Assignment 3.

Given March 11, due March 25.

Objectives: To use the FFT to compute a discrete convolution and compare with a direct computation; to understand how the FFT works.

Obtain an FFT routine. You can use routine `fft / ifft` in MATLAB or you can take one from netlib or from *Numerical Recipes* or you can write your own.

(1) Let s be a vector of length N that samples the function $s(t) = sin(t)$ at N equally spaced points in the interval $[0, 2\pi)$. Let $\tilde{s}$ be like s but with some random noise added: $\tilde{s} = s + 0.1 * randn(N, 1)$.

Let r be a vector of length N that samples the function $r(t) = \exp(-t^4)$ at N equally spaced points in the interval $[-\pi, \pi)$, with the first $N/2$ entries of r corresponding to positive times $(t = 0, \Delta t, \ldots, (N/2-1)\Delta t)$ and the last $N/2$ entries of r corresponding to negative times $(t = -(N/2)\Delta t, -(N/2-1)\Delta t, \ldots, -\Delta t)$. Compute the discrete *convolution* of r and $\tilde{s}$:

$$(r * \tilde{s})_k = \sum_{j=0}^{N-1} r_j \tilde{s}_{k-j}, \quad k = 0, 1, \ldots, N-1.$$

(Remember that the subscript $k-j$ means $k-j$ mod N, since $\tilde{s}$ is periodic with period N.)

Do the computation in two different ways: first by using the FFT, and then by directly computing the required sums. Compare your answers to make sure they are correct. Count the number of floating point operations required for each method with $N = 32$, $N = 64$, and $N = 128$. (If the direct calculation with $N = 128$ takes too long on your machine, then do $N = 16$ instead). Is it what you expected?

Turn in: a plot of $\tilde{s}(t)$, $r(t)$, and $(r*\tilde{s})(t)$; also the operation counts for $N = 32$, 64, 128 with a sentence or two of explanation.

(2) This exercise does *not* require programming. Let f be a vector of length 8 with components denoted $f_0, f_1, \ldots, f_7$. Write down (and explain clearly) the sequence of operations you would use to compute the discrete Fourier transform of f, using the Danielson-Lanczos lemma.