大连理工大学

本科实验报告二

课程名称：随机信号分析实验

学院（系）：电子信息与电气工程学部

专　　业：电子信息工程

班　　级：电信 1806 班

学　　号：201871080

学生姓名：刘祎铭

2020 年 11 月 17 日

大连理工大学实验报告

学院（系）：电子信息与电器工程学部 专业：电子信息工程 班级：电信 1806

姓　　名：刘祎铭 学号：201871080 组：

实验时间：2020.11.11 实验室：创新园 C220 实验台：

指导教师：李小兵

实验 II：系统对随机信号响应的统计特性分析、功率谱分析及应用实验

一、 实验目的和要求

掌握直接法估计随机信号功率谱的原理和实现方法；掌握间接法估计随机信号功率谱的原理和实现方法；掌握系统对随机信号响应的统计特性分析及仿真实现方法。熟悉 MATLAB 信号处理软件包的使用。

二、 实验原理和内容

（一）实验原理

1. 直接法估计随机信号功率谱原理

直接法又称为周期图法，它是把随机信号 x(n)的 N 点观察数据 xN(n)视为一能量有限信号，直接取 xN(n)的傅里叶变换，得到 XN(ejω)，然后取其模值的平方，并除以 N，作为对 x(n)真实的功率谱 P(ejω)的估计。工程上，常使用离散 Fourier 变换（DFT，编程上使用其快速算法 FFT），即 $P_X(k)=\frac{1}{N}|X_N(k)|^2$，进行计算。

2. 间接法估计随机信号功率谱

间接法的理论基础是 Wiener-Khintchine 定理，具体的实现方法是先由 xN(n)估计出自相关函数 r`(m)，然后对 r`(m)求傅里叶变换得到 xN(n)的功率谱，记之为 XN(ejω)，并以此作为对真实功率谱 P(ejω)的估计。工程上，常使用离散 Fourier 变换（DFT，编程上使用其快速算法 FFT），即：

$$P_X(k)=\sum_{m=-M}^{M}\hat{r}(m)e^{-j\frac{2\pi km}{2M+1}},\quad |M|\le N-1$$

进行计算。因为由这种方法求出的功率谱是通过自相关函数间接得到的，所以又称为间接法或 Blackman-Tuckey(BT)法，该方法是 FFT 出现之前常用的谱估计方法。

3. 时域中系统对随机信号响应的统计特性分析及仿真

根据系统卷积性质，计算系统输出信号的统计特性。有如下性质：

$$m_Y = m_X \sum_n h(n)$$

$$R_Y(m) = \sum_j \sum_k R_X(m+j-k)h(j)h(k)$$

4. 频域中系统对随机信号响应的统计特性分析及仿真

根据卷积定理，输入、输出信号功率谱的关系为:

$$R_Y(e^{j\omega}) = R_X(e^{j\omega})|H(e^{j\omega})|^2$$

在计算系统输 出信号功率谱时，如果在时域时计算困难，可以按照上式在频域计算。

（二）实验内容：

1. 直接法估计随机信号功率谱

(1) 生成 1024 点数据的随机信号

$$X(n) = 2\cos(2\pi f_1 t + \varphi_1) + 5\cos(2\pi f_2 t + \varphi_2) + N(n)$$

其中$f_1 = 30Hz$，$f_2 = 100Hz$，φ_1，φ_2为在$[0,2\pi]$内的均匀分布的随机变量，$N(n)$是数学期望为 0，方差为 1 的高斯白噪声。

(2) 用周期图法计算$X(n)$的功率谱，并绘图。

(3) 用 MATLAB 函数 periodogram 重新计算$X(n)$的功率谱，并与(2)做比较。

2．间接法估计随机信号功率谱

(1) 计算以上$X(n)$的自相关函数。

(2) 通过计算自相关函数的 Fourier 变换，求$X(n)$的功率谱并绘图。

(3) 利用 MATLAB 函数 psd、pwelch 重新计算$X(n)$的功率谱，并与(2)做比较。

3．系统对随机信号响应的统计特性分析及仿真

(1) 生成含 500 点数据的高斯分布白噪声随机信号$X(n)$。

(2) 设计一个带通系统$H(e^{j\omega})$，其上、下截止频率分别为 4KHz 和 3KHz.

(3) 计算$X(n)$通过以上带通滤波器的自相关函数和功率谱密度。

三、 主要仪器设备

Windows10(64 位)，MATLAB2016a

四、 实验步骤与操作方法

1. 直接法估计随机信号功率谱

(1) 准备实验环境。

(2) 生成随机信号样本。

```
%两个带随机相位的单谱信号与白噪声之和
N=1024;fs=1000;                          %序列长度和采样频率
t=(0:N-1)/fs;                            %时间序列
fai=random('unif',0,2*pi,1,2);           %产生 2 个[0, 2pi]内均匀随机数
xn=cos(2*pi*30*t+fai(1))+3*cos(2*pi*100*t+fai(2))+randn(1,N); %产生含噪声的随机序列
```

(3) 显示随机信号时域波形。

```
figure,plot(xn);
title('随机信号时域波形')
```

(4) 周期图法直接估计随机信号功率谱，通过 Fourier 变换取上述高斯白噪声 1024 点计算对应的功率谱并绘图。

```
%对样本进行傅里叶变换，取模平方时间平均
Sx1=abs(fft(xn)).^2/N;                   %估计功率谱
f=(0:N/2-1)*fs/N;                        %频率轴坐标
figure
subplot(211);
plot(f,10*log10(Sx1(1:N/2)));grid on; %用 dB/Hz 做功率谱单位，画图
xlabel('f （Hz）');
ylabel('Sx1(f)(dB/Hz)');
title('周期图法估计功率谱');
```

(5) 与内置函数 periodogram 对比。

```
Sx2=periodogram(xn); %用 periodogram 函数生成功率谱
subplot(212);
plot(f,10*log10(Sx2(1:N/2)));grid on; %用 dB/Hz 做功率谱单位，画图
xlabel('f(Hz)');
ylabel('Sx2(f)(dB/Hz)');
```

```
title('periodogram 函数估计功率谱');
```

(6) 尝试改变信号、噪声的幅度和频率观察效果

使用 GUI 设计幅值、频率、相位可实时调整的随机信号，使用各方法估计功率谱

2. 间接法估计随机信号功率谱

(1) 准备环境。

(2) 生成随机信号样本。

```
%两个带随机相位的单频谱信号与白噪声之和
N=1024;fs=1000;                                    %序列长度和采样频率
t=(0:N-1)/fs;                                      %时间序列
fai=random('unif',0,2*pi,1,2);                     %产生 2 个[0, 2pi]内均匀随机数
xn=cos(2*pi*30*t+fai(1))+3*cos(2*pi*100*t+fai(2))+randn(1,N);  %产生含噪声的随机序列
```

(3) 显示随机信号时域波形。

```
figure,plot(xn);grid on
title('随机信号时域波形')
```

(4) 计算高斯白噪声的自相关函数。

```
Rxx=xcorr(xn,'biased');                            %估计自相关函数 Rxx
```

(5) 间接法估计随机信号的功率谱

```
Rxx=xcorr(xn,'biased');                            %估计自相关函数 Rxx
Sx1=abs(fft(Rxx));                                 %对 Rxx 进行 FFT 得到功率谱
f=(0:N-1)*fs/N/2;                                  %频率轴坐标
figure;plot(f,10*log10(Sx1(1:N)));grid on;         %用 dB/Hz 做功率谱单位，画图
xlabel('f(Hz)');
ylabel('Sx(f)(dB/Hz)');
title('自相关函数法估计功率谱');
```

（6）与内置函数对比。

1.使用 MATLAB 内置的 psd 函数生成功率谱并比较

```
Nseg=256;                                          %分段间隔为 256
window=hanning(Nseg);                              %汉宁窗
noverlap=Nseg/2;                                   %重叠点数为 128
f=(0:Nseg/2)*fs/Nseg;                              %频率轴坐标
Sx2=psd(xn,Nseg,fs,window,noverlap,'none');  %psd 函数估计功率谱
figure;plot(f,10*log10(Sx2));grid on;
xlabel('f(Hz)');
ylabel('Sx(f)(dB/Hz)');
title('Bartlett 法估计功率谱');
```

2.使用MATLAB内置的pwelch函数生成功率谱并比较

```
Sx3=pwelch(xn,window,128,Nseg,fs,'onesided')*fs/2; %Welch 函数估计功率谱
figure;plot(f,10*log10(Sx3));grid on;              %用 dB/Hz 做功率谱单位，画图
xlabel('f(Hz)');
ylabel('Sx(f)(dB/Hz)');
title('Welch 法估计功率谱');
```

(7) 比较三者误差。

```
Sx11=Sx1';

err1=Sx11(1:16:N*2-1)-Sx2(1:N/8);        % 间接法谱估计与 psd 函数估计的误差

ff=f(1:length(f)-1);

figure;subplot(311);plot(ff,10*log10(err1));

xlabel('f(Hz)');
ylabel('Sx1(f)-Sx2(f) (dB/Hz)');
title('间接法谱估计与 psd 函数估计的误差');
err2=Sx11(1:16:N*2-1)-Sx3(1:N/8);          % 间接法谱估计与 pwelch 函数估计的误差
subplot(312);plot(ff,10*log10(err2));
xlabel('f(Hz)');
ylabel('Sx1(f)-Sx3(f) (dB/Hz)');
title('间接法谱估计与 pwelch 函数估计的误差');
err3=Sx2-Sx3;                    % psd 函数估计与 pwelch 函数估计的误差
subplot(313);plot(f,10*log10(err3));
xlabel('f(Hz)');
ylabel('Sx2(f)-Sx3(f) (dB/Hz)');
title('psd 函数估计与 pwelch 函数估计的误差');
```

（8）尝试改变信号、噪声的幅度和频率观察效果

使用 GUI 设计幅值、频率、相位可实时调整的随机信号，使用各方法估计功率谱。

3. 系统对随机信号响应的统计特性分析及仿真

(1) 准备环境。

(2) 生成高斯随机信号样本。

```
N=500;                                    %样本长度 N=500，对应时长 25ms
xt=random('norm',0,1,1,N);                  %产生 1*N 个高斯随机数
% 显示时域波形
figure,plot(xt);
title('随机信号时域波形')
```

(3)设计一个带通滤波器，截止频率为3KHz，4KHz，显示其时域频域波形和频域波形

```
%冲激响应
ht=fir1(101,[0.3 0.4]);                   % 101 阶带通滤波器，数字截止频率为 0.3 和 0.4
% 显示冲激响应函数 ht
figure,plot(ht)
```

```
title('冲激响应函数 ht')
HW=fft(ht,2*N);                          %2N 点滤波器频率响应（系统传输函数）
% 显示传递函数 HW
figure,plot((1:N)/N,abs(HW(1:N)));       %频域波形输出
title('传递函数 HW')
```

(4)计算输出信号的自相关函数、功率谱等统计量。

```
Rxx=xcorr(xt,'biased');                  %直接法估计白噪声的自相关函数
% 显示自相关函数 Rxx
figure,stem(Rxx)
title('输入自相关函数 Rxx')
```

```
Sxx=abs(fft(xt,2*N).^2)/(2*N);           %周期图法估计白噪声的功率谱
```

```
HW2=abs(HW).^2;                          %系统的功率传输函数
Syy=Sxx.*HW2;                            %输出信号的功率谱
```

