop τα οποία αρχικοποιούνται από τις μεθόδους διαχείρισης του ποντικιού (mouseMoveEvent, mousePressEvent). **Οι συντεταγμένες των σημείων αυτών είναι στο σύστημα συντεταγμένων του Plot και όχι στης σκηνής**. Την προηγούμενη κατάσταση του σήματος την αποθηκεύει στο Stack signal (κάνοντας push).

QPointF signalToScene(Point & p) const

Μετατρέπει συντεταγμένες του σήματος σε συντεταγμένες της σκηνής.

Point sceneToSignal(QPointF & p) const

Μετατρέπει συντεταγμένες της σκηνής σε συντεταγμένες του σήματος.

Public Slots της Κλάσης

void load()

Εμφανίζει το διάλογο ανοίγματος αρχείου και μετά καλεί την loadSignal().

void save()

Εμφανίζει το διάλογο αποθήκευσης σε αρχείο και αποθηκεύει το σήμα και τα Properties του (σε .cfg με το ίδιο όνομα αρχείου).

void saveAsImage()

Εμφανίζει το διάλογο αποθήκευσης σε αρχείο και αποθηκεύει το σήμα σε μορφή PNG εικόνας. Δεν αποθηκεύει όλο το σήμα αλλά μόνο την περιοχή που φαίνεται.

void toggleGrid()

Αλλάζει την κατάσταση του πλέγματος. Εάν εμφανίζεται το σβήνει και το αντίθετο.

void toggleLines()

Αλλάζει την κατάσταση των γραμμών. Εάν εμφανίζονται τις σβήνει και το αντίθετο.

void toggleZoomLock()

Αλλάζει την τιμή της μεταβλητής zoomLock. Εάν είναι true τότε δεν γίνεται zoom παρόλο που μπορεί να επιλεχθεί μια περιοχή. (Για να μην γίνεται zoom κατα λάθος)

void changeSignalType()

Εάν το σήμα ζωγραφίζεται αναλογικό τότε το κάνει ψηφιακό και το αντίθετο. Η μέθοδος αυτή κάνει την σκηνή να ξαναζωγραφιστεί καλώντας την redrawScene().

void zoomOut()

Γυρίζει το σήμα στην προηγούμενη κατάσταση zoom. Για να το κάνει αυτό χρησιμοποιεί το Stack signal, κάνει pop το πάνω σήμα (που είναι και η τωρινή κατάσταση) και ξαναζωγραφίζει τη σκηνή βάση της προηγούμενης κατάστασης.

void showPropDlg()

Κάνει update το διάλογο με τις ιδιότητες του σήματος (βλ. Κλάση PlotProperties) και εμφανίζει τον διάλογο.

void startStopAnim()

Εάν το σήμα κινείται τότε το σταματά και το αντίθετο.

void pauseResumeAnim()

Εάν το σήμα είναι παγωμένο τότε συνεχίζει το animation και το αντίθετο.

PlotProperties [top]

Η κλάση PlotProperties είναι ένας διάλογος μέσα από τον οποίο παρουσιάζονται τα στατιστικά ενός σήματος, γίνονται διάφορες αλλαγές σε ένα σήμα και αλλάζουν τα χρώματα από ένα διάγραμμα. Για να γνωρίζει η PlotProperties σε ποιο διάγραμμα αναφέρεται και από εκεί να παίρνει το σήμα που απεικονίζεται, πρέπει πάντα να δημιουργείται με κάποιο parent widget. Το widget αυτό γίνεται casting σε Plot μέσα στον δομητή. Έτσι ένα αντικείμενο της κλάσης αυτής δεν μπορεί να μην έχει parent. Ο διάλογος αυτός εκτελείται σε non-Modal mode, δηλαδή όταν εκτελείται, επιτρέπεται στον χρήστη να δουλεύει και την εφαρμογή. Για τον λόγο αυτό πρέπει να είναι πάντα συγχρονισμένος ο διάλογος με το διάγραμμα από το οποίο καλέστηκε. Τέλος να αναφέρουμε ότι ένα αντικείμενο της PlotProperties θα είναι πάντα συγχρονισμένο και με το SignalProperties του διαγράμματος ώστε αν ζητηθεί να γίνει αποθήκευση του σήματος, η Plot να αποθηκεύσει τα σωστά Properties στο .cfg αρχείο.

Public Μέθοδοι της Κλάσης

PlotProperties (QWidget *parent,Qt::WindowFlags f=0)

Ο δομητής της κλάσης. Δημιουργεί ένα αντικείμενο που υποχρεωτικά έχει parent.

virtual ~PlotProperties()

void updateContents()

Η μέθοδος αυτή κάνει update όλες τις καρτέλες του διαλόγου. Όταν καλείται, στην ουσία κάνει ξανά casting το parent αντικείμενο σε Plot και καλεί τις update μεθόδους τις κάθε καρτέλας με τα απαραίτητα ορίσματα.

Private Μέθοδοι της Κλάσης

void createUI()

Εδώ αρχικοποιούνται όλα τα αντικείμενα του GUI και καλείται η update των καρτελών ώστε να πάρουν τις σωστές τιμές.

Public Slots της Κλάσης

void okClicked()

Καλεί την applyChanges() της καρτέλας που είναι επιλεγμένη ώστε να εφαρμοστούν οι αλλαγές. Αμέσως μετά καλείται η accept() του διαλόγου και κλείνει.

void manageApply()

Ελέγχει εάν έχει αλλαχθεί η καρτέλα που είναι επιλεγμένη και αν έχει τότε ενεργοποιεί το κουμπί Apply αλλιώς το απενεργοποιεί.

void applyClicked()

Καλεί την applyChanges() της καρτέλας που είναι επιλεγμένη ώστε να εφαρμοστούν οι αλλαγές που έχουν γίνει, αλλά ο διάλογος παραμένει ανοιχτός.

void changeAnimSpeed(int value)

Η μέθοδος αυτή αλλάζει την ταχύτητα του animation χωρίς να έχει πατηθεί το κουμπί Apply και οι αλλαγή εφαρμόζεται αμέσως.

PlotPropTab [top]

Η κλάση αυτή χρησιμοποιείται για την λογική ομαδοποίηση των καρτελών του διαλόγου PlotProperties. Το widget της κάθε καρτέλας επεκτείνει την κλάση αυτή. Έτσι εξασφαλίζεται ότι κάθε υποτάξη θα έχει κάποιες μεθόδους και δεν χρειάζεται να γίνεται casting στη συγκεκριμένη υποκλάση αλλά αρκεί να γίνει casting σε PlotPropTab. Χρησιμοποιείται από την κλάση PlotProperties κυρίως στις λειτουργίες Apply και Ok.

Public Μέθοδοι της κλάσης

PlotPropTab(QWidget * parent=0)

Ο Δομητής της κλάσης.

virtual ~PlotPropTab()

virtual bool isModified()

Επιστρέφει το εάν έχει αλλάξει το περιεχόμενο του widget. (π.χ. Κάποιο QLineEdit άλλαξε περιεχόμενο)

Public Slots της κλάσης

virtual void applyChanges(Plot * plot)

Εφαρμόζει τις αλλαγές που ζητήθηκαν. Για όρισμα παίρνει ένα Plot πάνω στο οποίο θα εφαρμόσει τις αλλαγές.

Signals της κλάσης

void modified()

Είναι ένα διαθέσιμο signal που χρησιμοποιείται για να γνωστοποιήσει στα γύρο αντικείμενα ότι άλλαξε.

StatsTab [top]

Η StatsTab επεκτείνει την PlotPropTab και είναι το widget που χρησιμοποιείται στην καρτέλα που εμφανίζει τα στατιστικά στοιχεία του σήματος. Σε αυτήν την καρτέλα η μέθοδος modified() επιστρέφει πάντα false, διότι δεν υπάρχει κάτι να αλλάξει. Έτσι το κουμπί Apply θα είναι απενεργοποιημένο.

Public Μέθοδοι της κλάσης

StatsTab(QWidget * parent=0)

Ο δομητής της κλάσης .

virtual ~StatsTab()

void updateForSignal(Signal &sig)

Για κάθε στατιστικό του σήματος υπάρχει και ένα QLabel το οποίο πρέπει να ανανεώνεται όταν το σήμα αλλάζει. Τη δουλεία αυτή την κάνει η updateForSignal παίρνοντας σαν όρισμα το σήμα του διαγράμματος. Η μέθοδος αυτή καλείται από την updateContents() της PlotProperties όποτε ζητείται ανανέωση του διαλόγου.

EditTab [top]

Η EditTab επεκτείνει την PlotPropTab και είναι το widget από το οποίο μπορούμε να εφαρμόσουμε διάφορους μετασχηματισμούς στο σήμα. Η κλάση αυτή πρέπει να είναι συγχρονισμένη με την SignalProperties ώστε εάν ένα σήμα φορτωθεί σε ένα διάγραμμα, στην καρτέλα αυτή να φαίνεται ο θόρυβος που υπάρχει στο σήμα καθώς και οι μετατοπίσεις που έχουν γίνει.

Public Μέθοδοι της κλάσης

EditTab(QWidget * parent=0)

Ο δομητής της τάξης.

virtual ~EditTab()

void updateSignalProperties(SignalProperties &prop)

Η μέθοδος αυτή ανανεώνει τα SignalProperties του σήματος ώστε να είναι συγχρονισμένα με τα Properties του διαγράμματος. Καλείτε από την updateContents() της PlotProperties όποτε ζητήται ανανέωση του διαλόγου.

Public Slots της κλάσης

void setModified()

Όταν αλλάξει το περιεχόμενο κάποιου LineEdit τότε καλείται η μέθοδος αυτή. Θέτει την μεταβλητή mod που κληρονομείται από την PlotPropTab σε true και αμέσως μετά κάνει emit το signal modified() ώστε να ενημερωθεί η PlotProperties και να ενεργοποιήσει το κουμπί Apply του διαλόγου.

virtual void applyChanges(Plot * plot)

Η μέθοδος αυτή κάνει την μετατροπή που ζητήθηκε στο σήμα και καλεί την redrawScene() της Plot για να ξαναζωγραφιστεί το σήμα. Επίσης ανανεώνει την SignalProperties του διαγράμματος και καθαρίζει όλα τα πεδία. Τέλος αλλάζει την μεταβλητή mod σε false και κάνει emit την modified() για να ενημερωθεί το κουμπί Apply. **Η ίδια η μέθοδος δεν αλλάζει τα labels με τα Properties (παρόλο που θα μπορούσε) αλλά περιμένει να γίνει update από την redrawScene()**.

SettingsTab [top]

Η SettingsTab επεκτείνει την PlotPropTab και είναι το widget από το οποίο μπορούμε να αλλάξουμε τις παραμέτρους του διαγράμματος, όπως τα χρώματα, το περιθώριο και την απόσταση των γραμμών στο πλέγμα.

