ME 605

Computational Fluid Dynamics

Project – 3
Solution to 1D Unsteady State Convection Diffusion Equation using Finite Differences Method

Souritra Garai

18110166

Indian Institute of Technology Gandhinagar

Semester I, 2020-21

Problem Statement

We are given 1D unsteady state convection diffusion equation over a domain $x \in [0, L]$ –

$$\frac{\partial}{\partial t}(\rho\varphi) + \frac{\partial}{\partial x}(\rho u\varphi) = \frac{\partial}{\partial x}\left(\Gamma\frac{\partial\varphi}{\partial x}\right)$$

This is a parabolic PDE. Information φ travels in the forward direction in t and in both positive and negative directions in x.

We are given the boundary conditions –

1. $\varphi(x = 0, t) = 0$
2. $\varphi(x = 1, t) = 1$

We are given the initial condition –

1. $\varphi(x, t = 0) = 0,\ x \neq 1$

Given values of $\rho = 1$, $L = 1$, and $u = 1$. We note velocity is along positive x direction.

The terms in the given PDE are –

1. Transient term – $\frac{\partial}{\partial t}(\rho\varphi)$
2. Advection term – $\frac{\partial}{\partial x}(\rho u\varphi)$
3. Diffusion term - $\frac{\partial}{\partial x}\left(\Gamma\frac{\partial\varphi}{\partial x}\right)$

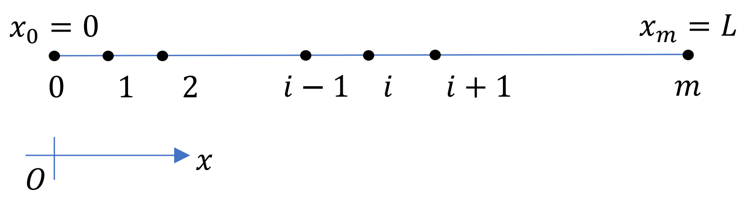
We need to solve the PDE using three schemes –

1. Euler's Explicit Scheme (Implemented 4th order Runge-Kutta method to improve accuracy)
2. Euler's Implicit Scheme
3. Crank Nicolson Method

Additionally, we will study effects of cell Peclet number and size of times steps on the solution obtained from the three methods.

Generating the Grid

We have assumed $m + 1$ equidistant points on the domain $[0, L]$, including the endpoints. The distance between consecutive points is $\Delta x = L/m$. The position of point i is given by $x_i = i \cdot \Delta x$, where $i \in \{0, 1, 2, \dots, m\}$



In the time dimension, we move in time steps of Δt and the number of time steps evolved is denoted by a superscript n.

We define the following dimensionless quantities, that will prove useful later –

1. Cell Peclet number, $Pe_{cell} = \frac{\rho u \Delta x}{\Gamma}$
2. Courant number, $c = \frac{u \Delta t}{\Delta x}$
3. $d = \frac{\Gamma \Delta t}{\rho (\Delta x)^2}$

Discretization of Spatial Derivative terms

We will use the following discretization schemes for the advection and diffusion terms –

1. Advection term

   We earlier noted velocity is along positive x. We have used third order upwind difference scheme to discretize the advection term, using two grid points to the upwind side and one to the downwind side –

$$\rho u \frac{\partial \varphi}{\partial x} \Big|_{x=x_i,\ t=t_n} \approx \rho u \varphi'_{i,n} = \rho u \frac{2\varphi_{i+1}^n + 3\varphi_i^n - 6\varphi_{i-1}^n + \varphi_{i-2}^n}{6\Delta x}$$

2. Diffusion term

   We use central difference scheme to discretize the second order derivative in x –

$$\Gamma \frac{\partial^2 \varphi}{\partial x^2} \Big|_{x=x_i,\ t=t_n} \approx \Gamma \varphi''_{i,n} = \Gamma \frac{-\varphi_{i+2}^n + 16\varphi_{i+1}^n - 30\varphi_i^n + 16\varphi_{i-1}^n - \varphi_{i-2}^n}{12(\Delta x)^2}$$

For the points neighbouring the boundary points, we will resort to using lower order schemes as number of available points are limited.

Discretization of Time Derivatives

We may choose either forward difference or backward difference of both to discretize the transient term. Depending on the difference scheme we end up with explicit or implicit or a hybrid of both methods to calculate changes in φ while marching forward in time.

Given PDE –

$$\frac{\partial \varphi}{\partial t}\Big|_{x=x_i,\ t=t_n} + \rho u \frac{\partial \varphi}{\partial x}\Big|_{x=x_i,\ t=t_n} = \Gamma \frac{\partial^2 \varphi}{\partial x^2}\Big|_{x=x_i,\ t=t_n}$$

Using forward difference in time –

$$\rho \frac{\varphi_i^{n+1} - \varphi_i^n}{\Delta t} + \rho u \varphi'_{i,n} = \Gamma \varphi''_{i,n}$$

$$\Rightarrow \varphi_i^{n+1} = \varphi_i^n + d \cdot \varphi''_{i,n} - c \cdot \varphi'_{i,n}$$

