Results Review of Detecting of Human Errors Algorithm for audio files

Author: Minh Anh Nguyen
Email: minhanhnguyen@q.com

Purpose

The purpose of this analysis is to determine if the method of comparing two audio files can be used for detecting of human errors for channel-not-loaded-correctly issue. There are many algorithms which can be used to compare two audio files. These algorithms are: Discrete Fourier Transform (DFT), Short Time Fourier Transform (STFT), Wavelet, and Fast Fourier Transform (FFT). In this project, FFT algorithm is used to compare two audio files.

Why use FFT

The fast Fourier transform (FFT) is an algorithm for converting a time-domain signal into a frequency-domain representation of the relative amplitude of different frequency regions in the signal. The FFT is an implementation of the Fourier transform. The Fourier transform is one of the most useful mathematical tools for many fields of science and engineering. The Fourier transform displays the frequency components within a time series of data. The result of the FFT contains the frequency data and the complex transformed result. The FFT works perfectly when analyzing exactly one cycle of a tone.

How FFT works

The FFT takes a chunk of time called a *frame* (number of samples) and considers that chunk to be a single period of a repeating waveform. Most sounds are locally stationary, which means that the sound does look like a regularly repeating function in a short period of time.

The FFT is the one-dimensional Fourier transform. If assuming a signal is saved as an array in the variable X, then preforming fft(X) will return the Fourier transform of X as a vector of the same size as X. However, the values returned by fft(X) are frequencies.

Applications of the FFT

There are many applications of FFT. The FFT algorithm tends to be better suited to analyzing digital audio recordings than for filtering or synthesizing sounds. The FFT is used to determine the frequency of a signal, to try to recognize different kinds of sound, etc.

Problem with FFT

The FFT function automatically places some restrictions on the time series to generate a meaningful, accurate frequency response. The FFT function uses $(n/2) \log_2 (n)$, it requires that the length of the time series or total number of data points precisely equal to a 2^n. Therefore, FFT can only calculate with a fixed length waveform such as 512 points, or 1024 points, or 2048 points, etc. Another problem with using the FFT for processing sounds is that the digital recordings must be broken up into chunks of *n* samples, where *n* always has to be an integer power of 2. When the audio is broken up into chunks like this and processed with the FFT, the filtered result will have discontinuities which cause a clicking sound in the output at each chunk boundary. For example, if the recording has a sampling rate of 44,100 Hz, and the blocks have a size n = 1024, then there will be an audible click every 1024 /(44,100 Hz) = 0.0232 seconds, which is extremely annoying to say the least. The input signal must be repeats periodically and that the periodic length is equal to the length of the actual input; otherwise leakage will occur and cause both the amplitude and position of a frequency measurement to be inaccurate.

Techniques for comparing two audio files

Step 1: Load audio files

- Read in two audio files into the workspace.

Step2: Truncate both signals so that their durations are equivalent.

Step 3: Perform FFT

- Compute normalized energy spectral density (ESD) from DFT's two signals

Step 4: Compute mean-square-error (MSE)

- Compute mean-square-error (MSE) between normalized ESD's of two signals
- Two perfectly identical signals will obviously have MSE of zero.

Step 5: Display error message on the monitor/computer screen, if MSE value is not zero.

