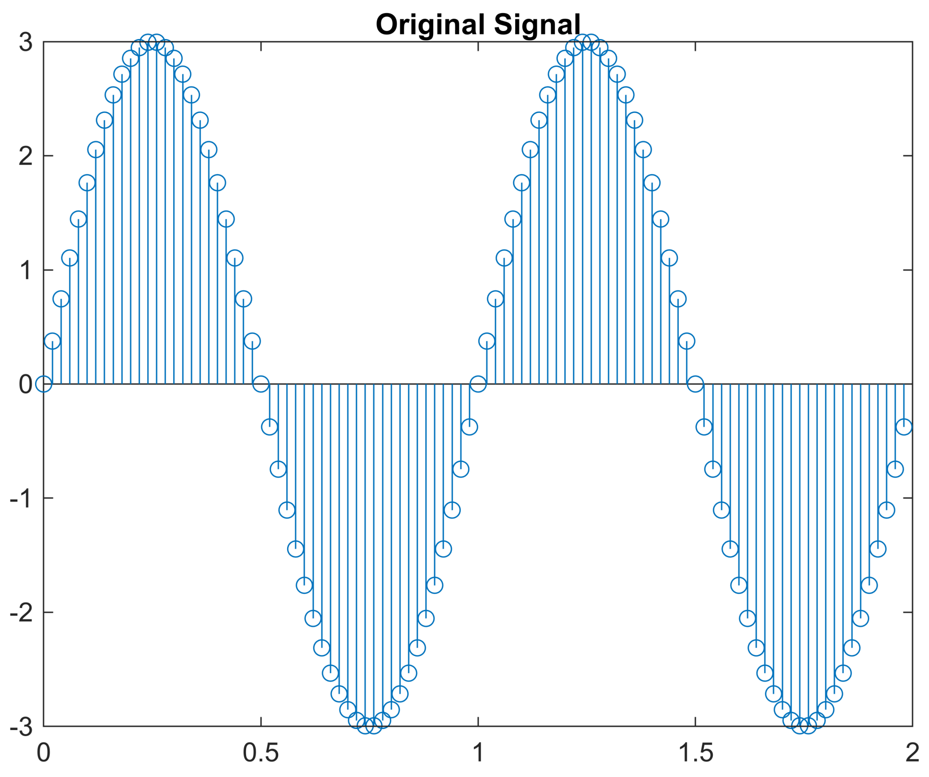
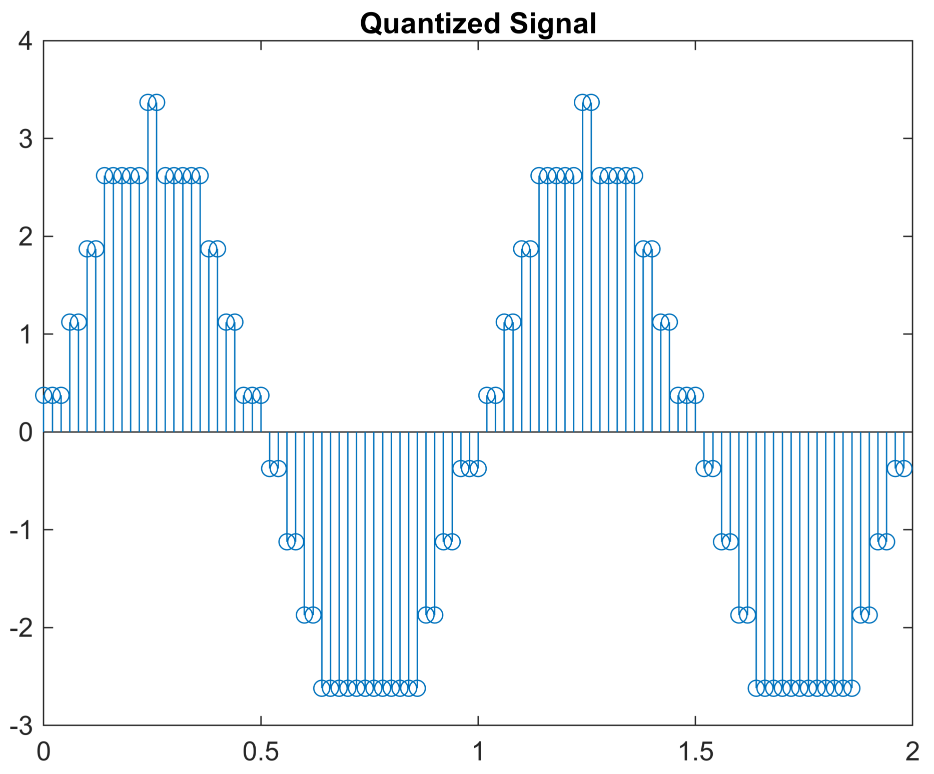
PART A (I)