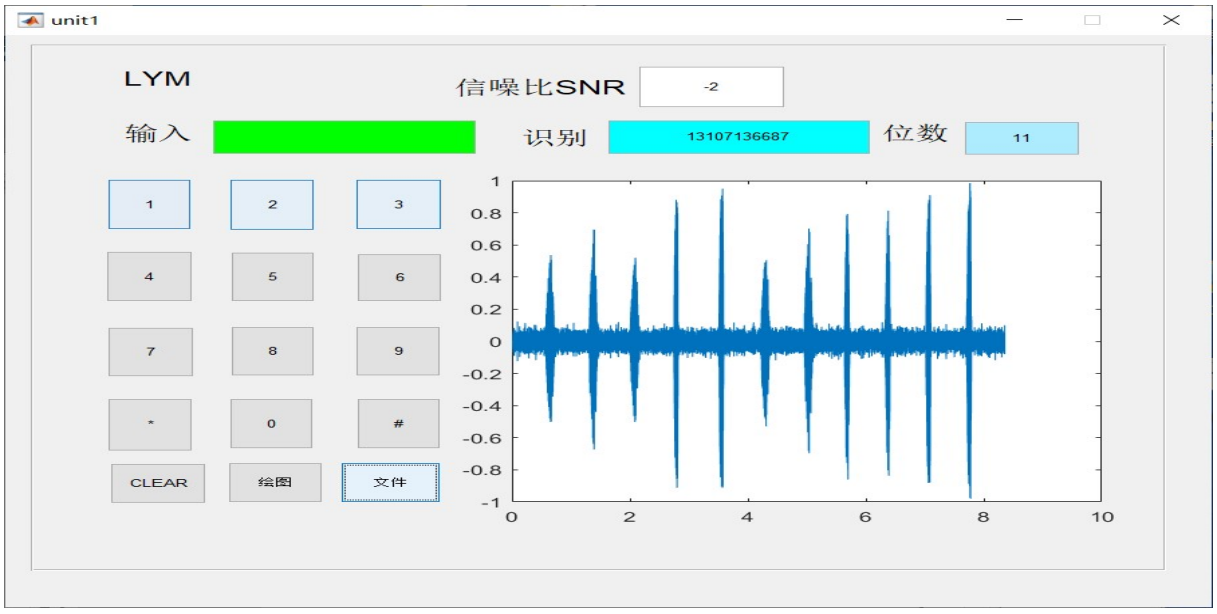
```
Ryy=fftshift(ifft(Syy));                 %用 IFFT 求输出信号的自相关函
数                                        %函数 fftshift 对数组进行移位
%Ryy=ifft(Syy);
```

(5)图形展示系统输出观察效果。

```
w=(1:N)/N;                               %功率谱密度横轴坐标
t=(-N:N-1)/N*(N/20000);                  %自相关函数横轴坐标
subplot(4,1,1);plot(w,abs(Sxx(1:N)));    %输入信号功率谱密度
xlabel('归一化频率 f');ylabel('Sxx(f)');title('输入信号功率谱密度');
subplot(4,1,2);plot(w,abs(HW2(1:N)));    %系统的功率传输函数
xlabel('归一化频率 f');ylabel('H2(f)');title('系统的功率传输函数');
subplot(4,1,3);plot(w,abs(Syy(1:N)));    %输出信号的功率谱密度
xlabel('归一化频率 f');ylabel('Syy(f)');title('输出信号的功率谱密度');
subplot(4,1,4);plot(t,Ryy);              %输出信号的自相关函数
xlabel('归一化频率 f');ylabel('Ryy(f)');title('输出信号的自相关函数');
```

4. 探究式实验内容 DTMF 电话号码识别

1. 利用 GUI 设计电话音号码识别界面



2. 将电话音信号进行分割，获取每一个拨号号码对应的信号

```
n=max(sampledata(1:length(sampledata)))%获得音频数据峰值
count1=0;%初始化
i=1;%初始化
numcount1=[];%初始化
while(i<=length(sampledata)-2640/48000*Fs)%最后一段长2640，如果按键发音，大小为2640的样本长度正好容纳，对之外的每一个点进行讨论
    for j=i:ceil(i+2640/48000*Fs)%对于采样率48000,用48000合适,但48000*Fs适用于各种音频。%对讨论是否认为是噪声并需要置0的点的后面的2640个点进行讨论
        if(abs(sampledata(j))>0.3*n)%我们认为峰值的0.39是区分按键音和环境噪音的标志，这个是由峰值1.79的取高度0.7作为分界点，这个是由图像特点，就是噪音和按键音幅度有显著差异决定的
            count1=count1+1;%计数
        end
    end
    if(count1>550/48000*Fs)%统计有多少个大于0.3的点，用550作为分界点进行保留，不是算出来的，而是使用matlab试出来的。不写count1的‘；’很容易区分出噪声含有的count1数目和按键音数目

numcount1=[numcount1,sampledata(i:i+ceil(4799/48000*Fs))'];%！！！！！！！！正常录音不需转置，但老师的测试音频需要！！！！！！%把按键音合并起来，该点和该点之后4800点，即100ms，会丢掉一部分按键音尾端数据，设计中也会舍弃很少一些首端数据，不过数量少，而且按键音中间的声音更具备特征，信号质量更好。
        i=ceil(i+5000/48000*Fs);%取完按键音后跳过该段，不重复截取声音片段。
%       else
%         i=ceil(i+600/48000*Fs)-count1;
    end
    i=i+1;%计数
```

```
            count1=0;%复位
    end
    t=0:1/Fs:(length(sampledata)-1)/Fs;%输出原始波形的时间转换
    plot(t,sampledata);%输出原始波形
    numlength=length(numcount1);%接收到的信号长度
    count=numlength/(1+ceil(4799/48000*Fs));%计算出发了多少个数字
set(handles.receivescreen,'string','');%清屏
```

更加简单但效果更好的基于频域的电话音信号分割

```
L=length(sampledata);
while(i<L)
    if(i<L-Fs/20)
    xh=sampledata(i:i+Fs/20);
    else
    xh=sampledata(i:L);
    end
    f=fft(xh,Fs);
    b=abs(f);
    p=b.*b/(Fs);
if(max(p(1:1000))>1.2&&max(p(1000:1700))>1.2)
numcount1=[numcount1,sampledata(i:i+4409)'];
i=i+Fs/12;
    else
i=i+Fs/200;
    end
end
```

3. 判断每个号码信号对应的电话号码

```
    for i=1:count%一共count个数字

d=numcount1(1+(1+ceil(4799/48000*Fs))*(i-1):(1+ceil(4799/48000*Fs))*i);%取每位拨号音的采样点
        f=fft(d,Fs); % 以N=2048 作FFT 变换d是取出来每位拨号音的采样点
        a=abs(f);
        p=a.*a/Fs; % 计算功率谱
        num(1)=find(p(1:1000)==max(p(1:1000))); % 找行频
        num(2)=1000+find(p(1000:1700)==max(p(1000:1700))); % 找列频
        if (num(1) < 730)
            row=1; % 确定行数
        elseif (num(1) < 810)
            row=2;
        elseif (num(1) < 900)
            row=3;
        else
            row=4;
```

```
        end
        if (num(2) < 1260)
            coloum=1; % 确定列数
        elseif (num(2) < 1395)
            coloum=2;
        elseif (num(2) < 1550)
            coloum=3;
        else coloum=4;
        end
        if row==4&&coloum==2%计算出对应的数字值
            num=0;
        elseif(row==4&&coloum==1)
            num='*';
        elseif(row==4&&coloum==3)
            num='#';
        elseif(coloum==4)
            if(row==1)
             num='A';
            elseif(row==2)
             num='B';
             elseif(row==2)
             num='C';
            else
             num='D';
            end
        else
            num=coloum+3*(row-1); %根据行列关系计算出对应的数字值
        end
        strnum=strcat(get(handles.receivescreen, 'string'),num2str(num));
        set(handles.receivescreen,'string',strnum);%将结果显示在屏幕上
    end
        set(handles.edit4,'string',count);%将位数显示在屏幕上
    end
```

4. 添加信噪比可改变的噪声信号

```
    [filename,filepath]=uigetfile('.wav','选择音频文件');%打开文件
    if(filename==0)
        return%判断文件是否为空
    else
    audeofile=strcat(filepath,filename);%形成音频文件路径
    [sampledata,Fs]=audioread(audeofile);%从音频文件路径中读取音频文件，获得采样率和音频数据
    SNR=get(handles.edit5,'String');
   sampledata=awgn(sampledata,str2num(SNR),'measured');
```

5.设计滤波器，对噪声信号进行滤波

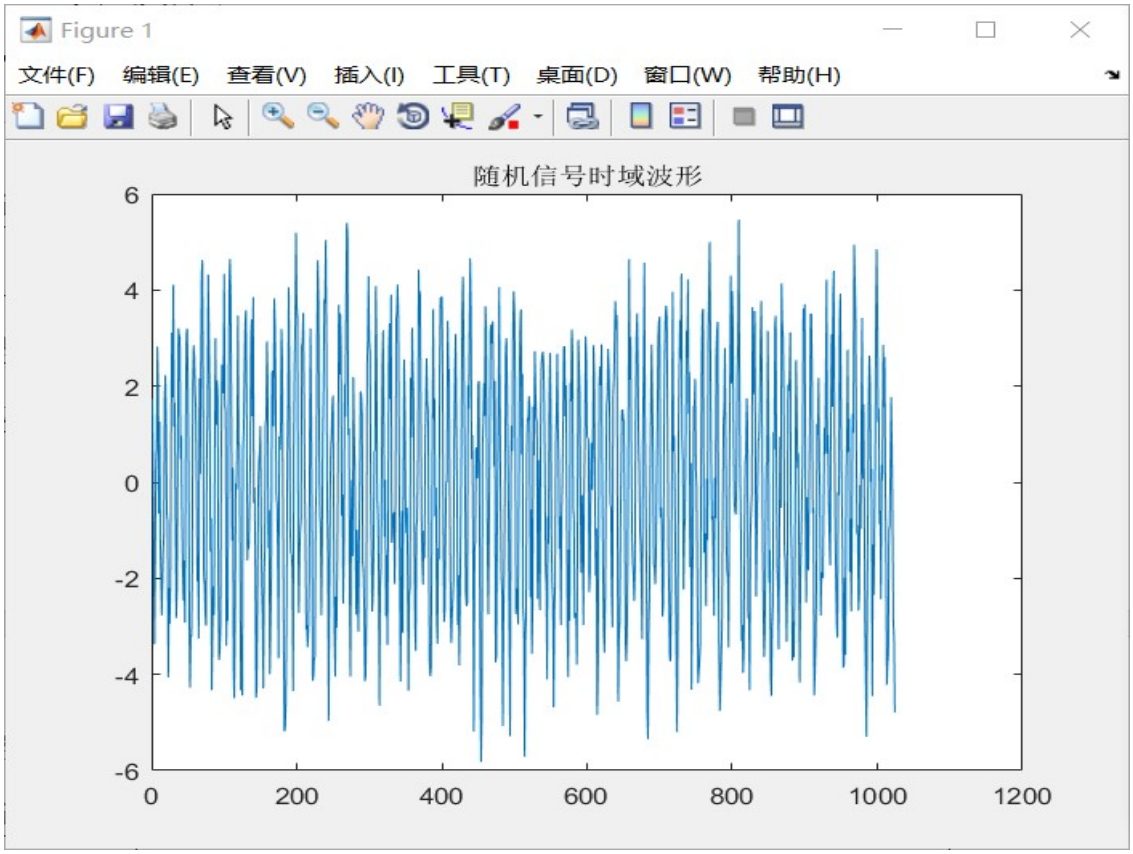
```
Hd=band_pass(600,690,1710,1800);                                    %带通滤波器
sampledata=filter(Hd,sampledata);


function Hd = band_pass(Fstop1,Fpass1,Fpass2,Fstop2);
%BAND_PASS Returns a discrete-time filter object.
% MATLAB Code
% Butterworth Bandpass filter designed using FDESIGN.BANDPASS.
% Generated by MATLAB(R) 9.0 and the Signal Processing Toolbox 7.2.
% Generated on: 18-Dec-2019 14:51:03
% All frequency values are in Hz.
Fs = 44100;  % Sampling Frequency
% Fstop1 = 500;              % First Stopband Frequency
% Fpass1 = 690;              % First Passband Frequency
% Fpass2 = 1710;             % Second Passband Frequency
% Fstop2 = 1800;             % Second Stopband Frequency
Astop1 = 70;               % First Stopband Attenuation (dB)
Apass  = 1;                % Passband Ripple (dB)
Astop2 = 70;               % Second Stopband Attenuation (dB)
match  = 'passband';  % Band to match exactly
% Construct an FDESIGN object and call its BUTTER method.
h  = fdesign.bandpass(Fstop1, Fpass1, Fpass2, Fstop2, Astop1, Apass, ...
                      Astop2, Fs);
Hd = design(h, 'butter', 'MatchExactly', match);
```

