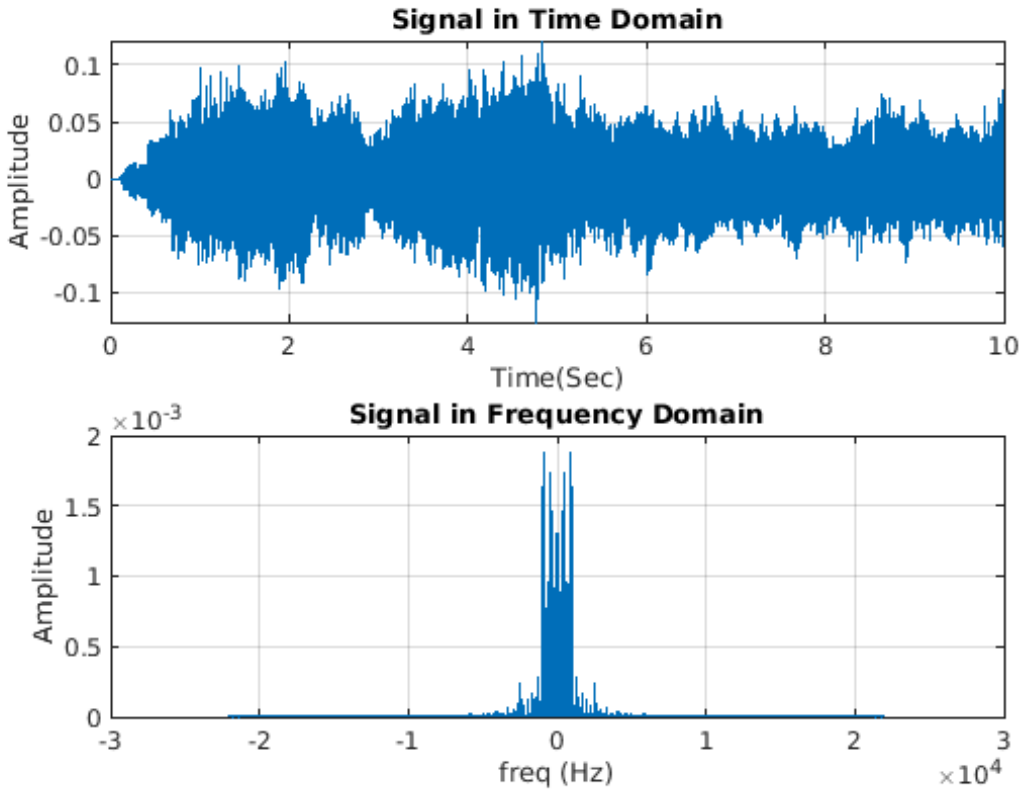
Table of Contents


```
close all;
clear;
clc;
```

2.3.a

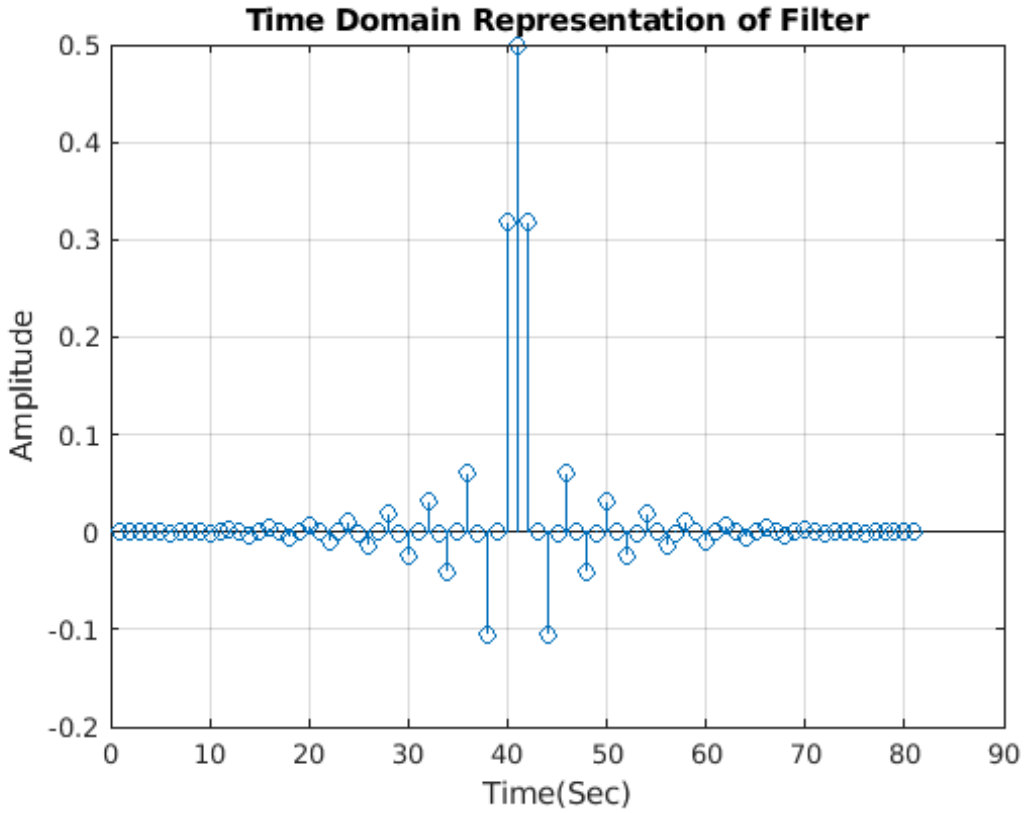
Loading the audio file

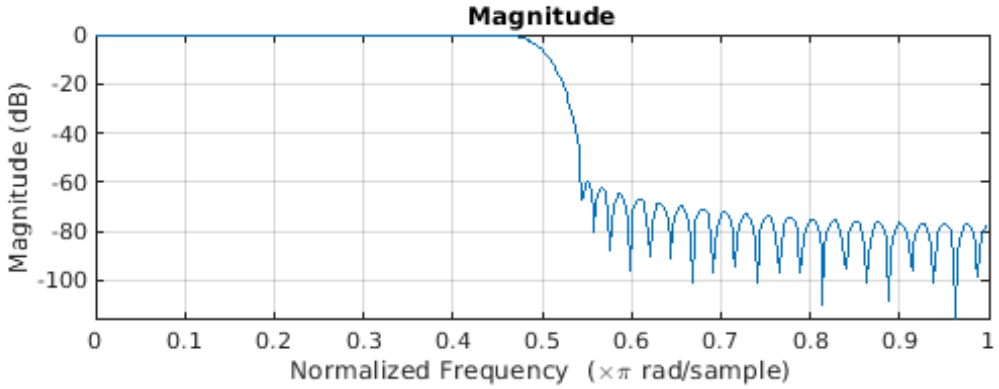
```
[x, Fs] = audioread('Audio01.wav');

% Plotting signal in time and freq domain
plot_time_freq(x, Fs, 'Original audio signal', 'Signal in Time
 Domain', 'Signal in Frequency Domain');

% Playing the audio
sound(x, Fs);
%pause(length(x) * (1/ Fs));
```

