CYFROWE PRZETWARZANIE SYGNAŁÓW Laboratorium Elektrotechnika, studia niestacjonarne drugiego stopnia	Instytut Metrologii, Elektroniki i Informatyki
Temat: **Filtracja cyfrowa okresowych sygnałów deterministycznych**	Ćwiczenie 2

1. Cel ćwiczenia

Celem ćwiczenia jest praktyczne poznanie zagadnień związanych z filtracją cyfrową oraz ich implementacją w jeżyku ANSI C przy wykorzystaniu środowiska programowania LabWindows/CVI firmy National Instruments.

2. Zagadnienia do przygotowania

- Próbkowanie sygnałów ciągłych – twierdzenie o próbkowaniu.
- Charakterystyczne cechy charakterystyk amplitudowych i fazowych filtrów NOI (Butterwortha, Czebyszewa, inwersyjnego filtru Czebyszewa oraz filtra eliptycznego) i SOI (z oknem trójkątnym, Hanninga, Hamminga i Blackmana).
- Wyznaczanie widma amplitudowego (fazowego) sygnału po filtracji na podstawie znajomości widma amplitudowego (fazowego) sygnału przed filtracją i charakterystyki amplitudowej (fazowej) filtra.
- Równania różnicowe określające zależność pomiędzy sygnałem wejściowym a wyjściowym w filtrach cyfrowych o nieskończonej odpowiedzi impulsowej (NOI) i skończonej odpowiedzi impulsowej (SOI).
- Algorytmy implementacji filtracji sygnałów przez cyfrowe filtry NOI i SOI przy znajomości współczynników filtru.
- Algorytm wyznaczania punktów charakterystyk częstotliwościowych cyfrowych filtrów NOI i SOI przy znajomości współczynników filtru.
- Funkcje języka ANSI C dotyczące wyznaczania wartości funkcji arcus tangens (`atan`, `atan2`).

3. Program ćwiczenia

Zadanie 1

a) Wykorzystując środowisko programowania LabWindows/CVI zmodyfikuj program napisany w ramach realizacji ćwiczenia 0, aby wyznaczał charakterystyki częstotliwościowe (amplitudową i fazową) filtrów o parametrach zadanych przez prowadzącego. W tym celu, posługując się funkcjami opisanymi w punkcie 4.3 niniejszej instrukcji, wyznacza współczynniki filtrów. Wykresy charakterystyk przedstaw na dwóch osobnych elementach *Graph* linią ciągłą. Oś odciętych charakterystyki amplitudowej i fazowej powinna być skalowana w jednostkach częstotliwości (hercach). Oś rzędnych charakterystyki fazowej powinna być skalowana w stopniach kątowych i przedstawiać wartości w zakresie -180° do +180°.

b) Sprawdź, czy kształt uzyskanych charakterystyk jest poprawny biorąc pod uwagę ich cechy charakterystyczne oraz założone parametry filtrów. Wnioski przedstaw prowadzącemu zajęcia.

Zadanie 2

a) Na każdym z elementów *Graph* zaimplementuj możliwość odczytu wartości poprzez kursory. Przy implementacji tego zadania posłuż się informacjami podanymi w punkcie 4.4 instrukcji do ćwiczenia 2.

Zadanie 3

a) Uzupełnij tworzony program o możliwość realizacji filtracji cyfrowej próbek sygnału wczytanego z pliku. W tym celu, posługując się funkcjami opisanymi w punkcie 4.3 niniejszej instrukcji, wyznacza współczynniki filtrów. Wykres sygnału przed i po filtracji przedstaw na dwóch osobnych elementach *Graph*.

b) Sprawdź wstępnie poprawność implementacji filtracji biorąc pod uwagę, że żaden z filtrów nie wzmacnia sygnału wejściowego oraz, że w sygnale wyjściowym znajduje się co najmniej jedną harmoniczną.

Zadanie 4

a) Rozwiń tworzony program o możliwość wyznaczania widma amplitudowego i fazowego zarówno sygnału przed jak i po filtracji. Zaimplementuj możliwość wyznaczania widm zarówno na podstawie wszystkich próbek sygnału, jak również na podstawie tylko drugiej ich połowy. Wykresy widm przedstaw na osobnych elementach *Graph*. Oś rzędnych widm fazowych powinna być skalowana w stopniach kątowych i przedstawiać wartości w zakresie -180° do +180°. Podczas implementacji tego zadania posłuż się informacjami zawartymi w punkcie 4.4 niniejszej instrukcji.

