Sparse Fourier Transform

Simple Algorithm and Implement

ZhiXiong Yang
He Zhang
Xi Zhang

Instructor: A.Petropulu

Abstract

Discrete Fourier Transform (DFT), as one of the most fundamental tools in modern signals processing, has its Fast Fourier Transform (FFT) algorithm to improve its computing rate. However, with the occurrence the big data problem, the size of the processed data growing so fast that FFT will take long time in computing. So we need faster ways to compute the spectrum of the target signal. Then, it comes to the Sparse Fourier Transform(SFT).The SFT is a new FFT algorithm that deal with signal with *K-sparse.* And our method is to realize this method, based on the idea introduced in the [1].

The goal of this report is to implement Sparse Fourier Transform. And our method can be expressed in four main stages. First, randomly permutation so as to separate the close dominant frequencies. Then, the process of binning (also denoted as filter bank) divides to divide the whole signal into multiple components, then each bin (frequency sub-band) may have one dominant frequency. Third, we implement methods of single frequency recovery[2] to locate the dominating frequency and decide its coefficient. Finally, we will map the selected the frequency back to original signal and get the spectrum of the *K-sparse* signal.

Finally, we compare the SFT of the target signal and FFT of the target signal. And experiment in low and high SNR environment. At last, evaluate the running time of the FFT and SFT.

SPARSE FOURIER TRANSFORM 1

ABSTRACT 2

1. BACKGROUND AND BASIC IDEA 4

1.1 MOTIVATION 4

1.2 BASIC IDEAS 5

2. METHODS 7

2.1 PERMUTATION 7

2.2 BINNING FREQUENCY (SHATTERING) 9

2.3 SINGLE FREQUENCY RECOVERY 9

2.3.1 PHASE ENCODING 9

2.3.2 AN ALIASING-BASED SEARCH 10

2.4 ITERATION 11

2.4.1 VOTING 11

2.4.2 THRESHOLD 11

2.5 ESTIMATION 12

3.SIMULATION AND EXPERIMENTAL RESULT 13

3.1 ROBUSTNESS TO NOISE 13

3.1.1 SIMULATION IN HIGH-SNR BACKGROUND 13

3.1.2 SIMULATION IN LOW-SNR BACKGROUND 14

3.2 RUN-TIME WITH THE CHANGE OF SIGNAL LENGTH 15

4. CONCLUSION 16

5. REFERENCE 17

1. Background and Basic Idea

1.1 Motivation

Discrete Fourier Transform (denoted as DFT) is the most fundamental computational tool in signal processing and scientific computing. It converts the finite samples from its original domain to frequency domain by the following equation[3]:

$$\hat{X}(\omega) = \sum_{n=0}^{N-1} x(n) \cdot e^{-j\frac{2\pi}{N}\omega n} \tag{1.1}$$

And the Inverse DFT is defined as:

$$x(n) = \sum_{\omega=0}^{N-1} \hat{X}(\omega) \cdot e^{j\frac{2\pi}{N}\omega n} \tag{1.2}$$

The general DFT requires $O(N^2)$ computation time when generating N outputs. And then, several efficient ways of computing DFT appear. And one of the most significant ways is Fast Fourier Transform (denoted as FFT) takes advantage of decomposition and improves efficiency to $O(N \log N)$ time.[3]