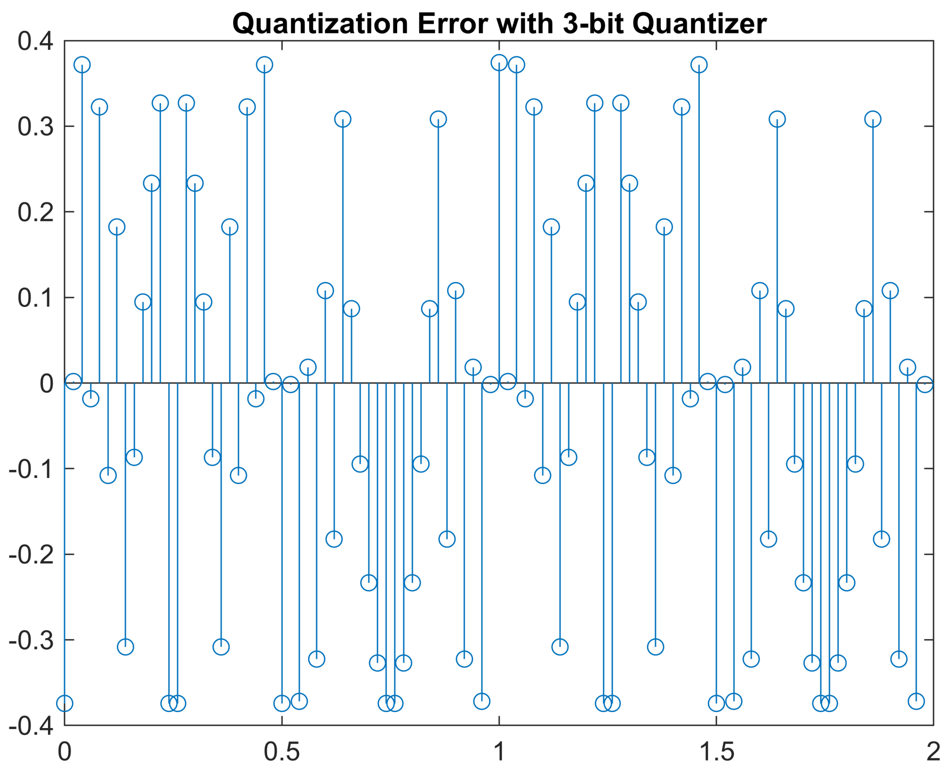
```
A = 3;
f = 1;
T = 2;
F = 50;

n = 0:1/F:T-1/F;
x = A * sin(2 * pi * f * n);

L = 2^3;
delta = (2 * max(x)) / L;
q_levels_a = (-max(x) + delta/2):delta:(max(x) - delta/2);

x_quantized = delta * floor(x/delta) + delta/2;

quantization_error = x - x_quantized;

signal_power = mean(x .^ 2);
noise_power = mean(quantization_error .^ 2);
SNR = 10 * log10(signal_power / noise_power);

figure(1);
stem(n, x);
title('Original Signal');
```



```
figure(2);
stem(n, x_quantized);
title('Quantized Signal');
```



```
figure(3);
stem(n, quantization_error);
title('Quantization Error with 3-bit Quantizer');
```

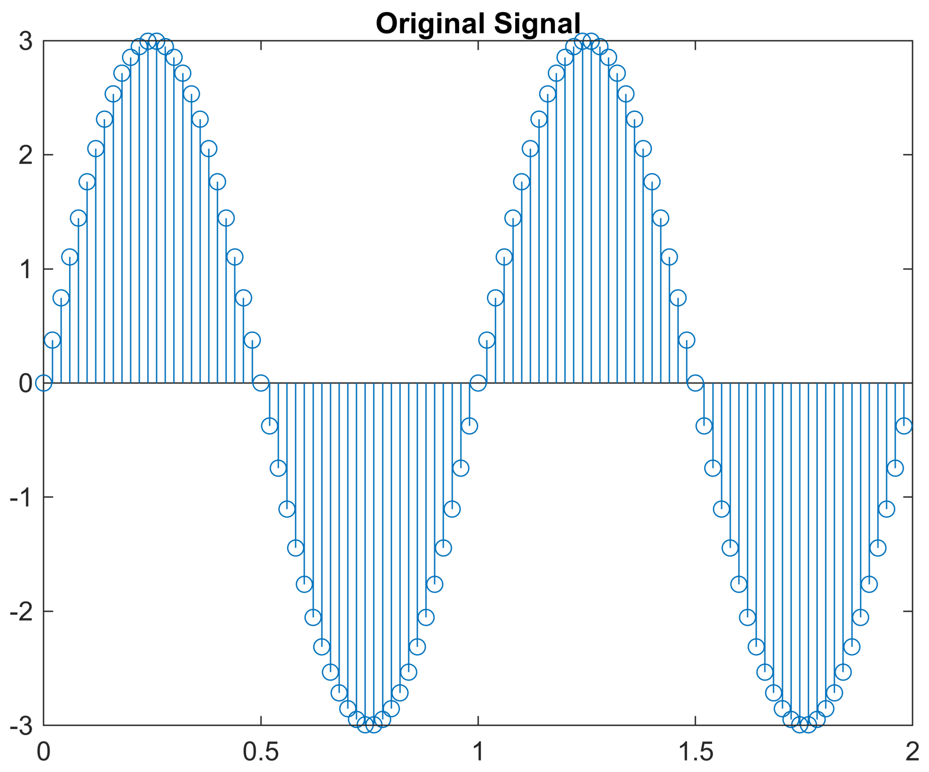


```
fprintf('Output Signal-to-Noise Ratio (SNR) with 3-bit Quantizer: %f dB\n', SNR);
```