Public Μέθοδοι της κλάσης

SettingsTab(QWidget * parent=0)

Ο Δομητής της κλάσης.

virtual ~SettingsTab()

void updateSettings(Plot *plot)

Η μέθοδος αυτή ανανεώνει τις τιμές της καρτέλας παίρνοντας σαν όρισμα ένα Plot. Αρχικοποιεί τα χρώματα και θέτει την μεταβλητή mod σε false. Καλείται συνήθως από την updateContents την PlotProperties για να συγχρονίσει το διάγραμμα με την καρτέλα.

Private Μέθοδοι της κλάσης

void chColor(QPushButton *but,QColor c)

Αλλάζει το background χρώμα του κουμπιού but στο χρώμα του που παίρνει σαν όρισμα.

QColor getButColor(QPushButton * but)

Επιστρέφει το background χρώμα ενός κουμπιού.

Public Slots της κλάσης

void setModified()

Θέτει την μεταβλητή mod σε true και κάνει emit το signal modified(). Χρησιμοποιείται για να ενεργοποιήσει το κουμπί Apply του διαλόγου.

virtual void applyChanges(Plot * plot)

Κάνει όλες τις αλλαγές που ζητήθηκαν στο διάγραμμα και ξαναζωγραφίζει το διάγραμμα με την redrawScene() της κλάσης Plot. Τέλος θέτει την mod σε false αφού πλέον δεν υπάρχουν αλλαγές.

void choseColor(QAbstractButton *)

Η μέθοδος αυτή εμφανίζει το διάλογο επιλογής χρώματος και αν ο χρήστης επιλέξει κάποιο χρώμα, τότε αλλάζει το background χρώμα του κουμπιού που πατήθηκε, στο επιλεγμένο. Η αλλαγή γίνεται καλώντας την chColor().

void animSpeedChanged(int v)

Η μέθοδος αυτή καλείται όταν αλλάζει θέση η μπάρα για την ταχύτητα του animation. **Δεν γίνεται emit το signal speedChanged() διότι οι τιμές της μπάρας είναι ανάποδα.** Όσο πιο δεξιά πάει η μπάρα και αυξάνεται η τιμή της, τόσο πιο πολύ μικραίνει η τιμή που πρέπει να περαστεί στην speedChanged. Αυτό συμβαίνει γιατί στην ουσία δεν υπάρχει ταχύτητα animation αλλά αλλάζει το animSleepTime της κλάσης Plot (που είναι αντιστρόφως ανάλογο της ταχύτητας).

Signals

void speedChanged(int v)

Ενημερώνει την κλάση PlotProperties για να αλλάξει το animSleepTime της Plot χωρίς να έχει πατηθεί το κουμπί Apply.

SignalProperties [top]

Η κλάση αυτή χρησιμοποιείται για να διαβάζουμε τα .cfg αρχεία. Τα αρχεία αυτά αποθηκεύονται στον ίδιο φάκελο που αποθηκεύεται και το αρχείο του σήματος και κρατάνε πληροφορίες για το αν το σήμα είναι αναλογικό η όχι, κατά πόσο είναι μετατοπισμένο ή κατά πόσο έχει θόρυβο. Τα στοιχεία αυτά θα μπορούσαν να είναι αποθηκευμένα στο ίδιο αρχείο με το σήμα αλλά τότε δεν θα μπορούσαμε να φορτώνουμε αρχεία που έχουν παραχθεί από άλλα προγράμματα, στο δικό μας.

Public Μέθοδοι της κλάσης

SignalProperties()

Ο δομήτης αυτός αρχικοποιεί ένα αντικείμενο SignalProperties με τις προκαθορισμένες τιμές.

SignalProperties(const char * filename)

Ο δομήτης αυτός αρχικοποιεί ένα αντικείμενο SignalProperties διαβάζοντας τις τιμές των μεταβλητών από το αρχείο με διαδρομή filename. Για να το κάνει αυτό χρησιμοποιεί το υπερφορτωμένο τελεστή >>.

~SignalProperties()

void loadSignalProp(const char * filename)

Η μέθοδος αυτή αλλάζει τις τιμές ενός αντικειμένου SignalProperties διαβάζοντας τις καινούργιες από το αρχείο filename. Εάν αυτό για οποιοδήποτε λόγο δεν μπορέσει να ανοίξει το αρχείο, τότε το αντικείμενο δεν αλλάζει τιμές.

void save(const char * filename)

Αποθηκεύει το αντικείμενο στο αρχείο filename. Για να το κάνει αυτό χρησιμοποιεί τον υπερφορτωμένο τελεστή <<.

bool isAnalog()

Επιστρέφει true εάν το σήμα είναι αναλογικό.

double getNoise()

Επιστρέφει το ποσοστό θορύβου (επί τις εκατό) στο σήμα.

double getXMove()

Επιστρέφει το κατά πόσο είναι μετατοπισμένο το σήμα στον άξονα X.

double getYMove()

Επιστρέφει το κατά πόσο είναι μετατοπισμένο το σήμα στον άξονα Y.

void setAnalog(bool a)

Θέτει το σήμα αναλογικό εάν το a είναι true και σε ψηφιακό εάν είναι false.

void setNoise(double n)

Θέτει το ποσοστό θορύβου στο σήμα σε n.

void setXMove(double xm)

Θέτει την μετατόπιση του σήματος στον άξονα X σε xm.

void setYMove(double ym)

Θέτει την μετατόπιση του σήματος στον άξονα Y σε ym.

friend ofstream & operator<<(ofstream &out, const SignalProperties & sp)

Υπερφορτωμένος τελεστής για εγγραφή του αντικειμένου σε αρχείο.

friend ifstream & operator>>(ifstream &in, SignalProperties & sp)

Υπερφορτωμένος τελεστής για ανάγνωση του αντικειμένου από αρχείο.

SignalEditor [top]

Η κλάση αυτή δημιουργεί το Widget του editor, από τον οποίο μπορούμε να δημιουργήσουμε σήματα βάσει της μαθηματικής τους συνάρτησης. Για να το κάνουμε αυτό χρησιμοποιήσαμε τον fparser των Juha Nieminen και Joel Yliluoma. Μέσω του fparser διαβάζουμε την συνάρτηση του σήματος, ελέγχουμε για συντακτικά σφάλματα και εκτελούμε την συνάρτηση. Επίσης ο parser έχει την δυνατότητα να μας λέει για σφάλματα κατά την εκτέλεση της συνάρτησης όπως για παράδειγμα διαίρεση με το 0 ή αρνητικό όρισμα σε αλγόριθμο (Για περισσότερες πληροφορίες για τον fparser στο ../fparser28/fparser.txt).

Public Μέθοδοι της κλάσης

SignalEditor(QWidget * parent=0)

Ο δομητής της κλάσης.

~SignalEditor()

Public Slots της κλάσης

void previewClicked()

Καλείται όταν πατηθεί το κουμπί Preview. Ελέγχει εάν είναι συμπληρωμένα όλα τα απαραίτητα πεδία καθώς και το αν τα "από", "εώς" είναι στη σωστή σειρά. Έπειτα δημιουργεί ένα αντικείμενο CreateSignalThread (νήμα) στο οποίο περνά τη συνάρτηση και τις παραμέτρους της. Αυτό το thread θα εκτελέσει την συνάρτηση χωρίς να κολλήσει κανένα άλλο παράθυρο, εκτός από τον SignalEditor.

void helpClicked()

Καλείται όταν πατηθεί το ερωτηματικό δίπλα από το Preview. Διαβάζει το μήνυμα βοήθειας από το αρχείο ":src/fparser.help" που είναι δηλωμένο και στα resources, και χρησιμοποιεί ένα HelpDialog για να εμφανίσει το παράθυρο βοήθειας με το κατάλληλο μήνυμα.

CreateSignalThread [top]

Η κλάση αυτή επεκτείνει την Qthread. Εκτελεί (με την χρήση του fparser) μία συνάρτηση και αρχικοποιεί ένα σήμα. Μετά την εκτέλεση του νήματος μπορούμε να πάρουμε τα αποτελέσματα και να δούμε αν έγινε κάποιο σφάλμα. Η CreateSignalThread δεν είναι υπεύθυνη για την ενημέρωση του χρήστη, με κάποιο διάλογο, για τυχόν σφάλματα. Αν γίνει σφάλμα σταματά η εκτέλεση του νήματος.

Public Μέθοδοι της κλάσης

CreateSignalThread(QString func, double from, double to, double step, QObject * parent = 0)

Ο δομητής της κλάσης, σε αυτόν περνιούνται παραμετρικά η συνάρτηση και οι παράμετροί της καθώς και το parent αντικείμενο. Αν καταστραφεί το parent αντικείμενο τότε τερματίζεται και το νήμα.

virtual ~CreateSignalThread()

Signal getSignal()

Επιστρέφει το σήμα που δημιουργήθηκε από την εκτέλεση του νήματος. ΠΡΟΣΟΧΗ: Πρέπει να καλείται μόνο μετά την εκτέλεση του νήματος (μετά το signal finished()) για να δώσει σωστά αποτελέσματα.

bool isOK()

Επιστρέφει true εάν δεν έχει παρουσιαστεί σφάλμα.

QString getErrorMsg()

Επιστρέφει το μήνυμα του πρώτου σφάλματος, εάν υπάρχει κάποιο.

void run()

Δημιουργεί ένα αντικείμενο FunctionParser, του προσθέτει την σταθερά π και τις συναρτήσεις u,δ (με τις μεθόδους AddConstant και AddFunction). Στη συνέχεια καλείται η Parse του αντικειμένου. Σε οποιοδήποτε λάθος καταγράφεται το κατάλληλο μήνυμα (στην μεταβλητή error) και η διαδικασία σταματά. Τέλος εκτελείται η συνάρτηση. Κατά την εκτέλεση γίνεται έλεγχος για τυχόν σφάλματα από τον fparser. Άν το αποτέλεσμα της εκτέλεσης είναι ένα σημείο τότε δεν θα ζωγραφιστεί τίποτα και θα η error θα έχει το κατάλληλο μήνυμα σφάλματος.

HelpDialog [top]

Η κλάση αυτή δημιουργήθηκε για διευκόλυνση και μόνο. Είναι στην ουσία ένα QDialog το οποίο παίρνει σαν παράμετρο στον δομητή ένα QString. Έτσι δημιουργεί ένα παράθυρο με ένα TextArea με το κείμενο που πήρε για όρισμα ο δομητής και μόνο το κουμπί OK. Σκοπός της κλάσης αυτής είναι να χρησιμοποιηθεί για εμφάνιση μηνυμάτων βοηθείας.

