UNIVERSITÄT HAMBURG

UNIVERSITÀ DEGLI STUDI DELL'AQUILA

UNIVERSITAT AUTÒNOMA DE BARCELONA

Erasmus Mundus Consortium MathMods

Joint Degree of Master of Science in Mathematical Modelling in Engineering: Theory, Numerics, Applications

In the framework of the Consortium Agreement and Award of a Joint/Multiple Degree 2013-2019

Master's thesis

Numerical approximation and error analysis of diffusion problems

Supervisor

Dr. Monika Twarogowska

Candidate

S. M. Atiqur Rahman Chowdhury

Matricola:227795

2014/2015

Laurea Magistrale in Ingegneria Matematica Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica Università degli Studi dell'Aquila

I hereby declare that this Thesis is my own work and effort. Where other sources of information have been used, they have been acknowledged.

Acknowledgments

Firstly, I would like to express my sincere gratitude to my advisor ***Dr. Monika Twarogowska*** *for the continuous support of my master thesis study and related research, for her patience, motivation, and immense knowledge. Her guidance helped me in all the time of research and writing of this thesis.*

Besides my advisor, I would like to thank gratefully to the coordinator of ***MathMods*** *program:* ***Prof. Bruno Rubino*** *for his guidance, teaching and encouragement from the begging to the end of my study, specially at the first semester at* ***University of L'Aquila*** *in Italy, I was demoralized because on that time I found study in abroad is too hard for me and I had a decision to quit this program. He suggested me and gave me courage to continue study by taking a challenge, and at the end of this stage, I found I have learned a lot from* ***MathMods*** *and surely acquired knowledge will be effective for my future life.*

I thank my colleges of MathMods for all study discussions, for the sleepless nights we were working together before exams, and for all the fun we have had in the last two years. Also I thank to my all friends from University of Hamburg in Germany for their support when I was studying there for my second semester.

Last but not the least, I would like to thank my family: my parents and to my brother in law and sister for supporting me spiritually throughout finishing this excellence ***MathMods*** *program, writing this thesis and my life in general.*

S. M. ATIQUR RAHMAN CHOWDHURY

To my parents.

Contents

Introduction

Partial differential equations (PDEs) form the basis of many mathematical models of physical, chemical and biological phenomena, and more recently their use has spread into economics, financial forecasting, image processing and other fields. To investigate the predictions of PDE models of such phenomena it is often necessary to approximate their solution numerically, commonly in combination with the analysis of simple special cases; while in some of the recent instances the numerical models play an almost independent role.

This thesis deals with numerical methods for solving partial differential equations (PDEs) coupling linear and non-linear diffusion equation, with a focus on time dependency. Comparison of numerical schemes and accuracy is presented of methods for both linear and non-linear diffusion equation that has been elaborately described in the book by W. Hundsdorfer and J. G. Verwer [9].
Many of those mathematical models are described by a (system of) partial differential equations (PDEs). Well established laws of nature and their mathematical counterparts have led to most of the development of suitable PDEs, e.g., heat conduction, fluid dynamics or deformation of solids [7, 4, 11, 9]. However, sometimes the development and understanding of a particular mathematical model can only be tackled by a trial and error approach, which is a two-step procedure.
In a first step, the simulated results are compared to experimental data and in a second step, these comparisons are used to modify the mathematical model, i.e., most of the case it is the underlying PDE. Hence, numerical simulations of PDEs are of tremendous importance when trying to understand real world processes. On the one hand, in the field of numerical ODEs highly valuable methods and results exist which are of practical use for solving time-dependent PDEs, something which is often not fully exploited by numerical PDE researchers [9].

Although many problems can be solved by Euler's method or the Crank-Nicolson method, better alternatives are often available which can significantly reduce the computational effort needed to solve practical problems [7]. On the other hand, many numerical ODE researchers are unaware of the vast amount of highly interesting results on discretization methods for PDEs [9]. Moreover, when solving PDEs, discretizations in space and time have to be matched, and different spatial discretizations may require different temporal discretizations.

Numerical approximation

Partial differential equations (PDEs) have become enormously successful as models of physical phenomena. With the rapid increase in computing power in recent years, such models have permeated virtually every physical and engineering problem. The phenomena modeled by partial differential equations become increasingly complicated, and so do the partial differential equations themselves. Often, one wishes a model to capture different aspects of a situation, for instance both convective transport and dispersive oscillations on a small scale. These different aspects of the model are then reflected in a partial differential equation, which may contain terms (operators) that are mathematically very different, making these models hard to analyze, both theoretically and numerically. (See [1, 9, 10, 11, 15, 18, 19, 20, 21, 22, 24]).

In order to study numerically any kind of problem we need a stable and consistent numerical scheme [7, 11, 9]. Besides, a good scheme has to reproduce all of the important features of the original model, which arise from the physical background of the problem. First of all the scheme has to preserve the non negativity of solutions as we deal with densities and concentrations. Then, if we consider bounded domains with no-flux boundary conditions, the numerical approximation has to conserve the total mass. Conservation laws with reaction terms are characterized by a special balance between the fluxes and the sources, which can lead to non constant stationary solutions. Their preservation is essential and in the case of geometric sources, containing for example space derivatives, it is impossible by using standard schemes treating the flux and the source at different time steps. Moreover, the presence of the vacuum states brings another difficulty as many schemes produce oscillations at the interface between the regions, where the density is strictly positive and where it vanishes. In order to study the behaviour of the boundary, the approximate solution not only has to be free from oscillations, but also it has to deal with tracking the movement of exact front. Using an appropriate numerical scheme for parabolic model of porous medium type that can satisfy all these properties is not an easy task (see [3, 12]).

Numerical Analysis

We focus our attention on degenerate parabolic equations such as the porous medium equation (PME). It is well known that the degeneracy implies a finite speed of propagation [4, 6]. Equivalently it means that when the initial density has compact support, it will remain compact for all times [14]. In this case, the classical smooth solution may not always exist in general, even if the initial solution is smooth. It is necessary to consider the weak energy solution, whose behavior causes many difficulties for a good numerical simulation. For example, the weak solution may lose its classical derivative at some (interface) points, and the sharp interface of support may propagate with finite speed if the initial data have compact support. Enamored of these interesting facts, there have been many works on the simulation for the non smooth solution of the PME, for example, the finite different method by Graveleau and Jamet [25], the interface tracking algorithm by DiBenedetto and Hoff [26], and the relaxation scheme referred to in [1].

In numerical point of view, implicit θ-scheme possesses several properties to make it very attractive for practical computations, such as parallelization, adaptivity, and simple treatment of boundary conditions. The most important properties of this method is its strong stability and high-order accuracy; as a result, it is very good at capturing discontinuous jumps and sharp transient layers. Explicit reluxation scheme has a good L_2 and L_∞-stability for the standard nonlinear diffusion equation, including the PME for short time simulation where as for long time time simulation explicit-implicit Crank-Nicolson scheme is better in the sense of L_2 and L_∞-stability [1]. We will compare between fully implicit Backward Euler scheme and explicit-implicit Crank-Nicolson scheme and measured the accuracy by the L_1, L_2 and L_∞-norm.

There are two main components in this thesis. The first is the simulation by the finite difference method (FDM) for smooth and non-smooth solutions of the linear diffusion equation and PME (see [9, 6]). We design a non-negativity-preserving the physical relevancy of the numerical solutions (Q. Zhang and Z. Wu. has been done by local degenerate Galarkin (LDG) method, see [12]). The given numerical results verify the above advantage of the FDM method that it has the ability to capture sharp interfaces accurately without or with very little numerical oscillation (see the Figure 3.8). As a comparison, time integration method based of non-negativity is also considered to show the importance to avoid the numerical difficulties. (See section 2.2.1 in Chapter 2).

The second component is an analysis for the non-negativity preservation principle of the considered θ-method, i.e., in each cell of numerical discretization of the Barenblatt solution [8] for the PME remains non-negative for different adiabatic exponents (see the Figure 3.8). At the same time we point out in this thesis that we need to take care of the oscillation for the the PME that for any choice of the adiabatic parameter ensures the stability not only in the L_2 sense but also in the sense of L_∞, then we are guaranteed that the approximation to our desired solution response will lie within the illustrated bounds.

To learn more about numerical methods for parabolic equations including PME, we invite the reader to take a closer look at one or several of the references [7, 9, 10, 11, 16, 17, 18, 20, 21, 24, 25, 26]

Outline of the thesis

The content of this thesis is organized as follows:

In Chapter 1, we studied the background of the research. There we give an introduction for the numerical solution of PDE using finite diffenece method (FDM), its numerical schemes and algebraic formulations. We talk about its stability, non-negativity, describe the endowed boundary conditions regarding on time dependency that has been elaborately described in some books of the references [7, 9, 10, 11]

In Chapter 2, we studied the numerical solution of linear diffusion equation (DE).

We show the numerical difficulties in the simulation for the DE, where the FDM is used to resolve the Gaussian solution (See sec 1.2 of chapter 1 for the general solution). In Sect. 2.1, we describe the total mass for DE for the different boundary conditions, in particular we have tested numerically the result in the case of Dirichlet, periodic and Neumann boundary condition. Numerical result is presented in Figure 2.1. A short analysis on time integration and importance of positivity test in numerical simulation is given in Sect. 2.2. Numerical results regarding stability and accuracy in the sense of L_p-norm (see sec. 2.3 [10]) are presented at the end of this chapter. For more details see [7, 9].

In Chapter 3, we studied the numerical solution of porous media equation (PME). We show the difference between PME and DE and difficulties in the simulation for the PME, FDM is used to resolve the self-similar Barenblatt solution. (for more details about self similar solution, see the book by Barenblatt [2]). In Sect. 3.2, we describe the numerical schemes to solve PME and use to the standard porous media equation ([8, 15]) using the Neumann boundary condition. Numerical result regarding accuracy is presented in Figure **??** and in Figure 3.7. By paying more attention to the movement of the numerical interface for different adiabatic exponents is presented in Figure 3.8. Results are compared with the exact Barenblatt solution and we conclude that implicit-explicit scheme is better to solve the PME.

At the end in Appendix A, generated MATLAB codes are presented for all simulations.

CHAPTER 1

INTRODUCTION TO NUMERICAL SOLUTION FOR DIFFUSION EQUATION

Mathematics has applications in almost all subjects including physics, chemistry, computer science and engineering. Biologists, sociologist,economists and psychologists have vastly benefited from mathematics in their work for drawing conclusions and developing novel techniques of investigation.

In the past decades mathematical thinking has been intensively applied on natural life sciences, especially in the field of ecology, physical processes in nature and many biological phenomena in general. The common goal is to map observable features of the real physical and biological processes to an abstract mathematical model and a corresponding discrete numericalsimulation in order to gain new insights in the underlying real world objectives. Moreover, in several cases the mathematical description of the real world system is the only possibility to provide reliable predictive analysis for the underlying process of nature, which results in templates for, e.g., industrial or medical purposes.

Many of those mathematical models are described by a (system of) partial differential equations (PDEs). Well established laws of nature and their mathematical counterparts have led to most of the development of suitable PDEs, e.g., heat conduction, fluid dynamics or deformation of solids. However, sometimes the development and understanding of a particular mathematical model can only be tackled by a trial-and-error approach, which is a two-step procedure. In a first step, the simulated results are compared to experimental data and in a second step, these comparisons are used to modify the mathematical model, i.e., most of the case it is the underlying PDE. Hence, numerical simulations of PDEs are of tremendous importance when trying to understand real world processes.

The finite difference method uses for the numerical solution of the equations of fluid dynamics. The fundamental idea is straightforward and implementation is relatively simple. Approximations of various levels of accuracy can be readily computed by the use of different finite difference formula. Consistency, stability and error analysis can be carried out.

1.1 Finite Difference Method (FDM)

The aim of finite difference method is to build a numerical scheme . It is based on approximation of the differential operator. The derivatives are replaced by differential quotients. Approximations of solutions are computed at the discretized space and time with respect to the initial and boundary conditions.

Here, we present a short description on finite difference approach for a differntial equation, describe the way of finding accuracy by using Taylor Expansion for the simple differential equation, its discreatization in space and time. Then we introduce the concept of Numerical Scheme in PDE and a brief description of the consistency and stability of the secheme. In the end, method to solve the algebraic equation on the basis of different schemes and approximation of Dirichlet and homogeneous Neumann boundary conditions will be carried out at the same time.

