Assignment 7.

Given April 20, due end of finals week.

Objective: To code the FFT and see an application

This assignment makes use of a Fourier sine series, which is closely related to exponential Fourier series we have worked with. Because $e^{it} = \cos(t) + i\sin(t)$, the exponential Fourier coefficients,

$$\hat{f}_\alpha = \frac{1}{2\pi}\int_0^{2\pi} e^{-i\alpha t} f(t)dt \ ,$$

can be written as the sum of a cosine coefficient and a sine coefficient:

$$\hat{f}_\alpha = a_\alpha - ib_\alpha \ ,$$

where

$$a_\alpha = \frac{1}{2\pi}\int_0^{2\pi} \cos(\alpha t) f(t)dt \ ,$$
$$b_\alpha = \frac{1}{2\pi}\int_0^{2\pi} \sin(\alpha t) f(t)dt \ .$$

In a similar way, we can define a discrete cosine transform and a discrete sin transform from the discrete (exponential) Fourier transform.

The sine and cosine transform can be useful in certain circumstances, particularly when the function, f, has special properties. In particular, the Fourier sine series is useful if f is real and skew symmetric about $t = \pi$. This means that $f(\pi + t) = -f(\pi - t)$ for any t. Such functions may seem odd and unlikely to arise in practice. They do arise in the following way. Suppose that $f(t)$ is initially defined for t only in the range from 0 to π, and that f is continuous on that range, and that $f(0) = 0$ and $f(\pi) = 0$. In solving differential equations, these would be called "Dirichlet boundary conditions". A skew symmetric periodic function that is continuous satisfies these conditions automatically. Conversely, for any continuous function defined on the interval $[0, \pi]$ with Dirichlet boundary values, we can define a periodic skew symmetric function, defined for all t, with the same values in the original $[0, \pi]$ interval. First extend f to the complementary interval $[\pi, 2\pi]$ by skew symmetry: $f(\pi+t) = -f(\pi-t)$ for $0 \leq t \leq \pi$. Then define f for t outside the fundamental period $[0, 2\pi]$ by $f(t+2\pi n) = f(t)$ for $t \in [0, 2\pi]$ and n any integer. Of course, we never actually do this extension in the computer. Instead, we think of doing it to help us derive properties of the Fourier sine series using what we already know about the exponential Fourier series.

We can make a discrete version of this by thinking at first of the numbers f_k as being samples of a function: $f_k = f(t_k)$. If we have N points $t_k = k\Delta t$, $\Delta t = 2\pi/N$, then the midpoint, π will be one of the t_k only if N is even. Therefore, suppose $N = 2(M + 1)$ and that we have M points uniformly spaced *inside* the interval $[0, \pi]$:

$$t_k = k\Delta t \ , \quad \Delta t = \frac{\pi}{M+1} \ , \quad \text{for } k = 1, \ldots, M. \tag{1}$$

We can (mentally) extend the samples f_k, defined for $k = 1, \ldots, M$ to be defined for $k = 0, \ldots, N-1$ by skew symmetry. First define $f_0 = 0$ (consistent with the Dirichlet boundary condition). Then define $f_{k+M+1} = -f_k$ for $k = 0, \ldots, M$.

The exponential Fourier coefficients of the extended f give the Fourier sine coefficients of the original f. In the continuous case,

$$\hat{f}_\alpha = -ib_\alpha \quad , \quad \text{where } \; b_\alpha = \frac{1}{\pi}\int_0^\pi \sin(\alpha t) f(t) dt \quad . \tag{2}$$

and

$$f(t) = 2\sum_{\alpha=1}^{\infty} b_\alpha \sin(\alpha t) \quad . \tag{3}$$

The factor of 2 in (**??**) comes from the fact that the contributions from α and $-\alpha$ are equal, and we have kept only positive α terms. All the cosine coefficients, a_α are zero in this case, because the integrals

$$\int_0^{2\pi} \cos(\alpha t) f(t) dt$$

are zero.

1. Derive the discrete version of the Fourier sine series. This starts with M samples (real numbers) , f_1, ..., f_M, and produces M coefficients $\tilde{b}_1$, ..., $\tilde{b}_M$. You can find these formulae by using the DFT applied to the skew symmetric extended f.

2. Suppose that $M = 2^L - 1$. Derive a fast algorithm for computing the discrete sine coefficients that uses only real arithmetic. This will be called the FST (fast sine transform)

3. Assuming that the f_k are samples of a smooth, skew symmetric function, find an FST based algorithm that gives a spectrally accurate estimate of $f''(t_k)$ for $k = 1, \ldots, M$. Call this operation A. That is, there is an $M \times M$ real matrix so that if $Af = g$, then g_k is the estimate of $f''(t_k)$ from the samples f_j, $j = 1$, ..., M. Write a program to calculate Af using teh FST. Check numerically that the matrix A is symmetric. Check that it produces spectrally accurate results. For the latter, you will have to be careful in choosing f.

4. Write a program to solve the partial differential equation

$$\frac{\partial u}{\partial t} = \frac{1}{2}\frac{\partial^2 u}{\partial x^2} + u^2 \quad , \tag{4}$$

with boundary conditions

$$u(x=0,t) = 0 \quad , \quad u(x=\pi,t) = 0 \quad , \tag{5}$$

and intitial conditions

$$u(x, t=0) = z\sin(x) \quad . \tag{6}$$

Find an approximate solution by replacing u by a discrete approximation $u_k(t) \approx u(x_k, t)$, where $x_k = k\Delta x$, $\Delta x = \pi/(M+1)$, and $k = 1, \ldots, M$. Approximate the equation (??) by

$$\dot{u}_k(t) = \frac{1}{2}\left(Au(t)\right)_k + u_k^2 \ . \tag{7}$$

This is a system of M ordinary differential equations. Compute the operation of A using the algorithm from part 3. Solve these ordinary differential equations using the fourth order Runge Kutta method. You must take the time step, Δt, to be very small for large N. At a minimum, you must take $\Delta t < \frac{1}{4}\Delta x^2$, or the numerical solution will "blow up"[1]. Do a convergence study to verify that your method has spectral accuracy. Compute solutions for various values of z. For small z the solutions tranquilly go to zero as $t \to \infty$. For larger z, the solution develops a spike near $x = \pi$ which leads to infinite[2] u at $x = \pi$ for some finite t. Observe how the computational method breaks down as the spike forms.

[1] This blow up has nothing to do with physical blowup of corectly computed solutions. It is a numerical artifact assiciated with errors in the discrete approximation.

[2] This blow up is real, and very hard to compute, as you will see.