ΕΡΓΑΣΤΗΡΙΟ ΨΗΦΙΑΚΗΣ ΕΠΕΞΕΡΓΑΣΙΑΣ ΣΗΜΑΤΩΝ

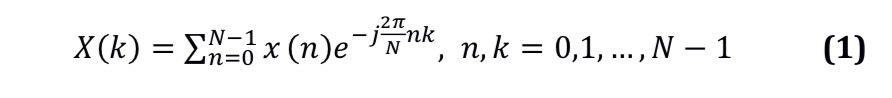
## Εργαστήριο 8

## Υλοποίηση φασματικού αναλυτή

|  |
| --- |
| Ομάδα 17 14-1-2019 |
| Ασημακόπουλος Κωνσταντίνος 1046966 |
| Λουκαρέας Παύλος 1046970 |

**Περίληψη**

Σε αυτή την εργαστηριακή άσκηση υλοποιήσαμε έναν φασματικό αναλυτή υλοποιώντας τον αλγόριθμο FFT. O FFT είναι ένας αποδοτικός αλγόριθμος υπολογισμού του DFT N σημείων, ο οποίος μπορεί να θεωρηθεί ως δειγματοληψία του μετασχηματισμού Fourier διακριτού χρόνου και δίνεται από την παρακάτω εξίσωση:



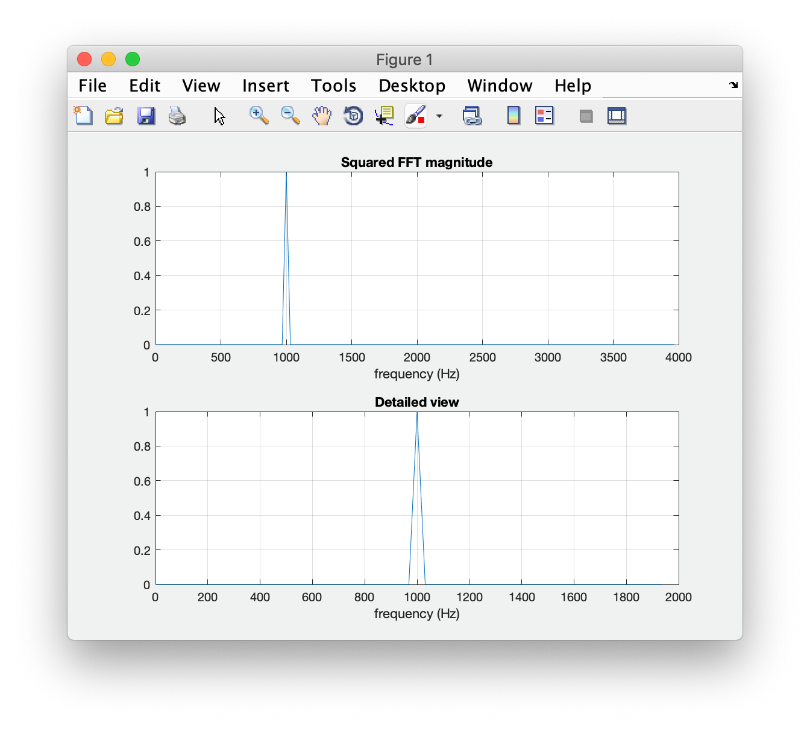
O FFT υπολογίζεται πάντα για ένα μπλοκ δεδομένων. Σε αυτή την άσκηση θεωρούμε ως μήκος μπλοκ 64 δείγματα. Η απότομη αποκοπή του σήματος στα άκρα του μπλοκ παραμορφώνει το φάσμα, γι’ αυτό το λόγο θα χρησιμοποιήσουμε παραθύρωση Hamming.

