UNIVERSITATEA TEHNICĂ "GH ASACHI" IAȘI FACULTATEA AUTOMATICĂ ȘI CALCULATOARE
SPECIALIZAREA CALCULATOARE ȘI TEHNOLOGIA INFORMAȚIEI
DISCIPLINA ACHIZIȚIA ȘI PRELUCRAREA DATELOR

Achiziția și prelucrarea datelor
Proiect

Profesor Coordonator,
Lupu Robert

Student,
Hriscu Cornelia-Ștefana
GRUPA 1310A

Iași, 2022

CERINȚELE PROIECTULUI

➢ ETAPA I

- Achiziția și prelucrarea în timp a semnalului audio asignat
- Afișarea semnalului si a parametrilor caracteristici (valorile minime și maxime, dar și indexii corespunzători acestora, medie, dispersie, mediană, zero-crossing, histogramă)
- Filtrarea semnalului în domeniul timp prin mediere (pe 16/32 eșantioane) și element de ordin I
- Reprezentarea anvelopei și a derivatei semnalului;
- Afișarea rezultatelor de mai sus pe întreg semnalul, dar și pe secunde

➢ ETAPA II

- Prelucrarea în frecvență a semnalului
- Ferestruirea semnalului prin metodele Blackman și Triunghiulară
- Filtrarea rezultatului ferestruirii folosind un filtru Notch sau Eliptic trece-jos cu frecvența de tăiere de 1500Hz
- Reprezentarea spectrului pe o anumită fereastră de timp din secunda aleasă (numărul de puncte este o putere a lui 2)

FIȘIERUL UTILIZAT

Fișierul audio utilizat în cadrul proiectului este reprezentat prin înregistrarea unui ton de apel de 6 secunde.

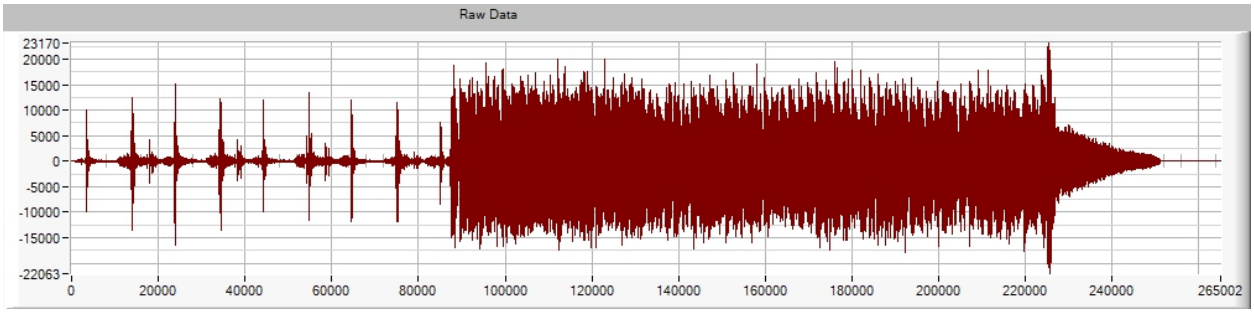
MEDIUL DE DEZVOLTARE

LabWindows/CVI este o platformă pentru dezvoltare de software cu orientare pe aplicaţii de instrumentaţie, având ANSI C ca limbaj. CVI ne pune la dispoziţie un mediu interactiv de dezvoltare a unor aplicaţii pentru sistemul de operare Windows.

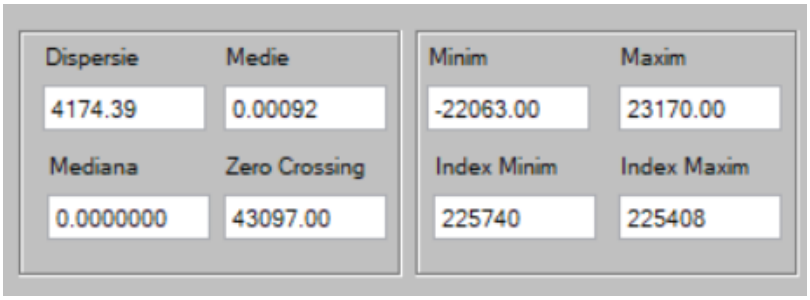
ETAPA I – ANALIZA ÎN DOMENIUL TIMP

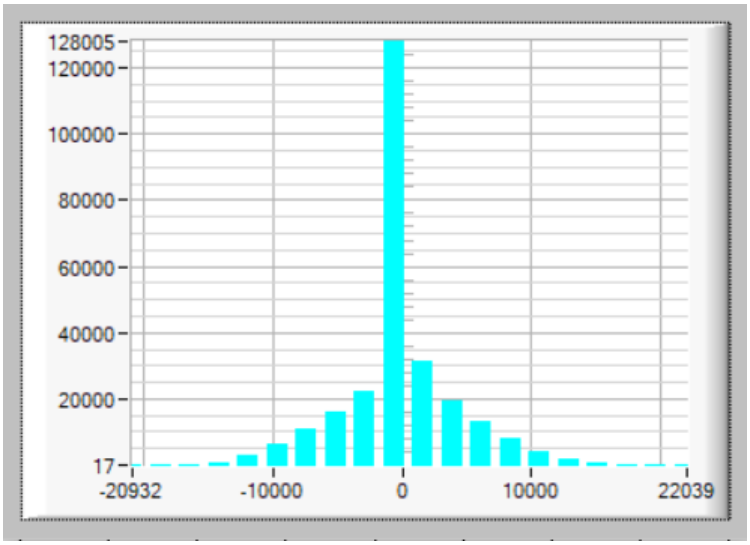
Inițial s-a achiziționat semnalul prin folosirea mediului de dezvoltare online COLAB unde s-a convertit semnalul dintr-un fișier WAV într-un fișier text în urma rulării unui cod scris în limbajul de programare Python.

SEMNALUL ACHIZIȚIONAT rezultat este reprezentat în următoarea imagine (afișare pe un control de tip Graph):



CARACTERISTICILE acestui semnal sunt:





Se observă că **mediana** este 0. Aceasta rezultă din faptul că semnalul achiziționat este simetric.

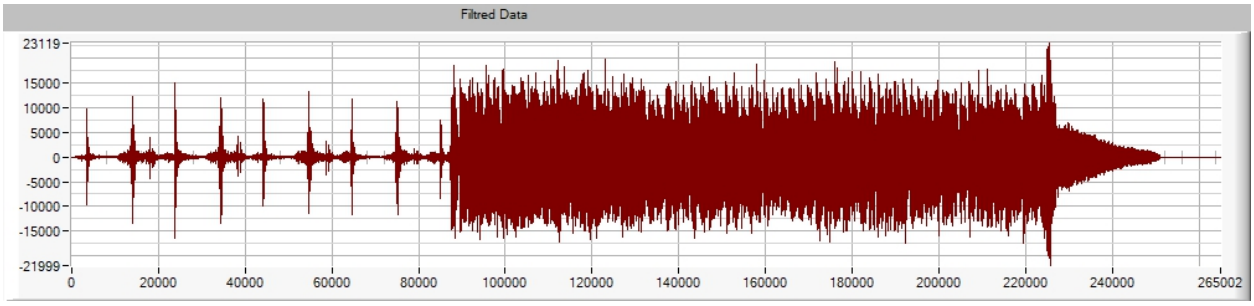
Media evidențiază în jurul cărei valori din semnal se aglomerează cele mai multe date din vector (aproximativ 0). Același aspect se observă și din **histogramă** (împărțită în 20 de intervale).

Dispersia nefiind mare reiese că valorile nu sunt foarte împrăștiate.

