# EMCH 792 Project 1- Simulation of a Passive Scalar Field in a Taylor-Green Vortex

Spencer Schwartz

## 1 Time Step computation

The time step size constraints are

$$\Delta t < \frac{\Delta x}{\mathrm{CFL}|u_{max}|} \tag{1}$$

and

$$\Delta t < \frac{(\Delta x)^2}{4\nu}. \tag{2}$$

Eq. (1) and (2) refer to the constraints caused by the convective and diffusive terms, respectively. For this project, CFL = 0.5 to maximize stability. For inviscid cases, Eq. (1) is used while the minimum $\Delta t$ between (1) and (2) is used for viscid cases. Every case is ran to $t = 10$.

## 2 Initial Conditions

The domain is a 1x1 square with 128x128 cells using symmetric boundary conditions for the tracer field, $T$. The velocity field is given as

$$u = -\sin \pi x \cos \pi y, v = \cos \pi x \sin \pi y \tag{3}$$

and the initial tracer field is defined by

$$T(x, y, t = 0) = exp\left[-\left(\frac{x - 0.75}{0.025\pi}\right)^2 - \left(\frac{y - 0.75}{0.025\pi}\right)^2\right] \tag{4}$$

and is illustrated in Figure 1.

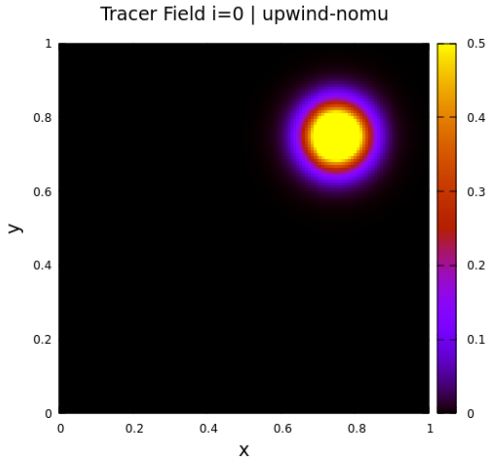


**Figure 1:** Initial tracer field at t = 0, used for all cases.

# 3 Results

For each case, inviscid and viscid, we test the 1st order upwind and 2nd order central method to solve the advection term.

## 3.1 Inviscid Flow

For this project, the problem was solved using Basilisk and a custom solver developed in C++. Four snapshots at t = 0.39, 1.95, 3.90, and 9.76 (corresponding to i = 100, 500, 1000, 2500) are shown and compared. First, Figure 2 and 3 show the evolution of the tracer field using the upwind method. As one can see, excellent agreement is found between our C++ program and the one made in Basilisk. A notable feature of the solution is the congregation of tracer values in the corners