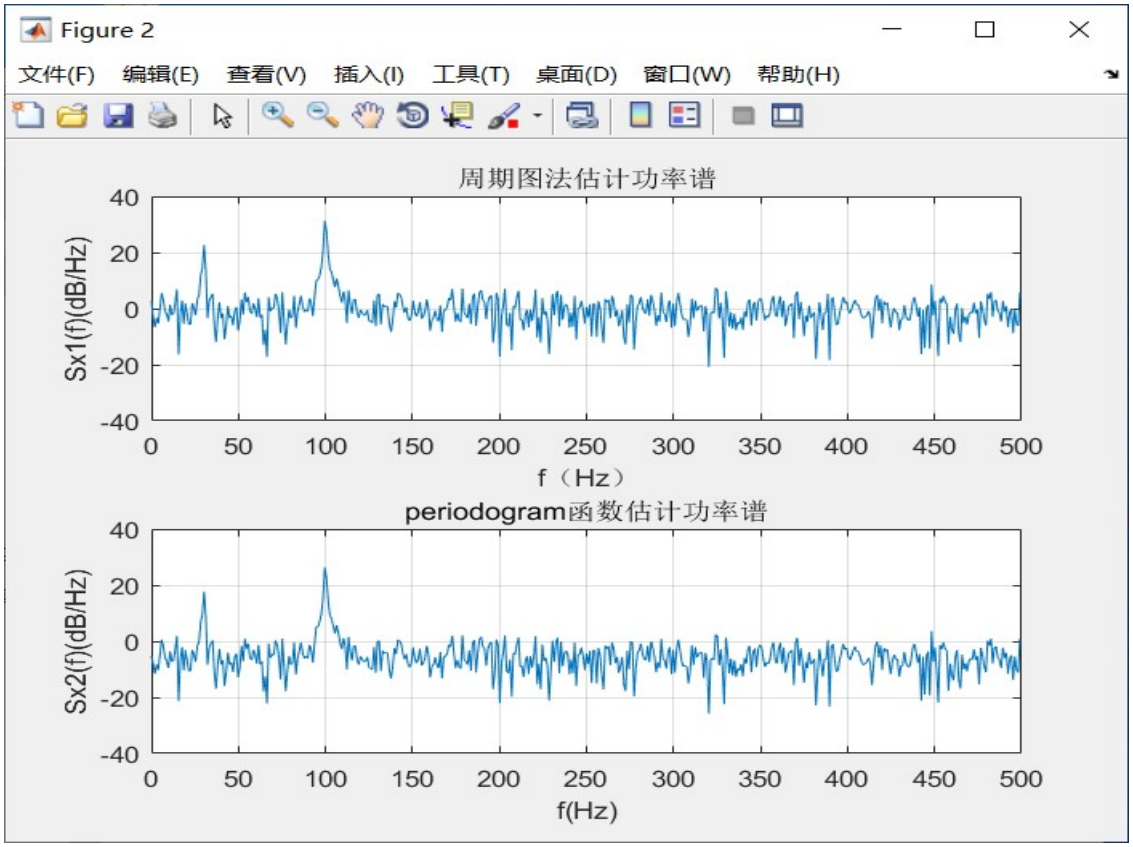
五、 实验数据记录和处理

实验一：直接法估计随机信号功率谱

1. 随机信号样本时域波形图



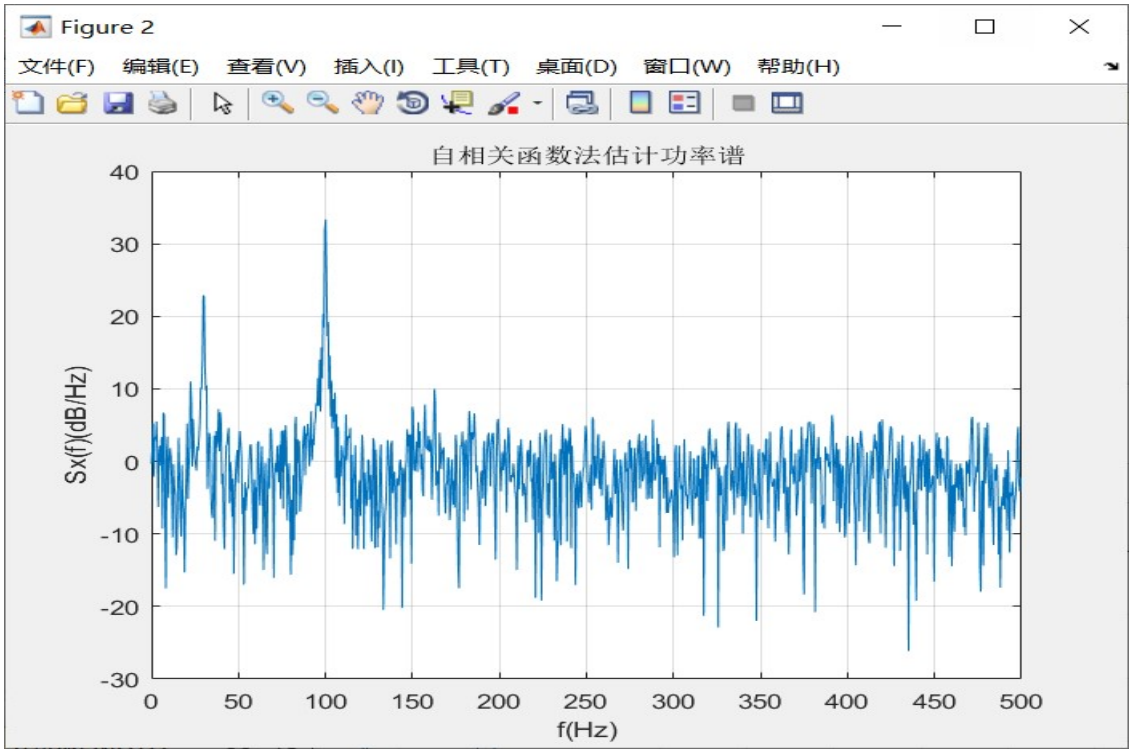
2.周期法直接生成功率谱与 MATLAB 内置的 periodogram 函数生成功率谱进行比较



如图所示首先用周期图法生成了功率谱，然后又调用函数 periodogram 函数生成了功率谱观察二者，发现区别不大。这些估计方法的相同点是都有两个峰值，分别在两个余弦函数的频率处，并且其余地方均有高斯噪声产生的波动。但发现两个问题，周期图法生成的功率谱一是毛刺太多，二是叠加的白噪声部分不是常数功率谱。这是由于 periodogram 默认使用海明窗，所以有细微差别，但高度有明显不同。为 periodogram 指定窗口类型为矩形窗后图形几乎一样，但仍然是高度不一样，发现差了一个倍数（大概 2 到 4 倍之间）。上图是经过调整后的频谱图。

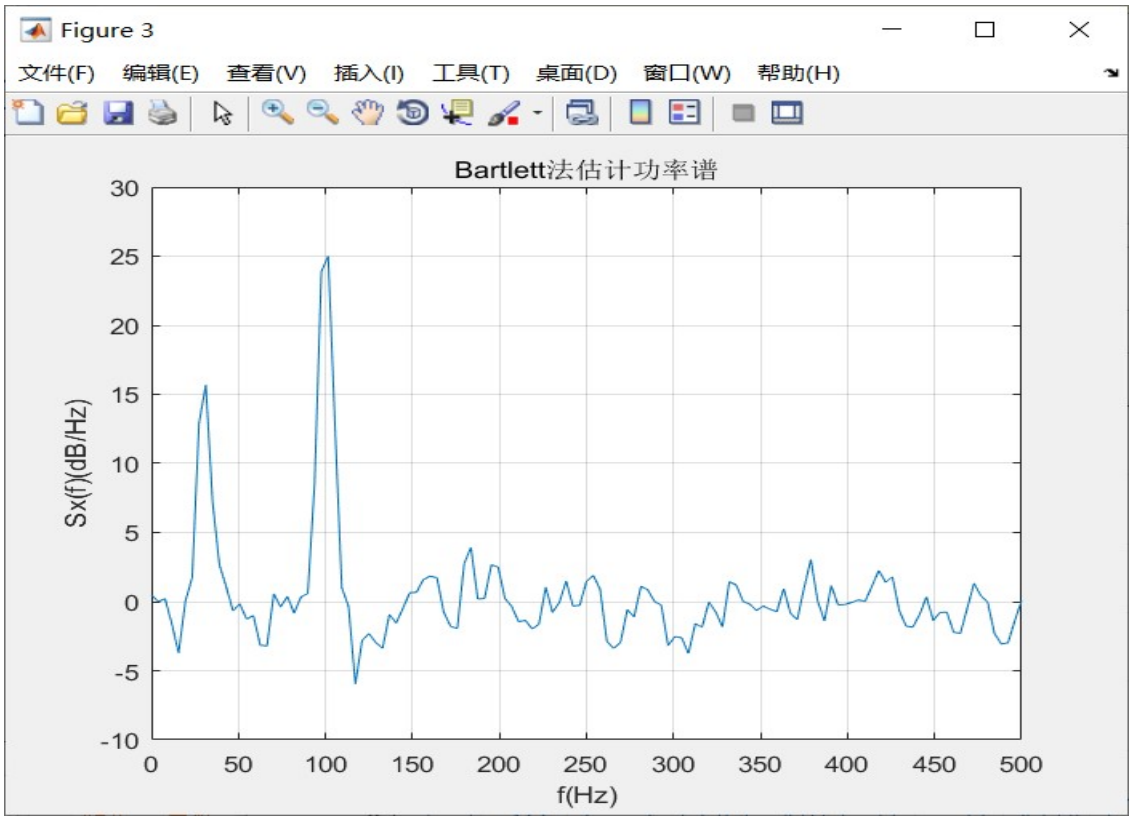
实验二：间接法估计随机信号功率谱

1. 自相关函数法间接估计随机信号的功率谱



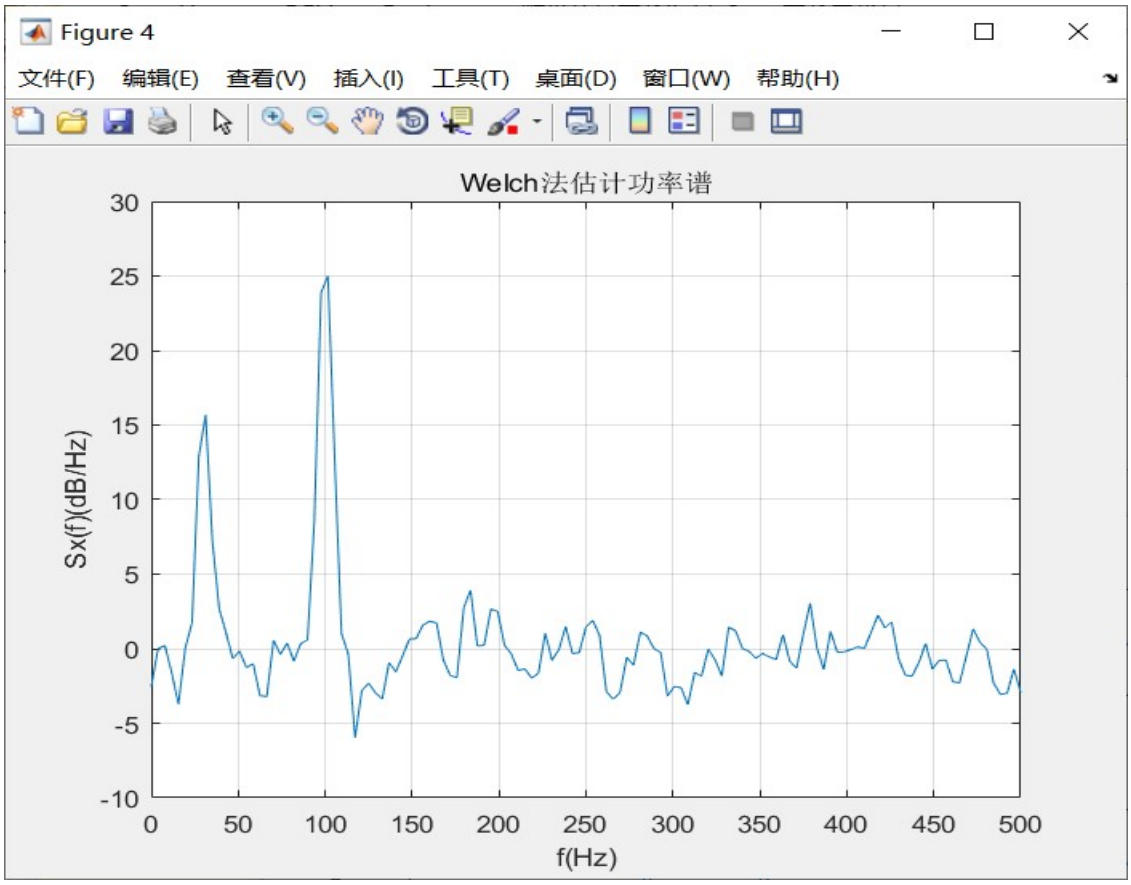
直接法和间接法的缺陷一是功率谱估计方差不随着频谱样本长度 N 的增加而减小趋于零，二是有泄漏问题：将实际频谱展宽，导致功率谱估计分辨率下降，弱信号的主瓣被强信号的副瓣淹没。

