National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Department of Electrical Engineering and Computer Science

Faculty Member: Dr. Ahmad Salman   Dated: 06/03/2023

Semester: 6th   Section: BEE 12C

EE-330 Digital Signal Processing

Lab 5: Sampling of Audio Signals in MATLAB

Group Members

		PLO4 - CLO4		PLO5 - CLO5	PLO8 - CLO6	PLO9 - CLO7
Name	**Reg. No**	**Viva / Quiz / Lab Performance**	**Analysis of data in Lab Report**	**Modern Tool Usage**	**Ethics and Safety**	**Individual and Teamwork**
		5 Marks	**5 Marks**	**5 Marks**	**5 Marks**	**5 Marks**
Danial Ahmad	331388					
Muhammad Umer	345834					
Tariq Umar	334943					

1 Table of Contents

2 Sampling of Audio Signals in MATLAB

2.1 Objectives

The objective of this lab is to down sample audio signals and perform signal analysis in the frequency domain.

- Familiarization with sampling
- Analysis of down sampled signal in frequency domain

2.2 Introduction

The ability to manipulate signals in the frequency domain is crucial in various fields, including audio processing and communication systems. By performing down sampling, we aimed to study the effect of reducing the signal's sampling rate on its frequency spectrum.

In this report, we will present our methodology for down sampling audio signals, followed by an analysis of the down sampled signals in the frequency domain. We will examine the resulting frequency spectra of the signals and compare them to the original signals to study the impact of down sampling on the signal's frequency content.

2.3 Software

MATLAB is a high-level programming language and numerical computing environment. Developed by MathWorks, it provides an interactive environment for numerical computation, visualization, and programming. MATLAB is widely used in various fields, including engineering, science, and finance, due to its capabilities for matrix and vector operations, implementation of algorithms, and creation of graphical representations of data. The objective of this lab is to provide a hands-on experience with the A-to-D sampling and the D-to-A reconstruction processes that are essential for digital image processing. We will also demonstrate a commonly used method of image zooming (reconstruction) that produces “poor” results, which will help illustrate the importance of understanding the underlying principles of digital image processing.

2.4 Lab Report Instructions

All questions should be answered precisely to get maximum credit. Lab report must ensure following items:

- Lab objectives
- MATLAB codes
- Results (graphs/tables) duly commented and discussed
- Conclusion

3 Lab Procedure

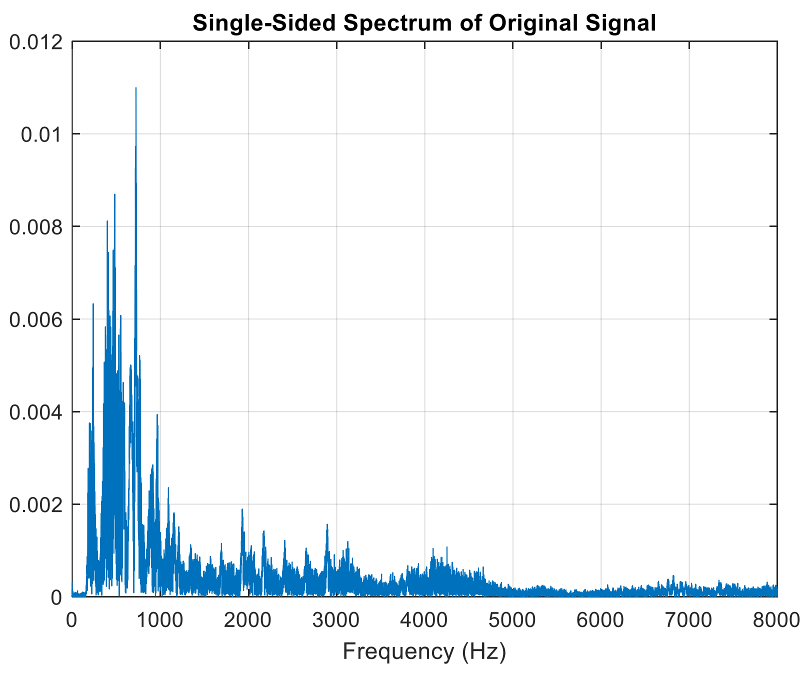
3.1 Sampling

In this lab you will gain some practical knowledge about how to handle real world signals. In any modern signal processing system (e.g., in a telecommunication system), signal acquisition, its processing and efficient storage/transmission are the critical steps. In the class lectures, you have gained the knowledge about sampling of a continuous-time signal, change of sampling rates and their hierarchical criteria. This lab covers the above-mentioned tasks i.e., change of sampling rate.