Flow chart for comparing two audio files

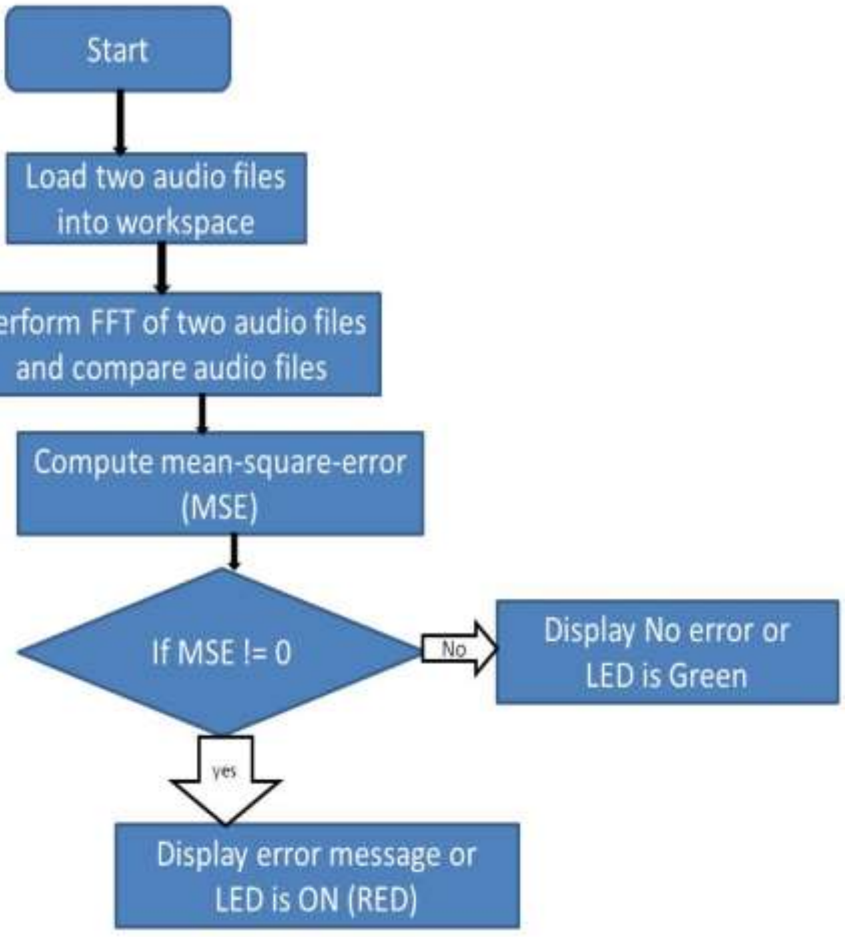

Start
Load two audio files into workspace
Perform FFT of two audio files and compare audio files
Compute mean-square-error (MSE)
If MSE != 0
No
Display No error or LED is Green
yes
Display error message or LED is ON (RED)

Figure1. Flow chart for comparing two audio files

Audio test in Simulation mode

The following steps are the procedure to perform audio test in simulation mode

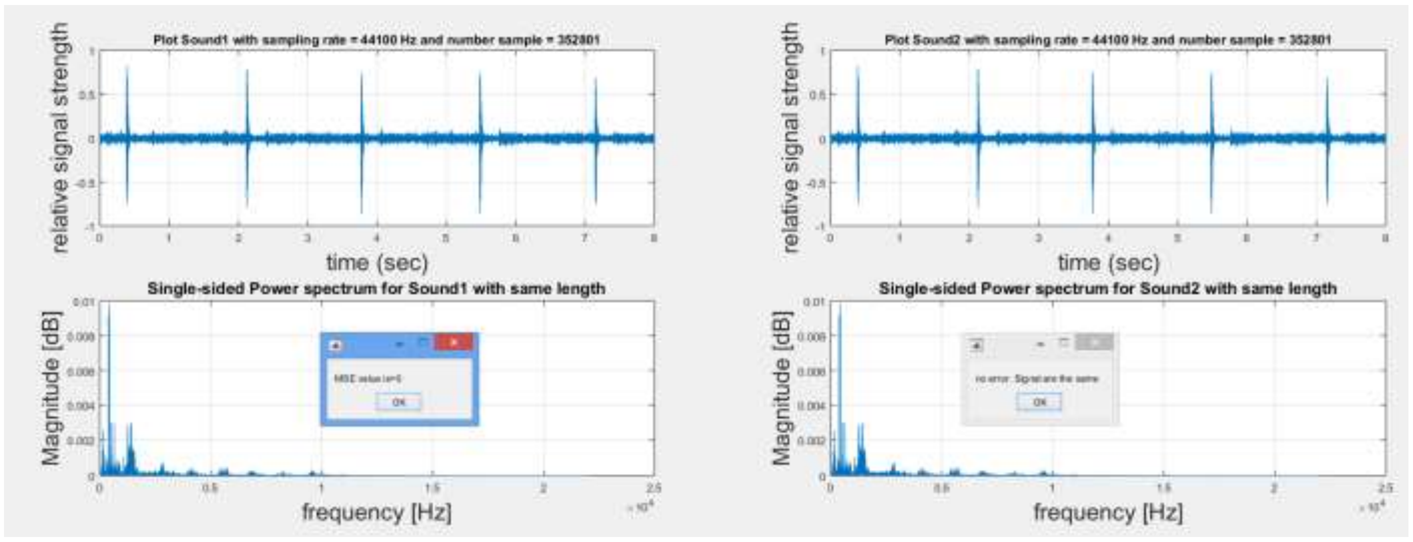
1. Make sure audio files are loaded into the workspace without any issues.
2. Plot and listen to audio files, make sure a correct file is loaded.
3. Verify that the FFT value is correct.
4. Positive test: load two audio files which have 2 different signals and verify that the test results show the difference.
5. Negative test: load two audio files which have 2 similar signals and verify that the test result show the same thing or match.

