ΕΡΓΑΣΤΗΡΙΟ ΨΗΦΙΑΚΗΣ ΕΠΕΞΕΡΓΑΣΙΑΣ ΣΗΜΑΤΩΝ

Εργαστήριο 6

Ομάδα 09 – Group 1

ΕΠΩΝΥΜΟ	ΟΝΟΜΑ	ΑΡΙΘΜΟΣ ΜΗΤΡΩΟΥ
ΔΑΣΟΥΛΑΣ	ΙΩΑΝΝΗΣ	1053711
ΔΟΥΡΔΟΥΝΑΣ	ΑΡΙΣΤΕΙΔΗΣ ΑΝΑΡΓΥΡΟΣ	1047398

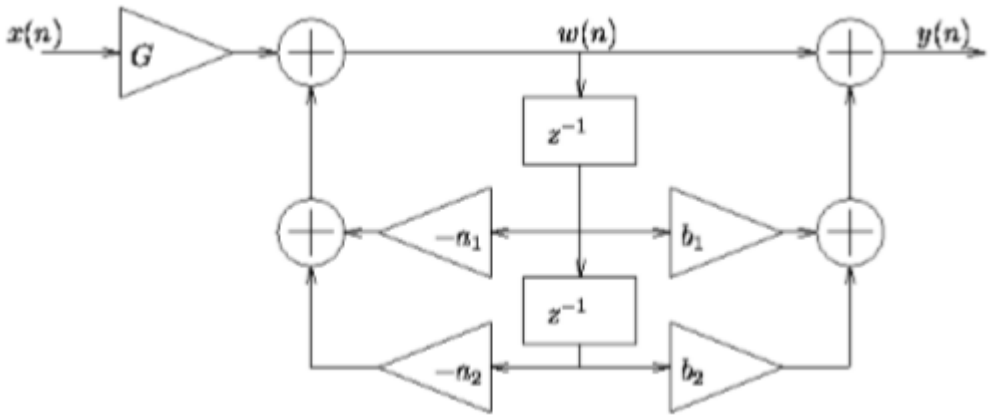
Σχεδιασμός και υλοποίηση IIR φίλτρων

Στόχος:

Αυτό το εργαστήριο αφορά τα φίλτρα κρουστικής απόκρισης άπειρου μήκους, ή IIR (Infinite Impulse Response Filters) φίλτρα. Όπως τα FIR έτσι και τα IIR φίλτρα είναι και αυτά γραμμικά, χρονικά ανεξάρτητα συστήματα, που μπορούν να αναπαράγουν μία μεγάλη γκάμα αποκρίσεων συχνοτήτων. Συγκρινόμενα με τα FIR φίλτρα, τα IIR απαιτούν πολύ μικρότερο αριθμό συντελεστών, όμως η μη γραμμικότητα της φάσης (η καθυστέρηση σημάτων διαφορετικής συχνότητας δεν είναι ίδια) και η χρήση ανάδρασης (feedback) κάνει δύσκολη τη χρήση τους. Συγκεκριμένα, η ανάδραση μπορεί να οδηγήσει σε προβλήματα αστάθειας, όταν η υλοποίηση γίνεται σε έναν επεξεργαστή σταθερής υποδιαστολής (fixed point arithmetic).

Σε αυτό το εργαστήριο ερευνώνται θέματα που έχουν σχέση με την υλοποίηση σε επεξεργαστή σταθερής υποδιαστολής ενός IIR συστήματος 4ης τάξης. Η υλοποίηση θα γίνει με διασύνδεση δύο συστημάτων δεύτερης τάξης, που χρησιμοποιούν την τύπου άμεσης δομής ΙΙ διάταξη.

Μπλοκ διάγραμμα της τύπου άμεσης δομής ΙΙ διάταξης:



Η συνάρτηση μεταφοράς της διάταξης δεύτερης τάξης που φαίνεται στην εικόνα είναι:

$$H(z) = G\frac{1 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

Ασκήσεις:

Άσκηση 6.1

1.Από το μπλοκ διάγραμμα της εικόνας 1, να υπολογιστεί η συνάρτηση διαφορών που σχετίζει την είσοδο x(n)με την έξοδο y(n).Πρώτα πρέπει να γραφεί η εξίσωση διαφορών που συνδέει την είσοδο x(n)με το ενδιάμεσο σήμα w(n).Μετά να γραφεί μία άλλη εξίσωση που συνδέει το ενδιάμεσο αποτέλεσμα w(n)με την έξοδο y(n).Συνδυάζοντας τις δύο αυτές εξισώσεις να βγει η τελική εξίσωση διαφορών, που συνδέει τις x(n)και y(n),αφαιρώντας τα w από τις επιμέρους εξισώσεις.

2.Χρησιμοποιώντας την πρώτη εξίσωση διαφορών, να δημιουργηθεί η συνάρτηση μεταφοράς $W(z)$ $X(z)$, από τη x(n)στο w(n). Χρησιμοποιώντας τη δεύτερη εξίσωση διαφορών να δημιουργηθεί η συνάρτηση μεταφοράς $Y(z)W(z)$, από τη w(n)στη y(n). Συνδυάζοντας τα δύο αποτελέσματα, να επαληθευτεί ότι το σύστημα έχει τη συνάρτηση μεταφοράς, που δίνεται από την εξίσωση (1).

3.Να ελεγχθεί η λειτουργία της συνάρτησης freqz της Matlab για την εμφάνιση της απόκρισης συχνότητας συνάρτησης μεταφοράς, καθώς και της συνάρτησης ellip για τον σχεδιασμό IIR φίλτρου.

4.Να σχεδιαστεί ένα IIR κατωπερατό φίλτρο 4ηςτάξης χρησιμοποιώντας την συνάρτηση ellip:

[B,A]=ellip(4,0.25,10,0.25); freqz(B,A);

5.Βρίσκοντας τους πόλους και τα μηδενικά του φίλτρου, να εκφραστεί η συνάρτηση μεταφοράς σαν το γινόμενο δύο διατάξεων δεύτερης τάξης. Να χρησιμοποιηθούν οι συναρτήσεις της Matlab roots και poly για το σκοπό αυτό. Αφού βρεθούν οι πόλοι και τα μηδενικά, να ομαδοποιηθούν οι συζυγείς πόλους/μηδενικά (ή δύο πραγματικοί πόλοι/μηδενικά). Τώρα η συνάρτηση μεταφοράς H(z) θα ισούται με το γινόμενο δύο συναρτήσεων μεταφοράς τάξης 2, καθεμία όμοιας μορφής με εκείνη της εξίσωσης (1).

