信号分析与处理综合实验报告

班级： 通信工程 171 班

姓名： YUANFEI

学号：XXXXXXXXXXXXXXX

基于MATLAB GUI的音乐合成综合实验

1.实验要求

① 给定一段乐谱，将其合成并播放。要求合成的结果比较自然。

② 用傅里叶级数分析一段音乐，画出频谱图，该段音乐是可选择的。

③ 分别设计一个低通滤波器和高通滤波器（截止频率自己设定一个），对音乐进行滤波并播放，比较滤波之后的效果；

④ 完成最终的GUI界面设计。

2. 实验基本原理

2.1 傅里叶变换

傅里叶变换建立了信号频谱的概念。所谓傅里叶分析即分析信号的频谱（频率构成）、频带宽度等。要想合成一段音乐，就要了解该段音乐的基波频率，谐波构成等，因此，必须采用傅里叶变换这一工具。对于连续时间信号$f(t)$，其傅里叶变换为：

$$F(w) = F|f(t)| = \int_{-\infty}^{\infty} f(t)e^{-jwt}dt$$

由于其变换两边的函数$f(t)$和$F(w)$都是连续函数，不适合于计算机处理。MATLAB语言提供了符号函数fourier来实现傅里叶变换，但该函数需要信号的解析表达式。而工程应用中经常需要对抽样数据进行傅里叶分析，这种情况下往往无法得到信号的解析表达式，必须采用傅里叶变换的数值计算方法。

如果$f(t)$的主要取值区间为$[\mathrm{t1,t2}]$，定义$\mathrm{T = t1 - t2}$为区间长度。在该区间抽样N个点，抽样间隔为：$\triangle \mathrm{t = T/N}$

则有：

$$\mathrm{F}(w_1 + \mathrm{k}\Delta\mathrm{w}) = \Delta\mathrm{t}\sum_{0}^{N-1} f(t_1 + n\Delta t)\, e^{-j(w_1 + kw)(t_1 + n\Delta t)}\Delta t$$

可以计算出任意频点的傅里叶变换值，假设$F(w)$的主要取值区间位于$[\mathrm{w1,w2}]$，要计算其间均匀抽样的k个值，则有：

$$\mathrm{F(w)} = \sum_{0}^{N-1} f(t_1 + n\Delta t)\, e^{-jw(t_1 + n\Delta t)}\Delta t$$

式中，$\triangle \mathrm{w} = (w_2 - w_1)/\mathrm{k}$为频域抽样间隔。

2.2 快速傅里叶变换

快速傅里叶变换（英语：Fast Fourier Transform, FFT），是快速计算序列的离散傅里叶变换（DFT）或其逆变换的方法。傅里叶分析将信号从原始域（通常是时间或空间）转换到频域的表示或者逆过来转换。FFT 会通过把 DFT 矩阵分解为稀疏（大多为零）因子之积来快速计算此类变换。因此，它能够将计算 DFT 的复杂度从只用 DFT 定义计算需要的$O(n^2)$，降低到$O(n\,log\,n)$，其中为数据大小。

快速傅里叶变换广泛的应用于工程、科学和数学领域。这里的基本思想在 1965 年才得到普及，但早在 1805 年就已推导出来。1994 年美国数学家吉尔伯特・斯特朗把 FFT 描述为“我们一生中最重要的数值算法”，它还被 IEEE 科学与工程计算期刊列入 20 世纪十大算法。

计算离散傅里叶变换的快速方法，有按时间抽取的 FFT 算法和按频率抽取的 FFT 算法。前者是将时域信号序列按偶奇分排，后者是将频域信号序列按偶奇分排。它们都借助于的两个特点：一是周期性；二是对称性，这里符号*代表其共轭。这样，便可以把离散傅里叶变换的计算分成若干步进行，计算效率大为提高。

2.3 FIR 滤波器

有限冲激响应（Finite impulse response，缩写 FIR）滤波器是数位滤波器的一种，简称 FIR 数位滤波器。这类滤波器对于脉冲输入信号的响应最终趋向于 0，因此是有限的，而得名。它是相对于无限冲激响应（IIR）滤波器而言。由于无限冲激响应滤波器中存在反馈回路，因此对于脉冲输入信号的响应是无限延续的。

有限冲激响应滤波器是一线性系统，输入信号$\mathrm{x}(0), \mathrm{x}(1), \dots, \mathrm{x}(\mathrm{n})$经过该系统后的输出信号，$y(n)$可表示为：

$$y(n) = h_0 x(n) + h_1 x(n-1) + \dots + h_N x(n-N)$$

其中，$h_0, h_1, \dots, h_N$是滤波器的冲激响应，通常称为滤波器的系数。N 是滤波器的阶数。上式也可表示为：

$$y(n) = \sum_{k=0}^{N} h_k\, x(m-k)$$

如果输入信号为脉冲信号

$$\delta(n) = \begin{cases} 1 & n = 0 \\ 0 & n \neq 0 \end{cases}$$

输出信号则为：

$$y(n) = \sum_{k=0}^{N} h_k\, \delta(n-k) = h_n$$

这也是冲激响应$h(n)$得名的原因，即，它是滤波器脉冲输入的响应。 有限冲激响应滤波器的传递函数可由其冲激响应的 z 变换获得：

$$H(z) = Z\{h(n)\} = \sum_{n=-\infty}^{\infty} h(n)\, Z^{-n} = \sum_{n=0}^{N} h(n)\, Z^{-n}$$

因此，有限冲激响应滤波器的频率响应为：

$$H(e^{-j\omega}) = \sum_{n=0}^{N} h(n)\, e^{-j\omega n}$$

滤波器设计是一个创建满足指定滤波要求的滤波器参数的过程。 只有完成了滤波器的设计和实现，才能最终完成数据的滤波。

MATLAB 的信号处理工具箱软件提供了两种方式设计滤波器：面向对象的和非面向对象的。面向对象的方法首先创建一个滤波器对象 fdesign，然后调用合适的 design 参数设计。