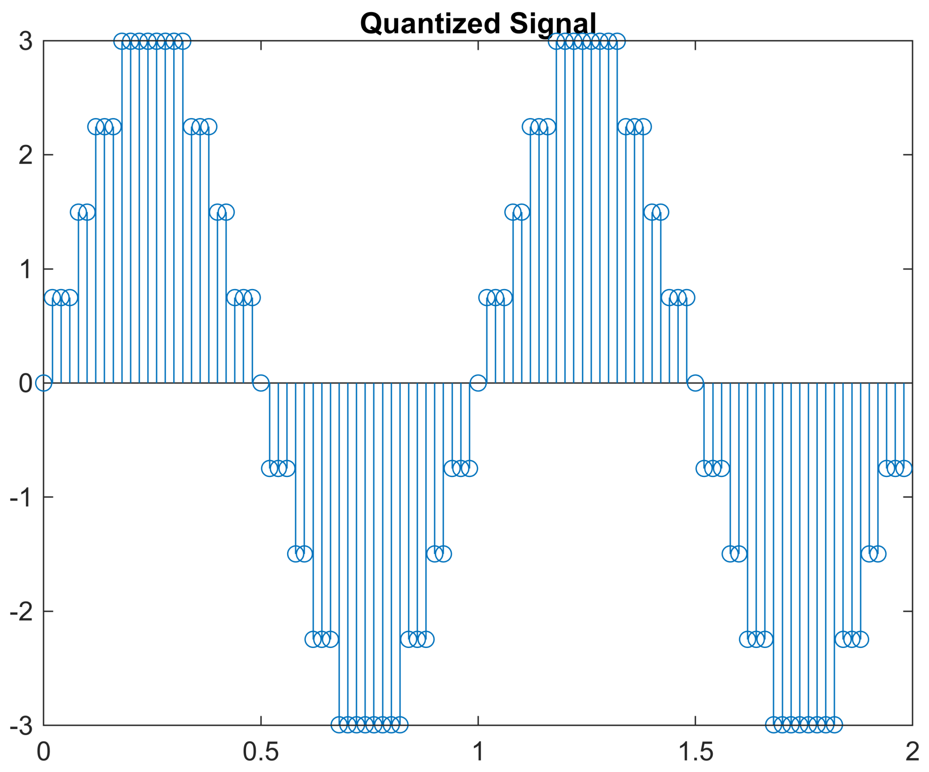
```
Output Signal-to-Noise Ratio (SNR) with 3-bit Quantizer: 18.633327 dB
```

PART A (II)

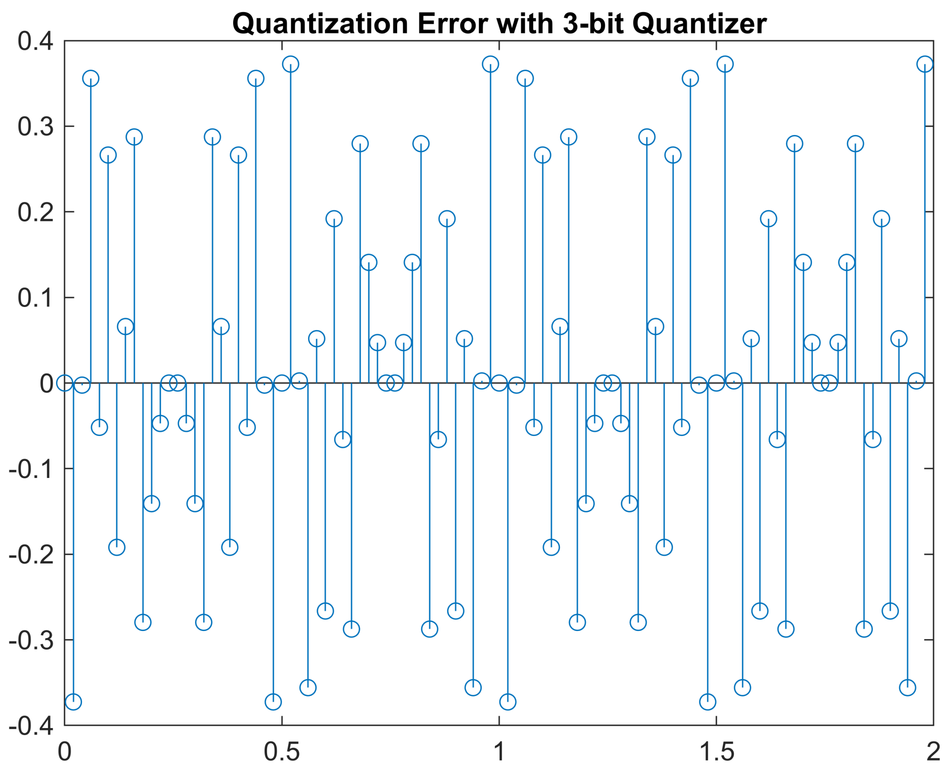
```
x_quantized = delta * round(x/delta);

quantization_error = x - x_quantized;

signal_power = mean(x .^ 2);
noise_power = mean(quantization_error .^ 2);
SNR = 10 * log10(signal_power / noise_power);

figure(4);
stem(n, x);
title('Original Signal');
```



```
figure(5);
stem(n, x_quantized);
title('Quantized Signal');
```



```
figure(6);
stem(n, quantization_error);
title('Quantization Error with 3-bit Quantizer');
```



```
fprintf('Output Signal-to-Noise Ratio (SNR) with 3-bit Quantizer: %f dB\n', SNR);
```

```
Output Signal-to-Noise Ratio (SNR) with 3-bit Quantizer: 19.995572 dB
```