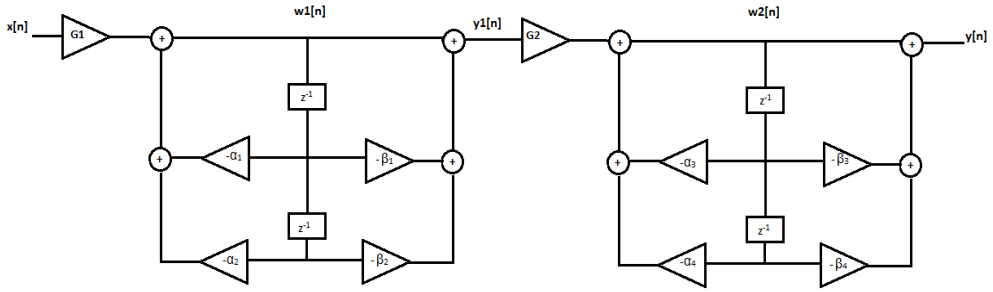
6.Για κάθε διάταξη δεύτερης τάξης, θα πρέπει να επιλεχθεί ένας συντελεστή κέρδους G, έτσι ώστε τα ενδιάμεσα αποτελέσματα να είναι μεταξύ [-1,1], έτσι ώστε να αποφευχθεί η υπερχείλιση.

Όταν$x(n)$∈[−1,1], πρέπει να εξασφαλιστεί ότι τα w(n) και y(n) ανήκουν στην δυναμική περιοχή. Αυτό διότι ισχύει: $w(n)$=∑$h[k]$×G×$x[n-k]$≤G|∑$h[k]$∞k=0|∞ k=0, (2)όπου $h[n]$είναι η κρουστική απόκριση που αντιστοιχεί στο $W(z)X(z)$. Όταν |$x(n)$|≤1, προκειμένου να ισχύει |$w(n)$|≤ 1πρέπει να επλιεχθεί G=1|∑$h[k]$∞k=0|.Ο ακριβής υπολογισμός του |∑$h[k]$∞k=0|δεν είναι γενικά απλή διαδικασία. Μπορεί

να υπολογιστεί κατά προσέγγιση, με τη βοήθεια της Matlab, βρίσκοντας την κρουστική απόκριση που αντιστοιχεί στη $W(z)X(z)$, χρησιμοποιώντας την συνάρτηση filter, με είσοδο ένα μοναδιαίο παλμό. Ακόμη και αν η κρουστική απόκριση είναι άπειρη σε μήκος, υπολογίζοντάς την για μία αρκετά μεγάλη χρονική περίοδο, προσεγγίζεται ικανοποιητικά το $|\sum h[k]\infty k=0|$, αφού τα $h[k]$ υφίστανται ταχεία απόσβεση στο χρόνο. Για τις ανάγκες αυτού του εργαστηρίου, να υπολογιστούν 500 δείγματα της h για την προσέγγιση του αθροίσματος. Χρησιμοποιώντας την έκφραση $W(z)X(z)$, να βρεθεί ο κατάλληλος συντελεστής κέρδους $G1$, έτσι ώστε $|w(n)|\leq 1$. Μετά, χρησιμοποιώντας την έκφραση $Y(z)W(z)$, να βρεθεί ο κατάλληλος συντελεστής κέρδους $G2$, έτσι ώστε $|y(n)|\leq 1$. Ο συνολικός συντελεστής κέρδους της διάταξης δεύτερης τάξης, που εξασφαλίζει ότι τα $w(n)$και $y(n)$παίρνουν τιμές στο διάστημα [-1,1] είναι $G=G1\times G2$. Να βρεθούν οι συντελεστές κέρδους για τις δύο διατάξεις δεύτερης τάξης για τον σχεδιασμό του φίλτρου 4ηςτάξης

Ερώτημα 6.1.1

Για την υλοποίηση του ου φίλτρου 4ης τάξης το φίλτρο χωρίζεται σε 2 επιμέρους υποσυστήματα 2ης τάξης. Το συνολικό σύστημα είναι της μορφής:



Οι συναρτήσεις μεταφοράς των επιμέρους φίλτρων είναι

$$\frac{W_1(z)}{X(z)} = \frac{G_1}{1 + a_1 z^{-1} + a_2 z^{-2}} \qquad \frac{W_2(z)}{Y_1(z)} = \frac{G_2}{1 + a_3 z^{-1} + a_4 z^{-2}}$$

$$\frac{Y_1(z)}{W_1(z)} = 1 + b_1 z^{-1} + b_2 z^{-2} \qquad \frac{Y(z)}{W_2(z)} = 1 + b_3 z^{-1} + b_4 z^{-2}$$

Τελικά :

$$\frac{Y_1(z)}{X(z)} = \frac{G_1(1 + b_1 z^{-1} + b_2 z^{-2})}{1 + a_1 z^{-1} + a_2 z^{-2}} \qquad \frac{Y(z)}{Y_1(z)} = \frac{G_2(1 + b_3 z^{-1} + b_4 z^{-2})}{1 + a_3 z^{-1} + a_4 z^{-2}}$$

Ερώτημα 6.1.4

Κατόπιν μελέτης των εντολών elip και freqz δημιουργήθηκε ένα ελλειπτικό φίλτρο με κανονικοποιημένη συχνότητα αποκοπής 0,25 που αντιστοιχεί σε συχνότητα 6 kHZ.

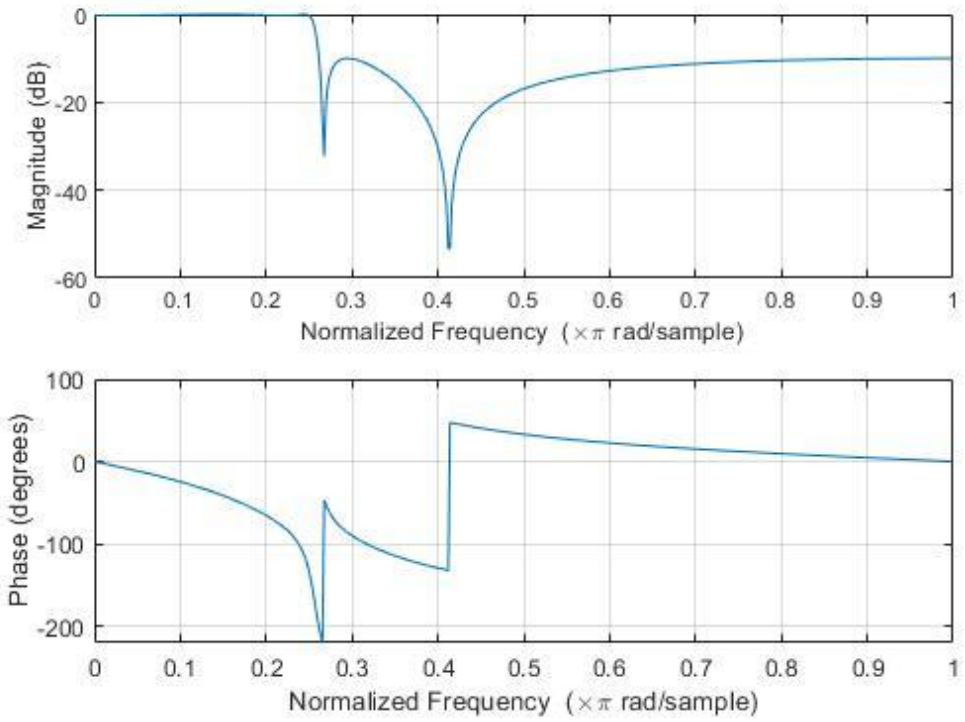
```
[B,A] = ellip(4, 0.25, 10, 0.25);
freqz(B,A);
```