非面向对象的方法则适用函数实现滤波器设计，如 butter、firpm。所有非面向对象的滤波器设计函数使用的是归一化频率，归一化频率$[0,1]$之间，1 表示πrad。

本次实验设计采用面向对象的滤波器。

3. 实现方法

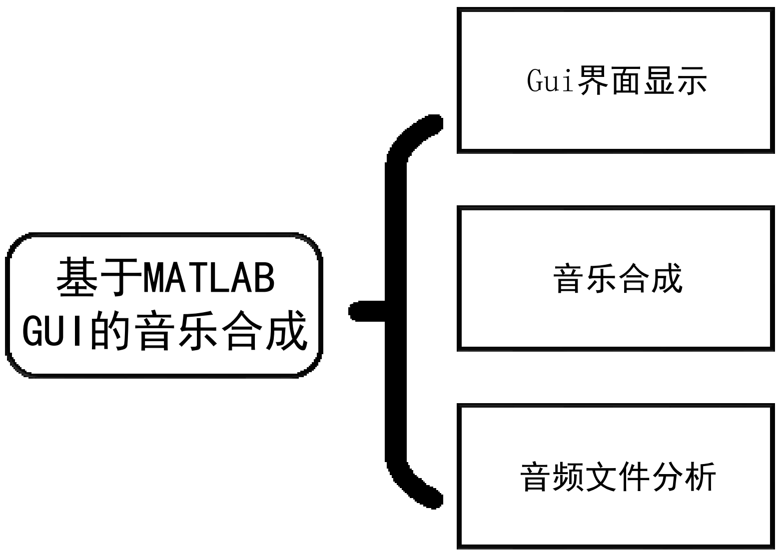

图 1 基于 MATLAB GUI 的音乐合成总系统框图

程序的总系统框图如上，本程序分为三大模块，GUI 界面显示，音乐合成，音频文件分析。程序的功能依托于 GUI 界面的显示；音乐合成模块实现了数字语音的合成；音频文件分析模块则实现了音频文件的分析和高通低通滤波。通过以上三个模块的配合，实现了系统的设计。

