
INDIAN INSTITUTE OF SCIENCE

END-SEMESTER PROJECT REPORT

---

# Analysis of Synchronous and Asynchronous schemes for partial differential equation

---

*Author:*
Suriyah R K
3rd year UG
suriyahr@iisc.ac.in

*Supervisor:*
Manish Jain
Physics Department

April, 2023

# Abstract

In this report, we will begin by using the finite differences method to solve partial differential equations. The main PDEs we will be looking at are the diffusion equation, Advection - Diffusion equation, and Wave equation. And progressively solving them in higher dimensions and animating them. Then continue by analyzing the time taken to perform these processes and the inefficiency in serial processing when faced with a highly complex system, followed by introducing synchronous and Asynchronous schemes. Analyzing the error profiles associated with these methods and finally suggesting some results and possible improvements to the schemes suggested. This is a basic method that is discussed in this report; other highly efficient schemes are available which are being actually used in simulating these equations in real applications

# Contents

# 1 Introduction

## 1.1 Finite Difference Method

Most of nature's laws and engineering models are dictated by partial differential equations; they are integral in determining how systems evolve. It is not always straightforward how to solve a PDE; a closed-form solution usually doesn't exist. Thus developing numerical models to handle such equations is a crucial part of science and technology. The primary and straightforward method to solve partial differential equations is through **Finite Difference Method**, where we approximate the derivative by " Finite differences." This method's accuracy depends on how many orders we go down the Taylor series of a derivative; at the same time, the accuracy gets better the smaller the interval we choose to use. ( This is not always true, as sometimes we need to obey stability criteria which ensures our solutions don't blow up). For the first-order approximation, there are three choices of finite difference we can employ,

- Forward Difference
- Backward difference
- Central difference

Depending upon the situation, these methods are used hand in hand.

$$\text{Forward - Difference} \implies \frac{dx}{dt} = \frac{x_{i+} - x_i}{\Delta t} \tag{1.1}$$

Similarly, the backward difference can be defined by replacing $i$ with $i - 1$. The central difference is defined as,

$$\text{Centered - Difference} \implies \frac{dx}{dt} = \frac{x_{i+1} - x_i}{2\Delta t} \tag{1.2}$$

we will mainly only be using forward and Central differences.

## 1.2 Heat equations

The heat equation governs the flow of heat in materials. The derivation of heat is observation-based, but proper motivation can be given to the terms involved. There are two terms in the Heat equation **Diffusion** and **Advection**. The diffusion term represents the process of heat transfer by thermal conductivity. It describes how heat diffuses from regions of high temperature to regions of low temperature until the temperature throughout the material reaches equilibrium. The diffusion term in the heat equation involves the second derivative of temperature with respect to position and is proportional to the material's thermal conductivity. This term is responsible for smoothing out temperature gradients over usually a long period of time.

The other is the Advection term which describes how heat is transported by the movement of the material itself, such as in the case of fluid flow. This term is proportional to the velocity of the material itself, for example, fluid flow. This involves the first derivative of temperature with respect to position. The Advection term can cause abrupt changes in temperature at boundaries or interfaces between materials and is responsible for heat transport over large distances in the material.

Both diffusion and Advection terms are present in the heat equation, and together they determine the evolution of temperature over time and space in a given region.

The general Heat equation is therefore written as

$$\frac{\partial u}{\partial t} = \vec{c}.\vec{\nabla}u + \kappa\nabla^2 u \tag{1.3}$$

Therefore one - dimensional version of the heat equation can be written as (we will be using the 1 - D version for analysis, but the results obtained can be easily generalized to higher dimensions)

$$\frac{\partial u}{\partial t} = c\frac{\partial u}{\partial x} + \kappa\frac{\partial^2 u}{\partial t^2} \tag{1.4}$$

Which, when converted into the language of finite difference, is,

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = c\frac{u_{i+1}^n - u_{i-1}^n}{2\Delta x} + \kappa\frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{(\Delta x)^2} + O(\Delta t, \Delta x^2) \tag{1.5}$$

Where $n$ represent the discrete points in time, and $i$ represent the position of a grid point.