Τα B , A είναι οι συντελεστές και έχουν τις ακόλουθες τιμές :

B = 0.2932 -0.5501 0.7978 -0.5501 0.2932

A =1.0000 -2.2948 2.6602 -1.4872 0.4141

Και δημιουργήθηκαν οι ακόλουθες γραφικές απόκρισης συχνότητας και φάσης:



Ερώτημα 6.1.5

Στη συνέχεια χρησιμοποιήθηκαν οι ακόλουθες εντολές για την εύρεση των ριζών :

```
poles = roots(A);
zeros = roots(B);
```

Μέσω των δύο εντολών roots υπολογίζονται τα μηδενικά και οι πόλοι .

poles =

0.6715 + 0.7012i

0.6715 - 0.7012i

0.4759 + 0.4613i

0.4759 - 0.4613i

zeros =

0.2697 + 0.9629i

0.2697 - 0.9629i

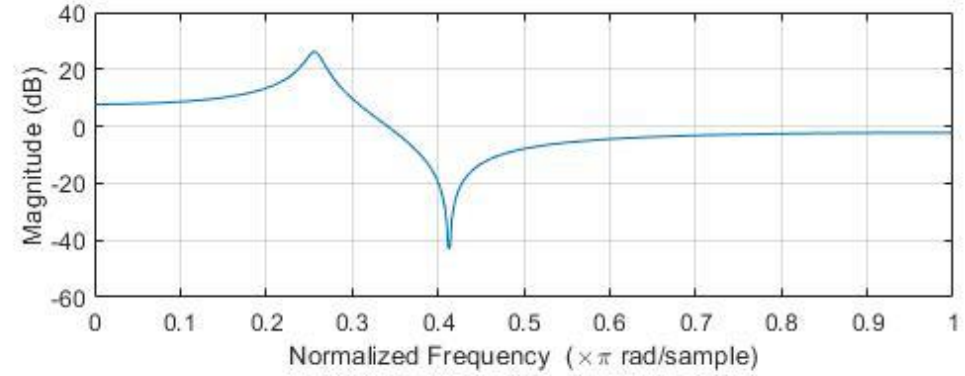
0.6684 + 0.7438i

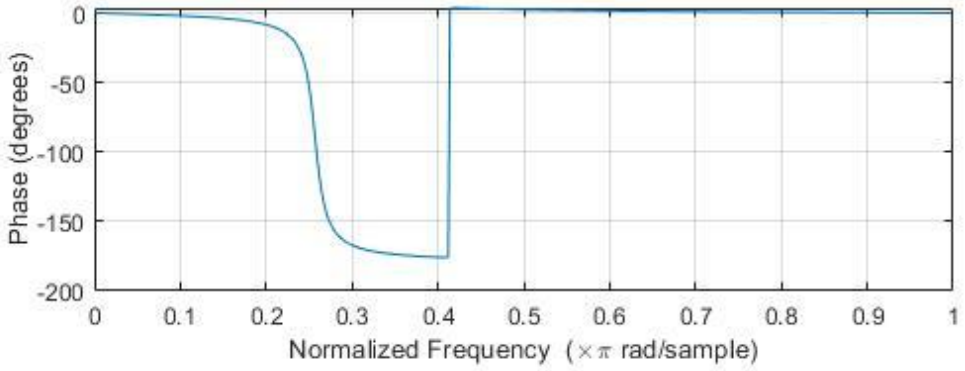
0.6684 - 0.7438i

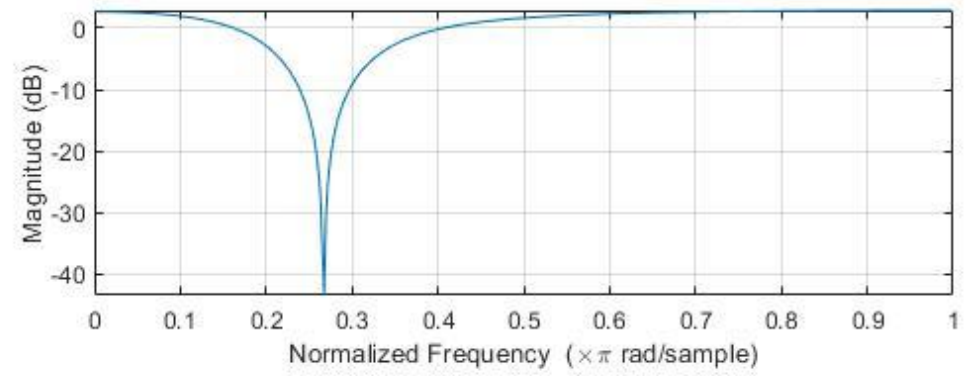
```
 7 -    A1 = poly ([0.6715 + 0.7012i 0.6715 - 0.7012i]);
 8 -    A2 = poly ([0.4759 + 0.4613i 0.4759 - 0.4613i]);
 9 -    B1 = poly ([0.2697 + 0.9629i 0.2697 - 0.9629i]);
10 -    B2 = poly ([0.6684 + 0.7438i 0.6684 - 0.7438i]);
11
12 -    figure(2);
13 -    freqz(B1,A1)
14 -    figure(3);
15 -    freqz(B2,A2)
```

