**稀疏傅里叶变换**

#### 实验目的

对于稀疏信号，以亚线性复杂度求出频谱。

#### 实验环境

我采用要求的matlab编写代码。MATLAB是美国 MathWorks 公司出品的商业数学软件，用于算法开发、数据可视化、数据分析以及数值计算的高级技术计算语言和交互式环境。我使用的是matlab R2023b和R2023a。

#### 实验原理

实际应用中，许多信号的频谱具有稀疏性，即只在个别频点上有非零值，因而可以对频域信号进行B倍降采样后处理，频域降采样可以通过时域混叠实现，即论文中的Claim 3.7。假设信号长度原始为N，稀疏度为k（即正负频共有k个非零频点），则可以将复杂度从降为。之后由于降采样未必保证可以抽到非零频点，因而需要多次计算采多组频点。假设重复实验L次，则DFT的复杂度为。

其中采多组频点的实现方式是将信号进行时域重排，即。然而如果全部重排就会发现仅仅一次重排的复杂度就有，为了节约重排以及后面处理的计算量，采用加窗的方法，即只保留重排后的W个值，这样重排的复杂度降低为，DFT的复杂度也降低为，所以总复杂度为。之后就每次采样获得的个候选频点，同时采用逆映射找到它们原来的位置，并对结果进行一定放缩还原出原来的频谱。一共有个候选频点，挑选其中出现次数大于的作为最终的结果。

因而总的复杂度为，其中待确定的参数有降采样倍数B、重复次数L、截取长度W、与选取最大值个数有关的d。按照2012年论文的取法，，，，。代入我所推导的公式，复杂度为，即论文给出的结果。同时论文推导出此种算法满足保证的概率为。

