ECE 161B: LAB 0
DISCRETE-TIME DOMAIN SIGNAL IN MATLAB

ECE 161B — WINTER 2020
May 27, 2020

Roy Kim
A13804992
Electrical and Computer Engineering
UC San Diego

Contents

1 Introduction

In this lab we will generate a discrete-time domain signal using MATLAB and analyze its power spectrum in the frequency domain. We will learn about **spectral leakage** and **signal aliasing**. We will observe aliasing due to under-sampling and how this relates to the Nyquist-Shannon sampling theorem, the minimum required sample frequency to maintain all the signal information is 2 times the frequency of the highest component. We will also write our signal into a ".wav" file, operate on the file, and read the file back into MATLAB.

2 Objectives

1. Generate a discrete-time domain signal in MATLAB
2. Write the result to file
3. Verify the results of your code in MATLAB

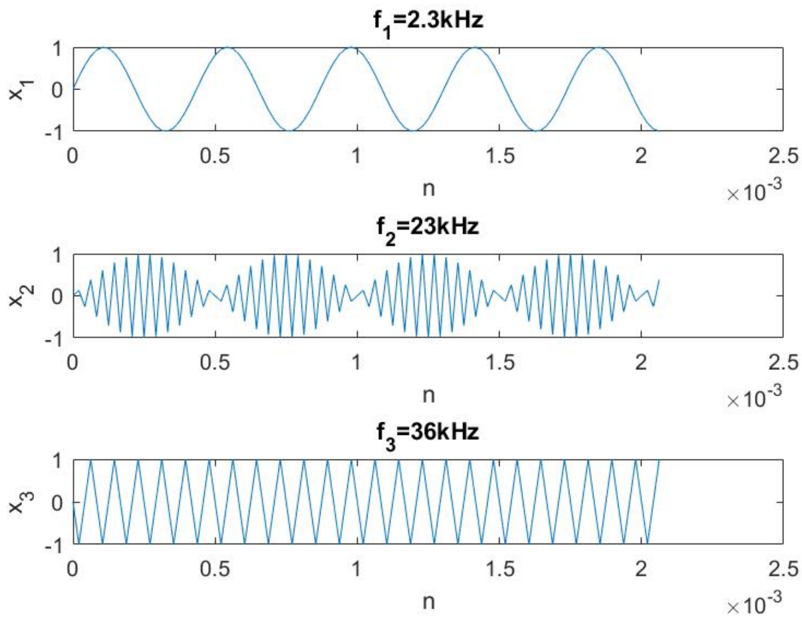
3 Results

3.1 Task 1.

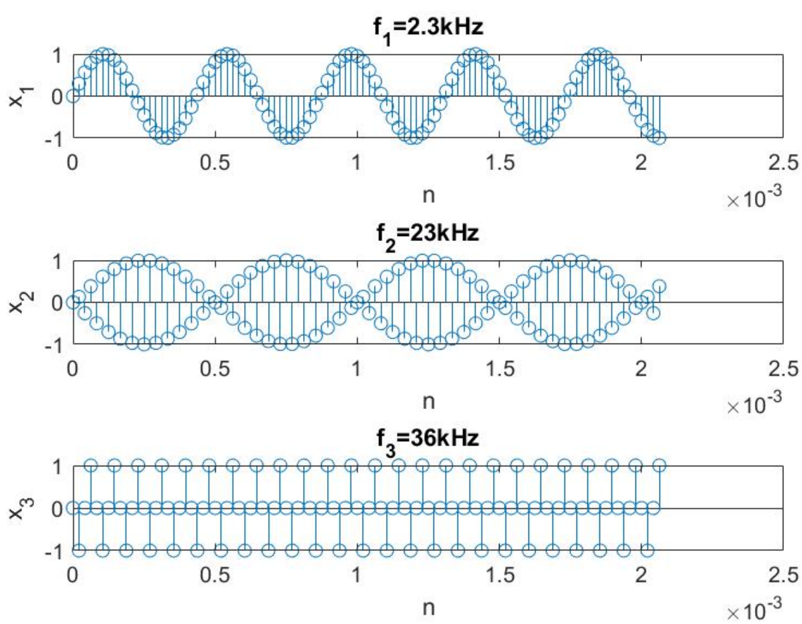
Sampling sinusoids,

$$\begin{aligned} x_1 &= sin(2\pi f_1 n) \qquad & f_1 &= 2.3kHz \\ x_2 &= sin(2\pi f_2 n) \qquad & f_2 &= 23kHz \\ x_3 &= sin(2\pi f_3 n) \qquad & f_3 &= 36kHz \end{aligned}$$

Continuous waveform plot,



Using stem() to get 100-discrete samples, we observe that the waveform looks as below:



Looking at the stem plot, we can easily see the 100 points of each sinusoid–given that $L = 100$.