2. MATLAB 内置函数 psd 法估计功率谱



采用分段周期图法（bartlett 法）用 psd 函数实现：首先将信号采样数据分成数据量相等的极端，对每段数据采用周期图法估计功率谱；接着对这些功率谱取平均，得到 bartlett 估计功率谱。但是该方法的缺点是分段数越多，每段数据量就越少，要结果更精准，就需要人为平衡好分段数与数据量的关系。

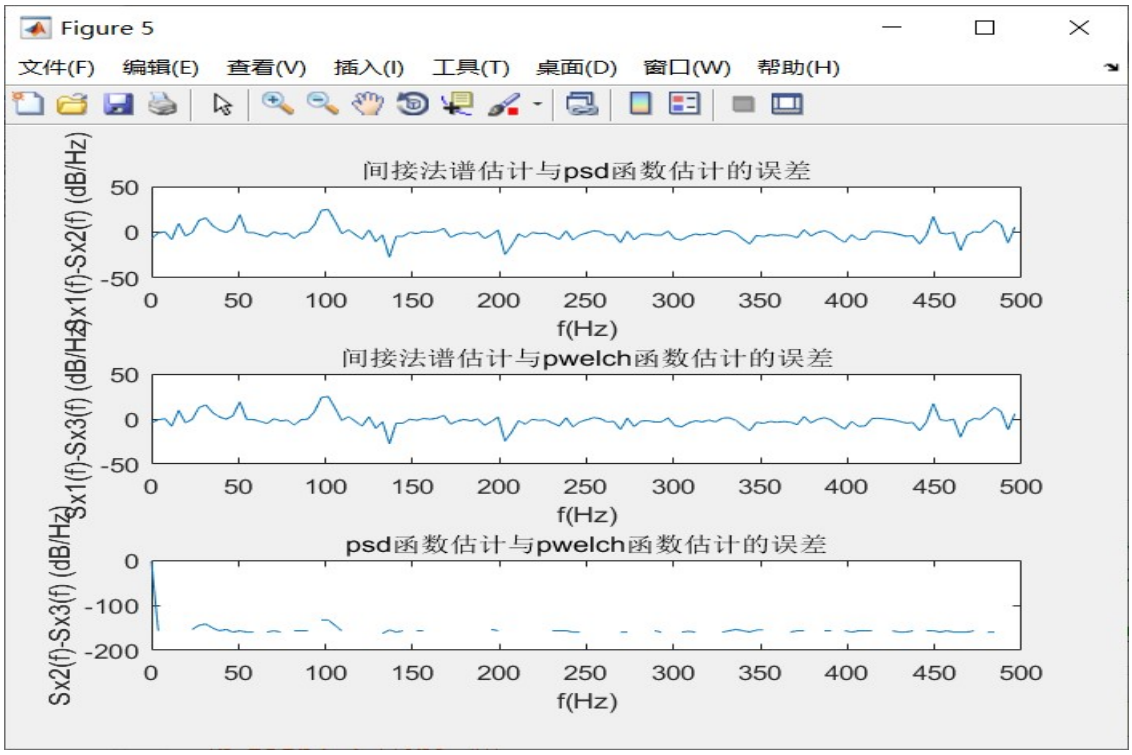
3. MATLAB内置函数Welch法估计功率谱



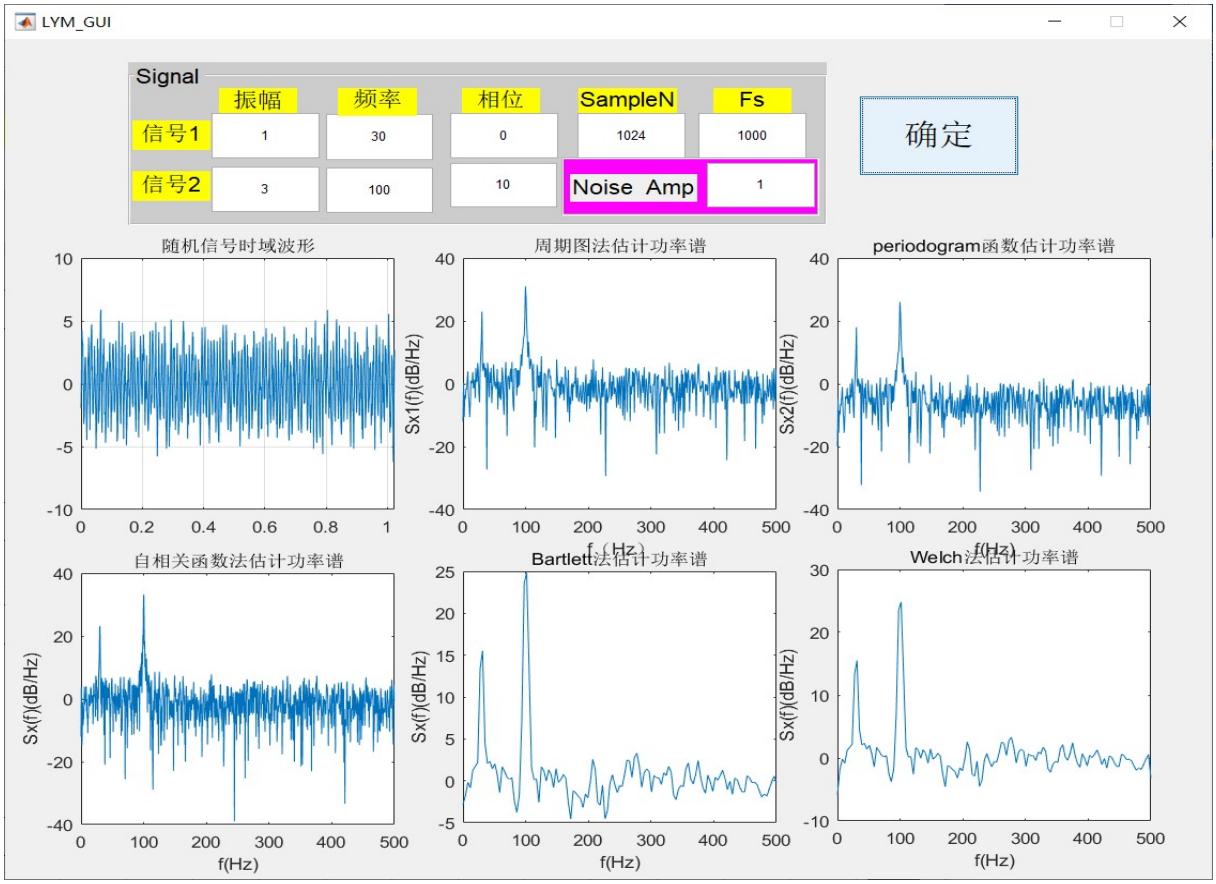
Welch 法汉宁窗平滑，对分段周期法进行改进，它允许分段的数据之间有重叠。

可以看出，welch 法和 bartlett 法的曲线平滑了很多，比较明显的是尖端高度降低了，是因为分段之后真正参与估计功率谱的数据长度减小了，减小为：N/分段数，导致分辨率降低。并且使用不同的窗函数也会影响分辨率，比如非矩形窗口会加宽主瓣，导致分辨率降低。