The simplest finite difference approximation is the difference quotient. Consider the differential equation $y' = u(y)$ with initial condition $y(\alpha) = y_0$ defined on some interval $I = [\alpha; \beta]$. We desire to find a solution $y(x)$. Many such problems have no closed form solution and must be approximated numerically. Thus we let $\{x_0 = \alpha; x_1; ....; x_{n+1} = \beta\}$ be a partition of I and we define

$$\Delta x_i = x_i - x_{i-1}. \tag{1.1.1}$$

Note that

$$\frac{y_{i+1} - y_i}{\Delta x_{i+1}} \approx u(y_i), \tag{1.1.2}$$

so that,

$$y_{i+1} \approx y_i + u(y_i)\Delta x_{i+1}.$$

An important concern in FDM is accuracy. To find the order of accuracy, let x_i be given in grid of points where $x_{i+1} - x_i = \Delta x$ for all i, we will use the Taylor series of $u(x)$ for points in neighborhood of x_i around x_i. As an example, say we wish to approximate $u'(xi)$ using a forward difference, meaning we will use the points x_i and x_{i+1}, then the Taylor series of $u(x_{i+1})$ around x_i is

$$u(x_{i+1}) = u(x_i + \Delta x) = u(x_i) + u'(x_i)\Delta x + u''(x_i)\frac{(\Delta x)^2}{2} + O((\Delta x)^3). \tag{1.1.3}$$

So, if we subtract over $u(x_i)$ and solve for $u'(x_i)$ to get

$$u'(x_i) = \frac{u(x_{i+1}) - u(x_i)}{\Delta x} + u(x_i) + u''(x_i)\frac{\Delta x}{2} + O((\Delta x)^2), \tag{1.1.4}$$

so we get our forward difference approximation of the first derivative at x_i

$$u'(x_i) = \frac{u(x_{i+1}) - u(x_i)}{\Delta x} + O(\Delta x). \tag{1.1.5}$$

We note that this is a first order approximation to the first derivative. We say a finite difference approximating an $n - th$ derivative is of order m if we can write it as a function of its neighbors,

$$u^n(x_i) = \sum_{i=0}^{n+1} a_i u(x_i) + O(\Delta x^m). \tag{1.1.6}$$

Likewise, we could do a backward difference (using xi and x_{i-1})

$$u(x_{i-1}) = u(x_i - \Delta x) = u(x_i) - u'(x_i)\Delta x + u''(x_i)\frac{(\Delta x)^2}{2} + O((\Delta x)^3), \tag{1.1.7}$$

and we get our backward difference approximation of the first derivative at x_i

$$u'(x_i) = \frac{u(x_i) - u(x_{i-1})}{\Delta x} + O(\Delta x).$$

We can do this in general for any set of grid points in a neighborhood of x_i, for example we have a second order centered difference (using an balanced amount of points on either side of x_i like for instance using x_{i+1}, x_i, and x_{i-1}) approximation to the first derivative we use the Taylor series

$$u(x_{i+1}) = u(x_i) + u'(x_i)\Delta x + u''(x_i)\frac{(\Delta x)^2}{2} + u'''(x_i)\frac{(\Delta x)^3}{6} + O((\Delta x)^4), \tag{1.1.8}$$

$$u(x_{i-1}) = u(x_i) - u'(x_i)\Delta x + u''(x_i)\frac{(\Delta x)^2}{2} - u'''(x_i)\frac{(\Delta x)^3}{6} + O((\Delta x)^4), \tag{1.1.9}$$

subtract the equation 1.1.9 term from the the equation 1.1.8 to get

$$u(x_{i+1}) - u(x_{i-1}) = 2u'(x_i)\Delta x + O((\Delta x)^3),$$

so

$$u'(x_i) = \frac{u(x_{i+1}) - u(x_{i-1})}{2\Delta x} + O((\Delta x)^2),$$

notice that the Δx^2 terms in the Taylor series cancel each other out giving us, after dividing by Δx, a second order centered difference approximation to the first derivative. Likewise, we can solve for a centered difference approximation to the second derivative. Add the two above Taylor series and subtract $2u(x_i)$ to get

$$u(x_{i+1}) - 2u(x_i) + u(x_{i-1}) = u''(x_i)(\Delta x)^2 + O((\Delta x)^4) \tag{1.1.10}$$

so we have

$$u''(x_i) = \frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{(\Delta x)^2} + O((\Delta x)^2), \tag{1.1.11}$$

FDM creates a particular discretized system using finite differences which approximate the differential equation we wish to solve. We hope that this discretized system will accurately approximate the solution of the differential equation.

1.2 General solutions

Consider the following partial differential equations (PDEs)

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = 0 \quad \text{for} \quad x \in \mathbb{R}, \quad t \geq 0 \tag{1.2.1}$$

and

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2} \quad \text{for} \quad x \in \mathbb{R}, \quad t \geq 0 \tag{1.2.2}$$

with given constants $a \in \mathbb{R}, \quad d > 0$, given initial value $u(x, 0) = u_0(x), \quad \forall x \in \mathbb{R}$ and the periodicity condition

$$u(x + L, t) = u(x, t), \quad \forall t \geq 0. \tag{1.2.3}$$

The reason for considering periodicity conditions is mainly for the ease of presentation of the main concepts. Boundary conditions cause additional theoretical and numerical problems, as we shall see gradually in later sections. Note that with this periodicity condition we only have to compute the solution for $0 \leq x \leq L$.

Equation 1.2.1 is an one-dimensional transport equation. General solution of the equation 1.2.1simply is $u(x, t) = u(x - at, 0)$. Initial profiles are shifted (carried along by the wind) with velocity a. The lines $x - at$ constant in the $(x, t) - plane$ are the characteristics of this transport equation. Along these characteristics the solution $u(x, t)$ is constant.

The equation 1.2.2 is a diffusion problem. Insight in the behavior of solutions can be obtained by Fourier decompositions. Consider

$$\phi_k(x) = e^{2\pi ikx} \quad for \quad k \in \mathbb{Z}, \tag{1.2.4}$$

and,

$$(\phi, \psi) = \int_0^L \overline{\phi(x)}\psi(x)dx. \tag{1.2.5}$$

The functions ϕ_k will be called Fourier modes, and (ϕ, ψ) is an inner product for the function space $L_2[0, L]$, consisting of all square integrable complex functions on $[0, L]$ with identification of functions that differ only on sets of measure zero. The set $\phi_k, k \in \mathbb{Z}$ is an orthonormal basis for this space. For any function $\psi \in L_2[0, L]$ we have

$$\psi(x) = \sum \alpha_k \phi_k(x), \quad with \quad \alpha_k = (\phi_k, \psi), \tag{1.2.6}$$

and,

$$\|\psi\|_{L_2}^2 = \int_0^L |\psi(x)|^2\, dx = \sum |\alpha_k|^2. \tag{1.2.7}$$

Now, consider for the equation 1.2.2 with initial profile $u(x, 0) = \phi(x)$. To find the solution, we apply the Fourier transform to the equation 1.2.2, we have

$$\hat{u}_t(\zeta, t) = D\hat{u}_{xx}(\zeta, t),$$
$$\Rightarrow \hat{u}_t = D(i\zeta)^2\hat{u} = -D\zeta^2\hat{u}.$$

Solving this ODE and using the initial condition above, we have,

$$\hat{u}_t(\zeta, t) = \hat{\phi}(\zeta)e^{-D\zeta^2 t}.$$

therefore,

$$\begin{aligned} u(x,t) &= \frac{1}{(2\pi)^{1/2}} \int_{\mathbb{R}} e^{ix\zeta}\hat{u}(\zeta, t)d\zeta \\ &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{ix\zeta}\hat{\phi}(\zeta)e^{-D\zeta^2 t}d\zeta \\ &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{ix\zeta}\left[\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} e^{-ix_0\zeta}\phi(x_0)dx_0\right] e^{-D\zeta^2 t}d\zeta \\ &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \phi(x_0)\left[\frac{1}{\sqrt{2\pi}} e^{-i(x_0-x)\zeta}e^{-D\zeta^2 t}d\zeta\right] dx_0 \\ &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \phi(x_0)\hat{f}(x_0 - x)dx_0 \end{aligned}$$

By applying inverse Fourier Transformation, we get

$$\hat{f}(x_0 - x) = \frac{1}{\sqrt{2Dt}} e^{-\frac{(x-x_0)^2}{4Dt}} \tag{1.2.8}$$

and consequently, the general solution of the diffusion equation 1.2.2 for, $t > 0$ and $x, x_0 \in \mathbb{R}$ is given by

$$u(x,t) = \frac{1}{\sqrt{4\pi Dt}} e^{-\frac{(x-x_0)^2}{4Dt}} \tag{1.2.9}$$

At $t = 0$ this is a Dirac delta function δ_0, so for computational purposes we must start to view the solution at some time $t = t_\epsilon > 0$. Replacing t by $t_\epsilon + t$ in 1.2.9 makes it easy to operate with a (new) t that starts at $t = 0$ with an initial condition with a finite width. At $x = 0$, (8) becomes

$$u(x,t) = \frac{1}{\sqrt{4\pi Dt}} \tag{1.2.10}$$

while we consider that initial total mass is $1 \geq 0$, Clearly, the concentration of particles at any other point than zero increases, and the profile of the distribution takes the shape of a bell with fatter and fatter tails as time increases. It is straightforward to notice that at a fixed time t, the profile of u has fatter tails when the diffusion coefficient D is larger. This corresponds to the intuition that a larger diffusion coefficient implies a faster spreading process of the particles.

Remark. If $u(x,t)$ is a concentration then $\int_0^L u(x,t)dx$ is the mass in $[0, L]$ at time t. This is a conserved quantity:

$$\frac{d}{dt}\int_0^L u(x,t)dx = \int_0^L u_t(x,t)dx = \int_0^L -au(x,t)dx = -a(u(L,t) - u(0,t)),$$

due to the periodicity for equation1.2.1, and also since,

$$\frac{d}{dt}\int_0^L u(x,t)dx = \int_0^L u_t(x,t)dx = \int_0^L Du_{xx}(x,t)dx = D(u_x(L,t) - u_x(0,t)),$$

due to the periodicity and homogenous Neumann boundary of the diffusion equation 1.2.2.

1.3 Discretizations

The first step in the discretization procedure is to replace the domain $[0, L] \times [0, T]$ by a set of mesh points. Here we apply equally spaced mesh points
$x_i = i\Delta x, \quad i = 0, \ldots, N_x$, and $t_n = n\Delta t, \quad n = 0, \ldots, N_t$
Moreover, u_i^n denotes the mesh function that approximates $u(x_i, t_n)$ for $i = 0, \ldots, N_x$ and $n = 0, \ldots, N_t$. Requiring the equation to be fulfilled at a mesh point (x_i, t_n) leads to the equation, where

$$(x_0, t_0) = (0, 0),$$

and

$$(x_{N_x}, t_{N_t}) = (L, T).$$

After the construction of a numerical domain we define approximations of partial derivatives. The basic differences are First order space derivative:

- backward Euler difference

$$u_x \approx \frac{u_i - u_{i-1}}{\Delta x} \tag{1.3.1}$$

- forward Euler difference

$$u_x \approx \frac{u_{i+1} - u_i}{\Delta x} \tag{1.3.2}$$

2nd order space derivative:

- centered difference

$$u_x \approx \frac{u_{i+1} - u_{i-1}}{\Delta x} \tag{1.3.3}$$

$$u_{xx} \approx \frac{u_{i+1} - 2u_i + u_{i-1}}{(\Delta x)^2} \tag{1.3.4}$$

and, time derivative is the first order **forward Euler difference** as:

$$u_t \approx \frac{u^{n+1} - u^n}{\Delta t} \tag{1.3.5}$$

1.3.1 Space discreatization

In this section we shall consider some simple space discretizations on a uniform grid $x_i = i\Delta x, \quad i = 0, \ldots, N_x$, with mesh width $\Delta x = 1/N_x$. Approximations $w_i(t) \approx u(x_i, t)$, $i = 0, \ldots, N_x$, are found by replacing the spatial derivatives by difference quotients. This gives a finite difference discretization in space. Setting

$$w(t) = (w_0(t), ..., w_{N_x}(t),$$

we then get a system of ordinary differential equations $(ODEs)$

$$w'(t) = F(t, w(t)) \tag{1.3.6}$$

with a given initial value $w(0)$. Often we shall deal with an F that is linear in w,

$$w'(t) = Aw(t) + g(t) \tag{1.3.7}$$

Discretized equation for the Transport Problem

Consider the advection equation 1.2.1 with $a > 0$ and $\Delta x = h$. The formula

$$\frac{1}{h}(u(x-h) - u(x)) = -u_x(x) + O(h), \tag{1.3.8}$$

leads to the I-st order upwind discretization

$$w_i'(t) = \frac{a}{h}(w_{i-1}(t) - w_i(t)), \quad \text{for} \quad i = 1, \ldots, N_x \tag{1.3.9}$$

with $w_0(t) = w_{N_x}(t)$ by periodicity. This is of the form 1.3.26 with $g = 0$ and

$$A = \tfrac{a}{h}\begin{pmatrix} -1 & & & & & 1 \\ 1 & -1 & & & & \\ & 1 & -1 & & & \\ & & \ddots & \ddots & & \\ & & & 1 & -1 & \\ & & & & 1 & -1 \end{pmatrix}.$$

The formula

$$\frac{1}{2h}(u(x-h) - u(x+h)) = -u_x(x) + O^2(h), \tag{1.3.10}$$

gives the $II-$ nd order central discretization

$$w_i'(t) = \frac{a}{2h}(w_{i-1}(t) - w_{i+1}(t)), \quad for \quad i = 1, \ldots, N_x \tag{1.3.11}$$

with $w_0(t) = w_{N_x}(t)$ and $w_{N_x+1}(t) = w_1(t)$ by periodicity. This is of the form 1.3.26 with $g = 0$ and

$$A = \tfrac{a}{2h}\begin{pmatrix} 0 & -1 & & & & 1 \\ 1 & 0 & -1 & & & \\ & 1 & 0 & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & 0 & -1 \\ -1 & & & & 1 & 0 \end{pmatrix}.$$

For smooth profiles the $II-$ nd order scheme is better, but the result of the $II-$ nd order scheme is also far from satisfactory: it gives oscillations, negative values and a significant phase error.