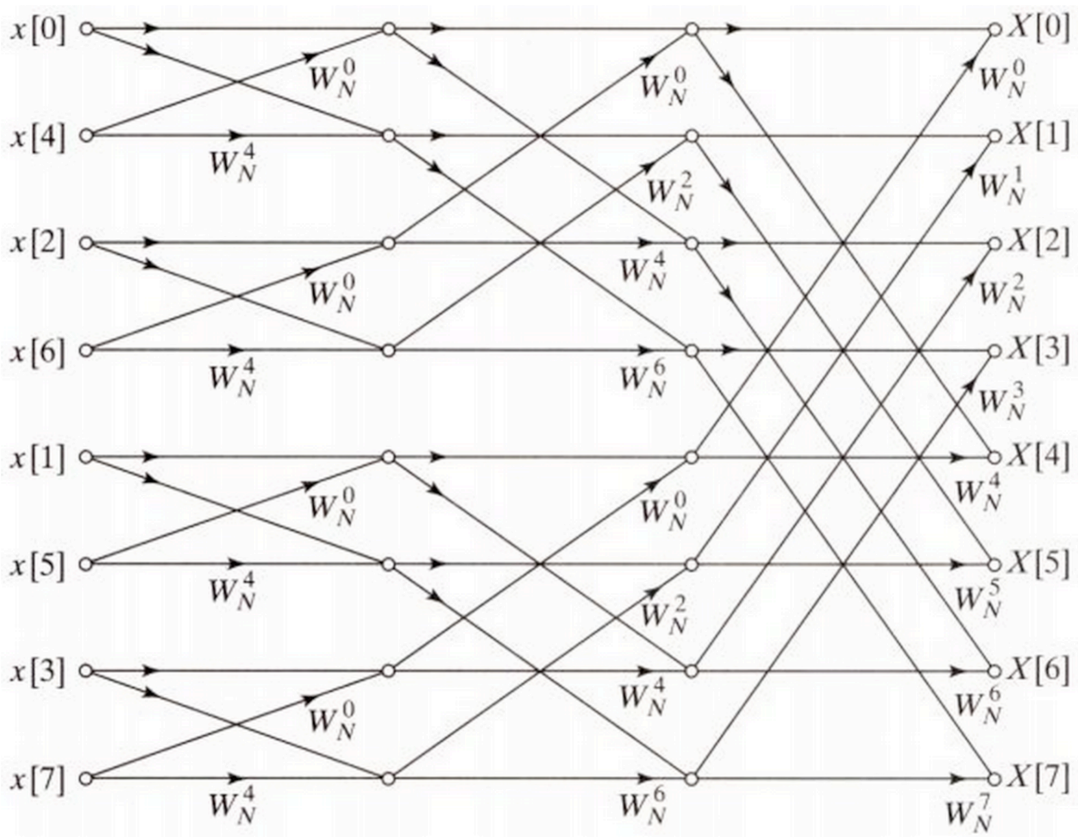

Figure 1: Architecture of 8-point Decimation-in-time FFT

Though the FFT algorithm is very fast, the computation time depend on the length of the sequence N. So the FFT is unable to deal with the big data problem efficiently, FFT seems not fast enough anymore. This motivates a new method to improve the efficiency of the Fourier transform, then it comes to the development of Sparse Fourier Transform (denoted as SFT).

When the signal just involves few frequencies, the existence of a very large number of zero coefficients led to inefficiency of FFT. In the contrast, Sparse Fourier Transform computes an approximation or compressed version of the DFT that focus on the sparsity (non-zero) of the spectrum.

Developed over the last decade, SFT algorithms use only a small subset of the input data, which is proportional to the sparsity in frequency. So, SFT can improve the efficiency by only computing nonzero or large frequencies and their complex amplitudes, rather than all frequencies.

1.2 Basic ideas

When a signal contains only 1 frequency in it, the DFT of this signal can be calculated by only looking at 2 points. The technique will be shown in the section Phase Encoding. Based on this technique, no matter how long the signal is, the DFT can be computed very fast.

Despite the fastness, however, Phase Encoding is not very robust against noises. We can easily observe that a little perturbation will change the result. To improve the performance, we introduce another method based on Chinese Remainder Theorem, which will be shown in the section An Aliasing-Based Search. We emphasis that these two methods will fail (or be much more complicated) if the signal contains more than one frequency.

Although the signal with only 1 frequency is very easy to deal with, it is not usually the case. We need a technique to decompose the signal into signals with 1 frequency. The process is called binning. We pass the signal through a filter to split the spectrum of the signal into several small bins. Since the bin is very small comparing to the whole spectrum, each bin will be very likely to have only 1 frequency in it. Then the analysis can be performed to each bin.

There are possibilities that 1 bin contains more than 1 frequency because two frequencies are very close to each other. To solve this problem, we need a random permutation process. Permutation is to circular shift every point in the spectrum. If the distance of shift is randomly generated, the non-zero frequencies will be randomly distributed. This process will separate two close frequencies apart to be ready for

binning. We repeat permutation several times, every frequency can be isolated into a small bin due to the sparsity of the spectrum. Of course we need a way to map the permuted spectrum back to the original. This will be introduced in the section Permutation.

Besides locating the non-zero frequency, we also interest in the value of each frequency. The value estimation will be performed after we locate each non-zero frequency. We will show that the value can be estimated using part of the result in the single frequency recovery. Putting the locations and values of each non-zero frequencies back together will give the result for SFT.

We implement the algorithm introduced above in Matlab. And the flow chart of our method is shown in the Figure 2.

