Mario Theme Song
Signals and System Term Project

Group #1 (Group Alpha)
John Carucci
Priyank Kashyap
Vitaliy Vorobets

EENG 341-M01
Dr. Ziqian Dong

Objectives:

• Use MATLAB to create music
• Graph the Time and Frequency components of the song
• Extra Credit: Add white Gaussian noise to your music and analyze and plot the output music in both time and frequency domains.
• Extra Credit: Design a filter to remove noise and analyze and plot the music in both time and frequency domain.

Part 1: Creating the Song

Our team has chosen the "Super Mario Bros. theme song", officially known as the "Ground Theme" or Overworld Theme, is a musical theme originally heard in the first stage of the 1985 Nintendo Entertainment System video game Super Mario Bros. It was one of six themes composed for Super Mario Bros. by composer Koji Kondo, who found it to be the game's most difficult track to compose. The theme has a calypso rhythm and usually receives a corresponding orchestration in games whose sound synthesizers can imitate steel drums.
Since being included in Super Mario Bros, it went on to become the theme of the series, and has been a fixture in most of its titles. It has been reused and remixed in other Nintendo-published games.

In MATLAB, we generated a19 second clip of song's main melody. We started by finding the original notes. Since none of us had any musical background we had to learn how to read music notes. Afterwards we interpreted each note into frequency and set a list variable to its representing note. Using MATLAB vectors and list of notes we were able to recreate the 19 seconds music. After listening to song once we noticed that rests were missing. Finding the rests was the most time consuming part. Replaying the song over and over to see where we needed rests was time consuming, but it was worth it. The song is saved to a Waveform Audio File Format, titled as ("Mario.wav"). The song can be found in the same directory as the ("Mario.m") file.

Part 2: Plotting the Signal in Time and Frequency domains

By using tools provided by MATLAB, we managed to plot Time and Frequency domains of the "Mario Song". Using the "audioread" function in MATLAB we were able to recreate the graphs as seen in Figure 1. The original music is shown in the first two panels, the time graph on the left and the frequency graph on the right.

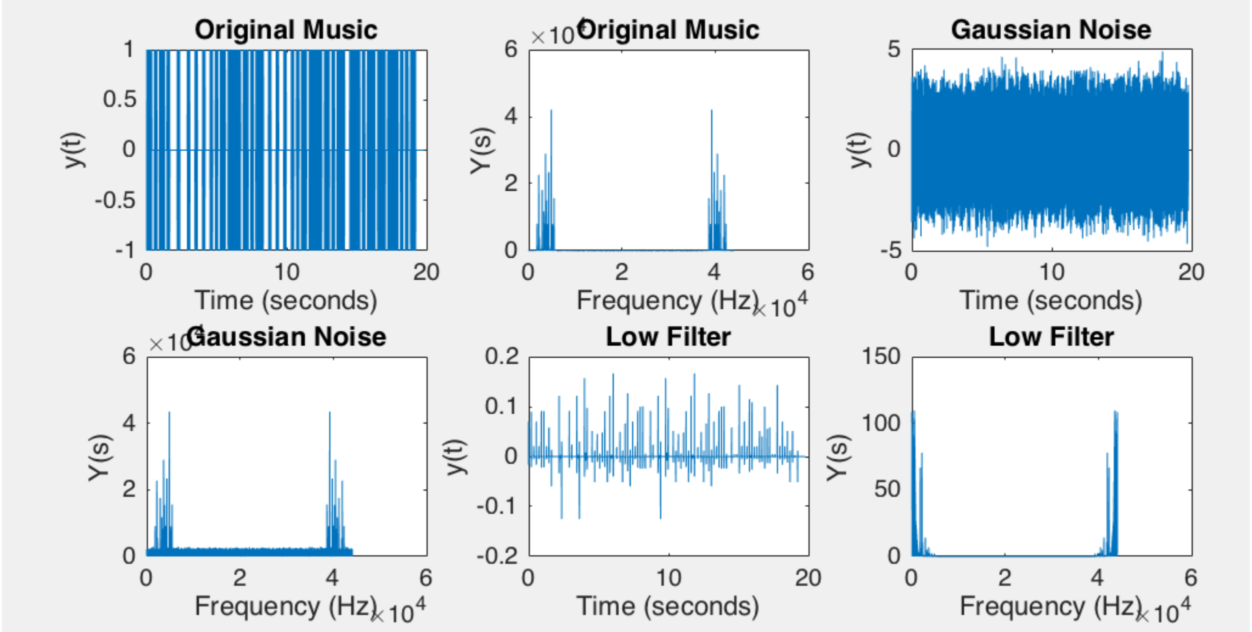

Figure 1

Part 3: Extra Credit

For extra credit we were given the task of adding white Gaussian noise, and then filtering it out. For both cases we needed to create respectable graphs in their Time and Frequency domains.