3.1 GUI 界面显示

此模块的设计依托于 MATLAB 内置的 GUI 设计模块，在命令行输入 guide 即可调出模块进行编程。

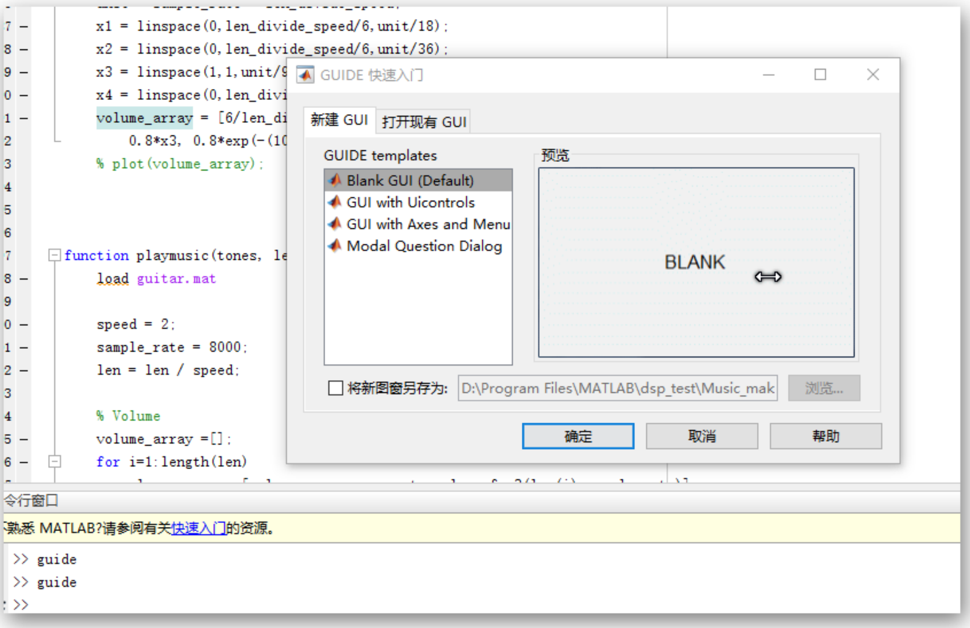

图 2 guide 模块示意图

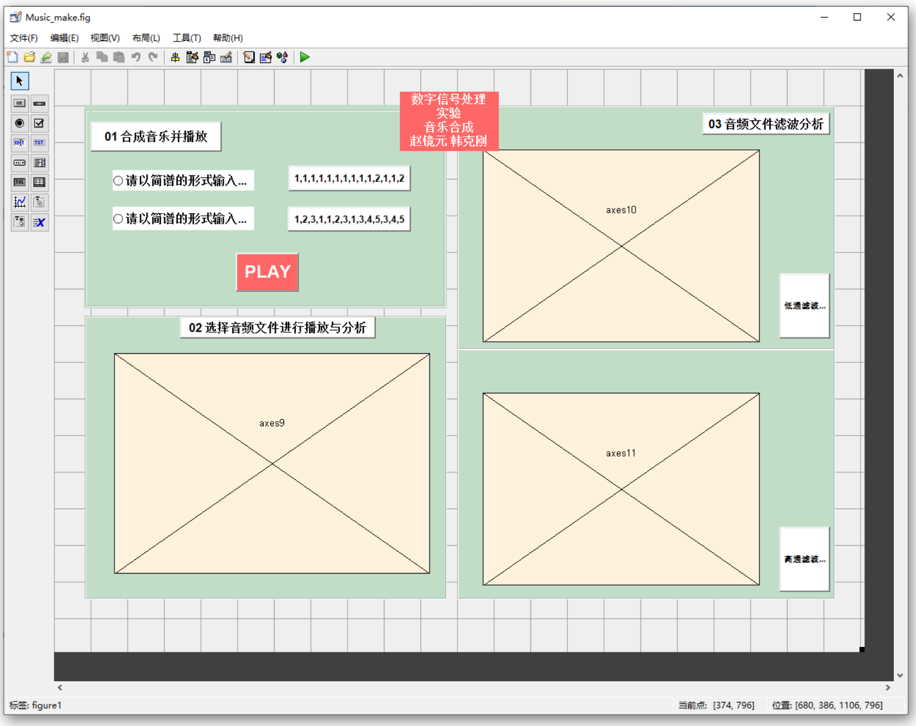

图 3 GUI 模块编程图

为其添加相应控件后，在控件上设置 callback 回调函数即可与 MATLAB 文件建立对应关系，从而完成 GUI 的设计。

定义 Music_make_OpeningFcn(hObject, eventdata, handles, varargin) 函数，在 Music_make 可见前执行，其中 varargin 将命令行参数转到 Music_make。

使用 handles.output = hObject 方法，选择音乐制作的默认命令行输出，并利用 function varargout = Music_make_OutputFcn(hObject, eventdata, handles) 将此函数的输出返回到命令行。实现用于返回输出参数的 varargout 单元数组。

在有了 GUI 图形设计后，开始编写相关函数文件，从而实现任务要求。

3.2 音乐合成模块的实现

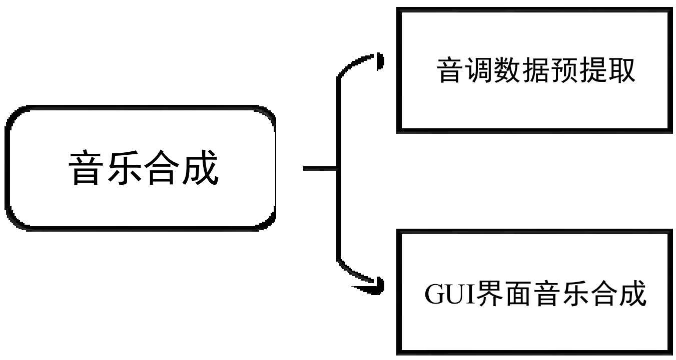

图 4 音乐合成模块程序框图

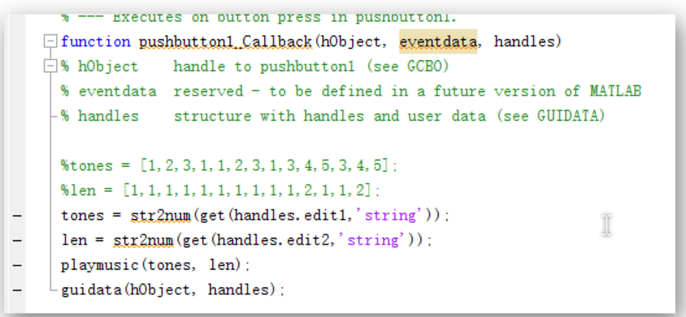

图 5 音乐合成 GUI 函数图

音乐合成模块分为两个部分，音调数据预提取；GUI 界面音乐合成。音调数据预提取子模块用于提取音调的频率，为了更精确，本程序提取了基本频率，2，3，4 次谐波分量。用 Analyze_fmt.m 分析音频的到数据 GUItar.mat，从而为 GUI 界面音乐合成提供原始音调，实现数字语言合成。

```
function playmusic(tones, len)
    load guitar.mat
    speed = 2;
    sample_rate = 8000;
    len = len / speed;
    % Volume
    volume_array =[];
    for i=1:length(len)
       volume_array = [volume_array, generate_volume_for3(len(i),sample_rate)];
    end
    %      F(1), G(2), A(3),  B-(4),  C(5),   D(6),   E(7)
    f = [349.23, 392,  440,  466.16, 523.25, 587.33, 659.25];
    tone = f(tones);
    % Generate Harmonic Sin Signal
    y = [];
    for i = 1:length(tone)
        t = linspace(0,len(i),len(i)*sample_rate);

        [val, index] = min( abs(tone(i) - base) );

        y = [y, [1, two_standard(index), three_standard(index), four_standard(index)] * ...
            [sin(2*pi*tone(i)*t); sin(2*pi*2*tone(i)*t);...
            sin(2*pi*3*tone(i)*t); sin(2*pi*4*tone(i)*t)]];
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % y suppressed by volume
    y = y .* volume_array;
    % Make sound
    sound(y, sample_rate);
```

图 6 playmusic()函数图

GUI 界面音乐合成模块依托于 3.1 中 GUI 界面显示的控件回调。当用户在 GUI 界面输入好要生成的音调和节拍的时候，点击“play”按钮即可生成音乐并播放。此功能依托于 GUI 控件的传值和 playmusic()函数的设计。先从 handles 找到需要的数据，传递给 playmusic()函数，对照 GUItar.mat 的音调得到音乐波形，最后用 sound()函数进行音乐播放，从而完成音乐合成的功能。

