Homework2_2

Programmers

Mohammad Mahdi Elyasi - 9823007

Moein Nasiri - 9823093

Clear Workspace

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

Homework_1

- Main purpose of this task is scrambling voice and reconstruct it
- Now we import audio and plot it

```
[x_t, fs] = audioread('Audio01.wav');
x_t = x_t';
t_axis = linspace(0, length(x_t) / fs, length(x_t));
f0 = 10000;
```

Now we plot the audio

```
figure('Name', 'Audio');
plot(t_axis, x_t);
xlabel('Samples');
ylabel('Amplitude');
title('time domain of Audio');
grid on;
```

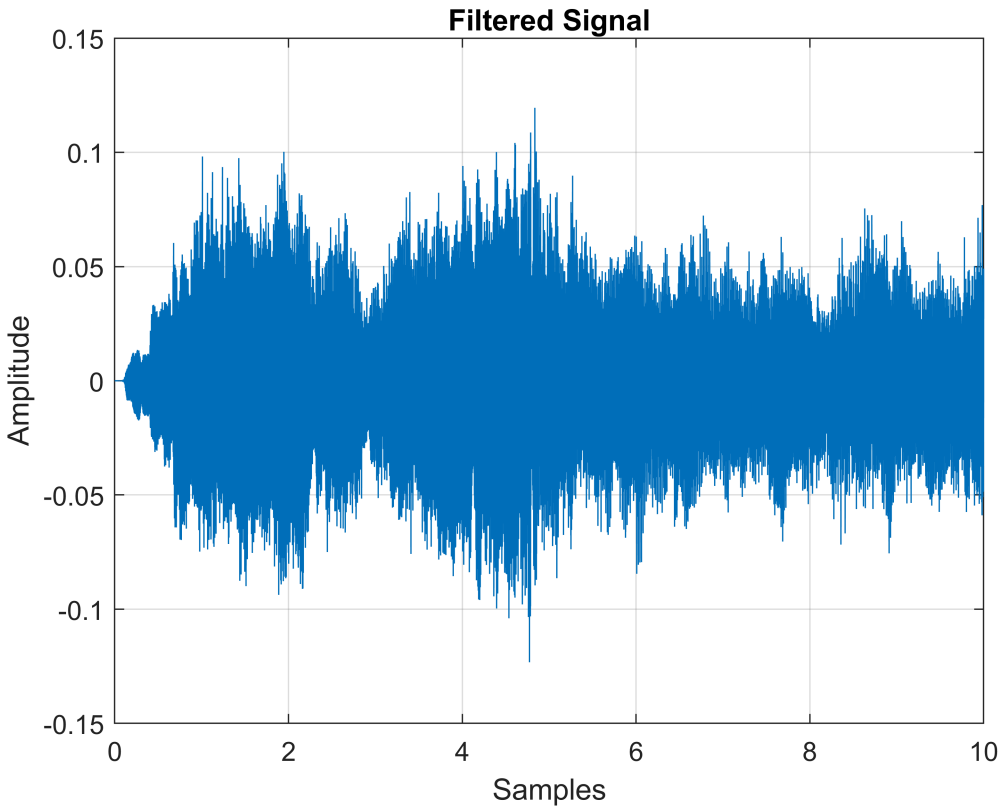
```
figure('Name', 'Audio');
plot(fftshift(abs(fft(x_t))));
xlabel('Samples');
ylabel('Amplitude');
title('fft of Audio');
grid on;
```

Here we hear the sound of audio

```
sound(x_t, fs);
pause(length(x_t) / fs);
```

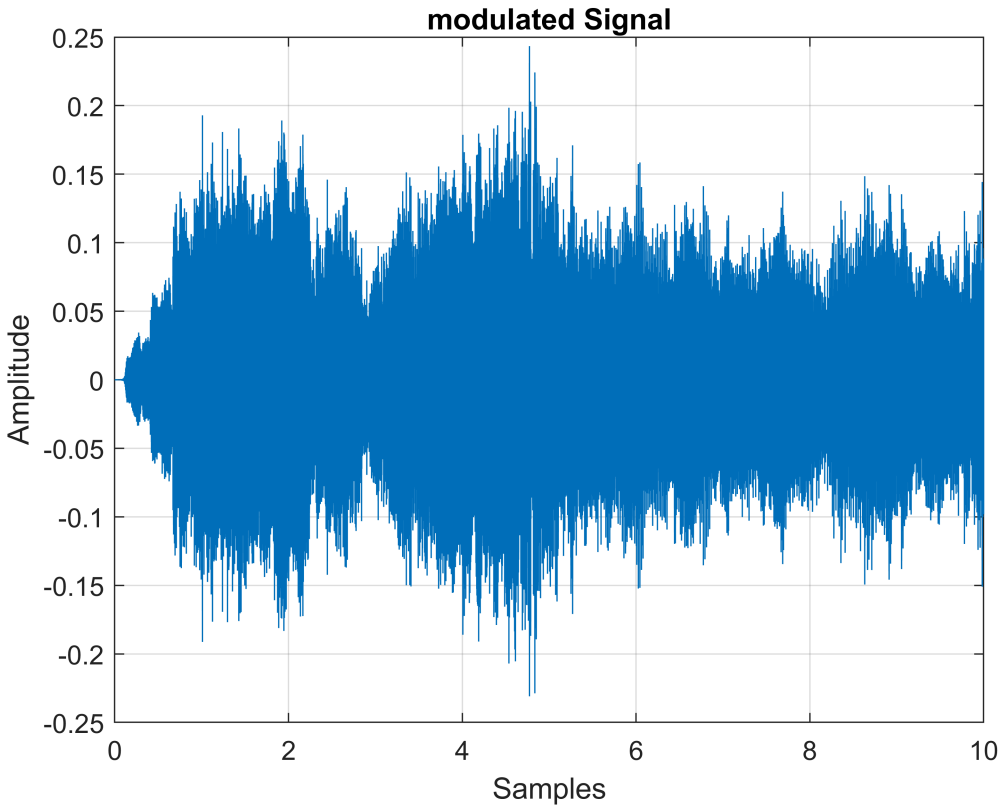
Now we declare needed variables and signals and import the constructed filter by filterDesigner tool

```
s = 2 * cos((2 * pi * f0) * t_axis);
FD_FIR = filter_FIR;
y = FD_FIR.filter(x_t);
figure('Name', 'Signal');
plot(t_axis, y);
xlabel('Samples');
ylabel('Amplitude');
title('Filtered Signal');
grid on;
```



