UNIVERSITY
OF GÄVLE

FACULTY OF ENGINEERING AND SUSTAINABLE DEVELOPMENT

Process and Analysis of Voice Signal by MATLAB

Nan Wu

Bofei Wang

June 2014

Bachelor thesis in Electronics

Program: Electronics
Examiner: Jose Chilo
Supervisor: Zain Ahmed Kahn

Abstract

Deliver message by voice is the most important, effective and common method of exchange information for mankind. Language is human specific features and human voice is commonly used tool which is also the important way to pass information to each other. The voice has large information capacity. So we can use modern method to study voice processing technology, so that people can easily transmit, store, access and apply the voice. In this thesis, we designed a collection system that can collect voice and use different filters to filter the noise. After filtering the noise, the voice will be more quality in mobile communication, radio, TV and so on.

In this thesis we use Microsoft recorder to collect a voice, and then analyze its time-domain, the frequency spectrum and the characteristics of the voice signal. We use MATLAB's function to remove the noise which has been added to the voice, further use bilinear transformation method to design a filter which is based on Butterworth simulation and window function and then filter the voice signal which has been added noise. After that we compare the time-domain and frequency-domain of the original voice and noised voice, then playback the noised voice and de-noising voice and then compare the application of signal processing in FIR filter and IIR filter, especially in the perspectives of the signal filtering de-noising characteristics and applications. According to the comparison, we can determine which filter is the best.

Key words: voice signal, MATLAB, filter.

Content

1. Introduction

This chapter gives the background of the voice process; this thesis is based on MATLAB so we give the background of MATLAB and the reason why we choose MATLAB to finish this thesis and the purpose of this study.

1.1 Background

Human voice is a commonly useful tool and it is the most important means to pass information to each other. Passing message by voice is the most important and effective way for mankind. Now with the development of the times, mankind has entered the information age, with the modern means of speech signal study, people can generate, transmit, store, access, and apply voice messaging more effectively, which has a very important significance for the promotion of social development. So in this thesis, we built a voice collection system, which can collect voice signal then analyze the signal, then filter the noise by using various type of filters.

MATLAB is a combination of two words matrix & laboratory, released by the U.S.Mathworks main face of scientific computing, visualization and interactive program designed in high-tech computing environment. Its numerical analysis, matrix computation, scientific data visualization and modeling as well as simulation of nonlinear dynamic systems, and many other powerful features are integrated in an easy way to use Windows environment, for scientific research, engineering design, and the need for effective numerical numerous scientific field provides a comprehensive solution, and largely combines with the traditional non-interactive programming languages (such as C, Fortran) in edit mode, on behalf of the current international advanced level of scientific computing software.[1]

The collection, analysis and processing of speech signals, from theoretical research to product development have gone through a dozen years and have made a long-term progress. It has been applied into industrial production sector voice control, telephone, automatic dialing in telecommunications systems, auxiliary function control to the query and the public health and welfare of the life support systems and other practical applications, and is expected to become the next generation operating system and the user interface of the application,[2]that is because we will do this to help people record, analyze the sound and filter it into a high quality sound.

1.2 Goal

The goal of our thesis is to build a voice collection system to record voice and then use it to filter the noise. In daily life there are some situations that need to filter the noise when we received voice, so filter the noise means a lot. Also, build this system can be a nice example to show others the filter’s performance and application, which can be surely used in university class. In this thesis, we decided to use data collection system by using voice card. After filtering the voice we can get higher quality sound which is helpful in electric communication like radio, mobile or TV and so on.

1.3Outline

In Chapter 1 is introduction of our work and tells people the background of voice collection and a few history of the voice signal process. Also the MATLAB is introduced in this chapter and we set a goal in chapter 1. Chapter 2 is the theory part. This part shows different ways to build a collection system, and the basic knowledge of FIR and IIR filters. Next is process part that we presented in chapter 3. This part recommended the whole flow chart of our structure, and how to collect voice, and how to design the filters. Then the results show in chapter 4. We get the results of the 6 different type filters so that we can find the best filters to filter our sound record. Finally, the conclusion and discussion is written in chapter 5. We found some shortage of the system and try to make it better in future.

2. Theory

This chapter introduces all structures of the voice collection system, and the theory of speech signal collection, different types of IIR and FIR filters. This offers the basic knowledge of this thesis.

2.1 Overall system

2.1.1 Four schemes of data collection system

Some common design plans for data collection system are as below:

(1) Use data collected by collection modules of RS322 serial communication and then transform them to PC.

When get the data then with the help of serial port object of instrument control toolbox in MATLAB we can control the serial port to communicate with peripherals like handle documents. First apply the serial function to create the serial port objects, then set up Baud rate, data position, stop position and other properties, which makes PC and data collection modules have same communication type [3]. When transform data, use JUfopen function to open the serial port, and use fwrite function and fread function by binary or ASCII to read and operate. Use fdose function to close the serial port.

(2) Use AT89C51 microcontroller (MCS51 series) and Texas Instruments Company's TLC2543make data collection card.

This collection system has good mobility and can reduce cost. Microcontroller will complete data collection task, then the data will be transmitted to PC fast then start processing and analyzing. Design this data collection card in external structure, the analog signal through A/D converter into digital signal then pass the serial port and get into PC.

(3) Apply USB- bus's data acquisition unit

The USB-bus has lots of advantages like fast speed, easy extension, bus power supply, flexibility for using. In MATLAB, we can control the bus directly to achieve real time data collection. But MATLAB can't control the USB by itself. In C language environment, USB will be easily controlled. So if C language is used to drive and control the data collection part, then it should be compiled into MEX file.

(4) Data collection system by using voice card.

Laptops have an integration voice card, so we can use microphone and voice card to record the voice, then read and analyze by MATLAB, meanwhile we can use the GUI in MATLAB to design a nice guest interface, so this can be a cheap data collection system for some uses that has few requirements. The application of this system can reduce cost and it is easy to be used.

2.1.2 Characteristics of system

Data collection system normally has 6 characteristics as below:

1) Generally modern collection system is controlled by computers that improve the quality and efficiency of data collection, and save hardware investment as well.
2) Software plays more and more important role in data collection system and increases the flexibility of system design.
3) Data collection combines with data process closer and closer so a data collection and process system should be produced. This system will do all jobs from data collection, processing to control.
4) The process of data collection has real time property. For normal collection system higher speed is much better to adapt more application environment.
5) Bustechnique has widespread use in data collection system and plays an important role.

2.2 The theory of speech signal collection

2.2.1 Sampling frequency

Sampling frequency means that pc collects how many speech samples per second, which not only is describing voice files' pitch, timbre but also can measure the sound card, sound file's quality. The frequency is higher while the interval time of sampling is shorter, which means computer gets more sound sample data in a unit time, so the shape of sound wave forms will be more precise. According to the Nyquist theory, only when the frequency of samples is twice higher than the speech signal’s peak frequency then we can convert digital signal to the origin sound. Thismeans the sampling frequency is key to sound card collection, record and sound filesrestoration.

2.2.2 Sampling resolution

Sampling resolution is the value of sampling which is a parameter use to measure sound wave change. It is a binary bit when the sound card collecting, playing sounds file.

Sampling resolution and sampling frequency are two important indexes to the audio interface which is also a standard to choose an audio interface. No matter how sampling frequency is, theoretically the sampling resolution decides the range of sound intensity. Increase 1 sampling resolution amounts to range increase 6dB. The more resolution has the more accurate the signal collect is.

2.2.3 Sampling theory

In the progress of analog to digital signal convert, when sampling frequency fs.max is twice higher than the highest frequency of the signal f max, fs.max ≥ 2 f.max, so the digital signal after sampling will keep the information of the origin signal. In normally application to guarantee the sampling frequency is 5 – 10 times to the signals highest frequency. This is called Nyquist theorem. [8]

2.3 Theory design of digital filter

2.3.1 The basic idea of digital filter

There are two important steps to achieve digital filter, one is to convert the digital domain to analog domain. This step makes the digital filter technology index convert to analog filter technology index come true, which also makes digital filter system functions convert to analog filter system functions come true. The other one is designing a digital filter technology index that satisfies the analog filter.

2.3.2The outline of analog filter

In order to use analog to digital converter to design IIR digital filter, first an analog filter is needed to be designed. There are so many different types and two of them are main ones:

(1) Butterworth filter. BW filter is a signal processing filter that is designed to have as flat a frequency response as possible in the pass-band.[10] For an N-order low pass filter, the flat characteristic is the derivative of (2N-1) order of analog function in $\omega=0$. The other feature of BW filter is its amplitude frequency characteristic which is always monotonically decreasing function of frequency, and its analog function is closer to the ideal low pass filter with N increase.

(2) Chebyshev filter. The analog functions of CB low pass filter is defined by Chebyshev polynomial, and the frequency response is fluctuate in pass-band while monotonic change in stop-band. [10]

2.3.3 The outline of IIR digital filter

IIR filter is Infinite Impulse Response and it has many features below:

(1) Closed function: the system function of IIR filter can be written as the closed function.