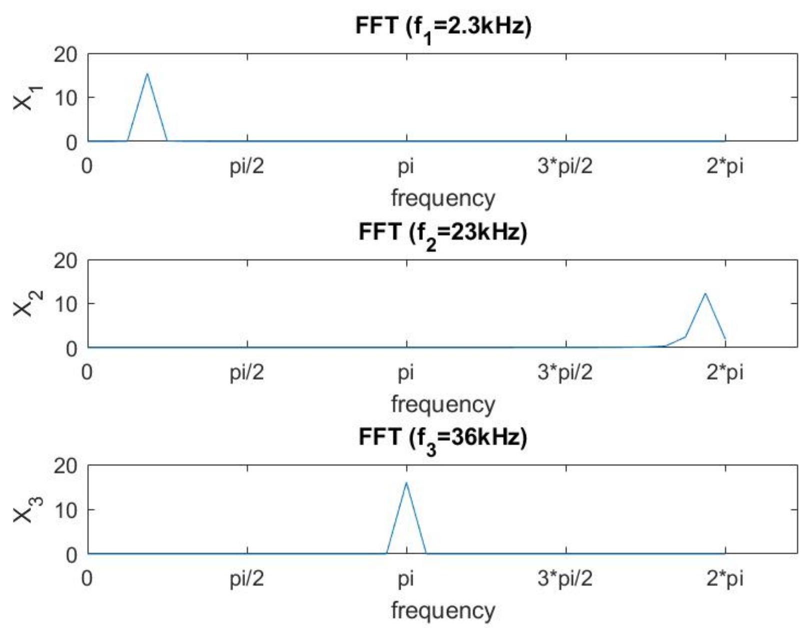
From observation the 2.3kHz sinusoid has a longer wavelength than the 36kHz sinusoid, and this we expect because wavelength is inversely proportional to frequency $\lambda = \frac{1}{f}$. So the higher frequency will have a shorter wavelength. The 23kHz sinusoid looks like an amplituded modulated signal. As if two sinusoids have been added together in such a way that the waveform looks like beats; where there is constructive interference at the peaks and destructive interference at the nodes. Initially, I was expecting to see a sine wave similar to signals x_1 and x_3 but with a wavelength in between the two.

3.2 Task 2.

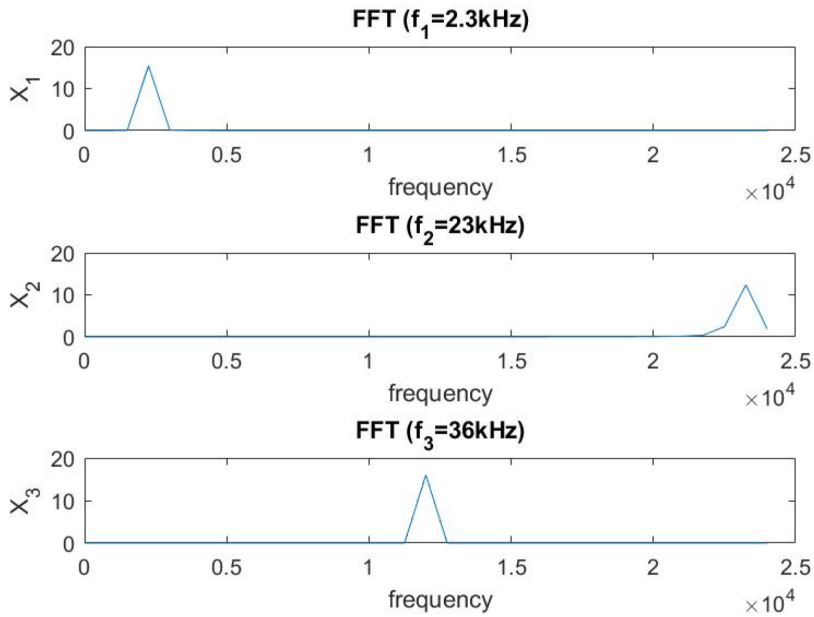
$$
\begin{aligned}
X_1 &= FFT(x_1) & f_1 &= 2.3kHz \\
X_2 &= FFT(x_2) & f_2 &= 23kHz \\
X_3 &= FFT(x_3) & f_3 &= 36kHz
\end{aligned}
$$

X_1, X_2, X_3 are power spectrums of their corresponding signals.

64-Fast Fourier Transform in radians,



For analysis purpose I will observe the power spectral densities in frequency. FFT in frequency,



