SPRAWOZDANIE 7

Zajęcia nr 8 - Próbkowanie i przekształcanie sygnałów ciągłych,

Michał Midor, gr. 5 - 26.11.2025, godz: 16:45

Wprowadzenie:

Celem ćwiczenń laboratoryjnych jest implementacja, weryfikacja oraz analiza właściwości Dyskretnej Transformacji Fouriera (DFT) w środowisku MATLAB. DFT stanowi numeryczny odpowiednik transformacji Fouriera dla sygnałów o skończonym czasie trwania i dyskretnej dziedzinie czasu. Kluczowym aspektem ćwiczenia jest zrozumienie relacji między parametrami próbkowania (f_p, N(i)) a rozdzielczością widmową, a także analiza zjawisk takich jak przeciek widma spectral leakage oraz aliasing. W ramach laboratorium badany jest wpływ okien czasowych na redukcję listków bocznych widma oraz weryfikowana jest liniowość przekształcenia.

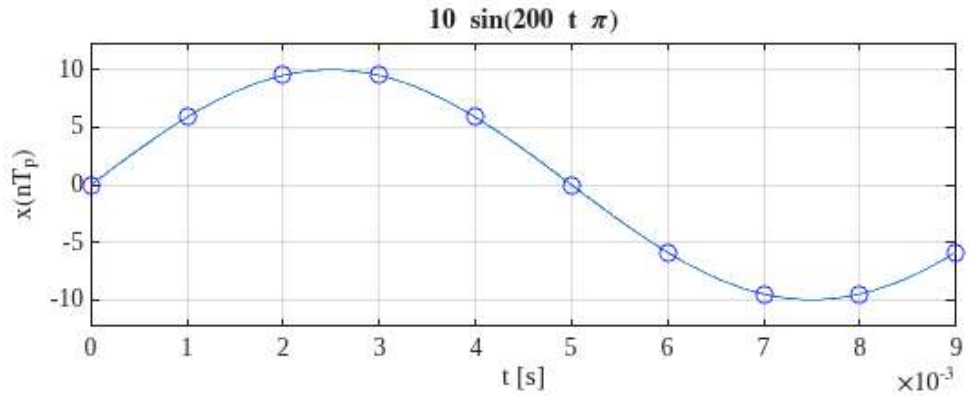
Ćwiczenia 1-5:

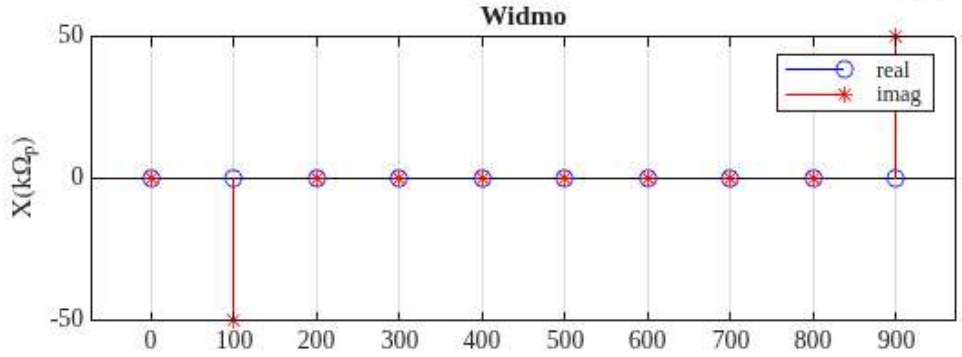
```
clear all; close all;
syms t w

N = [10, 15, 20]; % liczba próbek, zadanie 5
fp = 1000;%Hz
Tp = 1/fp;
A0 = 5;
A1 = 10;
f1 = 100; %Hz

x1 = A1*sin(2*pi*f1*t);
x = x1;
for i = 1:length(N)
    tn = [0:N(i)-1]*Tp; % wsp. czasowe próbek
    xn = double(subs(x,t,tn));
    Xk = zeros(1,N(i));

    for k = 0:N(i)-1 % impl. wzoru (8)
    for n = 0:N(i)-1
    % Zadanie 1)
    % Wprowadzenie właściwego wzoru (8) ze skryptu, wyliczającego DFT, punkt
po
    % punkcie.
    Xk(k+1) = Xk(k+1) + xn(n+1) * exp(-1j * 2 * pi * k * n / N(i));
    end
    end
    Xk
```