PART B

```
x1 = 0.5 * x;
x1_quantized = delta * floor(x1/delta) + delta/2;
quantization_error1 = x1 - x1_quantized;

x2 = 2 * x;
x2_quantized = delta * floor(x2/delta) + delta/2;
quantization_error2 = x2 - x2_quantized;

signal_power1 = mean(x1 .^ 2);
noise_power1 = mean(quantization_error1 .^ 2);
SNR1 = 10 * log10(signal_power1 / noise_power1);

signal_power2 = mean(x2 .^ 2);
noise_power2 = mean(quantization_error2 .^ 2);
SNR2 = 10 * log10(signal_power2 / noise_power2);

disp(['Output Signal-to-Noise Ratio (SNR) with x1[n] Quantizer: ', num2str(SNR1), '
dB']);
```

```
Output Signal-to-Noise Ratio (SNR) with x1[n] Quantizer: 12.6889 dB
```

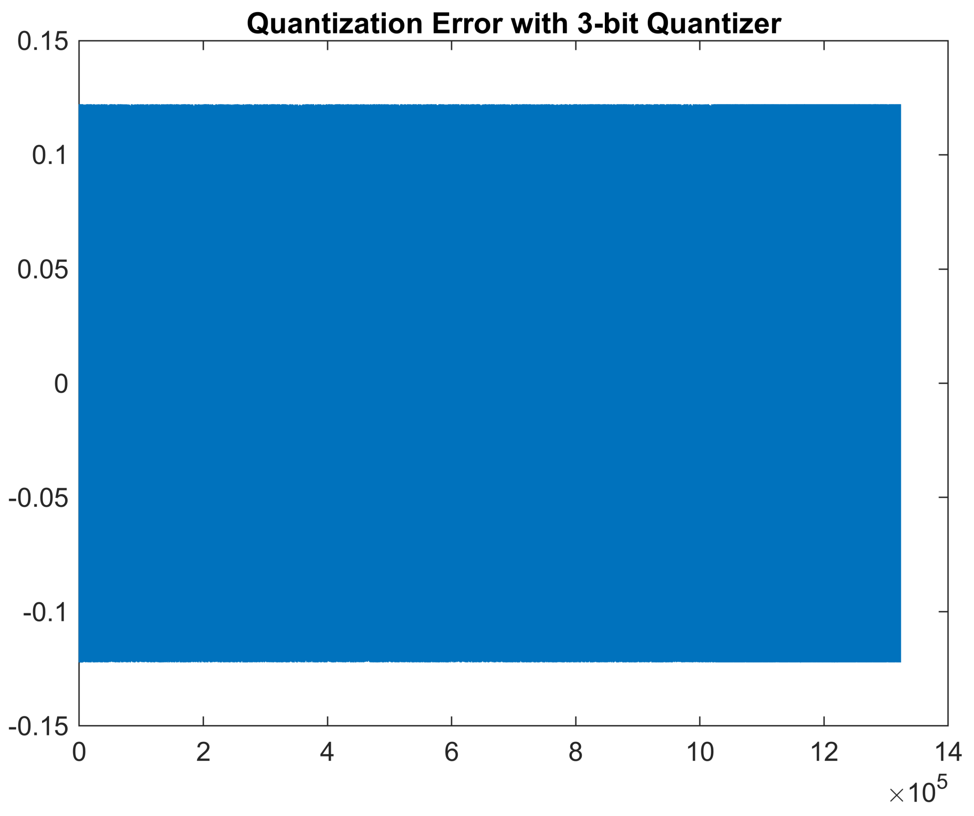
```
disp(['Output Signal-to-Noise Ratio (SNR) with x2[n] Quantizer: ', num2str(SNR2), '
dB']);
```

```
Output Signal-to-Noise Ratio (SNR) with x2[n] Quantizer: 23.9013 dB
```

```

% Comments
% The SNR for x1[n] is low because the signal level is lowered and the quantizer
levels remain the same.
% The SNR for x2[n] is high because the signal level is increased and the quantizer
levels remain the same.
```

PART C

```
[x, Fs] = audioread('C:\sound.wav');

A_max = max(abs(x));

L = 2^3;
delta = (2 * A_max) / L;

q_levels_c = (-A_max + delta/2):delta:(A_max - delta/2);

x_quantized = delta * floor(x/delta) + delta/2;

quantization_error = x - x_quantized;

%sound(x, Fs);
%pause(length(x)/Fs);
%sound(x_quantized, Fs);

signal_power = mean(x .^ 2);
noise_power = mean(quantization_error .^ 2);
SNR = 10 * log10(signal_power / noise_power);

figure(7);
plot(quantization_error);
title('Quantization Error with 3-bit Quantizer');
```