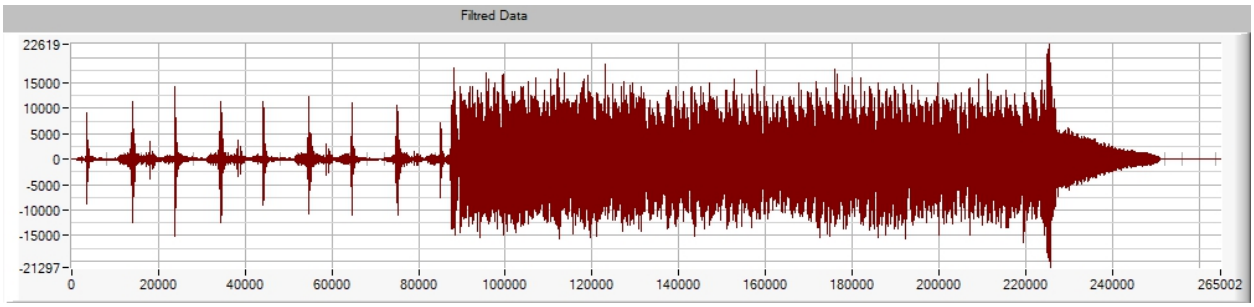
Pe interfață există un control de tip RING care permite comutarea între cele două tipuri de filtrări. Opțiunile care pot fi alese sunt:

I. FILTRARE DE ORDIN I

a) Alpha = 0.8



b) Alpha = 0.5



c) Implementare

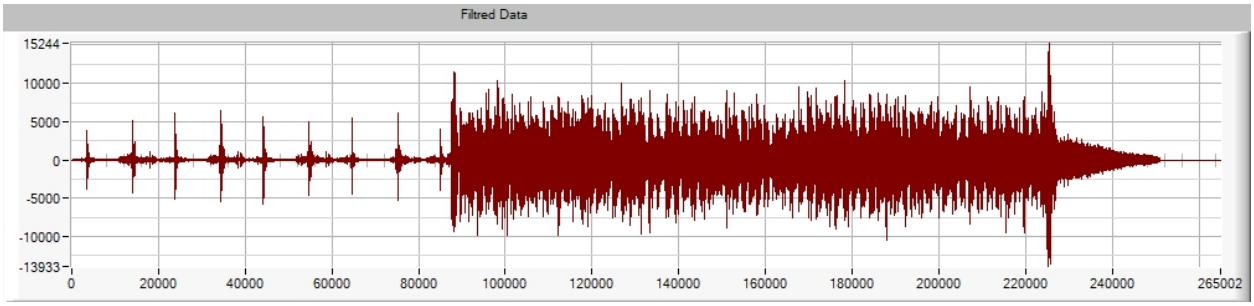
```
double * filtrare_ordinI (int numarpuncte, double * data)
{
    double * filtru_final = (double *) malloc (numarpuncte * sizeof(double));

    filtru_final[0] = data[0];

    for(int i = 1; i < numarpuncte; ++i)
    {
        filtru_final[i] =  (1 - alpha) * filtru_final[i - 1] + alpha * data[i];
    }

    return filtru_final;
}
```

Acest tip de filtrare s-a aplicat cu ajutorul formulei *filt[i]=(1-alpha)*filtru_final[i-1]+alpha*data[i]*, unde *data* este vectorul care contine valorile semnalului audio iar *filtru_final* este un vector care conține valorile filtrate.

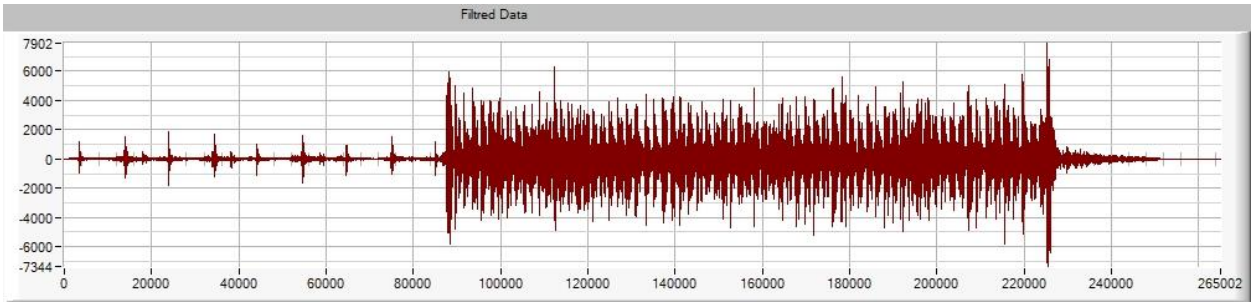
Valoare parametrului alpha se poate schimba prin intermediul unui control de tip NUMERIC din interfață, în timp real. Alpha poate lua valori doar în intervalul (0,1).

II. FILTRARE PRIN MEDIERE

a) 16 eșantioane



b) 32 eșantioane



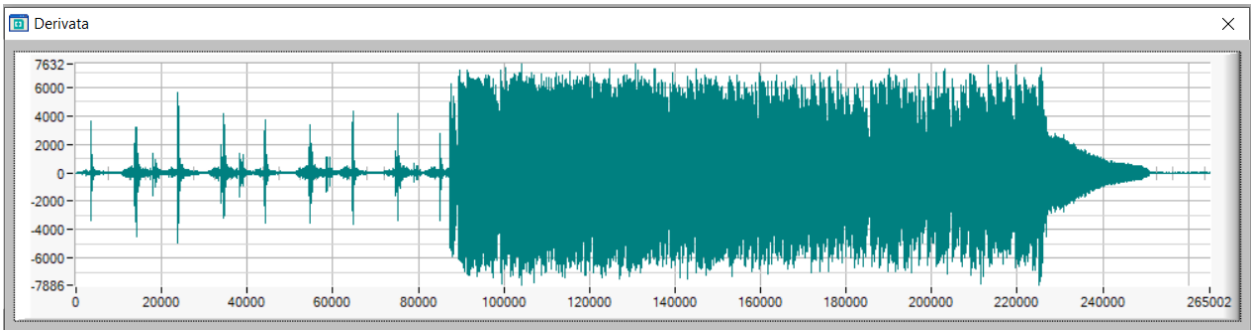
c) Implementare

```
double * filtrare_mediere (int numarpuncte, double * data)
{
    double * filtru_final = (double *) malloc (numarpuncte * sizeof(double));

    //copiez primele esantioane-1 valori -> primele 16 sau 32 de valori
    for (int i = 0 ; i < esantioane - 2 ; ++i){
        filtru_final[i] = data[i];
    }

    //de la esantioane incep sa fac filtrarea
    for(int i = esantioane ; i < numarpuncte ; ++i){
        double sum = 0.0;
        for(int j =  i - esantioane; j < i; ++j){
            sum += data[j];
}
        filtru_final[i] = sum / esantioane;
    }

    return filtru_final;
}
```

Acest filtru este aplicat pentru a scăpa de zgomotul din semnalul inițial. Numărul de eșantioane pe care se face filtrare se obține printr-un control de tip NUMERIC și poate lua doar două valori, 16 sau 32. S-a aplicat formula $filtru\_final[i] = \frac{\sum_{j=i-eșantioane}^{i-1} data[j]}{eșantioane}$, unde filtru_final este vectorul de output, data este inputul, iar eșantioane este 16 sau 32.

Inițial se copiază primele “eșantioane-1” de valori, deoarece, altfel s-ar fi accesat valori dinaintea indexului 0, ceea ce nu ar fi fost posibil (relație de recurență). Apoi, începând cu elementul de pe poziția 16/32 începe filtrarea propriu-zisă. Valoarea de la indexul curent este suma (variabila sum) tuturor celor 16/32 de valori anterioare ei supra numărul de eșantioane, 16/32 (variabila eșantioane).

III. DERIVATA SEMNALULUI

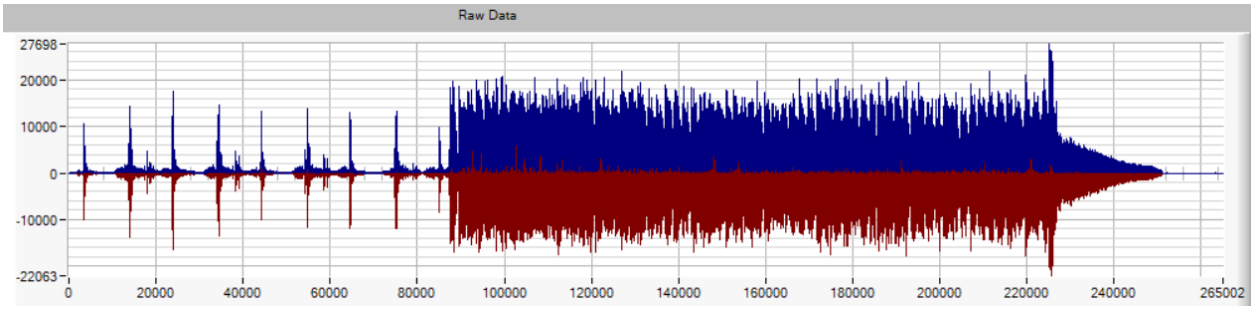


Implementare

```
/*****************************************************************************
    Functia de callback a butonului Derivata
*****************************************************************************/
int CVICALLBACK On_Derivata (int panel, int control, int event,
                             void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:
            panelDerivata = LoadPanel(1, "lucru.uir", PANEL_DERI);
            InstallPopup(panelDerivata);

            double* derivata = (double *) malloc (sizeof(double)*npoints);
            DifferenceEx(waveData, npoints, 1.0, waveData + 1, 1, waveData + npoints - 1, 1, DIFF_SECOND_ORDER_CENTRAL, derivata);

            PlotY(panelDerivata, PANEL_DERI_GRAPH_DERIVATA, derivata, npoints, VAL_DOUBLE,VAL_THIN_LINE,VAL_EMPTY_SQUARE,VAL_SOLID,VAL_CONNECTED_POINTS, VAL_DK_CYAN);


            break;
    }
    return 0;
}
```

Derivata semnalului în domeniul timp s-a folosit pentru a vedea rata de variație, cu alte cuvinte, derivata semnalului nostru în punctul i reprezintă panta tangentei la grafic în punctul i. Pentru implementare s-a folosit funcția DifferenceEx din CVI cu parametrul DIFF_SECOND_ORDER_CENTRAL. Aceasta va fi afișată pe un panou separat, pe un control de tip GRAPH, la apasarea butonului „Derivata”.