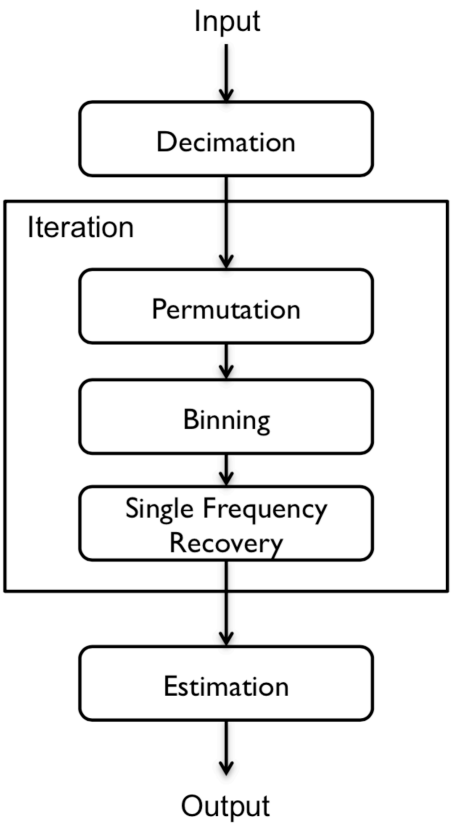

Figure 2: The Flow of the SFT Process

2. Methods

2.1 Permutation

As mentioned above, the frequencies in the frequency domain may be close to each other, so we are unable to isolate the different frequencies by way of band-pass filter. The largest magnitude entries can appear anywhere in principle. So we need methods to randomly permuting the spectrum. Also two important feature of permutation is as follows:

1. It is a randomized method to reconstruct the signal.
2. The permutation should be mapped back to the original signal

Then according to the method introduced in [4], We could permute the signal by way of dilation in time domain and then there will be permutation in the frequency domain. It is defined in Definition 1:

Definition 1: *Given a number* σ*, which is a co-prime positive integer of* N*. ie. gcd(*σ*,* N*)=1,and* $\sigma \in [N]$*.Then considering the dilation* d_σ*:* $n \mapsto \sigma n \bmod N$*, .Then, we could prove that :*

$$y[n] = x[\sigma n] \text{ for all } n \Leftrightarrow Y[w] = X[\sigma^{-1} w] \text{ for all } w \tag{2.1.2}$$

where σ^{-1} is *multiplicative inverse* of σ ,and $\sigma \cdot \sigma^{-1} \equiv 1 \bmod N$.

So according to the expression (2.1.2), after permutation the signal in time domain，the spectrum of signal also permutated. And the proof of equation (2.1.2) is shown in *Proof 1.*

*Proof 1:*Define the sequence after permutation as y(n), then

$$y(n) = x(\sigma n \bmod N) \tag{2.1.3}$$

The Fourier transform of y(n) is defined as

$$\hat{y}(\omega) = \sum_{n=0}^{N-1} y(n) e^{-j\frac{2\pi}{N}\omega n} \tag{2.1.4}$$

or

$$\hat{y}(\sigma\omega) = \sum_{n=0}^{N-1} y(n) e^{-j\frac{2\pi}{N}\omega\sigma n} = \sum_{n=0}^{N-1} x(\sigma n \bmod N) e^{-j\frac{2\pi}{N}\omega(\sigma n \bmod N)} \tag{2.1.5}$$

Let $m = \sigma n$

$$\hat{y}(\sigma\omega) = \sum_{m=0}^{N-1} x(m) e^{-j\frac{2\pi}{N}\omega m} = \hat{x}(\omega) \tag{2.1.6}$$

From this equation, we have shown that the time-domain permutation also leads to frequency- domain permutation. The equation also gives the way to map back from permuted spectrum to the original.

From *Proof 1,* it is obvious that permutation in time domain leads to the permutation in frequency domain. And Figure 3 shows the permutation result of a 256 -point sequence with four frequencies.