(2) IIR digital filter has structure with recursion, which means it has the structure with a feedback loop. The operation of IIR filter is usually composed by delay, multiply index and adding which can compose four types directly. All of these types have feedback loops. But there are always some errors in the operation; it will produce a weak parasitic oscillation sometimes.

(3) Need to add a phase to correction network. The phase of IIR digital filter is hard to control, which need to add a phase to correction network when the demand of phase is higher.

2.3.4 The outline of FIR digital filter

FIR digital filter is finite impulse response filter. The final response to the pulse input signal of this kind of filter tends to 0. There are several advantages of FIR filter.

(1) The impulse respond is finite. It makes the digital signal in output finite when the signal in input is finite.

(2) Compared with IIR filter, it is easier to maximize.

(3) Linear-phase. It results the transfer function to even symmetry or odd symmetry and it is finite.

(4) It must be stable because all of poles are in the unit circle after z transform.[11]

2.3.5 Compared FIR filter with IIR filter

The design of both IIR filter and FIR filter has three steps:

(1) Determine the performance index of the filter by the requirement of actual task.

(2) Use a causal, stable function to approach this performance index. Use IIR system function or FIR system function to approach the performance index according to different requirement.

(3) Use the finite precision algorithm to achieve system function, including the structure selection, byte selection etc.

The design of IIR filter is different from the design of FIR filter. The IIR filter design method has indirect method and direct method. The indirect method gets help from the design of analog filter.[12] The first step is to design a transition analog filter to get the system function, and then transform the system function of analog filter to the system function of digital filter by some method. FIR filter uses the indirect method and the most common method used is window function method, frequency sampling method and Chebyshev ripple approximation method. For linear phase filter usually uses FIR filter. [13]

2.3.6 Low-pass, high-pass and band-pass filter

Low pass filter: For different low-pass filter, there is different weakening of frequency in each signal. When using it in audio applications, sometimes it is called high-cut filter or treble cut filter [12]. The application of low-pass filter in signal processing is equal to the effect of other areas. There are many types of low-pass filter; the most common one is Butterworth filter and Chebyshev filter.

High-pass filter: High-pass filter is a filter that removes useless low frequency interference. Its function is to filter low-frequency component and increase the high-frequency component. Sometimes it is called low-cut filter or bass-out filter. In addition, high-pass filter always appears with low-pass filter. No matter high-pass filter or low-pass filter, their function is sending the sound frequencies to the appropriate unit.

Band-pass filter: A band-pass filter is a device that passes frequencies within a certain range and rejects frequencies outside that range. An example of band-pass filter is resistance-inductance-capacitance circuit. [14]

2.4 Design of GUI interface

GUI means that graphical user interfaces contains windows, cursors, buttons, menu, text and other objects. Using GUI can optimize performance, make the operation more humane, reduce the users' cognitive burden and enhance the competitiveness of products [15]. There is a tool FUIDE in MATLAB which can offer the GUI a more effective integrated development environment. GUIDE is interface design too l kit, GUI can be stored in FIG file and also create an M file.

M-file contains GUI design, control widget function and user control callback function defined as sub function. GUI creation has interface design parts and widget program parts. The main steps are as follow: first, set up GUIDE options; second, use interface design modifier to draw the figure, third write the widget callback control code.

3. Process

This chapter picks up the final collection system structure and some main code of the MATLAB process. Also GUI interface is introduced in this chapter. So this chapter recommends how to accomplish this thesis.

3.1 System Structure

We decide to use the last design plan to accomplish our design. The system structure will be shown in details below.

Data collection system mainly contains two parts: collection subsystem and computer subsystem, which means lower computer collection system and upper computer IIMI system. [5] Collection subsystem will collect data and convert it into digital signal which can be dealt by PC. The data can be controlled, stored and handled by computer subsystem.

Normal external data collection system structure is shown in figure 2-1. Analog signal collected by sensor then through signal conditioning module, after that, signal will be sent into data collection hardware equipment. The data will accomplish A/D conversion, including sample, quantization, encode in the data collection tool. Finally the digital signal will be transmitted into the computer. In this thesis, the leading end sensor and other hardware equipment are made up by integration voice card and microphone. According to difference requirement, the data will be undertaken real time analysis and process by using MATLAB and further programming. In the working condition users can control the data collection hardware equipment through man-machine interaction interface to modified, set various parameters, and get the analysis result, which is called data collection and analysis automation.[4]

The whole system can be divided into two parts of data collection and data analysis.

All data’s time domain and frequency domain will be shown in figure to users. Besides it can offer data saving and data playback functions.

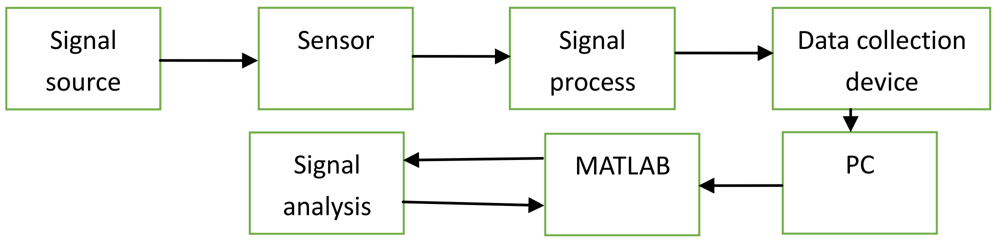


Figure 3-1 Normal external structure of data collection system

The data collection system by MATLAB's diagram is shown in figure 2-2, the main component data collection tool kit provides hardware drive program adapter, and data collection engine and M-file function between hardware drive program and MATLAB environment.

The hardware drive program adapter exchanges attribute value, data and events between hardware drive program and data collection engine. The data collection engine will store every device objects and their attribute value and store the collected data then makes out-sync events to be synchronized. M-file is used to create device objects, collect or output data, collect attribute value, test the status of data collection and data collection device. [6]

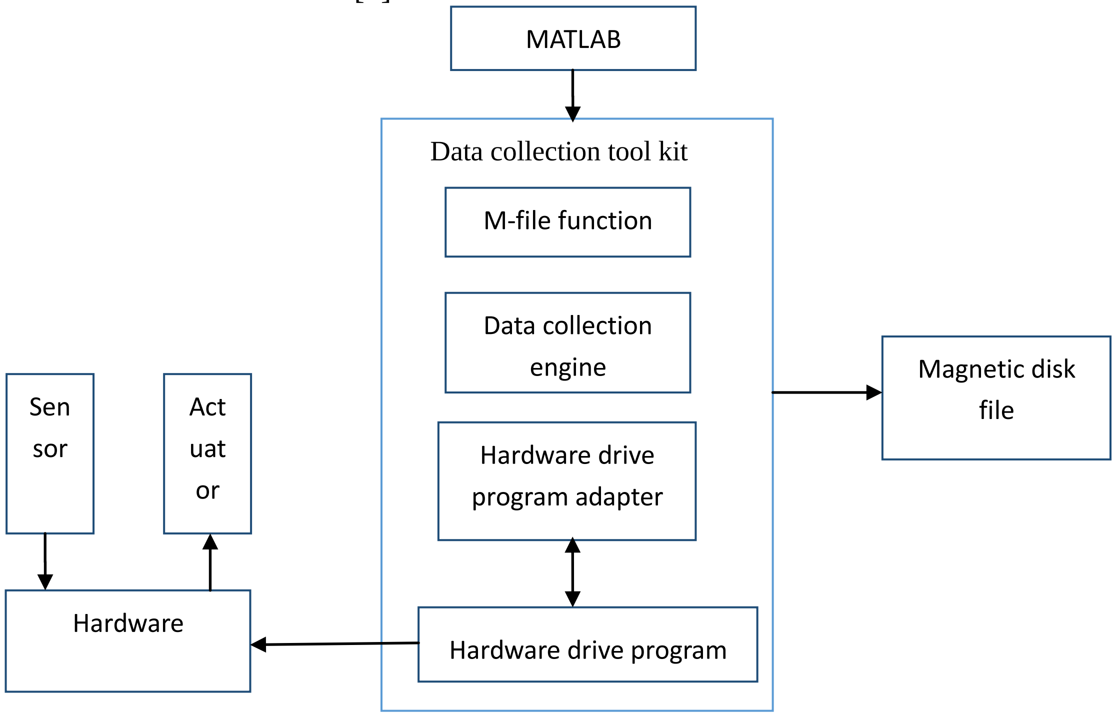


Figure 2-2 Data collection system by MATLAB

In figure 2-2 is our design's hardware and sensor, actuators are all accomplished by laptop's integration voice card and microphone.

For this system, we can have De-noising function as well. The basic flow chart is shown in figure 2-3.