IV. ANVELOPA SEMNALULUI



Implemntare

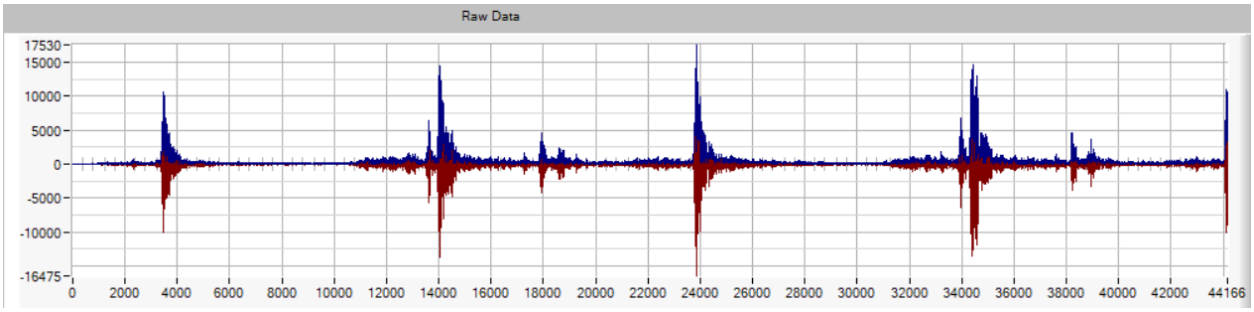
```
int CVICALLBACK On_Avelopa (int panel, int control, int event,
                            void *callbackData, int eventData1, int eventData2)
{
    switch (event)
    {
        case EVENT_COMMIT:
            //panelAnvelopa = LoadPanel(1, "lucru.uir", PANEL_ANVE);
            //InstallPopup(panelAnvelopa);

            //DeleteGraphPlot(panelAnvelopa, PANEL_ANVE_GRAPH_ANVELOPA, -1, VAL_IMMEDIATE_DRAW);
            am_apasat_anvelopa = 1;

            anvelopa = (double*)malloc(sizeof(double)*npoints);

            FileToArray("anvelopadata.txt", anvelopa, VAL_DOUBLE, npoints, 1, VAL_GROUPS_TOGETHER, VAL_GROUPS_AS_COLUMNS,VAL_ASCII);

            anvelopapersec = (double**) malloc (sizeof(double*)*intervalesec);
            for (int i = 0; i < intervalesec; ++i)
            {
                anvelopapersec[i] = (double*) malloc(sizeof (double) * npunctepersec); //alocare memorie coloane
            }

            //construirea vectorului cu secunde
            int count = 0; //numar de coloana -> adica in a cata coloana sunt
            for(int i = 0; i < intervalesec; ++i) //parcurg liniile -> adica fiecare secunda 0->1; 1->2 etc.
            {
                for(int j = npunctepersec * i; j < npunctepersec * (i + 1); ++j) // parcurg vectorul initial(waveData) pentru a lua secundele din el
                {
                    anvelopapersec[i][count] = anvelopa[j];
                    count++;
                }
                count = 0;
            }

            /*
            for(int i = 0; i < intervalesec; ++i)
            {
                for(int count = 0; count < npunctepersec; count++)
                {
                    printf("%lf ", anvelopapersec[i][count]);
                }

            }
            */
            //PlotY(panel, PANEL_GRAPH_RAW_DATA, waveData, npoints, VAL_DOUBLE, VAL_THIN_LINE, VAL_EMPTY_SQUARE, VAL_SOLID, VAL_CONNECTED_POINTS, VAL_MAGENTA); //semnalul waveData
            PlotY(panel, PANEL_GRAPH_RAW_DATA, anvelopa, npoints, VAL_DOUBLE,VAL_THIN_LINE,VAL_EMPTY_SQUARE,VAL_SOLID,VAL_CONNECTED_POINTS, VAL_DK_BLUE); //anvelopa
            break;
    }
    return 0;
```

Anvelopa semnalului (cea cu albatru închis) este afișată pe același control de tip GRAPH cu semnalul inițial pentru a se observa înfășurarea acestuia. Funcția pentru aceasta a fost construită pas cu pas, alegând toate maximele semnalului.

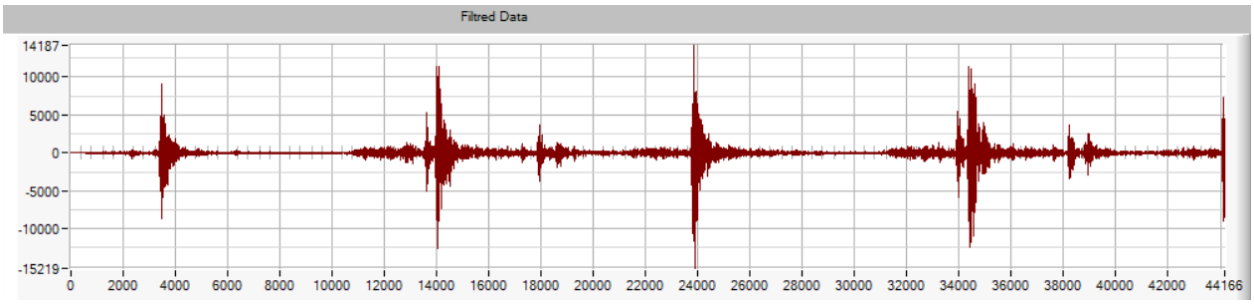
Pentru **divizarea în secunde** s-a folosit un pointer la pointer, în care s-a stocat, la apăsarea butonului de LOAD, toate valorile fiecărei secunde. Accesarea se face cu ajutorul unei variabile globale nextsec care permite navigarea împreună cu controalele PREV și NEXT din interfață. La secunda 0 pe interfață se va afișa întreg semnalul, urmând ca de la 1 la 6 să se afișeze secundele.

Se vor atașa imagini pentru prima secundă trecută prin fiecare aspect discutat mai sus:

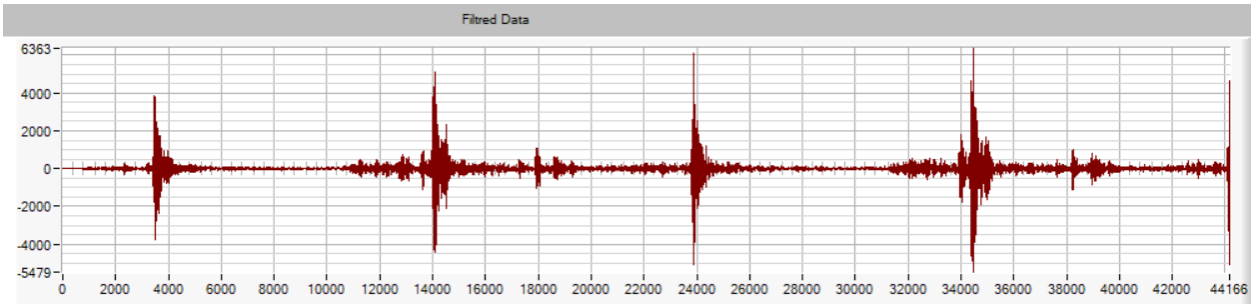
- Secunda inițială și anvelopa



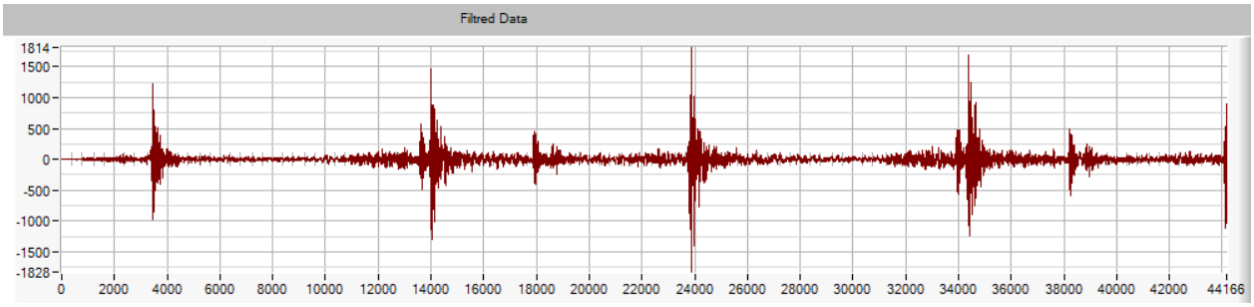
- Filtrarea de ordin întâi cu alpha = 0.5



- Filtrarea prin mediere pe 16 eșantioane



- Filtrarea prin mediere pe 32 de eșantioane



ETAPA II– ANALIZA ÎN DOMENIUL FRECVENȚĂ

Cu ajutorul unui control de pe interfață de tip SWITCH se va realiza comutarea între două paneluri care conțin prelucrarea în domeniul timp și frecvență.

