Solving the KdV equation with Fourier Transforms

March 21, 2007

We start by writing the KdV equation in the form

$$u(x,t)_t + 3\left(u^2(x,t)\right)_x + u(x,t)_{xxx} = 0. \tag{1}$$

The third derivative term causes problems for numerical solutions, being what is known as a 'stiff term'. In order to get rid of it, we use an integrating factor in Fourier space, as follows. We denote the Fourier transform (FT) by $\mathcal{F}$,

$$\mathcal{F}(u)(k,t) = \int_{-\infty}^{\infty} e^{ikx} u(x,t). \tag{2}$$

Taking the FT of (1), we get

$$\mathcal{F}(u)_t + 3ik\mathcal{F}(u^2) - ik^3 F(u) = 0, \tag{3}$$

since

$$\mathcal{F}\left(\frac{\partial u}{\partial x}\right) = ik\mathcal{F}(u)$$

with my definition (2) of the FT. We can now remove the stiff term by defining

$$\hat{U}(k,t) = e^{-ik^3 t}\mathcal{F}(u), \tag{4}$$

in terms of which (3) becomes

$$\begin{aligned} \hat{U}_t &= -3ike^{-ik^3 t}\mathcal{F}(u^2) \\ &= f(t,U) \equiv -3ike^{-ik^3 t}\mathcal{F}\left(\left(\mathcal{F}^{-1}\left(e^{ik^3 t}\hat{U}\right)\right)^2\right), \end{aligned} \tag{5}$$

where we have written u in terms of $\hat{U}$ using the inverse FT,

$$u(x,t) = \mathcal{F}^{-1}\mathcal{F}(u) = \mathcal{F}^{-1}\left(e^{ik^3 t}\hat{U}\right).$$

Equation (5) may look unpleasant, but computers can perform FTs extremely quickly using the Fast Fourier Transform (FFT) algorithm, for which Python has a very straightforward algorithm. All we need to do is to evaluate the right-hand side of (5), using the relevant Python library, and then use it to evolve forward in time with some sort of finite-differences stepping method. For better stability, I used a Runge-Kutta fourth-order time stepping routine, which is in

fact very simple and is explained well in the relevant Wikipedia article. I realise that $f(t, U)$ is not, strictly speaking, a function of U, involving instead two Fourier transforms, but it is the notation I used in my program.

This is more or less the whole story, except that the program needs to take account of a few subtleties in how Python actually implements the FFT algorithm. The most important of these is that the algorithm expects the function u to be periodic with period 2π, whereas our initial condition is not actually periodic at all. However, since $\mathrm{sech}^2(x)$ tends to zero very rapidly for large x, if we choose a large enough L, the initial configuration can be taken to be periodic on the interval $[-L/2, L/2]$. I then have to rescale the function by a factor of $2\pi/L$ to get the FTs to work out correctly, which is why there are quite a few instances of this factor in the program in seemingly mysterious places.