Zihan Xu

Homework 2

1. *Semi-Lagrangian method for the advection equation.*

Consider the square domain $\Omega = [-1, 1]^2$ and the linear advection equation

$$\frac{\partial \phi}{\partial t} + \mathbf{V} \cdot \nabla \phi = 0 \quad x \in \Omega, \tag{1}$$

with the following velocity field, initial condition and final time

$$\mathbf{V} = (-y, x),$$
$$\phi(x, y, 0) = \sqrt{(x - 0.25)^2 + y^2} - 0.2 t_f = 2\pi$$

Solve the above problem numerically using a second-order (in time and space) Semi-Lagrangian method. At any point $(\mathbf{x^{n+1}}, t^{n+1})$, the departure point $\mathbf{x_d}$ is constructed using a mid-point rule(RK2):

$$\mathbf{x}^* = \mathbf{x^{n+1}} - \frac{\Delta t}{2} \mathbf{V}(\mathbf{x^{n+1}}, t^{n+1})$$
$$\mathbf{x_d} = \mathbf{x^{n+1}} - \Delta t \mathbf{V}(\mathbf{x}^*, t^{n+\frac{1}{2}})$$

To compute the interpolation of the solution at $(\mathbf{x^d}, t^n)$, $\tilde{\phi}(\mathbf{x^d}, t^n)$, find to which grid cell C the point $\mathbf{x_d}$ belongs to and use ENO inspired quadratic interpolation.

(a) Test your method for increasing resolution and various $\frac{\Delta x}{\Delta t}$ ratios (for example 0.5, 1, 5 and 10). Compute the error and study the convergence of the method.

I used $(N-1) \times (N-1)$ grid in the domain $[-1, 1] \times [-1, 1]$ for N = 17, 33, 65, 129, since in Grid2d, we defined N as the number of nodes. In other words, my grid sizes are $\Delta x = 1/8, 1/16, 1/32$, and $1/64$. Here, I show the final numerical results of these different grid sizes with CFL $= \frac{\Delta x}{\Delta t} = 0.5$ in Figure 1.

