Universitatea POLITEHNICA din Bucureşti
Facultatea de electronică, Telecomunicații și Tehnologia Informației

TEMA 2

PROCESOARE DE SEMNAL IN COMUNICATII

Nume student: STANESCU VLAD-CONSTANTIN

Grupa: 444C

1. Partea 1

1.1 Cerinta temei

I. Proiectați în MATLAB un filtru digital folosind **metoda directă de proiectare** cu specificațiile de proiectare din tabelul de mai jos (se vor vedea și explicațiile din **Anexa la Tema 2**).

a) Determinați (dacă nu se specifică în tabel) ordinul și frecvența de tăiere. Dacă ordinul filtrului se obține mai mare ca 4 modificați unii parametri pentru a nu depăși acest ordin.

b) Determinați coeficienții funcției filtrului digital $H(z)$ și reprezentați grafic răspunsul în frecvență și poziția polilor și a zerourilor pentru filtrul digital proiectat.

c) Se consideră implementarea în **forma directă 2**. Realizați scalarea funcției de transfer și a semnalului folosind **regula de scalare L1**.
   - Determinați coeficienții obținuți după scalarea funcției de transfer.
   - Determinați coeficientul de scalare a semnalului de la intrarea structurii.

d) Desenați structura obținută după scalare inclusiv cu coeficienții de scalare.

STĂNESCU C.D. Vlad-Constantin	444C	FTJ	Cebâșev2			16000	2800	4400	1	40

a)

Funcția cheb2ord este utilizata pentru a determina ordinul unui filtru Cebisev de tip II. Dupa prima rulare a codului, ordinul era egal cu 5, in acest caz am modificat frecventa in banda de oprire astfel incat ordinul sa nu fie mai mare decat 4 cum se mentioneaza in cerinta. Dupa ce am aflat frecventa de taiere normalizata am putut afla si frecventa de taiere in Hz (frecventa la care castigul filtrului atinge -3 decibeli).

```
% Specificații de proiectare
Fs = 16000;          % Frecvența de esantionare (Hz)
Ftb = 2800;           % Frecvența limită în banda de trecere (Hz)
Fob = 4400;          % Frecvența în banda de oprire (Hz)
Rp = 1;              % Riplul maxim în banda de trecere (dB)
Rs = 40;            % Riplul maxim în banda de oprire (dB)

% Punctul a
% Calculează ordinul și frecvența de tăiere normalizata
[n, Wn] = cheb2ord(Ftb/(Fs/2), Fob/(Fs/2), Rp, Rs);

% Deoarece ordinul initial este egal cu 5, am modificat frecventa de oprire
if n > 4
    disp('Ordinul depășește 4. Modificăm parametrii.');
    Fob = 2*Ftb; % Modificăm frecvența in banda de oprire=>Fob=5600Hz
    [n, Wn] = cheb2ord(Ftb/(Fs/2), Fob/(Fs/2), Rp, Rs);
    fprintf('Ordinul este: %.2f\n', n);
end

% Frecvența de tăiere(Hz)
ft = Wn * (Fs / 2);
fprintf('Frecvența de tăiere: %.2f Hz\n', ft);
```