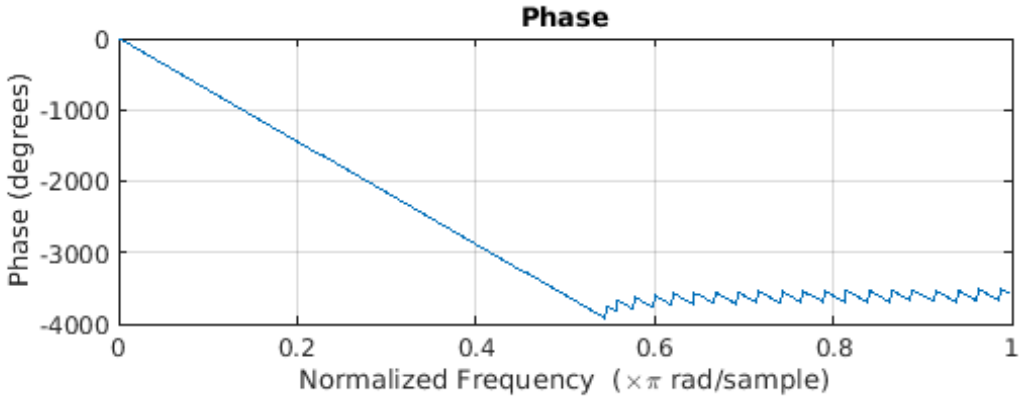


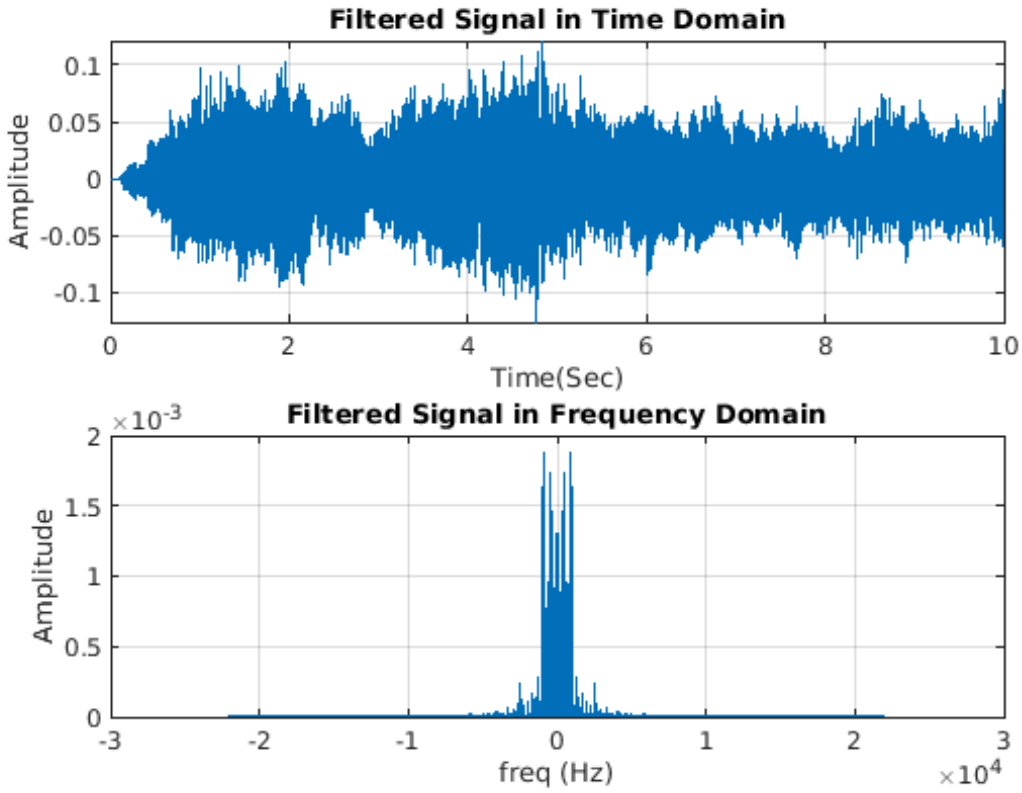
2.3.b

```
load('filter.mat');

% Plotting filter in time domain
figure('Name', 'Time Domain Representation of Filter');
stem(Num, 'LineWidth', 1.5);
grid on;
xlabel('Time(Sec)') ;
ylabel('Amplitude') ;
title('Time Domain Representation of Filter');

figure('Name', 'Filter freq-phase response');
freqz(Num); % Visualising filter freq-phase response

y_0 = filter(Num,1,x); % Applying filter

% Plotting signal in time and freq domain
plot_time_freq(y_0, Fs, 'Filtered audio signal', 'Filtered Signal in Time
 Domain', 'Filtered Signal in Frequency Domain');
```


Time Domain Representation of Filter
Amplitude
Time(Sec)
0.5
0.4
0.3
0.2
0.1
0
-0.1
-0.2
0
10
20
30
40
50
60
70
80
90

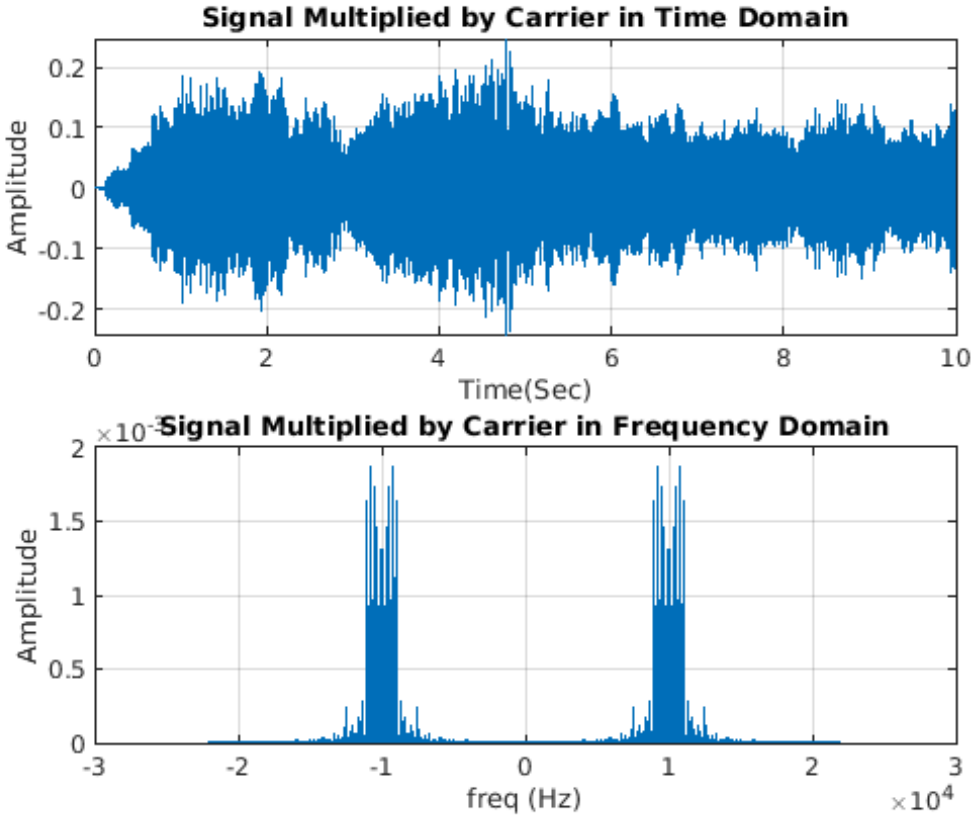

Magnitude
Magnitude (dB)
Normalized Frequency (×π rad/sample)
0
-20
-40
-60
-80
-100
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

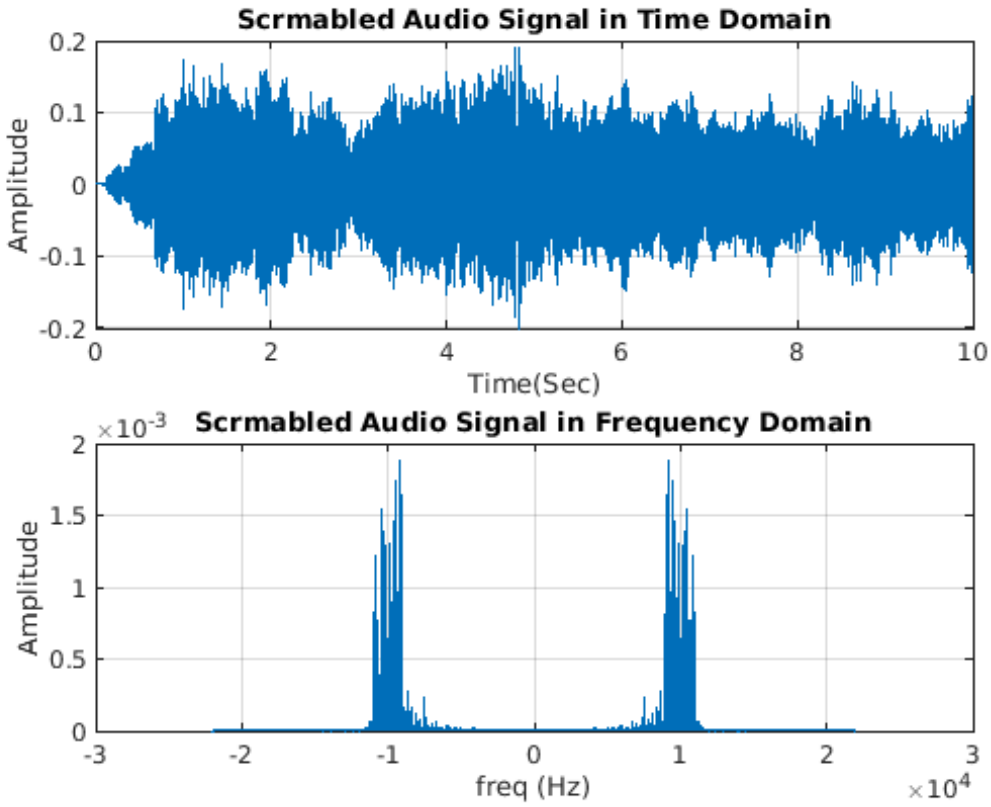

Phase
Phase (degrees)
Normalized Frequency (×π rad/sample)
0
-1000
-2000
-3000
-4000
0
0.2
0.4
0.6
0.8
1