Informațiile de interes pentru cel de-al doilea panou sunt ferestruirea, filtrarea și afișarea spectrului. Toate aceste operații se vor realiza, ca și la prima etapă, pe secunde, dar doar pe un anumit număr de puncte, putere a lui 2, care se va prelua de pe interfată. La activarea unui TOGGLE BUTTON, TIMER-ul va porni și operațiile vor fi aplicate succesiv până la finalul acesteia. În cazul în care pentru ultima parte din secundă nu avem destule eșantioane pentru fereastră, atunci se va face ZERO PADDING pentru completare

Pe interfață sunt prezente două SWITCHURI pentru comutarea între fereastra BLACKMAN sau TRIANGLE, respectiv între filtrul ELIPTIC trece-jos cu frecvența de tăiere de 1500Hz sau NOTCH.

FRESTRUIRE ȘI SPECTRU

Înainte de calcularea spectrului am aplicat una din cele două tipuri de ferestre pentru a limita scurgerea spectrală și pentru a putea evidenția liniile spectrale de frecvențe apropriate (duce la pierderea semnalului la capetele ferestrei). În cadrul calculului spectrului am ales tipurile de ferestre folosite în proiect.

Funcțiile ferestrelor elimină discontinuitățile de la capetele intervalului de timp aferent eșantioanelor.

- ### BLACKMAN

Implementare

```
//FERESTRUIRE + SPECTRU FERESTRUIRE
double * vectFerestruit = (double *) malloc (sizeof(double) * punctefereastra);

if(valfereastra == 0)
{
    //BlackMan
    Copy1D(Fereastra, punctefereastra, vectFerestruit);
    BkmanWin(vectFerestruit, punctefereastra);
    DeleteGraphPlot(panel, PANEL_FREQ_GRAPH_FERESTRUIRE, -1, VAL_IMMEDIATE_DRAW );
    PlotY(panel, PANEL_FREQ_GRAPH_FERESTRUIRE, vectFerestruit , punctefereastra, VAL_DOUBLE, VAL_THIN_LINE, VAL_EMPTY_SQUARE, VAL_SOLID, VAL_CONNECTED_POINTS, VAL_DK_CYAN);
    Spectru(punctefereastra, vectFerestruit, 1);

    //Salvare
    sprintf(fileName,"Imagine_semnal_ferestruit_blackman_secunda_%d_%d-%d.jpg", valsecunda, parcurgesecunda, parcurgesecunda + punctefereastra);
    GetCtrlDisplayBitmap(panel, PANEL_FREQ_GRAPH_FERESTRUIRE, 1, &imghandle);
    SaveBitmapToJPEGFile(imghandle, fileName, JPEG_PROGRESSIVE, 100);
    //Salvare
    sprintf(fileName,"Spectru_ferestruit_bkman_secunda_%d_%d-%d.jpg", valsecunda, parcurgesecunda, parcurgesecunda + punctefereastra);
    GetCtrlDisplayBitmap(panel, PANEL_FREQ_GRAPH_SPECTRU_1, 1, &imghandle);
    SaveBitmapToJPEGFile(imghandle, fileName, JPEG_PROGRESSIVE, 100);
}
```

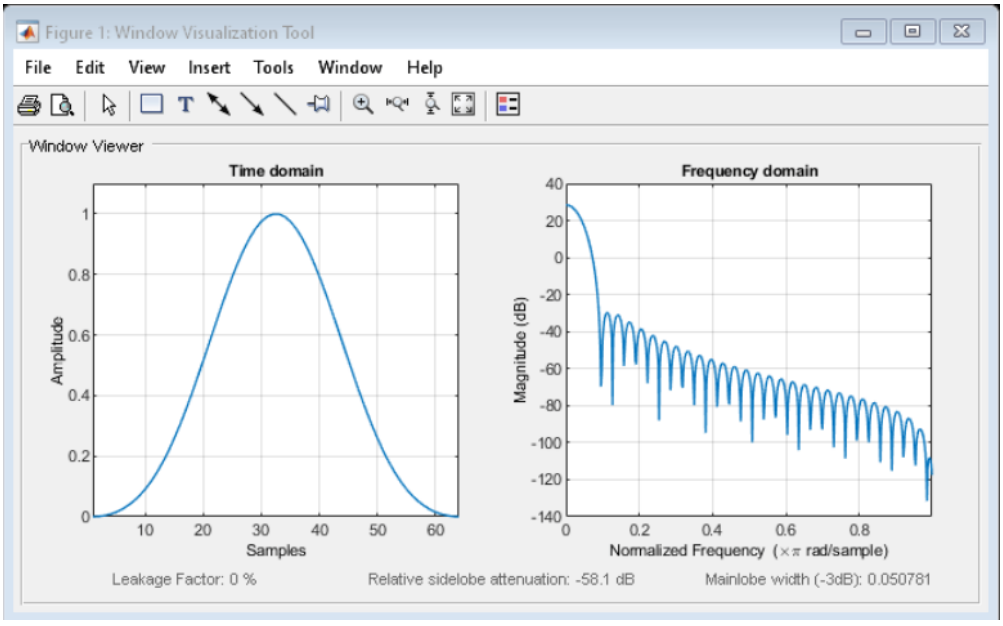
Următoarea ecuație definește fereastra **Blackman** de lungime N:

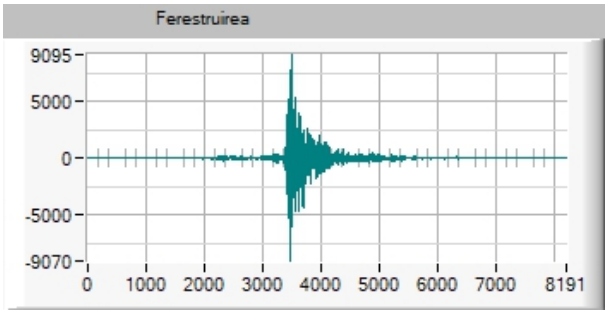
$$w(n)=0.42-0.5\cos_{(2\pi nL-1)}+0.08\cos_{(4\pi nL-1)},\quad 0\le n\le M-1$$

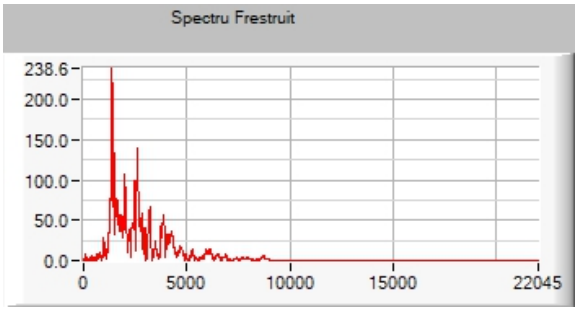
unde M este N/2 când N este par și (N + 1)/2 când N este impar.

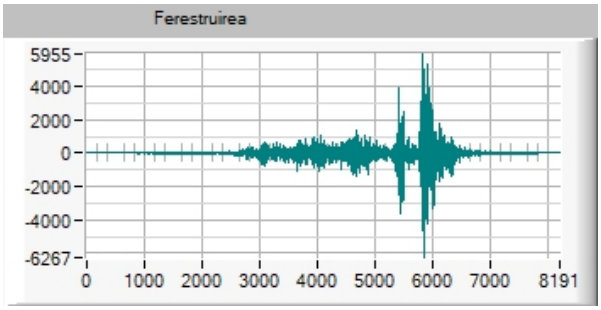
În cazul simetric, a doua jumătate a ferestrei Blackman, M ≤ n ≤ N – 1, se obține prin reflectarea primei jumătăți în jurul punctului de mijloc. Opțiunea simetrică este metoda preferată atunci când se utilizează o fereastră Blackman în designul filtrului FIR.