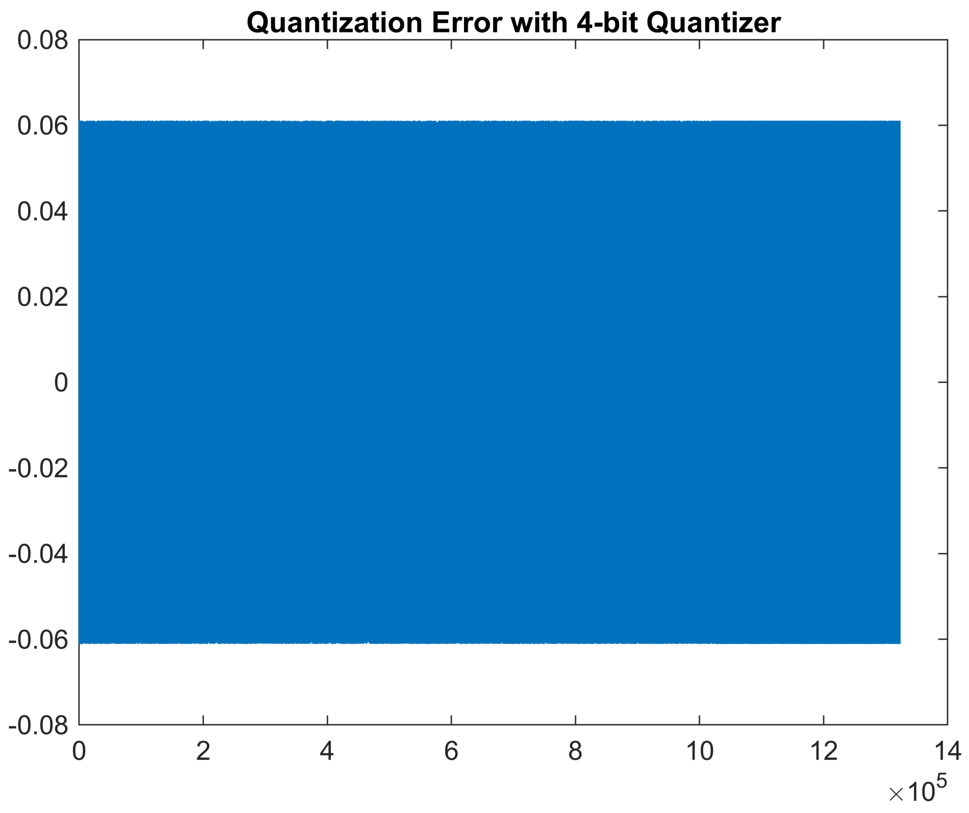
```
fprintf('Output Signal-to-Noise Ratio (SNR) with 3-bit Quantizer: %f dB\n', SNR);
```

```
Output Signal-to-Noise Ratio (SNR) with 3-bit Quantizer: 0.459233 dB
```

PART D

```
[x, Fs1] = audioread('C:\sound.wav');

A_max = max(abs(x));

num_bits = 4;
L = 2^num_bits;

delta = (2 * A_max) / L;

q_levels_d = (-A_max + delta/2):delta:(A_max - delta/2);

x_quantized = delta * floor(x/delta) + delta/2;

quantization_error = x - x_quantized;

%sound(x, Fs1);
%pause(length(x)/Fs1);
%sound(x_quantized, Fs1);
```

```
signal_power = mean(x .^ 2);
noise_power = mean(quantization_error .^ 2);

SNR = 10 * log10(signal_power / noise_power);

figure(8);
plot(quantization_error);
title('Quantization Error with 4-bit Quantizer');
```



```
fprintf('Output Signal-to-Noise Ratio (SNR) with 4-bit Quantizer: %f dB\n', SNR);
```

