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

Name	Reg. No	Lab Report Marks	Viva Marks	Total
		10 Marks	5 Marks	15 Marks
Ahmed Mohsin	333060			
Hassan Rizwan	335753			
Syeda Fatima Zahra	334379			

1 Table of Contents

Sampling and Quantization of audio signals in MATLAB **3**

1.1 Objectives 3

1.2 Introduction 3

1.3 Software 3

Lab Exercises **4**

2.1. Sampling 4

Conclusion **7**

Sampling and Quantization of audio signals in MATLAB

1.1 Objectives

The purpose of the lab is as follows:

- Down sampling audio signals
- Analyzing the audio signals in frequency domain
- Familiarizing with sampling of signals, aliasing, and cutoff frequency.
- Creating low pass filter with certain cutoff frequency.

1.2 Introduction

Digital audio processing involves the conversion of continuous analog signals to digital signals, which can be processed by a computer. Sampling and quantization are the two main steps involved in the digital conversion of analog signals. In this lab, we explore these two processes in detail using MATLAB.

1.3 Software

MATLAB R2022b

Lab Exercises

2.1. Sampling

1. You are given a speech signal. Consider it a discrete-time signal with the sampling frequency fs=16 kHz.

2. Load the signal in Matlab using the function *audioread*. Listen to the signal using *sound*.

```
%Part 1
%Reading the audio file
[y,Fc]=audioread('sample.wav');
sound(y,Fc);
```

3. 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 $fS/2$.

```
%Part 2- Designing low-pass butterworth filter

%Calculating normalized cutoff frequency for lowpass filter
Fs=16000;
Fn=Fc/2; %maximum frequency
F=Fn/(Fs/2); %normalized cut-off frequency (1)
```

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

```
%Part 3-Applying the filter

%Getting tf for lowpass filter
[b,a]=butter(6,0.99);

%Applying lowpass filter
y_filtered=filter(b,a,y);
```

5. Now downsample 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.

```
%Part 4-Downsamply Manually
y_downsampled=y_filtered(1:2:end);
sound(y_downsampled,Fc/2)
```

6. See the Matlab help for the function *downsample*. Apply this function for downsampling 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 subplots for original and three cases for different M.

```
% 5-Downsampling

%Downsampling signal by M=2,3,4,10
y_downsampled1=downsample(y_filtered,2);
sound(y_downsampled1,Fc/2)
y_downsampled2=downsample(y_filtered,3);
sound(y_downsampled2,Fc/3)
y_downsampled3=downsample(y_filtered,5);
sound(y_downsampled3,Fc/5)
y_downsampled4=downsample(y_filtered,10);
sound(y_downsampled4,Fc/10)
```

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

```
%Part 7- Without anti-aliasing filter
y_downsampled5=downsample(y,10);
sound(y_downsampled4,Fc/10)
```

