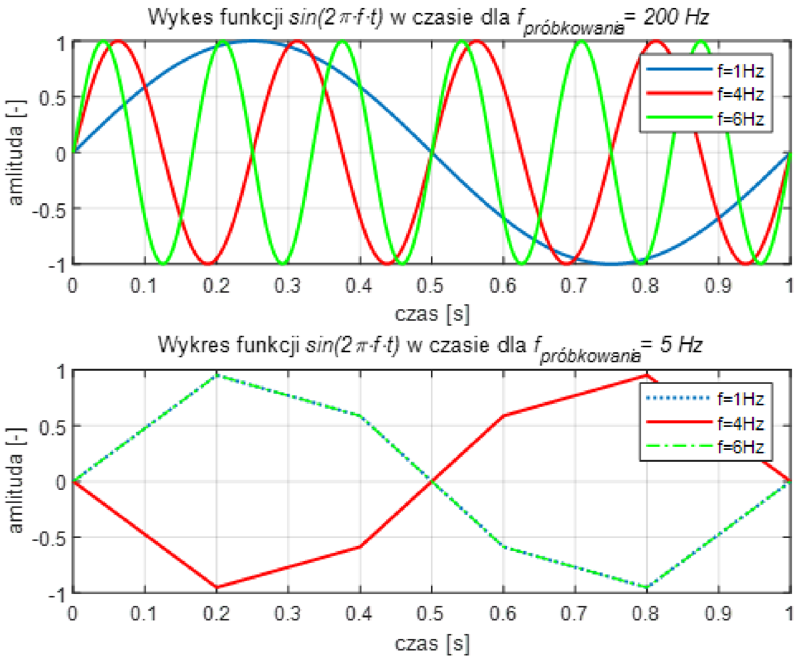
Numer ćwiczenia	1	Temat:
Grupa	L05	**Laboratorium #1 - Podstawy**
Data wykonania ćwiczenia:	25.02.2019	
Nazwisko i imię:	1. Wilk Michał	
	2. Zbroja Wiktor	

1. Sygnały i ich parametry

1.1. Częstotliwość próbkowania -przykład 1

```
% częstotliwości generowanych sygnałów
f1=1; %[Hz]
f2=4; %[Hz]
f3=6; %[Hz]
fs=200; %[Hz] częstotliwość próbkowania
t=0:1./fs:1; %wektor czasu
% wygenerowanie trzech sygnałów sinusoidalnych
y1 = sin(2*pi*f1*t);
y2 = sin(2*pi*f2*t);
y3 = sin(2*pi*f3*t);

%tworzenie wykresu
subplot(2,1,1)
plot(t,y1, 'LineWidth',1.5)
hold on
plot(t, y2,'r','LineWidth',1.5)
hold on
plot(t, y3,'g','LineWidth',1.5)
xlabel('czas [s]')
ylabel('amlituda [-]')
title(['\rmWykes funkcji \itsin(2\pi\cdotf\cdott)\rm w czasie dla
\itf_{próbkowania}= ', num2str(fs), ' Hz'])
grid on
legend(['f=', num2str(f1), 'Hz'],['f=', num2str(f2), 'Hz'],['f=', num2str(f3),
'Hz'])

fs=5 %[Hz] Zmiana częstotliwości próbkowania
t=0:1./fs:1; %nowy wektor czasu
%Generowanie sygnałów
y1 = sin(2*pi*f1*t);
y2 = sin(2*pi*f2*t);
y3 = sin(2*pi*f3*t);

%tworzenie kolejnego wykresu
subplot(2,1,2)
plot(t,y1,':', 'LineWidth',1.5)
hold on
plot(t, y2,'r','LineWidth',1.5)
hold on
plot(t, y3,'g-.','LineWidth',1)
xlabel('czas [s]')
ylabel('amlituda [-]')
title(['\rmWykes funkcji \itsin(2\pi\cdotf\cdott)\rm w czasie dla
\itf_{próbkowania}= ', num2str(fs), ' Hz'])
```

```
grid on
legend(['f=', num2str(f1), 'Hz'],['f=', num2str(f2), 'Hz'],['f=', num2str(f3),
'Hz'])
```



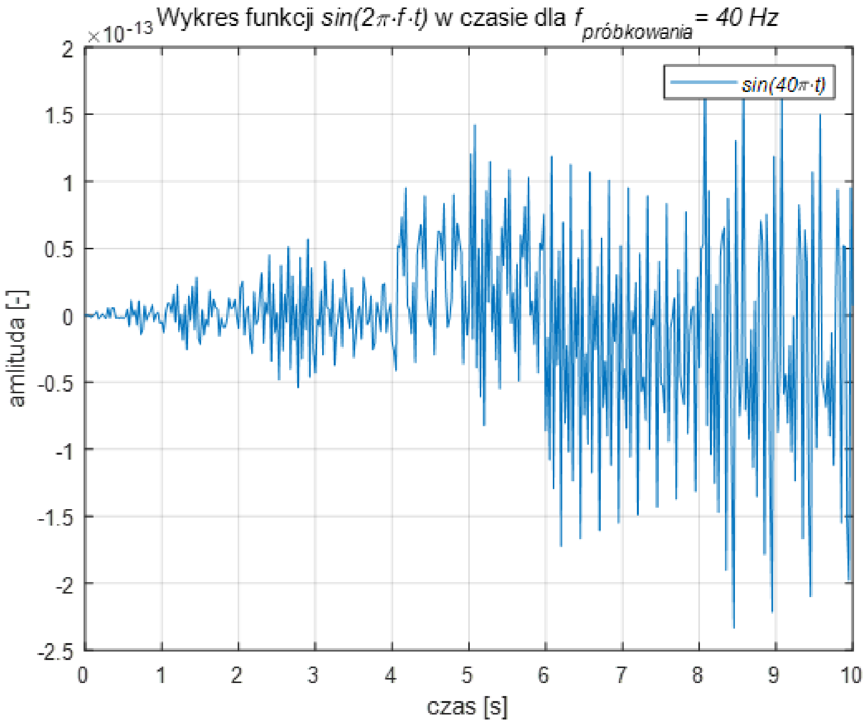
Rys. 1.1 Wykres funkcji otrzymanych w zadaniu 1.1

Zmniejszenie 40-krotne częstotliwości próbkowania doprowadziło do złamania warunku twierdzenia Kotielnikowa-Shannona dla sygnałów o częstotliwości 4 Hz i 6 Hz (odpowiednio minimalne częstotliwości próbkowania to 8 Hz i 12 Hz). Sygnały te nie zostały poprawnie odtworzone (**Rys. 1.1**). Funkcja sinus o częstotliwości 1 Hz pokrywa się z funkcją o częstotliwości 6 Hz. W obydwu przypadkach została utracona informacja o mierzonym sygnale, przez co niemożliwe jest poprawne ich odtworzenie.

1.2. Częstotliwość próbkowania -przykład 2

```
f1=20; %[Hz] częstotliwość sygnału
fs=40; %[Hz] częstotliwość próbkowania
t=0:1./fs:10; %wektor czasu
y = sin(2*pi*f1*t); % generowanie sygnału

%generowanie wykresu
plot(t,y)
set(gcf,'color','w');
xlabel('czas [s]')
ylabel('amlituda [-]')
title(['\rmWykres funkcji \itsin(2\pi\cdotf\cdott)\rm w czasie dla
\itf_{próbkowania}= ', num2str(fs), ' Hz'])
grid on
```

Komentarz [1]: Pomyłka - zamienione częstotliwości



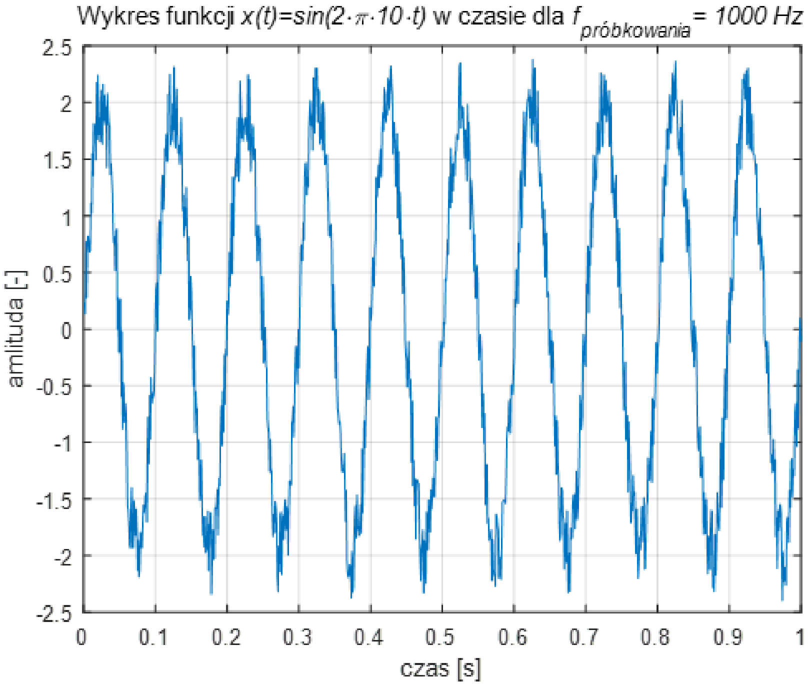
Rys. 1.2 Wykres funkcji otrzymanej w zadaniu 1.2 (*f=20 Hz*)

W tym zadaniu mamy do czynienia z przypadkiem granicznym, w którym częstotliwość próbkowania jest dokładnie równa minimalnej częstotliwości próbkowania z tw. Nyquista. Mimo to nie otrzymujemy poprawnego odwzorowania, co wynika z faktu, iż dla wygenerowanego przez nas wektora czasu wszystkie próbki znajdują się w momencie gdy sinus osiąga wartość zero. W przykładzie tym możemy zaobserwować natomiast błąd numeryczny związany z odwzorowaniem zera oraz jego narastanie.