Now we modulate the signal and plot it

```
y1 = y .* s;

figure('Name', 'Signal');
plot(t_axis, y1);
xlabel('Samples');
ylabel('Amplitude');
title('modulated Signal');
grid on;
```



Here we import filter and change a little in F_axis for showing filter better

```
y2 = FD_FIR.filter(y1);

imp_filter = load('filter.mat').Num;
reso_freq = 10000;
f_axis = linspace(-fs / 2, fs / 2, reso_freq);
f_axis_pos = f_axis(floor(reso_freq / 2) + 1:end);
FT_lowpass_filter = fftshift(fft(imp_filter, reso_freq));
FT_lowpass_filter = FT_lowpass_filter(floor(reso_freq / 2) + 1:end);

x2 = y2 .* s;
x4 = filter(imp_filter, 1, x2);
```

- y is filtered signal of input
- y1 is product of filtered signal and carrier
- y2 is filtered signal of y1
- x4 is reconstructed signal

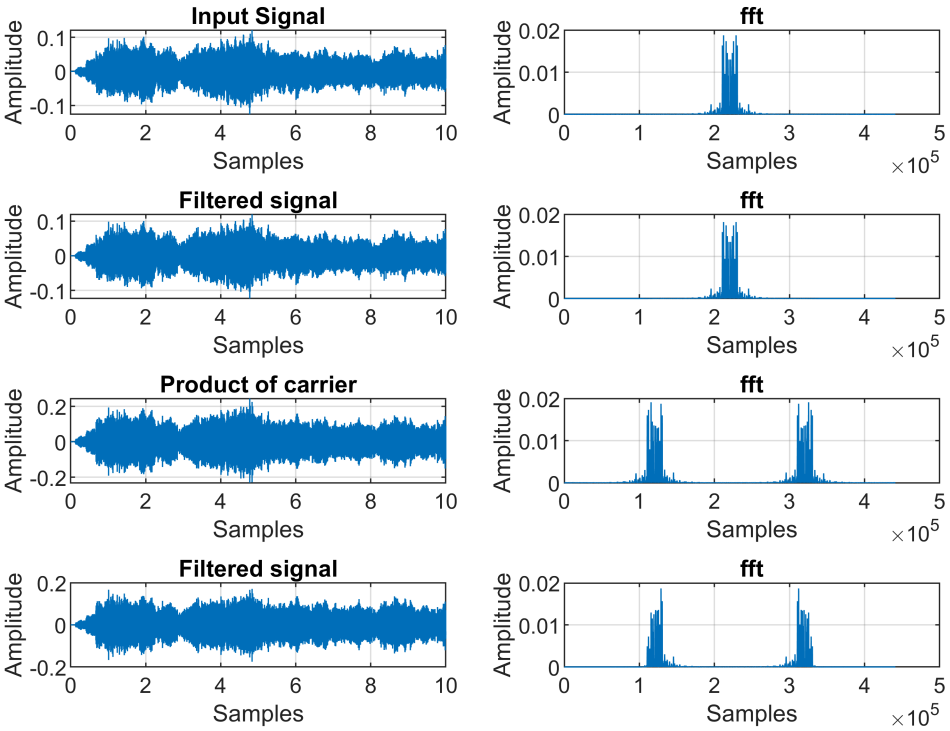
Here we hear the sound of modulated signal

```
sound(y2, fs);
pause(length(y2) / fs);
```

Now we plot signals without noise

```
figure('Name', 'Signals');
subplot(4, 2, 1)
plot(t_axis, x_t);
xlabel('Samples');
ylabel('Amplitude');
title('Input Signal');
grid on;

subplot(4, 2, 2)
plot(fftshift(abs(fft(x_t))) / fs);
xlabel('Samples');
ylabel('Amplitude');
grid on;
title('fft');

subplot(4, 2, 3)
plot(t_axis, y);
xlabel('Samples');
ylabel('Amplitude');
title('Filtered signal');
grid on;

subplot(4, 2, 4)
plot(fftshift(abs(fft(y))) / fs);
xlabel('Samples');
ylabel('Amplitude');
grid on;
title('fft');

subplot(4, 2, 5)
plot(t_axis, y1);
xlabel('Samples');
ylabel('Amplitude');
title('Product of carrier');
grid on;

subplot(4, 2, 6)
plot(fftshift(abs(fft(y1))) / fs);
xlabel('Samples');
ylabel('Amplitude');
grid on;
title('fft');

subplot(4, 2, 7)
plot(t_axis, y2);
xlabel('Samples');
ylabel('Amplitude');
title('Filtered signal');
grid on;

subplot(4, 2, 8)
plot(fftshift(abs(fft(y2))) / fs);
xlabel('Samples');
```

```
ylabel('Amplitude');
grid on;
title('fft');
```