Public Μέθοδοι της κλάσης

HelpDialog(QString helpMsg, QWidget *parent=0,Qt::WindowFlags=0)

Ο δομητής της τάξης.

~HelpDialog()

MultiPlot [top]

Η κλάση αυτή δημιουργεί ένα Widget με πολλά Plots ταυτόχρονα τα οποία αποθηκεύονται σε μια στοίβα με πρώτο το πάνω. Επίσης έχει και ένα Panel για την διαχείριση των διαγραμμάτων αυτών. Από εκεί μπορούν να προστεθούν και να αφαιρεθούν διαγράμματα από το Widget και γίνεται ο έλεγχος του animation. Για κάθε διάγραμμα υπάρχει και ένα check box. Όταν πατηθεί κάποιο από τα κουμπιά Play/Pause/Stop τότε τα επιλεγμένα διαγράμματα θα πάρουν την εντολή, τα μη επιλεγμένα θα παραμείνουν στην ίδια κατάσταση.

Σταθερές που έχουν οριστεί στην κλάση.

Όνομα	Προκαθορισμένη Τιμή	Περιγραφή
MAXPLOTS	5	Ο μέγιστος αριθμός διαγραμμάτων που μπορεί να έχει το widget.
DEFPLOTS	3	Ο προκαθορισμένος αριθμός διαγραμμάτων όταν ανοίγει το widget

Public Μέθοδοι της κλάσης

MultiPlot(QWidget* parent = 0)
Ο δομητής της κλάσης.

~MultiPlot()

QStack<Plot *> getPlots()
Επιστρέφει τη στοίβα με τα διαγράμματα που έχει το Widget αυτή τη στιγμή. Η μέθοδος αυτή χρησιμοποιείται από την CalcDlg για να μπορεί να κάνει αλλαγές στα διαγράμματα και να ελέγχει εάν είναι άδεια.

Private Μέθοδοι της κλάσης

void createUI()
Στη μέθοδο αυτή δημιουργείται η γραφική αναπαράσταση του Widget και μπαίνουν τα επιμέρους widgets στην θέση τους.

void fixSize()
Υπολογίζει και εφαρμόζει (κάνοντας resize) το καινούργιο μέγεθος του widget, όταν σε αυτό προστίθενται/αφαιρούνται διαγράμματα. Τα διαγράμματα μπαίνουν δυναμικά στο layout του widget και από μόνο του το widget δεν μπορεί να υπολογίσει το κατάλληλο μέγεθος.

Public slots της κλάσης

void addClicked()
Προσθέτει ένα Plot στο widget και στην στοίβα με τα διαγράμματα. Επίσης προσθέτει μια

ετικέτα και ένα check box στις αντίστοιχες στοίβες, ώστε να μπορούμε να ελέγχουμε το καινούργιο διάγραμμα. Ελέγχει για το εάν είναι το τελευταίο διάγραμμα που μπορούμε να προσθέσουμε (αυτό καθορίζεται από την σταθερά MAXPLOTS) και αν είναι απενεργοποιεί το κουμπί add. Το ίδιο ελέγχει και για το εάν μπορούν να αφαιρεθούν διαγράμματα και εάν μπορούν ενεργοποιεί το κουμπί remove. Τέλος καλεί την fixSize().

void removeClicked()

Αφαιρεί το τελευταίο διάγραμμα από την στοίβα και το widget, καθώς και τα controls του. Ελέγχει το πλήθος των διαγραμμάτων και εάν είναι μόνο ένα τότε απενεργοποιεί το κουμπί remove. Επίσης ελέγχει για το εάν μπορούν να προστεθούν διαγράμματα και εάν μπορούν ενεργοποιεί το κουμπί add. Τέλος καλεί την fixSize().

void playClicked()

Για καθένα από τα επιλεγμένα διαγράμματα ελέγχει εάν είναι σταματημένο ή εάν είναι παγωμένο. Στην περίπτωση που είναι σταματημένο καλεί την startAnimation() του διαγράμματος ενώ στην άλλη περίπτωση καλεί την resumeAnimation().

void pauseClicked()

Για κάθε επιλεγμένο διάγραμμα καλεί την pauseAnimation() του.

void stopClicked()

Για κάθε επιλεγμένο διάγραμμα καλεί την stopAnimation() του.

void editClicked()

Ελέγχει για το εάν υπάρχουν τουλάχιστον τρία διαγράμματα και τουλάχιστο ένα το οποίο να μην είναι άδειο. Εάν πληρούνται οι παραπάνω προϋποθέσεις ανοίγει τον διάλογο για τους υπολογισμούς μεταξύ σημάτων.

CalcDlg [top]

Η κλάση CalcDlg δημιουργεί έναν διάλογο μέσα από τον οποίο μπορούμε να κάνουμε πράξεις μεταξύ σημάτων ή μετασχηματισμούς ενός σήματος. Για κάθε πράξη ή μετασχηματισμό υπάρχει και ένα QWidget το οποίο έχει τα απαραίτητα πεδία που πρέπει να συμπληρωθούν για να γίνει η πράξη. Όταν επιλέγεται κάποιος υπολογισμός εμφανίζεται το κατάλληλο widget.

Public Μέθοδοι της κλάσης

CalcDlg(QWidget * parent,Qt::WindowFlags f=0)

Ο δομητής της τάξης, σε αυτόν αρχικοποιούνται όλα τα επιμέρους widget για κάθε υπολογισμό. Επίσης εδώ αρχικοποιείται το combo box για το source διάγραμμα, το οποίο δεν μπορεί να είναι κάποιο άδειο διάγραμμα. Για να γίνουν όλα αυτά, **ένα αντικείμενο της κλάσης CalcDlg πρέπει οποιοδήποτε να έχει parent ένα αντικείμενο της κλάσης MultiPlot**. Ένα από τα πρώτα πράγματα που γίνονται είναι να γίνει casting του parent από widget σε MultiPlot για να έχουμε πρόσβαση στα διαγράμματα.

~CalcDlg()

Private Μέθοδοι της κλάσης

void createSampQuantWidget()
Δημιουργεί το widget για δειγματοληψία και κβαντισμό.

void createAddMultiWidget()
Δημιουργεί το widget για τον πολλαπλασιασμό σημάτων.

void createFftWidget()
Δημιουργεί το widget για τον μετασχηματισμό Fourier.

void createErrorPane()
Δημιουργεί ένα widget που χρησιμοποιείται όταν δεν είναι επαρκή τα διαγράμματα ή τα σήματα, και εμφανίζει μόνο ένα μήνυμα σφάλματος.

int loadedSignals()
Επιστρέφει τον αριθμό τον σημάτων που είναι φορτωμένα στα διαγράμματα του MultiPlot αντικειμένου.

Public slots της κλάσης

void refresh2ndCombo()
Ανανεώνει το δεύτερο combo box από το widget της πράξης που είναι επιλεγμένη, χωρίς όμως να αλλάζει το πρώτο combo box. Αυτό χρησιμοποιείται όταν ο χρήστης αλλάξει την τιμή του πρώτου combo box.

void refresh3dCombo()
Ανανεώνει το τρίτο combo box από το widget της πράξης που είναι επιλεγμένη (μόνο ο fft έχει τρία), χωρίς όμως να αλλάζει το πρώτο και το δεύτερο combo box. Αυτό χρησιμοποιείται όταν ο χρήστης αλλάξει την τιμή του δεύτερου combo box.
void refreshCombos()
Ανανεώνει όλα τα combo box από το widget της πράξης που είναι επιλεγμένη. Χρησιμοποιείται όταν ο χρήστης αλλάξει την πράξη που θέλει να κάνει και χρησιμοποιείται και από τον δομητή για την αρχικοποίηση των combo box.

void okClicked()
Καλείται όταν πατηθεί το κουμπί OK. Ανάλογα με την πράξη που είναι επιλεγμένη γίνονται οι απαραίτητοι έλεγχοι για το αν είναι συμπληρωμένα όλα τα απαραίτητα πεδία, και στη συνέχεια γίνεται η πράξη. Στην δειγματοληψία και τον κβαντισμό μπορεί να γίνει μόνο το ένα ή μόνο το άλλο ανάλογα με τα πεδία που είναι συμπληρωμένα.

void calcChanged()
Καλείται όποτε αλλάζει η πράξη που είναι να γίνει. Πετιέται το widget της προηγούμενης πράξης και αντικαθίσταται με το καινούριο. Σε περίπτωση που τα διαγράμματα ή τα σήματα που είναι φορτωμένα σε αυτά δεν είναι αρκετά, εμφανίζεται το widget με το μήνυμα σφάλματος.

void fftFinished()
Ο μετασχηματισμός Fourier εκτελείται σε ένα ξεχωριστό thread. Η μέθοδος αυτή καλείται όταν ο fft τελιώσει και εμφανίζει τα αποτελέσματα στα διαγράμματα.

FftThread [top]

Η κλάση αυτή επεκτείνει την QThread και είναι υπεύθυνη για την εκτέλεση του μετασχηματισμού Fourier.

Public Μέθοδοι της κλάσης

FftThread(Signal & sig, QObject * parent = 0)

Ο δομητής της κλάσης, στον οποίο περνιέται παραμετρικά το σήμα στο οποίο θέλουμε να γίνει ο μετασχηματισμός Fourier.

virtual ~FftThread()

Signal getImag()

Επιστρέφει το σήμα με τα πραγματικά μέρη του μετασχηματισμού.

Signal getReal()

Επιστρέφει το σήμα με τα πραγματικά μέρη του μετασχηματισμού.

Signal getPow()

Επιστρέφει το σήμα με τις τιμές ισχύος του σήματος.

Protected μέθοδοι της κλάση

void run()

Καλεί την fft για το σήμα που έχει πάρει ο δομητής σαν όρισμα. Αυτή είναι και η χρονοβόρα διαδικασία που πρέπει να εκτελείται σε άλλο νήμα. Για μεγάλα σήματα μπορεί να διαρκέσει και πολύ παραπάνω από 15 λεπτά.

MathPlot [top]

Η MathPlot δημιουργεί ένα διάγραμμα το οποίο διαφέρει/συμπληρώνει το Plot. Αυτό το διάγραμμα δεν έχει απεριόριστο μήκος, δεν υποστηρίζει zoom και δεν μας δίνει τη δυνατότητα να κάνουμε καμία μετατροπή στα σήματα που περιέχει. Αλλά μπορεί να εμφανίσει πολλά σήματα ταυτόχρονα (το ένα πάνω από το άλλο), εμφανίζονται “χάρακες” με αρίθμηση στις δύο διαστάσεις και αναγράφεται η θέση του ποντικιού (με x,y) στο σύστημα συντεταγμένων του διαγράμματος.