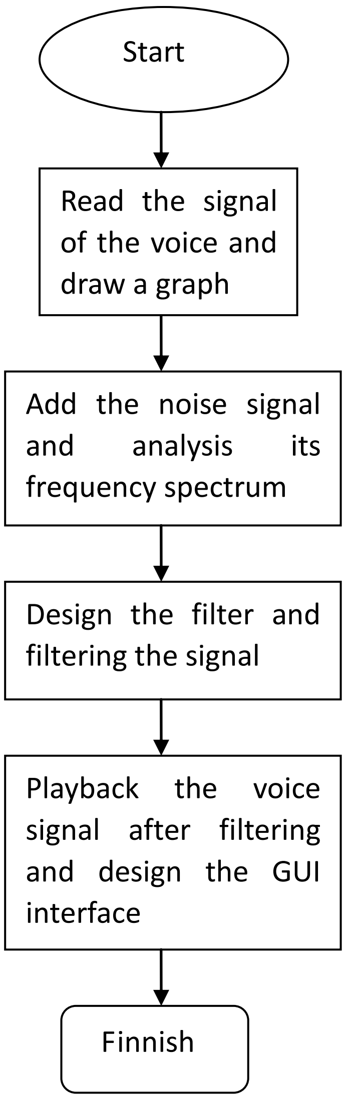

Start
Read the signal of the voice and draw a graph
Add the noise signal and analysis its frequency spectrum
Design the filter and filtering the signal
Playback the voice signal after filtering and design the GUI interface
Finnish

Figure 3.2 Flow chart of the program

3.2 Function of system design

This system is made up by two parts, one is data collection system and the other is data analysis system. The function of data collection is to obtain the samples from voice card. Data analyses parts mainly have functions below:

1) Get data from signal collection parts or read data from system.
2) Spectral analysis of the signal and getting the spectrogram, which will show characteristics of the signal in an intuitive way
3) Display voice signal’s frequency domain figure and time domain figure.

4) Put a random noise signal by using MATLAB into the original signal, and through the filters we design to filter the noise and replay the voice.
5) Use GUI graph tool in MATLAB, and create a user interface.

3.3 Achieve this system design

Voice signal collected is using MATLAB to control PC's voice card and convert the analog signal into digital signal and store it in PC. And the signal analysis is to analyze the time and frequency domain.

3.3.1 Method of the collection of voice signal

Collect the digital signals by using the voice card on pc and WINDOWS operating system. Plug the microphone to the voice input on pc, and start the recorder. Push the record button and say 'I love electronics' and then stop recording, the left of the screen will show the length of the voice. Push the play button can hear the speech we have made. Name the file *Orivoice* and save it to D:/. We could see that this filename extension is .wav which is the standard of WINDOWS operation system sound files. [7]

Certainly we can use MATLAB to achieve record voice function. In this thesis we have two methods as below:

(1) Operating the analog input items produced by sound card.

Sound card is a hardware that is supported by MATLAB data collecting tool kit. When we use sound card to complete the data collection, the microphone will be the sensor of this system. Here are the basic steps:

a) Create the target equipment. Here is the voice card AI.
b) Add channel to the target equipment. Here we add 1 channel.
c) Set the property value of the equipment, and control the data collection.
d) Data collection and result management.

First use Fourier transform to the collected data. Then turn the result into dB, and show the real number of the result.

e) Clear the target equipment in RAM

```
delete(AI);
```

clear AI;

This entire program is shown below:

```
% voice from microphone to the signal of MATLAB
AI=analoginput('Orivoice'); % sets up target of record
Chan=addchannel(AI,1); % adds record channel
Duration=5; % records 5 seconds voice
Set(AI,'SampleRate',8000); % sets the sanple frequency
ActualRate=get(AI,'SampleRate');
set(AI,' SamplesPerTrigger',duration*ActualRate)
set(AI,'TriggerType','Manual')
blocksize = get(AI,'SamplesPerTrigger');
Fs = AcutalRate;
start(AI); % starts recording, input voice signal by microphone.
trigger(AI); % stops recording
sn = getdata(AI); % gets voice signal data.
figure;
t = 1 : 40000 ;
plot ( t,sn); % shows the wave form of the voice signal.
xlabel('time');
ylabel('amplitudes');
```

(2) Use wave function in MATLAB to achieve record

```
The wave function
  % voice from microphone to the signal of MATLAB
  n=10;
  fs=16000; % sets n to be record time, sample rate is fs
  channel = 1;
  y = waverecord(n *fs,fs,channel,'unit8'); % Record
  pause % Pause
  wavplay(y,fs); % plays the voice of the record.
  figure;
  plot(y);
  wavwrite(y,fs,'Orivoice.wav);   %   Saves   the   file   with   name
Orivoice.wav
  pause;
  [y,fs,bits]= wavread('Orivoice.wav');
```

In this thesis, we use the second method to record and save the voice file.

3.4 Design filter

3.4.1 Design of FIR filter

There are big differences in designing FIR filter and IIR filter. Here is the design step by using window function.

(1) According to the index of stop-band attenuation and transition-band, choose the type of window functions and estimate the length of the window.

(2) Construct a function that approaches the frequency response.

(3) Calculate the transfer function.

(4) Add the window function and get the result.

3.4.2 Design of IIR filter

The steps of design IIR filter are:

(1) Transfer the technical index of the digital filter to the technical index of simulation low-pass filter.

(2) According to the technical index of simulation low-pass filter to design the filter.

(3) Transfer the function of simulation filter to the function of digital filter.

If the designed filter is low-pass filter, the design is finished. If the designed filter is high-pass filter or band-pass filter, there is also step.

(4) Transfer the technical index of high-pass or band-pass filter to the technical index of simulation low-pass filter. And based on steps design the transfer function of the filter, and then transfer this function to digital filter.[9]

3.4.3 Performance index of different filters

(2) Performance index of low-pass filter

fp=1000Hz, fc=1200HZ, As=100dB, Ap=1dB

(2) Performance index of high-pass filter

fp=3500Hz, fc=4000Hz, As=100dB, Ap=1dB

(3) Performance index of band-pass filter

fp1=1200HZ,fp2=3000Hz,fc1=1000Hz,fc2=3200Hz,As=100dB,Ap=1dB

3.5 Design of FIR and IIR filter in MATLAB

3.5.1 FIR filter

(1) FIR low pass filter

The boundary frequency in band pass of this filter is 1000Hz, the cutoff frequency in stop band is 1200Hz, sampling frequency is 8000Hz, the maximum attenuation in band pass is 1dB, and the minimum attenuation in stop band is 50dB. Here is the procedure:

(2) FIR high pass filter

The boundary frequency in band pass of this filter is 4000Hz, the cutoff frequency in stop band is 3500Hz, sampling frequency is 8001Hz, the maximum attenuation in band pass is 1dB, and the minimum attenuation in stop band is 50dB. Here is the procedure:

(3) FIR band pass filter

The boundary frequency in band pass of this filter is 1200Hz and 3000Hz, the cutoff frequency in stop band is 3500Hz and 3200Hz, sampling frequency is 7000Hz, the maximum attenuation in band pass is 1dB, the minimum attenuation in stop band is 50dB. Here is the procedure:

3.5.2 IIR filter

(1) IIR low pass filter

The boundary frequency in band pass of this filter is 1000Hz, the cutoff frequency in stop band is 1200Hz, sampling frequency is 8000Hz, the maximum attenuation in band pass is 1dB, the minimum attenuation in stop band is 50dB. Here is the procedure:

(2) IIR high pass filter

The boundary frequency in band pass of this filter is 4000Hz, the cutoff frequency in stop band is 3500Hz, sampling frequency is 8000Hz, the maximum attenuation in band pass is 1dB, and the minimum attenuation in stop band is 50dB. Here is the procedure:

(3) IIR band pass filter

The boundary frequency in band pass of this filter is 1200Hz and 3000Hz, the cutoff frequency in stop band is 1000Hz and 3200Hz, sampling frequency is 7800Hz, the maximum attenuation in band pass is 1dB, the minimum attenuation in stop band is 50dB. Here is the procedure:

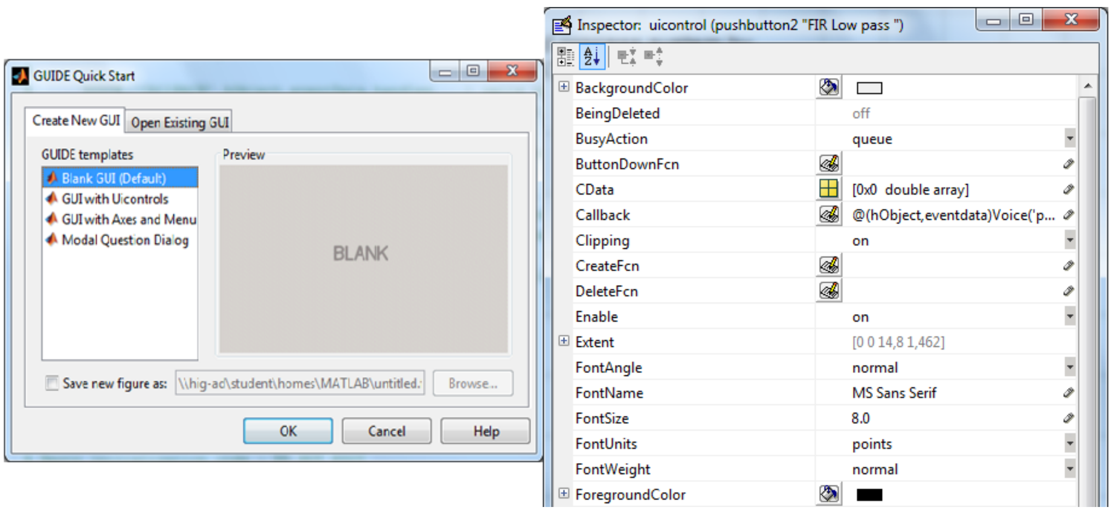
We use sin function in order to verify the usability of the designed filter. The results will be shown in next chapter.

