Math For Communication MATLAB Project

Done By Omar Elmahy ID:210237

Contents

Initializing the project

Started by clearing all the variables and the existing log to have a fresh start every run using these lines:

```
clear
clc
```

I then loaded in “handel” which is a sound file that is built into MATLAB. The sound file is a recording of Handel’s Hallelujah Chorus which consists of “Fs” that has the value of 8192 and “y” that holds 9 seconds of the audio. I used the line:

```
load handel
```

I then proceeded to play back the loaded audio using the line:

```
sound(y,Fs)
```

Part A

If we want to graph the signal using the appropriate time domain, we should use a vector that has the values in seconds for our time domain and y, which is also a vector, that we got when we loaded handel. The time domain by default starts from 1 but I made it start from 0 to the length of y but -1 since I shifted the initial value. All of these are divided by Fs which is the sampling frequency

```
plot((0:length(y)-1) / Fs, y)
```

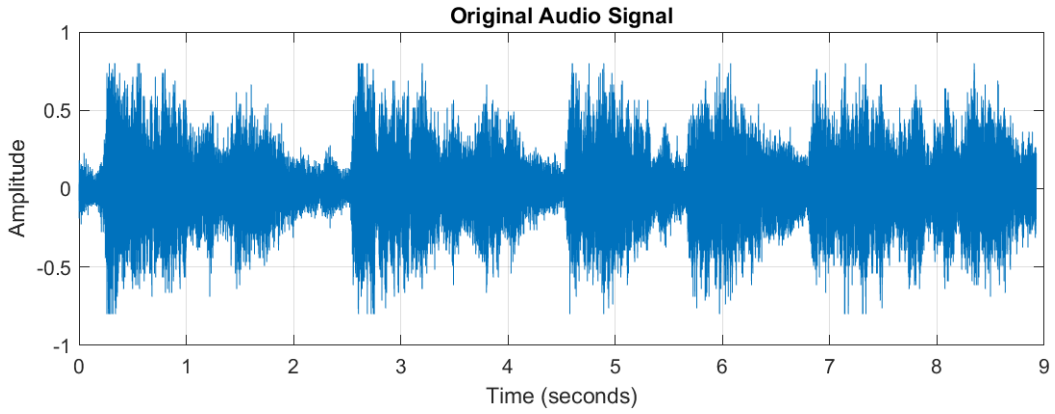
The title of the graph and its x,y labels are defined as the following:

```
title('Original Audio Signal');
xlabel('Time (seconds)');
ylabel('Amplitude');
```

For the continuous time variable, I created a variable t_original and the function t_scaled which is scaled by factor a. The factor a is set in the code to be 3 and the code is as follows:

```
a = 3;
t_original = 0:length(y)-1;
t_scaled = a.*t_original;
```

Audio Graph:



Part B

Computing the Frequency Spectrum

To compute the frequency spectrum of the signal, we first got the Discrete-time Fourier transform using the function `fft()` and then we rearranged the signal by shifting the zero-frequency components to the center of the array using the function: `fftshift()` This is useful when we get to analyze the frequency components of audio signals, as it allows you to see the low-frequency and high-frequency components of the signal more clearly. Lastly, to obtain the Magnitude of the signal we get the absolute value of the results of the previous functions. Here is the appropriate code: `abs(fftshift(fft(y)))`

Creating the Frequency Vector

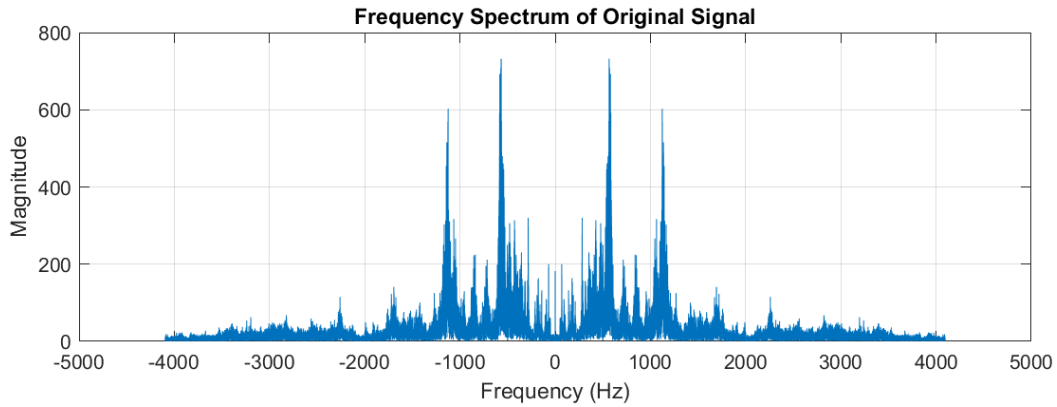
The frequency vector is the vector that will be plotted on the x-axis. In the code, it is symmetrical across 0, therefor we created the vector from the -length(y)/2 to (length/2)-1. The -1 as mentioned earlier is needed because the default value is 1 so since we shifted it, we minus by one. Here is the appropriate code: `-length(y)/2:length(y)/2-`