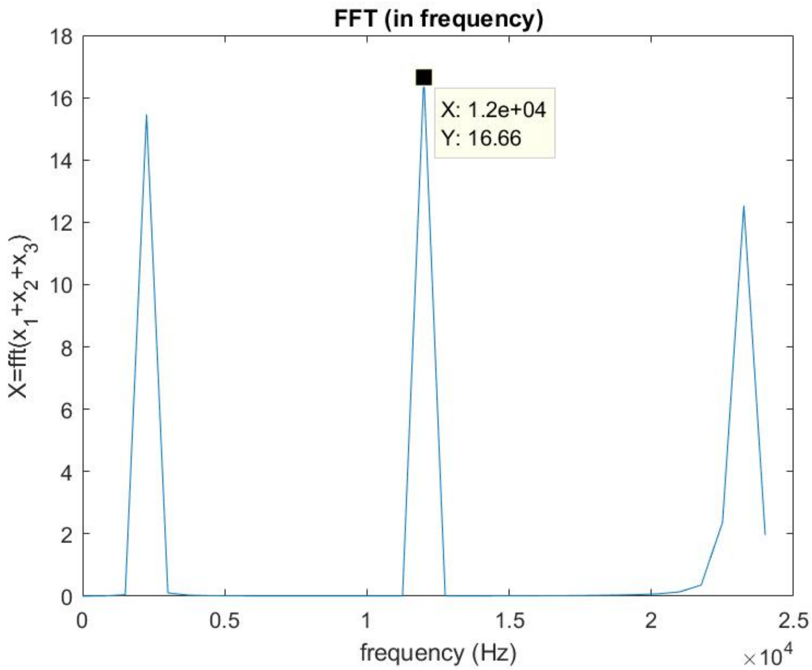
From the fft plot in frequency, we can observe that x_1 has a frequency of 2.3kHz, x_2 has a frequency of 23kHz, and x_3 has a frequency that is not 36kHz (between 10kHz and 15kHz); this is due to the under-sampling of signal x_3 caused by the sampling rate (discussed in detail in **Task 4**). However, we do expect to see three distinct peaks in the FFT plot since we have three distinct sinusoids.

When taking the fft() of the signals, we plot half the points since we know that sine and cosine have a positive and a negative frequency component in the frequency domain.

$$cos(\omega_0 t) = \pi[\delta(\omega - \omega_0) + \delta(\omega + \omega_0)]$$
$$sin(\omega_0 t) = \frac{\pi}{2}[\delta(\omega - \omega_0) - \delta(\omega + \omega_0)]$$

From this Fourier Transform, we expect to see delta functions in the frequency domain analysis. However in our observation, we do not see delta functions. And this is due to **spectral leakage** (discussed in **Task 3**) and how MATLAB handles windowing and bins.

FFT in frequency plotted together,



Here we can better see that signal x_3 has been undersampled. It's fft shows that x_3 has a frequency component of 1.2kHz when the original signal is 36kHz.

3.3 Task 3.

Explain why the magnitude plots are not delta functions.

1. `https://flylib.com/books/en/2.729.1/dft_leakage.html`
2. `https://dspillustrations.com/pages/posts/misc/spectral-leakage-zero-padding-and-frequency-resolution.html`

$$F_{analysis}(k) = k * \frac{f_s}{N} \tag{1}$$

The magnitude plots of the fft() are not delta functions due to DFT leakage. This is because the "input sequence does not have an integral number of cycles over the N-sampled DFT interval, so the input energy has leaked into all the other DFT output bins."
"the analytical frequencies always have an integral number of cycles over our total sample interval of 64 points."
"the DFT assumes that its input signal is one period of a periodic signal, its output are the discrete frequencies of this periodic signal" (1)

Because the DFT assumes a periodic repetition of the signal, we can see from **Task 1** that our signals are not a complete period with length $L = 100$. So there will be discontinuities between the transistions since we did not capture a complete period. So the DFT will not see a pure sinusoidal wave, so we do not see a delta function. This is an example of **spectral leakage**.

"**spectral leakage** – even though the signal x(t) is a periodic signal of frequency f_0, if we take a part of the signal and calculate the DFT spectrum from it, we see multiple frequencies occuring, due to the strange behaviour at the period's boundary" (2)

If we were to measure an integer multiple of the signal period, then we would observe that the leakage would disappear, since the fft() will see a complete periodic signal.

3.4 Task 4.

Describe how the plots in Task 2 relate to the famous Nyquist-Shannon sampling theorem. (If there is aliasing, at what frequency is it showing it in the spectrum and why?)

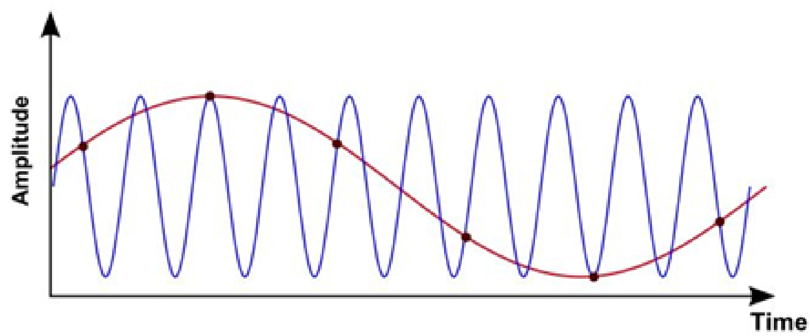
Nyquist-Shannon sampleing theorem:

`http://195.134.76.37/applets/AppletNyquist/Appl_Nyquist2.html`

"The minimum sampleing frequency of a signal that it will not distort its underlying information, should be double the frequency of its highest frequency component."

"If f_s is the sampling frequency, then the critical frequency (or Nyquist limit) f_N is defined as equal to $\frac{f_s}{2}$."

