# Instrukcja do laboratorium z cyfrowego przetwarzania sygnałów

# Ćwiczenie 6

## *Filtracja w dziedzinie widmowej*

*© Przemysław Korohoda, KE, AGH*

**Zawartość instrukcji:**

Na instrukcję składają się następujące części:

**1 Materiał z zakresu *DSP***
**2 Korzystanie z pakietu MATLAB**
**3 Zadania do wykonania**
**4 Przebieg ćwiczenia**

Do sprawnego wykonania ćwiczenia nie jest konieczna wcześniejsza praktyczna znajomość nie wprowadzonych w ramach poprzednich ćwiczeń funkcji pakietu MATLAB, jednak niezbędne jest przeanalizowanie części 4 i samodzielne sformułowanie propozycji odpowiednich wniosków w oparciu o zamieszczone tam przykłady.
W razie niejasności należy skonsultować się z prowadzącym przed zajęciami - na przykład w terminie konsultacji - bezpośrednio lub poprzez e-mail:
*korohoda@uci.agh.edu.pl*

# 1 Materiał z zakresu *DSP*

Odpowiednie wnioski teoretyczne należy wyciągnąć bazując na analizie przedstawionych przykładów w części "przebieg ćwiczenia". Wprawdzie ostateczne wnioski należy formułować dopiero po zajęciach, jednak bez wcześniejszego przemyślenia przykładów i wstępnego opracowania propozycji tych wniosków zadanie to będzie znacznie utrudnione.

# 2 Korzystanie z pakietu MATLAB

W tym ćwiczeniu, poza drobnymi wyjątkami, korzysta się z dotychczas poznanych możliwości pakietu MATLAB.

## 2.1 Opis wybranych funkcji

**triang** - generowanie ciągu o wartościach należących do obwiedni w kształcie trójkąta, np.:

```
>> triang(3)
```

da w wyniku ciąg: 0,5 1 0,5

**find** - wyszukiwanie indeksów dla elementów wektora lub macierzy, które spełniają podany warunek, np.:

```
>> ind=find(v==0);
```

wektor "índ" będzie zawierał indeksy tych elementów wektora "v", które są równe zero.

# 3 Zadania do wykonania

1. Zademonstrować za pomocą filtracji w dziedzinie częstotliwości dekompozycję sygnału na część gorno- i dolnopasmową (potwierdzenie liniowości transformacji *DFT*).

2. Pokazać za pomocą przykładów, jaką rolę odgrywa faza filtru (lub sygnału).

3. Potwierdzić, że liniowość fazy filtru *FIR* można zapewnić przez spełnienie odpowiedniego warunku dla transmitancji "Z" tego filtru.

4. Pokazać na czym polega różnica między splotem liniowym i kołowym oraz jak za pomocą splotu liniowego można wyznaczyć splot kołowy. Wykazać, że filtracja w dziedzinie *DFT* jest równoważna odpowiednio zastosowanemu splotowi kołowemu.

5. Zademonstrować związek jaki zachodzi pomiędzy transformatami *DFT*:
a) ciągu,
b) tego samego ciągu, ale z K-elementowymi ciągami elementów zerowych powstawianych pomiędzy kolejne elementy ciągu z punktu a) (w efekcie następuje wydłużenie ciągu).

6. Pokazać związek zachodzący pomiędzy filtracją poprzez splot z zadaną odpowiedzią impulsową (*FIR*) i interpolacją.

## 4 Przebieg ćwiczenia

**1. Zademonstrować za pomocą filtracji w dziedzinie częstotliwości dekompozycję sygnału na część gorno- i dolnopasmową (potwierdzenie liniowości transformacji *DFT*)**

```
>> x=rand(1,128);
>> plot(0:127,x);
>> X=fft(x);
>> figure(2);
>> plot(0:127,abs(X));
>> K=10;
>> HDP=[ones(1,K), zeros(1,(128-2*K+1)),ones(1,K-1)];
>> max(abs(imag(ifft(HDP))))                    sprawdzenie czy nie ma błędu symetrii
>> HGP=1-HDP;
>> hold on;
poniżej pomnożono amplitudę przez 10, żeby było widać wykres na tle widma sygnału:
>> plot(0:127,abs(HDP)*10,'r');      "abs" nie jest tu konieczne, ale dla zasady zostało użyte
>> plot(0:127,abs(HGP)*10,'g');      "abs" nie jest tu konieczne, ale dla zasady zostało użyte

>> XDP=X.*HDP;
>> XGP=X.*HGP;
>> plot(0:127,abs(XDP),'r');
>> plot(0:127,abs(XGP),'g');
>> xdp=real(ifft(XDP));                         zakładamy, że nie ma błędu symetrii
>> xgp=real(ifft(XGP));                         zakładamy, że nie ma błędu symetrii
>> figure(1); hold on
>> plot(0:127,xdp,'r',0:127,xgp,'g');
>> max(abs((xdp+xgp)-x))                        sprawdzenie, czy obie części dadzą kompletny
                                                sygnał
```