2.3.c

```
f0 = 10000;
n   = 1:length(x);
s_n = 2*cos(2*pi*f0*n/Fs);

y_1 = y_0 .* s_n';
y_2 = filter(Num,1,y_1); % Scrambled Signal

% Plotting signals in time and freq domain
plot_time_freq(y_1, Fs, 'Signal Multiplied by Carrier', 'Signal Multiplied by
 Carrier in Time Domain', 'Signal Multiplied by Carrier in Frequency Domain');
plot_time_freq(y_2, Fs, 'Scrmabled Audio Signal', 'Scrmabled Audio Signal in
 Time Domain', 'Scrmabled Audio Signal in Frequency Domain');

% Playing the audio
sound(y_2, Fs);
%pause(length(y_2) * (1/ Fs));
```


Signal Multiplied by Carrier in Time Domain
Amplitude
Time(Sec)
Signal Multiplied by Carrier in Frequency Domain
Amplitude
freq (Hz)



2.3.d

```
y_3 = filter(Num,1,y_2);
y_4 = y_3 .* s_n';
y_5 = filter(Num,1,y_4);

% Plotting signals in time and freq domain
plot_time_freq(y_4, Fs, 'Scrmabled Signal Multiplied by Carrier', 'Scrmabled
 Signal Multiplied by Carrier in Time Domain', 'Scrmabled Signal Multiplied by
 Carrier in Frequency Domain');
plot_time_freq(y_5, Fs, 'Descrmabled Audio Signal', 'Descrmabled Audio Signal
 in Time Domain', 'Descrmabled Audio Signal in Frequency Domain');

plot_freq_freq(x, y_5, Fs, 'Frequency Spectrums', 'Frequency Spectrum Of
 Original Signal', 'Frequency Spectrum Of Descrambled Signal');

% Playing the audio
sound(y_5, Fs);

% Calculating MSE and MAE
disp('MAE and MSE:');
MAE = mean(abs(x - y_5))
MSE = mean((x - y_5) .^ 2)

%pause(length(y_2) * (1/ Fs));
```

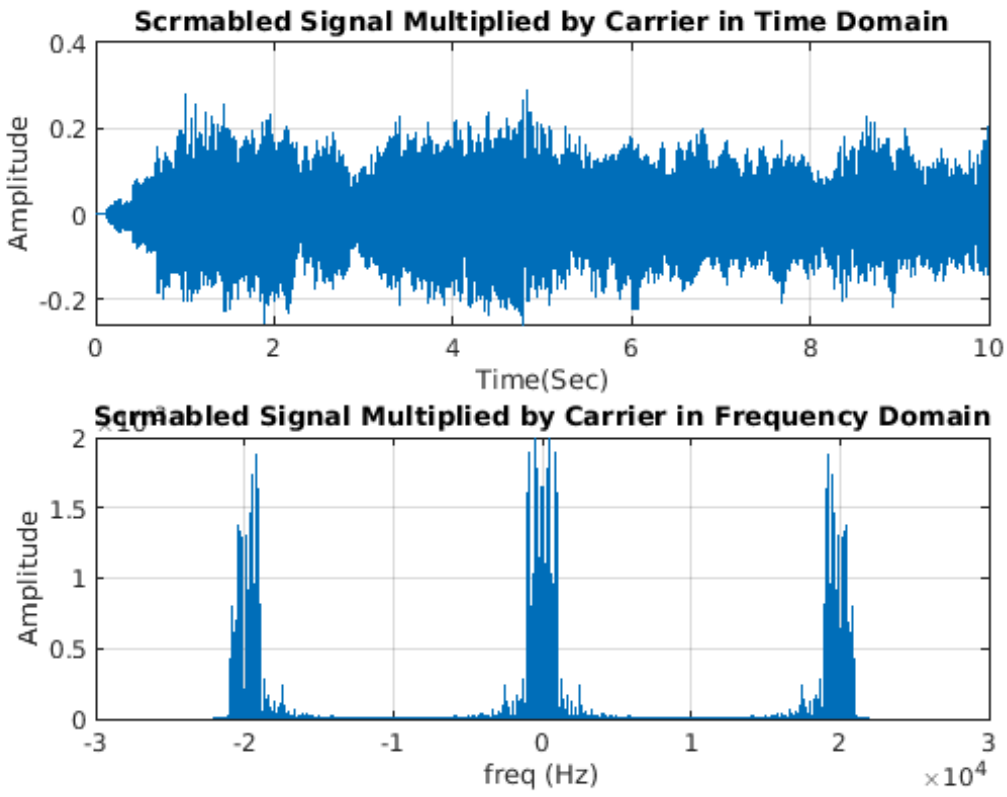
MAE and MSE:

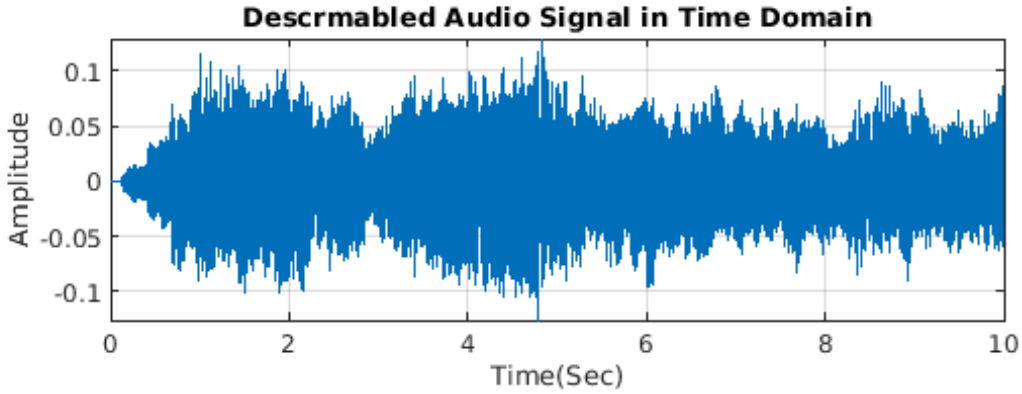
MAE =

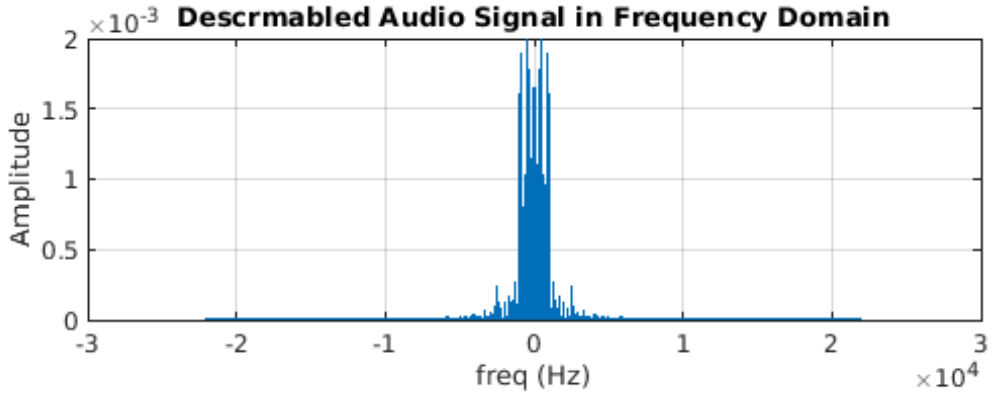
0.0311

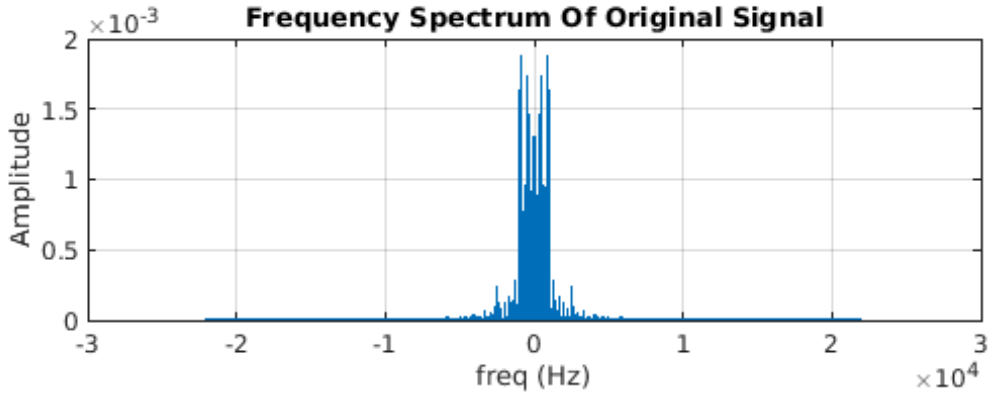
MSE =

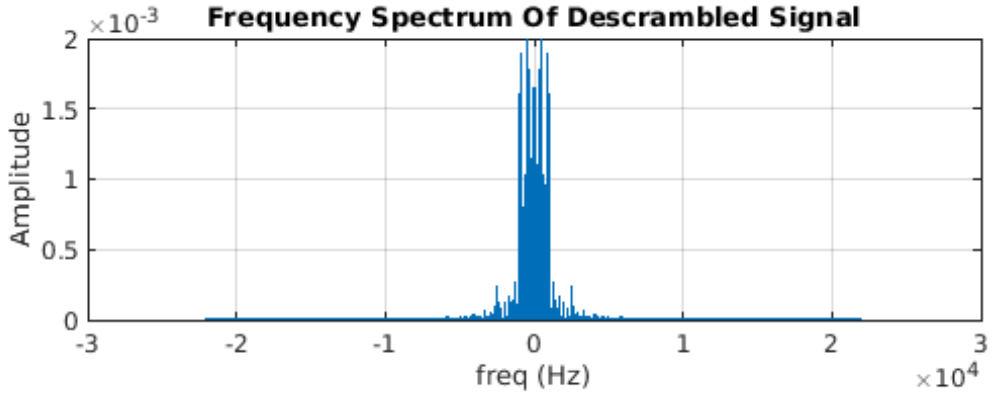
0.0017




Descrmabled Audio Signal in Time Domain
Amplitude
0.1
0.05
0
-0.05
-0.1
0
2
4
6
8
10
Time(Sec)