Software

Matlab R2015b
Labview 2014

Summary of Results

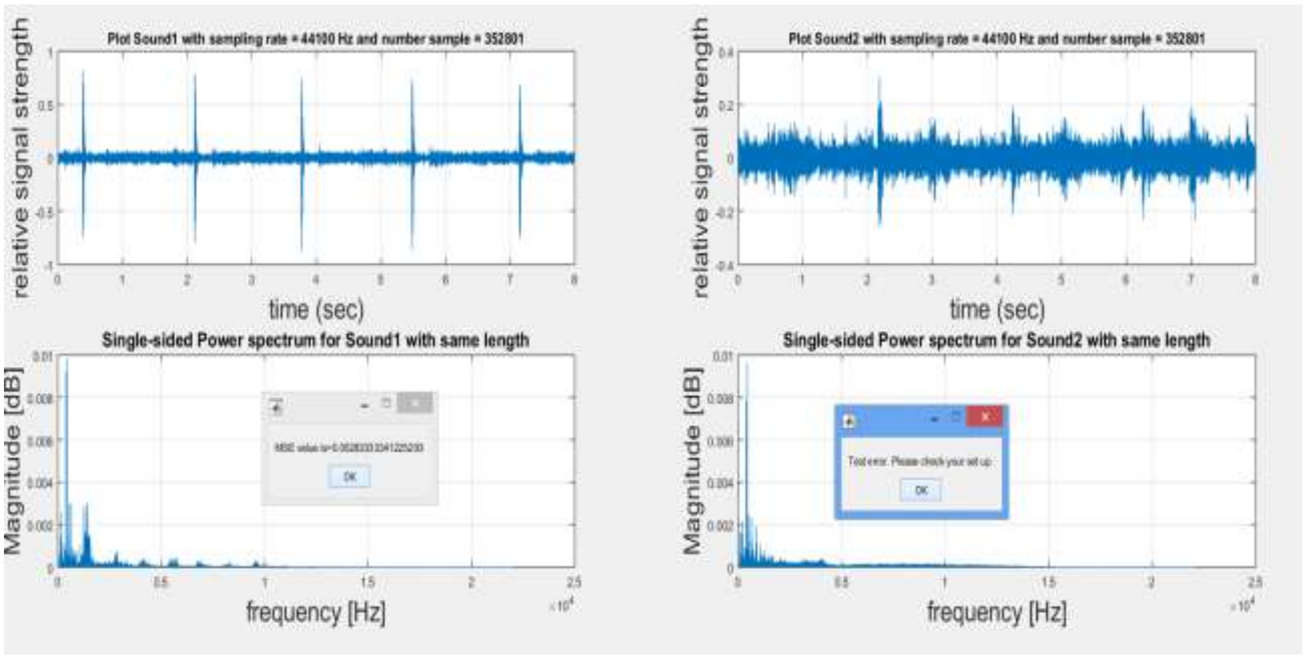
The following figure illustrates the difference between the displays of the FFT of the repeats periodically input signals and the results of the techniques and flow chart for comparing two audio files above.


Plot Sound1 with sampling rate = 44100 Hz and number sample = 352801
relative signal strength
time (sec)
Single-sided Power spectrum for Sound1 with same length
Magnitude [dB]
frequency [Hz]
OK
Plot Sound2 with sampling rate = 44100 Hz and number sample = 352801
relative signal strength
time (sec)
Single-sided Power spectrum for Sound2 with same length
Magnitude [dB]
frequency [Hz]
OK

```
MSE value is=0
```

```
no error: Signal are the same
```

