Fast Fourier Transform

*Report 4 on the course "Numerical Analysis".

1st Chen Yihang
Peking University
1700010780

Abstract

The fast Fourier transform and inverse fast Fourier transform are implemented in this report, and are tested via filtering the noisy signals. We compare various levels of filtering by plotting the true signals, observed signals and recovered signals.

CONTENTS

I. PROBLEM STATEMENT

Suppose

$$f(t) = e^{-t^2/10}(\sin(2t) + 2\cos(4t) + 0.4\sin(t)\sin(50t)) \tag{I.1}$$

is the observed signal, and hence the real signal is

$$f_0(t) = e^{-t^2/10}(\sin(2t) + 2\cos(4t)) \tag{I.2}$$

We plot the signals in Figure I.1 and we want to recover the real signal from the observed signal.

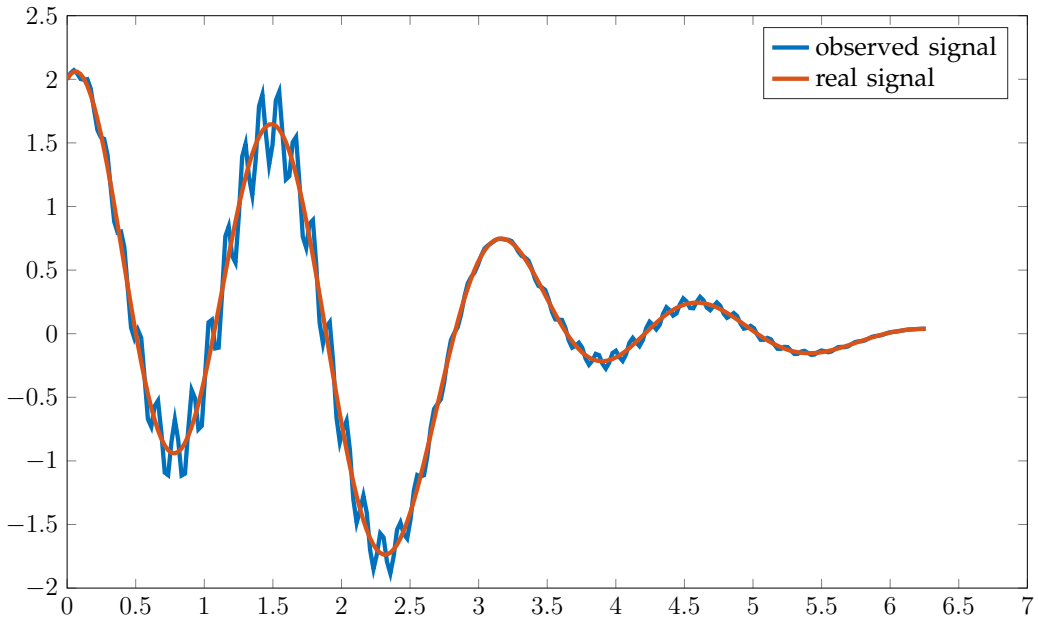

Figure I.1. Observed and real signal

Assume $y_k = f(2k\pi/256)$ $(k = 0, 1, \cdots, 255)$, $\hat{y}_k$ $(k = 0, 1, \cdots, 255)$ is the Fourier transform of y_k. Since $\hat{y}_{n-k} = \bar{\hat{y}}_k$, the low frequency terms are $\hat{y}_1, \hat{y}_2, \cdots, \hat{y_m}$ and $\hat{y}_{255-m}, \cdots, \hat{y}_{255}$ (for small m). We set $\hat{y}_k = 0 (m \leq k \leq 255 - m)$ to filter the high frequency terms out, and perform Fourier transform on new $\hat{y}'_k$, which produces y'_k. Plot y_k and y'_k and compare their differences, and try different m.

II. Fast Fourier Transform

A. Theory

Assume the inverse binary representation of integer i is invbin(i), and $w_N^j = e^{-\frac{2\pi ij}{N}}$ as usual. For $\mathbf{a} = (a_0, \cdots, a_{2^n-1})$, we can define

$$\mathbf{a}_{\text{invbin}(i)} = a_i \tag{II.1}$$

Then we regroup $\mathbf{a}_\varepsilon^{(n)}, \varepsilon^{(n)} \in \{0, 1\}^n$ by following method.

