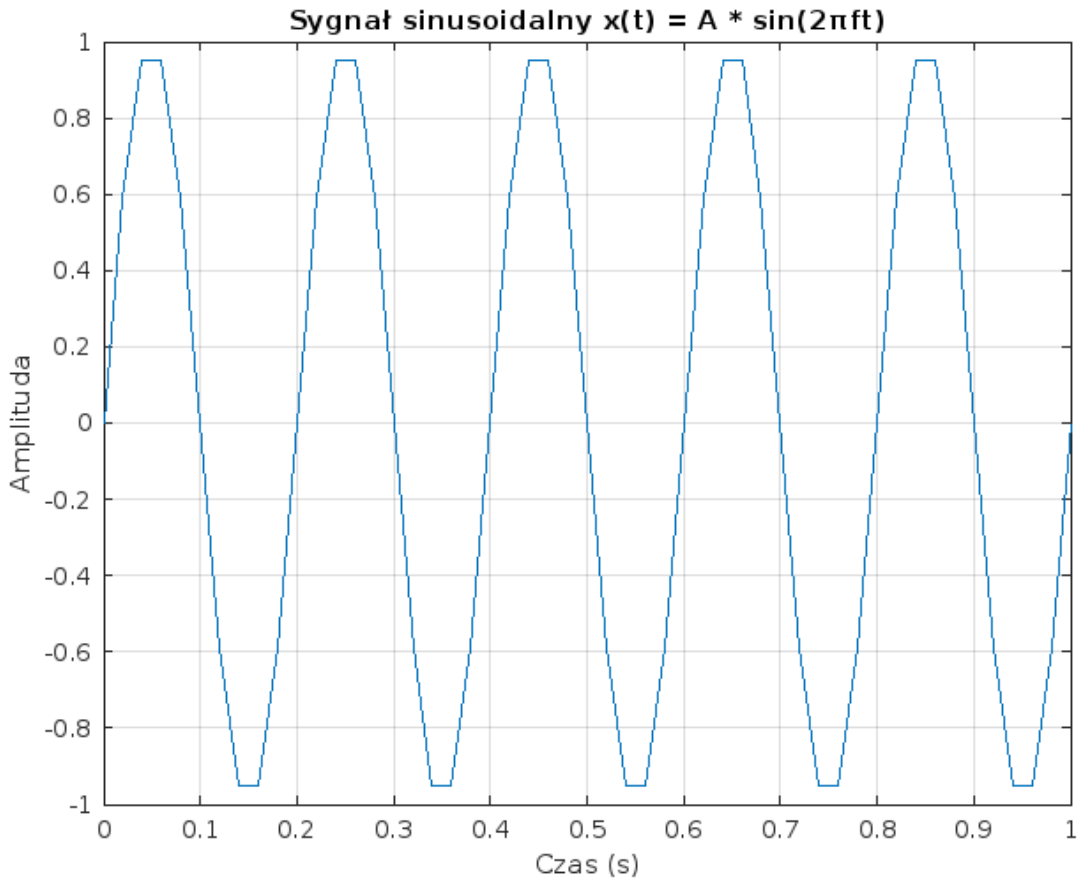
Table of Contents

Zadanie 1

```
A = 1;
f = 5;
fs = 50;
t = 0:1/fs:1;

x = A * sin(2 * pi * f * t);

figure;
plot(t, x);
title('Sygnał sinusoidalny x(t) = A * sin(2ft)');
xlabel('Czas (s)');
ylabel('Amplituda');
grid on;
```

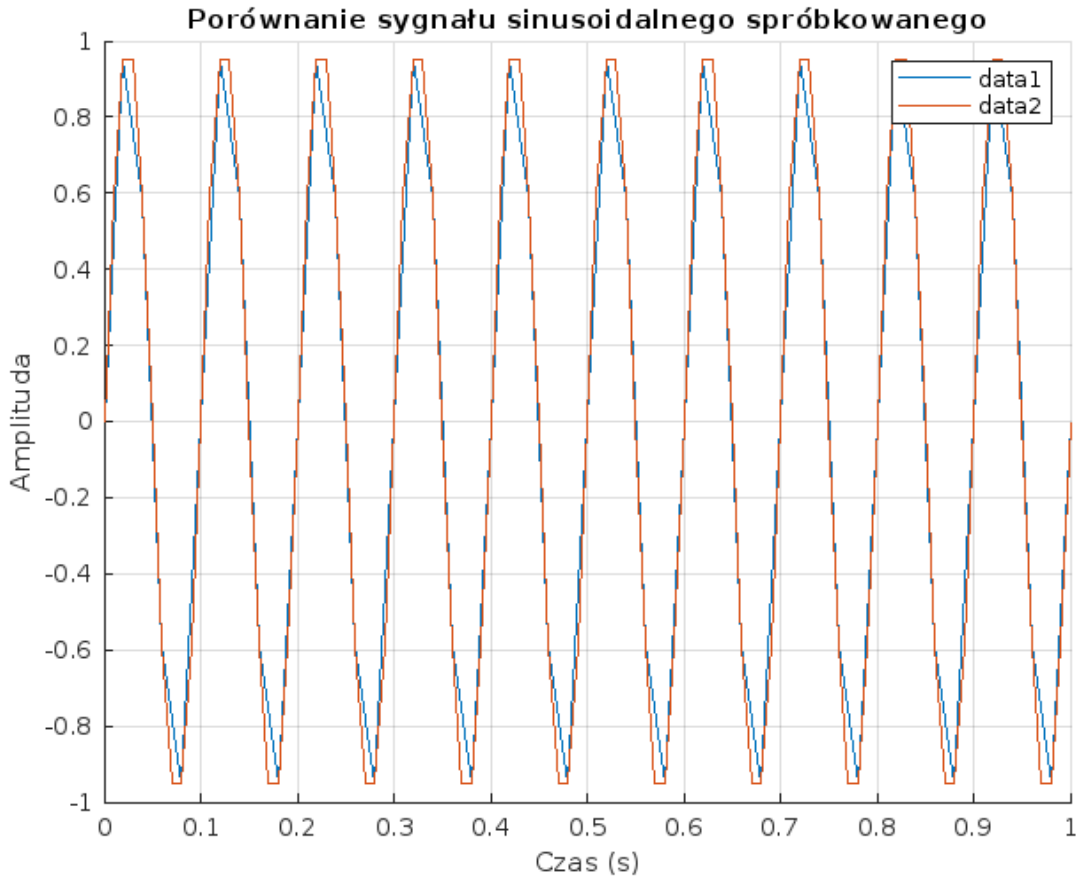


Zadanie 2

```
f = 10;

x = sin(2 * pi * f * t);

fs1 = 50;
fs2 = 100

t1 = 0:1/fs1:1;
t2 = 0:1/fs2:1;

x_sampled1 = sin(2 * pi * f * t1);
x_sampled2 = sin(2 * pi * f * t2);

% Wykres
figure;
hold on;
plot(t1, x_sampled1);
plot(t2, x_sampled2);
hold off;

% Ustawienia wykresu
```

```
title('Porównanie sygnału sinusoidalnego spróbkowanego');
xlabel('Czas (s)');
ylabel('Amplituda');
legend;
grid on;
```

```
fs2 =

   100
```



Zadanie 3

```
f = 30;
fs = 40;
t = 0:1/fs:1;
x = sin(2 * pi * f * t);

%figure;
%plot(t, x);
%title('Sygnał sinusoidalny');
%xlabel('Czas (s)');
```

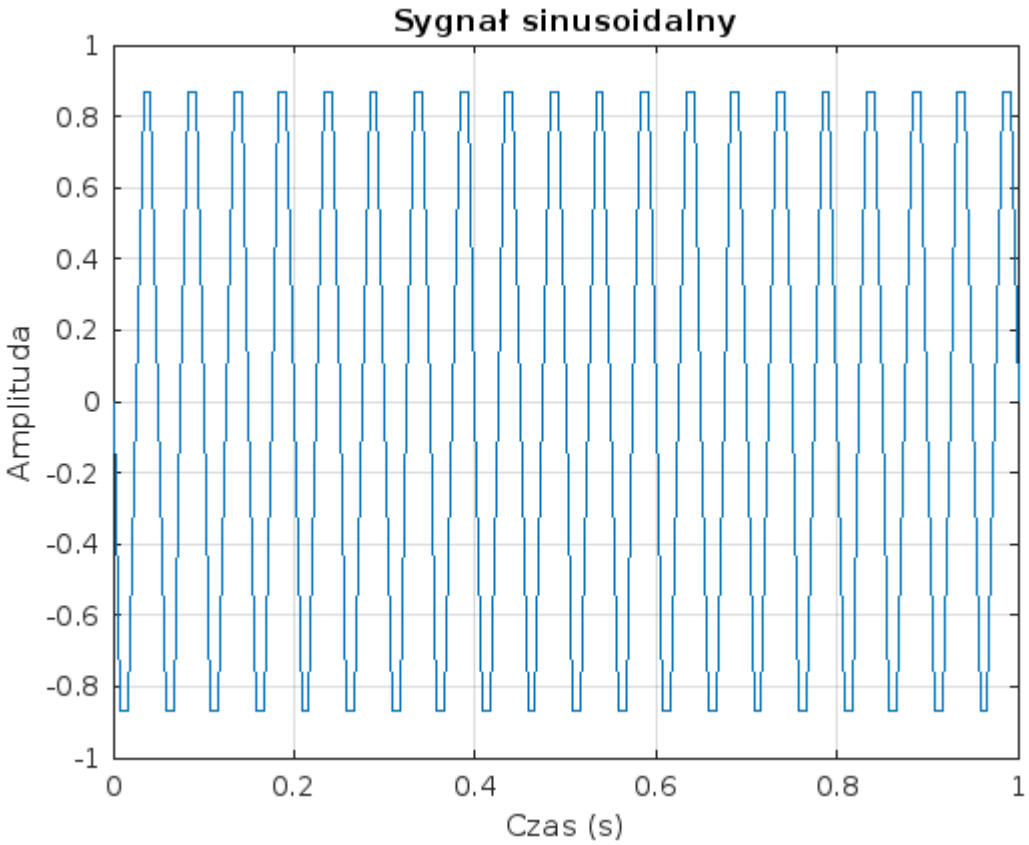
```
%ylabel('Amplituda');
%grid on;

if fs < 2 * f
    disp('Zjawisko aliasingu zachodzi: częstotliwość próbkowania jest
mniejsza niż dwukrotność częstotliwości sygnału.');
else
    disp('Nie zachodzi zjawisko aliasingu: częstotliwość próbkowania jest
wystarczająca.');
end
```

Zjawisko aliasingu zachodzi: częstotliwość próbkowania jest mniejsza niż dwukrotność częstotliwości sygnału.

Zadanie 4

```
f = 100;
fs = 120;
t = 0:1/fs:1;

x = sin(2 * pi * f * t);

figure;
plot(t, x);
title('Sygnał sinusoidalny');
xlabel('Czas (s)');
ylabel('Amplituda');
grid on;

if fs < 2 * f
    disp('Zjawisko aliasingu zachodzi: częstotliwość próbkowania jest
mniejsza niż dwukrotność częstotliwości sygnału.');
else
    disp('Nie zachodzi zjawisko aliasingu: częstotliwość próbkowania jest
wystarczająca.');
end
```

Zjawisko aliasingu zachodzi: częstotliwość próbkowania jest mniejsza niż dwukrotność częstotliwości sygnału.