Public Μέθοδοι της κλάσης

MathPlot(QWidget * parent=0)

Ο δομητής της κλάσης.

virtual ~MathPlot()

bool isEmpty()

Επιστρέφει true εάν δεν υπάρχει κανένα σήμα στο διάγραμμα.

Protected Μέθοδοι της κλάσης

Μέθοδοι που υπερβαίνουν τις αντίστοιχες του Qt και χρησιμοποιούνται για υποστηρίζει το διάγραμμα drag 'n' drop από διαγράμματα Plot.

void dragLeaveEvent(QDragLeaveEvent *)
void dragMoveEvent(QDragMoveEvent *)
void dragEnterEvent(QDragEnterEvent *)
void dropEvent(QDropEvent *)

Private Μέθοδοι της κλάσης

void createUI()

Αρχικοποιεί και τοποθετεί στην σωστή θέση τα επιμέρους widgets που χρησιμοποιούνται.

void createControl(int id)

Η μέθοδος αυτή δημιουργεί και προσθέτει στο Layout του widget μία γραμμή για κάθε καινούργιο σήμα που φορτώνεται στο διάγραμμα. Από εκεί μπορούμε να ελέγξουμε το αν το σήμα θα ζωγραφίζεται αναλογικά ή όχι καθώς και να αφαιρέσουμε από το διάγραμμα.

void updateColors(int from = 0)

Όταν ένα σήμα αφαιρείται από το διάγραμμα τότε όλα τα άλλα, από αυτό και κάτω θα αλλάξουν χρώμα (ένα χρώμα πίσω από τη σειρά των χρωμάτων). Η μέθοδος αυτή καλείται για να ενημερώσει τα widgets ελέγχου που έχουν το χρώμα του κάθε σήματος.

void updateLayout(int emptyPos = 0)

Όταν κάποιο σήμα αφαιρείται από το διάγραμμα, η θέση του στο Layout μένει κενή. Αυτό έχει σαν αποτέλεσμα το σήμα N να είναι να είναι στη θέση N+1...πράγμα που δεν το θέλουμε. Η μέθοδος αυτή μετατοπίζει όλα τα widgets από τη θέση emptyPos και κάτω, μία θέση πάνω. Καλείται μετά την αφαίρεση κάποιου σήματος.

void updateIds(int from = 0)

Κάθε κουμπί remove στο widget έχει και ένα id που είναι ο αριθμός του σήματος με αρίθμηση από το μηδέν. Καλείται όταν αφαιρείται ένα σήμα και ανανεώνει όλα τα κουμπιά από εκεί και κάτω. Προσοχή: **η removeButton(*but) κάνει segmentation fault ενώ δεν θα έπρεπε(???) και δεν χρησιμοποιείται**.

Public Slots Μέθοδοι της κλάσης

void signalAddedToPanel()

Όταν ένα σήμα φορτωθεί στο MathPlotPanel (είτε από drag 'n' drop είτε από load) τότε αυτό στέλνει ένα signal. Η μέθοδος αυτή είναι συνδεδεμένη με το σήμα αυτό και καλεί την createControl με το κατάλληλο id, για να δημιουργήσει ένα καινούργιο widget ελέγχου.

void removeClicked(int)

Καλείται όταν πατηθεί το κουμπί remove ενός σήματος. Αφαιρεί το σήμα από το διάγραμμα

και ανανεώνει όλα τα widget – layouts που πρέπει να ενημερωθούν.

void checkChanged()

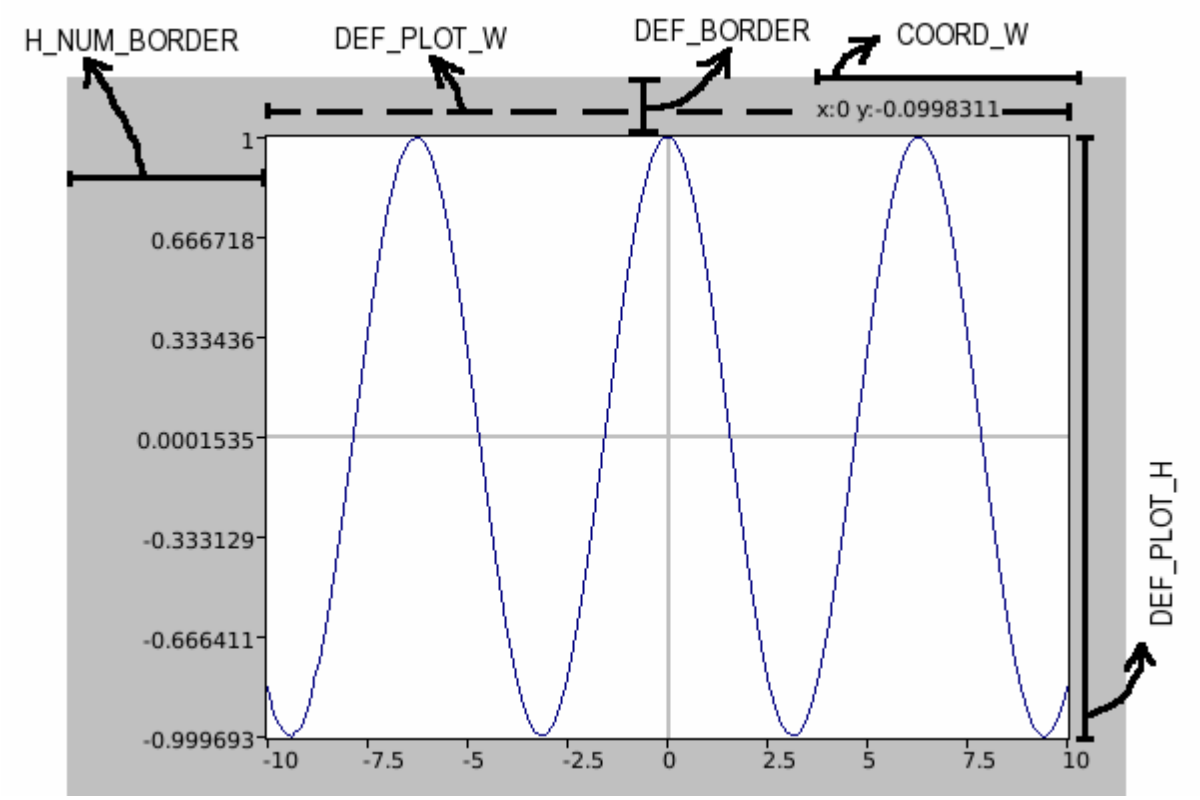
Καλείται όταν αλλάξει κατάσταση το check box ενός σήματος.

MathPlotPanel [top]

Η κλάση αυτή είναι το διάγραμμα που χρησιμοποιεί και διαχειρίζεται η MathPlot. Επεκτείνει την QWidget. Το widget αυτό κρατά χώρο για ζωγραφιστεί μια εικόνα QImage στην οποία υπάρχει σχεδιασμένο η περιοχή με τα σήματα, η αρίθμηση στους άξονες και οι συντεταγμένες του ποντικιού. Η εικόνα αυτή ζωγραφίζεται στο σημείο (0,0) του widget.

Σταθερές που έχουν οριστεί στην κλάση

Όνομα	**Προκαθορισμένη Τιμή**	**Περιγραφή**
DEF_PLOT_W	400	Το μήκος της περιοχής για σχεδίασμα των σημάτων.
DEF_PLOT_H	300	Το ύψος της περιοχής για σχεδίασμα των σημάτων.
H_NUM_BORDER	100	Το περιθώριο που αφήνεται για την σχεδίαση των μονάδων στον άξονα y'y (σε pixel).
DEF_BORDER	30	Το περιθώριο που έχει η περιοχή σχεδίασης από δεξιά, πάνω και κάτω.
DISPLAY_DIG	5	Το πλήθος των δεκαδικών ψηφίων που εμφανίζονται στον χάρακα χ'χ (για να μην υπάρχουν επικαλύψεις).
COORD_W	150	Το μήκος της περιοχής που εμφανίζονται οι συντεταγμένες του ποντικιού.



Εικόνα 17: Το αναλυτικό διάγραμμα (MathPlot)

Public Μέθοδοι της κλάσης

MathPlotPanel(QWidget * parent=0)
: Ο δομητής της κλάσης.

virtual ~MathPlotPanel()

bool isEmpty()
: Επιστρέφει true εάν δεν υπάρχουν φορτωμένα σήματα στο δίαγραμμα.

int getSignalsCount()
: Επιστρέφει το πλήθος των σημάτων που έχουν φορτωθεί στο διάγραμμα.

Signal getSignal(int index)
: Επιστρέφει το σήμα στη θέση index από το διάγραμμα. Το index ξεκινά από το μηδέν.

Qt::GlobalColor getColorFor(int index)
: Επιστρέφει το χρώμα του σήματος που βρίσκεται στη θέση index.

bool isAnalog(int index)
: Επιστρέφει true εάν το σήμα στη θέση index είναι αναλογικό.

void setSignal(int index,Signal &)
: Θέτει το σήμα της θέσης index. Δεν ξαναζωγραφίζεται η εικόνα από μόνη της, πρέπει να

καλεστεί η redrawAll().

void setAnalog(int index,bool a)

Θέτει ένα σήμα σαν αναλογικό. Δεν ξαναζωγραφίζεται η εικόνα από μόνη της, πρέπει να καλεστεί η redrawAll().

void addSignal(Signal &)

Προσθέτει ένα σήμα στο διάγραμμα. Ξαναζωγραφίζει την εικόνα και στέλνει το signal signalAdded().

void removeSignal(int)

Αφαιρεί ένα σήμα από το διάγραμμα. Δεν ξαναζωγραφίζεται η εικόνα από μόνη της, πρέπει να καλεστεί η redrawAll(). Δεν χρειάζεται να σταλθεί κάποιο signal διότι μόνο το widget που διαχειρίζεται το διάγραμμα μπορεί να καλέσει αυτή τη μέθοδο, το ίδιο το MathPlotPanel δεν την καλεί ποτέ.

void redrawAll()

Υπολογίζει τα στατιστικά ξανά και ζωγραφίζει τα πάντα. Καλείται όταν έχει γίνει κάποια αλλαγή και θέλουμε να ανανεωθεί η εικόνα.