Descrmabled Audio Signal in Frequency Domain
×10^-3
Amplitude
2
1.5
1
0.5
0
-3
-2
-1
0
1
2
3
freq (Hz)
×10^4


Frequency Spectrum Of Original Signal
×10^-3
Amplitude
2
1.5
1
0.5
0
-3
-2
-1
0
1
2
3
freq (Hz)
×10^4


Frequency Spectrum Of Descrambled Signal
×10^-3
Amplitude
2
1.5
1
0.5
0
-3
-2
-1
0
1
2
3
freq (Hz)
×10^4

2.3 (with noise)

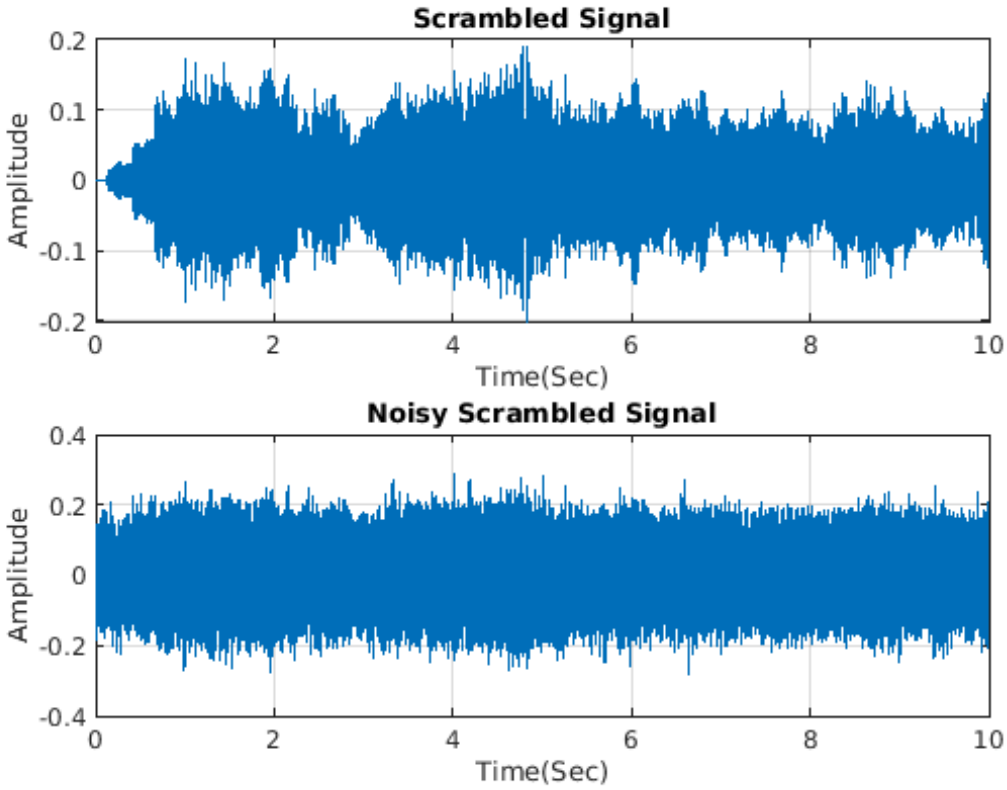
```
y_2_noisy = y_2 + 0.05 * randn(1, length(y_2))';

T = length(y_2) * (1/ Fs);
t = 0: 1/Fs:T-1/Fs;

figure('Name', 'Scrambled Signal VS Noisy Scrambled Signal');
subplot(2, 1, 1);
plot(t, y_2, 'LineWidth', 1.5);
grid on;
xlabel('Time(Sec)') ;
ylabel('Amplitude') ;
title('Scrambled Signal');
subplot(2,1,2) ;
plot(t, y_2_noisy, 'LineWidth',1.5) ;
grid on;
xlabel('Time(Sec)') ;
ylabel('Amplitude');
title('Noisy Scrambled Signal');


y_3 = filter(Num,1,y_2_noisy);
y_4 = y_3 .* s_n';
y_5 = filter(Num,1,y_4);

% Plotting signals in time and freq domain
plot_time_freq(y_3, Fs, 'Low-Passed Scrmabled Noisy Signal', 'Low-Passed
 Scrmabled Noisy Signal in Time Domain', 'Low-Passed Scrmabled Noisy Signal in
 Frequency Domain');
plot_time_freq(y_4, Fs, 'Noisy Scrmabled Signal Multiplied by Carrier', 'Noisy
 Scrmabled Signal Multiplied by Carrier in Time Domain', 'Noisy Scrmabled
 Signal Multiplied by Carrier in Frequency Domain');
plot_time_freq(y_5, Fs, 'Noisy Descrmabled Audio Signal', 'Noisy Descrmabled
 Audio Signal in Time Domain', 'Noisy Descrmabled Audio Signal in Frequency
 Domain');

plot_freq_freq(x, y_5, Fs, 'Frequency Spectrums (Channel with
 noise)', 'Frequency Spectrum Of Original Signal', 'Frequency Spectrum Of
 Descrambled Signal');
```

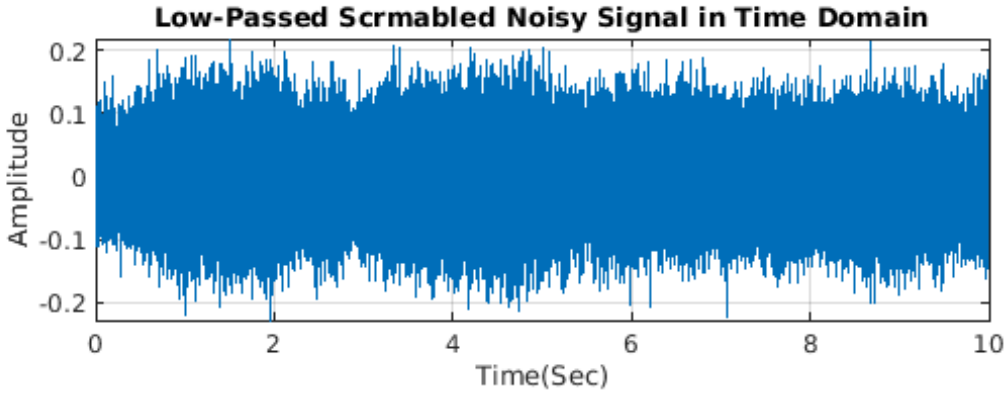
When the communication channel has noise, a noise floor is added to the signal which remains after descrambling and is audible in the final result. The noise floor can also be seen in the spectrum of the descrambled signal.