## 1.3 Wave Equation

Like the Heat equation, the Wave equation is a simple partial differential equation that can be profoundly found in most areas of physics. It can be concisely written as,

$$\frac{\partial^2 u}{\partial t^2} = c\nabla^2 u \tag{1.6}$$

Considering the 1D variant and applying finite difference, we have,

$$\frac{u_i^{n+1} - 2u_i^n + u_i^{n-1}}{(\Delta t)^2} = c\frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{(\Delta x)^2} + O(\Delta t^2, \Delta x^2) \tag{1.7}$$

## 1.4 Animations

To get a better visual appreciation for the evolution of heat across the material, I have animated the evolution using **matlplotlib.animation** package in Python. Several cases ranging from 1D, 2D, and 3D for the advection-diffusion and wave equation have been considered. Every plot can be found in the codes attached. The results have also been attached at the end of the report. Animation can be found in the results folder in Codes. Some animations which are on 2D have been saved, but the three animation can be seen only by running the code as the 3D plot is interactive in nature.

# 2 Synchronous and Asynchronous Schemes to solve PDE

## 2.1 Synchronous Scheme

The analysis we will be doing is for the heat equation. The method that was suggested in the previous chapters is what is called a synchronous scheme to solve a PDE. Where to determine the heat value at the time step uin+1, we require that the heat at neighboring points at time step $n$ can be seen from equation 1.5. We can quickly implement this procedure using a serial code. With information about $u_i^n$ at all grid points at a certain point in time, we can determine the heat at the next time step at all points (Assuming periodic boundary condition ).

But now consider the case where the grid points are broken into several Processing elements (PEs) where each PE element can perform computation parallelly without interference from the nearby PEs. The parallel processing will highly increase the rate at which our PDE is being solved. The computation at the interior points of a PE is trivial. However, updating the values of grid points near the boundary of the PE will require communication between the PEs, which will, in general, increase the total time required to perform the computation. In the Synchronous Scheme, the computation to the next step will not progress until all the points in the PE have received information and have computed to the next point in time. In large-scale parallel processing systems, these PEs are usually separated. The time taken to communicate between the PEs is usually much higher than the computation time. Thus this plays a vital role in how efficiently a code can be implemented for large-scale use. In the Global Synchronous scheme discussed above, there is a lot of **idle** time for the processors where they are not performing any computation.

This idling time can be significantly reduced by choosing different schemes, which we will discuss now.

## 2.2 Asynchronous Scheme

The problem of idling time can be significantly reduced by allowing the PEs to continue, irrespective of the current state of the message being received. It is unnecessary for us to always have the most updated value at the boundary point. Therefore the modified finite difference equation can be written as (considering the heat equation with only diffusion term)

$$u_i^{n+1} = u_i^n + \frac{\kappa \Delta t}{\Delta x^2}(u_i^{n+1} - 2u_i^n + u_{i-1}^{\tilde{n}}) \tag{2.1}$$

For a right boundary point and

$$u_i^{n+1} = u_i^n + \frac{\kappa \Delta t}{\Delta x^2}(u_i^{\tilde{n}+1} - 2u_i^n + u_{i-1}^n) \tag{2.2}$$

For a left boundary point. Here $\tilde{n}$ denotes the delay in communication between the PEs. There it can take the values say $(n-1)$, $(n-2)$, and so on. And moreover, the choice of $\tilde{n}$ is, in general, very well approximated by a random process; thus, we will perform the analysis assuming it's a random variable with corresponding probabilities of obtaining the values listed above. This process of different PEs performing computation independently without receiving the latest data is called **an Asynchronous scheme** of solving PDE. An excellent cartoon to explain the difference between both schemes can be seen in figure 2.1 We can see the

Asynchronous scheme is less accurate than its counterpart, but our major question should be how inaccurate it is and whether we can look over it. There are two important parts of numerical evolution which we need to pay close attention to **Accuracy** and **Stability.**