Discretizations for the Diffusion Equation

Consider the diffusion equation 1.2.2 with $D > 0$. We have

$$\frac{1}{h^2}(u(x-h) - 2u(x) + u(x+h)) = -u_{xx}(x) + O(h^2), \tag{1.3.12}$$

gives the $II-$ nd order central discretization, for the diffusion equation

$$w_i'(t) = \frac{D}{h^2}(w_{i-1}(t) - 2w_i(t) + w_{i+1}(t)), \quad for \quad i = 1, \ldots, N_x \tag{1.3.13}$$

with $w_0(t) = w_{N_x}(t)$ and $w_{N_x+1}(t) = w_1(t)$ by periodicity. This is of the form 1.3.26 with $g = 0$ and

$$A = \frac{D}{h^2}\begin{pmatrix} -2 & 1 & & & & 1 \\ 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \\ 1 & & & & 1 & -2 \end{pmatrix}.$$

1.3.2 Time Discreatization

We assume that, the equation 1.2.1, with solution $u(x,t)$, has been discretized in space, resulting in the semi-discrete system (of ODEs)

$$w'(t) = F(t, w(t)), \tag{1.3.14}$$

with $w(t) = (w_i(t))_{i=1}^{N_x} \in R^{N_x}$, N_x being proportional to the number of grid points in space. Fully discrete approximations $w_i^n \approx u(x_i, t_n)$ can now be obtained by applying some suitable ODE method with step size $k = \Delta t$ for the time levels $t_n = nk$. In the following we use $w^n = (w_i^n)_{i=1}^{N_x}$ to denote the vector (grid function) containing the discrete numerical solution. The approach of considering space and time discretizations separately is called the method of lines (MOL). This is not a "method" in the numerical sense, it is a way to construct and analyze certain numerical methods. For a smooth solution it can be assumed that both the differential equation.

A typical MOL reasoning goes as follows: if we know that $\|w(t) - w_h(t)\| \leq Ch^q$ for our space discretization and the ODE theory tells us that $\|w(t_n) - w^n\| \leq Ck^p$, then we have an error bound for the fully discrete approximations

$$\|w_h(t_n) - w^n\| \leq Ck^p + Ch^q.$$

In this section we shall consider these concepts. For the ODE method we consider, as an example, the $\theta - method$

$$w^{n+1} = w^n + k(1-\theta)F(t_n, w^n) + k\theta F(t_{n+1}, w^{n+1}) \tag{1.3.15}$$

with, $F(t_n, w^n) = w'(t_n)$ and as special cases the explicit (forward) Euler method $(\theta = 0)$, the trapezoidal rule $(\theta = \frac{1}{2})$ and the implicit (backward) Euler method $(\theta = 1)$. As we shall see, the order is $p = 2$ if $\theta = \frac{1}{2}$ and $p = 1$ otherwise.

Explicit ODE methods always have a bounded stability domain. Application to a transport equation will lead to a stability condition of the form

$$\frac{ak}{h} \leq C \tag{1.3.16}$$

a so-called CFL-restriction (after Courant-Friedrichs-Lewy), where C depends on the particular method and space discretization. If the space discretization is central then the eigenvalues will be on the imaginary axis, and the ODE method should be selected such that a portion of the imaginary axis is contained in the stability region.

Application of an explicit ODE method to a diffusion equation will give rise to a stability condition

$$\frac{Dk}{h^2} \leq C \tag{1.3.17}$$

with again C determined by the method and space discretization.
Since solutions of the diffusion problems often give rise to rather smooth solutions, this time step restriction makes explicit methods unattractive for such problems. With implicit methods we can avoid such restrictions.

As a rule, with exceptions, **explicit** methods are more efficient for a transport equation than **implicit** methods. For problems with significant diffusion the implicit methods are in general to be preferred. In the appendices some ODE methods and stability restrictions are listed.

The MOL approach, where space and time discretizations are considered separately, is conceptually simple and flexible. However, sometimes it is better to consider space and time errors simultaneously: there may be cancellation of the various error terms.

Explicit Scheme for Transport Equation

This is a one-step method in time, which is also called a two-level method in the context of PDE's since it involves the solution at two different time levels.

Consider $I-$ st order upwind discretization for the transport equation (1.2.1), with $a > 0$, given initial profile and periodicity condition at $x = 0, L$

$$w_i^{n+1} = w_i^n + \beta(w_{i-1}^n - w_i^n) \tag{1.3.18}$$

where, $\beta = \frac{ak}{h}$ and with $\frac{ak}{h} \leq 1$ for stability. This scheme is also known as the Courant-Isaacson-Rees scheme. Matrix form for 1.3.18 is

$$W^{n+1} = AW^n$$

Denote by W^n the vector of $\mathbb{R}^N$ whose components are $w_1^n, \ldots, w_{N_x-1}^n$ and

$$A = \begin{pmatrix} 1-\beta & 0 & & & & \beta \\ \beta & 1-\beta & 0 & & & \\ & \beta & 1-\beta & 0 & & \\ & & \ddots & \ddots & \ddots & \\ & & & \beta & 1-\beta & 0 \\ 0 & & & & \beta & 1-\beta \end{pmatrix}.$$

The terms at the end of the first line comes from the periodic boundary conditions. We use that $w_0^n = w_{N_x}^n$ and $w_1^n = w_{N_x-1}^n$.Except on the two diagonals all the terms vanish. If we insert the exact PDE solution into this difference scheme we get

$$u(x_i, t_{n+1}) = u(x_i, t_n) + \frac{ak}{h}(u(x_{i-1}, t_n)) + k\tau_i^n, \quad \forall, i = 1, \ldots, N_x$$

with a (residual) local truncation error

$$\begin{aligned} \tau_i^n &= [(u_t + \frac{1}{2}\tau u_{tt} + \ldots) + a(u_x - \frac{1}{2}\tau u_{xx} + \ldots)](x_i, t_n) \\ &= -\frac{1}{2}ah(1 - \frac{ak}{h})u_{xx}(x_i, t_n) + O(h^2), \end{aligned} \tag{1.3.19}$$

If we let $\tau \to 0$ with h fixed, we just re-obtain the bound for the spatial error. We see, however, that the error for the above scheme will actually decrease for $\tau \to 0$, and it will be less than the error of the semi-discrete system with exact time integration.

Explicit Scheme for Diffusion Equation

The computationally simplest method arises from using a forward difference in time and a central difference in space for diffusion Equation 1.2.2 is,

$$\frac{w_i^{n+1} - w_i^n}{k} = D\frac{w_{i+1}^n - 2w_i^n + w_{i-1}^n}{h^2}. \tag{1.3.20}$$

We have turned the equation into algebraic equation, also often called discrete equations. The key property of the equations is that they are algebraic, which makes them easy to solve. As usual, we anticipate that u_i^n is already computed such that u_i^{n+1} is the only unknown in Solving with respect to this unknown is easy:

$$w_i^{n+1} = u_i^n + F\left(w_{i+1}^n - 2w_i^n + w_{i-1}^n\right). \tag{1.3.21}$$

where,

$$F = D\frac{k}{h^2}. \tag{1.3.22}$$

We now apply a backward difference in time in 1.2.2, but the same central difference in space:

$$\frac{w_i^n - w_i^{n-1}}{k} = D\frac{w_{i+1}^n - 2w_i^n + w_{i-1}^n}{h^2}. \tag{1.3.23}$$

Now we assume u_i^{n-1} is computed, but all quantities at the new time level n are unknown. This time, it is not possible to solve with respect to u_i^n because this value couples to its neighbors in space, u_{i-1}^n and u_{i+1}^n, which are also unknown. Let us examine this fact for the case when $N_x = 3$. Equation (6) written for $i = 1, \dots, N_x - 1 = 1, 2$ becomes

$$\frac{w_1^n - w_1^{n-1}}{k} = D\frac{w_2^n - 2w_1^n + w_0^n}{h^2}$$
$$\frac{w_2^n - w_2^{n-1}}{k} = D\frac{w_3^n - 2w_2^n + w_1^n}{h^2}$$

The boundary values w_0^n and w_3^n are known as zero. Collecting the unknown new values w_1^n and w_2^n on the left-hand side gives

$$\begin{aligned}(1+2F)\,w_1^n - Fw_2^n &= w_1^{n-1},\\ -Fw_1^n + (1+2F)\,w_2^n &= w_2^{n-1}.\end{aligned}$$

This is a coupled 2×2 system of algebraic equations for the unknowns w_1^n and w_2^n The equivalent matrix form is

$$\begin{pmatrix} 1+2F & -F \\ -F & 1+2F \end{pmatrix}\begin{pmatrix} w_1^n \\ w_2^n \end{pmatrix} = \begin{pmatrix} w_1^{n-1} \\ w_2^{n-1} \end{pmatrix}$$

In the general case, equation 1.3.21 gives rise to a coupled $(N_x - 1) \times (N_x - 1)$ system of algebraic equations for all the unknown u_i^n at the interior spatial points $i = 1, \dots, N_x - 1$. Collecting the unknowns on the left-hand side can be written

$$-Fw_{i-1}^n + (1+2F)\,w_i^n - Fw_{i+1}^n = w_{i-1}^{n-1} \tag{1.3.24}$$

for $i = 1, \dots, N_x - 1$. One can either view these equations as a system for where the w_i^n values at the internal mesh points, $i = 1, \dots, N_x - 1$, are unknown, or we may append the boundary values w_0^n and $w_{N_x}^n$ to the system. In the latter case, all w_i^n for $i = 0, \dots, N_x$ are unknown and we must add the boundary equations to the $N_x - 1$ equations in 1.3.21. A coupled system of algebraic equations can be written on matrix form and correspond to the matrix equation

$$AW^{n+1} = W^n \tag{1.3.25}$$

where $W = (w_0^n, \dots, w_{N_x}^n)$, and the matrix A has the following structure:

$$A = \begin{pmatrix}
A_{0,0} & A_{0,1} & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\
A_{1,0} & A_{1,1} & 0 & \ddots & & & & & \vdots \\
0 & A_{2,1} & A_{2,2} & A_{2,3} & \ddots & & & & \vdots \\
\vdots & \ddots & & \ddots & \ddots & 0 & & & \vdots \\
\vdots & & \ddots & \ddots & \ddots & \ddots & \ddots & & \vdots \\
\vdots & & & 0 & A_{i,i-1} & A_{i,i} & A_{i,i+1} & \ddots & \vdots \\
\vdots & & & & \ddots & \ddots & \ddots & \ddots & 0 \\
\vdots & & & & & \ddots & \ddots & \ddots & A_{N_x-1,N_x} \\
0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & A_{N_x,N_x-1} & A_{N_x,N_x}
\end{pmatrix} \tag{1.3.26}$$

The nonzero elements are given by

$$\begin{aligned} A_{i,i-1} &= -F \\ A_{i,i} &= 1+2F \\ A_{i,i+1} &= -F \end{aligned}$$

for the equations for internal points, $i = 1, \dots, N_x - 1$. Other points,

$$A_{0,0};\, A_{0,1};\, A_{N_x,N_x-1};\, A_{N_x,N_x};$$

takes the value from the boundary condition.
The right-hand side W is written as

$$W = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_i \\ \vdots \\ w_{N_x} \end{pmatrix}$$

We observe that the matrix A contains quantities that do not change in time. Therefore, A can be formed once and for all before we enter the recursive formulas for the time evolution. The right-hand side b, however, must be updated at each time step.

Implicit Scheme for Transport Equation

Consider the advection equation (6),

$$u_t + au_x = 0.$$

Discretization using a backward difference in space and in time gives the scheme

$$(1+\beta)w_j^{n+1} - \beta w_{j-1}^{n+1} = w_j^n \tag{1.3.27}$$

In this case the scheme 1.3.18 can be written in matrix form

$$BW^{n+1} = W^n$$

Denote by W^n the vector of $\mathbb{R}^N$ whose components are $w_1^n, \dots, w_{N_x-1}^n$ and

$$B = \begin{pmatrix} 1+\beta & 0 & & & & -\beta \\ -\beta & 1+\beta & 0 & & & \\ & -\beta & 1+\beta & 0 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -\beta & 1+\beta & 0 \\ 0 & & & & -\beta & 1+\beta \end{pmatrix}$$

The terms at the end of the first line comes from the periodic boundary conditions. We use that $w_0^n = w_{N_x}^n$ and $w_1^n = w_{N_x-1}^n$.Except on the two diagonals all the terms vanish.

