Final Project 2015

Tang Qi

M-B5-5515-2

Department:
Electrical and Computer Engineering

Course Code:
ELCE705
Course Title:
Digital Signal Processing

Part I. (50%) Digital Filter for Guitar Wave

In this part, the spectrum of wav file guitar1.wav is analyzed. A digital filter is designed and used to remove interference (a single tone added) from an audio signal.

Signal Access and Exploration

1. **Use MATLAB's wavread command to load the guitar1.wav file:**
2. **Look at the guitar signal in the frequency domain by computing and plotting (in dB) the windowed DFT of various 8192-point blocks of the guitar signal. Please indicate in your report in what range the significant portion of the guitar signal's spectrum lies?**

Solution:

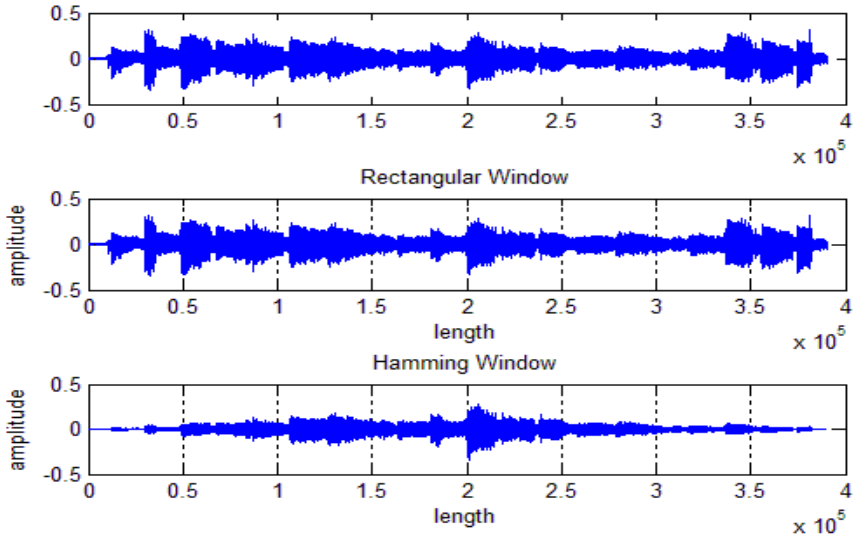

Fig.01 Original sample and truncated by rectangular & hamming window

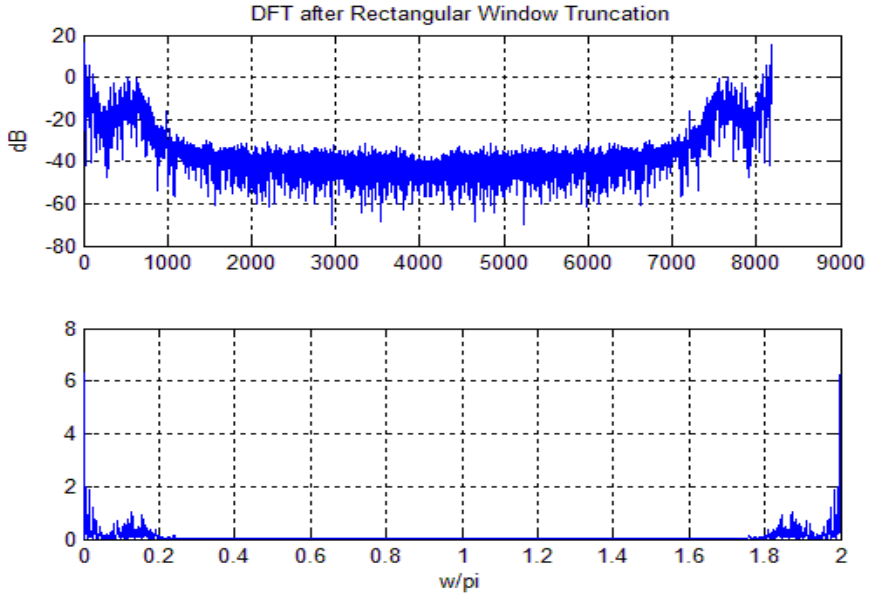

Fig.02 DFT after rectangular window truncation

Before doing the DFT, rectangular window and hamming window is applied to truncate the signal, as shown in Fig.01. Later, rectangular window is chosen to do further processing. The

guitar signal in frequency domain (in dB) after plotting the windowed DFT is as shown in Fig.02. The significant portion of the guitar signal's spectrum lies at [0 0.2pi] and [1.8pi 2pi]. The sound of guitar signal now is pure and clear.

Adding the Interference

1. Create a sinusoid whose frequency is 10kHz that is sampled at the same rate as the guitar signal and has the same length. The amplitude of this sinusoid should be one.

2. Add this signal to the guitar signal to create the simulated recorded signal that has the interference (call this signal y_10 to indicate that it has an interference at 10 kHz).

3. Look at the new signal in the frequency domain. Comment on it.

4. Listen to the new signal.

Solution:

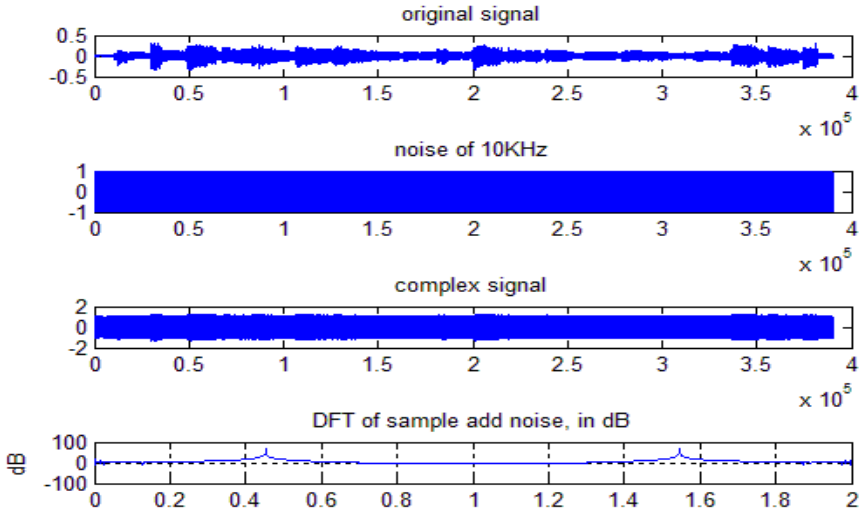

Fig.03 original, interference and complex signals in time domain and complex signal in frequency domain

Comparing the original signal and the new complex signal with sinusoidal interference of 10 KHz by observing Fig.03, we can find the sample guitar is totally buried by the interference in time domain. Listening to the complex sound, we can still tell the audio track of interference and the original sample as they obtain different main frequencies. The main frequency of the original sample signal is generally less than 5 KHz and the frequency of interference is of 10KHz. thus, we need to design a low pass filter which the passband edge frequency ends greater than 5 KHz but less than 10 KHz and stopband edge starts before 10 KHz but later than passband edge frequency.

Filter Design

Design the digital filter satisfying the following requirement:

- passband edge frequency: 7 kHz
- stopband edge frequency: 9 kHz
- passband ripple: 1 dB
- minimum stopband attenuation: 40 dB.

1) Design digital IIR low-pass filters of three types (*Butterworth, Type I Chebyshev, Elliptic*).

- **Write down the exact expression for the system function of the filter in your report.**

Solution:

We can use the matlab functions of corresponding filter to directly get the coefficients of numerator (b) and denominator (a) and thus get the system function H(z).

For Butterworth:

The order of Butterworth filter is 17. Coefficients of corresponding b and a are as shown above

$$\mathrm{H(z)} = \frac{b(z)}{a(z)} = \frac{\mathrm{b1z^{-17}} + \mathrm{b2z^{-16}} + \cdots + \mathrm{b17z^{-1}} + \mathrm{b18}}{a1z^{-17} + \mathrm{a2z^{-16}} + \cdots + \mathrm{a17z^{-1}} + \mathrm{a18}}$$

Corresponding coefficients of b and a are as followed:

```
>>    %% Butterworth filter
Fs=44100;
wp=7000/(Fs/2);% nyquist,1/2 of sampling frequency
ws=9000/(Fs/2);
rp=1;
rs=40;
format long
[n,Wn]=buttord(wp,ws,rp,rs)
[b,a]=butter(n,Wn)
fvtool(b,a);

n =
      17

Wn =

    0.329436991237154

b =
   Columns 1 through 5
    0.000000177046223    0.000003009785796    0.000024078286369    0.000120391431844
0.000421370011454
   Columns 6 through 10
    0.001095562029779    0.002191124059559    0.003443194950735    0.004303993688419
0.004303993688419
   Columns 11 through 15
    0.003443194950735    0.002191124059559    0.001095562029779    0.000421370011454
0.000120391431844
   Columns 16 through 18
    0.000024078286369    0.000003009785796    0.000000177046223

a =
   Columns 1 through 5
```

```
    1.000000000000000   -5.792404676059926   17.575282394679920  -35.799760278837383
54.022332289886066
   Columns 6 through 10
 -63.455910435183497   59.674849064779366  -45.658899861310601   28.662604785635907
-14.799938437242101
   Columns 11 through 15
    6.269443046639585   -2.161725624944414    0.598160983196085   -0.129830902464762
0.021316856935195
   Columns 16 through 18
   -0.002491796735785    0.000184950364189   -0.000006556757495
```

For cheby1:

The system function of Cheby1 filter is:

$$\mathrm{H(z)} = \frac{b(z)}{a(z)} = \frac{\mathrm{b1z^{-8}} + \mathrm{b2}z^{-7} + \cdots + \mathrm{b8z^{-1}} + \ \mathrm{b9}}{a1z^{-8} + \mathrm{a2z^{-7}} + \cdots + \mathrm{a8z^{-1}} + \ +\mathrm{a9}}$$

Corresponding coefficients of b and a are as followed:

```
>> %% Cheby1
Fs=44100;
wp=7000/(Fs/2);% nyquist,1/2 of sampling frequency
ws=9000/(Fs/2);
rp=1;
rs=40;
format long
[n,Wn]=cheb1ord(wp,ws,rp,rs)
[b,a]=cheby1(n,1,Wn)
fvtool(b,a);

n =
        8

Wn =
    0.317460317460317

b =
   Columns 1 through 5
     0.000043945936180   0.000351567489439   0.001230486213036   0.002460972426072
0.003076215532590
   Columns 6 through 9
    0.002460972426072   0.001230486213036   0.000351567489439   0.000043945936180

a =
   Columns 1 through 5
    1.000000000000000   -5.283162046892483   13.580260662755855  -21.758376686051886
23.564429672325677
```

```
   Columns 6 through 9
 -17.590817483886863   8.829889414005237   -2.729880612252208   0.400279966751325
```

For Elliptic:

The system function of the Elliptic filter is:

$$\mathrm{H(z)} = \frac{b(z)}{a(z)} = \frac{\mathrm{b1z^{-5}} + \mathrm{b2z^{-4}} + \cdots + b\mathrm{5z^{-1}} + \mathrm{b6}}{a\mathrm{1z^{-5}} + \mathrm{a2z^{-4}} + \cdots + \mathrm{a5z^{-1}} + \mathrm{a6}}$$

Corresponding coefficients of b and a are as followed:

```
>> Fs=44100;
wp=7000/(Fs/2);% nyquist,1/2 of sampling frequency
ws=9000/(Fs/2);
rp=1;
rs=40;
format long
[n,Wn]=ellipord(wp,ws,rp,rs)
[b,a]=ellip(n,wp,ws,Wn,'low')
fvtool(b,a);

n =
        5
Wn =
     0.317460317460317

b =
   Columns 1 through 5
     0.648941311005988   -0.758890708696803    0.653598369091293    0.653598369091292
-0.758890708696803
   Column 6
     0.648941311005989

a =
   Columns 1 through 5
     1.000000000000000   -1.870915164091089    2.528712313857923   -1.221144570402084
0.352420830065041
   Column 6
     0.298224533371165
```

2) Compute and plot the filter's magnitude and phase response. Does your design meet the specifications?

Solution:

As required, for all of the three filters, the passband edge frequency is 7 KHz and the stopband edge frequency is 9 KHz, satisfying the need that passes certain frequencies, original sample (frequency mainly less than 5 KHz), and stops certain frequencies, interference of 10 KHz, as designed. Thus, the design meets the general specification.

However, the three kinds of filters have own unique characteristics and hence they have been chosen for different applications as desired. Here in this signal processing, comparison will be made later.

