

IE523: Financial Computing
Fall, 2016
Programming Assignment 6: Using Fast-Fourier
Transforms (FFTs) to filter Tick Data
Due Date: 6 October 2016
©Prof. R.S. Sreenivas

This programming assignment is about using the *Fast Fourier Transform* (FFT) algorithm to clean-up/process financial data. I am only going to cover the computational aspects of this procedure. To learn about the theory behind what we are going to do, you might want to look at Oppenheim and Schaffer's classic text [1]. We will use the implementation of the FFT algorithm in NEWMAT.

I am going to give you the required “functional basics” for what an FFT is etc. There is a lot more to this topic than what I am going to cover here. The easiest way to get to the foundations of FFT is this – any periodic signal $x(t)$ with period T (i.e. $x(t+T) = x(t)$ for any t) can be written as a sum-of-sinewaves (click this [wikipedia page for an illustrative example](#)). That is,

$$x(t) = C_0 + \sum_{i=1}^{\infty} \left\{ A_i \sin \left(\left(i \times \frac{2\pi t}{T} \right) \right) + B_i \cos \left(\left(i \times \frac{2\pi t}{T} \right) \right) \right\}. \quad (1)$$

The terms A_i and B_i denote the *amplitude* (i.e. strength) of the sinewave/-cosinewave whose frequency is $\frac{i}{T}$ Hz – the larger these numbers are, the more “significant” is the “contribution” of this frequency-component to the value of $x(\bullet)$. To be a little more precise, the larger the value of $\sqrt{A_i^2 + B_i^2}$, there is a larger contribution from the $\frac{i}{T}$ Hz component.

In essence, we are going to take the ticker data for a given duration (in our case, it is for a day), and we are going to assume that it is a single-period of a (fictitious) periodic waveform. Consequently, we can write this ticker data “pattern” as a sum of sinewaves and cosinewaves. The summation is over a large number of such terms, but some of the sinewaves/cosinewaves might have a small contribution (i.e. $\sqrt{A_i^2 + B_i^2}$ is small, compared to the rest) – we just drop them – and we get a (smoother) “approximation” of the ticker data pattern. Such filtered/smoothened version of the data can be used in numerous ways – one of them is to do short-term prediction, possibly. Another, could be to come up with triggers for automated-trades, but that requires some experimentation on your part.

If the ticker data has n -many entries over a time period of T , it turns out we can generate the values of A_i and B_i ($0 \leq i \leq n$) from the *Fast-Fourier Transform* (FFT) routine. The version in NEWMAT ([click here for the NEWMAT documentation](#)) that we will use has the format `RealFFT(X, F, G);`, where X is the (real-valued) ticker data array, and F (resp. G) is the real-part (imaginary-part) of the FFT of X . With reference to equation 1, it can be shown that for $1 \leq i \leq \frac{n}{2} + 1$,

$$C_0 = \frac{1}{n}F(1); A_i = \frac{2}{n}F(i+1); B_i = \frac{2}{n}G(i+1).$$

To reconstruct the filtered tick data, you pick those components of F and G that contribute significantly (compared to the others) to the original signal. You compute a **magnitude** array whose i -th entry is $\sqrt{F(i)^2 + G(i)^2}$. Following this, you sort the entries of **magnitude** and pick the top-5, top-10, top-15 values of **magnitude** (and these are the sinewave and cosinewave components that are the “strongest” in the original tick data). After you picked the top, stronger components, in the F and G arrays, you zero-out the remaining entries of the F and G arrays, as their contributions are insignificant. You reconstruct the signal using the inverse of the FFT, which is `RealFFT1(F, G, X)`; in the NEWMAT library. This is how the red, black, blue curves of figure 2 are generated.

In this programming assignment you are going to reconstruct a filtered-version of a (highly noisy/fluctuating) price-path of an instrument¹.

Using the `hint.cpp` provided on Compass, and the video instructions, I want you to write a C++ program that takes the ticker/price-path data and takes as command-line inputs: (1) the #significant-terms in the fourier-series/fourier-expansion of the price-path, (2) the #data-points in the ticker-data, (3) input-file name that contains the ticker-data, and (4) the output-file that should contain the filtered-data (after the program completes). A sample output is shown in figure 1. The video instructions also tell you how to generate a sample output as shown in figure 2.