4. Wskazówki do ćwiczenia

4.1. Próbki sygnałów do analizy

Próbki sygnałów wykorzystywanych do badań znajdują się w plikach `WAVE3_XY.CPS`[1].

4.2. Typy i parametry filtrów

Tabela 1 zawiera typy oraz parametry filtrów przewidziane dla każdej z grup ćwiczeniowych.

Tabela 1. *Typy i parametry filtrów dla poszczególnych grup ćwiczeniowych*

Gr.	Filtr			Częstotliwość graniczna f_c filtru [Hz]	Częstotliwość próbkowania f_p [kHz]
	Typ	Rodzaj	Rząd		
1	NOI	dolnoprzepustowy Butterwortha	6 i 12	4000	51,2
	SOI	dolnoprzepustowy z oknem Hanninga	6 i 12	4000	51,2
2	NOI	dolnoprzepustowy Czebyszewa, nierównomierność pasma przepustowego 0,1 dB	6 i 12	1200	12,8
	SOI	dolnoprzepustowy z oknem Blackmana	6 i 12	1200	12,8
3	NOI	dolnoprzepustowy eliptyczny, nierównomierność pasma przepustowego 0,1 dB, tłumienie w paśmie zaporowym 60 dB	4 i 8	500	6,4
	SOI	dolnoprzepustowy z oknem Hamminga	4 i 8	500	6,4
4	NOI	dolnoprzepustowy Czebyszewa inwersyjny, tłumienie w paśmie zaporowym 60 dB	5 i 10	1200	15,36
	SOI	dolnoprzepustowy z oknem trójkątnym	5 i 10	1200	15,36
5	NOI	dolnoprzepustowy Butterwortha	7 i 14	5500	102,4
	SOI	dolnoprzepustowy z oknem Blackmana	7 i 14	5500	102,4
6	NOI	dolnoprzepustowy Czebyszewa inwersyjny, tłumienie w paśmie zaporowym 80 dB	8 i 16	550	5,12
	SOI	dolnoprzepustowy z oknem Hanninga	8 i 16	550	5,12
7	NOI	dolnoprzepustowy Czebyszewa, nierównomierność pasma przepustowego 0,2 dB	5 i 10	7000	76,8
	SOI	dolnoprzepustowy z oknem trójkątnym	5 i 10	7000	76,8

4.3. Wyznaczanie współczynników filtrów cyfrowych

4.3.1. Filtry NOI

Przy wyznaczaniu współczynników cyfrowych filtrów typu NOI należy posługiwać się następującymi funkcjami zgrupowanymi w bibliotece Advanced Analysis » Signal Processing » IIR Digital Filters » Old-Style Filter Functions:

- `Bessel_Coef` filtr Bessela
- `Bw_Coef` filtr Butterwortha
- `Ch_Coef` filtr Czebyszewa typu 1
- `InvCh_Coef` filtr Czebyszewa typu 2
- `Elp_Coef` filtr eliptyczny

Poszczególne parametry tych funkcji mają następujące znaczenie:

`Type` typ filtru (dolno-, górno-, pasmowoprzepustowy lub pasmowozaporowy),
`Order` rząd filtru,

[1] X oznacza numer grupy laboratoryjnej, natomiast Y to jednocyfrowy numer sygnału.

`Sampling Frequecy`	częstotliwość próbowania,
`Lower Cuttof Freq`	dolna częstotliwość załamania charakterystyki częstotliwościowej,
`Upper Cutoff Freq`	górna częstotliwość załamania charakterystyki częstotliwościowej (w przypadku filtrów dolno- i górnoprzepustowych jej wartość nie jest brana pod uwagę, jednak ze względu na konieczność wprowadzenia jakiejś wartości można wpisać 0.0),
`A Coefficients Array`	tablica, do której zostaną zapisane współczynniki mianownika transmitancji filtru cyfrowego,
`B Coefficients Array`	tablica, do której zostaną zapisane współczynniki licznika transmitancji filtru cyfrowego,
`Number of A Coeficients`	liczba współczynników mianownika (w przypadku filtru dolno- i górnoprzepustowego jest większa o 1 od rzędu filtru),
`Number of B Coeficients`	liczba współczynników licznika (w przypadku filtru dolno- i górnoprzepustowego, jest większa o 1 od rzędu filtru),
`Passband Ripple`	poziom zafalowań w paśmie przepustowym (tylko w filtrze Czebyszewa typu 1 i w filtrze eliptycznym),
`StopBand Atten`	minimalne tłumienie w paśmie zaporowym (tylko w filtrze Czebyszewa typu 2 i w filtrze eliptycznym).