**Επίδραση της συμπλήρωσης με μηδενικά**

Όταν εκτελούμε FFT N σημείων ο αριθμός των συντελεστών που δίνει η συνάρτηση fft() της MATLAB είναι επίσης Ν δείγματα. Αν και τα Ν δείγματα περιέχουν όλη την ισοδύναμη πληροφορία, είναι χρήσιμο να παίρνουμε δείγματα του DTFT σε περισσότερα σημεία. Αυτό γίνεται προσθέτοντας Μ – Ν μηδενικά στο σήμα πριν τον FFT.

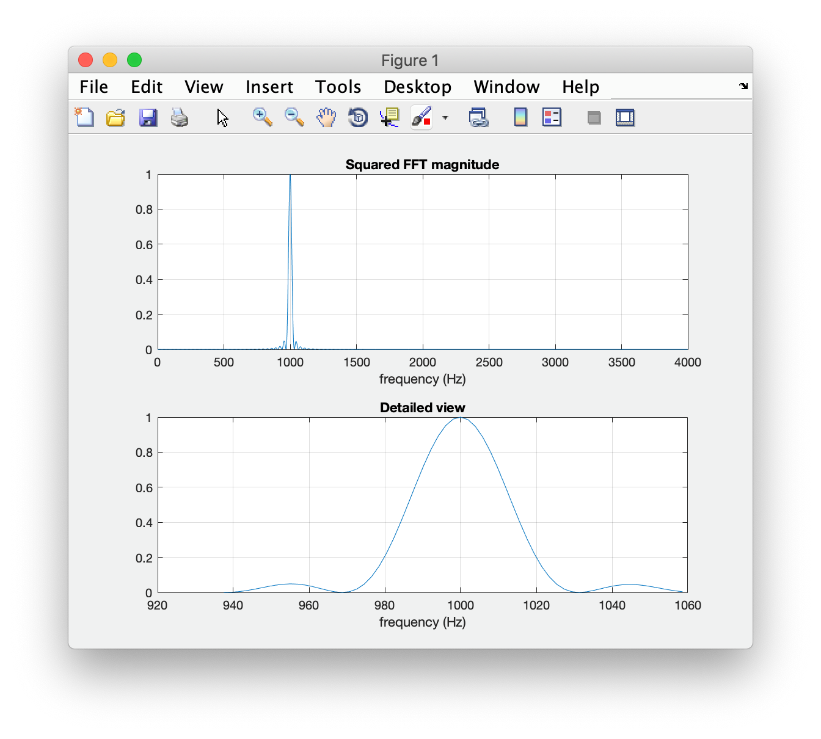
**Άσκηση 8.1**

1. Σε αυτή την άσκηση μας δίνεται ο παρακάτω κώδικας MATLAB ο οποίος υπολογίζει και σχεδιάζει το μέτρο του FFT 256 δειγμάτων από ημιτονοειδές σήμα 1 kHz με ρυθμό δειγματοληψίας 8 kHz. H έξοδος του κώδικα είναι η παρακάτω:



Να σημειωθεί ότι στον παραπάνω κώδικα στη γραμμή 2 ορίζεται το μήκος του διανύσματος δεδομένων μετά την προσθήκη μηδενικών. Αφού Μ = Ν δηλώνει ότι δεν κάνουμε συμπλήρωση μηδενικών.

1. Σε αυτό το ερώτημα χρησιμοποιούμε Μ = 4096 οπότε τα μηδενικά που συμπληρώνουμε είναι πολλά περισσότερα από τη χρήσιμη πληροφορία. Αυτό κάνει τη μετάβαση στα άκρα του φάσματος πολύ πιο ομαλή.

