ELECTRICAL ENGINEERING 3TP3

Lab 4 Report

Kevin Le (let36)

Student Number: 400385350

Joy Justin (justinj)

Student Number: 400364157

Submitted: November 5th, 2023

As a future member of the engineering profession, the student is responsible for performing the required work in an honest manner, without plagiarism and cheating. Submitting this work with my name and student number is a statement and understanding that this work is my own and adheres to the Academic Integrity Policy of McMaster University and the Code of Conduct of the Professional Engineers of Ontario. Submitted by

[Kevin Le, 400385350 Joy Justin, justinj, 400364157]

Part I

On playing the *tones2023.wav* file, we hear a high pitched sound and a background lower tone that starts fading in, hinting at an oscillating wave.

Plotting tones2023.wav

```
clear all;

% Read in the signal from the audio file
[signal, Fs] = audioread("tones2023.wav");
T = 1/Fs; % Sampling period; 1/(sampling frequency)
L = length(signal); % Number of points in 'signal'
t = [0:L-1] * T; % Time vector

t_plot = 0.005; % 5 milliseconds
num_of_samples = t_plot/T; % number of samples in 5 msec

% Plotting the first 5 msec of the signal
plot(t(1:num_of_samples), signal(1:num_of_samples));
title('Plot of Input Signal');
xlabel('time (seconds)');
grid('minor');
```

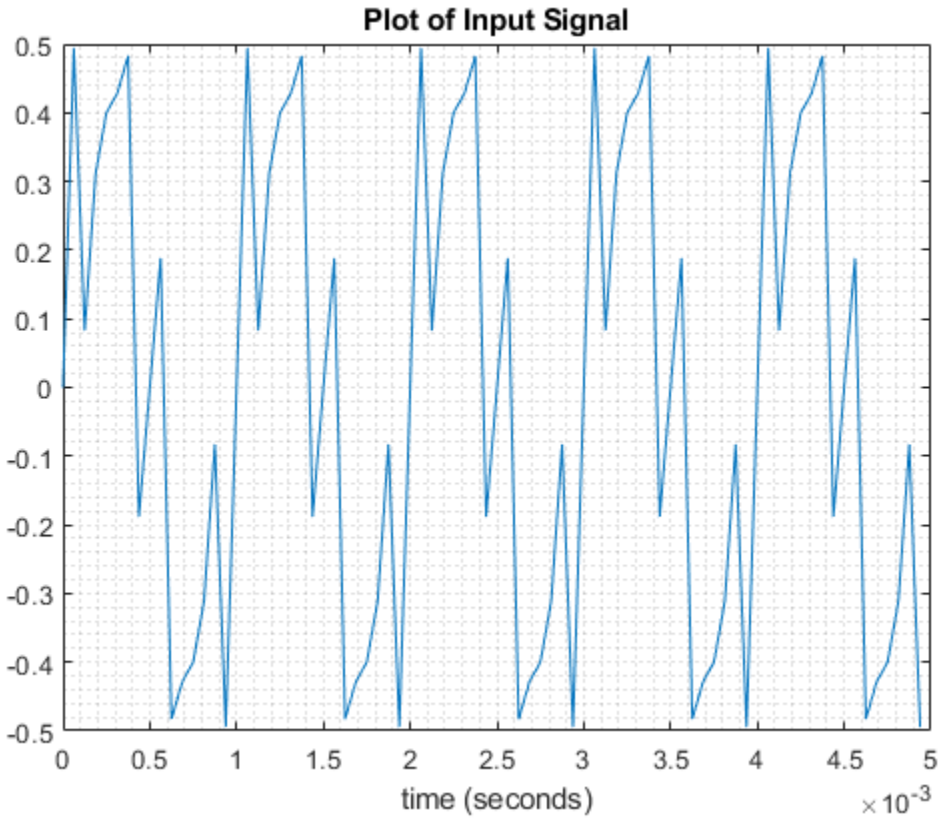

Figure 1: First 5 milliseconds of tones2023.wav, sampled and plotted

Visually observing the plot above, I am unable to tell how many sinusoids make up the signal. However I can estimate that the highest frequency of the sinusoids that make up the signal is around 4000 Hz.