Implicit Scheme for Diffusion Equation

Using a forward difference in time and a central difference in space for Diffusion Equation (1.2.2), an Implicit scheme is,

$$\frac{w_i^{n+1} - w_i^n}{k} = D\frac{w_{i+1}^{n+1} - 2w_i^{n+1} + w_{i-1}^{n+1}}{h^2} \, .$$

As usual, we anticipate that u_i^n is already computed such that u_i^{n+1} is the only unknown in solving with respect to this unknown is easy:

$$w_i^{n+1} = w_i^n + F\left(w_{i+1}^{n+1} - 2w_i^{n+1} + w_{i-1}^{n+1}\right) \, .$$

or,

$$(1 + 2F)w_i^{n+1} - F(w_{i+1}^{n+1} + w_{i-1}^{n+1}) = w_i^n \, .$$

where,

$$F = D\frac{k}{h^2} \, .$$

for $i = 1, \ldots, N_x - 1$. One can either view these equations as a system for where the w_i^{n+1} values at the internal mesh points, $i = 1, \ldots, N_x - 1$, are unknown, or we may append the boundary values w_0^n and $w_{N_x}^n$ to the system. A coupled system of algebraic equations can be written on matrix form and correspond to the matrix equation

$$BW^{n+1} = W^n \tag{1.3.28}$$

where $W = (w_0^n, \ldots, w_{N_x}^n)$, and the matrix B has the following structure:

$$B = \begin{pmatrix}
B_{0,0} & B_{0,1} & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\
B_{1,0} & B_{1,1} & 0 & \ddots & & & & & \vdots \\
0 & B_{2,1} & B_{2,2} & B_{2,3} & \ddots & & & & \vdots \\
\vdots & \ddots & & \ddots & \ddots & 0 & & & \vdots \\
\vdots & & \ddots & \ddots & \ddots & \ddots & \ddots & & \vdots \\
\vdots & & & 0 & B_{i,i-1} & B_{i,i} & B_{i,i+1} & \ddots & \vdots \\
\vdots & & & & \ddots & \ddots & \ddots & \ddots & 0 \\
\vdots & & & & & \ddots & \ddots & \ddots & B_{N_x-1,N_x} \\
0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & B_{N_x,N_x-1} & B_{N_x,N_x}
\end{pmatrix} \tag{1.3.29}$$

The nonzero elements are given by

$$\begin{aligned}
B_{i,i-1} &= -F \\
B_{i,i} &= 1 + 2F \\
B_{i,i+1} &= -F
\end{aligned}$$

for the equations for internal points, $i = 1, \ldots, N_x - 1$. Other points,

$$B_{0,0}; B_{0,1}; B_{N_x,N_x-1}; B_{N_x,N_x};$$

takes the value from the boundary condition.
The right-hand side W^n is written as

$$W^n = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_i \\ \vdots \\ w_{N_x} \end{pmatrix}$$

We observe that the matrix B contains quantities that do not change in time. Therefore, B can be formed once and for all before we enter the recursive formulas for the time evolution. The right-hand side W^n, however, must be updated at each time step.

Since the diffusion equation is so called "stiff", and hence this is an implicit method, which allows much larger time steps to be taken than an explicit method, is a very efficient method for the diffusion equation.

Crank-Nicolson Scheme

Another one-step method, which is much more useful in practice as we will see in chapter 2 is the Crank-Nicolson method which is an implicit-explicit method. The idea in the Crank-Nicolson scheme is to apply centered differences in space and forward differences in time, combined with an average in time and can be represent as:

$$\frac{w_i^{n+1} - w_i^n}{h} = D\frac{w_{i+1}^n - 2w_i^n + w_{i-1}^n + w_{i+1}^{n+1} - 2w_i^{n+1} + w_{i-1}^{n+1}}{2h^2}, \tag{1.3.30}$$

and solving as to find again w_i^{n+1} as:

$$w_i^{n+1} = w_i^n + \frac{Dk}{2h^2}\left(w_{i+1}^n - 2w_i^n + w_{i-1}^n + w_{i+1}^{n+1} - 2w_i^{n+1} + w_{i-1}^{n+1}\right), \tag{1.3.31}$$

or,

$$-\lambda w_{i-1}^{n+1} + (1 + 2\lambda)w_i^{n+1} - \lambda w_{i+1}^{n+1} = \lambda w_{i+1}^n - (1 - 2\lambda)w_i^n + \lambda w_{i-1}^n, \tag{1.3.32}$$

where,

$$\lambda = \frac{Dk}{2h^2} = \frac{1}{2}F \tag{1.3.33}$$

where

$$F = \frac{Dk}{h^2}.$$

Also here, as in the Backward Euler scheme, the new unknowns w_{i-1}^{n+1}, w_i^{n+1}, and w_{i+1}^{n+1} are coupled in a linear system

$$AW^{n+1} = BW^n \tag{1.3.34}$$

and find numerical updated solution as:

$$W^{n+1} = (A^{-1})BW^n \tag{1.3.35}$$

where A has the same structure as in 1.3.26, but with slightly different entries:

$$\begin{aligned} A_{i,i-1} &= -\frac{1}{2}F \\ A_{i,i} &= 1 + F \\ A_{i,i+1} &= -\frac{1}{2}F \end{aligned}$$

and, has the same structure as in 1.3.29, but with slightly different entries:

$$\begin{aligned} B_{i,i-1} &= \frac{1}{2}F \\ B_{i,i} &= -1 + F \\ B_{i,i+1} &= \frac{1}{2}F \end{aligned}$$

for the equations for internal points, $i = 1, \ldots, N_x - 1$.

for other points, $A_{0,0}; A_{0,1}; A_{N_x,N_x-1}; A_{N_x,N_x}; b_0, b_{N_x}$ takes the value from the boundary condition.

The right-hand side W^n is written as

$$W^n = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_i \\ \vdots \\ w_{N_x} \end{pmatrix}$$

1.3.3 Boundary Conditions

Periodicity conditions do not often occur in practice. It is more common to impose Dirichlet conditions, where the values at the boundaries are prescribed,

$$u(0,t) = \alpha_0, \quad u(L,t) = \alpha_1, \tag{1.3.36}$$

or, more general, with time dependent boundary values $\alpha_0(t)$ and $\alpha_1(t)$. In the linear transport equation 1.2.1, we need only condition at the inflow-outflow boundary, that is, at $x = 0$ if $a > 0$ and at $x = L$ if $a < 0$.

On the other hand, in the linear diffusion equation 1.2.2, if $D > 0$ then the **Dirichlet condition** (1.3.36) at the outflow boundary will give rise to a boundary layer. A boundary layer of this type will be absent if the homogeneous **Neumann condition** $u_x = 0$ is imposed at the outflow boundary. If $a > 0$ we then have

$$u(0,t) = \alpha_0, \quad u_x(L,t) = 0, \tag{1.3.37}$$

With this condition rapid changes may still occur in the spatial derivatives of u, but u itself will not show the nearly discontinuous behaviour that arises with Dirichlet conditions. In practice, finding correct boundary conditions is a difficult task for the modellers, and much physical insight is needed for systems of equations. Boundary conditions also give rise to several numerical difficulties, some of which will be shortly addressed in this section.

For instance, consider the advection equation

$$u_t + u_x = 0 \tag{1.3.38}$$

for $0 \leq x \leq L$, with given inflow condition $u(0,t) = \alpha_0(t)$ and outflow condition as, $u_x(L,t) = 0$ with initial profile $u(x,0) = u_0(x)$. For the numerical simulation, let $h = L/N_x$ and $x_j = jh$ for $j = 0, \ldots, N_x$. Second order central discretization gives

$$w_j'(t) = \frac{1}{2h}(w_{j-1}(t) - w_{j+1}(t)), \quad \forall, j = 1, \ldots, N_x \tag{1.3.39}$$

with $w_0(t) = \alpha_0(t)$. Here $w_{N_{x+1}}(t)$ represents the value at the virtual point $x_{Nx+1} = L+h$. This value can be found by extrapolation, for example in the form of $\theta-$ scheme for 1.3.38,

$$w_{N_{x+1}} = \theta w_{N_{x+1}}(t) + (1-\theta)w_{N_x}(t). \tag{1.3.40}$$

So that, if we consider $\theta = 1$ boundary condition takes the form by constant extrapolation and if, $\theta = 2$ it takes the form by linear extrapolation. The last choice seems more natural; in fact we can then apply the $I-$ st order upwind discretization at the outflow point.

Neumann boundary condition for the spatial accuracy

Let, the spatial truncation errors for each grid point be

$$\sigma_h(t) = (\sigma_{h,1}(t) \ldots, \sigma_{h,N_x}(t)), \tag{1.3.41}$$

for the transport equation 1.2.1 with $a = 1$ and we find $\sigma_{h,j}(t) = O(h^2)$ for $j < N_x$, whereas at the outflow point in 1.3.40 is

$$\begin{aligned}\sigma_{h,N_x} &= \frac{d}{dt}u(t,x_{N_x}) - \frac{1}{2h}(\theta u(t,x_{N_x}) - \theta u(t,x_{N_{x+1}})),\\ &= -\frac{1}{2}(2-\theta)u_x - \frac{1}{4}\theta u_{xx} + \ldots|_{(x_{N_x},t)}.\end{aligned}$$

So, for the space truncation error we have the bounds

$$\|\sigma_h\|_\infty = O(h^s), \quad \|\sigma_h\|_2 = O(h^{s+\frac{1}{2}}), \quad \|\sigma_h\|_1 = O(h^{s+1}),$$

in the L_∞, L_2 and L_1 norms, with $s=0$ if $\theta=1$ and $s=1$ if $\theta=2$.

Then, the error is,

$$\|w_h(t) - w(t)\| = O(h^{s+1})$$

for all three norms and in numerical simulation we have

$$A = \tfrac{1}{2h}\begin{pmatrix} 0 & -1 & & & & \\ 1 & 0 & -1 & & & \\ & 1 & 0 & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & 0 & -1 \\ & & & & \theta & -\theta \end{pmatrix}, \sigma_h = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \vdots \\ 1 \end{pmatrix} Ch^s + O(h^{s+1})$$

with $C = -\frac{1}{2}u_x(L,t)$ if $\theta=1$, and $C = -\frac{1}{2}u_{xx}(L,t)$ if $\theta=2$ and, hence, a matrix form of 1.2.1 for $a=1$ gives

$$A\zeta = \sigma_h,$$

where, $\|\zeta\| = O(h^{s+1})$ in the L_∞, L_2 and L_1 norms.

Now, consider the diffusion equation 1.2.2 for $D=1$ is

$$u_t = u_{xx}, \tag{1.3.42}$$

with, initial value $u(x,0) = u_0(x)$, and, boundary conditions, $u_x(0,t)=0, u_x(L,t) = 0$, truncation errors for each grid point are as same as 1.3.41.

$II-$nd order central discretization for the diffusion equation 1.3.42,

$$w_i'(t) = \frac{1}{h^2}(w_{i-1}(t) - 2w_i(t) + w_{i+1}(t)), \quad for \quad i = 1,\ldots,N_x \tag{1.3.43}$$

The homogenous Neumann condition at $x=0$ and $x=L$ can be discretized as $\frac{1}{2h}(w_0(t) - w_1(t)) = 0$ and, $\frac{1}{2h}(w_{N_{x+1}}(t) - w_{N_{x-1}}(t)) = 0$ respectively.

Thus we set, with parameter, θ in the equation 1.3.42, we have,

$$w_{N_{x+1}} = \theta w_{N_x+1}(t) + (1-\theta)w_{N_x}(t) \tag{1.3.44}$$

Neumann condition are valid at $x_0 = 0$ and $x_{N_x} = L$. This implies that $u_x(0,t) = u_x(L,t) = 0$. Inserting the exact solution in the difference scheme, we find a $II -$ nd order truncation error, except at x_0 and x_{N_x} where

$$\sigma_{h,1} = \frac{1}{2}\theta u_{xx} + O(h^2),$$

$$\sigma_{h,N_x} = \frac{1}{2}\theta u_{xx} + O(h^2).$$

So, for the space truncation error we have the bounds

$$\|\sigma_h\|_\infty = O(h^2), \quad \|\sigma_h\|_2 = O(h^2), \quad \|\sigma_h\|_1 = O(h^2),$$

in the L_∞, L_2 and L_1 norms and in numerical simulation we have,

$$A = \tfrac{1}{h^2}\begin{pmatrix} p & -p & & & & \\ 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \\ & & & & p & -p \end{pmatrix},$$

where, p any value that we could choose.

1.4 Monotonicity

Many spatial discretizations produce oscillations and negative values. In numerical simulation it is seen that with MOL discreatization for finite difference model of chemical reactions may lead to negative values and so instability can occur. In this section we study monotonic scheme under the consideration of those features mentioned above.