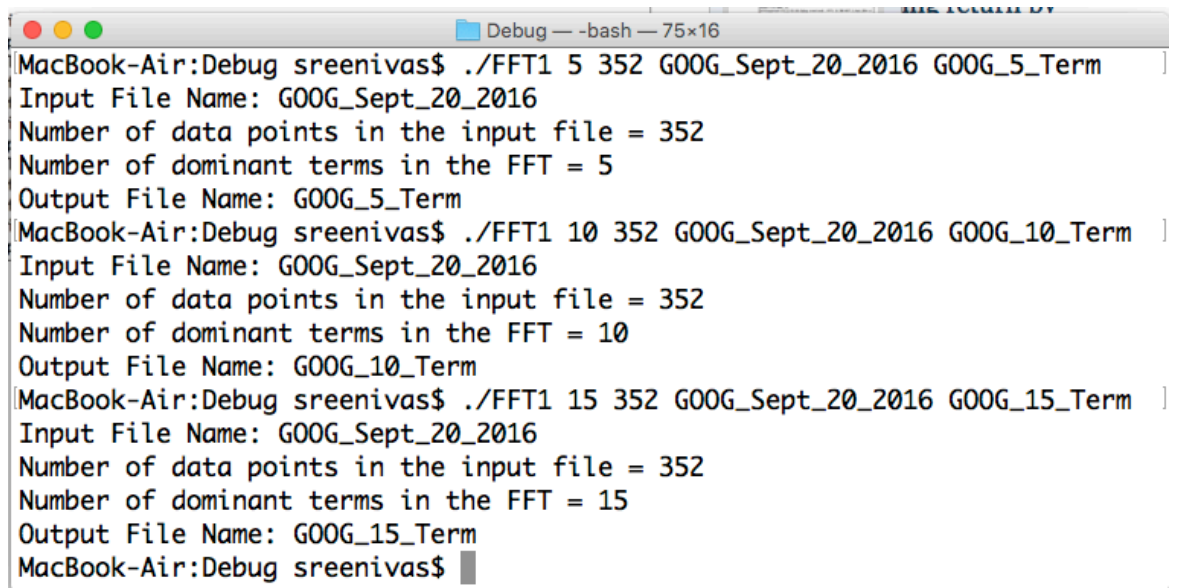
Here is what I want from you for this programming assignment

1. A short write-up that explains what your code does. No more than a page.
2. C++ code with appropriate Class definitions along the lines of `hint.cpp` that accomplishes what is required.

References

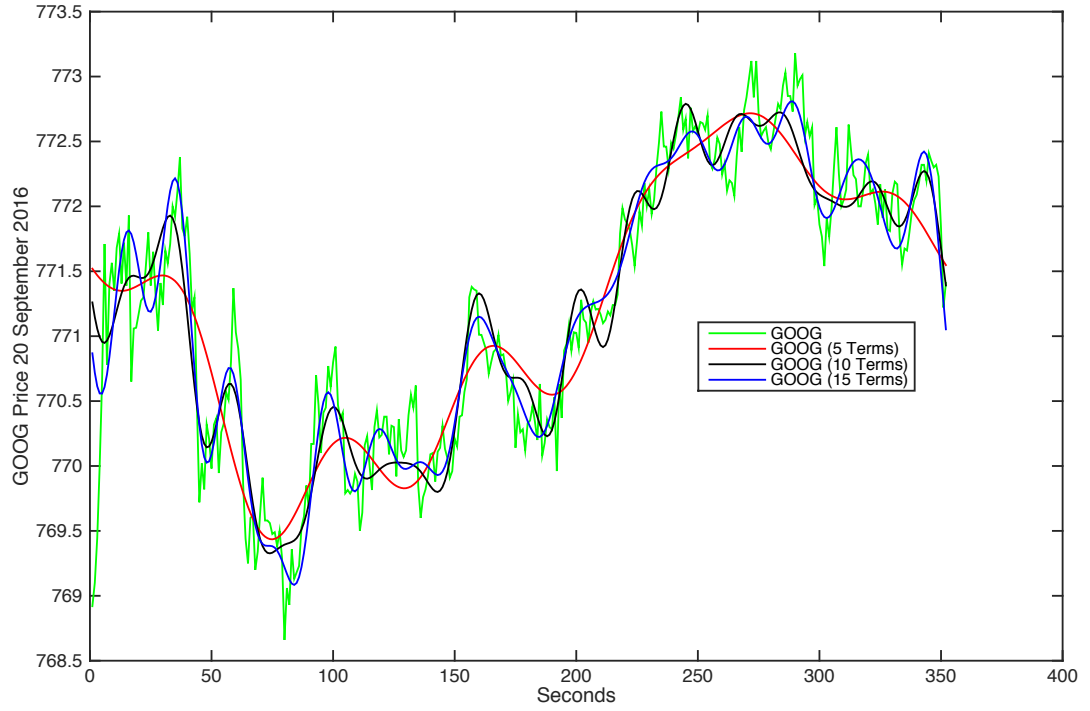
- [1] A.V. Oppenheim and R.W. Schaffer. *Digital Signal Processing*. Prentice Hall, 1975.