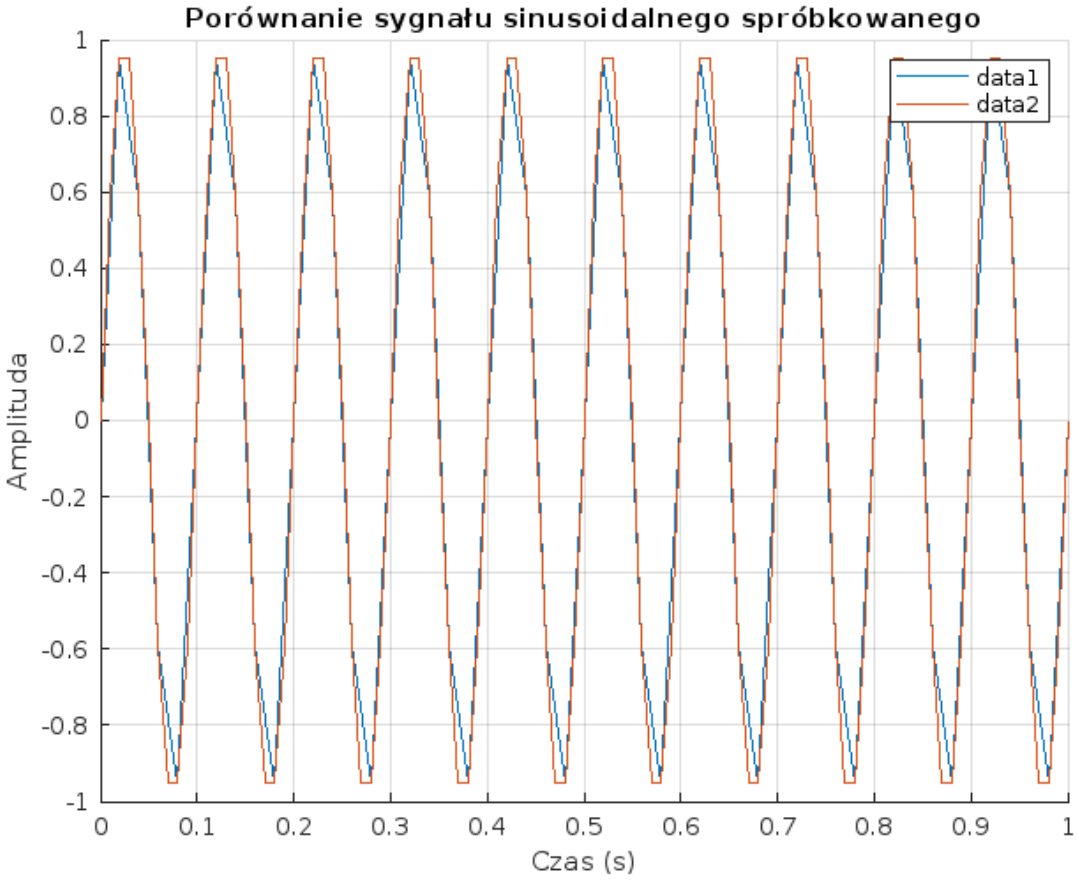


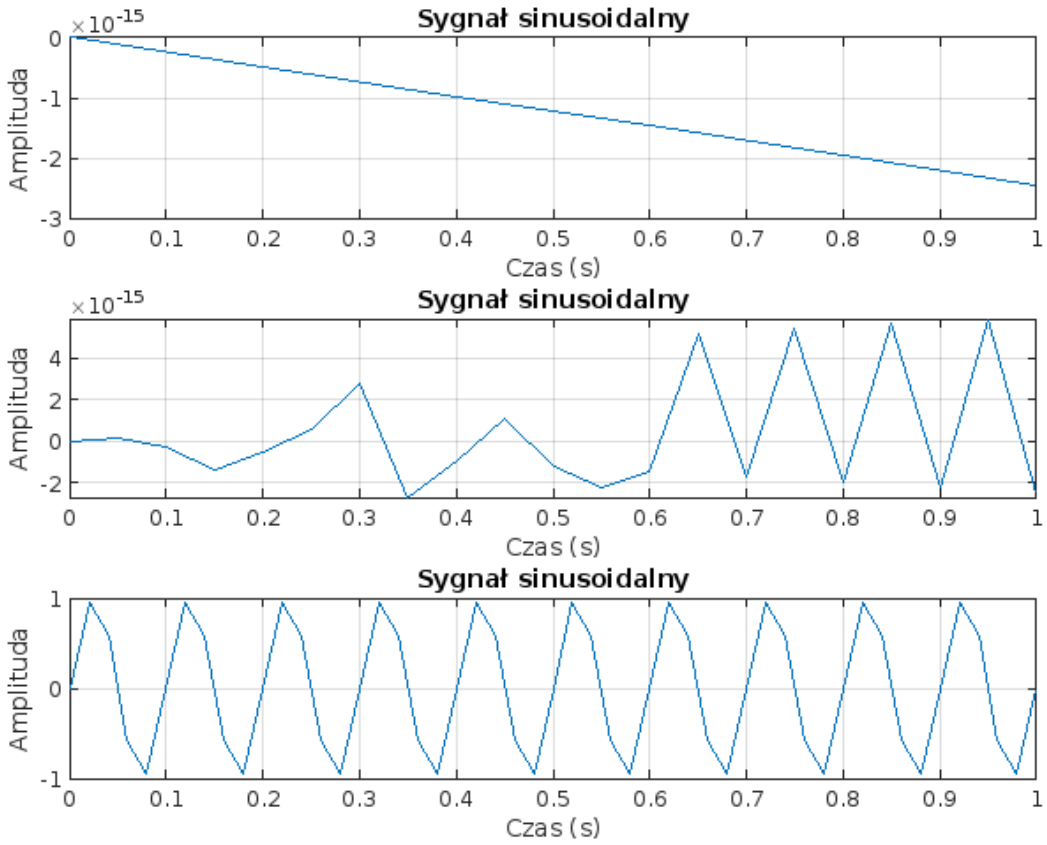
Zadanie 5

```
f = 10;
fs1 = 5;
fs2 = 20;
fs3 = 50;

t1 = 0:1/fs1:1;
t2 = 0:1/fs2:1;
t3 = 0:1/fs3:1;

x1 = sin(2 * pi * f * t1);
x2 = sin(2 * pi * f * t2);
x3 = sin(2 * pi * f * t3);

figure;
subplot(3,1,1)
plot(t1, x1);
title('Sygnał sinusoidalny');
xlabel('Czas (s)');
ylabel('Amplituda');
grid on;

if fs1 < 2 * f
```

```
    disp('Zjawisko aliasingu zachodzi: częstotliwość próbkowania jest
mniejsza niż dwukrotność częstotliwości sygnału.');
else
    disp('Nie zachodzi zjawisko aliasingu: częstotliwość próbkowania jest
wystarczająca.');
end

subplot(3,1,2)
plot(t2, x2);
title('Sygnał sinusoidalny');
xlabel('Czas (s)');
ylabel('Amplituda');
grid on;

if fs2 < 2 * f
    disp('Zjawisko aliasingu zachodzi: częstotliwość próbkowania jest
mniejsza niż dwukrotność częstotliwości sygnału.');
else
    disp('Nie zachodzi zjawisko aliasingu: częstotliwość próbkowania jest
wystarczająca.');
end

subplot(3,1,3)
plot(t3, x3);
title('Sygnał sinusoidalny');
xlabel('Czas (s)');
ylabel('Amplituda');
grid on;

if fs3 < 2 * f
    disp('Zjawisko aliasingu zachodzi: częstotliwość próbkowania jest
mniejsza niż dwukrotność częstotliwości sygnału.');
else
    disp('Nie zachodzi zjawisko aliasingu: częstotliwość próbkowania jest
wystarczająca.');
end
```

```
Zjawisko aliasingu zachodzi: częstotliwość próbkowania jest mniejsza niż
dwukrotność częstotliwości sygnału.
Nie zachodzi zjawisko aliasingu: częstotliwość próbkowania jest wystarczająca.
Nie zachodzi zjawisko aliasingu: częstotliwość próbkowania jest wystarczająca.
```

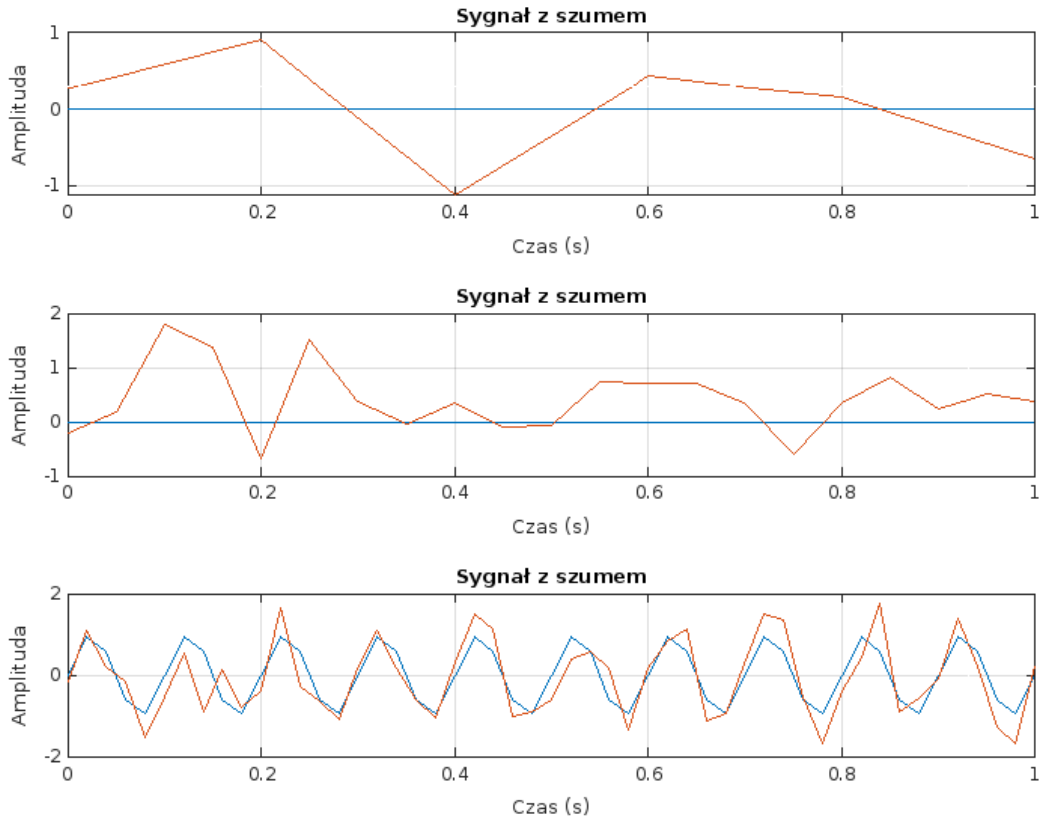

Porównanie sygnału sinusoidalnego spróbkowanego
data1
data2
Amplituda
Czas (s)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1



Zadanie 6

```
f = 10;

fs = [5,20,50];

amplitude_noise = 0.5;

figure
for i = 1:length(fs)
    subplot(length(fs),1,i)
    t = 0:1/fs(i):1;
    x_t = sin(2 * pi * f * t);
    plot(t, x_t);
    noise = amplitude_noise*randn(size(t));

    noisy_x = x_t + noise;
    hold on;
    plot(t, noisy_x);
    hold off;
    title('Sygnał z szumem');
    xlabel('Czas (s)');
    ylabel('Amplituda');
```

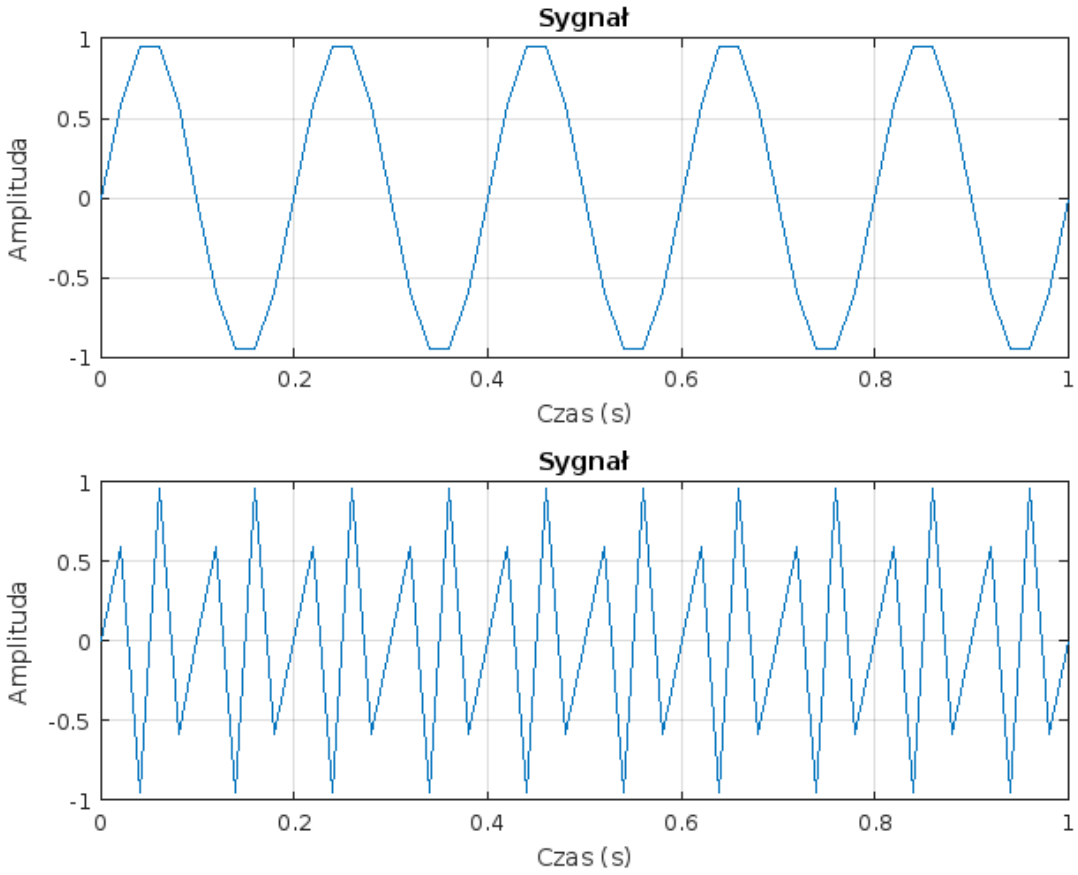
```
    grid on;

end
```



Zadanie 7

```
f = [5,20];

fs = 50;

t = 0:1/fs:1;

figure
for i = 1:length(f)
    subplot(length(f),1,i)
    x_t = sin(2 * pi * f(i) * t);
    plot(t, x_t);

    title('Sygnał');
    xlabel('Czas (s)');
    ylabel('Amplituda');
    grid on;
```