而对比FFT复杂度恒为，因而在即时，SFT可以比FFT更快。即要求k较小（足够稀疏）、不是很大（通带带宽，窗足够窄），不是很小（波纹幅度，一般很小）。

综上，总体算法流程如下，其中变量名与代码中的统一，且红色、黄色、绿色分贝表示main文件、outerloop文件、innerloop文件中进行的操作。

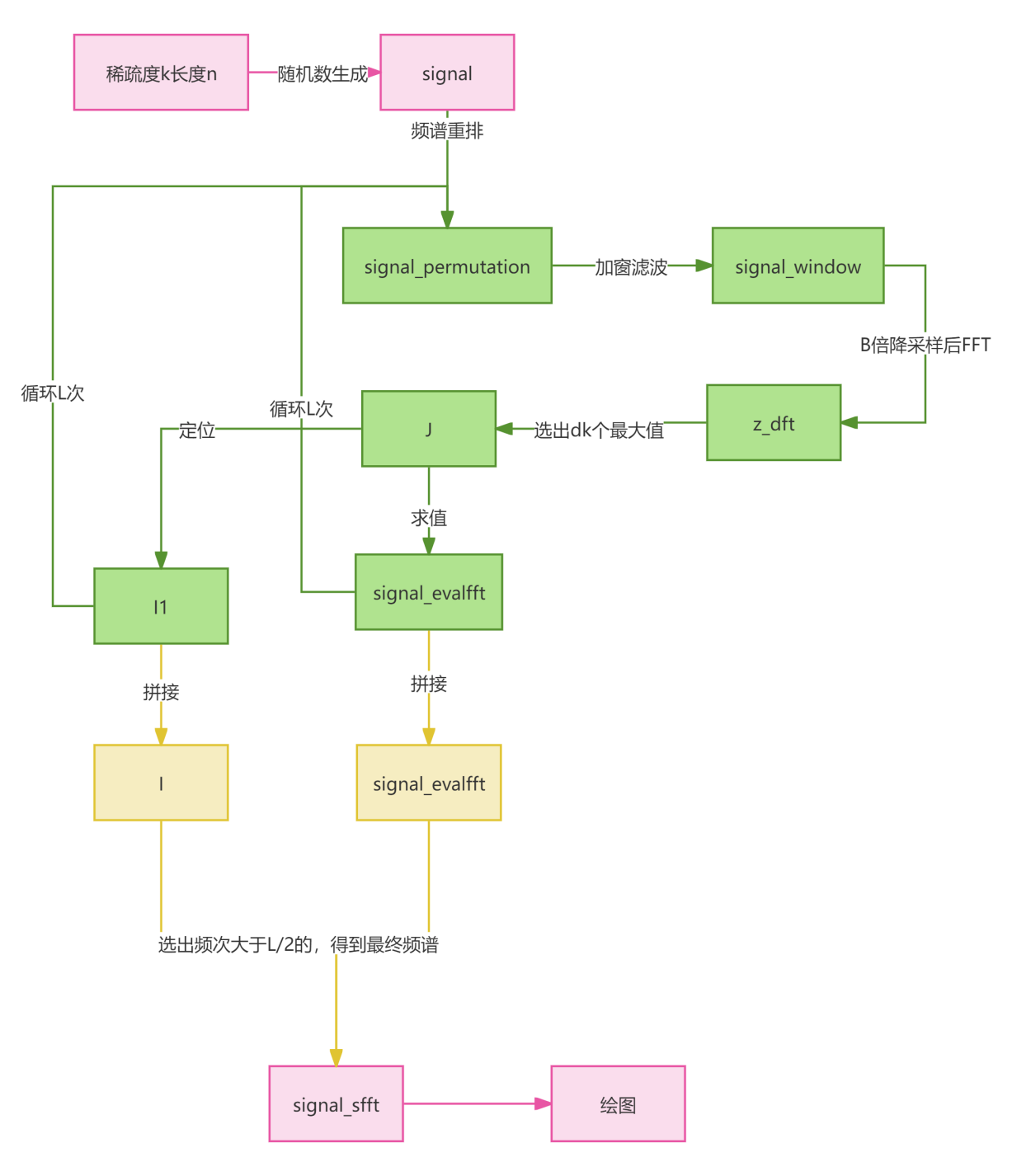


图 1 算法整体流程图

#### **实验内容**

我先根据论文完成了整体的代码，之后进行了广泛的实验和调参，这部分将放在消融实验里面介绍。

我首先以时间加随机数为随机数种子，生成频率随机、幅度随机、相位随机的规定稀疏度的信号。之后运行FFT并统计时间。然后进行SFT。SFT首先幅值参数，之后生成对应的窗。这一部分的耗时不计入总时间。之后调用定义的函数outerloop进行外循环；outerloop中调用L次innerloop。其中outerloop返回SFT的结果，innerloop返回频点坐标与频谱值。这一部分计时。由于SFT结果比较大，所以我没有直接输出，而是画图表示。

总体而言，我的代码可以比较大概率的情况下找到非零的频点，但是对于频谱的绝对大小不是很准确，我尝试了很多方法，也放在消融实验中。下面放一套结果图，参数为n=2048，k=6，L=11，B=128。时间上SFT大约是FFT的几倍到几十倍，应该是由于matlab对FFT做了充分优化，而我的代码没有很多底层优化，而且分成多个文件会增加延迟。

