ECE 4750: Digital Signal Processing Lab

Project 4: Encoding and Decoding Touch-Tone (DTMF) Signals

Angel Rodriguez

Utah Valley University

# Introduction

The objective of this lab is to learn how to use figure out which keypad buttons are pressed only given a .wav sound file. For the first parts, the signal in time domain, the FFT (fast Fourier transform) of the signal, and the spectrum are given for multiple button presses. Analyzing these figures, the MATLAB code, and the DTMF encoding table for Touch Tone dialing (Figure 1), it will be determined which buttons are pressed. Using the same analysis method, new MATLAB code will be created for three different .wav sound files which will generate the time domain function, FFTs, and spectrums to determine which keypads are pressed. Figure 2 depicts the filter bank consisting of bandpass filters (BPFs) which pass frequencies corresponding to those shown in Figure 1. The analysis and results are below.

# Tasks

**Problem 1:**  
The code first clears the workspace and command window and assigns Fs, the sampling frequency to 32768 (Hz). The time vector, t, goes from 0 to 0.25 seconds in increments of 1/Fs (the sampling period). Row matrices are made from the both the frequency row (fr) and the frequency column (fc). Two for loops are made to combine signals of fr and fc that will run a total of 12 times (1 time for each of the 12 keys: 0, 1, 2, …, 8, 9, \*, 0, #). Each of the signals are applied to sin(2\*pi\*fr(m)\*t) where m is the hertz value obtained from fr or fc. Combining two sinusoids that use a frequency from fc and fr, gives y = 0.5\*(y1+y2) where y1 is a sinusoid using a frequency from fr and y2 is the sinusoid from fc. The resulting sounds are announced, and their functions are displayed in the time domain. The sounds are just as a phone dial sounds, and the time domain functions (labeled ‘Signal/tone’) look messy.

**Problem 2:**The code repeats the process of the previous code with the addition of applying the fast Fourier transform (FFT, an algorithm of the Fourier transform) to the magnitude of y where y = 0.5\*(y1+y2). A Fourier transform (FT) is a mathematical transform that decomposes functions depending on space or time into functions depending on spatial frequency or temporal frequency. In addition to sounding out the signals and graphing them, the FFT of the signals are displayed which provide information much clearer than the time domain. The FFT is a graph in which the x-axis is the frequency values and the y-axis is |Y(jw)|.

**Problem 3:**Each FFT plot is showing 2 spikes which each spike representing a frequency from either the row frequencies or the column frequencies shown in Figure 1. At each of the FFTs, the approximate x-value at the top of each of these spikes will give you a frequency that will match the chart from Figure 1. For example, Figure 3 shows the x-values 699 and 1211 which are closes to the 697 Hz and 1209 Hz values of the Figure 1 table. Therefore, this is the FFT for #1 on the DTMF encoding table. Using this strategy, we can determine the which FFT plot corresponds to which keypad.

|  |  |
| --- | --- |
| **MATLAB Figure Number** | **Corresponding Keypad Button** |
| **1** | **1** |
| **2** | **2** |
| **3** | **3** |
| **4** | **4** |
| **5** | **5** |
| **6** | **6** |
| **7** | **7** |
| **8** | **8** |
| **9** | **9** |
| **10** | **\*** |
| **11** | **0** |
| **12** | **#** |

**Problem 4:**FFT (fast Fourier transform) convolution uses the principle that multiplication in the frequency domain corresponds to convolution in the time domain. The input signal is transformed into the frequency domain using the DFT (discrete Fourier transform), multiplied by the frequency response of the filter, and then transformed back into the time domain using the inverse DFT. In MATLAB, this is done with a signal x as: **y = fft(x);** with this file applying the FFT to the absolute value by **p = abs(fft(Z));** as to not graph the frequencies before 0 since the positive frequencies already tell us what we need to know to find which keypads are pressed. Figure 4 shows the time domain of the signals named “Signal” which tells there are 8 keypads pressed. In Figure 4, there is also the FFT of the signal labeled “FFT of Signal” which tells the 7 frequencies which make up the 8 keypad sounds. Looking at the FFT alone, it is difficult to tell which buttons are pressed since each keypad displayed in the time domain is a combination of 2 frequencies in the FFT. There are many different possibilities with just the information from Figure 1 alone. Figure 5, the spectrogram of the whole signal shows both frequencies as well as the power of the signal (loudness) over time with time being the x-axis. The figure shows us 8 columns of bright colors which are each of the 8 buttons pressed. Where the signal is the loudest, yellow is displayed. For each of these columns (buttons pressed), looking at the two centers of the yellow colors as it aligns with the frequency measurement will tell the frequencies of that keypad. Here are the buttons pressed based on this analysis.

