**DEPARTMENT OF COMPUTER & SOFTWARE ENGINEERING**

**COLLEGE OF E&ME, NUST, RAWALPINDI**

**Subject Name**

**Digital Signal Processing**

**Lab Number**

**5**

**SUBMITTED TO:**

**LE Sundas Ashraf**

**SUBMITTED BY:**

**Student Name**

1. Wahaaj Nasir

**Reg#413238**

**DE- 44 Dept C&SE**

**Objectives:**

Processing in MATLab

**Related Topic/Chapter in theory class:**

Basics Of Digital Signal Processing

**Hardware/Software required:**

Hardware: PC

Software Tool: MATLab

**Task :**

**Record an audio signal saying @ 8kHz: “*Welcome to Digital Signal Processing Course.*”**

**and plot the signal in time and frequency domain with all axes correctly labeled. You can play the audio by using the MATLAB command sound. Note: if you don’t have a working microphone, you can always use the recorded audio.**

**Solution:**

%%

recorder = audiorecorder(8000, 8, 1);

fs = 8000;

disp("Start Recording");

recordblocking(recorder, 2);

disp("Stop Recording");

voice = getaudiodata(recorder);

fourier = abs(fft(voice));

subplot(2, 1, 1)

plot(voice)

xlabel('Time')

ylabel('Amplitude')

title('Time Domain')

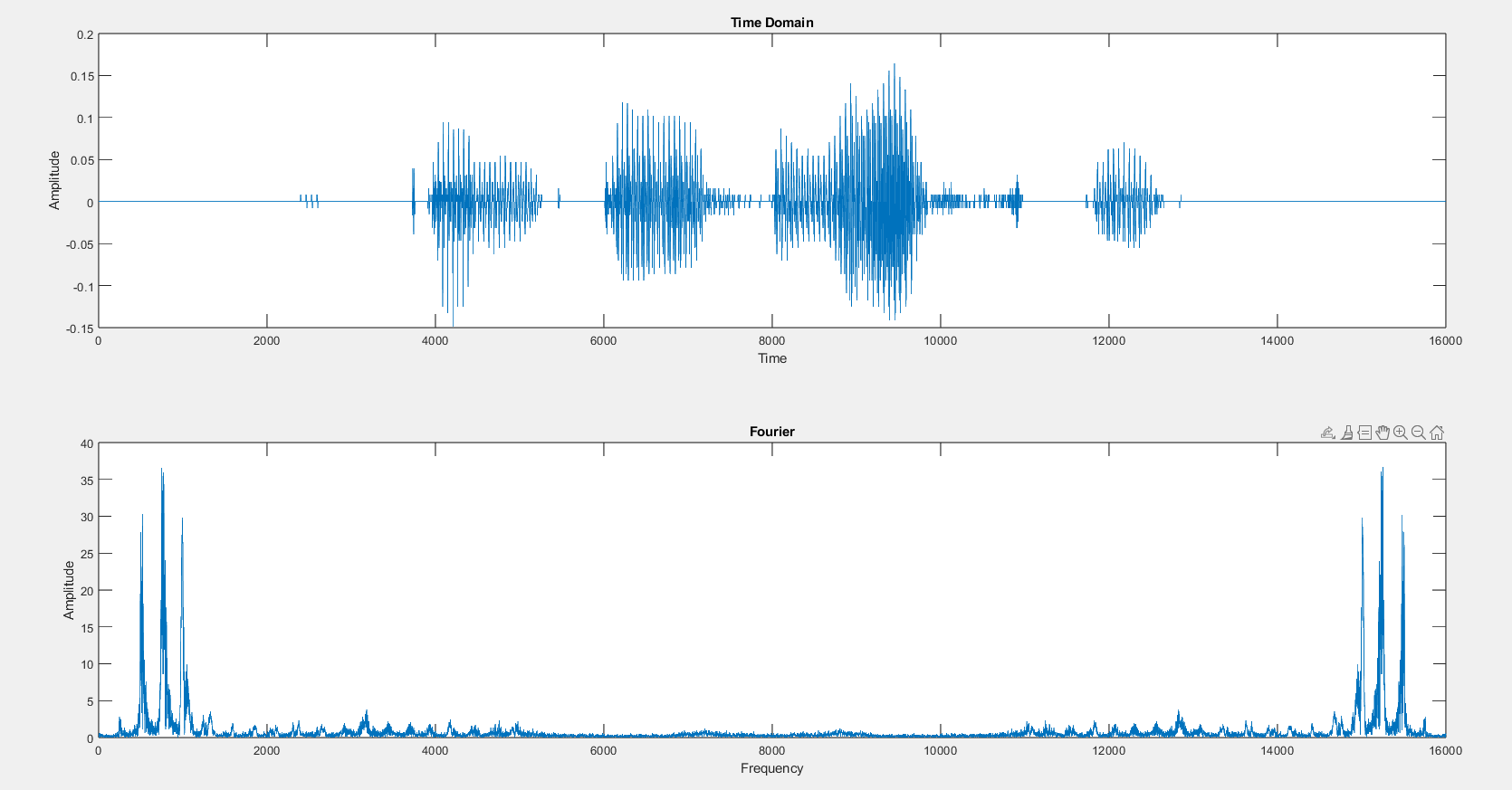
subplot(2, 1, 2)

