浮点 C 子程序的定点化实验

刘东楷 PB15061322

一、实验目的

1、掌握浮点程序定点化转换的方法；

2、理解分段卷积的保留舍去法思想；

3、掌握音频信号加噪声与滤波操作；

4、理解正弦信号采样的准则；

二、实验原理

1、Q 格式：

许多 DSP 都是定点 DSP，处理定点数据相当快，但是处理浮点数据就会非常慢。可以利用 Q 格式将浮点数据转化成定点数据再做运算，以节约运算时间。实际应用中，浮点运算大都时候都是既有整数部分，也有小数部分的。所以要选择一个适当的定标格式才能更好的处理运算。

Q 格式表示为：Qm.n，表示数据用 m 比特表示整数部分，n 比特表示小数部分，共需要 m+n+1 位来表示这个数据，多余的一位用作符合位。假设小数点在 n 位的左边（从右向左数），从而确定小数的精度

例如 Q15 表示小数部分有 15 位，一个 short 型数据，占 2 个字节，最高位是符号位，后面 15 位是小数位，就假设小数点在第 15 位左边，表示的范围是：-1<X<0.9999695 。

浮点数据转化为 Q15，将数据乘以 2^15；Q15 数据转化为浮点数据，将数据除以 2^15。

例如：假设数据存储空间为 2 个字节，0.333×2^15=10911=0x2A9F，0.333 的所有运算就可以用 0x2A9F 表示，同理 10911×2^(-15)=0.332977294921875，可以看出浮点数据通过 Q 格式转化后是有误差的。

2、Q 格式的运算

1> 定点加减法：须转换成相同的 Q 格式才能加减

2> 定点乘法：不同 Q 格式的数据相乘，相当于 Q 值相加，即 Q15 数据乘以 Q10 数据后的结果是 Q25 格式的数据

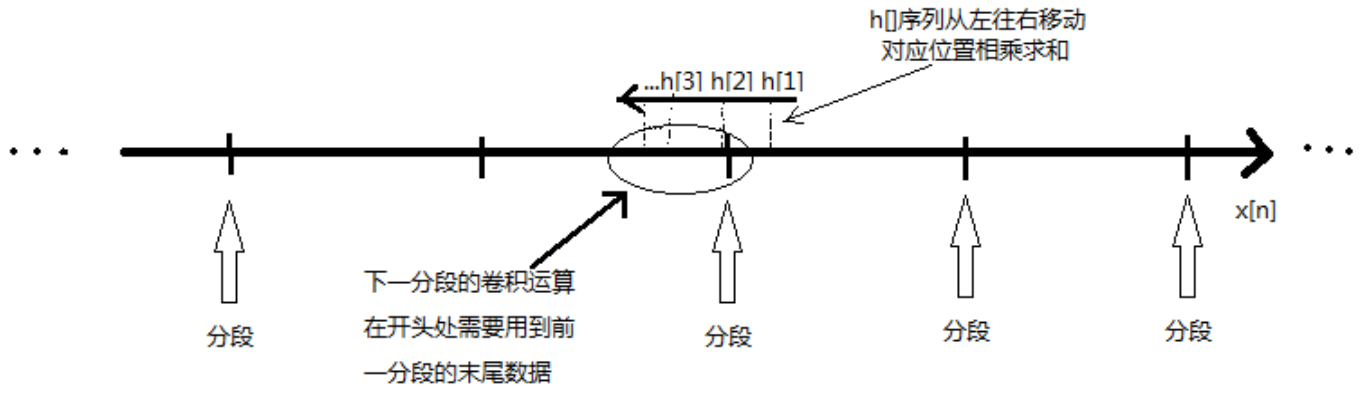
3> 定点除法：不同 Q 格式的数据相除，相当于 Q 值相减

4> 定点左移：左移相当于 Q 值增加

5> 定点右移：右移相当于 Q 减少

3、保留舍去法

由于 x[n]可能会很长，如果将 x[n]存储完毕再做卷积运算将会产生两个问题：一是要就计算机的存储量过大，二是要等待 x[n]输入的时间过长，不能实现“实时处理”，为了解决这一问题，可以将 x[n]分成小段，逐段与 h[n]做卷积。这样做还需要对分段处的运算做一定处理以保证分段运算结果相连等价于不分段的运算结果，本实验采用保留舍去法处理分段处的运算，示意图如下：



从分段卷积的示意图可以看出，下一分段的开头处需要用到前一分段的末尾数据，h[n]序列向右位移到分段末尾时结束该分段运算，即舍去了该分段末尾的运算结果，但将该段 x[n]末尾的数据保留下来并加在下一分段开头，便将前面舍去的运算工作转移到下一分段开头处进行。

4、正弦信号采样的几个准则：

1、抽样频率应为正弦信号频率的整倍数；

2、截短后的正弦信号包含完整周期；

3、每个周期抽样至少 4 个点，或为 2 的整次幂；

4、截短后不补零；

5、信号中包含多个正弦信号时，令其中最高频率的正弦信号满足以上条件。

三、实验内容

（1）C 滤波程序定点化

主要修改：

1> 将滤波器数组 h[]乘以 2^15 转换成 Q15 数，并强制类型转换用 short int 类型存储；

2> 存放从 wav 文件读取到的数据的 indate[]数组和存放滤波结果的 outdate[]数组类型都改成 short int；

3> 由于 wav 文件使用 Q15 数 16bit 存储采样值，所以读取到的数据可直接使用，为方便起见读文件和写文件分别使用 fread()和 fwrite()函数，一次性读写一帧数据量；

4> 由于 Q15 数做乘法运算会变成 Q30 数，所以使用一个 int 类型的数 sum 暂存乘法中间运算值，累加运算结束后左移 15 位恢复成 Q15 数，再强制类型转换成 short int 赋值给 outdate[]；