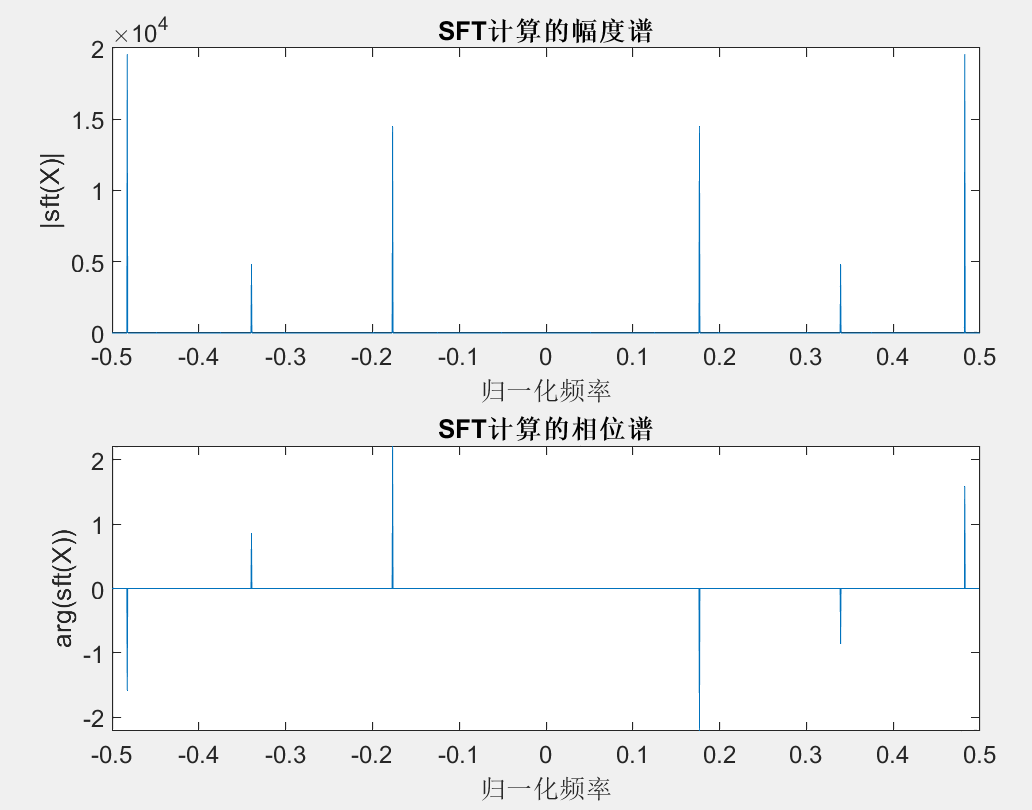


图 2 SFT结果

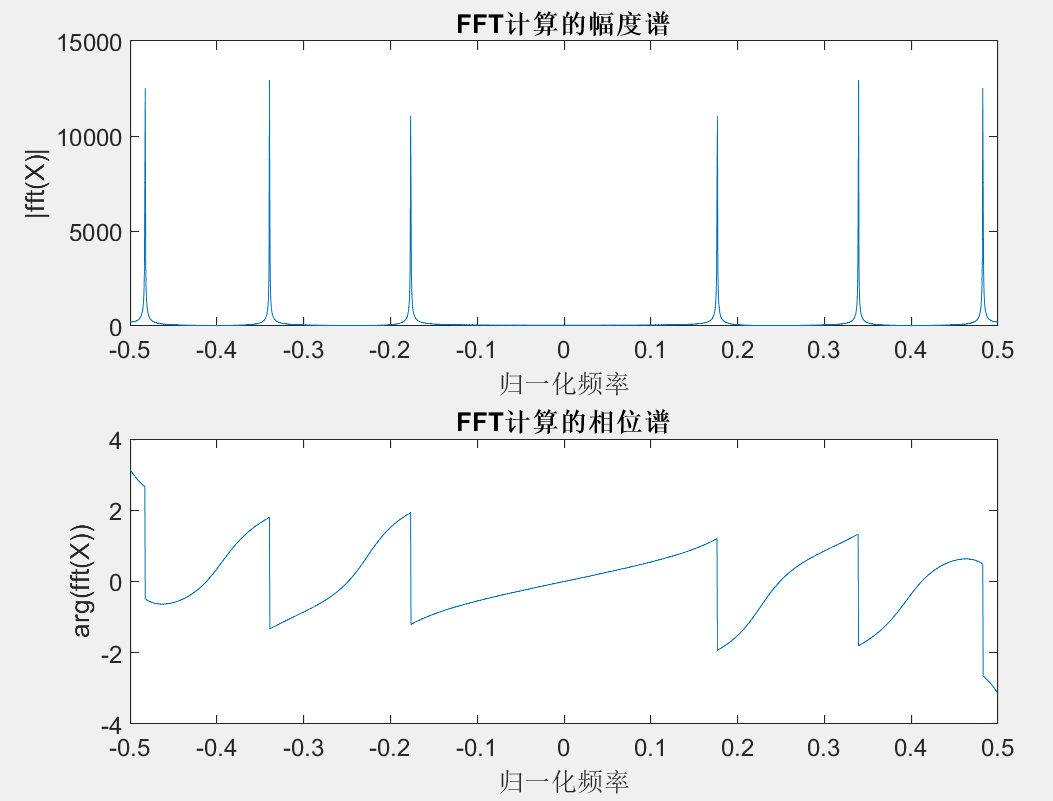


图 3 FFT结果

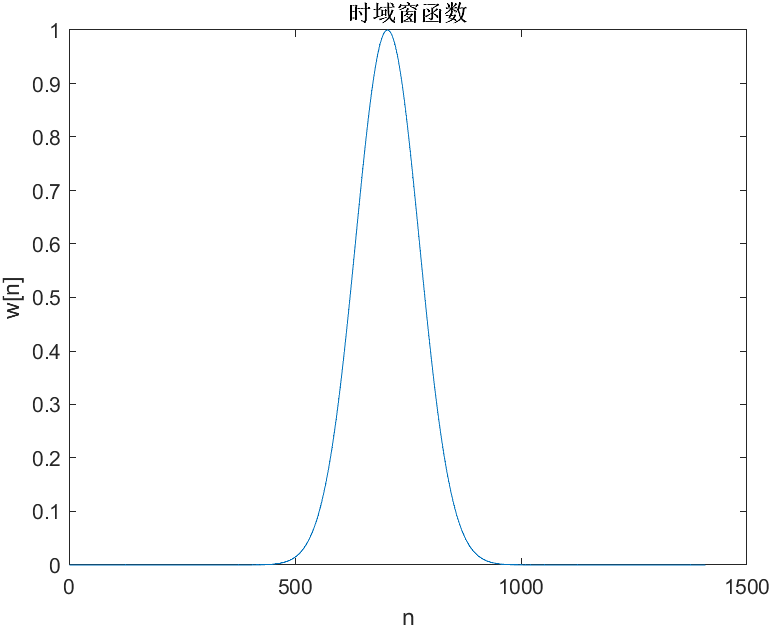


图 4 时域窗函数

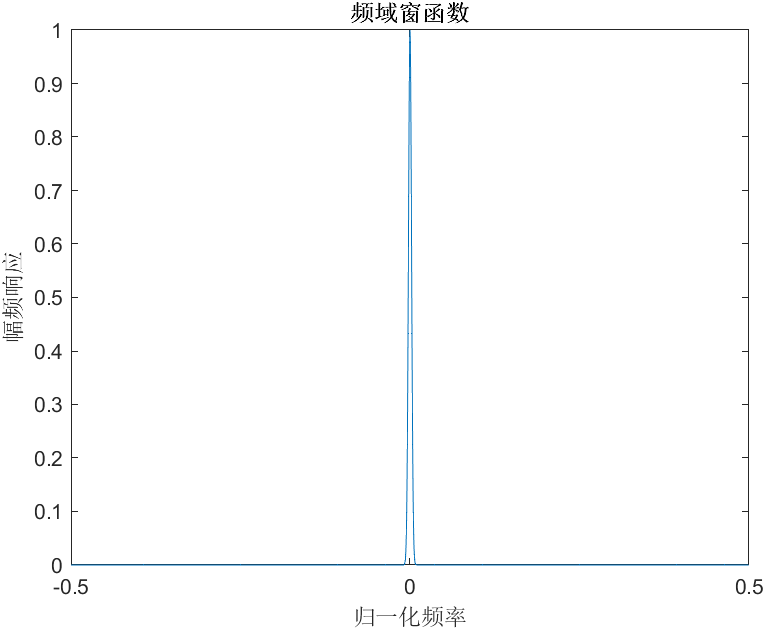


图 5 频域窗函数

#### **消融实验**

###### 窗函数的影响

实验开始我使用了频域为高斯卷方波的窗，但是像论文的图一样，通带比较宽如图6，导致SFT输出完全没有结果！分析原因是加窗之后对于重排后频谱平滑过于严重，导致即使降采样后，一个实际非零频点周围会有很多差不多大小的点，这就导致候选点的出现次数均小于L/2，于是没有点被选出。图7画出了这种情况下一次重排的FFT，从上到下依次是重排后的FFT、加窗后的FFT、降采样后的FFT。