3.2 Lab Task

1. You are given a speech signal. Consider it a discrete-time signal with the sampling frequency fs=16 kHz. Load the signal in MATLAB using the function *audioread*. Listen to the signal using *sound*.

```
%% Original Signal
[x, Fs] = audioread('sample.wav');
L = length(x);

NFFT = 2 ^ nextpow2(L); % Next power of 2 from length of y
Y1 = fft(x, NFFT) / L;
f = Fs / 2 * linspace(0, 1, NFFT / 2 + 1);
Fc = Fs / 2;
[b, a] = butter(6, Fc / (Fs / 2) - 0.01);

figure
plot(f, 2 * abs(Y1(1:NFFT / 2 + 1))), grid on
title('Single-Sided Spectrum of Original Signal')
xlabel('Frequency (Hz)')
sound(x, Fs)
```



2. Design a 6th order low-pass Butterworth filter. Hint: see MATLAB help for *butter* and *filter*. The butter command takes the normalized cutoff frequency (in the range 0-1) as an input argument where the maximum 1 means fS/2.

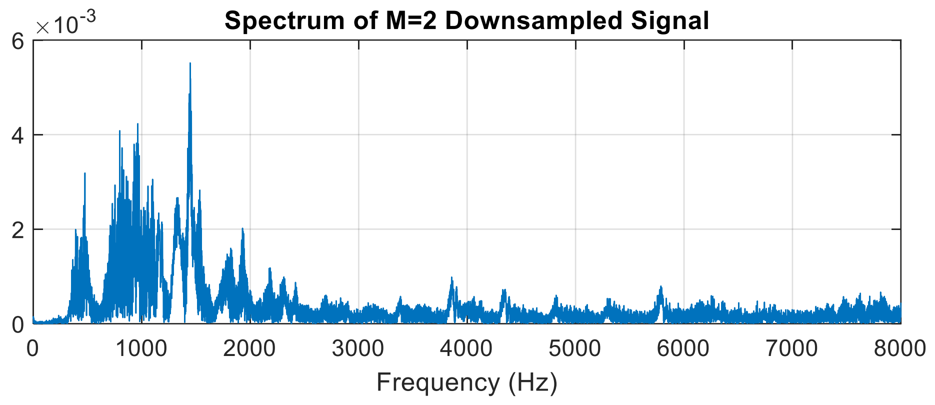
```
[b, a] = butter(6, Fc / (Fs / 2) - 0.01);
```

3. Consider the maximum frequency of the speech signal fN = fS/2. Apply the filter.

```
filtered_x = filter(b, a, x);
```

4. Now down sample the filtered signal by the factor of 2 i.e., *M*=2. Do this manually by picking up every alternative sample and storing it in a different array.

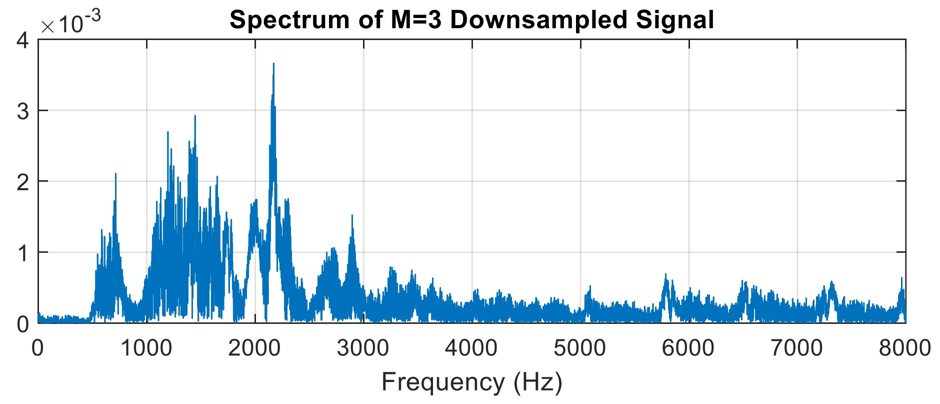
```
%% Downsampled by 2
filtered_x = filter(b, a, x);
downsampled_x = filtered_x(1:2:end);

Y2 = fft(downsampled_x, NFFT) / L;
sound(downsampled_x, Fs / 2)
```