5> wav 文件 44bytes 头文件原封不动地复制到滤波后的文件中。

```
#include <stdio.h>
#include<math.h>
#define length 180
void my_filter(short int xin[], short int xout[], int n, short int h[] );

short int x1[length+20]={0};
float h_float[19]={0.01218354, -0.009012882, -0.02881839, -0.04743239, 0.04584568, -0.008692503, 0.06446265,
0.1544655, 0.2289794, 0.257883, 0.2289794, 0.1544655, 0.06446265, -0.008692503, -0.04584568, -0.04743239, -
0.02881839, -0.009012882, 0.01218354 };   //实验提供的浮点数滤波器

void my_filter(short int xin[], short int xout[], int n, short int h[] )
{     int i, j;
      int sum;
      for(i=0; i<length; i++ )
            x1[n+i-1] = xin[i];
      for(i=0; i<length; i++ )
      {     for(sum=0, j=0; j<n; j++ )
                  sum+=h[j]*x1[i-j+n-1];   //卷积运算
            xout[i]=sum>>15;   //Q15 数做乘法后变成 Q30 数，需右移 15 位恢复成 Q15 数
      }
      for(i=0; i<n-1; i++)
            x1[n-i-2]=xin[length-1-i];       //保留舍去法
}


main()
{     FILE *fpin, *fpout;
      short int indate[length], outdate[length], h[19];
      int frame, i;
      fpin = fopen("indate.wav", "rb");
      fpout = fopen("outdate.wav", "wb");

      for(i=0; i<19; i++ )  //滤波器定点化
            h[i] = (short int)(h_float[i]*pow(2,15));

      fread(indate, 1, 44, fpin);      //44 字节 wav 文件头
      fwrite(indate, 1, 44, fpout);
```

```
    frame = 0;
    while(feof(fpin)==0)
    {    frame++;
         printf("frame = %d\n", frame);
         fread(indate, sizeof(short int), length, fpin); //读一帧数据
         my_filter(indate, outdate, 19, h);
         fwrite(outdate, sizeof(short int), length, fpout);   //写入运算结果
    }
    fclose(fpin);
    fclose(fpout);
    return(0);
}
```

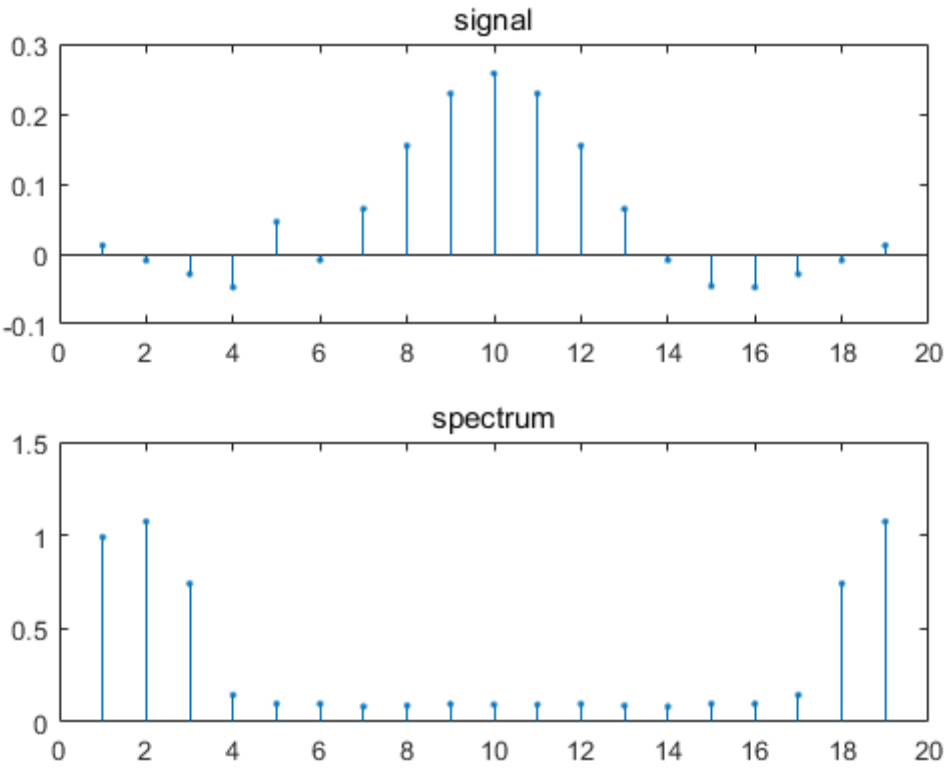
利用 Matlab 的 audioread()的'native'方式可以直接读取 Q15 数，将 C 程序的第一帧 indate[]数据 printf()出来，两者对比结果一致，可以证明 C 程序读取正确。

（2）MATLAB 频谱分析

1、滤波器

h=[0.01218354, -0.009012882, -0.02881839, -0.04743239, 0.04584568, -0.008692503, 0.06446265, 0.1544655, 0.2289794, 0.257883, 0.2289794, 0.1544655, 0.06446265, -0.008692503, -0.04584568, -0.04743239, -0.02881839, -0.009012882, 0.01218354];