3.6 Filtering

Use the designed filter to filter the voice signal with noise. FIR filter uses function fftfilt to filter the signal while IIR filter uses function filter to filter the signal. The function fftfilt uses overlap and adds operation to achieve the calculation of linear convolution.

3.7 Design GUI

First we create a new GUI file: Choose New/GUI, and then choose Blank GUI (Default).We can add buttons in the GUI, we can use the buttons indicate the function. When some buttons were set up, double clicked the button, so buttons properties were show in the figure below, then the performance and the value of the buttons will be adjusted easily. The button properties shows blow:



Inspector: uicontrol (pushbutton2 "FIR Low pass ")

BackgroundColor	▭
BeingDeleted	off
BusyAction	queue
ButtonDownFcn	
CData	[0x0 double array]
Callback	@(hObject,eventdata)Voice('p...
Clipping	on
CreateFcn	
DeleteFcn	
Enable	on
Extent	[0 0 14,8 1,462]
FontAngle	normal
FontName	MS Sans Serif
FontSize	8.0
FontUnits	points
FontWeight	normal
ForegroundColor	■

Figure 3-3 GUI interface

Figure 3-1 shows the quick way to build a GUI interface in MATLAB, the left figure shows how to create GUI interface and the right one is button properties.

Push save button to save the GUI into a FIG file, and an M-file will be created. Use the code that we wrote; just plug the code into the corresponding push buttons.

4. Result

Since we made 6 types of filters, this chapter will show the results in figures of these filters. The good results and bad results all will be shown in this chapter. Compared with these results we can get which one is the best.

4.1 Spectral of the voice

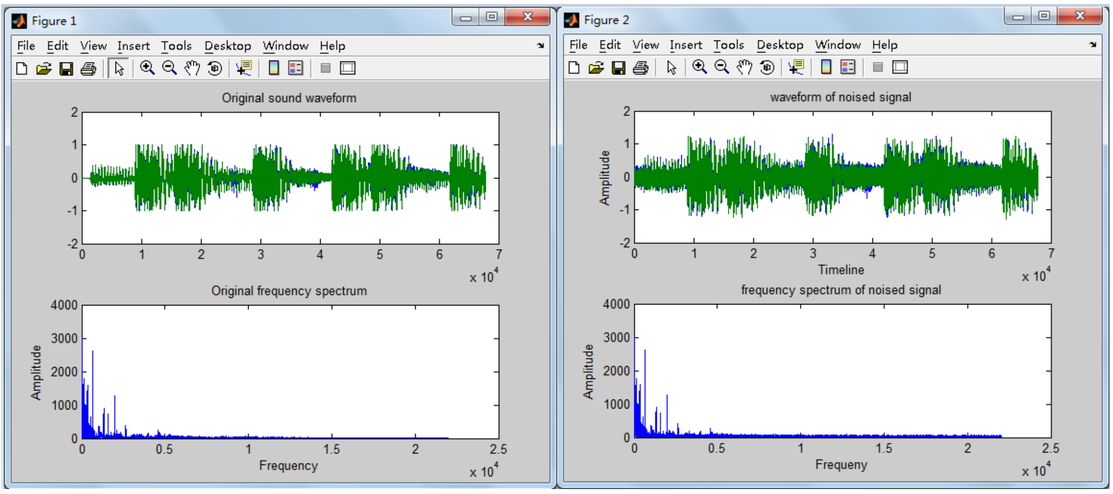


Figure 4-1 spectral analyze of the voice

From figure 4-1, the left figure is original waveform and frequency spectrum of the sound before adding noise, the results of waveform and frequency spectrum after adding noise shows in the right figure. In waveform figure, X-label is timeline and Y-label is amplitude of frequency. In spectrum figure, X-label is frequency and Y-label is amplitude of frequency also. Compared with the original waveform, it is obviously that the noised waveform has more range of amplitude. That shows the noise has been added successfully. Then we playback the sound, we cannot hear the sound clearly which means there is much noise in it.

4.2 Verify the filter

In order to verify the filter we use the sin wave to verify it and the result is shown in figure 4-2. From this figure, the X-label in waveform figure is timeline and in spectrum figure X-label is frequency. The Y-label is amplitude. It is obviously that the signal spectrum before filtering is not a smooth line, which has some small spectral, but after filtering the signal spectrum shows nice performance. Same reason, the waveform before filtering has lots noise in it but after filtering we can see a smooth sin wave.

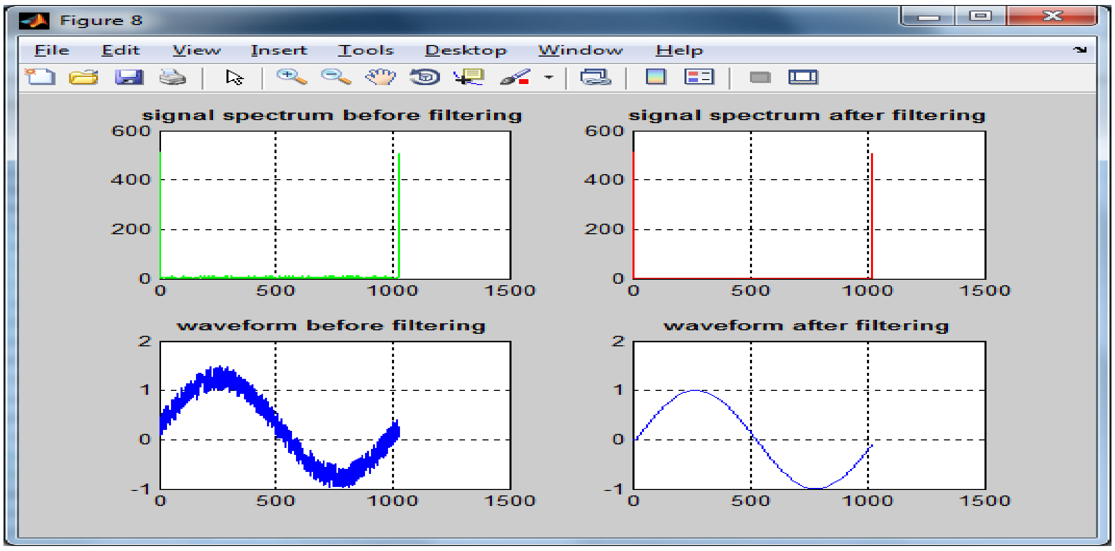


Figure 4-2 verified the filter

4.3 Filter the voice

4.3.1 FIR low pass filter

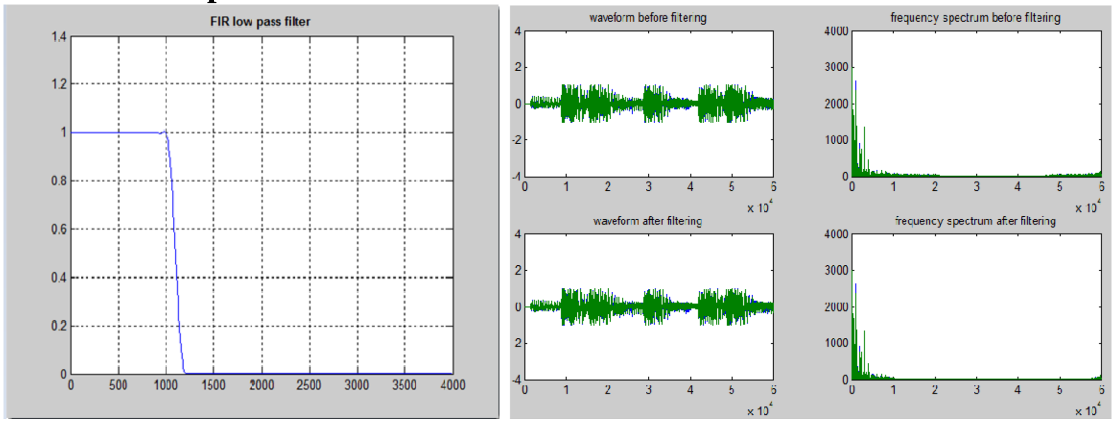


Figure 4-3 FIR low pass filter

In figure 4-3, the left figure is FIR low pass filter; we can see that the cutoff frequency is 1200, which means the performance of low pass filter is good. It has a perfect cutoff frequency that can filter all high components. The figure shows in right is the waveform and frequency spectrum before and after filtering. The X-label in waveform figure is timeline while Y-label is amplitude. In spectrum figure, X-label is frequency while Y-label is amplitude. We applied this filter to the noised signal; we can see that the high component has been removed. And the waveform’s width is reduced because of the removed noise. Then we playback the sound we heard the sound is lowering than before.