For *Butterworth:*

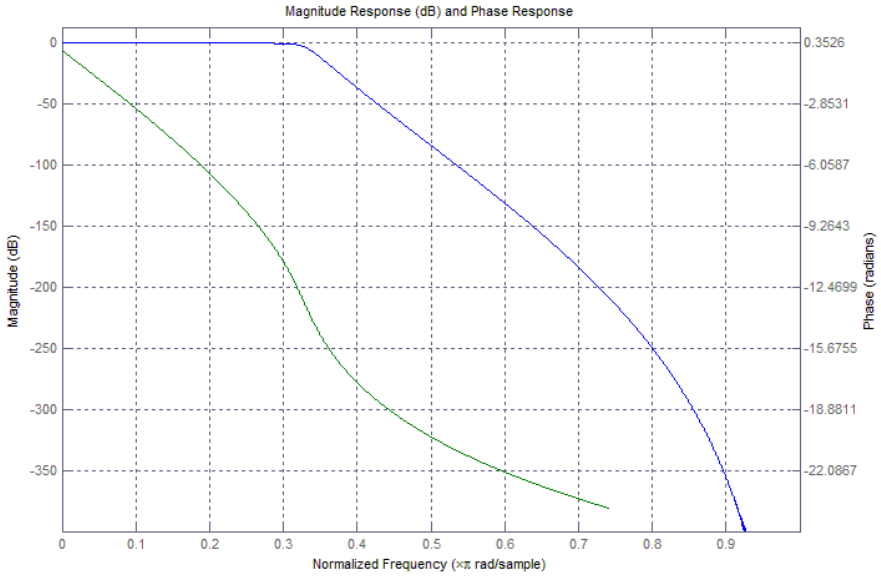

Fig.04 magnitude and phase response of the Butterworth filter

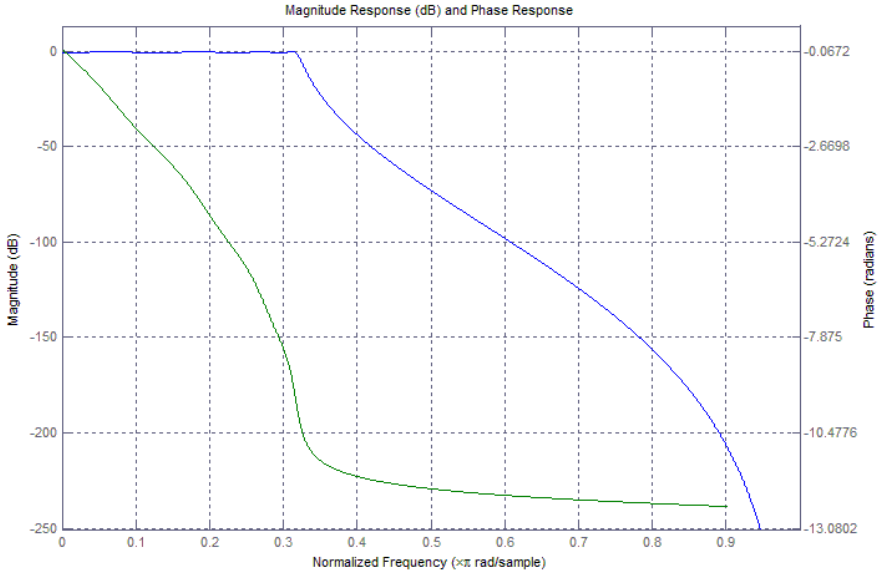

Fig.05 magnitude and phase response of the Cheby I filter

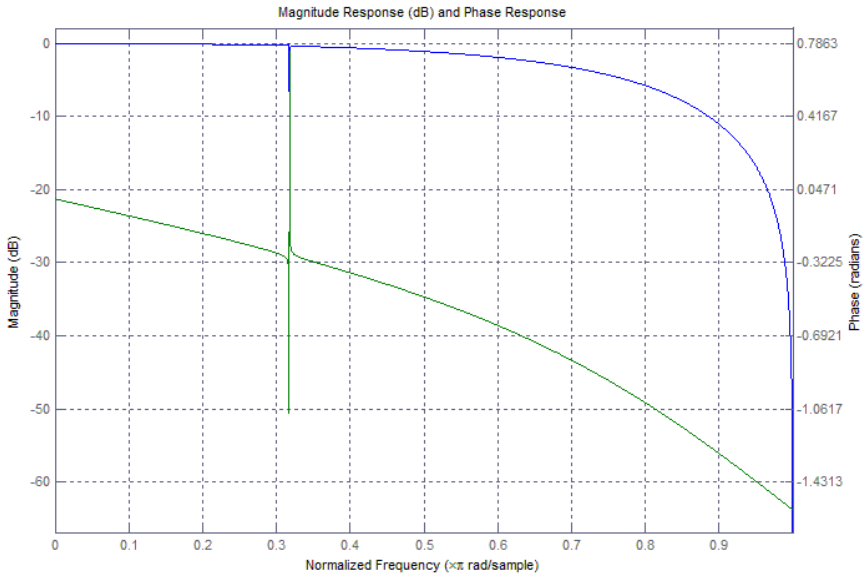

Fig.06 magnitude and phase response of the Elliptic filter

3) Also show the pole-zero plot for the designed filter in your report.

Solution:

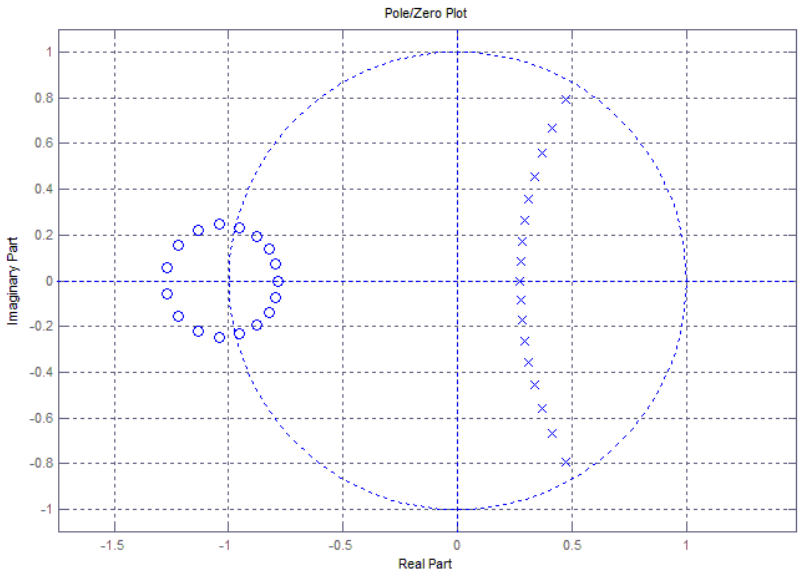

Fig.07 pole-zero plot of the Butterworth filter

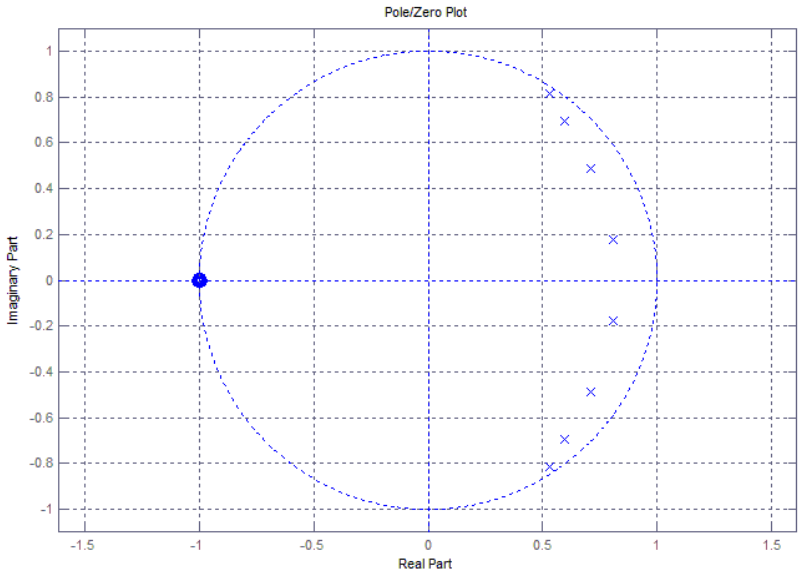

Fig.08 pole-zero plot of the Cheby I filter

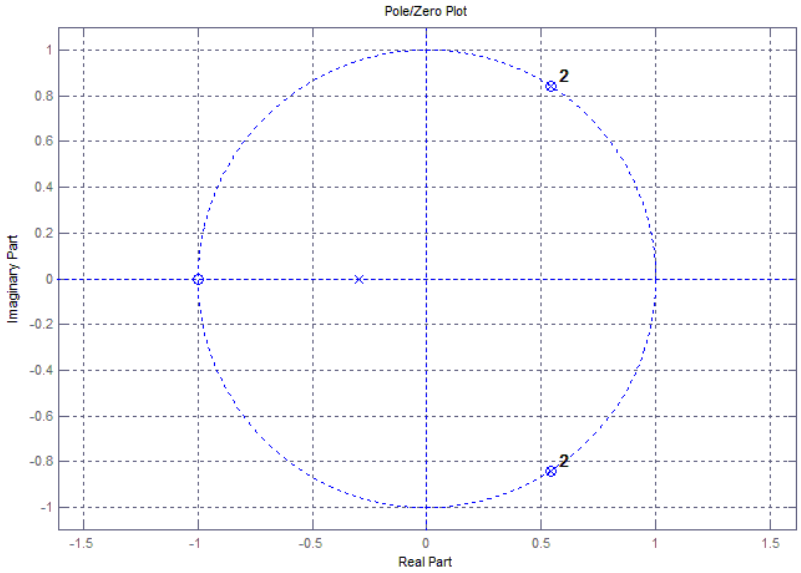

Fig.09 pole-zero plot of the Elliptic filter

4) Compare the three filters and discuss in your report.

Solution:

Observing the frequency response of three filters, we can find that the Butterworth filter has maximally flat response within the passband of the filter and moderate phase distortion. The Cheby I filter have steeper roll-off near cutoff frequency comparing to butterworth filter. But this will result into monotonicity in passband region along with poor transient response. It ripples in passband and has poorer group delay. For the Elliptic filter, the cutoff slope is shaper compare to Butterworth and Chebyshev. But it has ripples in passband and stopband of amplitude response and has very non-linear phase response.

The Butterworth LPF have all the poles and they are located on unit circle with equal angles

while the cheby I filter have all poles and are located on an ellipse inside the unit circle.

5) Use the designed filter to filter the interfered with guitar signal and test their effectiveness. (Use matlab function filter). Compare the two signals in frequency domain. Listen to the filtered signal. Comment on your observations.