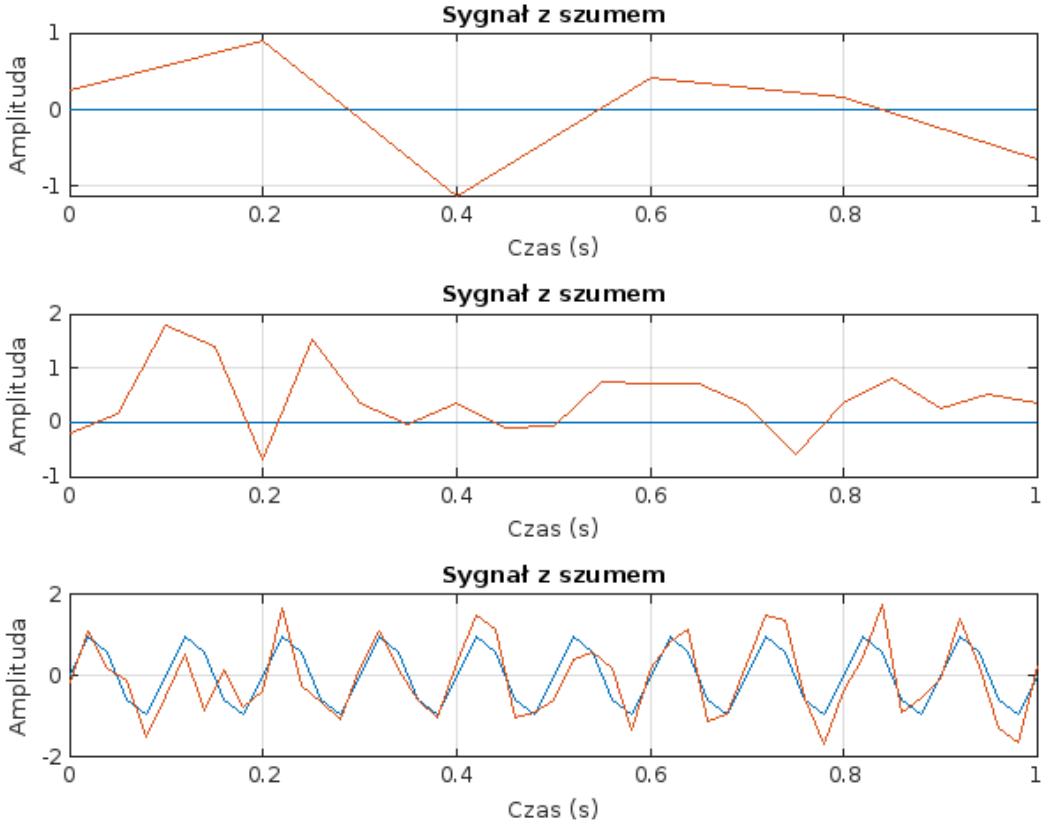
```
end
```

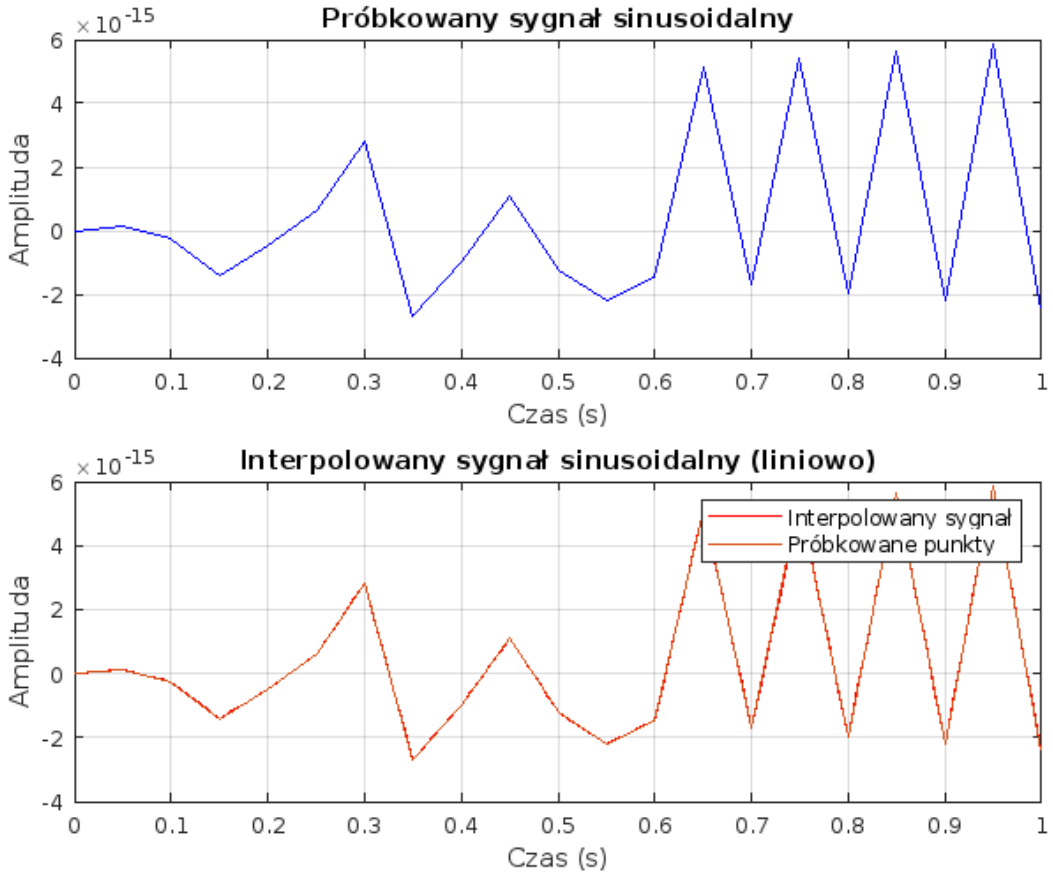


Zadanie 8

```
f = 10;
fs = 20;

t_sampled = 0:1/fs:1;
x_sampled = sin(2 * pi * f * t_sampled);

fs_interp = 100;
t_interp = 0:1/fs_interp:1;

x_interp = interp1(t_sampled, x_sampled, t_interp, 'linear');

figure;
subplot(2,1,1);
plot(t_sampled, x_sampled, 'b', 'MarkerFaceColor', 'b');
title('Próbkowany sygnał sinusoidalny');
xlabel('Czas (s)');
ylabel('Amplituda');
grid on;
```

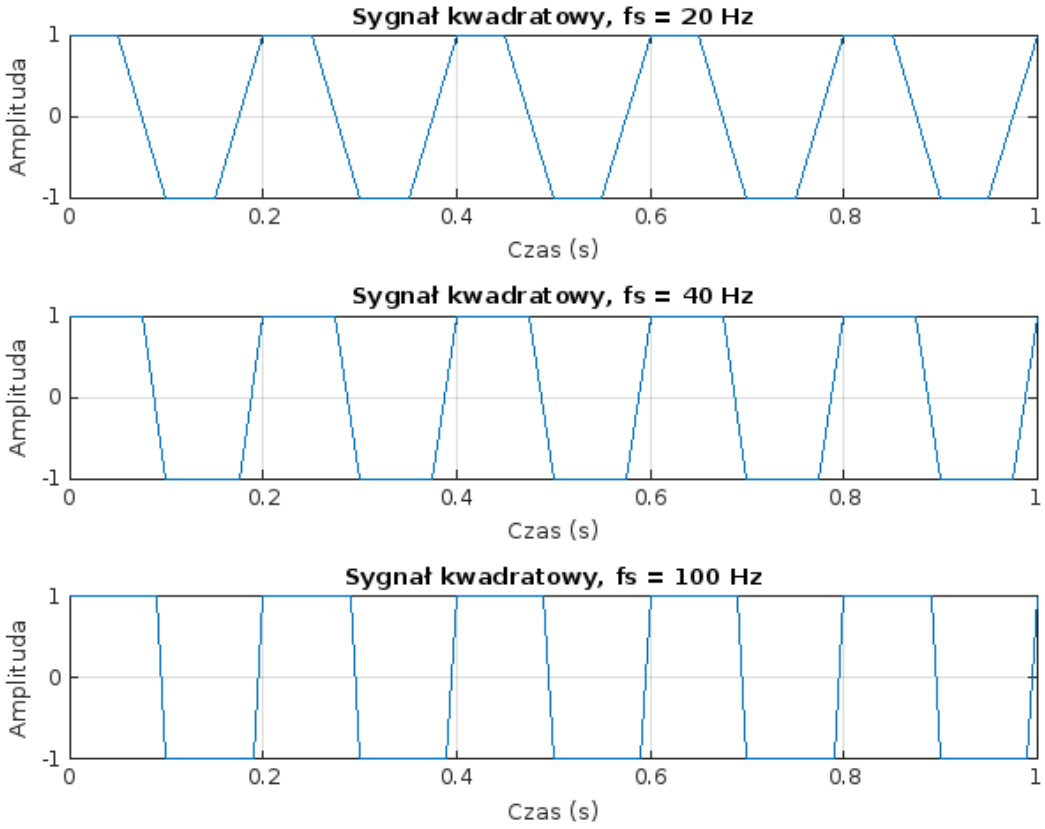
```
subplot(2,1,2);
plot(t_interp, x_interp, 'r');
hold on;
plot(t_sampled, x_sampled);
hold off;
title('Interpolowany sygnał sinusoidalny (liniowo)');
xlabel('Czas (s)');
ylabel('Amplituda');
grid on;
legend('Interpolowany sygnał', 'Próbkowane punkty');
```





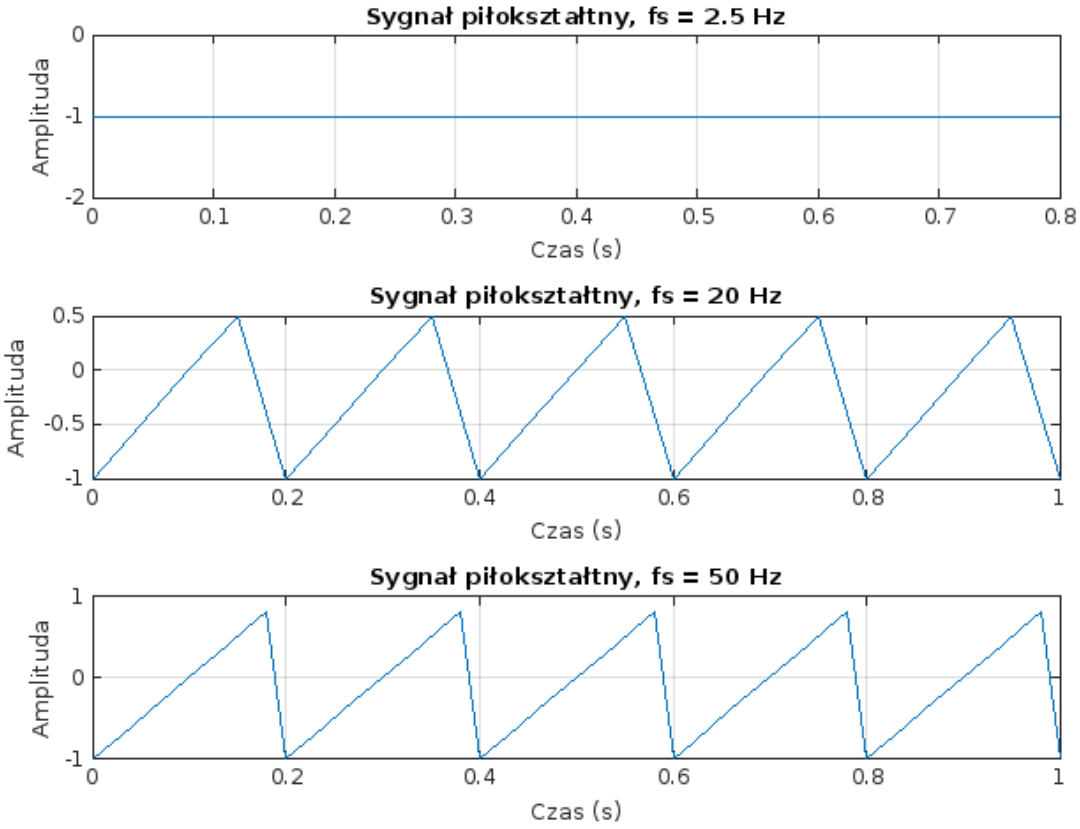
Zadanie 9

```
f = 5;
fs = [20, 40, 100];
amplitude_noise = 0.5;

figure;
for i = 1:length(fs)
    subplot(length(fs),1,i)
    t = 0:1/fs(i):1;
    x_t = square(2 * pi * f * t);
    plot(t, x_t);

    title(['Sygnał kwadratowy, fs = ' num2str(fs(i)) ' Hz']);
    xlabel('Czas (s)');
    ylabel('Amplituda');
    grid on;
end
```



Zadanie 10

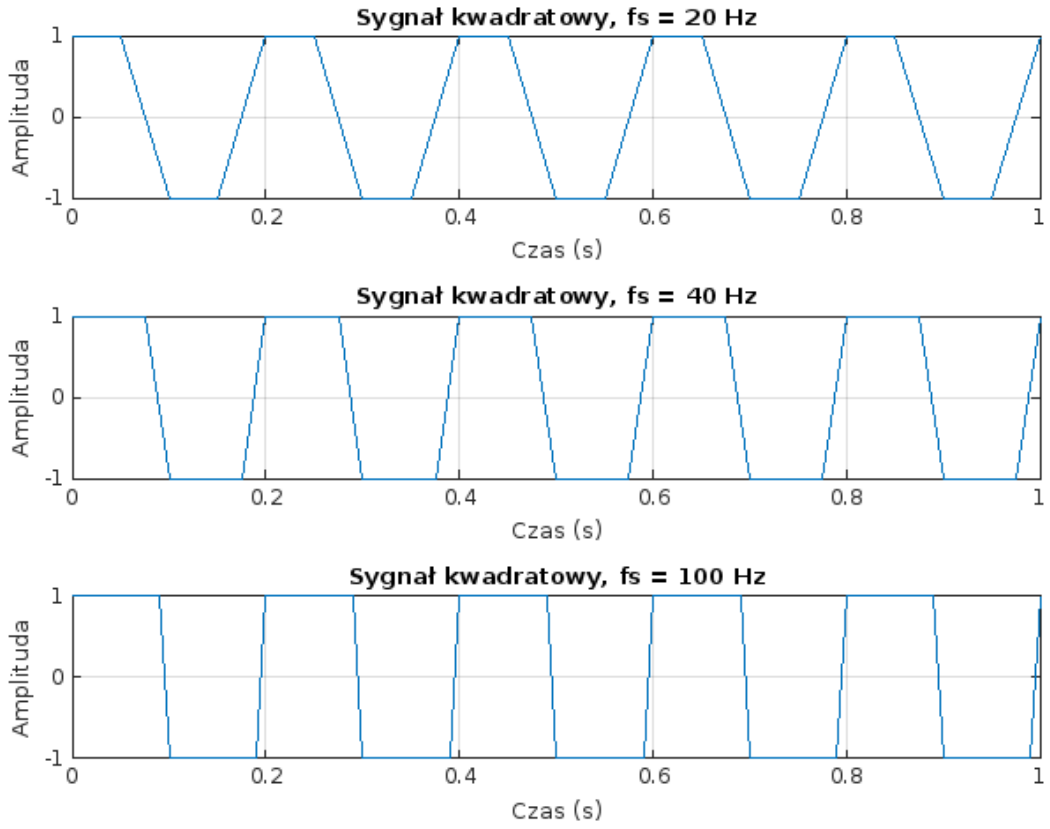
```
f = 5;
fs = [2.5, 20, 50];
amplitude_noise = 0.5;

figure;
for i = 1:length(fs)
    subplot(length(fs),1,i)
    t = 0:1/fs(i):1;
    x_t = sawtooth(2 * pi * f * t);
    plot(t, x_t);

    title(['Sygnał piłokształtny, fs = ' num2str(fs(i)) ' Hz']);
    xlabel('Czas (s)');
    ylabel('Amplituda');
    grid on;
end
```