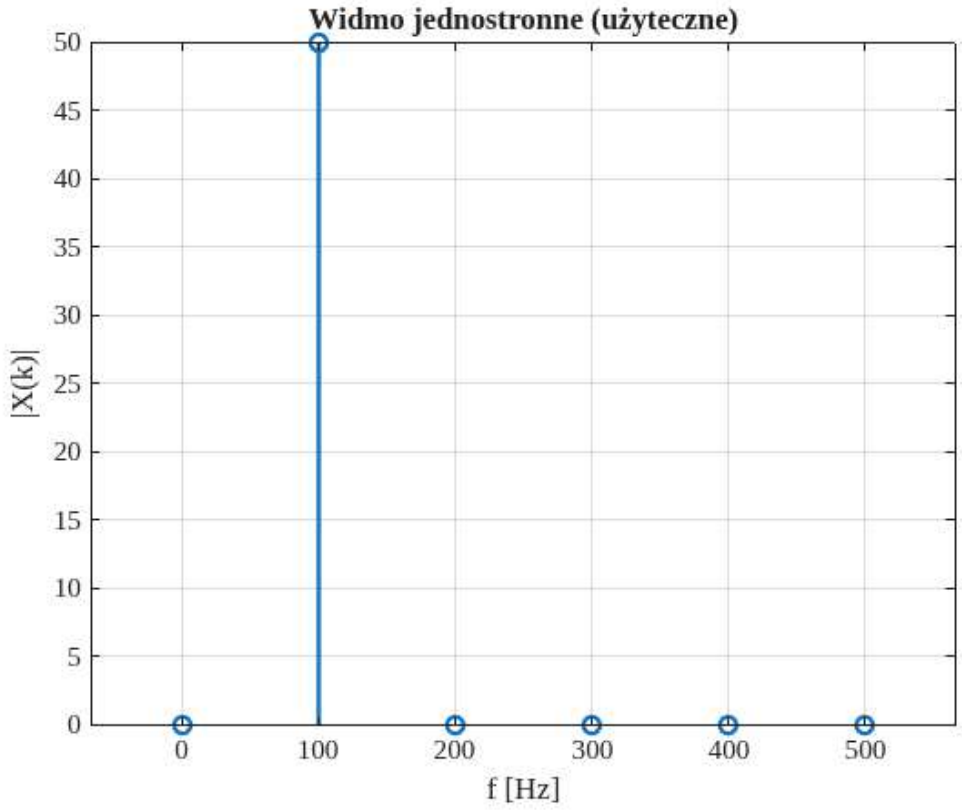
```
    figure('Name', 'Zadanie 1 i 2 - Poprawiona oś X');
    subplot(2,1,1);

    ezplot(x,[tn(1),tn(N(i))]); hold on; grid on

    plot(tn, xn,'ob');
    xlabel('t [s]'); ylabel('x(nT_p)');
    subplot(2,1,2)

    % Zadanie 2 - Należy przeliczyć oś X na Herce za pomocą wzoru f = k*f_p/
N(i).

    f_axis = (0:N(i)-1) * (fp / N(i));
    stem(f_axis, real(Xk),'ob'); grid on, hold on
    stem(f_axis, imag(Xk),'*r'); % Prążek występuje również w 900 Hz, bo to
symetryczne odbicie 100 Hz po lewej stronie osi Y (900-1000=-100)

    title('Widmo'),
    ylabel('X(k\Omega_p)'); %xlabel('f [Hz]')
    legend('real','imag')

    % Zadanie 3 - usunięcie ujemnych częstotliwości z widma

    N_half = floor(N(i)/2) + 1;
    Xk_half = Xk(1:N_half);
    f_axis_half = f_axis(1:N_half);

    figure('Name', 'Zadanie 3 - Widmo Jednostronne'); % Ograniczone do
częstotliwości Nyquista - 500 Hz
    stem(f_axis_half, abs(Xk_half), 'LineWidth', 1.5);
    grid on; xlabel('f [Hz]'); ylabel('|X(k)|');
    title('Widmo jednostronne (użyteczne)');

    % Zadanie 4 - Przejście z liczb zespolonych na amplitudę i fazę.

    tol = 1e-5; % Tolerancja błędów numerycznych
    Xk_clean = Xk;
    Xk_clean(abs(Xk) < tol) = 0; % Zerowanie szumu

    Amplituda = abs(Xk_clean)
    Faza = angle(Xk_clean) % Kąt w radianach

    figure('Name', 'Zadanie 4 - Amplituda i Faza');
    subplot(2,1,1);
    stem(f_axis, Amplituda, 'b', 'LineWidth', 1.5); grid on;
    title('Widmo Amplitudowe'); ylabel('|X(f)|');

    subplot(2,1,2);
    stem(f_axis, Faza, 'r', 'LineWidth', 1.5); grid on;
    title('Widmo Fazowe'); ylabel('Faza [rad]'); xlabel('f [Hz]');
```

```
end
```

```
Xk = 1×10 complex
  -0.0000 + 0.0000i   0.0000 -50.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i ···
```