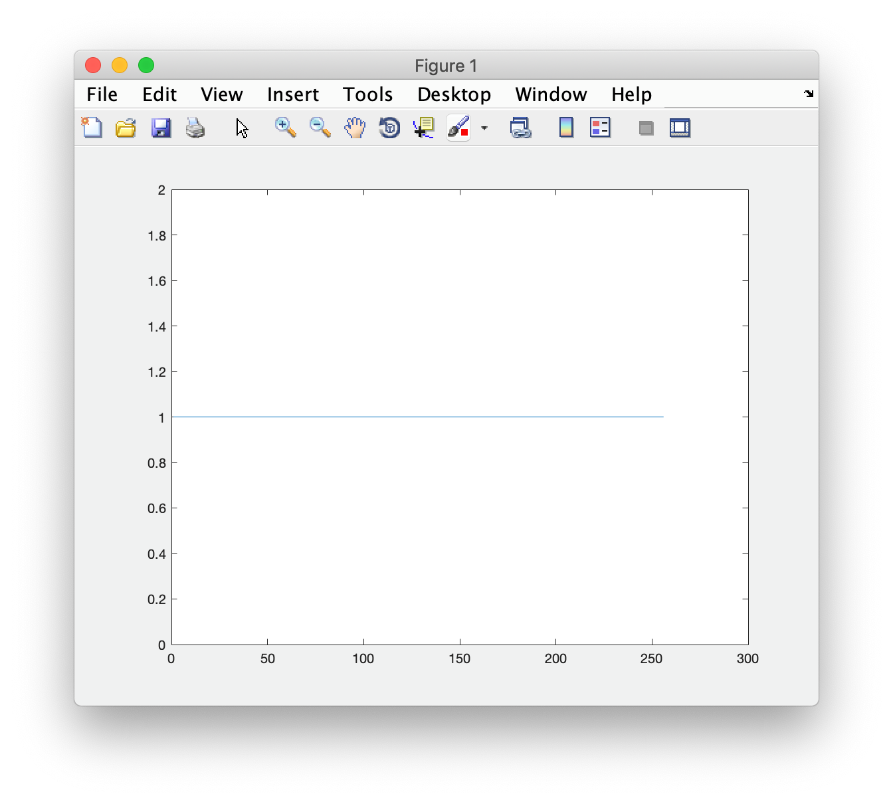


**Επίδραση της παραθύρωσης δεδομένων**

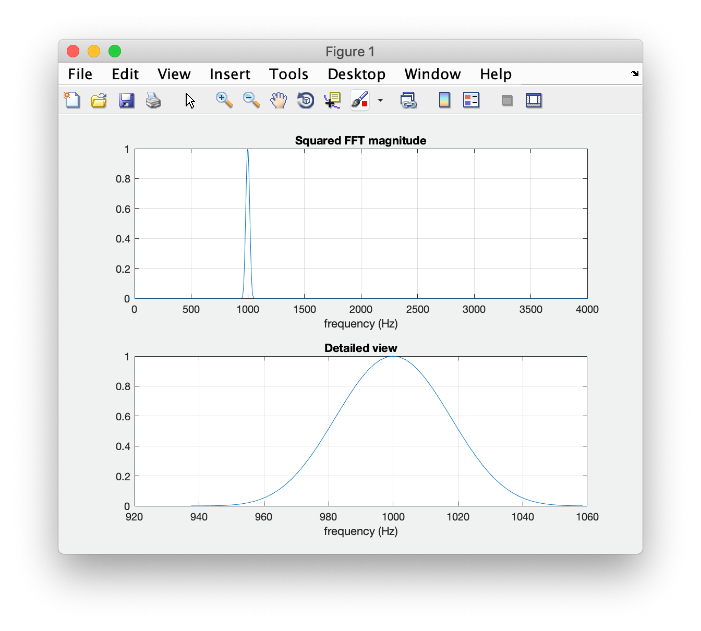
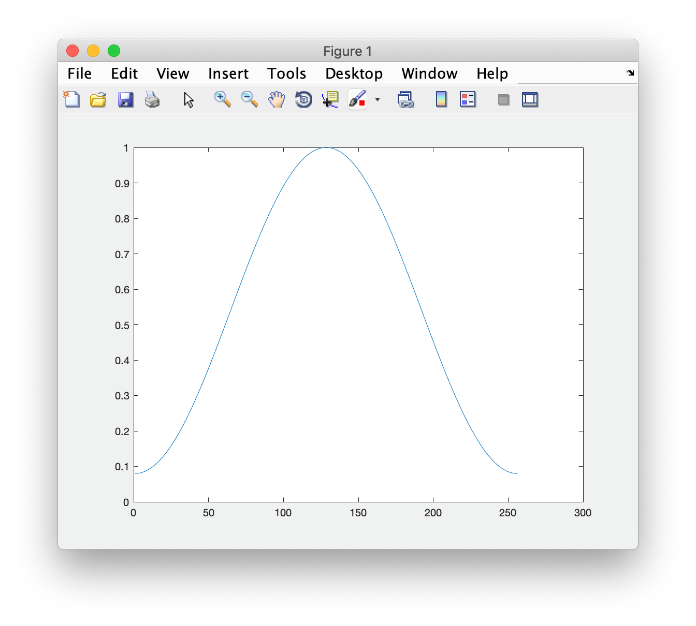
Ενώ η συμπλήρωση με μηδενικά παράγει φάσμα υψηλότερης ανάλυσης, το μέγεθος του δείγματος μεγαλώνει, με αποτέλεσμα να αυξάνεται η υπολογιστική πολυπλοκότητα του FFT. Για το λόγο αυτό μελετάμε την επίδραση της παραθύρωσης δεδομένων στο φάσμα.