We see all the terms to the right-hand side of the discretized equation is known while marching forward in time. This is Euler's Explicit method, where we can directly find φ^{n+1} for each time step. Additionally, we observe –

$$\varphi_i^{n+1} = \varphi_i^n + \Delta t \cdot f(\varphi_{i+2}^n, \varphi_{i+1}^n, \varphi_i^n, \varphi_{i-1}^n, \varphi_{i-2}^n)$$

or more compactly,

$$\varphi^{n+1} = \varphi^n + \Delta t \cdot f(\varphi^n)$$

To improve accuracy in each time step, we may employ 4th order Runge-Kutta method. Then the scheme takes the form –

$$\varphi^{n+1} = \varphi^n + \frac{\Delta t}{6} \cdot \left[f(\varphi^n) + 2f\left(\varphi^{*n+\frac{1}{2}}\right) + 2f\left(\varphi^{**n+\frac{1}{2}}\right) + f\left(\varphi^{*n+1}\right) \right]$$

where,

$$\varphi^{*n+\frac{1}{2}} = \varphi^n + \frac{\Delta t}{2} \cdot f(\varphi^n)$$

$$\varphi^{**n+\frac{1}{2}} = \varphi^n + \frac{\Delta t}{2} \cdot f\left(\varphi^{*n+\frac{1}{2}}\right)$$

$$\varphi^{*n+1} = \varphi^n + \Delta t \cdot f\left(\varphi^{**n+\frac{1}{2}}\right)$$

This scheme helps improve the global error due to discretization of transient term, from $O(\Delta t)$ in Euler's scheme to $O((\Delta t)^4)$, where the time step Δt is restricted by the inequalities from von Neumann stability analysis.

Using backward difference –

$$\rho \frac{\varphi_i^{n+1} - \varphi_i^n}{\Delta t} + \rho u \varphi'_{i,n+1} = \Gamma \varphi''_{i,n+1}$$

$$\Rightarrow \varphi_i^{n+1} - d \cdot \varphi''_{i,n+1} + c \cdot \varphi'_{i,n+1} = \varphi_i^n$$

While marching forward in time from an initial condition, we can directly determine the right-hand side. But the left-hand side forms a linear combination of $\{\varphi_{i+2}^{n+1}, \varphi_{i+1}^{n+1}, \varphi_i^{n+1}, \varphi_{i-1}^{n+1}, \varphi_{i-2}^{n+1}\}$, all of which are unknown at the current time step. We will need to solve this linear equation at each time step to find out φ^{n+1}. However, the system of linear equations forms a banded matrix that can be solved easily using Thomas Algorithm. This is known as Euler's Implicit scheme.

Using trapezoidal rule –

$$\rho \frac{\varphi_i^{n+1} - \varphi_i^n}{\Delta t} = \frac{1}{2}\left\{\left(\Gamma \varphi''_{i,n+1} - \rho u \varphi'_{i,n+1}\right) + \left(\Gamma \varphi''_{i,n} - \rho u \varphi'_{i,n}\right)\right\}$$

$$\Rightarrow \varphi_i^{n+1} - d \cdot \varphi''_{i,n+1} + c \cdot \varphi'_{i,n+1} = \varphi_i^n + d \cdot \varphi''_{i,n} - c \cdot \varphi'_{i,n}$$

Again, in this case, the right-hand side can be determined in a forward time marching problem, while the left-hand side will require us to solve a banded matrix equation. This is known as Crank-Nicolson Method.

The coefficients of $\{\varphi_{i+2}^n, \varphi_{i+1}^n, \varphi_i^n, \varphi_{i-1}^n, \varphi_{i-2}^n\}$ and $\{\varphi_{i+2}^{n+1}, \varphi_{i+1}^{n+1}, \varphi_i^{n+1}, \varphi_{i-1}^{n+1}, \varphi_{i-2}^{n+1}\}$ for each of the methods specified above, are determined from the script Coefficients.py.

Solutions

The following pages shows the solution obtained via different methods and at different cell Peclet numbers. They have been compared against a solution obtained using very small Δx and Δt, plotted in corresponding graphs with a black line. The following figures show the temporal evolution of distribution of φ –