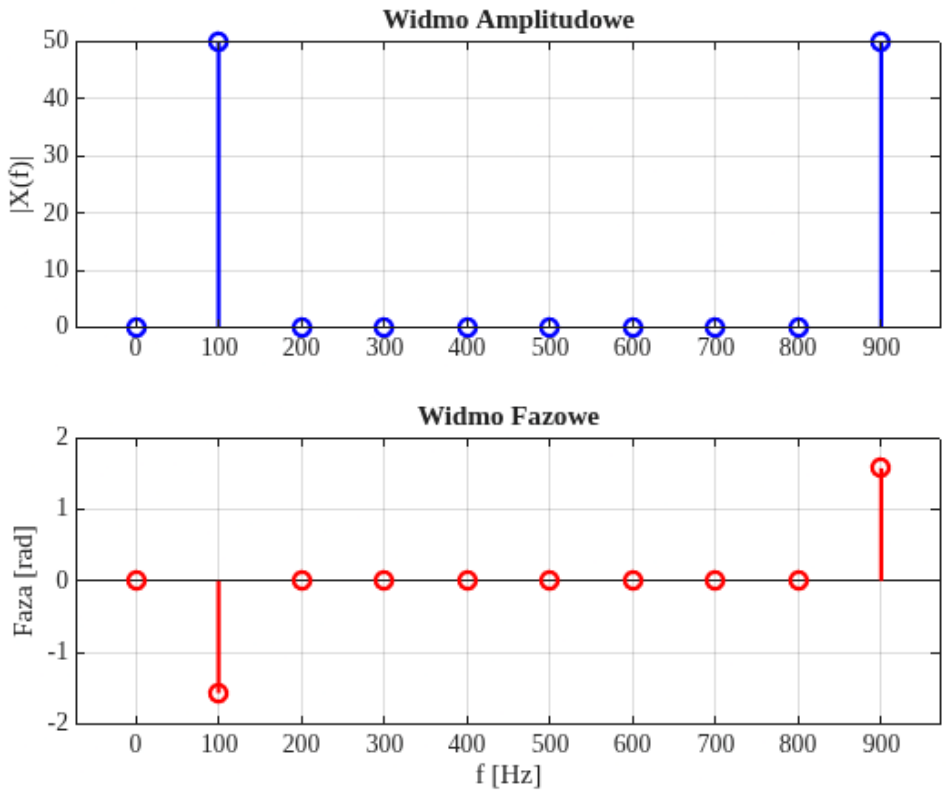




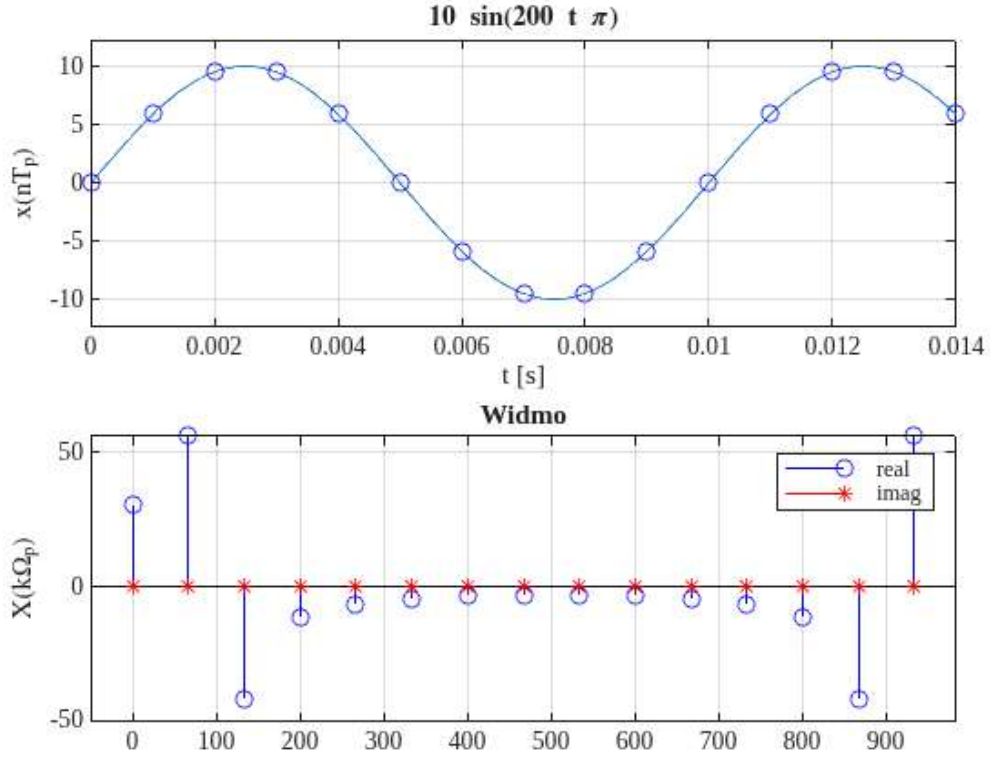


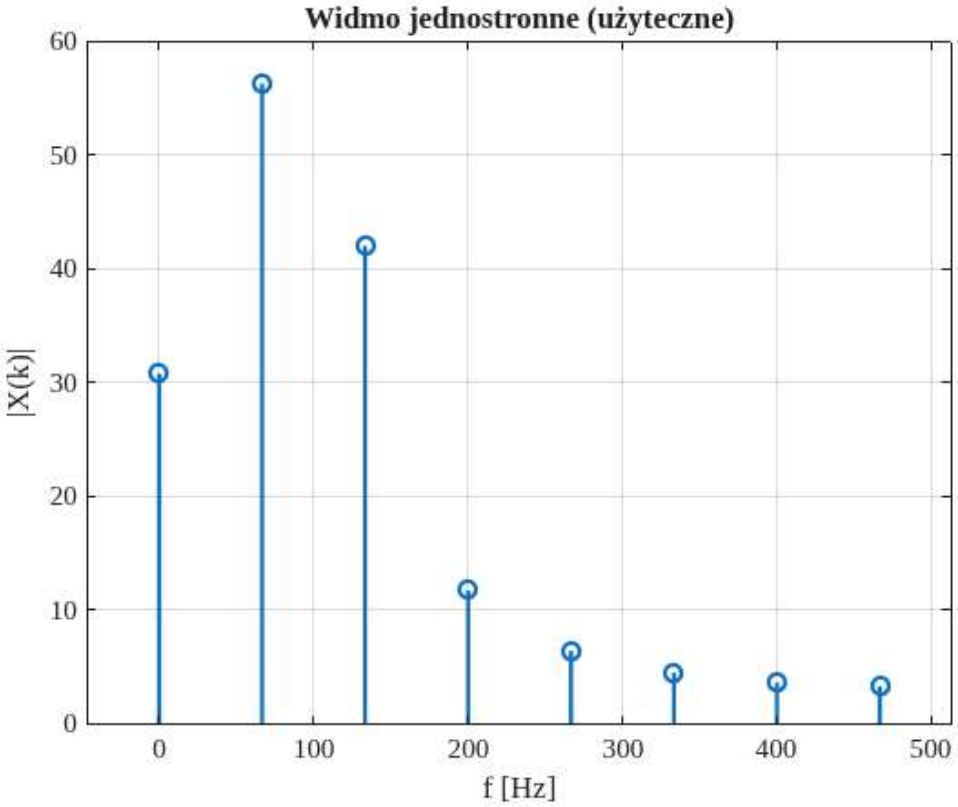

```
Amplituda = 1×10
```

```
         0   50.0000         0         0         0         0         0         0 ···
Faza = 1×10
         0   -1.5708         0         0         0         0         0         0 ···
```