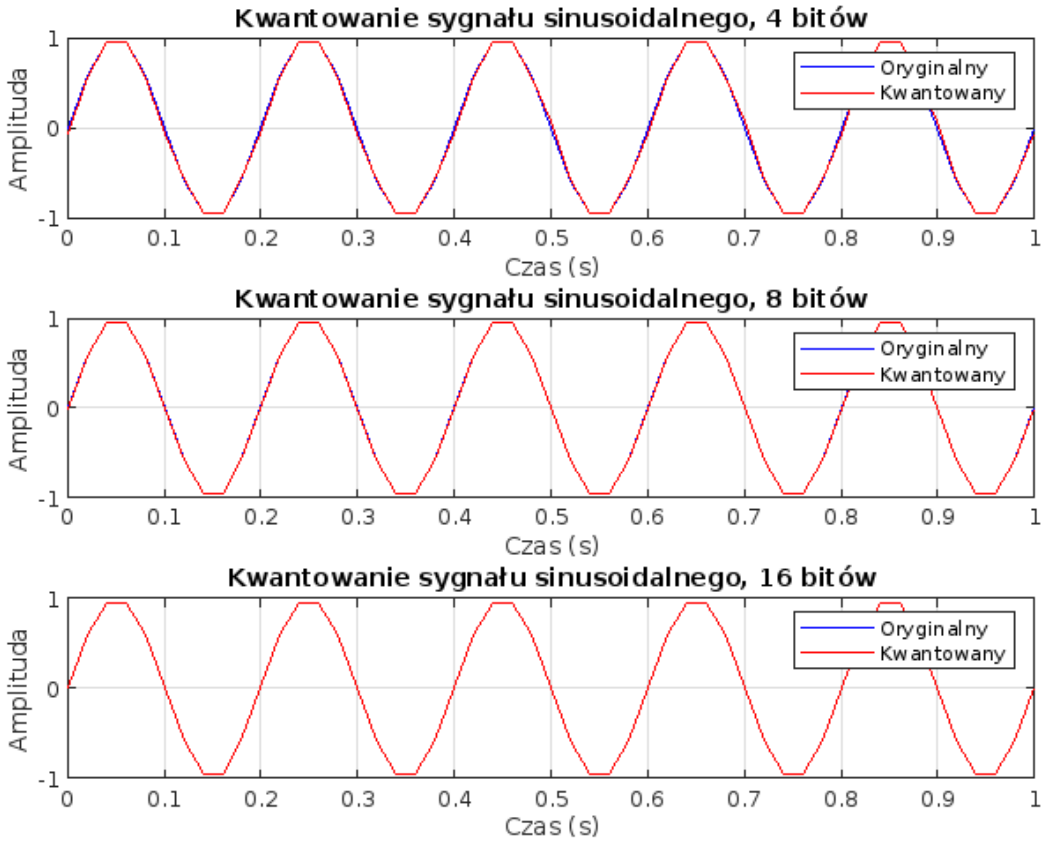


Zadanie 11

```
f = 5;
fs = 50;
duration = 1;
t = 0:1/fs:duration;

x_t = sin(2 * pi * f * t);


n_bits = [4, 8, 16];


figure;
for i = 1:length(n_bits)

    L = 2^n_bits(i);

    x_max = max(x_t);
    x_min = min(x_t);
    quant_step = (x_max - x_min) / (L - 1);
    x_quant = round((x_t - x_min) / quant_step) * quant_step + x_min;
```

```
    subplot(length(n_bits), 1, i);
    plot(t, x_t, 'b');
    hold on;
    plot(t, x_quant, 'r');
    hold off;
    title(['Kwantowanie sygnału sinusoidalnego, ', num2str(n_bits(i)), '
bitów']);
    xlabel('Czas (s)');
    ylabel('Amplituda');
    legend('Oryginalny', 'Kwantowany');
    grid on;
end
```

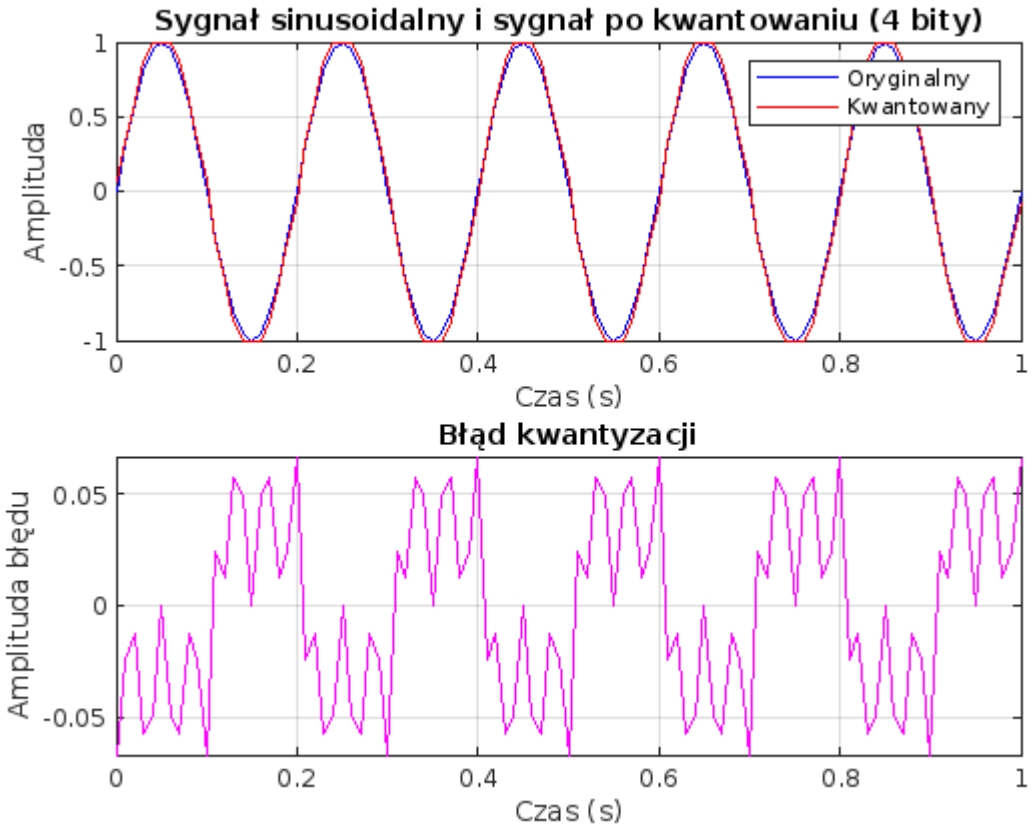




Zadanie 12

```
f = 5;
fs = 100;
t = 0:1/fs:1;

x_t = sin(2 * pi * f * t);


n_bits = 4;
L = 2^n_bits;

% Obliczanie kwantyzacji
x_max = max(x_t);
x_min = min(x_t);
quant_step = (x_max - x_min) / (L - 1);

x_quant = round((x_t - x_min) / quant_step) * quant_step + x_min;

quantization_error = x_t - x_quant;

figure;
```

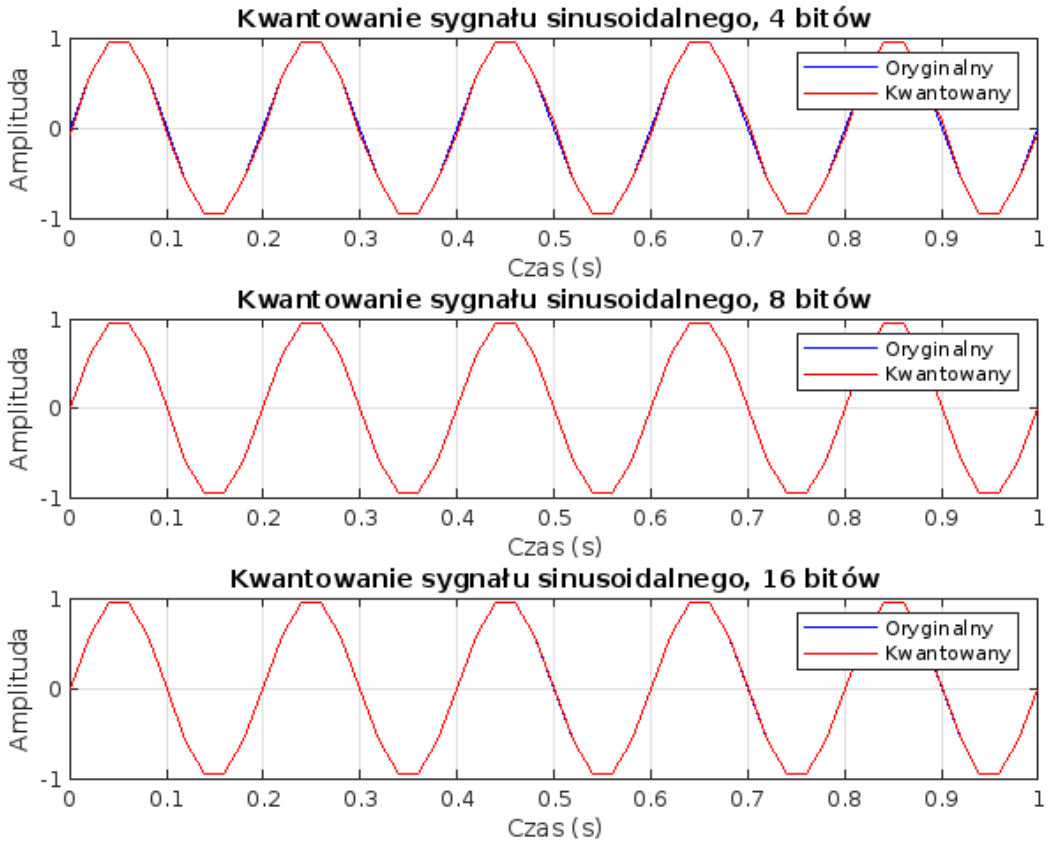
```
subplot(2,1,1);
plot(t, x_t, 'b');
hold on;
plot(t, x_quant, 'r');
hold off;
title('Sygnał sinusoidalny i sygnał po kwantowaniu (4 bity)');
xlabel('Czas (s)');
ylabel('Amplituda');
legend('Oryginalny', 'Kwantowany');
grid on;

% Błąd kwantyzacji
subplot(2,1,2);
plot(t, quantization_error, 'm');
title('Błąd kwantyzacji');
xlabel('Czas (s)');
ylabel('Amplituda błędu');
grid on;
```

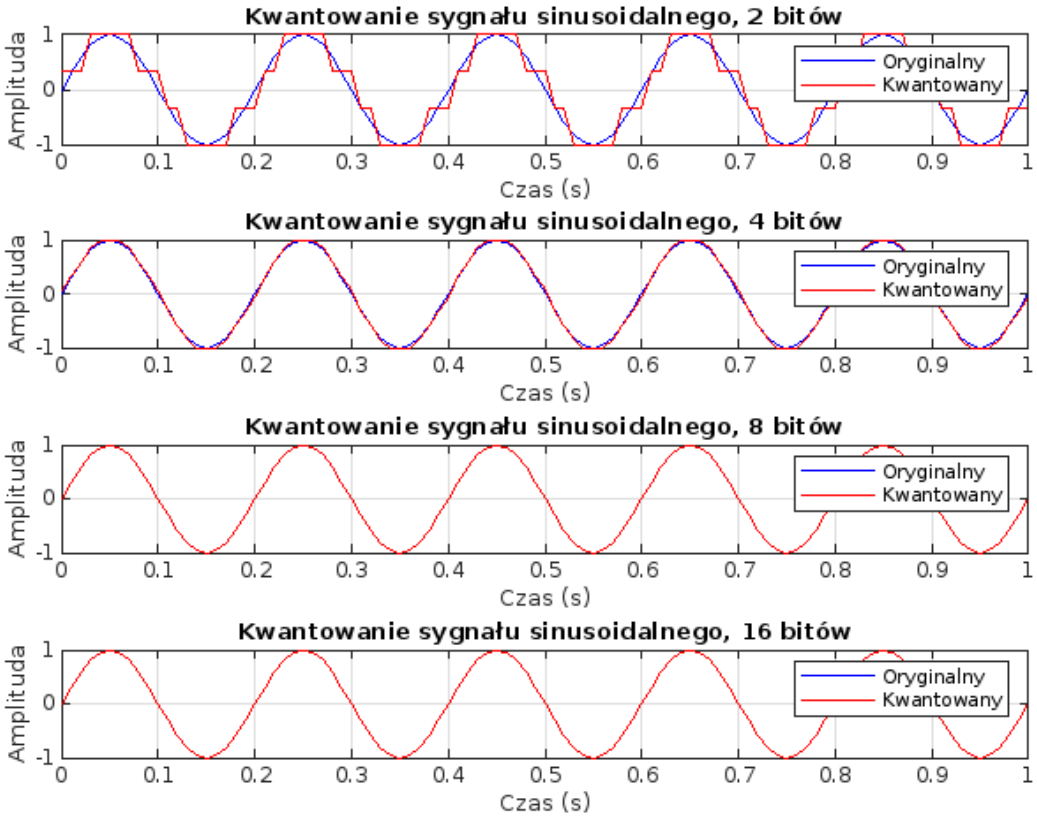


Zadanie 13

```
f = 5;
fs = 100;
t = 0:1/fs:1;
```

```
x_t = sin(2 * pi * f * t);

n_bits = [2, 4, 8, 16];

figure;
for i = 1:length(n_bits)
    L = 2^n_bits(i);

    x_max = max(x_t);
    x_min = min(x_t);
    quant_step = (x_max - x_min) / (L - 1);
    x_quant = round((x_t - x_min) / quant_step) * quant_step + x_min;

    subplot(length(n_bits), 1, i);
    plot(t, x_t, 'b');
    hold on;
    plot(t, x_quant, 'r');
    hold off;
    title(['Kwantowanie sygnału sinusoidalnego, ', num2str(n_bits(i)), '
bitów']);
    xlabel('Czas (s)');
    ylabel('Amplituda');
    legend('Oryginalny', 'Kwantowany');
    grid on;
end
```

