Math 56 Compu & Expt Math, Spring 2013: Homework 5

due 10am Thursday May 2nd

This week less theory, but more fun real-world signal applications

1. Spectral differentiation, i.e. approximating the derivative of a function from N samples of its value.[1]

   (a) Adapt your code from #3(c) from the previous homework to compute the *derivative* of the interpolant on the fine grid, generated from the same sets of N samples of $e^{\sin x}$. Plot the convergence vs N of the max error from the analytically-known derivative of f on the fine grid. [Hint: what are the Fourier coefficients of the derivative?]

   (b) Repeat for $|\sin x|^3$. What type and order of convergence results? Is it as predicted by #1(f) from HW4?

2. Write your own FFT code for $N = 2^n$, using the Cooley-Tukey recursive algorithm, including a driver code that computes the 2-norm of the error between the output of your FFT and Matlab's native `fft`. Measure the ratio of your runtime to Matlab's on a vector of length 2^{16}. (Again, marvel at the speed of the FFTW library that Matlab uses.)

3. The power of the DFT to extract a periodic signal in the presence of noise.

   (a) Download and read in the audio file `signoise.wav`. This signal $\mathbf{f}$ is known to contain one frequency component on top of (iid, i.e uncorrelated) noise. Can you see any periodicity in the graph of $\mathbf{f}$ visually? Use the FFT to answer: what are the *two* dominant Fourier mode indices m ? How are their indices related? (Explain.) How are their coefficients related? Explain using a property of the input signal.

   (b) Use the information that the signal was sampled at 44100 Hz to compute the true (physical) dominant frequency in Hz.

   BONUS Listen to the signal; can you hear any tone above the noise? Discuss by comparing the amplitude of the desired component to the typical size of the signal $\mathbf{f}$.

4. Discrete convolution theory.

   (a) Consider the signal $f = [1, 1, 1, 0, 0, \ldots]^T$, where the dots indicate continuation by zeros to the length of the vector, and the signal $g = [1, 1, 1, 1, 0, \ldots]^T$. Assuming that the length of both the signals is $N = 6$, work out by hand the (periodic) convolution $f * g$.

   (b) Do the same except assuming that the length of both is $N = 5$.

   (c) The answer to one of (a) or (b) has the same nonzero vector elements as the *non-periodic* convolution $\sum_i f_i g_{j-i}$. Which? Use this to answer: to what length N signals of lengths N_1 and N_2 should be zero-padded so that the N-periodic convolution correctly computes the non-periodic one.

   (d) Prove commutativity: for all $f, g \in \mathbb{C}^N$, it holds that $f * g = g * f$.

   (e) For any $f, g \in \mathbb{C}^N$, what is the sum of the signal $f * g$ expressed in terms of simple properties of f and g? [Hint: use (a) to make a conjecture then prove it]

[1] You should compare this to finite-differencing, which is the other numerical differentiation method you have seen.

5. Using convolution to apply a "reverb effect" to an audio signal (i.e. digital signal processing).

   Use `audacity`, or any audio software of your choice, to record (or take from a music track) an interesting sound of duration at most a few seconds. Export it as a WAV file (mono, 44100 Hz sample rate) and read it into Matlab via `wavread`.

   Download and read in `impulseresponse.wav`, which is the sound of a clap in an echoey space, i.e. the response of the space to an impulse (delta function).

   Use `fft` and `ifft` to (periodically) convolve the two signals. Make sure to zero-pad the vectors to a length N long enough so that the result is correct as a non-periodic convolution.

   Finally, use e.g. `wavwrite(y,44100,'output.wav');` to write out the result as a WAV file, and include in your HW. It should be playable by any audio player—listen and enjoy!

6. 2D image de-blurring application, when aperture function known.

   (a) Download and `textread` into Matlab the data `blurry.txt` and reshape it into a 512-by-512 array. View in monochrome with `imagesc` and `colormap(gray(256)); axis equal`, and note that the text is unreadable! Similarly read in `aperture.txt` that was the aperture image used for blurring; it is a single block of nonzero pixels (but appears in four corners due to periodic wrapping).

   (b) Deconvolve the blurry image using the 2D version of the technique from lecture. [Hint: the 2D FFT is just like the 1D FFT except it takes a $N \times N$ array to another $N \times N$ array. See `fft2` and `ifft2`.] Include the resulting (crisp!) image in your HW.

   (c) Add iid Gaussian white noise of standard deviation 3×10^{-5} to the original blurry image—this models a *tiny* amount of measurement error (e.g. from a camera)—then repeat (b). Explain why the result looks as it does.[2] [Hint: look at the DFT of the aperture image.]

BONUS Invent an improved algorithm that reduces noise in the deconvolved image while still preserving resolution, and show your result. [Hint: see above hint!]

[2]This extreme sensitivity to noise is why I didn't have you read in the images in TIFF format, even though it is easy in Matlab: even the rounding error to 8-bit accuracy in the TIFF format introduces way too much noise!