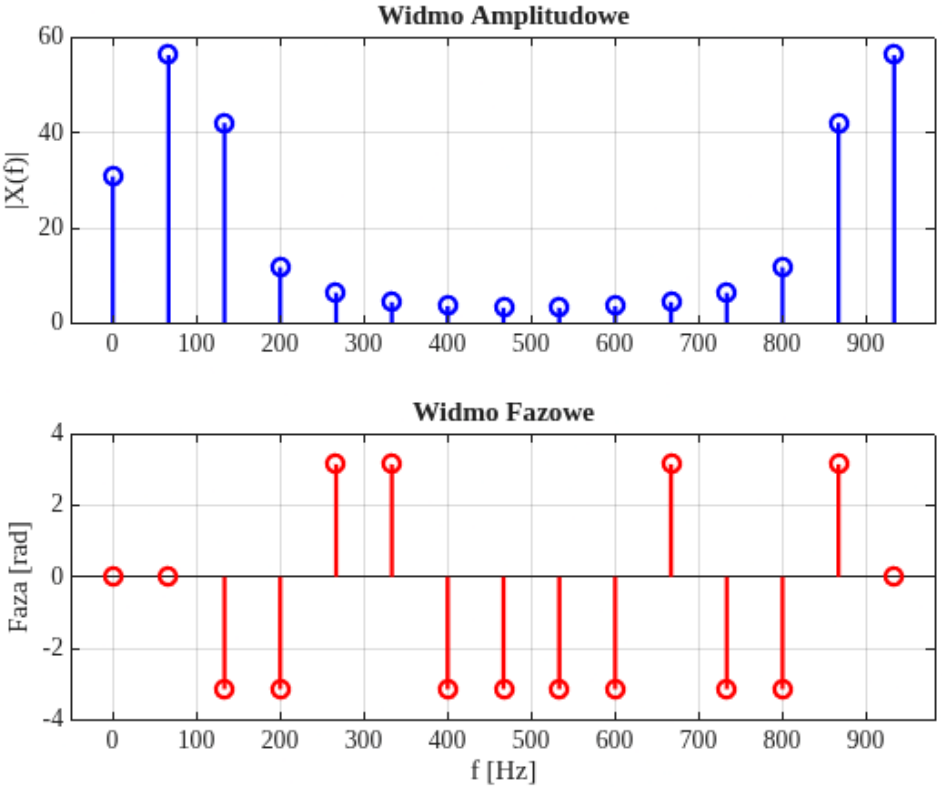



```
Xk = 1×15 complex
  30.7768 + 0.0000i  56.2321 + 0.0000i -42.0188 - 0.0000i -11.7557 - 0.0000i ···
```



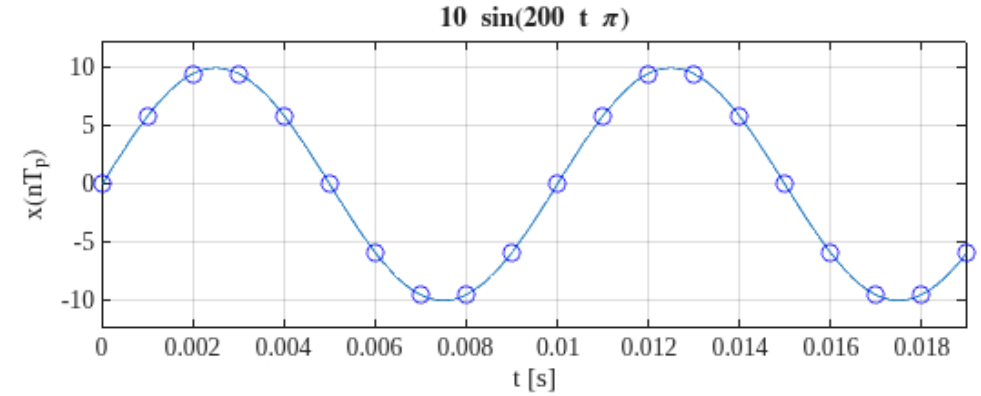


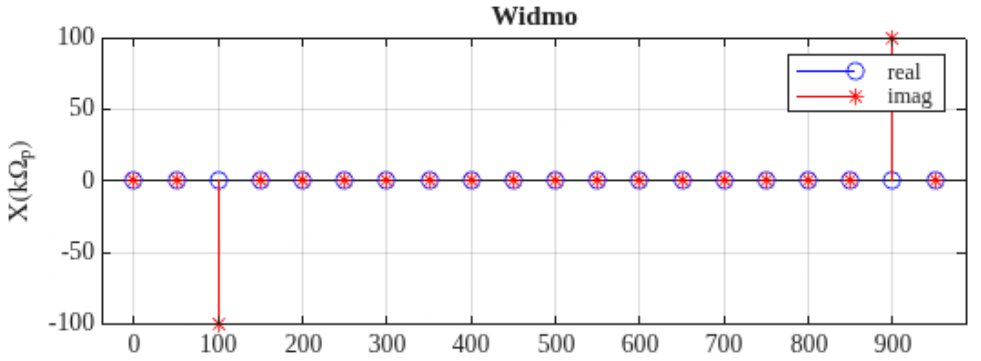

```
Amplituda = 1×15
   30.7768   56.2321   42.0188   11.7557    6.4341    4.4903    3.6327    3.2889 ···
Faza = 1×15
         0    0.0000   -3.1416   -3.1416    3.1416    3.1416   -3.1416   -3.1416 ···
```

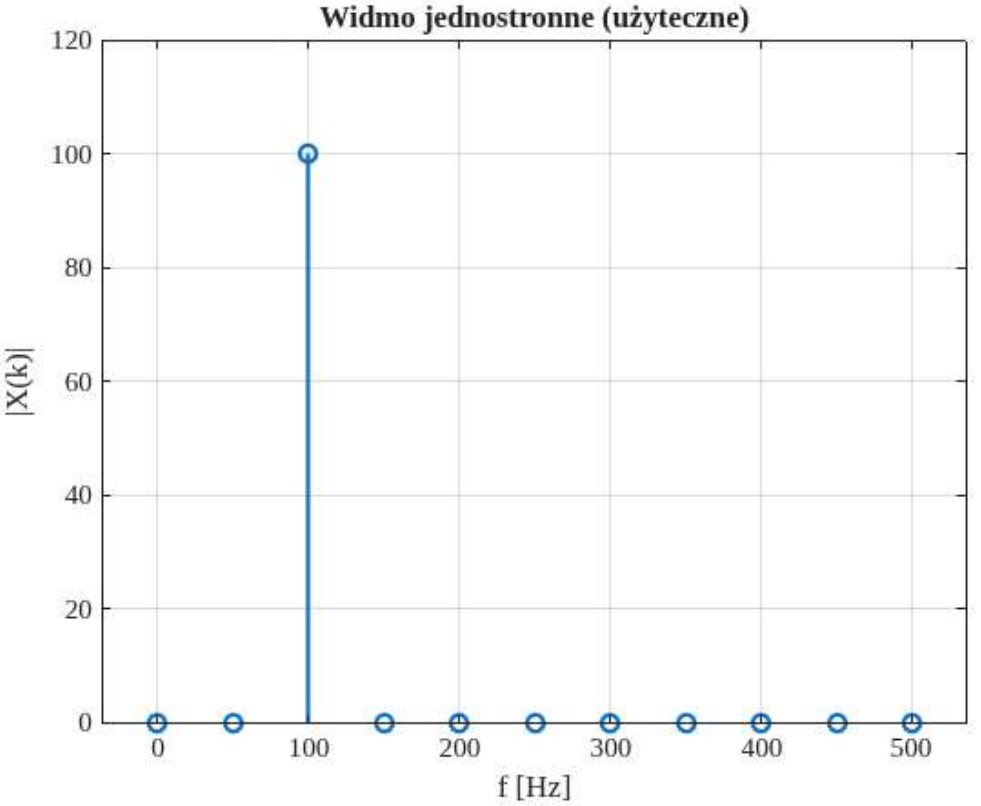



```
Xk = 1×20 complex
```

```
10^2 ×
  -0.0000 + 0.0000i  -0.0000 - 0.0000i   0.0000 - 1.0000i   0.0000 + 0.0000i ···
```