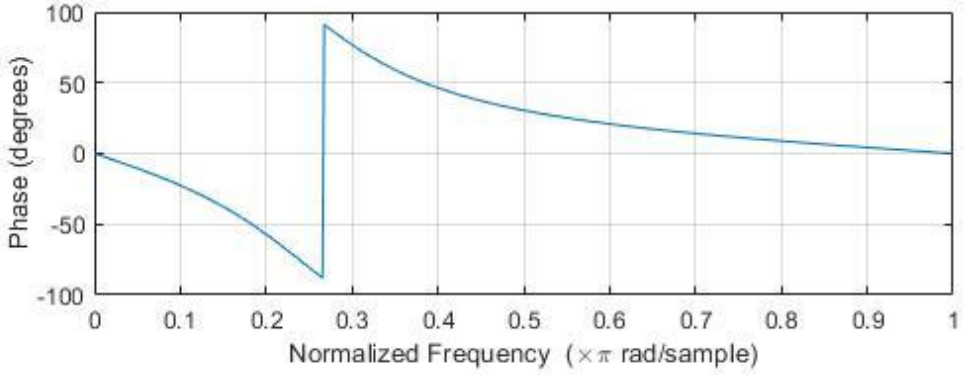
Με την χρήση της εντολής poly θα δημιουργηθούν οι συντελεστές των 2 επιμέρους φίλτρων A1 , A2 , B1 , B2. Η poly δέχεται σαν ορίσματα ένα ζευγάρι μηδενικών και πόλων αντίστοιχα. Η αποκρίσεις συχνότητας και ο φάσεις των δύο επιμέρους φίλτρων

φαίνονται στις ακόλουθες φωτογραφίες.









Ερώτημα 6.1.6

Ο κώδικας που αναπτύχθηκε είναι ο ακόλουθος:

```
clear all;
[B,A] = ellip(4, 0.25, 10, 0.25);
poles = roots(A);
zeroes = roots(B);

A1 = poly ([0.6715 + 0.7012i 0.6715 - 0.7012i]);
A1_two = twocomplement(A1/2);

A2 = poly ([0.4759 + 0.4613i 0.4759 - 0.4613i]);
A2_two = twocomplement(A2/2);

B1 = poly ([0.2697 + 0.9629i 0.2697 - 0.9629i]);
B1_two = twocomplement(B1/2);

B2 = poly ([0.6684 + 0.7438i 0.6684 - 0.7438i]);
B2_two = twocomplement(B2/2);

samples = zeros(1, 500);
samples(1) = 1;

A1_filter = filter(1,A1,samples);
G_A1 = (abs(sum(A1_filter)))^(-1);
G_A1_two = twocomplement(G_A1/2);

A2_filter = filter(1,A2,samples);
G_A2 = (abs(sum(A2_filter)))^(-1);
G_A2_two = twocomplement(G_A2/2);

B1_filter = filter(B1,1,samples);
G_B1 = (abs(sum(B1_filter)))^(-1);
G_B1_two = twocomplement(G_B1/2);

B2_filter = filter(B2,1,samples);
G_B2 = (abs(sum(B2_filter)))^(-1);
G_B2_two = twocomplement(G_B2/2);

G1 = G_A1 * G_B1;
G1_two = twocomplement(G1/2);

G2 = G_A2 * G_B2;
G2_two = twocomplement(G2/2);
```

Αρχικά, δημιουργείται το ελλειπτικό φίλτρο που ζητήθηκε στο ερώτημα 1.4 με συντελεστές A και B. Στη συνέχεια υπολογίζονται οι συντελεστές των επιμέρους φίλτρων A1 ,B1 και A2, B2 μέσω της συνάρτησης poly. Έπειτα χρησιμοποιείται η συνάρτηση twoscompliment που δίνεται στο eclass. Η συνάρτηση αυτή μετατρέπει τις τιμές του φίλτρου σε δυαδικούς αριθμούς 16 bit, ωστόσο για να λειτουργήσει σωστά πρέπει οι αριθμοί να βρίσκονται μεταξύ του [-1,1] για αυτό και γίνεται οι διαίρεση με το 2 η οποία τους περιορίζει οι αριθμοί σε αυτό το εύρος.

Με την εντολή samples δημιουργείται ένας buffer 500 μηδενικών στοιχείων με μία μόνο στήλη. Ο πίνακας αυτό χρησιμοποιείται έπειτα στην εντολή filter. Η filter υπολογίζει τις κρουστικές αποκρίσεις του εκάστοτε συντελεστή. Έτσι με βάση τον τύπο $G=\frac{1}{|\sum_{K=0}^{K=\infty} H(K)|}$. Υπολογίζεται το κέρδος για κάθε συντελεστή. Το συνολικό κέρδος του κάθε φίλτρου είναι το γινόμενο του κέρδους των επιμέρους συντελεστών και

αποθηκεύεται στις μεταβλητές G1 , G2 και στη συνέχεια μετατρέπεται σε συμπλήρωμα του 2 μέσω της twoscomplement/.

Για την άσκηση 6.2 πρέπει να χρησιμοποιηθούν οι συντελεστές που υπολογίστηκαν για αυτό με το παρακάτω κομμάτι κώδικα καταγράφονται σε ένα header file το οποίο μπορεί να διαβαστεί στην C μέσω της εντολής include.

```
45 -    fprintf(filtro, '#define A1_1 %d\n', A1_two(2));
46 -    fprintf(filtro, '#define A1_2 %d\n', A1_two(3));
47
48 -    fprintf(filtro, '#define A2_1 %d\n', A2_two(2));
49 -    fprintf(filtro, '#define A2_2 %d\n', A2_two(3));
50
51 -    fprintf(filtro, '#define B1_1 %d\n', B1_two(2));
52 -    fprintf(filtro, '#define B1_2 %d\n', B1_two(3));
53
54 -    fprintf(filtro, '#define B2_1 %d\n', B2_two(2));
55 -    fprintf(filtro, '#define B2_2 %d\n', B2_two(3));
56
57 -    fprintf(filtro, '#define G1_two %d\n', G1_two);
58 -    fprintf(filtro, '#define G2_two %d\n', G2_two);
59
60 -    fclose('all');
```

Άσκηση 6.2 (Υλοποίηση IIR φίλτρου)

1. Γράψτε ένα πρόγραμμα στη C, που υλοποιεί την διάταξη δεύτερης τάξης. 2. Γράψτε ένα πρόγραμμα στη C, που υλοποιεί το ολόκληρο φίλτρο 4ης τάξης. 3. Ελέγξτε την λειτουργία του φίλτρου, χρησιμοποιώντας ως είσοδο ένα σήμα ημιτόνου μεταβαλλόμενης συχνότητας.

Αρχικά, δημιουργήθηκε το πρόγραμμα για την διάταξη δεύτερης τάξης (6_2_1.c) , με την χρησιμοποίηση των συντελεστών που προέκυψαν από το Matlab και αποθηκεύτηκαν στο αρχείο ‘filtro.h’.