Solution:

Using the function of "filter(b,a,signal)",we can get the three groups of filter signal comparison before and after filtering. Corresponding results are as followed:

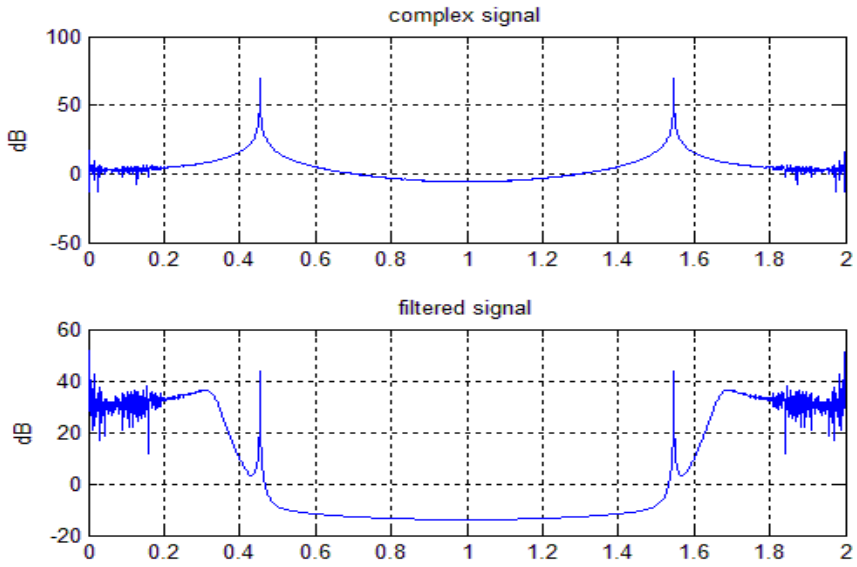

Fig.10 signal before and after Butterworth filter

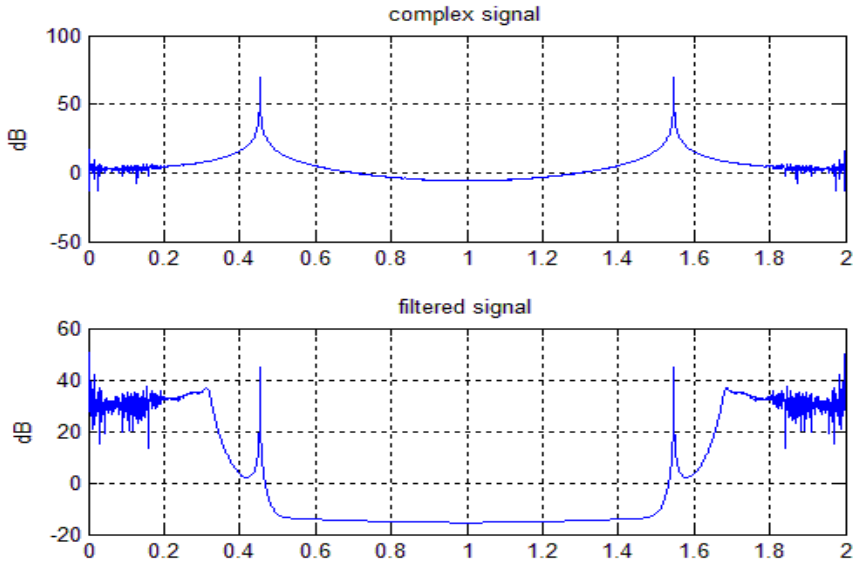

Fig.11 signal before and after Cheby I filter

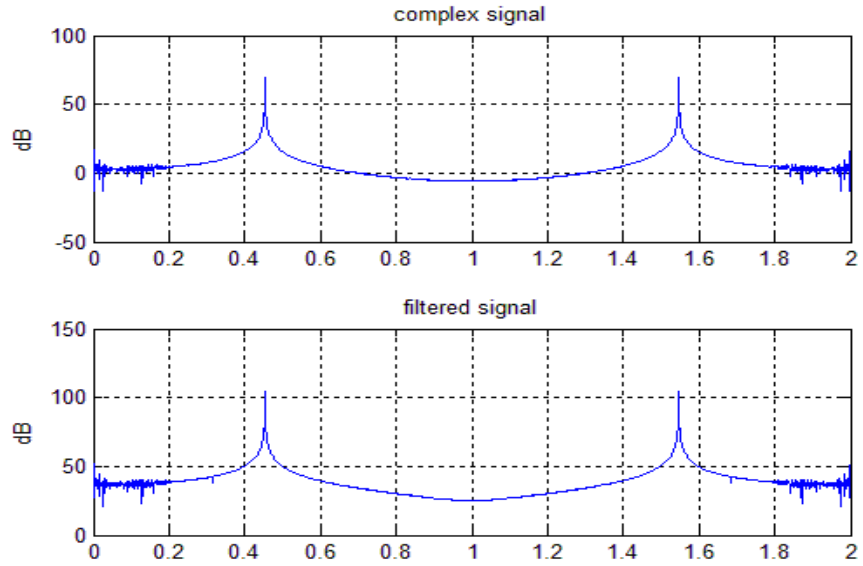

Fig.12 signal before and after Elliptic filter

Doing the filtering, we should care about the DC gain of the transfer function in numerator. After several trials, here I apply 57.87.By hearing, the Butterworth and Cheby type I filter preforms quite good to me. The sinusoidal interference has been reduced to a certain extent. However, the clearing of interference is not exhaustive; I can still tell the background interference. To the Elliptic filter here, the filtering effectiveness is the worst. Maybe because the phase responses of these filters are all non-linear, signals will have distortion. Among all the three filters, the elliptic filter has the most non-linear phase response, which leads to not clean filtering result here. Compensation should be added, which will increase the complexity of the digital filter. Or FIR filter should be applied.

2) Design digital FIR filters using two different windows.

• Provide the shape and length of the selected window, explain why?

Solution:

The passband edge is at 7 kHz. The stopband edge is at 9 kHz. The passband ripple is smaller than 1dB. The smallest stopband attenuation is 40dB. Corresponding attenuation parameters for frequently-used windows are as shown:

window	Stopband Attenuation /dB	Approximate transition bandwidth	Precise transition bandwidth
Rectangular	21	4*pi/N	1.8*pi/N
Hanning	44	8*pi/N	6.2*pi/N
Hamming	53	8*pi/N	6.6*pi/N
Blackman	74	12*pi/N	11*pi/N
Kaiser	80	10*pi/N	10*pi/N

Table.1 attenuation parameters of windows

Considering the stopband attenuation, Hanning and Hamming window are chosen.

• Compute and plot the filter's magnitude and phase response. Does your design meet the

specifications?

Solution:

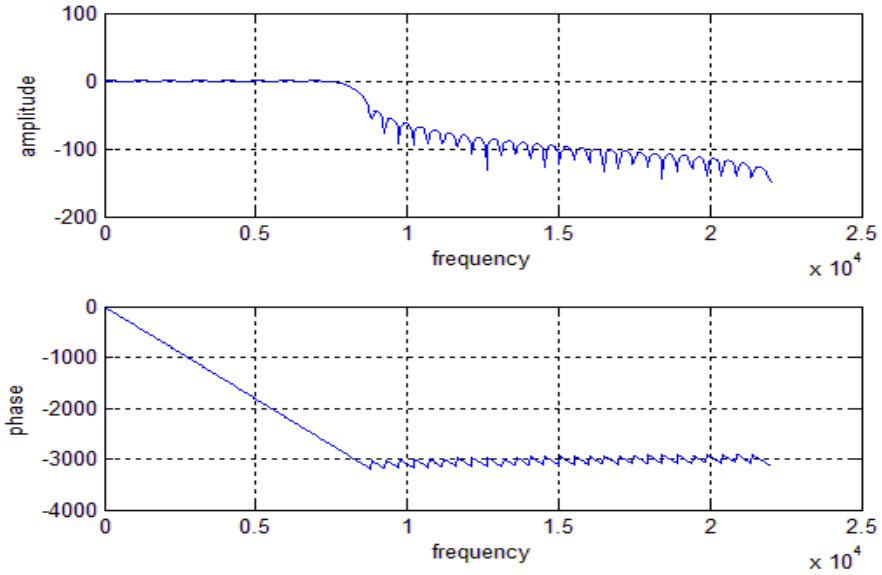

Fig.13 magnitude and phase response applying Hannning window

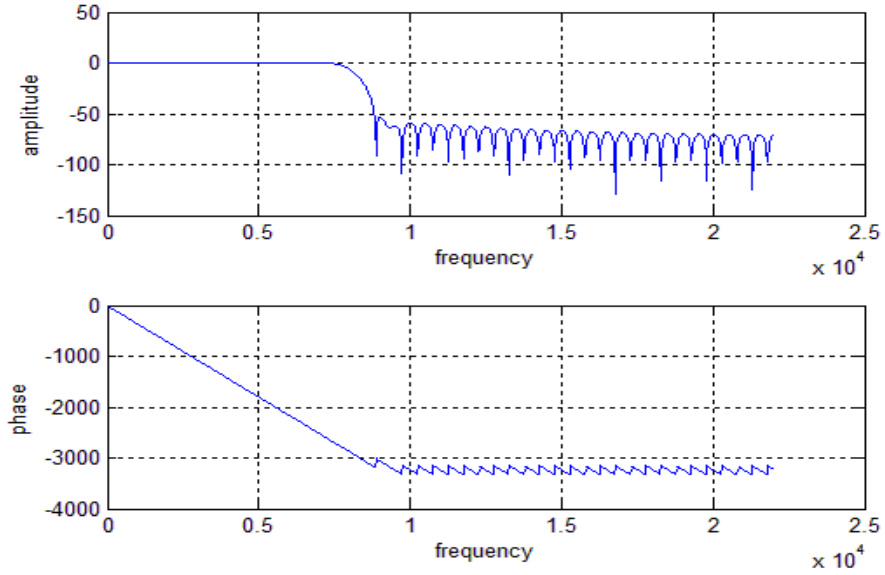

Fig.14 magnitude and phase response applying Hamming window

The FIR filter using Hanning meets the specification as the wp set to be 7000/(Fs/2) and ws set to be 9000/(Fs/2).

• Use the designed filter to filter the interfered with guitar signal and test their effectiveness. (Use matlab function filter). Compare the two signals in frequency domain. Listen to the filtered signal. Comment on your observations.

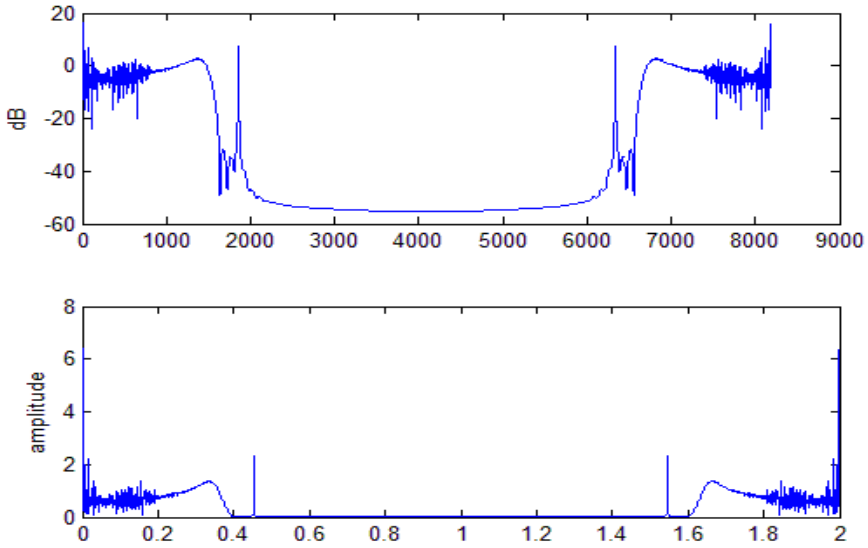

Fig.15 signal before and after filtering by applying Hanning window

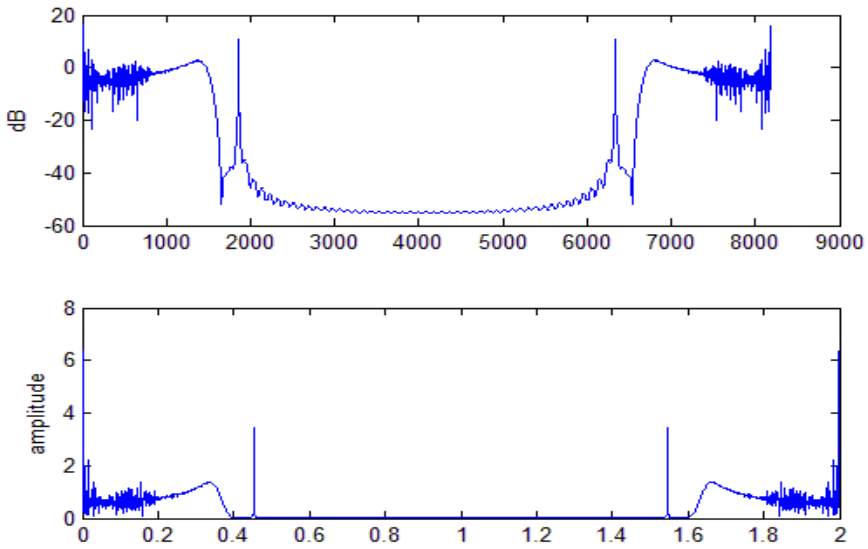

Fig.16 signal before and after filtering by applying Hamming window

After applied the FIR filter with designed Hannning and Hamming window, the complex signal got cleared. And the effectiveness of the FIR filter is way better than the IIR filter of Butterworth, Cheby I and Elliptic filter as all of them have non-linear phase response leading to distortion.

Part II. Decode DTMF

Touch-tone telephones use a dual tone multi frequency (DTMF) scheme to encode key-presses as audio tones. Thus for every key pressed a unique combination of two distinct audio tones is created, with frequencies specified in the following table:

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

The aim of this part is to first remove the noise to get the DTMF audio tones. Secondly, investigate and implement methods for decoding a sequence of key presses.