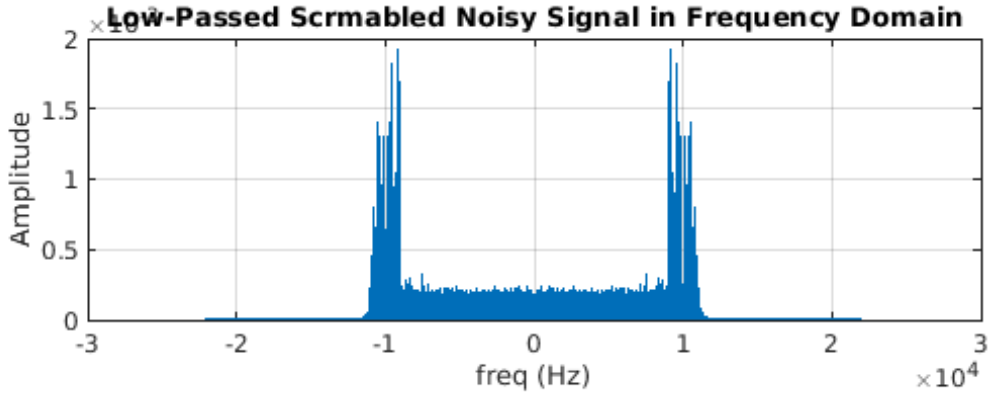
```
% Playing the audio
sound(y_5, Fs);

% Calculating MSE and MAE
disp('MAE and MSE for noisy signal:');
MAE = mean(abs(x - y_5))
MSE = mean((x - y_5) .^ 2)
```

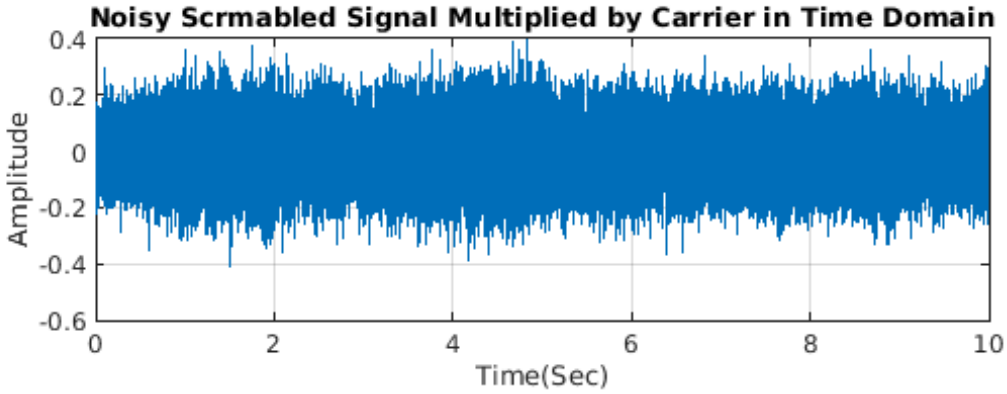
MAE and MSE for noisy signal:

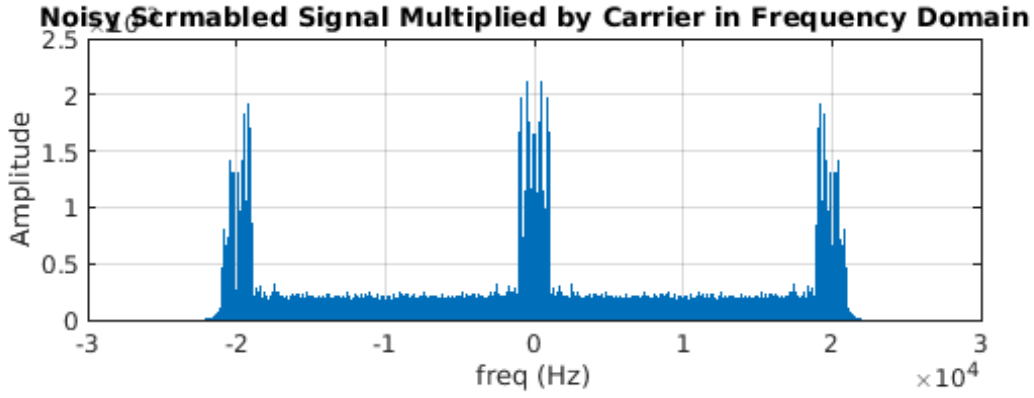
```
MAE =

    0.0429

MSE =

    0.0030
```

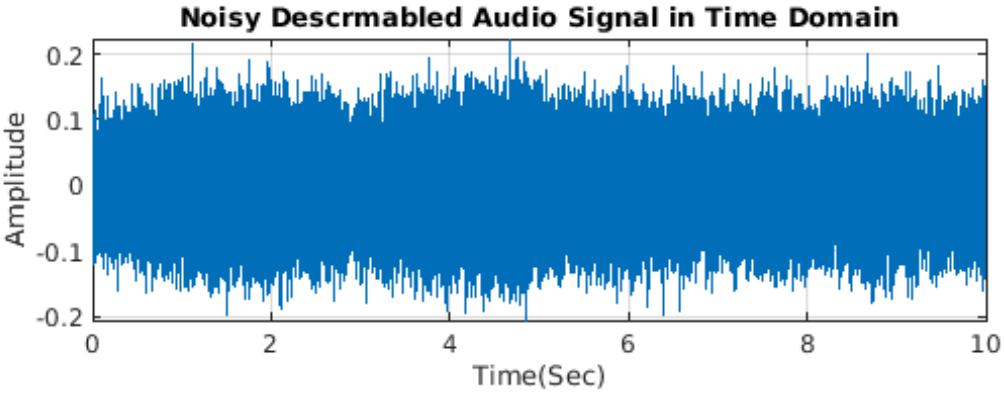


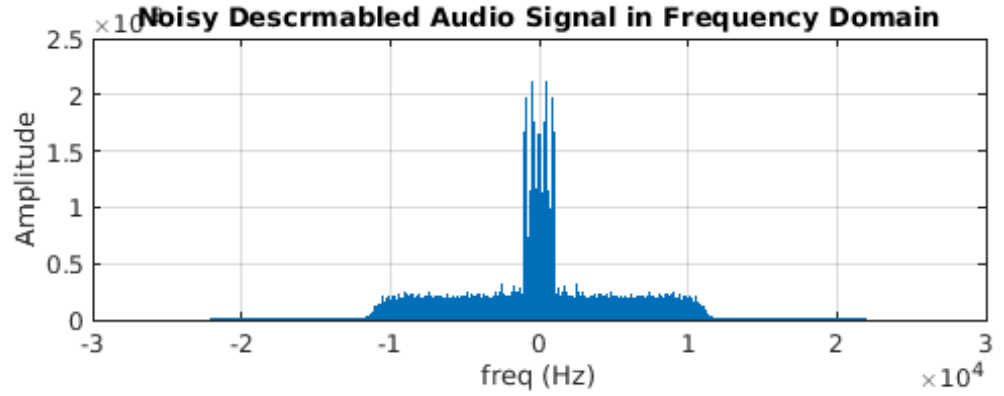

Low-Passed Scrmabled Noisy Signal in Time Domain
Amplitude
Time(Sec)

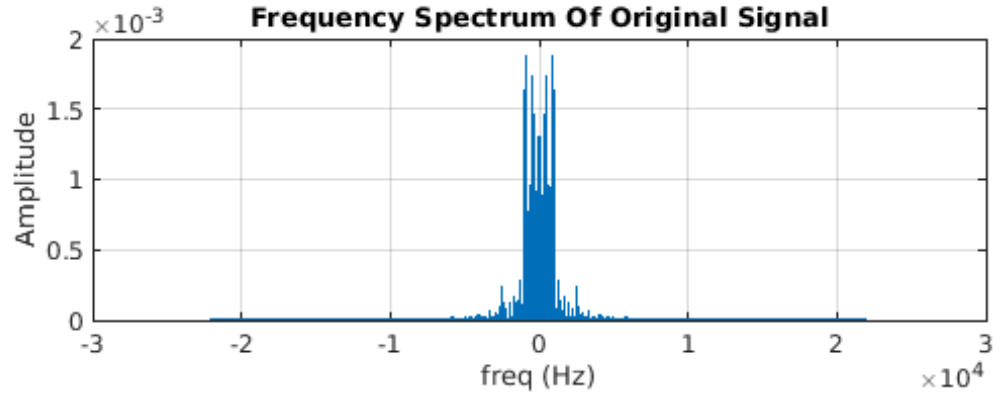

Low-Passed Scrmabled Noisy Signal in Frequency Domain
Amplitude
freq (Hz)
×10^4

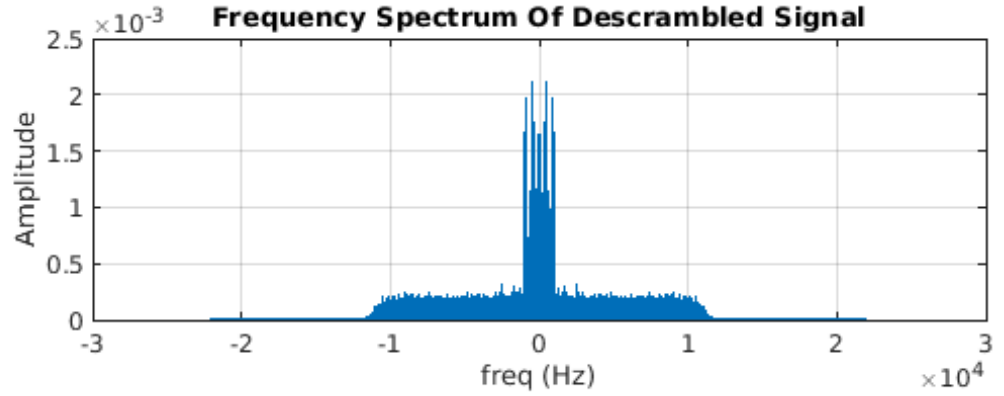

Noisy Scrmabled Signal Multiplied by Carrier in Time Domain
Amplitude
Time(Sec)