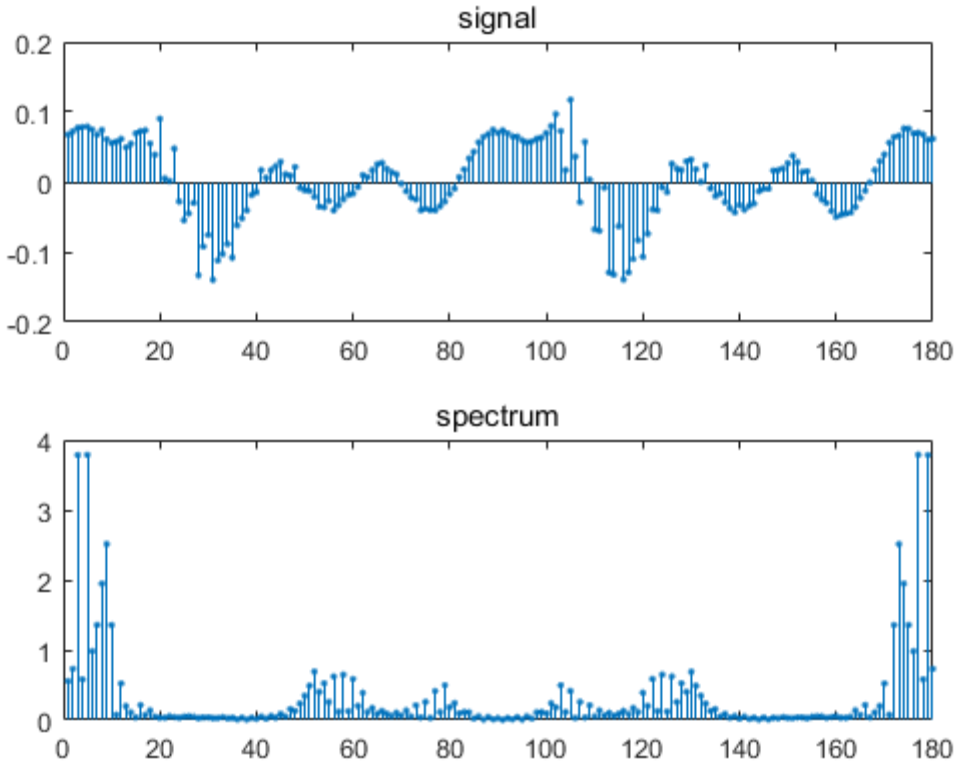
绘制滤波器的波形及频谱



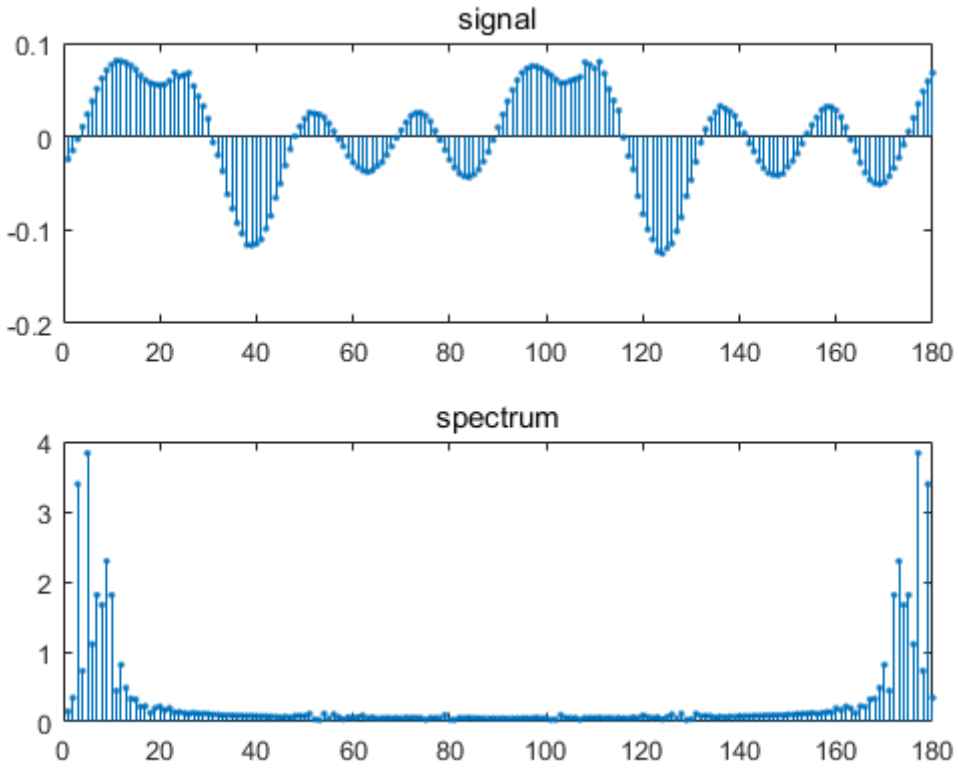
可以看出这是个低通滤波器

2、滤波效果

取一帧分析，原信号：

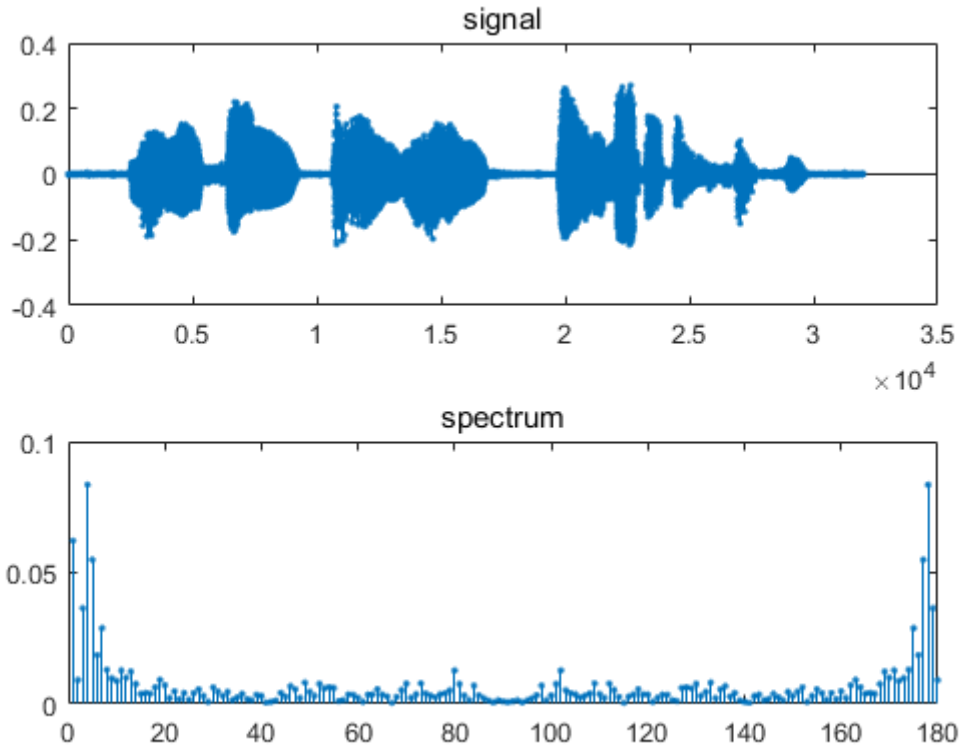


滤波后：

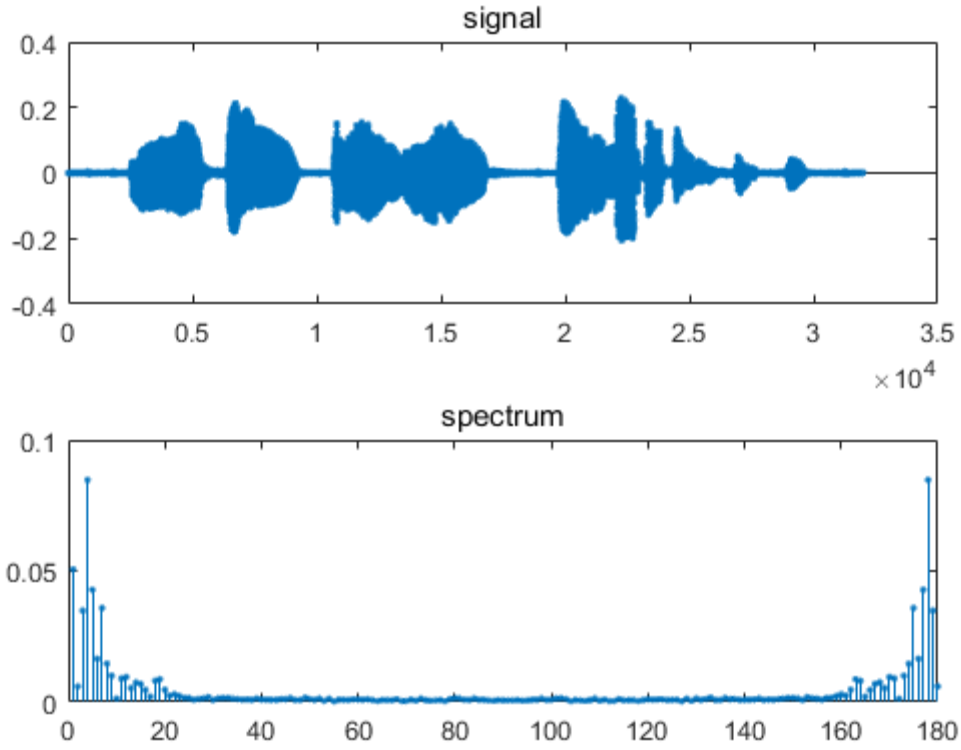


对比滤波前后频谱，高频部分被滤掉了，低频部分基本不变，时域信号滤波后变得光滑。

对信号整体作频谱分析，原始信号：



滤波后：



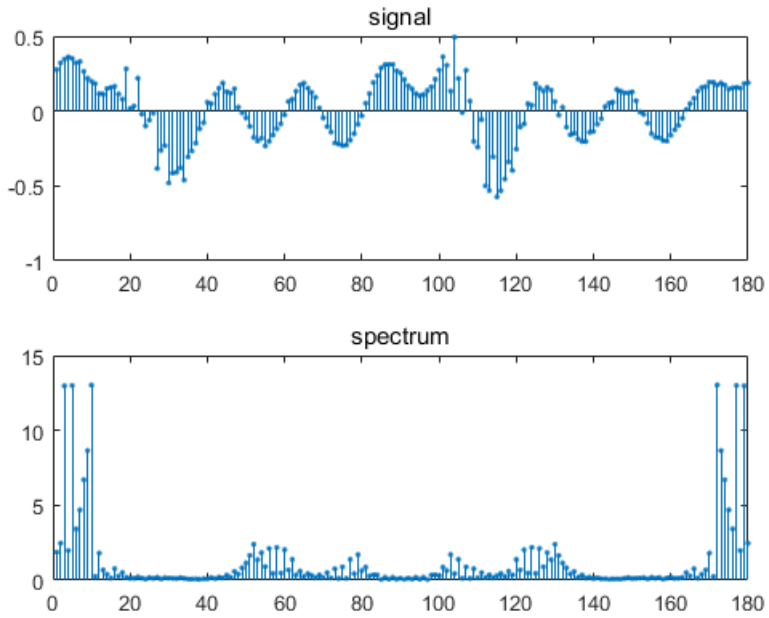
同样可以明显观察到高频几乎全被滤掉。

3、添加噪声