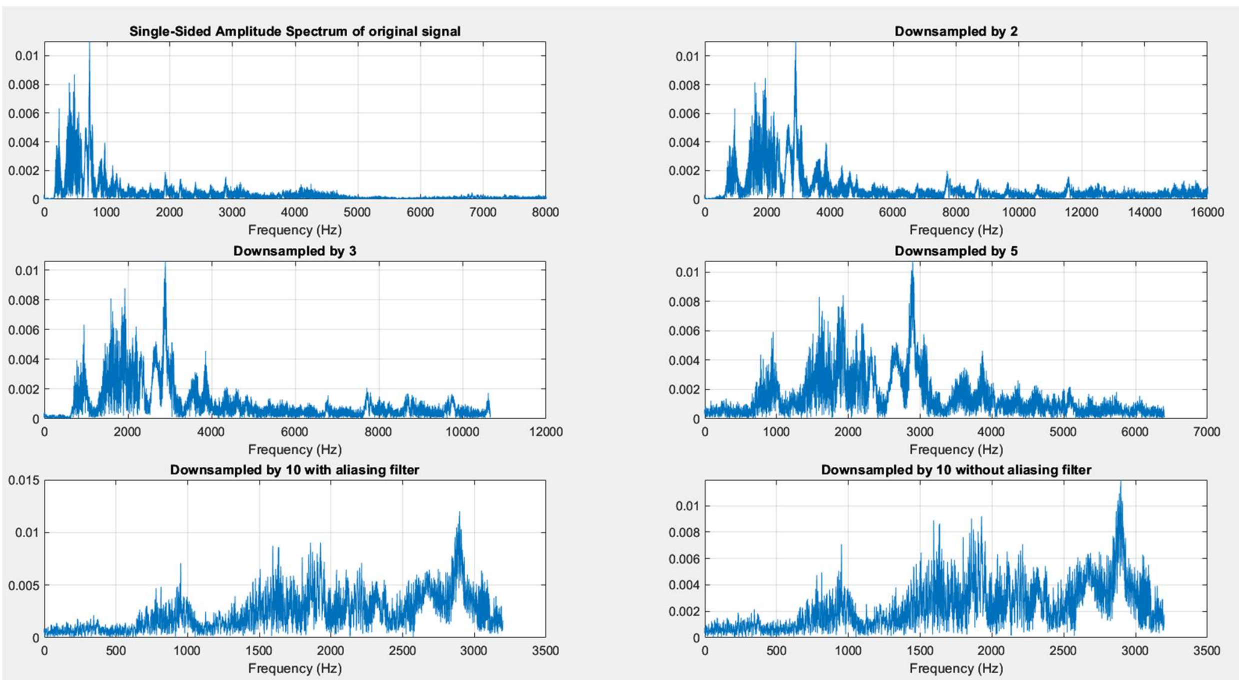
Code for plotting:

```
%Plots
L=zeros(1,6);

%For original signal
L(1)=length(y);
NFFT = 2^nextpow2(L(1));% Next power of 2 from length of y
Y = fft(y,NFFT)/L(1);
f = Fc/2*linspace(0,1,NFFT/2+1);
```

```
subplot(321)
plot(f,2*abs(Y(1:NFFT/2+1))),grid on
title('Single-Sided Amplitude Spectrum of original signal')
xlabel('Frequency (Hz)')

%For Downsampling by 2
L(2)=length(y_downsampled1);
NFFT = 2^nextpow2(L(2));% Next power of 2 from length of y
Y1 = fft(y_downsampled1,NFFT)/L(2);
f = Fc/2*2*linspace(0,1,NFFT/2+1);

subplot(322)
plot(f,2*abs(Y1(1:NFFT/2+1))),grid on
title('Downsampled by 2')
xlabel('Frequency (Hz)')

%For Downsampling by 3
L(3)=length(y_downsampled2);
NFFT = 2^nextpow2(L(3));% Next power of 2 from length of y
Y2 = fft(y_downsampled2,NFFT)/L(3);
f = Fc/3*2*linspace(0,1,NFFT/2+1);

subplot(323)
plot(f,2*abs(Y2(1:NFFT/2+1))),grid on
title('Downsampled by 3')
xlabel('Frequency (Hz)')

%For Downsampling by 5
L(4)=length(y_downsampled3);
NFFT = 2^nextpow2(L(4));% Next power of 2 from length of y
Y3 = fft(y_downsampled3,NFFT)/L(4);
f = Fc/5*2*linspace(0,1,NFFT/2+1);

subplot(324)
plot(f,2*abs(Y3(1:NFFT/2+1))),grid on
title('Downsampled by 5')
xlabel('Frequency (Hz)')

%For Downsampling by 10
L(5)=length(y_downsampled4);
NFFT = 2^nextpow2(L(5));% Next power of 2 from length of y
Y4 = fft(y_downsampled4,NFFT)/L(5);
f = Fc/10*2*linspace(0,1,NFFT/2+1);

subplot(325)
plot(f,2*abs(Y4(1:NFFT/2+1))),grid on
title('Downsampled by 10 with aliasing filter')
xlabel('Frequency (Hz)')

%For Downsampling by 10
L(6)=length(y_downsampled5);
```

```
NFFT = 2^nextpow2(L(6));% Next power of 2 from length of y
Y5 = fft(y_downsampled5,NFFT)/L(6);
f = Fc/10*2*linspace(0,1,NFFT/2+1);

subplot(326)
plot(f,2*abs(Y5(1:NFFT/2+1))),grid on
title('Downsampled by 10 without aliasing filter')
xlabel('Frequency (Hz)')
```

Plots:



Conclusion

The lab demonstrates how to create a low-pass filter with a certain cutoff frequency, which is essential for preventing aliasing in the down-sampled signal. The lab also discusses the trade-offs between signal quality, sampling rate, and bit depth, and provides practical examples of how to optimize these parameters for different types of audio signals.