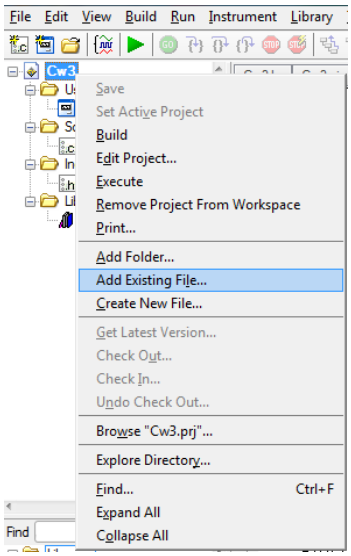
4.3.2. Filtry SOI

Przy wyznaczaniu współczynników cyfrowych, dolnoprzepustowych filtrów typu SOI, należy posługiwać się funkcją `Wind_LPF` znajdująca się w bibliotece Advanced Analysis » Signal Processing » FIR Digital Filters. Poszczególne parametry tej funkcji mają następujące znaczenie:

`Sampling Frequecy`	częstotliwość próbowania,
`Cuttof Frequency`	częstotliwość załamania charakterystyki częstotliwościowej,
`Number of Coef`	liczba współczynników filtru (większa o 1 od rzędu filtru),
`Coefficient Array`	tablica, do której zostaną zapisane wyznaczone współczynniki filtru,
`Window type`	typ zastosowanego okna.

4.4. Funkcja wyznaczająca widmo amplitudowe i fazowe sygnałów

W celu uproszczenia realizacji wyznaczania widma amplitudowego i fazowego na stronie WWW z materiałami do niniejszego przedmiotu udostępniono bibliotekę DLL zawierającą funkcję `ComputeSpectrum`, która na podstawie próbek umożliwia wyznaczenie widma amplitudowego lub fazowego.



Rys. 1. *Widok menu podręcznego z wyróżnioną pozycją umożliwiającą dodawanie do projektu dodatkowych plików*

W celu wykorzystania omawianej biblioteki w tworzonym programie należy ze strony WWW wskazanej przez prowadzącego ściągnąć archiwum o nazwie `cw3cps.zip` i wypakować znajdujące się w nim pliki do folderu, w którym znajdują się pliki tworzonego programu. Następnie w środowisku LabWindows/CVI przy kursorze umieszczonym na nazwie projektu należy kliknąć prawy klawisz myszy i w rozwiniętym menu podręcznym wybrać pozycję Add Existing File, co pokazano na rysunku 1.

W otwartym oknie wyboru plików należy wskazać pliki `cw3cps.h` oraz `cw3cps.lib` i zaakceptować wybór klikając klawisz OK. Po tej operacji należy dołączyć plik `cw3cps.h` do kodu tworzonego programu wykorzystując dyrektywę `#include`.

Prototyp funkcji `ComputeSpectrum` ma następującą postać:

```
int ComputeSpectrum (double *samples, unsigned int numberOfSamples,
                     unsigned int numberOfSamplesRemoved, double samplingFrequency,
                     int typeOfSpectrum, double *spectrum, double *frequency);
```

Poszczególne parametry funkcji mają następujące znaczenie:

`samples`	wskaźnik na tablicę zawierającą próbki sygnału;
`numberOfsamples`	całkowita liczba próbek znajdujących się w tablicy wskazywanej przez wskaźnik `samples`;
`numberOfsamplesRemoved`	liczba próbek z początku sygnału, jaka nie będzie brana do wyznaczania widma;
`samplingFrequency`	zastosowana podczas pobierania próbek częstotliwość próbkowania wyrażona w hercach;
`numberOfsamples`	rodzaj widma, jakie ma być wyznaczane (0 – amplitudowe lub 1 – fazowe);
`spectrum`	wskaźnik do tablicy, gdzie po wykonaniu funkcji zostaną zapisane wyznaczone wartości widma amplitudowego (wyrażone w tych samych jednostkach co próbki) lub fazowego (wyrażone w stopniach);
`frequency`	wskaźnik do tablicy, gdzie zostaną zapisane wartości częstotliwości (wyrażone w hercach) odpowiadające poszczególnym wartościom prążków widma zapisanych do tabeli wskazywanej przez wskaźnik `spectrum`.