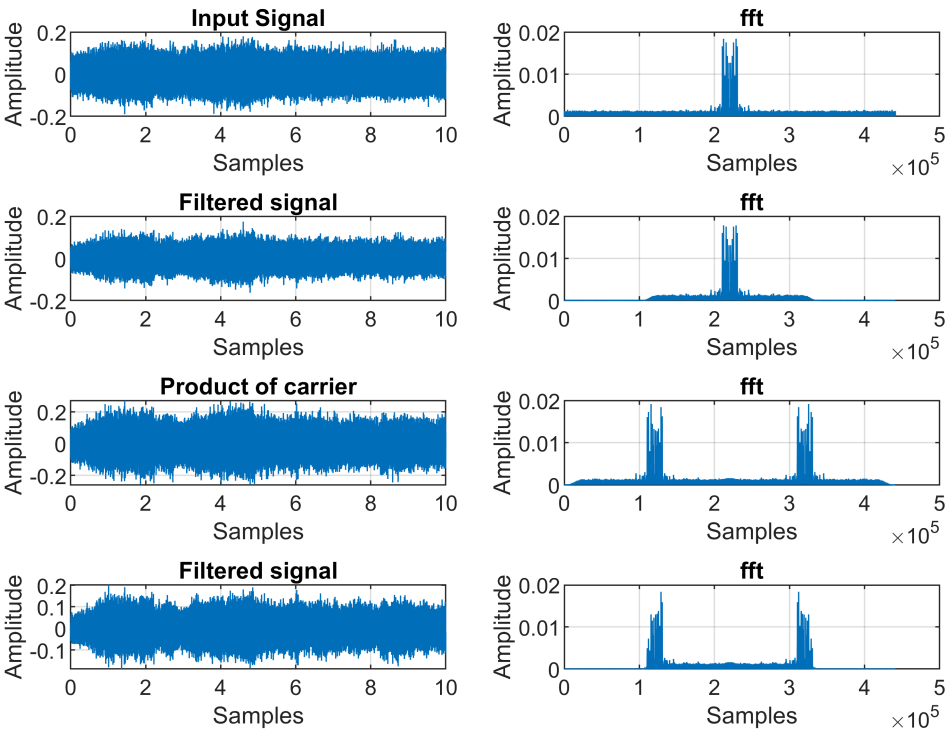
Here We hear the sound of reconstructed signal

```
sound(x4, fs);
pause(length(x4) / fs);
```

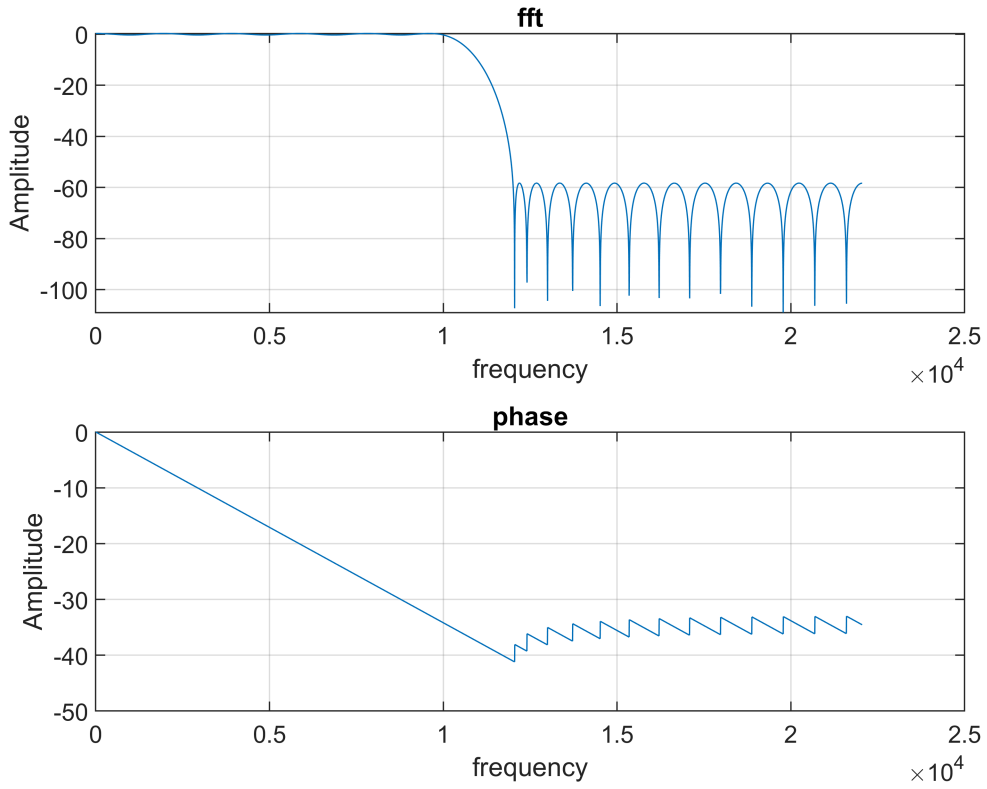
Now we add a gaussian noise to our signal and then plot it

```
Noisy_signal = x_t + 0.03 * randn(1, length(y));
y_noisy = FD_FIR.filter(Noisy_signal);
y1_noisy = y_noisy .* s;
y2_noisy = FD_FIR.filter(y1_noisy);
y2_mo = y2_noisy .* s;
Noisy_rec = filter(imp_filter, 1, y2_mo);

figure('Name', 'Noisy Signals');
subplot(4, 2, 1)
plot(t_axis, Noisy_signal);
xlabel('Samples');
ylabel('Amplitude');
title('Input Signal');
grid on;

subplot(4, 2, 2)
plot(fftshift(abs(fft(Noisy_signal))) / fs);
```

```
xlabel('Samples');
ylabel('Amplitude');
grid on;
title('fft');

subplot(4, 2, 3)
plot(t_axis, y_noisy);
xlabel('Samples');
ylabel('Amplitude');
title('Filtered signal');
grid on;

subplot(4, 2, 4)
plot(fftshift(abs(fft(y_noisy))) / fs);
xlabel('Samples');
ylabel('Amplitude');
grid on;
title('fft');

subplot(4, 2, 5)
plot(t_axis, y1_noisy);
xlabel('Samples');
ylabel('Amplitude');
title('Product of carrier');
grid on;

subplot(4, 2, 6)
plot(fftshift(abs(fft(y1_noisy))) / fs);
xlabel('Samples');
ylabel('Amplitude');
grid on;
title('fft');

subplot(4, 2, 7)
plot(t_axis, y2_noisy);
xlabel('Samples');
ylabel('Amplitude');
title('Filtered signal');
grid on;

subplot(4, 2, 8)
plot(fftshift(abs(fft(y2_noisy))) / fs);
xlabel('Samples');
ylabel('Amplitude');
grid on;
title('fft');
```




```
figure('Name', 'Signals');
subplot(2, 1, 1);
plot(f_axis_pos, 20 * log10(abs(FT_lowpass_filter)));
xlabel('frequency');
ylabel('Amplitude');
title('fft');
grid on;

subplot(2, 1, 2);
plot(f_axis_pos, unwrap(angle(FT_lowpass_filter)));
xlabel('frequency');
ylabel('Amplitude');
grid on;
title('phase');
```