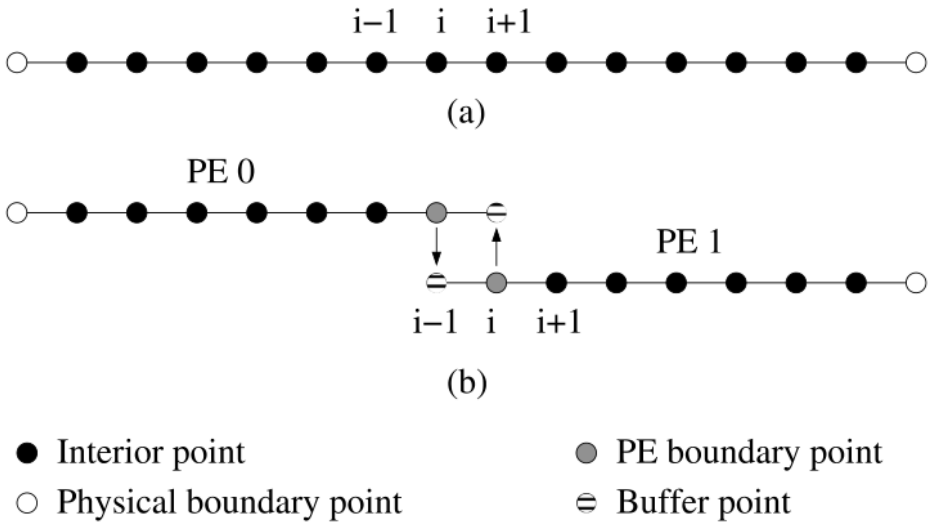



**Figure 2.1:** Discretized 1D domain - (a) Domain in serial code (b) Domain decamped to PEs

## 2.3 Stability

A scheme is said to be stable if the error after each step doesn't blow up during the iteration process, and the values at subsequent time steps converge or increase only infinitesimally. Using Von Neumann analysis, it can be shown the PDE (1.5) ( assume c = 0 and periodic boundary condition ) will be stable in the synchronous scheme for,

$$r_\kappa = \frac{\kappa \Delta t}{\Delta x^2} \leq \frac{1}{2}$$

We will be using the matrix formulation to understand these stabilities. Equation (1.5) can be written as,

$$u_i^{n+1} = r_\kappa u_{i-1}^n + (1 - 2r_\kappa) u_i^n + r_\kappa u_{i+1}^n \tag{2.3}$$

$$V^{n+1} = A^n V^n \tag{2.4}$$

Where $V^n = [u_1^n \; u_2^n \ldots]$ and $A^n$ is

$$A^n = \begin{bmatrix} (1-2r_\kappa) & r_\kappa & 0 & \ldots & r_\kappa \\ r_\kappa & (1-2r_\kappa) & r_\kappa & 0 & \ldots \end{bmatrix}$$

Where $A^n$ is an N × N matrix. Extrapolating the equation to previous moments in time we have,

$$V^{n+1} = A^n A^{n-1} A^{n-2} \ldots A^0 V^0 \tag{2.5}$$

As all $A^i$ are the same. For $V^{n+1}$ to not blow up, we require $||A^n|| \leq 1$, where the norm can be defined in several ways. We will use the $\infty$ norm defined as $||A||_\infty = max_i \Sigma_j |b_{ij}|$. This directly leads to the result that can be obtained via Von Neumann's analysis.

$$|r_\kappa| + |1 - 2r_\kappa| + |r_\kappa| \leq 1 \tag{2.6}$$

Which is satisfied when $0 \leq r_\kappa \leq \frac{1}{2}$

The very same analysis can be done for the asynchronous scheme. We will take $\tilde{n}$ and can take only $n$ or $n-1$ to simplify the analysis, where $k_i$ can take values 0 and 1 depending on whether it is n or n-1. As there are two possible time steps, our time evolution matrix will take the form

$$\tilde{C} = \begin{bmatrix} \tilde{A}_0^n & \tilde{A}_1^n \\ I & 0 \end{bmatrix}$$

Where $\tilde{A}_0^n$ and $\tilde{A}_1^n$ are

$$A^n = \begin{bmatrix} (1-2r_\kappa) & (1-k_2)r_\kappa & 0 & \dots & r_\kappa \\ r_\kappa & (1-2r_\kappa) & (1-k_3)r_\kappa & 0 & \dots \end{bmatrix}$$

$$A^n = \begin{bmatrix} 0 & k_2 r_\kappa & 0 & \dots & r_\kappa \\ 0 & 0 & k_3 r_\kappa & 0 & \dots \end{bmatrix}$$

Again applying the $\infty$ - norm we have,

$$|r_\kappa| + |1 - 2r_\kappa| + |(1-k_i)r_\kappa| + |k_i r_\kappa| \leq 1 \tag{2.7}$$

Which is satisfied if $r_\kappa \leq 1/2$. Same as the result we obtained for Synchronous. Hence if the Synchronous method converges, the corresponding Asynchronous scheme also converges.