plot(fourier)

xlabel('Frequency')

ylabel('Amplitude')

title('Fourier')

****

**Task 2:**

Now decimate the audio signal using **downsample**() and plot the resultant signal in both time and frequency domain. Listen and observe the change in the audio. Kindly note that the decimation in Time domain results in spread of the spectrum in Frequency domain by the same factor.

Keep on doing decimation and plot signals in time domains until aliasing starts to occur and you can feel the change in voice while playing it.

**Solution**

%%

function y = self\_down(x, D)

y = x(1:D:length(x));

end

factor = 1

fs\_new = 8000/factor

downsampled\_voice = self\_down(voice, factor);

downsampled\_voice = downsampled\_voice.';

downsample\_function = downsample(voice, factor);

fourier = abs(fft(downsampled\_voice));

fourier\_fun = abs(fft(downsample\_function));

subplot(2, 2, 1)

plot(downsampled\_voice)

xlabel('Time')

ylabel('Amplitude')

title('Time Domain (Downsample Own)')

subplot(2, 2, 2)

plot(fourier)

xlabel('Frequency')

ylabel('Amplitude')

title('Fourier (Downsample Own)')

subplot(2, 2, 3)

plot(downsample\_function)

xlabel('Time')

ylabel('Amplitude')

title('Time Domain (Downsample Function)')

subplot(2, 2, 4)

plot(fourier\_fun)

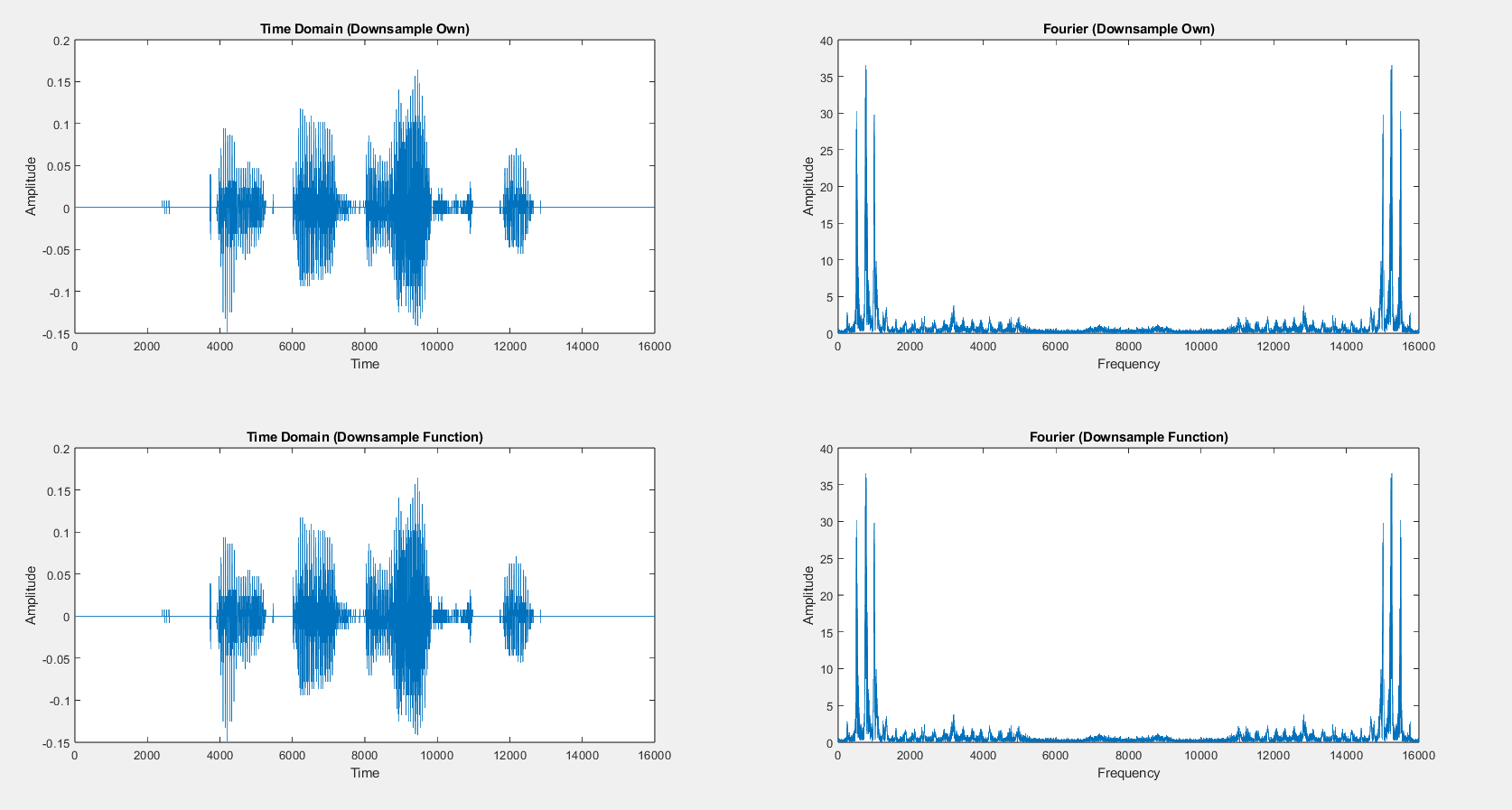
xlabel('Frequency')