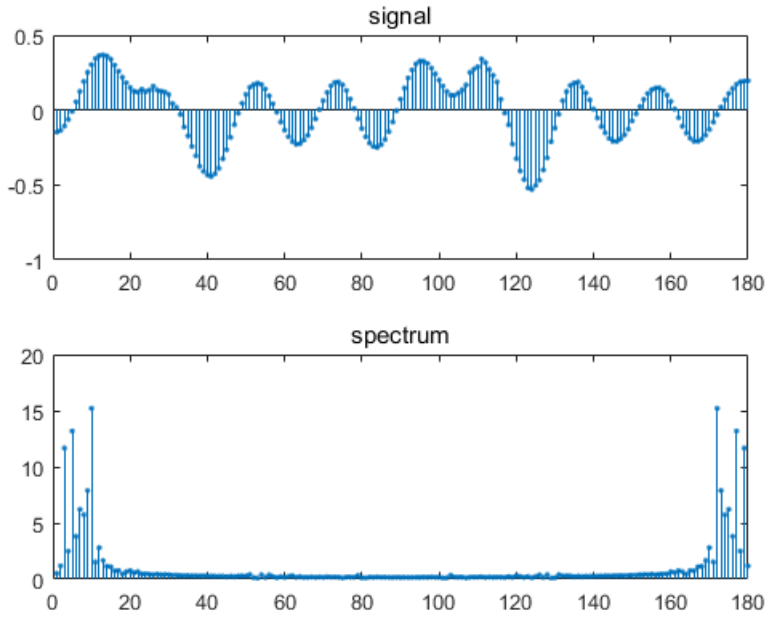
在 MATLAB 中为原始声音信号添加正弦噪声

```
[x,fs] = audioread('bluesky1.wav');
fl=400;%低频
fh=2000;%高频
k=0.1*max(x);%控制噪声幅度
n=1:length(x);
noise_l=sin(2*pi*fl/fs.*n); %低频噪声
noise_h=sin(2*pi*fh/fs.*n); %高频噪声
x_sin=x+k*(noise_l+noise_f)';%添加噪声
max_num=max(abs(x_sin));
x_sin=x_sin/max_num;%归一化信号
audiowrite('indate.wav',x_sin,fs);%生成含噪声文件，供c程序滤波
```