Here We hear the sound of reconstructed noisy signal

```
sound(Noisy_rec, fs);
pause(length(Noisy_rec) / fs);
```

Here we calculate mean square errer and mean absolute error for each noisy and without noise signal

```
MSE = sum((x4 - x_t) .^ 2) / length(x_t);
MAE = sum(abs(x4 - x_t)) / length(x_t);

fprintf('MSE between original signal and descrambled signal: %d', MSE);
```

```
MSE between original signal and descrambled signal: 1.479939e-03
```

```
fprintf('MAE between original signal and descrambled signal: %d', MAE);
```

```
MAE between original signal and descrambled signal: 3.049262e-02
```

```
MSE_n = sum((Noisy_rec - x_t) .^ 2) / length(x_t);
MAE_n = sum(abs(Noisy_rec - x_t)) / length(x_t);

fprintf('MSE between original signal and descrambled noisy signal: %d', MSE);
```

```
MSE between original signal and descrambled noisy signal: 1.479939e-03
```

```
fprintf('MAE between original signal and descrambled noisy signal: %d', MAE);
```

```
MAE between original signal and descrambled noisy signal: 3.049262e-02
```

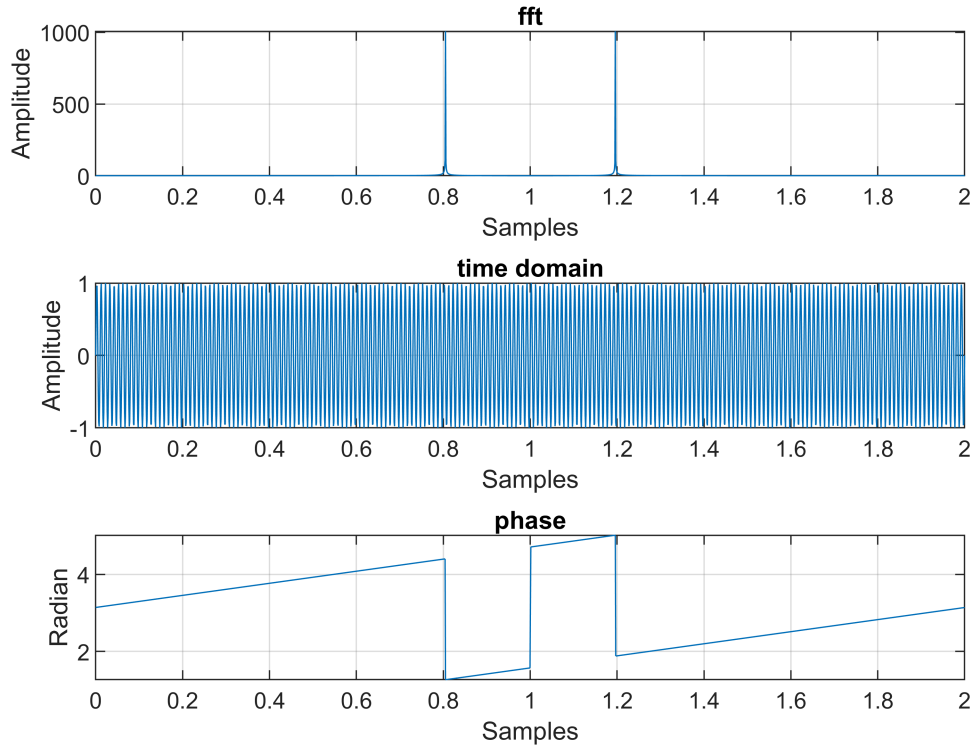
Homework_2

- Here in this task we want to use spectogtram and see chirp
- Now we declare variables and signals

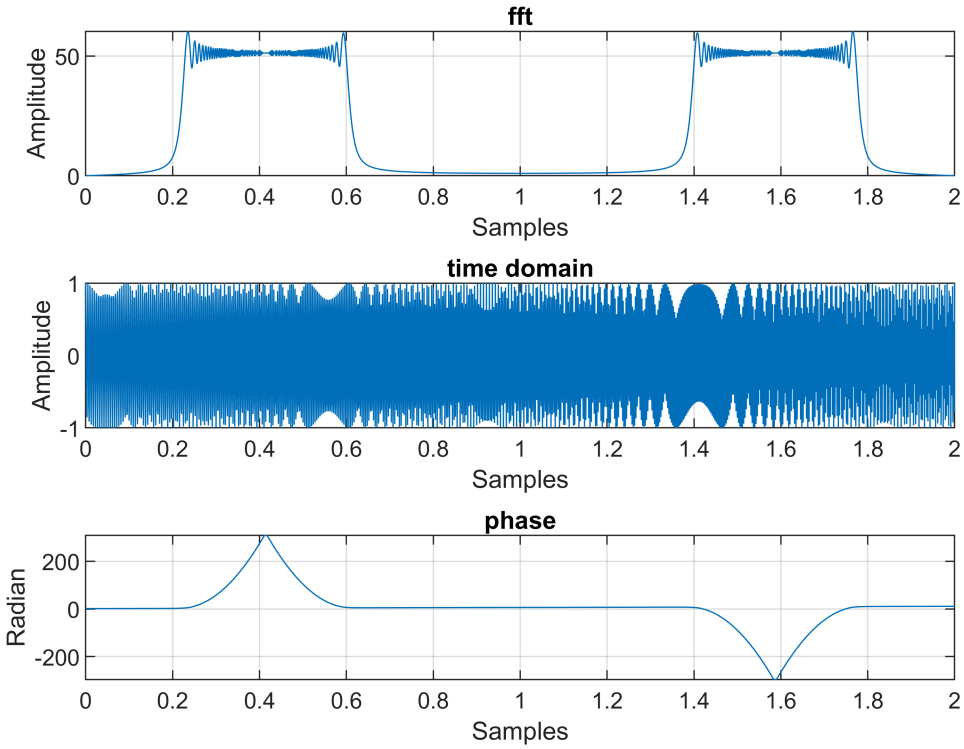
```
f0 = 100;
t = linspace(0, 2, 2 ^ 11);
Ts = t(2) - t(1);
fs = 1 / Ts;
x_1 = sin(2 * pi * f0 * t);
x_2 = chirp(t, 200, 2, 400);
x_3 = 50 * [zeros(1, 1023) 1 zeros(1, 1024)];
x3 = x_1 + x_2 + x_3;
```