1.3. Parametry sygnałów

```
fs=1000; %częstotliwość próbkowania
N=1000; %ilość próbek
t=[1:N]*1/fs; %określenie wektora czasu
x=2*sin(2*pi*10*t); %generowanie sygnału
white_noise=0.8*rand(1,1000)-0.4; %generowanie szumu

y=x+white_noise; %sumowanie sygnału i szumu

%generowanie wykresu
plot(t,y)
set(gcf,'color','w');
xlabel('czas [s]')
ylabel('amlituda [-]')
```

```
title(['\rmWykres funkcji \itx(t)=sin(2\cdot\pi\cdot10\cdott)\rm w czasie dla
\itf_{próbkowania}= ', num2str(fs), ' Hz'])
grid on

wart_sr=mean(y) % wartość średnia
wart_max=max(y) % wartość maksymalna
wart_min=min(y) % wartość minimalna

wariancja_ML=var(y) %wariancja przy użyciu wzoru wbudowanego (MatLab korzysta z
wersji nieobciążonej)
wariancja=sum(power(y-wart_sr,2))./(N-1) %własna wersja komendy będąca
estymatorem nieobciążonym

odchylenie_std_ML=std(y) %odchylenie standardowe przy użyciu wzoru wbudowanego
odchylenie_std=sqrt(wariancja) %odchylenie standardowe

energia=sum(y.^2) %energia sygnału

moc_ML=bandpower(y) %moc średnia sygnału przy użyciu wzoru wbudowanego
moc=energia/N %moc średnia sygnału

wart_sk_ML=rms(y) %wartość skuteczna sygnału przy użyciu wzoru wbudowanego
wart_sk=sqrt(moc_sr)%wartość skuteczna sygnału

SNR_ML=snr(x,white_noise)% SNR sygnału wbudowany wzór
moc_syg_baz=sum(x.^2)/N; % moc sygnału bazowego

moc_szumu=sum(white_noise.^2)/N; %moc szumu niezbędna do wyznaczenia SNR
SNR=10*log10(moc_syg_baz/moc_szumu) %SNR funkcja własna
```



Rys. 1.3 Wykres funkcji otrzymanej w zadaniu 1.3

Wyniki:

Otrzymane przy użyciu własnych funkcji wartości parametrów sygnału zgadzały się z tymi otrzymanymi przy użyciu wbudowanych funkcji MatLaba.

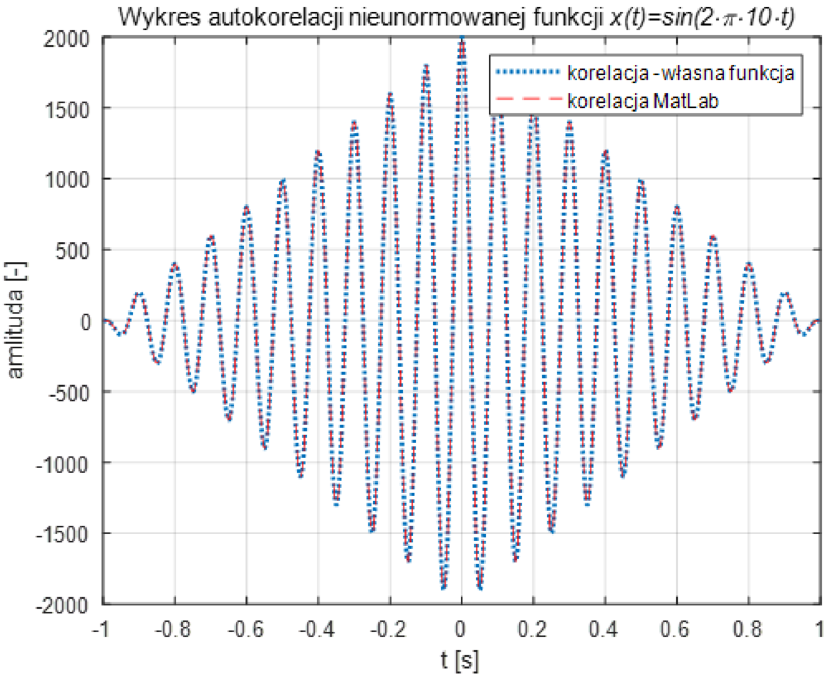
Parametr sygnału	**Obliczona wartość**	**Co odzwierciedla**
Wartość średnia	-0.0089 [j]	Wartość średnia dla idealnego sygnału sinusoidalnego powinna wynosić zero. Różnica mówi nam o występowaniu szumu
Wartość maksymalna	2.3799 [j]	Wartości określają przedział wewnątrz którego zawierają się wszystkie pomiary
Wartość minimalna	-2.3837 [j]	
Odchylenie standardowe	1.4356 [j]	Miara rozrzucenia wartości analizowanej funkcji wokół średniej
Wariancja	2.0609 [j^2]	Średnia arytmetyczna kwadratów odchyleń, niezbędna do obliczenia odchylenia standardowego
Energia	$2.0590 \cdot 10^3$ [j^2]	Wartość potencjalnej energii rozpraszanej przez sygnał na jednostkowym odbiorniku
Moc średnia	2,0460 [j^2]	Wartość ta świadczy o ilości energii generowanej przez sygnał na jednostkowym odbiorniku w czasie w danym czasie
Wartość skuteczna	1.4349 [j]	Jest to wartość amplitudy dla sygnału prostokątnego, który dostarczy tę samą ilość energii co sygnał pierwotny
SNR	15.9029 [dB]	Jest to miara porównująca poziom sygnału użytecznego do poziomu szumu. Im większa wartość tego parametru tym lepiej