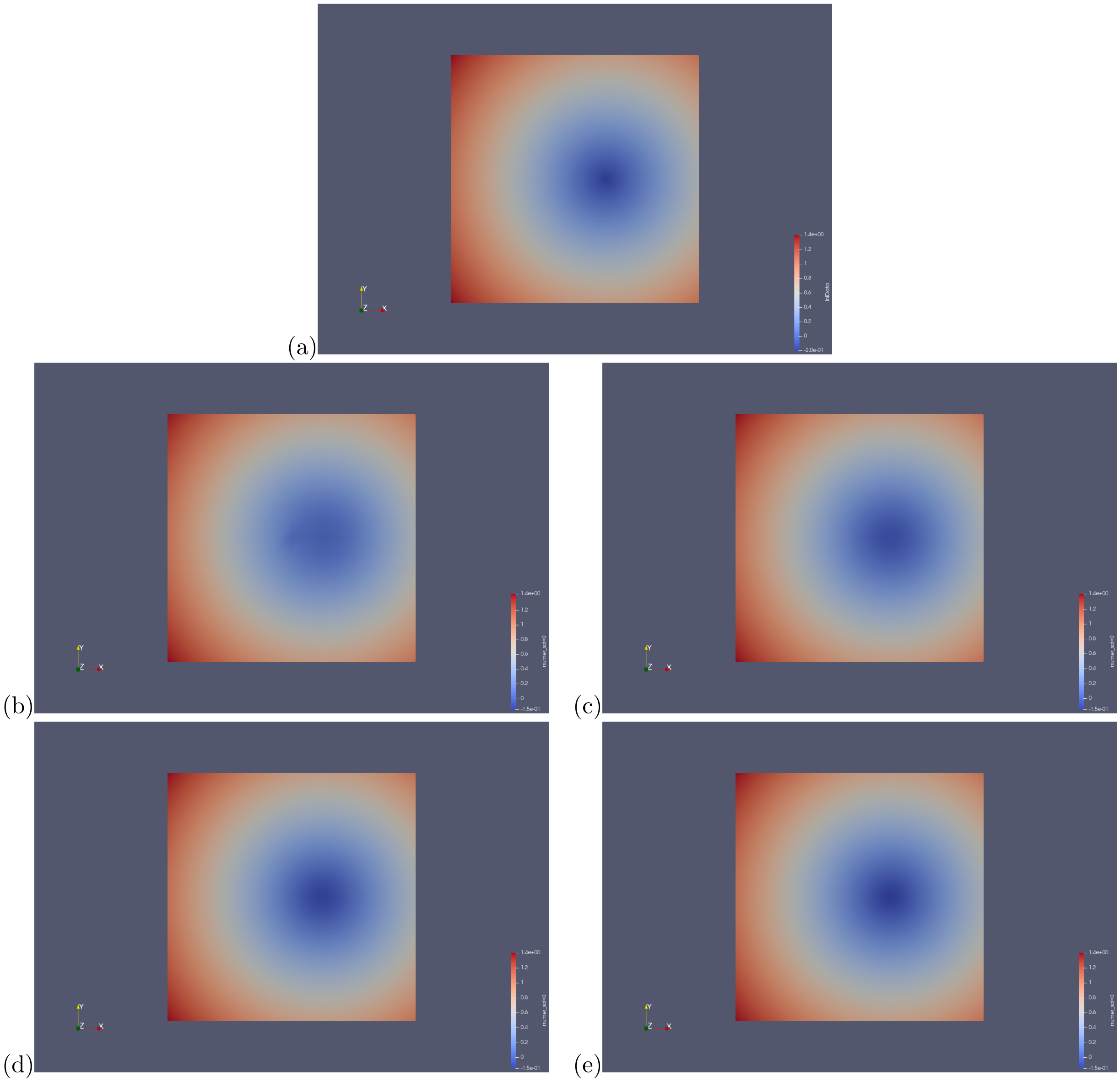

Figure 1: Comparison of Analytical Solution (a) and Numerical Solutions with N=17, 33, 65, and 129 correspond to (b)-(e).

From Figure 2: error maps of numerical solutions with various spatial sizes, we can observe that if we choose smaller Δx, the result would be better. Note that CFL $= \dfrac{\Delta x}{\Delta t} = 0.5$. Table 1 shows the result $||e||_\infty$ for different grid sizes and various $\dfrac{\Delta x}{\Delta t}$ ratios.