When experimenting on Desmos to find an answer for this question, I was unable to notice any patterns when varying the number of sinusoids of different frequencies in the summation. However, one pattern that I noticed is that if the highest frequency in the summation is, for example, 7000 Hz, I see around seven rises and falls in a 0.001s time period.

So looking at figure 1, in a 1ms time period, I notice four rises and falls, which is why I predicted that the highest frequency in the signal is 4000 Hz.

One flaw with this approach is that if our signal has rises and falls that are close together in time, we may not be able to see it due to the limitation of our sampling frequency.

Plotting the Discrete Fourier Transform

```
clear all;

% Read in the signal from the audio file
[signal, Fs] = audioread("tones2023.wav");
T = 1/Fs; % Sampling period; 1/(sampling frequency)
L = length(signal); % Number of points in 'signal'
t = [0:L-1] * T; % Time vector

Y = fft(signal); % perform DFT

% '(0:L-1)' is cycles per frame; converts to cycles per second
f = (0:L-1)*Fs/L;

f_double_sided = (-L/2:ceil(L/2)-1)*Fs/L;

% plotting the discrete fourier transform
plot(f,abs(Y)/L);
title('Discrete Fourier Transform');
ylabel('Magnitude');
xlabel('Frequency (Hz)');
axis([0 Fs 0 0.5])
grid('minor');

figure()

% plotting the two-sided frequency spectrum (including the negative
% frequencies)
% fftshift shifts points between ceil(L/2)+1 and L to the negative side
plot(f_double_sided, fftshift(abs(Y)/L));
title('Two-Sided Magnitude Spectrum');
ylabel('Magnitude');
xlabel('Frequency (Hz)');
axis([-Fs/2 Fs/2 0 0.5])
grid('minor');
```

```
figure()

% multiply magnitude of the output of DFT by 2/L to get the peak amplitude
A = abs(Y)*2/L;

% plotting only the positive frequencies; the use of '(1:ceil(L/2))'
% below assures that code will work with even or odd number of samples
plot(f(1:ceil(L/2)),A(1:ceil(L/2)));
title('Single-Sided Magnitude Spectrum');
ylabel('Amplitude');
xlabel('Frequency (Hz)');
axis([0 Fs/2 0 0.5])
grid('minor');
```