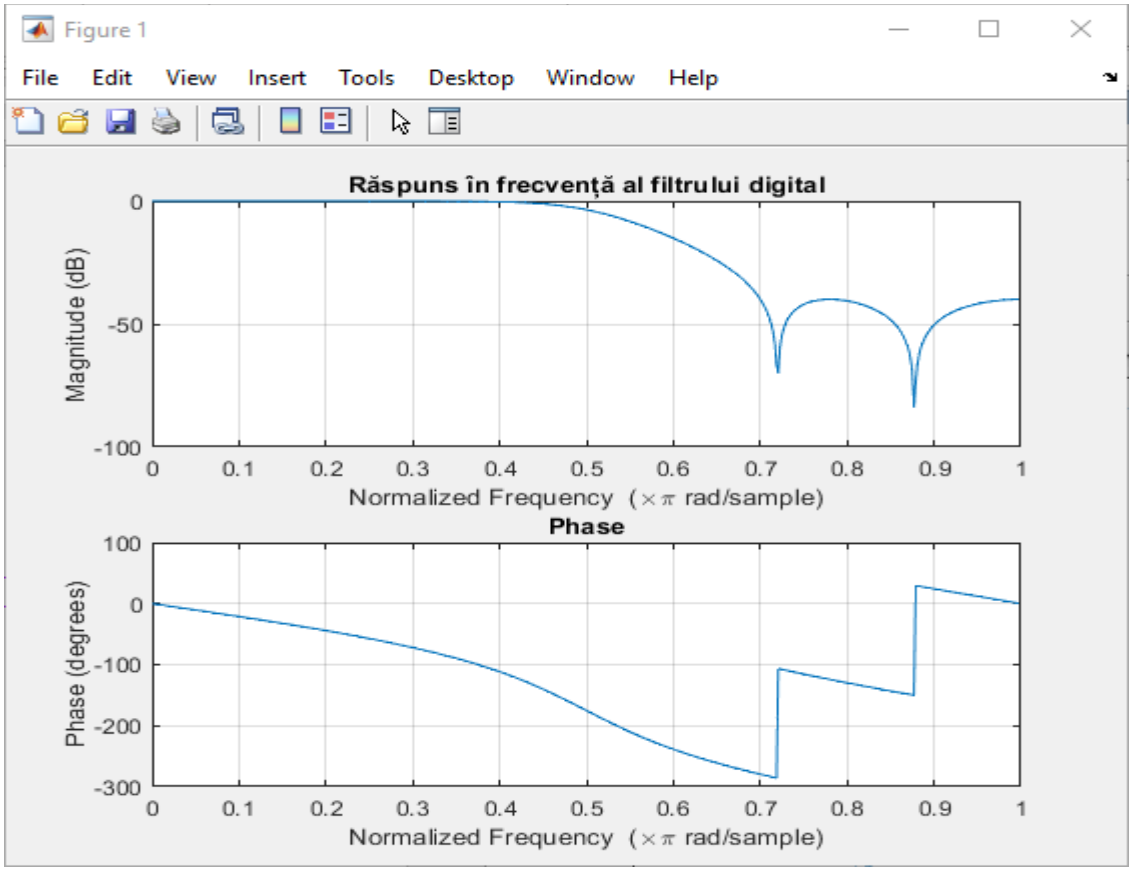
```
Ordinul depășește 4. Modificăm parametrii.
Ordinul este: 4.00
Frecvența de tăiere: 5600.00 Hz
```

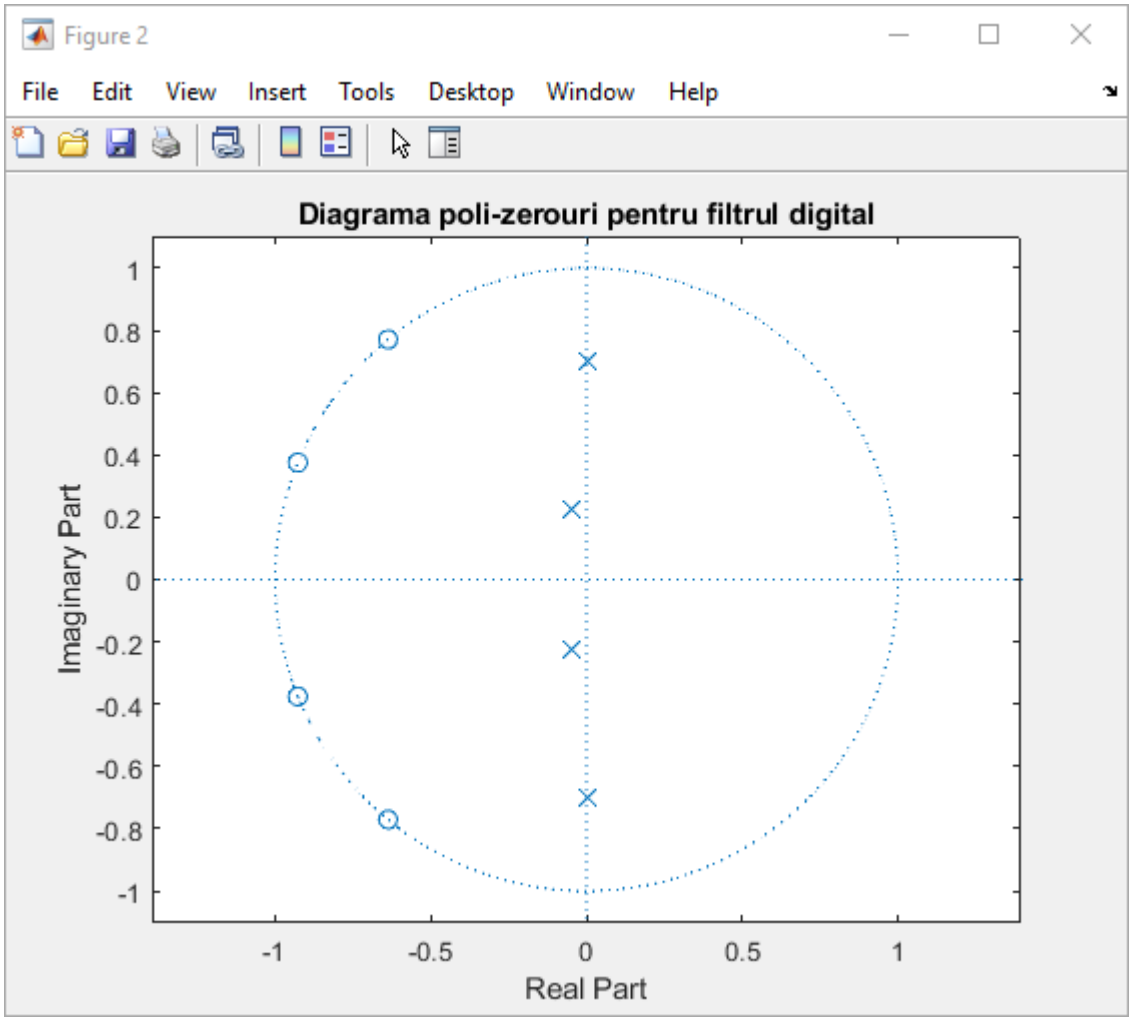
b)