ylabel('Amplitude')

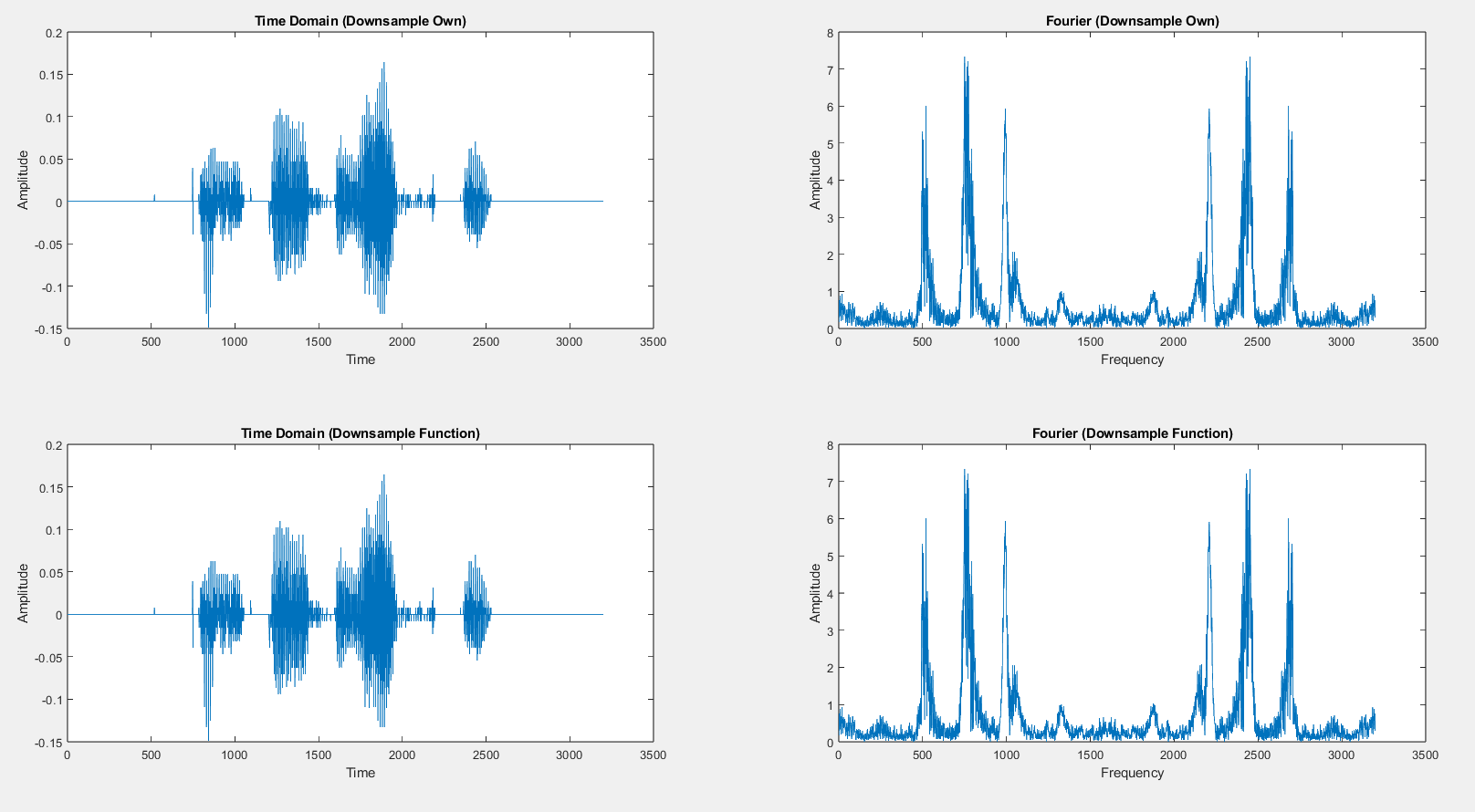
title('Fourier (Downsample Function)')