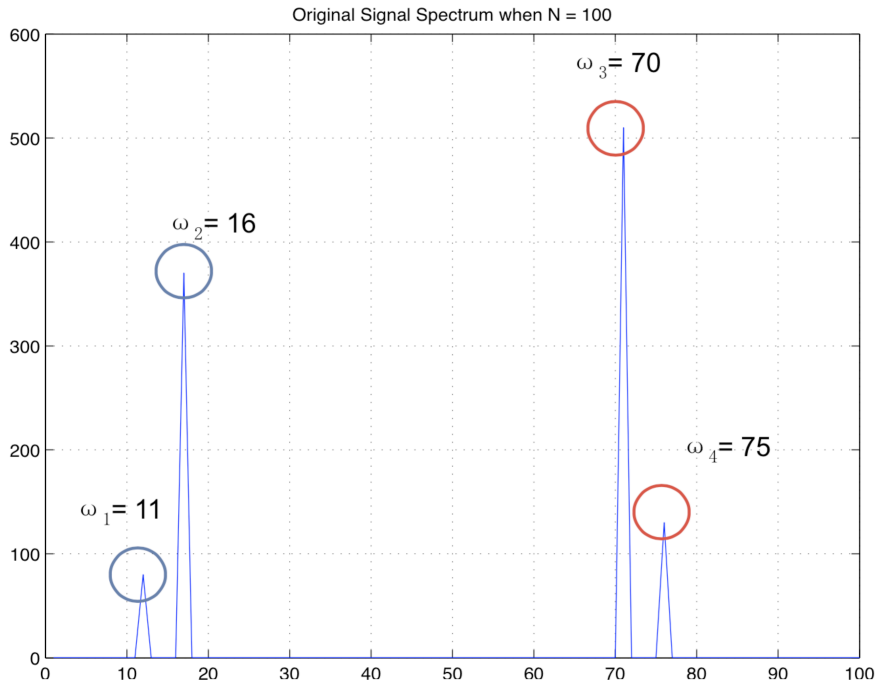

Figure 3: The spectrum of the original signal

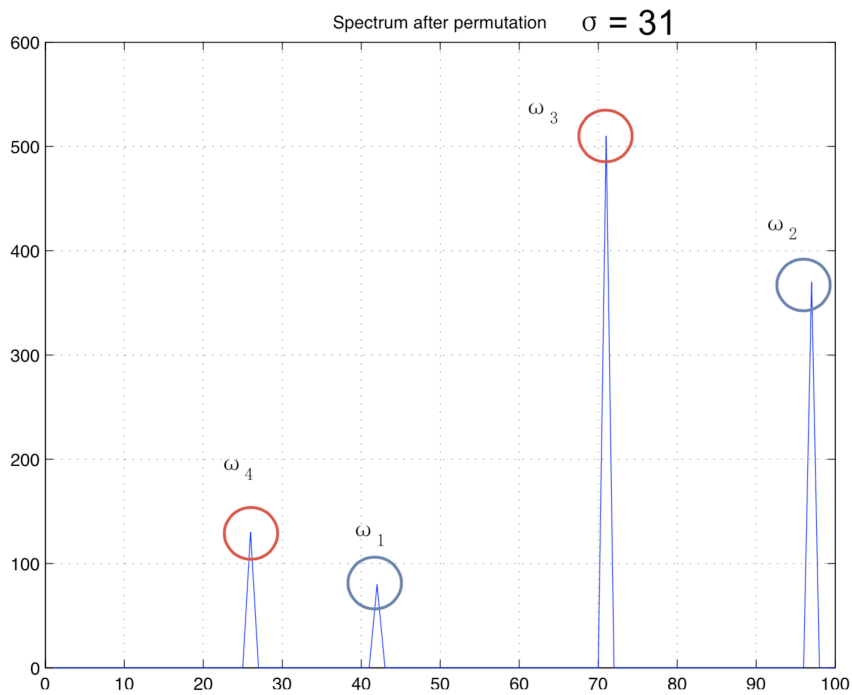

Figure 4: The spectrum of the permuted signal

2.2 Binning Frequency (Shattering)

The process of permutation randomly reorders the spectrum, and then dominant frequencies may be separated from each other though some of them are close each other in the original frequency domain. Then we could use multiple band-pass filters, so each dominant frequency may occupy one 'bin'. Then, we could use the single frequency recovery technique to find the dominant frequency efficiently [5].

The design of band-pass filter bank is very simple. Let h_{low} to be a low-pass filter with cut off frequencies π / K rad. Then, the filter bank consists of K frequency modulation of this low-pass filter, spaced $2\pi / K$ rad apart. In the time domain, it can be expressed as:

$$h_k[n] = e^{-2\pi jk/K} h[n] \quad k = 0,1,\ldots,K-1 \tag{2.2.1}$$

And it is possible that more sophisticated low-pass filters or windows may lead to better result.

Also, in practice we should take essential ideas into consideration:

1.The filter should be highly sparse in time domain, which means there will be less order for the low-pass filter.

2.The DFT of such filters should be in the construction of low-leakage.

After shattering the signal to different 'bins' (band-pass), we could use efficient method to recover the single dominant frequency in each bin.

2.3 Single Frequency Recovery

In each bin, if we have just one dominant frequency, then SFT is able to capture the dominant frequency much faster the $O(N \log N)$ - time FFT. In this part, I will introduce two methods to illustrate how to locate the domain frequency and why it is faster than the FFT. The first method is denoted as Phase encoding and second method is denoted as aliasing-based DFT search. Before going deep in the two methods, I will assume that our vector is in the following form:

$$x(n) = \hat{X}(\omega_0) \cdot e^{j\frac{2\pi}{N}\omega_0 n} \tag{2.3}$$

where $w \in \{0,1,\ldots,N-1\}$.