Plotting the Graph

Here is the appropriate code to plot the graph:

```
plot((-length(y)/2:length(y)/2-1) * Fs/length(y), abs(fftshift(fft(y))));
title('Frequency Spectrum of Original Signal');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
grid on;
```

Frequency Graph:



Part C

Downsampling the Audio Signal

We are going to downsample the audio at a rate of `D = 2;`. The function that we will be using is called `downsample(y,D)` and to be able to use it on MATLAB, you need to install an Add-on called "Signal Processing Toolbox." That way we got out our downsampled y and to get our downsampled Fs, we divide it by the downsampling rate. Here is the appropriate code:

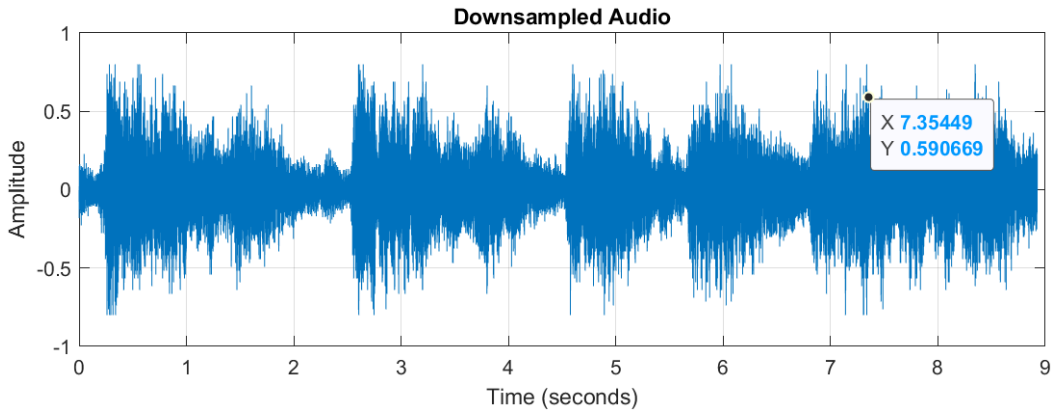
```
D = 2;
y_downsampled = downsample(y, D);
Fs_downsampled = Fs / D;
```

Plotting the Downsampled Audio Signal

We are going to do what we did previously in Part A when plotting the audio signal only changing the y and Fs to our newly made y_downsampled and Fs_downsampled. Here is the appropriate code:

```
plot((0:length(y_downsampled)-1) / Fs_downsampled, y_downsampled);
title('Downsampled Audio');
xlabel('Time (seconds)');
ylabel('Amplitude');
grid on;
```

Audio Graph:



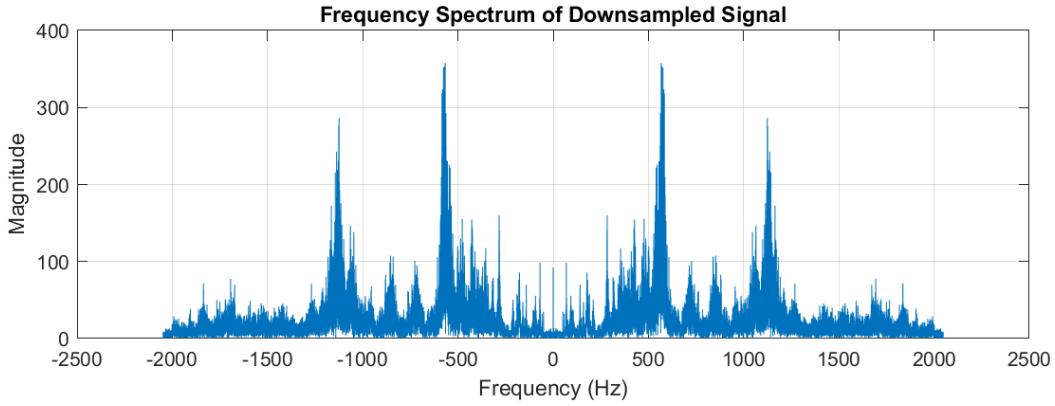
Plotting the Downsampled Frequency Signal