For simplicity, we define $\varepsilon^{(k)} \in \{0, 1\}^k$, $\mathbf{c}_{\varepsilon^{(n)}} = \mathbf{a}_{\varepsilon^{(n)}}$. Assume all $\mathbf{c}_{\varepsilon^{(k+1)}}$ has been

grouped, then for any $\varepsilon^{(k)} \in \{0,1\}^k$

$$\mathbf{c}_{\varepsilon^{(k)}} = \begin{pmatrix} \mathbf{c}_{\varepsilon^{(k)}+\{0\}} + \mathbf{w}_{2^{n-k}} \odot \mathbf{c}_{\varepsilon^{(k)}+\{1\}} \\ \mathbf{c}_{\varepsilon^{(k)}+\{0\}} - \mathbf{w}_{2^{n-k}} \odot \mathbf{c}_{\varepsilon^{(k)}+\{1\}} \end{pmatrix} \tag{II.2}$$

where

$$\mathbf{w}_{2^k} = (w_{2^k}^0, w_{2^k}^1, \cdots, w_{2^k}^{2^{k-1}-1})^\top \tag{II.3}$$

B. Implementation

In the programme, we use "complex.h" in C to perform complex operations.

In the begining, we need to reshuffle y_k. In general, we reverse the binary representation of each number in the following code:

```
int reverse_bit(int i, int n) {
int ans = 0;
for (int j = 0; j < n; ++j)
{
    ans = (ans << 1) | (i & 1);
    i >>= 1;
}
return ans;
}
```

Since the process of FFT and inverse FFT are similar, we use the variable "choice" to specify the difference. We list the code below:

```
void FFT(double complex *x, double complex *y, int n, int choice) {
    int N = 1 << n;
    for (int i = 0; i < N; ++i){
        int ind = reverse_bit(i, n);
        y[i] = x[ind];
    }
    for (int i = 1; i <= n; ++i) {
        int j = 1 << i;
        for (int s = 0; s < N; s += j) {
            for (int k = 0; k < j / 2; ++k) {
                double complex w = cos(2 * pi * k / j)
                - choice * sin(2 * pi * k / j) * _Complex_I;
                double complex u = y[s + k];
                double complex v = y[s + k + j / 2];
                y[s + k] = u + w * v;
                y[s + k + j / 2] = u - w * v;
            }
        }
    }
```

```
20      if (choice < 0){
21          for (int i = 0; i <= N; ++i){
22              y[i] = y[i] / N;
23          }
24      }
25  }
```

The begining loop in Line 3-6 is the reshuffling process, and the last loop in Line 20-24 is to resize the output in inverse FFT.

The main part is in the Line 7-19, which describes the regrouping process. The outer loop determine the stepsize j of the regrouping method by Line 8. Latter, the middle loop, in Line 9-18, divides the sequence into subsequences with length j. Finally, the innermost loop, in Line 10-17, join two consecutive subsequences together.

III. Results

We plot the frequency of the signal in Figure III.1. Hence, we want to filter the high frequency term while keeping the most of the low frequency term. We mainly want to filter the frequency term around 50-th. The choice of m should be around 10.

For $m = 3, 6, 12, 18, 24, 48$, we plot the results in Figure III.2, III.3, III.4. Clearly, as m increases, the recovery is more and more accurate. m being too small ($m = 3$) will increases the error, while m being too large will not filter the high frequency terms (eg. $m = 50$, see Figure III.1). Setting $m = 12$ or 18 is the optimal choice, which is aligned with our previous observation.

From Figure III.3, we find that the error mainly comes from the boundry region, since the true signals are not periodic. Larger m will reduce the infinity error of the oscillations, but will oscillates in higher frequency.

Actually, we find that setting $m = 48$ can filter the higher frequency terms efficiently, while setting $m = 50$ cannot, which is aligned with the frequency figure in III.1. We plot the results in Figure III.5.