W przypadku, gdy:

- wskaźnik `samples` jest równy NULL,
- wskaźnik `spectrum` jest równy NULL,
- wskaźnik `frequency` jest równy NULL,
- wartość parametru `numberOfSamplesRemoved` jest większa lub równa od wartości parametru `numberOfSamples`,

wtedy funkcja zwróci wartość -1. W przypadku poprawnego wyznaczenia widma funkcja zwraca liczbę wyznaczonych wartości widma.

Przedstawiony poniżej przykład obrazuje sposób wywołania funkcji `ComputeSpectrum` do wyznaczenia widma amplitudowego. W przykładzie tym założono, że dysponujemy próbkami sygnału zapisanymi w tablicy o nazwie `probki`, w zmiennej o nazwie `lp` przechowujemy liczbę próbek, w zmiennej o nazwie `fp` zastosowaną podczas pobierania próbek częstotliwość próbkowania i chcemy wyznaczyć widmo amplitudowe na podstawie wszystkich próbek (trzeci parametr wskazujący ile próbek z początku sygnału będzie odrzuconych wynosi 0).

```
ComputeSpectrum (probki, lp, 0, fp, SPECTRUM_AMPL, widmo, f);
```

W powyższym wywołaniu wykorzystano stałą `SPECTRUM_AMPL`, która jest zdefiniowana w pliku `cw3cps.h` i może być wykorzystana w celu wskazania, jakie widmo ma być wyznaczone. W przypadku chęci wyznaczenia widma fazowego należy zastosować stałą `SPECTRUM_PHASE`. Dodatkowo założono, że `widmo` i `f` są tablicami o komórkach typu `double` i ich liczbie równej co najmniej wartości zmiennej `lp`.

Trzeci parametr omawianej funkcji pozwalają na wyznaczenie widma na podstawie fragmentu sygnału. Jeśli jednak funkcja zostanie wywołana w następujący sposób

```
ComputeSpectrum (probki, lp, lp/2, fp, SPECTRUM_AMPL, widmo, f);
```

to pierwsza połowa próbek znajdujących się w tablicy `probki` nie będzie brana pod uwagę (trzeci parametr wywołania funkcji wskazuje, że pierwsze `lp/2` próbek nie będzie brane pod uwagę), a widmo zostanie wyznaczone na podstawie próbek znajdujących się w drugiej połowie tablicy. W takim wypadku liczba komórek tablic `widmo` i `f` może być równa `lp/2`.

Poniżej pokazano szerszy przykład wykorzystujący funkcję `ComputeSpectrum` wraz z deklaracjami niezbędnych tablic i prezentacją widma na elementach *Graph*. Zakłada się w nim, że na panelu użytkownika znajduje się element typu *Ring*, oznaczony stałą `TYP_WIDMA`, który służy do wyboru rodzaju widma (amplitudowe, fazowe) do wyznaczenia. Wykorzystując niniejszy przykład w swojej aplikacji należy pamiętać o dostosowaniu stałych identyfikujących elementy *Graph* do swojej wersji programu.

```
  int typWidma;
  unsigned int x;
```

```
double *widmo, *f;

// Pobranie informacji jakie widmo ma być wyznaczone
GetCtrlVal(panelHandle, PANEL_TYP_WIDMA, &typWidma);
// Przydzielenie pamięci dla tablic
widmo = malloc(lp*sizeof(double));
f = malloc(lp*sizeof(double));
// Wyznaczenie widma
x = ComputeSpectrum(probki, lp, 0, fp, typWidma, widmo, f);
// Prezentacja widma na elemencie Graph
DeleteGraphPlot(panelHandle, PANEL_GRAPH, -1, VAL_IMMEDIATE_DRAW);
PlotXY(panelHandle, PANEL_GRAPH, f, widmo, (x/2)+1, VAL_DOUBLE, VAL_DOUBLE,
       VAL_BASE_ZERO_VERTICAL_BAR, VAL_EMPTY_SQUARE, VAL_SOLID, 1, VAL_RED);
// Zwolnienie pamięci przydzielonej dla tablic
free(widmo);
free(f);
```

5. Opracowanie wyników

Sprawozdanie z ćwiczenia powinno zawierać wymienione poniżej elementy ***wraz z komentarzami słownymi***.