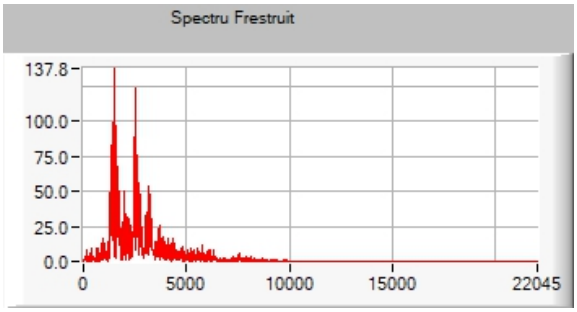
Fereastra Blackman periodică este construită prin extinderea lungimii ferestrei dorite cu un eșantion la N + 1, rezultând construirea unei ferestre simetrice și îndepărtarea ultimului eșantion. Versiunea periodică este metoda preferată atunci când se utilizează o fereastră Blackman în analiza spectrală, deoarece transformata Fourier discretă presupune extinderea periodică a vectorului de intrare.

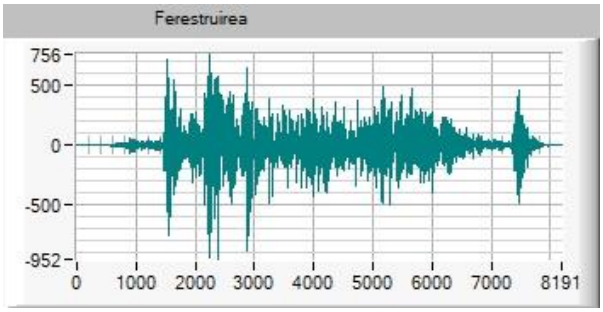


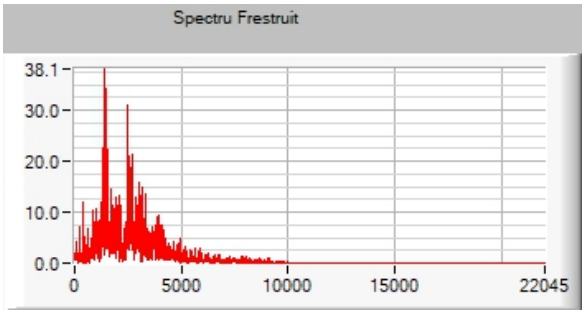

Ferestruirea
9095
5000
0
-5000
-9070
0 1000 2000 3000 4000 5000 6000 7000 8191

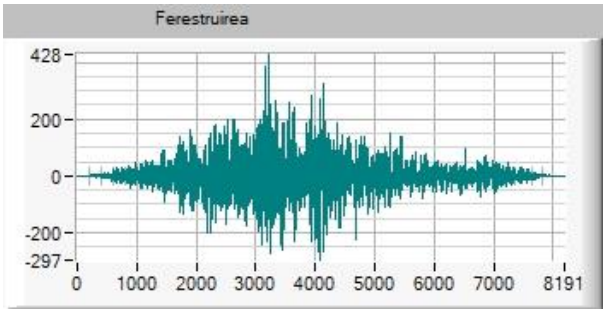

Spectru Frestruit
238.6
200.0
150.0
100.0
50.0
0.0
0 5000 10000 15000 22045

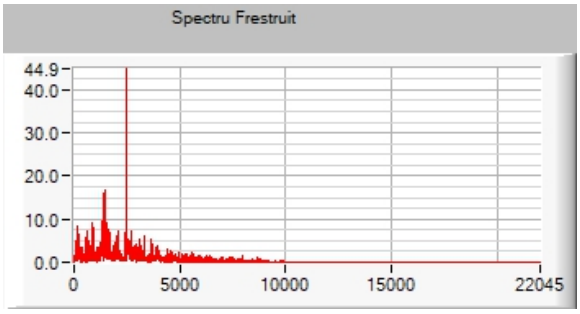

Ferestruirea
5955
4000
2000
0
-2000
-4000
-6267
0 1000 2000 3000 4000 5000 6000 7000 8191

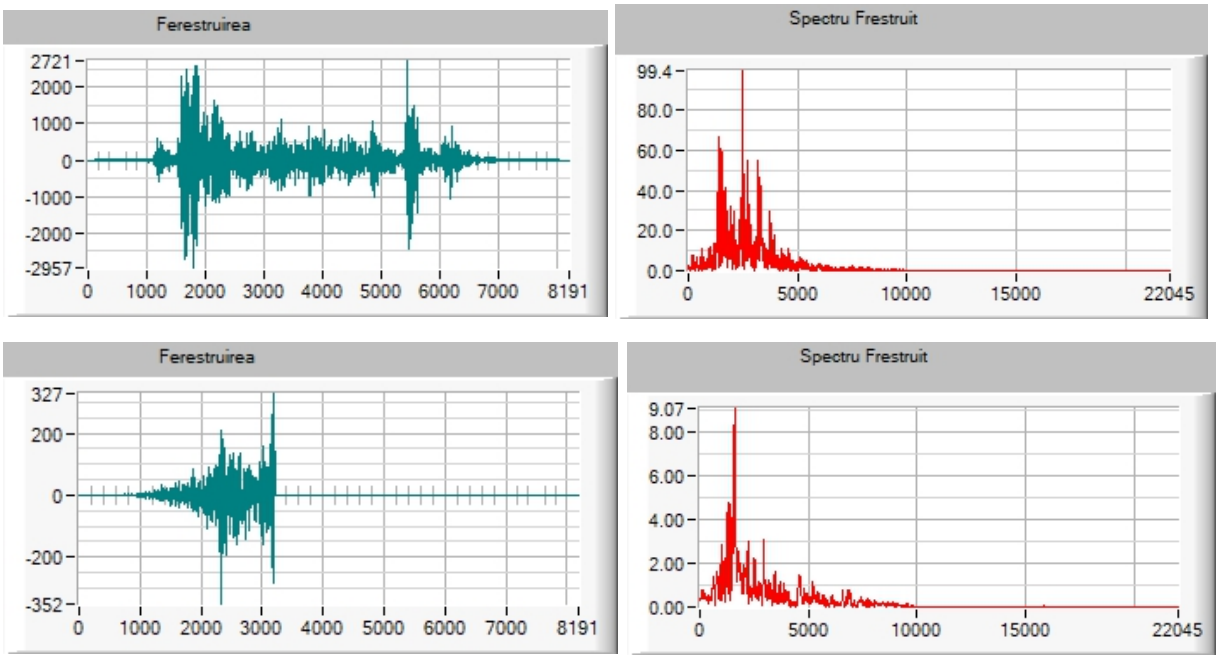

Spectru Frestruit
137.8
100.0
75.0
50.0
25.0
0.0
0 5000 10000 15000 22045


Ferestruirea
756
500
0
-500
-952
0 1000 2000 3000 4000 5000 6000 7000 8191


Spectru Frestruit
38.1
30.0
20.0
10.0
0.0
0 5000 10000 15000 22045


Ferestruirea
428
200
0
-200
-297
0 1000 2000 3000 4000 5000 6000 7000 8191


Spectru Frestruit
44.9
40.0
30.0
20.0
10.0
0.0
0 5000 10000 15000 22045



- **TRIANGLE**

  Implementare

```
else
{
    //Triunghiular
    Copy1D(Fereastra, punctefereastra, vectFerestruit);
    TriWin(vectFerestruit, punctefereastra);
    DeleteGraphPlot(panel, PANEL_FREQ_GRAPH_FERESTRUIRE, -1, VAL_IMMEDIATE_DRAW );
    PlotY(panel, PANEL_FREQ_GRAPH_FERESTRUIRE, vectFerestruit , punctefereastra, VAL_DOUBLE, VAL_THIN_LINE, VAL_EMPTY_SQUARE, VAL_SOLID, VAL_CONNECTED_POINTS, VAL_DK_CYAN)
    Spectru(punctefereastra, vectFerestruit, 1);

    //Salvare
    sprintf(fileName,"Imagine_semnal_ferestruit_triunghiular_secunda_%d_%d-%d.jpg", valsecunda, parcurgesecunda, parcurgesecunda + punctefereastra);
    GetCtrlDisplayBitmap(panel, PANEL_FREQ_GRAPH_FERESTRUIRE, 1, &imghandle);
    SaveBitmapToJPEGFile(imghandle, fileName, JPEG_PROGRESSIVE, 100);
    //Salvare
    sprintf(fileName,"Spectru_ferestruit_tri_secunda_%d_%d-%d.jpg", valsecunda, parcurgesecunda, parcurgesecunda + punctefereastra);
    GetCtrlDisplayBitmap(panel, PANEL_FREQ_GRAPH_SPECTRU_1, 1, &imghandle);
    SaveBitmapToJPEGFile(imghandle, fileName, JPEG_PROGRESSIVE, 100);
}
```