4.3.2 FIR high pass filter

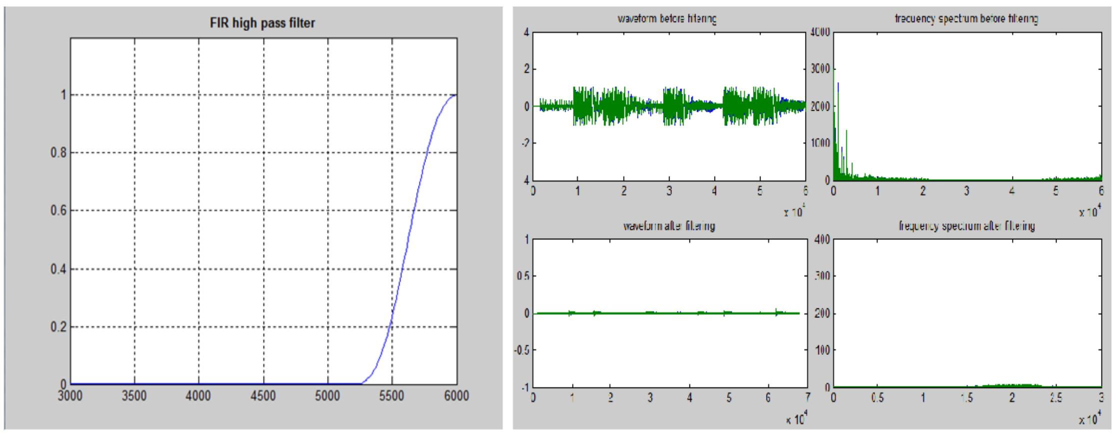


Figure 4-4 FIR high pass filter

In figure 4-4, the left figure is FIR high pass filter; the cutoff frequency is 5300Hz which means the frequency that blow 5300Hz can’t pass this filter so before the cutoff frequency the amplitude is 0 all the time. It has a perfect cutoff frequency that can filter all low components. The figure shows in right is the waveform and frequency spectrum before and after filtering. The X-label in waveform figure is timeline while the Y-label is amplitude. In spectrum figure, X-label is frequency while Y-label is amplitude. Form the figure we can see that the low frequency components has been removed and the waveform after filtering becomes almost 0. Then we playback the sound, we only heard some little voice that is because the sound that we applied is almost consist of low components.

4.3.3 FIR band pass filter

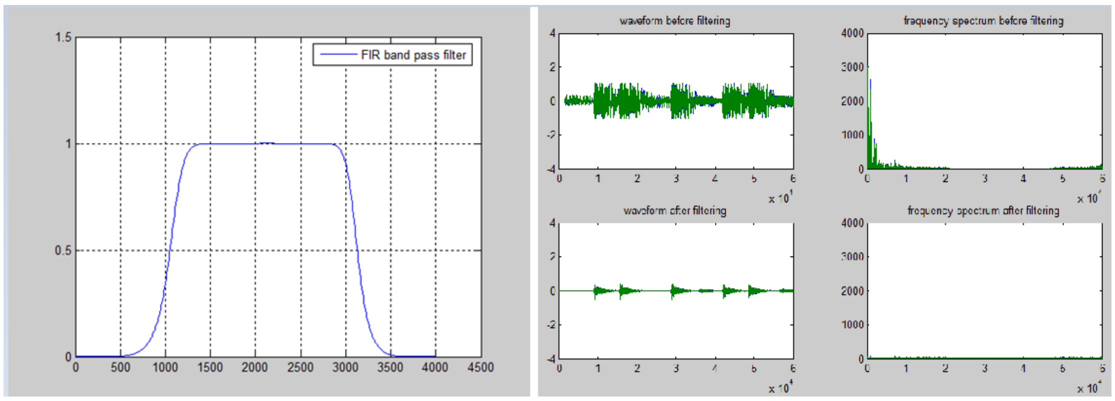


Figure 4-5 FIR band pass filter

In figure 4-5 shows FIR band pass filter. The left one is the performance of the FIR band pass filter and the right one is the waveform and frequency spectrum before filtering and after filtering. From the left figure, the cutoff frequency can be easily found, 500Hz and 3500Hz which means this filter only can let the frequency between 500Hz and 3500Hz pass and remove all other components. From the right figure, the X-label in waveform figure is timeline and the Y-label is amplitude. In the spectrum figure, the X-label is frequency and the Y-label is amplitude of the frequency. We can see that the difference of frequency spectrum between before filtering and after filtering very clearly, the unit of vertical axis is smaller, that means we removed the noise. And in frequency spectrum figure, the components besides two cutoff frequencies have been removed.

4.3.4 IIR low pass filter

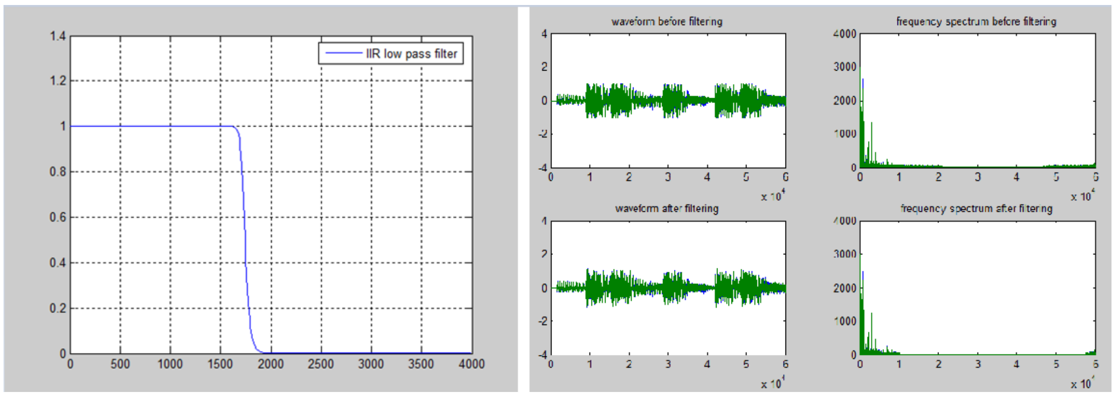


Figure 4-6 IIR low pass filter

In figure 4-6, the left figure is the performance of IIR low pass filter; the cutoff frequency can be seen is 1700, which means the performance of the low pass filter shows good. It has a perfect cutoff frequency and can filter all high components. And this figure is seems like the FIR low pass filter. The right figure shows the waveform and frequency spectrum before and after filtering. The X-label in waveform figure is timeline while Y-label is amplitude. In spectrum figure, the X-label is frequency while the Y-label is the amplitude of frequency. We compare these two figures, the frequency spectrum after filtering has good performance, it removes the high frequency and save the low frequency which is similar to FIR low pass filter and then we playback the sound, we also heard the lowering voice.

4.3.5 IIR high pass filter

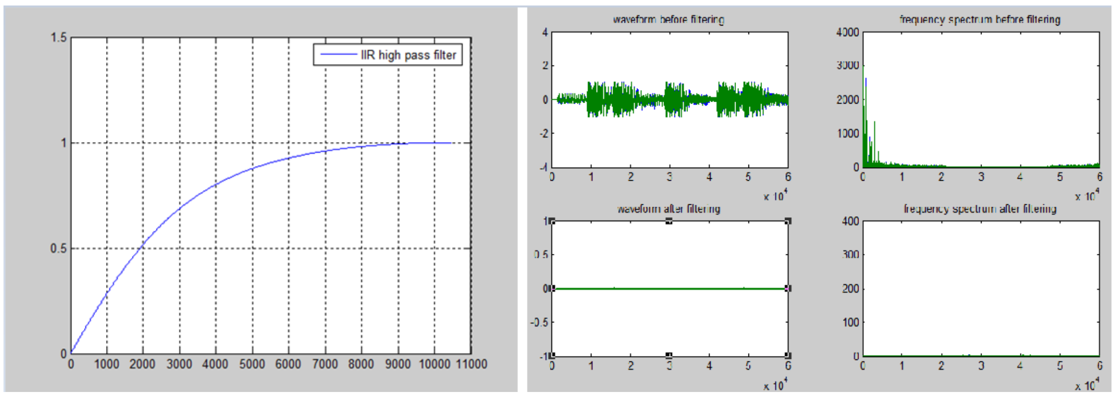


Figure 4-7 IIR high pass filter

In figure 4-7, the left figure shows good performance of IIR high pass filter which can filter all low frequency components. It is obviously that the cutoff frequency of IIR high pass filter is 3500Hz. And in the right figure, it shows the waveform and frequency spectrum before filtering and after filtering. The X-label and Y-label in waveform figure represents timeline and amplitude respectively while the X-label and Y-label in spectrum figure represents frequency and amplitude of frequency respectively. This result is similar to FIR high pass filter, the low components almost has been removed. Then we playback the sound, the result is similar to FIR high pass filter also, we can only hear little voice.