We are going to do what we did previously in Part B when plotting the frequency signal only changing the y and Fs to our newly made y_downsampled and Fs_downsampled. Here is the appropriate code:

```
plot((-length(y_downsampled)/2:length(y_downsampled)/2-1) *
Fs_downsampled/length(y_downsampled), abs(fftshift(fft(y_downsampled))));

title('Frequency Spectrum of Downsampled Signal');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
grid on;
```

Frequency Graph:



Part D

Anti-aliasing Lowpass Filter

To create an anti-aliasing lowpass filter, we are going to use the cheby1 function that was provided in the Project Description. We are going to use the results of that in the function called filter with the parameters (b, a and y_downsampled) where y_downsampled is the signal that we going to filter and a and b are the confections of the rational transfer function as shown in this image:

$$Y(z) = \frac{b(1) + b(2)z^{-1} + \ldots + b(n_b + 1)z^{-n_b}}{1 + a(2)z^{-1} + \ldots + a(n_a + 1)z^{-n_a}} X(z),$$

Figure 1 Rational Transfer Function

Here is the appropriate code for this portion:

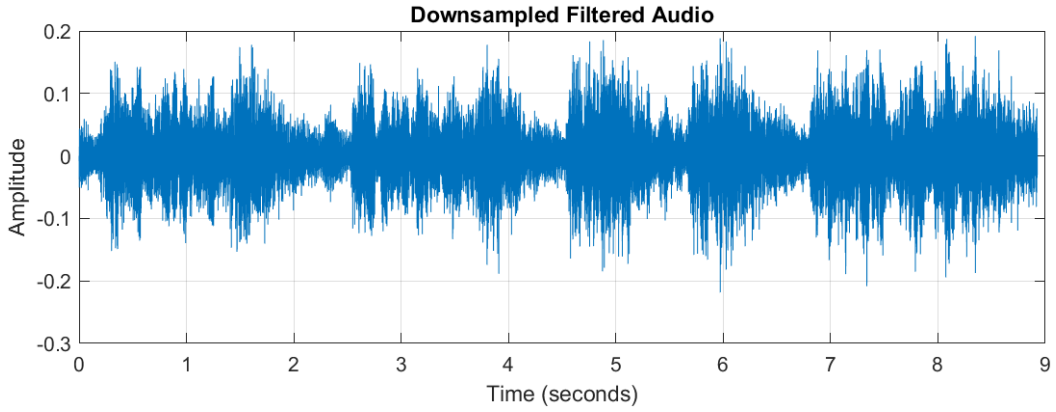
```
b=cheby1(5, 1, 0.45);
a=1;
y_filtered = filter(b, a, y_downsampled);
Fs_filtered = Fs / D;
```

Plotting the Filtered Audio Signal

We are going to do what we did previously in Part A when plotting the audio signal only changing the y and Fs to our newly made y_filtered and Fs_filtered. Here is the appropriate code:

```
plot((0:length(y_filtered)-1) / Fs_filtered, y_filtered);
title('Downsampled Filtered Audio');
xlabel('Time (seconds)');
ylabel('Amplitude');
grid on;
```

Audio Graph:



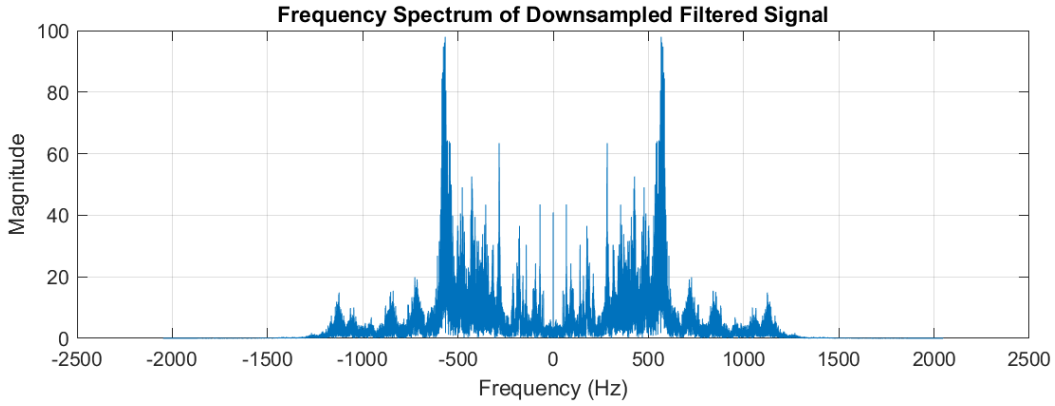
Plotting the Filtered Frequency Signal

We are going to do what we did previously in Part B when plotting the frequency signal only changing the y and Fs to our newly made y_filtered and Fs_ filtered. Here is the appropriate code:

```
plot((-length(y_filtered)/2:length(y_filtered)/2-1) *
Fs_filtered/length(y_filtered), abs(fftshift(fft(y_filtered))));

title('Frequency Spectrum of Downsampled Filtered Signal');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
grid on;
```

Frequency Graph:



Extracting the .wav Files

To extract the .wav files, I used the function audiowrite and with the 3 different parameters. Here is the appropriate code:

```
audiowrite ('Original Audio.wav',y,Fs)
audiowrite ('DownSampled Audio.wav',y_downsampled,Fs_downsampled)
audiowrite ('Filtered Audio.wav',y_filtered,Fs_filtered)
```

All .wav files are linked to the zip file with this report.

The Code:

```
% Initalzing
clear
clc
load handel
sound(y, Fs);

% The title of the figure that will contain all the graphs
figure('Name','The Three Audio Signal with Six Graphs','NumberTitle','off');

% Original Signal Time Graph
subplot(3, 2, 1);
plot((0:length(y)-1) / Fs, y);  % Plotting the original audio signal
title('Original Audio Signal');
xlabel('Time (seconds)');
ylabel('Amplitude');
grid on;

% Continous Time Variable
a = 3; %Defineing the Scaling Factor
t_original = 0:length(y)-1;
t_scaled = a.*t_original;

% Original Signal Frequency Graph

subplot(3, 2, 2);
plot((-length(y)/2:length(y)/2-1) * Fs/length(y), abs(fftshift(fft(y))));  % Plotting
the frequency spectrum of the original signal
title('Frequency Spectrum of Original Signal');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
grid on;

pause(9);  % Pause to allow the first audio to finish before plotting the next one

% Downsampled Audio Signal
D = 2;
y_downsampled = downsample(y, D);  % Downsampling the audio
Fs_downsampled = Fs / D;
```

```
% Downsampled Signal Time and Frequency
subplot(3, 2, 3);
plot((0:length(y_downsampled)-1) / Fs_downsampled, y_downsampled);  % Plotting the
downsampled audio signal
title('Downsampled Audio');
xlabel('Time (seconds)');
ylabel('Amplitude');
grid on;

subplot(3, 2, 4);
plot((-length(y_downsampled)/2:length(y_downsampled)/2-1) *
Fs_downsampled/length(y_downsampled), abs(fftshift(fft(y_downsampled))));  % Plotting
the frequency spectrum of the downsampled signal
title('Frequency Spectrum of Downsampled Signal');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
grid on;

% Play back the downsampled audio
sound(y_downsampled, Fs_downsampled);  % Playing back the second audio
pause(9);

% Filtered Audio Signal
b=cheby1(5, 1, 0.45);
a=1;
y_filtered = filter(b, a, y_downsampled);  % Downsampling the audio with an anti-
aliasing or lowpassfilter
Fs_filtered = Fs / D;

% Filtered Signal with Time and Frequency
subplot(3, 2, 5);
plot((0:length(y_filtered)-1) / Fs_filtered, y_filtered);  % Plotting the downsampled
and filtered audio signal
title('Downsampled Filtered Audio');
xlabel('Time (seconds)');
ylabel('Amplitude');
grid on;

subplot(3, 2, 6);
plot((-length(y_filtered)/2:length(y_filtered)/2-1) * Fs_filtered/length(y_filtered),
abs(fftshift(fft(y_filtered))));  % Plotting the frequency spectrum of the
downsampled and filtered signal
title('Frequency Spectrum of Downsampled Filtered Signal');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
grid on;

% Play back the downsampled and filtered audio
sound(y_filtered, Fs_filtered);  % Playing back the third audio

% Saving all the three Audio into .wav
audiowrite ('Original Audio.wav',y,Fs)
audiowrite ('DownSampled Audio.wav',y_downsampled,Fs_downsampled)
audiowrite ('Filtered Audio.wav',y_filtered,Fs_filtered)
```