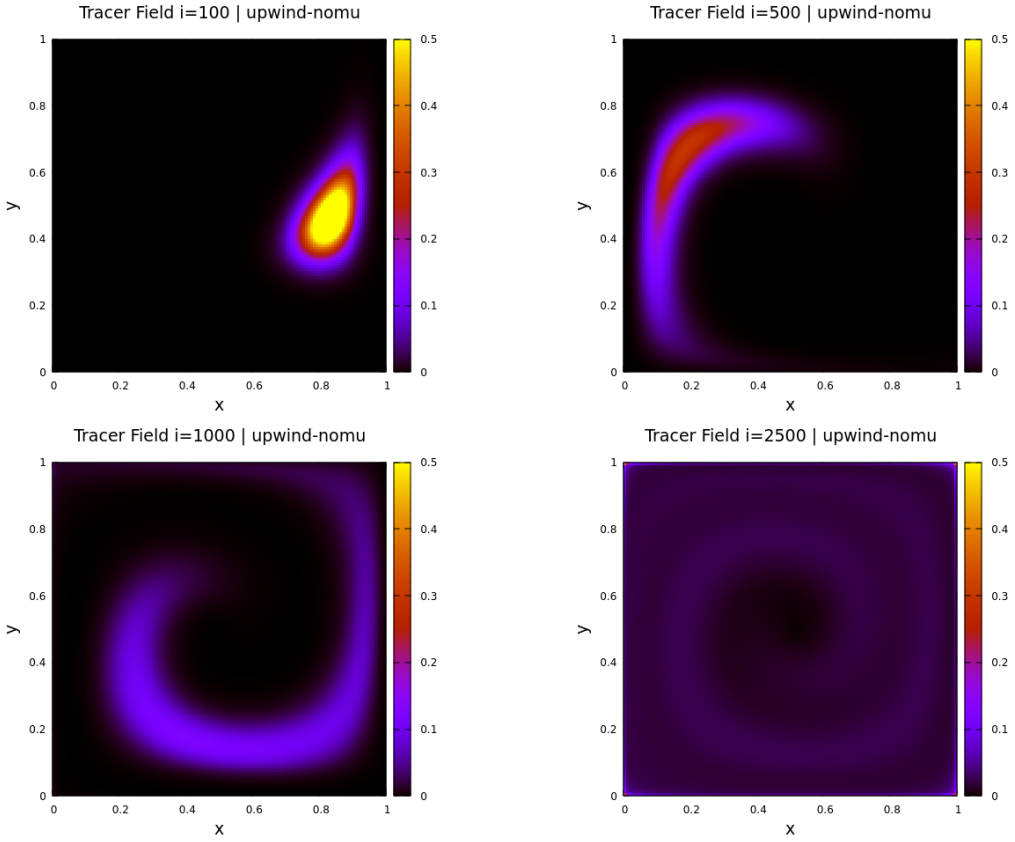


**Figure 2:** C++ program results for inviscid flow using the upwind method to solve the advection term.

of the domain. This is likely caused by the numerical diffusion introduced in the upwind method, allowing the tracer field to diffuse into areas with zero velocity. Furthermore, numerical diffusion is what makes the upwind method conditionally stable for solving the advection equation.

Other than the upwind method, we run an identical case except using the central method to determine $T_{face}$. As expected, the simulation fails unconditionally due to the absence of any diffusion, i.e., artificial numerical or physical. Somewhat surprisingly, both programs fail in almost identical ways, as shown in Figures 4 and 5. These similarities highlight the validity of the C++ program with the more robust Basilisk software.

