18.336 Problem Set 1

Due Tuesday, 21 February 2006.

Problem 1: Trigonometric interpolation polynomials

You are given a function $f(x_n) = f_n$ at N points $x_n = \frac{2\pi}{N} n$, $n = 0, \cdots, N-1$, and you construct the trigonometric interpolation polynomial

$$f(x) = \sum_{k=0}^{N-1} c_k e^{ikx}$$

via the DFT. We showed in class that one would get the same coefficient c_k if you replace any e^{ikx} term with $e^{i(k+\ell_k N)x}$ for any integer ℓ_k (we showed for $\ell_k = 1$, but other values follow by induction). This means that there are actually many possible trigonometric interpolations with the same c_k coefficients passing through the same f_n.

(a) Assume N to be odd for simplicity. Show that the unique choice of integer ℓ_k's that *minimizes* the mean-square slope $\frac{1}{2\pi} \int_0^{2\pi} |f'(x)|^2 dx$ (for arbitrary f_n) is the "symmetric" polynomial:

$$f(x) = \sum_{k=0}^{(N-1)/2} c_k e^{ikx} + \sum_{k=(N+1)/2}^{N-1} c_k e^{i(k-N)x}$$

(b) Suppose that the f_n values are real numbers. We would like the interpolated $f(x)$ to be real as well for all $x \in [0, 2\pi)$. Show that the above "symmetric" polynomial satisfies this (i.e. real $f(x)$ for arbitrary real f_n). Is it the unique polynomial with real $f(x)$?

Problem 2: Solving Poisson's equation

Here, you will use Matlab to explore the solution of Poisson's equation $\phi''(x) = \rho(x)$ with periodic boundary conditions on $x \in [0, 2\pi)$ via spectral methods and FFTs. In particular, the following Matlab code solves for $\phi(x)$ given a "sawtooth" function $\rho(x) = x$ for $0 \le x < \pi$ and $\rho(x) = x - 2\pi$ for $\pi < x < 2\pi$ using $N = 100$ points to get an approximate solution $\phi_N(x)$:

```
N=100;
x = linspace(0,2*pi, N+1); x = x(1:end-1);
rho = x .* (x < pi) + (x - 2*pi) .* (x > pi); rho(N/2+1)=
k = [ 0:N/2-1, -N/2:-1 ]; k(1) = 1;
phi = ifft(- fft(rho) ./ k.^2);
```

(The k(1)=1 command is a hack to avoid dividing zero by zero when we divide by k^2.)

(a) Show by a simple finite-difference evaluation of $\phi_N''(x)$ (applying Matlab's "diff" command twice to "phi" and scaling by $1/\Delta x^2$) that $\phi_N''(x) \approx \rho(x)$.

(b) Estimate the error by repeating the calculation for $2N$ points to get $\phi_{2N}(x)$, and compute the root-mean-square difference

$$\Delta_N \equiv \sqrt{\frac{1}{N} \sum_{n=0}^{N} |\phi_N(\frac{2\pi}{N} n) - \phi_{2N}(\frac{2\pi}{N} n)|^2}$$

Compute Δ_N for a sequence of N values from $N = 10$ to $N = 10000$ and plot Δ_N vs. N on a log-log scale. You should get a power-law dependence (a straight line). What is the exponent?

Problem 3: Fast Fourier Transforms

Consider the Cooley-Tukey decimation-in-time (DIT) radix-2 FFT algorithm applied to compute the DFT of length $N = 2^m$. As we showed in class, this recursively decomposes the problem into two DFTs of length $N/2$ of the even- and odd-indexed inputs, respectively.

(a) Assume that the inputs are complex numbers, and that a complex addition takes 2 real additions and a general complex multiplication takes 4 real multiplications and 2 real additions, and that all twiddle factors (cosine and sine values) are precomputed. The *exact* count of real adds+multiplies is of the form $\# \cdot N \log_2 N + O(N)$. Show what # is. (Count multiplications by ± 1 and $\pm i$ as free, or equivalently subtraction counts as addition.)

(b) Suppose that the inputs are purely real, in which case the outputs c_k have the symmetry $c_{N-k} = c_k^*$. Show that, by applying radix-2 Cooley-Tukey directly to this data and tossing out the redundant computations, we can compute the DFT in $(\#/2) \cdot N \log_2 N + O(N)$ real adds+multiplies. (*Numerical Recipes* fans may know that you can compute a real-input DFT for even N by embedding it into a complex-input DFT of length $N/2$. Don't use this trick here.)