4.3.6 IIR band pass filter

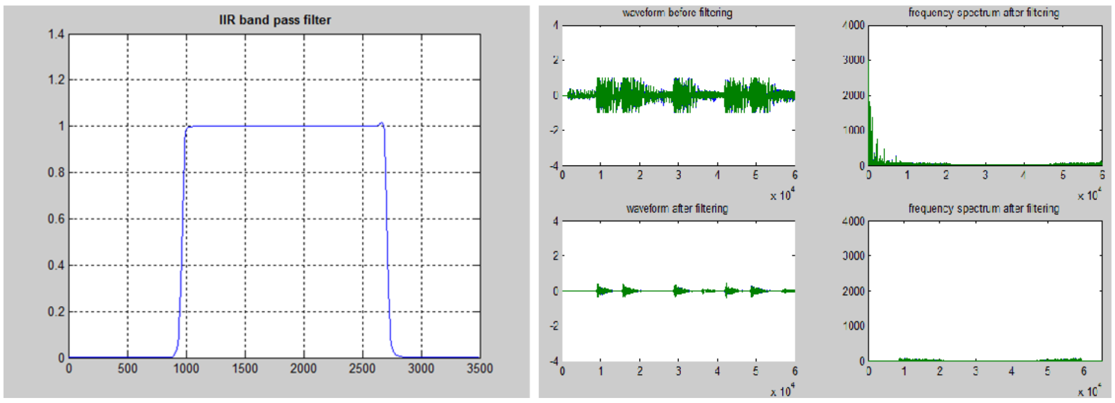


Figure 4-8 IIR band pass filter

From the figure 4-7, the left figure shows the performance of IIR band pass filter and the cutoff frequency can be easily found, 1000Hz and 2700Hz which means this filter only can let the frequency between 1000Hz and2700Hz pass and remove all other components. The right figure shows the waveform and frequency spectrum before filtering and after filtering. The X-label and Y- label represent timeline and amplitude in waveform figure and in spectrum figure X-label represents frequency, Y-label represents the amplitude of frequency. When the noised sound goes through the IIR band pass filter, we can find the spectrum is changed. The components besides two cutoff frequencies have been removed.

After hearing all the voice, the high pass filters has best performance that is because we used a low frequency sound source. These figures all show nice performance. Every filter can run well and will have good reliability to filter the noise of the sound.

4.4 BUILD GUI

In this design, 14 buttons, 4 static texts and the properties of buttons can be changed by double click. When you press the button in the interface, it will show the corresponding figure. We do this to help people use this system easily.

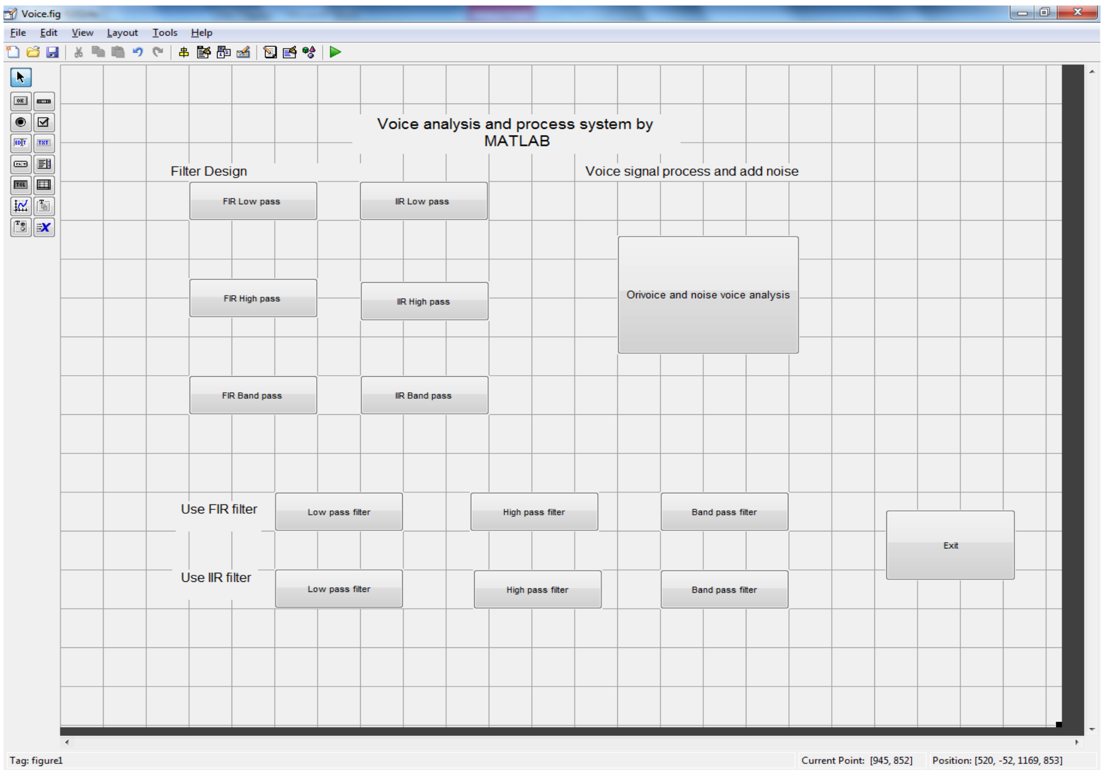


Figure 4-9 GUI design

5 .Conclusion and Discussion

This chapter will know the best filter among those filters, and our work will be discussed and some shortage will be known. That will help to improve this collector in future.

Play the sound in MATLAB by plus the noise processing. Its format is sound(y,Fs), sound(y) and sound(y,Fs,bits). There are big changes between filter. Since the voice we used is at high frequency, the high pass filter is the best.

This design chooses a voice signal to be the research object, uses MATLAB to do frequency spectrum, add noise to simulate a noised voice signal. Design FIR and IIR digital filter based on digital signal processing theory, and filter the noised voice signal, then analyze the characteristics of time domain and frequency domain, finally replay the voice signal. Furthermore, create a signal processing interface; complete a system to analysis and process voice signal by MATLAB. Do spectrum analysis of the original voice signal and noised voice signal well, use FFT we get the spectrogram of the signal. In filter part, this project is mainly started with Butterworth filter to design filter, and has achieved the expected de-noising effect well. Finally we create an interface design, barely accomplished record, open voice signal file, show the wave form before and after filtering, but this interface has not used other widgets like list box, pull-downed menu and so on, which has not made full use of the GUI interface design function.

From this thesis and design, conclusions can be achieved:

1) Using voice card and microphone in laptop which can be a sensor and collector in data collection system.
2) Using the strong data processing function in MATLAB, various voice signal analysis and process can be down well.
3) Using the filter in MATLAB, noise can be filtered and original sound will reappear.
4) Using GUI interface design in MATLAB, a good UI can be designed and we can get a complete system

Of course, there are lots of things which be improved in this design like power spectrum, making a more accurate filter parameter or smarter filter method and also this can be connected to LABVIEW, which optimizes the UI very much.

Reference

[1] P.I. Kattan and G.Z. Voyiadjis *MATLAB Guide to Finite Elements: An Interactive Approach*. 1st ed. New York: Springer Berlin Heidelberg, 2010.

[2]L.R.Rabiner and R.W.Schafer,"Introduction to Digital Speech Processing", *Journal of Foundations and Trends in Signal Processing,* vol 1, no.1, pp:1-149, Jan. 2007.

[3]Y. LI and M. GUO, "Audio Data Acquisition System Based on MATLAB", *Journal of Audio Engineering*, vol31, no.1, pp: 38-48, May 2007.

[4]X. Chen, J. Liu and C.Wu, "Long-distance Data Acquisition System Design Based on MATLAB" ,*Journal of New Technology & New Process*,vol 10, no.1, pp:108-210. August 2007.

[5]A.H. VanDoren, "Data Acquisition Systems",*Journal of Automatic data collection systems*, vol 1, no.1, p291, Jan.1982.

[6]H. Wang, G. Wang, X Jin and N Bai "Application of data acquisition system based on sound card and MATLAB in Young modulus measurement",*Journal of Physics Experimentation,* vol 4, no.1, pp:205-246, April 2005.

[7] M. D. Lutovac, D.V. Tošić and B. L. Evans. *Filter Design for Signal Processing Using MATLAB and Mathematica.* 4sted. Beijing: House of Electronics Industry, 2004.

[8]K. Feher, *Telecommunications Measurements, Analysis, and Instrumentation*, 1sted. New York: Englewood,1997.

[9]M. D. Lutovac, D.V. Tošić and B. L. Evans. *Filter Design for Signal Processing Using MATLAB andMathematica*, 4sted. Beijing: House of Electronics Industry, 2004.

[10]L.D. Paarmann, *Design and Analysis of Analog Filters: A Signal Processing Perspective*, 3rded. America: Springer, 2001.

[11] B. Venkataramani, *Digital Signal Processors*,2nded. New Delhi: Tata McGraw Hill Education Private Limited, 2010.