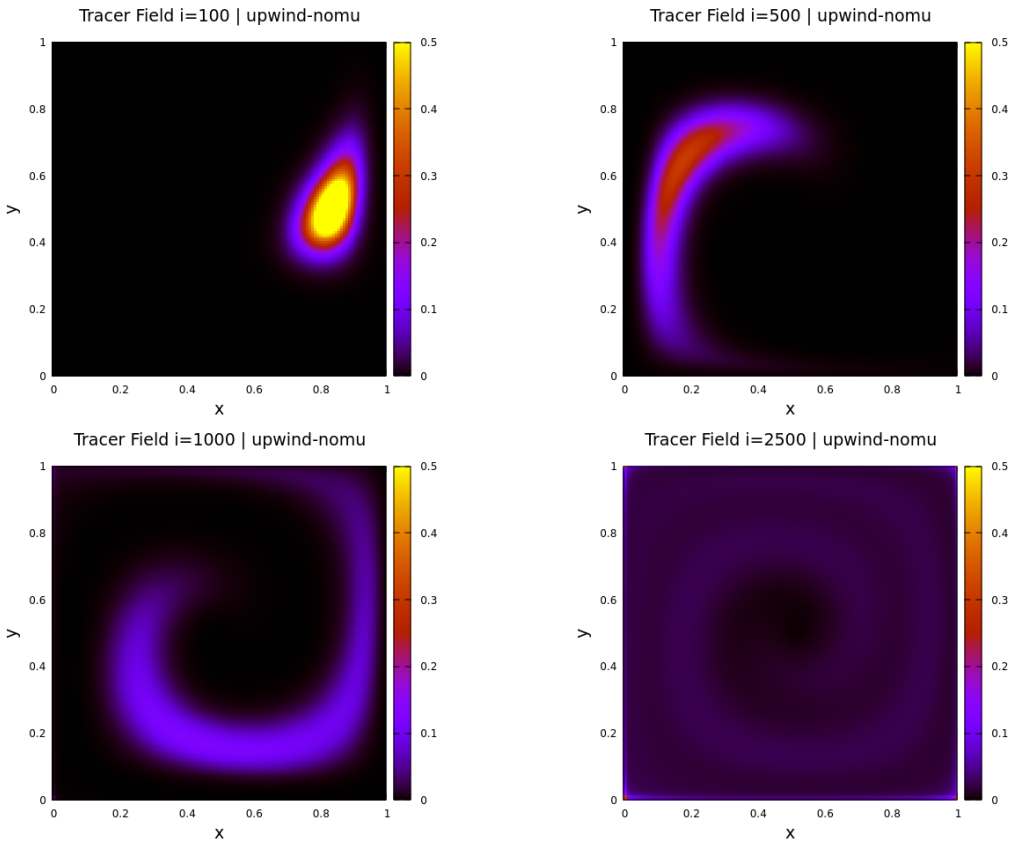


**Figure 3:** Basilisk program results for inviscid flow using the upwind method to solve the advection term.

### 3.2 Viscid Flow

Introducing the diffusive term in the advection equation helps stabilize otherwise unstable methods, such as the central method. This effect is directly shown in Figures 6 and 7, which use the central method to solve the advection-diffusion equation. Equation (2) significantly reduces the $\Delta t$ which increases the number of time steps. Therefore, in Figure 6-9, i = 1000; 10,000; 15,000; 786,000, corresponds to t = 0.01, 0.13, 0.19, and 9.92, resp.

Unlike the inviscid simulation, the additional diffusive term quickly disperses the tracer field, and by t=0.2, the tracer field is almost homogeneous throughout the domain. This is supported by simulations using the upwind method, as shown in Figures 8-9.

## 4 Conclusion

Simulations of a passive scalar field in a Taylor-Green vortex were developed and ran in two separate programs: one coded in Basilisk while the other coded in source C++. In each program, we used the upwind and central method to solve the 2D advection equation and noticed the unconditionally unstable nature of the central method compared to the conditionally stable upwind method. Furthermore, each program is used to solve the 2D advection-diffusion equation using the central and upwind methods. It is shown that the introduction of diffusivity through the kinematic