4.误差估计

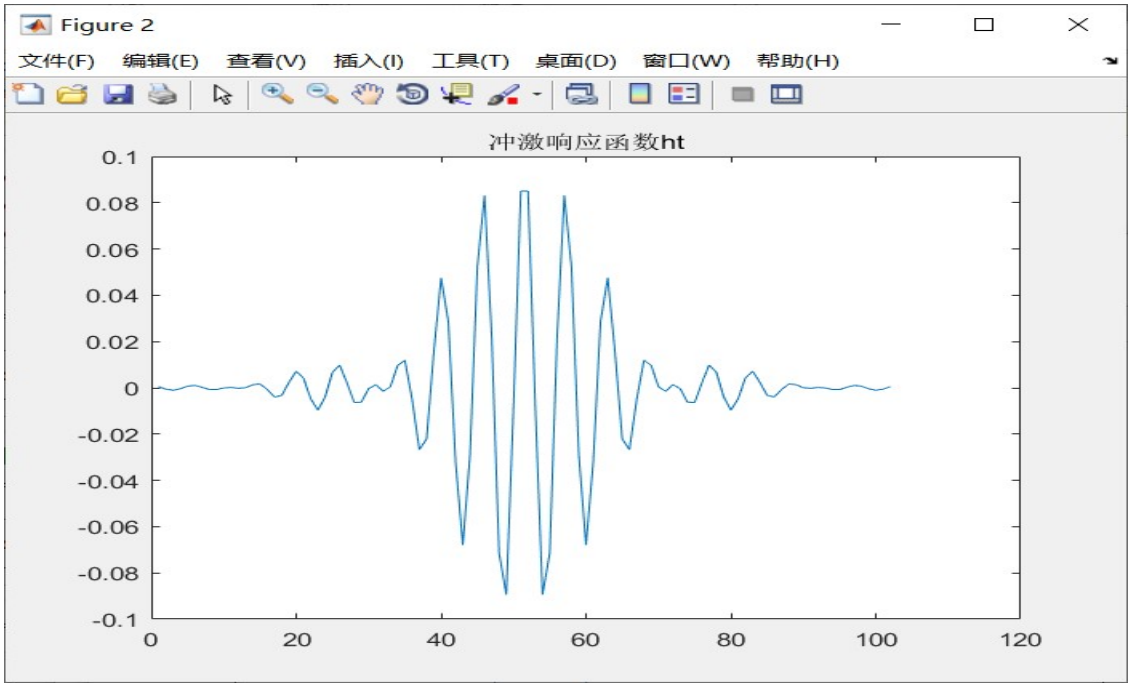


5. 使用 GUI 进行联合验收与展示

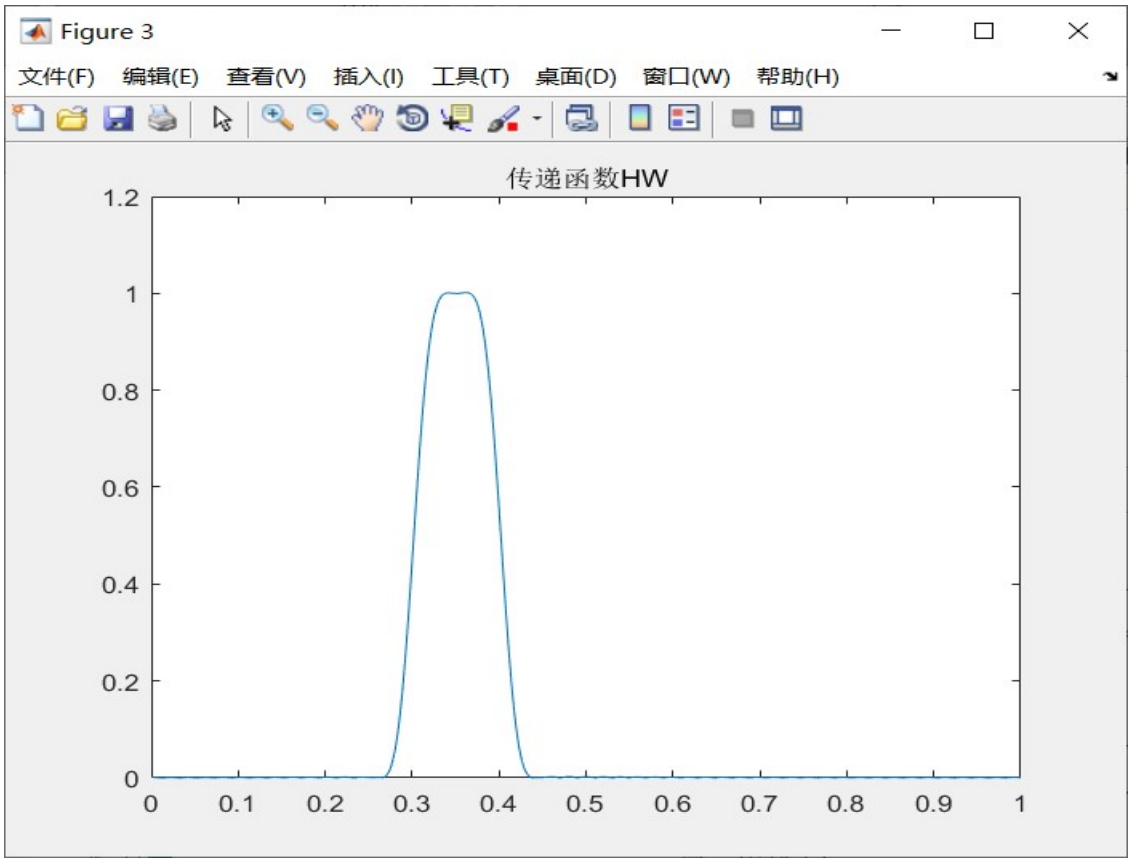


实验三：系统对随机信号响应的统计特性分析及仿真

1. 传递函数的时域冲击响应

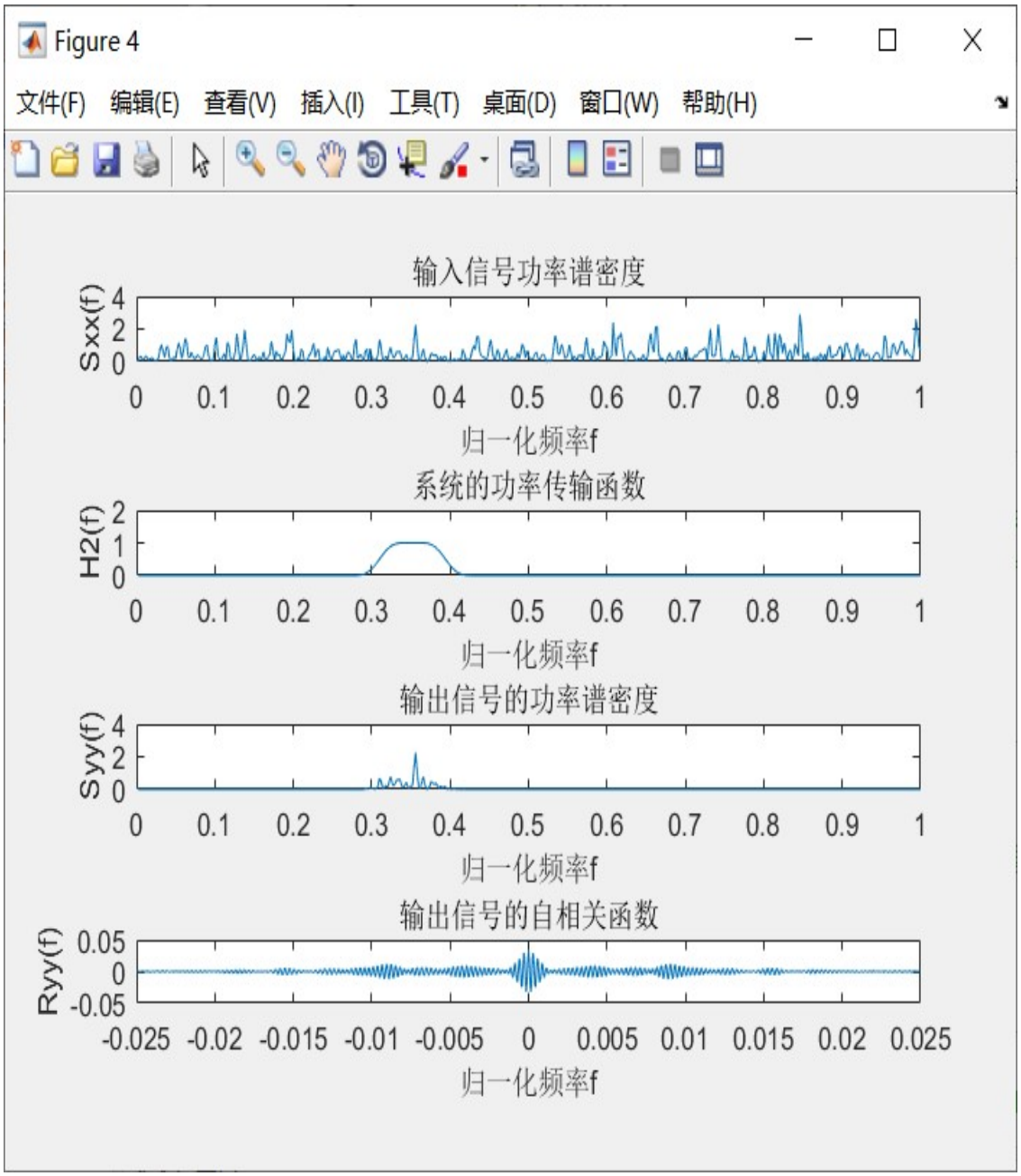


2.传递函数的时域波形



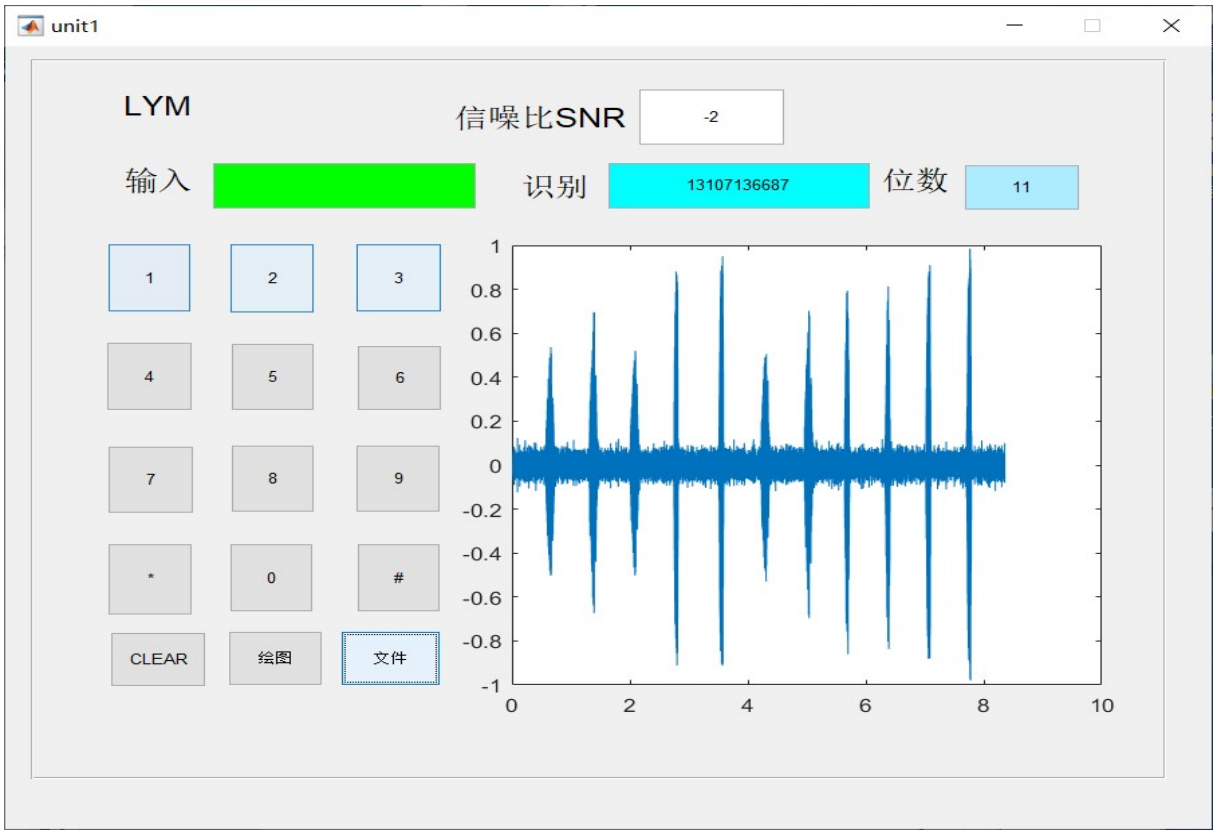
3. 图形展示系统输出观察效果。

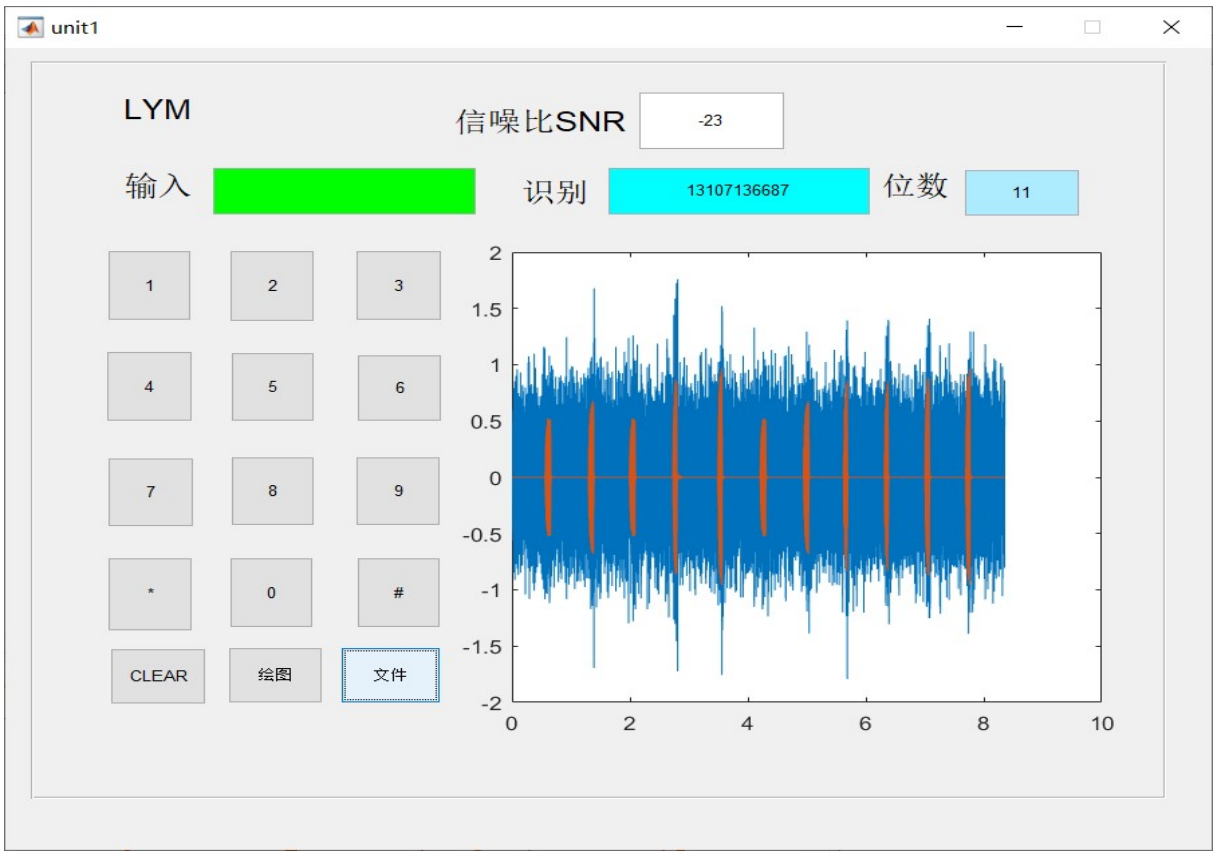
信号通过带通系统，输出的时域与频域波形：



实验四：探究式实验内容 DTMF 电话号码识别

识别的号码	13107136687





红色为原始信号时域波形图，蓝色为加-23dB 噪声后信号的时域波形图。噪声已经完全掩盖住了原始信号，仍旧能够正常识别。

六、 实验结果与分析

1. 直接法估计随机信号功率谱

由于输入信号 xn=a1*cos(2*pi*30*t+fai(1))+a2*cos(2*pi*100*t+fai(2))+randn(1,N);所以功率谱的两个峰值应该在 30Hz 和 100Hz 附近，根据画出的功率谱图，可以看出符合理论知识。用周期图法自编的函数得到的功率谱和内置的 periodogram 函数得到的功率谱，由图像可以看出差别不大，说明自编函数符合要求。

2. 间接法估计随机信号功率谱

输入信号与实验 1 一样，通过功率谱可以看出符合理论要求。比较自编自相关法求功率谱的函数得到的功率谱，和内置函数 periodogram 函数和 psd 函数得到的功率谱，可以验证自编自相关法函数的正确性。对比周期图法和 Welch 法，周期图谱估计曲线的波动很大，即估计的方差较大，而 Welch 法谱估计曲线较为平滑，方差减小，但分辨率降低；对 Welch 法，当数据分段数增加，各段数据长度较短时，谱的分辨率明显下降，而谱估计曲线较为平滑，方差较小；反之，当数据分段数减小，各段数据长度较长时，谱分辨率明显提高，而谱估计曲线波动较大，方差较大。查阅资料显示 psd 的原理是求功率谱而 pwelch 是求功率谱密度，psd 的谱线在输入信号更复杂的情况下不只信号源频率处谱线突出，其他频率也会有突出，本实验可能由于输入信号还不够复杂没能显示出来。