2.3.1 Phase Encoding

If the signal contains only one frequency, we use the expression() to get the following equation:

$$\frac{x(n+1)}{x(n)} = e^{j\frac{2\pi}{N}\omega_0} = \cos(\frac{2\pi}{N}\omega_0) + j \cdot \sin(\frac{2\pi}{N}\omega_0) \tag{2.3.1}$$

where $n+1 = (n+1) \bmod N$

Once we know the phase of the $x(n+1)/x(n)$, then we could obtain the frequency w, $\hat{X}(\omega_0)$ could be calculated according to the expression (2.3). It's obvious that this procedure effectively find DFT by using only two entries of original signal. The complexity of this procedure is improved to $O(1)$-time. However, this method is not robust to noise. Once the signal contains noise, then the expression of (2.3.1) are distorted, so we are unable to locate the location of dominating frequency.

2.3.2 An Aliasing-Based Search

The phase encoding is not robust to noise, so we need some new method to locate the domain frequency. Compared with phase coding, an alias-based search method is easy to implement and more robust to noise. But there will be some guarantee to the total point of the signal , the length N should be is a product of several prime integers.

Recall equation (2.3), assume

$$a_0 = x(0) = \hat{X}(\omega_0) \tag{2.3.2}$$

After calculating:

$$\hat{A}_0 = \hat{X}(\omega) \cdot \frac{1+(-1)^{\omega_0}}{2} \tag{2.3.3}$$

$$a_1 = x(\frac{N}{2}) = \hat{X}(\omega_0) \cdot e^{j\pi\omega_0} = \hat{X}(\omega_0) \cdot (-1)^{\omega_0} \tag{2.3.4}$$

$$\hat{A}_1 = \hat{X}(\omega) \cdot \frac{1+(-1)^{\omega_0+1}}{2} \tag{2.3.5}$$

The value of $\hat{A}_0 \neq 0$ means $\omega_0 \equiv 0 \pmod 2$, and $\hat{A}_1 \neq 0$ means $\omega_0 \equiv 1 \pmod 2$. For instance, assume the point of DFT is N = 2 x 5 x 7.

Above analysis can expand to mod 5, mod 7 or mod any prime integer. For any $\omega_0 \in [0,1,..,N-1]$, the location of ω_0 can be calculated based on Chinese Remainder Theorem (CRT).

Suppose we have a sequence $x[n]$ of length N and its dominant frequency is at $w = M$. When we take 3 point DFT, the spectrum will be aliased. Let $a = DFT_3(x[n])$ and let $b = DFT_N(x[n])$. Compare the 3 point DFT to N point DFT, we can find the relation

$$a_1 = \sum_{m=3k+1} b_m, a_2 = \sum_{m=3k+2} b_m, a_3 = \sum_{m=3k} b_m \quad \text{k=0,1,2...} \tag{2.3.6}$$

In the single frequency case, only 1 of a_1, a_2, a_3 could be non-zero. The index of non-zero point in a equals $M \bmod 3$ ($M \bmod 3 = 0$ if the index is 3). With the choice of different points of DFT, we can collect remainders of M with different denominator. We can then implement CRT algorithm with enough remainders and denominators.

Chinese Remainder Theorem (CRT):

Any integer x is uniquely specified modulo N by its remainders modulo m relatively prime integers $m_1, m_2, ..., m_k$, *as long as* $\prod_{i=1}^{k} m_i > N$.

Then, I will use an example to illustrate the First, assume $m_1 = 2, m_2 = 5, m_3 = 7$, then $M_1 = 35$, $M_2 = 14$, $M_3 = 10$. Then the reciprocal of number theory is $t_1 = 1$, $t_2 = 4$, $t_3 = 5$.If x has the equations.

Then x can be written as:

$$1 \times 35 \times 1 + 2 \times 14 \times 4 + 5 \times 10 \times 5 + k \cdot 2 \times 5 \times 7 \Rightarrow 397 + k \cdot 70 \qquad (2.3.7)$$

If $x \in [0, N]$, x = 47 when k = -5.

2.4 Iteration

Since the permutation process is a random process, we cannot guarantee that each bin contains only one frequency. But the repeating the process for several times will give it a high likelihood to isolate a frequency. During each iteration, it is also important to determine if a bin truly contains only one frequency (other possibilities are more than 1 frequency and no frequency at all).

2.4.1 Voting

We discard the wrong result caused by a bin containing more than one frequency by voting. We let the CRT search return the index of largest valued frequency found in a bin and vote for that frequency once. Since the right index has much higher probability to show up in the result, its number of votes will become larger than all the other indexes after several iterations. When all iterations finish, we keep the largest k frequencies and discard all the rest.

2.4.2 Threshold

When a bin has no dominate frequency in it, the voting process will still locate the largest value in the bin, which is, however, very likely to be noise. This result could prevent us from finding the right answer as well as waste time. To eliminate this possibility, we set a threshold to the largest value found. The vote will only be considered valid when there is a large enough value in the bin.