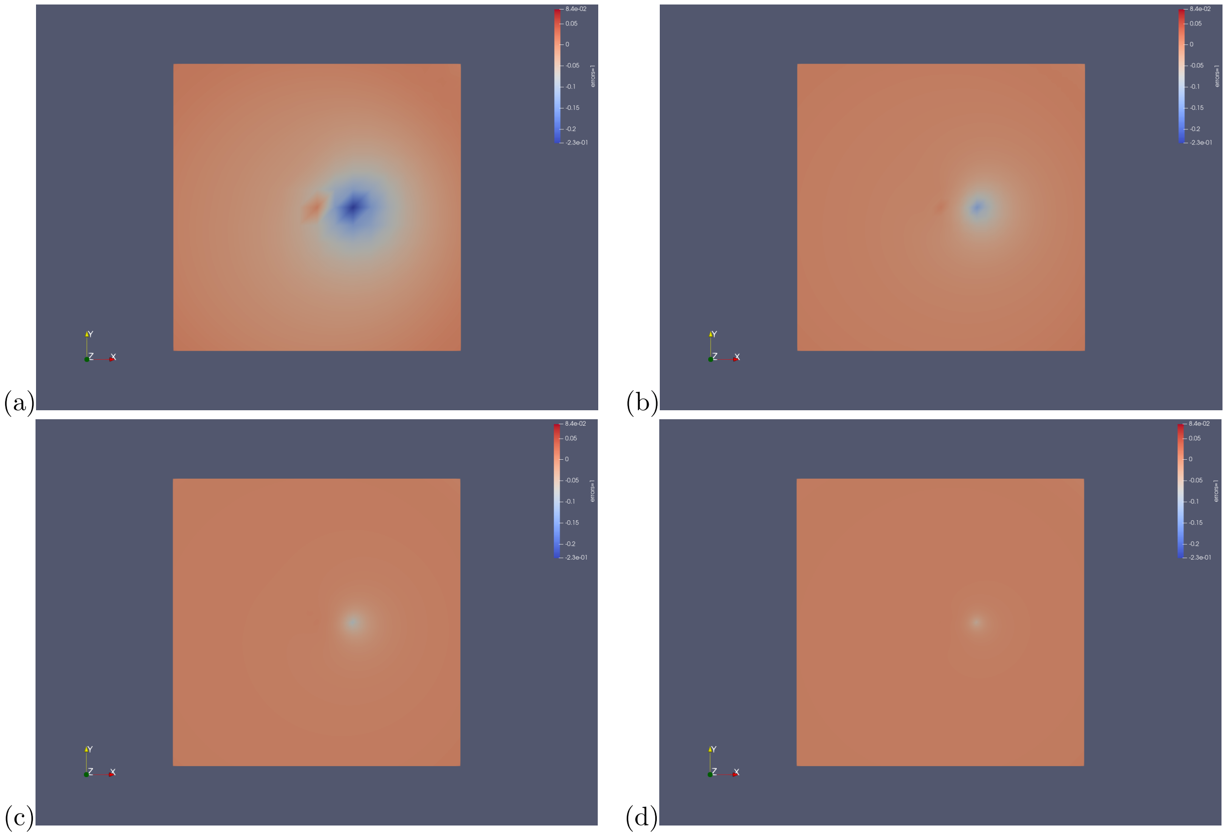

Figure 2: Error Maps of Numerical Solutions with N=17, 33, 65, and 129 correspond to (a)-(d).

$\|\|LTE\|\|_\infty$ vs Δx and Δt					
Δx	$\Delta t = 2\Delta x$	$\Delta t = \Delta x$	$\Delta t = \Delta x/5$	$\Delta t = \Delta x/10$	$\Delta t = \Delta x^2$
1/8	0.1978	0.2328	0.2622	0.2661	0.2651
1/16	0.1215	0.1388	0.1528	0.1546	0.1553
1/32	0.0778	0.0870	0.0940	0.0955	0.1078
1/64	0.0496	0.0551	0.0688	0.0841	0.0986

Table 1: Convergence of SL method with various $\frac{\Delta x}{\Delta t}$ ratios and last column $\Delta t = \Delta x^2$. Note that first column is grid size Δx.

The SL methods include second-order interpolation and RK2, so the local truncation error should be

$$e = O(\frac{\Delta x^2}{\Delta t}) + O(\Delta t^2).$$

The regression of the errors with respect to $\frac{\Delta x^2}{\Delta t}$ and Δt^2 is $e = .0667 + .1787 * \frac{\Delta x^2}{\Delta t} + 2.3991 * \Delta t^2$. When we analyze the convergence rate of Δt by plotting errors vs Δt (Figure 3), we can find a critical point (local minimum), i.e., $\Delta t_{critical} = 0.0835, 0.0526, 0.0331, 0.0209$ correspond to $\Delta x = 1/8, 1/16, 1/32$, and $1/64$ respectively. Thus, when Δt gets smaller, the error explodes instead of decreases with a fixed Δx.

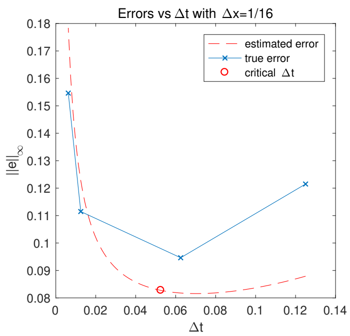

Figure 3: $||e||_\infty$ vs Δt with various Δx. The Dashed lines are regression result and the circled points are critical points. The star points are true errors from the numerical solutions.

To study the convergence rate for grid size, I fixed the time step size $\Delta t = 1/16$. The log-log fit slope of infinity norm of the errors vs Δx is close to 2 (around 1.8). In order to verify $O(\Delta t^2)$, we can set $\Delta t = \Delta x^2$, and the log-log fit is also close to 2.

(b) What can you say about the stability of the method?

The Semi-Lagrangian scheme is unconditionally stable, so any choice of Δt would guarantee a stable solution. As our final time is 2π, we always have a small remainder. If we clip the remainder, it may lower the convergence rate for time. To keep accuracy, we should set `dt = min( dt, t_final - t_n)` to guarantee its second-order convergence rate for time.

(c) Does the solution remain reinitialized?

No, from Figure 4, we can observe that the gradient of the solution is larger than 1 at $t = 2\pi$.

