Math 56 Compu & Expt Math, Spring 2013: Homework 4

due 10am Thursday April 25th

Exploring beautiful properties and applications of things Fourier.

1. Fourier series theory. As in lecture, let $\|\cdot\|$ be the L_2-norm on $(0, 2\pi)$.
   (a) What is $\|e^{inx}\|$ for any $n \in \mathbb{Z}$?
   (b) Find a function orthogonal to $f(x) = x$ on $0 \le x < 2\pi$.
   (c) Combine the Fourier series for the 2π-periodic function defined by $f(x) = x$ for $0 \le x < 2\pi$ that we computed on the worksheet with Parseval's relation to evaluate $\sum_{m=1}^{\infty} m^{-2}$
   (d) Compute the Fourier series for the 2π-periodic function defined by $f(x) = x^2$ in $-\pi \le x < \pi$. [Hint: shift the domain of integration to a convenient one.] Comment on how $\hat{f}_m$ decays for this continuous (but not C^1) function compared to the discontinuous function in (c).
   (e) Take the expression for a general f written as a Fourier series. By taking the derivative of both sides (you may assume that you can pass the derivative through the sum), prove that if $|f'|$ is bounded, $|\hat{f}_m| = O(1/|m|)$.
   (f) Use the previous idea to prove a bound on the decay of Fourier coefficients when f has k bounded derivatives. What bound follows if $f \in C^\infty$ (arbitrarily smooth)? Can you give this a name?

2. Getting to know your DFT. Use numerical exploration followed by proof (each proof is very quick):
   (a) Produce a color image of the real part of the DFT matrix F for $N = 256$. Explain one of the *curves* seen.
   (b) What is F^2 ? [careful: matrix product, also don't forget the 0-indexing]. What does F^2 do to a vector? (this should be very simple!) Now, for general N, prove your claim [Hint: use ω]
   (c) What then is F^4 ? Prove this.
   (d) What are the eigenvalues of F? Use your previous result to prove this.
   (e) What is the condition number of F? Prove this using a result from lecture.

3. The power of trigonometric interpolation, i.e. using just a few samples of a periodic function to reconstruct the function *everywhere*. (Applications to modeling data, etc.)
   (a) Let's interpolate $f(x) = e^{\sin x}$. For $N = 40$, by using the $1/N$-weighted samples at the nodes $x_j = 2\pi j/N$, $j = 0, \ldots, N-1$, and `fft`, find $\tilde{f}_m$. Plot their magnitudes on a log vertical scale vs $m = 0, \ldots, N-1$. Relate to #1(f). By what $|m|$ have the coefficients decayed to $\varepsilon_{\text{mach}}$ times the largest? (This is the effective band-limit of the function at this tolerance.)
   (b) Using $\tilde{f}_m$ as good approximations to the true Fourier coefficients in $-N/2 < m < N/2$, plot the "intepolant" given by this truncated Fourier series, on the fine grid `0:1e-3:2*pi`. Overlay the samples $Nf_j = f(x_j)$ as blobs. [Hint: debug until the interpolant passes through the samples]
   (c) By looping over the above for different N, make a labeled semi-log plot of the maximum error (taken over the fine grid) between the interpolant and f, vs N, for even N between 2 and 40. At what N is convergence to $\varepsilon_{\text{mach}}$ reached? (Pretty amazing, eh?) Relate to (a).

4. Let's prove that, amongst all trigonometric polynomials of degree at most $N/2$, the $N/2$-truncated Fourier series for f is the *best approximation* to f in the $L_2(0, 2\pi)$ norm.

(a) Any trig. poly. can be written $\sum_{|n|<\leq N/2}(\hat{f}_n + c_n)e^{inx}$ for some coefficient "deviations" c_n. Write the squared L_2-norm of the function which is the difference of the above and the true f.

(b) Expand out to four terms, use the definition of $\hat{f}_n$, then expand further, and cancel stuff to leave all the c_n dependence in a sum of squares. Your LaTeX only needs to show 3-4 key steps. The proof is now easy.

(c) From that, extract an expression for the square of this (best) error.

5. How fast do things run?

(a) Consider the matvec problem computing $\mathbf{y} = A\mathbf{x}$ for A an n-by-n matrix. Write a little code using random A and $\mathbf{x}$ for $n = 4000$, which measures the runtime of Matlab's native `A*x` and your own naive double loop to compute the same thing. Express your answers in "flops" (flop per sec), and give the ratio. Marvel at how fast the built-in library is.

(b) Make a plot showing how many microseconds (μs $= 10^{-6}$ s) it takes to do a FFT of a random vector of length n, varying n over integers between 8100 to 8200 (use at least 100 repetitions for each n to get runtimes that are long enough to measure accurately). Overlay on your plot the reciprocal of the number of prime `factors` of n, scaled vertically so as to have the same max value. Which n has the fastest FFT (why?). How many times slower is the slowest?