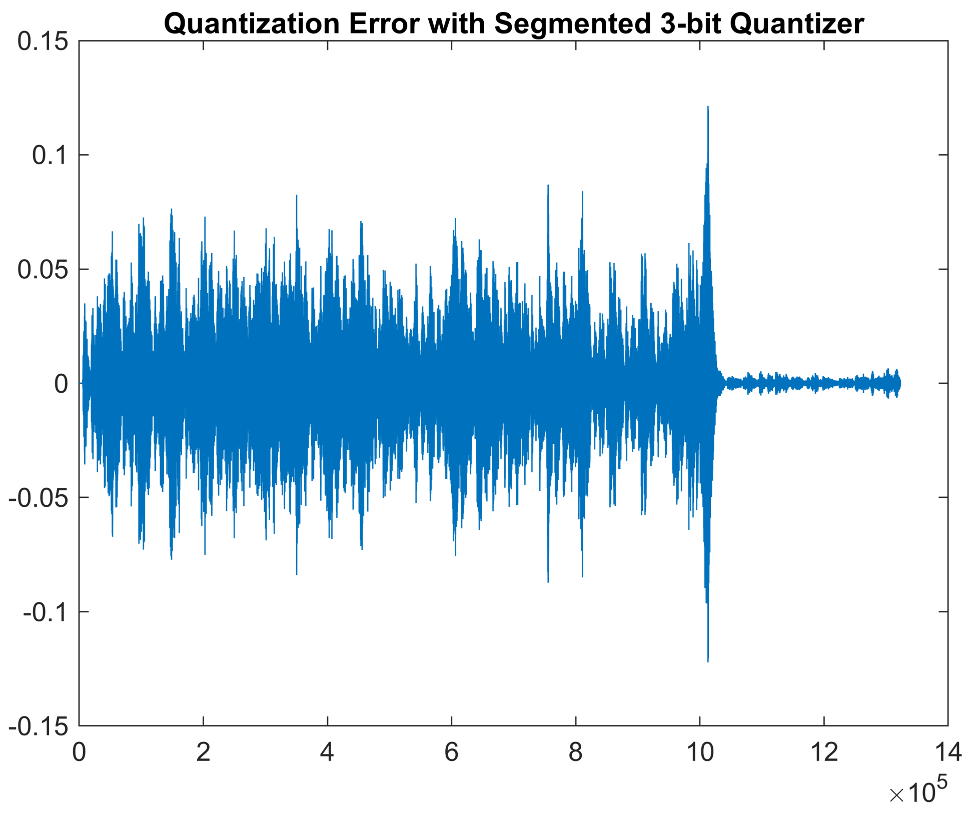
```
Output Signal-to-Noise Ratio (SNR) with 4-bit Quantizer: 7.018802 dB
```

```
% Comment on the results by compare with PART C
% The SNR should be higher with a 4-bit quantizer compared to a 3-bit quantizer
% because the quantization step size is smaller, leading to a finer resolution and
% less quantization error.
```

PART E

```
[x, Fs] = audioread('C:\sound.wav');
segment_length = 100;
num_segments = floor(length(x) / segment_length);
quantized_segments = zeros(size(x));
quantization_error_segments = zeros(size(x));
```

```
for i = 1:num_segments
    segment_start = (i-1) * segment_length + 1;
    segment_end = i * segment_length;
    segment = x(segment_start:segment_end);

    A_max_segment = max(abs(segment));
    L = 2^3;
    delta = (2 * A_max_segment) / L;
    q_levels = (-A_max_segment + delta/2):delta:(A_max_segment - delta/2);

    segment_quantized = delta * floor(segment/delta) + delta/2;
    quantized_segments(segment_start:segment_end) = segment_quantized;

    quantization_error = segment - segment_quantized;
    quantization_error_segments(segment_start:segment_end) = quantization_error;
end

if length(x) > num_segments * segment_length
    last_segment_start = num_segments * segment_length + 1;
    last_segment = x(last_segment_start:end);

    A_max_last_segment = max(abs(last_segment));
    delta = (2 * A_max_last_segment) / L;
    q_levels_last = (-A_max_last_segment + delta/2):delta:(A_max_last_segment -
delta/2);

    last_segment_quantized = delta * floor(last_segment/delta) + delta/2;
    quantized_segments(last_segment_start:end) = last_segment_quantized;

    quantization_error_last = last_segment - last_segment_quantized;
    quantization_error_segments(last_segment_start:end) = quantization_error_last;
end

signal_power = mean(x .^ 2);
noise_power = mean(quantization_error_segments .^ 2);
SNR = 10 * log10(signal_power / noise_power);

%sound(x, Fs);
%pause(length(x)/Fs);
%sound(quantized_segments, Fs);

figure(9);
plot(quantization_error_segments);
title('Quantization Error with Segmented 3-bit Quantizer');
```



```
fprintf('Output Signal-to-Noise Ratio (SNR) with Segmented 3-bit Quantizer: %f
dB\n', SNR);
```