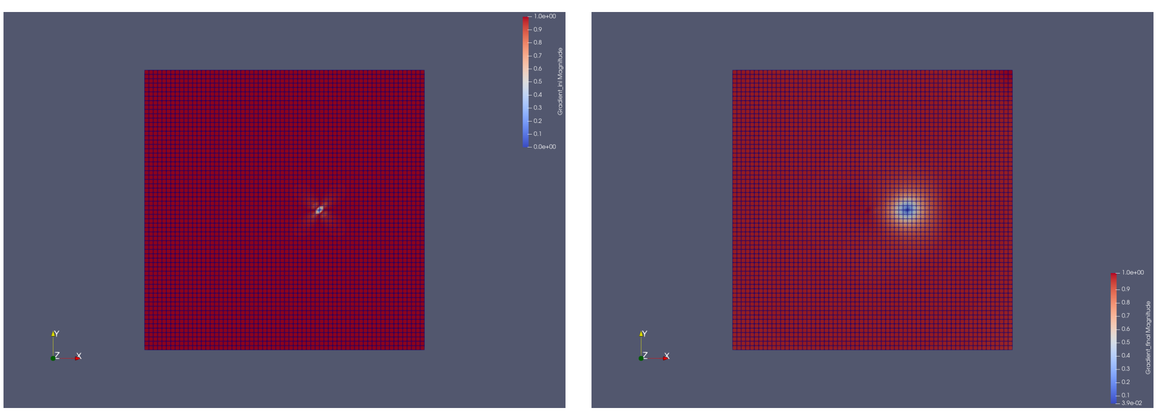

Figure 4: Left is the gradient of the initial. Right is the gradient of numerical solution.

2. *Reinitialization Equation.*

(a) To test your method, consider a contour for which you know the reinitialized level set function (like the initial solution of the above problem) and construct a non reinitialized level set function by perturbing the reinitialized one (make sure that the contour is preserved). Use your code to reinitialize that level set function.

I choose the initial contour in question 1 to multiply it by 0.5 and perturb it to reinitialize. If it works, the gradient of the contour will converge to 1 to obtain a "good" reinitialized function. Results are shown in part (b) Figure 5 and Figure 6.

(b) Show the solution after various number of iterations.