1.4. Autokorelacja

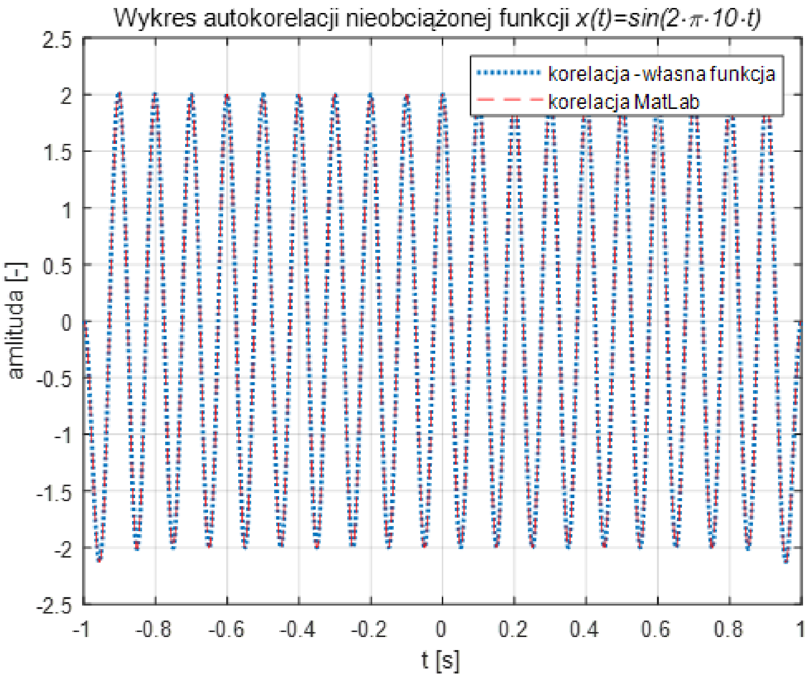
```
fs=1000; %częstotliwość próbkowania
N=1000; %ilość próbek
t=[1:N]*1/fs; %określanie wektora czasu
x=2*sin(2*pi*10*t); %generowanie sygnału

kor_ML=xcorr(x,x);%koreleacja przy użyciu wbudowanej funkcji MatLab

% utworzenie dwóch macierzy, które będą natępnie względem siebie przesuwane
```

```
x1=horzcat(zeros(1,N), x); % dodanie N zer przed wektorem funkcji
x2=horzcat(x, zeros(1,N)); % dodanie N zer za wektorem funkcji

%2 razy mniej obliczeń hehehe
y1 = [x zeros(1,N)];
kor = zeros(1,2*N-1);
kor_unbiased = kor;

for k = 0:N-1
   y_mv = [zeros(1,k) y1(1:end-k)];
   kor(k+N) = sum(y_mv.*y1);
   kor_unbiased(k+N) = kor(k+N)/(N-k);
end

kor(1:N-1) = fliplr(kor(N+1:2*N-1));
kor_unbiased(1:N-1) = fliplr(kor_unbiased(N+1:2*N-1));
kor_biased = kor./N;

t2=[1:length(kor)]*1/fs-1;
%tworzenie wykresu
plot(t2,kor,':','LineWidth',2)
set(gcf,'color','w');
hold on
plot(t2,kor_ML, 'r--')
legend('korelacja - własna funkcja', 'korelacja MatLab')
xlabel('t [s]')
ylabel('amlituda [-]')
title('Wykres autokorelacji funkcji x(t)')
title(['\rmWykres autokorelacji nieunormowanej funkcji
\itx(t)=sin(2\cdot\pi\cdot10\cdott)\rm '])
grid on

figure(2)
plot(t2,kor_biased,':','LineWidth',2)
set(gcf,'color','w');
hold on
plot(t2,xcorr(x,x,'biased'), 'r--')
legend('korelacja - własna funkcja', 'korelacja MatLab')
xlabel('t [s]')
ylabel('amlituda [-]')
title('Wykres autokorelacji funkcji x(t)')
title(['\rmWykres autokorelacji obciążonej funkcji
\itx(t)=sin(2\cdot\pi\cdot10\cdott)\rm '])
grid on

figure(3)
plot(t2,kor_unbiased,':','LineWidth',2)
set(gcf,'color','w');
hold on
plot(t2,xcorr(x,x,'unbiased'), 'r--')
legend('korelacja - własna funkcja', 'korelacja MatLab')
xlabel('t [s]')
ylabel('amlituda [-]')
title('Wykres autokorelacji funkcji x(t)')
title(['\rmWykres autokorelacji nieobciążonej funkcji
\itx(t)=sin(2\cdot\pi\cdot10\cdott)\rm '])
grid on
```

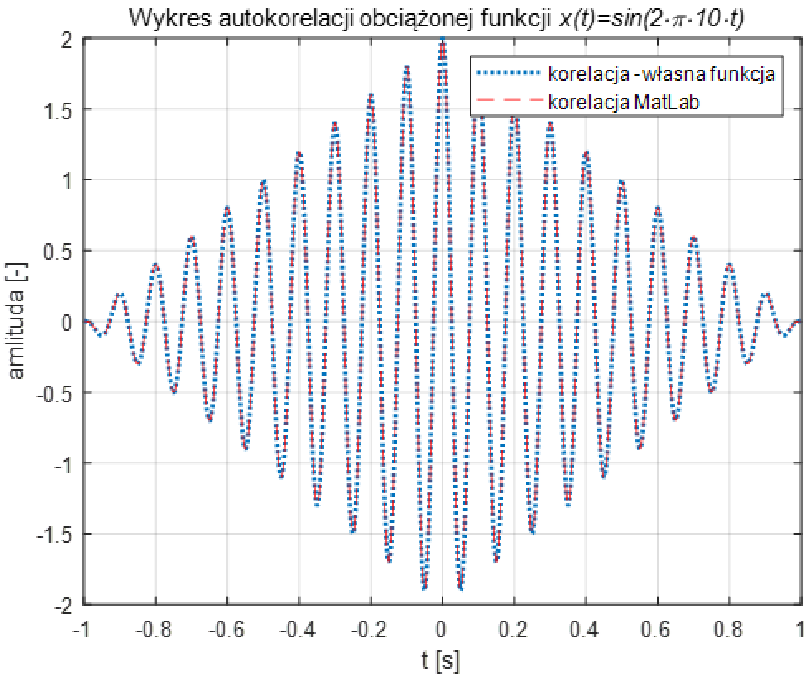
Komentarz [2]: hehe



Rys 1.4a Wykres funkcji autokorelacji nieunormowanej



Rys 1.4b Wykres funkcji autokorelacji nieobciążonej



Rys 1.4b Wykres funkcji autokorelacji obciążonej

Estymator znormalizowany - wartości estymaty wyjściowej są podzielone przez taką wartość (odpowiadającą norm(x)*norm(y)), że współczynnik korelacji przy zerowym opóźnieniu wynosi 1.

Estymator nieobciążony - estymator jest nieobciążony gdy różnica pomiędzy wartością oczekiwaną rozkładu estymatora a wartością szacowanego parametru jest równa zero, w przeciwnym przypadku **estymator** jest **obciążony**. Charakteryzuje się tym że unika losowej, dużej wariancji w końcowych punktach korelowanych sygnałów.

2. Splot sygnałów

2.1. Reprezentacja funkcji splotu

```
clear all;  %wyczyszczenie przestrzeni roboczej

h=[0 1 1 1 0 0 0]; %definiowanie sygnału h
x=[0 1 1 1 1 0 0]; %definiowanie sygnału x

h_fliped=fliplr(h);% obrócenie horyzontalnie macierzy h
```

```
x_new=[zeros(1,length(h)) x]; % utworzenie nowego wektora sygnału x poprzez dodanie przed starym wektorem sygnału zer o ilości równej liczbie próbek sygnału h
h_fliped=[h_fliped zeros(1,length(x))]; % utworzenie nowego wektora sygnału h poprzez dodanie po starym wektorze sygnału zer o ilości równej liczbie próbek sygnału x

figure(1) %wyświetlanie na jednym wykresie przebiegu sygnałów h i x
stem(h_fliped)
hold on
stem(x_new)
legend('h','x')

for n=1:length(x)+length(h)-1
   h_fliped_moved=[zeros(1,n) h_fliped(1:end-n) ]; %przesuwanie macierzy h
   y_conv(n)=sum(h_fliped_moved.*x_new); %sumowanie iloczynów wartości obu funkcji dla danego aktualnego przesunięcia h

   figure(2)
   clf %czyszczenie wykresu
   subplot(2,1,1)
   stem(h_fliped_moved,'r') %prezentacja aktualnego przesunięcia h
   hold all
   stem(x_new,'--b')
   legend('h','x')
   xlabel('m')

   subplot(2,1,2)
   stem(y_conv,'-*k') %wynik kowariancji
   title('Wynik splotu sygnału x[m] i h[m]')
   xlabel('Przesunięcie n')
   pause(1)% zatrzymanie wykonywania się programu na sekundę
end
```

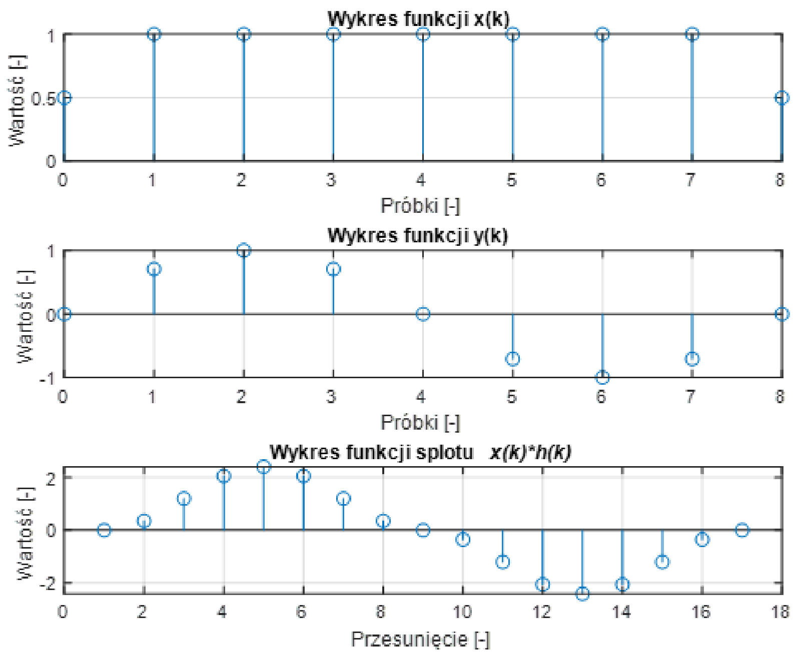
Skrypt wyświetla dwa wykresy. Pierwszy z nich przedstawia wykresy sygnałów x[m] oraz h[-m]. Druga figura, pierwszy podwykres prezentuje, jak w miarę z zmianą opóźnienia n, sygnał transponowany przesuwa się względem pierwotnego w czasie. Natomiast na drugim podwykresie prezentowany jest wyniku splotu sygnałów w trakcie jego obliczania. Ostatnia iteracja prezentuje zatem odpowiedź jednego sygnału przy wymuszeniu będącym tym drugim sygnałem, traktując ten pierwszy sygnał jako odpowiedź impulsową (h to wektor wymuszenia a x to odpowiedź impulsowa układu x lub vice versa).