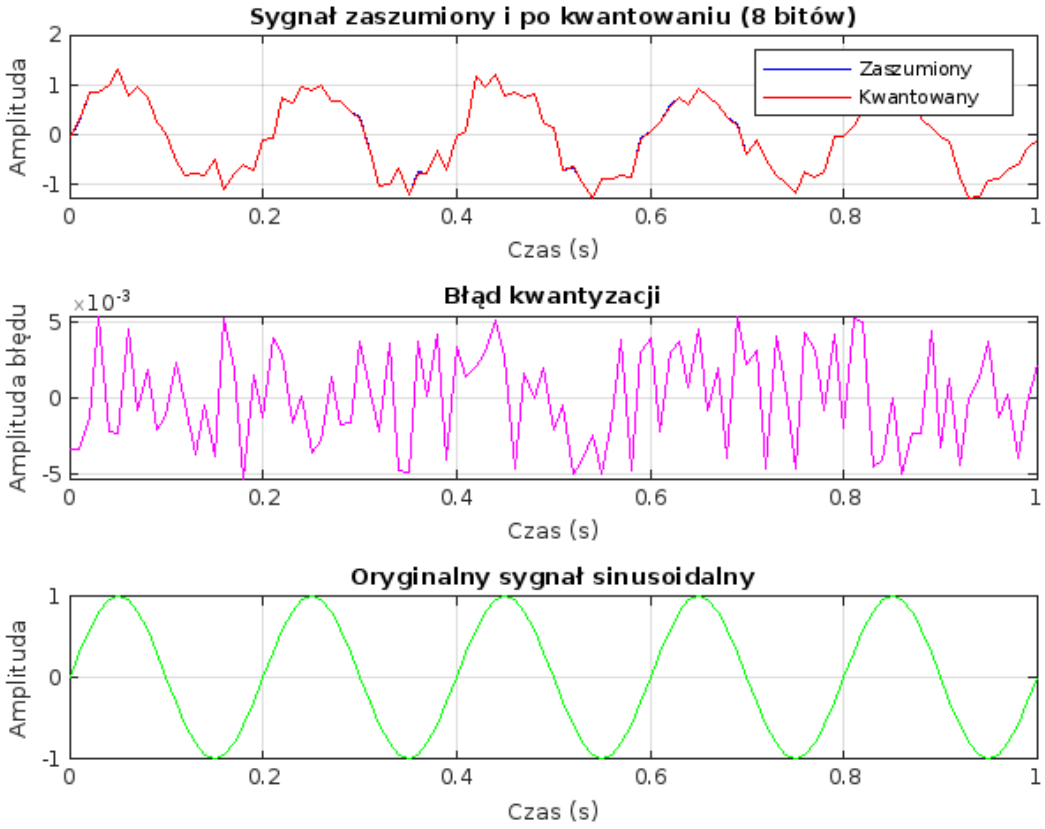

Kwantowanie sygnału sinusoidalnego, 4 bitów
Amplituda
Czas (s)
Oryginalny
Kwantowany
Kwantowanie sygnału sinusoidalnego, 8 bitów
Amplituda
Czas (s)
Oryginalny
Kwantowany
Kwantowanie sygnału sinusoidalnego, 16 bitów
Amplituda
Czas (s)
Oryginalny
Kwantowany



Zadanie 14

```
f = 5;
fs = 100;
t = 0:1/fs:1;

x_t = sin(2 * pi * f * t);

noise_amplitude = 0.2;
noise = noise_amplitude * randn(size(t));
x_noisy = x_t + noise;

n_bits = 8;
L = 2^n_bits;

x_max = max(x_noisy);
x_min = min(x_noisy);
quant_step = (x_max - x_min) / (L - 1);

x_quant = round((x_noisy - x_min) / quant_step) * quant_step + x_min;

quantization_error = x_noisy - x_quant;
```

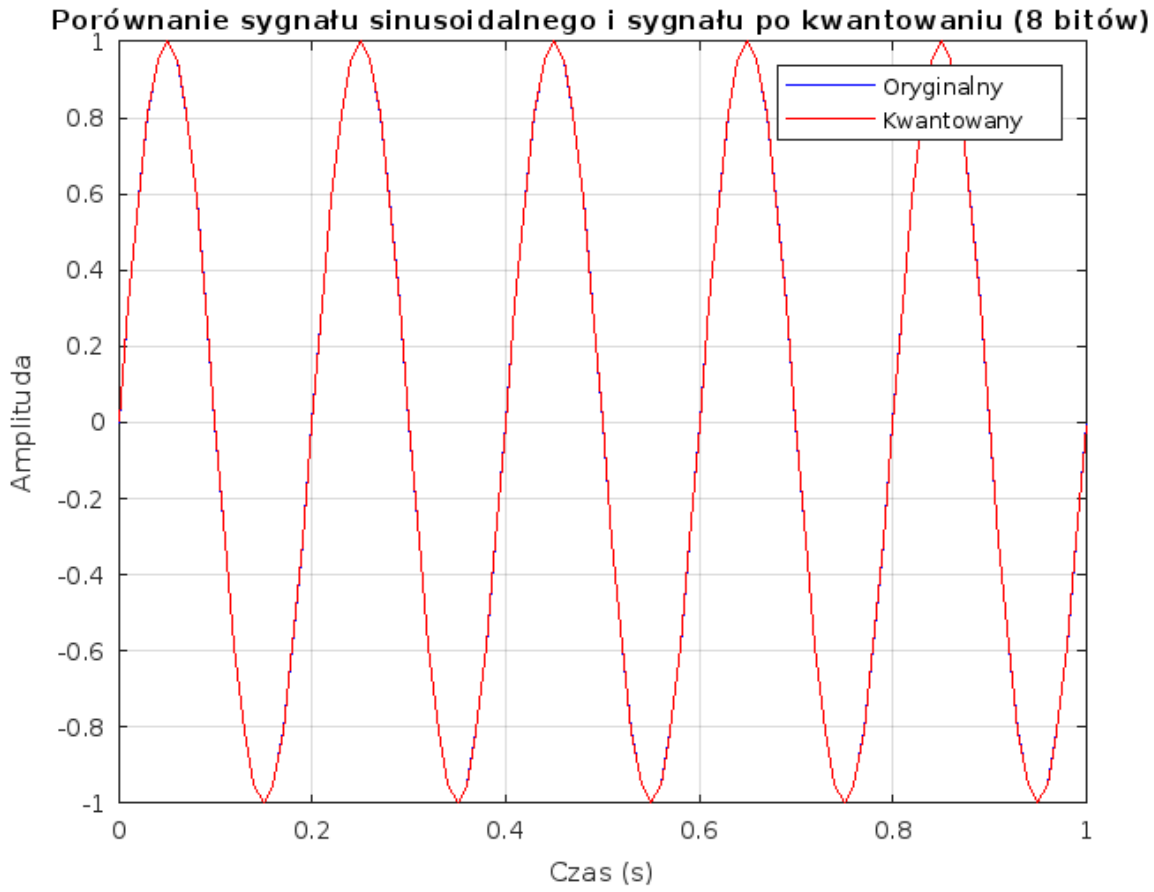
```
figure;
subplot(3,1,1);
plot(t, x_noisy, 'b');
hold on;
plot(t, x_quant, 'r');
hold off;
title('Sygnał zaszumiony i po kwantowaniu (8 bitów)');
xlabel('Czas (s)');
ylabel('Amplituda');
legend('Zaszumiony', 'Kwantowany');
grid on;

subplot(3,1,2);
plot(t, quantization_error, 'm');
title('Błąd kwantyzacji');
xlabel('Czas (s)');
ylabel('Amplituda błędu');
grid on;

subplot(3,1,3);
plot(t, x_t, 'g');
title('Oryginalny sygnał sinusoidalny');
xlabel('Czas (s)');
ylabel('Amplituda');
grid on;
```



Zadanie 15

```
f = 5;
fs = 100;
t = 0:1/fs:1;

x_t = sin(2 * pi * f * t);

n_bits = 8;
L = 2^n_bits;

x_max = max(x_t);
x_min = min(x_t);
quant_step = (x_max - x_min) / (L - 1);

x_quant = round((x_t - x_min) / quant_step) * quant_step + x_min;

figure;

plot(t, x_t, 'b');
hold on;
plot(t, x_quant, 'r');
hold off;
```

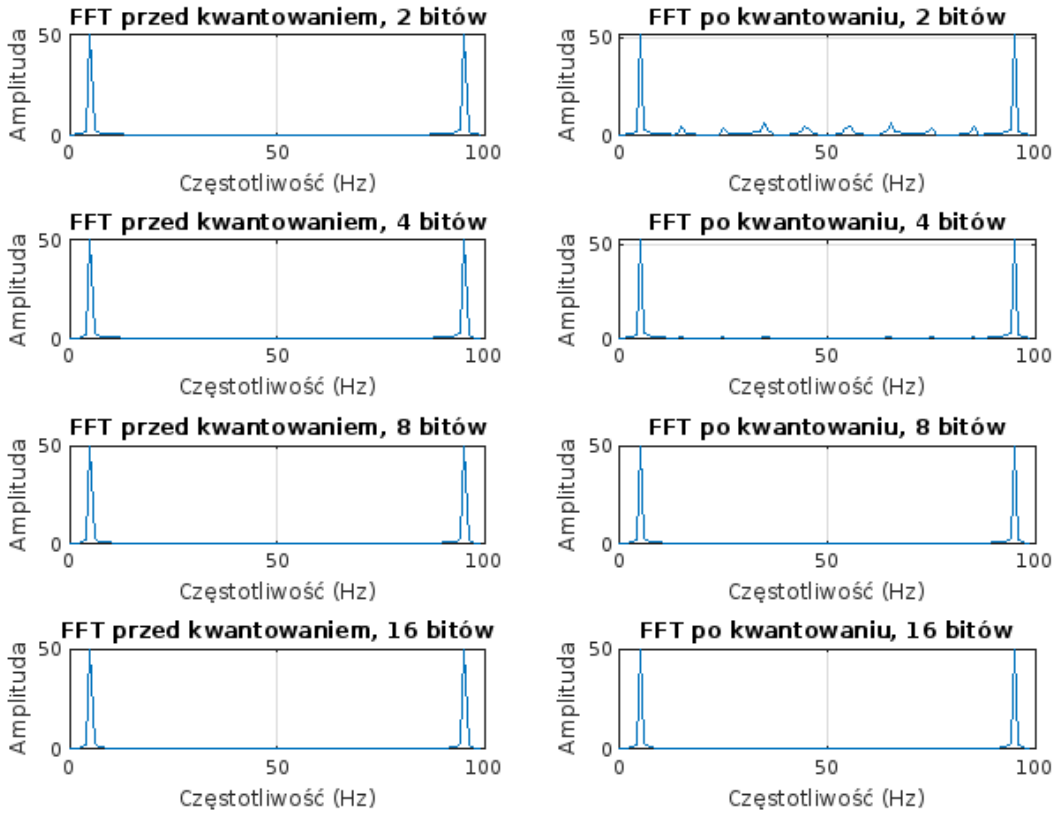
```
title('Porównanie sygnału sinusoidalnego i sygnału po kwantowaniu (8 bitów)');
xlabel('Czas (s)');
ylabel('Amplituda');
legend('Oryginalny', 'Kwantowany');
grid on;
```