sound(downsampled\_voice, fs\_new)

**Original:**

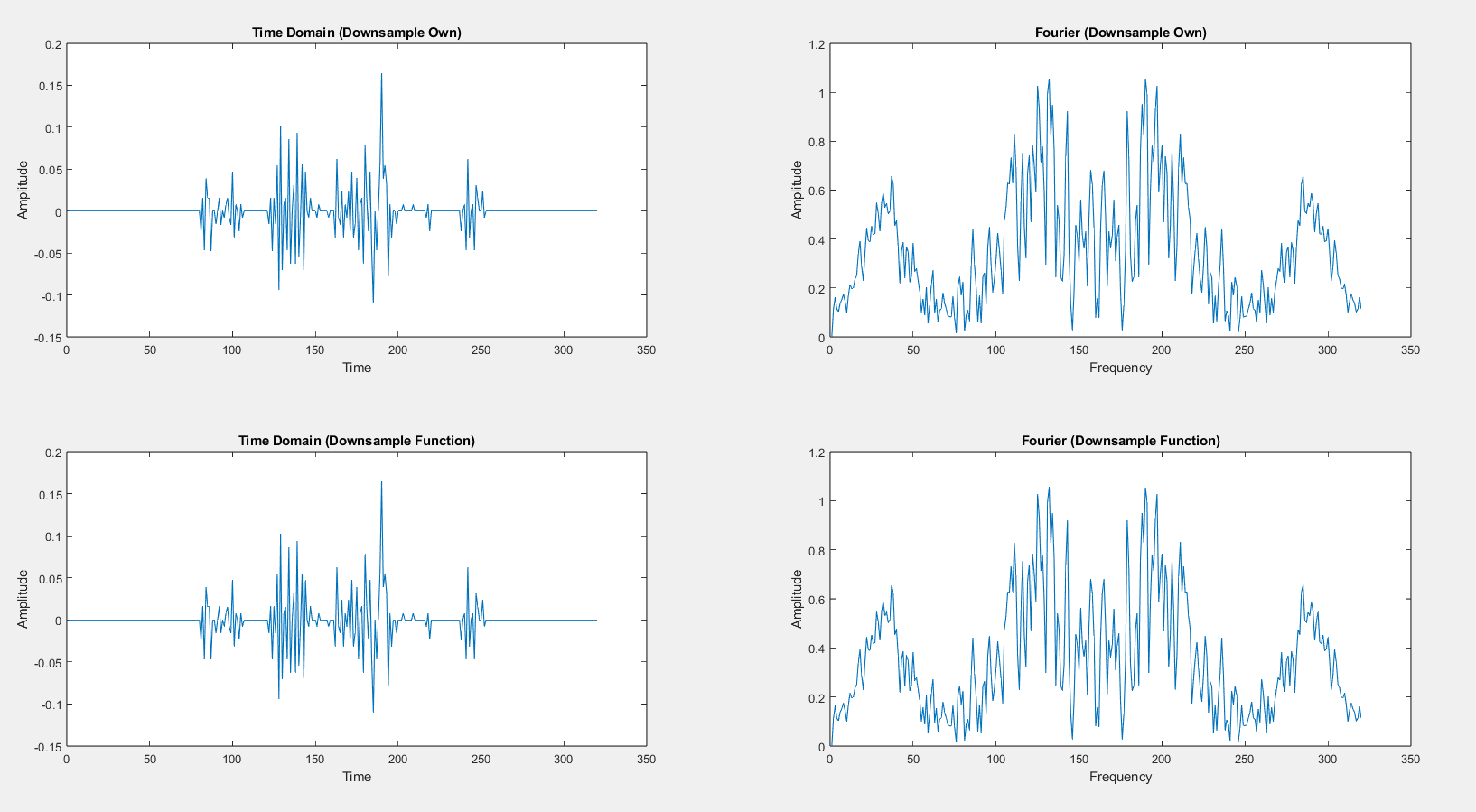


**Downsampling by 5:**



**The Fourier transform shows shrinkage, but no aliasing yet.**

**Downsampling By 50:**



**Clear aliasing. We cannot hear the signal as the sampling rate is too low for the hardware of modern laptops.**

**Task 3:**

Upsample the audio signal repeatedly by a factor 2 and plot the resultant signal in both time and frequency domain using **upsample()** function. **Note the changes happening to the signal in time domain and corresponding spectrum whenever we add upsample by adding zeros in the time domain.** Listen and observe the change in the audioAW. Do follow the same steps and plot the input signal in time and frequency domain.

**Solution**

function y = self\_up(x, U)

y = zeros(1, U \* length(x));

y(1:U:end) = x;

end

factor = 1;

fs\_new = 8000 \* factor;

upsampled\_voice = self\_up(voice, factor);

upsampled\_voice = upsampled\_voice.';

upsample\_function = upsample(voice, factor);

fourier = abs(fft(upsampled\_voice));

fourier\_fun = abs(fft(upsample\_function));

subplot(2, 2, 1)

plot(upsampled\_voice)

xlabel('Time')

ylabel('Amplitude')

title('Time Domain (Upsample Own)')

subplot(2, 2, 2)

plot(fourier)

xlabel('Frequency')

ylabel('Amplitude')

title('Fourier (Upsample Own)')

subplot(2, 2, 3)

plot(upsample\_function)

xlabel('Time')

ylabel('Amplitude')

title('Time Domain (Upsample Function)')

subplot(2, 2, 4)

plot(fourier\_fun)