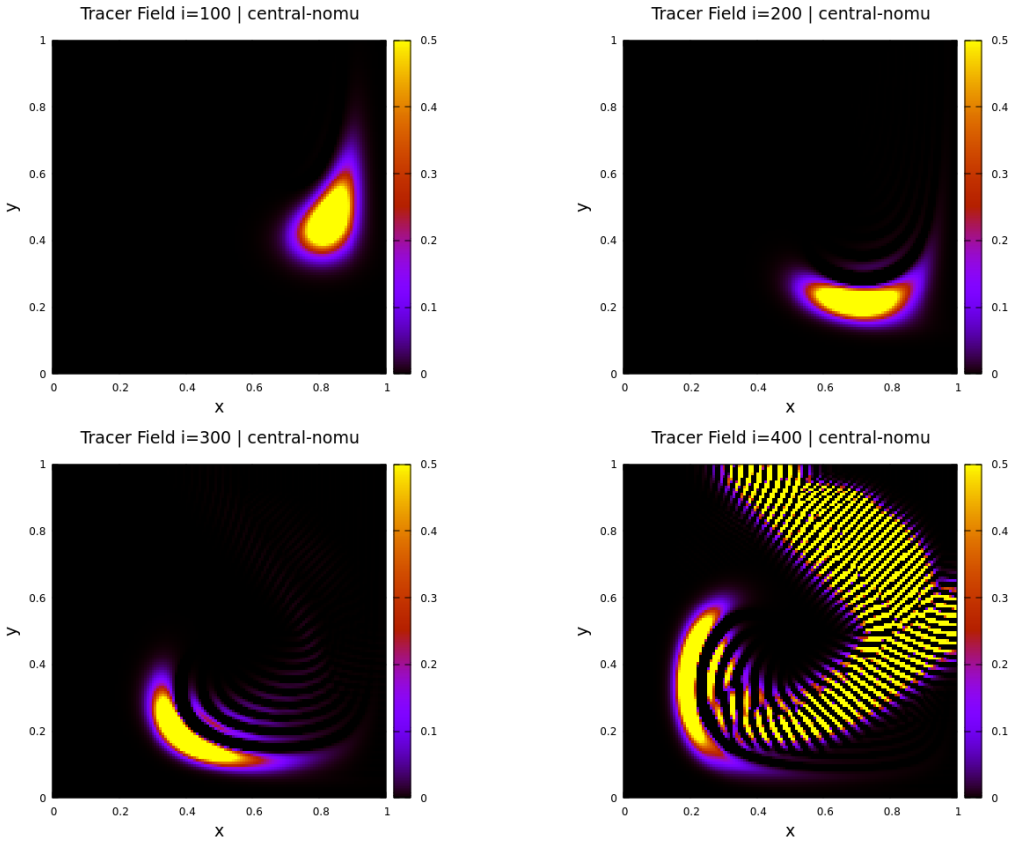


**Figure 4:** C++ program results for inviscid flow using the central method to solve the advection term.

viscosity stabilizes the central method while also quickly dispersing the tracer field in and out of the domain. Overall, excellent agreement is found between the custom C++ program and the one coded in Basilisk.

## 5 Resources

All code for this project can be found here:
https://github.com/SpencerSchwart/multiphase-cfd/tree/main/project1