Zadanie 16

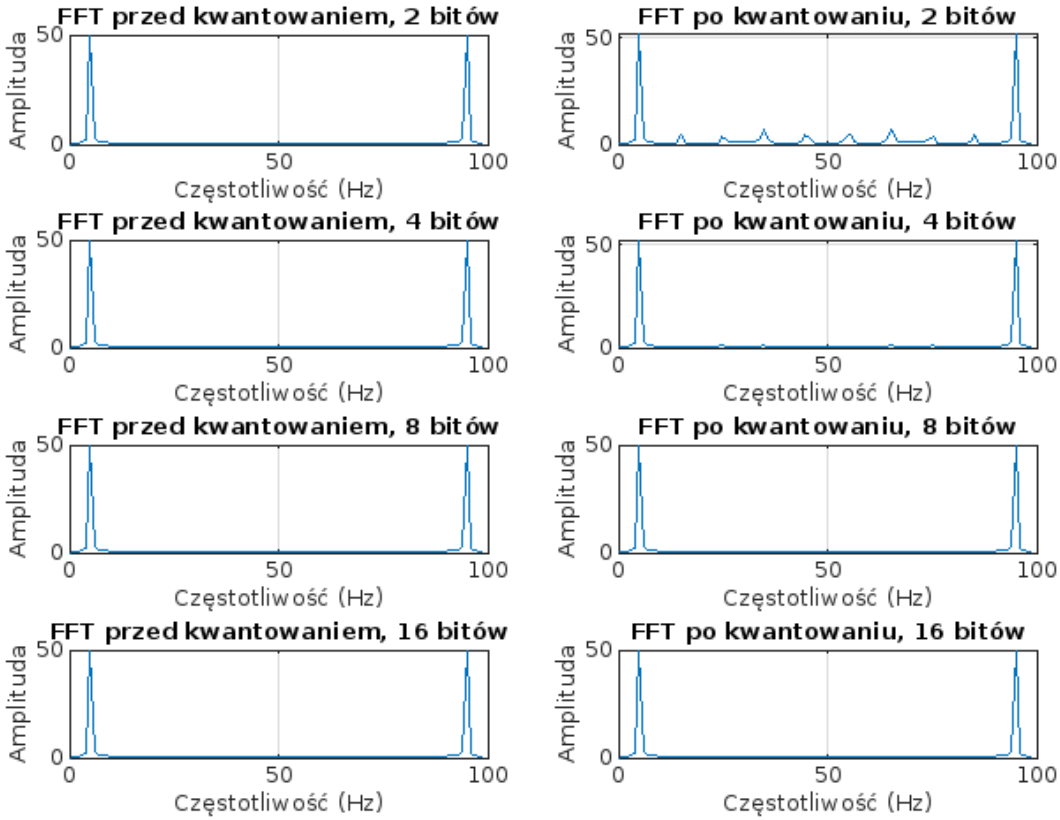
```
f = 5;
fs = 100;
t = 0:1/fs:1;
x_t = sin(2 * pi * f * t);

n_bits = [2, 4, 8, 16];

figure;
for i = 1:length(n_bits)
    L = 2^n_bits(i);
    x_max = max(x_t);
    x_min = min(x_t);
    quant_step = (x_max - x_min) / (L - 1);
    x_quant = round((x_t - x_min) / quant_step) * quant_step + x_min;
```

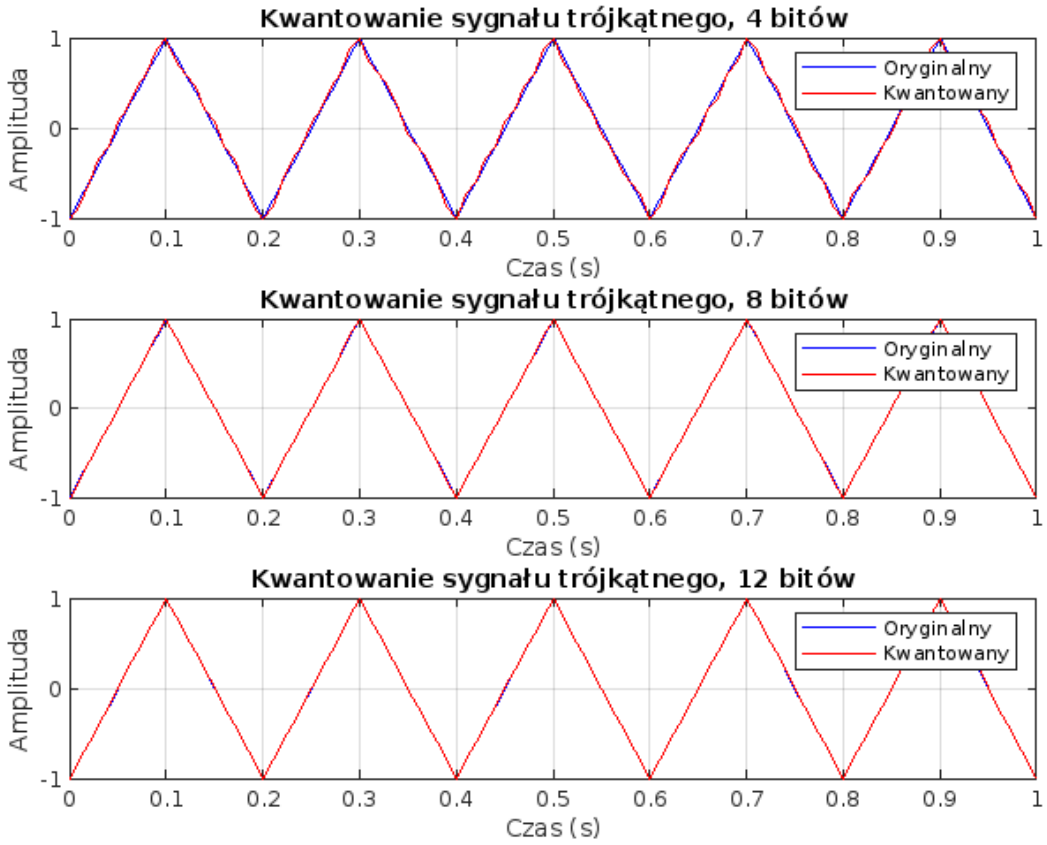
```
    X_fft = abs(fft(x_t));
    X_quant_fft = abs(fft(x_quant));

    freq = (0:length(X_fft)-1) * fs / length(X_fft);

    subplot(length(n_bits), 2, 2*i-1);
    plot(freq, X_fft);
    title(['FFT przed kwantowaniem, ', num2str(n_bits(i)), ' bitów']);
    xlabel('Częstotliwość (Hz)');
    ylabel('Amplituda');
    grid on;

    subplot(length(n_bits), 2, 2*i);
    plot(freq, X_quant_fft);
    title(['FFT po kwantowaniu, ', num2str(n_bits(i)), ' bitów']);
    xlabel('Częstotliwość (Hz)');
    ylabel('Amplituda');
    grid on;
end
```



Zadanie 17

```
f = 5;
fs = 100;
t = 0:1/fs:1;
x_t = sawtooth(2 * pi * f * t, 0.5);

n_bits = [4, 8, 12];

figure;
for i = 1:length(n_bits)
    L = 2^n_bits(i);
    x_max = max(x_t);
    x_min = min(x_t);
    quant_step = (x_max - x_min) / (L - 1);
    x_quant = round((x_t - x_min) / quant_step) * quant_step + x_min;

    subplot(length(n_bits), 1, i);
    plot(t, x_t, 'b');
    hold on;
    plot(t, x_quant, 'r');
    hold off;
    title(['Kwantowanie sygnału trójkątnego, ', num2str(n_bits(i)), '
bitów']);
    xlabel('Czas (s)');
    ylabel('Amplituda');
    legend('Oryginalny', 'Kwantowany');
    grid on;
end
```


FFT przed kwantowaniem, 2 bitów
FFT po kwantowaniu, 2 bitów
FFT przed kwantowaniem, 4 bitów
FFT po kwantowaniu, 4 bitów
FFT przed kwantowaniem, 8 bitów
FFT po kwantowaniu, 8 bitów
FFT przed kwantowaniem, 16 bitów
FFT po kwantowaniu, 16 bitów
Amplituda
Częstotliwość (Hz)
0
50
100

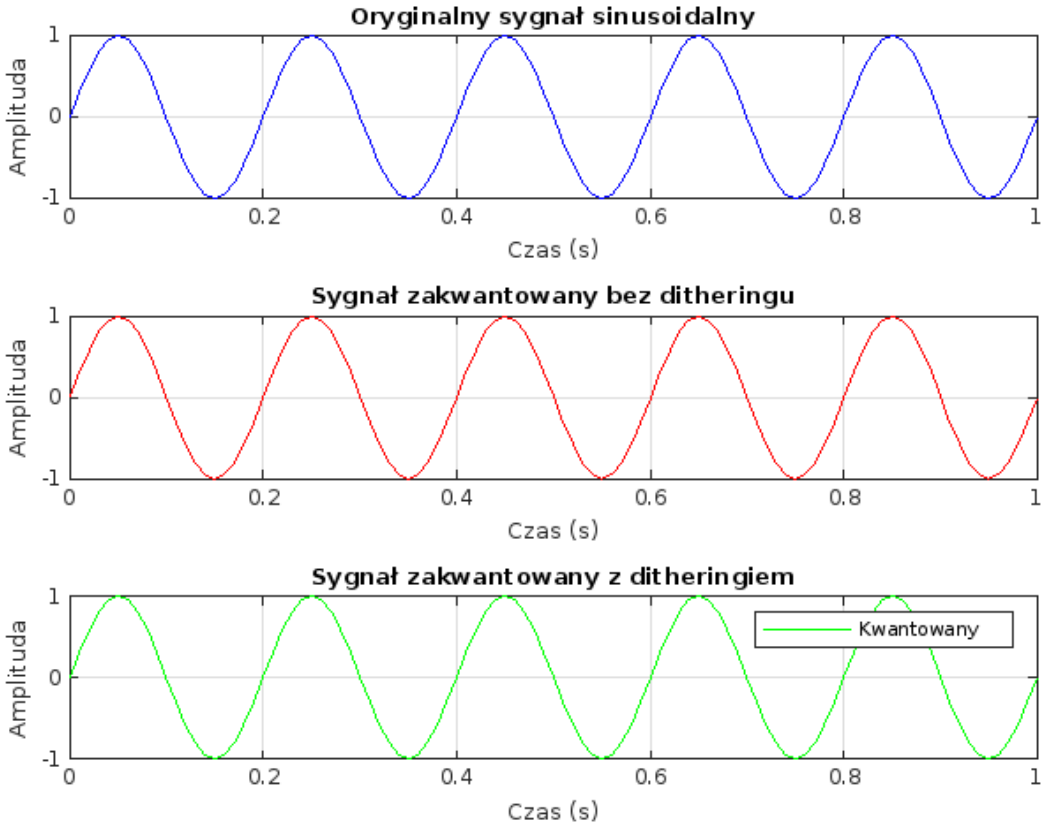


Zadanie 18

```
f = 5;
fs = 100;
t = 0:1/fs:1;
x_t = sin(2 * pi * f * t);

n_bits = 8;
L = 2^n_bits;
x_max = max(x_t);
x_min = min(x_t);
quant_step = (x_max - x_min) / (L - 1);

dither_amplitude = quant_step / 2;
dither = dither_amplitude * (rand(size(x_t)) - 0.5);
x_dithered = x_t + dither;

x_quant_dithered = round((x_dithered - x_min) / quant_step) * quant_step +
x_min;
x_quant = round((x_t - x_min) / quant_step) * quant_step + x_min;

figure;
```

```
subplot(3, 1, 1);
plot(t, x_t, 'b');
title('Oryginalny sygnał sinusoidalny');
xlabel('Czas (s)');
ylabel('Amplituda');
grid on;

subplot(3, 1, 2);
plot(t, x_quant, 'r');
title('Sygnał zakwantowany bez ditheringu');
xlabel('Czas (s)');
ylabel('Amplituda');
grid on;

subplot(3, 1, 3);
plot(t, x_quant_dithered, 'g');
title('Sygnał zakwantowany z ditheringiem');
xlabel('Czas (s)');
ylabel('Amplituda');
grid on;

legend('Kwantowany', 'Kwantowany z ditheringiem');
```

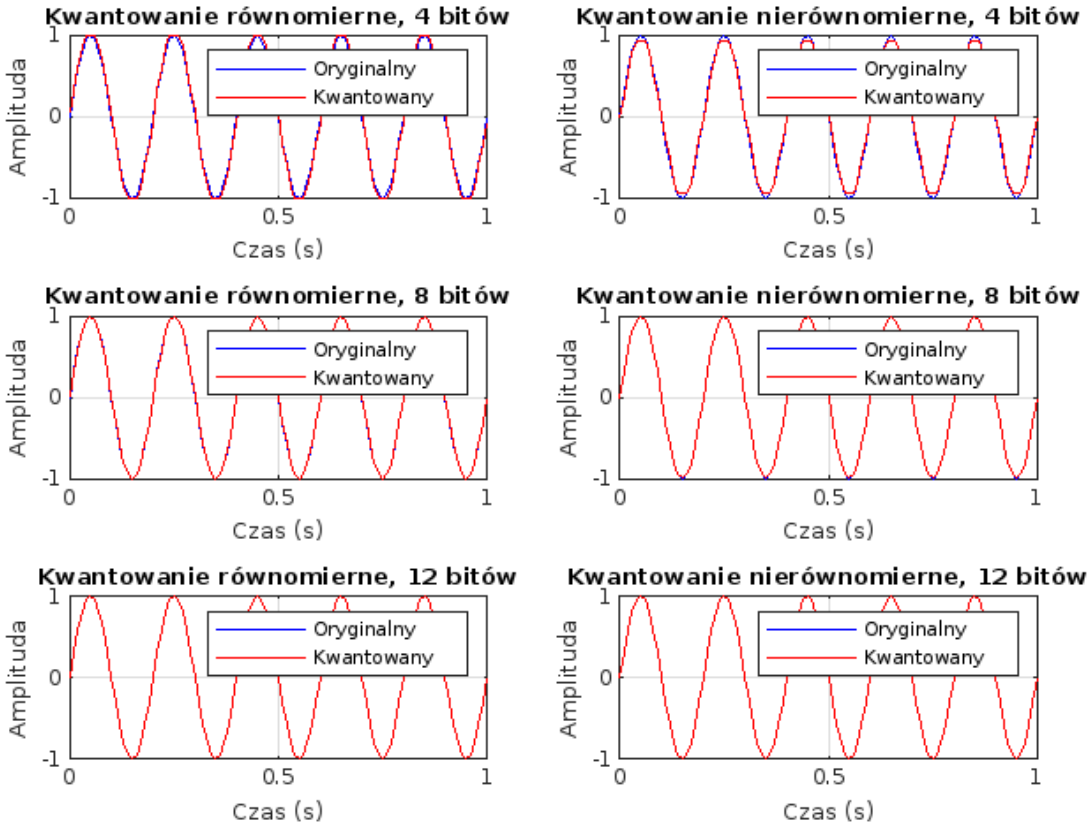
Warning: Ignoring extra legend entries.