1. De-noiser Filter Design

A. Analyze the signal in frequency domain. Indicate the frequencies of the noises in your report.

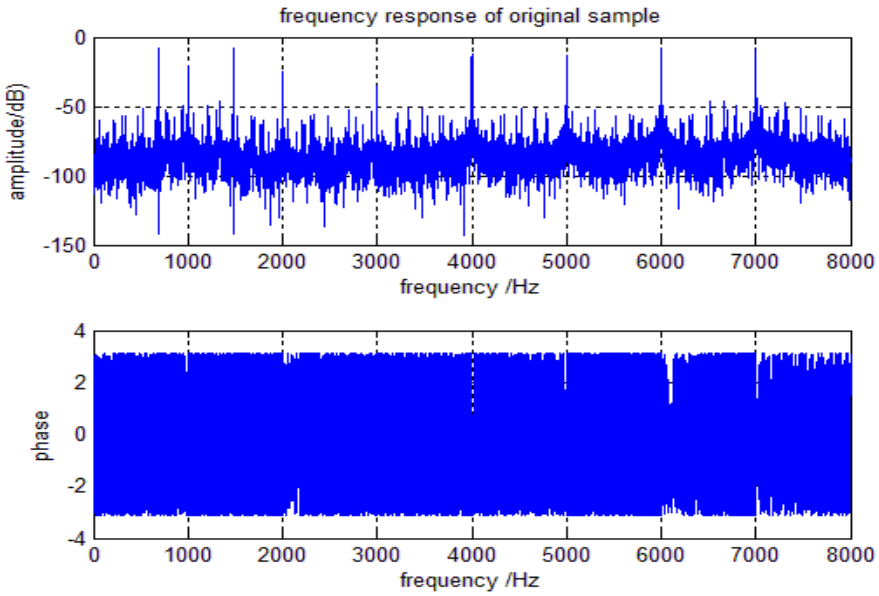

Fig.17 signal in frequency domain

Observing the magnitude response of the original sample signal, we can find that the signal is interfered regularly after 3000 Hz, where the waveforms interfered in every 1000 Hz since then. Still, there are interferences before 3000 Hz, at 1000Hz. In addition, from the signal forms in time domain, as shown in Fig.18, they are only related to the frequency. Thus, two bandpass filters should be applied to do the de-noising.

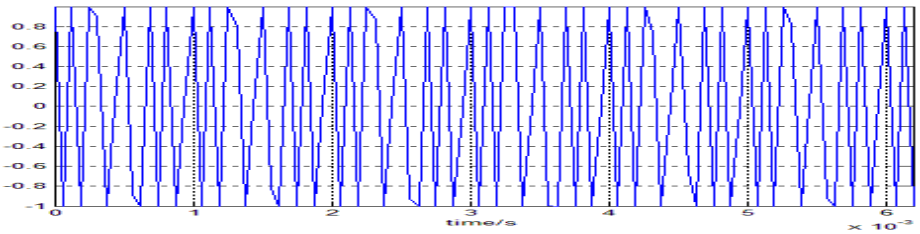

Fig.18 original sample signal in time domain

B. Design a discrete-time filter to filter the noise.

1) List the specifications of the digital filter in your report.

Because the desired signal is composed of two frequencies ranges from 697 Hz to 1633 Hz. From the frequency response of the original sample, there are several interferences inside and outside ranges from 697 Hz to 1633 Hz. Thus, we need a bandpass filter.

For the first bandpass filter:

Specifications:

Passband: from 697Hz to 941Hz; Stopbands: smaller than 397 Hz, larger than 1241Hz; Maximum passband ripple: 3 dB; Minimum stopband attenuation: 60dB;

For the first bandpass filter:

Passband: from 1209 Hz to 1633Hz; Stopbands: smaller than 909 Hz, larger than 1933 Hz; Maximum passband ripple: 3 dB; Minimum stopband attenuation: 60dB;

2) Provide the following of the designed filter in your report:

a. Plot the impulse response if a FIR filter is designed.

Referring to Table.1, parameters of windows, we can find the Blackman and Kaiser windows both satisfy the specifications. In our course, Kaiser is more illustrated. Thus, the Kaiser Window FIR bandpass filter is applied.

Specifications :

Delta_w= ws-wp= 300/Fs*2*pi;
Normalized Delta_w = 0.2356;
M=(A-8)/(2.285* Delta_w);
beta= 0.1101*(A-8.7), as A>50 from A=-20log10 epsilon;
wn=kaiser(M,beta);
 	
ideal bandpass filter impulse response :
hn=max(cut_off_w)./pi.*sinc(max(cut_off_w)./pi.*(n-(M-1)/2))-min(cut_off_w)./pi.*sinc(min(cut_off_w)./pi.*(n-(M-1)/2));

Kaiser Window FIR filter impulse response:

```
hn_Kai=hn.*wn';
```

For the first Kaiser window FIR bandpass filter:

```
f_min=697;
f_max=941;
```

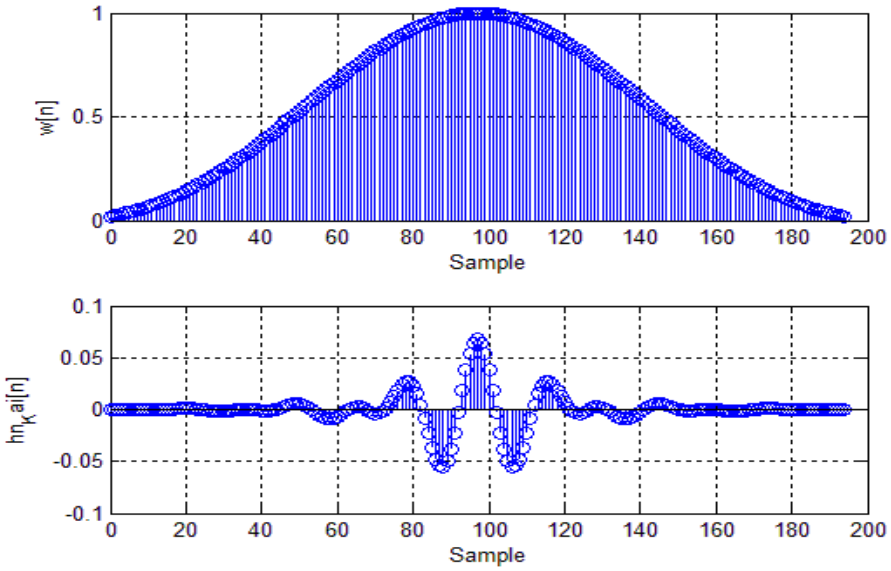

Fig.19 impulse response of designed Kaiser Window bandpass filter 1

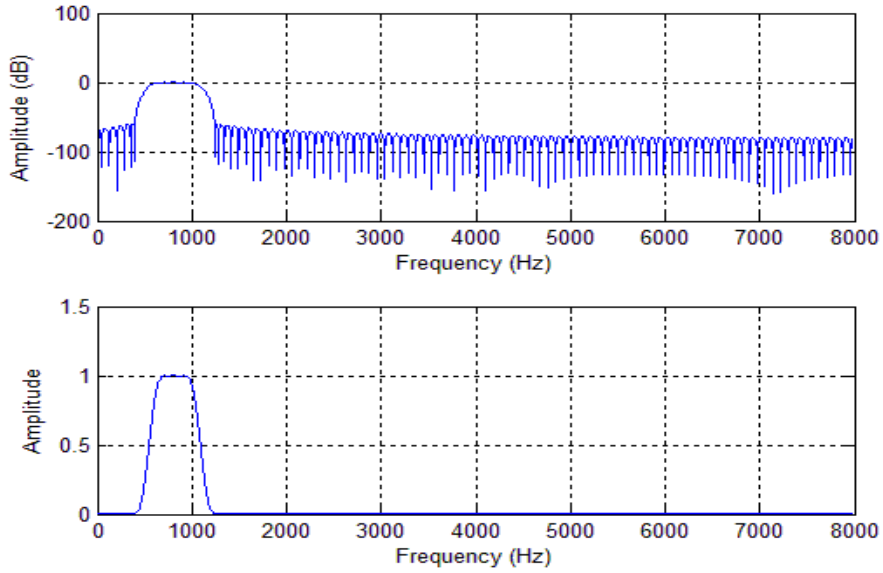

Fig.20 magnitude response of Kaiser Window bandpass filter 1

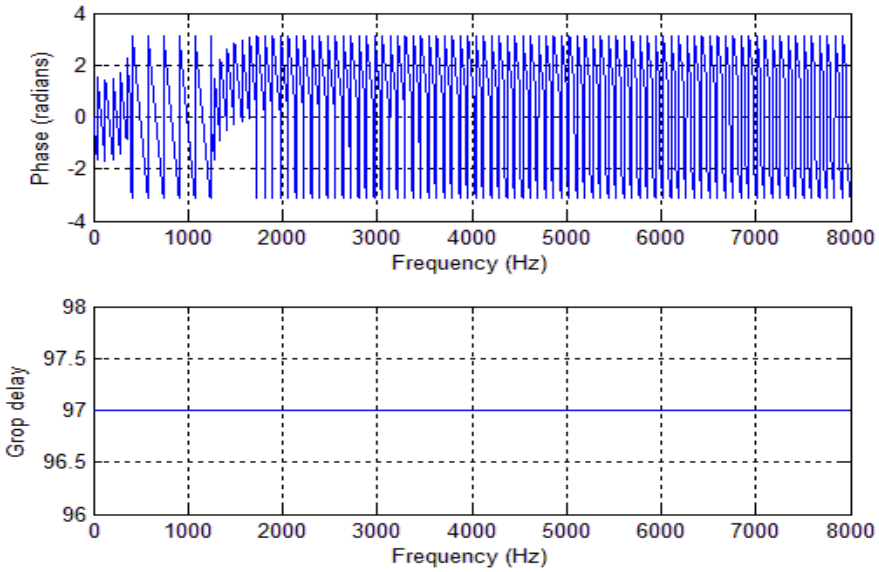

Fig.21 phase response and group delay of Kaiser Window bandpass filter 1

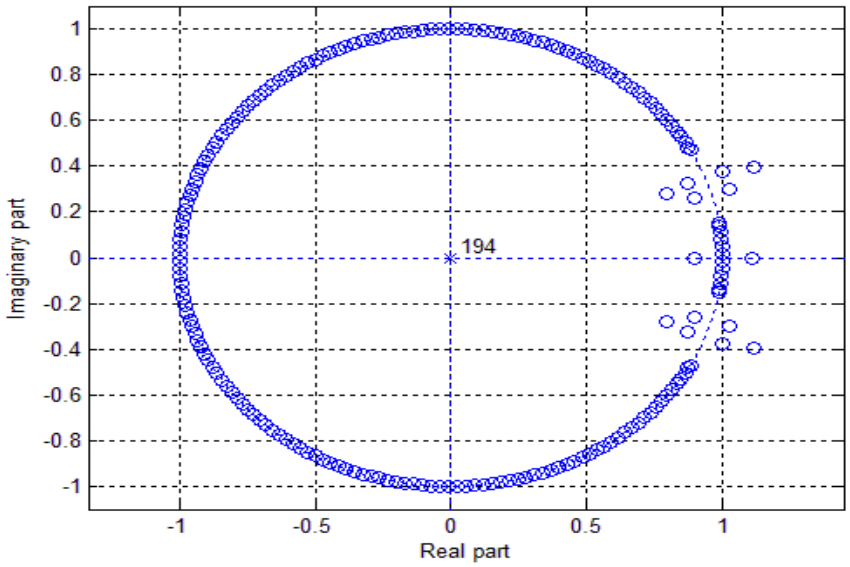

Fig.21 pole-zero plot of Kaiser Window bandpass filter 1

For the second Kaiser window FIR bandpass filter:

```
f_min=1209;
f_max=1633;
```