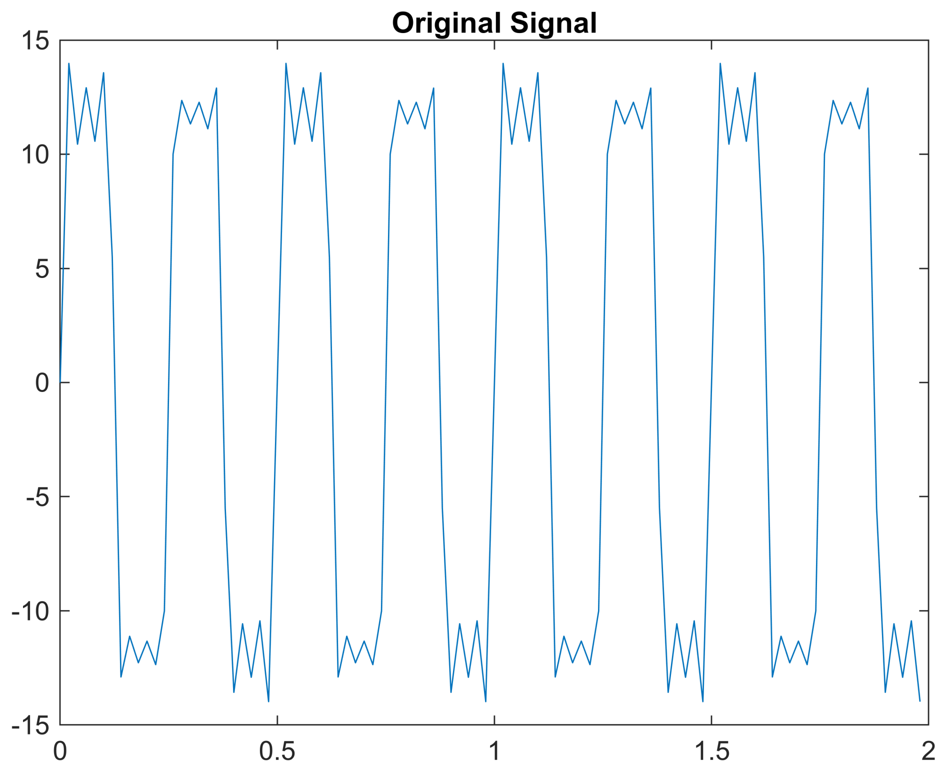
```
Output Signal-to-Noise Ratio (SNR) with Segmented 3-bit Quantizer: 15.567592 dB
```

```
% Comment on the results by comparing with PART C
% The segmented 3-bit quantizer approach used in this experiment provides a
% unique advantage over the uniform 3-bit quantizer applied in question (c).
% By partitioning the speech signal into segments and quantizing each segment
% individually, we cater to the varying amplitude ranges within different parts
% of the speech. This method allows for a more efficient use of the quantizer's
% dynamic range, potentially resulting in a higher Signal-to-Noise Ratio (SNR)
% when compared to the uniform quantization. The adaptive nature of this segmented
% quantization can lead to a more accurate representation of the speech signal,
% with reduced quantization error and improved auditory quality upon playback.
```

PART F

```
A1 = 15; f1 = 4; T1 = 2; F1 = 50;
A2 = 5; f2 = 12; T2 = 2; F2 = 50;
A3 = 3; f3 = 20; T3 = 2; F3 = 50;

t = 0:1/F1:T1-1/F1;
```

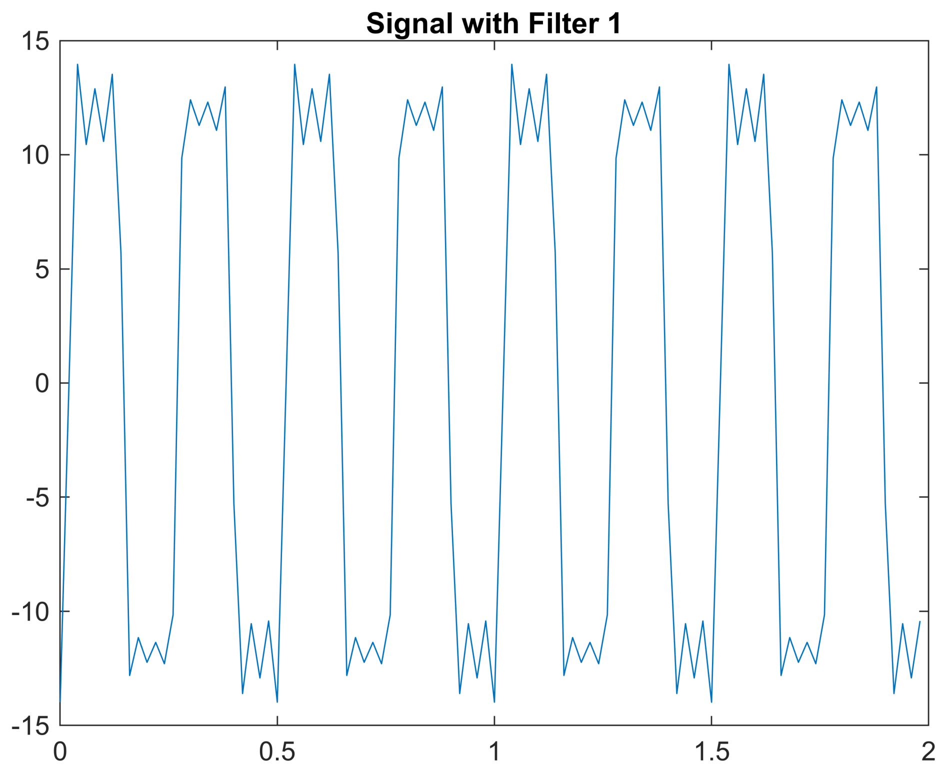
```
x = A1*sin(2*pi*f1*t) + A2*sin(2*pi*f2*t) + A3*sin(2*pi*f3*t);

H1 = @(w) exp(-1i*w);
H2 = @(w) exp(-1i*w.^2);

w = linspace(-pi, pi, length(t));

X = fftshift(fft(x));
X_filtered1 = X .* H1(w);
X_filtered2 = X .* H2(w);

x_filtered1 = ifft(ifftshift(X_filtered1));
x_filtered2 = ifft(ifftshift(X_filtered2));

figure(10);
plot(t, x);
title('Original Signal');
```



```
figure(11);
plot(t, x_filtered1);
```

```
Warning: Imaginary parts of complex X and/or Y arguments ignored.
```