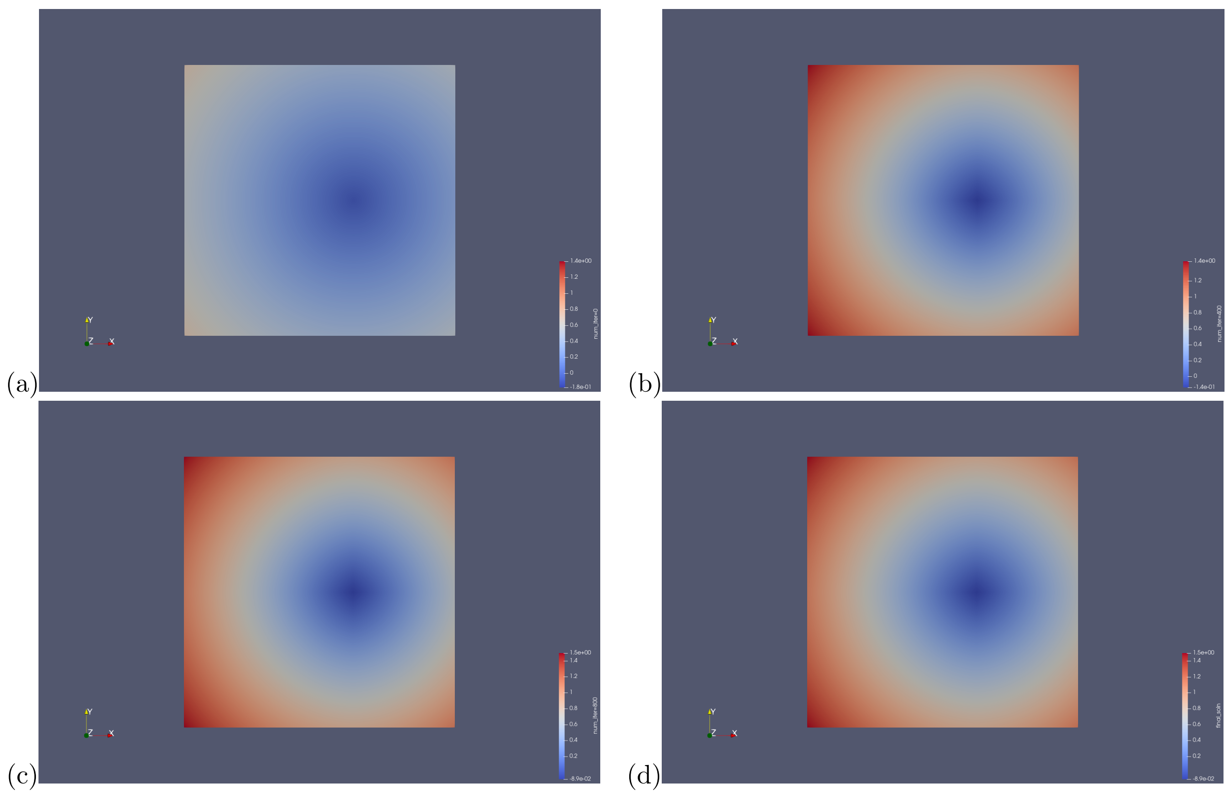

Figure 5: (a)-(d) are numerical solutions for N=129 correspond to t=0, 400 iterations, 800 iterations and final time $t = 2\pi$.

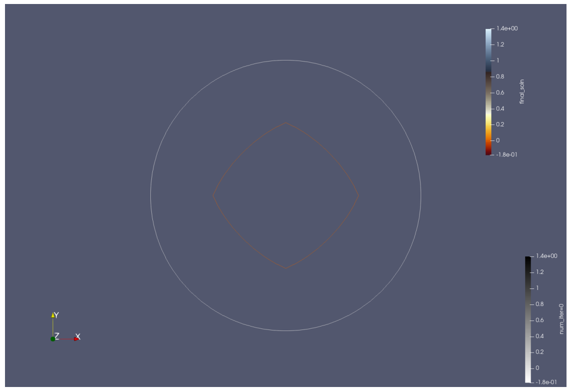

Figure 6: Contours: white outside one is initial and red inside one is the reinitialized solution at the final time.

(c) Approximately, how many does it take to obtained a "good" reinitialized function?

After 400 iterations, we already obtained a "good" reinitialized function since its gradient is close to 1.

3. *Level set method* Use the method you developed in part 1 and 2 to simulate the advection of a circular contour of radius 0.2 and center $(0.25, 0)$ under the deforming velocity field defined previously. What should you look at to verify that your method is working correctly?

I combined part 1 and 2 for this question. In other words, I reinitialized the advected solution at each time step. Figure 7 shows the numerical solution after various number of iterations and Figure 8 shows the gradient of initial and final solutions. Compared with 1(c) in Figure 4, the result in Figure 8 remains reinitialized at the final time, since $||\nabla\phi||$ is around 1 at final time $t = 2\pi$. This result implies my method is working correctly.