Pentru a determina coeficientii funcției H(z) se folosește funcția cheby2. Aceasta este utilizata pentru a proiecta filtre digitale IIR folosind algoritmul Cebisev tip II. Coeficientii functiei H(z) sunt a si b.

```
% Punctul b
% Proiectează filtrul Chebyshev de tip 2
[b, a] = cheby2(n, Rs, Wn,"low");


% Afiseaza răspunsul în frecvență
figure(1),freqz(b, a),title('Răspuns în frecvență al filtrului digital');
figure(2), zplane(b,a), title('Diagrama poli-zerouri pentru filtrul digital');
```

```
b =

    0.1353    0.4233    0.5904    0.4233    0.1353

a =

    1.0000    0.0869    0.5468    0.0476    0.0264
```





Pentru reprezentarea grafica a raspunsului in frecventa se folosește functia freqz. Aceasta este utilizata pentru oferi informatii despre modul in care filtrul se comporta la diferite frecvente, argumentele acestei functii reprezinta coeficientii functiei filtrului digital H(z). Pentru reprezentarea grafica a pozitiei polilor si zerourilor pentru filtrul digital proiectat am folosit functia zplane.

c)

```
%% Punctul c
%Realizare scalare functiei de transfer
h = impz(b, a); %functia de transfer
k0 = sum(abs(h)); %constanta de scalare
disp(k0);

% Determinare coeficienti obtinuti dupa scalare
if k0 > 1
    b = b/k0;
end
disp(b); %coeficienti B(z) dupa scalare
s0 = 1/k0; %coeficient de scalare

%Determinare coeficient de scalare a semnalului de la intrarea structurii
h0 = impz(1, a); %functia de transfer de la nodul de intrare
k1 = sum(abs(h0)); %constanta de scalare
disp(k1);
s1 = 1/k1; %coeficient de scalare la nod intrare
```

Scalarea dupa regula L1 presupune ca semnalul de intrare este in modul subunitar, scalarea functiei de transfer necesita ca iesirea sistemului sa fie subunitara (în modul), iar daca k0 >1 exista posibilitatea depasirilor la iesire si trebuie scalata functia de transfer cu factorul de scalare.

$$|x(n)| \le 1$$

$$|y(n)| = \left|\sum_{k=0}^{\infty} h(k)x(n-k)\right| \le \sum_{k=0}^{\infty} |h(k)||x(n-k)| \le \sum_{k=0}^{\infty} |h(k)| = k_0$$

$$s_0 = 1/k_0, \quad H_s(z) = s_0 H(z)$$

Pentru a scala functia de transfer, intai am determinat raspunsul la impuls h(n), apoi am calculat coeficientul k0 cu care trebuie facuta scalarea dupa urmatoarea formula:

$$k_0 = \sum_{n=0}^{\infty} |h(n)|$$

Pe prima linie este afisat coeficientul k0, iar pe a doua coeficientii lui b dupa scalare obtinuti prin raportul coeficientilor lui b cu constanta de scalare k0:

```
1.7016

0.0795    0.2488    0.3469    0.2488    0.0795
```

Avand in vedere ca parametrul k0 este mai mare decat 1, inseamna ca exista posibilitatea de depasire la iesire a capacitatii registrelor, asadar coeficientul de scalare s0 este egal cu 1/k0 , adica este egal cu 0.5877, iar functia scalata va fi Hs(z) = s0 * H(z).

Pentru a determina coeficientul de scalare a semnalului de la intrarea structurii se afla intai functia de transfer de la nodul de intrare h0. Constanta de scalare a semnalului de la intrarea structurii este k1=2.2407. Coeficientul de scalare este raportul 1/k1 si este egal cu 0.4463.