3.3 音频文件分析模块的实现

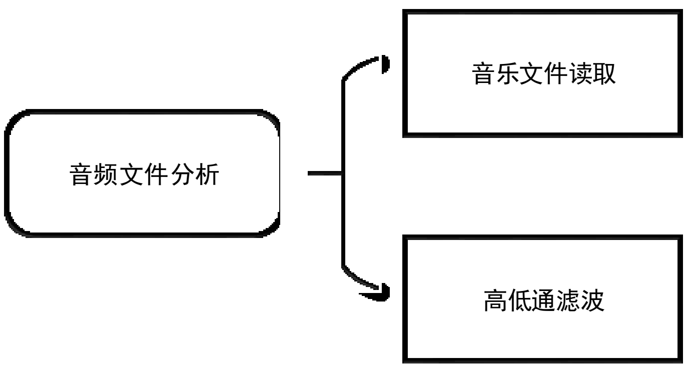

图 7 音频文件分析模块程序框图

音频文件分析模块实现了音乐文件的读取和高通低通波形分析。利用 GUI 控件结合 MATLAB 内置函数读取音乐文件。此处引入 uigetfile()弹框函数获取文件路径，用 audioread()读取文件，得到数据后用 FFT 变换获得频谱，从而完成幅度谱的绘制。

```
% --- 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)
%clear sound
[FileName,PathName]=uigetfile('*','select the file');
file = fullfile(PathName,FileName);
%data = load(file);
%handles.data = data;
[audio,fs ]= audioread(file);
sound(audio,fs);
figure(1);
NFFT = 2^nextpow2(fs);
audio_fft = fft(audio,NFFT);

audio_fft = audio_fft(1:ceil(length(audio_fft) / 2));
subplot(2,1,1);
plot(audio);
title('时域信号');
subplot(2,1,2);
plot(abs(audio_fft));
title('频域信号');
handles.data_audio = audio;
handles.data_audio_fft = audio_fft;
handles.data_fs = fs;
guidata(hObject,handles);
```

图 8 音频文件分析 GUI 函数图

在读取文件后，本程序会保存数据到 GUI 的 handles 容器里面。为高通低通滤波提供源数据。

```
% --- 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)
fs = handles.data_fs;
d = fdesign.highpass('Fst,Fp,Ast,Ap',3000/fs,5000/fs,60,1);
fliter_high = design(d,'equiripple');
data_t = handles.data_audio;
fs = handles.data_fs;
result_t = filter(fliter_high,data_t);
NFFT = 2^nextpow2(fs);
result_f=fft(result_t,NFFT);
result_f = result_f(1:ceil(length(result_f) / 2));
figure(3);
audio=handles.data_audio ;
subplot(3,1,1);
plot(audio);
title('原始时域信号');
subplot(3,1,2);
plot(result_t);
title('时域信号');
subplot(3,1,3);
plot(abs(result_f));
title('频域信号');
sound(result_t,fs)
```

图 9 音频文件分析滤波器实现函数图

高低通滤波的实现没有本质区别，本程序直接引入 MATLAB 中类滤波器进行设计，面向对象的方法首先创建一个滤波器对象 fdesign，然后调用 design 参数进行设计。这中面向对象的设计方法是 MATLAB 提供的简单实现方法，开发人员只需给的相关类型和技术指标即可生成滤波器。极大地方便了我们对于滤波器的设计。

4. 实验结果和讨论

本实验程序运行的结果如下图所示，可以看出，此程序正确、完整的完成了所要求的任务和功能。

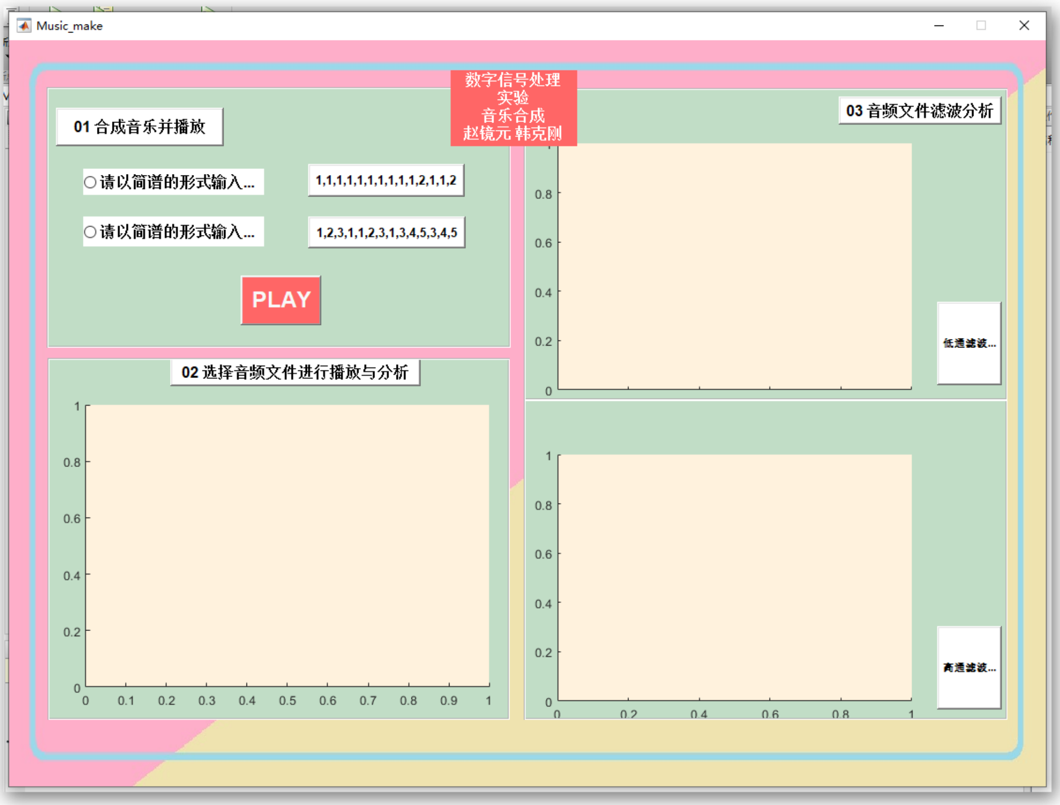

图 10 程序主界面运行图

图 10 为主程序主界面最初的运行图，可以明显的看到，程序分为三个模块，每个模块都对应一个实验要求，当输入好对应的音调和节拍后，点击 PLAY 键即可产生音乐。运行后 GUI 界面没有变化，但计算机会产生相应音乐，图 10 对应的音乐为 “两只老虎”，可以明显，清晰地听到他的声音。从而完成了给定一段乐谱，将其合成并播放。要求合成的结果比较自然的实验要求。