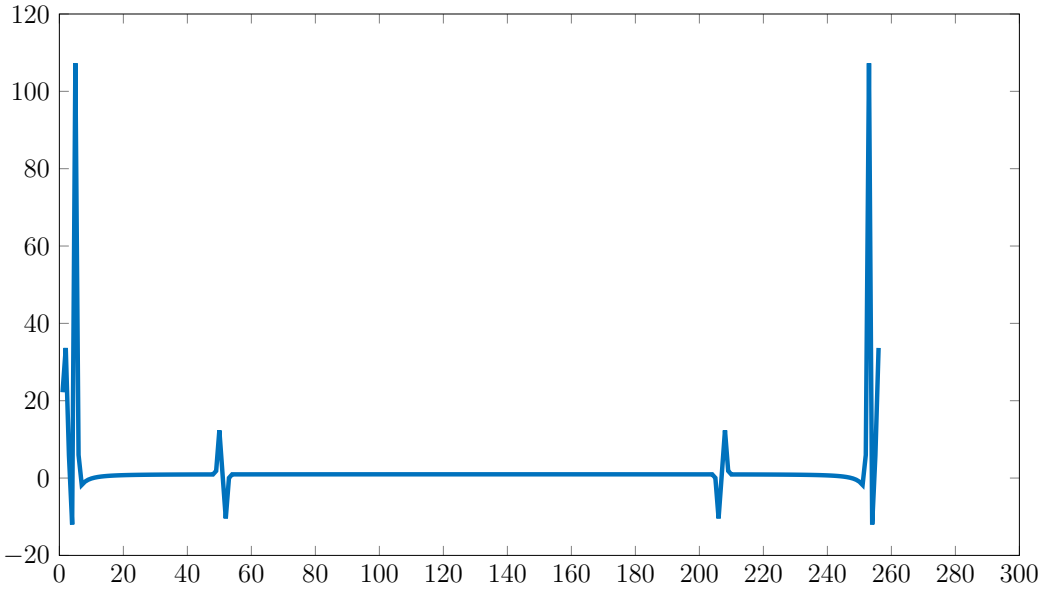

Figure III.1. Frequency of the Signal

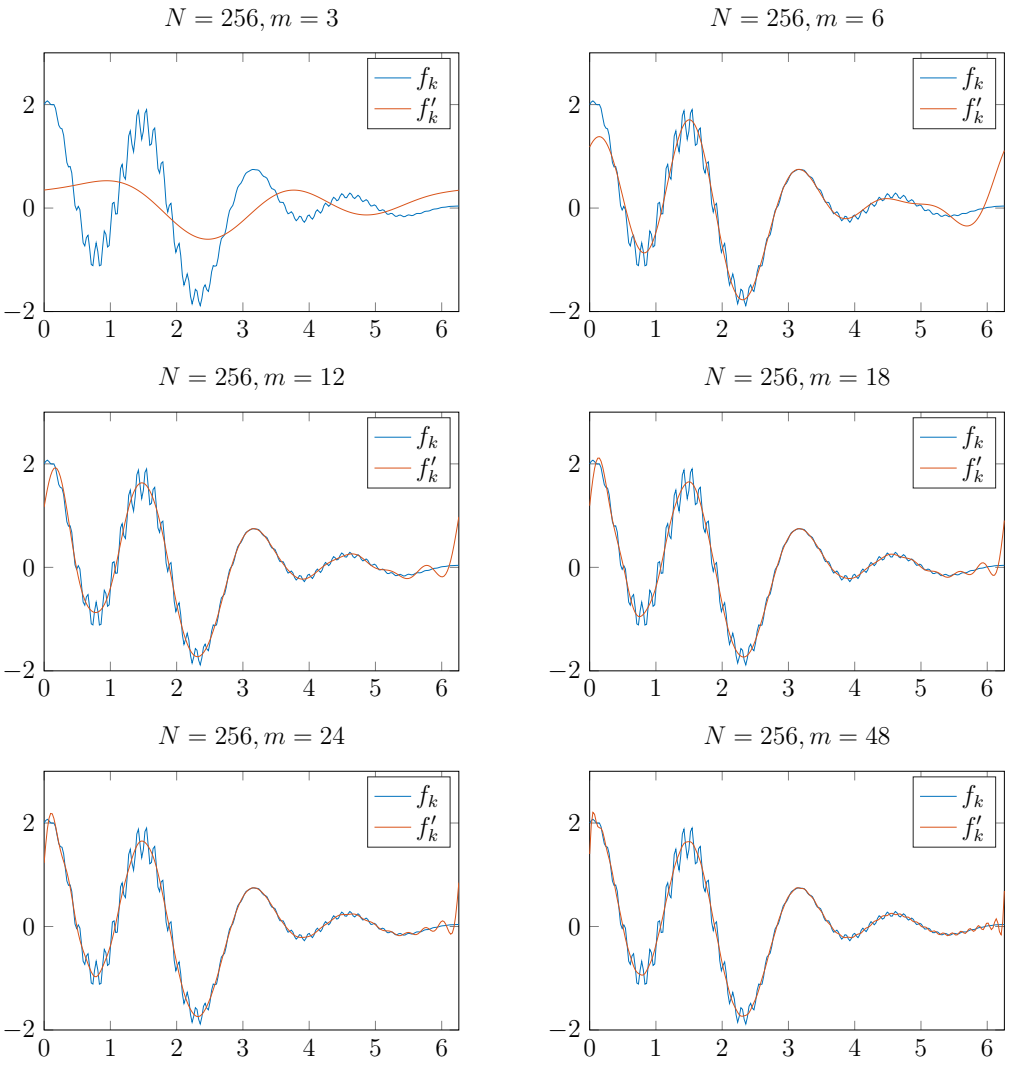