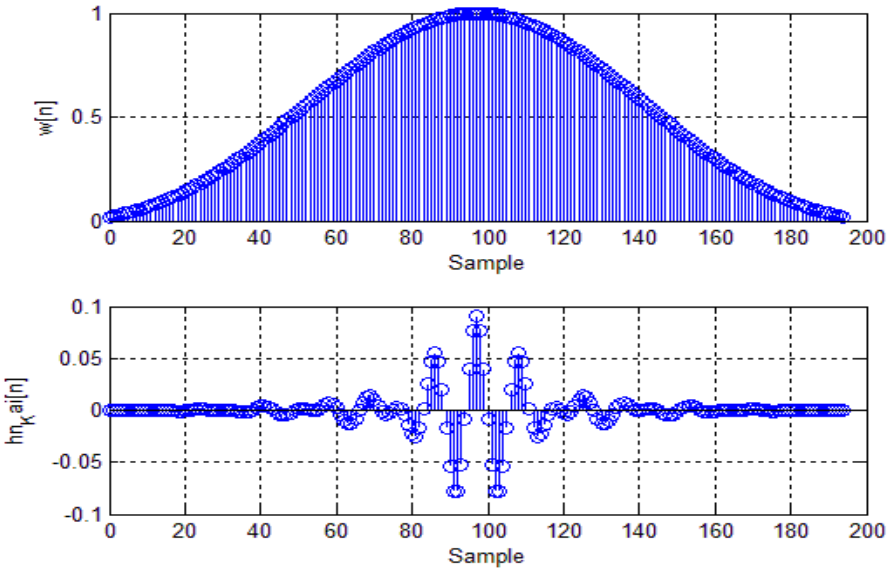

Fig.22 impulse response of designed Kaiser Window bandpass filter 2

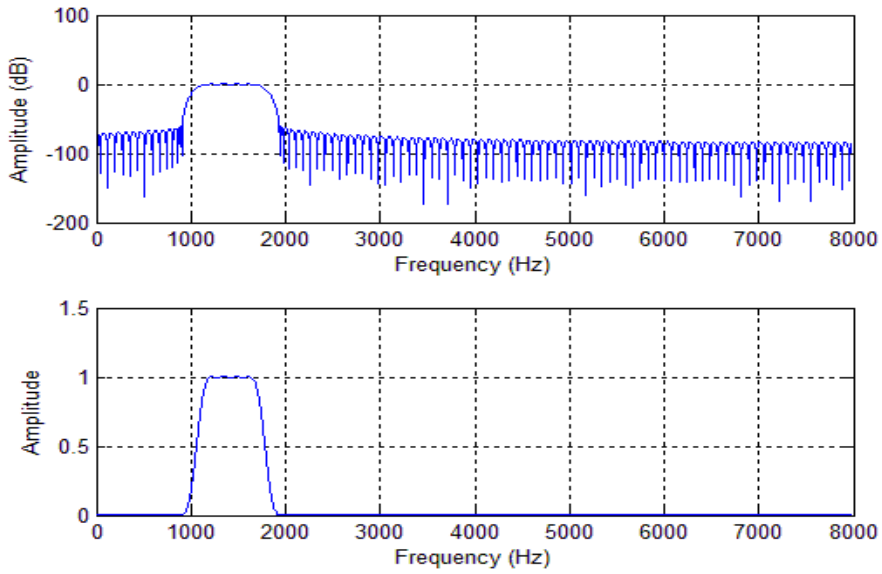

Fig.23 magnitude response of Kaiser Window bandpass filter 2

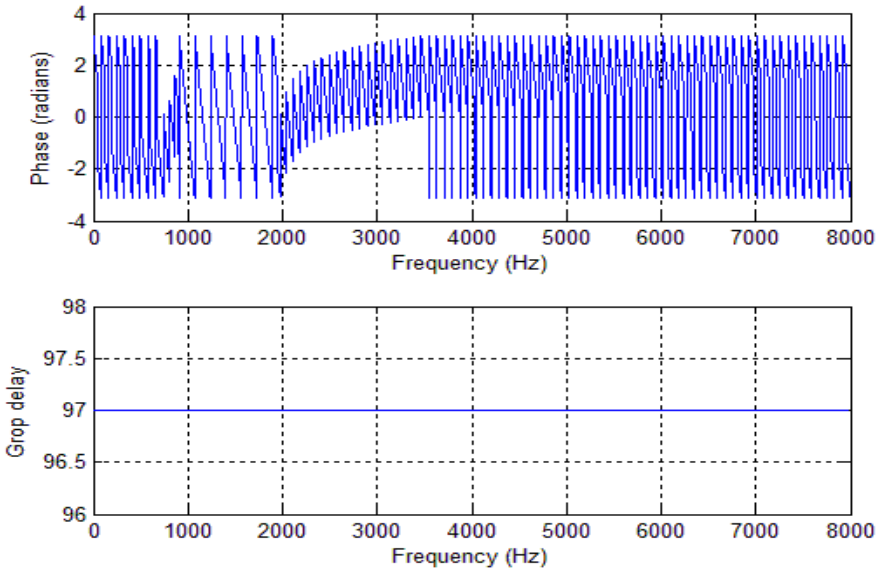

Fig.24 phase response and group delay of Kaiser Window bandpass filter 2

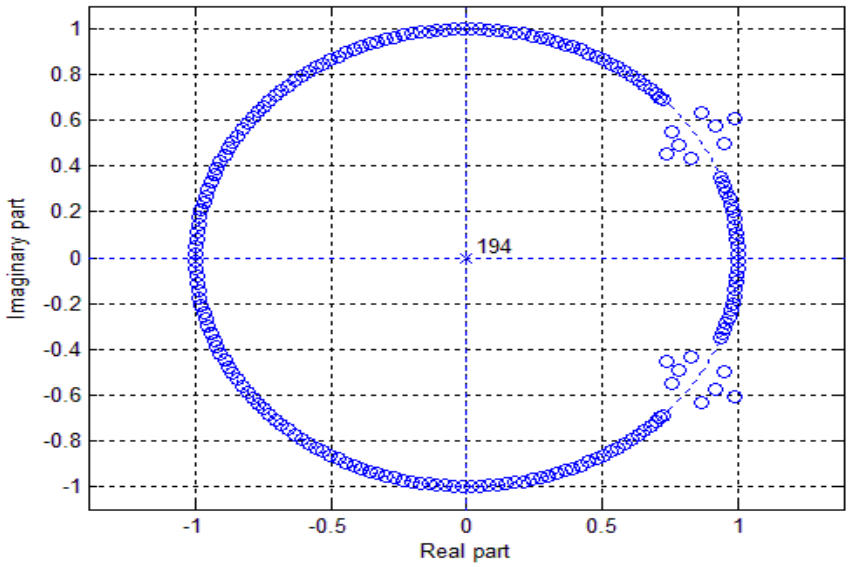

Fig.25 pole-zero plot of Kaiser Window bandpass filter 2

b. Give the system function H(z) if a IIR filter is designed.

Here I choose to use two Butterworth bandpass filters:

For Butterworth bandpass filter I:

The system function is

$$\mathrm{H(z)} = \frac{b(z)}{a(z)} = \frac{\mathrm{b1z^{-14} + b2z^{-13} + \cdots + b14z^{-1} + b15}}{a1z^{-14} + \mathrm{a2z^{-13} + \cdots + a14z^{-1} + +a15}}$$

Corresponding coefficients of b and a are as followed:

```
n =
     7
wn =
```

```
    0.0858    0.1194

b =

   1.0e-07 *

  Columns 1 through 8

    0.0088         0   -0.0618         0    0.1855         0   -0.3092         0

  Columns 9 through 15

    0.3092         0   -0.1855         0    0.0618         0   -0.0088


a =

   1.0e+03 *

  Columns 1 through 8

    0.0010   -0.0128    0.0773   -0.2884    0.7460   -1.4145    2.0273   -2.2311

  Columns 9 through 15

    1.8946   -1.2354    0.6089   -0.2200    0.0551   -0.0086    0.0006
```

For Butterworth bandpass filter 2:

The system function is

$$\mathrm{H(z)} = \frac{b(z)}{a(z)} = \frac{\mathrm{b1z^{-14}} + \mathrm{b2}z^{-13} + \cdots + \mathrm{b14z^{-1}} + \ \mathrm{b15}}{a1z^{-14} + \mathrm{a2z^{-13}} + \cdots + \mathrm{a14z^{-1}} + \ + \mathrm{a15}}$$

Corresponding coefficients of b and a are as followed:

```
n =

     9

wn =

    0.1507    0.2046

b =

   1.0e-07 *

  Columns 1 through 8

    0.0014         0   -0.0128         0    0.0512         0   -0.1194         0

  Columns 9 through 16

    0.1791         0   -0.1791         0    0.1194         0   -0.0512         0

  Columns 17 through 19

    0.0128         0   -0.0014


a =

   1.0e+04 *

  Columns 1 through 8

    0.0001   -0.0014    0.0101   -0.0455    0.1462   -0.3576    0.6897   -1.0716

  Columns 9 through 16

    1.3593   -1.4181    1.2195   -0.8625    0.4981   -0.2317    0.0850   -0.0237

  Columns 17 through 19

    0.0047   -0.0006    0.0000
```

c. The number of multiplication operations per input sample required to implement the filter.

Both numerator and denominator do multiplication. And here we use two Butterworth bandpass filter. Counting the number of multiplications of the transfer function,

14+14+18+18=64

d. Plot the magnitude (in dB) and phase response of the filter.

For Butterworth bandpass filter I:

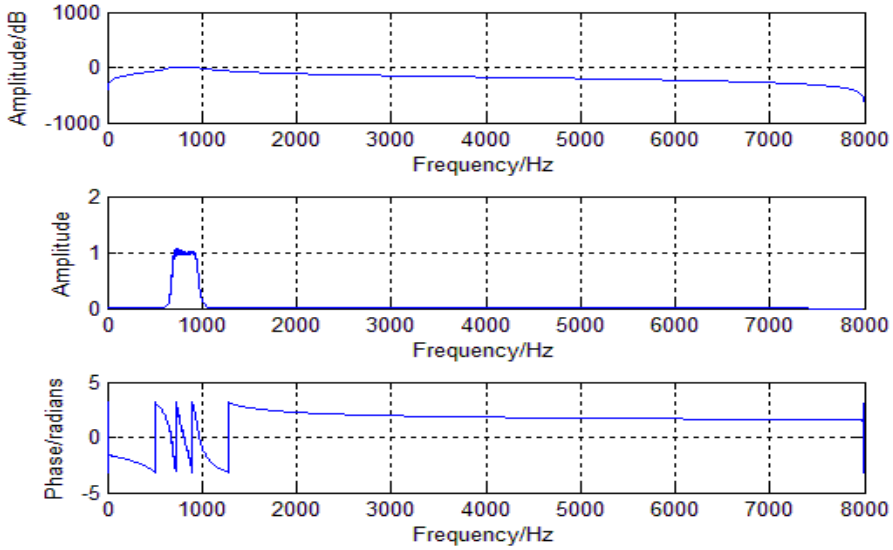

Fig.26 frequency response of Butterworth bandpass filter 1

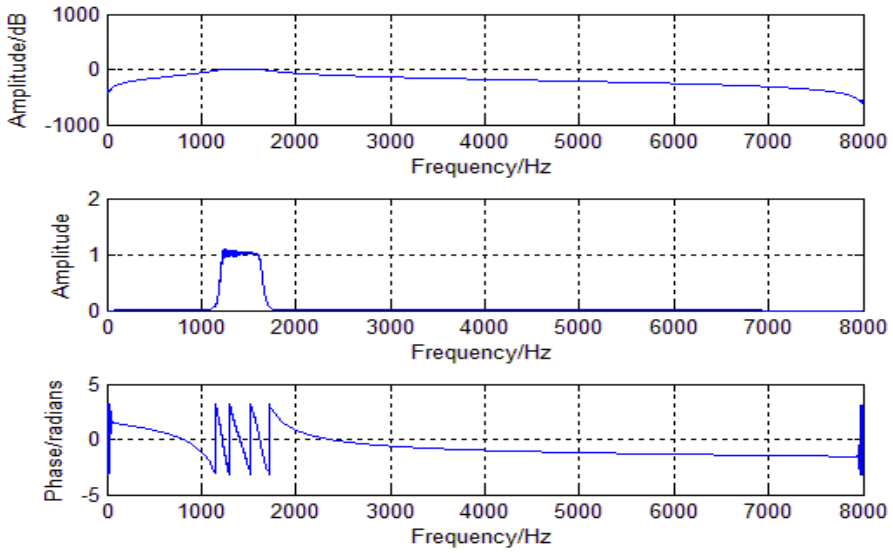

Fig.27 frequency response of Butterworth bandpass filter 2

e. Plot a detail of the magnitude response, focusing on the passband ripple (linear scale).