Noisy Scrmabled Signal Multiplied by Carrier in Frequency Domain
Amplitude
freq (Hz)
×10^4


Noisy Descrmabled Audio Signal in Time Domain
Amplitude
Time(Sec)

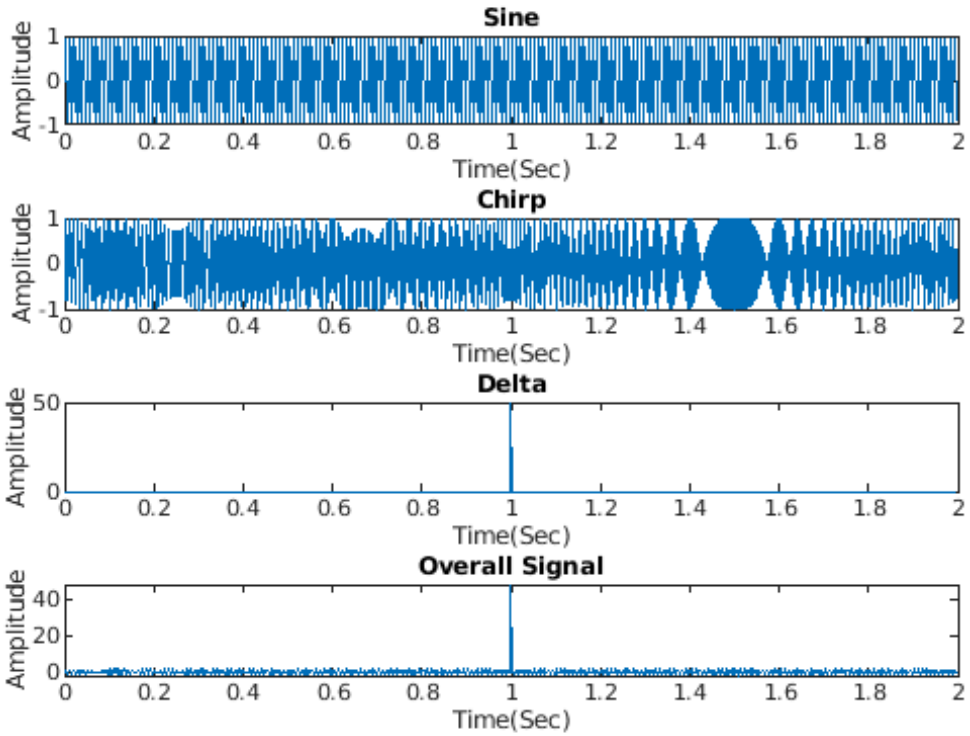

Noisy Descrmabled Audio Signal in Frequency Domain
Amplitude
freq (Hz)

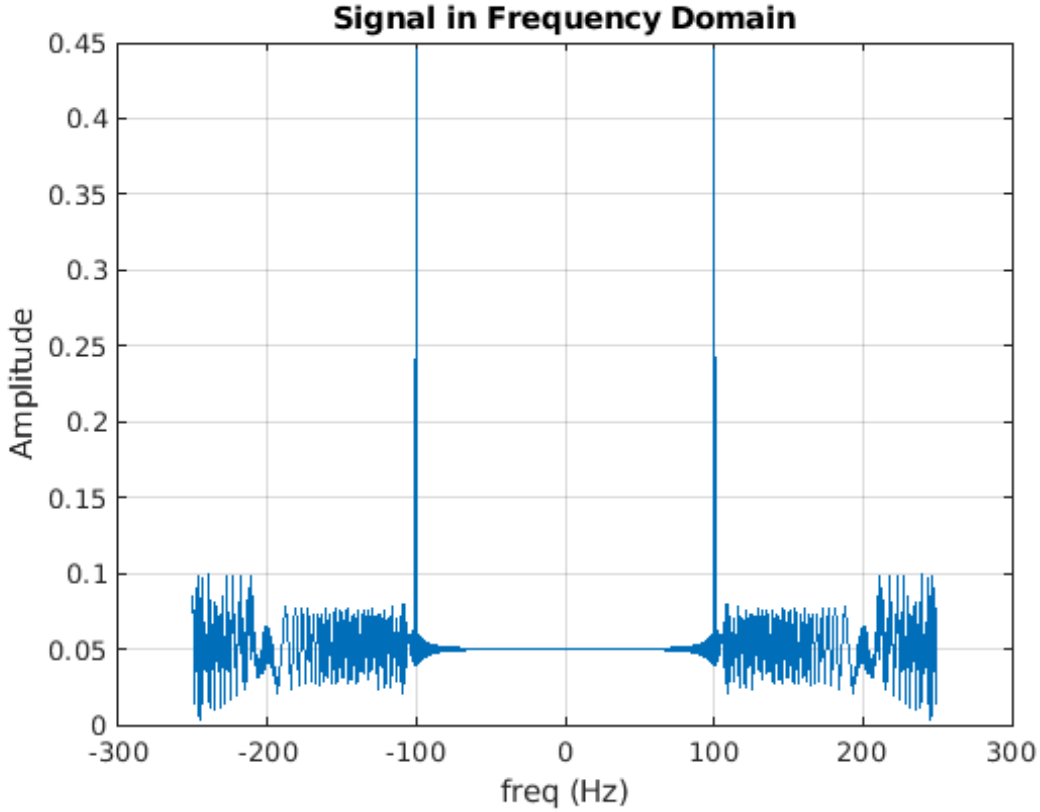

Frequency Spectrum Of Original Signal
Amplitude
freq (Hz)


Frequency Spectrum Of Descrambled Signal
Amplitude
freq (Hz)

2.4.a

```
f0    = 100;
fs    = 500;
t0 = 0;
t1 = 2;
t     = t0 : 1/fs : t1 - 1/fs;

s1   = sin(2 * pi * f0 * t);
f1    = 400;
f2    = 200;
s2   = chirp(t,f1,t1,f2);
s3   = zeros(1,length(s1));
s3(500) = 50;
s     = s1 + s2 + s3;

figure('Name', 'Signals');
subplot(4,1,1);
plot(t, s1);
xlabel('Time(Sec)') ;
ylabel('Amplitude');
title('Sine');
subplot(4,1,2);
plot(t, s2);
xlabel('Time(Sec)') ;
ylabel('Amplitude');
title('Chirp');
subplot(4,1,3);
plot(t, s3);
xlabel('Time(Sec)') ;
ylabel('Amplitude');
title('Delta');
subplot(4,1,4);
plot(t, s);
xlabel('Time(Sec)') ;
ylabel('Amplitude');
title('Overall Signal');


L_x       = length(s);
f_x       = (fs/L_x) * (-L_x/2:L_x/2-1);
fft_x     = fftshift(fft(s))/L_x;

figure();
plot(f_x, abs(fft_x), 'LineWidth',1.5) ;
grid on;
xlabel('freq (Hz)');
ylabel('Amplitude');
title('Signal in Frequency Domain');
```


Sine
Amplitude
Time(Sec)
Chirp
Amplitude
Time(Sec)
Delta
Amplitude
Time(Sec)
Overall Signal
Amplitude
Time(Sec)


Signal in Frequency Domain
Amplitude
freq (Hz)

2.4.b

```
for window_size=[64, 128, 256, 512]
    figure('Name', 'Spectrogram of a Chirp Signal');
    spectrogram( ...
        s, ...
        hamming(window_size), ...
        window_size - 1, ...
        window_size, ...
        fs, ...
        "centered", ...
        "yaxis" ...
    );
    title(sprintf('Spectrogram of a Chirp Signal (window size=%d)',
 window_size));
end

figure('Name', 'Spectrogram of a Chirp Signal in 3D');
spectrogram( ...
    s, ...
    hamming(128), ...
    127, ...
    128, ...
    fs, ...
    "centered", ...
    "yaxis" ...
);
view(-45, 50);
colormap bone;
title('Spectrogram of Signal (window size=128)');
```