2.2. Testowanie funkcji splotu

a)

```
k=0:1:8;% wektor kolejnych próbek
%określenie funkcji i ich autokorelacji
x=heaviside(k)-heaviside(k-8);
h=sin(2*pi*k/8).*(heaviside(k)-heaviside(k-8));
y=conv(x,h);

%tworzenie wykresów
subplot(3,1,1)
```

```
set(gcf,'color','w');
stem(x)
set(gcf,'color','w');
xlabel('Próbki [-]')
ylabel('Wartość [-]')
title('Wykres funkcji x(k)')
grid on

subplot(3,1,2)
stem(h)
xlabel('Próbki [-]')
ylabel('Wartość [-]')
title('Wykres funkcji y(k)')
grid on

subplot(3,1,3)
stem(y)
xlabel('Próbki [-]')
ylabel('Wartość [-]')
title('Wykres funkcji splotu x(k)*h(k)')
grid on
```

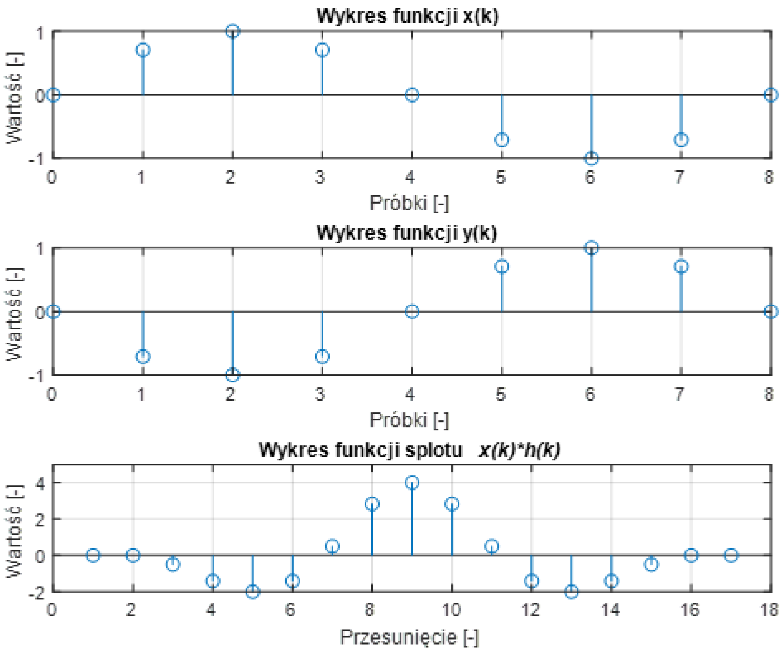


Rys. 2.2a Wykres funkcji otrzymanej w zadaniu 2.2

b)

kod programu różni się jedynie definiowanym sygnałem

```
k=0:1:8
x=sin(2*pi*k/8).*(heaviside(k)-heaviside(k-8));
h=-sin(2*pi*k/8).*(heaviside(k)-heaviside(k-8));
y=conv(x,h)
```



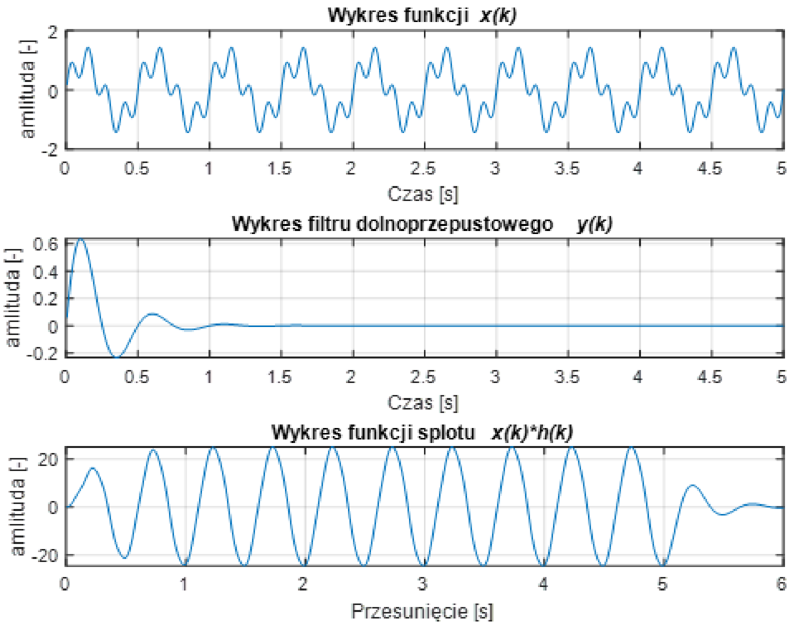
Rys. 2.2b Wykres funkcji otrzymanej w zadaniu 2.2

Zarówno w przykładzie a) jak i b), podobnie jak w zadaniu 2.1, obserwujemy realizację splotu dwóch sygnałów, z których jeden jest sygnałem wyjściowym a drugi odpowiedzią impulsową obiektu. Nie trudno zauważyć związek pomiędzy korelacją a splotem dwóch sygnałów. Polega on na sumowaniu ilorazów wartości sygnałów w danej próbce czasowej przy ich względnym przesunięciu tylko że dla splotu jeden z sygnałów jest odbity względem osi wartości.

2.3. Splot - filtracja

```
N=1000; % ilość próbek
T=5 %czas trwania sygnału
t=(T./N):(T./N):T; %wektor czasu

%określenie funkcji i ich autokorelacji
x=sin(2*pi*2*t)+0.5*sin(2*pi*8*t);
h=sin(2*pi*2*t).*exp(-4*t);
y=conv(x,h); %korelacja sygnałów

%generowanie wykresów
subplot(3,1,1)
plot(t,x)
xlabel('Czas [-]')
ylabel('amlituda [-]')
title('Wykres funkcji \itx(k)')
```

```
grid on

subplot(3,1,2)
plot(t,h)
xlabel('Czas [-]')
ylabel('amlituda [-]')
title('Wykres funkcji \ity(k)')
grid on

subplot(3,1,3)
tt=(T/N):(T/N):(length(x)+length(h)-1)*(T/N);
plot(tt,y)
xlabel('Przesunięcie [s]')
ylabel('amlituda [-]')
title('Wykres funkcji splotu \itx(k)*h(k)')
grid on
set(gcf,'color','w');
```



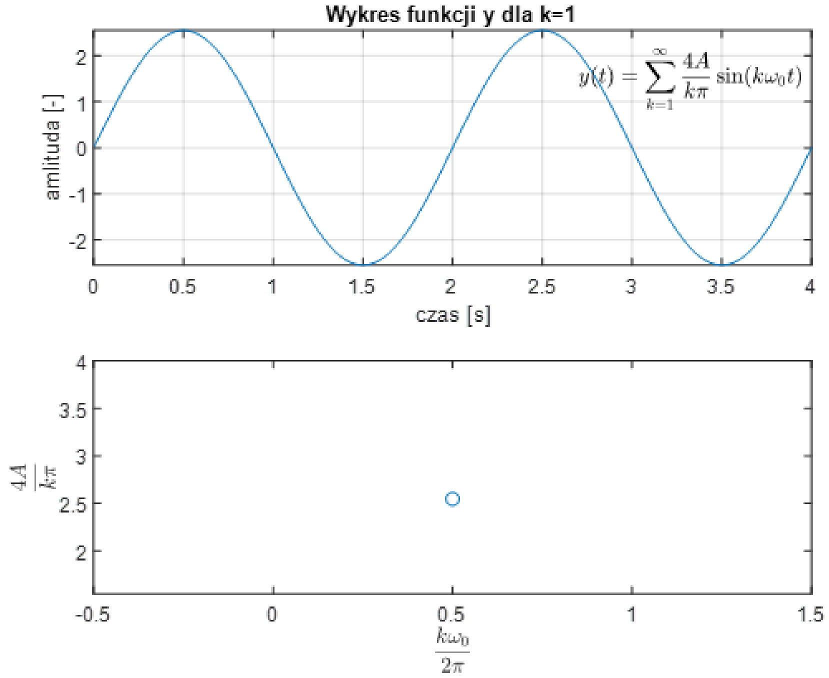
Rys. 2.3 Wykres funkcji otrzymanej w zadaniu 2.3

Celem powyższego zadania było zaprezentowanie działania filtru dolnoprzepustowego (h) o częstotliwości granicznej 2 Hz, na którego wejście podano sygnał (x). Za pomocą splotu wyznaczono odpowiedź, którą jest odszumiony (usunięto wpływ 8 Hz zakłócenia) sygnał sinusoidalny o czasie trwania równym 5 sekund. Ponadto możemy zaobserwować odpowiedź po ustaniu sygnału wymuszającego oraz początkową bezwładność układu co świadczy o magazynach energii w nim występujących.