|  |  |
| --- | --- |
| **Column Number (from Left to Right)** | **Corresponding Keypad Button** |
| **1** | **8** |
| **2** | **#** |
| **3** | **6** |
| **4** | **3** |
| **5** | **0** |
| **6** | **8** |
| **7** | **8** |
| **8** | **5** |

**Problem 5:**Using the same analysis of the spectrum used for Problem 4, below are the results of the numbers which were pressed for file Test1.wav. The MATLAB figures are shown in Figure 6.

|  |  |
| --- | --- |
| **Column Number (from Left to Right)** | **Corresponding Keypad Button** |
| **1** | **#** |
| **2** | **\*** |
| **3** | **5** |
| **4** | **5** |
| **5** | **#** |
| **6** | **0** |
| **7** | **9** |
| **8** | **3** |

**Problem 6:**Using the same analysis of the spectrum used for Problem 4, below are the results of the numbers which were pressed for file Tone1.wav. The MATLAB figures are shown in Figure 7.

|  |  |
| --- | --- |
| **Column Number (from Left to Right)** | **Corresponding Keypad Button** |
| **1** | **3** |
| **2** | **0** |
| **3** | **7** |
| **4** | **1** |
| **5** | **1** |
| **6** | **3** |
| **7** | **1** |
| **8** | **4** |

**Problem 7:**Using the same analysis of the spectrum used for Problem 4, below are the results of the numbers which were pressed for file Test2.wav. The MATLAB figures are shown in Figure 8.

|  |  |
| --- | --- |
| **Column Number (from Left to Right)** | **Corresponding Keypad Button** |
| **1** | **\*** |
| **2** | **\*** |
| **3** | **#** |
| **4** | **2** |
| **5** | **8** |
| **6** | **0** |
| **7** | **5** |
| **8** | **\*** |

# Figures

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| FREQS | 1209 Hz | 1336 Hz | 1477 Hz | 1633 Hz |
| 697 Hz | 1 | 2 | 3 | A |
| 770 Hz | 4 | 5 | 6 | B |
| 852 Hz | 7 | 8 | 9 | C |
| 941 Hz | \* | 0 | # | D |

**Figure 1:**  
Extended DTMF encoding table for Touch Tone dialing. When any key is pressed the tones of the corresponding column and row are generated and summed. Keys A-D (in the fourth column) are not implemented on commercial and household telephone sets but are used in some military and other signaling applications.

Diagram

Description automatically generated  
**Figure 2:**  
Filter bank consisting of bandpass filters (BPFs) which pass frequencies corresponding to the eight DTMF component frequencies listed in Tab. 1. The number is each box is the center frequency of the BPF.

Graphical user interface

Description automatically generated  
**Figure 3:**  
MATLAB Figure 1 of TouchTone\_project\_part2.m

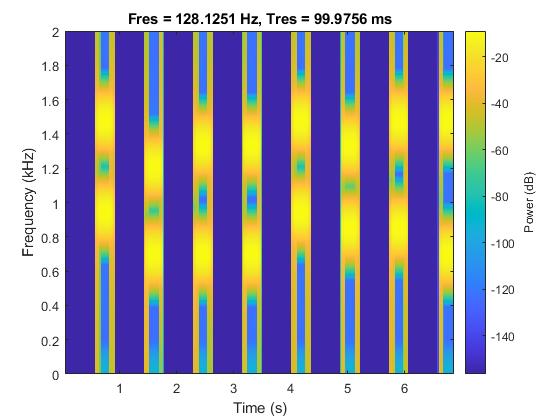
Graphical user interface, chart, Excel, bar chart

Description automatically generated  
**Figure 4:**  
Figure 1 of TouchTone\_project\_part3.m

A picture containing chart

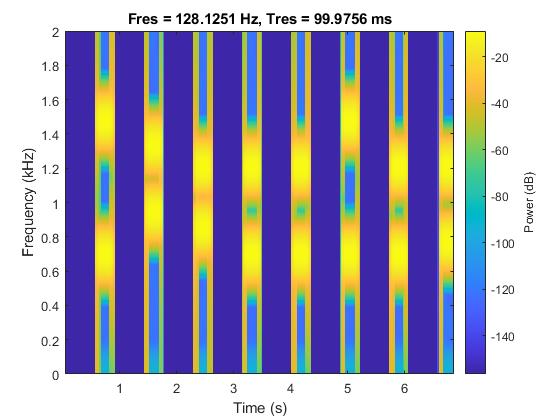
Description automatically generated  
**Figure 5:**  
Figure 2 of TouchTone\_project\_part3.m