**2. Pokazać za pomocą przykładów, jaką rolę odgrywa faza filtru (lub sygnału)**
Należy pozamykać wszystkie okna graficzne oraz usunąć zmienne.
Uwaga - przykład dotyczy parzystej długości ciągu.

```
>> N=16;
>> x=[1:10,8:-2:0,0];
>> stem(0:15, x);                    oglądamy sygnał
>> X=fft(x);
>> figure(2);
>> stem(0:15,angle(X)/pi); grid      faza sygnału(podzielona przez pi)
>> figure(3);
>> stem(0:15,abs(X));                oglądanie amplitudy sygnału
>> Fi=ones(1,7)*pi/2;                stała wartość fazy
>> Fi1=[angle(X(1)),Fi,angle(X(9)),-Fi(7:-1:1)];   wprowadzenie odpowiedniej symetrii
>> H1=exp(j*Fi1);                    amplituda filtru jest więc wszędzie równa 1
>> X1=X.*H1;                         filtracja w dziedzinie FFT
>> x1=ifft(X1);
>> max(abs(imag(x1)))                sprawdzenie poprawności symetrii
>> x1=real(x1);                      jeśli symetrie są poprawne, to eliminujemy szczątkową
                                     część urojoną
>> figure(1); hold on
>> plot(0:15,x1,'ro');               porównanie graficzne - od razu widać różnice
```

A teraz liniowa faza filtru:

```
>> m=1;
>> Fi=(0:N/2)*(m*pi)/(N/2);
>> Fi2=[Fi,-Fi(N/2:-1:2)];
>> figure(4)
>> stem(0:15, Fi2);
```

```
>> H2=exp(j*Fi2);
>> X2=X.*H2;
>> x2=real(ifft(X2));                 już bez sprawdzania symetrii
>> figure(1); hold off                poniżej ponownie porównujemy:
>> stem(0:15,x); hold on
>> plot(0:15,x2,'ro');                i powinno wyjść jedynie cykliczne przesunięcie
Powyższe ćwiczenie można powtórzyć dla różnych całkowitych wartości "m"
>> figure(4);
>> plot(unwrap([Fi2,Fi2,Fi2]));              bo objerzeniu wykresu powinno być jasne co
                                        oznacza termin "liniowa faza"
```

Faza liniowa, a jednak niewłaściwa - ponieważ nie przechodzi przez punkt (0,0):

```
>> Fi=(N/2:-1:0)*(m*pi)/(N/2);
>> Fi3=[Fi,-Fi(N/2:-1:2)];
>> figure(4);
>> plot(unwrap([Fi3,Fi3,Fi3]));
>> H3=exp(j*Fi3);
>> max(abs(imag(ifft(H3))))           weryfikacja symetrii
>> X3=X.*H3;
>> x3=real(ifft(X3));
>> figure(1); hold off
>> stem(0:15,x); hold on
>> plot(0:15,x3,'ro');
```

Faza tylko pozornie wyglądająca na liniową:

```
>> m=0.5;
```

i ciąg dalszy jak podczas demonstracji dla fazy liniowej;

**3. Potwierdzić, że liniowość fazy filtru *FIR* można zapewnić przez spełnienie odpowiedniego warunku dla transmitancji "Z" tego filtru**

Należy pozamykać wszystkie okna graficzne oraz usunąć zmienne.

```
>> M=2;                               rząd filtru będzie zatem 4*M
>> z0=(rand(1,M)-0.5)+j*rand(1,M);
>> z=[z0';conj(z0')];
>> z=[z;1./z];                        wektor zer powinien być kolumnowy
>> p=zeros(4*M,1);                    odpowiednia ilość biegunów w punkcie (0,0)
>> zplane(z,p);
>> [b,a]=zp2tf(z,p,1);
>> d=zeros(1,128); d(1)=1;            delta Kroneckera
>> h=filter(b,a,d);
>> H=fft(h);
>> figure(1);
>> plot(0:127, angle(H));
>> figure(2);
>> plot(0:127, abs(H));
```

**4. Pokazać na czym polega różnica między splotem liniowym i kołowym oraz jak za pomocą splotu liniowego można wyznaczyć splot kołowy. Wykazać, że filtracja w dziedzinie *DFT* jest równoważna odpowiednio zastosowanemu splotowi kołowemu**