The plots in **Task 2** show the frequency domain of the signals x_1, x_2, and x_3. There is aliasing on signal x_3. The original sinusoid has a frequency of 36kHz, but due to aliasing, the 64-fft shows that x_3 has a frequency of 12kHz. This means that x_3 with a frequency of $f_3 = 36kHz$ has been **under-sampled or distorted**, and we can say x_3 **is an aliased signal due to undersampling.**



This can be explained by the Nyquist-Shannon sampling theorem, that says *"The minumum sampling frequency of a signal that it will not distort its underlying information, should be double the frequency of its highest frequency component."*

$$x_1 = sin(2\pi f_1 n) \qquad f_1 = 2.3kHz$$
$$x_2 = sin(2\pi f_2 n) \qquad f_2 = 23kHz$$
$$x_3 = sin(2\pi f_3 n) \qquad f_3 = 36kHz$$

$F_s = 48kHz$ should be used for frequency components $\leq 24kHz$ signals, so x_1 and x_2 will retain their signal information; frequency components higher than

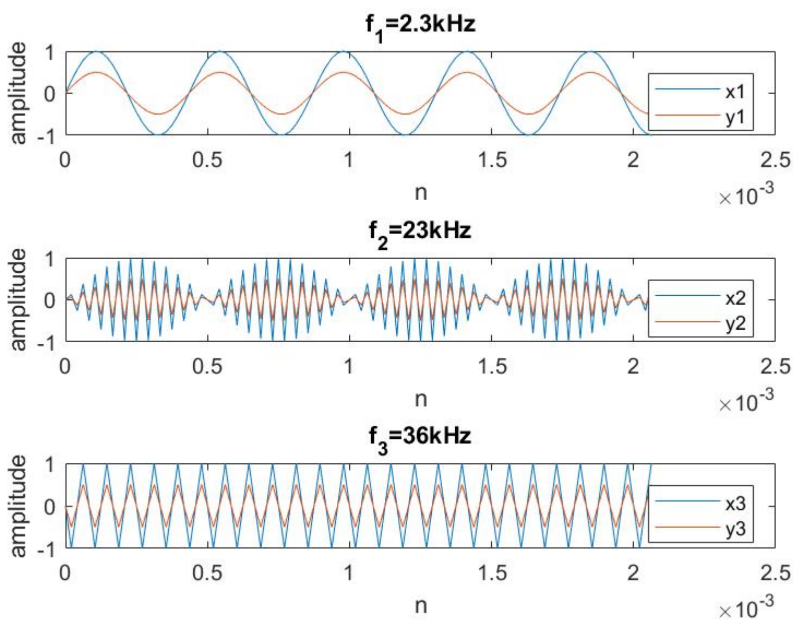
24kHz will be aliased–as seen with x_3 which has a frequency component of 36kHz.

The minimum sampling rate to maintain the original signal of all x_1, x_2, and x_3 is $F_N = 2 \times max(f_1, f_2, f_3) = 2 \times 36000 = 72kHz$

3.5 Task 5.

Used **audiowrite()** and to write each signal to a wav file. Using the c-file, I generated a c-skeleton that reduced the amplituded of the signals by one half in an output wav file. I then read the signal back into MATLAB using **audioread()**.

Plotting the processed signals from the c-generated file,



x is the original signal
y is the modified signal

The c-generated wav files show an output signal that has a max amplitude of one half. This verifies that the c-skeleton did reduce the amplitudes of each signal by half. Without changing the shape of the waveform, we reduced the amplitudes by dividing by two each component of the signal, element by element wise.

4 Code Appendix

4.1 MATLAB Code:

```
%% Task 1
clear all; close all; clc
Fs=48000; %48kHz, sample rate
L=100; %100, number of samples
f=[2300 23000 36000]; %2.3kHz, 23kHz, 36kHz

%sinusoids
n=0:(1/Fs):((L/Fs)-(1/Fs));
x1=sin(2*pi*f(1)*n);
x2=sin(2*pi*f(2)*n);
x3=sin(2*pi*f(3)*n);

figure;
subplot(3,1,1);
plot(n,x1); title('f_1=2.3kHz'); ylabel('x_1'); xlabel('n');
subplot(3,1,2);
plot(n,x2); title('f_2=23kHz'); ylabel('x_2'); xlabel('n');
subplot(3,1,3);
plot(n,x3); title('f_3=36kHz'); ylabel('x_3'); xlabel('n');

figure;
subplot(3,1,1);
stem(n,x1); title('f_1=2.3kHz'); ylabel('x_1'); xlabel('n');
subplot(3,1,2);
stem(n,x2); title('f_2=23kHz'); ylabel('x_2'); xlabel('n');
subplot(3,1,3);
stem(n,x3); title('f_3=36kHz'); ylabel('x_3'); xlabel('n');

%% task 1 test 1
t=0:(1/Fs):((L/Fs)-(1/Fs));
x1=sin(2*pi*f(1)*t);
figure;
stem(t,x1)

%% task 1 test 2
w=[2*pi*f(1) 2*pi*f(2) 2*pi*f(3)]; %w=2*pi*f rad/s
n=0:1/Fs:((L/Fs)-(1/Fs));
figure;
stem(n,sin(n*w(1)))