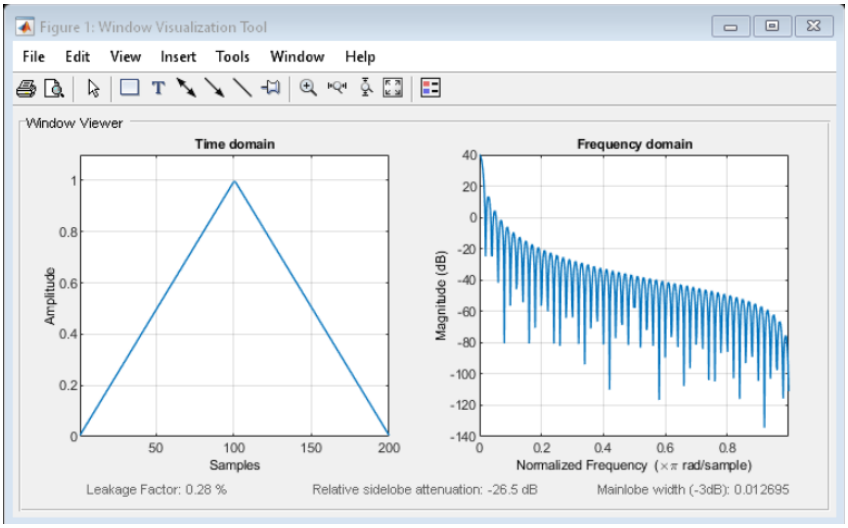
Coeficienții ferestrei triunghiulare sunt:

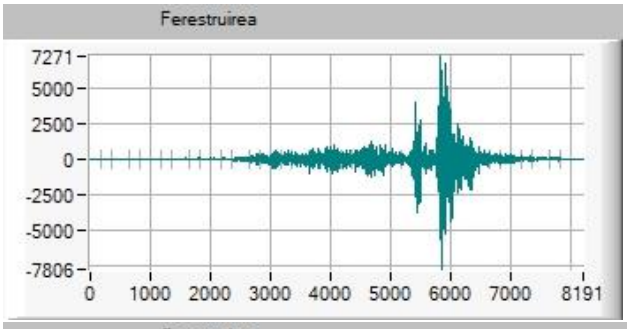
- Pentru L impar

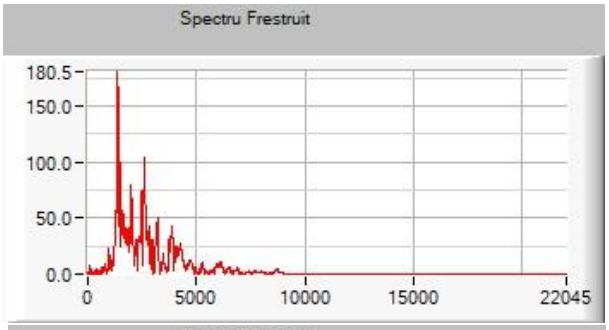
$$w(n) = \begin{cases} \dfrac{2n}{L+1} & 1 \le n \le \dfrac{L+1}{2} \\ 2 - \dfrac{2n}{L+1} & \dfrac{L+1}{2} + 1 \le n \le L \end{cases}$$

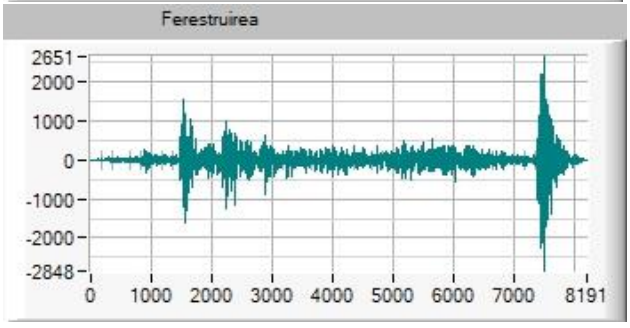
- Pentru L par

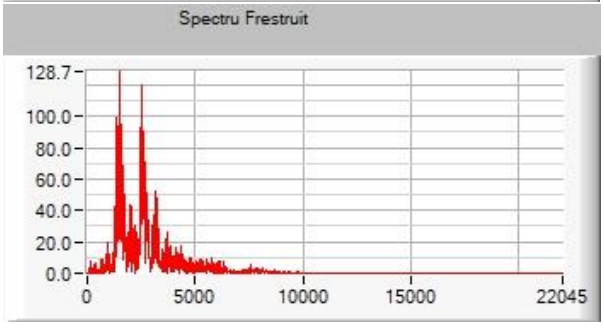
$$w(n) = \begin{cases} \dfrac{(2n-1)}{L} & 1 \le n \le \dfrac{L}{2} \\ 2 - \dfrac{(2n-1)}{L} & \dfrac{L}{2} + 1 \le n \le L \end{cases}$$

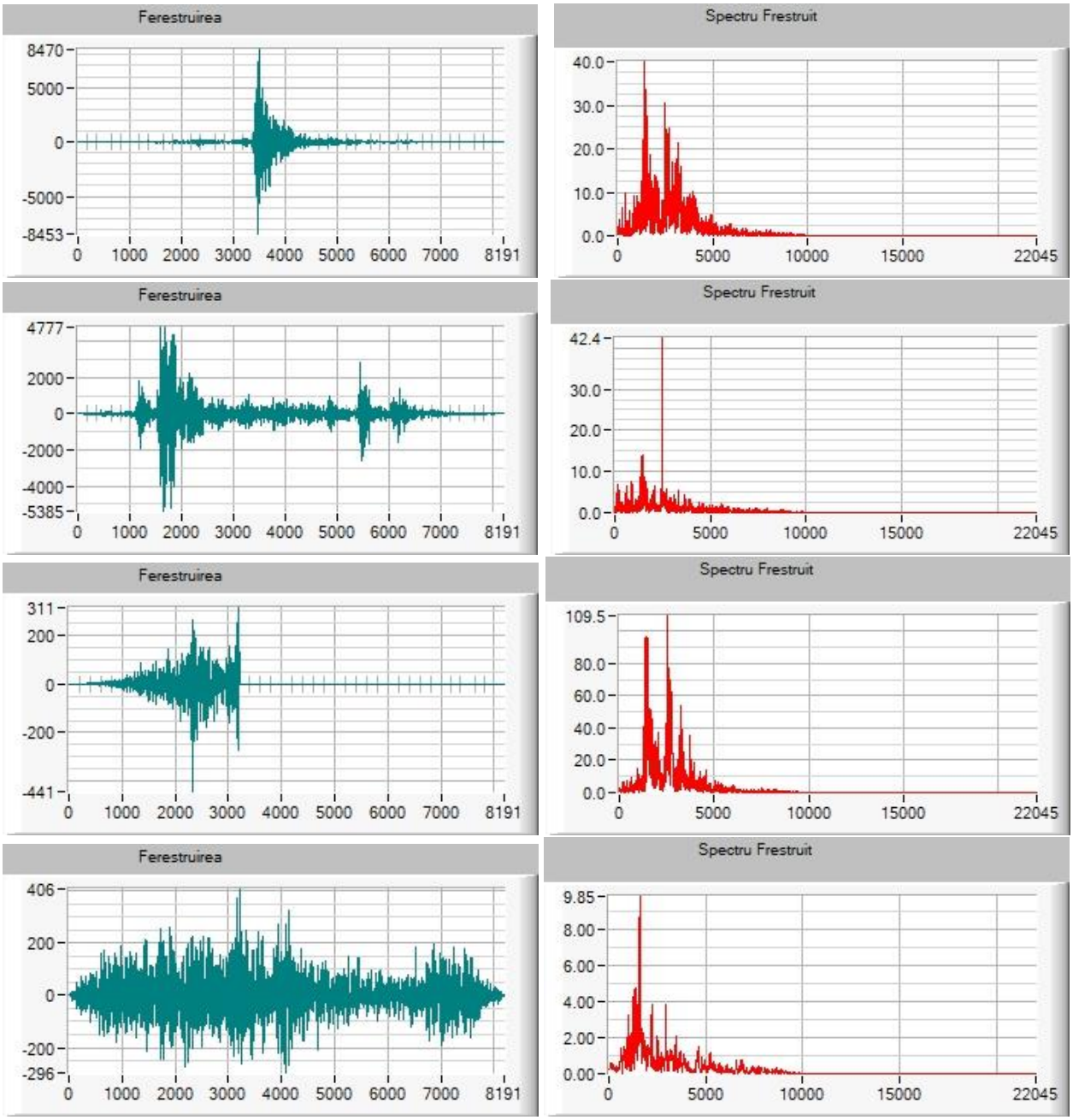











Ferestruirea
8470
5000
0
-5000
-8453
0 1000 2000 3000 4000 5000 6000 7000 8191
Spectru Frestruit
40.0
30.0
20.0
10.0
0.0
0 5000 10000 15000 22045
Ferestruirea
4777
2000
0
-2000
-4000
-5385
0 1000 2000 3000 4000 5000 6000 7000 8191
Spectru Frestruit
42.4
30.0
20.0
10.0
0.0
0 5000 10000 15000 22045
Ferestruirea
311
200
0
-200
-441
0 1000 2000 3000 4000 5000 6000 7000 8191
Spectru Frestruit
109.5
80.0
60.0
40.0
20.0
0.0
0 5000 10000 15000 22045
Ferestruirea
406
200
0
-200
-296
0 1000 2000 3000 4000 5000 6000 7000 8191
Spectru Frestruit
9.85
8.00
6.00
4.00
2.00
0.00
0 5000 10000 15000 22045