Należy zamknąć wszystkie okna graficzne oraz usunąć zmienne.

```
>> N=32; K=8;
>> x=rand(1,N);
>> h=rand(1,K);                       zakładamy K mniejsze lub równe N
>> x1=conv(x,h);
>> X=fft(x);
```

```
>> H=fft(h,N);
>> X2=X.*H;
>> x2=real(ifft(X2));
>> xper=[x,x,x];                    symulacja ciągu okresowego
>> xper1=conv(xper,h);
>> max(abs(x2-xper1(N+1:2*N)))      porównujemy
>> x21=[x1(1:K-1)+x1(N+1:N+K-1), x1(K:N)];
>> max(abs(x2-x21))                 i znów porównujemy
```

**5. Zademonstrować związek jaki zachodzi pomiędzy transformatami *DFT*:**
**a) ciągu,**
**b) tego samego ciągu, ale z K-elementowymi podciągami elementów zerowych powstawianych pomiędzy kolejne elementy ciągu z punktu a) (w efekcie następuje K+1 krotne wydłużenie ciągu)**
Należy zamknąć wszystkie okna graficzne oraz usunąć zmienne.

```
>> N=8; K=3;
>> x=rand(1,N);
>> x2=zeros(1,N+N*K);
>> N2=length(x2)
>> x2(1:(K+1):N2)=x;
>> stem(0:N2-1,x2);
>> X=fft(x);
>> X2=fft(x2);
>> stem(0:N-1,abs(X));
>> hold on
>> plot(0:N2-1,abs(X2),'r');
```

**6. Pokazać związek zachodzący pomiędzy filtracją z zadaną odpowiedzią impulsową (*FIR*) i interpolacją**
Należy zamknąć wszystkie okna graficzne oraz usunąć zmienne.

Najpierw pokażemy, że interpolację można uzyskać przez splot z odpowiedzią impulsową:

```
>> N=64;
>> x=zeros(1,N);
>> sp=8;                            okres próbkowania
>> x(1:sp:N)=rand(1,N/sp);          generujemy wartości tylko w punktach próbkowania
>> stem(0:N-1,x);
>> xint=[triang(sp*2-1)',0];        ciąg trójkątny o długości 16 elementów, pewna
                                       asymetria jest tutaj celowa
>> figure(2);
>> stem(-(sp-1):sp,xint);
>> xconv=conv(xint,x);              interpolacja przez splot liniowy
>> xr=xconv(sp:N-1+sp);             zostawiamy tylko "środkową" część wyniku splatania
>> figure(1); hold on
>> plot(0:N-1,xr,'r');

>> xs=x(1:sp:N);                    wybieramy tylko wartości próbek
poniżej generowanie tylu ciągów bazowych, ile jest próbek - czyli 8:
>> for k=0:7, B(k+1,1:(N+2*sp))=[zeros(1,k*sp), xint, zeros(1,(8-k)*sp)]; end;
>> b1=B(1,1:(N+2*sp-1));
>> b2=B(2,1:(N+2*sp-1));
>> b3=B(3,1:(N+2*sp-1));
>> b4=B(4,1:(N+2*sp-1));
>> b5=B(5,1:(N+2*sp-1));
>> b6=B(6,1:(N+2*sp-1));
>> b7=B(7,1:(N+2*sp-1));
>> b8=B(8,1:(N+2*sp-1));
>> figure(3);
>> stem(-(sp-1):9*sp-1, b6);        przykładowy ciąg bazowy
```

```
>> xext=[zeros(1,sp-1),x,zeros(1,sp)];      ciąg pierwotny wydłużony zerami - 7 w lewo i 8 w prawo
>> xrext=xs(1)*b1+xs(2)*b2+xs(3)*b3+xs(4)*b4+xs(5)*b5+xs(6)*b6+xs(7)*b7 +xs(8)*b8;
>> figure(4);                                wynik interpolacji w postaci graficznej:
>> plot(-(sp-1):9*sp-1, x2rext , 'g'); hold on;
>> stem((0:7)*sp, x(1:sp:N));
>> max(abs(xrext-xconv))                     dokładne porównanie obu wyników - ze splatania i za pomocą układu ciągów bazowych
```

A teraz pokażemy jak połączyć korzystanie z "xint" oraz **fft**

```
>> xintext=fftshift([zeros(1,N/2-sp+1),xint,zeros(1,N/2-sp-1)]);      przedłużamy zerami i przesuwamy cyklicznie
>> figure(5);
>> stem(0:N-1,xintext);
>> XINTEXT=fft(xintext);
>> figure(6);
>> stem(0:N-1,abs(XINTEXT));
>> X=fft(x);
>> XRE=X.*XINTEXT;
>> xre = ifft(XRE);
>> max(abs(imag(xre)))
>> xre=real(xre);
>> figure(7)
>> stem(0:N-1,x); hold on; plot(0:N-1,xre,'r');
```