```

```
%% task 1 test 3
Fs = 48000; % Sampling frequency
T = 1/Fs; % Sampling period
L = 100; % Length of signal
t = (0:L-1)*T; % Time vector

signal=sin(2*pi*f(1)*t);
stem(t,signal)
%concl.: all same methods

%% Task 2 test
clear all; close all; clc
Fs=48000; %48kHz, sample rate
L=100; %100, number of samples
f=[2300 23000 36000]; %2.3kHz, 23kHz, 36kHz

%sinusoids
n=0:(1/Fs):((L/Fs)-(1/Fs));
x1=sin(2*pi*f(1)*n);
x2=sin(2*pi*f(2)*n);
x3=sin(2*pi*f(3)*n);
x=x1+x2+x3;

%signals x1, x2, x3
%L=64; %look here
X=fft(x,64); %64point-fft
P2=abs(X/L);
P1=P2(1:L/2+1);
P1(2:end-1)=2*P1(2:end-1);
freq = Fs*(0:(L/2))/L;
plot(freq,P1);
%xlim([0 4000]);

%mapping (0-48000) to (0-2pi)

%% Task 2 fft() in frequency
clear all; close all; clc
Fs=48000; %48kHz, sample rate
L=100; %100, number of samples
f=[2300 23000 36000]; %2.3kHz, 23kHz, 36kHz

%sinusoids
n=0:(1/Fs):((L/Fs)-(1/Fs));
x1=sin(2*pi*f(1)*n);
x2=sin(2*pi*f(2)*n);
x3=sin(2*pi*f(3)*n);
```

```
N=64;

X1=fft(x1,N);
X2=fft(x2,N);
X3=fft(x3,N);

P1=X1.*conj(X1)/N;
P2=X2.*conj(X2)/N;
P3=X3.*conj(X3)/N;
f=Fs*(0:0.5*N)/N;

figure;
subplot(3,1,1);
plot(f,P1(1:0.5*N+1)); title('FFT (f_1=2.3kHz)'); ylabel('X_1'); xlabel('frequency');
subplot(3,1,2);
plot(f,P2(1:0.5*N+1)); title('FFT (f_2=23kHz)'); ylabel('X_2'); xlabel('frequency');
subplot(3,1,3);
plot(f,P3(1:0.5*N+1)); title('FFT (f_3=36kHz)'); ylabel('X_3'); xlabel('frequency');

%% Task 2 fft() in radians
clear all; close all; clc
Fs=48000; %48kHz, sample rate
L=100; %100, number of samples
f=[2300 23000 36000]; %2.3kHz, 23kHz, 36kHz

%sinusoids
n=0:(1/Fs):((L/Fs)-(1/Fs));
x1=sin(2*pi*f(1)*n);
x2=sin(2*pi*f(2)*n);
x3=sin(2*pi*f(3)*n);

N=64;

X1=fft(x1,N);
X2=fft(x2,N);
X3=fft(x3,N);

P1=X1.*conj(X1)/N;
P2=X2.*conj(X2)/N;
P3=X3.*conj(X3)/N;
f=Fs*(0:0.5*N)/N;
%rad=f*2*pi/size(f,2);
```

```
rad=linspace(0,2*pi,size(f,2)); %# of rad points = # freq points

figure;
subplot(3,1,1);
plot(rad,P1(1:0.5*N+1)); title('FFT (f_1=2.3kHz)'); ylabel('X_1'); xlabel('frequency');
set(gca,'XTick',0:pi/2:2*pi)
set(gca,'XTickLabel',{'0','pi/2','pi','3*pi/2','2*pi'})
subplot(3,1,2);
plot(rad,P2(1:0.5*N+1)); title('FFT (f_2=23kHz)'); ylabel('X_2'); xlabel('frequency');
set(gca,'XTick',0:pi/2:2*pi)
set(gca,'XTickLabel',{'0','pi/2','pi','3*pi/2','2*pi'})
subplot(3,1,3);
plot(rad,P3(1:0.5*N+1)); title('FFT (f_3=36kHz)'); ylabel('X_3'); xlabel('frequency');
set(gca,'XTick',0:pi/2:2*pi)
set(gca,'XTickLabel',{'0','pi/2','pi','3*pi/2','2*pi'})
%% Task 2 FFT(x1+x2+x3) in frequency
clear all; close all; clc
Fs=48000; %48kHz, sample rate
L=100; %100, number of samples
f=[2300 23000 36000]; %2.3kHz, 23kHz, 36kHz

%sinusoids
n=0:(1/Fs):((L/Fs)-(1/Fs));
x1=sin(2*pi*f(1)*n);
x2=sin(2*pi*f(2)*n);
x3=sin(2*pi*f(3)*n);
x=x1+x2+x3;

N=64; %64 point-fft
x=x1+x2+x3;
X=fft(x,N); %N point-fft
Pxx=X.*conj(X)/N;
f=Fs*(0:0.5*N)/N;
figure;
plot(f,Pxx(1:0.5*N+1));
title('FFT (in frequency)'); xlabel('frequency (Hz)'); ylabel('X=fft(x_1+x_2+x_3)');

%% Task 5
clear all; close all; clc
Fs=48000; %48kHz, sample rate
L=100; %100, number of samples
f=[2300 23000 36000]; %2.3kHz, 23kHz, 36kHz
```

```

%sinusoids
n=0:(1/Fs):((L/Fs)-(1/Fs));
x1=sin(2*pi*f(1)*n);
x2=sin(2*pi*f(2)*n);
x3=sin(2*pi*f(3)*n);

%% write to WAV file
filename1='signal_1.wav';
audiowrite(filename1,x1,Fs);

filename2='signal_2.wav';
audiowrite(filename2,x2,Fs);

filename3='signal_3.wav';
audiowrite(filename3,x3,Fs);

% on linux terminal run:
% gcc -lm -o skeleton Lab0.c $(pkg-config sndfile --cflags --libs
    ↪ )
% ./skeleton signal_1.wav output_1.wav

%% read output file into MATLAB
[y1,Fs1] = audioread('output_1.wav');
[y2,Fs2] = audioread('output_2.wav');
[y3,Fs3] = audioread('output_3.wav');

% plot
figure;
subplot(3,1,1);
plot(n,x1); hold on;
plot(n,y1);title('f_1=2.3kHz'); ylabel('amplitude'); xlabel('n');
legend('x1','y1'); hold off;
subplot(3,1,2);
plot(n,x2); hold on;
plot(n,y2); title('f_2=23kHz'); ylabel('amplitude'); xlabel('n');
legend('x2','y2'); hold off;
subplot(3,1,3);
plot(n,x3); hold on;
plot(n,y3); title('f_3=36kHz'); ylabel('amplitude'); xlabel('n');
legend('x3','y3'); hold off;


%% example time domain
%http://matlab.izmiran.ru/help/techdoc/ref/fft.html
t = 0:0.001:0.6;
```

```
x = sin(2*pi*50*t)+sin(2*pi*120*t);
y = x + 2*randn(size(t));
plot(1000*t(1:50),y(1:50))
title('Signal Corrupted with Zero-Mean Random Noise')
xlabel('time (milliseconds)')

%% example frequency domain
Y = fft(y,512);
Pyy = Y.* conj(Y) / 512;
f = 1000*(0:256)/512;
plot(f,Pyy(1:257))
title('Frequency content of y')
xlabel('frequency (Hz)')
```