Now we plot time domain and frequency domain of each signal

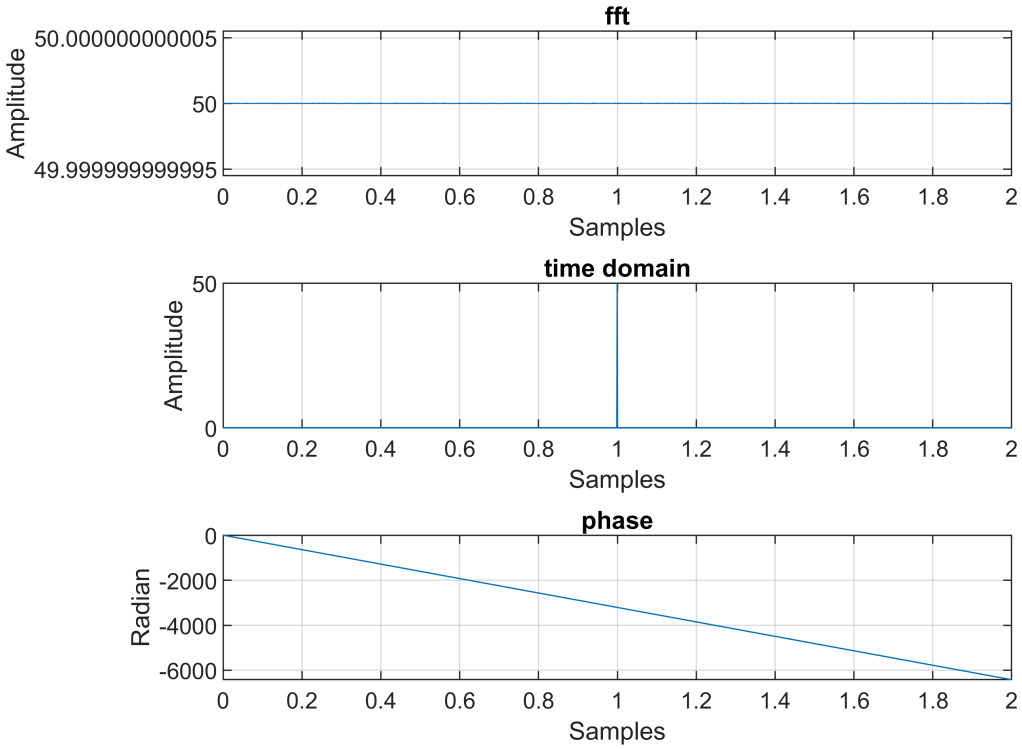
```
figure('Name', 'Signal x_1');
subplot(3, 1, 1);
plot(t, abs(fftshift(fft(x_1))));
xlabel('Samples');
ylabel('Amplitude');
title('fft');
grid on;

subplot(3, 1, 2);
plot(t, x_1);
xlabel('Samples');
ylabel('Amplitude');
grid on;
title('time domain');

subplot(3, 1, 3);
plot(t, unwrap(angle(fftshift(fft(x_1)))));
xlabel('Samples');
ylabel('Radian');
grid on;
title('phase');
```