Protected Μέθοδοι της κλάσης

Μέθοδοι που υπερβαίνουν τις αντίστοιχες του Qt και χρησιμοποιούνται για υποστηρίζει το διάγραμμα drag 'n' drop από διαγράμματα Plot, να κάνει repaint την εικόνα του διαγράμματος και να διαχειρίζεται το ποντίκι.

void paintEvent(QPaintEvent *);
void dragLeaveEvent(QDragLeaveEvent *)
void dragMoveEvent(QDragMoveEvent *)
void dragEnterEvent(QDragEnterEvent *)
void dropEvent(QDropEvent *)
void contextMenuEvent(QContextMenuEvent *)
void mouseMoveEvent (QMouseEvent * event)

Private Μέθοδοι της κλάσης

void createMenus()

Δημιουργεί το popup μενού και τα actions που αυτό περιλαμβάνει.

void createUI()

Αρχικοποιεί την εικόνα του διαγράμματος και το ίδιο το widget. Άλλα widgets δεν χρησιμοποιούνται.

void initDrawArea()

Ξαναζωγραφίζει την εικόνα του διαγράμματος κενή.

void drawDigitalSignal(int index)

Ζωγραφίζει το σήμα στη θέση σαν αναλογικό.

void drawAnalogSignal(int index)

Ζωγραφίζει το σήμα στη θέση σαν ψηφιακό.

QPointF SignalToImage(const Point &p)
Παίρνει σαν όρισμα ένα σημείο στο σύστημα συντεταγμένων του σήματος και επιστρέφει ένα σημείο στο σύστημα συντεταγμένων της εικόνας.

Point ImageToSignal(const QPointF &p)
Παίρνει σαν όρισμα ένα σημείο στο σύστημα συντεταγμένων της εικόνας και επιστρέφει ένα σημείο στο σύστημα συντεταγμένων του σήματος.

void calcSteps()
Υπολογίζει τα βήματα (XStep και YStep) που χρησιμοποιούνται για την σχεδίαση των σημάτων. Βασίζεται στα στατιστικά όλων των σημάτων που έχουν φορτωθεί στο διάγραμμα.

void addNumbers()
Ζωγραφίζει τους χάρακες και τους αριθμούς πάνω σε αυτούς. Βασίζεται στα στατιστικά όλων των σημάτων που έχουν φορτωθεί στο διάγραμμα.

void drawAxes()
Ζωγραφίζει του άξονες, εάν αυτό είναι απαραίτητο.

void calcMinMaxValues()
Υπολογίζει τα στατιστικά που χρειάζονται από της άλλες μεθόδους. Αυτά είναι το μέγιστο από τα μέγιστα X και Y των σημάτων, και το ελάχιστο από τα ελάχιστα X και Y των σημάτων.

Public Slots της κλάσης

void load()
Ανοίγει έναν διάλογο επιλογής αρχείου και φορτώνει το σήμα από το αρχείο που έχει επιλέξει ο χρήστης.

void saveAsImage()
Αποθηκεύει το διάγραμμα σαν εικόνα PNG.

Signals της κλάσης

void signalAdded()
Το σήμα αυτό στέλνεται όταν φορτώνεται καινούργιο σήμα στο διάγραμμα για να ενημερωθεί το widget που διαχειρίζεται το διάγραμμα.

MainWin [top]

Η κλάση MainWin δημιουργεί το κεντρικό παράθυρο της εφαρμογής, έχει το menu bar καθώς και το workspace μέσα στο οποίο θα ανοίξουν όλα τα υπόλοιπα παράθυρα.

Public Μέθοδοι της κλάσης

MainWin(QWidget* parent = 0, Qt::WindowFlags flags = 0)
Ο δομητής της κλάσης.

virtual ~MainWin()

Private Μέθοδοι της κλάσης

void connectActions()
Εδώ γίνεται η σύνδεση όλων των signals των αντικειμένων με τα slots της κλάσης. Έγινε ξεχωριστή μέθοδος διότι είναι αρκετές οι συνδέσεις που χρειάζεται να γίνουν.

void createMenuBar()
Αρχικοποιεί το menu bar και τα υπο-μενού του. Εδώ ορίζονται και και οι συντομεύσεις του πληκτρολογίου για την εφαρμογή.

void createUI()
Στη μέθοδο αυτή δημιουργείται η γραφική αναπαράσταση του Widget και μπαίνουν τα επιμέρους widgets στην θέση τους.

Private slots της κλάσης

void openNewSingle()
Ανοίγει ένα παράθυρο στο χώρο εργασίας με ένα μόνο διάγραμμα.

void openNewMulti()
Ανοίγει ένα παράθυρο στο χώρο εργασίας με τρία διάγραμμα (MultiPlot).

void openEditor()
Ανοίγει ένα παράθυρο στο χώρο εργασίας με το SignalEditor.

void openNewMath()
Ανοίγει ένα παράθυρο στο χώρο εργασίας με κενό διάγραμμα MathPlot.

void load()
Εμφανίζει τον διάλογο επιλογής αρχείων. Σε αυτόν μπορούν να επιλεχθούν παραπάνω από ένα αρχεία. Για κάθε ένα αρχείο ανοίγει και ένα διάγραμμα με φορτωμένο το σήμα του.

Signal [top]

Η Signal είναι η κλάση η οποία κρατά και διαχειρίζεται τα σημεία ενός σήματος. Είναι υπεύθυνη να κράτα τα στατιστικά του σήματος ενημερωμένα, να κάνει πράξεις μεταξύ σημάτων ή σήματος και σημείου καθώς και μετασχηματισμούς στο ίδιο σήμα. Όλοι οι μαθηματικοί υπολογισμοί, εκτός αποό τον μετασχηματισμό Fourier, είναι ορισμένοι σε αυτήν την κλάση. Τέλος σε αυτήν ορίζονται οι υπερφορτωμένοι τελεστές για εγγραφή και ανάγνωση σήματος από αρχείο.

Public Μέθοδοι της κλάσης

Signal()

Δομητής της τάξης που δημιουργεί ένα σήμα χωρίς κανένα σημείο.

Signal(const char * fname)

Δομητής που δημιουργεί ένα σήμα και διαβάζει τα σημεία του από αρχείο. Εάν δεν ανοίξει το αρχείο (για οποιοδήποτε λόγο) το σήμα θα είναι χωρίς σημεία.

Signal(const Signal & sig)

Δομητής που αντιγράφει το σήμα που πέρνει σαν όρισμα.

Signal(vector<Point> pts)

Δομητής που δημιουργεί ένα σήμα και παίρνει τα σημεία του από τον vector που παίρνει σαν όρισμα.

~Signal()

void load(const char * fname)

Διαβάζει σημεία από το αρχείο fname και δημιουργεί καινούριο σήμα. Εάν το σήμα έχει ήδη σημεία τότε αυτά διαγράφονται. Τέλος θέτει τιμή στη μεταβλητή sourceFile και υπολογίζει όλα τα στατιστικά του καινούριου σήματος. Εάν δεν ανοίξει το αρχείο τότε το καινούριο σήμα είναι άδειο.

void save(const char * fname)

Αποθηκεύει τα σημεία του σήματος στο αρχείο fname και αλλάζει την τιμή της μεταβλητής sourceFile σε fname.

uint size()

Επιστρέφει το πλήθος των σημείων του σήματος.

bool isEmpty()

Επιστρέφει true εάν το σήμα είναι άδειο.

string toString() const

Επιστρέφει μια λίστα με τα σημεία του σήματος σε std::string. Χρησιμοποιείται για έλεγχο αποτελεσμάτων.

string getFileName()

Επιστρέφει το όνομα του αρχείου από το οποίο φορτώθηκε το σήμα.

double getMinXdist()

Επιστρέφει την μικρότερη από τις αποστάσεις μεταξύ των δειγμάτων του σήματος.

double getMinYdist()

Επιστρέφει την μικρότερη από τις αποστάσεις μεταξύ των τιμών του σήματος.

double getMaxY()

Επιστρέφει την μέγιστη τιμή του σήματος.

double getMaxX()
Επιστρέφει το μέγιστο δείγμα του σήματος.

double getMinY()
Επιστρέφει την μικρότερη τιμή του σήματος.

double getMinX()
Επιστρέφει το μικρότερο δείγμα του σήματος.

double getMean()
Επιστρέφει την μέση τιμή του σήματος.

double getVariance()
Επιστρέφει την διακύμανση του σήματος.

double getDispersion()
Επιστρέφει την διασπορά του σήματος.

double getStdDeviation()
Επιστρέφει την τυπική απόκλιση του σήματος.

double getAvgDeviation()
Επιστρέφει την μέση απόκλιση του σήματος.

double getSkewness()
Επιστρέφει την λοξότητα του σήματος.

double getKurtosis()
Επιστρέφει την κυρτότητα του σήματος.

void moveHor(double pts)
Μετακινεί το σήμα στον οριζόντιο άξονα κατά pts μονάδες. Έπειτα υπολογίζει ξανά τα στατιστικά του σήματος.

void moveVer(double pts)
Μετακινεί το σήμα στον κάθετο άξονα κατά pts μονάδες. Έπειτα υπολογίζει ξανά τα στατιστικά του σήματος.

void addNoise(double max)
Προσθέτει max%, της μέγιστης διαφοράς πλάτους (maxY-minY), τυχαίο θόρυβο σε κάθε σημείο του σήματος.

Signal getSampledBy(int step)
Επιστρέφει ένα καινούριο σήμα με σημεία τα αποτελέσματα της δειγματοληψίας στο σήμα με βήμα step.

Signal getQuantedWith(int levels)
Επιστρέφει ένα καινούριο σήμα με σημεία τα αποτελέσματα του κβαντισμού του σήματος με levels διακριτές στάθμες.

Point & operator[](uint index)

Υπερφορτωμένος τελεστής ώστε να μπορεί το σήμα να χρησιμοποιηθεί σαν πίνακας. Επιστρέφει το σημείο της θέσης index στο σήμα.

Signal& operator+=(const Point&)

Προσθέτει ένα σημείο στο τέλος του σήματος.

Signal operator+(const Signal&)

Κάνει πρόσθεση μεταξή δύο σημάτων. Τα δύο σήματα πρέπει να έχουν το ίδιο σταθερό βήμα μεταξύ των σημείων τους, αλλιώς δεν έχει νόημα η πράξη. Στις περιοχές που ορίζεται μόνο ένα από τα δύο σήματα το αποτέλεσμα είναι το ίδιο το σήμα ενώ στις περιοχές που ορίζονται και τα δύο, το αποτέλεσμα για κάθε σημείο είναι ένα σημείο με το ίδιο x και y το άθροισμα του πλάτους των δύο σημείων των σημάτων.