[12]B.A.Shenoi, *Introduction to Digital Signal Processing and Filter Design*, 1sted. Canada: John Wiley & Sons, 2006.

[13]E. Lai, *Practical Digital Signal Processing*, 1sted. America:Newnes, 2003.

[14]D. Patrick, S. Fardo, *Understanding AC Circuits*, 2ndEd. America:Newnes,1999

[15]J.D. Foley andA.V. Dam, *FundamentalsofInteractiveComputerGraphics*,1sted. California:Wesley Publishing Company, 2005.

Appendix

Program in GUI

```
functionvarargout = Voice(varargin)
% VOICE MATLAB code for Voice.fig
%      VOICE, by itself, creates a new VOICE or raises the existing
%      singleton*.
%
%      H = VOICE returns the handle to a new VOICE or the handle to
%      the existing singleton*.
%
%      VOICE('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in VOICE.M with the given input arguments.
%
%      VOICE('Property','Value',...) creates a new VOICE or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before Voice_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to Voice_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Voice

% Last Modified by GUIDE v2.5 26-May-2014 13:51:10

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
'gui_Singleton',  gui_Singleton, ...
'gui_OpeningFcn', @Voice_OpeningFcn, ...
'gui_OutputFcn',  @Voice_OutputFcn, ...
'gui_LayoutFcn',  [] , ...
'gui_Callback',   []);
ifnargin&&ischar(varargin{1})
gui_State.gui_Callback = str2func(varargin{1});
```

```
end

ifnargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before Voice is made visible.
functionVoice_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Voice (see VARARGIN)

% Choose default command line output for Voice
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Voice wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
functionvarargout = Voice_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
F1=7000;
F2=800;
F3=1000;
a=2*F2/F1;
b=2*F3/F1
r1=2;
r2=60;
p1=1-10.^(-r1/20);
s1=10.^(-r2/20);
FF=[a b];
ma=[1 0];
v=[p1 s1];
[A21,wA21,bt,Yp]=kaiserord(FF,ma,v);
B21=fir1(A21,wA21,kaiser(A21+1,bt));
[h,w]=freqz(B21,1);
figure;
plot(w*7000*0.5/pi,abs(h));
title('FIR low pass filter','fontweight','bold');
grid;



% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
F1=4000;
F2=3500;
F3=3000;
a=2*F2/F1;
b=2*F3/F1
r1=2;
r2=60;
p=1-10.^(-r1/20);
s=10.^(-r2/20);
FF=[b a];
ma=[0 1];
v=[p s];
[A23,wA23,bt,Yp]=kaiserord(FF,ma,v);
B23=fir1(A23,wA23,'high',kaiser(A23+1,bt));
 [h,w]=freqz(B23,1);
figure;
plot(w*10000*0.5/pi,abs(h));
```

```
title('FIR high pass filter', 'fontweight', 'bold');
grid;



% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
F21=1000;
F22=2500;
F31=3200;
F32=3000;
F1=7000;
a1=tan(pi*F21/F1);
a2=tan(pi*F22/F1)
b1=tan(pi*F31/F1);
b2=tan(pi*F32/F1);
w=a1*a2/b2;
bw=a2-a1;
a=2;
b=(a*a2-w.^2)/(bw*w);
 [A02,wA02]=buttord(a,b,2,60, 's');
[B02,A02]=butter(A02,wA02, 's');
 [n2,d2]=lp2bp(B02,A02,sqrt(a1*a2),bw);
[n22,deA02]=bilinear(n2,d2,0.5);
[h,w]=freqz(n22,deA02);
figure;
plot(w*6000*0.5/pi,abs(h));
title('FIR band pass filter', 'fontweight', 'bold');
grid;



% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
F1=7500;
F2=900;
F3=1100;
a=2*pi*F2/F1;
```

```
b=2*pi*F3/F1;
F2=2*F1*tan(a/2);
F3=2*F3*tan(a/2);
[C11,wC11]=buttord(a,b,2.60,'s');
[D11,C11]=butter(C11,wC11, 's');
 [m11,deC11]=bilinear(D11,C11,0.5);
 [h,w]=freqz(m11,deC11);
figure;
plot(w*7000*0.5/pi,abs(h));
title('IIR low pass filter', 'fontweight', 'bold');
grid;



% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
F1=8000;
F2=4000;
F3=3600;
a1=tan(pi*F2/F1);
b1=tan(pi*F3/F1);
a=2;
b=a1*a/b1;
[D13,wD13]=cheb1ord(a,b,2,60, 's');
[C13,D13]=cheby1(D13,1,wD13, 's');
[SHL,den]=lp2hp(C13,D13,wD13);
[n13,deD13]=bilinear(SHL,den,0.5);
[h,w]=freqz(n13,deD13);
figure;
plot(w*18000*0.5/pi,abs(h));
title(' IIR high pass filter', 'fontweight', 'bold');
grid;



% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
F21=1100;
```

```
F22=3000;
F31=900;
F32=3100;
F1=7800;
a1=tan(pi*F21/F1);
a2=tan(pi*F22/F1)
b1=tan(pi*F31/F1);
b2=tan(pi*F32/F1);
w=a1*a2/b2;
ww=a2-a1;
a=1;
b=(a1*a2-w.^2)/(ww*w);
[D12,wD12]=buttord(a,b,1,50, 's');
[C12,D12]=butter(D12,wD12, 's');
[n2,d2]=lp2bp(C12,D12,sqrt(a1*a2),ww);
 [SHL12,deD12]=bilinear(n2,d2,0.5);
[h,w]=freqz(SHL12,deD12);
figure;
plot(w*7000*0.5/pi,abs(h));
title(' IIR band pass filter', 'fontweight', 'bold');
grid;



% --- Executes on button press in pushbutton8.
function pushbutton8_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
n=10;
fs=16000;
channel=1;
y=audiorecorder(n*fs,fs,channel,'unit8');
pause
audioplayer(y,fs);
figure;
plot(y);
audiowrite(y,fs,'OriSound.wav');
pause
[y,fs,bits]=('OriSound.wav');

% --- Executes on button press in pushbutton9.
function pushbutton9_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)



% --- Executes on button press in pushbutton10.
function pushbutton10_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton11.
function pushbutton11_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton12.
function pushbutton12_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

[y1,fs,bits]=wavread('D:\Orivoice.wav');
nor1=y1/max(y1);
sound(nor1,fs,bits);
data=round(32767*nor1);
t1=fopen('data.dat','w');
fprintf(t1,'1651 1 0 0 0\n');
fprintf(t1,'%d\n',data);
fclose(t1);
n=length(y1);
Noise=0.1*randn(size(y1));
s=y1+Noise;
S=fft(s,n);
normal=s/max(s);
cha=round(32767*normal);
pause(4);
sound(normal,fs,bits);
ch=fopen('cha.dat','w');
fprintf(ch,'1651 1 0 0 0\n');
fprintf(ch,'%d\n',cha);
```

```
fclose(ch);
F1=7000;
F2=1000;
F3=1100;
w1=2*pi*F2/F1;
w2=2*pi*F3/F1;
r1=1;
r2=50;
p=1-10.^(-r1/20);
q=10.^(-r2/20);
fp=[w1 w2];
mg=[1 0];
de=[p q];
[A21,wA21,beta,ftype]=kaiserord(fp,mg,de);
C21=fir1(A21,wA21,kaiser(A21+1,beta));
D21=fftfilt(C21,s);
sound(D21,fs,bits);
%sound(D21);
E21=fft(D21);
figure;
subplot(3,3,1);
plot(abs(S), 'g');
title('signal spectrum before filtering', 'fontweight', 'bold');
grid;
subplot(3,3,2);
plot(abs(E21), 'r');
title('signal spectrum after filtering', 'fontweight', 'bold');
grid;
subplot(3,3,3);
plot(s);
title('waveform before filtering', 'fontweight', 'bold');
grid;
subplot(3,3,4);
plot(D21);
title('waveform after filtering', 'fontweight', 'bold');
grid;


% --- Executes on button press in pushbutton13.
function pushbutton13_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
[y,fs,bits]=wavread('D:\Orivoice.wav');
m=max(y);
normal1=y/max(y);
sound(normal1,fs,bits);
indata=round(32767*normal1);
tem1=fopen('indata.dat','w');
fprintf(tem1,'1651 1 0 0 0\n');
fprintf(tem1,'%d\n',indata);
fclose(tem1);
%sound(y,fs,nbits);
n=length(y);
Noise=0.1*randn(size(y));
s=y+Noise;
S=fft(s,n);
normal=s/max(s);
chang=round(32767*normal);
pause(4);
sound(normal,fs,bits);
ch=fopen('chang.dat','w');
