实践任务二报告

刘正浩 2019270103005

1. 快速傅里叶变换 FFT 的原理和计算复杂度分析

《信号与系统》中，离散时间周期信号的 FS 的分析公式为

$$a_k = \frac{1}{N}\sum_{n=<N>} x[N]e^{-jk\frac{2\pi}{N}n}$$

其中，共有 N 个 FS 系数需要计算。对于每个 FS 系数，需要进行 N 次复数乘法运算和 N-1 次复数加法运算。显然，复数乘法运算的计算量远大于复数加法运算。由此可知，一个周期为 N 的 FS 的计算的复杂度为$O(N^2)$。

FFT（Fast Fourier Transform）是 DFT（Discrete-time Fourier Transform）的一种快速计算的方法，它通过利用 DFT 中系数 W 的性质——对称性、周期性、可约性，来对 DFT 进行分解。

在进行时间抽取 FFT 时，要将整个序列按奇数项和偶数项分为两半序列并分别求和。偶序列为 x(0),x(2),x(4)…，奇序列为 x(1),x(3),x(5)…。这样，x(N)的 N 点 DFT 可以写为：

$$X(k)=\sum_{n=0}^{\frac{N}{2}-1}x(2n)W_N^{2nk}+\sum_{n=0}^{\frac{N}{2}-1}x(2n+1)W_N^{(2n+1)k}$$

考虑到W_N的性质，即

$$W_N^2=[e^{-j\frac{2\pi}{N}}]^2=e^{-j\frac{2\pi}{\frac{N}{2}}}=W_{\frac{N}{2}}$$

因此有

$$X(k)=\sum_{n=0}^{\frac{N}{2}-1}x(2n)W_{\frac{N}{2}}^{nk}+W_N^k\sum_{n=0}^{\frac{N}{2}-1}x(2n+1)W_{\frac{N}{2}}^{nk}$$

或者可以写为

$$X(k)=Y(k)+W_N^kZ(k)$$

由于$Y(k)$与$Z(k)$的周期为$\frac{N}{2}$，并且利用W_n的对称性和周期性，即

$$W_N^{k+\frac{N}{2}}=-W_N^k$$

可以得到

$$X(k+\frac{N}{2})=Y(k)-W_N^kZ(k)$$

经过查阅资料可以知道，蝶形算法的时间复杂度减少为$O(NlogN)$。

附：蝶形算法 MATLAB 源码与验证

（1）源码：

```
function[Xk]=my_fft(Xn,N)  %函数：快速傅里叶变换，传入数列Xn 和长度N

 %如果没有输入N，将N 记为Xn 的长度
 if nargin<2
   N=length(Xn);
 end

 %如果传入的N 大于Xn 的长度，Xn 补零（0 不影响fft 的结果）
 Xn=[Xn,zeros(1,N-length(Xn))];

 %计算蝶形运算的级数M
 tempbinary=dec2bin(N);  %将数字转换为二进制数
 M=length(tempbinary(2:end)); %级数M 就是二进制数的位数
 tempdata=zeros(N,M+1); %中间操作存储矩阵

 %对输入进行排序
 for n=0:N-1
   tempbinary=dec2bin(n,M);
   tempbinary=rot90(tempbinary,2);
   tempdata(n+1,1)=Xn(bin2dec(tempbinary)+1);%数组都从1 开始计数
 end

 %开始fft
 for m=0:M-1
 %产生W
   for r=1:2^m
     W(r)=cos((pi/2^m)*(r-1))-1j*sin((pi/2^m)*(r-1)); %W 数组从1 开始存数
   end
   for i=1:r
     for groupseq=1:N/(2^(m+1))
       p=i+(groupseq-1)*2^(m+1);%用组号和W 因子号确定P 的坐标
       q=p+2^m;
       tempdata(p,m+2)=tempdata(p,m+1)+W(i)*tempdata(q,m+1);%计算蝶形单元
       tempdata(q,m+2)=tempdata(p,m+1)-W(i)*tempdata(q,m+1);
     end
   end
 end

 Xk=tempdata(:,M+1);
 m=size(Xk);
```