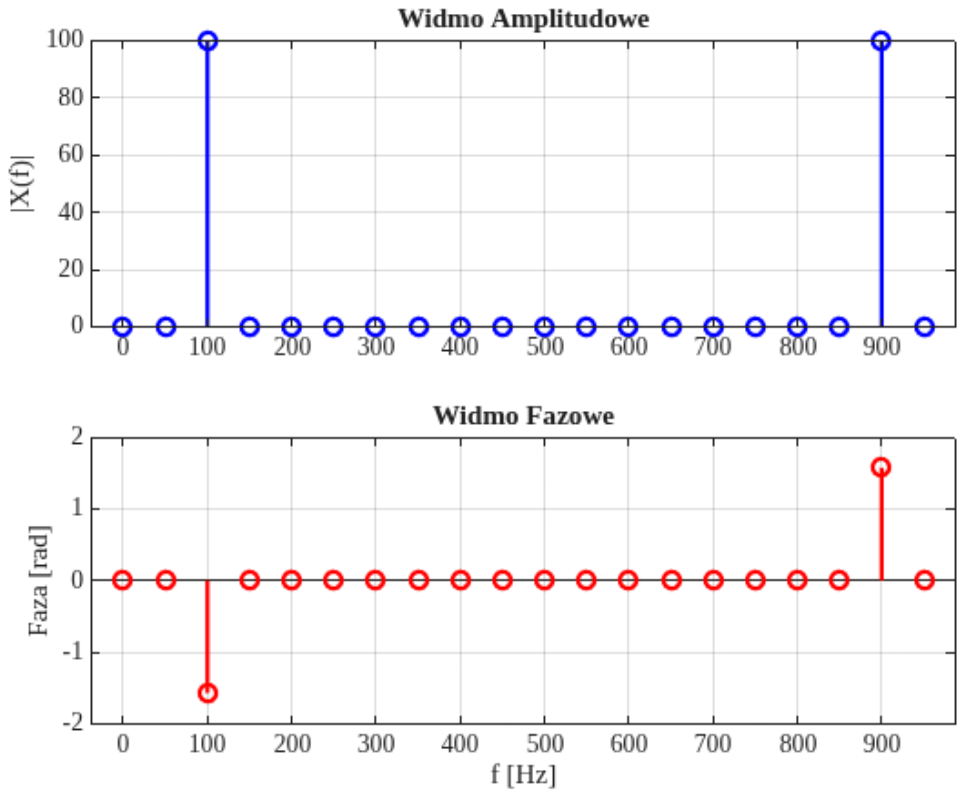







```
Amplituda = 1×20
         0         0  100.0000         0         0         0         0         0 ···
Faza = 1×20
```

```
    0         0   -1.5708         0         0         0         0         0 ···
```




```
Warning: An error occurred while drawing the scene: SceneModel error in command compositeCommand:
TypeError: Cannot read properties of undefined (reading 'deferred')
    at https://matlab-1b.mathworks.com/toolbox/matlab/uitools/figurelibjs/release/
bundle.mwBundle.gbtfigure-lib.js?mre=https
```

Wniosek:

(do zadania 5)

Widać, że gdy liczba próbek N zmienia się z 10 na 15, to rozdzielczość widma liczona ze wzoru f_p/N (1000/10) zmienia się ze 100Hz na 1000/15, co powoduje, że sygnał 100 Hz nie trafia już idealnie w prążek widma, lecz jego enrergia "rozlewa się" na sąsiednie prążki. Takie zjawisko jest nazywane przeciekiem widma.

To samo zjawisko widać w kolejnym zadaniu, gdzie zmieniona zostaje częstotliwość, zamiast liczby próbek.

Zadanie 6)

```
f3 = 150;
N = 10;

x3 = A1*sin(2*pi*f3*t);
x = x3;
tn = [0:N-1]*Tp; % wsp. czasowe próbek
xn = double(subs(x,t,tn));
Xk = zeros(1,N);

for k = 0:N-1 % impl. wzoru (8)
```

```
for n = 0:N-1
Xk(k+1) = Xk(k+1) + xn(n+1) * exp(-1j * 2 * pi * k * n / N);
end
end

Xk_fft = fft(xn,N); %funkcja wbudowana
dft_err = sum(abs(Xk_fft-Xk))
```

```
dft_err =
1.9041e-13
```

```
disp('DFT error:'); disp(dft_err);
```

```
DFT error:
   1.9041e-13
```