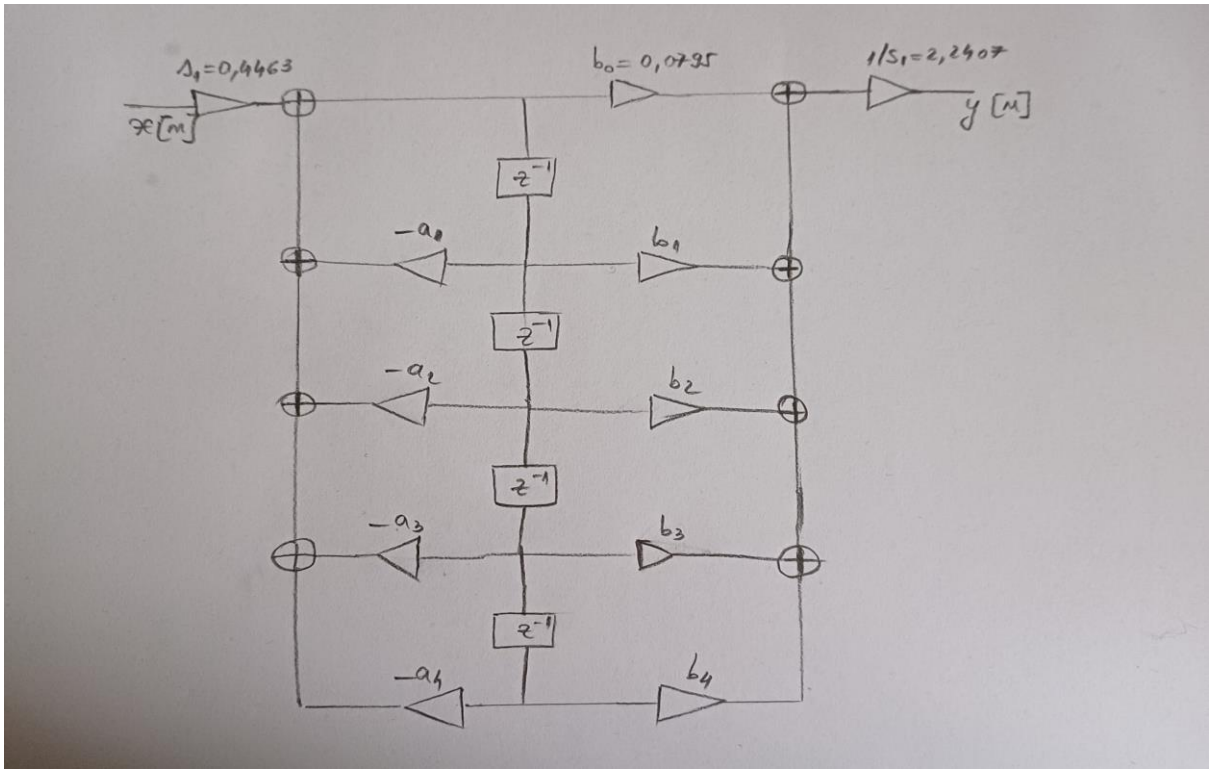
In proiectare si implementare trebuie evaluate efectele reprezentarii semnalelor utilizand un numar finit de biti. In cazul reprezentarii numerelor in formate cu virgula fixa apare o depasire daca rezultatul unei operații aritmetice este de modul supraunitar. Analiza posibilitatii depasirilor comporta doua aspecte:

• analiza functiei de transfer, H(z) și eventual scalarea acesteia astfel incat sa se elimine sau sa se reduca suficient de mult probabilitatea ca semnalul de iesire sa fie de modul supraunitar

• analiza posibilitatii depasirilor in toate nodurile retelei, conducand (daca este necesar) la o scalare a semnalului compensata in final, astfel încat sa nu afecteze realizarea functiei de transfer impuse.

a	[1,0.0869,0.5468,0.0476,0.0264]
b	[0.0795,0.2488,0.3469,0.2488,0.0795]
Fob	5600
Fs	16000
ft	5600
Ftb	2800
h	*28x1 double*
h0	*28x1 double*
k0	1.7016
k1	2.2407
n	4
Rp	1
Rs	40
s0	0.5877
s1	0.4463
Wn	0.7000

d)