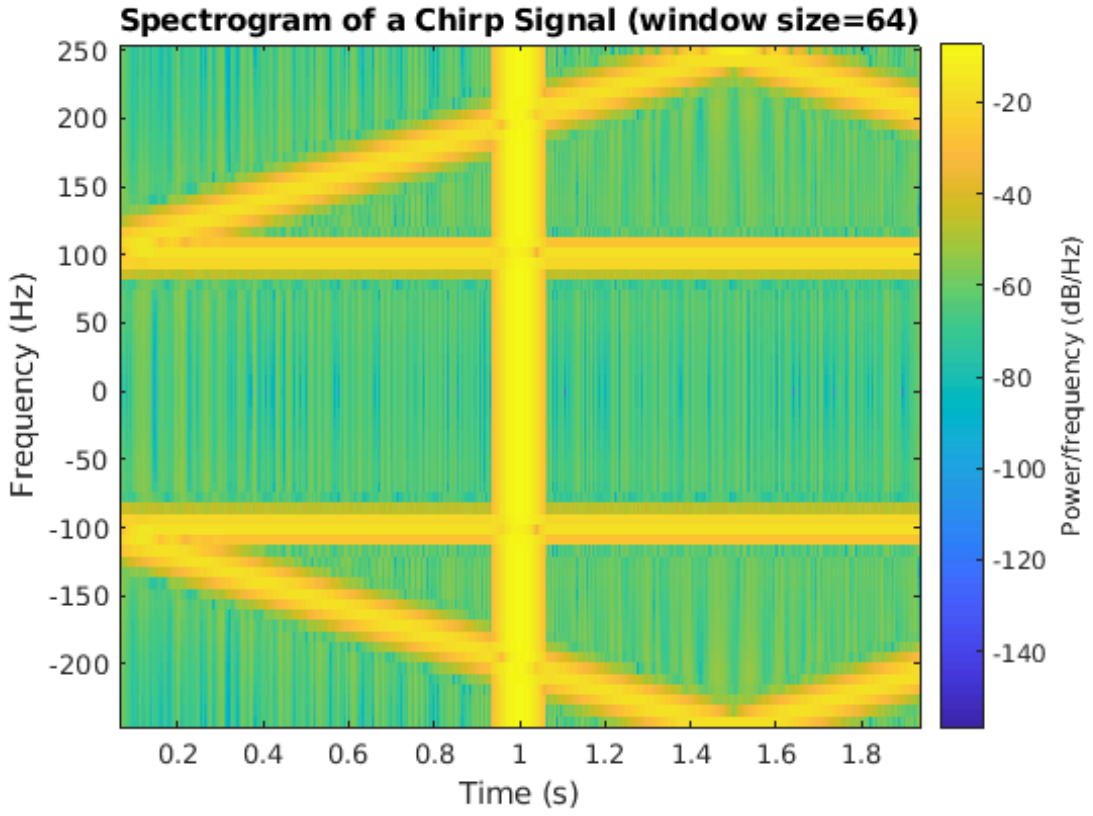
STFT analyzes small chunks of a longer signal, while DFT looks at the entire signal. This allows STFT to provide time-localized frequency information, while DFT only provides overall frequency content. STFT uses a sliding window function to isolate segments of the signal for analysis. DFT does not use a window, it processes the whole signal. STFT provides a 2D representation (time vs. frequency) of the signal. DFT provides only 1D frequency information.

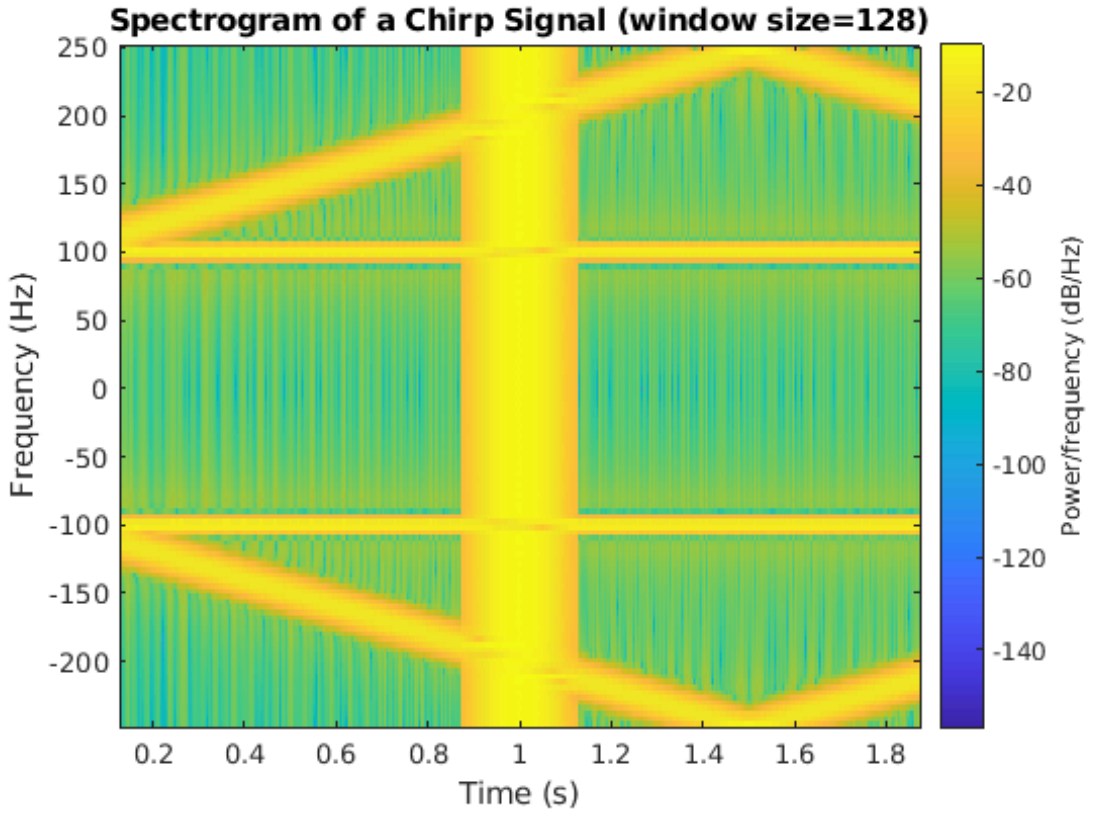
The length of the sliding window in STFT affects the time and frequency resolution:

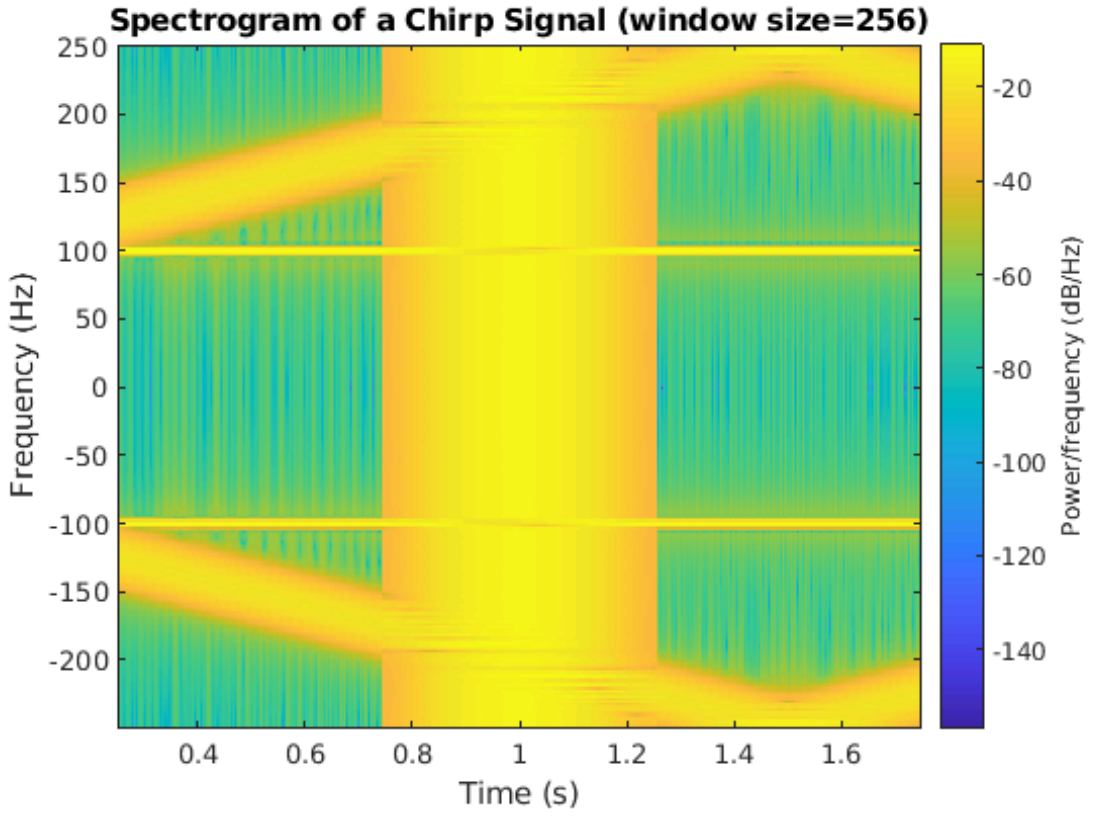
Shorter window results in better time resolution, worse frequency resolution. The window can isolate shorter segments of the signal, but with less cycles/frequency content.