For the equation 1.2.1 and 1.2.2 we know, by physical interpretation,

$$u(x,0) \geq 0, \quad \forall, x \to u(x,t) \geq 0, \forall x \quad and \quad t > 0,$$

space discretizations may destroy this property. We would like to have a criterion that tells us when positivity is preserved. Consider a semi-discrete ODE system

$$w'(t) = F(t, w(t)). \quad i = 1, \ldots, N_x$$

This system will be called positive if

$$w(0) \geq 0 \Rightarrow w(t) \geq 0 \quad \text{for all} \quad t > 0.$$

Positivity may also imply a maximum principle,

$$\text{min}_i w_i(0) \leq w_i(t) \leq \text{max}_i w_i(0), \text{for all} \quad t > 0.$$

Positivity for transport equation discretizations

Consider the transport equation 1.2.1,

$$u_t + au_x = 0.$$

The general spatial discretization formula

$$w_i'(t) = \frac{1}{h}\sum_{k=-s}^{r} \gamma_k w_{i+k}(t), \quad i = 1, \ldots, N_x,$$

with $w_i(t) = w_{i+m}(t)$ to impose the periodicity condition. Here we use r grid points to the left and s to the right of x_i so that in total $r+s+1$ points are used. The set $x_{i-r}, \ldots, x_{i+s}$ is called the stencil of the discretization around x_i. With wider stencils higher orders can be achieved. With $w_{i+N_x} \equiv w_i$. The spatial truncation error is

$$\begin{aligned} u_t(x,t) = \frac{1}{h}u(x+kh,t) &= -au_x - \frac{1}{h}\sum \gamma_k(u + khu_x + \frac{1}{2}k^2h^2u_{xx} + \ldots)|_{(x,t)}, \\ &= -\frac{1}{h}\gamma_k u - (a + \sum_k k\gamma_k)u_x \ldots |_{(x,t)}. \end{aligned} \tag{1.4.1}$$

The conditions for orders are

$$\sum_k \gamma_k = 0, \sum_k k\gamma_k = -a, \ldots, \sum_k k^q\gamma_k = 0,$$

where, q is the order of k in Taylor series for the equation 1.4.1 we see that the requirement for positivity is,

$$\gamma_k \geq 0, \tag{1.4.2}$$

for all, $k \neq 0$. This is satisfied by the $I-$ st order upwind discretization, which is very inaccurate and very diffusive. Unfortunately, it is also optimal under the positive transport discretizations: For, $q \geq 2$, we then have, $\sum_k k^2\gamma_k = 0$, and therefore, from the inequality 1.4.2, we get

$$q \leq 1.$$

Furthermore, if, $q = 1$ then the leading term in the truncation error is proportional to $\sum_k k^2\gamma_k$, and since we have $\sum_k k\gamma_k = -a$, it follows the inequality by 1.4.2

$$\sum_k k^2\gamma_k \geq a,$$

and the minimal error coefficient, $\sum_k k^2\gamma_k = a$ is achieved by the $I-$ st order upwind discreatization.

Positivity for Diffusion Discretizations

In the same way as for the tansport equation, we can consider linear discretizations for the diffusion equation

$$u_t = Du_{xx},$$

with periodicity condition and given initial values. A general formula for the spatial discretization is

$$w_i'(t) = \frac{1}{h^2}\sum_{k=-s}^{r} \gamma_k w_{i+k}(t), \quad i = 1, \ldots, N_x,$$

with $w_{i+N_x} \equiv w_i$. We assume that $s = r$ and $\gamma - k = \gamma_k$, symmetry in space.
For the the symmetric discreatization the spatial truncation error is

$$\begin{aligned} u_t(x,t) &= \frac{1}{h^2}\gamma_k u(x+kh,t) \\ &= -Du_{xx} - \frac{1}{h^2}\sum_k \gamma_k (u + khu_x + \frac{1}{2}k^2h^2u_{xx} + \ldots)|_{(x,t)} \\ &= -\frac{1}{h^2}\sum_k \gamma_k u + (d - \frac{1}{2}\sum_k k^2\gamma_k)u_{xx} \ldots |_{(x,t)}. \end{aligned}$$

So, the conditions for order q (q is even, due to symmetry) are

$$\sum_k \gamma_k = 0, \sum_k k^2\gamma_k = 2D, \sum_k k^4\gamma_k = 0 \ldots, \sum_k k^q\gamma_k = 0.$$

We can see, positivity preserve for the diffusion equation 1.2.2 for $q \leq 2r$.

1.4.1 Monotone scheme

Consider an equation:

$$u_t + f(u)_x = 0, \forall -\infty \leq x \leq \infty, \tag{1.4.3}$$

subject to the initial condition,

$$u(x,0) = \Phi(x), \quad , \forall -\infty \leq x \leq \infty.$$

A finite difference scheme

$$v_i^{n+1} = H(v_{i-k}^n, v_{i-k+1}^n, \ldots, v_{i+k}^n), \tag{1.4.4}$$

is said to be monotone if H is a monotone increasing function of each of its arguments. Let,

$$\begin{aligned} v_i^{n+1} &= H_f(v_{i-k}^n, v_{i-k+1}^n, \ldots, v_{i+k}^n) \\ &= v_i^n - \lambda(h_f(v_{i-k+1}^n, \ldots, v_{i+k}^n) - h_f(v_{i-k}^n, \ldots, v_{i+k-1}^n)), \end{aligned} \tag{1.4.5}$$

be a finite difference approximation to 1.4.3 in conservation form, i,e.,

$$h_f(w, w, \dots, w) = f(w),$$

which is monotone as

$$\frac{\partial H_f}{\partial w_i}(w_{-k}, \dots, w_k) \geq 0,$$

for all $-k \leq i \leq k$.

1.5 System of Nonlinear Equation

The boundary value problem become nonlinear due to the nonlinearity of the governing differential equations, or of the boundary conditions or both. Most physical problems actually nonlinear. There is no difficulty in applying the FDM to discreatize a nonlinear problem; but the difficulty is associated with the solution of the resulting system of algebraic equations. One important technique can be solve efficiently the nonlinear algebraic equation, known as The Newton-Raphson method, or Newton Method.

It is a powerful technique for solving equations numerically. As in the differential calculus, it is based on the idea of linear approximation. The Newton Method, is a method for finding successively better approximations to the roots of a real-valued function or to the zeros of a complex-valued function.

1.5.1 The Newton-Raphson Iteration

The idea of the method is as follows: let $f(x)$ be a well behaved function, and let r be a root of the equation $f(x) = 0$. We start with an estimate x_0 of r. From x_0, we produce an improved-we hope-estimate x_1. From x_1, we produce a new estimate x_2. From x_2, we produce a new estimate x_3. We go on until we are close enough to r. The above general style of proceeding is called iterative. Of the many iterative root-finding procedures, the Newton-Raphson method, with its combination of simplicity and power, is the most widely used.

Let x_0 be a good estimate of r and let $r = x_0 + h$. Since the true root is r, since h is 'small,' we can use the linear (tangent line) approximation to obtain that $h = r - x_0$, the number h measures how far the estimate x_0 is from the exact line.
So that, we have,

$$0 = f(r) = f(x_0 + h) = f(x_0) + hf'(x_0). \tag{1.5.1}$$

As a result unless $f'(x_0)$ is close to 0 we have

$$h \approx -\frac{f(x_0)}{f'(x_0)}.$$

It follows that

$$r = x_0 + h \approx x_0 - \frac{f(x_0)}{f'(x_0)}.$$

Our new zero iteration x_1 of r is therefore given by

$$x_1 \approx x_0 - \frac{f(x_0)}{f'(x_0)}.$$

The next estimate x_2 is obtained from x_1 in exactly the same way as x_1 was obtained from x_0 The second iteration is,

$$x_2 \approx x_1 - \frac{f(x_1)}{f'(x_1)},$$

and so on. Continue in this way. If x_n is the current estimate of (n-1)'th iteration, then the next estimate x_{n+1} is given by

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}. \tag{1.5.2}$$

After each iteration we check to see if the convergence condition

$$|x_{n+1} - x_n| < \epsilon,$$

is satisfied, where the parameter ϵ called tolerance. It is better to choose small parameter ϵ in numerical simulation.

1.5.2 Geometric interpretation

Figure 1.1 represents the curve $y = f(x)$ meets the $x - axis$ at r. Let a be the current estimate of r. The tangent line to $y = f(x)$ at the point $(x_0; f(x_0))$ has equation

$$y = f(x_0) + (x - x_0)f'(x_0). \tag{1.5.3}$$

Let, x_1 be the $x -$ intercept of the tangent line. Then

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}. \tag{1.5.4}$$

The new estimate x_1 is obtained by drawing the tangent line at $x = x_0$, and then sliding to the $x -$ axis along this tangent line. Now draw the tangent line at $(x_1; f(x_1))$ and ride the new tangent line to the $x - axis$ to get a new estimate c. and continue like this until converge to the root r.

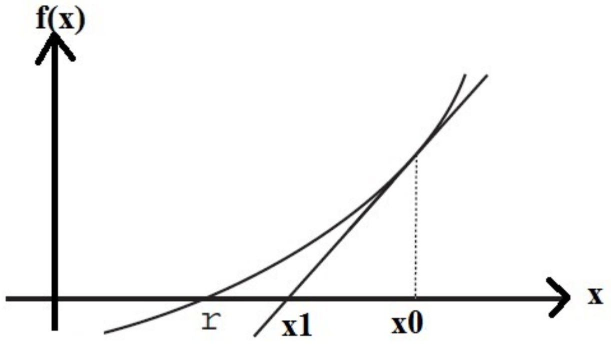

Figure 1.1: Newton-Rapshon procedure, where: x_1 is just the 'next' Newton-Raphson estimate of r.

We can use the geometric interpretation to design functions and starting points for which the Newton Method runs into trouble. For example, by putting a little bump on the curve at $x = x_0$ we can make x_0 fly far away from r.When a Newton Method calculation is going badly, a picture can help us diagnose the problem and fix it. It would be wrong to think of the Newton Method simply in terms of tangent lines. The Newton Method is used to find complex roots of polynomials, and roots of systems of equations in several variables, where the geometry is far less clear, but linear approximation still are accurate. The initial estimate is called x_0, is called a guess. The Newton Method is prefers well if x_0 is close to r, and can lead to large otherwise. The guess, x_0 should be chosen with care.

1.5.3 The Convergence of the Newton method

Suppose x_r is a root of $f(x) = 0$ and x_n is an estimate of x_r s.t. $|x_r - x_n| = \delta << 1$. Then by Taylor series expansion we have

$$0 = f(x_r) = f(x_n + \delta) = f(x_n) + f'(x_n)(x_r - x_n) + \frac{f''(\zeta)}{2}(x_r - x_n)^2, \quad (1.5.5)$$

for some ζ between x_n and x_r. From 1.5.2 we know that

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)},$$

i.e. using 1.5.2 in 1.5.5 we get say, $e_n = (x_r - x_n), \quad e_{n+1} = x_r - x_{n+1}$, where $e_n, \quad e_{n+1}$ denote the error in the solution at n^{th} and $(n+1)^{th}$ iterations. Therefore

$$e_{n+1} = -\frac{f''(\zeta)}{2f'(x_n)} \approx e_n^2, \quad (1.5.6)$$

$$\Rightarrow e_{n+1} \quad \propto \quad e_n^2.$$

Thus, Newton-Raphson method is said to have quadratic convergence. That is, the difference between the root and the approximation is squared (the number of accurate digits roughly doubles) at each step. However, there are some difficulties with the method.
For example, let $f(x) = x^2 - 1$ if $x = 1$, $f(1) = 0$. Then the behavior of $f(x)$ near 1 gives no conclusion to the fact that $f(1) = 0$. So that, successive approximation Newton-Raphson method can't arrive at the solution of $f(x) = 0$.

We assume that $f(x)$ is smooth. We suppose that $f''(x)$ exists and it is continuous near root r. The tangent line approximation is an approximation. Let's try to get a handle on the error. Imagine a particle traveling in a straight line, and let $f(x)$ be its position at time x. Then $f'(x)$ is the velocity at time x. If the acceleration of the particle were always 0, then the change in position from time x_0 to time $x_0 + h$ would be $hf'(x_0)$. So the position at time $x_0 + h$ would be $f(x_0) + hf'(x_0)$ note that this is the tangent line approximation, which is called the zero-acceleration approximation. If the velocity varies in the time from x_0 to $x_0 + h$, then in general the tangent line approximation will not correctly predict the displacement at time $x_0 + h$. And bigger acceleration gives a large error. It can be shown that if f is twice differentiable then the error in the tangent line approximation is $\frac{1}{2}h^2 f''(c)$ for some c between x_0 and $x_0 + h$. In particular, if $|f''(x)|$ is large between $x0$ and $x_0 + h$, then the error in the tangent line approximation is large. Thus we can expect large second derivatives to be bad for the Newton Method. These informal considerations can be turned into positive side about the behavior of the error in the Newton Method. For example, if $\left|\frac{f''(x)}{f'(x)}\right|$ is not too large from near r, and we start with an x_0 close enough to r, the Newton Method converges very fast to r. thus, the study of the convergence of the Newton Method is an important area of Numerical Analysis.

Now, consider the following system of N-algebraic equations:

$$\begin{aligned} f_1(x_1, x_2, \ldots, x_n) &= 0 \\ f_2(x_1, x_2, \ldots, x_n) &= 0 \\ \vdots \qquad\qquad & \vdots \\ \vdots \qquad\qquad & \vdots \\ f_n(x_1, x_2, \ldots, x_n) &= 0 \end{aligned} \tag{1.5.7}$$

We need to find $x_1, x_2, \ldots, x_n$ such that this system of equations is satisfied. To develop the iteration scheme, the equation are written in the vector form as

$$F(X) = 0$$

and it's Taylor series expansion is considered

$$F(X^{m+1}) = F(X^m) + \frac{\partial F}{\partial X}(X^{m+1} - X^m) + \ldots\ldots \tag{1.5.8}$$

We need $F(X^{m+1}) = 0$. The Taylor series is turncated and this condition is imposed to obtain

$$F(X^m) + \frac{\partial F}{\partial X}(X^{m+1} - X^m) = 0, \tag{1.5.9}$$

which is solved for X^{m+1} as

$$X^{m+1} = X(^m) - (\frac{\partial F}{\partial X})^{-1} F(X^m),$$

where $\frac{\partial F}{\partial X}$ is the Jacobian matrix **J** defined as

$$J = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \cdots & \frac{\partial f_1}{\partial x_m} \\ \vdots & & & \vdots \\ \vdots & & & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix}.$$

CHAPTER 2

NUMERICAL SIMULATION OF DIFFUSION EQUATION

We consider basic numerical techniques for solving the one-dimensional diffusion equation. the aim is to study the behavior of solutions of this equation and how this behavior depends on boundary conditions.we consider the implicit-explicit-method and analyze the total mass.
At first we focus on Numann boundary condition, since, in case of homogenious Neumann boundary condition, there is no in-flow and out-flow flux at the boundary, so that Total mass will be conserved, i.e, total mass will be constant for all time. This case will be same for periodic boundary condition, i.e, Total mass will be conserved too, because as same as homogeneous Neumann boundary condition there is now flux flow at the boundary for this case.
Next, we consider Dirichlet boundary condition $u(0,t) = u(L,t) = 0 \quad \forall, x \in [0,L]$ for all time ($t \in [0,T]$), that is, mass at the boundaries will depend on incoming and outgoing flux at the boundary and the total mass will decrease towards final time.

