信号处理原理Matlab大作业

2018011324

尤艺霖

计82

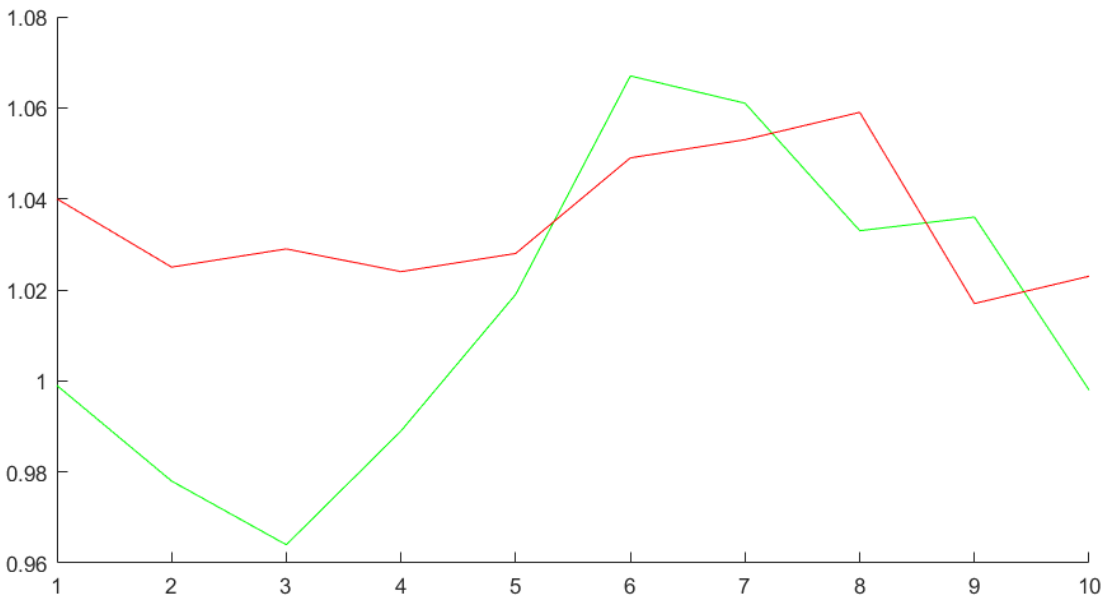
1.按键识别

文件内容：

- wav目录 存放用于测试的音频文件，由于手机的问题采用了他人的音频
- fft_analysis.m 使用fft的算法来分析音频
- goertzel.m 实现goertzel算法来分析音频
- test.m 测试两种算法