¹You can get the second-by-second price-path of *GOOG* and *MSFT* recorded at COB on 20 September, 2016. The *GOOG* data has 352 data-points, while the *MSFT* data has 390 data-points.

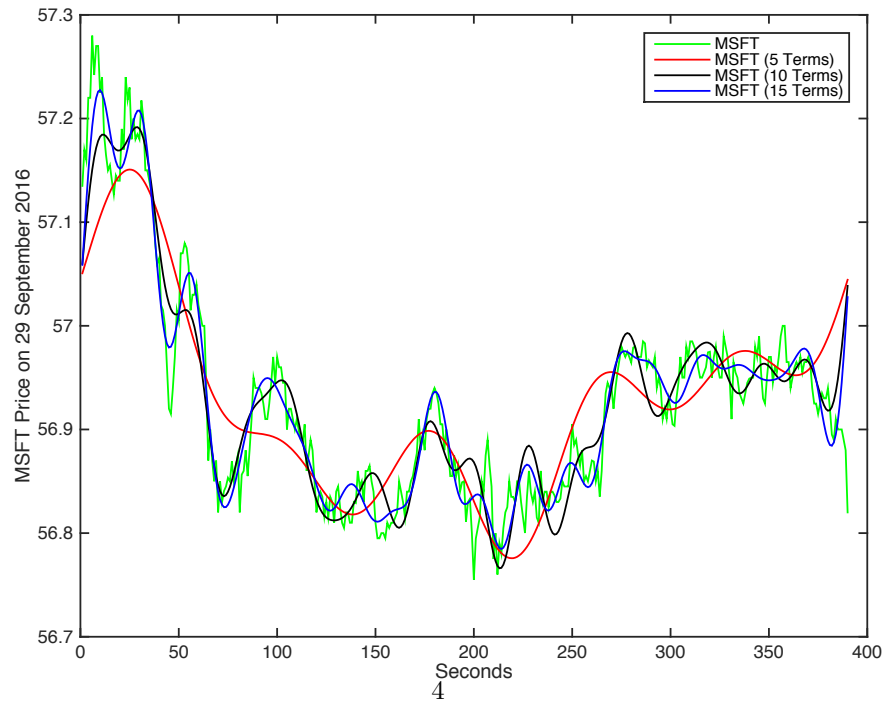


```
MacBook-Air:Debug sreenivas$ ./FFT1 5 352 G00G_Sept_20_2016 G00G_5_Term
Input File Name: G00G_Sept_20_2016
Number of data points in the input file = 352
Number of dominant terms in the FFT = 5
Output File Name: G00G_5_Term
MacBook-Air:Debug sreenivas$ ./FFT1 10 352 G00G_Sept_20_2016 G00G_10_Term
Input File Name: G00G_Sept_20_2016
Number of data points in the input file = 352
Number of dominant terms in the FFT = 10
Output File Name: G00G_10_Term
MacBook-Air:Debug sreenivas$ ./FFT1 15 352 G00G_Sept_20_2016 G00G_15_Term
Input File Name: G00G_Sept_20_2016
Number of data points in the input file = 352
Number of dominant terms in the FFT = 15
Output File Name: G00G_15_Term
MacBook-Air:Debug sreenivas$
```

Figure 1: A sample output.



(a) The original, and fitered-price-paths for GOOG on 9/20/2016



(b) The original, and fitered-price-paths for MSFT on 9/20/2016

Figure 2: The original, and fitered-price-paths for GOOG and MSFT on 9/20/2016.