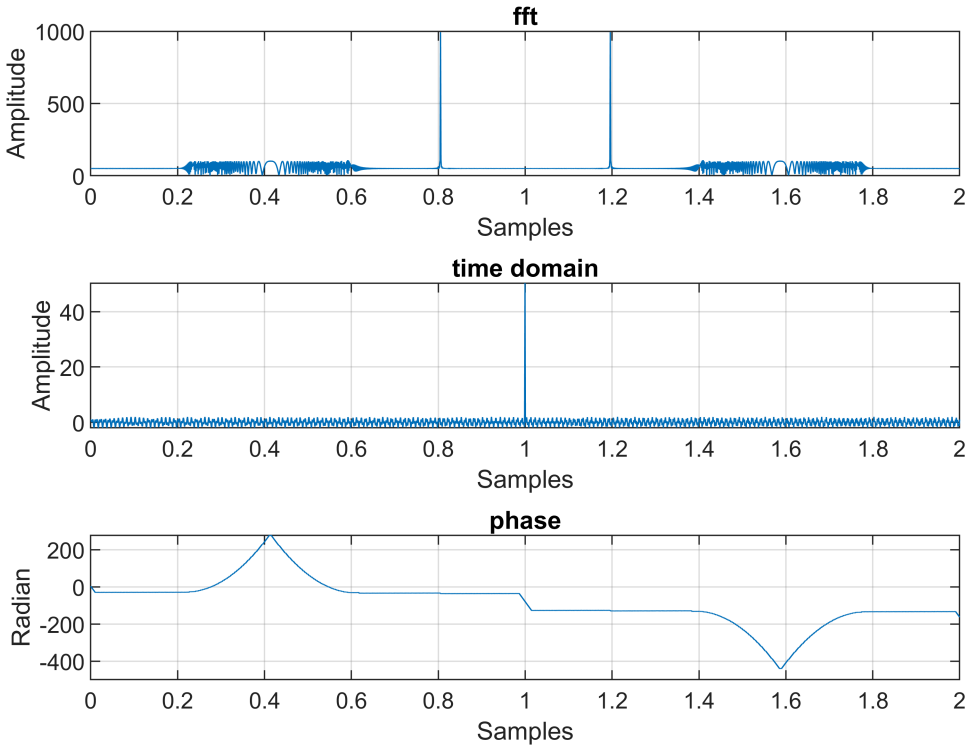

```
figure('Name', 'Signal x_2');
subplot(3, 1, 1);
plot(t, abs(fftshift(fft(x_2))));
xlabel('Samples');
ylabel('Amplitude');
title('fft');
grid on;

subplot(3, 1, 2);
plot(t, x_2);
xlabel('Samples');
ylabel('Amplitude');
grid on;
title('time domain');

subplot(3, 1, 3);
plot(t, unwrap(angle(fftshift(fft(x_2)))));
xlabel('Samples');
ylabel('Radian');
grid on;
title('phase');
```




```
figure('Name', 'Signal x_3');
subplot(3, 1, 1);
plot(t, abs(fftshift(fft(x_3))));
xlabel('Samples');
ylabel('Amplitude');
title('fft');
grid on;

subplot(3, 1, 2);
plot(t, x_3);
xlabel('Samples');
ylabel('Amplitude');
grid on;
title('time domain');

subplot(3, 1, 3);
plot(t, unwrap(angle(fftshift(fft(x_3)))));
xlabel('Samples');
ylabel('Radian');
grid on;
title('phase');
```



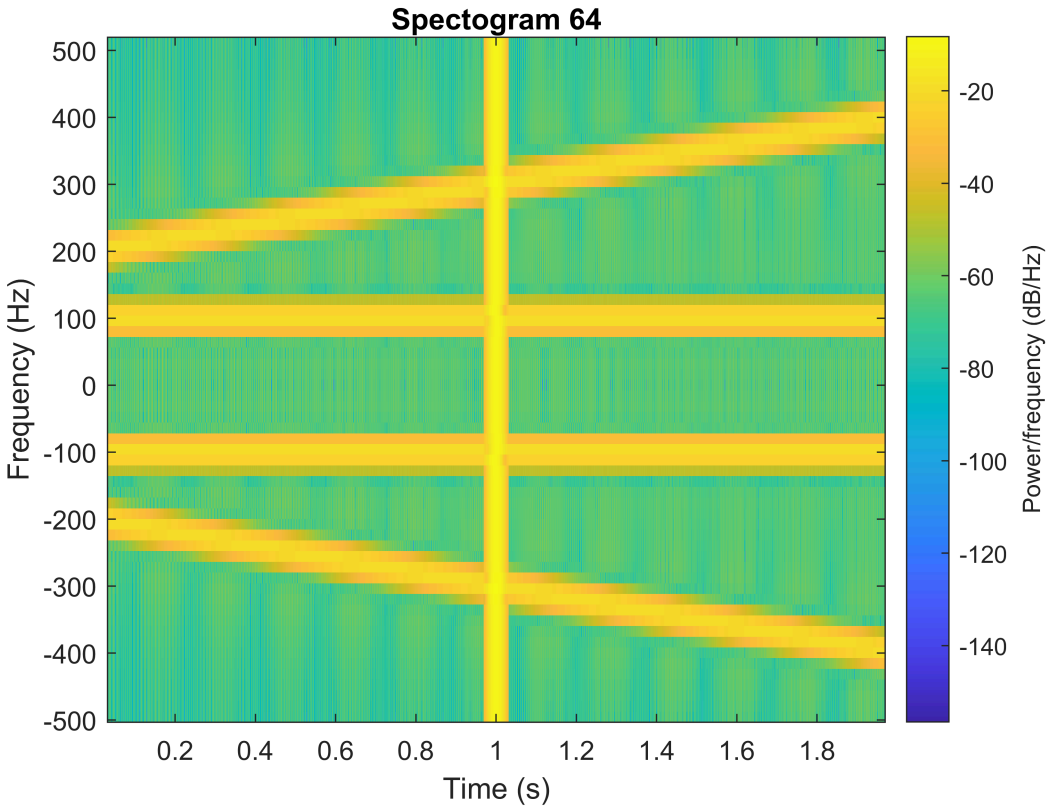

```
figure('Name', 'Signal x3');
subplot(3, 1, 1);
plot(t, abs(fftshift(fft(x3))));
xlabel('Samples');
ylabel('Amplitude');
title('fft');
grid on;

subplot(3, 1, 2);
plot(t, x3);
xlabel('Samples');
ylabel('Amplitude');
grid on;
title('time domain');

subplot(3, 1, 3);
plot(t, unwrap(angle(fftshift(fft(x3)))));
xlabel('Samples');
ylabel('Radian');
grid on;
title('phase');
```