图 11 为音频文件分析示意图，通过操作，实现音频文件分析的功能。点击按键“02 选择音频文件进行播放与分析后会自动弹出”文件选择框。

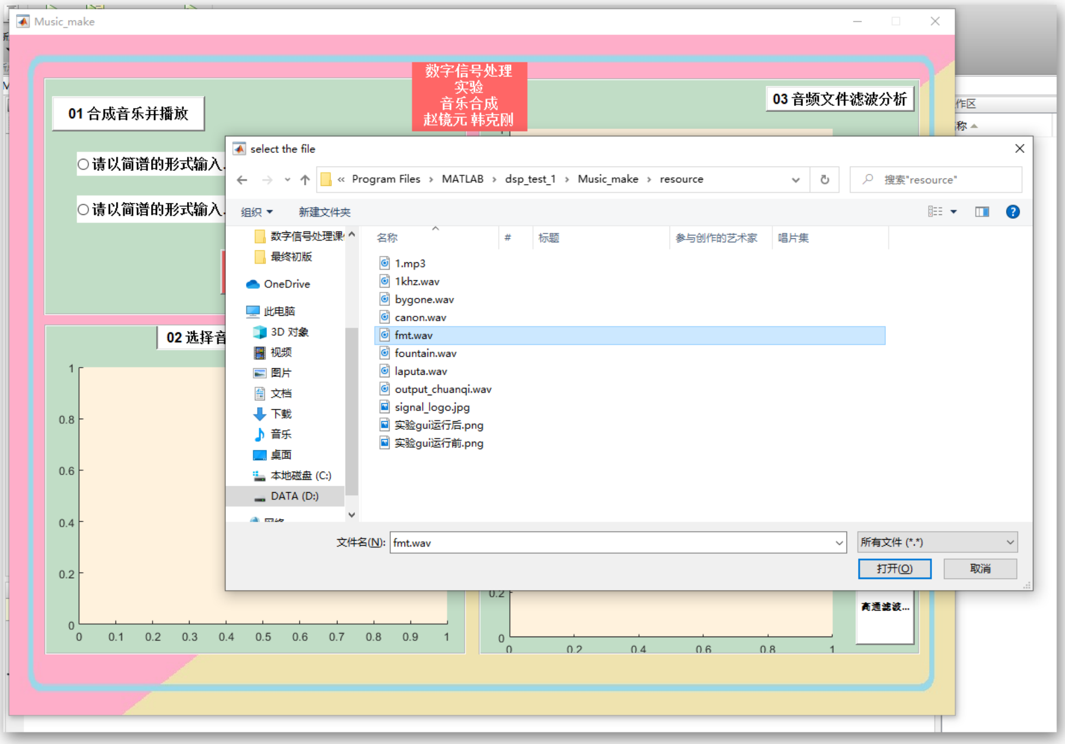

图 11 选择分析文件图

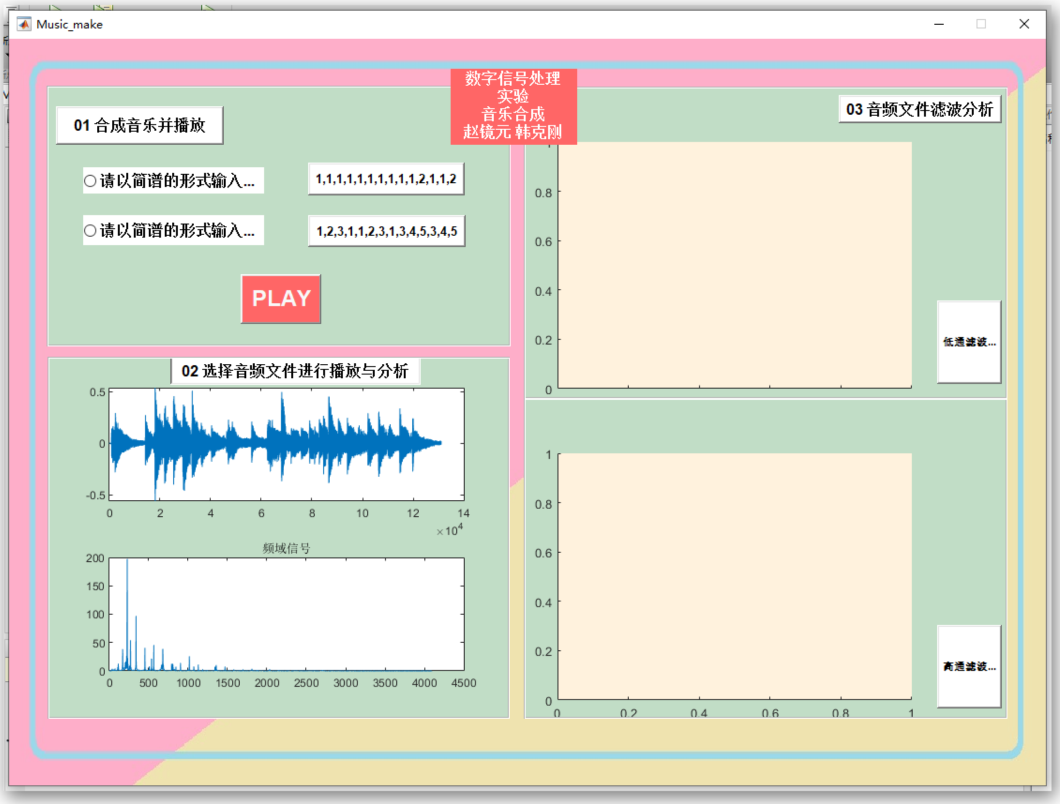

图 12 分析文件波形图

图 12 为所选音频文件的时域波形和频谱图，没有滤波处理，符合实际情况。在 02 区域绘图过程中，同时也会伴随音频播放，此音频来自于采样后的数字波形数据，播放出来的声音较好的对真实数据进行了还原，从而实现了用傅里叶级数分析一段音乐，画出频谱图的功能。