Chart

Description automatically generated

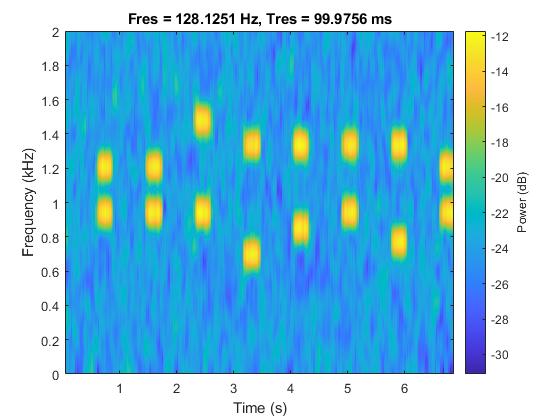
**Figure 6:**  
Figures of Problem 5

Graphical user interface, chart

Description automatically generated

**Figure 7:**  
Figures of Problem 6

Chart

Description automatically generated

**Figure 8:**  
Figures of Problem 7

# Conclusion

It has been observed that the time domain function with the FFT alone is adequate to determine which keypad has been pressed only when a single button was pressed. As soon as multiple buttons are pressed, the possibilities can become exponential, and the spectrum is required to determine which buttons are pressed. Any errors that arise from the results may come from human error of trying to distinguish which frequency aligns with the center of the yellow colors in the spectrum as eyeballing the data is not an exact measurement. From the results, it is clear that the spectrums provide valuable information to determine which buttons are pressed even where there are many buttons pressed and/or background noise.

# Attachments

**MATLAB Code of TouchTone\_project\_part1.m:**% ECE 4755 Project 3: Touch Tone

% Part 1

clc, clear; close all;

Fs = 32768;

t = 0 : 1/Fs : 0.25;

fr = [ 697 , 770 , 852 , 942];

fc = [ 1209 , 1336 , 1477];

for m = 1:4

for n = 1:3

y1 = sin(2\*pi\*fr(m)\*t);

y2 = sin(2\*pi\*fc(n)\*t);

y = 0.5\*(y1+y2);

sound(y,Fs);

pause(2);

figure; plot(t,y); grid

title('Signal/tone'); ylabel('y(t)'); xlabel('t(sec)');

end

end

**MATLAB Code of TouchTone\_project\_part2.m:**  
% ECE 4755 Project 4: Touch Tone

% Part 2

clc, clear; close all;

Fs = 32768;

t = 0 : 1/Fs : 0.25;

fr = [ 697 , 770 , 852 , 942];

fc = [ 1209 , 1336 , 1477];

for m = 1:4

for n = 1:3

y1 = sin(2\*pi\*fr(m)\*t);

y2 = sin(2\*pi\*fc(n)\*t);

y = 0.5\*(y1+y2);

sound(y,Fs);

pause(2);

y = double(y);

Len\_y = length(y);

p = abs(fft(y));

f = (Fs/Len\_y)\*(1:Len\_y);

figure; subplot(211); plot(t,y); grid

title('Signal/tone'); ylabel('y(t)'); xlabel('t(sec)');

subplot(212); plot(f,p); title('FFT of Signal');

ylabel('|Y(jw)|'); xlabel('f(Hz)'); grid;

axis([0 2000 0 2000]);

end

end

**MATLAB Code of TouchTone\_project\_part3.m:**% ECE 4755 Project 4: Touch Tone

% Part 3

clc, clear; close all;

Fs = 32768;

t = 0 : 1/Fs : 0.25;

fr = [ 697 , 770 , 852 , 942];

fc = [ 1209 , 1336 , 1477];

for m = 1:4

for n = 1:3

y1 = sin(2\*pi\*fr(m)\*t);

y2 = sin(2\*pi\*fc(n)\*t);

y = 0.5\*(y1+y2);

y = double(y);

Len\_y = length(y);

p = abs(fft(y));

f = (Fs/Len\_y)\*(1:Len\_y);

end

end

% ---------------------------

% Problem 4

Z = [];