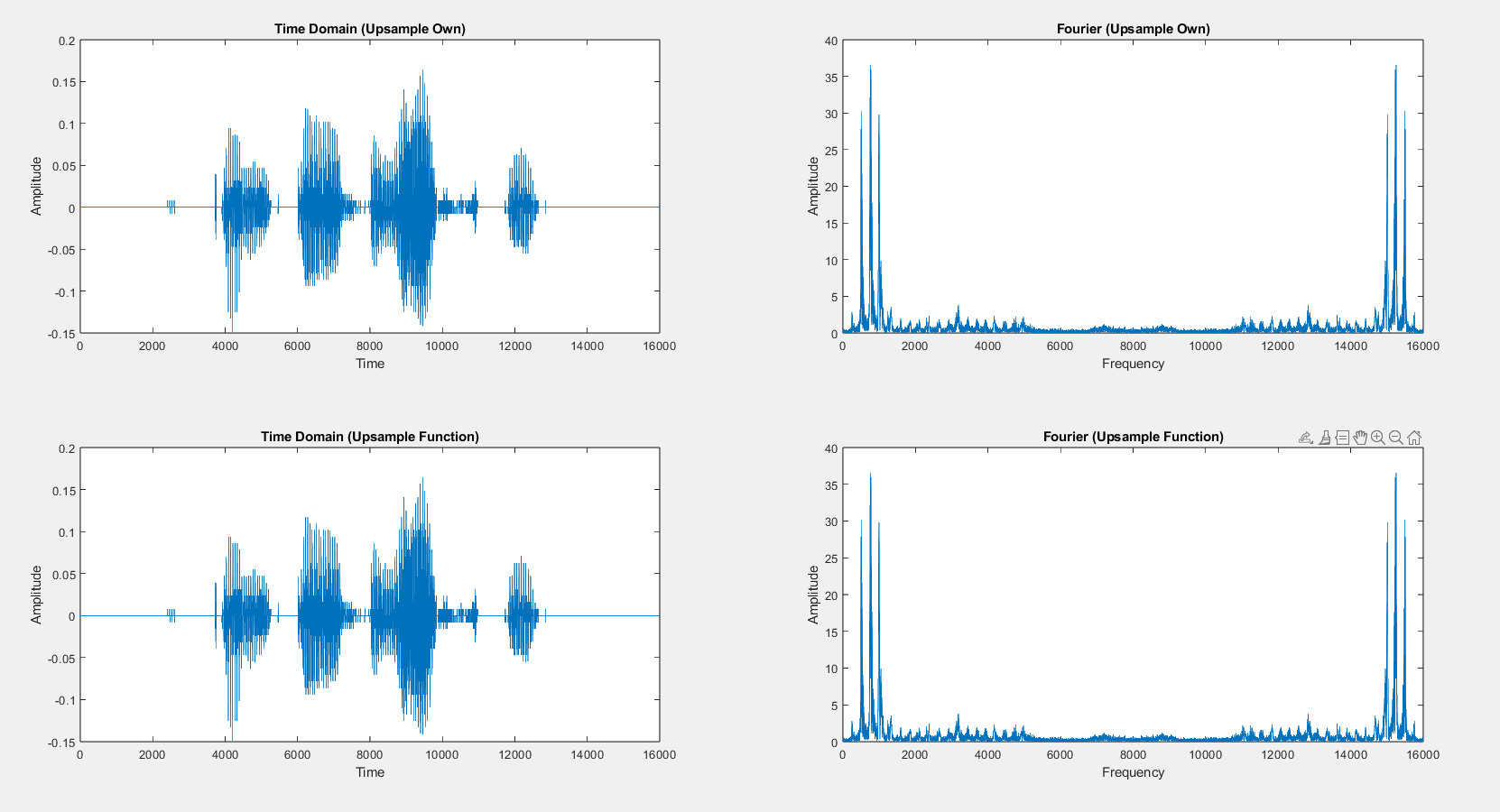
xlabel('Frequency')

ylabel('Amplitude')