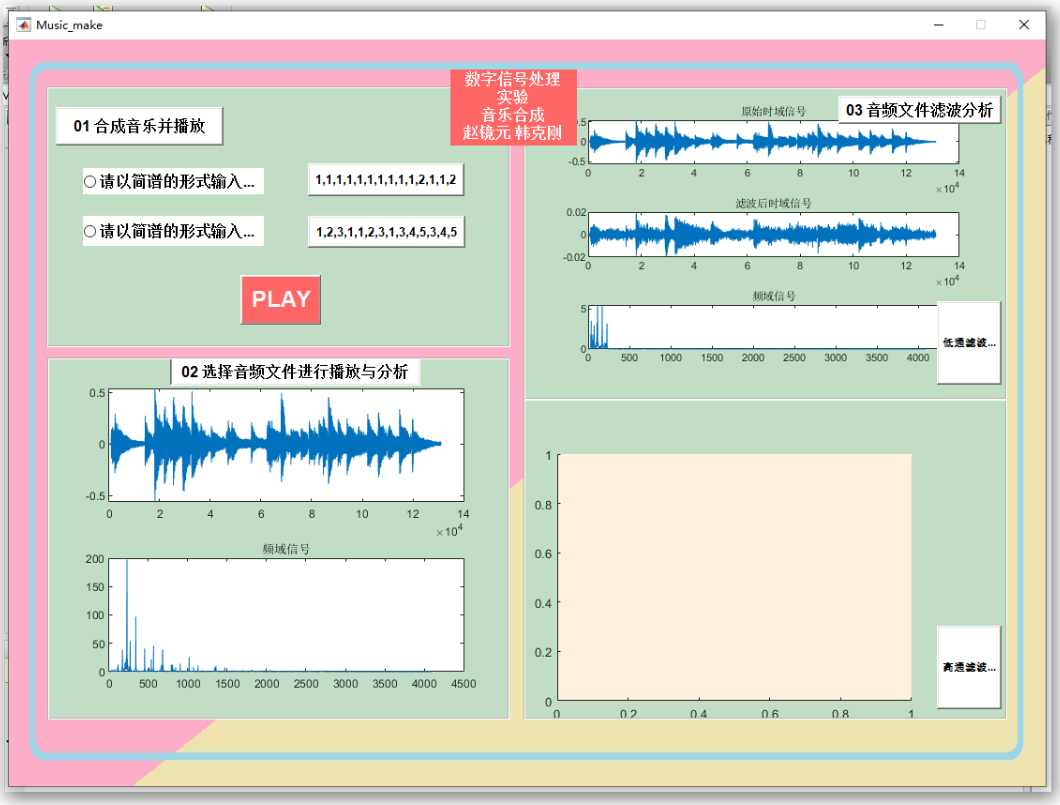

图 13 低通滤波波形图

图 13 为所选音频文件的低通滤波后的时域波形和频谱图，点击“低通滤波”按键即可让程序进行低通滤波分析，从而在右上角完成了波形绘制。滤波器选用了等波纹滤波器，相较原始数据，低频波形得到保留，高频波形被滤除，高频截止频率在 250 左右，符合实际情况。

因为本程序的低通截止频率选取相对过低，所以导致低音音乐输出的声音太小，在嘈杂的场所很难识别，所以在绘图后在程序内进行了音量放大，用 MATLAB 中的语句$result_t(:,1) = result_t(:,1) .* 50$;实现这样播放出的音频声音就很洪亮，也从中可以听出，低通滤波后的音频很雄厚，滤波效果十分明显。

图 14 为所选音频文件的高滤波后的时域波形和频谱图，点击“高通滤波按键”后会得到，滤波器选用了等波纹滤波器，相较原始数据，高频波形得到保留，低频波形被滤除，低频截止频率在 1750 左右。文件本身高音较少，所以滤波效果较为明显，符合实际情况。

图 14 高通滤后波形图

程序在绘制图形的过程中，也会伴随发出清脆的音乐，明显听出其符合高频滤波的特性，这也实现了高通滤波的功能。

通过以上运行结果可以看出，程序完整的实现了试验所要求的功能，运行正常，效果良好，从而顺利的完成了本次实验。

5. 总结

通过基于 MATLAB 的音乐分析与合成实验,我们了解了处理音频信号的基本操作。学会了简单的音乐合成，用傅里叶变换分析音乐和基于傅里叶级数的音乐合成。加深了对傅里叶变换的原理、方法所依据理论的理解，培养应用知识和独立思考的能力,提高分析问题和解决问题的能力，从而加深了我们对模拟信号数字处理化的理解。我们对 MATLAB 的基本使用提高了，学会设计简单的 GUI 界面，学会了怎样发现问题，怎样排错，独立设计仿真的能力得到了明显提升。