4.2 C Code:

```c
#include <stdlib.h>
#include <stdio.h>
#include <float.h>
//#include "wave.h"
#include <sndfile.h>
#include <math.h>

#define PI 3.14159265

int main(int argc, char *argv[])
{
    int ii;

    //Require 2 arguments: input file and output file
    if(argc < 3)
    {
        printf("Not enough arguments \n");
        return -1;
    }

    SF_INFO sndInfo;
    SNDFILE *sndFile = sf_open(argv[1], SFM_READ, &sndInfo);
    if (sndFile == NULL) {
        fprintf(stderr, "Error reading source file '%s': %s\n", argv[1], sf_strerror(sndFile));
        return 1;
    }

    SF_INFO sndInfoOut = sndInfo;
    sndInfoOut.format = SF_FORMAT_WAV | SF_FORMAT_PCM_16;
    sndInfoOut.channels = 1;
    sndInfoOut.samplerate = sndInfo.samplerate;
    SNDFILE *sndFileOut = sf_open(argv[2], SFM_WRITE, &sndInfoOut);

    // Check format - 16bit PCM
    if (sndInfo.format != (SF_FORMAT_WAV | SF_FORMAT_PCM_16)) {
        fprintf(stderr, "Input should be 16bit Wav\n");
        sf_close(sndFile);
        return 1;
    }

```

```
    // Check channels - mono
    if (sndInfo.channels != 1) {
        fprintf(stderr, "Wrong number of channels\n");
        sf_close(sndFile);
        return 1;
    }

    // Allocate memory
    float *buffer = malloc(sizeof(double));
    if (buffer == NULL) {
        fprintf(stderr, "Could not allocate memory for file\n");
        sf_close(sndFile);
        return 1;
    }

    // Load data
    for(ii=0; ii < sndInfo.frames; ii++)
    {
        sf_readf_float(sndFile, buffer, 1);
                //Do something to the variable buffer here
                    //buffer[ii]=buffer[ii]/2;
                    *buffer = *buffer/2;
        sf_writef_float(sndFileOut, buffer, 1);
    }

    sf_close(sndFile);
    sf_write_sync(sndFileOut);
    sf_close(sndFileOut);
    free(buffer);

    return 1;
}
```