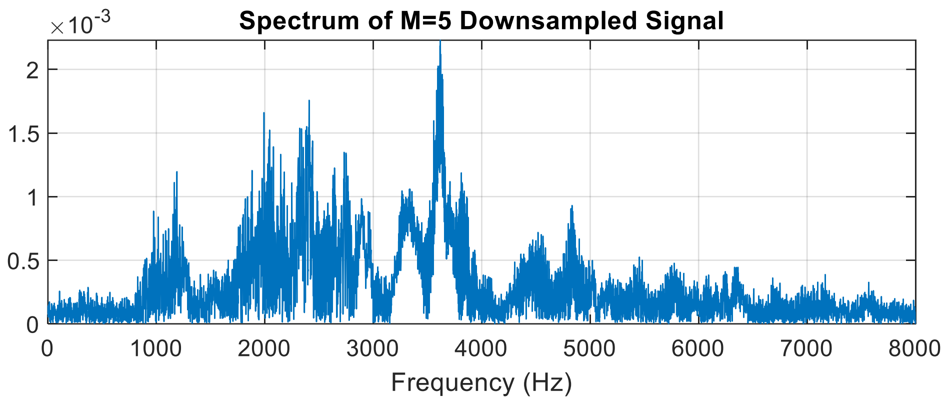


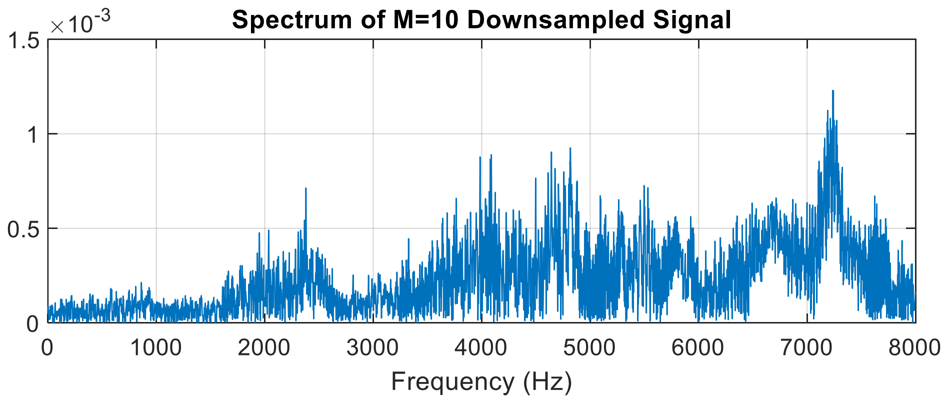
5. See the MATLAB help for the function *downsample*. Apply this function for down sampling the signal by the factors *M* = 3,5,10. Listen to the output signal in every case and prepare your conclusions. Also plot the spectrum of the input and output signal in a subplot for original and three cases for different *M*.

```
%% Downsampled by 3
filtered_x = filter(b, a, x);
downsampled_x = downsample(filtered_x, 3);

Y3 = fft(downsampled_x, NFFT) / L;
sound(downsampled_x, Fs / 3)

%% Downsampled by 5
filtered_x = filter(b, a, x);
downsampled_x = downsample(filtered_x, 5);

Y4 = fft(downsampled_x, NFFT) / L;
sound(downsampled_x, Fs / 5)

%% Downsampled by 10
```

```
filtered_x = filter(b, a, x);
downsampled_x = downsample(filtered_x, 10);

Y5 = fft(downsampled_x, NFFT) / L;
sound(downsampled_x, Fs / 10)
```