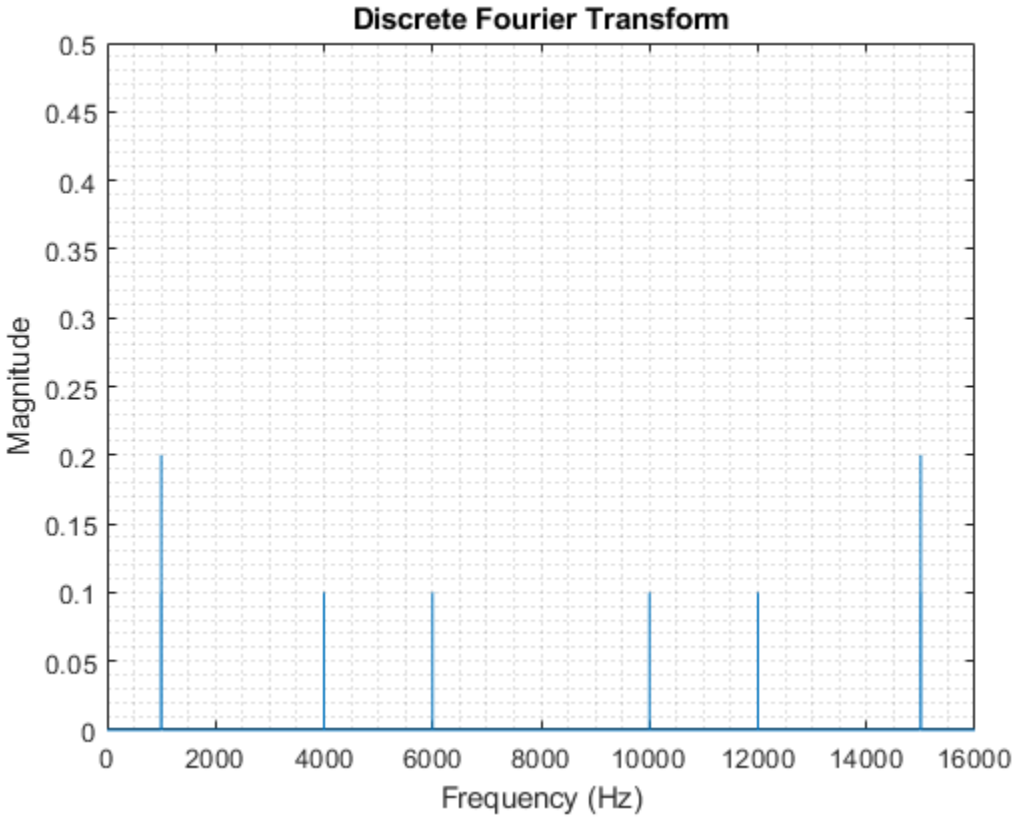

Figure 2: Discrete Fourier Transform of the signal; plotted with fft()

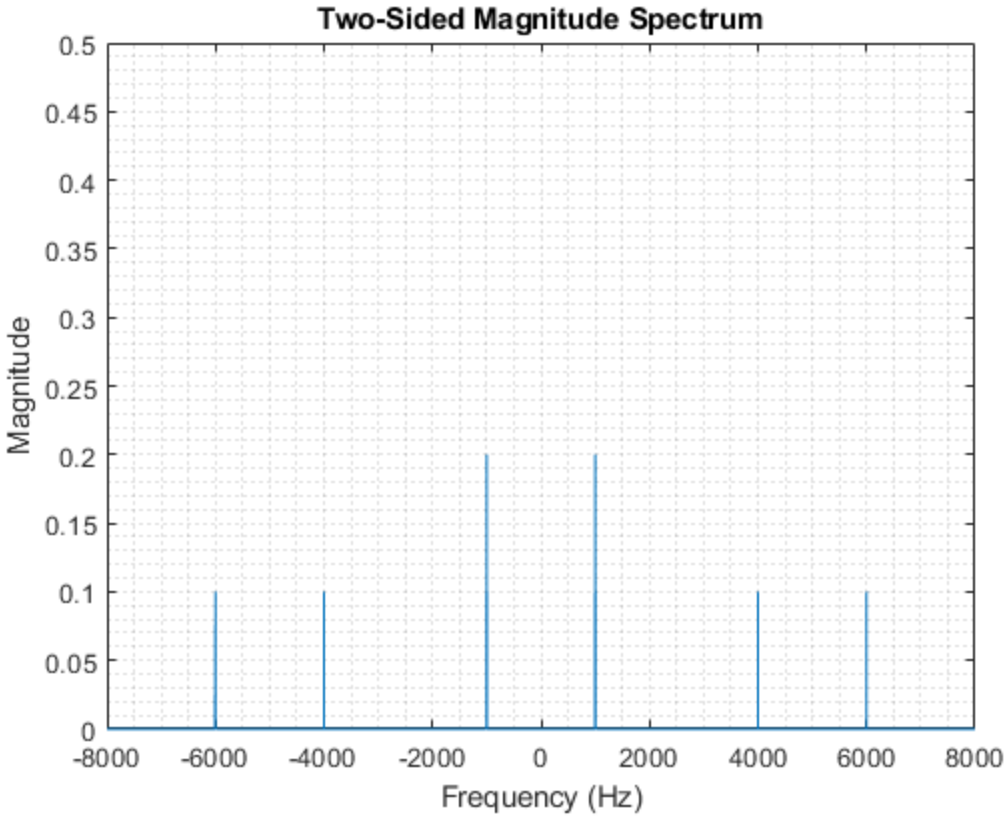

Figure 3: DFT of the signal plotted with fft() and fftshift(); plotting the two-sided magnitude spectrum