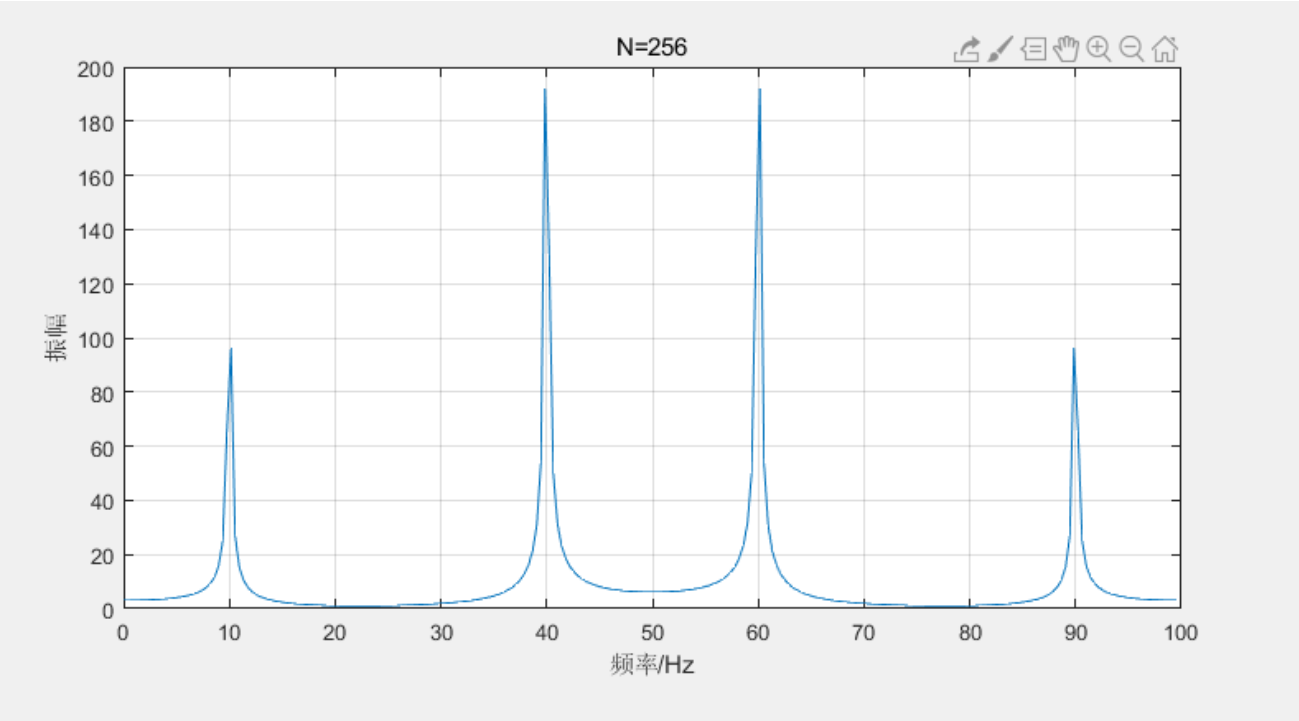
```
  m1 = size(Xn);
  if m ~= m1
    Xk = Xk';
  end
end
```

（2）验证：

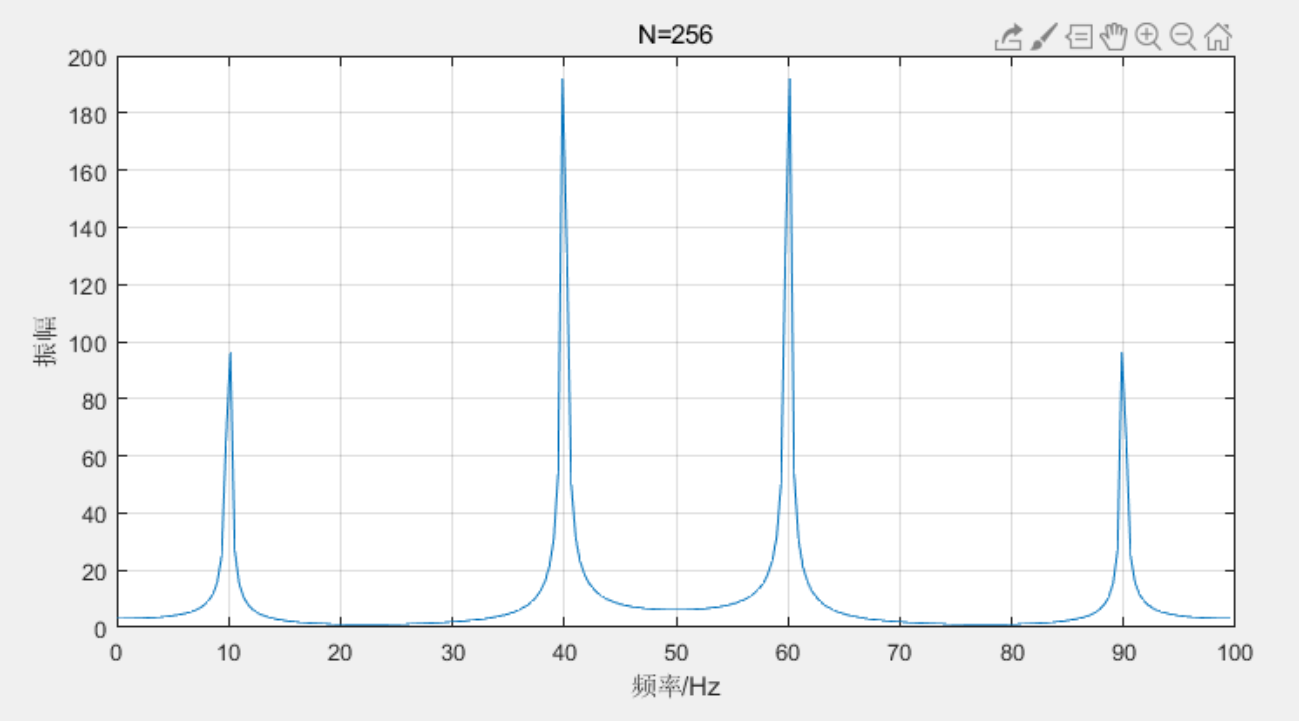
设需要进行 FT 操作的信号为 $x(t)=\sin(20\pi t)+2\sin(80\pi t)$，利用自己编写的蝶形算法计算频谱。计算的源码如下：

```
fs = 100;%设置采样频率为100
N = 256;%设置数据点数为256

%得到时间序列
n = 0 : N - 1;
t = n / fs;

%待处理信号
x = sin(2 * pi * 10 * t) + 2 * sin(2 * pi * 40 * t);
%对信号进行fft
y = my_fft(x, N);

%求fft后的振幅
mag = abs(y);

%频率序列
f = n * fs / N;

%画振幅-频率图
subplot(2,2,1),plot(f,mag);
xlabel('频率/Hz');
ylabel('振幅');
title('N=256');
grid on;
```