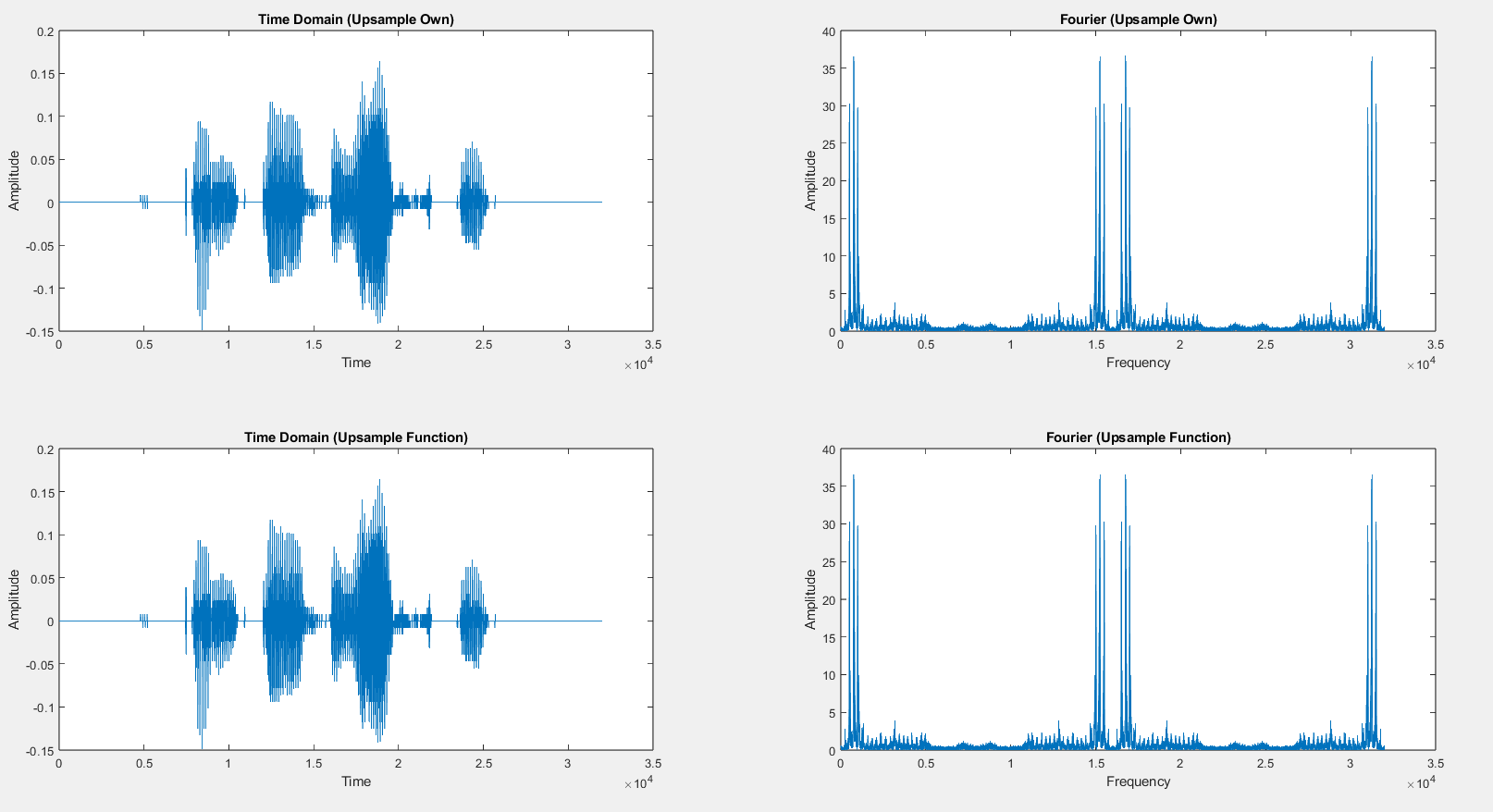
title('Fourier (Upsample Function)')