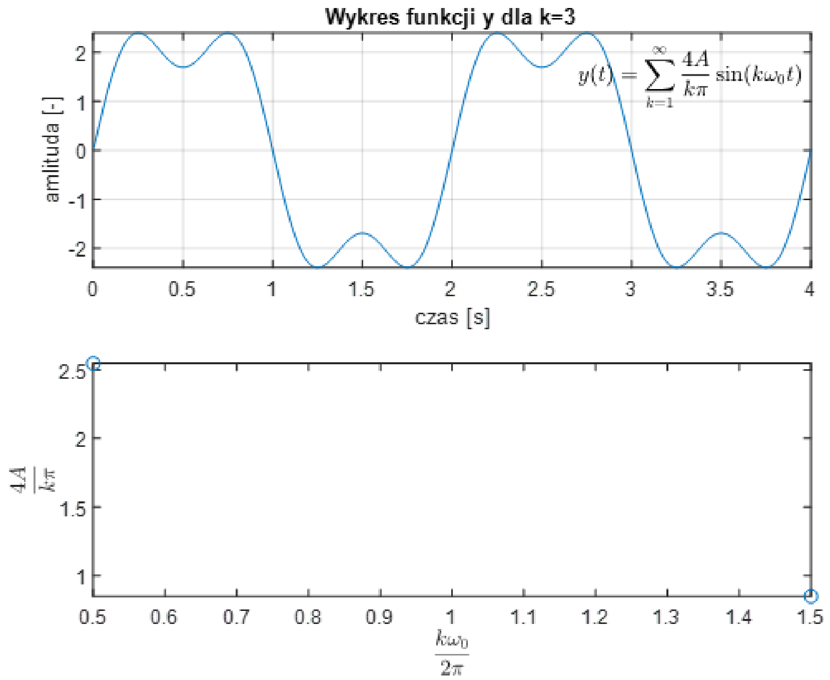
3. Sygnały i ich parametry

3.1. Szereg Fouriera

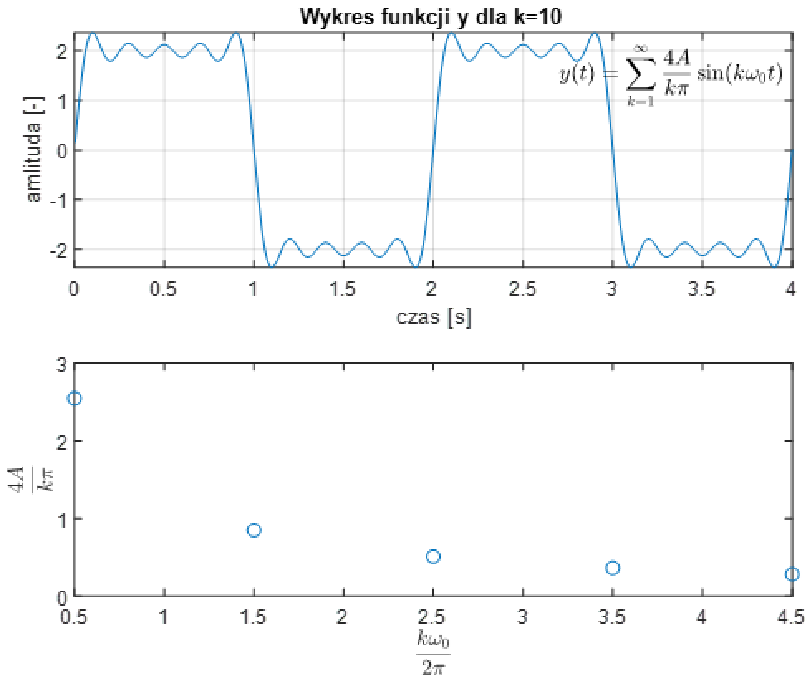
```
A=2;       % amplituda sygnału
w0=pi;   % częstość
ilosc_k=[1 3 10 20 100]; %wektor maksymalnych składowych
t=(T./N):(T./N):T; % wektor czasu

for j=1:length(ilosc_k) % pętla w celu uzyskania wykresu dla każdej zadanej
ilości składowych
y=zeros(1,N); %zerowanie macierzy
     for i=1:length(t)
          for k=1:2:ilosc_k(j) % sumowanie składowych
            y(i)=y(i)+(4*A/(k*pi)*sin(k*w0*t(i)));
          end
     end

figure(j)
set(gcf,'color','w');
subplot(2,1,1)
plot(t,y)
xlabel('czas [s]')
ylabel('amlituda [-]')
title(['Wykes funkcji y dla k=', num2str(ilosc_k(j))])
grid on
txt='$$ y(t)=\displaystyle\sum_{k=1}^{\infty} \frac{4A}{k\pi} \sin (k\omega_0t)
$$';
text(2.7,1.5,txt,'Interpreter','latex')
subplot(2,1,2)
k=1:2:ilosc_k(j);
plot( (k*w0/(2*pi)),4*A./(k*pi),'o' )
ylabel('$$ \frac{4A}{k\pi} $$','Interpreter','latex')
xlabel('$$ \frac{k\omega_0}{2\pi} $$','Interpreter','latex')
end
```



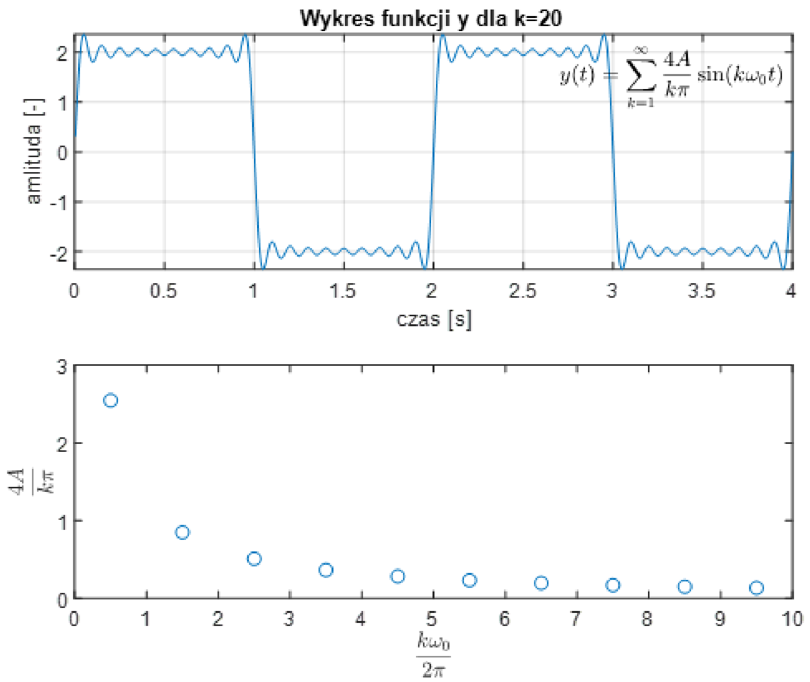
Rys. 3.1a Wykres funkcji otrzymanej w zadaniu 3.1



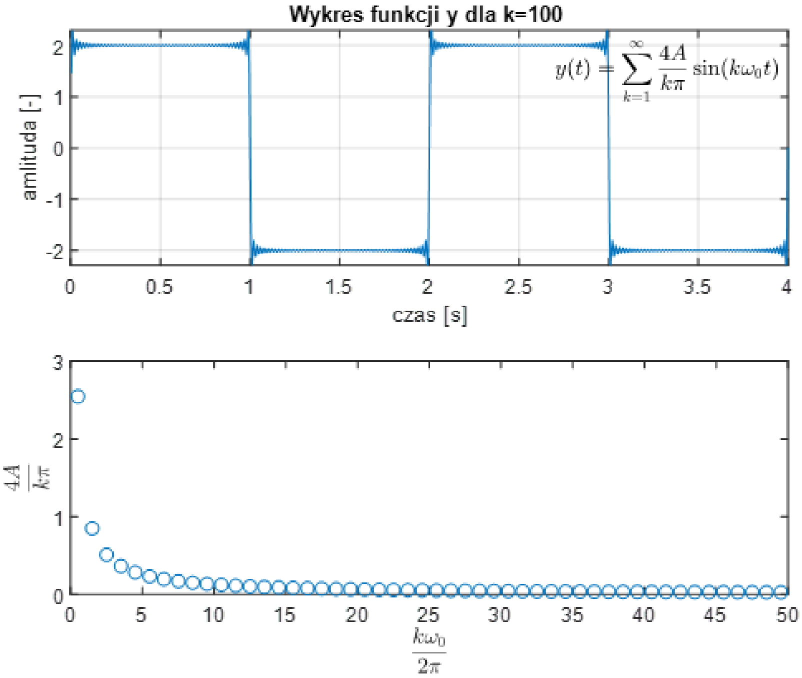
Rys. 3.1b Wykres funkcji otrzymanej w zadaniu 3.1



