Numerically Solving Continuity Equations

Carl Denton

September 26, 2016

1 Introduction

Continuity equations are of interest for a variety of reasons, the most important of which is that they model a wide range of physical phenomena including conservation of mass in fluids. However, they are generally far more difficult to solve than the closely related Advection-Diffusion equations due to the fact that they have no diffusive term and are thus far more unstable.

Here we implement the upwind and SHASTA methods for solving these equations and test a number of their initial conditions.

Code at `https://github.com/cfdenton/ac274` in continuity.py

2 Problem

The continuity equation is given by

$$\frac{\partial}{\partial t}\varphi + \nabla \cdot \vec{J}[\varphi] = S$$

for flux $\hat{J}$, density φ, and source S. We focus here on flux of the form

$$\vec{J}[\varphi] = \vec{u}(x,t)\varphi - D(x)\nabla\varphi$$

This equation becomes particularly difficult to solve for $D = 0$, as the lack of diffusion makes it much harder to maintain stability.

3 Methods

3.1 Upwind Finite Differences

The first method we explore is the upwind finite differences scheme. This is similar to standard centered finite-differences, with the main difference being that the spatial derivatives are one-sided, in the direction of positive advection. That is, to solve the continuity equation

$$\frac{\partial}{\partial t}\varphi = -\frac{\partial}{\partial x}u(x)\varphi$$

we use the scheme

$$\frac{\varphi_j^{n+1} - \varphi_j^n}{h} = \begin{cases} \frac{u_j\varphi_j^n - u_{j-1}\varphi_{j-1}^n}{d} & \text{if } u_j \geq 0 \\ \frac{u_{j+1}\varphi_{j+1}^n - u_j\varphi_j^n}{d} & \text{if } u_j < 0 \end{cases}$$

This is typically quite stable— unfortunately, this stability comes at the cost of high degrees of numerical diffusion.

3.2 SHASTA

The second method we implement is the SHASTA algorithm, a type of flux-corrected transport. The essential intuition of the algorithm is to move each grid point with the motion of the fluid, and then interpolate the results back to the original grid. Finally an antidiffusive term is added to compensate for the excess numerical diffusion created in the process. The result is an algorithm of the form

$$\varphi_j^{n+1} = \frac{1}{2}Q_-^2\left(\varphi_{j-1}^n - \varphi_j^n\right) + \frac{1}{2}Q_+^2\left(\varphi_{j+1}^n - \varphi_j^n\right) + (Q_+ + Q_-)\varphi_j^n$$

where

$$Q_\pm = \left(\frac{1}{2} \mp u_j\frac{h}{d}\right) / \left[1 \pm (u_{j\pm1} - u_j)\frac{h}{d}\right]$$

followed by an antidiffusive term given by

$$\bar{\varphi}_j^n = \varphi_j^n - f_{j+1/2} + f_{j-1/2}$$

where

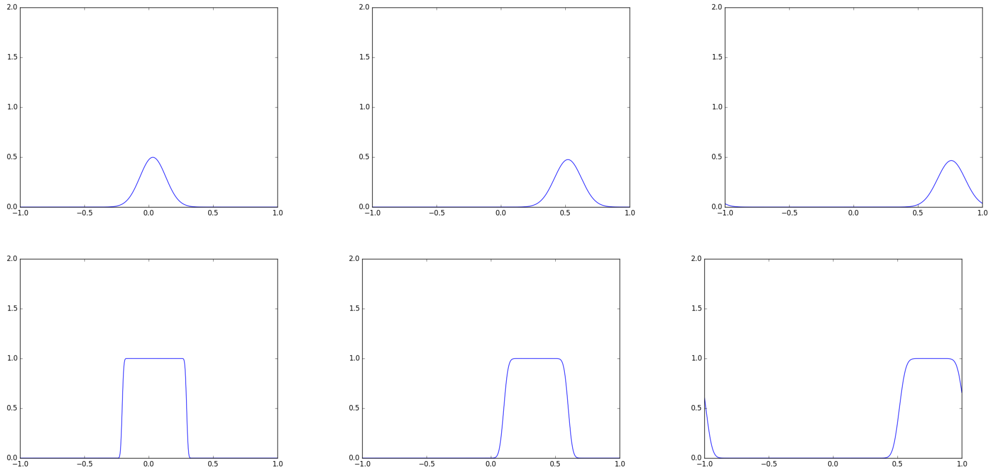
$$f_{j+1/2} = \text{sgn }\Delta_{j+1/2}\max\left\{0, \min\left\{\Delta_{j-1/2}\text{ sgn }\Delta_{j+1/2}, \frac{1}{8}\left|\Delta_{j+1/2}\right|, \Delta_{j+3/2}\text{ sgn }\Delta_{j+1/2}\right\}\right\}$$

and $f_{j-1/2}$ is obtained by setting $j \to j-1$. This set of conditions ensures positivity while eliminating excess diffusion. The authors of the original paper note that further accuracy can be achieved by modifying the $\frac{1}{8}$ term.