**Άσκηση 8.2**

Σε αυτή την άσκηση μελετάμε διαφορετικές τεχνικές παραθύρωσης. Αρχικά, παρατηρούμε το σχήμα του παραθύρου boxcar το οποίο δίνεται παρακάτω.



Στην προηγούμενη άσκηση στο φάσμα που αναλύσαμε χρησιμοποιούσαμε αυτή την τεχνική παραθύρωσης. Σε αυτή την άσκηση θα αναλύσουμε το φάσμα του σήματος με παράθυρο Hamming. To παράθυρο καθώς και το φάσμα φαίνονται παρακάτω:



Όπως βλέπουμε, η χρήση παραθύρου Hamming εξομαλύνει τις ασυνέχειες στα άκρα του παραθύρου. Τα ακραία σημεία πολλαπλασιάζονται με πολύ μικρές τιμές εξασθενώντας έτσι τις ασυνέχειες που εμφανίζονται. Σαν αποτέλεσμα εξαφανίζονται οι παράπλευροι λοβοί και διευρύνεται ο κεντρικός. Αντιθέτως, το boxcar παράθυρο πολλαπλασιάζει όλες τις τιμές με το ένα, αφήνοντας τους λοβούς. Η επιλογή παραθύρου διαμορφώνει τελικά τη σχέση εύρους κεντρικού και παράπλευρων λοβών. Η σχέση αυτή διαμορφώνει αντίστοιχα τη ζώνη μετάβασης και τις διακυμάνσεις των ψηφιακών φίλτρων.

**Άσκηση 8.3**

Σε αυτό το ερώτημα μας ζητήθηκε να υλοποιήσουμε στην πλακέτα έναν φασματικό αναλυτή ο οποίος θα δέχεται ηχητικό σήμα από το μικρόφωνο και θα εμφανίζει στην οθόνη το φάσμα του σήματος αυτού για ένα δεδομένο μήκος δειγμάτων. Στην άσκηση μας ζητήθηκε να υλοποιήσουμε τον αλγόριθμο FFT 64 σημείων χρησιμοποιώντας έτοιμες βιβλιοθήκες κώδικα τις οποίες παρείχε ο κατασκευαστής της πλακέτας. Για το λόγο αυτό κάθε εικόνα που θα πάρουμε θα αφορά μόνο τα 64αυτά δείγματα την στιγμή πριν κλείσουμε το πρόγραμμα σε debug mode.