同时也要感谢老师的帮助和教导，实验的成功和老师的帮助是分不开的，我们在以后的学习中一定会更加的努力，使自己的水平的到不断地提高。

附录 1：程序

主程序：

```
    function varargout = Music_make(varargin)
% MUSIC_MAKE MATLAB code for Music_make.fig
%MUSIC_MAKE, by itself, creates a new MUSIC_MAKE or raises the existing
%      singleton*.
% H = MUSIC_MAKE returns the handle to a new MUSIC_MAKE or the handle to
%      the existing singleton*.
%      MUSIC_MAKE('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in MUSIC_MAKE.M with the given input arguments.
%
%      MUSIC_MAKE('Property','Value',...) creates a new MUSIC_MAKE or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Music_make_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Music_make_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 Music_make% Last Modified by GUIDE v2.5 16-Dec-
2019 20:09:17% Begin initialization code - DO NOT EDIT
    GUI_Singleton = 1;
    GUI_State = struct('GUI_Name',       mfilename, ...
                   'GUI_Singleton',  GUI_Singleton, ...
                   'GUI_OpeningFcn', @Music_make_OpeningFcn, ...
                   'GUI_OutputFcn',  @Music_make_OutputFcn, ...
                   'GUI_LayoutFcn',  [] , ...
                   'GUI_Callback',   []);
    if nargin && ischar(varargin{1})
       GUI_State.GUI_Callback = str2func(varargin{1});
    endif nargout
       [varargout{1:nargout}] = GUI_mainfcn(GUI_State, varargin{:});
```

else

GUI_mainfcn(GUI_State, varargin{:});

end

% End initialization code - DO NOT EDIT% --

- Executes just before Music_make is made visible.

function Music_make_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 Music_make (see VARARGIN)

ha=axes('units','normalized','position',[0 0 1 1]);

uistack(ha,'down')

II=imread('./resource/signal_logo.jpg');

image(II)

colormap gray

set(ha,'handlevisibility','off','visible','off');% Choose default command line output for Music_make

handles.output = hObject;% Update handles structure

GUIdata(hObject, handles);% UIWAIT makes Music_make wait for user response (see UIRESUME)

% uiwait(handles.figure1);% --- Outputs from this function are returned to the command line.

function varargout = Music_make_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;

function edit1_Callback(hObject, eventdata, handles)

% hObject   handle to edit1 (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles   structure with handles and user data (see GUIDATA)% Hints: get(hObject,'String') returns contents of edit1 as text

%   str2double(get(hObject,'String')) returns contents of edit1 as a double% --

```
- Executes during object creation, after setting all properties.
        function edit1_CreateFcn(hObject, eventdata, handles)
        % hObject    handle to edit1 (see GCBO)
        % eventdata  reserved - to be defined in a future version of MATLAB
        % handles    empty - handles not created until after all CreateFcns called% Hint: edit control
s usually have a white background on Windows.
        %       See ISPC and COMPUTER.
        if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColo
r'))
            set(hObject,'BackgroundColor','white');
        end
        function edit2_Callback(hObject, eventdata, handles)
        % hObject    handle to edit2 (see GCBO)
        % eventdata  reserved - to be defined in a future version of MATLAB
        % handles    structure with handles and user data (see GUIDATA)% Hints: get(hObject,'Strin
g') returns contents of edit2 as text
        %        str2double(get(hObject,'String')) returns contents of edit2 as a double% --
- Executes during object creation, after setting all properties.
        function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called% Hint: edit controls usua
lly have a white background on Windows.
%       See ISPC and COMPUTER.
        if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColo
r'))
            set(hObject,'BackgroundColor','white');
        end% --- Executes on button press in pushbutton1.
        function pushbutton1_Callback(hObject, eventdata, handles)
        % hObject    handle to pushbutton1 (see GCBO)
        % eventdata  reserved - to be defined in a future version of MATLAB
        % handles    structure with handles and user data (see GUIDATA)%tones = [1,2,3,1,1,2,3,1,3
,4,5,3,4,5];
        %len = [1,1,1,1,1,1,1,1,1,1,2,1,1,2];
        tones = str2num(get(handles.edit1,'string'));
```

```
len = str2num(get(handles.edit2,'string'));
playmusic(tones, len);
GUIdata(hObject, handles);
% --- Executes on key press with focus on pushbutton1 and none of its controls.
function pushbutton1_KeyPressFcn(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  structure with the following fields (see MATLAB.UI.CONTROL.UICONTRO
L)
%   Key: name of the key that was pressed, in lower case
%   Character: character interpretation of the key(s) that was pressed
%   Modifier: name(s) of the modifier key(s) (i.e., control, shift) pressed
% handles    structure with handles and user data (see GUIDATA)% --
- If Enable == 'on', executes on mouse press in 5 pixel border.
% --- Otherwise, executes on mouse press in 5 pixel border or over pushbutton1.
function pushbutton1_ButtonDownFcn(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes during object creation, after setting all properties.
function pushbutton1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called% --
- Executes during object deletion, before destroying properties.
function pushbutton1_DeleteFcn(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (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 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)
%clear sound
[FileName,PathName]=uigetfile('*','select the file');
```

```
file = fullfile(PathName,FileName);
%data = load(file);
%handles.data = data;
[audio,fs ]= audioread(file);
sound(audio,fs);NFFT = 2^nextpow2(fs);
audio_fft = fft(audio,NFFT);
audio_fft = audio_fft(1:ceil(length(audio_fft) / 2));
axes('parent',handles.uipanel4)
%axes(handles.axes9)
subplot(2,1,1);
plot(audio);
title('时域信号');
subplot(2,1,2);
plot(abs(audio_fft));
title('频域信号');
handles.data_audio = audio;
handles.data_audio_fft = audio_fft;
handles.data_fs = fs;
GUIdata(hObject,handles);
% --- 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)% dedigin fliter
%Fs= 8000; %Sampling Frequency
% Data vector
% d=fdesign.lowpass(); %lowpass filter specification object
% d.Fpass=1500/8000;
% d.Fstop=2000/8000;
% d.Astop=40;
% d
% designmethods(d)
fs = handles.data_fs;
d = fdesign.lowpass('Fp,Fst,Ap,Ast',1/fs,500/fs,1,60);
% Invoke Butterworth design method
```

```
fliter_low =design(d,'equiripple');
% fvtool(fliter_low)
data_t = handles.data_audio;result_t = filter(fliter_low,data_t);
NFFT = 2^nextpow2(fs);
result_f=fft(result_t,NFFT);
result_f = result_f(1:ceil(length(result_f) / 2));
audio=handles.data_audio ;
axes('parent',handles.uipanel8)
%axes(handles.axes10)subplot(3,1,1);
plot(audio);
title('原始时域信号');
subplot(3,1,2);
plot(result_t);
title('滤波后时域信号');
subplot(3,1,3);
plot(abs(result_f));
title('频域信号');
result_t(:,1) = result_t(:,1).*50;
sound(result_t,fs)
% --- 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)
fs = handles.data_fs;
d = fdesign.highpass('Fst,Fp,Ast,Ap',3000/fs,5000/fs,60,1);
fliter_high = design(d,'equiripple');
data_t = handles.data_audio;
fs = handles.data_fs;
result_t = filter(fliter_high,data_t);
NFFT = 2^nextpow2(fs);
result_f=fft(result_t,NFFT);
result_f = result_f(1:ceil(length(result_f) / 2));
audio=handles.data_audio ;
%axes(handles.axes11)
```

```
axes('parent',handles.uipanel5)subplot(3,1,1);
plot(audio);
title('原始时域信号');
subplot(3,1,2);
plot(result_t);
title('时域信号');
subplot(3,1,3);
plot(abs(result_f));
title('频域信号');
sound(result_t,fs)% return an array of volume strength
function volume_array = generate_volume_for3(len_divide_speed,sample_rate)
    unit = sample_rate * len_divide_speed;
    x1 = linspace(0,len_divide_speed/6,unit/18);
    x2 = linspace(0,len_divide_speed/6,unit/36);
    x3 = linspace(1,1,unit/9);
    x4 = linspace(0,len_divide_speed/3,unit-length([x1,x2,x3]));
    volume_array = [6/len_divide_speed*x1, 1-1.2/len_divide_speed*x2,...
        0.8*x3, 0.8*exp(-(100-90*len_divide_speed)*x4)];
    % plot(volume_array);
function playmusic(tones, len)
    load GUItar.mat
    speed = 2;
    sample_rate = 8000;
    len = len / speed;
    % Volume
    volume_array =[];
    for i=1:length(len)
      volume_array = [volume_array, generate_volume_for3(len(i),sample_rate)];
    end
    %      F(1), G(2), A(3),  B-(4),  C(5),   D(6),   E(7)
    f = [349.23, 392,  440,  466.16, 523.25, 587.33, 659.25];
    tone = f(tones);
    % Generate Harmonic Sin Signal
    y = [];
    for i = 1:length(tone)
```

```
    t = linspace(0,len(i),len(i)*sample_rate);

    [val, index] = min( abs(tone(i) - base) );
    y = [y, [1, two_standard(index), three_standard(index), four_standard(index)] * ...
      [sin(2*pi*tone(i)*t); sin(2*pi*2*tone(i)*t);...
      sin(2*pi*3*tone(i)*t); sin(2*pi*4*tone(i)*t)]];
  end
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  % y suppressed by volume
  y = y .* volume_array;
  % Make sound
  sound(y, sample_rate);
```