实验结果：

经过测试，两种做法均能正确预测所有的按键结果。为了对比两种做法的效率，将每种做法在每个按键上运行了100次来进行测试，最终结果如下图。



图中红线为fft方法的开销，绿线为geortzel方法的开销，可以看出大部分时候，geortzel算法的时间效率是优于fft的。

2.序列卷积

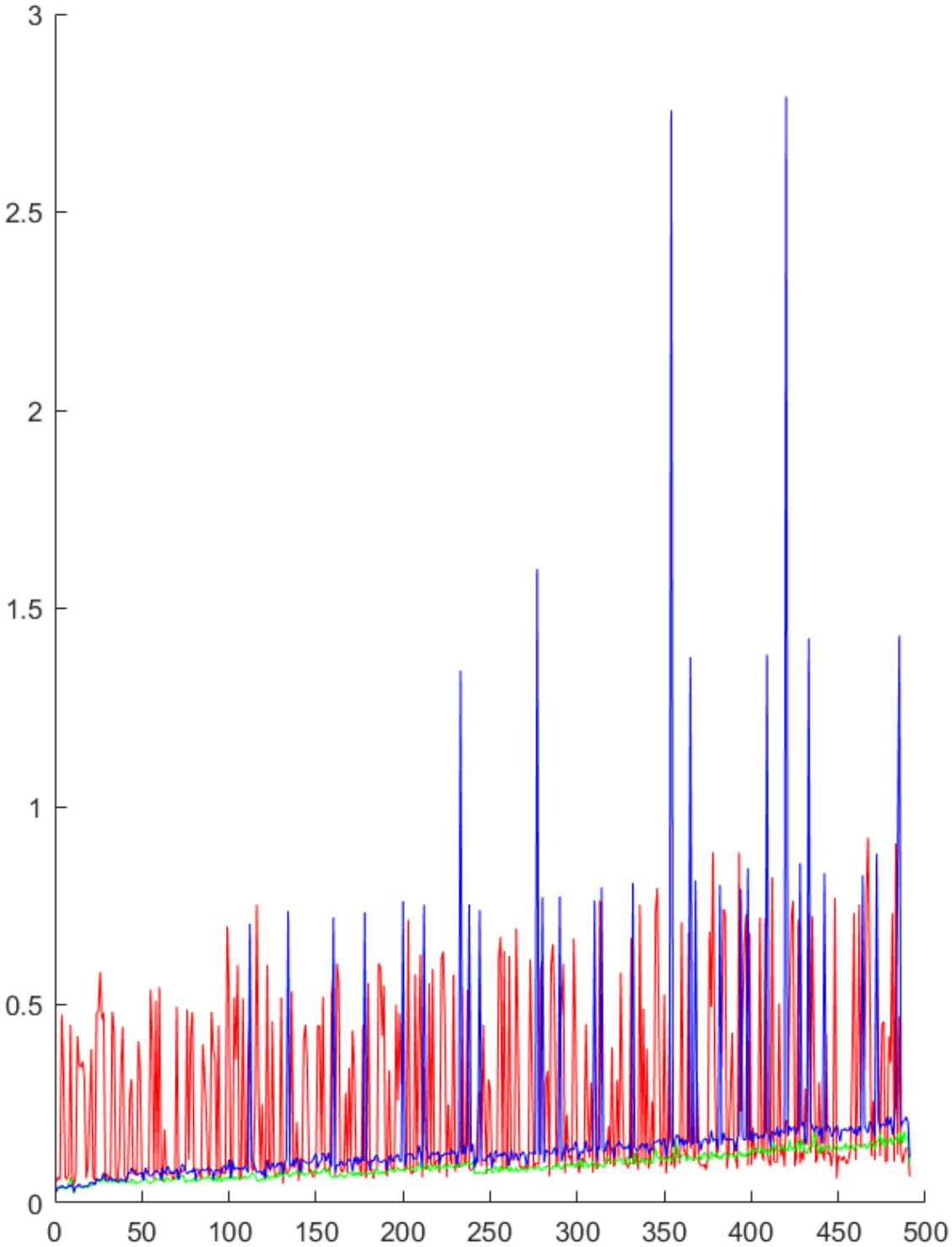
文件内容：

- brute_force.m 实现双重循环的朴素卷积
- matlab_fft.m 使用自带的快速傅里叶变换实现卷积
- overlap_add.m 使用overlap-add算法实现卷积
- overlap_save.m 使用overlap-save算法实现卷积

- test.m 测试以上三个程序
- figure.bmp 展示运行结果的图表

实验结果：

考虑到overlap-add/save算法主要是为了一个较长序列和一个较短滤波器的卷积而设计的。故选择的测试数据规则如下：一个序列长度始终为500000，另一个则以1000的间隔从10000遍历到500000。以方便对比各种算法在不同类型数据范围下的效果。由于朴素做法时间复杂度在任意情况下都不会比剩余三者更优，而且在较大数据范围下耗时太大，故没有对朴素做法进行测试。



上图中红线表示fft的运行时间，绿线表示overlap-save的运行时间，蓝线表示overlap-add的运行时间。可以观察到在图的左侧，当两个序列长度相差较大时，overlap-add/save算法效率是高于简单的fft的；随着短序列长度的增加，很快就由于额外开销而劣于简单的fft算法，但没有特别大的差距；其中蓝线又表明overlap-add在个别情况下会有较低的效率，绿线则表明overlap-save发挥稳定且效率出色。