Γίνονται οι απαραίτητες αρχικοποιήσεις, μετατρέποντας τους συντελεστές σε short int μεταβλητές και δημιουργώντας δύο πίνακες, έναν για αποθήκευση της εισόδου και έναν για αποθήκευση της εξόδου, καθώς και άλλες χρήσιμες για την λειτουργία μεταβλητές.

```
short a1_1 = A1_1, a1_2 = A1_2, a2_1 = A2_1, a2_2 = A2_2;
short b1_1 = B1_1, b1_2 = B1_2, b2_1 = B2_1, b2_2 = B2_2;
short G1 = G1_two, G2 = G2_two;
short buffer_in[500] = {0}, buffer_out[500] = {0};
int current = 0;
int data;
short prev[2]= {0};
```

Οι απαραίτητοι υπολογισμοί για την εύρεση της εξόδου γίνονται στην συνάρτηση του interrupt (serial_port_rcv_isr()). Λαμβάνεται η είσοδος και έπειτα μέσω των μαθηματικών τύπων του συστήματος, που δίνονται και ως σχόλια στο πρόγραμμα, υπολογίζεται η έξοδος και αποθηκεύεται στον πίνακα εξόδου.

```
// interrupt service routine
interrupt void serial_port_rcv_isr()
{

    data = input_leftright_sample();
    // "data" contains both audio channels

    data = (short) data;
    short int w, y = 0;

    buffer_in[current] = data;

    //w[n] = Gx[n] + ( -a1w[n-1] - a2w[n-2] )
    w = mul(G1,data) + ( - mul(a1_1, prev[0]) - mul(a1_2, prev[1]) );

    //y[n] = w[n] + ( b1w[n-1] + b2w[n-2] )
    y = w + mul(b1_1, prev[0]) + mul(b1_2, prev[1]);

    prev[1] = prev[0];
    prev[0] = w;

    buffer_out[current]= y;
    current++;
    if (current == 500) current = 0;

    // process "data", or pass another variable to change the output
    output_leftright_sample(y);

    return;
}
```

Οι πολλαπλασιασμοί γίνονται με την βοήθεια της συνάρτησης mul(short int, short int), η οποία δέχεται ως όρισμα δύο short integers, διπλασιάζει τον έναν, αρχικά, διότι προηγουμένως, στην δημιουργία των συντελεστών στο Matlab, οι συντελεστές διαιρούνται διά δύο για να γίνει η κβάντιση. Έπειτα, πολλαπλασιάζονται οι δύο παράμετροι και γίνεται shift 15 bits προς τα δεξιά για την εφαρμογή του Q-15 format.

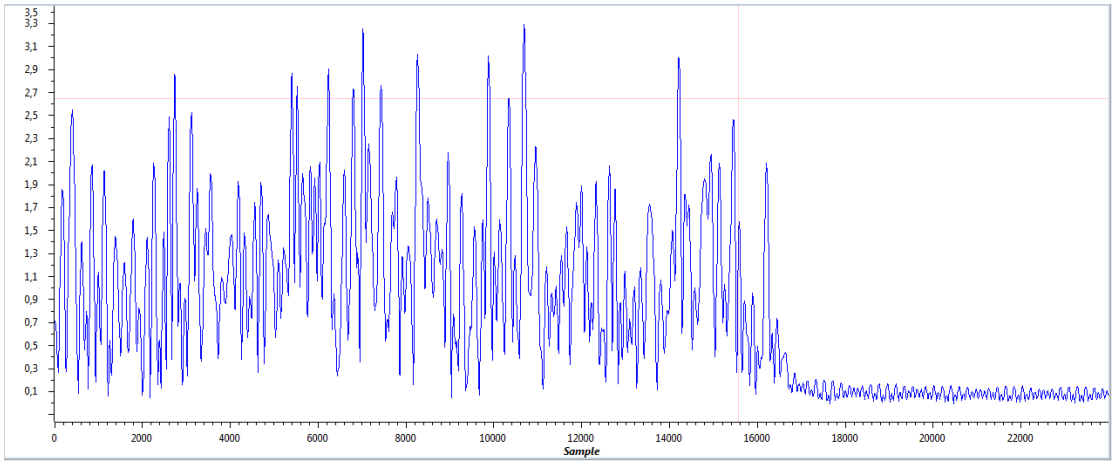
```
short int mul(short int m1, short int m2){
    short int result;

    m1 = m1 << 1;
    result = (short) (m1* ((int)m2) >> 15);

    return result;
}
```

Ύστερα, δημιουργήθηκε το πρόγραμμα για την διάταξη ολόκληρου του φίλτρου 4ης τάξης (6_2_2.c), με τον συνδυασμό δύο φίλτρων 2ης τάξης. Οι υπολογισμοί γίνονται ανάλογα με πριν, με το δεύτερο σύστημα να δέχεται ως είσοδο την έξοδο του πρώτου για να προκύψει το τελικό το αποτέλεσμα·

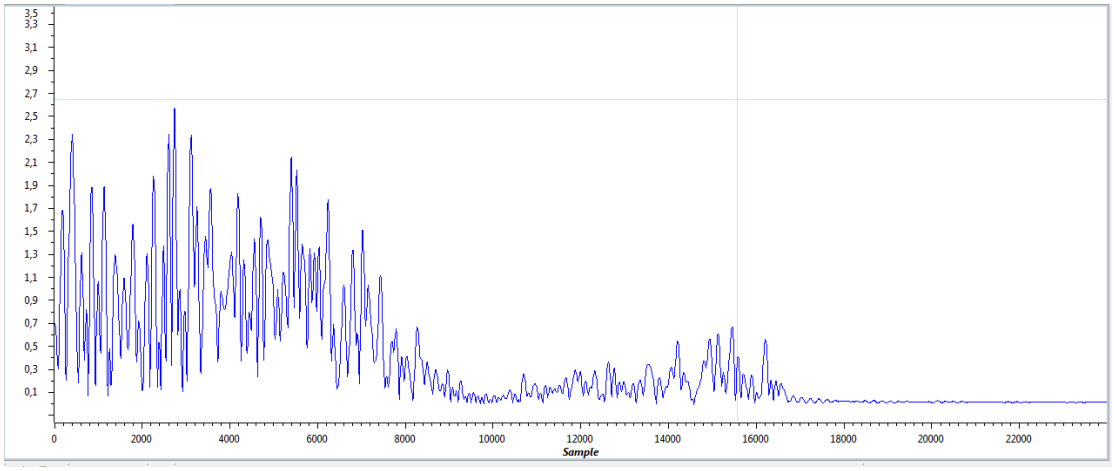
```
short int calc(short int data){
    short int w1, w2, y1, y2;
        //w1[n] = Gx[n] + ( -a1w1[n-1] - a2w1[n-2] )
        w1 = mul(G1,data) + ( - mul(a1_1, prev1[0]) - mul(a1_2, prev1[1]) );

        //y1[n] = w[n] + ( b1w1[n-1] + b2w1[n-2] )
        y1 = w1 + mul(b1_1, prev1[0]) + mul(b1_2, prev1[1]);

        prev1[1] = prev1[0];
        prev1[0] = w1;


        //w2[n] = Gy[n] + ( -a1w2[n-1] - a2w2[n-2] )
        w2 = mul(G2,y1) + ( - mul(a2_1, prev2[0]) - mul(a2_2, prev2[1]) );

        //y2[n] = w2[n] + ( b1w2[n-1] + b2w2[n-2] )
        y2 = w2 + mul(b2_1, prev2[0]) + mul(b2_2, prev2[1]);

        prev2[1] = prev2[0];
        prev2[0] = w2;

        return y2;
}
```