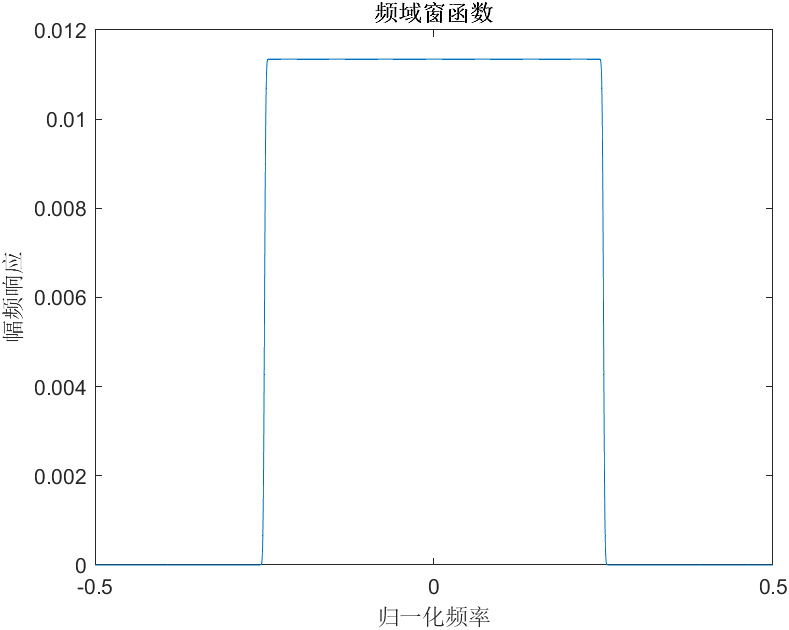


图 6 较宽的窗

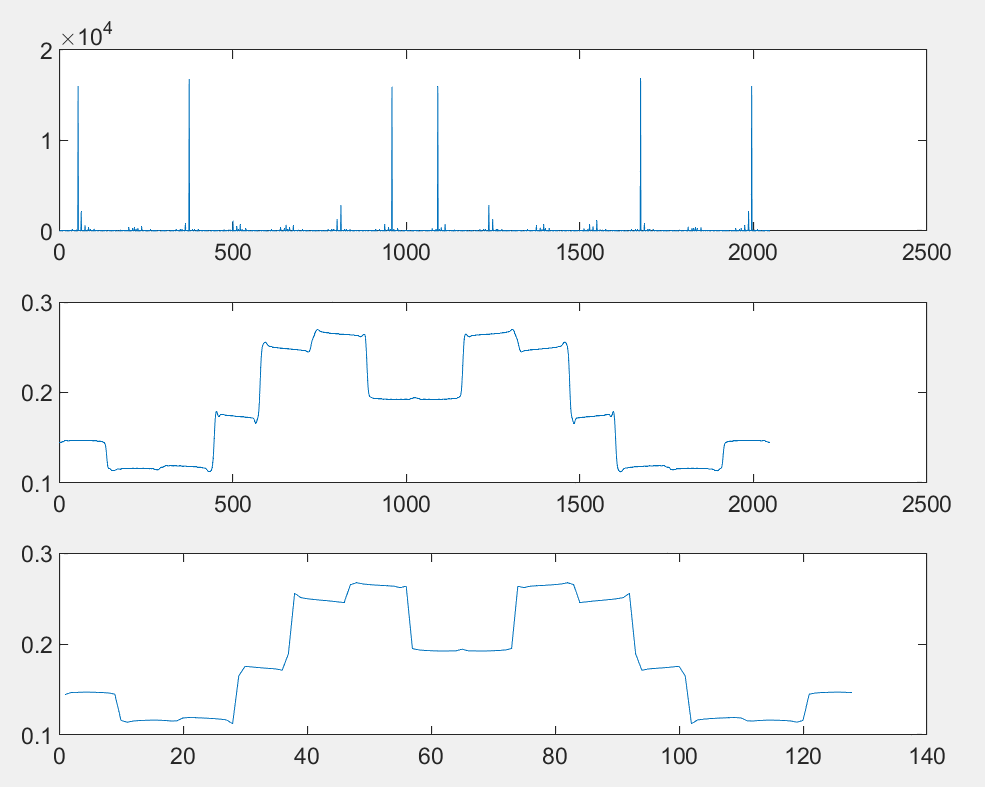


图 7 各信号的FFT

之后我有尝试了不卷高斯窗，仅使用截断的矩形窗。这种情况下也可以运行，但是效果略差于之前。

###### 重复次数L的影响

理论上来讲L的次数越大，SFT的准确率越高，但是耗时也会越大。一般而言我采取。之后我也对不同的L进行了测试。时，FFT仅仅比SFT快10倍及以下，并且极少数情况下会比SFT慢。不过在此精度下精度较难保证，往往会出现高于稀疏度个频点的结果。而时频点找的就基本上准确，但是会慢1-50倍。