Temp = [];

for h = 1:8

m = randperm(4);

m = m(1);

n = randperm(3);

n = n(1);

y1\_t = sin(2\*pi\*fr(m)\*t);

y2\_t= sin(2\*pi\*fc(n)\*t);

y\_t = 0.5\*(y1\_t+y2\_t);

Z = [Z,zeros(1,20000), y\_t];

Temp = [Temp, [m ; n]];

end

disp(Temp);

sound(Z,Fs)

Z = double(Z);

Len\_Z = length(Z);

tt = (1:Len\_Z)/Fs;

p = abs(fft(Z));

f = (Fs/Len\_Z)\*(1:Len\_Z);

figure; subplot(211); plot(tt,Z); grid

title('Signal'); ylabel('y(t)'); xlabel('t(sec)');

subplot(212); plot(f,p); title('FFT of Signal');

ylabel('|Y(jw)|'); xlabel('f(Hz)'); grid;

axis([0 2000 0 2000]);

figure;

pspectrum(Z,Fs,'spectrogram', ...

'FrequencyLimits',[0 2000],'TimeResolution',0.1)

% -------------------------------

% This is how the noisy signal was generated. You can uncomment this

% section and see the figures. Since this was generated randomly, you will

% get the same signal Test2.wav as is given to you in the porject.

% pause(10)

% Z = Z+randn(size(Z));

% soundsc(Z,Fs)

% Z = double(Z);

% Len\_Z = length(Z);

% tt = (1:Len\_Z)/Fs;

% p = abs(fft(Z));

% f = (Fs/Len\_Z)\*(1:Len\_Z);

% figure; subplot(211); plot(tt,Z); grid

% title('Signal'); ylabel('y(t)'); xlabel('t(sec)');

% subplot(212); plot(f,p); title('FFT of Signal');

% ylabel('|Y(jw)|'); xlabel('f(Hz)'); grid;

% axis([0 2000 0 2000]);

% figure;

% pspectrum(Z,Fs,'spectrogram', ...

% 'FrequencyLimits',[0 2000],'TimeResolution',0.1)

% audiowrite('Test2.wav',Z,Fs)

**MATLAB Code of Problem5.mlx:**clear; clc; close;

% Problem 5:

filename = 'Test1.wav';

[y,Fs] = audioread(filename);

sound(y,Fs);

t = linspace(0,10,length(y));

pause(2);

y = double(y);

Len\_y = length(y);

p = abs(fft(y));

f = (Fs/Len\_y)\*(1:Len\_y);

figure; subplot(211); plot(t,y); grid

title('Signal/tone'); ylabel('y(t)'); xlabel('t(sec)');

subplot(212); plot(f,p); title('FFT of Signal');

ylabel('|Y(jw)|'); xlabel('f(Hz)'); grid;

axis([0 2000 0 2000])

figure;

pspectrum(y,Fs,'spectrogram', ...

'FrequencyLimits',[0 2000],'TimeResolution',0.1)

**MATLAB Code of Problem6.mlx:**clear; clc; close;

% Problem 6:

filename = 'Tone1.wav';

[y,Fs] = audioread(filename);

sound(y,Fs);

t = linspace(0,10,length(y));

pause(2);

y = double(y);

Len\_y = length(y);

p = abs(fft(y));

f = (Fs/Len\_y)\*(1:Len\_y);

figure; subplot(211); plot(t,y); grid

title('Signal/tone'); ylabel('y(t)'); xlabel('t(sec)');

subplot(212); plot(f,p); title('FFT of Signal');

ylabel('|Y(jw)|'); xlabel('f(Hz)'); grid;

axis([0 2000 0 2000])

figure;

pspectrum(y,Fs,'spectrogram', ...

'FrequencyLimits',[0 2000],'TimeResolution',0.1)

**MATLAB Code of Problem7.mlx:**clear; clc; close;

% Problem 7:

filename = 'Test2.wav';

[y,Fs] = audioread(filename);

sound(y,Fs);

t = linspace(0,10,length(y));

pause(2);

y = double(y);

Len\_y = length(y);

p = abs(fft(y));

f = (Fs/Len\_y)\*(1:Len\_y);

figure; subplot(211); plot(t,y); grid

title('Signal/tone'); ylabel('y(t)'); xlabel('t(sec)');

subplot(212); plot(f,p); title('FFT of Signal');

ylabel('|Y(jw)|'); xlabel('f(Hz)'); grid;

axis([0 2000 0 2000])

figure;

pspectrum(y,Fs,'spectrogram', ...

'FrequencyLimits',[0 2000],'TimeResolution',0.1)