Zadanie 19

```
f = 5;
fs = 100;
t = 0:1/fs:1;
x_t = sin(2 * pi * f * t);

n_bits = [4, 8, 12];

figure;
for i = 1:length(n_bits)
    L = 2^n_bits(i);
    x_max = max(x_t);
    x_min = min(x_t);
    quant_step = (x_max - x_min) / (L - 1);
    x_quant_uniform = round((x_t - x_min) / quant_step) * quant_step + x_min;

    mid_levels = linspace(x_min, x_max, L);
    quant_levels = mid_levels + quant_step / 2;
    x_quant_nonuniform = zeros(size(x_t));

    for j = 1:length(x_t)
        [~, idx] = min(abs(quant_levels - x_t(j)));
```

```
        x_quant_nonuniform(j) = quant_levels(idx);
    end

    subplot(length(n_bits), 2, (i-1)*2 + 1);
    plot(t, x_t, 'b');
    hold on;
    plot(t, x_quant_uniform, 'r');
    hold off;
    title(['Kwantowanie równomierne, ', num2str(n_bits(i)), ' bitów']);
    xlabel('Czas (s)');
    ylabel('Amplituda');
    legend('Oryginalny', 'Kwantowany');
    grid on;

    subplot(length(n_bits), 2, (i-1)*2 + 2);
    plot(t, x_t, 'b');
    hold on;
    plot(t, x_quant_nonuniform, 'r');
    hold off;
    title(['Kwantowanie nierównomierne, ', num2str(n_bits(i)), ' bitów']);
    xlabel('Czas (s)');
    ylabel('Amplituda');
    legend('Oryginalny', 'Kwantowany');
    grid on;
end
```

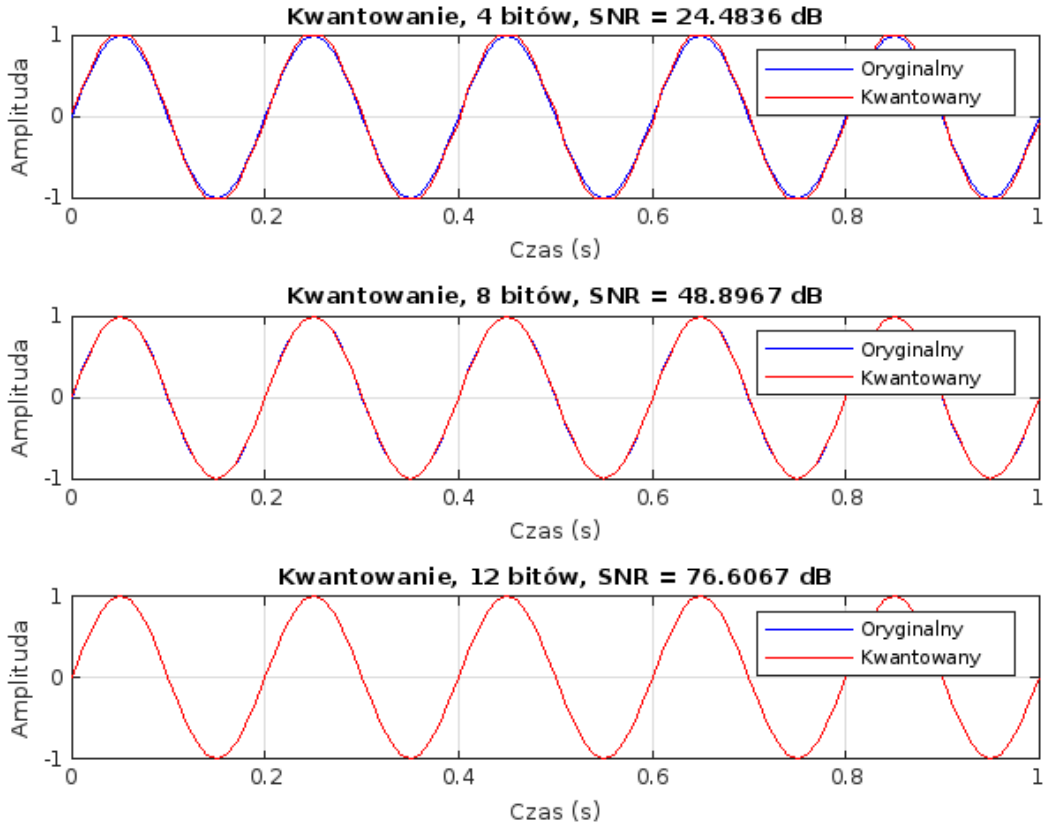


Zadanie 20

```
f = 5;
fs = 100;
t = 0:1/fs:1;
x_t = sin(2 * pi * f * t);
n_bits = [4, 8, 12];
snr_values = zeros(1, length(n_bits));

figure;
for i = 1:length(n_bits)
    L = 2^n_bits(i);
    x_max = max(x_t);
    x_min = min(x_t);
    quant_step = (x_max - x_min) / (L - 1);
    x_quant = round((x_t - x_min) / quant_step) * quant_step + x_min;

    noise = x_t - x_quant;
    signal_power = sum(x_t.^2) / length(x_t);
    noise_power = sum(noise.^2) / length(noise);
    snr_values(i) = 10 * log10(signal_power / noise_power);

    subplot(length(n_bits), 1, i);
```

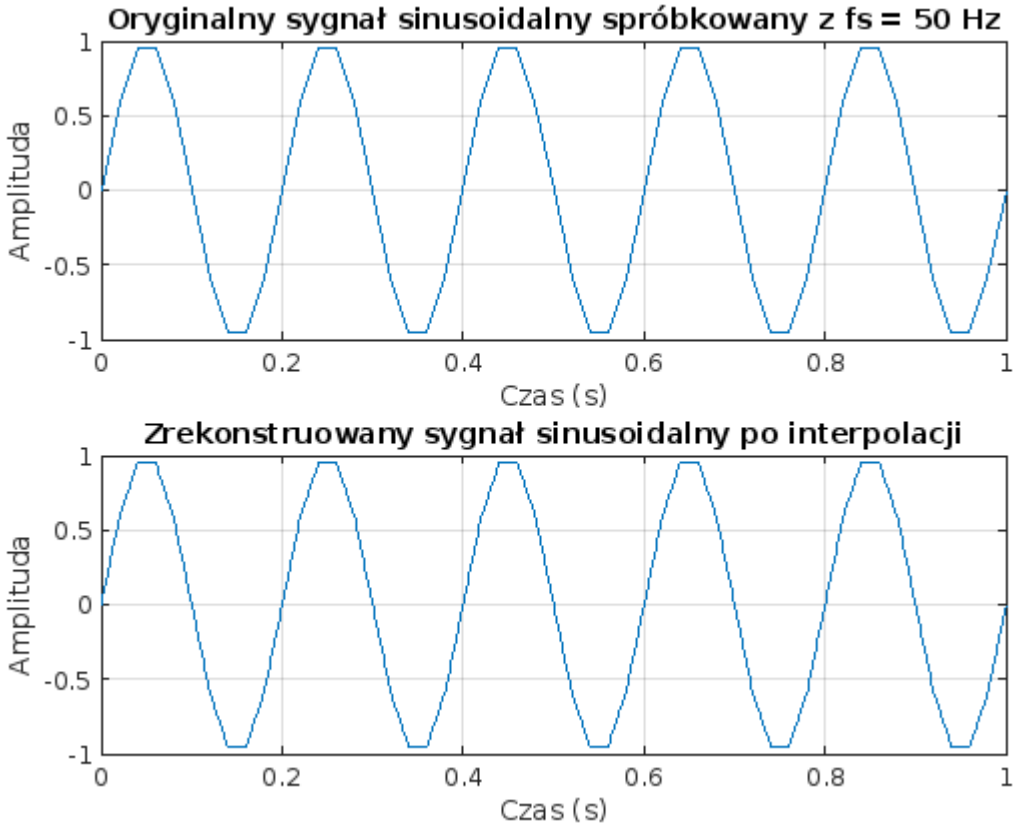
```
    plot(t, x_t, 'b');
    hold on;
    plot(t, x_quant, 'r');
    hold off;
    title(['Kwantowanie, ', num2str(n_bits(i)), ' bitów, SNR = ',
num2str(snr_values(i)), ' dB']);
    xlabel('Czas (s)');
    ylabel('Amplituda');
    legend('Oryginalny', 'Kwantowany');
    grid on;
end

disp('Wartości SNR dla różnych poziomów bitów:');
for i = 1:length(n_bits)
    fprintf('%d bitów: SNR = %.2f dB\n', n_bits(i), snr_values(i));
end
```

```
Wartości SNR dla różnych poziomów bitów:
4 bitów: SNR = 24.48 dB
8 bitów: SNR = 48.90 dB
12 bitów: SNR = 76.61 dB
```



Zadanie 21

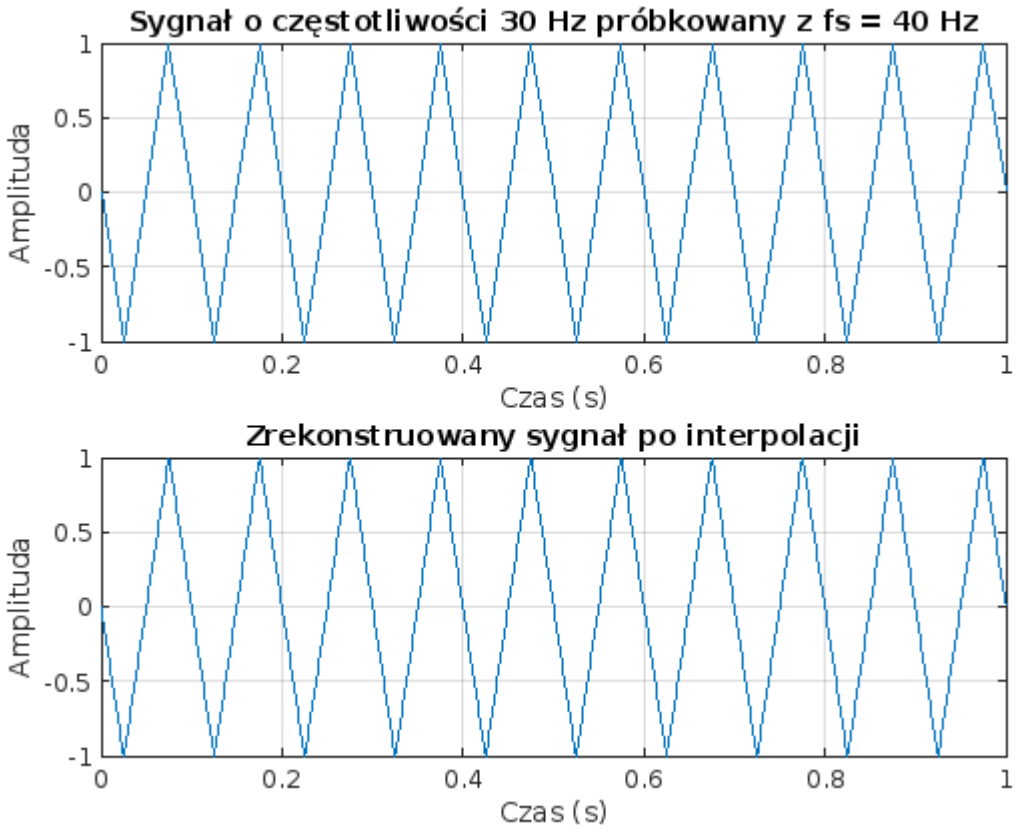
```
f = 5;
fs = 50;
t = 0:1/fs:1;
x = sin(2 * pi * f * t);

figure;
subplot(2, 1, 1);
plot(t, x);
title('Oryginalny sygnał sinusoidalny spróbkowany z fs = 50 Hz');
xlabel('Czas (s)');
ylabel('Amplituda');
grid on;

fs_interp = 1000;
t_interp = 0:1/fs_interp:1;
x_interp = interp1(t, x, t_interp);

subplot(2, 1, 2);
plot(t_interp, x_interp);
title('Zrekonstruowany sygnał sinusoidalny po interpolacji');
xlabel('Czas (s)');
ylabel('Amplituda');
grid on;
```



Zadanie 22

```
A = 1;
f = 30;
fs = 40;
t = 0:1/fs:1;
x = A * sin(2 * pi * f * t);

figure;
subplot(2, 1, 1);
plot(t, x );
title('Sygnał o częstotliwości 30 Hz próbkowany z fs = 40 Hz');
xlabel('Czas (s)');
ylabel('Amplituda');
grid on;

fs_interp = 1000;
t_interp = 0:1/fs_interp:1;
x_interp = interp1(t, x, t_interp);

subplot(2, 1, 2);
plot(t_interp, x_interp);
title('Zrekonstruowany sygnał po interpolacji');
xlabel('Czas (s)');
```