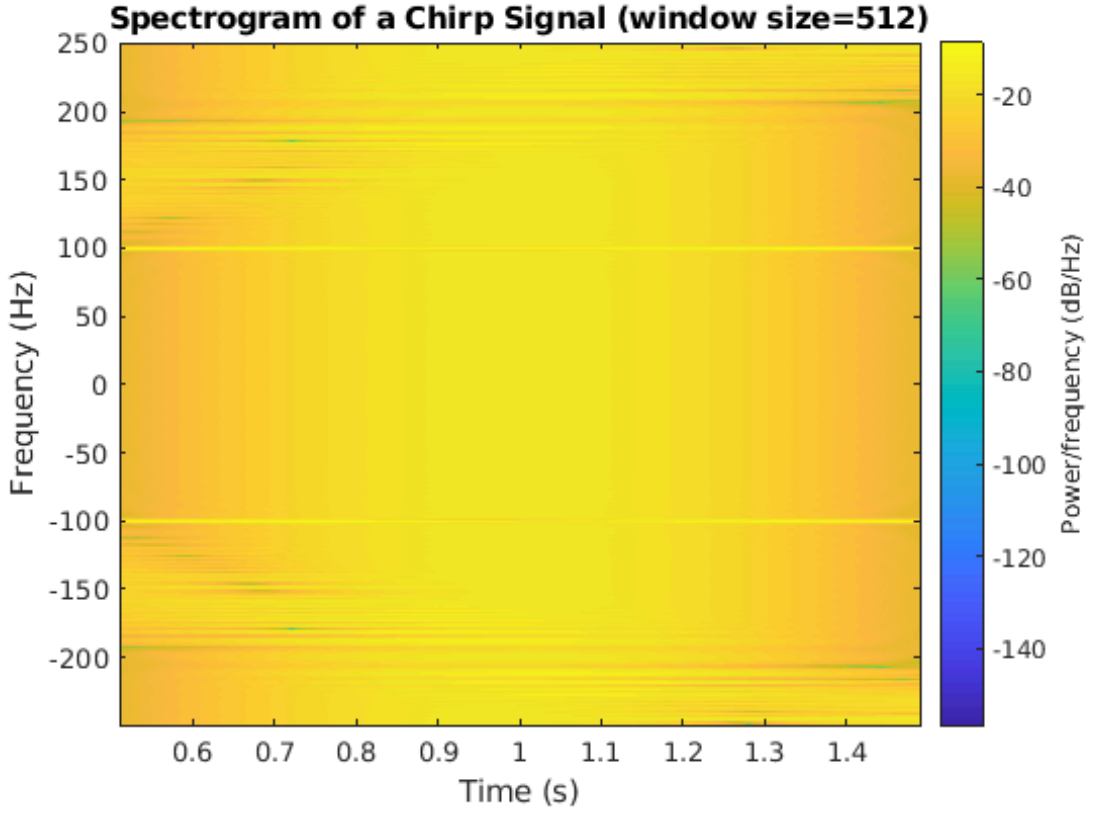
Longer window results in worse time resolution, better frequency resolution. Longer segments of the signal contain more cycles and provide better frequency information, but precise time localization is lost.

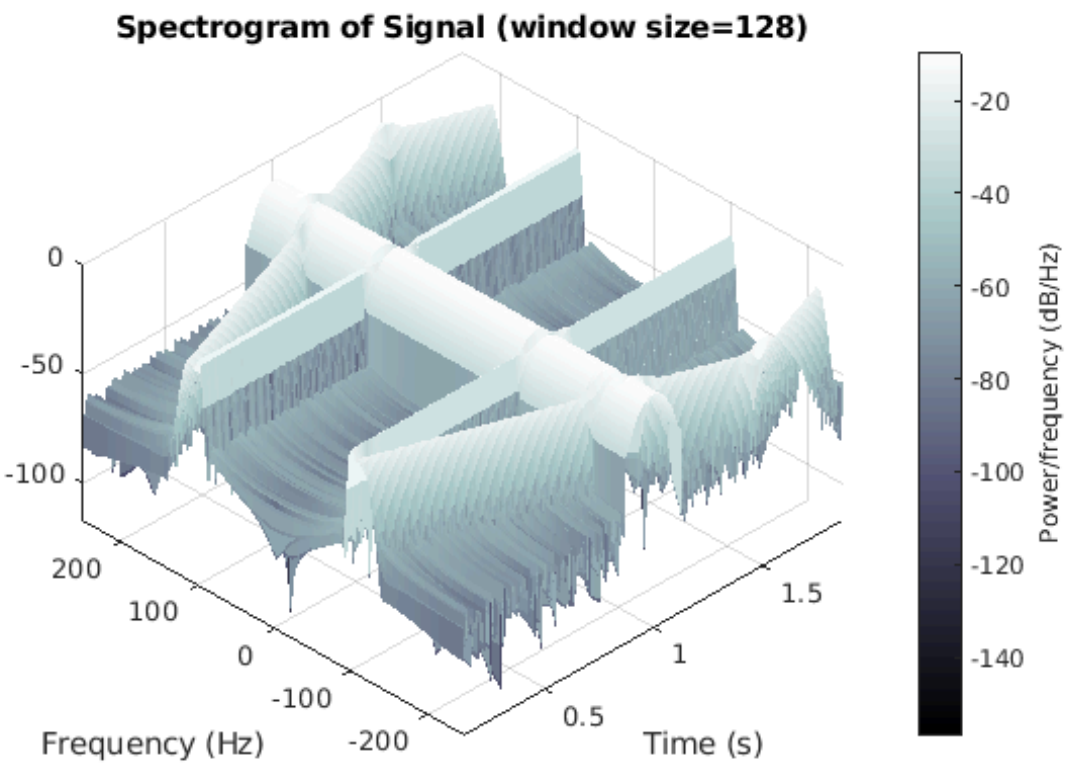
So there is a tradeoff between time and frequency precision based on STFT window length.


Spectrogram of a Chirp Signal (window size=64)
Frequency (Hz)
250
200
150
100
50
0
-50
-100
-150
-200
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
Time (s)
-20
-40
-60
-80
-100
-120
-140
Power/frequency (dB/Hz)


Spectrogram of a Chirp Signal (window size=128)
Frequency (Hz)
250
200
150
100
50
0
-50
-100
-150
-200
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
Time (s)
-20
-40
-60
-80
-100
-120
-140
Power/frequency (dB/Hz)


Spectrogram of a Chirp Signal (window size=256)
Frequency (Hz)
250
200
150
100
50
0
-50
-100
-150
-200
0.4
0.6
0.8
1
1.2
1.4
1.6
Time (s)
-20
-40
-60
-80
-100
-120
-140
Power/frequency (dB/Hz)


Spectrogram of a Chirp Signal (window size=512)
Frequency (Hz)
250
200
150
100
50
0
-50
-100
-150
-200
0.6
0.7
0.8
0.9
1
1.1
1.2
1.3
1.4
Time (s)
-20
-40
-60
-80
-100
-120
-140
Power/frequency (dB/Hz)



Plotting functions

```
function plot_time_freq(y, Fs, title1, title2, title3)
    T = length(y) * (1/ Fs);
    t = 0: 1/Fs:T-1/Fs;

    % Constructing the FFT spectrum of the signal
    L_y = length(y);
    f_y = (Fs/L_y) * (-L_y/2:L_y/2-1);
    fft_y = fftshift(fft(y))/L_y;

    % Plotting signal in time and freq domain
    figure('Name', title1);
    subplot(2, 1, 1);
    plot(t, y, 'LineWidth', 1.5);
    grid on;
    xlabel('Time(Sec)') ;
    ylabel('Amplitude') ;
    title(title2);
    subplot(2,1,2) ;
    plot(f_y, abs(fft_y), 'LineWidth',1.5) ;
    grid on;
    xlabel('freq (Hz)');
    ylabel('Amplitude');
    title(title3);
end
```

```
function plot_freq_freq(y1, y2, Fs, title1, title2, title3)
    % Constructing the FFT spectrum of the signals
    L_y1 = length(y1);
    f_y1 = (Fs/L_y1) * (-L_y1/2:L_y1/2-1);
    fft_y1 = fftshift(fft(y1))/L_y1;

    L_y2 = length(y2);
    f_y2 = (Fs/L_y2) * (-L_y2/2:L_y2/2-1);
    fft_y2 = fftshift(fft(y2))/L_y2;

    % Plotting signal in time and freq domain
    figure('Name', title1);
    subplot(2, 1, 1);
    plot(f_y1, abs(fft_y1), 'LineWidth', 1.5);
    grid on;
    xlabel('freq (Hz)') ;
    ylabel('Amplitude') ;
    title(title2);
    subplot(2,1,2) ;
    plot(f_y2, abs(fft_y2), 'LineWidth',1.5) ;
    grid on;
    xlabel('freq (Hz)');
    ylabel('Amplitude');
    title(title3);
end
```

Published with MATLAB® R2023a