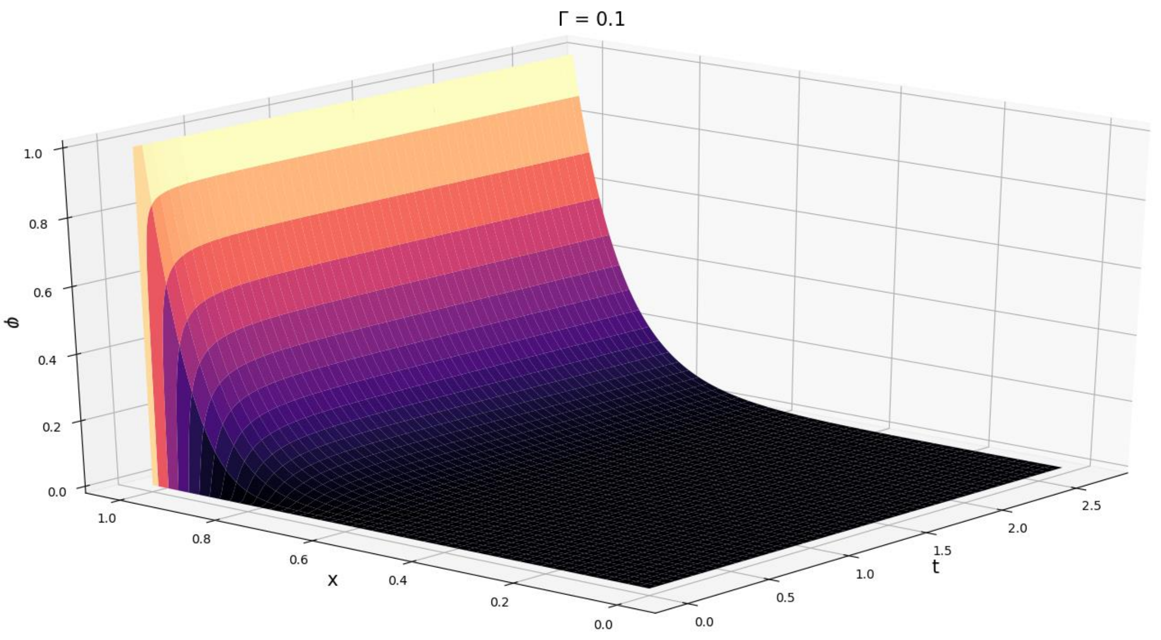


Figure 1: Convection dominated transport with positive Velocity

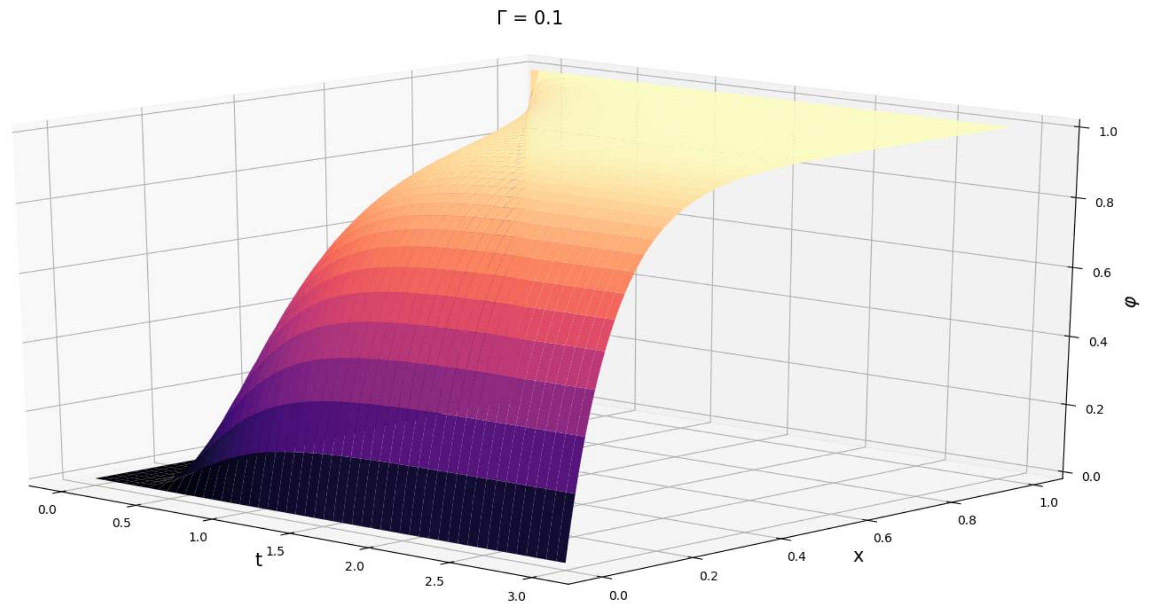


Figure 2: Convection dominated transport with negative Velocity

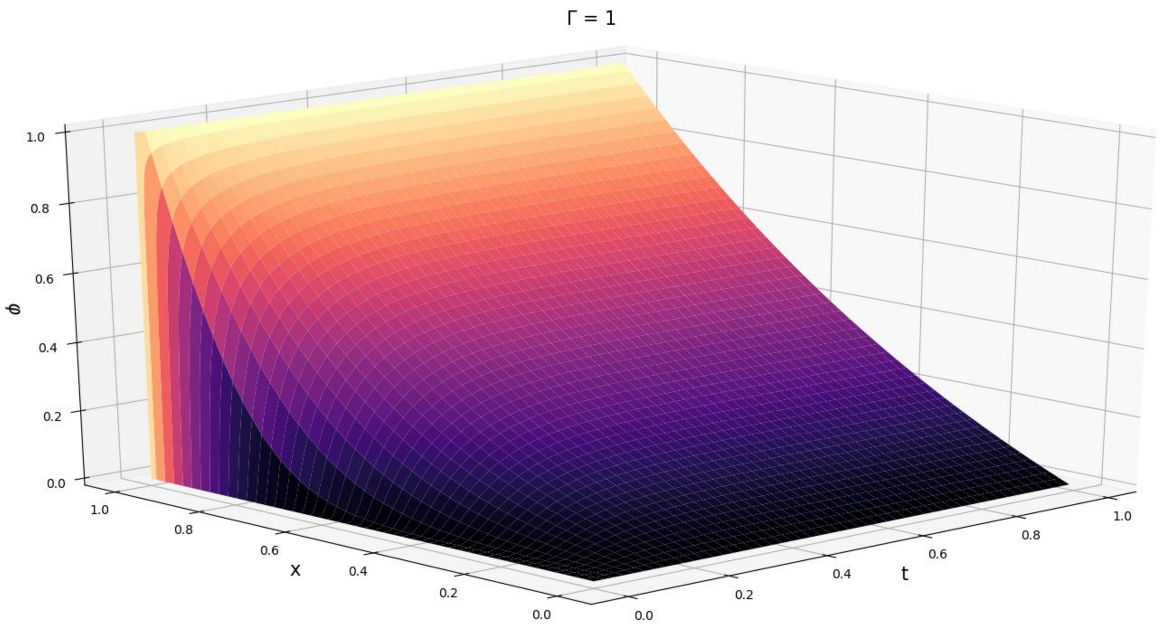


Figure 3: Moderate Diffusion

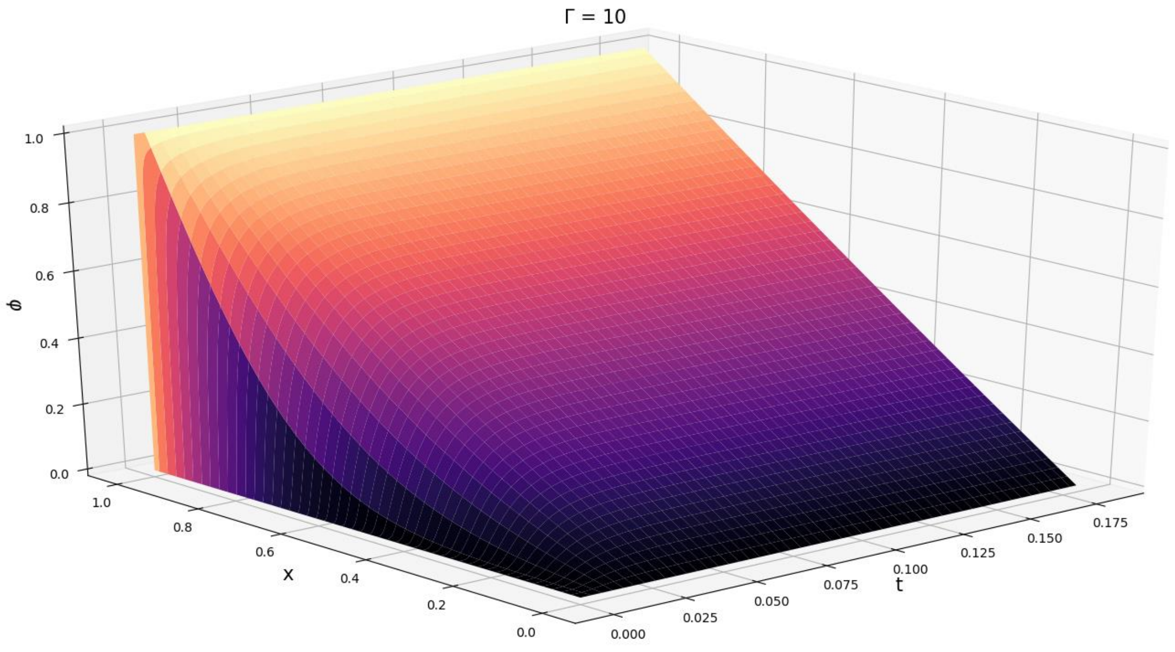


Figure 4: Strong Diffusion

The following figures show the difference in exact solution and those solutions obtained with $\Delta t = 0.01$ and $m = 20$, i.e., $\Delta x = 0.05$, using numerous different solutions, along the time axis. Kindly switch the view to landscape mode, since the following graphs were difficult to fit in portrait mode.