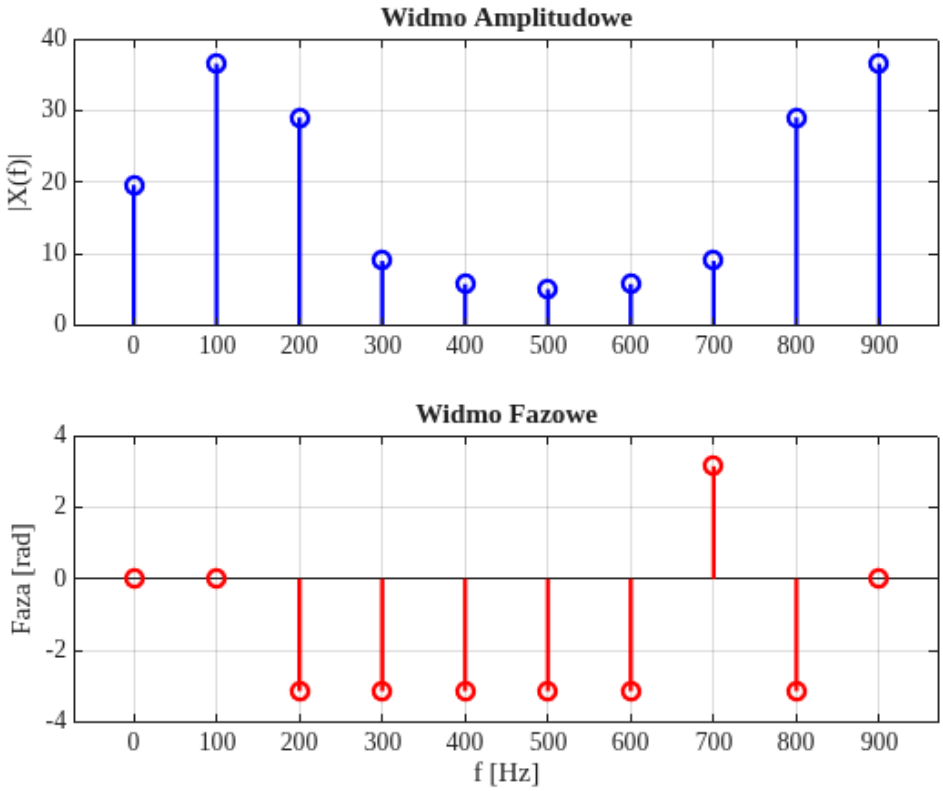
```


tol = 1e-5; % Tolerancja błędów numerycznych
Xk_clean = Xk;
Xk_clean(abs(Xk) < tol) = 0; % Zerowanie szumu

Amplituda = abs(Xk_clean)
```

```
Amplituda = 1×10
   19.6261   36.5688   29.0211    9.0211    5.7919    5.0953    5.7919    9.0211 ···
```

```
Faza = angle(Xk_clean) % Kąt w radianach
```

```
Faza = 1×10
         0    0.0000   -3.1416   -3.1416   -3.1416   -3.1416   -3.1416    3.1416 ···
```

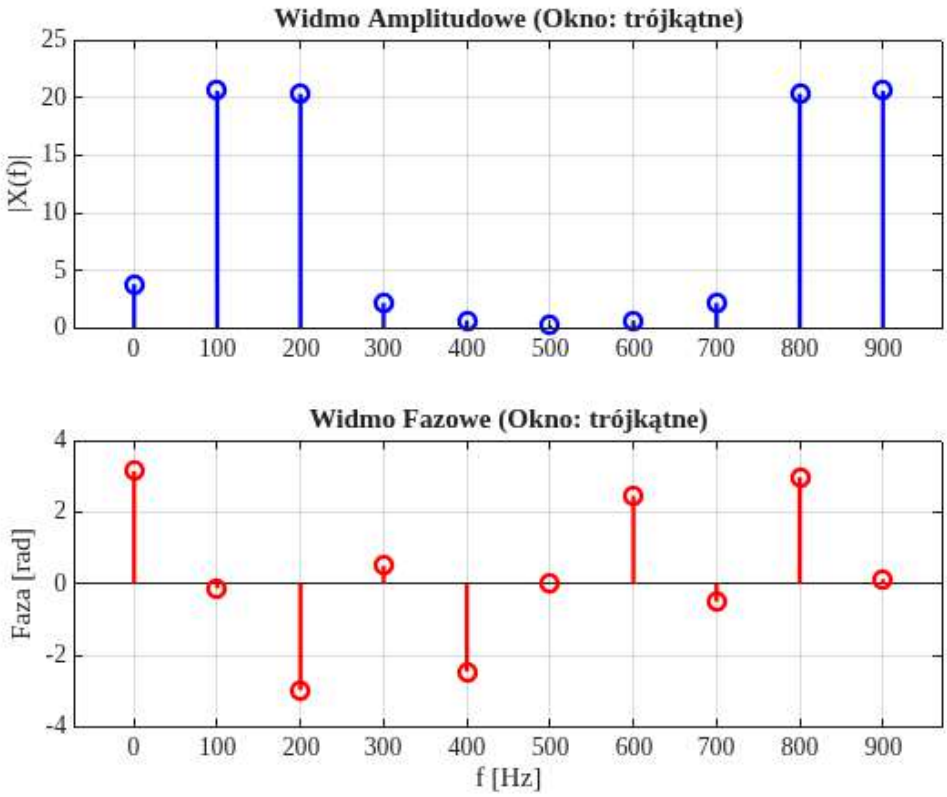
```


f_axis = (0:N-1) * (fp / N);

figure('Name', 'Zadanie 6 - F = 150 Hz, N = 10 | Amplituda i Faza');
subplot(2,1,1);
stem(f_axis, Amplituda, 'b', 'LineWidth', 1.5); grid on;
title('Widmo Amplitudowe'); ylabel('|X(f)|');

subplot(2,1,2);
stem(f_axis, Faza, 'r', 'LineWidth', 1.5); grid on;
title('Widmo Fazowe'); ylabel('Faza [rad]'); xlabel('f [Hz]');
```