Η σωστή λειτουργία του φίλτρου επιβεβαιώθηκε χρησιμοποιώντας ως είσοδο ένα σήμα ημιτόνου μεταβαλλόμενης συχνότητας και ένα σήμα λευκού θορύβου. Η αναμενόμενη συχνότητα αποκοπής του βαθυπερατού φίλτρου υπολογίζεται από την τέταρτη παράμετρο (0.25) της κλήσης της συνάρτησης ellip() που χρησιμοποιείται στο Matlab για την εύρεση των συντελεστών, με συχνότητα δειγματοληψίας 48KHz. Συγκεκριμένα: F = (0.25π * 48KHz) / 2π = 6KHz.
Επομένως, η συχνότητα αποκοπής είναι ίση με 6KHz.

Είσοδος λευκού θορύβου:



Έξοδος λευκού θορύβου:



Όπως, φαίνεται στην έξοδο, μετά τα 6KHz αποκόπτονται σταδιακά οι συχνότητες.

Έπειτα, έγινε προσπάθεια για υλοποίηση του φίλτρου 4ης τάξης χωρίς συνδυασμό των φίλτρων 2ης τάξης (6_2_3.c). Δημιουργήθηκαν οι συντελεστές στο Matlab, ξανά, και αποθηκεύτηκαν στο αρχείο ‘filtro2.h’ διαιρώντας τους διά το 4 αυτή τη φορά. Συμπληρώθηκαν οι απαραίτητοι υπολογισμοί, ενώ πλέον στην συνάρτηση mul(short int, short int), γίνεται ο πολλαπλασιασμός με το 4.

```
short int mul(short int m1, short int m2){
    short int result;
    result = (short) ((m1* m2) >> 13);

    return result;
}

short int calc_4(short int data){
    short w1, w2, y1, y2;
        //w[n] = Gx[n] + ( -a1w[n-1] - a2w[n-2] - a3w[n-3] - a4w[n-4])
        w1 = (int) mul(G,data) - (int) mul(a1, prev[0]) -(int) mul(a2, prev[1]);
        w2 = (int)w1 -(int) mul(a3, prev[2]) -(int) mul(a4, prev[3]);

        //y[n] = w[n] + ( b1w[n-1] + b2w[n-2] + b3w[n-3] + b4w[n-4] )
        y1 = (int) mul(0.25, w2) + (int) mul(b1, prev[0]) + (int) mul(b2, prev[1]);
        y2 = (int)y1 + (int) mul(b3, prev[2]) + (int) mul(b4, prev[3]);

        prev[3] = prev[2];
        prev[2] = prev[1];
        prev[1] = prev[0];
        prev[0] = w2;

        return (short)y2;
}
```

Παρόλα αυτά, λόγω κάποιου προβλήματος που πρόκυπτε, πιθανότατα λόγω υπερχειλίσεων, η έξοδος δεν είχε το σωστό αποτέλεσμα.

Άσκηση 6.3

Με την εντολή butter, να δημιουργηθεί ένα notchφίλτρο (εγκοπής):

[B,A] = butter(2,[0.07 0.10],'stop');

Να χρησιμοποιηθεί η εντολή freqz για τη σύγκριση του φίλτρου χωρίς κβάντιση με τις εξής δύο κβαντισμένες εκδόσεις:

•Πρώτα να κβαντιστεί το αρχικό φίλτρο 4ηςτάξης.

•Μετά να κβαντιστούν τα επιμέρους φίλτρα δεύτερης τάξης ξεχωριστά και συνδυάστε ξανά τα δύο φίλτρα.

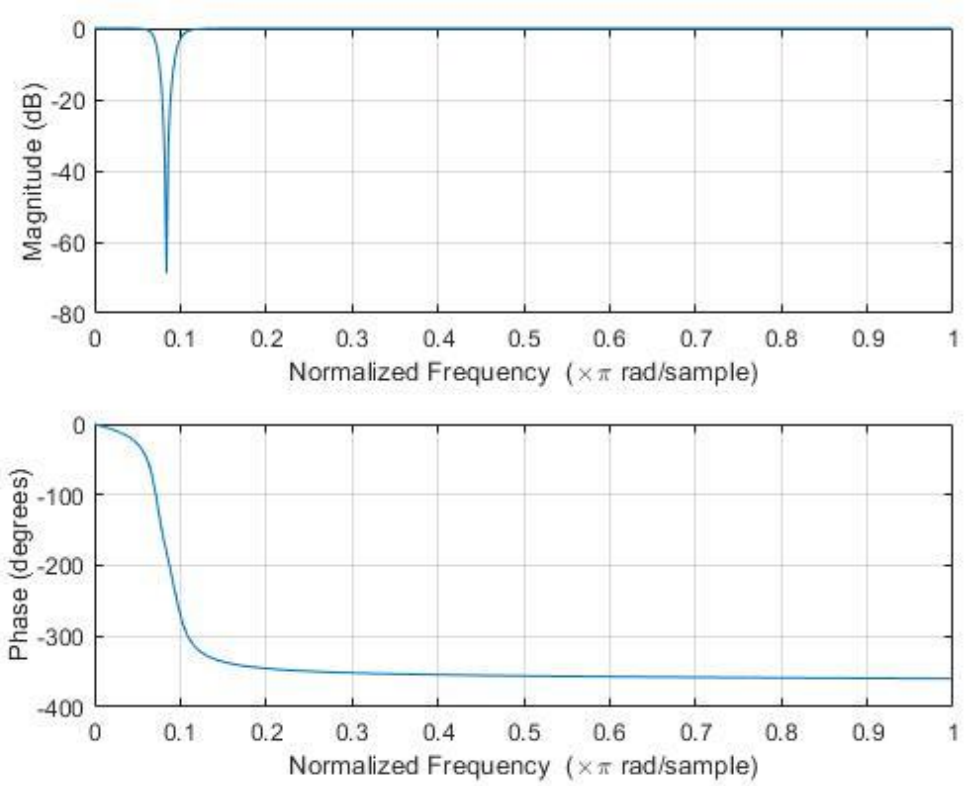
Να ελεγχθούν οι αποκρίσεις των τριών εκδόσεων του φίλτρου.