Signal operator*(const Signal&)

Κάνει πολλαπλασιασμό δύο σημάτων. Τα δύο σήματα πρέπει να έχουν το ίδιο σταθερό βήμα μεταξύ των σημείων τους, αλλιώς δεν έχει νόημα η πράξη. Στις περιοχές που ορίζεται μόνο ένα από τα δύο σήματα το αποτέλεσμα είναι 0 ενώ στις περιοχές που ορίζονται και τα δύο, το αποτέλεσμα για κάθε σημείο είναι ένα σημείο με το ίδιο x και y το γινόμενο του πλάτους των δύο σημείων των σημάτων.

friend ofstream & operator<<(ofstream & out, const Signal &sig)

Υπερφορτωμένος τελεστής για εγγραφή του σήματος σε αρχείο.

friend ifstream & operator>>(ifstream & in, Signal &sig)

Υπερφορτωμένος τελεστής για ανάγνωση του σήματος από αρχείο.

friend ostream & operator<<(ostream & out, const Signal &sig)

Υπερφορτωμένος τελεστής για εμφάνιση του σήματος στην οθόνη. Χρησιμοποιείται μόνο για να γίνονται έλεγχοι αποτελεσμάτων.

Signal convolution(const Signal&)

Κάνει την συνέληξη του σήματος με το σήμα που παίρνει σαν όρισμα και επιτρέφει το αποτέλεσμα σαν ένα νεο σήμα (χωρίς να αλλάζει κανένα από τα δύο πρώτα σήματα).

void fft(Signal &real, Signal &imag, Signal &power)

Εφαρμόζει τον μετασχηματισμό Fourier στο σήμα και επιστρέφει ένα σήμα με τα πραγματικά μέρη, ένα με τα φανταστικά μέρη (μιγαδικοί) και τις τιμές ισχύος. Η μέθοδος αυτή δεν υλοποιείται στην κλάση Signal αλλά καλεί την fft από το αρχείο fft.cpp. Στην ουσία η μέθοδος αυτή κάνει τις μετατροπές από Signal σε πίνακες (double *) ώστε να καλέσει την fft, και το αντίστροφο για να επιστρέψει Signals. Προσοχή, το πλήθος των δειγμάτων στο σήμα πρέπει να είναι δύναμη του δύο. Εάν δεν είναι τότε στον μετασχηματισμό περνάμε σαν όρισμα ένα πίνακα με όσο περισσότερα σημεία γίνεται αλλά το πλήθος τους να είναι δύναμη του 2. Έτσι τα σήματα που έχουμε σαν αποτέλεσμα είναι λογικό να έχουν μικρότερο μέγεθος.

Private Μέθοδοι της κλάσης

void calcMinMaxMean()

Υπολογίζει τα μέγιστα, τα ελάχιστα και την μέση τιμή του σήματος.

void calcVariance()
Υπολογίζει τη διακύμανση του σήματος.

void calcDispersion()
Υπολογίζει τη διασπορά του σήματος.

void calcStdDeviation()
Υπολογίζει την τυπική απόκλιση του σήματος.

void calcAvgDeviation()
Υπολογίζει την μέση απόλιση του σήματος.

void calcSkewness()
Υπολογίζει την λοξότητα του σήματος.

void calcKurtosis()
Υπολογίζει την κυρτότητα του σήματος.

void calcStats()
Υπολογίζει όλα τα στατιστικα του σήματος, με την σειρά που πρέπει. Η μέθοδος αυτή είναι private διότι η κλάση Signal είναι υπεύθυνη να κρατά ενημερωμένα τα στατιστικά του σήματος και να τα ξανα υπολογίζει όταν αυτό αλλάζει.

uint sizeToPowOfTow(uint size)
Παίρνει σαν όρισμα έναν αριθμό και επιστρέφει τον πρώτο μικρότερο αριθμό από αυτόν, που είναι δύναμη του δύο. Θα μπορούσε να παίρνει τον αριθμό από την μεταβλητή size του σήματος αλλά υλοποιήθηκε έτσι για να μπορεί να μετατρέψει οποιονδήποτε αριθμό και να γίνει static άμα χρειαστεί.

Point [top]

Η κλάση Point έχει δύο double (x, y) για την αναπαράσταση ενός σημείου. Τα double αυτά είναι public για να μπορεί να χρησιμοποιήται σαν struck, αλλά να έχει και operators για ανάγνωση και εγγραφή σε αρχείο.

Public Μέθοδοι της κλάσης

Point()
Ο default δομητής επιστρέφει ένα σημείο (0,0).

Point(double x, double y)
Ο κανονικός δομητής της κλάσης.

Point(const Point & pt)

~Point()

void operator=(const Point & pt)
Τελεστής ανάθεσης.

bool operator==(const Point &pt)
Τελεστής ελέγχου ισότητας.

bool operator!=(const Point &pt)
Τελεστής ελέγχου ισότητας.

friend ofstream & operator<<(ofstream & out, const Point &pt)
Τελεστής για εγγραφή Point σε αρχείο.

friend ifstream & operator>>(ifstream & in, Point &pt)
Τελεστής για ανάγνωση Point από αρχείο.

string toString()
Επιστρέφει το σημείο στη μορφή "χ ψ" σε string. Χρησιμοποιείται για έλεγχο αποτελεσμάτων σε κονσόλα.

4.3 Άλλα αρχεία κώδικα [περιεχόμενα]

main.cpp

Η main όλης της εφαρμογής. Αρχικοποιεί τα resources του προγράμματος, δημιουργεί και εκτελεί ένα αντικείμενο QApplication το οποίο έχει για MainWindow το δικό μας MainWin. Εδώ καθορίζεται το στύλ και οι γραμματοσειρές που θα χρησιμοποιεί η εφαρμογή.

fft (include/fft.h και src/fft.cpp)

Ο μετασχηματισμός Fourier με τις βοηθητικές μεθόδους και το header του.

fparser28/custom_functions (.h και .cpp)

Στα αρχεία αυτά δηλώνονται και υλοποιούνται κάποιες συναρτήσεις οι οποίες όπου χρειάζονται, προστίθενται στον fparser και μπορούν να χρησιμοποιηθούν από τον SignalEditor.

4.4 Βασικά αρχεία του Project [περιεχόμενα]

ergasia.pro

Σε αυτό το αρχείο οργανώνεται το Project. Καθορίζονται τα αρχεία κώδικα, τα αρχεία δηλώσεων καθώς και ρυθμίσεις για το που θα τοποθετηθούν τα ενδιάμεσα αρχεία και τα εκτελέσιμα μετά το compile. Το αρχείο αυτό μπορεί να παραχθεί αυτόματα από την qmake (-project) αλλά αυτό δεν συστήνεται. Από την qmake διαβάζεται το αρχείο αυτό και παράγεται το κατάλληλο Makefile.

resources/resources.qrc

Είναι ένα αρχείο με σύνταξη xml το οποίο καθορίζει τις εικόνες και τα κείμενα της εφαρμογής. Διαβάζεται από έναν pre-processor και παράγεται το ένα αρχείο .cpp με τις εικόνες ενσωματωμένες σε πίνακες. Το τελευταίο χρησιμοποιείται από τον compiler και την εφαρμογή. Με αυτόν τον τρόπο δεν εξαρτάται η εφαρμογή από την απόλυτη διαδρομή των εικόνων αλλά τις ενσωματώνει στο εκτελέσιμο που παράγεται, για αυτό και δεν χρειάζεται εγκατάσταση της εφαρμογής με κάποιον installer.

5. Βιβλιογραφία [περιεχόμενα]

- Press, Flannery, Teukolsky, Vetterling “Numerical Recipes: The art of scientific computing” Cambridge, 1986.
- Αθανάσιος Ι. Μάργαρης: «Σημειώσεις Ψηφιακής Επεξεργασίας Σήματος και Εικόνας», Τμήμα Πληροφορικής, ΤΕΙ Θεσσαλονίκης
- Το official site για τον fparser: http://warp.povusers.org/FunctionParser
- Qt Assistant by Trolltech

Ανάπτυξη Εκπαιδευτικού Λογισμικού Ψηφιακής Επεξεργασίας Σήματος

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Αντρέας Μποντόζογλου

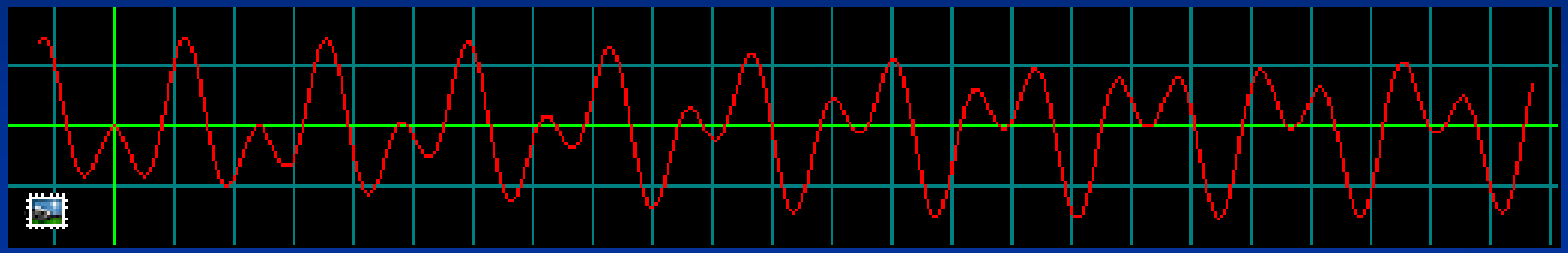
Επιβλέπων καθηγητής

Αθανάσιος Ι. Μάργαρης

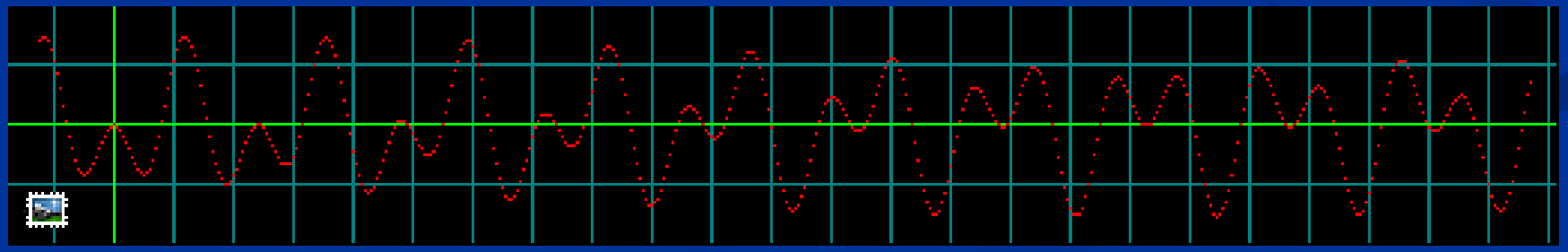
Βασικές Έννοιες Σημάτων

- Αναλογικά και ψηφιακά σήματα
- Σήματα διακριτού χρόνου
- Θεμελιώδεις πράξεις με διακριτά σήματα
- Στατιστικές ιδιότητες μονοδιάστατων διακριτών σημάτων
- Μετατροπή αναλογικού σήματος σε ψηφιακό
- Συνέλιξη (convolution)
- Μετασχηματισμός Fourier

Αναλογικά – Ψηφιακά Σήματα



Τα ψηφιακά σήματα είναι συναρτήσεις διακριτής μεταβλητής και παίρνουν τιμές οι οποίες ανήκουν σε μια ομάδα από διακριτές στάθμες



Σήματα διακριτού χρόνου

Ορίζονται ως συναρτήσεις μίας ανεξάρτητης ακέραιας μεταβλητής και παίρνουν οποιεσδήποτε τιμές.