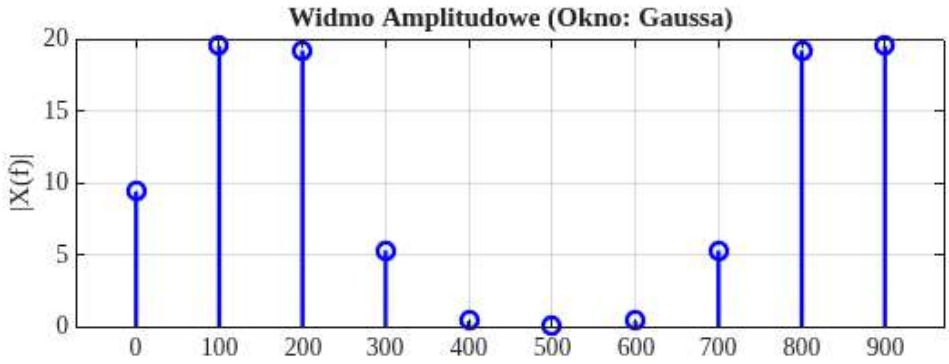


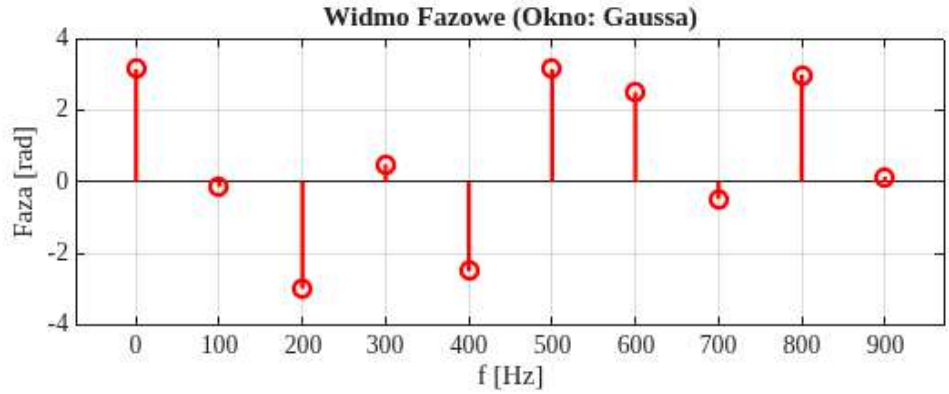
Zadanie 7)

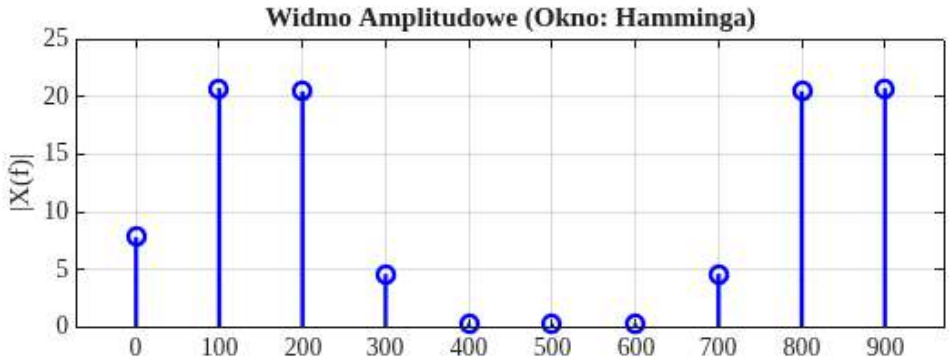
```
win_tri = triang(N)';          % Okno trójkątne (transpozycja na wiersz)
win_gauss = gausswin(N, 2.5)'; % Okno Gaussa
win_hamm = hamming(N)';        % Okno Hamminga (wybrane dowolne)
windows = {win_tri, win_gauss, win_hamm}; % macierz z oknami
windows_names = {'trójkątne', 'Gaussa', 'Hamminga'};

for i=1:length(windows)
    current_win = windows{i};
    xn_windowed = xn .* current_win;

    Xk = zeros(1,N); % reset Xk przed każdym oknem, by wynik nie był sumą z
poprzedniej iteracji pętli

    for k = 0:N-1 % impl. wzoru (8)
    for n = 0:N-1
    Xk(k+1) = Xk(k+1) + xn_windowed(n+1) * exp(-1j * 2 * pi * k * n / N);
    end
    end

    Xk_clean = Xk;
    Xk_clean(abs(Xk) < tol) = 0; % Zerowanie szumu

    Amplituda = abs(Xk_clean);
```

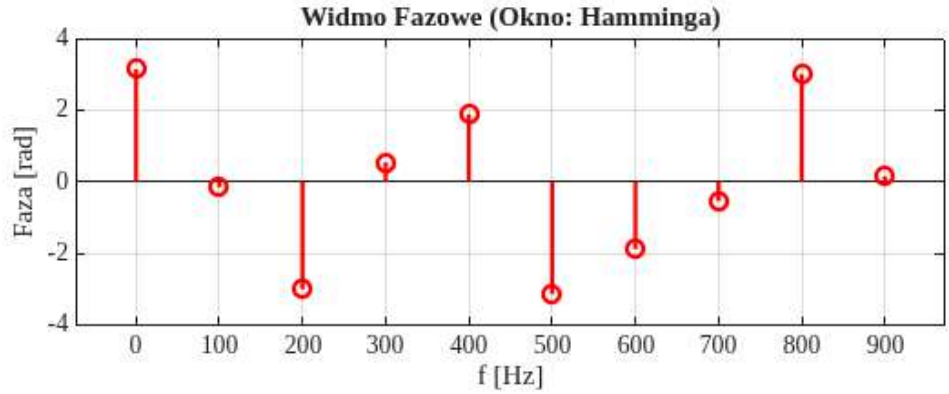
```
    Faza = angle(Xk_clean); % Kąt w radianach

    figure('Name', ['Zadanie 7: Okno ' windows_names{i}]);

    subplot(2,1,1);
    stem(f_axis, Amplituda, 'b', 'LineWidth', 1.5); grid on;
    title(['Widmo Amplitudowe (Okno: ' windows_names{i} ')']); ylabel('|
X(f)|');

    subplot(2,1,2);
    stem(f_axis, Faza, 'r', 'LineWidth', 1.5); grid on;
    title(['Widmo Fazowe (Okno: ' windows_names{i} ')']); ylabel('Faza
[rad]'); xlabel('f [Hz]');

    drawnow;
end
```




Widmo Amplitudowe (Okno: Gaussa)
|X(f)|
0
5
10
15
20
0
100
200
300
400
500
600
700
800
900


Widmo Fazowe (Okno: Gaussa)
Faza [rad]
4
2
0
-2
-4
0
100
200
300
400
500
600
700
800
900
f [Hz]