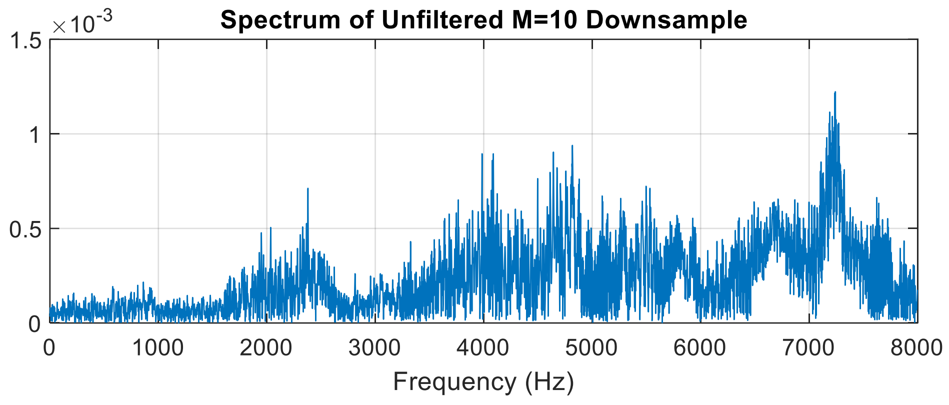






6. For M = 10, avoid the anti-aliasing filter and directly sample the speech to listen if there is any difference. Also plot the spectrum using code given below for input and output signal.

```
%% Unfiltered Downsampling by 10
downsampled_x = downsample(x, 10);

Y6 = fft(downsampled_x, NFFT) / L;
sound(downsampled_x, Fs / 10)
```



Code for Plot

```
 %% Plots
figure
subplot(2, 1, 1)
plot(f, 2 * abs(Y1(1:NFFT / 2 + 1))), grid on
title('Spectrum of Original Signal')
xlabel('Frequency (Hz)')
subplot(2, 1, 2)
plot(f, 2 * abs(Y2(1:NFFT / 2 + 1)))
grid on
title('Spectrum of M=2 Downsampled Signal')
xlabel('Frequency (Hz)')

figure
subplot(2, 1, 1)
plot(f, 2 * abs(Y1(1:NFFT / 2 + 1))), grid on
title('Spectrum of Original Signal')
xlabel('Frequency (Hz)')
subplot(2, 1, 2)
plot(f, 2 * abs(Y3(1:NFFT / 2 + 1)))
grid on
title('Spectrum of M=3 Downsampled Signal')
xlabel('Frequency (Hz)')

figure
subplot(2, 1, 1)
plot(f, 2 * abs(Y1(1:NFFT / 2 + 1))), grid on
title('Spectrum of Original Signal')
xlabel('Frequency (Hz)')
subplot(2, 1, 2)
plot(f, 2 * abs(Y4(1:NFFT / 2 + 1)))
grid on
title('Spectrum of M=5 Downsampled Signal')
xlabel('Frequency (Hz)')

figure
subplot(2, 1, 1)
plot(f, 2 * abs(Y1(1:NFFT / 2 + 1))), grid on
title('Spectrum of Original Signal')
xlabel('Frequency (Hz)')
subplot(2, 1, 2)
plot(f, 2 * abs(Y5(1:NFFT / 2 + 1)))
grid on
```

```
title('Spectrum of M=10 Downsampled Signal')
xlabel('Frequency (Hz)')

figure
subplot(2, 1, 1)
plot(f, 2 * abs(Y1(1:NFFT / 2 + 1))), grid on
title('Spectrum of Original Signal')
xlabel('Frequency (Hz)')
subplot(2, 1, 2)
plot(f, 2 * abs(Y6(1:NFFT / 2 + 1)))
grid on
title('Spectrum of Unfiltered M=10 Downsample')
xlabel('Frequency (Hz)')
```

4 Conclusion

In this lab experiment, we aimed to down sample audio signals and analyze the resulting signals in the frequency domain. Through our analysis, we observed that down sampling a signal leads to a loss of high-frequency content in the frequency domain. This is because, during down sampling, we remove some of the original signal's high-frequency components by averaging the adjacent samples. We also observed that the lower the sampling rate, the more significant the loss of high-frequency content. Our analysis also demonstrated that the down sampled signals' frequency content is significantly different from the original signal's frequency content. This observation is essential in signal processing applications where preserving the signal's frequency content is crucial.