FILTRARE ȘI SPECTRU

ELIPTIC

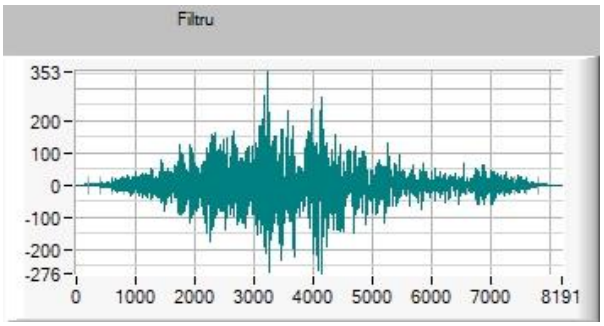
Implementare

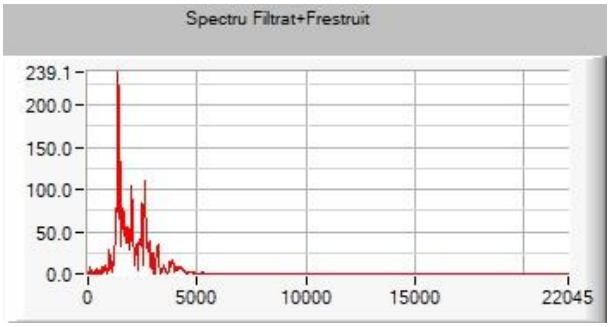
```
//FILTRU + FEREASTRA + SPECTRU
double * Filtru = (double *) malloc (sizeof(double) * punctefereastra);

if(valfiltru == 0)
{
    //Eliptic
    double ripple;
    double atenuare;
    int ordin;
    GetCtrlVal(panel, PANEL_FREQ_RIPPLE, &ripple);
    GetCtrlVal(panel, PANEL_FREQ_ATENUARE, &atenuare);
    GetCtrlVal(panel, PANEL_FREQ_ORDIN, &ordin);
    Elp_LPF(Fereastra, punctefereastra, frecventa, 1500, ripple, atenuare, ordin, Filtru);

        if(valfereastra == 0)
        {
            //BlackMan
            //Copy1D(Fereastra, punctefereastra, vectFerestruit);
            BkmanWin(Filtru, punctefereastra);
            DeleteGraphPlot(panel, PANEL_FREQ_GRAPH_FILTRU, -1, VAL_IMMEDIATE_DRAW );
            PlotY(panel, PANEL_FREQ_GRAPH_FILTRU, Filtru , punctefereastra, VAL_DOUBLE, VAL_THIN_LINE, VAL_EMPTY_SQUARE, VAL_SOLID, VAL_CONNECTED_POINTS, VAL_DK_CYAN);
            Spectru(punctefereastra, Filtru, 0);

            //Salvare
            sprintf(fileName,"Imag_filtrat_elp_ord_%d_ferestruit_bkman_secunda_%d_%d-%d.jpg", ordin, valsecunda, parcurgesecunda, parcurgesecunda + punctefereastra);
            GetCtrlDisplayBitmap(panel, PANEL_FREQ_GRAPH_FILTRU, 1, &imghandle);
            SaveBitmapToJPEGFile(imghandle, fileName, JPEG_PROGRESSIVE, 100);
            //Salvare
            sprintf(fileName,"Spectru_filtrat_elp_ord_%d_ferestruit_bkman_secunda_%d_%d-%d.jpg", ordin, valsecunda, parcurgesecunda, parcurgesecunda + punctefereastra);
            GetCtrlDisplayBitmap(panel, PANEL_FREQ_GRAPH_SPECTRU_2, 1, &imghandle);
            SaveBitmapToJPEGFile(imghandle, fileName, JPEG_PROGRESSIVE, 100);
        }
```

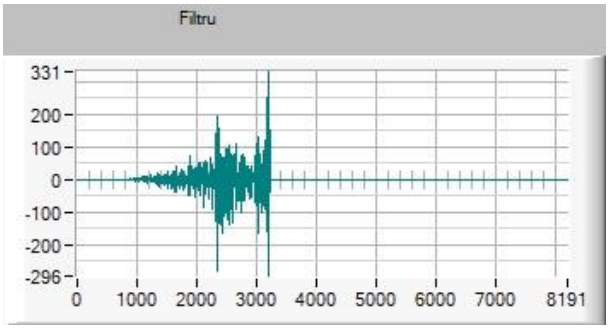
```
        else
        {
            //Triunghiular
            //Copy1D(Fereastra, punctefereastra, vectFerestruit);
            TriWin(Filtru, punctefereastra);
            DeleteGraphPlot(panel, PANEL_FREQ_GRAPH_FILTRU, -1, VAL_IMMEDIATE_DRAW );
            PlotY(panel, PANEL_FREQ_GRAPH_FILTRU, Filtru , punctefereastra, VAL_DOUBLE, VAL_THIN_LINE, VAL_EMPTY_SQUARE, VAL_SOLID, VAL_CONNECTED_POINTS, VAL_DK_CYAN);
            Spectru(punctefereastra, Filtru, 0);

            //Salvare
            sprintf(fileName,"Imag_filtrat_elp_ferestruit_tri_secunda_%d_%d-%d.jpg", valsecunda, parcurgesecunda, parcurgesecunda + punctefereastra);
            GetCtrlDisplayBitmap(panel, PANEL_FREQ_GRAPH_FILTRU, 1, &imghandle);
            SaveBitmapToJPEGFile(imghandle, fileName, JPEG_PROGRESSIVE, 100);
            //Salvare
            sprintf(fileName,"Spectru_filtrat_elp_ferestruit_tri_secunda_%d_%d-%d.jpg", valsecunda, parcurgesecunda, parcurgesecunda + punctefereastra);
            GetCtrlDisplayBitmap(panel, PANEL_FREQ_GRAPH_SPECTRU_2, 1, &imghandle);
            SaveBitmapToJPEGFile(imghandle, fileName, JPEG_PROGRESSIVE, 100);
        }
}
```

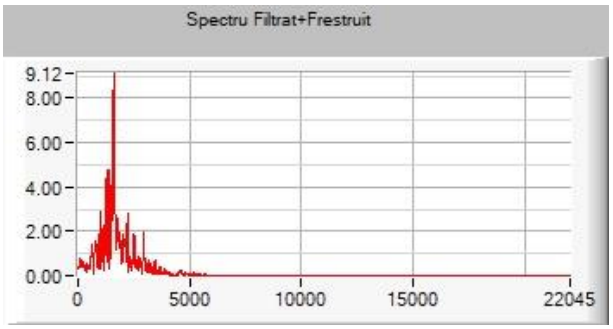
Filtrele eliptice oferă caracteristici de rulare mai abrupte decât filtrele Butterworth sau Chebyshev, dar sunt egale atât în banda de trecere, cât și în banda de oprire. În general, filtrele eliptice îndeplinesc specificațiile de performanță date cu cel mai mic ordin al oricărui tip de filtru.