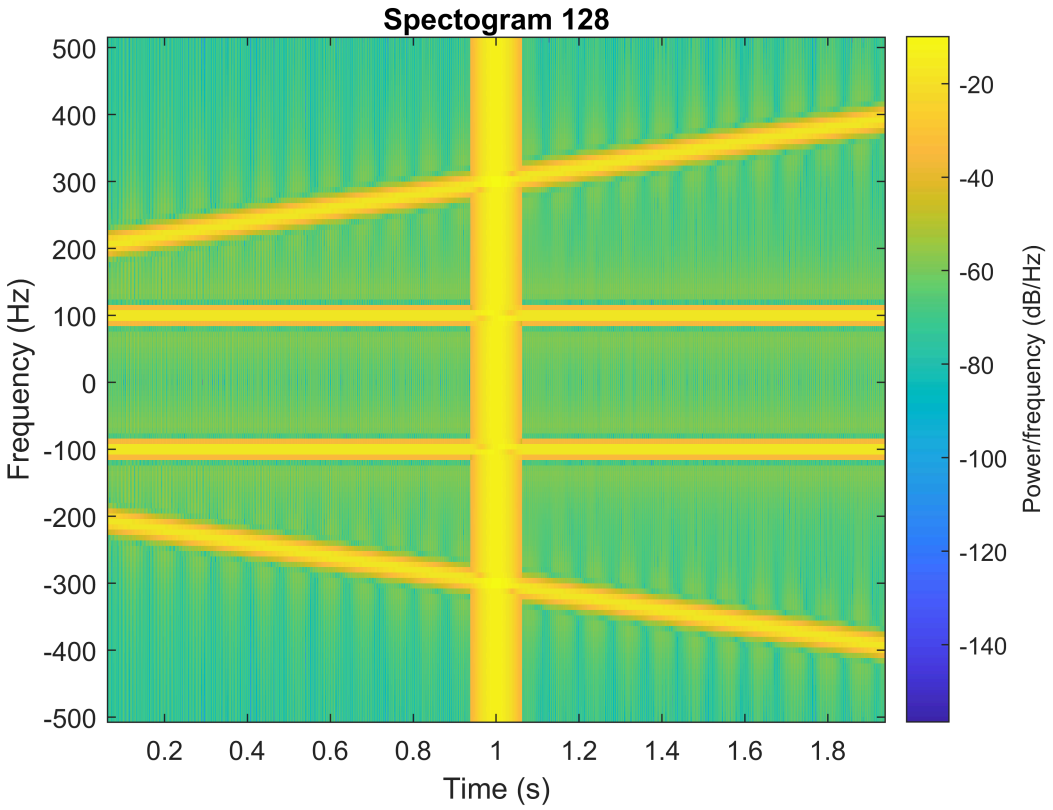
We use spectogram function to show STFT

Now we see what spectrogram of 4 Hamming windows look like

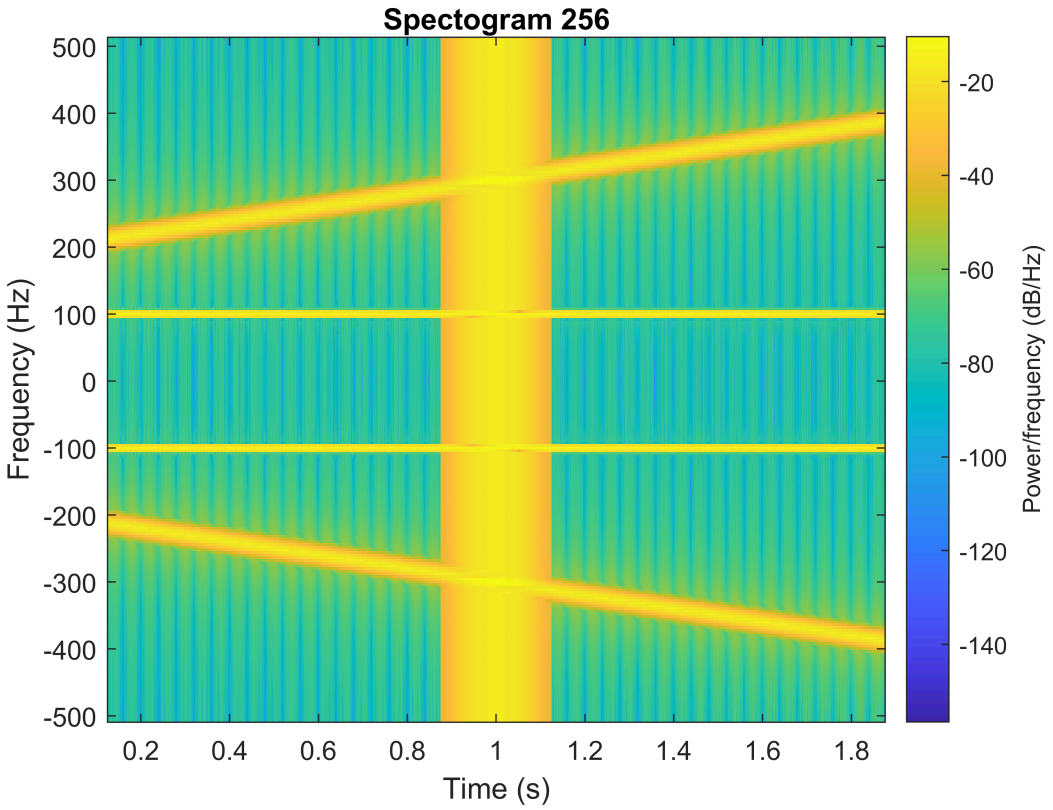
```
w1 = hamming(64);
figure('Name', 'Hamming');
spectrogram(x3, w1, 63, 64, fs, 'centered', 'yaxis');
title('Spectogram 64');
```