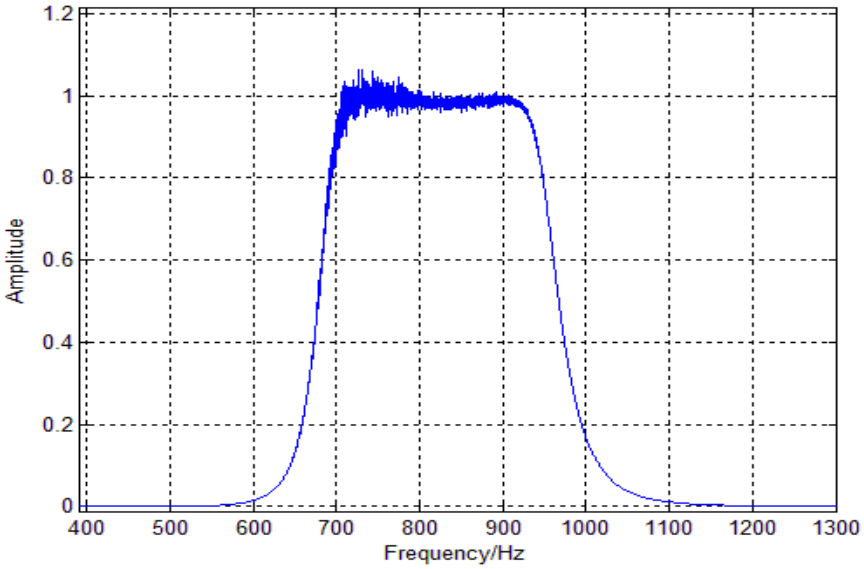

Fig.28 detailed magnitude response of Butterworth bandpass filter 1

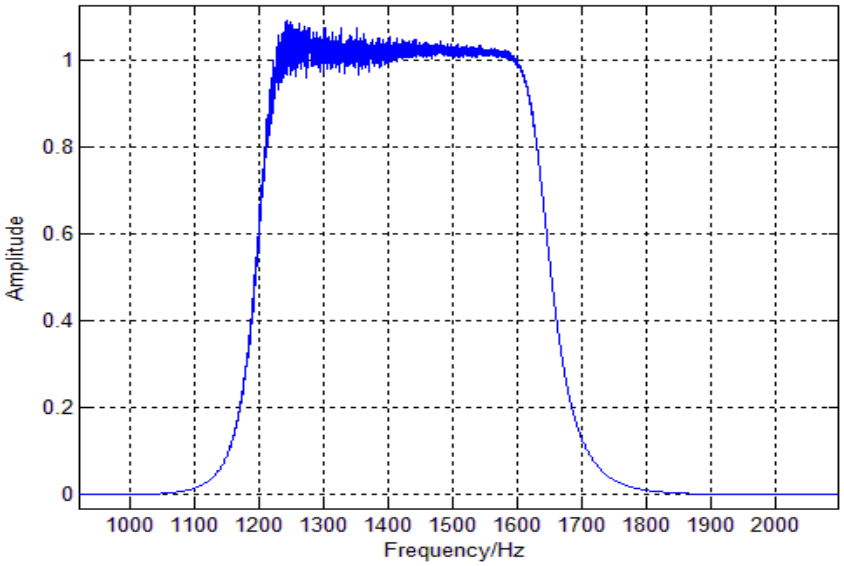

Fig.29 detailed magnitude response of Butterworth bandpass filter 2

f. Plot the group delay (in samples) using grpdelay.

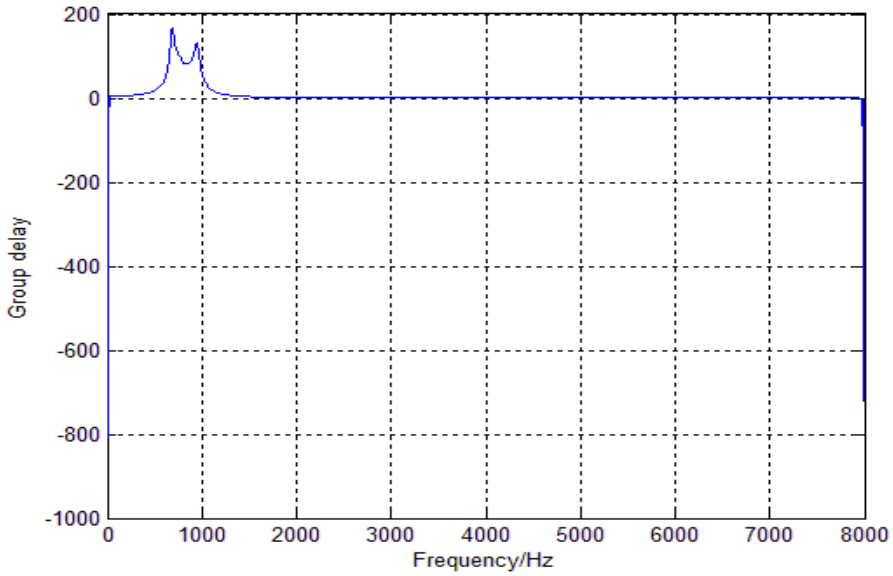

Fig.30 group dealy of Butterworth bandpass filter 1

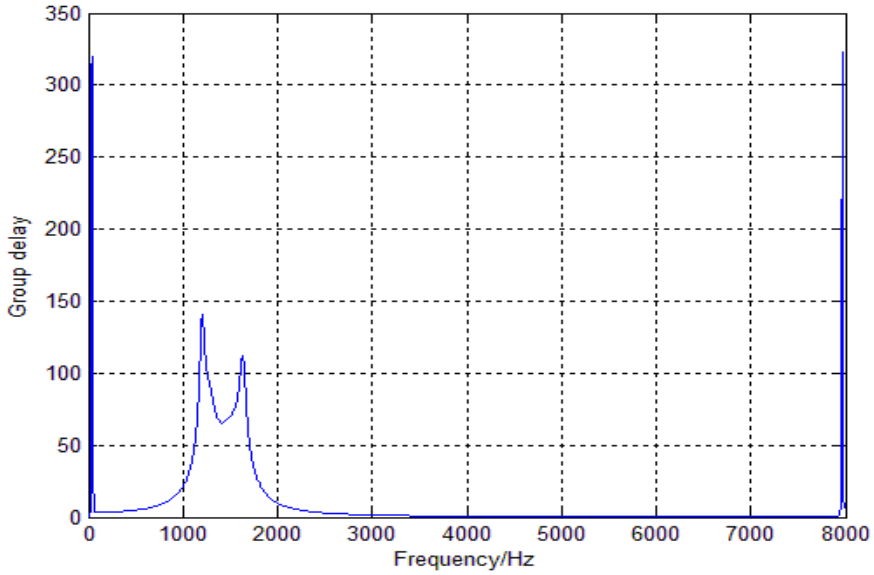

Fig.31 group dealy of Butterworth bandpass filter 2

g. Plot the pole-zero plot.

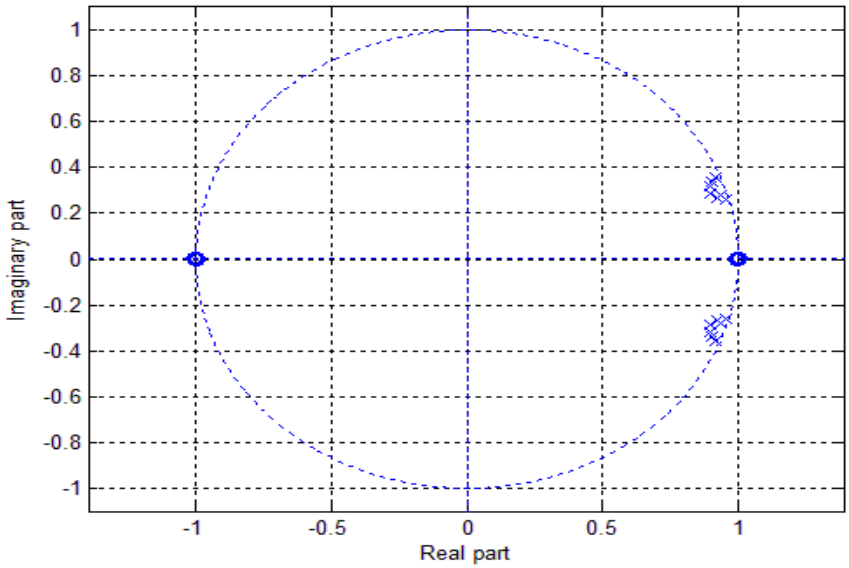

Fig.32 pole-zero plot of Butterworth bandpass filter 1

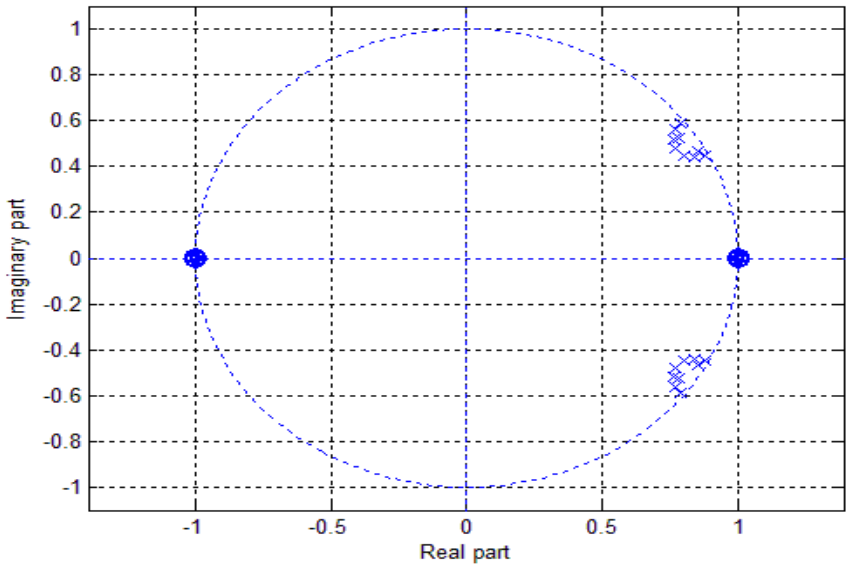

Fig.32 pole-zero plot of Butterworth bandpass filter 2

3) Explain why you select this type of the filter.

Observing the original sample signal, the interferences exist separately, generally greater than 3000Hz, but also interference at 1000Hz. Fortunately, the frequency domain of the desired signal ranges from 697Hz to 941Hz and 1209 Hz to 1633 Hz. Thus, we can choose two bandpass filters or one LFP and one bandstop filter.

Here two bandpass filter is chosen. And butterworth filter applied because it has maximally flat response within the passband of the filter and moderate phase distortion

4) Filter the audio signal using your de-noising filter. Listen to the filtered and original signals. Compare the performance and comment on your results.

Separately get the bandpass signal, using:

```
filtered_y1=filter(b1,a1,y);
filtered_y2=filter(b2,a2,y);
fil_y= filtered_y1+ filtered_y2;
sound(fil_y,Fs)
```

The filtered signal is much clearer than the original sample that the loud and sharp noise of “Di” has been cleaned. After filtering, we can hear the sounds when we call people on a telephone. The designed butterworth bandpass filter performs very good here.

2 Interpret the number by analyze the audio signal.

Find a method to analyze the audio signal and find the 12 keys, which are pressed in the given audio file. Describe your method clearly in the report, with figures, equations to support your discussions.

After filtering, we can clearly hear that there are 12 presses. The filtered signal is applied to analyze which are pressed at certain time and sequences. Then the total audio can be divided into 12 parts based on the time interval and the signal amplitude.