## 2.4 Accuracy

The analysis of the accuracy of a PDE is highly complex and requires Numerical analysis involving Taylor series expansions and more. We will just be quoting the formulas and results derived in the paper, which will be used to recreate the results.

We will define the error in a finite difference method to be

$$E_i^n = u_i^n - u_a(x_i, t_n) \tag{2.8}$$

Where $u_a$ is the real solution ( For Advection - diffusion equation, the real solution can be easily determined ), we will sum $E_i^n$ over all the grid points and compare it between synchronous and Asynchronous. It can be found the behavior of the logarithm of average error as a function of the logarithm of no of grid points in the case of the synchronous scheme has a linear relation with a slope $-2$ asynchronous with a slope $-1$.

## 2.5 Order recovery

We can try to recover some of the accuracies that were lost by following an asynchronous method by fixing the step size time appropriately with the time step in space. By numerical

analysis from the original paper, we can recover the original behavior in the asynchronous case for the advection equation by choosing $\Delta t \sim \Delta x^3$

## 2.6 Numerical Simulation

In the numerical simulation, We can see the error behavior under different dynamical variables of the Asynchronous scheme. Initially, I plotted the entire solution of the PDE both in synchronous and Asynchronous schemes. They are very similar to each other, which can be seen from the overlap of both of the solutions.

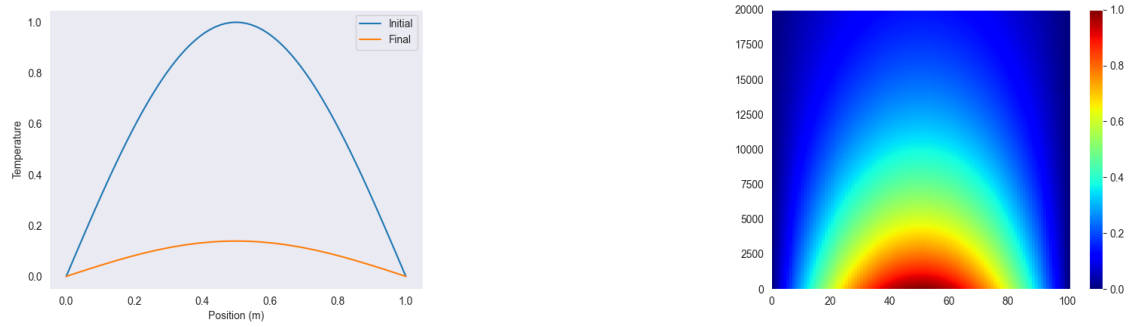
To better understand the two schemes' deviation, we have plotted the average error $\bar{E}_i$ vs. time and tried to understand how it evolves. The error also has a wave pattern similar to the solution. But large peaks in error in the case of Asynchronous methods can be seen near the Boundary points, which is to be expected.

I have also plotted the average error as a function of no grid points for different values of probabilities, where these probabilities correspond to at what chance the value from $n$ is chosen over $n - 1$. We can see the asynchronous methods reaching a slope $-1$, which is theoretically derived and can also be seen in the graph from the original paper.