- **ELIPTIC + BLACKMAN**

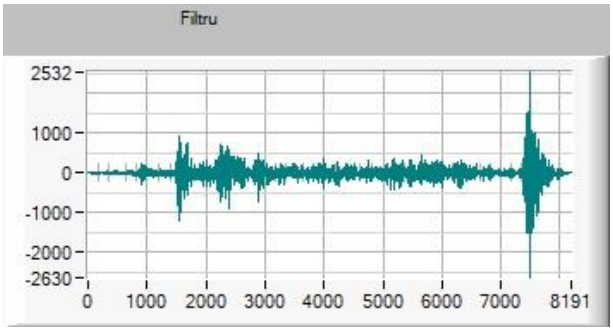


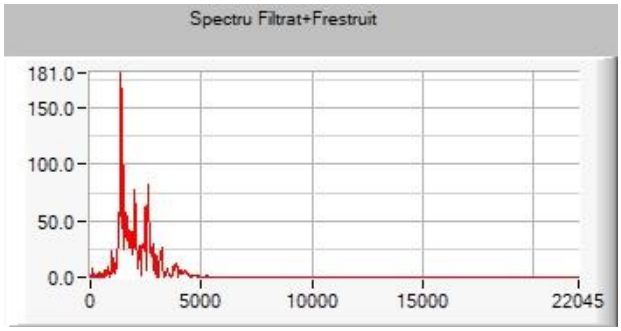


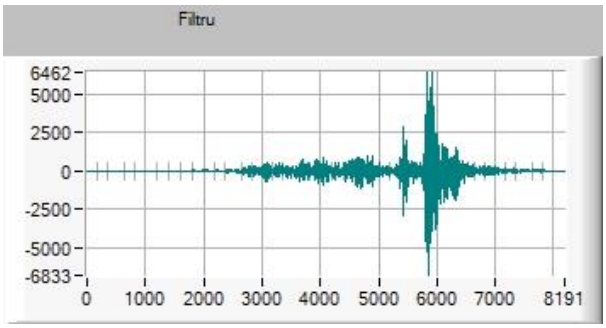


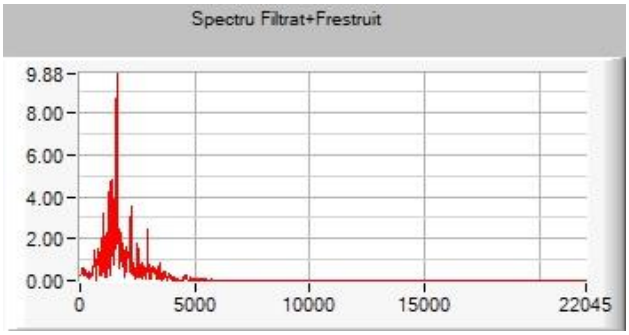


- **ELIPTIC + TRIANGULAR**









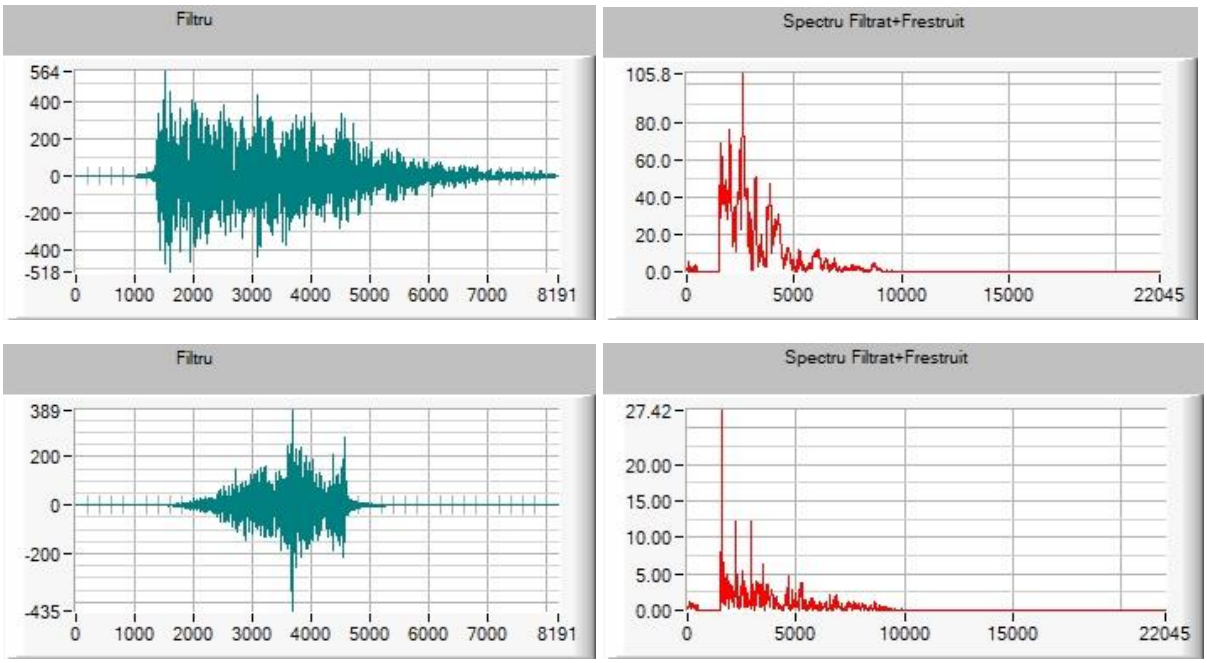
- **NOTCH**

Implementare

```
else
{
    //Notch
    int length, m;
    int impulse_length = punctefereastra;
    double ripple;
    double atenuare;
    double delta, freq_stopband = 950.0, freq_passband = 1050.0;

    GetCtrlVal(panel, PANEL_FREQ_RIPPLE, &ripple);
    GetCtrlVal(panel, PANEL_FREQ_ATENUARE, &atenuare);
    int numtaps_FIR;
    double rs = pow(10, - atenuare/20.0);

    //double * vectFiltru =(double *) malloc(sizeof (double) * punctefereastra);

    //FIIR
    delta = (freq_passband - freq_stopband) * 0.5 / frecventa;
    numtaps_FIR = (int)((-20.0 * log( sqrt(ripple * rs) ) - 13) / (14.6 * delta) * 1.1);
    FIRCoefStruct structFir;
    FIRNarrowBandCoef(frecventa, freq_passband, freq_stopband, 1000.0, ripple, atenuare,  BANDSTOP__, &structFir);

    if(structFir.interp == 1)
    {
        length = impulse_length + structFir.Mtaps - 1;
    }
    else
    {
        length = impulse_length + (structFir.Mtaps - 1) * structFir.interp + structFir.Itaps - 1;
    }

    m = punctefereastra + (structFir.Mtaps - 1) * structFir.interp + structFir.Itaps - 1;
    double *vectFil = (double *) malloc(m * sizeof(double));

    FIRNarrowBandFilter(Fereastra, punctefereastra, structFir, vectFil);
    if(valfereastra == 1)
    {

        //Blackman
        BkmanWin(vectFil, punctefereastra);
        DeleteGraphPlot(panel, PANEL_FREQ_GRAPH_FILTRU, -1, VAL_IMMEDIATE_DRAW );
        PlotY(panel, PANEL_FREQ_GRAPH_FILTRU, vectFil , punctefereastra, VAL_DOUBLE, VAL_THIN_LINE, VAL_EMPTY_SQUARE, VAL_SOLID, VAL_CONNECTED_POINTS, VAL_DK_CYAN);
        Spectru(punctefereastra, vectFil, 0);

        //Salvare
        sprintf(fileName,"Imag_filtrat_notch_ferestruit_bkman_secunda_%d_%d-%d.jpg", valsecunda, parcurgesecunda, parcurgesecunda + punctefereastra);
        GetCtrlDisplayBitmap(panel, PANEL_FREQ_GRAPH_FILTRU, 1, &imghandle);
        SaveBitmapToJPEGFile(imghandle, fileName, JPEG_PROGRESSIVE, 100);
        //Salvare
        sprintf(fileName,"Spectru_filtrat_notch_ferestruit_bkman_secunda_%d_%d-%d.jpg", valsecunda, parcurgesecunda, parcurgesecunda + punctefereastra);
        GetCtrlDisplayBitmap(panel, PANEL_FREQ_GRAPH_SPECTRU_2, 1, &imghandle);
        SaveBitmapToJPEGFile(imghandle, fileName, JPEG_PROGRESSIVE, 100);
    }
    else
    {
        //Triunghiular
        TriWin(vectFil, punctefereastra);
        DeleteGraphPlot(panel, PANEL_FREQ_GRAPH_FILTRU, -1, VAL_IMMEDIATE_DRAW );
        PlotY(panel, PANEL_FREQ_GRAPH_FILTRU, vectFil , punctefereastra, VAL_DOUBLE, VAL_THIN_LINE, VAL_EMPTY_SQUARE, VAL_SOLID, VAL_CONNECTED_POINTS, VAL_DK_CYAN);
        Spectru(punctefereastra, vectFil, 0);

        //Salvare
        sprintf(fileName,"Imag_filtrat_notch_ferestruit_tri_secunda_%d_%d-%d.jpg", valsecunda, parcurgesecunda, parcurgesecunda + punctefereastra);
        GetCtrlDisplayBitmap(panel, PANEL_FREQ_GRAPH_FILTRU, 1, &imghandle);
        SaveBitmapToJPEGFile(imghandle, fileName, JPEG_PROGRESSIVE, 100);
        //Salvare
        sprintf(fileName,"Spectru_filtrat_notch_ferestruit_tri_secunda_%d_%d-%d.jpg", valsecunda, parcurgesecunda, parcurgesecunda + punctefereastra);
        GetCtrlDisplayBitmap(panel, PANEL_FREQ_GRAPH_SPECTRU_2, 1, &imghandle);
        SaveBitmapToJPEGFile(imghandle, fileName, JPEG_PROGRESSIVE, 100);
}
```

Un **filtru Notch** este cunoscut și sub numele de filtru Band Stop sau Band Reject Filter. Aceste filtre resping/atenuează semnalele într-o bandă de frecvență specifică numită interval de frecvență al benzii de oprire și trec semnalele deasupra și sub această bandă.

- ## NOTCH + BLACKMAN



- ## NOTCH + TRIANGULAR