The original audio signal is made up of a summation of sinusoids at different frequencies. The discrete fourier transform of the original signal (shown in figure 2) produces a plot that graphs the amplitude of sinusoids with frequencies from 0 Hz to 15,999.9 Hz that make up the signal (not exactly 16,000 Hz, because there are 160,000 samples and sampling frequency is 16,000 Hz, so 0 Hz to 15,999.9 with increment 0.1 has 160,000 samples). In other words, if a signal is made from one sinusoid with frequency 1000 Hz, the plot of the DFT of that signal will have one spike at 1000 Hz. However, that is not all, because if that signal was sampled at 16,000 Hz, there will also be a spike at 15,000 Hz on that DFT graph. This is due to aliasing, as we have seen in a previous lab.

Because the sampling frequency is 16,000 Hz, the Nyquist frequency is 8,000 Hz. As we see in figure 2, the plot is mirrored on 8,000 Hz. As we have discovered in a previous lab, signals with frequencies 1,000 Hz and 15,000 Hz, or 1 Hz and 15,999 Hz, as an example, look the same if sampled at 16,000 Hz because the frequencies are the same distance from the Nyquist frequency of 8,000 Hz.

Because only frequencies up to the Nyquist limit are valid, all frequencies from the Nyquist frequency and above get moved to the negative side. This is done using the fftshift() function. So this plot tells us that the original signal is made from:

0.1*cos(2*pi*-6000*t + phase1) + 0.1*cos(2*pi*-4000*t + phase2) + 0.2*cos(2*pi*-1000*t + phase3) + 0.2*cos(2*pi*1000*t + phase4) + 0.1*cos(2*pi*4000*t + phase5) + 0.1*cos(2*pi*6000*t + phase6)

If we notice the magnitude at 1000 Hz, 4000 Hz and 6000 Hz in figure 3 above compared to the magnitude in figure 4 below, the magnitude from figure 3 is half of the magnitude in figure 4. The reason is because the single-sided magnitude spectrum gets rid of the negative frequencies and represents the frequency components of the original signal with positive frequencies only. Sinusoids with the negative frequency actually represent the one with the positive frequency, so when added with the corresponding positive frequency, their sum becomes just the positive frequency with the amplitude doubled. It is the same as:

0.4*cos(2*pi*1000*t + phase3) + 0.2*cos(2*pi*4000*t + phase2) + 0.2*cos(2*pi*6000*t + phase1)

Notice that the magnitudes are doubled.