2.1 Total mass

Evaluation of conserved quantities such as mass is one of the fundamental physical principles used to build mathematical models in the natural sciences. Numerical simulations are very useful methods to describe mass propagation phenomena for the linear diffusion equation 1.2.2 and total mass, M implies

$$M = \int_0^L u(x,t)dx \quad \text{for all}, t \in [0,T]. \tag{2.1.1}$$

Time derivative of M gives

$$\frac{dM}{dt} = \frac{d}{dt}\int_0^L u = \int_0^L \frac{\partial}{\partial t} u dx = \int_0^L D u_{xx} dx = D u_x(L,t) - D u_x(0,t) \tag{2.1.2}$$

Equation 2.1.2 implies that the total mass is conserved for the periodic boundary condition as $u(0,t) = u(L,t)$ and for the homogeneous Neumann boundary condition as $u_x(0,t) = u_x(L,t) = 0$. In these two cases there are no flux through the boundary condition, thus the total mass is constant for all time.
For numerical point of view for the explicit scheme the total mass is

$$M^n = M(t^n) = \sum_{j=1}^{n} u_j \Delta x, \tag{2.1.3}$$

and, for the Trapizoidal rule, the total mass is

$$M^n = M(t^n) = \sum_{j=2}^{n-1} u_j \Delta x + \frac{1}{2} u_1 \Delta x + \frac{1}{2} u_0 \Delta x \tag{2.1.4}$$

Numerical Simulation

In numerical simulation the computation of the mass depends on the mesh refinement, more smaller of the space Δx gives the accurate numerical result as compared with the exact value.
We study the total mass for the equation 1.2.2 in the case of periodic, Dirichlet and Neumann boundary condition with the initial value

$$u(x,0) = \begin{cases} 5 & 0.40 \leq x \leq 0.60 \\ 0 & \text{otherwise} \end{cases}$$

According to our initial condition the total mass for the periodic boundary condition or for the homogeneous Neumann boundary condition is

$$M = \int_{0.4}^{0.6} 5dx = 5 * 0.2 = 1 \tag{2.1.5}$$

Now our interest to find the time evolution of the total mass for the equation 1.2.2 in the case of three different boundary conditions (periodic, Dirichlet and Neumann) with the defining initial value.
Where, for all simulation we consider

- Diffusion coefficient: D = 1,
- Space length from 0 to $1.$, i.e. $0 \leq x \leq 1$,
- space step size: $\Delta x = 1/100$,
- Time step size: $\Delta t = (\Delta x)^2 / D$

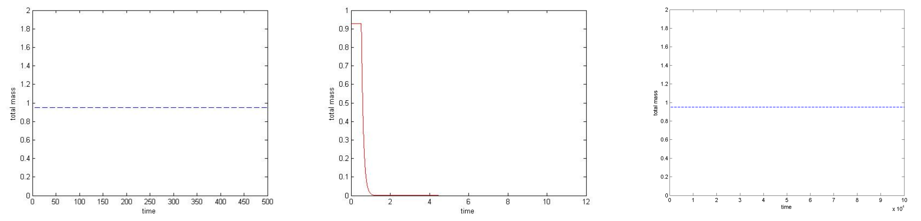

Figure 2.1: Time evolution of the total mass of the equation 1.2.2 with $D = 1$ with periodic (on the left), Dirichlet (in the middle) and homogeneous Nuemann (on the right) boundary condition

Figure 2.1 represents the time evolution of the total mass. We simulate from initial time $t_0 = 1$ to the final time $T = 5$ for the Dirichlet boundary condition and observe that the total mass is constant until initial time $t_0 = 1$ and then started decreasing. At the final time $T = 5$ we see that the total mass becomes almost zero which satisfy the theory for the total mass in the case of Dirichlet boundary condition. For the periodic we simulate from initial time $t_0 = 1$ to the final time $T = 500$ and homogeneous Neumann boundary condition from initial time $t_0 = 1$ to the final time $T = 1000$, we see that the total mass is conserved for all time due to no flux flow at the boundary.
We also notice that for all simulations time evolution of the total mass is not exactly 1 as for the exact calculation that we have found in equation 2.1.5. The reason is numerical error and that could be better if we refine our mesh and make its size smaller.

2.2 Time Integration

Stability is an important property of numerical schemes. Stability of numerical schemes is connected with both special and temporal discreatization, In this case of linear equations for the schemes can be written as a linear condition of values at nodes and it is possible to find condition at Δt, finding strict condition for time step is cumbersome.
In practice, thus it is particular interest for implicit schemes which are used with large time steps has a benefit from the favorable stability. To do so, we can consider $\theta -$ scheme, where we know, if $\theta = 1$, it executes as implicit Euler scheme, and when $\theta = 1/2$ it executes as implicit Trapizoidal rule. However, in both implicit schemes it needs to satisfy positivity property on time integration methods, because, theory tells that, if $\theta \leq 1/2$ it is unconditionally stable, but, when $1/2 < \theta \leq 1$ it gives oscillation under the violation of positivity property which is $\frac{\Delta t}{\Delta x^2} \leq 1$, but by the fully implicit Euler scheme ($\theta = 1$) we can simulate our model as much time as we can without notable oscillation, in other words, implicit Euler method is unconditionally positive for all time.
Positivity properties put restrictions on time integration methods. In this section we are going to discuss the positivity for the $\theta -$ method and we study the importance to

chose a scheme for the time consideration in practice.
Further, it should also be noted that, for the linear diffusion equation 1.2.2 numerical diffusive terms are usually very stiff, that is, some numerical result take place on very small time scales compared to the overall time scale, due to large diffusion constants. This implies that such terms have to be solved implicitly, which makes them difficult and time consuming.
Moreover, with large diffusive constants for the numerical schemes has drawback too. A very simple example, consider, $f(\zeta) = -D\zeta^2$ with reaction constant $D >> 1$. Then solutions of $\zeta'(t) = f(\zeta(t))$ are only stable if we start with $\zeta(0) \geq 0$. With $\zeta(0) < 0$ there will be a blow-up of the solution.

2.2.1 Positivity of Numerical Scheme

Linear Positivity

A linear, semi-discrete scheme for diffusion equation can be written as

$$w'(t) = Bw(t) \tag{2.2.1}$$

where, $w = u(x,t)$, and $w'(t) = \frac{\partial}{\partial t}u(x,t)$. $B \in \mathbb{R}^{n\times n}$ satisfies for the time discreatization

$$b_{ij} \geq 0 \quad for \quad i \neq j \quad \text{and} \quad b_{ii} \geq -\beta, \quad \text{for all} \quad i, \tag{2.2.2}$$

where, $\beta > 0$ is a fixed number.
Let us consider the forward and backward Euler methods. Equation 2.2.1, becomes for the forward Euler method as,

$$w_i^{n+1} = (I + \Delta tB)w_i^n, \tag{2.2.3}$$

where we can see that $I + \Delta tB \geq 0$ provided $1 + \Delta tb_{ii} \geq 0$. This holds if the time step satisfies such that

$$\Delta t\beta \leq 1. \tag{2.2.4}$$

The backward Euler method gives

$$w_i^{n+1} = (I - \Delta tB)^{-1}w_i^n. \tag{2.2.5}$$

Suppose that B has no eigenvalues in the positive real axis. Then $I - \Delta tB$ is invertible for all $\Delta t > 0$ and thus the implicit relation in the backward Euler scheme has unique solution. In fact, this solution is also positive.
In case of, $\theta -$ method since it's a Explicit-Implicit scheme, that is

$$w_i^{n+1} = w_i^n + (1-\theta)\Delta tBw_i^n + \theta\Delta tBw_i^{n+1}, \tag{2.2.6}$$

according to the previous result, positivity is guaranteed if the step size is restricted such that

$$\Delta t\beta \leq 1/\left(1-\theta\right). \tag{2.2.7}$$

Numerical Simulation

For, sufficiently small time step, it is difficult to understand the difference between Trapizoidal rule and backward Euler scheme because, then $\theta -$ method meets the criteria for the positivity condition, i.e, $\beta\Delta t < 1$, but if we consider time step Δt large, numerical result shows that, for the backward Euler scheme $(\theta = 1)$ in $\theta -$ method has no oscillation where as for $\theta = 1/2$, that is, for the Trapezoidal rule, we have oscillated result.

Thus, for the time integration method, for a large time step in order to get best approximated result in numerical computation, it is important to use Implicit Euler method than Explicit-Implicit scheme to avoid negative values and oscillation.
As an illustration we consider a diffusion equation with homogeneous Dirichlet boundary condition value problem

$$\begin{aligned} \partial_t u &= \partial_{xx} u \quad x \in (0,1), \quad t > 0 \\ u(0,t) &= 0, \quad t > 0 \\ u(1,t) &= 0, \quad t > 0 \end{aligned} \tag{2.2.8}$$

on the interval [0,1] with a discontinuous initial condition:

$$u(x,0) = \begin{cases} 0 & \text{if}, 0 \le x < 0.5 \\ 1 & \text{if}, 0.5 \le x < 1 \end{cases}$$

we use space a second order central difference discretization that gives positive approximations

$$\partial_{xx} u(x,t) = \frac{1}{\Delta x^2} \begin{bmatrix} -2 & 1 & 0 & \dots & 0 \\ 1 & -2 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 & -2 \end{bmatrix} u(x,t)$$

Time evolution of the solution $\Delta t = \Delta x = 1/50$ for the backward Euler and the Trapizoidal rule is presented in Figure 2.2

A sufficient condition for positivity of the Trapizoidal rule is $\frac{\Delta t}{\Delta x^2} \le 1$, which is clearly not satisfied here. The behavior of the trapezoidal rule in this example is actually determined by several properties.

Due to lack of positivity the scheme produces large over- and undershoots in the initial phase. Secondly, due to the poor damping properties of the trapezoidal rule, these over- and undershoots persist for a long time. Moreover, due to the fact that $R(z) = -1$ for $z = -\infty$, high-frequency spatial Fourier modes are amplified by a factor close to -1, and this causes the oscillatory behavior in the figure 2.2.

In practice, problems with positivity are not very often encountered with linear parabolic equations. The solutions for such problems are in general rather smooth and

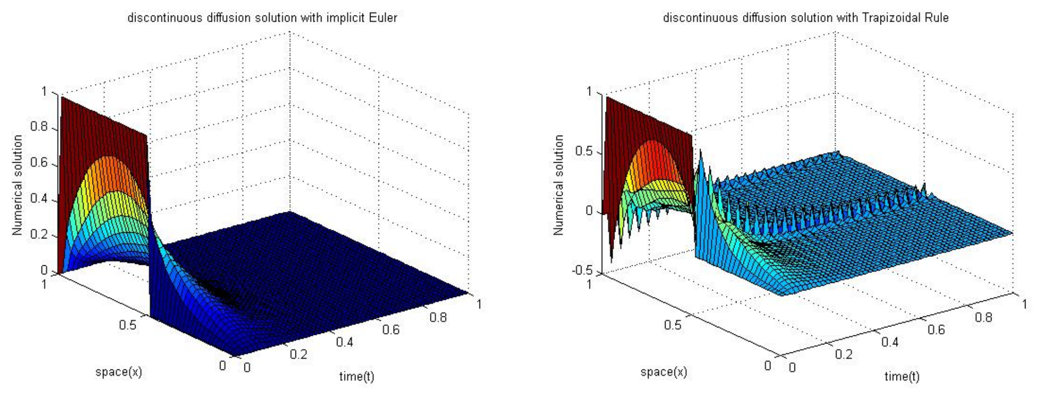

Figure 2.2: Discontinuous diffusion solutions with backward Euler (left) and the trapezoidal rule (right). Time evolution for increasing t in upper-right direction.

then negative values will not show up due to sufficient accuracy. Also in the discontinuous example presented here that tells us to focus on negative values in computations, negative values could have been avoided by starting the trapezoidal rule with small Δt and then gradually increasing the time step.

2.3 L$_p$ – error

When we study finite difference scheme to solve a differential equation, it is generally a good idea to test the order of accuracy of a scheme to ensure that it is producing correct results with the expected accuracy. How can we do this?