Rys. 3.1c Wykres funkcji otrzymanej w zadaniu 3.1



Rys. 3.1d Wykres funkcji otrzymanej w zadaniu 3.1



Rys. 3.1d Wykres funkcji otrzymanej w zadaniu 3.1

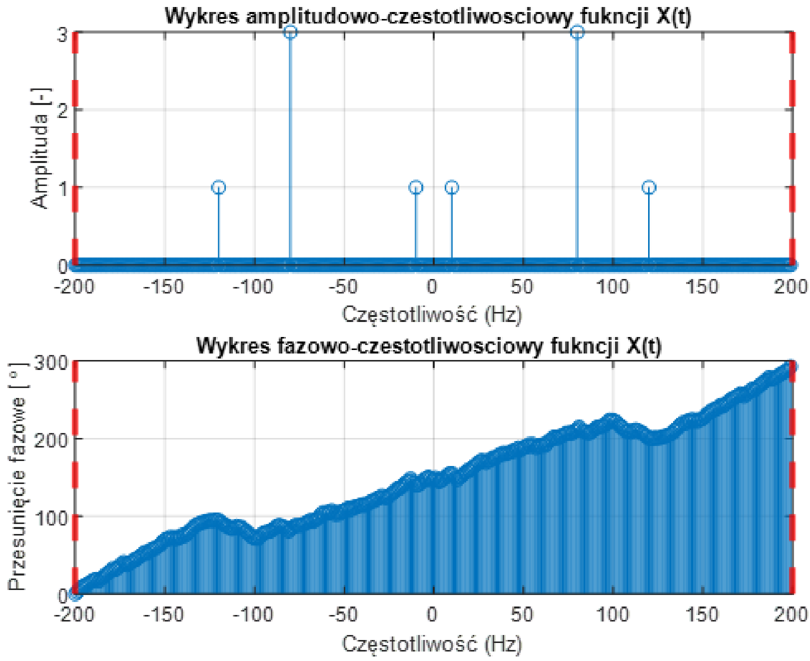
Wraz ze wzrostem ilości składowych wzrasta dokładność odwzorowania sygnału prostokątnego. Dla małych k czyli dla małych częstotliwości składowego sinusa wartość współczynnika przed sinusem jest największa. Dodawanie kolejnych składowych o dużych częstotliwościach i małej amplitudzie wpływa na zwiększenie dokładności odwzorowania.

Komentarz [3]: efekt gibbsa

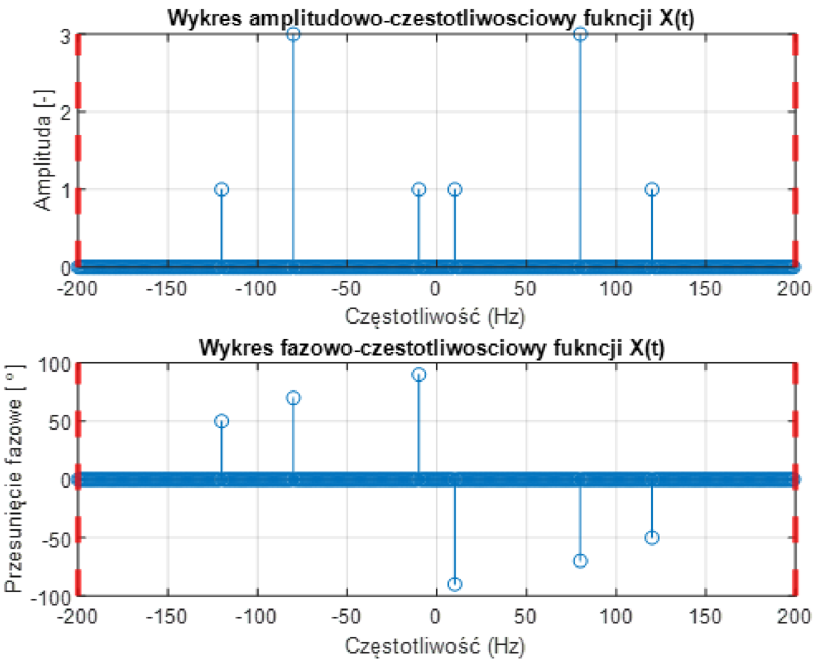
3.2. Transformata Fouriera

```
f1=10;
f2=80;
f3=120;
A=2;
fs = 400; % częstotliwość próbkowania
T = 1/fs; % okres próbkowania
t = 0:T:1-T; %wektor czasu
N=length(t); % zapisanie długości wektora czasu do późniejszych obliczeń

fn=fs/2; %częstotliwość Nyquista

%tworzenie trzech składowych
y1 = A*sin(2*pi*f1*t);
y2 = 3*A*sin(2*pi*f2*t+deg2rad(20));
y3 = A*sin(2*pi*f3*t+deg2rad(40));
y=y1+y2+y3; % złożenie sygnałów

Y1=fft(y); %szybka transformata Fouriera
```

```
Y2 = 1/N*fftshift(Y1);%przeskalowanie wyniku FFT i odpowiednie rozłożenie
częstotliwości

df=fs/N; %określenie zmian częstotliwości
f=(-N/2:N/2-1)*df; %wektor częstotliwości
M=abs(Y2); %obliczanie amplitudy


%małe wartości(zaokrąglone zera) składowych rzeczywistych i urojonych
%programu powodują błędne określenie kąta
X2=Y2;
prog = max(abs(Y2))/10000; %określenie progu tolerancji
X2(abs(Y2)<prog) = 0; %dla małych wartości - przyrównane do zera
Fi=rad2deg(angle(X2)); % obliczanie częstotliwości, przeliczanie na stopnie
Fi2=angle(Y2);% faza bez filtracji do zadania 3.4

%Generowanie wykresów
subplot(2,1,1)
set(gcf,'color','w');
stem(f,M)
title('Wykres amplitudowo-częstotliwościowy funkcji X(t)')
xlabel('Częstotliwość (Hz)')
ylabel('Amplituda [-]')
xline(fn, '--r', 'Linewidth', 3);
xline(-fn, '--r', 'Linewidth', 3);
grid on

subplot(2,1,2)
stem(f,Fi)
title('Wykres fazowo-częstotliwościowy funkcji X(t)')
xlabel('Częstotliwość (Hz)')
ylabel('Przesunięcie fazowe [\circ]')
xline(fn, '--r', 'Linewidth', 3);
xline(-fn, '--r', 'Linewidth', 3);
grid on
```



Rys. 3.2a Wykres funkcji otrzymanej w zadaniu 3.2



Rys. 3.2b Wykres funkcji otrzymanej w zadaniu 3.2

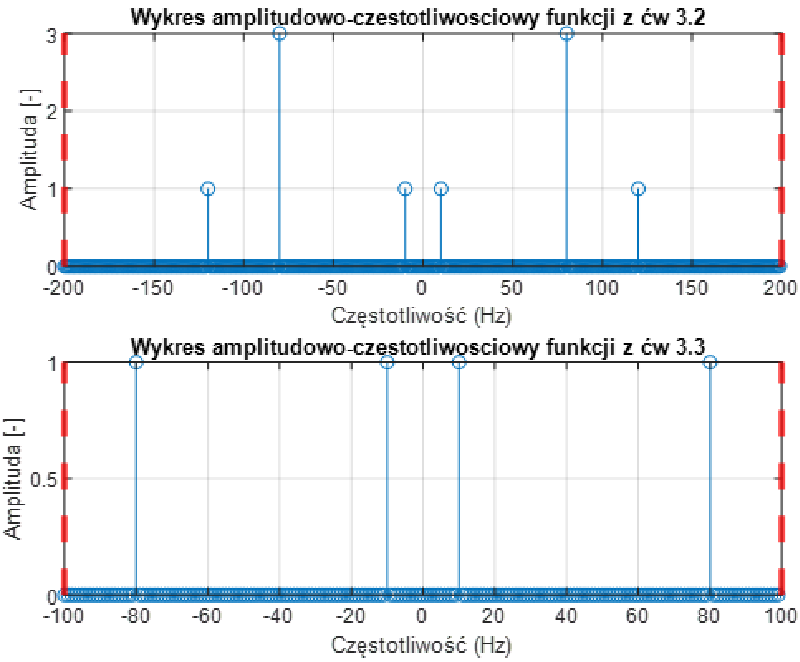
Komentarz [4]: trochę inny wykres powinien wyjść

Otrzymane wykres amplitudowo częstotliwościowy jest funkcją parzystą na której widać 6 charakterystycznych prążków, symetrycznie po trzy na półosi dodatniej i ujemnej. Oznacza to że mamy styczność z sygnałem okresowym o trzech składowych równych 10, 80, i 120 Hz z amplitudami wynoszącymi kolejno 1, 3 i 1 jednostka. Są one dwukrotnie mniejsze niż w sygnale oryginalnym, ze względu na występowanie składowych ujemnych (rozkład amplitudy po równo na składowe dodatnie i ujemne). Czerwonymi przerywanymi liniami zaznaczono częstotliwość Nyquist'a odpowiadającą maksymalnej częstotliwości możliwej do poprawnego zarejestrowania (bez zjawiska aliasingu) przy zadanej częstotliwości próbkowania. Wykres fazowo częstotliwościowy jest nieparzysty a widoczne na nim prążki różnią się od siebie o 20 stopni. Otrzymanie tego wykresu było możliwe dopiero w momencie odrzucenia wartości bliskich zeru z transformaty sygnału, w przeciwnym wypadku otrzymywano niepoprawne odwzorowanie (widoczne na **3.2a**). Na podstawie parzystości widma amplitudowego i nieparzystości widma fazowego można stwierdzić że analizowany sygnał jest sygnałem rzeczywistym.