2.5 Estimation

The final operation is to assign values to the located dominant frequencies. There are several ways of realizing it. As described in phase encoding, we can calculate the values directly from equation (2.3). Once again, the method is not very robust. If we scan every point of the signal and calculate the value of one exact point of its DFT, the value will be accurate. But the cost is reducing the calculation speed, which is against our goal.

Here we introduce a method that combines the two methods we just mentioned. The method is based on the unbiased estimator. If we extract equally spaced M points to calculate its FFT, the value corresponding to the dominating frequency will be

$$FFT_M(\omega') = \frac{M}{N} FFT_N(\omega) \tag{2.5.1}$$

ω' is the index of the dominate frequency in M points FFT. Recall that during the single frequency recovery we already calculated FFT of a downsampled version of the signal. We can take advantage of that process by keeping the value as well as the location of the dominating frequency during each iteration. Note that there may be different values appear in different iteration due to the performance of the filter. We take the largest value to improve accuracy.

3.Simulation and Experimental Result

3.1 Robustness to Noise

With the input with 5 frequencies, we set $K = 5$. The input contains 50Hz, 70Hz, 123Hz, 145Hz, 450Hz and 668Hz with magnitude of one. And to test the robustness to noise, the input signal is combined with the white Gaussian noise, whose variance depends on the SNR. And the low-pass filer is designed through Equiripple FIR filters with 30 orders, then modulated 6 times to generate the filter bank.

We will experiment both in low SNR and high SNR environment.

3.1.1 Simulation in High-SNR Background

With no noise in the input signal, we use the input denoted in the chapter 4.1 and then plot the result of SFT and FFT.

Then, Figure 5 shows the FFT and the SFT result of the input signal. It is encouraging that we can recover the signal perfectly. The SFT is almost the same as the FFT.

When the SNR is 5, the result is shown in Figure 6,it is obvious that the method is robust to noise to some degree.

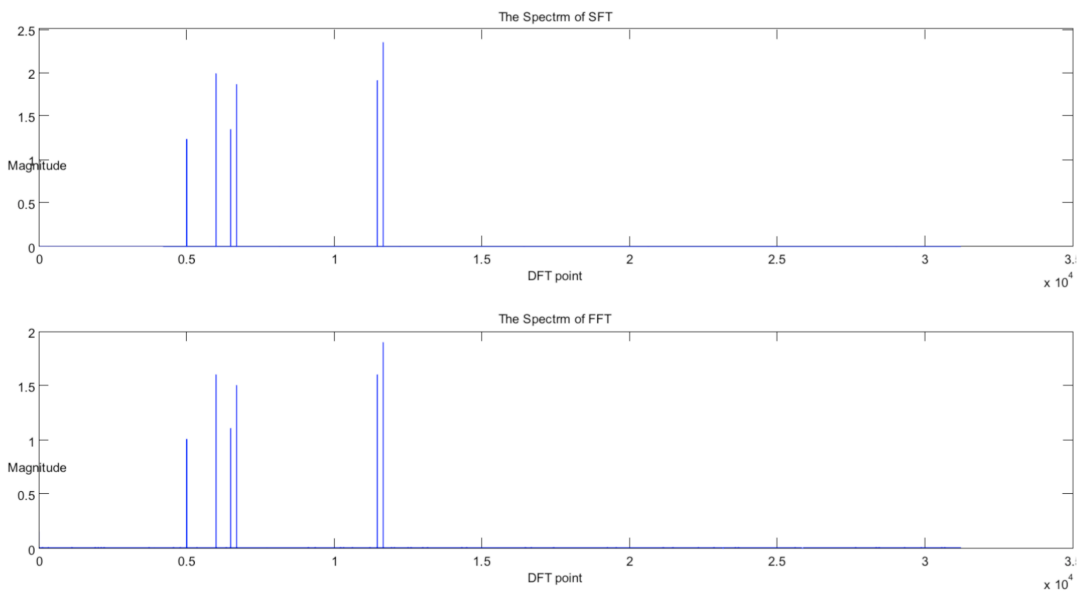

Figure 5: The SFT and FFT of the input signal (with no noise)