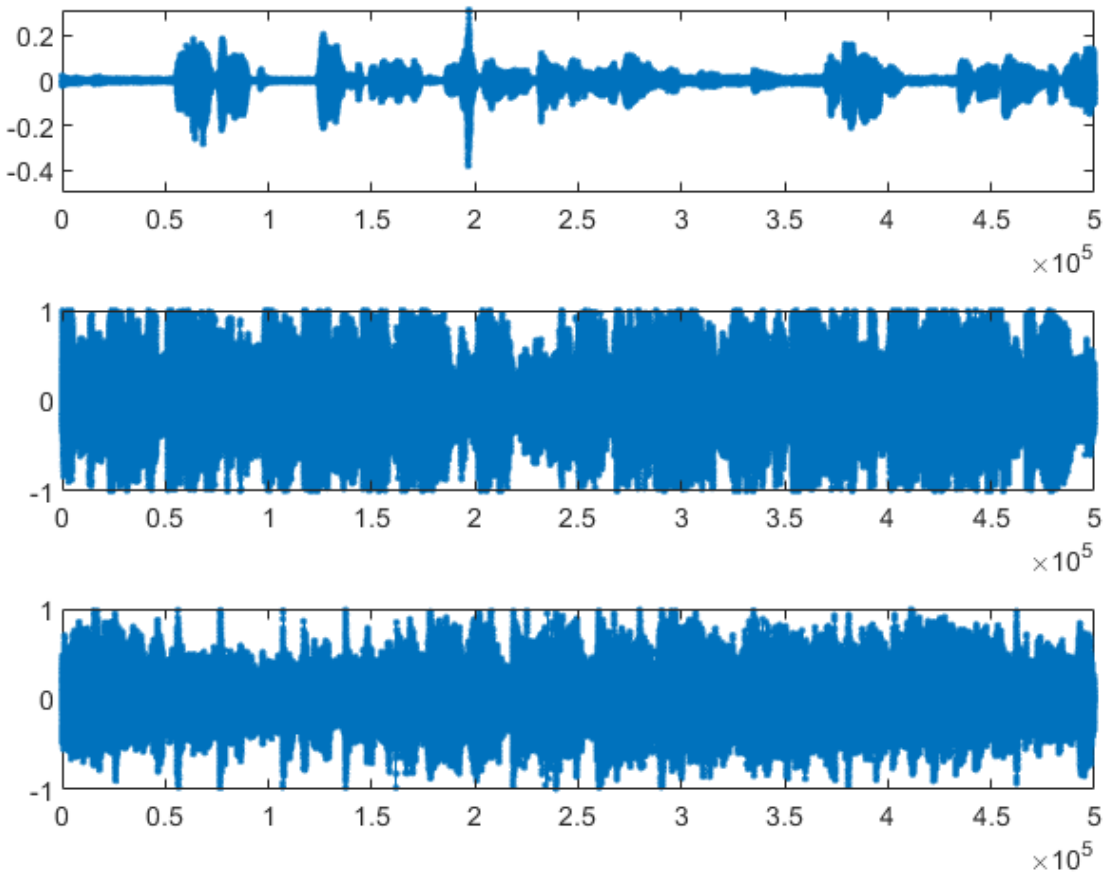
3.频分复用

文件内容：

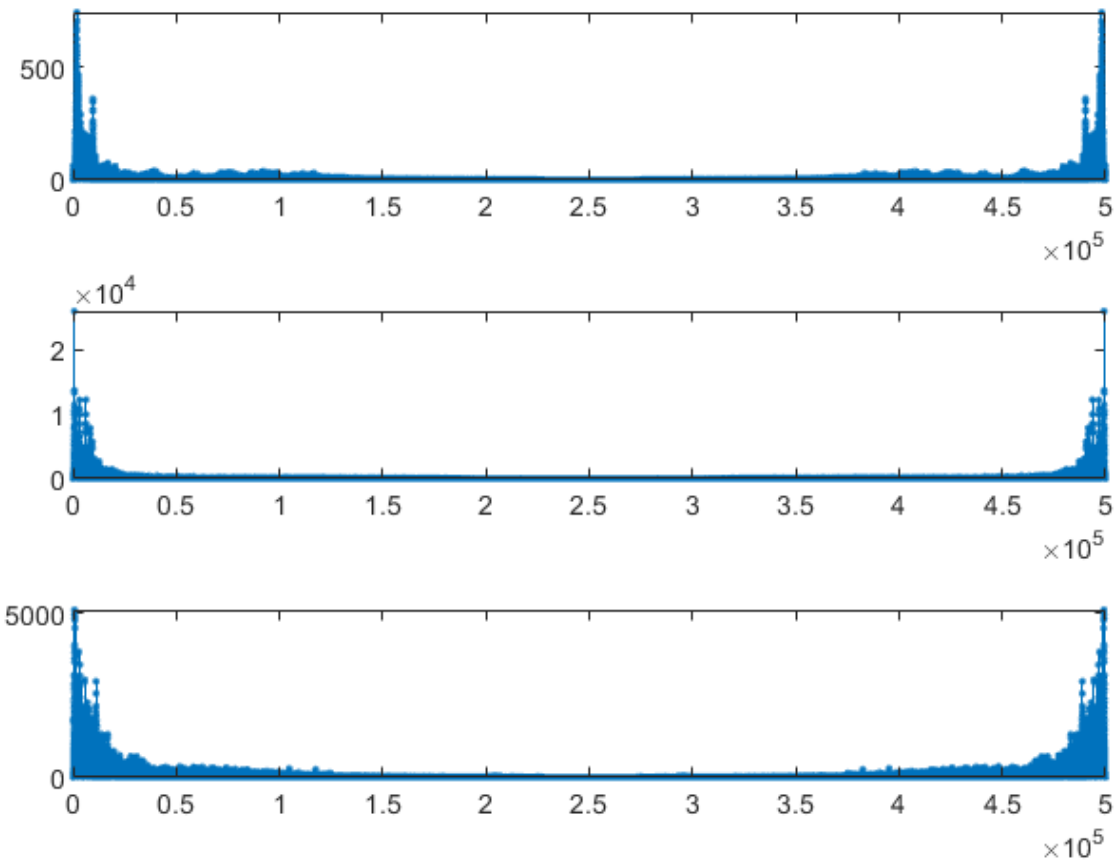
- wav目录 存放用于频分复用的音频
