2D Advection-Diffusion Equation: Pseudo-Spectral Methods and Finite Difference Schemes

Bin Hoang, Horacio Lopez, Matteo Polimeno

San Diego State University

05/03/2018

Outline

Derivation

- Fluid with constant density
- The vorticity $\omega = \nabla \times \mathbf{u}$. Assuming $\nabla \cdot u = 0$, from [1] we write the momentum Navier-Stokes equation

$$\rho \frac{\mathcal{D}\mathbf{u}}{\mathcal{D}t} = -\nabla p + \rho \mathbf{g} + \mu \Delta \mathbf{u} \tag{1}$$

which can use to derive an equation for the vorticity, see [1]. Rescaling leads to

$$\frac{\mathcal{D}\mathbf{u}}{\mathcal{D}t} = -\frac{1}{\rho} \nabla p + \mathbf{g} + \nu \Delta \mathbf{u} \tag{2}$$

where $\nu = \frac{\mu}{\rho}$ is the kinematic viscosity and $\mathbf{g}$ is gravity. It holds $\mathbf{g} = -\nabla \phi$.

Derivation

In order to derive the vorticity equation we take the curl of both sides and, when the vector calculus and the algebra settle, we end up with

$$\frac{\mathcal{D}\omega}{\mathcal{D}t} = (\omega \cdot \nabla)\mathbf{u} + \nu\Delta\omega \tag{3}$$

which is the field equation governing the vorticity in a fluid with constant density.

In 2D space (x,y), it holds

$$\mathbf{u} = \begin{bmatrix} u \\ v \\ 0 \end{bmatrix}, \tag{4}$$

which implies

$$(\omega \cdot \nabla)\mathbf{u} = \left(\omega_x \partial_x + \omega_y \partial_y + \omega_z \partial_z\right) \mathbf{u} = 0 \tag{5}$$

therefore, we end up with

$$\frac{\mathcal{D}\omega}{\mathcal{D}t} = \nu \Delta \omega \tag{6}$$

which is the vorticity equation in 2D.

As we know, in fluids it is typical to work with streamfunctions. Therefore let $\psi(x, y, t)$ be a streamfunction defined as

$$u = -\partial_y \psi, \; v = \partial_x \psi \tag{7}$$

After more algebra, we can write the advection-diffusion equation for the vorticity

$$\partial_t \omega + [\psi, \omega] = \nu \Delta \omega \tag{8}$$

where $[\psi, \omega] = \partial_x \psi \partial_y \omega - \partial_y \psi \partial_x \omega$

Moreover, the vorticity can now be written as

$$\omega = \partial_x v - \partial_y u = \Delta\psi \tag{9}$$

which represents the Poisson equation and thus gives us a set of equations to be solved as

$$\partial_t\omega + [\psi, \omega] = \nu\Delta\omega, \tag{10}$$

$$\Delta\psi = \omega \tag{11}$$

Problem set-up

- We are given initial vorticity, ω_0.
- Solve for streamfunction, ψ_0, by solving poisson equation

$$\Delta\psi_0 = \omega_0$$

- Use time stepper to solve advection-diffusion and obtain ω_1
- Use new vorticity to obtain new streamfunction, and repeat.

Solving Poisson Equation

- Poisson equation:

$$\Delta\psi = \omega$$

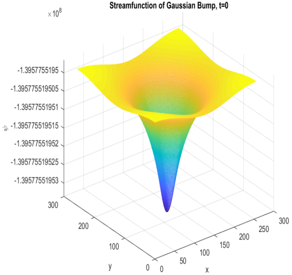
- Laplacian, Δ, given by $\partial_x^2 + \partial_y^2$.
- If we fast fourier transform, $FFT2$, our matrix, ω, and reshape/flatten into a vector, we can use kronecker tensors to represent the operators. So,

$$\widehat{\Delta\psi} = \widehat{\omega}$$

$$\Rightarrow \left(\widehat{\partial_x^2} + \widehat{\partial_y^2}\right)\widehat{\psi} = \widehat{\omega}$$

$$\Rightarrow \widehat{\psi} = \frac{\widehat{\omega}}{\left(\widehat{\partial_x^2} + \widehat{\partial_y^2}\right)}$$

Solving Poisson Equation



- Exponential, addition, subtraction, multiplication, division operations done elementwise.
- To avoid division by zero, first term of $\vec{k} = 1e - 6$ or other small number.
- To plot streamfunction, reshape resulting vector from Poisson solution, ψ, into $(N \times N)$ matrix and perform inverse fast fourier transform, $IFFT2$.

Time-stepping Vorticity

- Now that we have our streamfunction for time step n, ψ_n, we solve for ω_{n+1} by solving
$$\frac{\partial \omega_{n+1}}{\partial t} = \nu \Delta \omega_n - \left(\frac{\partial \psi_n}{\partial x} \frac{\partial \omega_n}{\partial y} - \frac{\partial \psi_n}{\partial y} \frac{\partial \omega_n}{\partial x} \right)$$
- All of this is done in fourier space, so we calculate the differentials
$$\frac{\partial \psi_n}{\partial x}; \; \frac{\partial \omega_n}{\partial y}; \; \frac{\partial \psi_n}{\partial y}; \; \frac{\partial \omega_n}{\partial x}$$
using kronecker tensor multiplication.
- We then calculate the advection term, $[\psi_n, \omega_n]$ by reshaping each of the above calculated derivatives into an $(N \times N)$ matrix, applying $IFFT2$, and multiplying and adding the respective values to get $[\psi_n, \omega_n]$.

