AMATH 481/581 Autumn Quarter 2024

Homework 5: Vorticity-Streamfunction Equations

DUE: Friday, November 22 at midnight

The time evolution of the vorticity $\omega(x, y, t)$ and streamfunction $\psi(x, y, t)$ are given by the governing equations:

$$\omega_t + [\psi, \omega] = \nu \nabla^2 \omega \tag{1}$$

where $[\psi, \omega] = \psi_x \omega_y - \psi_y \omega_x$, $\nabla^2 = \partial_x^2 + \partial_y^2$, and the streamfunction satisfies

$$\nabla^2 \psi = \omega \tag{2}$$

Initial Conditions: Assume a Gaussian shaped mound of initial vorticity for $\omega(x, y, 0)$. In particular, assume that the vorticity is elliptical with a ratio of 4:1 or more between the width of the Gaussian in the x- and y-directions. I'll let you pick the initial amplitude (one is always a good start). In most applications, the diffusion is a small parameter. This fact helps the numerical stability considerably. Here, take $\nu = 0.001$.

Boundary Conditions: Assume periodic boundary conditions for both vorticity and streamfunction. Also, I'll let you experiment with the size of your domain. One of the restrictions is that the initial Gaussian lump of vorticity should be well-contained within your spatial domains.

Numerical Integration Procedure: Discretize (2nd order) the vorticity equation and use ODE45 to step forward in time with **solve_ivp**.

(a) Solve these equations where for the streamline ($\nabla^2 \psi = \omega$) use a Fast Fourier Transform (NOTE: set $k_x(0) = k_y(0) = 10^{-6}$).

ANSWERS: With $x, y \in [-10, 10]$, $n = 64$, $\omega(x, y, 0) = \exp(-x^2 - y^2/20)$ and $tspan = 0 : 0.5 : 4$, write out the solution of your numerical evolution for the vorticity from **solve_ivp** as A1.

(b) Solve these equations where for the streamline ($\nabla^2 \psi = \omega$) use the following methods (NOTE: Take $A(0, 0) = 2$ instead of $A(0, 0) = -4$):

- A/b
- LU decomposition
- BICGSTAB
- GMRES

Compare all of these methods with your FFT routine developed in part (a). You can informally check this with the following code:

```
import time
start_time = time.time()  # Record the start time
end_time = time.time()  # Record the end time
elapsed_time = end_time - start_time
print(f"Elapsed time: {elapsed_time:.2f} seconds")
```

In particular, keep track of the computationl speed of each method. Also, for BICGSTAB and GMRES, for the first few times solving the streamfunction equations, keep track of the residual as a function of the number of iterations needed to converge to the solution. Note that you should adjust the tolerance settings in BICGSTAB and GMRES to be consistent with your accuracy in the time-stepping. Experiment with the tolerance to see how much more quickly these iteration schemes converge.

ANSWERS: With $x, y \in [-10, 10]$, $n = 64$, $\omega(x, y, 0) = \exp(-x^2 - y^2/20)$ and $tspan = 0 : 0.5 : 4$, write out the solution of your numerical evolution for the vorticity from **solve_ivp** as A2 for $\mathbf{A}\backslash\mathbf{b}$ and A3 for the **LU** method.

NOTE: For the LU method, when you solve you have to use the following code structure for $\mathbf{Ax} = \mathbf{b}$

```
from scipy.linalg import lu, solve_triangular
P, L, U = lu(A)

Pb = np.dot(P, b)
y = solve_triangular(L, Pb, lower=True)
x = solve_triangular(U, y)
```

This will make to just forward- or backward-substitute for the LU since the solve command does not exploit that.

(c) Try out these initial conditions with your favorite/fastest solver on the streamfunction equations.

- Two oppositely “charged” Gaussian vorticies next to each other, i.e. one with positive amplitude, the other with negative amplitude.
- Two same “charged” Gaussian vorticies next to each other.
- Two pairs of oppositely “charged” vorticies which can be made to collide with each other.
- A random assortment (in position, strength, charge, ellipticity, etc.) of vorticies on the periodic domain. Try 10-15 vorticies and watch what happens.

(d) Make a 2-D movie of the dynamics. Color and coolness are key here. I would very much like to see everyone’s movies and you can put these up on your github.