- fdm.m 实现频分复用算法

实验结果：

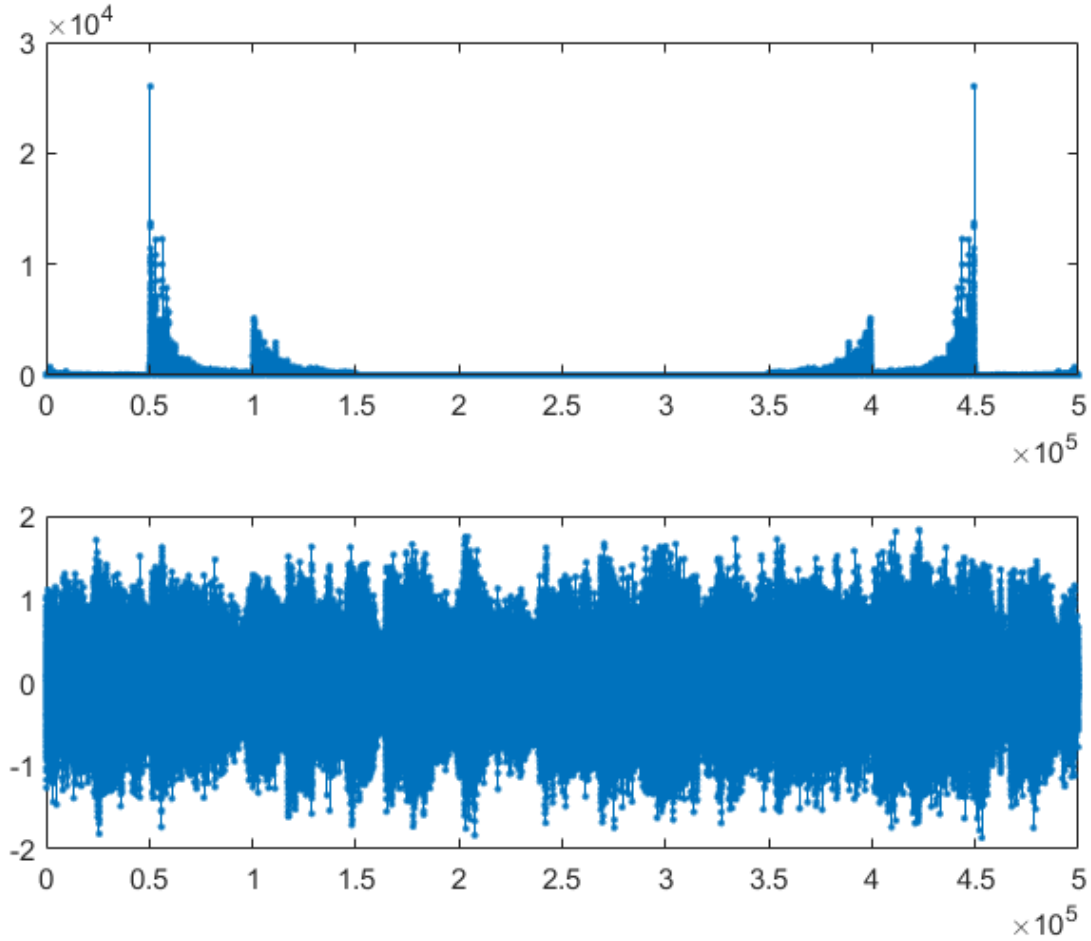
首先将三个音频做处理，从mp3格式转为wav格式，并保留500000个采样点的长度，有时域如下：