- **Συναρτησιακή αναπαράσταση:**

$$x(n)=\begin{cases}5, & n=-2,-1\\ 3, & n=0\\ -2, & n=1,2,3\end{cases}$$

- **Ακολουθιακή αναπαράσταση:** x(n) = {...0, 3, -1, -5, 5, 3 , 0, 0, -2, 8}

↑

- **Αναπαράσταση σε μορφή πίνακα:**

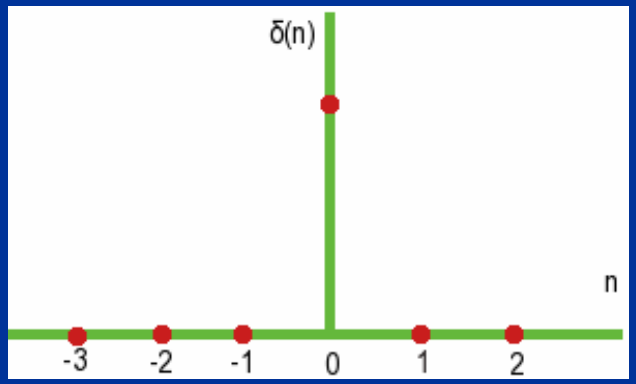
n	…	-3	-2	-1	0	1	…
x(n)	…	15	-9	-2	2	-6	…

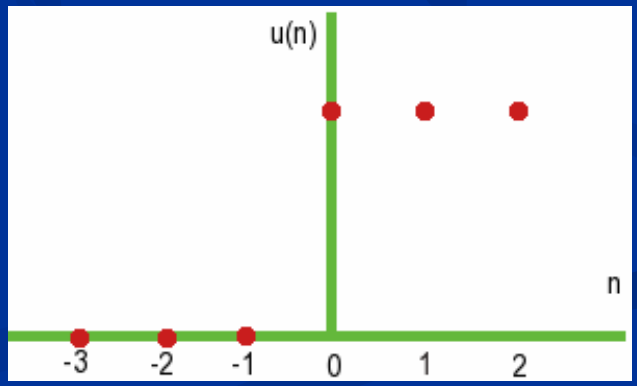
Συνάρτηση μοναδιαίας διακριτής ώσης:

$$\delta(n)=\begin{cases}1, & n=0\\ 0, & \text{οπουδήποτεαλλού}\end{cases} = u(n) - u(n-1)$$

Μοναδιαίας συνάρτησης βήματος:

$$u(n)=\begin{cases}1, & n\geq 0\\ 0, & \text{οπουδήποτεαλλού}\end{cases} = \sum_{k=-\infty}^{n} \delta(k)$$





Θεμελιώδεις πράξεις με διακριτά σήματα.

- Μεταβολή της ανεξάρτητης μεταβλητής
- Κλιμάκωση πλάτους: $x(n) = Ax(n), \quad -\infty < n < \infty$
- Πρόσθεση σημάτων: $y(n) = x1(n) + x2(n),$
- Πολλαπλασιασμός σημάτων: $y(n) = x1(n) * x2(n).$

Στατιστικές ιδιότητες διακριτών σημάτων

- Μέση τιμή
- Διασπορά
  - Μεταβλητότητα
  - Τυπική απόκλιση
  - Μέση απόλυτη απόκλιση
- Η τρίτη ροπή (skewness)
- Η τέταρτη ροπή (kurtosis)

Μετατροπή αναλογικού σήματος σε ψηφιακό

- **Δειγματοληψία** (Sampling)

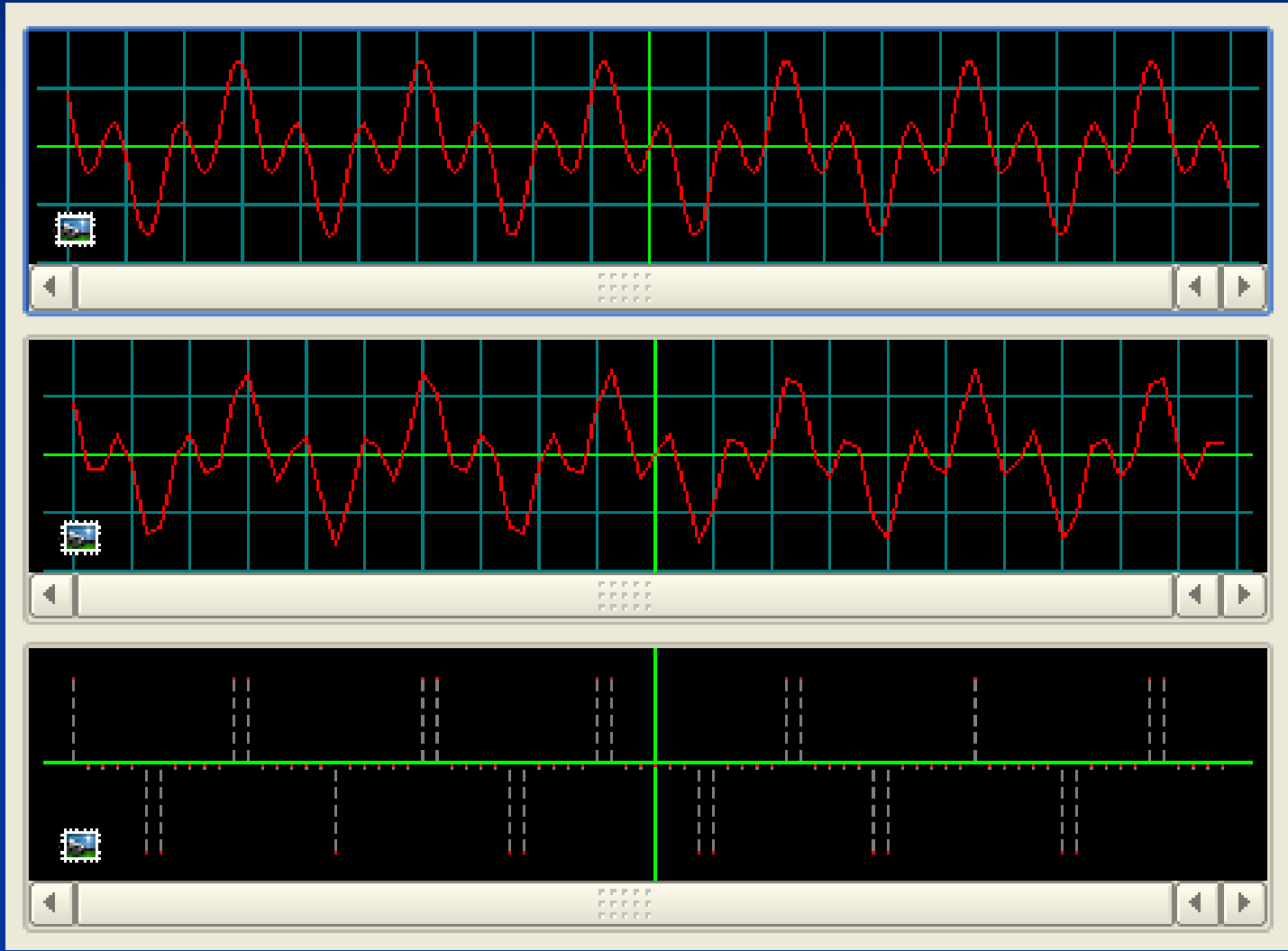
  Γίνεται η μετατροπή του αναλογικού σήματος σε σήμα διακριτού χρόνου.

- **Κβαντισμός** (Quantization)

  Οι τιμές του σήματος προσαρμόζονται έτσι ώστε οι νέες τιμές που θα προκύψουν να ανήκουν σε ένα σύνολο προκαθορισμένων τιμών (στάθμες).

- **Κωδικοποίηση**

Μετατροπή αναλογικού σήματος σε ψηφιακό



Συνέλιξη

Ορίζεται από τους τύπους:

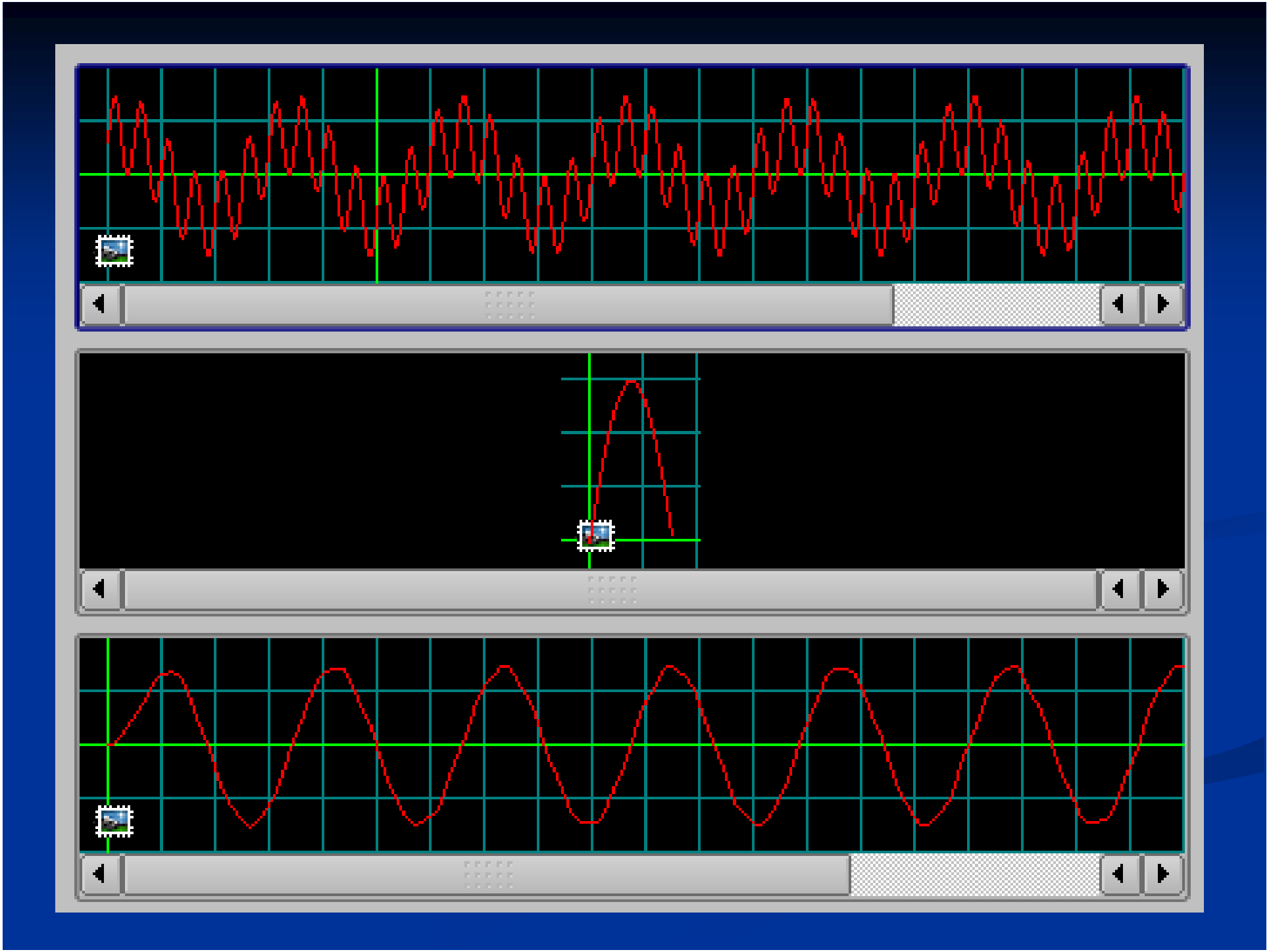
$$g * h = \int_{-\infty}^{\infty} g(\tau) h(t-\tau) d\tau$$