A first step is often to try on a problem for which the exact solution is known, in which case we can compute the error in the numerical solution exactly. Not only can we then check that the error is small on some grid, we can also refine the grid and check how the error is behaving asymptotically, to verify that the expected order of accuracy and perhaps even error constant are seen.

To do so, we shall focus on two types of normed based error estimates for the finite difference method, one is L_2- norm and another one is $L_\infty-$norm. These two types of estimates serve very different purposes. The main feature of these estimates is that they tell us the order of convergence of a given finite difference method, that is, they tell us that the finite difference error. The goal of these estimates is to give us a reasonable measure of the efficiency of a given method by telling us how fast the error decreases as we decrease the mesh size. In adaptive mesh refinement, a time influence error estimators are used to indicate where the error is particularly high, and more mesh intervals are then placed in those locations.

Before going to describe error estimation, in the next section we are going to give some numerical simulation to show the importance to check errors in numerical methods.

Numerical Simulation

It is important to test a computer program by refining grid even if the results look quite good in one particular grid. A subtle error in programming can lead to a program that gives reasonable results and able to track location to the exact solution more accurately. To see this, consider the following example

$$\begin{aligned} \partial_t u &= D\partial_{xx}u \quad x \in [0, L], \quad t \in [0, T] \\ u(x, 0) &= f(x) \quad x \in [0, L] \\ u(0, t) &= u(L, t), \quad t > 0 \end{aligned} \tag{2.3.1}$$

We study the qualitative behavior of the numerical solution and results are compared with exact Gaussian solution (1.2.9). For the numerical simulation we choose initial value $u(x, 0)$ is a Gaussian solution at initial time $t_0 = 0.1$. Figure 2.3 represents the difference between explicit and implicit scheme at the final time $T = 5$. Without taking care of CFL condition, we choose time step Δt as equal to space step $\Delta x = 0.1$, from left to right we plot the result for explicit forward Euler scheme, implicit backward Euler scheme and implicit-explicit Crank-Nicolson scheme and compared the result with Gaussian solution G by the formula in 1.2.9. Although definite conclusions cannot be drawn from the simple test, but from this plot it is clear that the implicit discretizations give quite good results compared to their explicit counterparts in terms of accuracy for the large time steps.

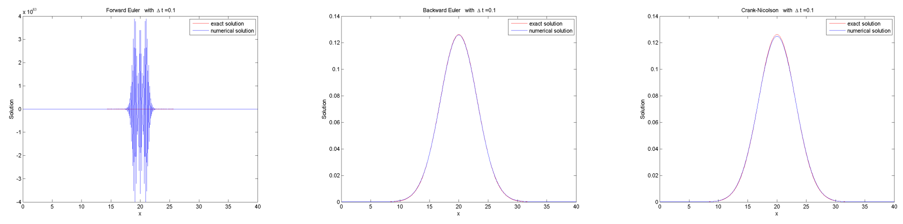

Figure 2.3: Numerical solution at final time, $T = 5$ the diffusion equation for Forward Eluer scheme (on the left), Backward Eluer scheme (in thev middle) and Crank-Nicolson scheme (on the right) with periodic boundary condition, where diffusive constant $D = 1$, space step $\Delta x = 0.1$. Result are compared with Gaussian solution.

We now pay more attention to the movement of the numerical solution by considering CFL condition. We take CFL condition $\Delta t = 0.5 * \Delta x^2$ and plot the result in Figure 2.4. We see at final time $T = 5$ we see all schemes are now giving good results and can able to track exact Gaussian solution more accurately compare to the Figure 2.3.

To get better approximation we see explicit scheme needs small time steps such as $\Delta t = 0.005$ where as implicit scheme doesn't need to take care of time step restriction to get good result for the equation 2.3.1 but smaller time step find larger final time which is time consuming and laborious in numerical simulation. We conclude

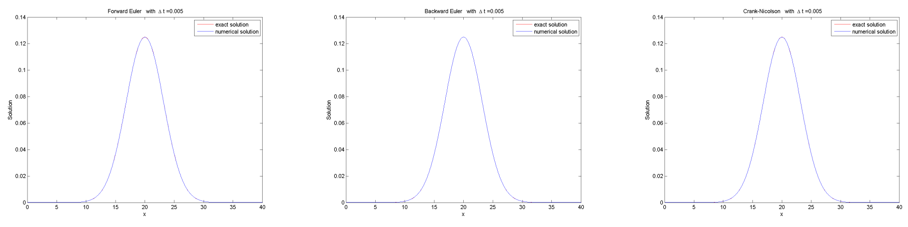

Figure 2.4: Numerical solution at final time, $T = 5$ the diffusion equation for Forward Eluer scheme (on the left), Backward Eluer scheme (in the middle) and Crank-Nicolson scheme (on the right) with periodic boundary condition, where diffusive constant $D = 1$, space step $\Delta x = 0.1$. Result are compared with Gaussian solution.

from both Figures 2.3 and 2.4 that the implicit scheme is a good choice to simulate the equation 2.3.1 but concluding the result by the accuracy point of view, it is important to check numerical error from each grid points.

Now we study numerical for the implicit backward Euler scheme. Figure 2.5 represents from the previous results as in Figure 2.4, if we fix time steps i.e $\Delta t = 0.0005$ at the final time $T = 5$ we see, it gives much accurate result if we refine mesh by decreasing its lengths. Here, we start simulation using mesh size $\Delta x = 0.5$ and finish our simulation after refining 8 mesh size and end-up with $\Delta x = 0.08333$, we see numerical solution gives much accurate result as mesh size becomes smaller, for each simul;ation results are compared with exact Gaussian solution.

All simulation represents that, even after getting good appropriation result, some times it is very critical to distinguish two solutions at a glance. To make accurate comments from any simulation it is thus necessary to analyze the error of the scheme to choose better scheme for simulation. Thus for the next section, we are going to present, numerical error for the three different schemes where error will be calculated by the L^p- norm as

$$\|v\|_p = \left(\Delta x \sum_{i=1}^{n} |v_i| \right)^{1/p} \quad ; \quad \forall 1 \leq p < \infty \qquad (2.3.2)$$

2.3.1 Error Analysis

In above section we have shown simulations to show the importance for the grid refinement and necessity to check the error to conclude the result is accurate. In this sections, we are interested to check the error in each grid points, to get better accuracy first suppose, we know the true solution (U^n) at time $t = t_n$. Let E^n denote the error in the calculation with time and space, that is in each grid point, as computed using the exact solution. We suppose that E^n is a scalar, typically some norm of the error over the grid points, i.e., $E^n = \left\| u^{\text{numerical}} - U^{\text{exact}} \right\|$ where $u^{\text{numerical}}$ is the numerical solution vector and U^{exact} is the true solution evaluated at each grid point by using

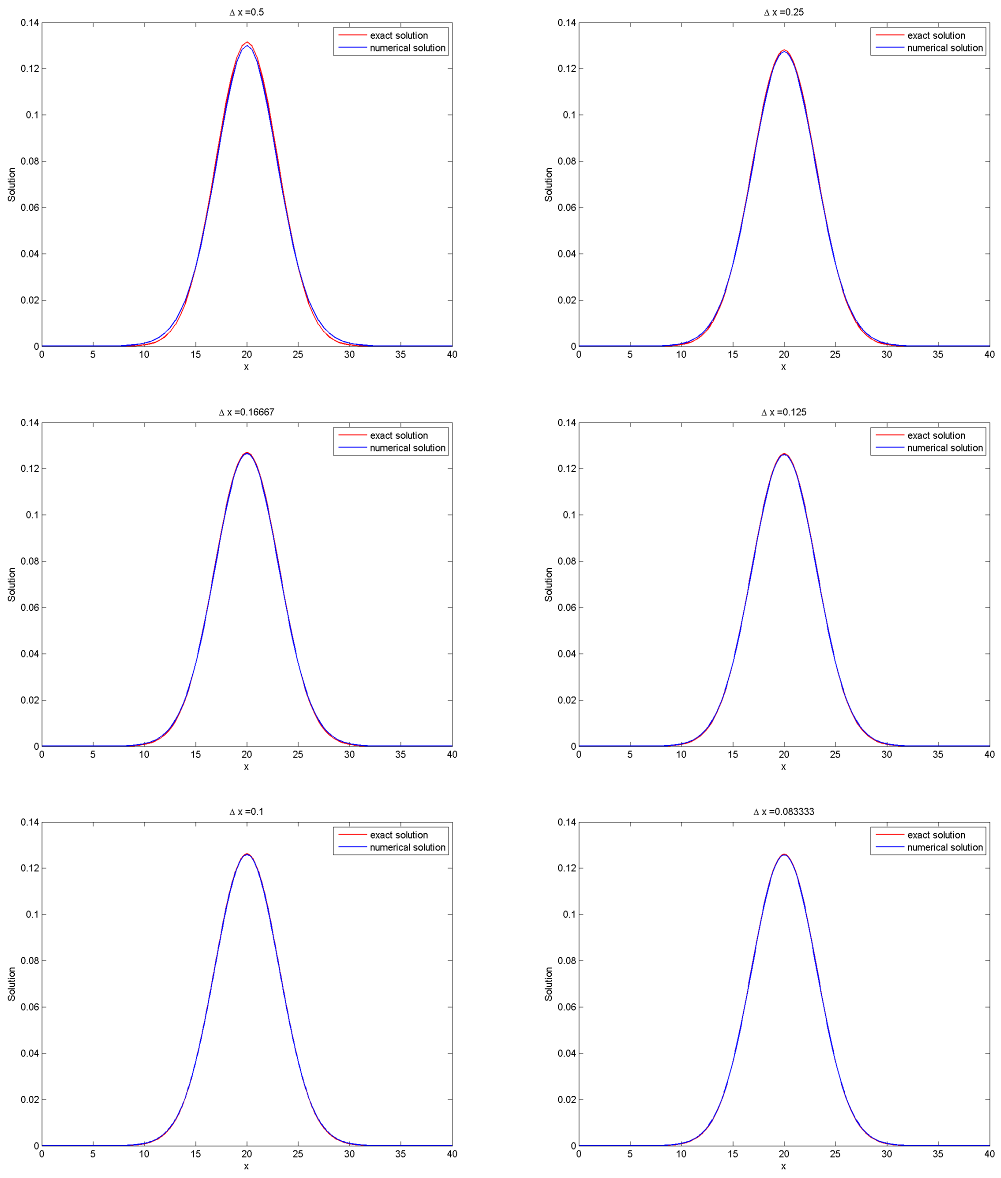

Figure 2.5: Numerical solution for the diffusion equation 2.3.1 with $D = 1$, $\Delta t = 0.0005$, $\Delta x = 0.5$ (top-left),. . ., $\Delta x = 0.08333$(bottom-right), results are compared with Gaussian solution.

time steps and space steps. So, At each grid point there will be some error which is important to check and now we are going to check error on the basis of L_1, L_2 and L_∞ norm and we will observe the qualitative behavior of error on each grid point with grid refinement to have better result.

L_1, L_2 and $L_\infty-$ error by refining time steps and mesh sizes

We study the L_1, L_2 and $L_\infty-$norm to measure the error for the equation 2.3.1 different values of Δx, Δt. For any vector $v \in \mathbb{R}$, L_1, L_2 and L_∞ norm performed by the formula as 2.3.2

$$\|v\|_1 = \Delta x \sum_{i=1}^{n} |v_i|\,,$$

$$\|v\|_2 = \left(\Delta x \sum_{i=1}^{n} |v_i|\right)^{1/2},$$

and,

$$\|v\|_\infty = max_{1\le i\le n}\,|v_i|\,.$$

Using, L_1 – norm we study mass error at the final time in each grid point and see how it response after each grid refining. Using, L_2 – norm we are going to measure error to know at final time response and see how our simulation is differ from exact solution at each final time. But, we also want the best fit to a specified numerical response in an L_∞ – norm sense rather than the L_2 sense. This minimizes the maximum error of the approximation. If the L_∞ – norm of the error is small, then we are guaranteed that the approximation to our desired solution response will lie within the illustrated bounds.

We are interested first, to check the influence of the time step in the error for L_1, L_2 and L_∞ sense. At each time step we have computed the errors starting at initial time $t_0 = 0.01$ up to the final time $T = 5$ for the computational simplicity we used periodic boundary condition for the numerical simulation and result are compared with Gaussian solution and accuracy table is presented in 2.1, 2.2 and 2.3 for the forward Euler, backward Euler and Crank-Nicolson schemes respectively.