Σε πρώτη φάση αλλάξαμε τα properties του κώδικα ώστε να μπορέσουμε να χρησιμοποιήσουμε μέσα από τα αρχεία κεφαλίδων και το αρχείο .lib στο οποίο είναι ορισμένες οι συναρτήσεις DSP\_radix2(), bitrev\_index(), DSP\_bitrev\_cplx() οι οποίες θα χρησιμεύσουν στον υπολογισμό του φάσματος από τα δείγματα. Στη συνέχεια ορίσαμε global μεταβλητές, ώστε να είναι εύκολη η μεταφορά των δεδομένων μεταξύ των συναρτήσεων. Αρχικά, με τη χρήση των διακοπών συλλέγουμε τα 64 δείγματα και τα βάζουμε σε πίνακα 128 θέσεων, όπου κάθε δεύτερη θέση θα έχει 0, συμβολίζοντας το φανταστικό μέρος του σήματος. Όταν ο πίνακας συμπληρωθεί καλούμε τη ρουτίνα του FFT για τον υπολογισμό του φάσματος, ενοποιώντας τα 16-bit πραγματικά και φανταστικά δεδομένα σε 32-bit αριθμό. Παρακάτω βλέπουμε τον τρόπο υλοποίησης της ρουτίνας διακοπής.

interrupt void serial\_port\_rcv\_isr() {

    int data;

    data = input\_leftright\_sample();

    if (n != 128) {

        signal[n] =  (short)data;

        signal[n + 1] = 0;

    } else {

        n = 0;

        signal[n] = (short)data;

        signal[n + 1] = 0;

        calculate\_coeff(coeff);

        FFT(signal, coeff);

        abs\_val((int \* )signal, amplitude);

    }

    n += 2;

    output\_leftright\_sample(data);

}

Η πρώτη συνάρτηση που καλείται όταν γεμίσει ο πίνακας είναι η FFT() και δέχεται ως ορίσματα τα δεδομένα του ήχου και τους συντελεστές με τα εκθετικά w[] (coeff). Τα δεδομένα όπως γνωρίζουμε είναι 64 και με βάσει τους ορισμούς και τον τρόπο υλοποίησης του αλγόριθμου FFT γράφουμε τον παρακάτω κώδικα. Επίσης, ορίζουμε τα εκθετικά ως συντελεστές με την παρακάτω συνάρτηση.

void FFT(short \* signal, short \* coeff) {

    DSP\_radix2(64, signal, coeff);

    bitrev\_index(index, 64);

    DSP\_bitrev\_cplx((int \*)signal, index, 64);

}

void calculate\_coeff(short \* coeff){

    int i, nx = 64;

    for (i = 0; i < nx/2; i++) {

        coeff[i\*2] = 32767 \* (-cos(i\*2\*PI/nx));

        coeff[i\*2 + 1] = 32767 \* (-sin(i\*2\*PI/nx));

    }

}

Στη συνέχεια, υπολογίζουμε το τετράγωνο του φάσματος με τη συνάρτηση abs\_val() με ορίσματα τα ενοποιημένα πραγματικά και φανταστικά δεδομένα που έχουν εξαχθεί στον πίνακα signal. Για να συμβεί αυτό θα πρέπει να χωρίσουμε τα δεδομένα ξανά σε πραγματικά και φανταστικά και να υπολογίσουμε το άθροισμα των τετραγώνων τους, όπως συμβαίνει στις παρακάτω εντολές.

void abs\_val(int \* signal, int \* amplitude){

    short real, imag;

    int i;

    for (i = 0; i < 64; i++) {

        imag = signal[i] & 0xFFFF;

        real = (signal[i] >> 16) & 0xFFFF;

        amplitude[i] = (int)(real\*real) + (int)(imag\*imag);

    }

}

Τέλος για να κάνουμε τη γραφική συνάρτηση του πίνακα amplitude, τρέχουμε με μία ηχητική είσοδο στο μικρόφωνο της πλακέτας και έπειτα σταματάμε το πρόγραμμα επιλέγοντας στη συνέχεια τις κατάλληλες ρυθμίσεις στο CCS.