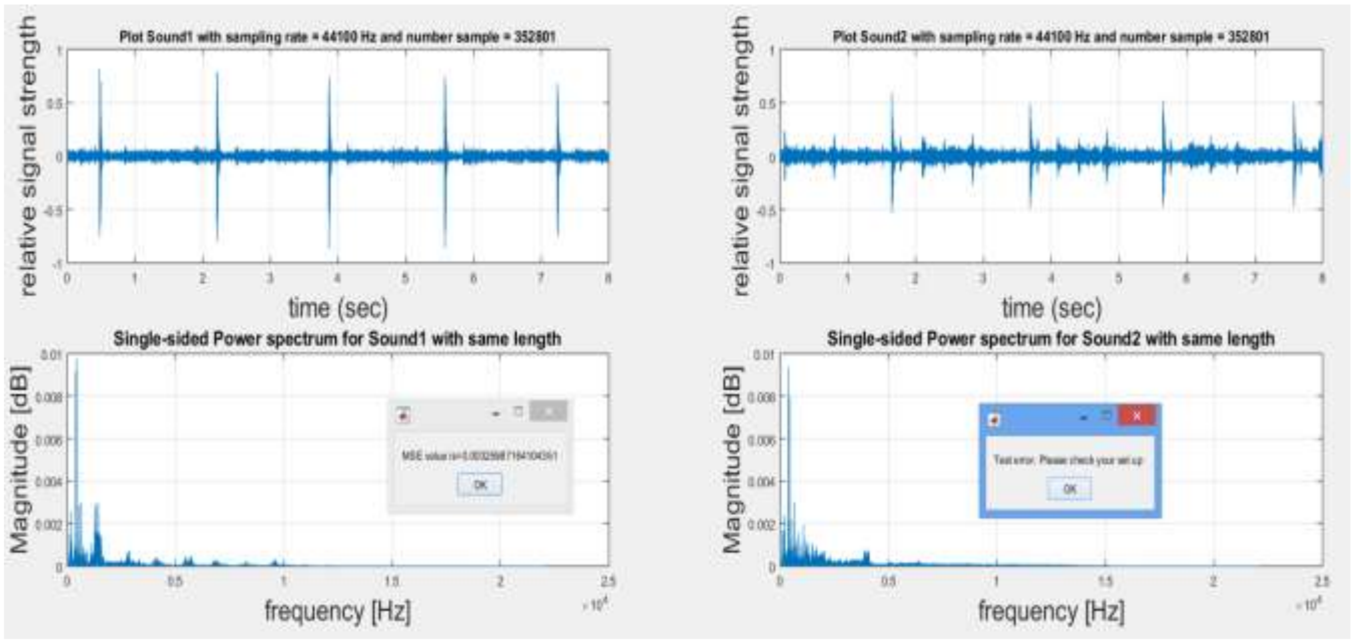
Figure2. Matlab results for comparing two audio files which are identical


Plot Sound1 with sampling rate = 44100 Hz and number sample = 352801
relative signal strength
time (sec)
Single-sided Power spectrum for Sound1 with same length
Magnitude [dB]
frequency [Hz]
OK
Plot Sound2 with sampling rate = 44100 Hz and number sample = 352801
relative signal strength
time (sec)
Single-sided Power spectrum for Sound2 with same length
Magnitude [dB]
frequency [Hz]
OK

```
MSE value is=0.0528333341225200
```