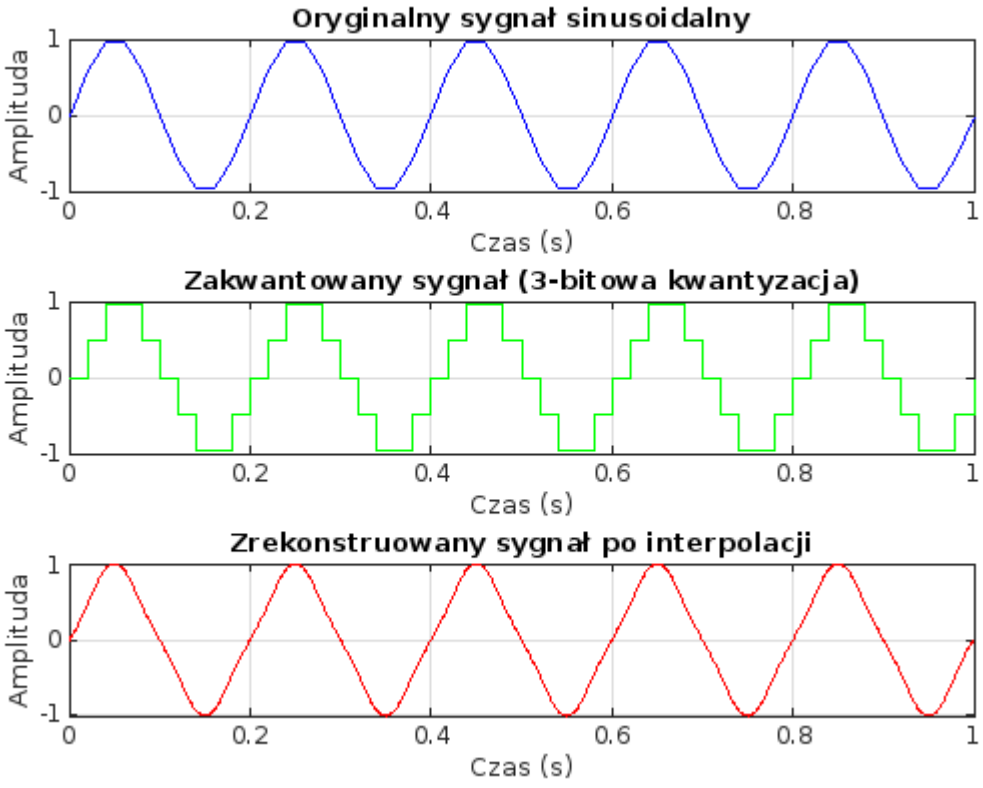
```
ylabel('Amplituda');
grid on;
```



Zadanie 23

```
A = 1;
f = 5;
fs = 50;
t = 0:1/fs:1;
x = A * sin(2 * pi * f * t);

n_bits = 3;
levels = 2^n_bits;
x_min = min(x);
x_max = max(x);
step_size = (x_max - x_min) / levels;
x_quant = round((x - x_min) / step_size) * step_size + x_min;

fs_interp = 1000;
t_interp = 0:1/fs_interp:1;
x_interp = interp1(t, x_quant, t_interp, 'spline');

figure;

subplot(3, 1, 1);
```

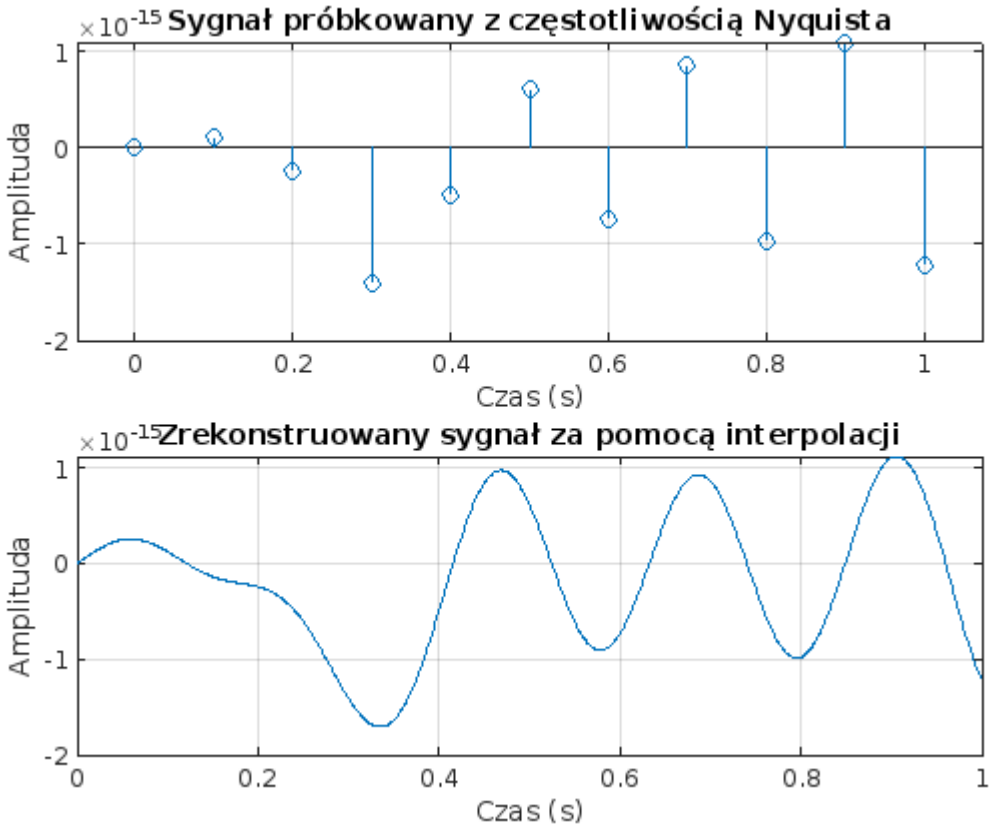
```
plot(t, x, 'b');
title('Oryginalny sygnał sinusoidalny');
xlabel('Czas (s)');
ylabel('Amplituda');
grid on;

subplot(3, 1, 2);
stairs(t, x_quant, 'g');
title(['Zakwantowany sygnał (', num2str(n_bits), '-bitowa kwantyzacja)']);
xlabel('Czas (s)');
ylabel('Amplituda');
grid on;

subplot(3, 1, 3);
plot(t_interp, x_interp, 'r');
title('Zrekonstruowany sygnał po interpolacji');
xlabel('Czas (s)');
ylabel('Amplituda');
grid on;
```



Zadanie 24

```
A = 1;
f = 5;
fs = 2 * f;
```

```
t = 0:1/fs:1;
x = A * sin(2 * pi * f * t);

fs_interp = 1000;
t_interp = 0:1/fs_interp:1;

x_interp = zeros(size(t_interp));
for i = 1:length(t)
    x_interp = x_interp + x(i) * sinc(fs * (t_interp - t(i)));
end

figure;
subplot(2, 1, 1);
stem(t, x);
title('Sygnał próbkowany z częstotliwością Nyquista');
xlabel('Czas (s)');
ylabel('Amplituda');
grid on;

subplot(2, 1, 2);
plot(t_interp, x_interp);
title('Zrekonstruowany sygnał za pomocą interpolacji');
xlabel('Czas (s)');
ylabel('Amplituda');
grid on;
```



Zadanie 25

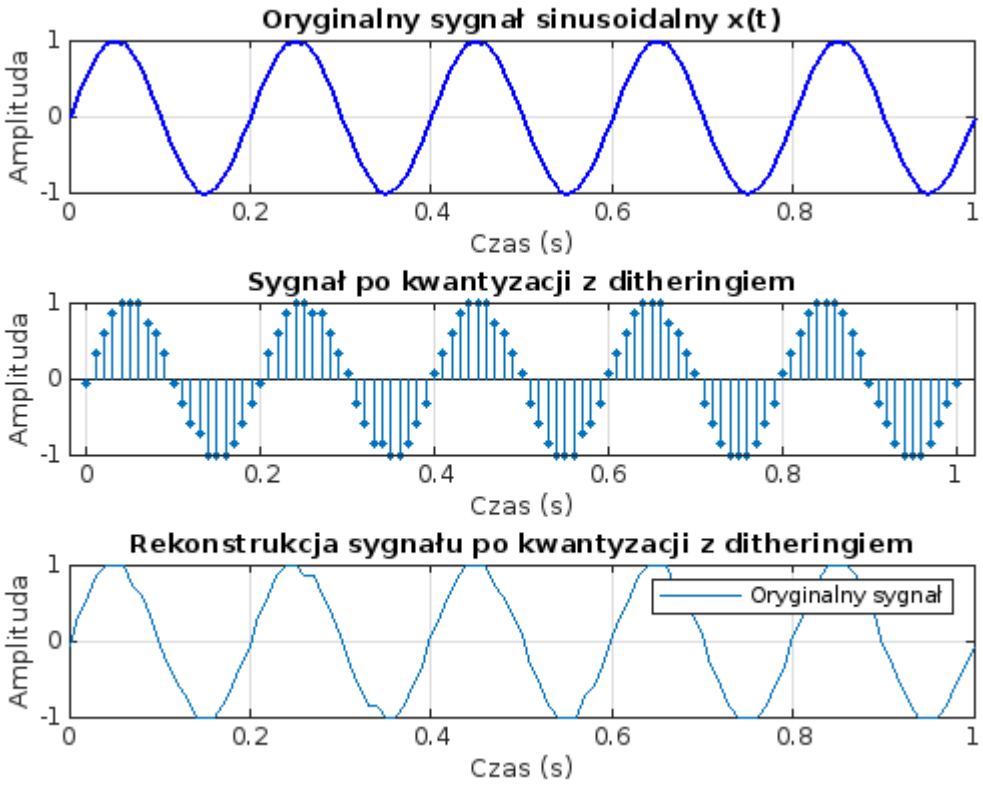
Zadanie 26

Zadanie 27

Zadanie 28

```
A = 1;
f = 5;
fs = 100;
t = 0:1/fs:1;

x = A * sin(2 * pi * f * t);

levels = 16;
step_size = 2 * A / (levels - 1);

dither = rand(size(x)) * (step_size / 2) - (step_size / 4);
x_dithered = x + dither;

x_quantized = round((x_dithered + A) / (2 * A) * (levels - 1)) / (levels - 1)
* 2 * A - A;

x_reconstructed = interp1(t, x_quantized, t, 'linear', 'extrap');

figure;

subplot(3, 1, 1);
plot(t, x, 'b', 'LineWidth', 1.5);
title('Oryginalny sygnał sinusoidalny x(t)');
xlabel('Czas (s)');
ylabel('Amplituda');
grid on;

subplot(3, 1, 2);
stem(t, x_quantized, 'filled', 'MarkerSize', 3);
title('Sygnał po kwantyzacji z ditheringiem');
xlabel('Czas (s)');
ylabel('Amplituda');
grid on;

subplot(3, 1, 3);
plot(t, x);
plot(t,x_reconstructed);
title('Rekonstrukcja sygnału po kwantyzacji z ditheringiem');
xlabel('Czas (s)');
ylabel('Amplituda');
legend('Oryginalny sygnał', 'Zrekonstruowany sygnał');
grid on;
```

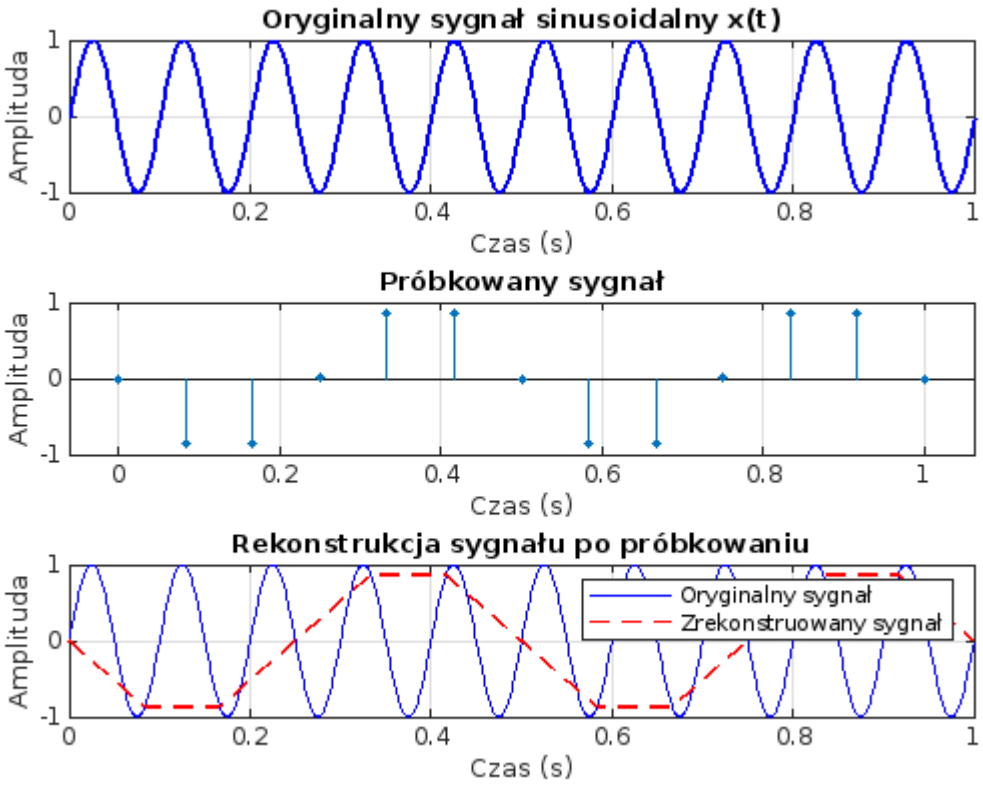
Warning: Ignoring extra legend entries.



Zadanie 29

```
A = 1;
f = 10;
fs = 12;
t = 0:0.001:1;

x = A * sin(2 * pi * f * t);


t_sampled = 0:1/fs:1;
x_sampled = A * sin(2 * pi * f * t_sampled);

x_reconstructed = interp1(t_sampled, x_sampled, t, 'linear', 'extrap');

figure;
subplot(3, 1, 1);
plot(t, x, 'b', 'LineWidth', 1.5);
title('Oryginalny sygnał sinusoidalny x(t)');
xlabel('Czas (s)');
ylabel('Amplituda');
grid on;
```

```
subplot(3, 1, 2);
stem(t_sampled, x_sampled, 'filled', 'MarkerSize', 3);
title('Próbkowany sygnał');
xlabel('Czas (s)');
ylabel('Amplituda');
grid on;

subplot(3, 1, 3);
plot(t, x, 'b', t, x_reconstructed, 'r--', 'LineWidth', 1);
title('Rekonstrukcja sygnału po próbkowaniu');
xlabel('Czas (s)');
ylabel('Amplituda');
legend('Oryginalny sygnał', 'Zrekonstruowany sygnał');
grid on;
```



Published with MATLAB® R2024b