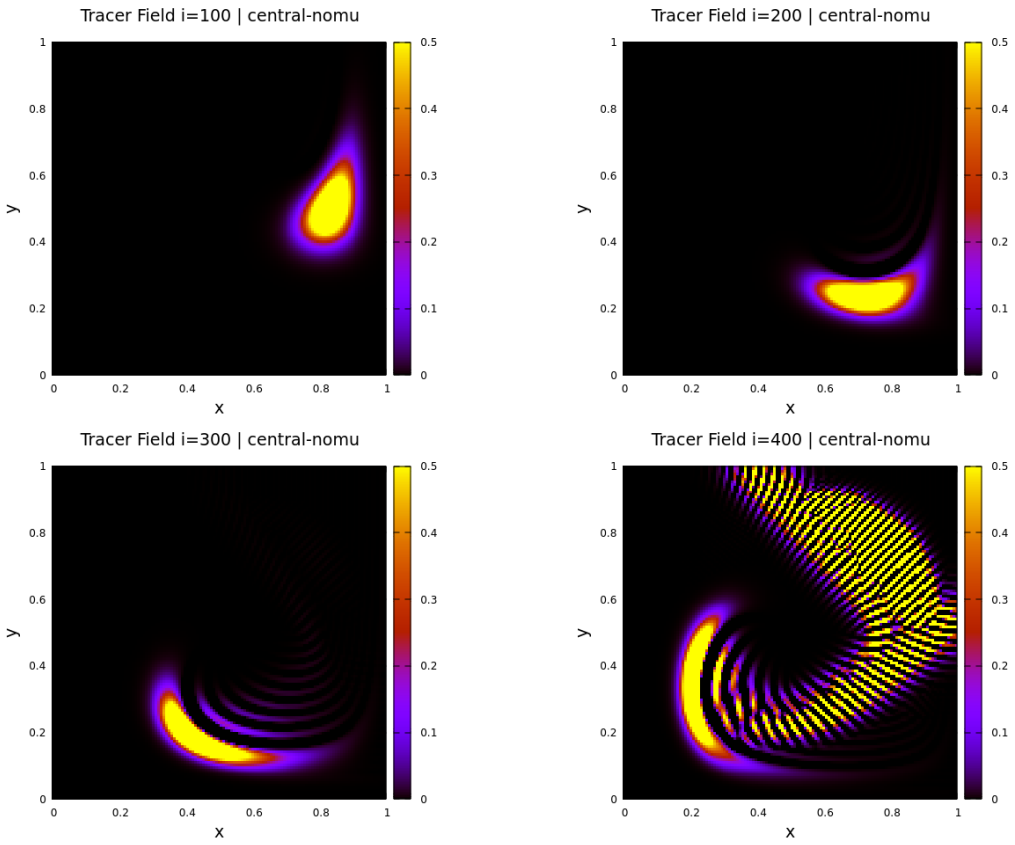


**Figure 5:** Basilisk program results for inviscid flow using the central method to solve the advection term.

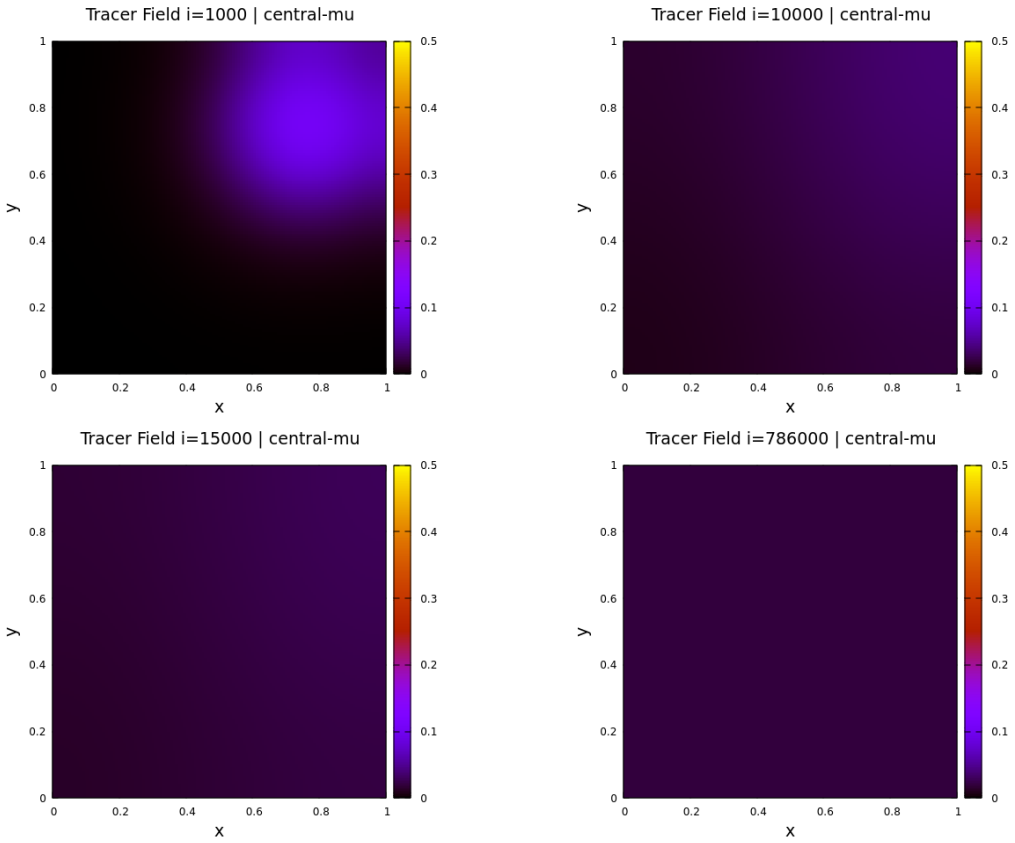


**Figure 6:** C++ program results for viscid flow