To add noise we used provided by MATLAB Gaussian noise tool, and by altering its parameters, we were able to get a balanced song where, noise and song are balanced and both can be heard. The output file is “MarioWithNoise.wav”. Furthermore, the music was graphed after the Gaussian noise was added. The Gaussian noise graphs are in panel 3 (time graph) in the first row, and panel 1(frequency graph) in the second row.

For filtering the noise we used the Butterworth Filter utility. This utility acts a low pass filter, which simply attenuates high frequency and doesn’t change original signal. The result file created is “MarioFiltered.wav”. The graph for the filtered version of the song occupies the last 2 panels on the second row, panel 5 is the time graph and panel 6 is the frequency graph.

Conclusion

This project has brought us a better understanding of signals and systems and the use of MATLAB in this subject. We now better understand how signals behave and how they can be altered to produce desired results. We learned that signals can be applied in different applications and this project showed us how to apply signals to create music.

All our team members had evenly balanced the workload to complete this project. We learned how to read music, write, debug code, manage our time and work as a group.

APPENDIX A

```
close all
clear
clc
%Super Mario Bros. Theme Song
%Note Beats and Rests
E5=sin(2*pi*659.25*(0:0.000125:1));
E53=sin(2*pi*659.25*(0:0.000125:1.33));
C5=sin(2*pi*523.25*(0:0.000125:1));
G5=sin(2*pi*783.99*(0:0.000125:1));
G5F=sin(2*pi*739.99*(0:0.000125:1));
G53=sin(2*pi*783.99*(0:0.000125:1.33));
G4=sin(2*pi*392.00*(0:0.000125:1));
G4S=sin(2*pi*415.30*(0:0.000125:1));
G43=sin(2*pi*392.00*(0:0.000125:1.33));
E4=sin(2*pi*329.63*(0:0.000125:1));
A5=sin(2*pi*880.00*(0:0.000125:1));
A4=sin(2*pi*440.00*(0:0.000125:1));
B5=sin(2*pi*987.77*(0:0.000125:1));
B5F=sin(2*pi*932.33*(0:0.000125:1));
A6=sin(2*pi*1760.00*(0:0.000125:1));
F5=sin(2*pi*698.46*(0:0.000125:1));
D5=sin(2*pi*587.33*(0:0.000125:1));
D5S=sin(2*pi*622.25*(0:0.000125:1));
R = sin(2*pi*0*(0:0.000125:1));

line1 = [E5,E5,R,E5,R, C5,E5,R, G5,R,R,R G4,R, R, R];
line2 = [C5,R, R,G4,R,R,E4,R,R,A5,R,B5,R,B5F,A5,R];
line3 = [G43,E53, G53,A5,R,F5,G5,R,E5,R,C5,D5,B5,R,R];
line4 = [C5,R, R,G4,R,R,E4,R,R,A5,R,B5,R,B5F,A5,R];
line5 = [G43,E53, G53,A5,R,F5,G5,R,E5,R,C5,D5,B5,R];
line6 = [R,G5,G5F,F5,D5S,R,E5,R,G4S,A5,E5,R,A5,C5,D5];
line7 = [R,G5,G5F,F5,D5S,R,E5,R,A5,R,A5,A5,R,R,R];
song=[line1,line2,line3,line4,line5,line6,line7];
audiowrite('MARIO.wav',song, 44100);

% music components
[wave, fs] = audioread('MARIO.wav');   % sampling frequency (fs = 30000)
n = length(wave)-1; % discrete increments
t = 0:1/fs:n/fs; % time axis increments
f = 0:fs/n:fs;  % frequency axis increments
y = awgn(wave,1);   %white Gaussian noise
wavefft = abs(fft(wave));   % fast fourier transform of the original music
yfft = abs(fft(y));       % fast fourier transform of the Gaussian noise

noise= awgn(wave,5,'measured');
audiowrite('MarioWithNoise.wav',noise,44100);

% t-domain filtered components
nn = 5;
mag = 700;
w = mag*2/fs;
[b, a] = butter(nn,w,'low');
res = filter(b,a,wave);
```

```
out=filter(b,a,noise);
audiowrite('MarioFiltered.wav',out,44100);
```