这是用自己编写的蝶形算法进行 FFT 的结果：



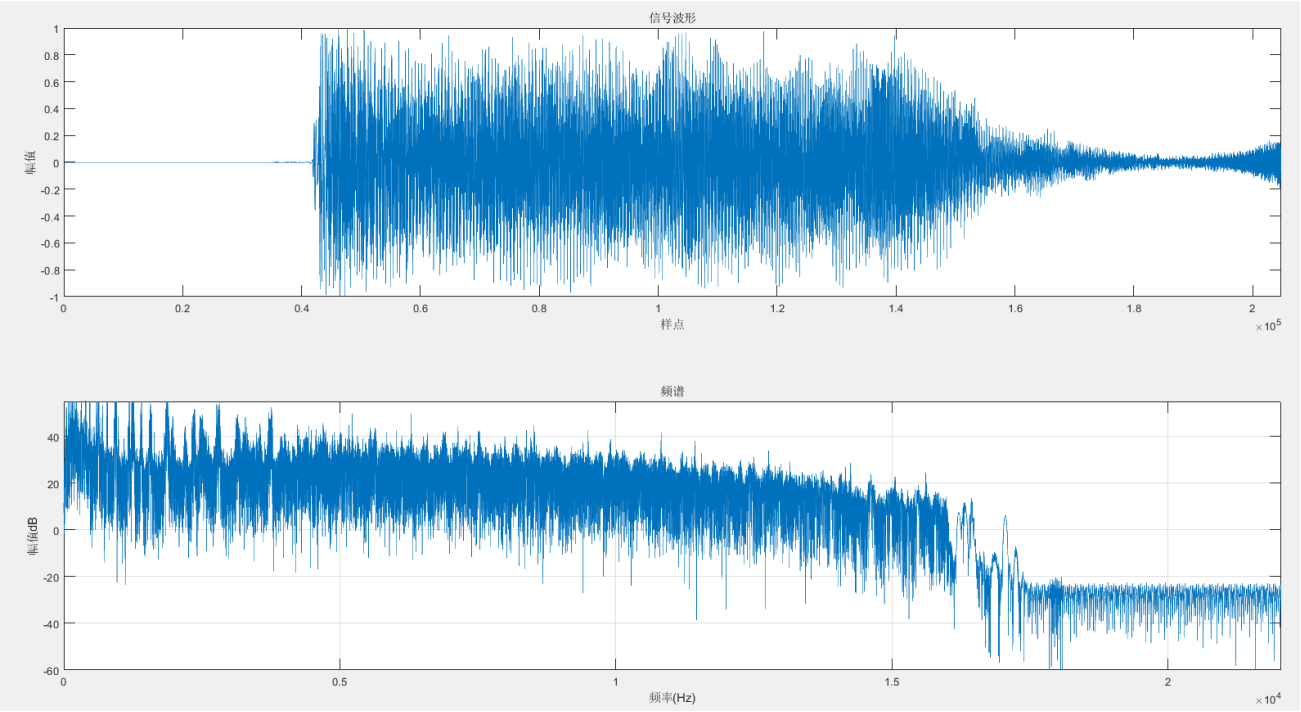
这是 MATLAB 自带的 FFT 函数计算的结果：



可以看出，本报告中给出的蝶形算法的效果与 MATLAB 自带的 FFT 函数效果基本一致。

2. 基于 FFT 的数字语音信号的谱分析

本实验旨在通过画出波形与通过 FFT 形成的频谱来分析一段音乐。在这里我选用了《Государственный гимн СССР》（前苏联国歌）作为例子。这首音乐作为一首恢弘的合唱曲，具有较宽的频谱（因为乐器的种类与人声部的高低较多且差别较大）和较强的响度。