2. Partea 2

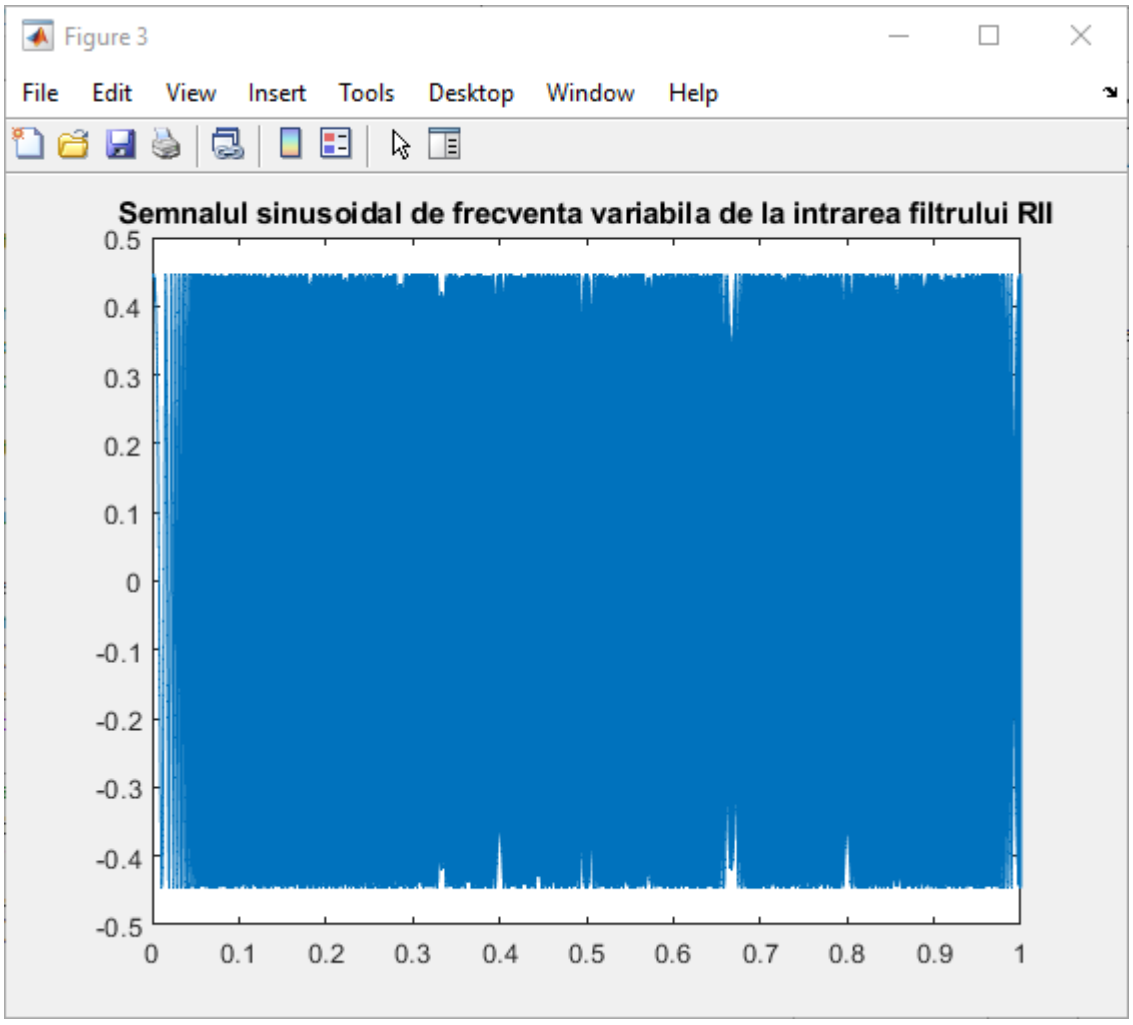
2.2 Cerinta

II. Realizaţi un proiect C pentru StarCore140 care să implementeze filtrul RII proiectat cu structura cerută şi cu coeficienţii obţinuţi în urma scalării funcţiei de transfer şi a semnalului. Aplicaţi la intrarea filtrului proiectat un semnal sinusoidal de frecvenţă variabilă între 0 Hz si $F_s/2$ generat cu ajutorul funcţiei Matlab **chirp** (>> **help chirp**). Durata semnalului generat este de 1 secunda. (t = 0:1/Fs:1).
Reprezentaţi semnalele de la intrarea şi ieşirea filtrului obţinută în MATLAB şi după rularea programului C.

STĂNESCU C.D. Vlad-Constantin	444C	FTJ	Cebâşev2			16000	2800	4400	1	40

Am aplicati la intrarea filtrului un semnal sinusoidal de frecventa variabila între 0 Hz si FS/2 generat cu ajutorul functiei Matlab chirp. Durata semnalului generat notata cu t este de 1 secunda. Apoi am scris vectorul de intrare in fisierul in.dat, folosit pentru a prelua input-ul in Code Warrior.

```
%% Generarea semnalului sinusoidal de frecventa variabila
t = 0: 1/Fs : 1-1/Fs; % durata semnalului generat este de o secunda
in = chirp(t, 0, 1, Fs/2); % generarea semnalului sinusoidal de frecventa variabila
in = in * s1; % scalarea semnalului de intrare in
figure(3), plot(t, in), title('Semnalul sinusoidal de frecventa variabila de la intrarea filtrului RII');
fid=fopen('..\in.dat','w','b'); % crearea si deschiderea fisierului in.dat
fwrite(fid,in.*2^15,'int16'); % scrierea in fisierul in.dat
fclose(fid);
```