数据生成函数

```
function [base, two_standard, three_standard, four_standard] = Analyze_fmt
    close all;
%       load GUItar.MAT;
    music = audioread('./resource/fmt.wav');
    start_time =[700,   2300 ,14000, 18000, 22000, 25000, 29000,...
        32000, 36000, 40000, 46000, 48000, 56000, 62000, 68000,...
        72000, 76000, 79000, 81000, 83000, 84500, 86500, 90000,...
        94000, 98000,   102000, 106000, 110000, 114500, 119000];
    end_time =   [2300, 14000,18000, 22000, 25000, 29000, 32000,...
        36000, 40000, 46000, 48000, 56000, 62000, 68000, 72000,...
        76000, 79000, 81000, 83000, 84500, 86500, 90000, 94000,...
        98000, 102000, 106000, 110000, 114500, 119000, 131000];
    len = [];
    base = [];
    one_amp = [];
    two_amp = [];
    three_amp = [];
    four_amp = [];
    tone = {};

    for i = 1:length(start_time)
```

```
            [base_uut, one_amp_uut,two_amp_uut,three_amp_uut,four_amp_uut ,tone_uut]
= ...
            Freq_Analyze( music(start_time(i):end_time(i)), 6, 9);

            leng = ( end_time(i) - start_time(i) )*2 / 4000;
            leng = round(leng) / 2;

            len(i,1) = leng;
            base(i,1) = base_uut;
            one_amp(i,1) = one_amp_uut;
            two_amp(i,1) = two_amp_uut;
            three_amp(i,1) = three_amp_uut;
            four_amp(i,1) = four_amp_uut;
            tone{i,1} = (tone_uut);
        end

        two_standard = two_amp./one_amp;
        three_standard = three_amp./one_amp;
        four_standard = four_amp./one_amp;

        report = table(base, len, two_standard,...
            three_standard, four_standard, tone,...
            'VariableNames', {'Base' 'length' 'two_amp' 'three_amp' 'four_amp' 'Tone'})
        save('GUItar','base','two_standard', 'three_standard', 'four_standard');

    end
```

“信号分析与处理综合实验”评分表

_YUANFEI___同学：

1. 实验过程评价（占 30%）

☐优秀　　☐良好　　☐中等　　☐及格　　☐不及格

2. 实验结果（占 30%）

☐优秀　　☐良好　　☐中等　　☐及格　　☐不及格

3. 报告撰写（占 40%）

☐优秀　　☐良好　　☐中等　　☐及格　　☐不及格

总评成绩： ____________

指导教师： ____________