Digital Signal Processing Course Project Generation and Detection of DTMF Signals

Xie Kunyang, 2017200501038

April 17, 2020

Abstract

I proudly present a cool project about the generation and detection of Dual-Tone Multi-Frequency (DTMF) Signals. We can use the sine or cosine function provided by MatLab directly to generate the input signals; besides, we can also generate it by digital oscillator. The cosine function can be generated by the recursive difference equation whose frequency domain is the z transformation of that cosine function. In this case, the impulse response of that system is the cosine function. As for the detection part, analyze the frequency spectrum is a good method, the DTFT of the input signal shows that the spectrum has four peaks, and the two on the left correspond to the frequency of the input signals. Furthermore, Goertzel Algorithm is another good method to detect the frequency components. This common approach to this estimation problem is to compute the DFT samples close to the seven fundamental tones. For a DFT-based solution, it has been shown that using 205 samples in the frequency domain minimizes the error between the original frequencies and the points at which the DFT is estimated [1].

1 Introduction

2 Problem Formulation

DTMF signals are widely used in telephone systems, 4 low-frequency signals and 4 high-frequency signals consist a 4×4 matrix, which has 16 intersections, each key represents for an intersection. If one key is pressed, the corresponding 2 signals will be superposed together as the input of the system.

In the first task, we need to generate the input signal by MatLab, there are lots of methods to generate the input signal, we can use the sine and cosine functions that come from MatLab or digital oscillator.

As for the second task, we're going to design an approach to separate the two superimposed cosine waves, in this report, we give two methods, analyzing the frequency spectrum and Goertzel Algorithm.

2.1 DTMF Signals Generation

The sine and cosine function can be generated by `y = cos(x);` directly, besides, it can also be generated by a digital oscillator, in this case, we know that the z transformation of a cosine function is:

$$\frac{Y(z)}{X(z)} = H(z) = \sum_{n=-\infty}^{+\infty} \cos[\omega_0 n] u[n] z^{-n} = \frac{1 - \cos\omega_0 z^{-1}}{1 - 2\cos\omega_0 z^{-1} + z^{-2}}$$

If we regard this as a system H(z), and if the initial condition is all zero, the reverse z transformation can be described as:

$$y[n] = 2\cos\omega_0 y[n-1] - y[n-2] + x[n] - \cos\omega_0 x[n-1]$$

The impulse response of this system is $\cos[\omega_0 n]$.

We can use this method to generate cosine waves, the block diagram is shown as follow:

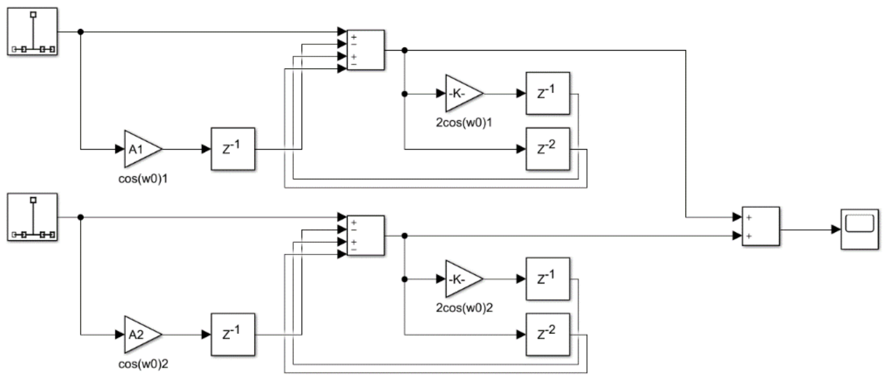

Figure 1 The Block Diagram of the System

Take Key-1 for an example, we see the impulse response is:

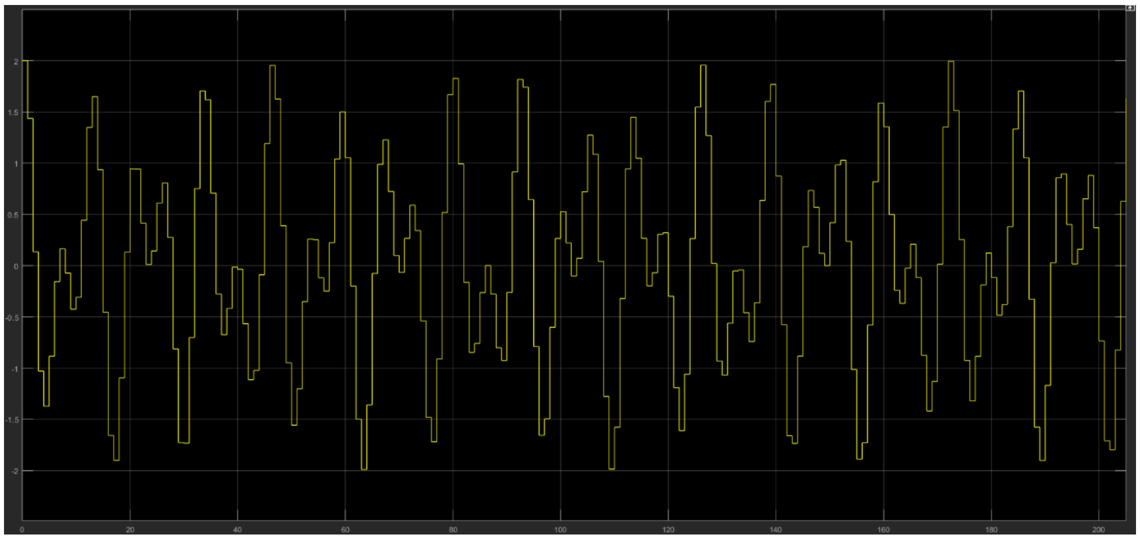

Figure 2 The Output of Simulink

Which is same to the output of MatLab:

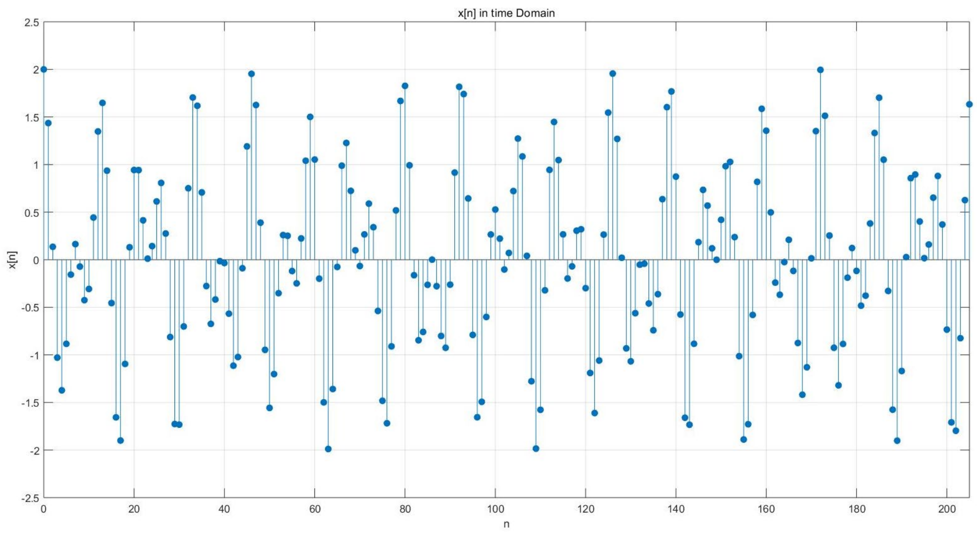

Figure 3 The Output of MatLab

The code is shown in code 1:

Code 1: DTMF Generation:

```
%% Initial Part
clear;
clc;
prompt = 'Input the Number:\n';
key_in = input(prompt);
Ts = 1/8000;
n = 0:205;
w = n/length(n)/Ts;
row = [1209 1336 1477 1633];
col = [697 770 852 941];
%% Input Part
switch key_in
   case 1
      freq1 = row(1,1);
      freq2 = col(1,1);
   case 2
      freq1 = row(1,2);
      freq2 = col(1,1);
   case 3
      freq1 = row(1,3);
      freq2 = col(1,1);
   case 'A'
      freq1 = row(1,4);
      freq2 = col(1,1);
   case 4
      freq1 = row(1,1);
      freq2 = col(1,2);
   case 5
      freq1 = row(1,2);
      freq2 = col(1,2);
   case 6
      freq1 = row(1,3);
      freq2 = col(1,2);
   case 'B'
      freq1 = row(1,4);
      freq2 = col(1,2);
   case 7
      freq1 = row(1,1);
      freq2 = col(1,3);
   case 8
      freq1 = row(1,2);
      freq2 = col(1,3);
   case 9
      freq1 = row(1,3);
      freq2 = col(1,3);
   case 'C'
      freq1 = row(1,4);
      freq2 = col(1,3);
   case '*'
      freq1 = row(1,1);
      freq2 = col(1,4);
   case 0
      freq1 = row(1,2);
      freq2 = col(1,4);
   case '#'
      freq1 = row(1,3);
      freq2 = col(1,4);
   case 'D'
      freq1 = row(1,4);
      freq2 = col(1,4);
   otherwise
      disp('Invalid Input£¡');
end
%% Processing by cos()
% x = cos(2*pi*freq1.*n*Ts) + cos(2*pi*freq2.*n*Ts);
% X = fft(x);
%% Processing by Digital Oscillator
omega1 = 2*pi*freq1*Ts;
omega2 = 2*pi*freq2*Ts;
A1 = cos(omega1);
A2 = cos(omega2);

m = zeros(1,length(n)+2); % Generate the impulse signal
for i = 1:length(n)+2
    if(i == 3)
        m(1,i) = 1;
    end
end

x1 = zeros(1,length(n)+2);    % Generate the input signal
x2 = zeros(1,length(n)+2);

for i = 3:length(n)+2
    x1(1,i) = 2*A1*x1(1,i-1) - x1(1,i-2) + m(1,i) - A1*m(1,i-1);
    x2(1,i) = 2*A2*x2(1,i-1) - x2(1,i-2) + m(1,i) - A2*m(1,i-1);
end

x1(:,1)=[]; % Delete initial condition components
x1(:,1)=[];
x2(:,1)=[];
x2(:,1)=[];
%% Plotting Part
figure
stem(n,real(x),'filled');
title('x[n] in time Domain');
xlabel('n');
ylabel('x[n]');
grid on;
axis([0 256 -2 2]);
```

2.2 DTMF Signals Detection

The output signal is the superposition of 2 cosine waves, by applying the Fourier Transform we can get the frequency component, which are shown as follow:

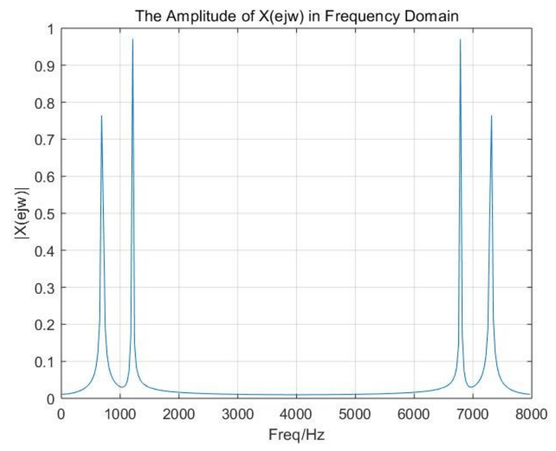

Figure 4 Amplitude in Frequency Domain

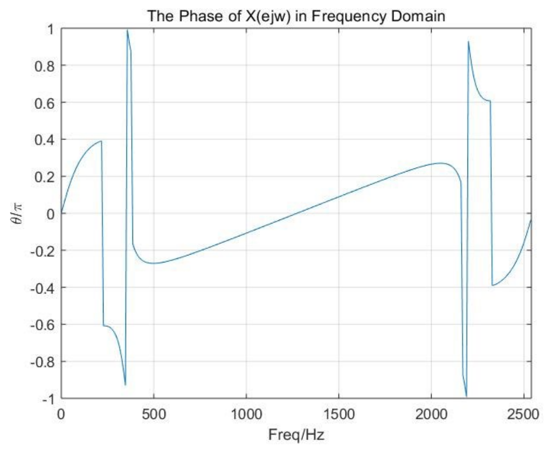

Figure 5 Phase in Frequency Domain

The left two peaks in figure 4 describe the frequency of the signals, which are 697Hz and 1209Hz for Key-1. In this case, the two frequency components are separated, which is beneficial to detected. The code is shown in Code 2:

Code 2: DTMF Detection by FFT:

```
%% Transforming Part
x = x1 + x2;
X = fft(x);
%% Plotting Part
figure
plot(w,abs(X)*2/length(X));
title('The Amplitude of X(ejw) in Frequency Domain');
xlabel('Freq/Hz');
ylabel('|X(ejw)|');
grid on;

figure
plot(w/pi,angle(X)/pi);
title('The Phase of X(ejw) in Frequency Domain');
xlabel('Freq/Hz');
ylabel('\theta/\pi');
grid on;
axis([0 2540 -1 1]);
```

However, this method is very complicated which may increase the calculation load for chips. In this case, Goertzel Algorithm is another good method to detect the frequency components. In MatLab, we can directly use this method by coding `Xgk = goertzel(x,K+1);` where x is the input time domain signal and K is the frequency order vector.

Now we should consider how to find the frequency order vector. We want to reconstruct as close to the true frequency as possible, which should satisfy the equation:

$$f = \frac{T_s k}{N}$$

Where Ts is 8000, f should as close as possible to the frequency in keyboard, k is the component in frequency order vector and they should be integers, finally, we need to find a suitable N to make k as close as an integer which can reduce the error. According to the Wikipedia [1], N is equal to 205, the correspondence is shown in the following table:

Table 1 Value of k

Frequency (Hz)	Real k Value	Value of k in K matrix
697	17.86	18
770	19.53	20
852	21.83	22
941	24.11	24
1209	30.98	31
1336	34.24	34
1477	37.85	38
1677	41.85	42

The block diagram is shown in figure 6:

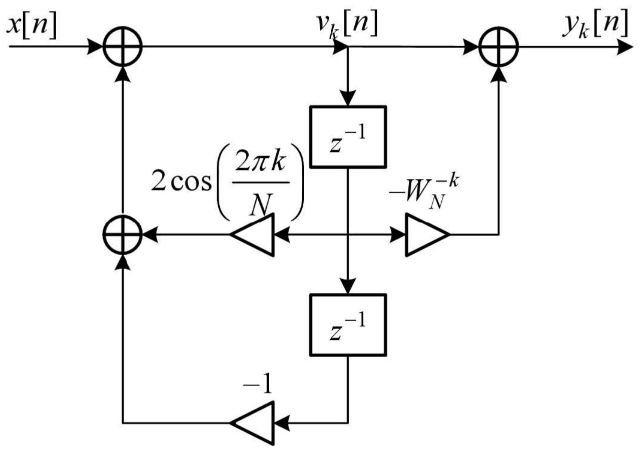

Figure 6 Diagram of Goertzel Algorithm

Finally, we need can plot the result by MatLab, the results are shown here:

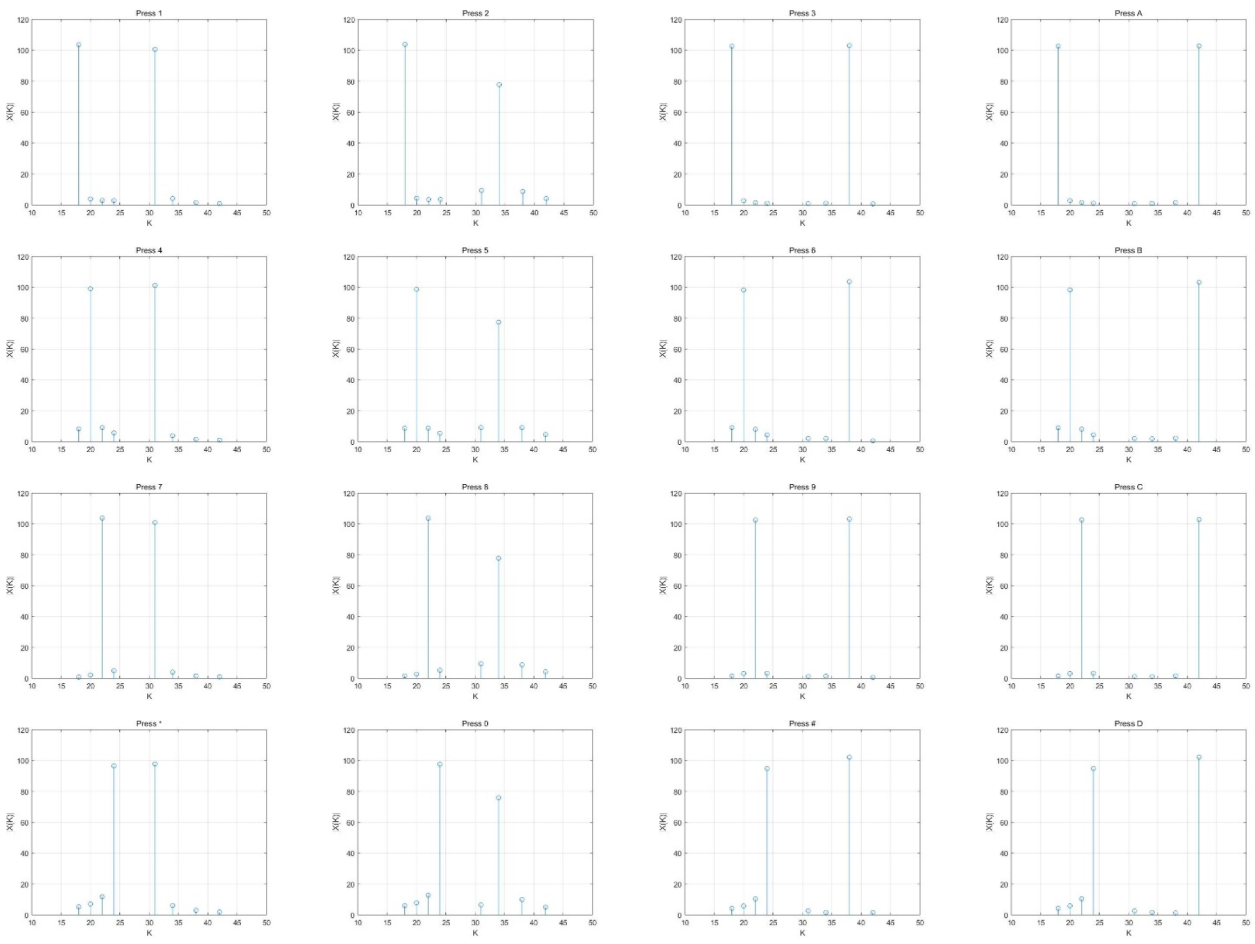

Figure 7 Results of Goertzel Algorithm

Now, different results are corresponding to different keys we press, which have the detection function.

The code is shown as follow in Code 3:

Code 3: Goertzel Algorithm:

```
%% Detecting Part
K = [18 20 22 24 31 34 38 42];
Xgk = goertzel(x,K+1);
figure
stem(K,abs(Xgk));
title('Press 1');
xlabel('K');
ylabel('|X(K)|');
grid on;
axis([10 50 0 120]);
```

3 Concluding Remarks

In this project, we used various methods to achieve the generation, superposition and separation of sine and cosine signals through MatLab and the results are satisfactory.

This project we derive a more profound understanding of the application of Fourier transform and its wide application in life, furthermore, it improves the level of MatLab coding skills.

References

[1] MathWorks Help Center, DFT Estimation with the Goertzel Algorithm, https://www.mathworks.com/help/signal/examples/dft-estimation-with-the-goertzel-algorithm.html, [Accessed on: 20/Apr/2020]
[2] Wikipedia, Goertzel algorithm, https://en.wikipedia.org/wiki/Goertzel_algorithm, [Accessed on: 19/Apr/2020]