Figure III.2. FFT based signal recovery, compared to observed signals

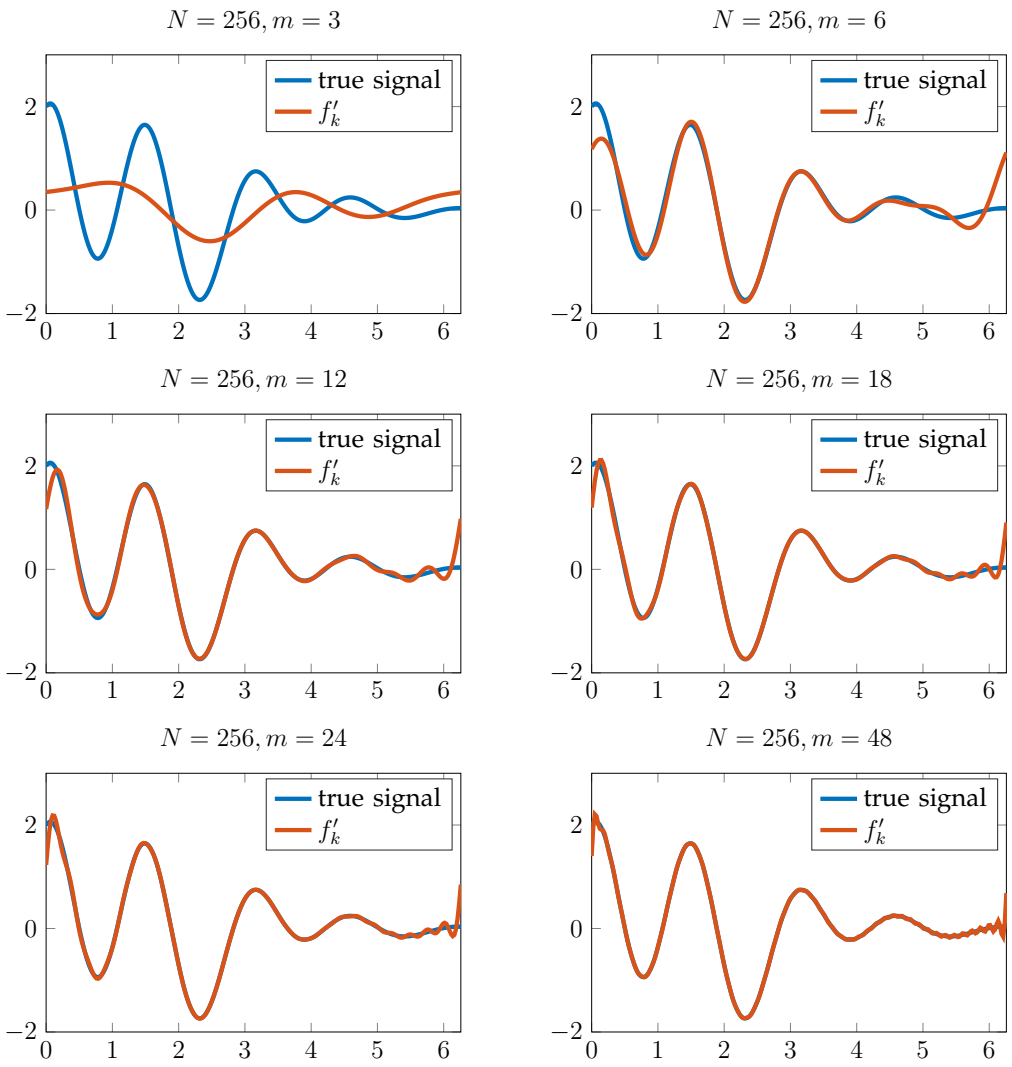

Figure III.3. FFT based signal