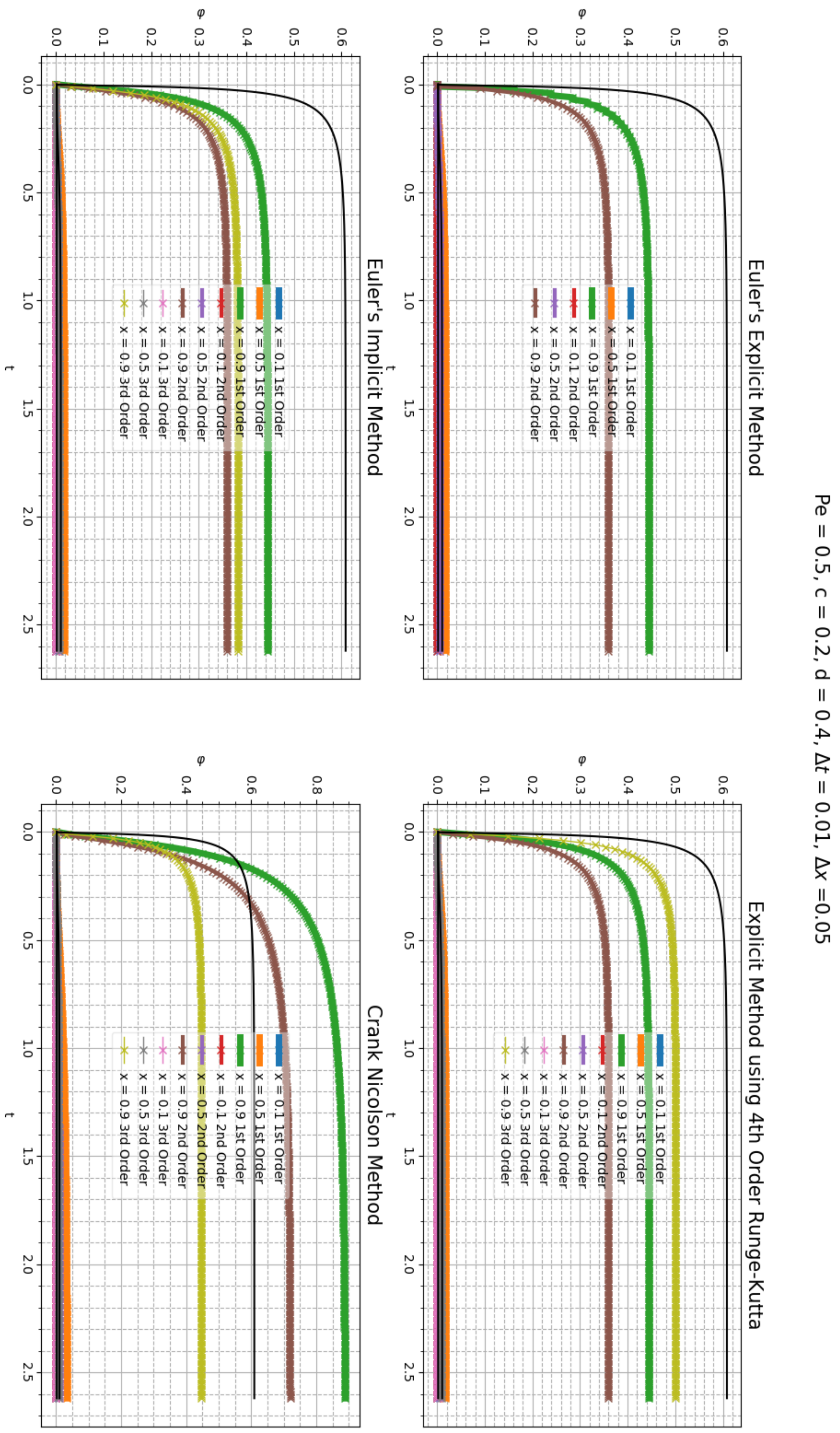


Figure 5: Convection dominated transport with positive Velocity

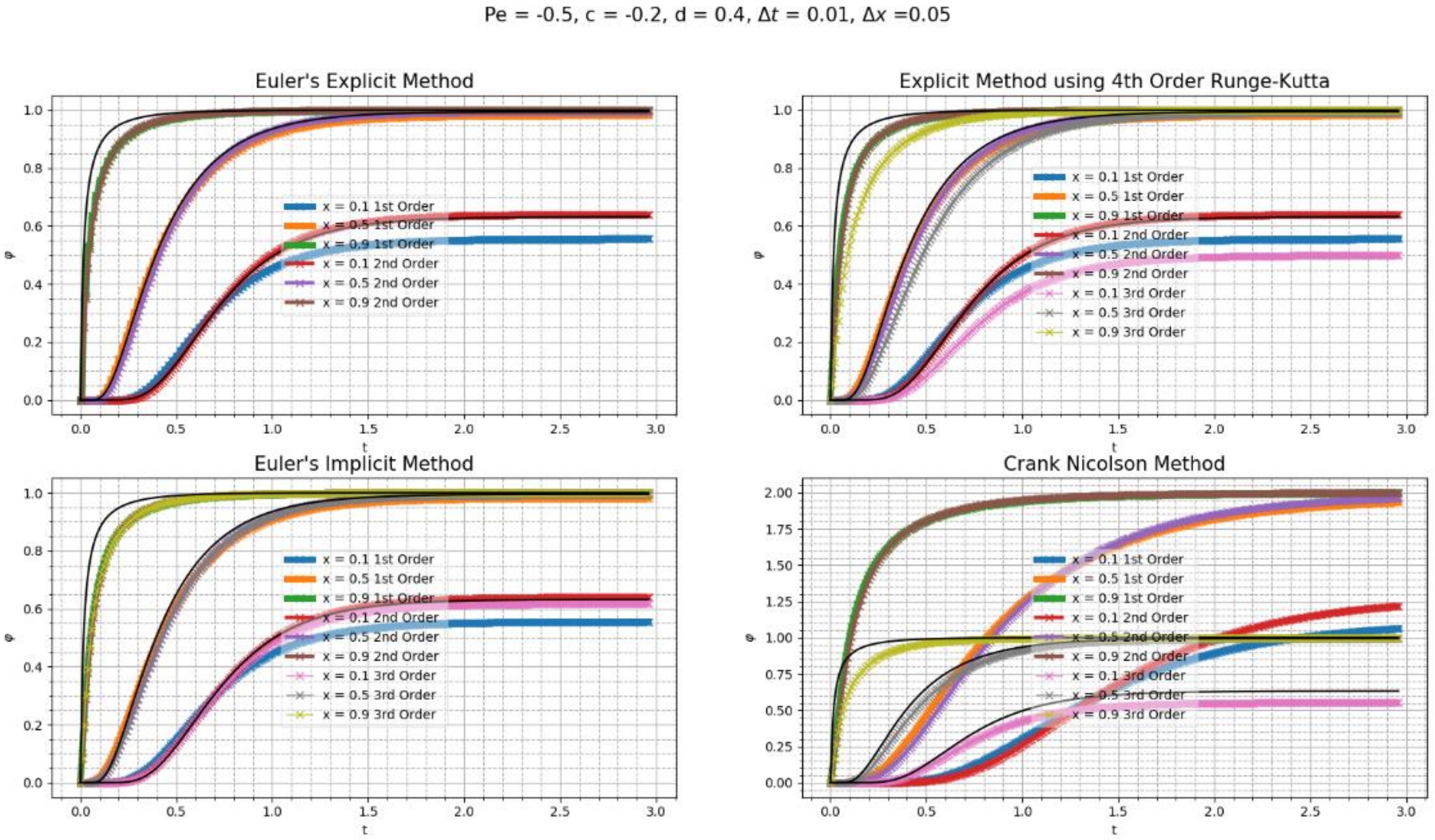


Figure 6: Convection dominated transport with negative Velocity

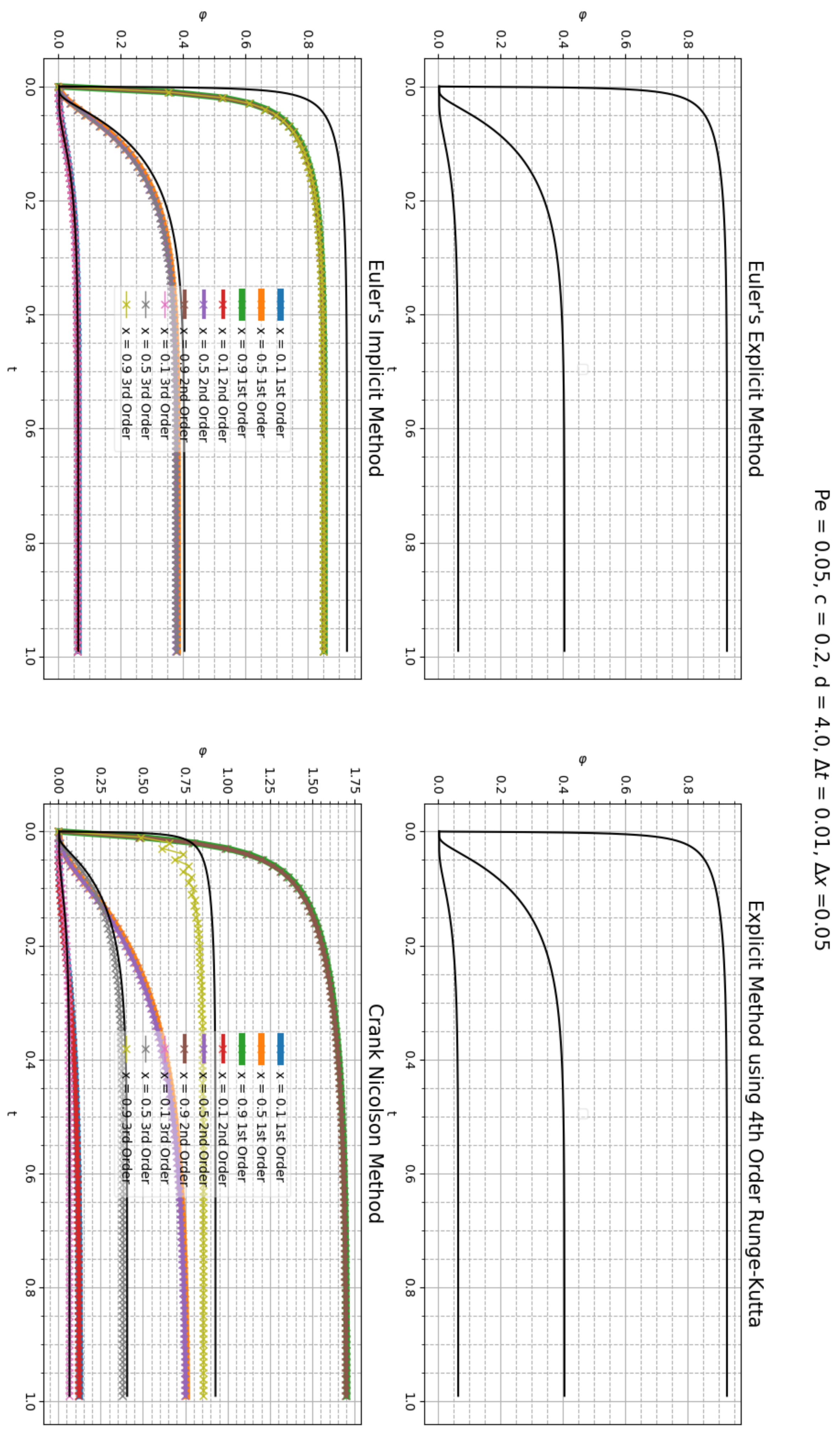


Figure 7: Moderate Diffusion

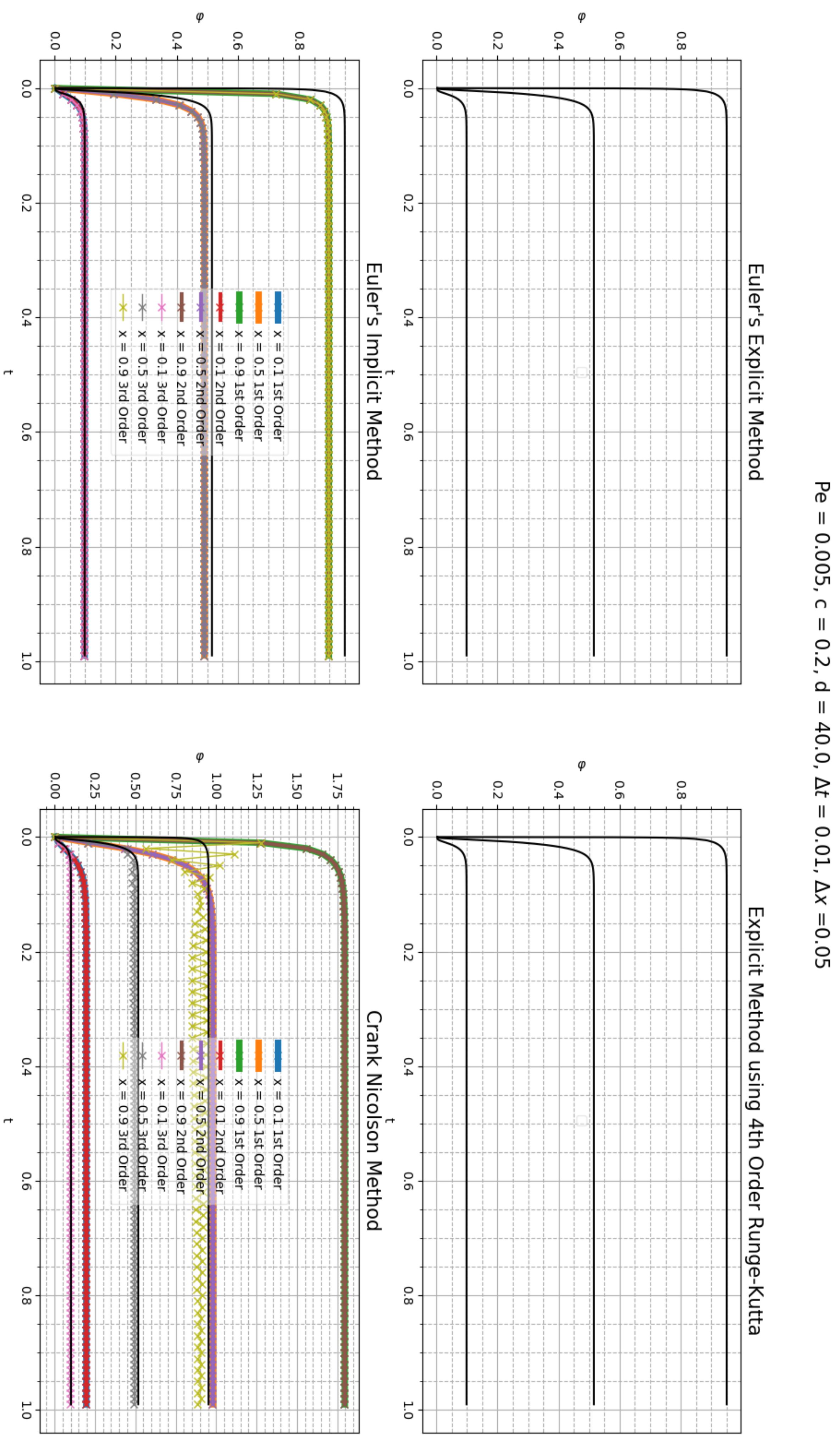


Figure 8: Strong Diffusion

In these plots, the order in legend denotes the lowest order of truncation error in the spatial derivatives.

In Figure 5, all methods seem stable except higher order explicit schemes for convection dominated transport. The higher order scheme probably induces stricter rules for stability in explicit schemes, than the general condition $d \leq 0.5$ obtained from von Neumann analysis. In fact, the higher order explicit scheme is unstable for all the examples, since Peclet number is decreasing, Γ is increasing and hence, d is increasing. The application of Runge-Kutta method for explicit scheme has resulted in a stable solution even for higher order scheme. This may be due to the predictor-corrector step in the Runge-Kutta algorithm. All the methods have underpredicted the value of φ, except lower order Crank Nicolson methods.

Figure 6 shows a convection dominated transport in the opposite direction. Almost all methods approach the exact solution except the lower order Crank Nicolson methods. Another peculiarity is that lower order Runge-Kutta solutions are closer to the exact solution than the higher order one.

In Figure 7 and Figure 8, where diffusion is dominant, we see explicit schemes fail to stabilize due to decreasing Pe and d overshooting 0.5. The higher order Crank Nicolson method starts oscillating but the mean value follows the trend of the exact solution closely. The lower order Crank Nicolson solutions have consistently overpredicted the value of φ. All the three implicit schemes, even with different order of truncation error in spatial discretization, have similar and one of the best accuracies.

Accuracy with Time Step

In Figure 9, we plotted the mean square error for solution of $\varphi(x, t)$ for all methods. The error was again calculated with respect to a solution obtained using $\Delta x = 0.001$ and $\Delta t = 0.0001$.