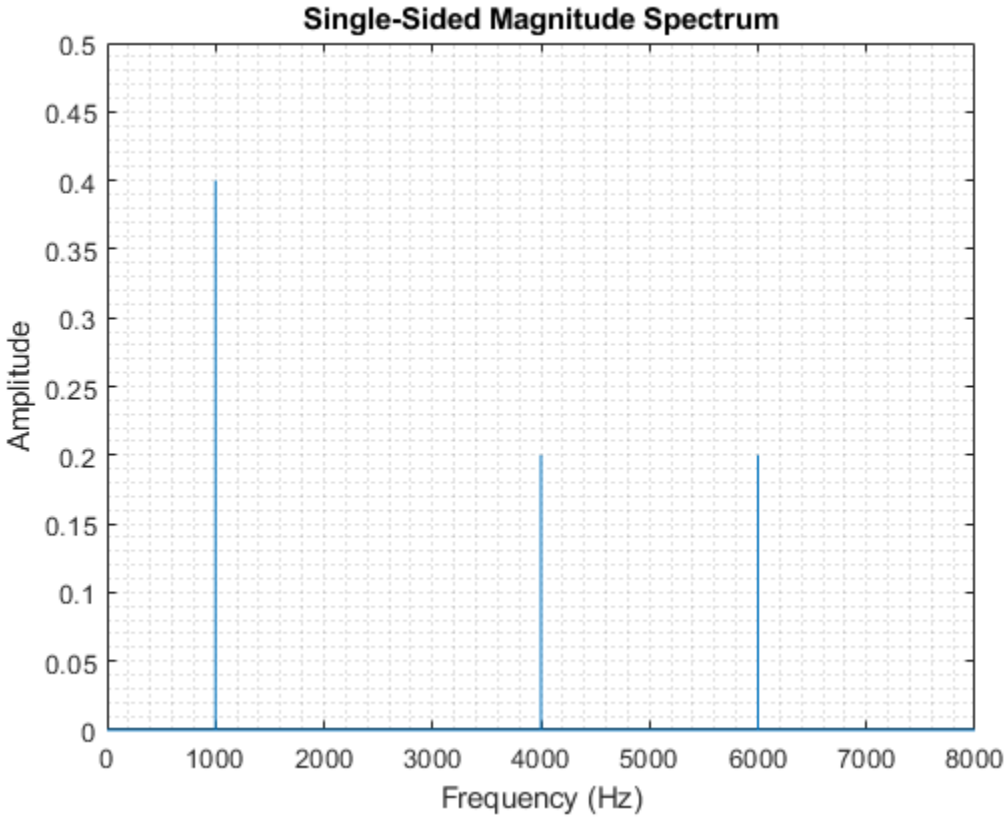

Figure 4: DFT of the signal plotting the single-sided magnitude spectrum

Sinusoids	Frequencies(f)	Amplitudes(A)
1	1000 Hz	0.4
2	4000 Hz	0.2
3	6000 Hz	0.2

Plotting the Discrete Fourier Transform on a Complex Plane

```
clear all;

% Read in the signal from the audio file
[signal, Fs] = audioread("tones2023.wav");
T = 1/Fs; % Sampling period; 1/(sampling frequency)
L = length(signal); % Number of points in 'signal'
t = [0:L-1] * T; % Time vector
Y = fft(signal); % perform DFT

% Plotting Y on a complex plane; x-axis is real, y-axis is imaginary
plot(real(Y(1:1+ceil((L-1)/2))),imag(Y(1:1+ceil((L-1)/2))),"o");
axis equal
grid on
xlabel("Re(z)")
ylabel("Im(z)")
```