using the central method to solve the advection term.

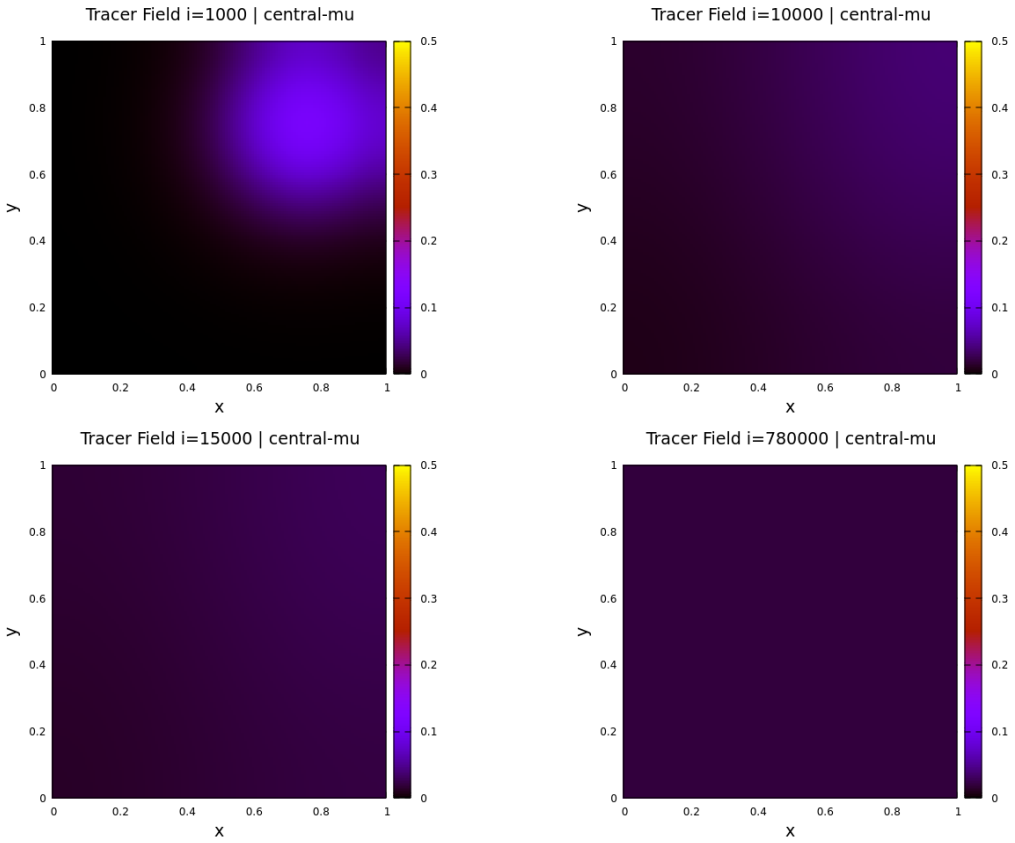


**Figure 7:** Basilisk program results for viscid flow using the central method to solve the advection term.

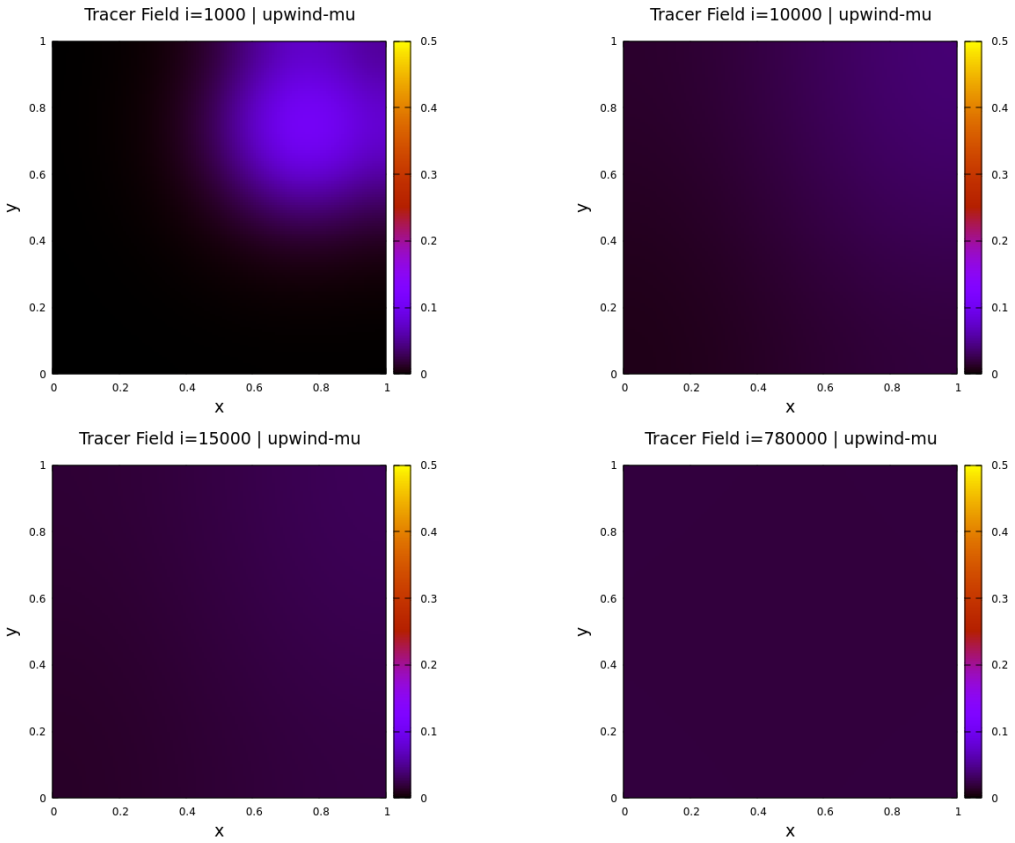