添加低频噪声，并取一帧分析：

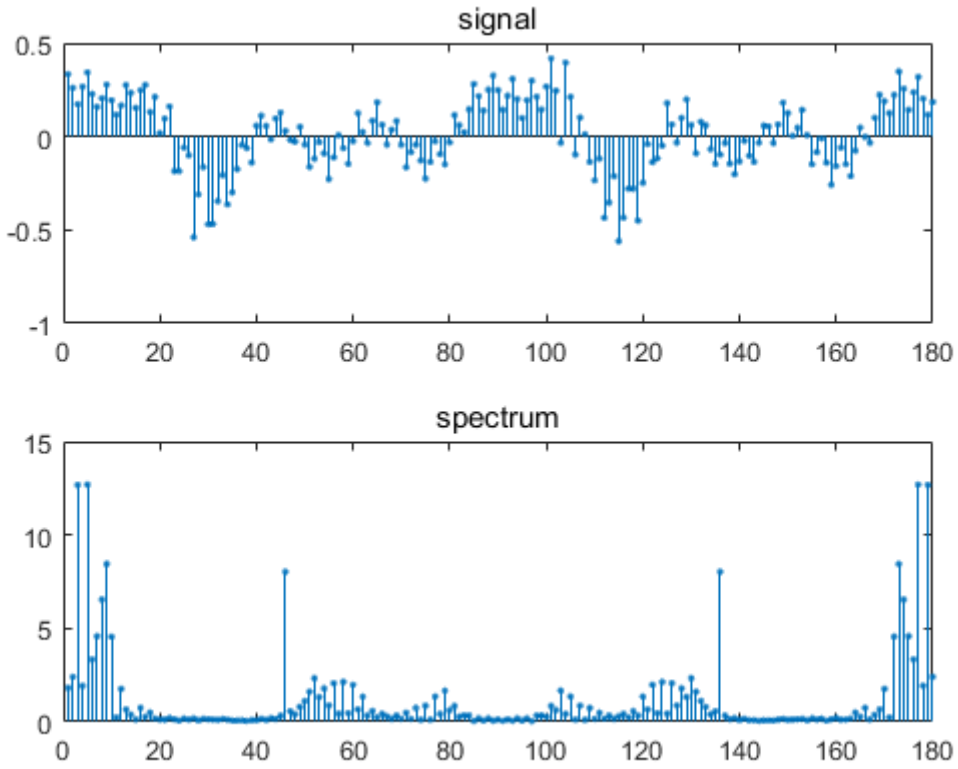


与前一步的原始信号对比，低频部分一根谱线明显变高，滤波后：

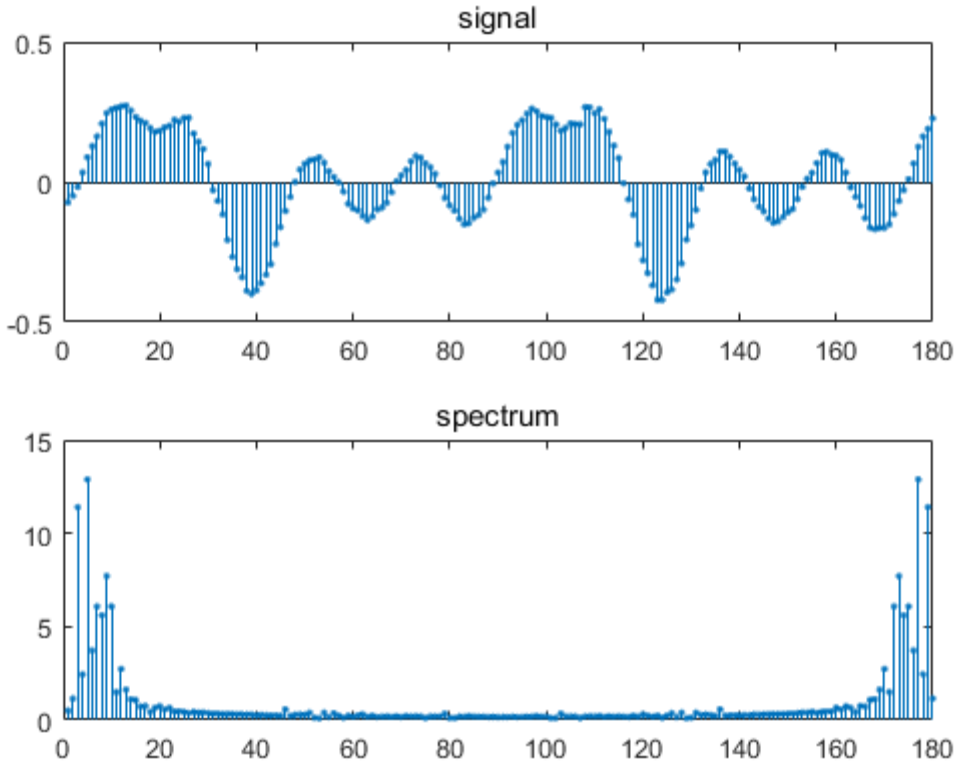


新增高峰仍然存在

添加高频噪声：