3.3. Transformata Fouriera – aliasing

```
%Wywołać zad3_2.m
subplot(2,1,1)
set(gcf,'color','w');
stem(f,M)
title('Wykres amplitudowo-częstotliwościowy funkcji z ćw 3.2')
xlabel('Częstotliwość (Hz)')
ylabel('Amplituda [-]')
xline(fn, '--r', 'Linewidth', 3);
xline(-fn, '--r', 'Linewidth', 3);
grid on

f1=10;
f3=120;
A=2;
fs = 200; % częstotliwość próbkowania
T = 1/fs; % okres próbkowania
t = 0:T:1-T; %wektor czasu
N=length(t); % zapisanie długości wektora czasu do późniejszych obliczeń

fn=fs/2; %częstotliwość Nyquista

%tworzenie składowych
y1 = A*sin(2*pi*f1*t);
y3 = A*sin(2*pi*f3*t+deg2rad(40));
y=y1+y3; % złożenie sygnałów

Y1=fft(y); %szybka transformata Fouriera


Y2 = 1/N*fftshift(Y1);%przeskalowanie wyniku FFT i odpowiednie rozłożenie
częstotliwości

df=fs/N; %określenie zmian częstotliwości
f=(-N/2:N/2-1)*df; %wektor częstotliwości
M=abs(Y2); %obliczanie amplitudy

subplot(2,1,2)
stem(f,M)
title('Wykres amplitudowo-częstotliwościowy funkcji z ćw 3.3')
```

```
xlabel('Częstotliwość (Hz)')
ylabel('Amplituda [-]')
xline(fn, '--r', 'Linewidth', 3);
xline(-fn, '--r', 'Linewidth', 3);
grid on
```



Rys. 3.3 Wykres funkcji otrzymanej w zadaniu 3.3

W sygnale poddawanym transformacie Fouriera z zadania 3.3 znajduje się składowa o częstotliwości większej od częstotliwości Nyquist'a, przez co pojawiają się one na widmie amplitudowym jako prążki odsunięty od częstotliwości granicznych o różnicę pomiędzy prawdziwą wartością częstotliwości a częstotliwością graniczną. Bez tej wiedzy można błędnie podejrzewać że sygnał w zadaniu 3.3 zawiera składowe o częstotliwościach 10 i 80 Hz, gdy w rzeczywistości wynoszą one 10 i 120 Hz.

3.4. Odwrotna Transformata Fouriera

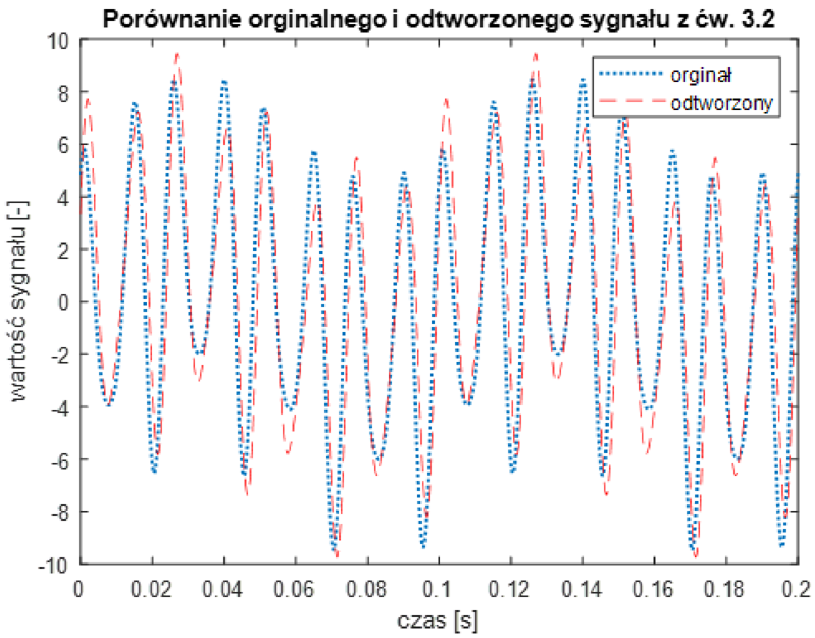
W programie 3.2 dodano poniższą komendę oraz zwiększono częstotliwość próbkowania do 4000 Hz:

```
save ('wyniki3_2.mat','f','M','Fi2','y');
```

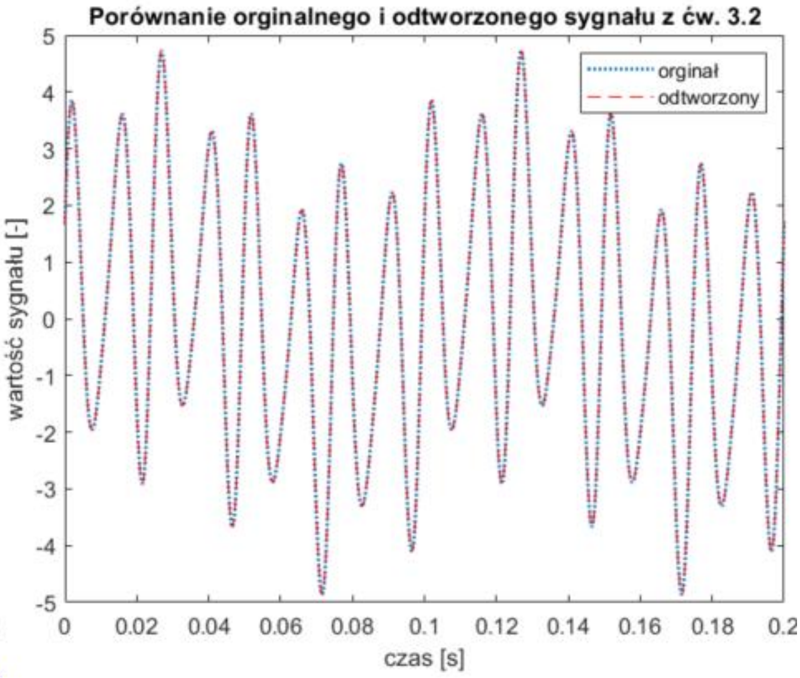
Program 3.4:

```
load('wyniki3_2.mat')
```

```
N=4000;
fs=4000;
T = 1/fs; % okres próbkowania
t = 0:T:1-T; %wektor czasu

Y=M.*cos(Fi2)+i*M.*sin(Fi2);
y2=N*ifft(ifftshift(Y));

t = [0:1:length(y)-1]./fs;
plot(t,y2,':');
hold on
plot(t,y, 'r--')
set(gcf,'color','w');
xlim([0 0.1])
title('Porównanie oryginalnego i odtworzonego sygnału z ćw. 3.2')

legend('oryginał','odtworzony')
xlabel('czas [s]')
ylabel('wartość sygnału [-]')
```



Rys. 3.4a Wykres funkcji otrzymanej w zadaniu 3.4 (faza po filtracji)



