Ay190 – Worksheet 06
David Vartanyan
Date: January 29, 2014

1 Exercise 1

1.1 a

My function equivalent of dft(x) is labeled f(x), the Fourier transform of x. By running the hashtag lines, we see that f(x) is identical to g(x), the machine definition of the Fourier transform of x using numpy.fft.fft(x).

1.2 b

We run timeit to find the time it takes to run f(x) ten times on a random vector of the length N. I create and plot an array of N vs the time to run f(x), where $N = range(10, 110, 10)$.

Interestingly, I find that the time to run f(x) scales as N, not as N^2. This may be due to my stepwise definition of f(x). Since I define f(x) with all the subvariables (w(N), k, b) pre-defined, I wondered if this would affect the time to call f(x). However, after redefining f(x) to encompass all subvariables, I still got a linear fit of timeit with respect to N.

1.3 c

See Figure 1. I plot the results for f(x) and g(x) on two differently scaled y-axes. Note how g(x) takes several orders of magnitude less time to run and further does not depend on N as stiffly, illustrating the $NlogN$ convergence for the fast Fourier transform.

1.4 d

One way I made my code faster is referenced in Section 1.2. I simply redefined f(x) so that all the subvariables (e.g. w(N), k, b) were defined separately so calling f(x) was faster. Bending the rules? Sure. Returning a faster evaluation of f(x)? Definitely. For everything else, there's MasterCard.

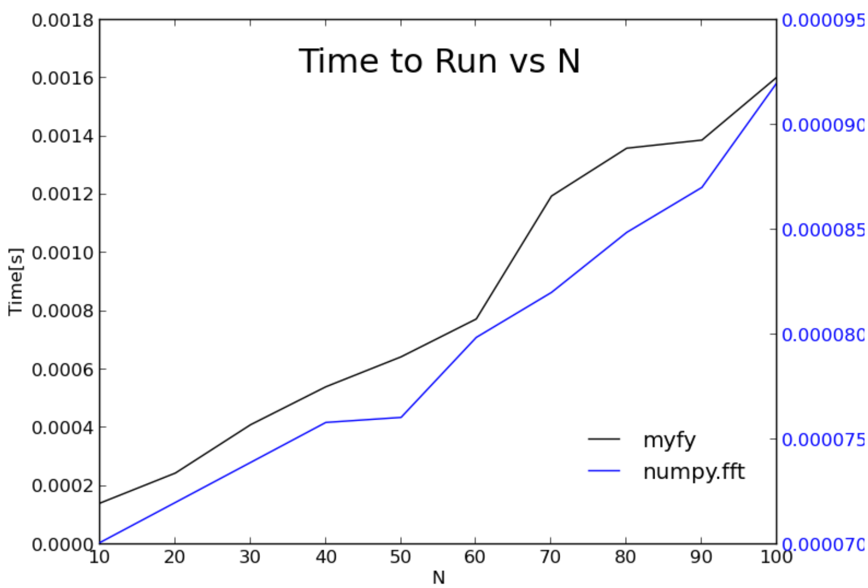

Figure 1: Timeit vs N.