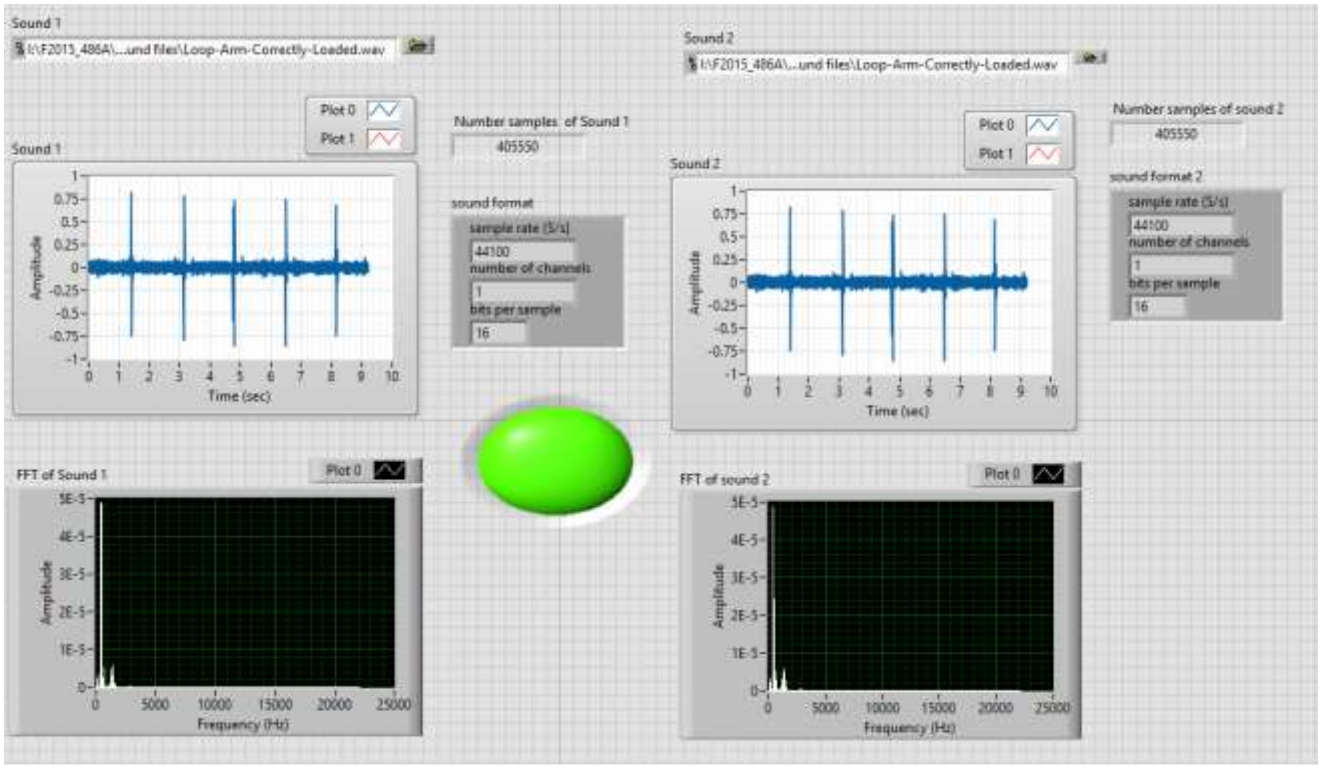
```
Test error. Please check your set up
```

Figure3. Matlab results for comparing two audio files which are not identical


Plot Sound1 with sampling rate = 44100 Hz and number sample = 352801
relative signal strength
time (sec)
Single-sided Power spectrum for Sound1 with same length
Magnitude [dB]
frequency [Hz]
Plot Sound2 with sampling rate = 44100 Hz and number sample = 352801
relative signal strength
time (sec)
Single-sided Power spectrum for Sound2 with same length
Magnitude [dB]
frequency [Hz]
OK
OK
MSE value is=0.00325987184104361