- Widok interfejsu użytkownika stworzonej aplikacji i krótki opis jej obsługi.
- Uzyskane charakterystyki filtrów (amplitudowe i fazowe).
- Analizy porównawcze otrzymanych charakterystyk (amplitudowych i fazowych), tj. porównanie charakterystyk dwóch filtrów tego samego typu, lecz różnych rzędów oraz filtrów tego samego rzędu, lecz różnych typów.
- Przebiegi sygnałów przed i po filtracji oraz ich widma (amplitudowe i fazowe).
- Analiza poprawności przeprowadzonej filtracji w oparciu o dziedzinę częstotliwości. W tym celu należy w tabeli zebrać informacje dotyczące częstotliwości, amplitud i faz harmonicznych występujących w sygnale filtrowanym. Następnie należy określić wartości wzmocnienia i przesunięcia fazowego dla każdej z częstotliwości, przy której występują harmoniczne w sygnale filtrowanym. Na podstawie tych informacji należy obliczyć amplitudy i fazy harmonicznych, jakie znajdą się sygnale po przeprowadzeniu filtracji. Otrzymane wyniki należy porównać z wynikami uzyskanymi przez wyznaczenie widma amplitudowego i fazowego na podstawie próbek sygnału po filtracji.

Na końcową ocenę z ćwiczenia mają również wpływ:

- systematyczna realizacja poszczególnych zadań ćwiczenia;
- poprawne skomentowanie kodu całej tworzonej aplikacji;
- poprawny styl kodowania (stosowanie wcięć w kodzie itp.);
- bezbłędne działanie stworzonego programu przy różnych niestandardowych działaniach użytkownika;
- implementacja wszystkich założeń dla aplikacji przedstawionych przez prowadzącego zajęcia;
- końcowy wygląd interfejsu użytkownika aplikacji, który powinien być estetyczny i przejrzysty;
- przejrzyste i estetyczne formatowanie pisemnego sprawozdania z ćwiczenia.

6. Literatura

[1] Borodziewicz W., Jaszczak K.: „*Cyfrowe przetwarzanie sygnałów – wybrane zagadnienia*”. WNT, Warszawa 1987.
[2] Dąbrowski A.: „*Przetwarzanie sygnałów przy użyciu procesorów sygnałowych*”. Wydawnictwo Politechniki Poznańskiej 2000.
[3] Lyons R.G.: „*Wprowadzenie do cyfrowego przetwarzania sygnałów*”. WKŁ, Warszawa 1999.
[4] Oppenheim V, Schaffer R.W.: „*Cyfrowe przetwarzanie sygnałów*”. WKŁ, Warszawa 1979.
[5] Oppenheim V, Willsky A.S.: „*Signals & Systems*”. Prentice Hall, 1997.
[6] Oppenheim V, Schaffer R.W.: „*Discrete-Time Signal Processing*”. WKŁ, Warszawa 1979.
[7] Roberts R.A., Mullis C.T.: „*Digital Signal Processing*”. Addison-Wesley Publishing Company, 1987.
[8] Stabrowski M.: „*Miernictwo elektryczne. Cyfrowa technika pomiarowa*”. Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa 1994.
[9] Sydenham P.H.: „*Podręcznik metrologii – tom I*”. WKŁ, Warszawa 1988.
[10] Tietze U, Schenk Ch.: „*Układy półprzewodnikowe*” WNT, Warszawa 1997.
[11] Zieliński T.P.: „*Od teorii do cyfrowego przetwarzania sygnałów*”. Wydział EAIiE AGH Kraków 2002.
[12] Zieliński T.P.: „*Cyfrowe przetwarzanie sygnałów. Od teorii do zastosowań*”. WKŁ, Warszawa 2005.