4.3 LaTeX Code:

```
\documentclass{article}
\usepackage[utf8]{inputenc}
\usepackage{graphicx}
\usepackage{hyperref}
%\usepackage[a4paper,width=150mm,top=1in,bottom=1in]{geometry}
%\usepackage[a4paper,margin=1in]{geometry}
\usepackage{indentfirst}

\usepackage{amsmath}
\pagenumbering{arabic}
\usepackage{subcaption}
\usepackage[numbered]{mcode} %using mcode.sty to convert .m file
    ↪ code to latex format
\usepackage{listings}
\usepackage{adjustbox}
\usepackage{minted}
\graphicspath{{./images/}}

\lstset{
  basicstyle=\ttfamily,
  columns=fullflexible,
  frame=single,
  breaklines=true,
  postbreak=\mbox{\textcolor{red}{$\hookrightarrow$}\space},
}

\begin{document}

\input{titlepage}

%
    ↪ --------------------------------------------------------------------------------
    ↪
% Table of contents
%
    ↪ --------------------------------------------------------------------------------
    ↪
\hspace{0pt}
\vfill
\tableofcontents
\vfill
\hspace{0pt}
```

```
\newpage
% ----------------------------------------------------------------------------------
% Content
% ----------------------------------------------------------------------------------
\section{Introduction}
    In this lab we will generate a discrete-time domain signal using MATLAB and analyze its power spectrum in the frequency domain. We will learn about \textbf{spectral leakage} and \textbf{signal aliasing}. We will observe aliasing due to under-sampling and how this relates to the Nyquist-Shannon sampling theorem, the minimum required sample frequency to maintain all the signal information is 2 times the frequency of the highest component. We will also write our signal into a ".wav" file, operate on the file, and read the file back into MATLAB.

% \begin{figure}[h!]
% \centering
% \includegraphics[scale=1.7]{universe}
% \caption{The Universe}
% \label{fig:universe}
% \end{figure}

\section{Objectives}
    \begin{enumerate}
      \item Generate a discrete-time domain signal in MATLAB
      \item Write the result to file
      \item Verify the results of your code in MATLAB
    \end{enumerate}

\section{Results}
    \subsection{Task 1.} Sampling sinusoids,
        \begin{align*}
            x_1&=sin(2 \pi f_1 n) & f_1&=2.3kHz\\
            x_2&=sin(2 \pi f_2 n) & f_2&=23kHz\\
            x_3&=sin(2 \pi f_3 n) & f_3&=36kHz
        \end{align*}
        Continuous waveform plot,
        \flushleft\includegraphics[width=\textwidth]{task1b.jpg}
        Using stem() to get 100-discrete samples, we observe that the waveform looks as below:
```

```
        \flushleft\includegraphics[width=\textwidth]{task1a.jpg}
        Looking at the stem plot, we can easily see the 100 points of each sinusoid--given that $L=100$.\\
        \vspace{5mm}
        From observation the 2.3kHz sinusoid has a longer wavelength than the 36kHz sinusoid, and this we expect because wavelength is inversely proportional to frequency $\lambda=\frac{1}{f}$. So the higher frequency will have a shorter wavelength.\\
        The 23kHz sinusoid looks like an amplituded modulated signal. As if two sinusoids have been added together in such a way that the waveform looks like beats; where there is constructive interference at the peaks and destructive interference at the nodes. Initially, I was expecting to see a sine wave similar to signals $x_1$ and $x_3$ but with a wavelength in between the two.

    \subsection{Task 2.}
        \begin{align*}
            X_1&=FFT(x_1) & f_1&=2.3kHz\\
            X_2&=FFT(x_2) & f_2&=23kHz\\
            X_3&=FFT(x_3) & f_3&=36kHz
        \end{align*}
        $X_1$, $X_2$, $X_3$ are power spectrums of their corresponding signals.\\
        \vspace{5mm}
        64-Fast Fourier Transform in radians,
        \includegraphics[width=\textwidth]{task2b.jpg}
        \newpage
        For analysis purpose I will observe the power spectral densities in frequency.\\
        FFT in frequency,
        \includegraphics[width=\textwidth]{task2a.jpg}
        From the fft plot in frequency, we can observe that $x_1$ has a frequency of 2.3kHz, $x_2$ has a frequency of 23kHz, and $x_3$ has a frequency that is not 36kHz (between 10kHz and 15kHz); this is due to the under-sampling of signal $x_3$ caused by the sampling rate (discussed in detail in \textbf{Task 4}). However, we do expect to see three distinct peaks in the FFT plot since we have three distinct sinusoids.\\
        \vspace{5mm}
        When taking the fft() of the signals, we plot half the points since we know that sine and cosine have a
```

```
        positive and a negative frequency component in the frequency domain.
        \begin{align*}
            cos(\omega_0 t)=\pi[\delta(\omega-\omega_0)+\delta(\omega+\omega_0)]\\
            sin(\omega_0 t)=\frac{\pi}{2}[\delta(\omega-\omega_0)-\delta(\omega+\omega_0)]
        \end{align*}
        From this Fourier Transform, we expect to see delta functions in the frequency domain analysis. However in our observation, we do not see delta functions. And this is due to \textbf{spectral leakage} (discussed in \textbf{Task 3}) and how MATLAB handles windowing and bins.
        \newpage
        FFT in frequency plotted together,
        \includegraphics[width=\textwidth]{task2.jpg}
        Here we can better see that signal $x_3$ has been undersampled. It's fft shows that $x_3$ has a frequency component of 1.2kHz when the original signal is 36kHz.
    \subsection{Task 3.} Explain why the magnitude plots are not delta functions.\\