fprintf(ch,'1651 1 0 0 0\n');
fprintf(ch,'%d\n',chang);
fclose(ch);
F1=8001;
F2=4000;
F3=3200;
w1=2*F2/F1;
w2=2*F3/F1;
r1=1;
r2=100;
p=1-10.^(-r1/20);
q=10.^(-r2/20);
fp=[w2 w1];
mg=[0 1];
de=[p q];
[A23,wA23,BT,ftype]=kaiserord(fp,mg,de);
C23=fir1(A23,wA23,'high',kaiser(A23+1,BT));
D23=fftfilt(C23,s);
sound(D23,fs,bits);
E23=fft(D23);
figure;
subplot(3,3,1);
plot(abs(S), 'g');
title('signal spectrum before filtering', 'fontweight', 'bold');
grid;
```

```
subplot(3,3,2);
plot(abs(E23), 'r');
title('signal spectrum after filtering', 'fontweight', 'bold');
grid;
subplot(3,3,3);
plot(s);
title('waveform before filtering', 'fontweight', 'bold');
grid;
subplot(3,3,4);
plot(D23);
title('waveform after filtering', 'fontweight', 'bold');
grid;
% --- Executes on button press in pushbutton14.
function pushbutton14_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[y,fs,bits]=wavread('D:\Orivoice.wav');
m=max(y);
normal1=y/max(y);
sound(normal1,fs,bits);
indata=round(32767*normal1);
tem1=fopen('indata.dat','w');
fprintf(tem1,'1651 1 0 0 0\n');
fprintf(tem1,'%d\n',indata);
fclose(tem1);
n=length(y);
Noise=0.1*randn(size(y));
s=y+Noise;
S=fft(s,n);
normal=s/max(s);
chang=round(32767*normal);
pause(4);
sound(normal,fs,bits);
ch=fopen('chang.dat','w');
fprintf(ch,'1651 1 0 0 0\n');
fprintf(ch,'%d\n',chang);
fclose(ch);
F11=1200;
F12=3000;
Fc1=1000;
Fc2=3200;
Fa=2200;
w11=tan(pi*F11/Fa);
```

```
w12=tan(pi*F12/Fa);
w21=tan(pi*Fc1/Fa);
w22=tan(pi*Fc2/Fa);
w=w11*w12/w22;
bw=w12-w11;
w1=1;
w2=(w1*w12-w.^2)/(bw*w);
[A22,wA22]=buttord(w1,w2,1,50, 's');
[C22,a22]=butter(A22,wA22, 's');
D22=fftfilt(C22,s);

sound(D22,fs,bits);
E22=fft(D22);
figure;
subplot(3,3,1);
plot(abs(S), 'g');
title('signal spectrum before filtering', 'fontweight', 'bold');
grid;
subplot(3,3,2);
plot(abs(E22), 'r');
title('signal spectrum after filtering', 'fontweight', 'bold');
grid;
subplot(3,3,3);
plot(s);
title('waveform before filtering', 'fontweight', 'bold');
grid;
subplot(3,3,4);
plot(D22);
title('waveform after filtering', 'fontweight', 'bold');
grid;


% --- Executes on button press in pushbutton15.
function pushbutton15_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[y,fs,bits]=wavread('D:\Orivoice.wav');
m=max(y);
normal1=y/max(y);
sound(normal1,fs,bits);
indata=round(32767*normal1);
tem1=fopen('indata.dat','w');
fprintf(tem1,'1651 1 0 0 0\n');
fprintf(tem1,'%d\n',indata);
fclose(tem1);
n=length(y);
Noise=0.1*randn(size(y));
s=y+Noise;
S=fft(s,n);
normal=s/max(s);
chang=round(32767*normal);
pause(4);
sound(normal,fs,bits);
ch=fopen('chang.dat','w');
fprintf(ch,'1651 1 0 0 0\n');
fprintf(ch,'%d\n',chang);
fclose(ch);
F1=8000;
F2=1000;
F3=1200;
w1=2*pi*F2/F1;
w2=2*pi*F3/F1;
fp=2*F1*tan(w1/2);
fs=2*F3*tan(w2/2);
[A11,wA11]=buttord(w1,w2,1,50,'s');
[C11,B11]=butter(A11,wA11,'s');
[N11,deA11]=bilinear(C11,B11,0.5);
[h,w]=freqz(N11,deA11);
D11=filter(N11,deA11,s);
pause(4);
normal2=D11/max(D11);
sound(normal2,fs,bits);
chang1=round(32767*normal2);
ch1=fopen('chang1.dat','w');
fprintf(ch1,'1651 1 0 0 0\n');
fprintf(ch1,'%d\n',chang1);
fclose(ch1);
F11=fft(D11);
figure;
subplot(3,3,1);
plot(abs(S), 'g');
title('signal spectrum before filtering', 'fontweight', 'bold');
grid;
subplot(3,3,2);
plot(abs(F11), 'r');
title('signal spectrum after filtering', 'fontweight', 'bold');
grid;
```

```
subplot(3,3,3);
plot(s);
title('waveform before filtering', 'fontweight', 'bold');
grid;
subplot(3,3,4);
plot(D11);
title('waveform after filtering', 'fontweight', 'bold');
grid;



% --- Executes on button press in pushbutton16.
function pushbutton16_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton16 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[y,fs,bits]=wavread('D:\Orivoice.wav');
m=max(y);
normal1=y/max(y);
sound(normal1,fs,bits);
indata=round(32767*normal1);
tem1=fopen('indata.dat','w');
fprintf(tem1,'1651 1 0 0 0\n');
fprintf(tem1,'%d\n',indata);
fclose(tem1);
n=length(y);
Noise=0.1*randn(size(y));
s=y+Noise;
S=fft(s,n);
normal=s/max(s);
chang=round(32767*normal);
pause(4);
sound(normal,fs,bits);
ch=fopen('chang.dat','w');
fprintf(ch,'1651 1 0 0 0\n');
fprintf(ch,'%d\n',chang);
fclose(ch);
F1=8000;
F2=1200;
F3=1000;
w11=tan(pi*F2/F1);
w21=tan(pi*F3/F1);
w1=1;  w2=w11*w1/w21;
[A12,wA12]=cheb1ord(w1,w2,1,50,'s');
```

```
[B12,C12]=cheby1(A12,1,wA12,'s');
[NM,DE]=lp2hp(B12,C12,wA12);
[NG12,deA12]=bilinear(NM,DE,0.5);
[h,w]=freqz(NG12,deA12);
F12=filter(NG12,deA12,s);
sound(F12,fs,bits);
G12=fft(F12);
figure;
subplot(3,3,1);
plot(abs(S), 'g');
title('signal spectrum before filtering', 'fontweight', 'bold');
grid;
subplot(3,3,2);
plot(abs(G12), 'r');
title('signal spectrum after filtering', 'fontweight', 'bold');
grid;
subplot(3,3,3);
plot(s);
title('waveform before filtering', 'fontweight', 'bold');
grid;
subplot(3,3,4);
plot(F12);
title('waveform after filtering', 'fontweight', 'bold');
grid;


% --- Executes on button press in pushbutton17.
function pushbutton17_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton17 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[y,fs,bits]=wavread('D:\Orivoice.wav');
m=max(y);
normal1=y/max(y);
sound(normal1,fs,bits);
indata=round(32767*normal1);
tem1=fopen('indata.dat','w');
fprintf(tem1,'1651 1 0 0 0\n');
fprintf(tem1,'%d\n',indata);
fclose(tem1);
n=length(y);
Noise=0.1*randn(size(y));
s=y+Noise;
S=fft(s,n);
```

```
normal=s/max(s);
chang=round(32767*normal);
pause(4);
sound(normal,fs,bits);
ch=fopen('chang.dat','w');
fprintf(ch,'1651 1 0 0 0\n');
fprintf(ch,'%d\n',chang);
fclose(ch);
F11=1200;
F12=3000;
F01=1000;
F02=3200;
F1=8000;
w11=tan(pi*F11/F1);
w12=tan(pi*F12/F1);
w21=tan(pi*F01/F1);
w22=tan(pi*F02/F1);
w=w11*w12/w22;
bw=w12-w11;
w1=1;
w2=(w1*w12-w.^2)/(bw*w);
[A12,wA12]=buttord(w1,w2,1,50, 's');
[B12,a12]=butter(A12,wA12, 's');
[NM2,DE2]=lp2bp(B12,a12,sqrt(w11*w12),bw);
[NG12,deA12]=bilinear(NM2,DE2,0.5);
C21=filter(NG12,deA12,s);
sound(C21,fs,bits);
E21=fft(C21);
figure;
subplot(3,3,1);
plot(abs(S), 'g');
title('signal spectrum before filtering', 'fontweight', 'bold');
grid;
subplot(3,3,2);
plot(abs(E21), 'r');
title('signal spectrum after filtering', 'fontweight', 'bold');
grid;
subplot(3,3,3);
plot(s);
title('waveform before filtering', 'fontweight', 'bold');
grid;
subplot(3,3,4);
plot(C21);
title('waveform after filtering', 'fontweight', 'bold');
```

```
grid;



% --- Executes on button press in pushbutton20.
function pushbutton20_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton20 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


%
--------------------------------------------------------------------
function Untitled_1_Callback(hObject, eventdata, handles)
% hObject    handle to Untitled_1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```