recovery, compared to true signals

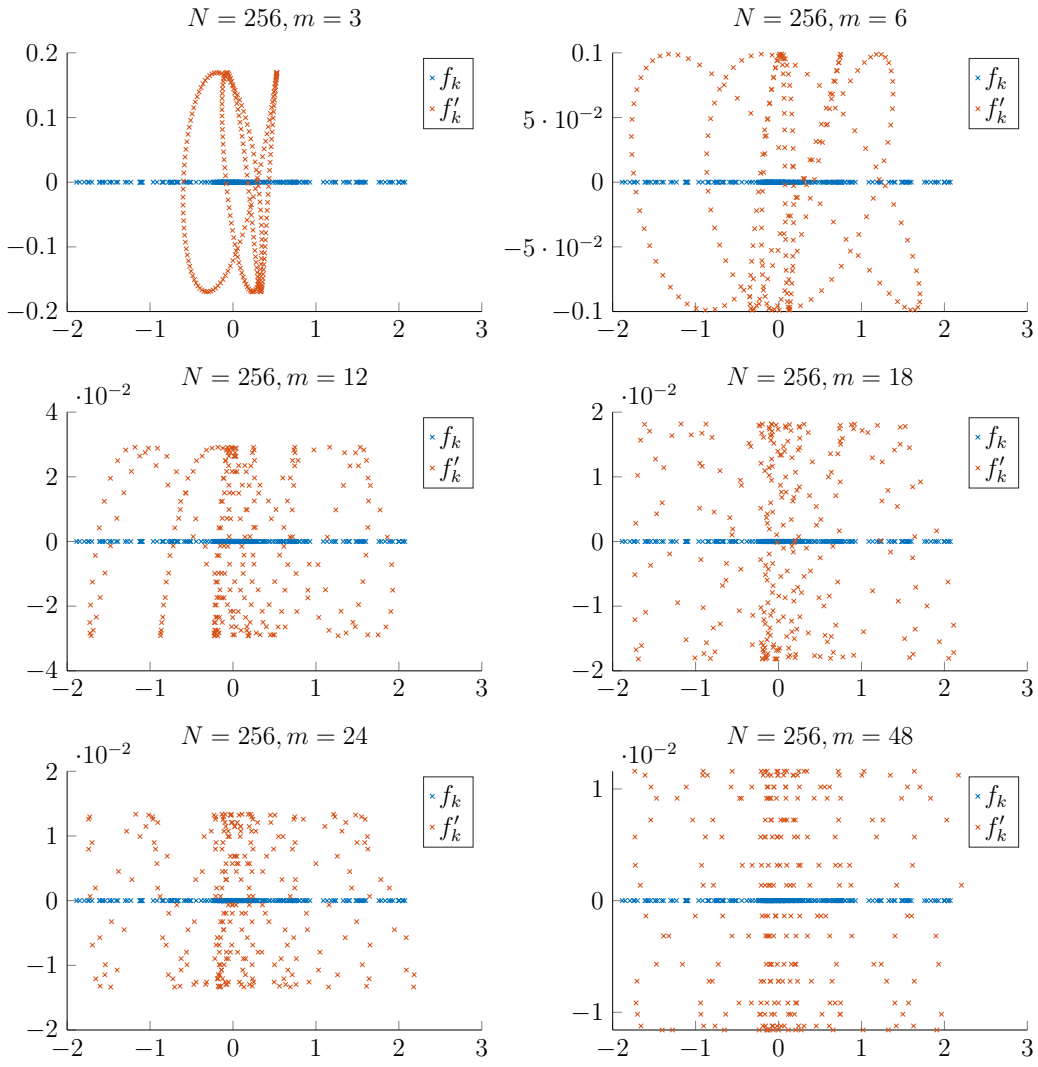

Figure III.4. FFT based signal recovery, in the complex plane