        \vspace{5mm}
        \begin{enumerate}
            \item\url{https://flylib.com/books/en/2.729.1/dft_leakage.html}
            \item\url{https://dspillustrations.com/pages/posts/misc/spectral-leakage-zero-padding-and-frequency-resolution.html}
        \end{enumerate}
        \begin{equation}
            F_{analysis}(k)=k*\frac{f_s}{N}
        \end{equation}
        The magnitude plots of the fft() are not delta functions due to DFT leakage. This is because the "input sequence does not have an integral number of cycles over the N-sampled DFT interval, so the input energy has leaked into all the other DFT output bins."
        \newline
        "the analytical frequencies always have an integral number of cycles over our total sample interval of 64 points."
        \newline
```

```
        "the DFT assumes that its input signal is one period of a periodic signal, its output are the discrete frequencies of this periodic signal" (1)\\
        \vspace{5mm}
        Because the DFT assumes a periodic repetition of the signal, we can see from \textbf{Task 1} that our signals are not a complete period with length $L=100$. So there will be discontinuities between the transistions since we did not capture a complete period. So the DFT will not see a pure sinusoidal wave, so we do not see a delta function. This is an example of \textbf{spectral leakage}.\\
        \vspace{5mm}
        "\textbf{spectral leakage} -- even though the signal x(t) is a periodic signal of frequency $f_0$, if we take a part of the signal and calculate the DFT spectrum from it, we see multiple frequencies occuring, due to the strange behaviour at the period's boundary" (2)\\
        \vspace{5mm}
        If we were to measure an integer multiple of the signal period, then we would observe that the leakage would disappear, since the fft() will see a complete periodic signal.
    \pagebreak
    \subsection{Task 4.} Describe how the plots in Task 2 relate to the famous Nyquist-Shannon sampling theorem. (If there is aliasing, at what frequency is it showing it in the spectrum and why?)\\
        \vspace{5mm}
        \begin{center}
            \textbf{\Large{Nyquist-Shannon sampleing theorem:}}\\
            \url{http://195.134.76.37/applets/AppletNyquist/Appl_Nyquist2.html}
            "The minimum sampleing frequency of a signal that it will not distort its underlying information, should be double the frequency of its highest frequency component."
            \newline
            "If $f_s$ is the sampling frequency, then the critical frequency (or Nyquist limit) $f_N$ is defined as equal to $\frac{f_s}{2}$."
        \end{center}
        \vspace{5mm}
        The plots in \textbf{Task 2} show the frequency domain of the signals $x_1$, $x_2$, and $x_3$.
```

```
        There is aliasing on signal $x_3$. The original sinusoid has a frequency of 36kHz, but due to aliasing, the 64-fft shows that $x_3$ has a frequency of 12kHz. This means that $x_3$ with a frequency of $f_3=36 kHz$ has been \textbf{under-sampled or distorted}, and we can say \textbf{$x_3$ is an aliased signal due to undersampling.}
        \includegraphics[width=\textwidth]{aliasing.jpg}
        \newline\newline
        This can be explained by the Nyquist-Shannon sampling theorem, that says \textit{"The minumum sampling frequency of a signal that it will not distort its underlying information, should be double the frequency of its highest frequency component."}
        \begin{align*}
            x_1&=sin(2 \pi f_1 n) & f_1&=2.3kHz\\
            x_2&=sin(2 \pi f_2 n) & f_2&=23kHz\\
            x_3&=sin(2 \pi f_3 n) & f_3&=36kHz
        \end{align*}
        $F_s=48kHz$ should be used for frequency components $\leq 24kHz$ signals, so $x_1$ and $x_2$ will retain their signal information; frequency components higher than 24kHz will be aliased--as seen with $x_3$ which has a frequency component of 36kHz.\\
        \vspace{5mm}
        The minimum sampling rate to maintain the original signal of all $x_1$, $x_2$, and $x_3$ is $F_N=2\times max(f_1,f_2,f_3)=2 \times 36000=72kHz$

    \subsection{Task 5.}
        Used \textbf{audiowrite()} and to write each signal to a wav file. Using the c-file, I generated a c-skeleton that reduced the amplituded of the signals by one half in an output wav file. I then read the signal back into MATLAB using \textbf{audioread()}.\\
        \vspace{5mm}
        Plotting the processed signals from the c-generated file,
        \includegraphics[width=\textwidth]{task5.jpg}
        \begin{center}
            $x$ is the original signal\\
            $y$ is the modified signal
        \end{center}
        The c-generated wav files show an output signal that has a max amplitude of one half. This verifies that the c-skeleton did reduce the amplitudes of each signal
```

```
             by half. Without changing the shape of the
            waveform, we reduced the amplitudes by dividing by
            two each component of the signal, element by
            element wise.
\newpage
\section{Code Appendix}
    \subsection{MATLAB Code:}
        %\begin{adjustbox}{max width=\textwidth}
            \lstinputlisting[frame=single]{code-files/lab0.m}
        %\end{adjustbox}
    \newpage
    \subsection{C Code:}
        \lstinputlisting[frame=single]{code-files/Lab0.c}
    \newpage
    \subsection{\LaTeX Code:}
        \lstinputlisting[frame=single]{main.tex}
\end{document}
```