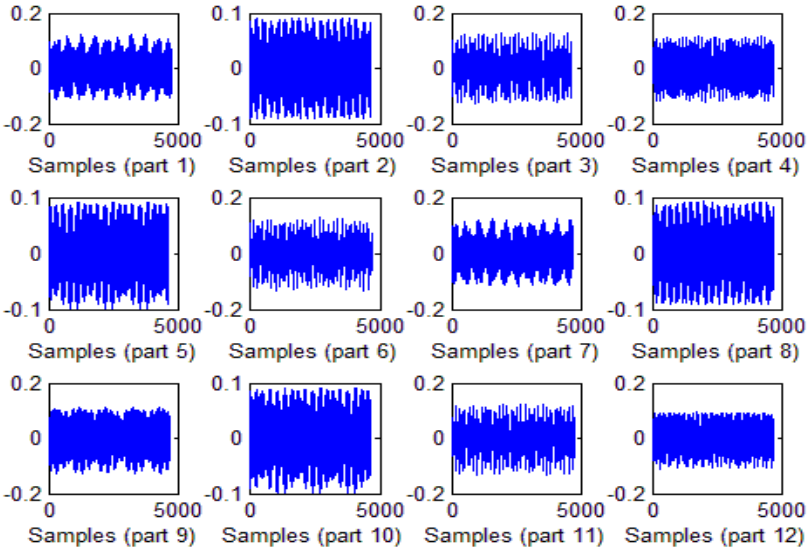

Fig.34 12 parts of presses based on filtered signal

Determine spectrum of 12 presses

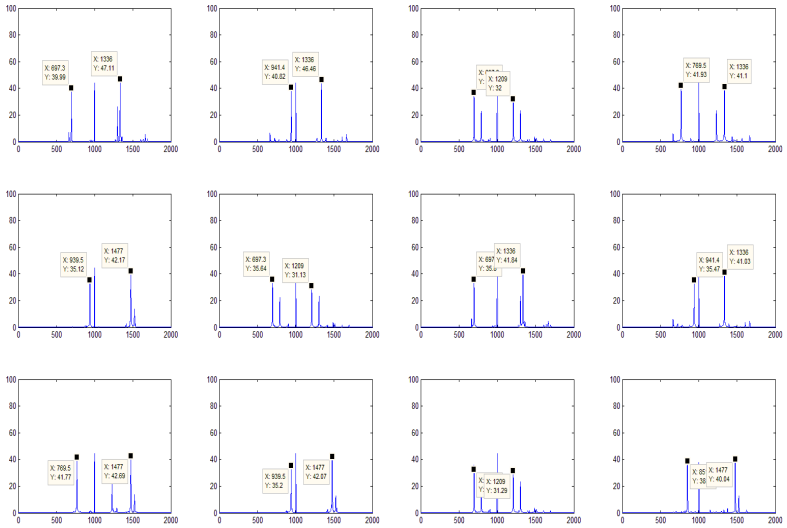

Fig.35 spectrum of 12 presses

The audio is 4 seconds long, thus, we can divide it into 12 intervals, and corresponding spectrums are as shown in Fig.35. From the spectrum, we can see that the inference at 1000Hz is not totally cleaned. This may be caused by the overlay of two bandpass filters whose Minimum stopband attenuation to be 60dB. However, we can still observe the two components of every key press.

	1	2	3	4	5	6	7	8	9	10	11	12
Comp 1	697	941	697	770	941	697	697	941	770	941	697	852
Comp 2	1336	1336	1209	1336	1477	1209	1336	1336	1477	1447	1209	1477
dial	2	0	1	5	#	1	2	0	6	#	1	9

Referring to the dial, we can tell that the 12 keys are: 2015#1206#19, which is the due date of this report.

Appendix A
Matlab code for Part I

```
% load the guitar1.wav
[fil_y,Fs]=wavread(( 'guitar1.wav'));
% sample
y_rec= fil_y.*rectwin(length(fil_y));
figure
subplot(3,1,1)
plot(0:length(fil_y)-1,fil_y);
% rectangular window
subplot(3,1,2)
plot(0:length(fil_y)-1,y_rec);
grid on
xlabel('length');
ylabel('amplitude');
title('Rectangular Window');
% hamming window
y_ham=fil_y.*hamming(length(fil_y));
subplot(3,1,3)
plot(0:length(fil_y)-1,y_ham);
grid on
xlabel('length');
ylabel('A mplitude');
title('Hamming Window');
%-------------------------------------choose rectangular window
 Y_REC=fft(y_rec,8192);
 figure
 subplot(2,1,1);
 plot(0:8191,20.*log10(abs(Y_REC)));
 grid on
 ylabel('dB');
 title('DFT after Rectangular Window Truncation');
 %_____________________________________DFT after rectangular window, in dB
 subplot(2,1,2);
 plot((0:8191)/8192*2,abs(Y_REC));
 grid on
 xlabel('w/pi');
 %----------------------------------significant portion
 sound(fil_y,Fs);
 %----------------------------------sound command
```

```
 %%
[fil_y,Fs]=wavread('guitar1.wav');
```

```
n=0:length(fil_y)-1;
figure
subplot(4,1,1)
plot(n,fil_y);
title('original signal');

 y_10=sin(2*pi*10000*n./Fs);% angular frequency = 2* pi * f
 subplot(4,1,2);
 plot(n,y_10);
%  set(gca,'xlim',[0 0.0001*length(y)],'ylimmode','auto');
 title ('noise of 10KHz');
 %--------------------10K Hz sin interference of same length & rate

 y_tot=y_10.'+fil_y;
 subplot(4,1,3);
 plot(n,y_tot);
 title('complex signal');
 %--------------------complex signal in time domain

 Y_tot=fft(y_tot,8192);
 subplot(4,1,4);
 plot((0:8191)/8192*2,20.*log10((abs(Y_tot))));
 title('DFT of sample add noise, in dB');
 grid on;
 ylabel('dB');
 %--------------------complex signal in freqency domain
 sound(y_tot,Fs);
 %% Butterworth filter
Fs=44100;
wp=7000/(Fs/2);% nyquist,1/2 of sampling frequency
ws=9000/(Fs/2);
rp=1;
rs=40;
format long
[n,Wn]=buttord(wp,ws,rp,rs)
[b,a]=butter(n,Wn)
fvtool(b,a);
%% Cheby1
Fs=44100;
wp=7000/(Fs/2);
ws=9000/(Fs/2);
rp=1;
rs=40;
format long
```

```
[n,Wn]=cheb1ord(wp,ws,rp,rs)
[b,a]=cheby1(n,1,Wn)
fvtool(b,a);
%%  Elliptic
Fs=44100;
wp=7000/(Fs/2);
ws=9000/(Fs/2);
rp=1;
rs=40;
format long
[n,Wn]=ellipord(wp,ws,rp,rs)
[b,a]=ellip(n,wp,ws,Wn,'low')
fvtool(b,a);
```

```
%% comparison before and after filtering
 % Butterworth filter
[fil_y,Fs]=wavread('guitar1.wav');
n=0:length(fil_y)-1;
y_10=sin(2*pi*10000*n./Fs);
y_tot=y_10.'+fil_y;
Fs=44100;
wp=7000/(Fs/2);
ws=9000/(Fs/2);
rp=1;
rs=40;
[n,Wn]=buttord(wp,ws,rp,rs)
[b,a]=butter(n,Wn)
[Hz,w]=freqz((10^(35.25/20))*b,a,1024);% DC gain of transfer function in
numerator
 Y_tot=fft(y_tot,8192);
 subplot(2,1,1);
 plot((0:8191)/8192*2,20.*log10((abs(Y_tot))));
 grid;
 ylabel('dB');
 title('complex signal');
 %-------------before filtered
 y_fil=filter((10^(35.25/20)).*b,a,y_tot);
 fil_Y=fft(y_fil,8192);
 subplot(2,1,2);
 plot((0:8191)/8192*2,20.*log10(abs(fil_Y)));
 grid;
 ylabel('dB');
 title('filtered signal');
 sound(y_fil,Fs)
```

```
%-------------------------------------------------------------
% Cheby1
[fil_y,Fs]=wavread('guitar1.wav');
n=0:length(fil_y)-1;
y_10=sin(2*pi*10000*n./Fs);
y_tot=y_10.'+fil_y;
Fs=44100;
wp=7000/(Fs/2);
ws=9000/(Fs/2);
rp=1;
rs=40;
[n,Wn]=cheb1ord(wp,ws,rp,rs)
[b,a]=cheby1(n,1,Wn)
[Hz,w]=freqz((10^(35.25/20))*b,a,1024);% DC gain of transfer function in
numerator
 Y_tot=fft(y_tot,8192);
 subplot(2,1,1);
 plot((0:8191)/8192*2,20.*log10((abs(Y_tot))));
 grid;
 ylabel('dB');
 title('complex signal');
 %-------------before filtered
 y_fil=filter((10^(35.25/20)).*b,a,y_tot);
 fil_Y=fft(y_fil,8192);
 subplot(2,1,2);
 plot((0:8191)/8192*2,20.*log10(abs(fil_Y)));
 ylabel('dB');
 title('filtered signal');
 sound(y_fil,Fs)
 %-------------------------------------------------------------------
-
```

```
%  Elliptic
[fil_y,Fs]=wavread('guitar1.wav');
n=0:length(fil_y)-1;
y_10=sin(2*pi*10000*n./Fs);
y_tot=y_10.'+fil_y;
Fs=44100;
wp=7000/(Fs/2);
ws=9000/(Fs/2);
rp=1;
rs=40;
[n,Wn]=ellipord(wp,ws,rp,rs)
[b,a]=ellip(n,wp,ws,Wn,'low')
```

```
[Hz,w]=freqz((10^(35.25/20))*b,a,1024); % DC gain of transfer function
in numerator
 Y_tot=fft(y_tot,8192);
 subplot(2,1,1);
 plot((0:8191)/8192*2,20.*log10((abs(Y_tot))));
 ylabel('dB');
 title('complex signal');
 %-------------before filtered
 y_fil=filter((10^(35.25/20)).*b,a,y_tot);
 fil_Y=fft(y_fil,8192);
 subplot(2,1,2);
 plot((0:8191)/8192*2,20.*log10(abs(fil_Y)));
 ylabel('dB');
 title('filtered signal');
 sound(y_fil,Fs)
```

```
 %% FIR filter
 %Hanning
[fil_y,Fs]=wavread('guitar1.wav');
 n=0:length(fil_y)-1;
 y_10=sin(2*pi*10000*n./Fs);% angular frequency = 2* pi * f
 y_tot=y_10.'+fil_y;
 Y_tot=fft(y_tot,8192);
 wp=7000/(Fs/2);
 ws=9000/(Fs/2);
 n=ceil(8/(ws-wp));
 b=fir1(n,(wp+ws)/2,hanning(n+1)); % call the funcation b =
fir1(n,Wn,ftype)
 [Hz,w]=freqz(b,1,1024,Fs);
 figure
 subplot(2,1,1),plot(w,20*log10(abs(Hz)));
 xlabel('frequency');ylabel('amplitude');grid
 subplot(2,1,2),plot(w,180/pi*unwrap(angle(Hz)));% use unwrap to
prevent vibration
 xlabel('frequency');ylabel('phase');grid
 %Hammming
 [fil_y,Fs]=wavread('guitar1.wav');
 n=0:length(fil_y)-1;
 y_10=sin(2*pi*10000*n./Fs);% angular frequency = 2* pi * f
 y_tot=y_10.'+fil_y;
 Y_tot=fft(y_tot,8192);
 wp=7000/(Fs/2);
 ws=9000/(Fs/2);
 n=floor(8/(ws-wp));
```

```
 b=fir1(n,(wp+ws)/2,hamming(n+1)); % call the funcation b =
fir1(n,Wn,ftype)
 [Hz,w]=freqz(b,1,1024,Fs);
 figure
 subplot(2,1,1),plot(w,20*log10(abs(Hz)));
 xlabel('frequency');ylabel('amplitude');grid
 subplot(2,1,2),plot(w,180/pi*unwrap(angle(Hz)));% use unwrap to
prevent vibration
 xlabel('frequency');ylabel('phase');grid
 %% designed filter with two windows
 [fil_y,Fs]=wavread('guitar1.wav');
 n=0:length(fil_y)-1;
 y_10=sin(2*pi*10000*n./Fs);% angular frequency = 2* pi * f
 y_tot=y_10.'+fil_y;
 Y_tot=fft(y_tot,8192);
 wp=7000/(Fs/2);
 ws=9000/(Fs/2);
 n=floor(8/(ws-wp));
 b=fir1(n,(wp+ws)/2,hanning(n+1)); % call the funcation b =
fir1(n,Wn,ftype)
 [Hz,w]=freqz(b,1,1024,Fs);
 fil_y=fftfilt(b,y_tot);
 subplot(2,1,1);
 Y_hanfil=fft(fil_y,8192);
 figure
 subplot(2,1,1)
 plot((0:8191),20*log10((abs(Y_hanfil))));
 ylabel('dB');
 subplot(2,1,2);
 plot((0:8191)/8192*2,(abs(Y_hanfil)));
 ylabel('amplitude');
 sound(fil_y,Fs)
```

```
 %-----------------hamming filtering
  [fil_y,Fs]=wavread('guitar1.wav');
 n=0:length(fil_y)-1;
 y_10=sin(2*pi*10000*n./Fs);% angular frequency = 2* pi * f
 y_tot=y_10.'+fil_y;
 Y_tot=fft(y_tot,8192);
 wp=7000/(Fs/2);
 ws=9000/(Fs/2);
 n=floor(8/(ws-wp));
 b=fir1(n,(wp+ws)/2,hamming(n+1)); % call the funcation b =
fir1(n,Wn,ftype)
 [Hz,w]=freqz(b,1,1024,Fs);
```

```
fil_y=fftfilt(b,y_tot);
subplot(2,1,1);
Y_hanfil=fft(fil_y,8192);
figure
subplot(2,1,1)
plot((0:8191),20*log10((abs(Y_hanfil))));
ylabel('dB');
subplot(2,1,2);
plot((0:8191)/8192*2,(abs(Y_hanfil)));
ylabel('amplitude');
sound(fil_y,Fs)
```