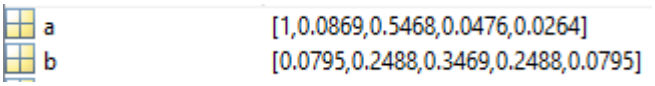
2.3 Prelucrarea semnalului sinusoidal de frecventa variabila in Code Warrior

Pentru realizarea acestei cerinte, am folosit algoritmul ecuatiilor cu diferente finite pentru filtrele IIR prezentat in cadrul laboratorului 5. La baza codului in C stau urmatoarele ecuatii:

$$w(n) = x(n) - a_1w(n-1) - a_2w(n-2) - a_3w(n-3) - a_4w(n-4)$$

$$y(n) = b_0w(n) - b_1w(n-1) - b_2w(n-2) - b_3w(n-3) - b_4w(n-4)$$

In Matlab, am obtinut urmatoarele valori ale coeficientilor a si b dupa scalare:



Am creat vectorul de intrare x si vectorul de iesire y, ce contin cate 160 (DataBlockSize) de elemente, adica din fisierul in.dat vom prelua cate 160 de elemente si le vom prelucra pe rand. De asemenea, input-ul prezinta un total de 16000 de elemente (Fs), deci vom prelucra in total 16000/160 = 100 (BlockLength) blocuri de date.

```
#define DataBlockSize 160 // luam cate 160 de elemente din fisier si le prelucram
#define BlockLength 100 // in total sunt 16000/160 = 100 blocuri de date
// Definirea vectorilor de intrare, respectiv de iesire:
Word16 x[DataBlockSize];
Word16 y[DataBlockSize];
// Vectorii ce contin coeficientii b si a ai filtrului:
Word16 b[]={WORD16(0.0795), WORD16(0.2488), WORD16(0.3469), WORD16(0.2488), WORD16(0.0795)}; // coeficientii b
Word16 a[]={WORD16(0.999), WORD16(0.0869), WORD16(0.5468), WORD16(0.0476), WORD16(0.0264)}; // coeficientii a
Word16 w[5]; // vector de 5 elemente: {w(n-0), w(n-1), w(n-2), w(n-3), w(n-4)}
Word32 suma; // cu ajutorul sau vom parcurge toate elementele din fisierul de intrare
```

In continuare am aplicat algoritmul ecuatiilor cu diferente finite, am creat 2 bucle FOR, prima parcurge cele 100 blocuri de date, iar a doua fiecare set de 160 de elemente continut in fiecare bloc. In variabila suma va fi stocata valoarea x[n] pe 32 de biti ajutorul functiei L_deposit_h(), care incarca un numar pe 16 biti intr-un registru de 32 de biti, pastrand semnul. Folosesc apoi succesiv un numar de 4 functii intrinseci L_msu() (inmultire cu scadere) pentru a obtine treptat ecuatia finala w(n). L_msu(c, a, b) >>> c = c – a*b. Cu ajutorul functiei round() transform valoarea pe 32 de biti inapoi pe 16 biti.