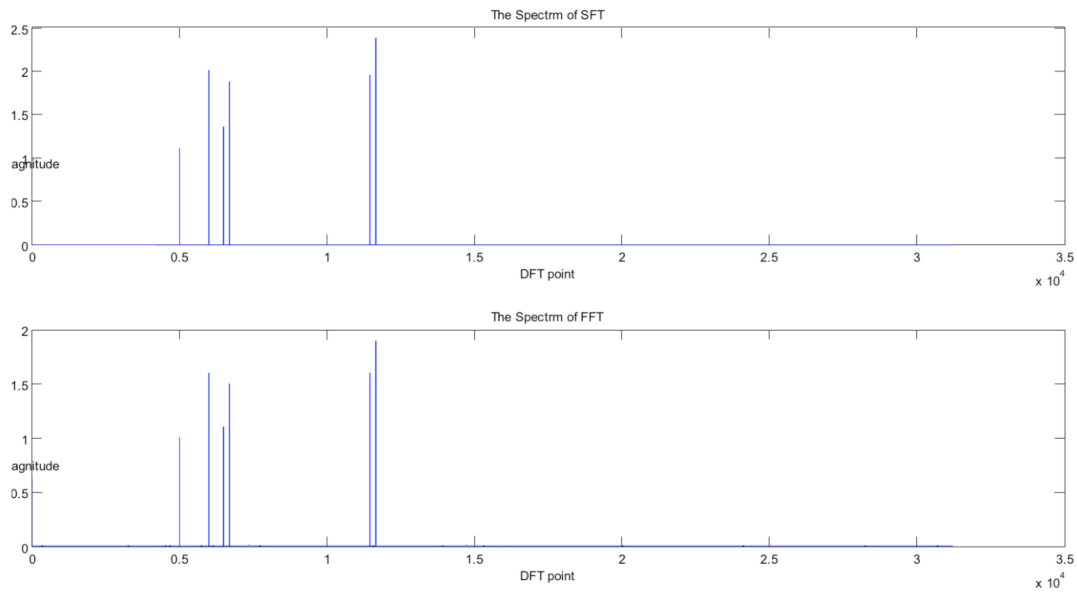

Figure 6: The SFT and FFT of the input signal (with SNR=5)

3.1.2 Simulation in Low-SNR Background

Then we add more noise to the input signal, we plot the result of SFT and FFT.

When the SNR is 0.5, then, Figure 7 shows the result. When there is so much noise added in the input, we may wrongly detect the dominant frequency.

However, if we iterated twenty or more times, then we could filter out the 'chasing' frequency and get accurate result.

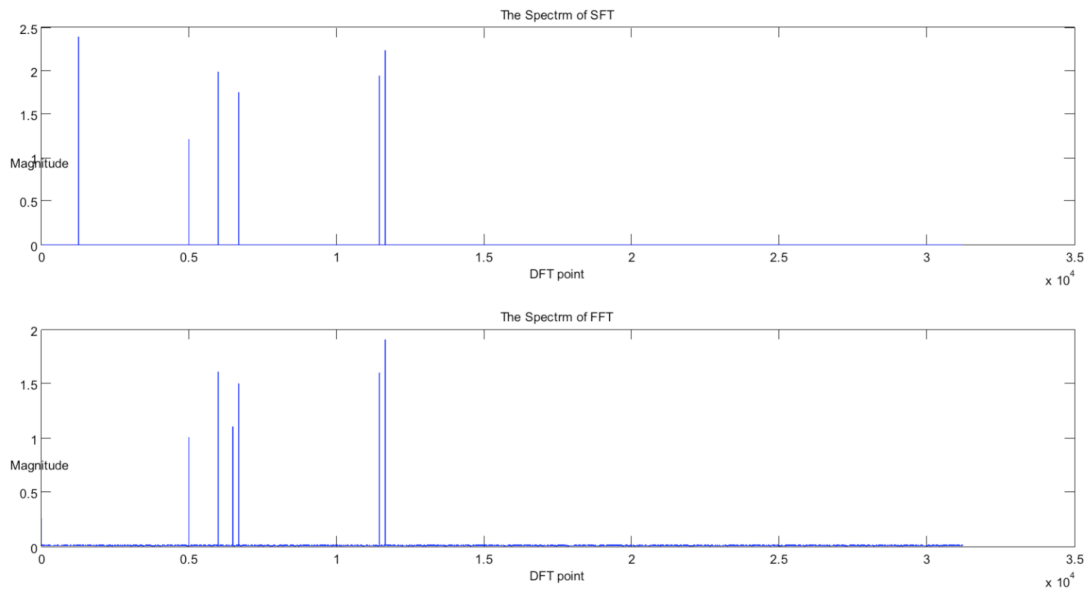

Figure 7: The SFT and FFT of the input signal (with SNR=0.5)

The reason leads to the 'locating' the wrong frequencies lies in that the noise may occupy some 'bin' during binning process, but it is noise, it will not dominate just one frequency, so if we increase the iteration times, the result will be more guaranteed.

3.2 Run-time with the change of signal length

Chapter 4.1 shows that the method designed is robust to noise to some degree and can accurately reflect the spectrum of the input signal. And then, we will calculate the efficiency of the SFT method compared with the running time of classical FFT with the same signal size. It is shown in Figure 8.