Test error, Please check your set up

Figure4. Matlab results for comparing two audio files which are not identical



Sound 1: I:\F2015_486A\...und files\Loop-Arm-Correctly-Loaded.wav

Number samples of Sound 1: 405550

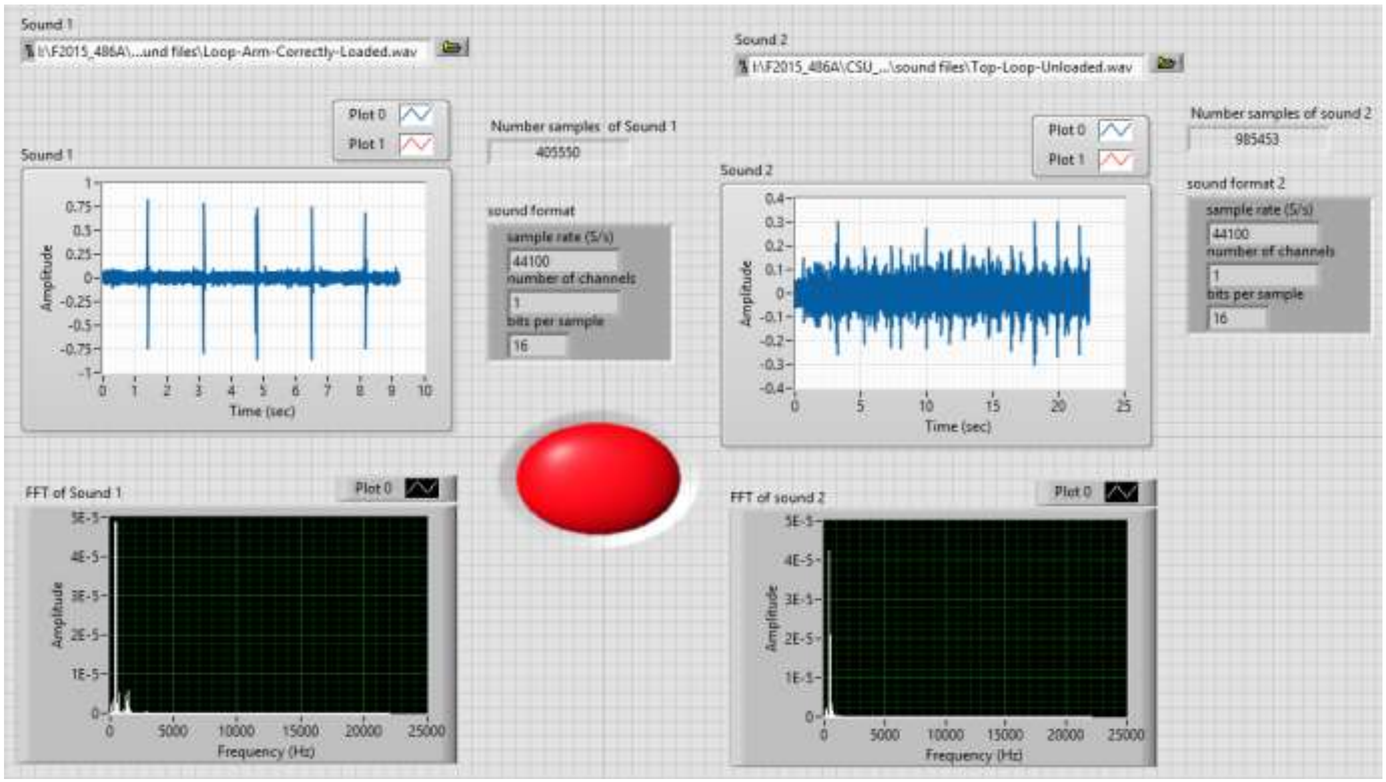
sound format
sample rate (S/s): 44100
number of channels: 1
bits per sample: 16

Sound 2: I:\F2015_486A\...und files\Loop-Arm-Correctly-Loaded.wav

Number samples of sound 2: 405550

sound format 2
sample rate (S/s): 44100
number of channels: 1
bits per sample: 16

Figure5. Labview results for comparing two audio files which are identical


Sound 1

I:\F2015_486A\...und files\Loop-Arm-Correctly-Loaded.wav

Number samples of Sound 1

405550

sound format

sample rate (S/s)

44100

number of channels

1

bits per sample

16

Sound 2

I:\F2015_486A\CSU_...\sound files\Top-Loop-Unloaded.wav

Number samples of sound 2

985453

sound format 2

sample rate (S/s)

44100

number of channels

1

bits per sample

16

Figure6. Labview results for comparing two audio files which are not identical

These results showed that the method of comparing two audio files can be used for detecting of human errors for channel-not-loaded-correctly issue. FFT of a time domain signal takes the samples and calculate a new set of numbers representing the frequencies, amplitudes, and phases of the sine waves that make up the sound.

Matlab Code for comparing two audio files.