Να εξηγηθεί ο λόγος για τον οποίο πάντα η υλοποίηση των IIRφίλτρων γίνεται με συνδυασμό διατάξεων δεύτερης τάξης.

Ακολουθώντας τις οδηγίες που δίνονται στην εκφώνηση της άσκησης δημιουργείται ένα φίλτρο butter $4^{ης}$ τάξης.

```
4 -     [B,A] = butter(2,[0.07 0.10],'stop')
5 -     poles1 = roots(A)
6 -     zeroes1 = roots(B)
7 -     figure(1)
8 -     freqz(B,A);
```

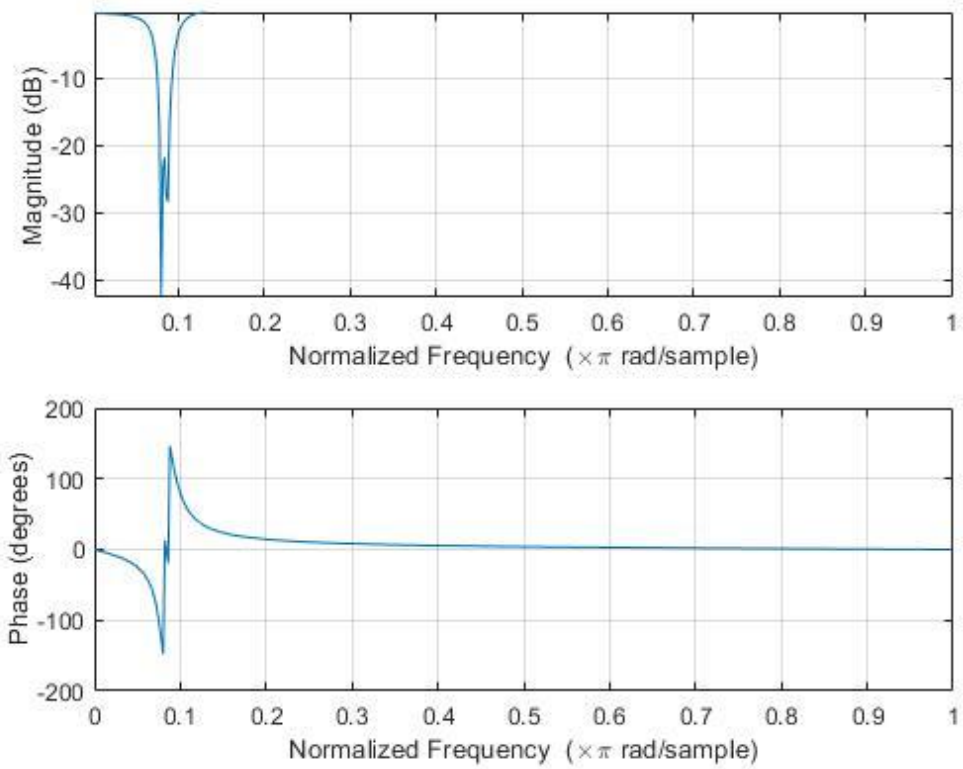
Με απόκριση συχνότητας και φάση :



Στη συνέχεια γίνεται οι κβάντιση των συντελεστών του φίλτρου 4ης τάξης μέσω του ακόλουθου κώδικα

```
10 -    B = 8*((round((B*32768/8)))/32768)
11 -    A = 8*((round((A*32768/8)))/32768)
12 -    poles2 = roots(A)
13 -    zeroes2 = roots(B)
14 -    figure(2)
15 -    freqz(B,A);
```

Ο πολλαπλασιασμός και η διαίρεση με το 8 γίνεται για τον περιορισμό τον τιμών στο διάστημα [-1,1], καθώς η μεγαλύτερη τιμή των συντελεστών είναι λίγο μικρότερη του 8 . Με την κβάντιση των συντελεστών λαμβάνονται οι ακόλουθες γραφικές παραστάσεις



Οι οι δύο κορυφές που φαίνονται στις δύο γραφικές παραστάσεις των κβαντισμένων συντελεστών στις θέσεις που κανονικά γινόταν εγκοπή οφείλονται στην μετακίνηση των πόλων και των μηδενικών που προκύπτει από την κβάντιση των συντελεστών.

Χωρίς κβάντιση :

poles1 = 0.9213 + 0.2813i

0.9213 - 0.2813i

0.9457 + 0.2210i

0.9457 - 0.2210i

zeroes1 = 0.9656 + 0.2599i

0.9656 - 0.2599i

0.9656 + 0.2599i

0.9656 - 0.2599i

Με κβάντιση :