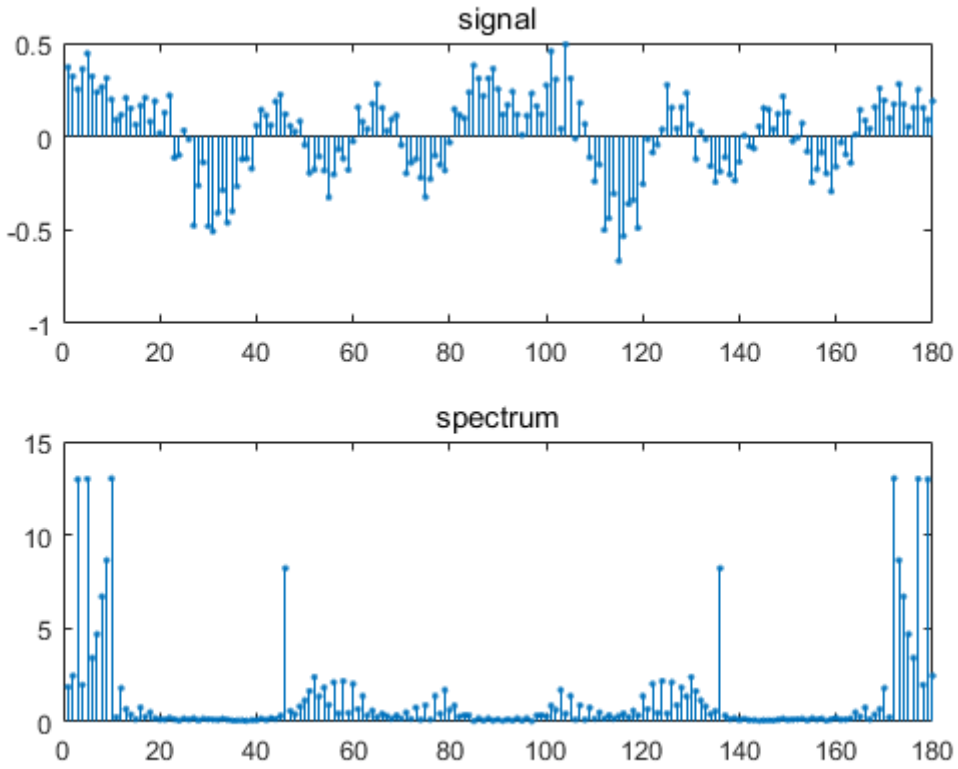


滤波后：

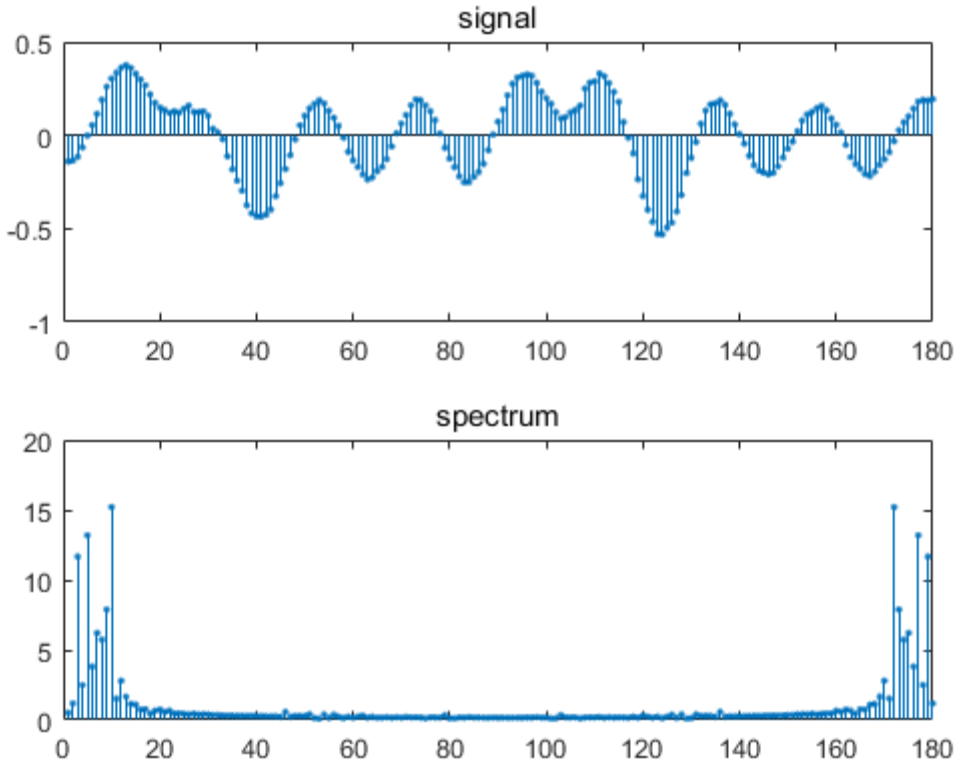


添加高频噪声后新增一根高频谱线，信号也变得更曲折，滤波后新增的高频谱线消失，信号也变得平滑。

同时添加高频和低频噪声：



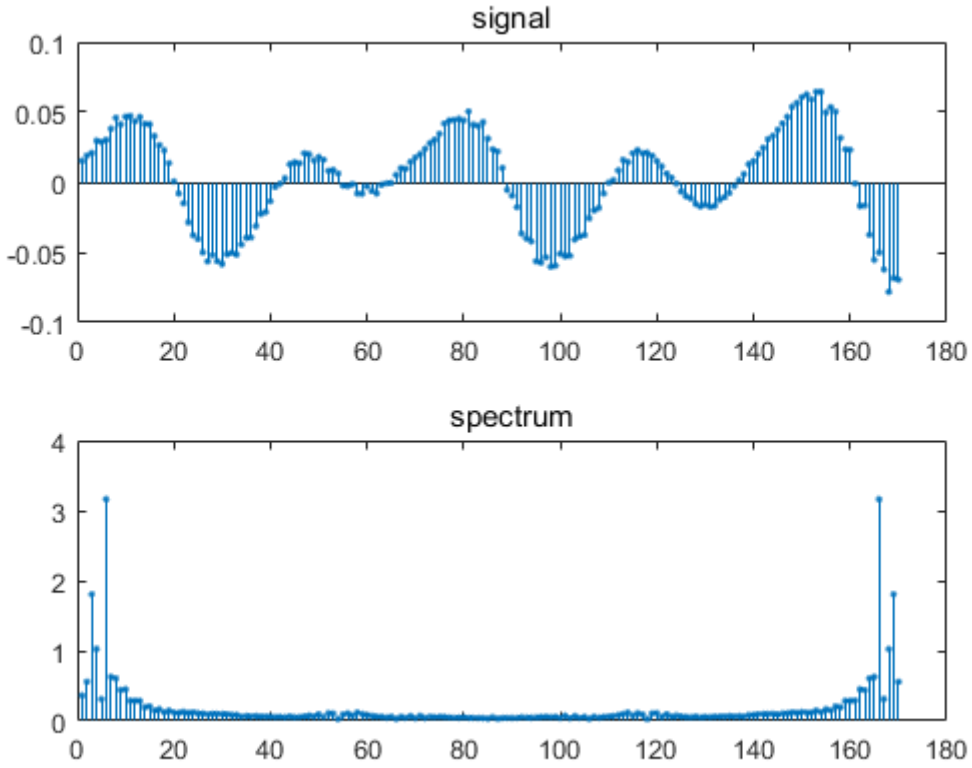