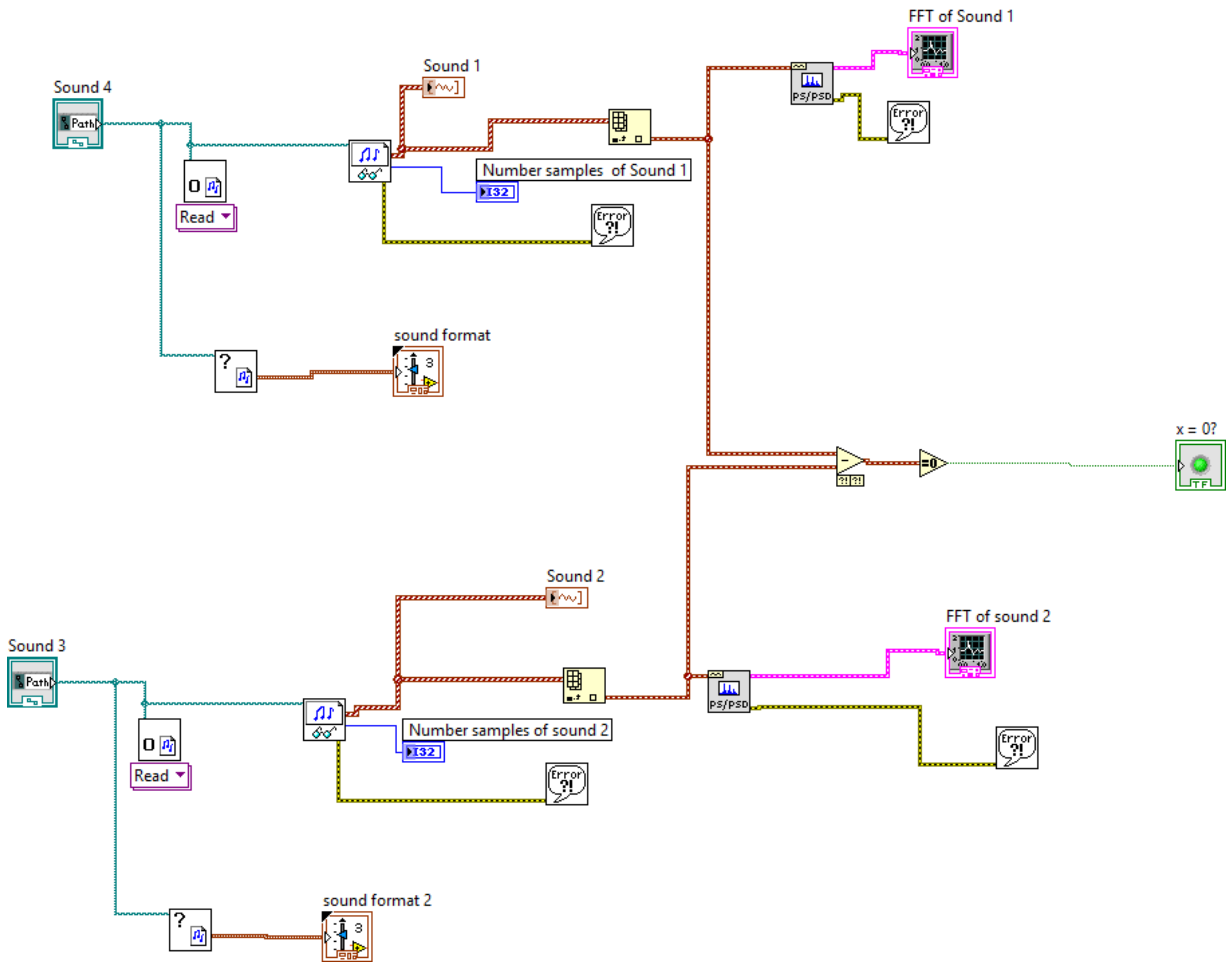
```
% MatLab Code:
% Author: Minh Anh Nguyen (minhanhnguyen@q.com)
% Expects a .wav soundfile as input.
% usage: Soundfunction('myfile.wav')
% This function will read in two wav files, perform fft, and compare these
files.  it
% will display on the screen whether or not the file are identifcal.
% this function is also perfom autocorrelation
% This beginning part just defines the function to be used in MatLab: takes a
% "wav" sound file as an input, and spits out a graph.


function [Soundfunction] = Soundfunction( file1, file2 );

clf % Clears the graphic screen.
close all;% Close all figures (except those of imtool.)
clc;% Clear the command window.
fontSize = 20;
```

```
fontSize1 = 14;

% Reads in the sound files, into a big array called y1 and y2.
%y = wavread( file );
[y1, fs1]= audioread( file1 );
[y2, fs2]= audioread( file2 );




% Normalize y1; that is, scale all values to its maximum. Note how simple it
is
% to do this in MatLab.

yn1 = y1/max(abs(y1));
yn2 = y1/max(abs(y1));

sound1 = y1;
sound2 = y2;

%% Do not modify here
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%% time of sound
N = length(sound1); % number of points to analyze
ls = size (sound1); % find the length of the data per second
nbits = 2^(length(sound1));
t1 = (0:1:length(sound1)-1)/fs1; %% time1
fx = fs1*(0:N/2-1)/N;   %Prepare freq data for plot
T1 = 1/fs1 % period between each sample


N2 = length(sound2);
ls2 = size (sound2); % find the length of the data per second
t2 = (0:1:length(sound2)-1)/fs2;
fx2 = fs2*(0:N2/2-1)/N2;   %Prepare freq data for plot
T2 = 1/fs2 % period between each sample


%%%%% cut signal for comparing
% cut signal to same length
voice1 = y1(fs1*1 : fs1*9);
voice2 = y2(fs2*1 : fs2*9);
%% find new length
N2x = length(voice2);
N1x = length(voice1);

% find new time
t1x = (0:1:length(voice1)-1)/fs1;
t2x = (0:1:length(voice2)-1)/fs2;

%% find new frequency
f2x = fs2*(0:N2x/2-1)/N2x;
f1x = fs1*(0:N1x/2-1)/N1x;

%% fft of cut signal
```

```
NFFT1x = 2 ^ nextpow2(N1x);
Y1x = fft(voice1, NFFT1x)/ N1x;
f1xx = (fs1/ 2 * linspace(0, 1, NFFT1x / 2+1))'; % Vector containing
frequencies in Hz