3. 系统对随机信号响应的统计特性分析及仿真

本实验从时域和频域两个角度来观察信号通过带通系统的变化。可以看出，在通过带通系统后，输出信号的频率被较好地限制在了所设的频率范围内，说明滤波器设计的比较合理。由于输入的信号是一段白噪声，理论上在通过频谱为窗函数的系统函数时，输出的自相关函数为抽样函数，可以从图中看出，输出的自相关确实是抽样函数的趋势，与理论所学相符。而要想使抽样函数泄露尽可能小，可以加大传递函数的宽度，使相关函数尽可能集中在中间，但这种泄露不可避免。

4. 探究式实验内容：DTMF 电话号码识别

本实验在原有的基础上，加上了噪声后再进行识别，所设计的代码在信噪比为**-20dB**的时候仍比较稳定，能够识别出结果。具体步骤为：读入一段拨号音，得到数据点以及采样率，加上噪声。进行时域分割，进行 fft 变换，计算功率，判断功率是否满足条件，进行相应的操作，最后显示拨号码以及展现出加噪声后以及本来的时域波形图。

分割法的设计思路，一种是进行幅值识别，只要信号未完全被噪声淹没就可使用，根据实际情况进行改进，加入带通滤波器减小噪声干扰，成功将噪声降低至信号幅值之下。为了避免噪声的较大点造成识别错误，采用统计一个窗内判断超过设定阈值的点数，当大于一个数量，说明此段中包含信号，将其截取出来存放，如此循环结束后，再分别对含有信号的部分进行傅里叶变换，求其频域的两个最大值（1000Hz 以下，1000Hz 以上各一个），得到对应的生成频率。

另一种是频域识别，由于信号功率和噪声功率又明显差别，对读入信号在时域进行加窗分段，并对每一段进行 fft 变换，之后计算功率，根据信噪比计算出合理的有效功率阈值，将每一段的功率谱的 0-1700Hz 的两个峰值（分别在 0-1000Hz 和 1000-1700Hz）与有效功率阈值比较，如果满足要求，则将此段视为有效段，将其截取出来存放，并将窗移动一个拨号音的长度即 fs/10，再分别对含有信号的部分进行傅里叶变换，对其进行最大频率搜索，并对应识别出拨号号码。

为了避免同一拨号音的多次识别，防止漏掉号码（无效段所进步长较短，几率较小），可以将满足要求的窗步进一大步，对于不满足要求的无效段，将窗前进一个较小的步长，最小可以为 1，而为了提高识别效率，采用了 i=ceil(i+600/48000*Fs)-count1 语句，对无效段的步长进行动态调整，使识别效率有了极大提高，尽管准确率不如最小步长，但速度的提高的利远大于弊。

循环进行截取直到只剩一个小于窗的长度的数据段，为了防止取值超过数据点数量，此次取窗只取剩余点数，保证了整段的识别。

从幅度角度，加滤波器是有用的，但是滤波器的作用不是无限的，它很难滤掉信号频率附近的噪声，当这些噪声的影响足够大时，幅值被噪声淹没时，就没有意义了。从频域角度，加滤波器意义不大，因为信号两侧的噪声功率不大于信号功率时，对系统无影响。

七、讨论、建议、质疑

首先本次实验使用的是 atlab2016a，包含 psd 函数，操作起来比较方便，不过我也意识到了每次实验报告中写的实验设备和操作环境的重要性，它决定了实验成果能否实现。

此外最后设计了 GUI 想去仔细比较 psd 和 welch 的区别，不过只有用更复杂的信号，二者的区别才能显现。

这次实验运用到了不少数字信号处理方面的知识，比如模拟频率转换为数字频率、滤波器的设计和各种窗的特点，以前不扎实的基础为我这次实验造成了很多麻烦。

有的时候在老师给的 HTML 框架下会影响自己的探究性想法，今后会尽量减少对 HTML 提示的依赖性，多自己找函数、写函数，多出错才能多思考。

本次实验所得结果与理论知识基本相符，而仿真又一次加深了我对理论的理解。通过本次实验，我再次体会到 MATLAB 在学习、科研过程中所能起到的巨大作用

八、MATLAB 代码

```
%% task2_1
%%  配置环境
clear all;
close all;
clc;
%%  生成随机信号
%两个带随机相位的单谱信号与白噪声之和
N=1024;fs=1000;                          %序列长度和采样频率
t=(0:N-1)/fs;                            %时间序列
fai=random('unif',0,2*pi,1,2);       %产生 2 个[0，2pi]内均匀随机数
xn=cos(2*pi*30*t+fai(1))+3*cos(2*pi*100*t+fai(2))+randn(1,N); %产生含噪声的随机序列
figure,plot(xn);
title('随机信号时域波形')

%%  直接法谱估计
%对样本进行傅里叶变换，取模平方时间平均
Sx1=abs(fft(xn)).^2/N;                   %估计功率谱
f=(0:N/2-1)*fs/N;                        %频率轴坐标
figure
subplot(211);
plot(f,10*log10(Sx1(1:N/2)));grid on; %用 dB/Hz 做功率谱单位，画图
xlabel('f（Hz）');
ylabel('Sx1(f)(dB/Hz)');
title('周期图法估计功率谱');
```

```
%% 与内置函数对比
%periodogram
Sx2=periodogram(xn);
subplot(212);
plot(f,10*log10(Sx2(1:N/2)));grid on;
xlabel('f(Hz)');
ylabel('Sx2(f)(dB/Hz)');
title('periodogram 函数估计功率谱');


%% task2_2
%% 配置环境
clear all;
close all;
clc;
%% 生成随机信号
%两个带随机相位的单谱信号与白噪声之和
N=1024;fs=1000;                                    %序列长度和采样频率
t=(0:N-1)/fs;                                      %时间序列
fai=random('unif',0,2*pi,1,2);                     %产生 2 个[0，2pi]内均匀随机数
xn=cos(2*pi*30*t+fai(1))+3*cos(2*pi*100*t+fai(2))+randn(1,N);   %产生含噪声的随机序列
figure,plot(xn);grid on;
title('随机信号时域波形')

%% 间接法谱估计
Rxx=xcorr(xn,'biased');                            %估计自相关函数 Rxx
Sx1=abs(fft(Rxx));                                 %对 Rxx 进行 FFT 得到功率谱
f=(0:N-1)*fs/N/2;                                  %频率轴坐标
figure;plot(f,10*log10(Sx1(1:N)));grid on;          %用 dB/Hz 做功率谱单位，画图
```

```
xlabel('f(Hz)');
ylabel('Sx(f)(dB/Hz)');
title('自相关函数法估计功率谱');

%% 内置函数 psd
Nseg=256;                                   %分段间隔为 256
window=hanning(Nseg);                       %汉宁窗
noverlap=Nseg/2;                            %重叠点数为 128
f=(0:Nseg/2)*fs/Nseg;                       %频率轴坐标
Sx2=psd(xn,Nseg,fs,window,noverlap,'none');  %psd 函数估计功率谱
figure;
plot(f,10*log10(Sx2));grid on;
xlabel('f(Hz)');
ylabel('Sx(f)(dB/Hz)');
title('Bartlett 法估计功率谱');
%% 内置函数 pwelch
Sx3=pwelch(xn,window,128,Nseg,fs,'onesided')*fs/2;
figure;                                     %Welch 函数估计功率谱
plot(f,10*log10(Sx3));grid on;
xlabel('f(Hz)');
ylabel('Sx(f)(dB/Hz)');
title('Welch 法估计功率谱');

%% 比较三者的误差
% Sx1:1*2047,Sx2:129*1,Sx3:129*1
Sx11=Sx1';
err1=Sx11(1:16:N*2-1)-Sx2(1:N/8);
ff=f(1:length(f)-1);
figure;subplot(311);plot(ff,10*log10(err1));
xlabel('f(Hz)');
```

```
ylabel('Sx1(f)-Sx2(f) (dB/Hz)');
title('间接法谱估计与 psd 函数估计的误差');
err2=Sx11(1:16:N*2-1)-Sx3(1:N/8);
subplot(312);plot(ff,10*log10(err2));
xlabel('f(Hz)');
ylabel('Sx1(f)-Sx3(f) (dB/Hz)');
title('间接法谱估计与 pwelch 函数估计的误差');
err3=Sx2-Sx3;
subplot(313);plot(f,10*log10(err3));
xlabel('f(Hz)');
ylabel('Sx2(f)-Sx3(f) (dB/Hz)');
title('psd 函数估计与 pwelch 函数估计的误差');
```

```
%% task2_3
%% 配置环境
clear all;
close all;
clc;
%% 产生高斯白噪声
N=500;                                          %样本长度 N=500，对应时长 25ms
xt=random('norm',0,1,1,N);                      %产生 1*N 个高斯随机数
% 显示时域波形
figure,plot(xt);
title('随机信号时域波形')
%% 设计带通滤波器
%冲激响应
ht=fir1(101,[0.3 0.4]);                         % 101 阶带通滤波器，数字截止频率为 0.3 和 0.4
% 显示冲激响应函数 ht
```

```
figure,plot(ht)
title('冲激响应函数 ht')
%% 传递函数
HW=fft(ht,2*N);                                    %2N 点滤波器频率响应（系统传输函数）
% 显示传递函数 HW
figure,plot((1:N)/N,abs(HW(1:N)));
title('传递函数 HW')
%% 估计输入信号的自相关和功率谱
Rxx=xcorr(xt,'biased');                            %直接法估计白噪声的自相关函数
% 显示自相关函数 Rxx
figure,stem(Rxx)
title('输入自相关函数 Rxx')
%% 功率谱
Sxx=abs(fft(xt,2*N).^2)/(2*N);                     %周期图法估计白噪声的功率谱
%% 系统求解  频域求解：输出功率谱
HW2=abs(HW).^2;                                    %系统的功率传输函数
Syy=Sxx.*HW2;                                      %输出信号的功率谱
%% 时域求解：输出自相关
Ryy=fftshift(ifft(Syy));                           % 用 IFFT 求输出信号的自相关函
数                                                 %函数 fftshift 对数组进行移位
%Ryy=ifft(Syy);
%% 图形展示
w=(1:N)/N;                                         %功率谱密度横轴坐标
t=(-N:N-1)/N*(N/20000);                            %自相关函数横轴坐标
subplot(4,1,1);plot(w,abs(Sxx(1:N)));              %输入信号功率谱密度
xlabel('归一化频率 f');ylabel('Sxx(f)');title('输入信号功率谱密度');
subplot(4,1,2);plot(w,abs(HW2(1:N)));              %系统的功率传输函数
xlabel('归一化频率 f');ylabel('H2(f)');title('系统的功率传输函数');
subplot(4,1,3);plot(w,abs(Syy(1:N)));              %输出信号的功率谱密度
xlabel('归一化频率 f');ylabel('Syy(f)');title('输出信号的功率谱密度');
```

```
subplot(4,1,4);plot(t,Ryy);                    %输出信号的自相关函数
xlabel('归一化频率 f');ylabel('Ryy(f)');title('输出信号的自相关函数');
```

实验任务 1、2 的 GUI

```
function varargout = LYM_GUI(varargin)
% LYM_GUI MATLAB code for LYM_GUI.fig
%      LYM_GUI, by itself, creates a new LYM_GUI or raises the existing
%      singleton*.
%
%      H = LYM_GUI returns the handle to a new LYM_GUI or the handle to
%      the existing singleton*.
%
%      LYM_GUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in LYM_GUI.M with the given input arguments.
%
%      LYM_GUI('Property','Value',...) creates a new LYM_GUI or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before LYM_GUI_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to LYM_GUI_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 LYM_GUI

% Last Modified by GUIDE v2.5 09-Nov-2020 23:03:43
```

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

if 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 LYM_GUI is made visible.
function LYM_GUI_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 LYM_GUI (see VARARGIN)