**Figure 8:** C++ program results for viscid flow using the upwind method to solve the advection term.

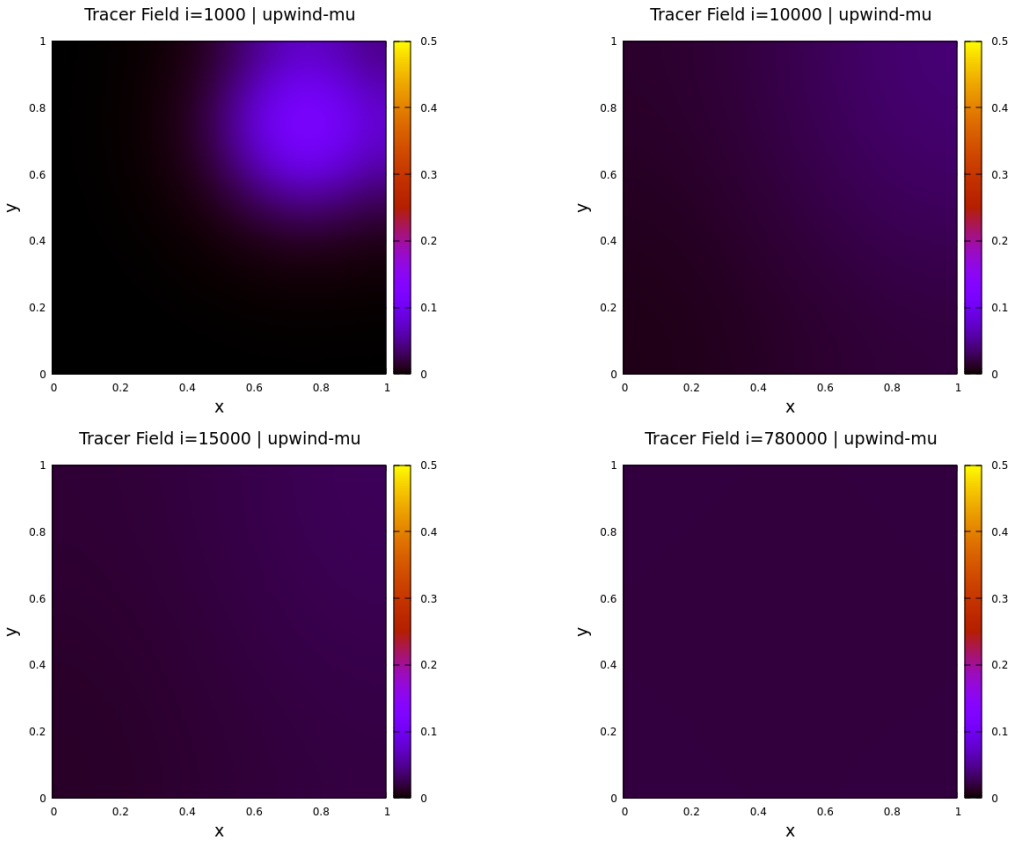


**Figure 9:** Basilisk program results for viscid flow using the upwind method to solve the advection term.