STFFT1x = ( 2 * abs(Y1x(1: NFFT1x / 2+1))); % Vector containing corresponding
amplitudes


NFFT2x = 2 ^ nextpow2(N2x);
Y2x = fft(voice2, NFFT2x) / N2x;
f2xx = (fs2 / 2 * linspace(0, 1, NFFT2x / 2+1))'; % Vector containing
frequencies in Hz
STFFT2x = ( 2 * abs(Y2x(1: NFFT2x / 2+1))); % Vector containing corresponding
amplitudes

%% plot for the cut signal
%% plot for the cut signal
figure; subplot (3,2,1); plot(t1x, voice1);
str1=sprintf('Plot Sound1 with sampling rate = %d Hz and number sample = %d',
fs1, N1x);
title(str1);
xlabel('time (sec)'); ylabel('relative signal strength'); grid on;

subplot (3,2,2); plot(t2x, voice2);
str2=sprintf('Plot Sound2 with sampling rate = %d Hz and number sample = %d',
fs2, N2x);
title(str2); xlabel('time (sec)','Fontsize', fontSize); ylabel('relative
signal strength','Fontsize', fontSize); grid on;

%% fft of cut signal
subplot (3,2,3); plot(f1xx, STFFT1x); title ('Single-sided Power spectrum for
Sound1 with same length','Fontsize', fontSize1);
ylabel ('Magnitude [dB]','Fontsize', fontSize); xlabel ('frequency
[Hz]','Fontsize', fontSize); grid on;


subplot (3,2,4); plot(f2xx, STFFT2x); title ('Single-sided Power spectrum for
Sound2 with same length','Fontsize', fontSize1);
ylabel ('Magnitude [dB]','Fontsize', fontSize); xlabel ('frequency
[Hz]','Fontsize', fontSize); grid on;

%% corlation

[C1, lag1] = xcorr(abs((fft(voice1))),abs((fft(voice2)) ));
 figure, plot(lag1/fs1,C1);
    ylabel('Amplitude'); grid on
    title('Cross-correlation between Sound1 and sound2 files')

%% calculate mean
% Calculate the MSE
D= voice1 - voice2;
MSE=mean(D.^2);
    %msgbox(strcat('MSE value is= ',mat2str(),' MSE'));
        msgbox(strcat('MSE value is= ',mat2str(MSE)));
if MSE ==0;
```

```
    msgbox('no error. Signal are the same');
    disp('Signal are the same');
else
    disp('Signal are not the same');
    msgbox('Test error. Please check your set up');
end
```

Labview Code for comparing two audio files.


Sound 4
Path
0
Read
Sound 1
Number samples of Sound 1
I32
Error ?!
sound format
3
FFT of Sound 1
PS/PSD
Error ?!
x = 0?
=0
Sound 2
FFT of sound 2
PS/PSD
Error ?!
Sound 3
Path
0
Read
Number samples of sound 2
I32
Error ?!
sound format 2
3