sound(upsampled\_voice, fs\_new)

**Original Signal**

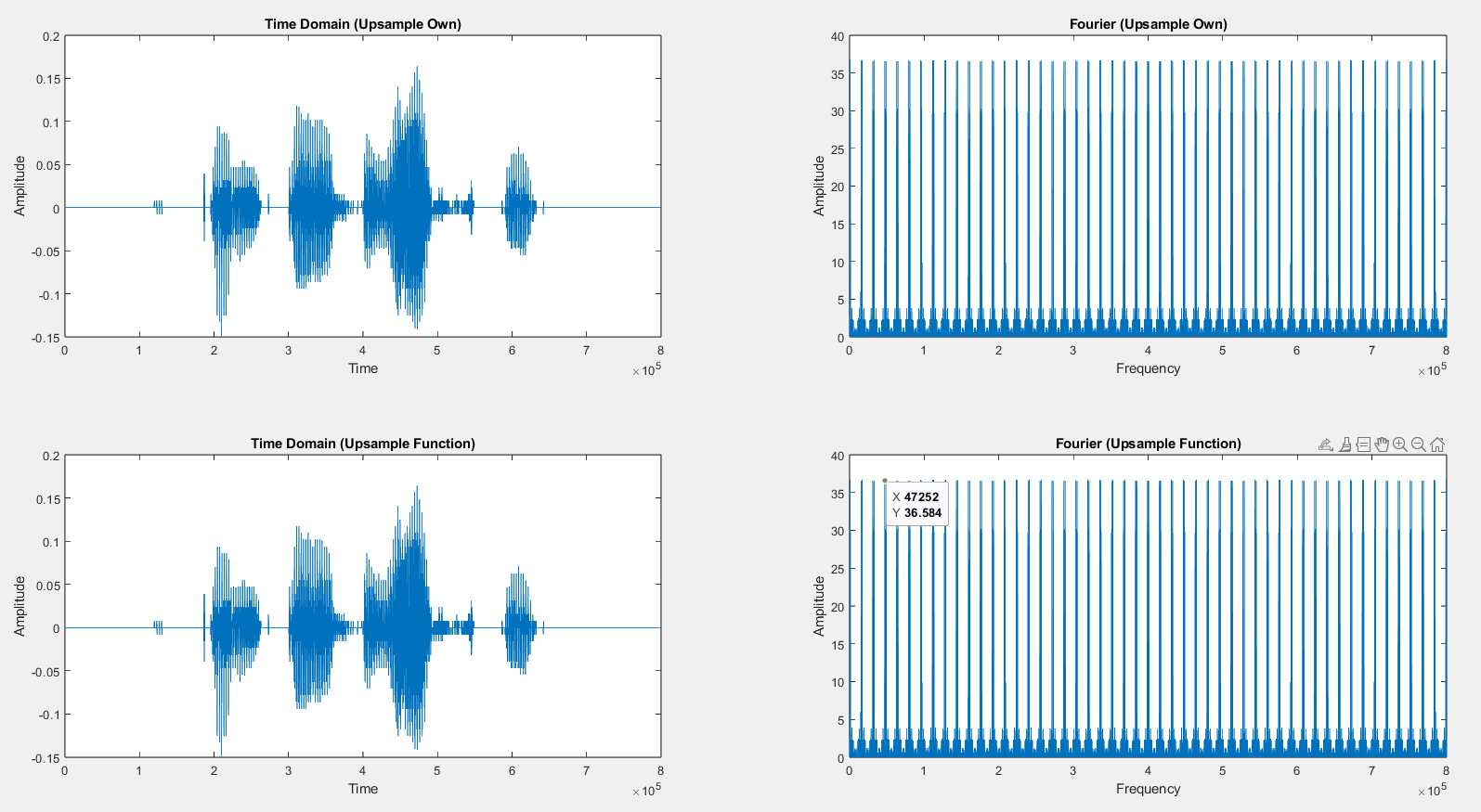


**Upsampling By 2:**



**As we can see, by upsampling a factor of 2, we get replicated peaks in higher frequencies.**

**Upsampling by 50:**



**Many peaks are located at higher frequencies due to excessive upsampling**

**Task 4:**

Resample the audio signal to a sampling rate of 3/5 by using the functions downsample() and upsample(). Repeat the task for sampling rate of 5/3.

Analyze the relation between original audio signal and its decimated and interpolated versions based on the magnitude spectrum and by listening the audio after each step.

**Solution**

%% Resampling 3/5

down\_factor = 3;

up\_factor = 5;

fs\_3\_5 = 8000 \* (3/5);

downsampled\_voice\_3 = downsample(voice, down\_factor);

upsampled\_voice\_5 = upsample(downsampled\_voice\_3, up\_factor);

fourier\_3\_5 = abs(fft(upsampled\_voice\_5));