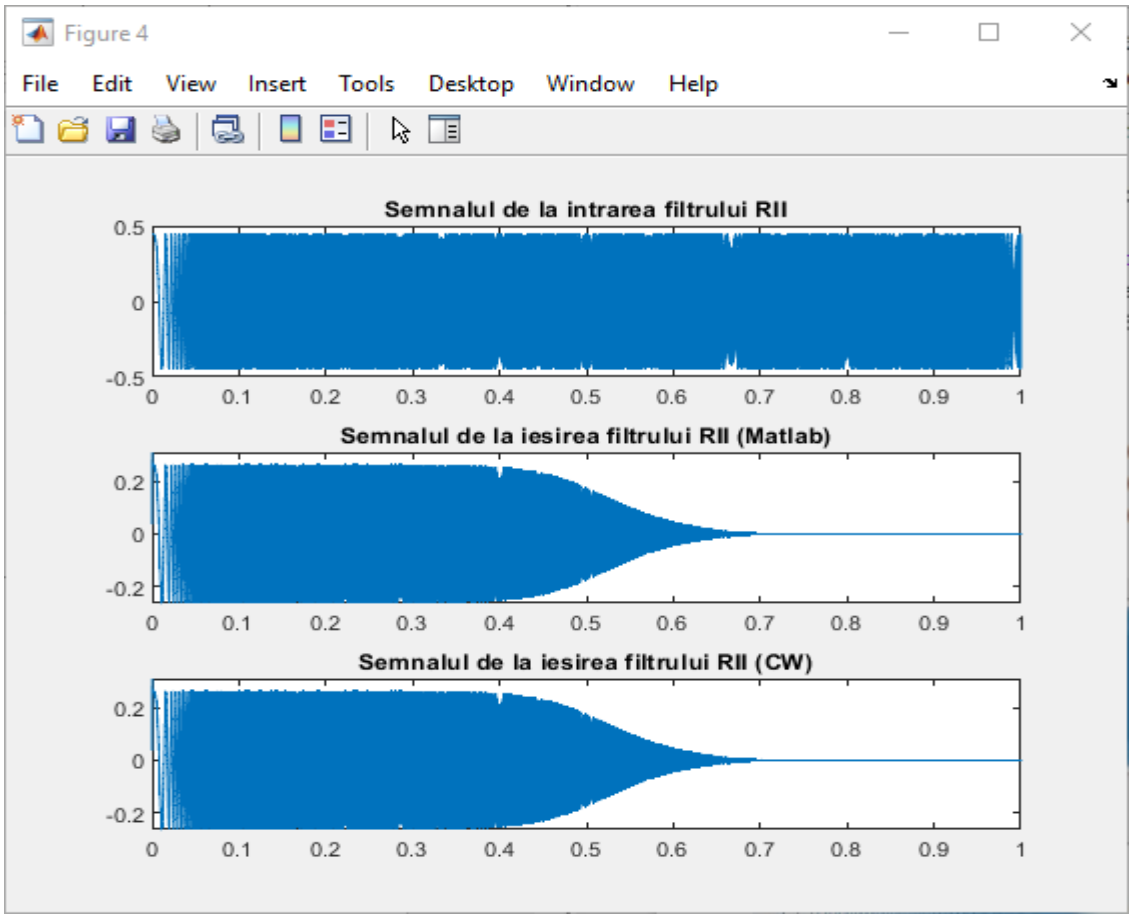
```
int main()
{
    short n,i;
    FILE *fpx,*fpy;
    fpx=fopen("in.dat","r+b");
    if (!fpx)
        printf("\nNu s-a deschis");
    fpy=fopen("out.dat","w+b");
    if (!fpy)
        printf("\nNu s-a deschis");
    for (i=0; i<BlockLength; i++)
    {
        fread(x,sizeof(Word16),DataBlockSize,fpx);
        for (n=0; n<DataBlockSize; n++)
        {
// prima ecuatie diferentiala
            suma = L_deposit_h(x[n]);// suma = x[n]
            suma = L_msu(suma, w[1], a[1]); // suma = x[n] - a1 * w(n-1)
            suma = L_msu(suma, w[2], a[2]); // suma = x[n] - a1 * w(n-1) - a2 * w(n-2)
            suma = L_msu(suma, w[3], a[3]); // suma = x[n] - a1 * w(n-1) - a2 * w(n-2) - a3 * w(n-3)
            suma = L_msu(suma, w[4], a[4]); // suma = x[n] - a1 * w(n-1) - a2 * w(n-2) - a3 * w(n-3) - a4 * w(n-4)
            w[0] = round(suma); // w(n) = suma
```

In cazul celei de-a doua ecuatii diferentiale, variabila suma va lua valoarea w(n) inmultita cu primul coeficient b_0, apoi am folosit succesiv 4 functii intrinseci L_mac() (inmultire cu acumulare) pentru a se obtine treptat ecuatia finala y(n). L_mac(c, a, b) >>> c = c + a*b.