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

```
% Update handles structure
guidata(hObject, handles);

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


% --- Outputs from this function are returned to the command line.
function varargout = LYM_GUI_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 Amp1_Callback(hObject, eventdata, handles)
% hObject    handle to Amp1 (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 Amp1 as text
%        str2double(get(hObject,'String')) returns contents of Amp1 as a double


% --- Executes during object creation, after setting all properties.
```

```
function Amp1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Amp1 (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 usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function Freq1_Callback(hObject, eventdata, handles)
% hObject    handle to Freq1 (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 Freq1 as text
%        str2double(get(hObject,'String')) returns contents of Freq1 as a double


% --- Executes during object creation, after setting all properties.
function Freq1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Freq1 (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 usually have a white background on Windows.
%       See ISPC and COMPUTER.
```

```
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function Pha1_Callback(hObject, eventdata, handles)
% hObject    handle to Pha1 (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 Pha1 as text
%        str2double(get(hObject,'String')) returns contents of Pha1 as a double


% --- Executes during object creation, after setting all properties.
function Pha1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Pha1 (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 usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function Amp2_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to Amp2 (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 Amp2 as text
%        str2double(get(hObject,'String')) returns contents of Amp2 as a double


% --- Executes during object creation, after setting all properties.
function Amp2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Amp2 (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 usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function Freq2_Callback(hObject, eventdata, handles)
% hObject    handle to Freq2 (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 Freq2 as text
%        str2double(get(hObject,'String')) returns contents of Freq2 as a double
```

```
% --- Executes during object creation, after setting all properties.
function Freq2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Freq2 (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 usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function Pha2_Callback(hObject, eventdata, handles)
% hObject    handle to Pha2 (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 Pha2 as text
%        str2double(get(hObject,'String')) returns contents of Pha2 as a double


% --- Executes during object creation, after setting all properties.
function Pha2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Pha2 (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 usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function NUM_Callback(hObject, eventdata, handles)
% hObject    handle to NUM (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 NUM as text
%        str2double(get(hObject,'String')) returns contents of NUM as a double


% --- Executes during object creation, after setting all properties.
function NUM_CreateFcn(hObject, eventdata, handles)
% hObject    handle to NUM (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 usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function Fs_Callback(hObject, eventdata, handles)
% hObject    handle to Fs (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 Fs as text
%        str2double(get(hObject,'String')) returns contents of Fs as a double


% --- Executes during object creation, after setting all properties.
function Fs_CreateFcn(hObject, eventdata, handles)
% hObject    handle to Fs (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 usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function AmpN_Callback(hObject, eventdata, handles)
% hObject    handle to AmpN (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 AmpN as text
```

```
%               str2double(get(hObject,'String')) returns contents of AmpN as a double


% --- Executes during object creation, after setting all properties.
function AmpN_CreateFcn(hObject, eventdata, handles)
% hObject       handle to AmpN (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 usually have a white background on Windows.
%             See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    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)
Amp1 = eval(get(handles.Amp1,'String'));
Pha1 = eval(get(handles.Pha1,'String'));
Freq1 = eval(get(handles.Freq1,'String'));
Amp2 = eval(get(handles.Amp2,'String'));
Pha2 = eval(get(handles.Pha2,'String'));
Freq2 = eval(get(handles.Freq2,'String'));
Num =eval( get(handles.NUM,'String'));
AmpN = eval(get(handles.AmpN,'String'));
Fs = eval(get(handles.Fs,'String'));
```

```
%两个带随机相位的单谱信号与白噪声之和
t=(0:Num-1)/Fs;                                    %时间序列
xn=Amp1*cos(2*pi*Freq1*t+Pha1)+Amp2*cos(2*pi*Freq2*t+Pha2)+AmpN*randn(1,Num); %产生含噪声的随机序列
plot(handles.axes1,t,xn);
title(handles.axes1,'随机信号时域波形')
xlim(handles.axes1,[0,(Num-1)/Fs]);
%% 直接法谱估计
%对样本进行傅里叶变换，取模平方时间平均
Sx1=abs(fft(xn)).^2/Num;                           %估计功率谱
f=(0:Num/2-1)*Fs/Num;                              %频率轴坐标
plot(handles.axes2,f,10*log10(Sx1(1:Num/2))); %用 dB/Hz 做功率谱单位，画图
xlabel(handles.axes2,'f（Hz）');
ylabel(handles.axes2,'Sx1(f)(dB/Hz)');
title(handles.axes2,'周期图法估计功率谱');
%% 与内置函数对比
%periodogram
Sx2=periodogram(xn);
plot(handles.axes3,f,10*log10(Sx2(1:Num/2)));
xlabel(handles.axes3,'f(Hz)');
ylabel(handles.axes3,'Sx2(f)(dB/Hz)');
title(handles.axes3,'periodogram 函数估计功率谱');

%% 间接法谱估计
Rxx=xcorr(xn,'biased');                            %估计自相关函数 Rxx
Sx3=abs(fft(Rxx));                                 %对 Rxx 进行 FFT 得到功率谱
f=(0:Num-1)*Fs/Num/2;                              %频率轴坐标
plot(handles.axes4,f,10*log10(Sx3(1:Num)));        %用 dB/Hz 做功率谱单位，画图
xlabel(handles.axes4,'f(Hz)');
ylabel(handles.axes4,'Sx(f)(dB/Hz)');
```

```
title(handles.axes4,'自相关函数法估计功率谱');
%% 内置函数 psd
Nseg=256;                                    %分段间隔为 256
window=hanning(Nseg);                        %汉宁窗
noverlap=Nseg/2;                             %重叠点数为 128
f=(0:Nseg/2)*Fs/Nseg;                        %频率轴坐标
Sx4=psd(xn,Nseg,Fs,window,noverlap,'none');  %psd 函数估计功率谱
plot(handles.axes5,f,10*log10(Sx4));
xlabel(handles.axes5,'f(Hz)');
ylabel(handles.axes5,'Sx(f)(dB/Hz)');
title(handles.axes5,'Bartlett 法估计功率谱');

%% 内置函数 pwelch
Sx5=pwelch(xn,window,128,Nseg,Fs,'onesided')*Fs/2;                    %Welch 函
数估计功率谱
plot(handles.axes6,f,10*log10(Sx5));grid on;
xlabel(handles.axes6,'f(Hz)');
ylabel(handles.axes6,'Sx(f)(dB/Hz)');
title(handles.axes6,'Welch 法估计功率谱');

guidata(hObject, handles);
```