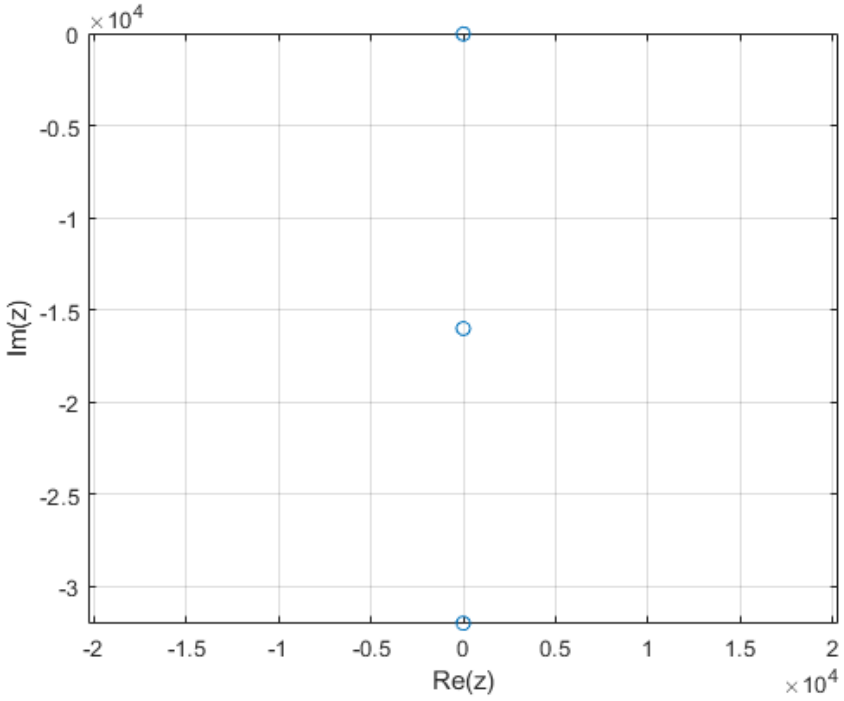

Figure 5: DFT of the signal plotted on the complex plane

It can be observed that the points lie on the negative imaginary axis of the complex plane. This indicates that the sinusoids that make up the original signal have a phase shift of 3*pi/2 or -pi/2. This is important for recreating the sound signal.

Recreating the audio signal

```
clear all;
% The frequencies were found to be 1000 Hz, 4000 Hz and 6000 Hz,
% with amplitudes 0.4, 0.2 and 0.2 respectively
f1 = 1000;
f2 = 4000;
f3 = 6000;
A1 = 0.4;
A2 = 0.2;
A3 = 0.2;

L = 160000; % Original audio being read in had 160000 samples
Fs = 16000; % Audio was sampled at 16kHz in previous step
T = 1/Fs;
t = (0:L-1)*T; % Time vector

% When Y(f1,f2,f3) are plotted on a complex plane, it can be noted that
% there is no real component and only a negative imaginary component.
% This means that there is a -90 degrees (-pi/2) phase shift
phase1 = -pi/2;
phase2 = -pi/2;
phase3 = -pi/2;

sinusoid1 = A1*cos(2*pi*f1*t+phase1);
sinusoid2 = A2*cos(2*pi*f2*t+phase2);
sinusoid3 = A3*cos(2*pi*f3*t+phase3);

final_wave = sinusoid1+sinusoid2+sinusoid3;

t_plot = 0.005; % 5 milliseconds
num_of_samples = t_plot/T; % number of samples in 5 msec

% Plotting the first 5 msec of the signal
plot(t(1:num_of_samples),final_wave(1:num_of_samples));
title('Plot of Recreated Signal');
xlabel('time (seconds)');
grid('minor');
```