滤波后：



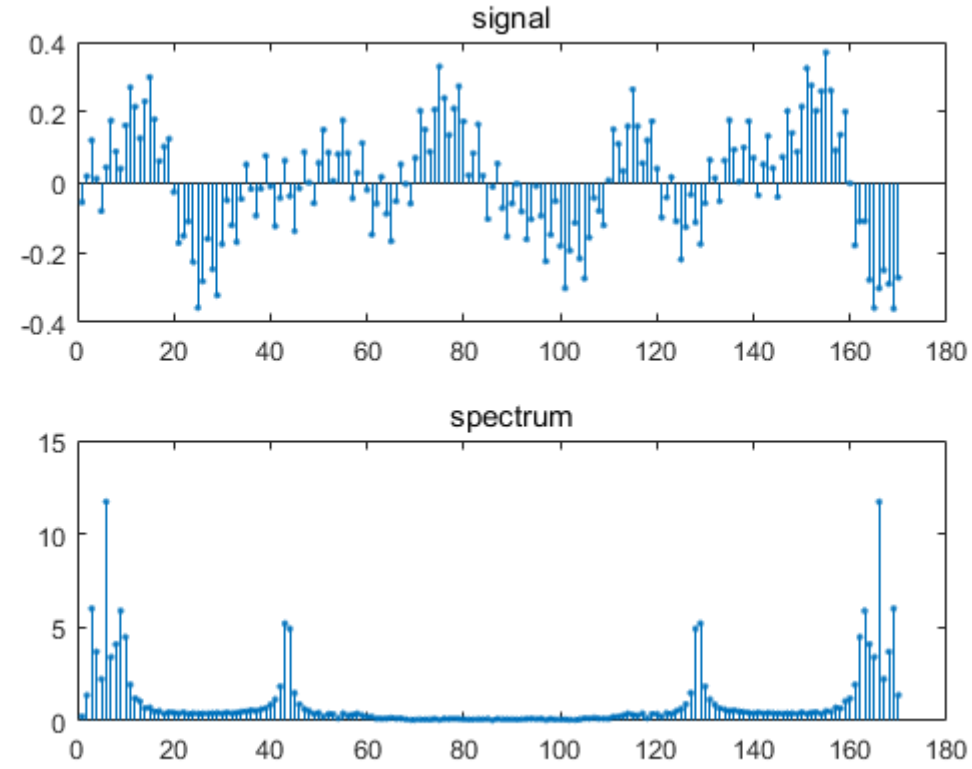
效果为前两个单独噪声的综合

3、正弦信号的采样

由于前面的实验帧长度取 180 个点，采样率 8000hz，噪声分别为 400hz 和 2000hz，所以都满足正弦信号采样要求，若取长度 170 个点，则不满足取完整周期要求，原信号：