We note that after a certain Δt, even change in order of Δt results in barely noticeable reduction in error. Strangely, some schemes with lower order discretization in space, produce more accurate results than other higher order ones.

Most of the curves for explicit schemes do not extend beyond a Δt toward increasing Δt. This is because these schemes become unstable after beyond a threshold of Δt, that can be shown using von Neumann analysis. Interestingly, for cases when the method is indeed stable, the second order explicit schemes provide one of the best accuracies among all the methods.

Conclusion

We observed not all methods converge or give a stable solution for all scenarios. But some them may provide better accuracy compared to methods that are unconditionally stable.

Codes –

1. Coefficients.py holds supplementary data and functions for the solvers.
2. TDMA_solver.py and PDMA_solver.py are implementations of TDMA and PDMA algorithms respectively.
3. Scripts Implicit_Method.py, Explicit_Method.py, and Crank_Nicolson_Method.py respectively contain functions for implementation of respective algorithms. They also contain provision at the end for solving examples.
4. Exact_Solution.py solves and saves a highly accurate solution (requires a few minutes to generate)
5. Scripts Comparing_Methods_with_Peclet_Number.py and Comparing_Methods_with_Time_Step.py generates the graphs presented in the report.

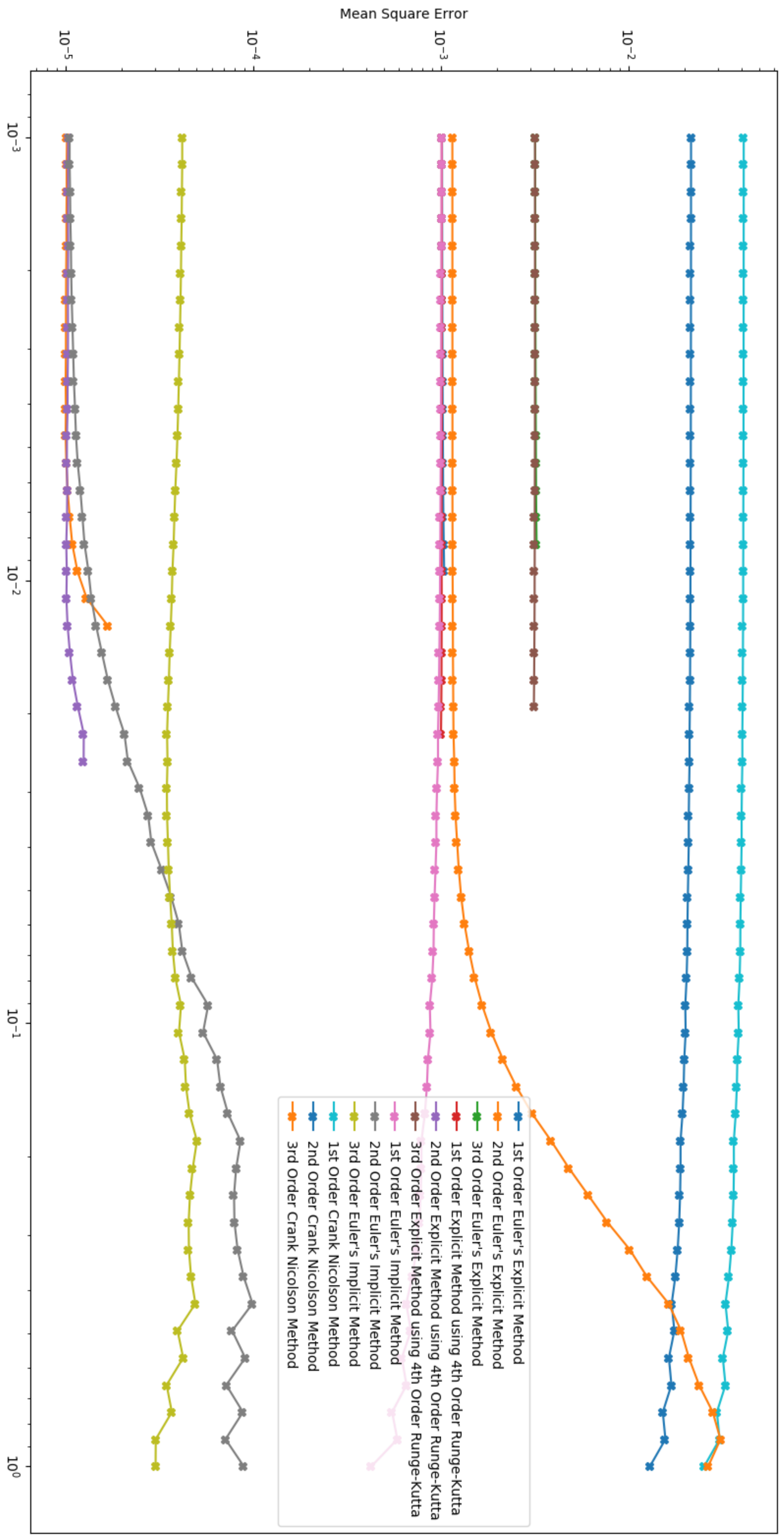


Figure 9: Graph of Order of Mean Square Error vs Order of Time Step for various methods