Appendix B
Matlab code for Part II

```
%% ------------------------------PART II   DECODE DTMF
% signal characteristics
[fil_y,Fs]=audioread('2015_final.wav');
% Plot original signal
figure('Name','Original Signal')
subplot(2,1,1)
plot((0:length(fil_y)-1)./Fs,fil_y);
grid on
%set(gca,'Fontsize',fontsize);
xlabel('Time (s)');
subplot(2,1,2)
plot((0:100-1)./Fs,fil_y(1:100));
grid on
xlim([0 100-1]./Fs)
ylim([min(fil_y(1:100)) max(fil_y)])
%set(gca,'Fontsize',fontsize);
xlabel('Time (s)');
% plot freqeuncy response of original signal
L=length(fil_y);
NFFT=2^nextpow2(L);
fft_result_temp=fft(fil_y,NFFT)/L;
freq=Fs/2*linspace(0,1,NFFT/2+1);
fft_result=fft_result_temp(1:NFFT/2+1);
figure('Name','Frequency Response of Original Signal')
subplot(2,1,1);
plot(freq,20*log10(abs(fft_result)));
```

```
%% Kaiser FIR filter 1
[fil_y,Fs]=audioread('2015_final.wav');
Fs
 f_min1=697;
 f_max1=941;

 A=60;
 Delta_w=300/Fs*2*pi;
 M=ceil((A-8)/(2.285*Delta_w)+1);
 if mod(M,2)==0
    M=M+1;
 end
 beta=0.1102*(A-8.7);
 wn=kaiser(M,beta);
```

```
 cut_off_w=([f_min1 f_max1]+[f_min1-300 f_max1+300])./2./Fs.*2.*pi;
 n=0:M-1;

hn=max(cut_off_w)./pi.*sinc(max(cut_off_w)./pi.*(n-(M-1)/2))-min(cut_
off_w)./pi.*sinc(min(cut_off_w)./pi.*(n-(M-1)/2));
 hn_Kai=hn.*wn';
 % Kaiser window in time domain
 figure
 subplot(2,1,1)
 stem(n,wn);
 grid on
 xlabel('Sample')
 ylabel('w[n]')
% impulse response of Kaiser
 subplot(2,1,2)
 stem(n,hn_Kai);
 grid on
 xlabel('Sample')
 ylabel('hn_Kai[n]')

 % Frequency response
 [H,f]=freqz(hn_Kai,1,length(fil_y),'half',Fs);
 figure('Name','Frequency response')
 subplot(2,1,1)
 plot(f,20*log10(abs(H)));
 grid on
 xlabel('Frequency (Hz)')
 ylabel('Amplitude (dB)')

 subplot(2,1,2)
 plot(f,abs(H));
 grid on
 xlabel('Frequency (Hz)')
 ylabel('Amplitude')

 figure('Name','group delay')
 subplot(2,1,1)
 plot(f,angle(H));
 grid on
 xlabel('Frequency (Hz)')
 ylabel('Phase (radians)')

 subplot(2,1,2)
 [gd,f]=grpdelay(hn_Kai,1,length(fil_y),'half',Fs);
```

```
plot(f,gd);
grid on
xlabel('Frequency (Hz)')
ylabel('Grop delay')

figure('Name','zero-pole plot')
[z,p,k]=zplane(hn_Kai,1);
grid on
xlabel('Real part')
ylabel('Imaginary part')
% filtered signal by using FIR bandpass filter
filtered_signal_FIR=conv(fil_y',hn_Kai);
filtered_signal_FIR=filtered_signal_FIR(1:length(fil_y))';
```

```
%% Kaiser FIR filter 2
[fil_y,Fs]=audioread('2015_final.wav');
Fs
f_min1=1209;
f_max1=1633;

A=60;
Delta_w=300/Fs*2*pi;
M=ceil((A-8)/(2.285*Delta_w)+1);
if mod(M,2)==0
   M=M+1;
end
beta=0.1102*(A-8.7);
wn=kaiser(M,beta);
cut_off_w=([f_min1 f_max1]+[f_min1-300 f_max1+300])./2./Fs.*2.*pi;
n=0:M-1;

hn=max(cut_off_w)./pi.*sinc(max(cut_off_w)./pi.*(n-(M-1)/2))-min(cut_
off_w)./pi.*sinc(min(cut_off_w)./pi.*(n-(M-1)/2));
hn_Kai=hn.*wn';
% Kaiser window in time domain
figure
subplot(2,1,1)
stem(n,wn);
grid on
xlabel('Sample')
ylabel('w[n]')
% impulse response of Kaiser
subplot(2,1,2)
stem(n,hn_Kai);
```

```
grid on
xlabel('Sample')
ylabel('hn_Kai[n]')

% Frequency response
[H,f]=freqz(hn_Kai,1,length(fil_y),'half',Fs);
figure('Name','Frequency response')
subplot(2,1,1)
plot(f,20*log10(abs(H)));
grid on
xlabel('Frequency (Hz)')
ylabel('Amplitude (dB)')

subplot(2,1,2)
plot(f,abs(H));
grid on
xlabel('Frequency (Hz)')
ylabel('Amplitude')

figure('Name','group delay')
subplot(2,1,1)
plot(f,angle(H));
grid on
xlabel('Frequency (Hz)')
ylabel('Phase (radians)')

subplot(2,1,2)
[gd,f]=grpdelay(hn_Kai,1,length(fil_y),'half',Fs);
plot(f,gd);
grid on
xlabel('Frequency (Hz)')
ylabel('Grop delay')

figure('Name','zero-pole plot')
[z,p,k]=zplane(hn_Kai,1);
grid on
xlabel('Real part')
ylabel('Imaginary part')
% filtered signal by using FIR bandpass filter
filtered_signal_FIR=conv(fil_y',hn_Kai);
filtered_signal_FIR=filtered_signal_FIR(1:length(fil_y))';
```

```
%% IIR filter
% butterworth bandpass filter 1
```

```
[fil_y,Fs]=audioread('2015_final.wav');
f_min1=697;
f_max1=941;
wp=[f_min1 f_max1]./(Fs/2);
ws=[f_min1-300 f_max1+300]./(Fs/2);
rp=3;
rs=60;

[n,wn]=buttord(wp,ws,rp,rs)
[b,a]=butter(n,wn)



[H,f]=freqz(b,a,length(fil_y),'half',Fs);

figure('Name','Frequency response of Butterworth bandpass filter')
subplot(3,1,1)
plot(f,20*log10(abs(H)));
grid on
xlabel('Frequency/Hz')
ylabel('Amplitude/dB')

subplot(3,1,2)
plot(f,abs(H));
grid on
xlabel('Frequency/Hz')
ylabel('Amplitude')

subplot(3,1,3)
plot(f,angle(H));
grid on
xlabel('Frequency/Hz')
ylabel('Phase/radians')

[gd,f]=grpdelay(b,a,length(fil_y),'half',Fs);
figure
plot(f,gd);
grid on
xlabel('Frequency/Hz')
ylabel('Group delay')

figure('Name','zero-pole plot of Butterworth bandpass filter')
[z,p,k]=zplane(b,a);
grid on
xlabel('Real part')
```

```
ylabel('Imaginary part')
```

```
%%
% butterworth bandpass filter 2
[fil_y,Fs]=audioread('2015_final.wav');
f_min1=1209;
f_max1=1633;
wp=[f_min1 f_max1]./(Fs/2);
ws=[f_min1-300 f_max1+300]./(Fs/2);
rp=3;
rs=60;

[n,wn]=buttord(wp,ws,rp,rs)
[b,a]=butter(n,wn)
%filtered Y=filter(b,a,Y);

[H,f]=freqz(b,a,length(fil_y),'half',Fs);

figure('Name','Frequency response of Butterworth bandpass filter')
subplot(3,1,1)
plot(f,20*log10(abs(H)));
grid on
xlabel('Frequency/Hz')
ylabel('Amplitude/dB')

subplot(3,1,2)
plot(f,abs(H));
grid on
xlabel('Frequency/Hz')
ylabel('Amplitude')

subplot(3,1,3)
plot(f,angle(H));
grid on
xlabel('Frequency/Hz')
ylabel('Phase/radians')

[gd,f]=grpdelay(b,a,length(fil_y),'half',Fs);
figure
plot(f,gd);
grid on
xlabel('Frequency/Hz')
ylabel('Group delay')
```

```
figure('Name','zero-pole plot of Butterworth bandpass filter')
[z,p,k]=zplane(b,a);
grid on
xlabel('Real part')
ylabel('Imaginary part')
```

```
%% Butterworth bandpass filter 1 detailed scale
[fil_y,Fs]=audioread('2015_final.wav');

f_min1=697;
f_max1=941;
%  f_min=1209;
%  f_max=1633;
wp=[f_min1 f_max1]./(Fs/2);
ws=[f_min1-300 f_max1+300]./(Fs/2);
rp=3;
rs=60;

[n,wn]=buttord(wp,ws,rp,rs);
[b,a]=butter(n,wn);
%filtered Y=filter(b,a,Y);

[H,f]=freqz(b,a,length(fil_y),'half',Fs);
figure
plot(f,abs(H));
grid on
xlabel('Frequency/Hz')
ylabel('Amplitude')

%% filtering the original sample using two butherworth bandpass filter
[fil_y,Fs]=audioread('2015_final.wav');
Fs
f_min1=697;
f_max1=941;
wp1=[f_min1 f_max1]./(Fs/2);
ws1=[f_min1-300 f_max1+300]./(Fs/2);
rp=3;
rs=60;
[n1,wn1]=buttord(wp1,ws1,rp,rs);
[b1,a1]=butter(n1,wn1);
filtered_y1=filter(b1,a1,fil_y);

f_min2=1209;
f_max2=1633;
```

```
wp2=[f_min2 f_max2]./(Fs/2);
ws2=[f_min2-300 f_max2+300]./(Fs/2);
[n2,wn2]=buttord(wp2,ws2,rp,rs)
[b2,a2]=butter(n2,wn2)
filtered_y2=filter(b2,a2,fil_y);

fil_y= filtered_y1+ filtered_y2;
sound(fil_y,Fs)
```

```
%% decoding key presses
[y,Fs]=wavread('2015_final.wav');
length(fil_y);
f = Fs*(0:8191)/8192;

y1=fft((fil_y(1:5600)).*hanning(5600),8192);
subplot(3,4,1);
plot(f,abs(y1));
axis([0 2000 0 100]);
y1=fft((fil_y(5600:11199)).*hanning(5600),8192);
subplot(3,4,2);
plot(f,abs(y1));
axis([0 2000 0 100]);
y1=fft((fil_y(12000:17599)).*hanning(5600),8192);
subplot(3,4,3);
plot(f,abs(y1));
axis([0 2000 0 100]);
y1=fft((fil_y(17600:23199)).*hanning(5600),8192);
subplot(3,4,4);
plot(f,abs(y1));
axis([0 2000 0 100]);
y1=fft((fil_y(23200:28799)).*hanning(5600),8192);
subplot(3,4,5);
plot(f,abs(y1));
axis([0 2000 0 100]);
y1=fft((fil_y(28800:34399)).*hanning(5600),8192);
subplot(3,4,6);
plot(f,abs(y1));
axis([0 2000 0 100]);
y1=fft((fil_y(34400:39999)).*hanning(5600),8192);
subplot(3,4,7);
plot(f,abs(y1));
axis([0 2000 0 100]);
y1=fft((fil_y(40000:45599)).*hanning(5600),8192);
subplot(3,4,8);
```

```
plot(f,abs(y1));
axis([0 2000 0 100]);
y1=fft((fil_y(45600:51199)).*hanning(5600),8192);
subplot(3,4,9);
plot(f,abs(y1));
axis([0 2000 0 100]);
y1=fft((fil_y(51200:56799)).*hanning(5600),8192);
subplot(3,4,10);
plot(f,abs(y1));
axis([0 2000 0 100]);
y1=fft((fil_y(56800:62399)).*hanning(5600),8192);
subplot(3,4,11);
plot(f,abs(y1));
axis([0 2000 0 100]);
y1=fft((fil_y(62400:67199)).*hanning(4800),8192);
subplot(3,4,12);
plot(f,abs(y1));
axis([0 2000 0 100]);
```