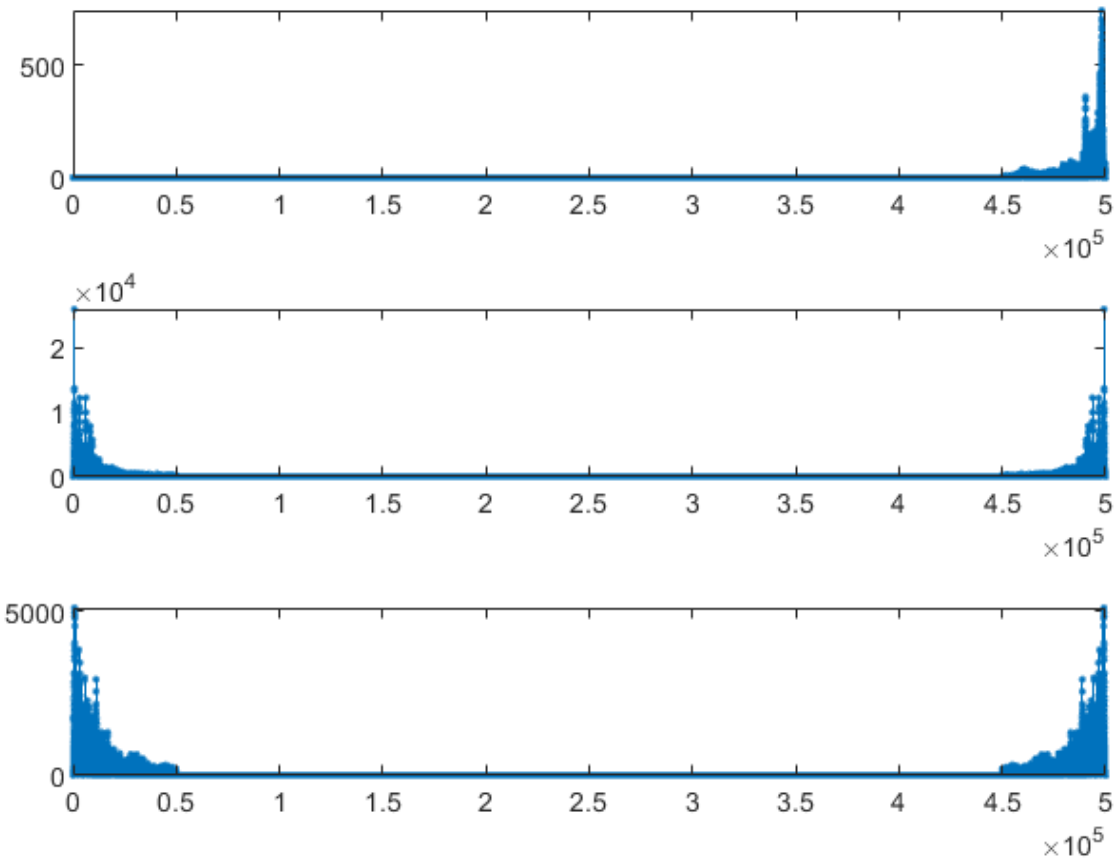
然后进行fft获得频域如下：



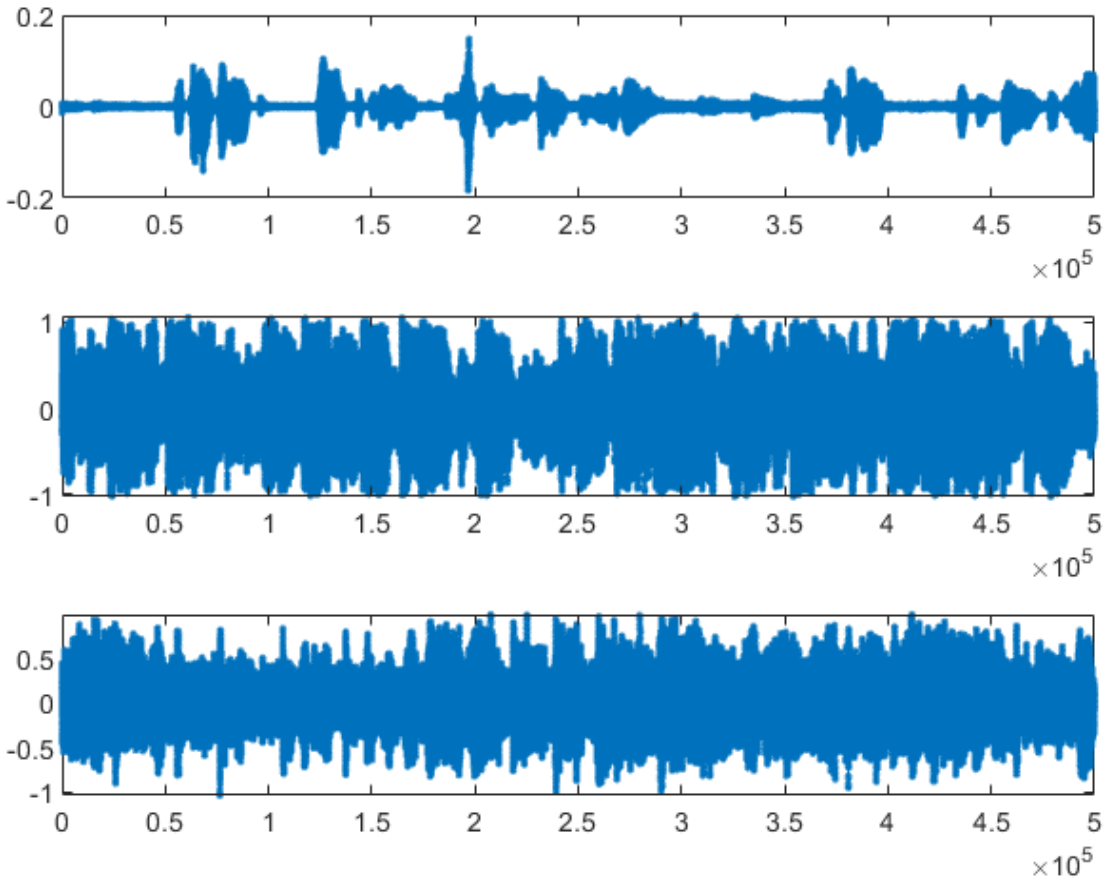
随后将这三个频域的高频部分组合到一起，形成一个新的频谱，并转换到时域。如下：



然后再将这些频谱分量分开到原先的三个分量的状态：



然后进行ifft获得时域信号：



从频谱的角度看，频分复用导致了低频分量的损失，最终在时域上导致了一定的偏差，在耳机里听到的效果体现为一些奇怪的杂音。但总体上基本能很好的恢复出原先的信号。