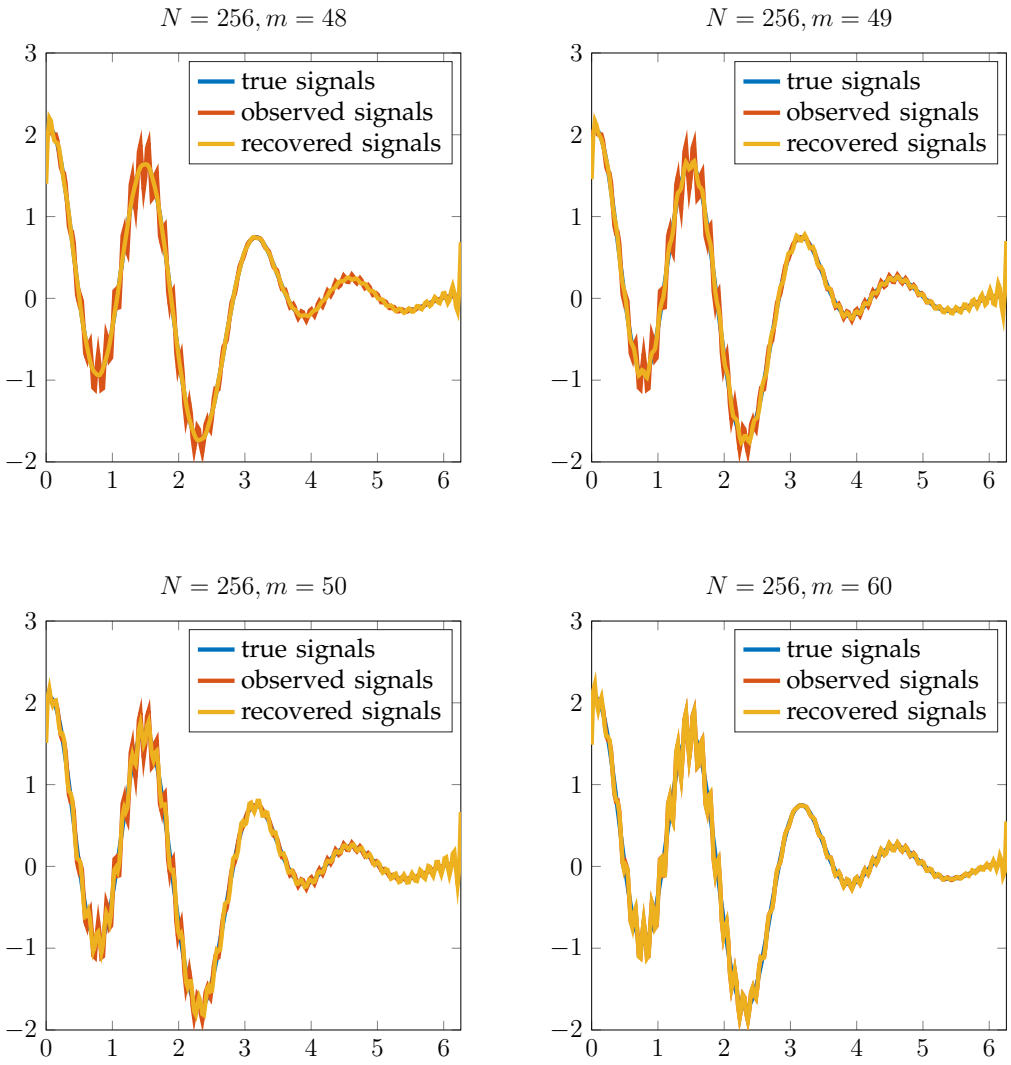

Figure III.5. FFT based signal recovery, compared to true and observed signals