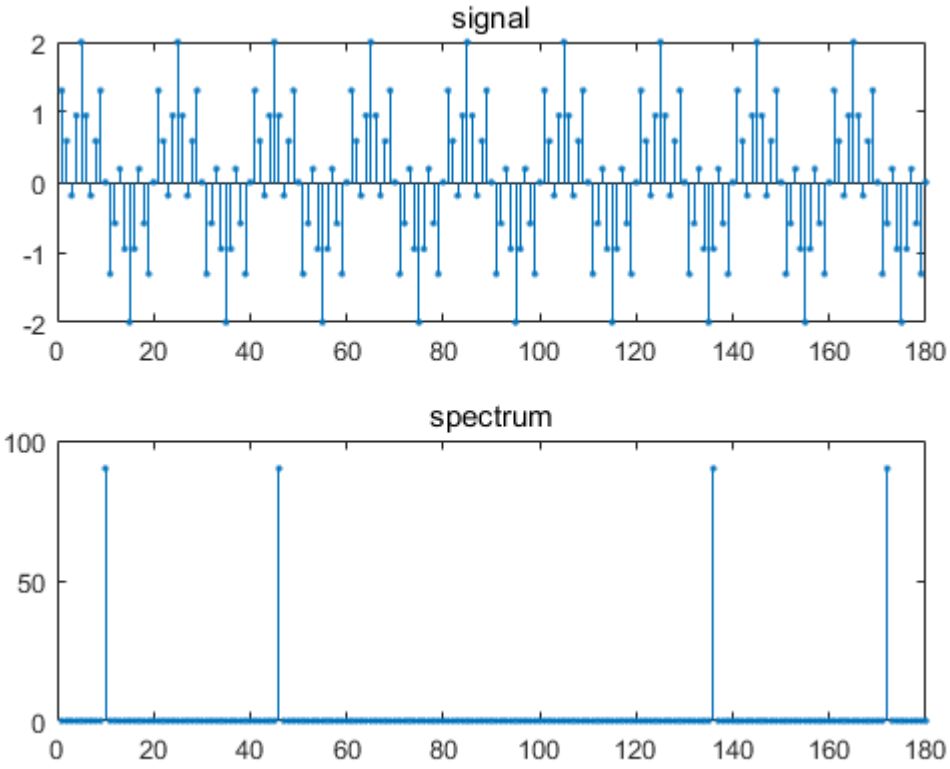


加噪声：

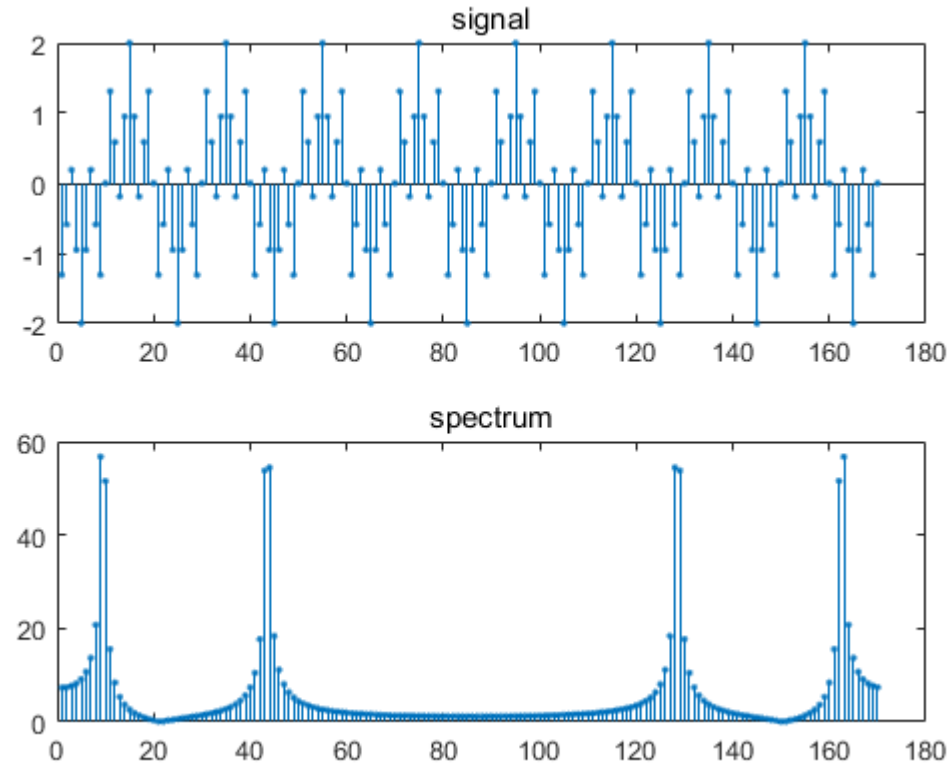


可以观察到高频和低频新增谱线为一丛而不是单根，信号确实如预期发生了泄露。

单独对正弦信号分析将能看到更明显效果：
取 180 个点（完整周期）：



取 170 个点（不完整）：



发生明显泄露。

实验心得

通过本次实验，我学到了用 Q 格式数处理数据的方法，掌握了分段卷积的舍去保留，学习了正弦信号采样规则对实验分析的影响，并通过对音频信号的滤波与加噪，亲身实践并体会到了信号处理工作对实际生活的影响。

附完整 MATLAB 程序：

```
[x,fs] = audioread('bluesky1.wav');
h=[0.01218354,  -0.009012882,  -0.02881839,  -0.04743239,  0.04584568,  -0.008692503,  0.06446265,
0.1544655,  0.2289794,  0.257883,  0.2289794,  0.1544655,  0.06446265,  -0.008692503,  -0.04584568,  -
0.04743239,  -0.02881839,  -0.009012882,  0.01218354];

frame=80;%帧号，取一帧数据分析
len=180;%帧长度

n=(frame-1)*len:frame*len-1;%对原始信号取一帧做频谱分析
x_frame=x(n);
figure('name','bluesky1','NumberTitle','off');
subplot(2,1,1),stem(x_frame,'.');title('signal');
subplot(2,1,2),stem(abs(fft(x_frame,len)), '.');title('spectrum');

fl=400;%低频
fh=2000;%高频
k=0.1*max(abs(x));%噪音幅度
n=1:length(x);
noise_l=sin(2*pi*fl/fs.*n);%低频噪音
noise_h=sin(2*pi*fh/fs.*n);%高频噪音
x_sin=x+k*(noise_l+noise_h)';%添加噪音
max_num=max(abs(x_sin));
x_sin=x_sin/max_num;%归一化
audiowrite('indate.wav',x_sin,fs);%生成含噪音的文件，供 c 程序滤波

n=(frame-1)*len+1:frame*len;
x_sin_frame=x_sin(n);%取一帧
figure('name','noise indate','NumberTitle','off');%含噪音信号的频谱分析
subplot(2,1,1),stem(x_sin_frame,'.');title('signal');
subplot(2,1,2),stem(abs(fft(x_sin_frame,len)), '.');title('spectrum');

figure('name','h','NumberTitle','off');%滤波器频谱分析
subplot(2,1,1),stem(h,'.');title('signal');
subplot(2,1,2),stem(abs(fft(h,length(h))), '.');title('spectrum');

%%
%C 程序滤波后再运行
[y,fs] = audioread('outdate.wav');%c 语言程序滤波的结果
n=(frame-1)*len+1:frame*len;
y_frame=y(n);%取一帧
figure('name','outdate','NumberTitle','off');%滤波结果的频谱分析
subplot(2,1,1),stem(y_frame,'.');title('signal');
subplot(2,1,2),stem(abs(fft(y_frame,len)),'.');title('spectrum');
```