figure;

subplot(2,1,1)

plot(upsampled\_voice\_5)

xlabel('Samples')

ylabel('Amplitude')

title('Time Domain (3/5)')

subplot(2,1,2)

plot(fourier\_3\_5)

xlabel('Frequency')

ylabel('Magnitude')

title('Fourier (3/5)')

sound(upsampled\_voice\_5, fs\_3\_5)

%% Resampling 5/3

up\_factor = 3;

down\_factor = 5;

fs\_5\_3 = 8000 \* (5/3);

downsampled\_voice\_5 = downsample(voice, down\_factor);

upsampled\_voice\_3 = upsample(downsampled\_voice\_5, up\_factor);

fourier\_5\_3 = abs(fft(upsampled\_voice\_3));

figure;

subplot(2,1,1)

plot(upsampled\_voice\_3)

xlabel('Samples')

ylabel('Amplitude')

title('Time Domain (5/3)')

subplot(2,1,2)

plot(fourier\_5\_3)

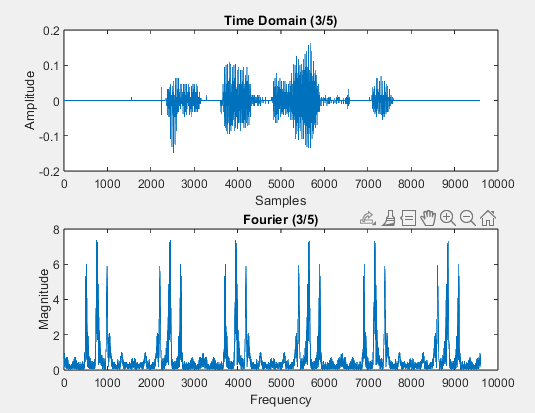
xlabel('Frequency')

ylabel('Magnitude')

title('Fourier (5/3)')

sound(upsampled\_voice\_3, fs\_5\_3)

**Downsample by 3, then Upsample by 5:**

****

**Downsample by 5, then Upsample by 3:**