3. 基于 FFT 的音调的频率分析

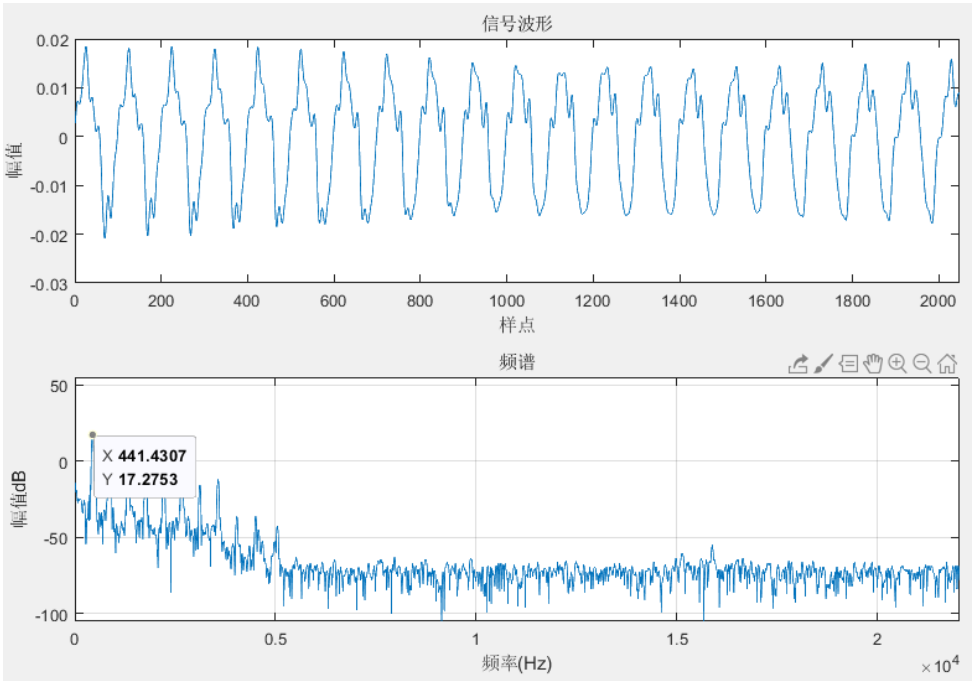
本实验旨在通过 FFT 分析钢琴的七个基本音（do、re、mi、fa、so、la、si），获得七个音的频率，并与标准频率进行比较，验证该频率值与理论值一致。

源码如下：

```
%读取钢琴音高文件
[xx,fs]=audioread('la.wav');

%单独取出读到的数据，并计算数据个数和信号持续时间
x=xx(:,1);
N=length(x);  %数据个数
time=(0:N-1)/fs;  %信号持续时间

%画出读到的数据
plot(x);
```

```
%采样，进行fft
M = 2048;
nfft = 8192;
win = hanning(M);
freq = (0 : nfft/2) * fs / nfft;
y = x(9001 : 9000+M);
y = y - mean(y);
Y = fft(y.*win,nfft);

%画图
subplot 211;plot(y);
xlim([0 M]);
title('信号波形');
xlabel('样点');
ylabel('幅值');
subplot 212;
plot(freq,20*log10(abs(Y(1:nfft/2+1))));
grid;
axis([0 max(freq) -60 55]);
title('频谱');
xlabel('频率(Hz)');
ylabel('幅值 dB');
```

下面是 1a 音的信号波形与频谱：



可以看到，钢琴发出的 1a 音并不是简单的正弦信号，而是波形更复杂的周期信号；频谱峰值对应的频率为 441.4307Hz，与标准音 1a 的 440Hz 基本一致，说明这里的频谱分析是成功的。

用同样的方法分析其他六个基本音以及小提琴的，得到的结论与分析 1a 的结论一致，都与理论值基本相同。

4. 讨论

（1）在进行 FT 的最大频率的确定时，需要先粗略了解信号大致的频率分布。这样做的目的有两个：一是因为采样时必须遵循采样定理，否则得到频谱后无法恢复原信号；二是因为如果频率范围选择不恰当，可能会使 FT 后得到的频谱分布不便于观察。

（2）在画图时要灵活设置横纵坐标的标度，采用均匀标度还是采用对数标度取决于信号的分布情况和我们需要详细观察的范围。