poles2 = 0.9461 + 0.2378i

0.9461 - 0.2378i

0.9208 + 0.2679i

0.9208 - 0.2679i

zeroes2 = 0.9628 + 0.2701i

0.9628 - 0.2701i

0.9684 + 0.2494i

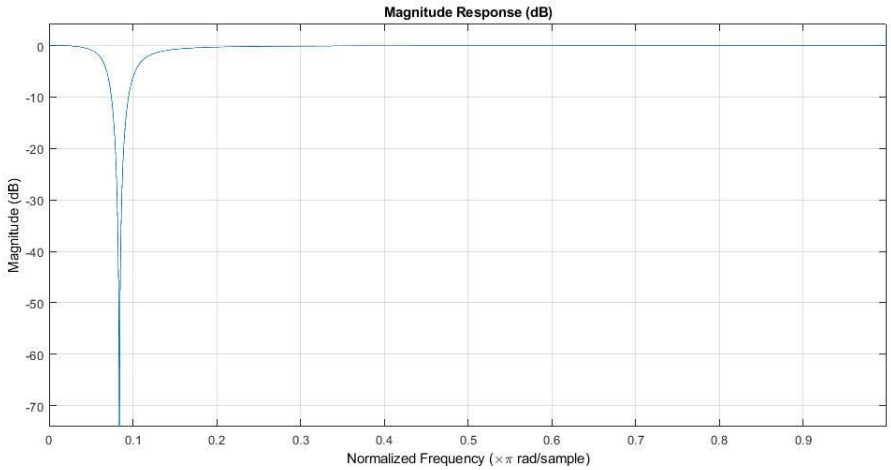
0.9684 - 0.2494i

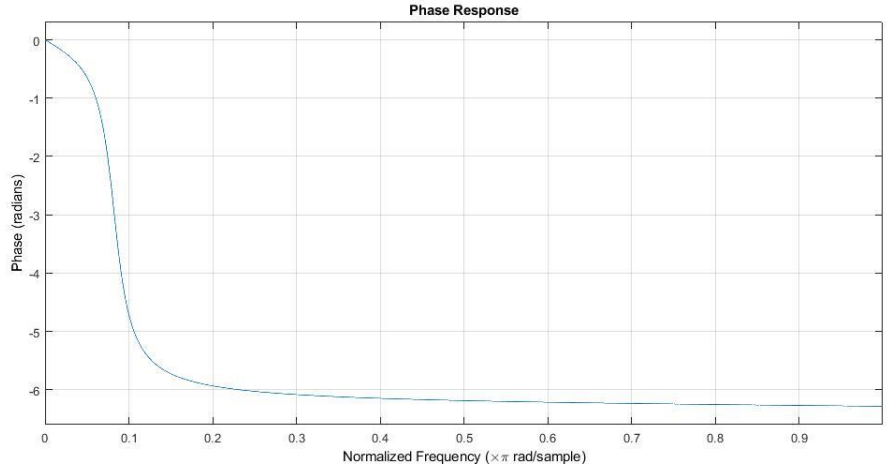
Για το δέυτερο υποερώτημα αναπτύχθηκε ο ακόλουθος κώδικας:

```
%%

clc;
clear;
[B1,A1] = butter(1,[0.07 0.10],'stop')
[B2,A2] = butter(1,[0.07 0.10],'stop')

B1 = 8*((round((B1*32768/8)))/32768)
A1 = 8*((round((A1*32768/8)))/32768)

B2 = 8*((round((B2*32768/8)))/32768)
A2 = 8*((round((A2*32768/8)))/32768)


H1=dfilt.df2t(B1,A1);
H2=dfilt.df2t(B2,A2);


Hcas=dfilt.cascade(H1,H2)

freqz(Hcas);
phasez(Hcas);
```

Μέσω του παραπάνω κώδικα υπολογίζονται οι συντελεστές δύο φίλτρων butter δεύτερης τάξης. Οι συντελεστές κβαντίζονται με την ίδια διαδικασία που ακολουθήθηκε στο προηγούμενο ερώτημα. Στη συνέχεια δημιουργούνται τα φίλτρα με τους κβαντισμένους συντελεστές και υπολογίζεται το τελικό φίλτρο ,που αποτελείται από τα 2 φίλτρα συνδεδεμένα

σε σειρά , μέσω της εντολής Hcas=dfilt.cascade(H1,H2). Οι αποκρίσεις συχνότητας και φάσης είναι οι ακόλουθες:





Όταν πραγματοποιηθεί κβάντιση σε στα δύο επιμέρους φίλτρα με την ίδια διαδικασία το αποτέλεσμα είναι αρκετά καλύτερο καθώς οι γραφικές είναι αρκετά όμοιες με την αρχική. Η κβάντιση δουλεύει καλύτερα στα δύο φίλτρα $2^{ης}$ τάξης από ότι στο ένα φίλτρο $4^{ης}$ λόγω του μεγαλύτερου αριθμού πράξεων καθώς και των βρόχων ανάδρασης στο φίλτρο $4^{ης}$ τάξης. Πίο συγκεκριμένα ο τύπος που χρησιμοποιείται σε ένα φίλτρο $4^{ης}$ τάξης απαιτεί παραπάνω πράξεις. Έτσι κατά την υλοποίηση υπάρχει μεγαλύτερη πιθανότητα να γίνουν σφάλματα στην στρογγυλοποίηση και να υπάρξει υπερχείλιση σε κάποια από τις μεταβλητές. Τα παραπάνω έχουν ως αποτέλεσμα την αύξηση των σφαλμάτων και συνεπώς την τελική δημιουργία ενός χειρότερου φίλτρου $4^{ης}$ τάξης. Τα ακόλουθα φαίνονται καλύτερα στο επιρπόσθετο ερώτημα υλοποίησης του ενός φίλτρου $4^{ης}$ τάξης στο οποίο συναντήθηκαν τα προβλήματα αυτά.

Υλοποίηση φίλτρου 4ης τάξης

Ακολουθώντας την διαδικασία που περιεγράφηκε στα προηγούμενα ερωτήματα, ο κώδικας σε Matlab για την υλοποίηση ενός φίλτρου 4η τάξης είναι ο ακόλουθος:

```
%% Filtro 4hs taxis
clc;
clear;

[B,A] = ellip(4, 0.25, 10, 0.25);
A_two = twocomplement(A/4);
B_two = twocomplement(B/4);

samples = zeros(1, 500);
samples(1) = 1;

A_filter = filter(1,A,samples);
G_A = (abs(sum(A_filter)))^(-1);
G_A_two = twocomplement(G_A/4);

B_filter = filter(B,1,samples);
G_B = (abs(sum(B_filter)))^(-1);
G_B_two = twocomplement(G_B/4);

G = G_A * G_B;
G_two = twocomplement(G/4);

filtro = fopen('filtro4.h', 'w+');

fprintf(filtro, '#define A1 %d\n', A_two(2));
fprintf(filtro, '#define A2 %d\n', A_two(3));
fprintf(filtro, '#define A3 %d\n', A_two(4));
fprintf(filtro, '#define A4 %d\n', A_two(5));

fprintf(filtro, '#define B1 %d\n', B_two(2));
fprintf(filtro, '#define B2 %d\n', B_two(3));
fprintf(filtro, '#define B3 %d\n', B_two(4));
fprintf(filtro, '#define B4 %d\n', B_two(5));

fprintf(filtro, '#define G_two %d\n', G_two);

fclose('all');
```

Με λογική αντίστοιχη με αυτή των προηγούμενων ερωτημάτων υπολογίζονται οι συντελεστές ενός ελλειπτικού φίλτρου 4ης τάξης. Οι συντελεστές μετατρέπονται σε εκφράσεις 16 bit μέσω τις twocompliment. Στην συνέχεια με την filter υπολογίζονται οι κρουστικές αποκρίσεις των συντελεστών A και B και έπειτα υπολογίζονται τα κέρδη

G_A και G_B των συντελεστών. Το συνολικό κέρδος είναι ίσο με το γινόμενο των G_A και G_B και μετατρέπεται και αυτό σε 16 bit αριθμό. Οι συντελεστές του φίλτρου και τα κέρδη γράφονται σε ένα header αρχείο με όνομα filtro4.h το οποίο συμπεριλαμβάνεται στην C με χρήση της εντολής include.

Βιβλιογραφία:

- **TMS320C67x/C67x+ DSP CPU and Instruction Set Reference Guide**
- **ΕΚΦΩΝΗΣΗ ΑΣΚΗΣΗΣ LAB-4 – ECLASS**
- **TMS320C6713 DATASHEET**