Widmo Amplitudowe (Okno: Hamminga)
|X(f)|
0
5
10
15
20
25
0
100
200
300
400
500
600
700
800
900

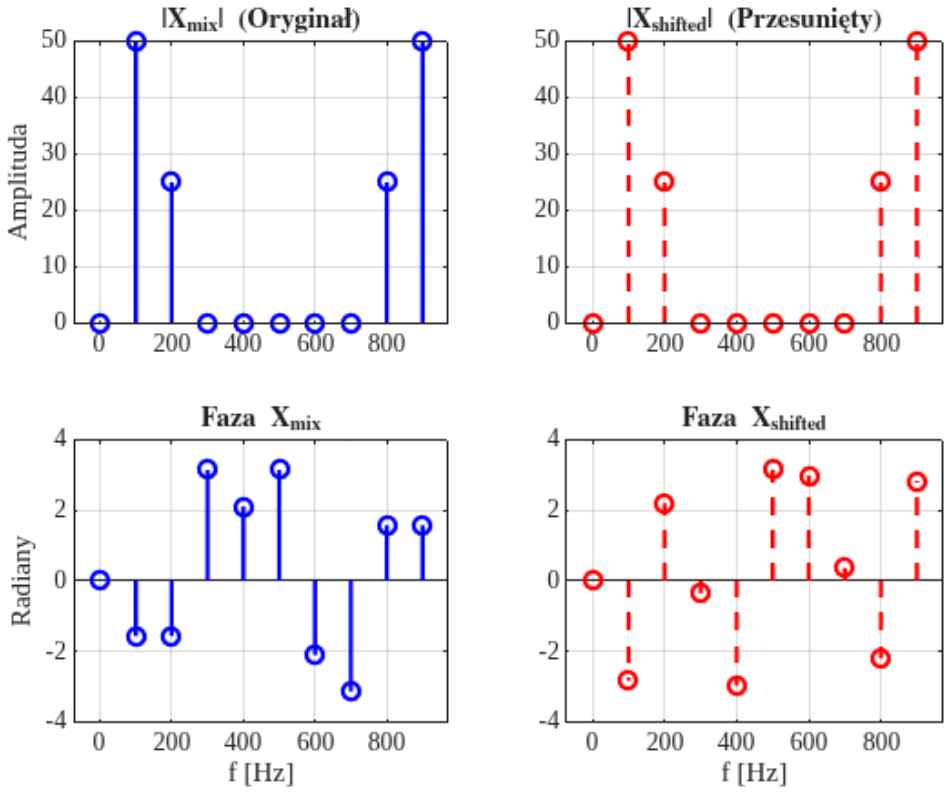

Widmo Fazowe (Okno: Hamminga)
Faza [rad]
4
2
0
-2
-4
0
100
200
300
400
500
600
700
800
900
f [Hz]

Wniosek:

Na wykresach każdego z okien widać, że przeciek widmowy udało się w pewnym stopniu wyeliminować (najbardziej dla okna trójkątnego), lecz niezupełnie, gdyż widma amplitudowe i fazowe nadal ukazują więcej niezerowych prążków niż powinno być.

Zadanie 8)

```
f2 = 200; % Hz
A2 = 5.0;

x_mix = A1*sin(2*pi*f1*tn) + A2*sin(2*pi*f2*tn);

shift_val = 2;
x_shifted = circshift(x_mix, [0, shift_val]); % przesunięcie wektora
poziomego


X_mix = fft(x_mix);
X_shifted = fft(x_shifted);
f_axis = (0:N-1)*(fp/N);

figure('Name', 'Zadanie 8 - Przesunięcie cykliczne');

subplot(2,2,1); stem(f_axis, abs(X_mix), 'b', 'LineWidth', 1.5); grid on;
title('|X_{mix}| (Oryginał)'); ylabel('Amplituda');

subplot(2,2,2); stem(f_axis, abs(X_shifted), 'r--', 'LineWidth', 1.5); grid
on;
title('|X_{shifted}| (Przesunięty)');

% Fazy
subplot(2,2,3); stem(f_axis, angle(X_mix), 'b', 'LineWidth', 1.5); grid on;
title('Faza X_{mix}'); ylabel('Radiany'); xlabel('f [Hz]');

subplot(2,2,4); stem(f_axis, angle(X_shifted), 'r--', 'LineWidth', 1.5);
grid on;
title('Faza X_{shifted}'); xlabel('f [Hz]');
```



Wniosek:

Wykres amplitudowy przesuniętego sygnały jest identyczny do wykresu amplitudowego sygnału oryginalnego, natomiast wykresy fazowe są różne.

Zadanie 9)

Zastąpiono iteracyjną implementację DFT operacją mnożenia macierzowego X = W_N*x. Macierz jądra transformacji W_N o rozmiarze N x N została wygenerowana wektorowo, gdzie element w_kn = e^(-j*2*pi/N*kn). Podejście to jest znacznie bardziej efektywne obliczeniowo w środowisku MATLAB, które jest zoptymalizowane do operacji macierzowych, eliminując narzut czasowy interpretacji pętli for.

```
col_vec = (0:N-1)'; % Wektor kolumnowy
row_vec = (0:N-1);  % Wektor wierszowy

W_matrix = exp(-1j * 2 * pi * (col_vec * row_vec) / N);

X_matrix_result = W_matrix * xn.' % xn musi być wektorem kolumnowym
```

```
X_matrix_result = 10×1 complex
  19.6261 + 0.0000i
  36.5688 + 0.0000i
 -29.0211 - 0.0000i
  -9.0211 - 0.0000i
  -5.7919 - 0.0000i
  -5.0953 - 0.0000i
  -5.7919 - 0.0000i
```

```
  -9.0211 + 0.0000i
 -29.0211 - 0.0000i
  36.5688 + 0.0000i
```

```
err_matrix = sum(abs(X_matrix_result.' - fft(xn))); % Transponujemy wynik z
powrotem
disp(['Błąd metody macierzowej: ', num2str(err_matrix)]);
```

```
Błąd metody macierzowej: 1.9041e-13
```