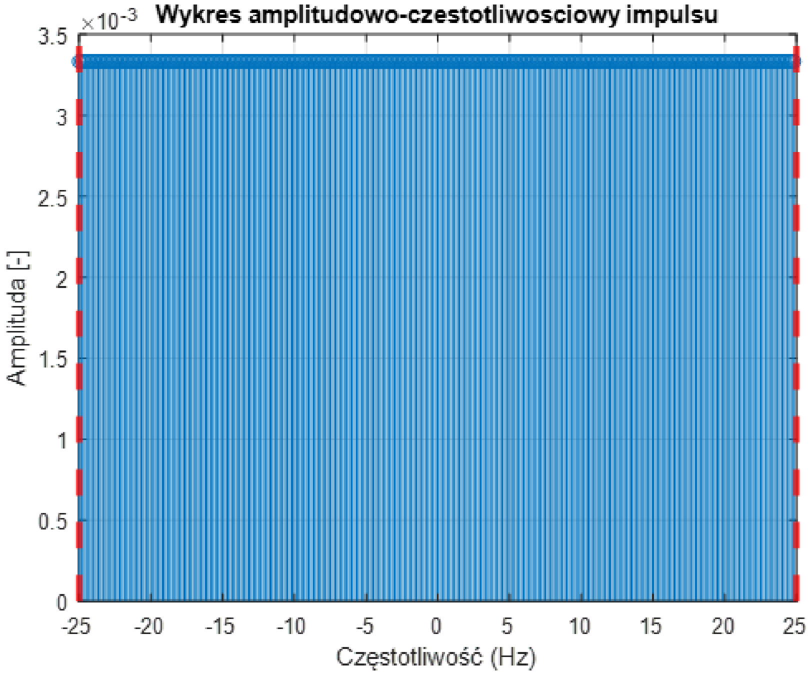
Rys. 3.4b Wykres funkcji otrzymanej w zadaniu 3.4 (faza przed filtracją)

Do odtworzenia sygnału wykorzystano dane wygenerowane do narysowania wykresów w zadaniu 3.2. Okazało się że wykorzystanie wartości argumentów po filtracji (która umożliwia odfiltrowanie małych wartości z FFT - uwydatnienie charakterystycznych prążków) powoduje nieznaczne różnice między oryginalnym a odtworzonym sygnałem (**Rys 3.4a)**. Do wygenerowania wykresu (**Rys 3.4b**) użyto sygnału przed filtracją. Umożliwiło to uzyskanie przebiegu dokładnie pokrywającego się z oryginałem.

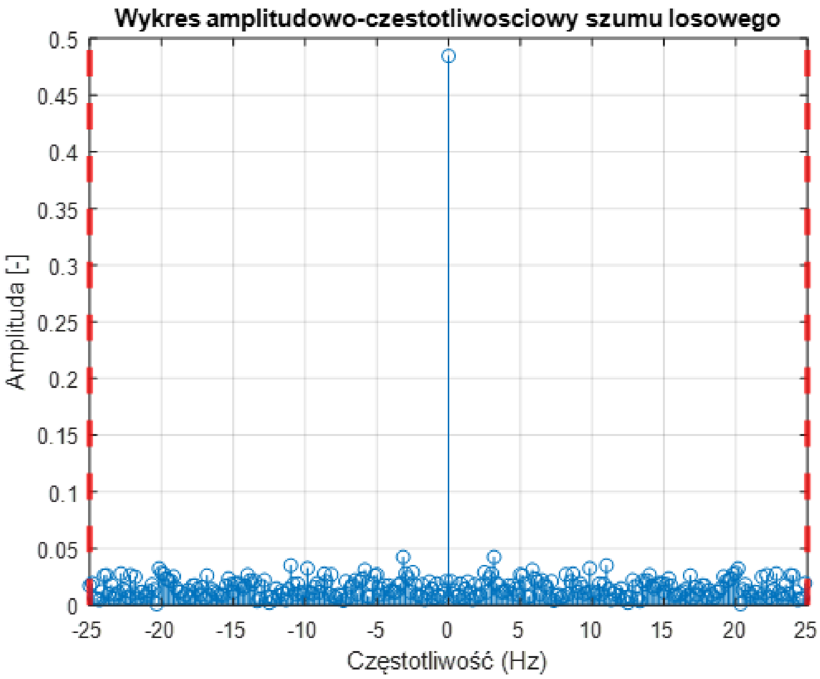
3.5. Transformata Fouriera dla wybranych przebiegów

```
fs = 50; % częstotliwość próbkowania
T = 1/fs; % okres próbkowania
t = -3:T:3-T; %wektor czasu
N=length(t); % zapisanie długości wektora czasu do późniejszych obliczeń
fn=fs/2; %częstotliwość Nyquista

%tworzenie sygnału
y1=dirac(t);
idx = y1 == Inf; % znalezienie indexu w którym następuje pik
y1(idx) = 1;  % przypisanie mu realnej wartości
y2=rand(1,length(t));
y3=heaviside(t-2);
y4=sawtooth(2*pi*t,0.5);

Y1 = 1/N*fftshift(fft(y1));%przeskalowanie wyniku FFT i odpowiednie rozłożenie
częstotliwości
Y2 = 1/N*fftshift(fft(y2));
```

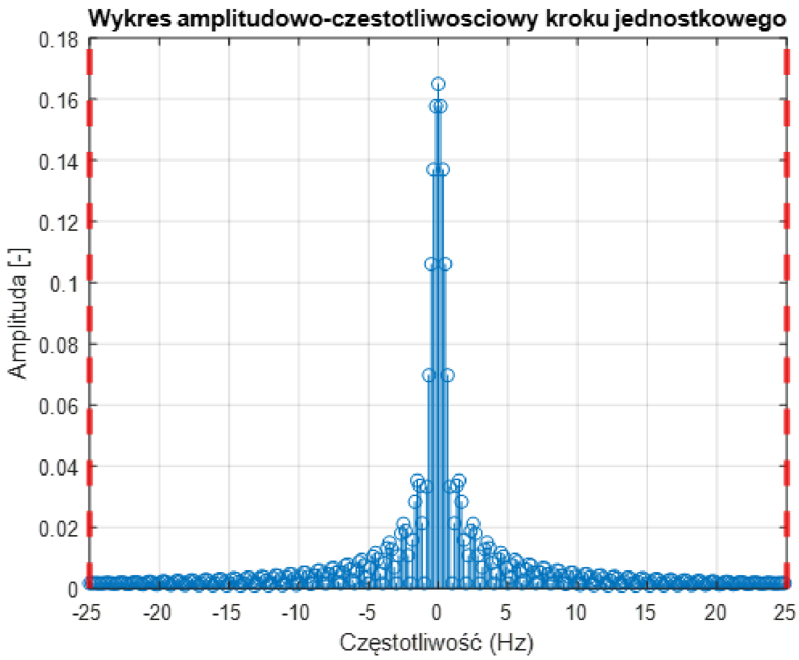
```
Y3 = 1/N*fftshift(fft(y3));
Y4 = 1/N*fftshift(fft(y4));

df=fs/N; %określenie zmian częstotliwości
f=(-N/2:N/2-1)*df; %wektor częstotliwości
M1=abs(Y1);M2=abs(Y2);M3=abs(Y3);M4=abs(Y4); %obliczanie amplitudy
```



Rys. 3.5a Charakterystyka amplitudowo fazowa impulsu

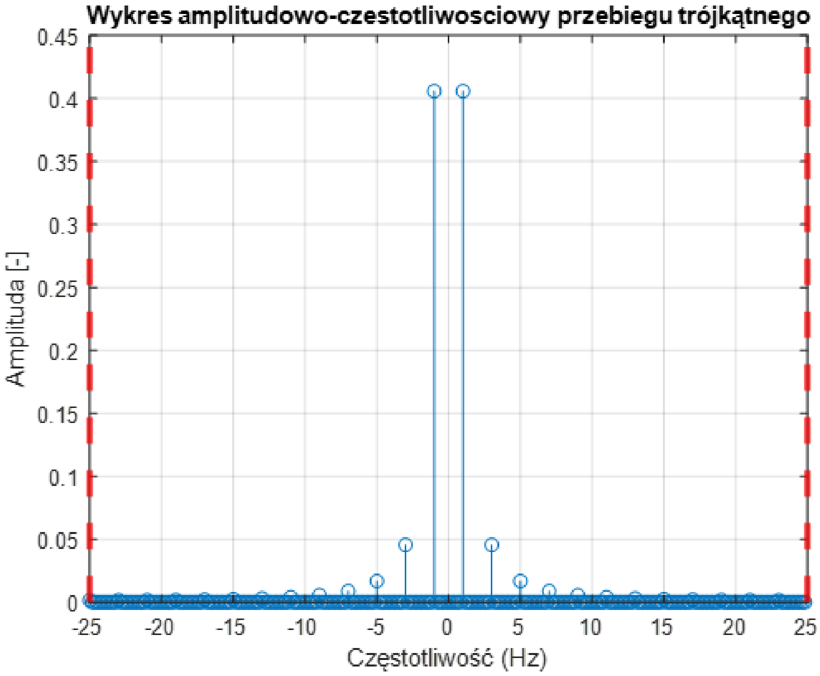


Rys. 3.5b Charakterystyka amplitudowo fazowa impulsu



Rys. 3.5c Charakterystyka amplitudowo fazowa kroku jednostkowego

Komentarz [5]: źle zdefiniowano szum (powinien oscylować wokół 0)



Rys. 3.5a Charakterystyka amplitudowo fazowa przebiegu trójkątnego