探究实验 GUI

```
function varargout = unit1(varargin)
% UNIT1 MATLAB code for unit1.fig
%      UNIT1, by itself, creates a new UNIT1 or raises the existing
%      singleton*.
%
%      H = UNIT1 returns the handle to a new UNIT1 or the handle to
%      the existing singleton*.
```

```
%
%       UNIT1('CALLBACK',hObject,eventData,handles,...) calls the local
%       function named CALLBACK in UNIT1.M with the given input arguments.
%
%       UNIT1('Property','Value',...) creates a new UNIT1 or raises the
%       existing singleton*.  Starting from the left, property value pairs are
%       applied to the GUI before unit1_OpeningFcn gets called.  An
%       unrecognized property name or invalid value makes property application
%       stop.  All inputs are passed to unit1_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 unit1

% Last Modified by GUIDE v2.5 17-Nov-2020 21:40:09

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',          mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @unit1_OpeningFcn, ...
                   'gui_OutputFcn',  @unit1_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
```

```
if 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 unit1 is made visible.
function unit1_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 unit1 (see VARARGIN)

% Choose default command line output for unit1
handles.output = hObject;
handles.numcount=[];
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes unit1 wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = unit1_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 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)
n=[1:410]; %  每个数字用 410  个采样点表示
d1=sin(2*pi*697/8192*n)+sin(2*pi*1209/8192*n); %  对应行频列频叠加
space=zeros(1,410); %410  个 0  模拟静音信号
phone=[d1,space];%要发送的信号
strnum1=strcat(get(handles.numscreen,'string'),'1');%之前的数字连上数字‘1’
set(handles.numscreen,'string',strnum1);%显示在显示屏上
handles.numcount=[handles.numcount,phone];%整合要发送的信号
sound(d1,8192);%播放按键的声音
guidata(hObject,handles);%更新



% --- 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)
```

```
n=[1:410]; %  每个数字用 410  个采样点表示
d2=sin(2*pi*697/8192*n)+sin(2*pi*1336/8192*n); %  对应行频列频叠加
space=zeros(1,410); %410  个 0  模拟静音信号
phone=[d2,space];%要发送的信号
strnum2=strcat(get(handles.numscreen,'string'),'2');%之前的数字连上数字‘2’
set(handles.numscreen,'string',strnum2);%显示在显示屏上
handles.numcount=[handles.numcount,phone];%整合要发送的信号
sound(d2,8192);%播放按键的声音
guidata(hObject,handles);%更新


% --- 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)
n=[1:410]; %  每个数字用 410  个采样点表示
d3=sin(2*pi*697/8192*n)+sin(2*pi*1477/8192*n); %  对应行频列频叠加
space=zeros(1,410); %410  个 0  模拟静音信号
phone=[d3,space];%要发送的信号
strnum3=strcat(get(handles.numscreen,'string'),'3');%之前的数字连上数字‘3’
set(handles.numscreen,'string',strnum3);%显示在显示屏上
handles.numcount=[handles.numcount,phone];%整合要发送的信号
sound(d3,8192);%播放按键的声音
guidata(hObject,handles);%更新


% --- 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)
n=[1:410]; %  每个数字用 410  个采样点表示
d4=sin(2*pi*770/8192*n)+sin(2*pi*1209/8192*n); %  对应行频列频叠加
space=zeros(1,410); %410  个 0  模拟静音信号
phone=[d4,space];%要发送的信号
strnum4=strcat(get(handles.numscreen,'string'),'4');%之前的数字连上数字‘4’
set(handles.numscreen,'string',strnum4);%显示在显示屏上
handles.numcount=[handles.numcount,phone];%整合要发送的信号
sound(d4,8192);%播放按键的声音
guidata(hObject,handles);%更新


% --- 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)
n=[1:410]; %  每个数字用 410  个采样点表示
d5=sin(2*pi*770/8192*n)+sin(2*pi*1336/8192*n); %  对应行频列频叠加
space=zeros(1,410); %410  个 0  模拟静音信号
phone=[d5,space];%要发送的信号
strnum5=strcat(get(handles.numscreen,'string'),'5');%之前的数字连上数字‘5’
set(handles.numscreen,'string',strnum5);%显示在显示屏上
handles.numcount=[handles.numcount,phone];%整合要发送的信号
sound(d5,8192);%播放按键的声音
guidata(hObject,handles);%更新


% --- 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)
n=[1:410]; %  每个数字用 410  个采样点表示
d6=sin(2*pi*770/8192*n)+sin(2*pi*1477/8192*n); %  对应行频列频叠加
space=zeros(1,410); %410  个 0  模拟静音信号
phone=[d6,space];%要发送的信号
strnum6=strcat(get(handles.numscreen,'string'),'6');%之前的数字连上数字‘6’
set(handles.numscreen,'string',strnum6);%显示在显示屏上
handles.numcount=[handles.numcount,phone];%整合要发送的信号
sound(d6,8192);%播放按键的声音
guidata(hObject,handles);%更新


% --- 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)
n=[1:410]; %  每个数字用 410  个采样点表示
d7=sin(2*pi*852/8192*n)+sin(2*pi*1209/8192*n); %  对应行频列频叠加
space=zeros(1,410); %410  个 0  模拟静音信号
phone=[d7,space];%要发送的信号
strnum7=strcat(get(handles.numscreen,'string'),'7');%之前的数字连上数字‘7’
set(handles.numscreen,'string',strnum7);%显示在显示屏上
handles.numcount=[handles.numcount,phone];%整合要发送的信号
sound(d7,8192);%播放按键的声音
guidata(hObject,handles);%更新
```

```
% --- 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=[1:410]; %  每个数字用 410  个采样点表示
d8=sin(2*pi*852/8192*n)+sin(2*pi*1336/8192*n); %  对应行频列频叠加
space=zeros(1,410); %410  个 0  模拟静音信号
phone=[d8,space];%要发送的信号
strnum8=strcat(get(handles.numscreen,'string'),'8');%之前的数字连上数字‘8’
set(handles.numscreen,'string',strnum8);%显示在显示屏上
handles.numcount=[handles.numcount,phone];%整合要发送的信号
sound(d8,8192);%播放按键的声音
guidata(hObject,handles);%更新


% --- 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)
n=[1:410]; %  每个数字用 410  个采样点表示
d9=sin(2*pi*852/8192*n)+sin(2*pi*1477/8192*n); %  对应行频列频叠加
space=zeros(1,410); %410  个 0  模拟静音信号
phone=[d9,space];%要发送的信号
strnum9=strcat(get(handles.numscreen,'string'),'9');%之前的数字连上数字‘9’
set(handles.numscreen,'string',strnum9);%显示在显示屏上
handles.numcount=[handles.numcount,phone];%整合要发送的信号
sound(d9,8192);%播放按键的声音
```

```
guidata(hObject,handles);%更新



% --- 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)
num=get(handles.numscreen,'string');%获取屏幕上的字符串
l=length(num);%求其字符串长度 L
n11=strrep(num,num,num(1:l-1));%取前(L-1)个字符替换现在的字符串，相当于删掉最后一个字符
set(handles.numscreen,'string',n11);%显示后的字符结果
L=length(handles.numcount);%记录现在的发送数据信号长度
handles.numcount=handles.numcount(1:L-820); %取前(L-1)个字符信号替换现在的信号，相当于发送时
删掉最后一个字符
n=[1:410];
d10=sin(2*pi*941/8192*n)+sin(2*pi*1209/8192*n); %对应行频列频叠加
sound(d10,8192); %播放按键声音
guidata(hObject,handles);%更新



% --- 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)
n=[1:410]; %  每个数字用 410  个采样点表示
d0=sin(2*pi*941/8192*n)+sin(2*pi*1336/8192*n); %  对应行频列频叠加
space=zeros(1,410); %410  个 0  模拟静音信号
```

```
phone=[d0,space];%要发送的信号
strnum0=strcat(get(handles.numscreen,'string'),'0');%之前的数字连上数字‘0’
set(handles.numscreen,'string',strnum0);%显示在显示屏上
handles.numcount=[handles.numcount,phone];%整合要发送的信号
sound(d0,8192);%播放按键的声音
guidata(hObject,handles);%更新



% --- 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)
n=[1:410];
d11=sin(2*pi*941/8192*n)+sin(2*pi*1477/8192*n); % 对应行频列频叠加
sound(d11,8192);%发出按键声音
numlength=length(handles.numcount);%接收到的信号长度
count=numlength/820;%计算出发了多少个数字
set(handles.receivescreen,'string','');%清屏
for i=1:count%一共 count 个数字
    d=handles.numcount(1+820*(i-1):820*i);%取每位拨号音的采样点
    f=fft(d,8192); % 以 N=2048 作 FFT 变换 d 是取出来每位拨号音的采样点
    a=abs(f);
    p=a.*a/8192; % 计算功率谱
    num(1)=find(p(1:1000)==max(p(1:1000))); % 找行频
    num(2)=1000+find(p(1000:1700)==max(p(1000:1700))); % 找列频
    if (num(1) < 730)
        row=1; % 确定行数
    elseif (num(1) < 810)
        row=2;
```

```
elseif (num(1) < 900)
    row=3;
else
    row=4;
end
if (num(2) < 1250)
    coloum=1; %  确定列数
elseif (num(2) < 1390)
    coloum=2;
elseif (num(2) < 1510)
    coloum=3;
else
    coloum=4;
end
if row==4&&coloum==2%计算出对应的数字值
    num=0;
elseif(row==4&&coloum==1)
    num='*';
elseif(row==4&&coloum==3)
    num='#';
elseif(coloum==4)
    if(row==1)
      num='A';
    elseif(row==2)
      num='B';
      elseif(row==2)
      num='C';
    else
      num='D';
    end
```

```
    else
        num=coloum+3*(row-1); %根据行列关系计算出对应的数字值
    end
    strnum=strcat(get(handles.receivescreen, 'string'),num2str(num));%连接新旧字符串
    set(handles.receivescreen,'string',strnum);%将结果显示在屏幕上
end
     set(handles.edit4,'string',count);%将位数显示在屏幕上
guidata(hObject,handles);

function numscreen_Callback(hObject, eventdata, handles)
% hObject    handle to numscreen (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 numscreen as text
%        str2double(get(hObject,'String')) returns contents of numscreen as a double


% --- Executes during object creation, after setting all properties.
function numscreen_CreateFcn(hObject, eventdata, handles)
% hObject    handle to numscreen (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 usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function receivescreen_Callback(hObject, eventdata, handles)
% hObject    handle to receivescreen (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 receivescreen as text
%        str2double(get(hObject,'String')) returns contents of receivescreen as a double


% --- Executes during object creation, after setting all properties.
function receivescreen_CreateFcn(hObject, eventdata, handles)
% hObject    handle to receivescreen (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 usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton1.
function pushbutton14_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)
```

```
% --- 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)
set(handles.numscreen,'string','');
set(handles.receivescreen,'string','');
set(handles.edit4,'string','');
handles.numcount=[];
guidata(hObject,handles);

% --- 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)
sampledata1=handles.numcount;
%语音信号的时域输出
if(length(handles.numcount)==0)
plot(0);
else
t=0:1/8192:(length(sampledata1)-1)/8192;
plot(t,sampledata1);
guidata(hObject,handles);
end
```

```
% --- Executes on button press in pushbutton19.
function pushbutton19_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton19 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[filename,filepath]=uigetfile('.wav','选择音频文件');%打开文件
if(filename==0)
    return%判断文件是否为空
else
audeofile=strcat(filepath,filename);%形成音频文件路径
[sampledata1,Fs]=audioread(audeofile);%从音频文件路径中读取音频文件，获得采样率和音频数据
SNR=get(handles.edit5,'String');
sampledata=awgn(sampledata1,str2num(SNR),'measured');
% ----------------------------------------带通滤波
Hd=band_pass(600,690,1710,1800);                                    %带通滤波器
                                    %带通滤波器
sampledata=filter(Hd,sampledata);
% Hd=bpass(700,710,750,760);
% sampledata=filter(Hd,sampledata);
%滤波
%Hd=band_pass(700,710,750,760);
%sound(y,Fs)
sound(sampledata,Fs);%播放音频文件
n=max(sampledata(1:length(sampledata)))%获得音频数据峰值
count1=0;%初始化
i=1;%初始化
numcount1=[];%初始化
while(i<=length(sampledata)-2640/48000*Fs)%最后一段长 2640，如果按键发音，大小为 2640 的样本长度正好容纳，对之外的每一个点进行讨论
    for j=i:ceil(i+2640/48000*Fs)%对于采样率 48000，用 48000 合适，但 48000*Fs 适用于各种音频。%
```

对讨论是否认为是噪声并需要置 0 的点的后面的 2640 个点进行讨论

if(abs(sampledata(j))>0.3*n)%我们认为峰值的 0.39 是区分按键音和环境噪音的标志，这个是由峰值 1.79 的取高度 0.7 作为分界点，这个是由图像特点，就是噪音和按键音幅度有显著差异决定的

count1=count1+1;%计数

end

end

if(count1>550/48000*Fs)%统计有多少个大于 0.39 的点，用 700 作为分界点进行保留，不是算出来的，而是使用 matlab 试出来的。不写 count1 的‘;’很容易区分出噪声含有的 count1 数目和按键音数目

numcount1=[numcount1,sampledata(i:i+ceil(4799/48000*Fs))']; %!!!!!!!! 正常录音不需转置，但老师的测试音频需要!!!!!! %把按键音合并起来，该点和该点之后 4800 点，即 100ms，会丢掉一部分按键音尾端数据，设计中也会舍弃很少一些首端数据，不过数量少，而且按键音中间的声音更具备特征，信号质量更好。

i=ceil(i+5000/48000*Fs);%取完按键音后跳过该段，不重复截取声音片段。

% else

% i=ceil(i+600/48000*Fs)-count1;

end

i=i+1;%计数

count1=0;%复位

end

t=0:1/Fs:(length(sampledata)-1)/Fs;%输出原始波形的时间转换

plot(t,sampledata);%输出原始波形

hold on;

plot(t,sampledata1);%输出原始波形

hold off;

numlength=length(numcount1);%接收到的信号长度

count=numlength/(1+ceil(4799/48000*Fs));%计算出发了多少个数字

set(handles.receivescreen,'string','');%清屏

for i=1:count%一共 count 个数字

```
d=numcount1(1+(1+ceil(4799/48000*Fs))*(i-1):(1+ceil(4799/48000*Fs))*i);%取每位拨号音的采样点
f=fft(d,Fs); % 以 N=2048 作 FFT 变换 d 是取出来每位拨号音的采样点
a=abs(f);
p=a.*a/Fs; % 计算功率谱
num(1)=find(p(1:1000)==max(p(1:1000))); % 找行频
num(2)=1000+find(p(1000:1700)==max(p(1000:1700))); % 找列频
if (num(1) < 730)
    row=1; % 确定行数
elseif (num(1) < 810)
    row=2;
elseif (num(1) < 900)
    row=3;
else
    row=4;
end
if (num(2) < 1260)
    coloum=1; % 确定列数
elseif (num(2) < 1395)
    coloum=2;
elseif (num(2) < 1550)
    coloum=3;
else coloum=4;
end
if row==4&&coloum==2%计算出对应的数字值
    num=0;
elseif(row==4&&coloum==1)
    num='*';
elseif(row==4&&coloum==3)
    num='#';
```

```
    elseif(coloum==4)
        if(row==1)
          num='A';
        elseif(row==2)
          num='B';
          elseif(row==2)
          num='C';
        else
          num='D';
        end
    else
        num=coloum+3*(row-1); %根据行列关系计算出对应的数字值
    end
    strnum=strcat(get(handles.receivescreen, 'string'),num2str(num));
    set(handles.receivescreen,'string',strnum);%将结果显示在屏幕上
end
     set(handles.edit4,'string',count);%将位数显示在屏幕上
end
guidata(hObject,handles);



function edit4_Callback(hObject, eventdata, handles)
% hObject        handle to edit4 (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 edit4 as text
%              str2double(get(hObject,'String')) returns contents of edit4 as a double
```

```
% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (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 usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function Hd = band_pass(Fstop1,Fpass1,Fpass2,Fstop2);
%BAND_PASS Returns a discrete-time filter object.

% MATLAB Code

% Butterworth Bandpass filter designed using FDESIGN.BANDPASS.

% Generated by MATLAB(R) 9.0 and the Signal Processing Toolbox 7.2.
% Generated on: 18-Dec-2019 14:51:03
% All frequency values are in Hz.
Fs = 44100;  % Sampling Frequency

% Fstop1 = 500;          % First Stopband Frequency
% Fpass1 = 690;          % First Passband Frequency
% Fpass2 = 1710;         % Second Passband Frequency
% Fstop2 = 1800;         % Second Stopband Frequency
```

```
Astop1 = 70;                % First Stopband Attenuation (dB)
Apass  = 1;                 % Passband Ripple (dB)
Astop2 = 70;                % Second Stopband Attenuation (dB)
match  = 'passband';  % Band to match exactly

% Construct an FDESIGN object and call its BUTTER method.
h  = fdesign.bandpass(Fstop1, Fpass1, Fpass2, Fstop2, Astop1, Apass, ...
                      Astop2, Fs);
Hd = design(h, 'butter', 'MatchExactly', match);
function Hd = bpass(Fstop1,Fpass1,Fpass2,Fstop2);
%BAND_PASS Returns a discrete-time filter object.

% MATLAB Code

% Butterworth Bandpass filter designed using FDESIGN.BANDPASS.

% Generated by MATLAB(R) 9.0 and the Signal Processing Toolbox 7.2.
% Generated on: 18-Dec-2019 14:51:03
% All frequency values are in Hz.
Fs = 44100;  % Sampling Frequency

% Fstop1 = 500;               % First Stopband Frequency
% Fpass1 = 690;               % First Passband Frequency
% Fpass2 = 1710;              % Second Passband Frequency
% Fstop2 = 1800;              % Second Stopband Frequency
Astop1 = 30;                % First Stopband Attenuation (dB)
Apass  = 1;                 % Passband Ripple (dB)
Astop2 = 30;                % Second Stopband Attenuation (dB)
match  = 'stopband';  % Band to match exactly
```

```
% Construct an FDESIGN object and call its BUTTER method.
h    = fdesign.bandpass(Fstop1, Fpass1, Fpass2, Fstop2, Astop1, Apass, ...
                                Astop2, Fs);
Hd = design(h, 'butter', 'MatchExactly', match);




function edit5_Callback(hObject, eventdata, handles)
% hObject       handle to edit5 (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 edit5 as text
%              str2double(get(hObject,'String')) returns contents of edit5 as a double



% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject       handle to edit5 (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 usually have a white background on Windows.
%             See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```