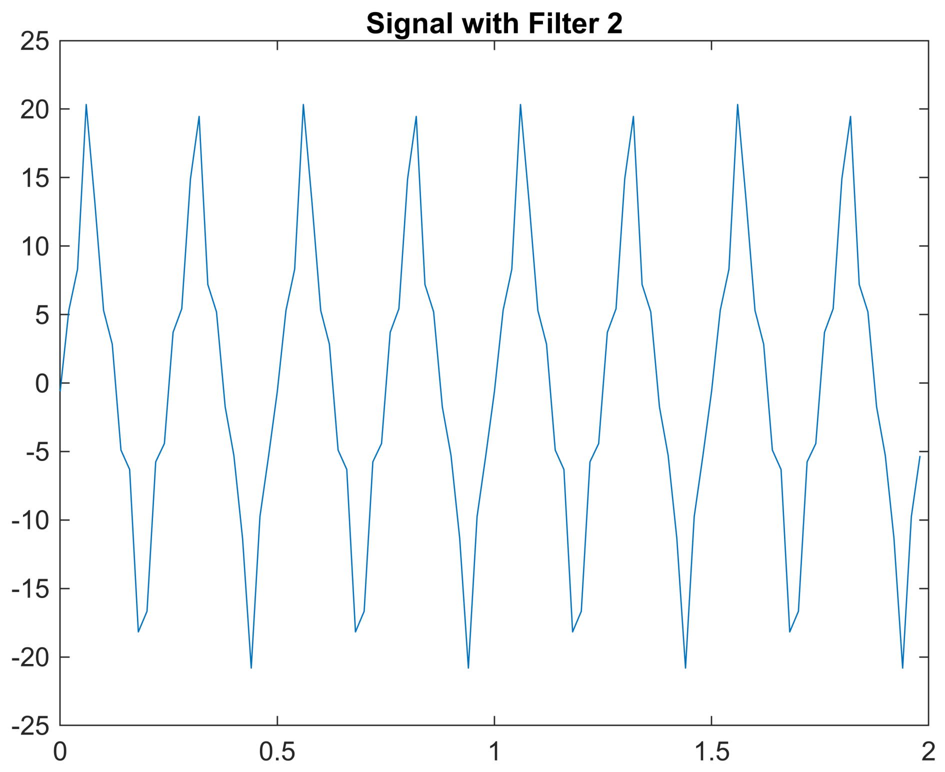
```
title('Signal with Filter 1');
```



```
figure(12);
plot(t, x_filtered2);
```

```
Warning: Imaginary parts of complex X and/or Y arguments ignored.
```

```
title('Signal with Filter 2');
```



```
% Comment on the results
% Filter 1 is a linear phase filter and does not change the shape of the signal.
% Filter 2 is a nonlinear phase filter and can change the shape of the signal.
```

PART G

```
[x, Fs] = audioread('C:\sound1.wav');

t = 0:1/Fs:(length(x)-1)/Fs;

w = linspace(-pi, pi, length(x));

X = fftshift(fft(x));

H1 = exp(-1i * w);
H2 = exp(-1i * w.^2);

X_filtered1 = X .* H1;
```

```
Requested 381952x381952 (2173.9GB) array exceeds maximum array size preference (15.8GB). This might cause
MATLAB to become unresponsive.

Related documentation
```

```
X_filtered2 = X .* H2;
```

```
x_filtered1 = ifft(ifftshift(X_filtered1));
x_filtered2 = ifft(ifftshift(X_filtered2));

figure(13);
plot(t, x);
title('Original Signal');
xlabel('Time (s)');
ylabel('Amplitude');

figure(14);
plot(t, real(x_filtered1));
title('Signal with Linear Phase Filter');
xlabel('Time (s)');
ylabel('Amplitude');

figure(15);
plot(t, real(x_filtered2));
title('Signal with Quadratic Phase Filter');
xlabel('Time (s)');
ylabel('Amplitude');

%sound(real(x_filtered1), Fs);
%pause(length(x)/Fs);
%sound(real(x_filtered2), Fs);

% Commets
% The application of the two different filters, H1 and H2, to the original
% signal x has resulted in two distinct filtered signals, x_filtered1 and
x_filtered2.
% Filter H1, which applies a linear phase shift, preserves the waveform of the
% original signal but shifts it in time. On the other hand, Filter H2 introduces a
% nonlinear phase shift, which can cause distortion in the waveform of the original
% signal. By examining the plots of the original and filtered signals, we can
observe
% these effects visually. Additionally, listening to the filtered signals would
% provide an auditory perspective on the impact of the phase shifts. The real part
% of the filtered signals is used for both plotting and sound playback, as the
% imaginary part should be negligible after inverse Fourier transform if the signal
% is real-valued. It's important to note that the quality of the filtered signals
% can also be quantitatively assessed by calculating the Signal-to-Noise Ratio
(SNR),
% which would give us an objective measure of the filters' performance.
```