APD Proiect

Etapa 2

1. Se include în aplicație un nou panou conform implementării SwitchPanels de pa Moodle. Se înlocuiește ”Acquisition” cu ”Frequency”.
2. In panoul nou se va implementa prelucrarea în frecvență a fișierului wav.
3. Se incluse un control de tip Graph pentru reprezentarea spectrului întregului semnal si un control numeric tip intreg pentru selectarea numărului de puncte (N) pentru Transformata Fourier.

Pentru a obține spectrul de putere a fisierului Wav se vor utiliza mai multe funcţii CVI, care vor fi apelate în următoarea ordine:

```
ScaledWindowEx (double xArray[], ssize_t numberOfElements, int windowType,
double windowParameter, WindowConst *windowConstants);
```

unde:

- *xArray[]* – bufferul cu eşantioane;
- *numberOfElements* – dimensiunea bufferului;
- *windowType* – tipul de fereastră, care pentru moment va fi *RECTANGLE_*;
- *windowParameter* – nu intereseaza pentru moment;
- *windowConstants* – parametru returnat. Atenţie: Acesta va trebui declarant (WindowConst winConst).

Această funcţie este necesară deoarece, unui semnal eşantionat a cărui perioadă nu se cunoaşte (dacă semnalul este periodic) i se aplică în general o fereastră cu scopul de a “aplatiza” forma semnalului la capetele intervalului de eşantioane analizat. În acest fel, fiecare buffer de eşantioane va fi asimilat cu o perioadă a semnalului (detalii la curs şi în laboratoarele viitoare).

```
AutoPowerSpectrum(double inputArray[], ssize_t numberOfElements, double dt,
double autoSpectrum[], double *df);
```

Această funcţie calculează partea pozitivă a spectrului scalat de putere pentru un semnal eşantionat, după formula:

$$(FFT(X) \cdot FFT^*(X))/n^2, \qquad (1)$$

unde *n* reprezintă numărul de puncte din bufferul cu eşantioane iar *FFT** reprezintă transformata Fourier complex conjugată. Semnificaţia parametrilor rezultă din help-ul contextual. Atenţie: pasul în domeniul timp *dt* trebuie sa fie 1.0/dimensiune buffer.

Funcţia returnează:

- *autoSpectrum[]* – spectrul de putere cu un număr de valori egal cu jumătate din dimensiunea bufferuluide intrare *(inputArray);*

- *df* – pasul în domeniul frecvenţei.

*PowerFrequencyEstimate (double autoSpectrum[], ssize_t numberOfElements, double searchFrequency, WindowConst windowConstants, double df, ssize_t frequencySpan, double *frequencyPeak,);*

Parametrii sunt furnizaţi de funcţiile precedente sau se utilizează valorile implicite. Funcţia returnează:

- *frequencyPeak* – frecvenţa estimată pentru spectrul de putere (maxim) din vectorul autoSpectrum.
- *powerPeak* – valoarea maxima din spectru de putere (din autoSpectrum).

Valorile *frequencyPeak* şi *powerPeak* se vor afişa pe interfaţa grafică.

*SpectrumUnitConversion ((double spectrum[], ssize_t numberOfElements, int type, int scalingMode, int displayUnits, double df, WindowConst windowConstants, double convertedSpectrum[], char *unitString);*

Funcţia converteşte spectrul de intrare (care poate fi puterea, amplitudinea sau amplificarea) în formate alternative (linear, logarithmic, dB) ce permit o reprezentare grafică mai convenabilă.

Precizări:

- *scalingMode - SCALING_MODE_LINEAR*
- *displayUnits - DISPLAY_UNIT_VRMS*
- *convertedSpectrum[]* – vectorul utilizat pentru reprezentarea spectrului
- *unitString* – unit, declarat in funcţia *EveryNCallback* (*char unit[32]="V"*, dacă mărimea de intrare este în volţi);

Abordari alternalive:

spectrul semnalului de intrare utilizând funcția FFT a cărei implementare ilustrează modul teoretic în care am prezentat la curs Transformata Fourier Discretă:

//a doua metoda

```
for(int i=0; i<nSamples; i++)
  {   WfR[i]=rowSignal[i]; //partea reală
       WfI[i]=0;        //partea imaginară – o in cazul dat
  }
  FFT(WfR,WfI,nSamples);

   for (i=0;i<nSamples-1;i++)
    { //spectrul după formulă
            p[i]=(WfR[i]*WfR[i]+WfI[i]*WfI[i]);
            ps[i]=p[i]/pow((double)nSamples,2.0);
```

//A treia metoda

```
Spectrum (wfm3, 1024);//Funcţie CVI- se poate incerca ca alternativă
```

```
delta_t=1/rate;   //pas in timp
      delta_f=1/(nSamples*delta_t); //pas in frecventă
//construiesc axa pentru frecvență
      frequency_array[0]=0.0;
      frequency_array[nSamples-1]=1/(2*delta_t);
      for (i=0;i<nSamples/2;i++)
      {
        frequency_array[i]=i*delta_f;
        frequency_array[nSamples-1-i]=-1*delta_f;
      }
```

4. Se va calcula și reprezenta spectrul pe semnalul intreg sau pe cate o sec., similar cu modul in care s-a realizat analiza pe 1 sec în prima etapa.
5. ***Alte facilități se for adăuga ulterior.***