I have also plotted how the error behaves as we increase the grid points for different values of constant $\Delta t$. It can be seen the error starts to increase as we increase the no of grid points to a large number, therefore clearly showcasing the stability condition stated in Section **??** (All the exact details can be found in the captions of plotted graphs )

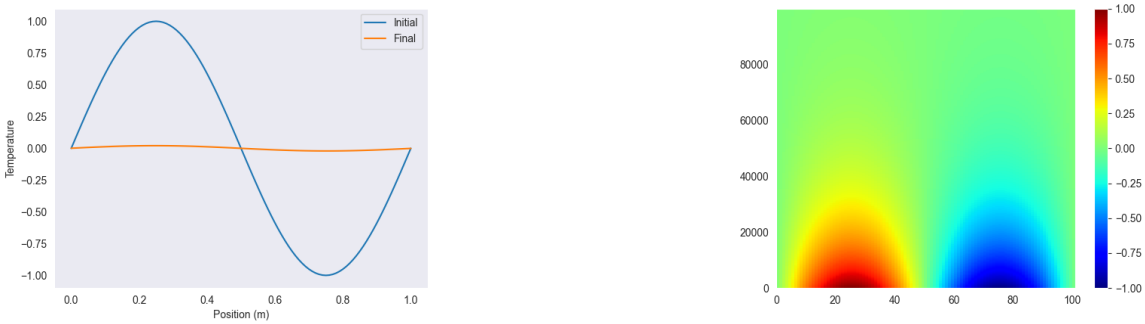
## 2.7 Challenges encountered

- For the initial parts of the projects, trying to animate the results obtained after evolving the partial differential equation required a lot of effort as I found the exact use of the animation package of Matplotlib a little challenging to understand.
- Most of the simulation in the original paper was done in high-processing powered systems which implemented the technique as it was intended initially using parallel processing. The code I have written doesn't use parallel processing aspects and instead implements a serial code to simulate it. The time taken to get the results was demandingly high, and several crashes were encountered while running the code.
- It was a struggle to consider all the dynamic variables affecting the Error. Some of their behavior was difficult to replicate, as found in the original paper.
- Implementing the periodic boundary condition was difficult in higher dimensional cases.
- **Overflow** and **Underflow** error is the biggest that I faced during the report. Plenty of times, the given input variables were off or didn't obey the stability condition, which led to the errors or solutions themselves blowing up to very large values making the results wrong, Finding the perfect set of input variables was a really hard task.



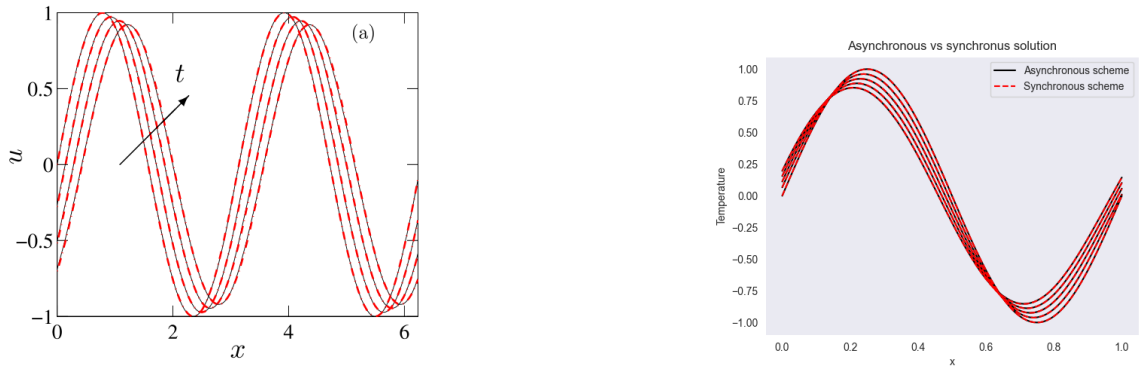

(a) Line plot (b) Color bar plot

**Figure 3.1:** Figure (a) depicts the temperature distribution at the final and initial points in time. Figure (b) depicts the same thing in terms of a color bar but for all points in time (the y-axis is the time step). Both calculations are done for fixed boundaries; therefore, heat is not conserved.