###### 稀疏度k的影响

首先必须承认我的代码不如论文中的测试结果那么快和准确。然后我也测试了下图k=50的结果，FFT比SFT快了70倍左右：

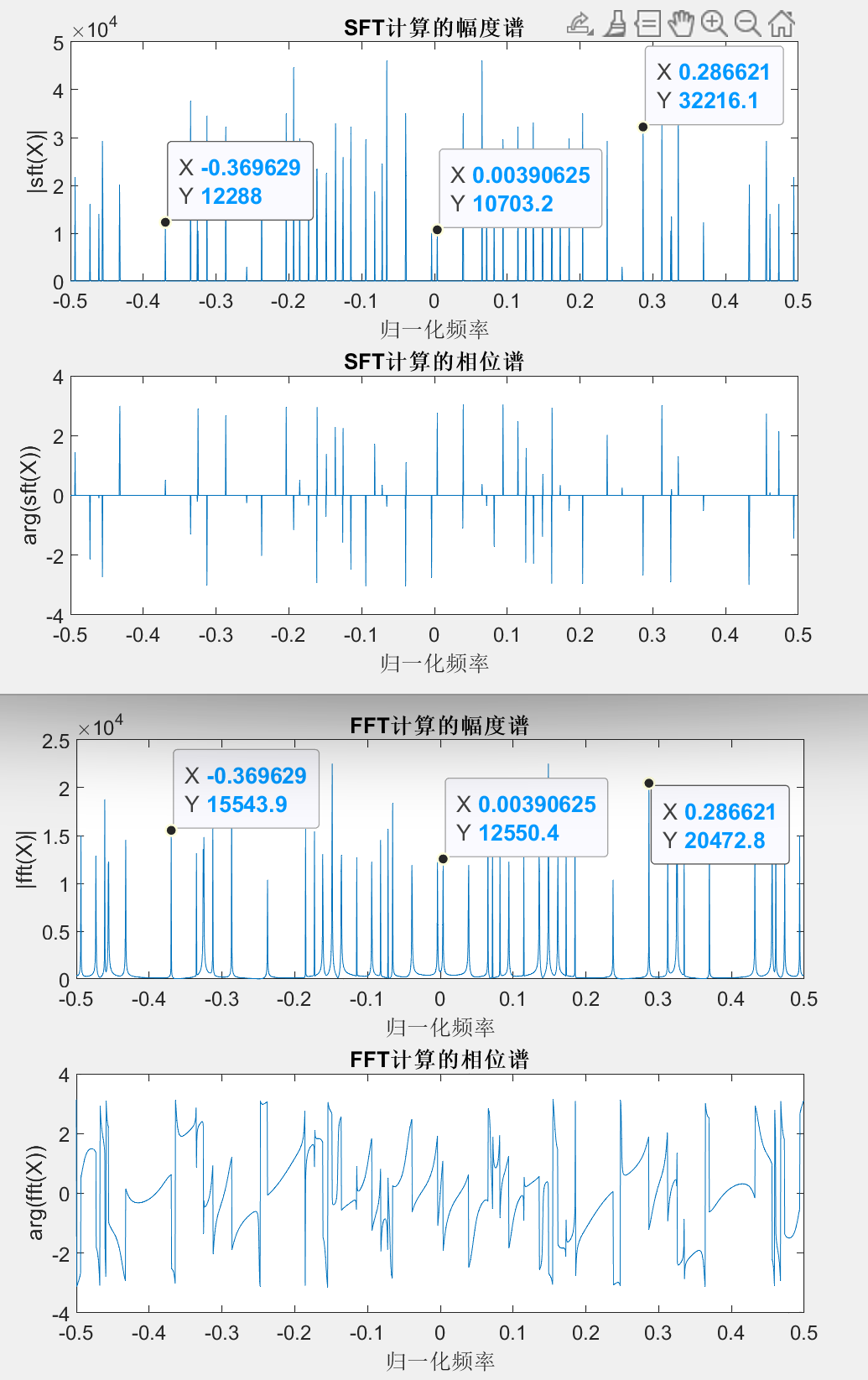


图 8 k=50

通过尝试可以确认K越大，SFT运行的越慢。不过SFT的运行时长存在随机性，与随机数的选取有很大关系，对长为2048的序列，我进行了简单定量分析如下，曲线与论文中转换为线性坐标基本一致：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| k | SFT耗时 | | | |
| 测试1 | 测试2 | 测试3 | 平均 |
| 2 | 0.00576 | 0.00757 | 0.00597 | 0.00643 |
| 10 | 0.00555 | 0.00825 | 0.00876 | 0.00752 |
| 30 | 0.01025 | 0.00656 | 0.00895 | 0.00859 |
| 50 | 0.01235 | 0.00791 | 0.01061 | 0.01029 |
| 100 | 0.01418 | 0.03696 | 0.01256 | 0.02123 |

表 1 稀疏度的影响

图 9 稀疏度的影响

###### 关于求值循环

实验开始我使用了论文中的求值循环，当然要对offset模w。但是我发现输出的绝对值总是与FFT结果不太一致。原因是我的窗函数频域很窄，导致除之后会出现很大的结果。于是我尝试了很多方法之后找到了适当修正的方案，可以保证n在256-8192范围内的绝对值相差小于1个数量级。但是相对值基本保真。如果仅仅关心模值，其实不用修正，仅仅乘固定倍数即可。

###### 关于筛选与求值

论文提出了采用频次大于L/2的判据。这种判据在论文中证明了正确概率相当高。然而实际发现这一判决导致受L影响很大，且可能选出多于k个值。我们还可以采用选取绝对值最大的k个值的方法，效果也很好。

同时，论文中采用中位数求值，我尝试了平均数，个人感觉效果好于中位数。

#### 实验总结

这次实验锻炼了我从论文中学习的能力，也让我学会了SFT。感觉还有理论很美妙，但是实现比较工程，调节参数花费了很长时间，也让我感受到了DSP较为前沿的研究，很有收获。

**附录**

文件清单：

代码文件：

main.m

innerloop.m

outerloop.m

genwindow.m

实验报告