Time-stepping Vorticity

- After obtaining $[\psi_n, \omega_n]$, we go back into fourier space by applying $FFT2$ and row wise reshaping into a vector of length N^2.
- I selected built in Runge-Kutta 4 with adaptive time-stepping.
- After obtaining next value for vorticity,ω, we again solve the poisson to solve for streamfunction, ψ, and apply the timestepper again.

Results

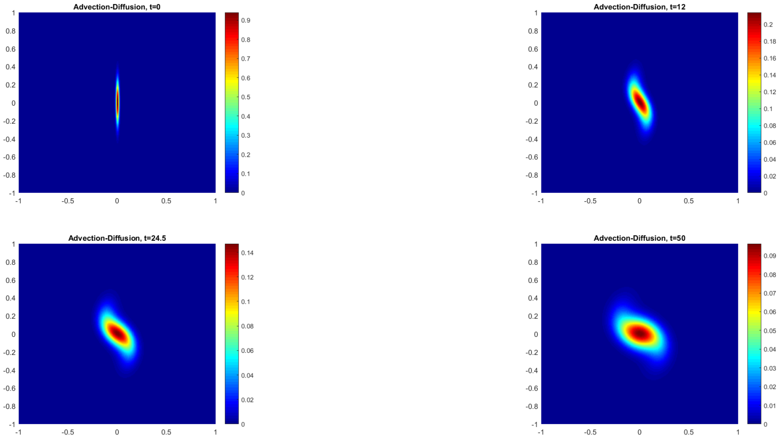


Figure: $\nu = 0.0001$, Gaussian Bump ω_0, and RK-4 timestepper

Solving the Advection-Diffusion using Finite Different Method

- 1) Solving the Poisson Equation
  - Introducing finite difference methods for Poisson Equation: Jacobi, Gauss-Seidel, and SOR. For this problem, we will only solve the Poisson Equation on the square grid
  - Comparing finite difference schemes with FFT.
- 2) Updating the Vorticity
  - Time Splitting Method
  - Comments on speed

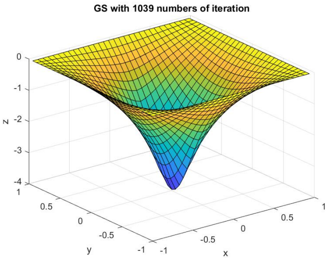
Jacobi, Gauss Seidel, and SOR

- Given a Poisson equation $\nabla^2\psi = w$, let h be the length of the space grid, the Jacobi scheme is given by:
  - $v_{i,j}^{k+1} = \frac{1}{4}(v_{i+1,j}^{k} + v_{i-1,j}^{k} + v_{i,j+1}^{k} - v_{i,j-1}^{k} - h^2 w_{i,j})$.
- Similar, Gauss Seidel scheme is :
  - $v_{i,j}^{k+1} = \frac{1}{4}(v_{i+1,j}^{k} + v_{i-1,j}^{k+1} + v_{i,j+1}^{k} - v_{i,j-1}^{k+1} - h^2 w_{i,j})$.
- Last and not least, the SOR scheme is:
  - $v_{i,j}^{k+1} = \frac{bb}{4}(v_{i+1,j}^{k} + v_{i-1,j}^{k+1} + v_{i,j+1}^{k} - v_{i,j-1}^{k+1} - h^2 w_{i,j}) + (1 - bb)v_{i,j}^{k}$

Jacobi, Gauss Seidel, and SOR (part 2)

- Results for the same w_0, we get

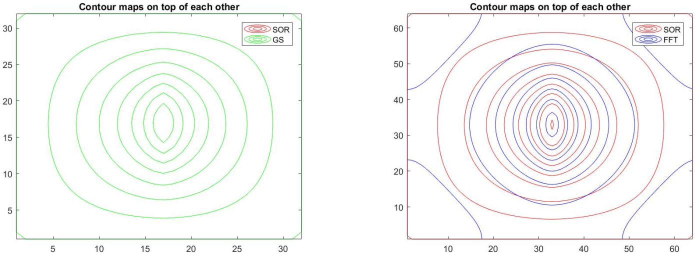
Figure 1: SOR converges more than 10 times faster.



Comparision with FFT

- Since what we are really after are the derivatives of the stream function and not the stream function itself, it makes sense to make comparison using contour plots.

Figure 2: FFT and SOR are very similar, but not quite the same.



Update the vorticity using finite difference scheme

- Wave-like Behavior:

$$\frac{\partial w}{\partial t} + [\Psi, w] = 0$$

  - forward Euler: unstable for all λ
  - leap-frog(2,2): stable for $\lambda < 1$
- Heat-like Behavior

$$\frac{\partial w}{\partial t} = \nu \nabla^2 \omega$$

  - forward Euler: stable for $\lambda < \frac{1}{2}$
  - leap-frog(1,2): unstable for $\lambda < 1$
- For our model, we used FTCS for heat-like portion and RK-4 for the wave-like portion of the advection diffusion equation.

Discussion

Psuedo Spectral Method:

- Pros:
  - Much higher speed, $\mathcal{O}(N Log N)$
  - Allows for better resolution; higher grid point number.
  - Spectral Accuracy.
- Cons:
  - Have to work in periodic square grid.

Finite Differencing Method:

- Pros:
  - Allows for work in rectangular grids
  - Can use periodic, Neumann, and Dirichelet boundary conditions.
- Cons:
  - Very slow compared to Pseudo Spectral method.

- Kundu P., Cohen I, Dowling D. *Fluid Mechanics*. Elsevier, 2016
- J.Nathan Kutz. *Data-Driven Modeling & Scientific Computation Methods for Complex Systems and Big Data*. Oxford University Press, 2013.
- Peter J. Schmit. *Dynamic Mode Decomposition of Numerical and Experimental Data*. Journal of Fluid Mechanics, vol. 656, 2010, pp. 5–28.