4 Experiments

4.1 Upwind

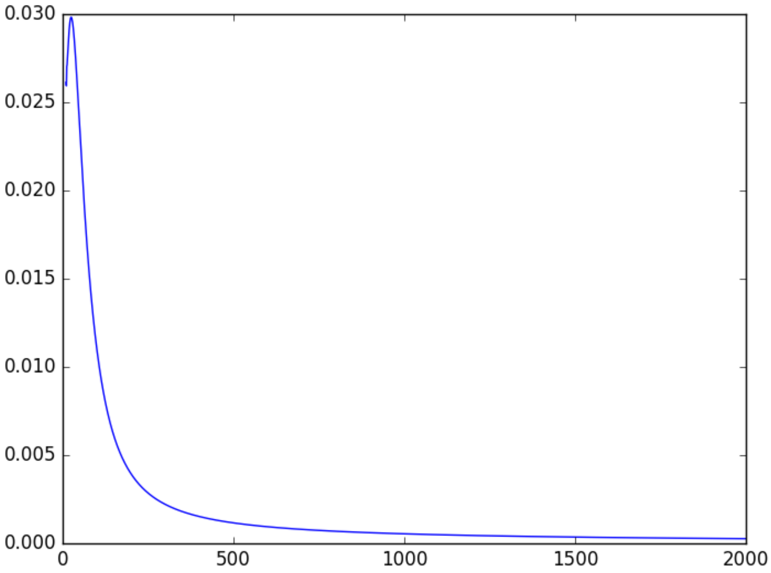
We first test the method with gaussian and box initial conditions, under spatially uniform advection. The results are plotted below.



Note the numerical diffusion, particularly noticeable in the case of a box wave, which appears as a side effect of the method.

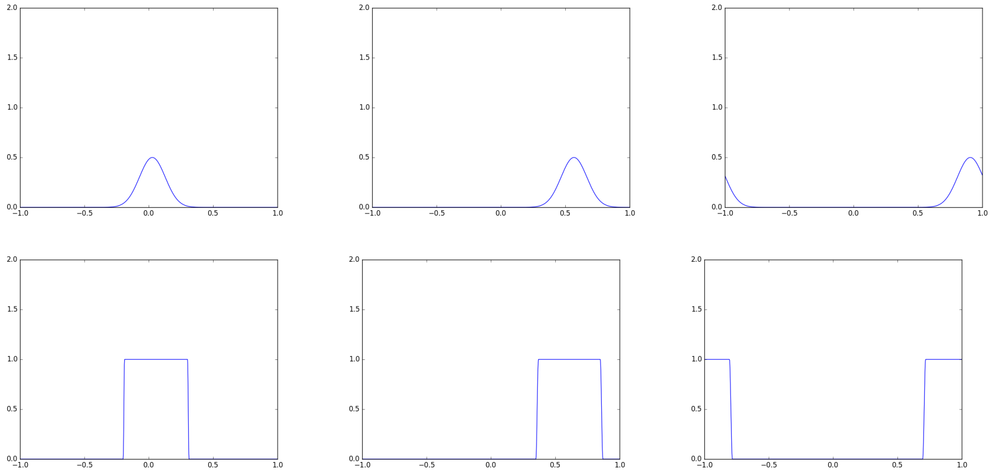
4.1.1 Validation

We perform a grid convergence analysis by running the simulation with uniform advection for grid sizes N and $2N$, where $N = 10, \ldots, 2000$. The root-mean-square error of the result at time step $t = 100$ between the simulation on a grid of size N and a grid of size $2N$ is plotted below. Clearly the solution converges quite well for fine meshes.



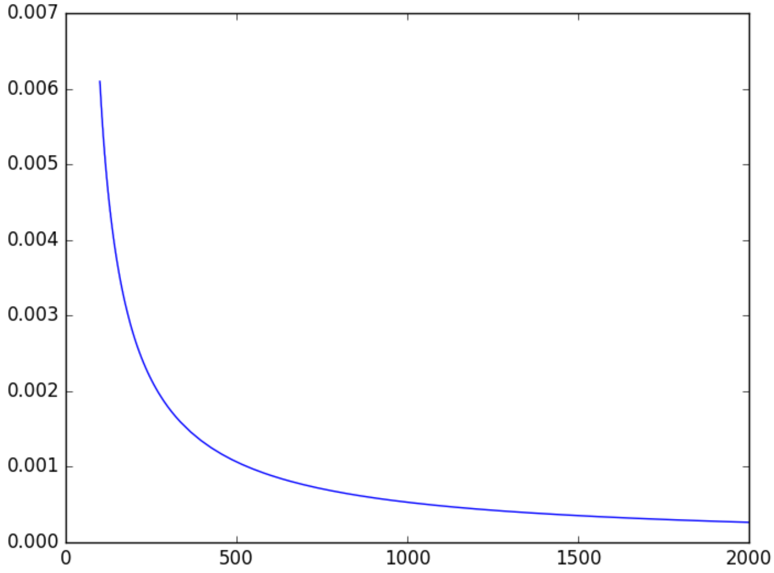
4.2 SHASTA

We perform the same experiments using the SHASTA method, with the results shown below. In this case numerical diffusion plays a far lesser role; the box wave can be seen to maintain its shape almost perfectly.



4.2.1 Validation

We perform a grid convergence analysis identical to the one for the upwind method, with the following results:



Again, the solutions converge well.