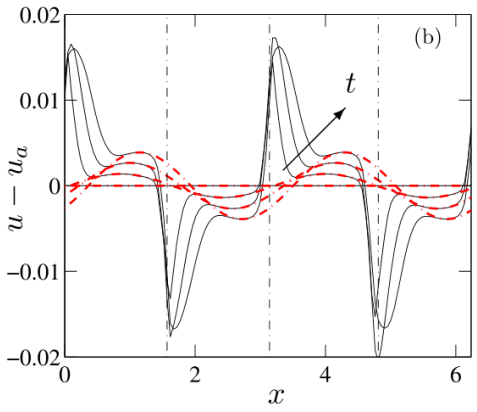

(a) Line plot (b) Color bar

**Figure 3.2:** The same Diffusion equation as the previous figure but using periodic boundary conditions

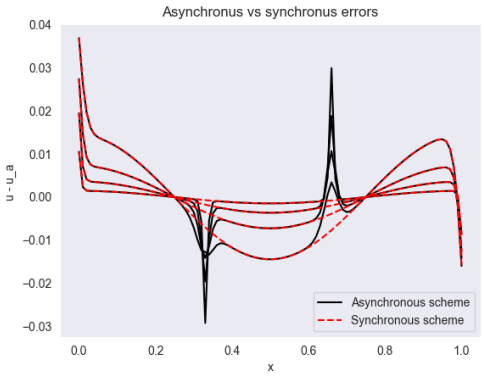



(a) Original (b) recreated

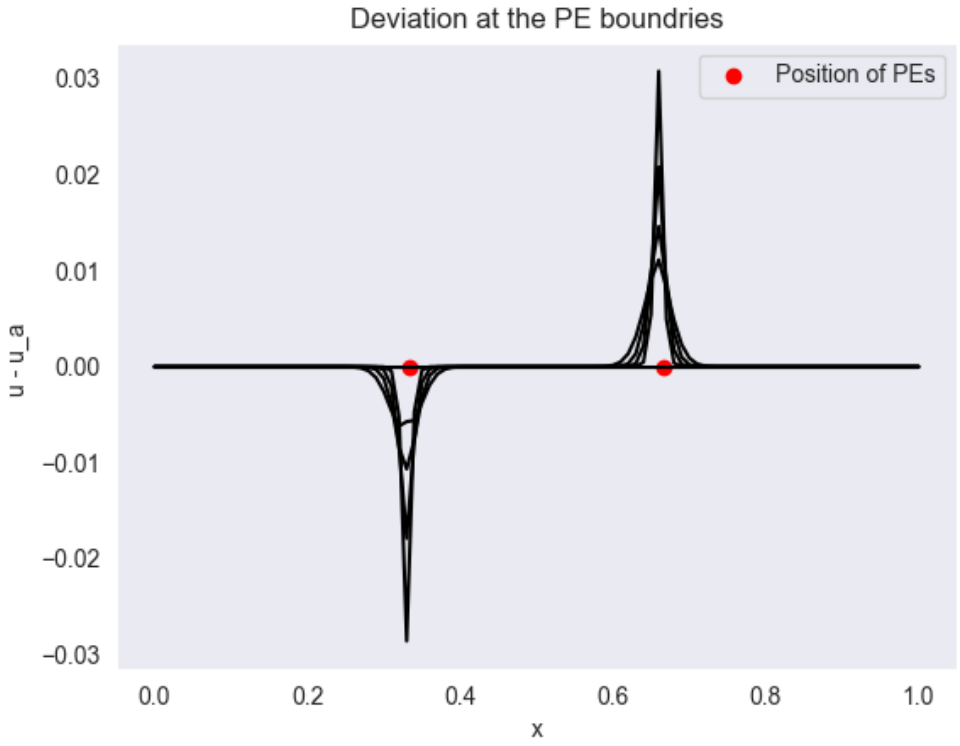
**Figure 3.3:** We can see a high similarity between the original solution in the paper and the recreated solution. The original image contains several cycles but recreated has only one cycle