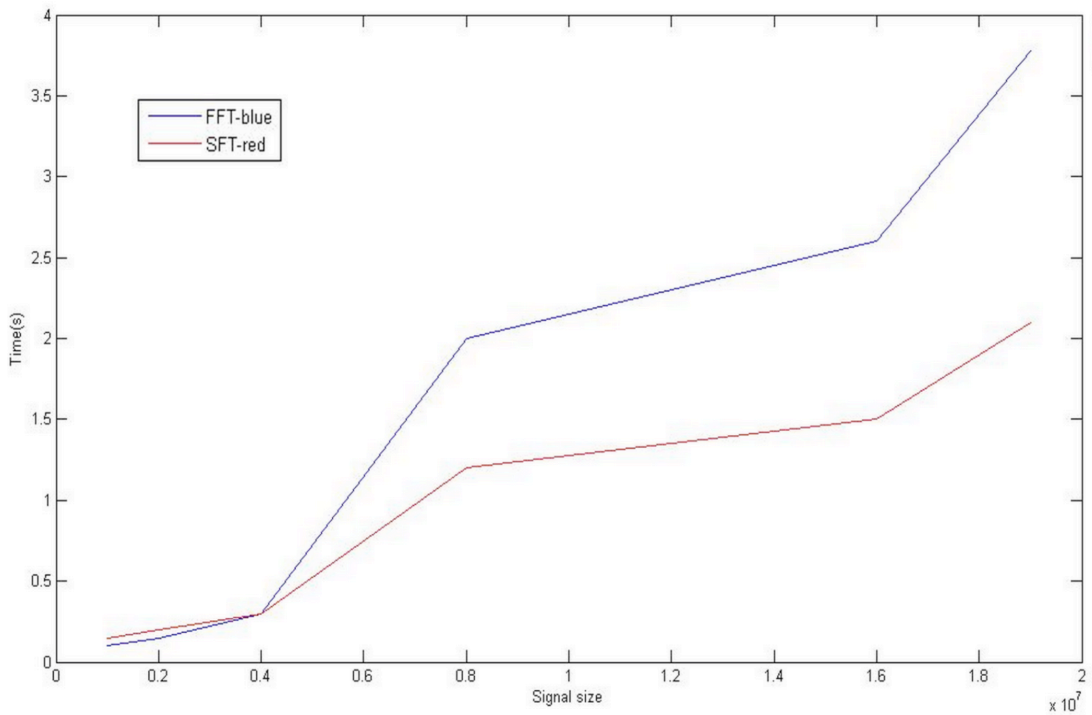

Figure 8: The time of SFT and FFT changed with signal size

4. Conclusion

It is shown that the SFT can be faster than FFT under certain large data condition. This brings application potentials to this algorithm. But it this is only one beneficial attempt. In order to apply the algorithm to different working environment, there are still many parts that can be improved.

The permutation algorithm consumes a lot of time comparing to other stages while filtering and frequency recovery only requires a short sequence. This means that the waste of resources on permutation can be avoided or reduced. Our future work direction is now targeting the combination of the permutation and filtering process which basically calculating only necessary elements from the original signal.

On the other hand, Matlab programming structure is very different from C or C++. While implementing and simulating algorithms under Matlab environment, we should try to adjust our programme compensate for Matlab. For example, Matlab has no parallel process. Every parallel process will be realized by iteration, which means that parallel process is no longer as computationally beneficial as in C++. In this direction, there still are future works to be done.

5. Reference

[1] A. C. Gilbert, P. Indyk, M. Iwen, and L. Schmidt, "Recent developments in the sparse Fourier transform: A compressed Fourier transform for big data," IEEE Signal Process. Mag., vol. 31, no. 5, pp. 91–100,Sep. 2014.
[2] M. A. Iwen, "Improved approximation guarantees for sublinear-time Fourier algorithms," Appl. Computat. Harm. Anal., vol. 34, pp. 57–82, 2013.
[3] Oppenheim, Alan V., Ronald W. Schafer, and John R. Buck. Discrete-time signal processing. Vol. 2. Englewood Cliffs: Prentice-hall, 1989.
[4] H. Hassanieh, P. Indyk, D. Katabi, and E. Price, "Simple and practical algorithm for sparse Fourier transform," in Proc. 23rd Annu. ACM-SIAM Symp. On Discrete Algorithms (SODA), 2012.
[5] A. C. Gilbert, M. J. Strauss, and J. A. Tropp, "A tutorial on fast Fourier sampling,"IEEE Signal Processing Mag., 2008.
[6] Schumacher, Jörn. High performance sparse fast Fourier transform. Diss. ETH Zurich, Department of Computer Science, 2013.
[7] T. Sreenivas and P. Rao. FFT algorithm for both input and output pruning. IEEE Transactions on Acoustics, Speech, and Signal Processing, 27(3):291–292, June 1979.