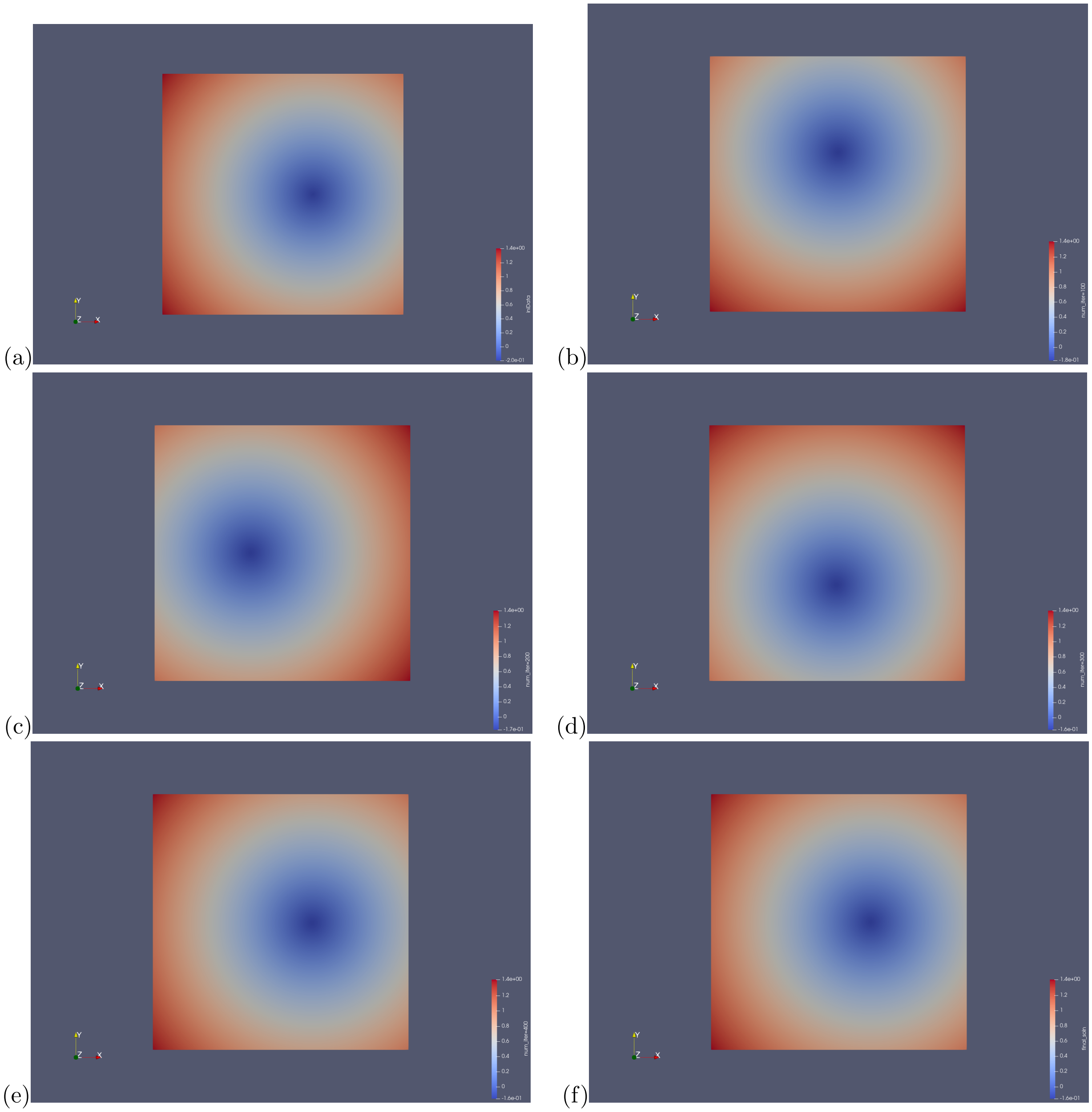

Figure 7: (a)-(f) are numerical solutions for N=129 correspond to t=0, after 100 iterations, 200 iterations, 300 iterations, 400 iterations and final time $t = 2\pi$.

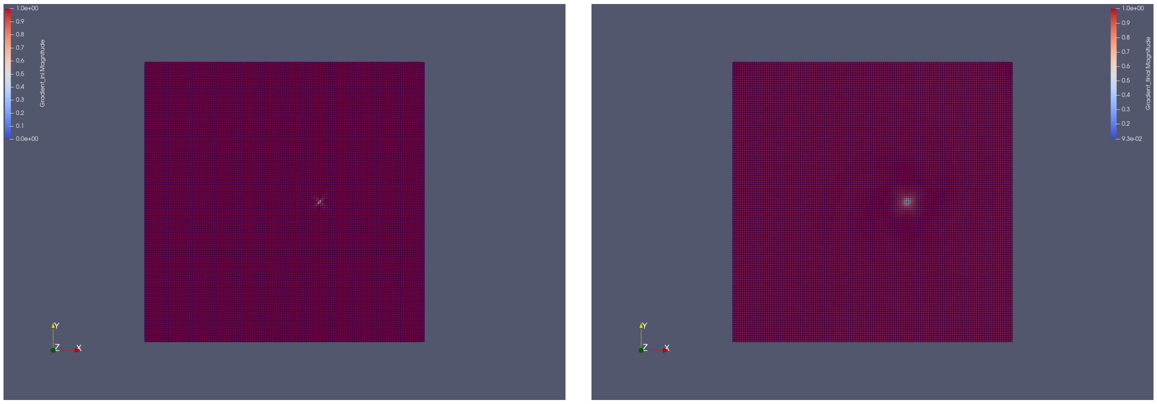

Figure 8: Left is the gradient of the initial. Right is the gradient of numerical solution.