(a) Original

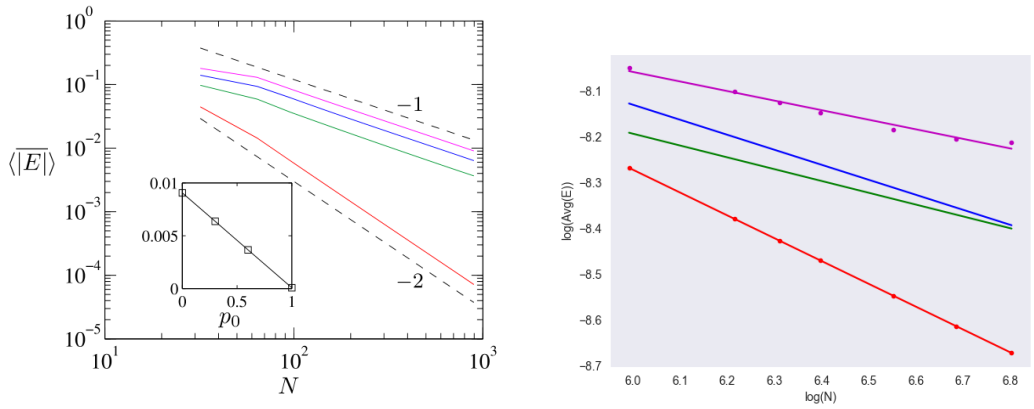



(b) Recreated



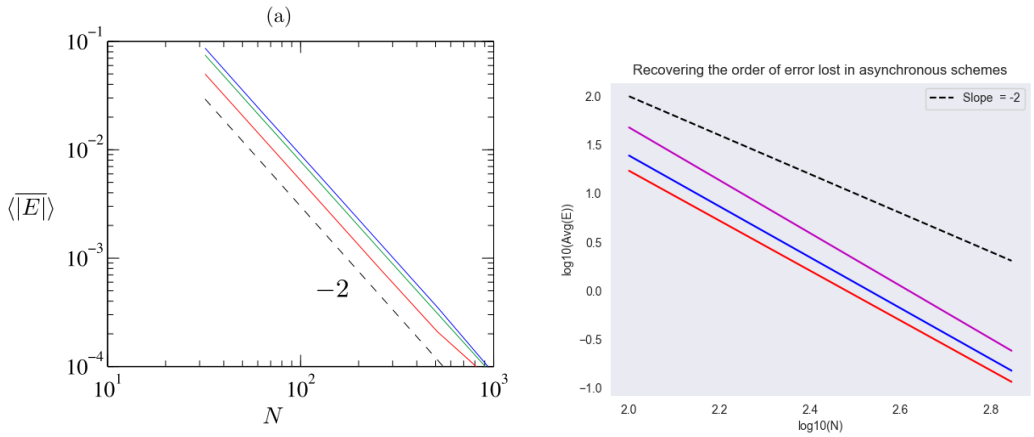

(c) Recreated enhanced

**Figure 3.4:** The original figure and the replicated figure, although having similar properties, are not the exact mirror image of each other. One reasoning is the original has several periods of the sine wave while I have modeled only a small range to make the analysis easier. The other reasoning for this is that the error's magnitude was not as large as the error in the original report. But to understand the general characteristics better, figure (c) has been plotted where we can see the apparent variation from ideal behavior near the PE boundaries.




(a) Original (b) Recreated

**Figure 3.5:** The Original and recreated figure has very close similarities. Here magenta - p = 0.0, blue - p = 0.3, green -p = 0.6, red - p = 1 . Where p is the probability of the boundary point picking the value at $n - 1$ rather than n. Only the points of p = 0 and p =1 have been plotted, as the other points involve randomness owing to p not being 0 or 1




(a) Original (b) Recreated

**Figure 3.6:** These figures correspond to the order recovery process of the asynchronous scheme, which from the earlier discussion, we found will repeat the behavior of synchronous if $\Delta t \sim \Delta x^3$, which was implemented in the code, and we have recreated the results to a very close degree.

# 4 Conclusion

The report showcases only the behavior of the error when we introduce the asynchronous scheme and explains how much we can look past it and the basic way to rectify it. But in real-world applications, except for very rare situations, can we implement this idea successfully. We require correction to the problem introduced by PEs. One such explanation introduced in the paper would be to use higher-order finite difference formula for the derivative. It can be shown using a 5-point finite difference formula instead of a three point for the second derivative would improve the accuracy of the results greatly. A further region to progress is performing the analysis done in case of 1D to higher dimensions as. Usually, the stability condition in case of higher dimensions is much tighter, and even a small violation can cause the system to blow up. The main issue with the partial differential equation is not figuring out the formula to iterate but making sure the error is well bound and doesn't produce very inaccurate results, which I experienced many times during the coding processes. Hence from this report we can conclude a lot of progress in numerical methods research to solve PDE is still required and its a active region of research.

# Thank You