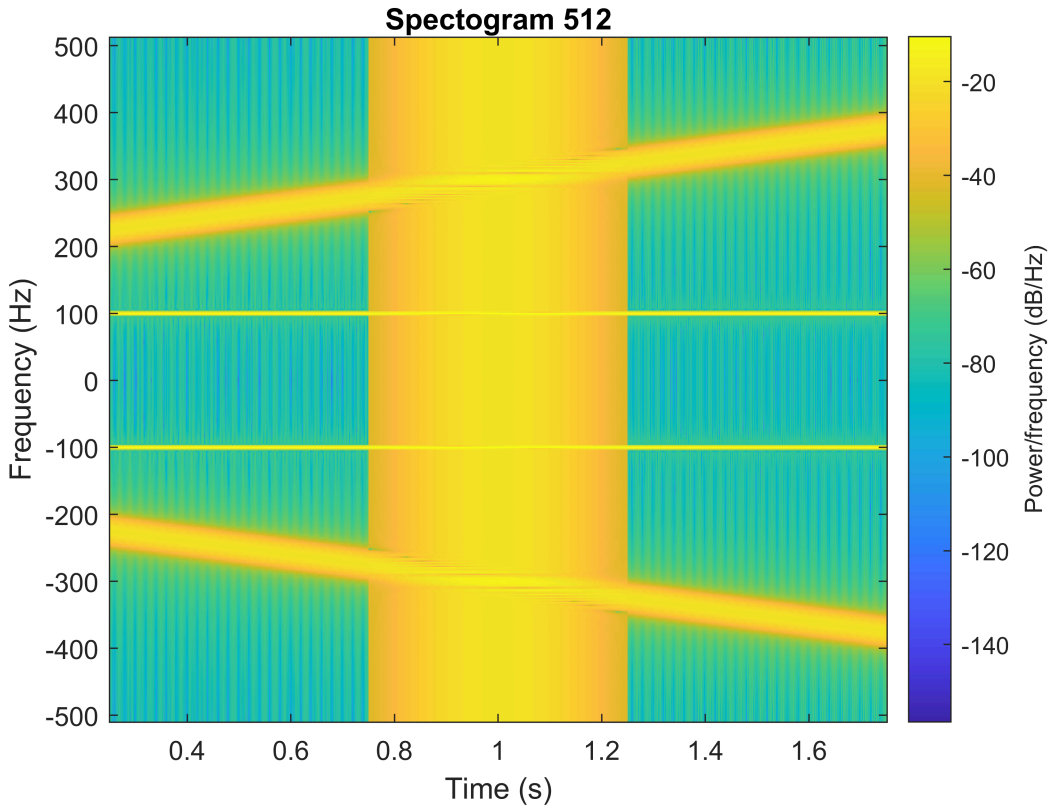

```
w2 = hamming(128);
figure('Name', 'Hamming');
spectrogram(x3, w2, 127, 128, fs, 'centered', 'yaxis');
title('Spectogram 128');
```




```
w3 = hamming(256);
figure('Name', 'Hamming');
spectrogram(x3, w3, 255, 256, fs, 'centered', 'yaxis');
title('Spectogram 256');
```

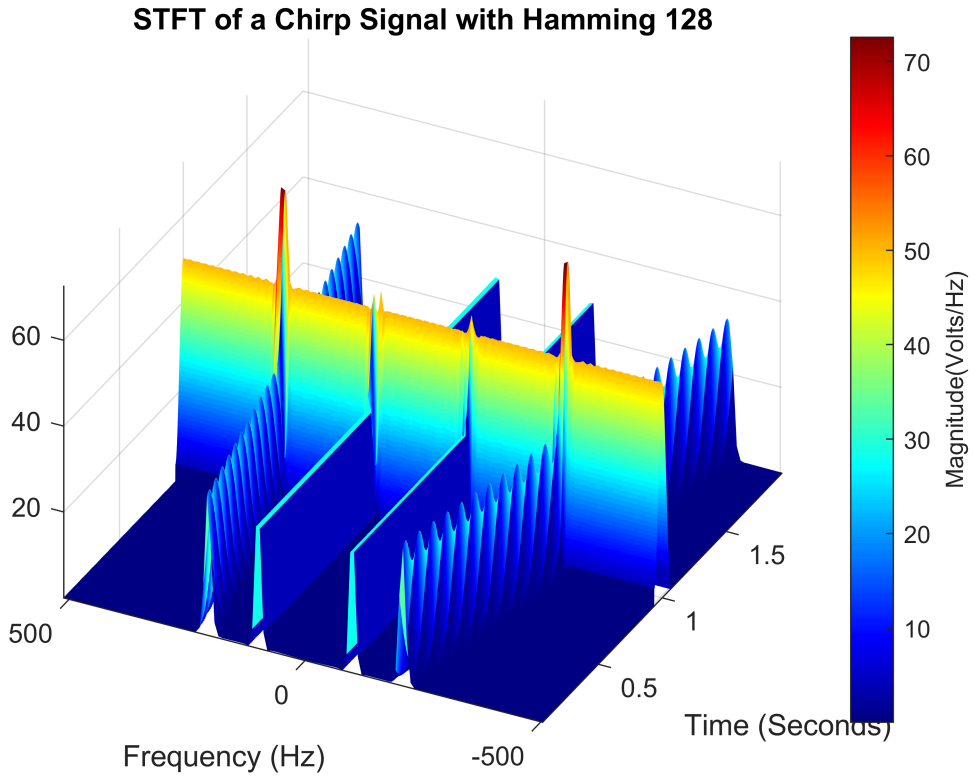



```
w4 = hamming(512);
figure('Name', 'Hamming');
spectrogram(x3, w4, 511, 512, fs, 'centered', 'yaxis');
title('Spectogram 512');
```



Here we extract the info of signals with stft function and then plot 3-D with surf function

```
[s, t, f] = stft(x3, fs, 'window', w2, 'OverLapLength', 127, 'FFTLength', 128);

figure('Name', 'Short-time Fourier Transform');
surf(f, t, abs(s), 'EdgeColor', 'none');
axis tight;
colormap(jet);
cb = colorbar;
view([-1, -0.5, 1]);
ylabel(cb, 'Magnitude(Volts/Hz)');
xlabel('Time (Seconds)');
ylabel('Frequency (Hz)');
title('STFT of a Chirp Signal with Hamming 128');
```


STFT of a Chirp Signal with Hamming 128
60
40
20
500
0
-500
Frequency (Hz)
0.5
1
1.5
Time (Seconds)
70
60
50
40
30
20
10
Magnitude(Volts/Hz)