$$(r * s)_j = \sum_{k=-M/2+1}^{M/2} s_{j-k} r_k$$

για τα αναλογικά και για τα ψηφιακά σήματα αντίστοιχα.

Τι είναι η Συνέλιξη

- Η μία συνάρτηση, είναι ένα σήμα που συνεχίζεται απεριόριστα στον χρόνο.
- Η άλλη ονομάζεται κρουστική απόκριση, και έχει συνήθως σχήμα κορυφής/καμπάνας.
- Το αποτέλεσμα της συνέλιξης είναι να εξομαλύνεται το σήμα της πρώτης σύμφωνα με την μορφή της δεύτερης.



Μετασχηματισμός Fourier

Κάθε σήμα μπορεί να γραφεί ως άθροισμα ημιτονοειδών συναρτήσεων.

- Αν ξέρουμε τον τύπο h(t) που περιγράφει μαθηματικά ένα σήμα, τότε μπορούμε να βρούμε το πλάτος της συχνότητας f που αντιστοιχεί σε κάθε επιμέρους ημιτονοειδή συνάρτηση, με την παρακάτω σχέση:

$$H(f) = \int_{-\infty}^{\infty} h(t) e^{2\pi i f t} dt$$

Αντίστροφος Μετασχηματισμός Fourier

- Αντίστροφα, αν γνωρίζουμε το πλάτος κάθε συχνότητας του σήματος μπορούμε να ανακατασκευάσουμε το σήμα σύμφωνα με τον αντίστροφο μετασχηματισμό Fourier που περιγράφεται από τη σχέση:

$$h(t) = \int_{-\infty}^{\infty} H(f) e^{-2\pi i f t} df$$

Διακριτός Μετασχηματιοσμός Fourier

- Ο διακριτός μετασχηματισμός βρίσκεται προσεγγίζοντας το αντίστοιχο ολοκλήρωμα ως άθροισμα και ορίζεται από τη σχέση:

$$H_n = \sum_{k=0}^{N-1} h_k e^{2\pi i k n / N}$$

- Ο διακριτός αντίστροφος μετασχηματισμός Fourier υπολογίζει τις τιμές h_k αν γνωρίζουμε τις H_n και δίνεται από τη σχέση:

$$h_k = \frac{1}{N} \sum_{n=0}^{N-1} H_n e^{-2\pi i k n / N}$$

Γρήγορος μετασχηματισμός Fourier (FFT)

- Ο αλγόριθμος αυτός έγινε γνωστός στα μέσα της δεκαετίας του 1960 από την εργασία των Cooley και Tukey

- Βασίζεται σε ένα λήμμα που διατυπώθηκε από τους Danielson και Lanczos το 1942 :

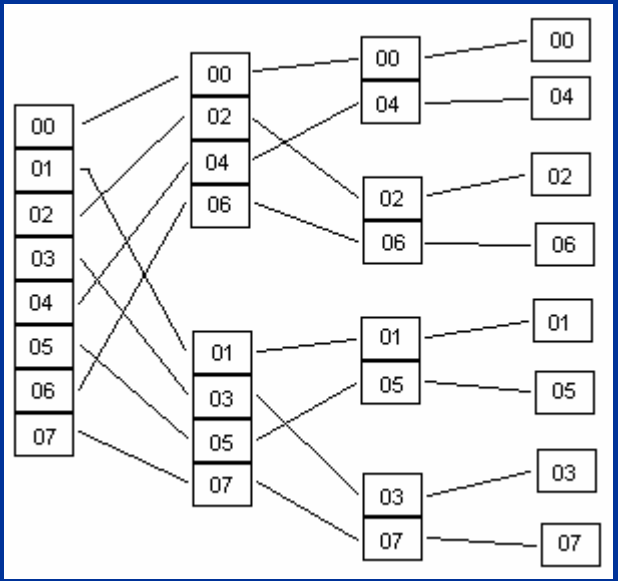
Ο μετασχηματισμός Fourier μιας ακολουθίας τιμών με μέγεθος N μπορεί να γραφεί ως το άθροισμα δύο διακριτών μετασχηματισμών, καθένας από τους οποίους έχει μήκος N/2.

Πολυπλοκότητα Fourier: N^2

Πολυπλοκότητα FFT: $N\log_2 N$

Ο FFT στην πράξη

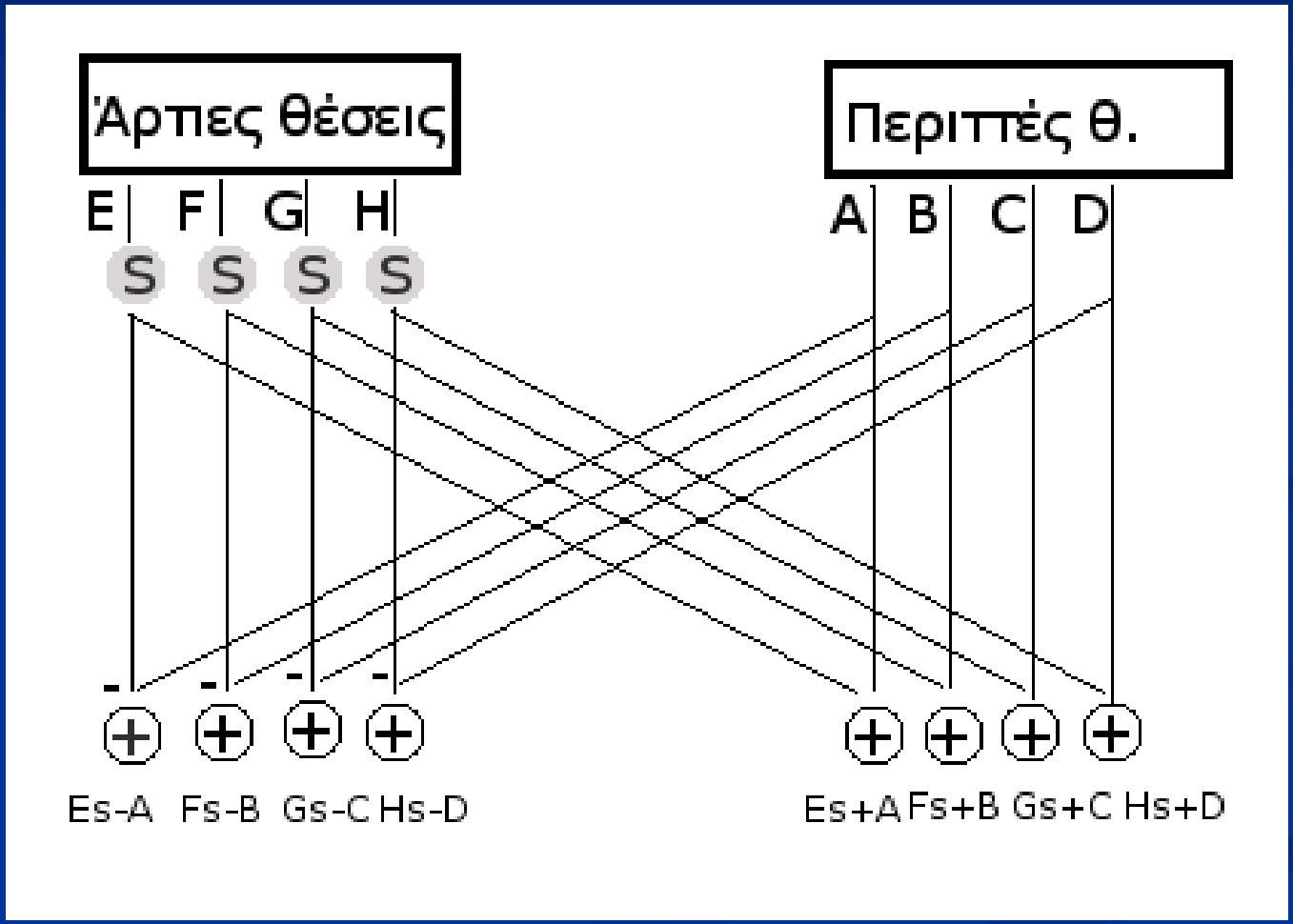
Από τους δύο αυτούς μετασχηματισμούς ο ένας υπολογίζεται από τα δείγματα που βρίσκονται στις άρτιες θέσεις της ακολουθίας εισόδου ενώ ο άλλος από τις περιττές :



Τιμή Δείκτη	Δυαδική Τιμή	Κατοπτρική Τιμή	Δεκαδική Τιμή
00	000	000	00
01	001	100	04
02	010	010	02
03	011	110	06
04	100	001	01
05	101	101	05
06	110	011	03
07	111	111	07

Σωστή θέση των δειγμάτων Αλγοριθμικά

Επανασύνθεση της Αρχική Ακολουθίας



Λειτουργίες Εφαρμογής

- Το απλό διάγραμμα

  Ιδιότητες σήματος/διαγράμματος

- Πολλαπλά διαγράμματα

  Πράξεις μεταξύ σημάτων

- Το αναλυτικό διάγραμμα

  Σύγκριση/Λεπτομέρειες σημάτων

- Επεξεργαστής σημάτων

  Παραγωγή διαγράμματος από συναρτήσεις

Διάγραμμα Οντοτήτων – Κλάσεων