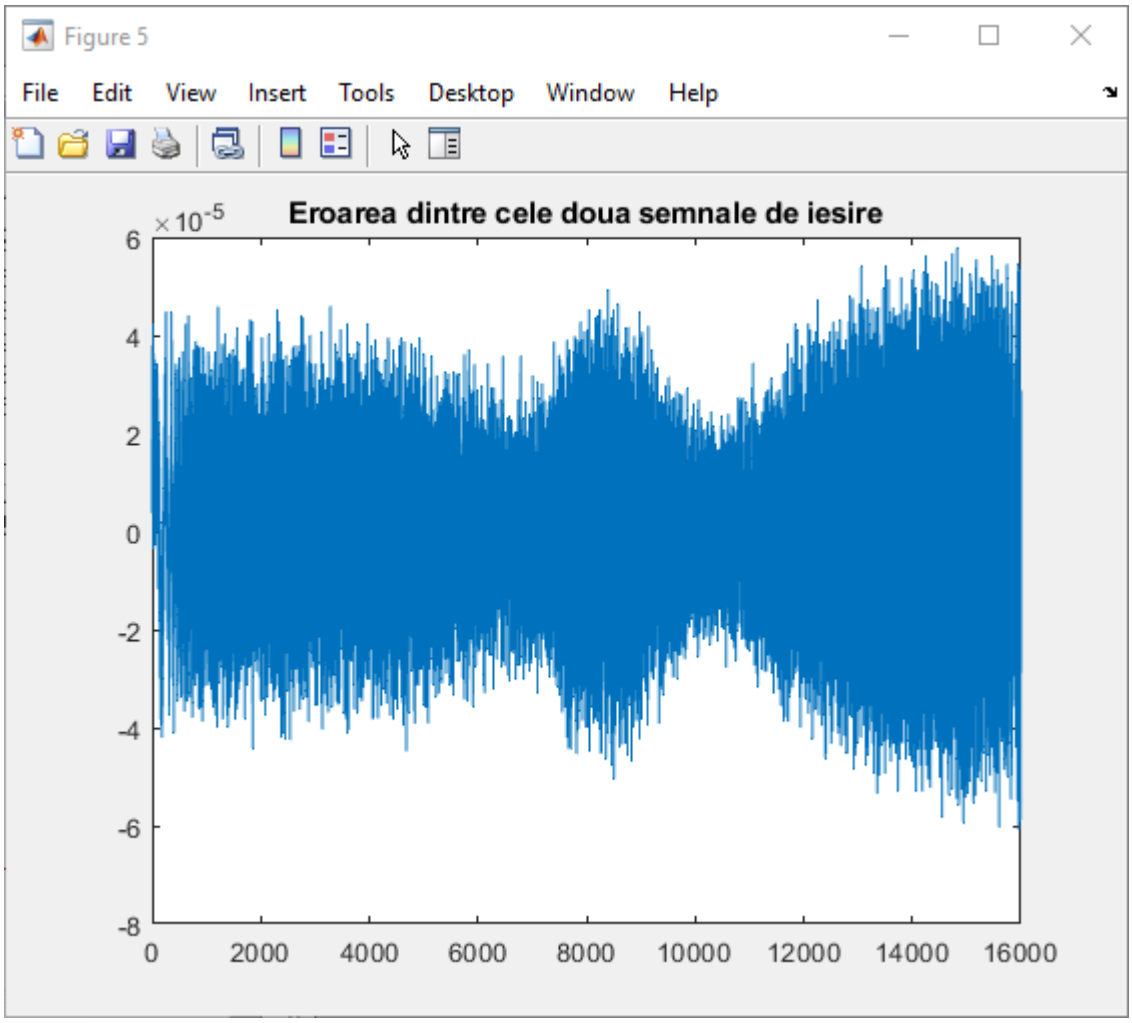
```
// a doua ecuatie diferentiala
            suma = L_mult(w[0], b[0]); // suma = w(n)*bd0
            suma = L_mac(suma, w[1], b[1]); // suma = b0 * w(n) + b1*w(n-1)
            suma = L_mac(suma, w[2], b[2]); // suma = b0 * w(n) + b1*w(n-1) + b2*w(n-2)
            suma = L_mac(suma, w[3], b[3]); // suma = b0 * w(n) + b1*w(n-1) + b2*w(n-2) + b3*w(n-3)
            suma = L_mac(suma, w[4], b[4]); // suma = b0 * w(n) + b1*w(n-1) + b2*w(n-2) + b3*w(n-3) + b4*w(n-4)
// scrierea in vectorul de iesire y:
            y[n] = round(suma); // y(n) = suma
            w[4] = w[3];
            w[3] = w[2];
            w[2] = w[1];
            w[1] = w[0];
        }
        fwrite(y,sizeof(Word16),DataBlockSize,fpy);
    }
    fclose(fpx);
    fclose(fpy);
    return 0;
}
```

2.4 Semnalul de la intrare/iesire + eroarea (Matlab/Code Warrior)

```
%% Reprezentarea semnalului de la intrarea si iesirea filtrului din Matlab / CW
out_m = filter(b, a, in); % semnalul de la iesirea filtrului din Matlab
fid=fopen('..\out.dat','r','b');
out_cw = fread(fid,Fs,'int16');
out_cw = out_cw/(2^15); % semnalul de la iesirea filtrului din CW
fclose(fid);
figure(4),
subplot(3,1,1), plot(t, in), title('Semnalul de la intrarea filtrului RII'),
subplot(3,1,2), plot(t, out_m), title('Semnalul de la iesirea filtrului RII (Matlab)');
subplot(3,1,3), plot(t, out_cw), title('Semnalul de la iesirea filtrului RII (CW)');
```



```
%% Calculul si reprezentarea grafica a erorii
error = out_m - out_cw';
e = 0 : (length(out_m) - 1);
figure(5), plot(e, error), title('Eroarea dintre cele doua semnale de iesire');
```



Observam ca eroarea este centrata in jurul valorii $x*10^{-5}$, deci extrem de mica, asadar indiferent de programul utilizat, rezultatele nu sunt influentate.