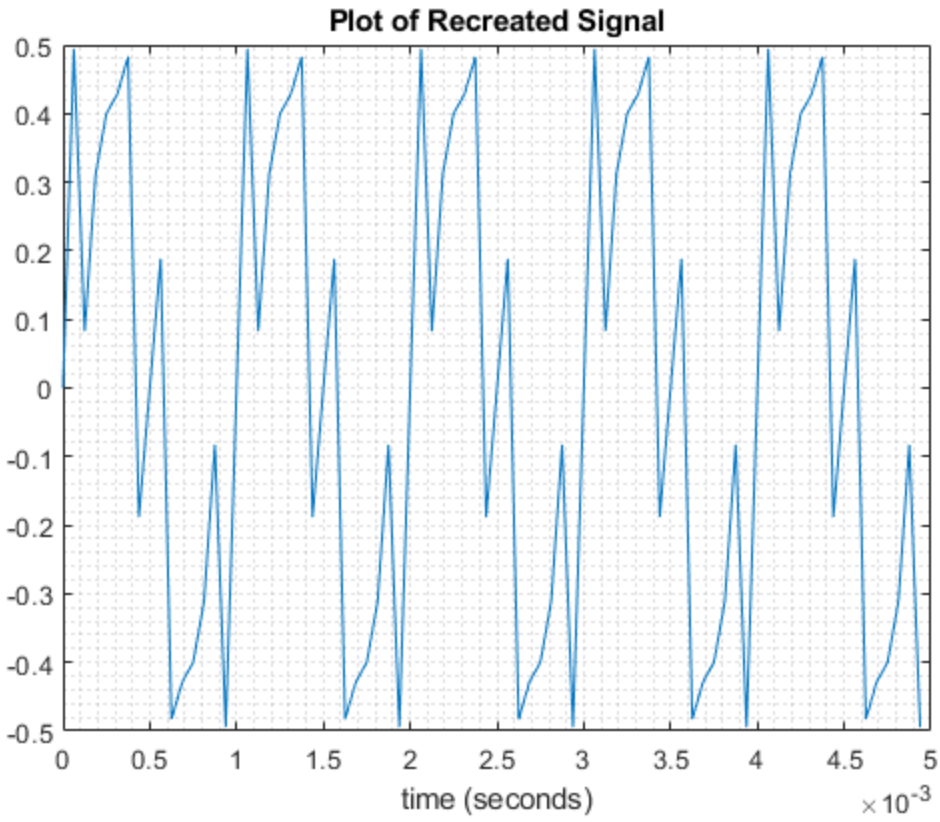

Figure 6: Plotting the first 5 milliseconds of the recreated signal originating from tones2023.wav

Part II

Listening to *SecretMessage2023.wav*, we hear a loud static noise and very quiet and subtle tones behind the garbage noise. Without listening closely, the tones in the background can easily go unnoticed. When listened carefully, the tone sounds to be changing between a low pitch and high pitch at varying time intervals.

Finding all the frequencies used

```
clear all;

% Read in the signal from the audio file
[signal, Fs] = audioread("SecretMessage2023.wav");
T = 1/Fs; % Sampling period; 1/(sampling frequency)
L = length(signal); % Number of points in 'signal'
t = [0:L-1] * T; % Time vector
Y = fft(signal); % perform DFT

% multiply magnitude of the output of DFT by 2/L to get the peak amplitude
A = abs(Y)*2/L;

% '(0:L-1)' is cycles per L points; converts to cycles per second
f = (0:L-1)*Fs/L;

% plotting only the positive frequencies;
plot(f(1:ceil(L/2)),A(1:ceil(L/2)));
title('Single-Sided Magnitude Spectrum');
ylabel('Amplitude');
xlabel('Frequency (Hz)');
axis([0 Fs/2 0 0.1])
grid('minor');
```

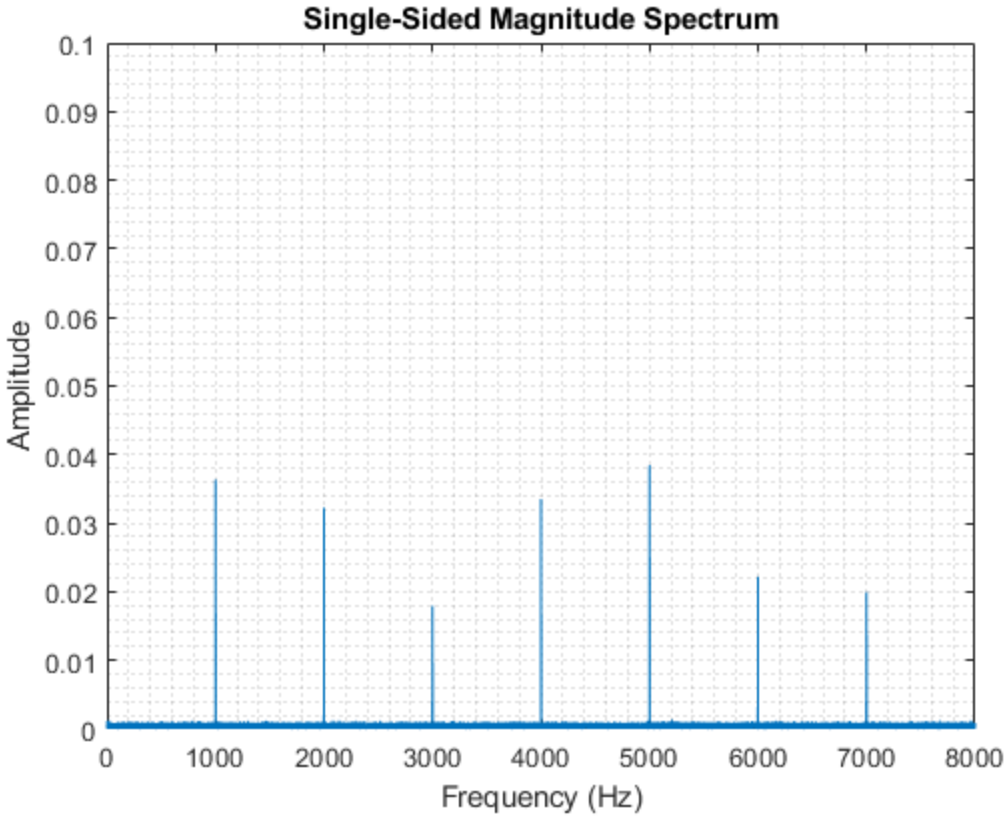

Figure 7: DFT of signal in SecretMessage2023.wav; Plot of the single-sided magnitude spectrum

The frequencies used in this audio file are:

1000 Hz, 2000 Hz, 3000 Hz, 4000 Hz, 5000 Hz, 6000 Hz, 7000 Hz.

Decoding the message

```
clear all;

% Read in the signal from the audio file
[signal, Fs] = audioread("SecretMessage2023.wav");
T = 1/Fs; % Sampling period; 1/(sampling frequency)
L = length(signal); % Number of points in 'signal'
t = [0:L-1] * T; % Time vector

symbol_period = 1; % message is encoded in 1-second duration symbol periods
symbol_sample = symbol_period/T; % number of samples in 1 second

for i = 0:75
   j = i+1;

   % i and j determine which symbol period to perform DFT
   Y = fft(signal(symbol_sample*i+1:symbol_sample*j)); % perform DFT
```

```
    % multiply magnitude of the output of DFT by 2/L to get the peak amplitude
    A = abs(Y)*2/symbol_sample;

    % '(0:L-1)' is cycles per L points; converts to cycles per second
    f = (0:symbol_sample-1)*Fs/symbol_sample;

    %plotting all 76 symbol periods
    subplot(13,6,i+1)
    plot(f(1:ceil(symbol_sample/2)),A(1:ceil(symbol_sample/2)));
    axis([0 Fs/2 0 0.1])
    grid('minor');
end
```

Figure 8: Plotting the DFT of every symbol period

A	1000	2000	3000	4000
B	1000	2000	3000	5000
C	1000	2000	3000	6000
D	1000	2000	3000	7000
E	1000	2000	4000	5000
F	1000	2000	4000	6000
G	1000	2000	4000	7000
H	1000	2000	5000	6000
I	1000	2000	5000	7000
J	1000	2000	6000	7000
K	1000	3000	4000	5000
L	1000	3000	4000	6000
M	1000	3000	4000	7000
N	1000	3000	5000	6000
O	1000	3000	5000	7000
P	1000	3000	6000	7000
Q	1000	4000	5000	6000
R	1000	4000	5000	7000
S	1000	4000	6000	7000
T	1000	5000	6000	7000
U	2000	3000	4000	5000
V	2000	3000	4000	6000
W	2000	3000	4000	7000
X	2000	3000	5000	6000
Y	2000	3000	5000	7000
Z	2000	3000	6000	7000
SPACE	2000	4000	5000	6000
PERIOD	2000	4000	5000	7000

Figure 9: Codebook used to decode the message

Cross-referencing each subplot against the codebook above, it revealed the following code when read left to right, row by row.

“NEVER LET THE FEAR OF STRIKING OUT KEEP YOU FROM PLAYING THE GAME. BABE RUTH.”