Δt	Number of time steps (T-step)	L_1 – error	L_2 – error	L_∞ – error
4.0000e-02	1.2600e+02	6.4155e-03	1.0144e-02	5.2213e-04
2.0000e-02	2.5100e+02	3.4931e-03	5.5231e-03	3.0370e-04
1.3333e-02	3.7600e+02	2.5343e-03	4.0071e-03	2.3100e-04
1.0000e-02	5.0100e+02	2.0587e-03	3.2552e-03	1.9461e-04
8.0000e-03	6.2600e+02	1.7803e-03	2.8149e-03	1.7276e-04
6.6667e-03	7.5100e+02	1.5947e-03	2.5215e-03	1.5819e-04
5.7143e-03	8.7600e+02	1.4657e-03	2.3174e-03	1.4778e-04
5.0000e-03	1.0010e+03	1.3732e-03	2.1712e-03	1.3997e-04

Table 2.1: Estimated error for the diffusion equation 2.3.1 by refining time step Δt, numerical simulation has been done by using Forward Euler scheme from initial time $t_0 = 0.1$ to the final time $T = 5$, where, $x \in [0, 40]$ and grid forms with step sizes $\Delta x = 0.4$ and $D = 1$

Δt	Number of time steps (T-step)	L_1 – error	L_2 – error	L_∞ – error
4.0000e-02	1.2600e+02	1.0248e-02	1.6204e-02	1.0390e-03
2.0000e-02	2.5100e+02	5.4882e-03	8.6776e-03	5.6109e-04
1.3333e-02	3.7600e+02	3.9168e-03	6.1930e-03	4.0226e-04
1.0000e-02	5.0100e+02	3.1356e-03	4.9579e-03	3.2293e-04
8.0000e-03	6.2600e+02	2.6699e-03	4.2214e-03	2.7535e-04
6.6667e-03	7.5100e+02	2.3608e-03	3.7328e-03	2.4365e-04
5.7143e-03	8.7600e+02	2.1422e-03	3.3872e-03	2.2101e-04
5.0000e-03	1.0010e+03	1.9783e-03	3.1280e-03	2.0403e-04

Table 2.2: Estimated error for the diffusion equation 2.3.1 by refining time step Δt, numerical simulation has been done by using Backwar Euler scheme from initial time $t_0 = 0.1$ to the final time $T = 5$, where, $x \in [0, 40]$ and grid forms with step sizes $\Delta x = 0.4$ and $D = 1$

Δt	Number of time steps(T-step)	L_1 – error	L_2 – error	L_∞ – error
4.0000e-02	1.2600e+02	8.2212e-03	1.2999e-02	7.7820e-04
2.0000e-02	2.5100e+02	4.4450e-03	7.0282e-03	4.3194e-04
1.3333e-02	3.7600e+02	3.1865e-03	5.0383e-03	3.1643e-04
1.0000e-02	5.0100e+02	2.5632e-03	4.0528e-03	2.5865e-04
8.0000e-03	6.2600e+02	2.1960e-03	3.4722e-03	2.2398e-04
6.6667e-03	7.5100e+02	1.9594e-03	3.0980e-03	2.0087e-04
5.7143e-03	8.7600e+02	1.7927e-03	2.8346e-03	1.8436e-04
5.0000e-03	1.0010e+03	1.6696e-03	2.6399e-03	1.7197e-04

Table 2.3: Estimated error for the diffusion equation 2.3.1 by refining time step Δt, numerical simulation has been done by using Crank-Nicolson scheme from initial time $t_0 = 0.1$ to the final time $T = 5$, where, $x \in [0, 40]$ and grid forms with step sizes $\Delta x = 0.4$ and $D = 1$

Figure 2.6 represents the comparison of numerical solution for three different schemes and show the result in terms of L_1 , L_2 and L_∞-error for the equation 2.3.1, error are executed by the by the norm of E^n that has been described at the first paragraph of this section. We start simulation from time step $\Delta t = 4e - 02$ and end-up simulation with $\Delta t = 5e - 03$, we see more smaller time steps produce more accurate result and under the CFL condition, at each final time $T = 5$ with different time steps explicit forward Euler scheme can track much accurately the exact solution than implicit schemes.

Figures 2.7 represents the accuracy result from Tables 2.1, 2.2 and 2.3 for forward Euler scheme, backward Euler scheme and Crank-Nicolson scheme. We see L_∞-error ensures that our simulation gives all solutions lies within the boundary which satisfy the theory that we mentioned before.

Now we are interested to check the error by refining space steps. Previously in Figure 2.5 we have shown an importance of refining mesh to get more better result. For

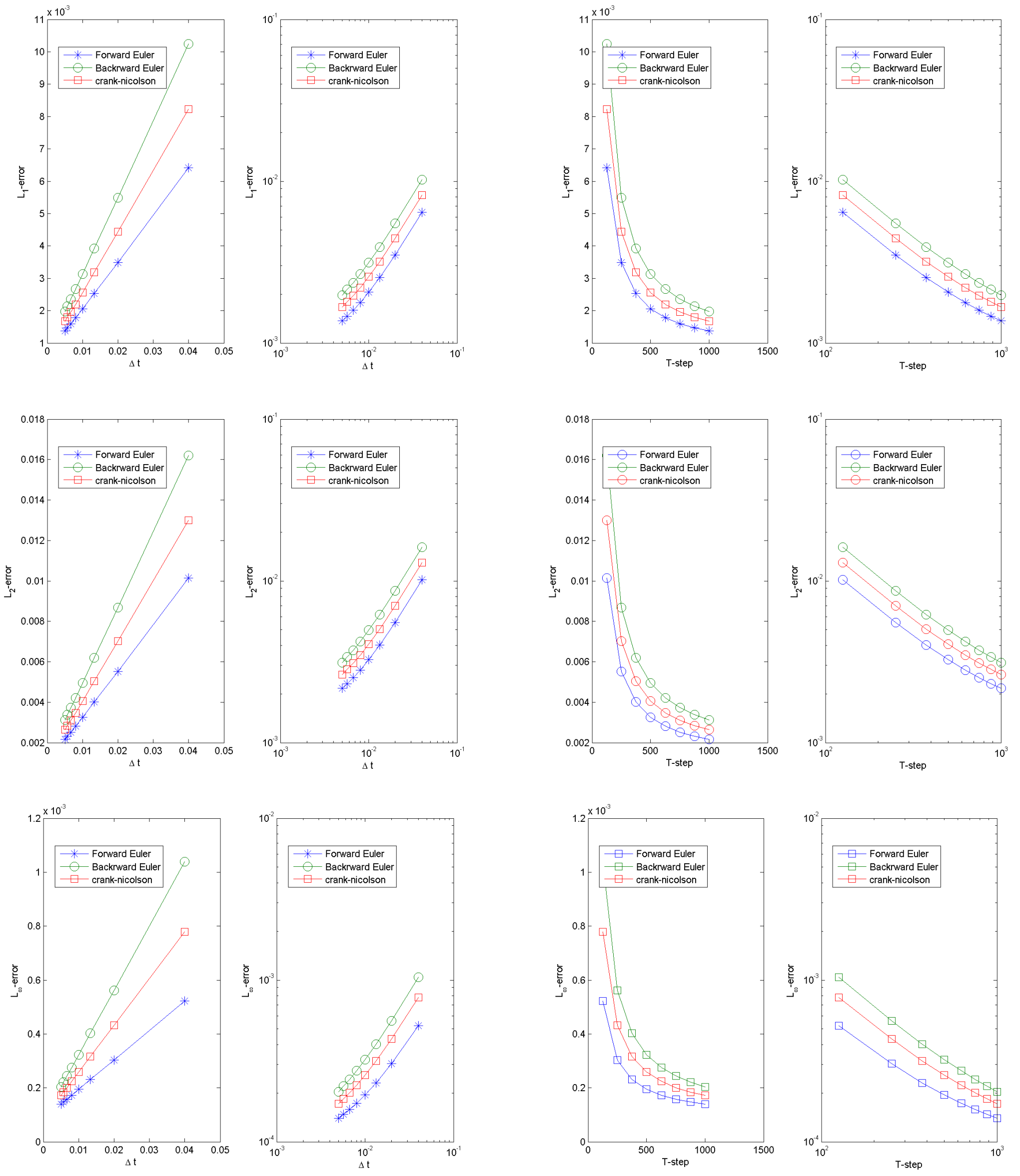

Figure 2.6: From upper-left to bottom-right, comparison of numerical error at time $T = 5$ in L_1, L_2 and L_∞-norm, obtained by explicit Forward Euler scheme, implicit Backward Euler Scheme and implicit-explicit Crank-Nicolson schemes with the space and time step, $\Delta x = 0.4$ and $\Delta t = 4 \times 10^{-2}$, ..., $\Delta t = 5 \times 10^{-3}$ for the equation 2.3.1, T-step represents the number of time steps.

this region, we start refine mesh size Δx, from 0.5 and end up refining at $\Delta x = 0.05$. The accuracy results in Table2.4, Table2.5 and Table2.6 are given for the error between numerical and Gaussian solution for the equation 2.3.1 with diffusive parameter $D = 1$ for the forward Euler scheme, backward Euler scheme and Crank-Nicolson scheme,

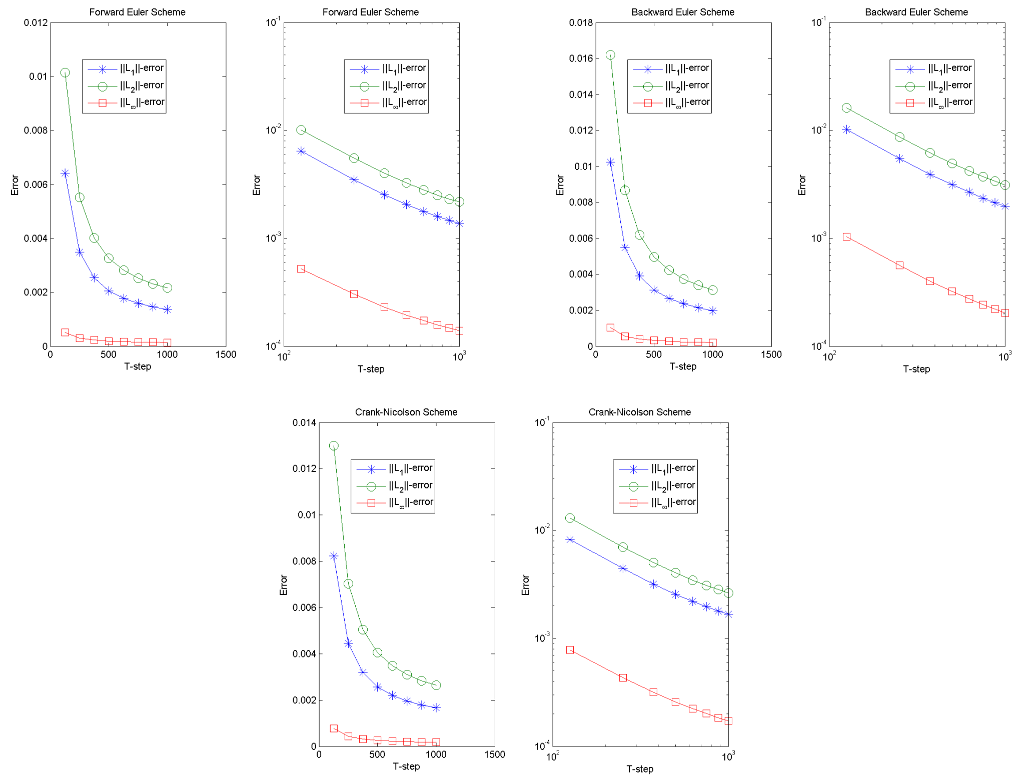

Figure 2.7: Comparison of numerical error at time $T = 5$ in L_1, L_2 and L_∞-norm, obtained by explicit Forward Euler scheme (upper-left), implicit Backward Euler Scheme (upper-right) and implicit-explicit Crank-Nicolson schemes (bottom) with the space and time step, $\Delta x = 0.4$ and $\Delta t = 4 \times 10^{-2}$, . . ., $\Delta t = 5 \times 10^{-3}$ for the equation 2.3.1, T-step represents the number of time steps.

where we consider space, $x \in [0, 40]$ and time, $t \in [0, 5]$. It shows that the implicit backward Euler scheme is better than all others scheme. And comparison results are plotted in Figure 2.8.

we deserve our error in each grid point is a scalar value for any time steps $\Delta t > 0$, to avoid the time steps order concern we take our time steps by CFL condition, i.e: $\Delta t = 0.5 \times \Delta x^2$ and simulate from initial time $t_0 = 0.1$ to the final time $T = 5$. Figure 2.8 represents for the maximum space step $\Delta x = 0.5$ gives much bigger error than the space steps $\Delta x = 0.01$ in L_1, L_2 and L_∞ norm by the formula of E^n. Figure 3.1 represents that, all schemes gives satisfy the boundedness property in L_∞ sense, but from Figure 2.8 we conclude that, fully implicit backward Euler scheme is a best choice to simulate the equation 2.3.1.

Δx	Number of Cells (N)	L_1 – error	L_2 – error	L_∞ – error
5.0000e-01	8.0000e+01	4.0329e-02	5.7034e-02	7.6371e-03
2.5000e-01	1.6000e+02	4.0452e-03	8.0905e-03	5.7729e-04
1.6667e-01	2.4000e+02	1.7874e-03	4.3783e-03	2.5483e-04
1.2500e-01	3.2000e+02	1.0035e-03	2.8384e-03	1.4301e-04
1.0000e-01	4.0000e+02	6.4188e-04	2.0298e-03	9.1424e-05
8.3333e-02	4.8000e+02	4.4555e-04	1.5434e-03	6.3452e-05
7.1429e-02	5.6000e+02	3.2722e-04	1.2243e-03	4.6601e-05
6.2500e-02	6.4000e+02	2.5049e-04	1.0019e-03	3.5671e-05

Table 2.4: Estimated error for the diffusion equation 2.3.1, numerical simulation has been done by using Forward Euler scheme from initial time $t_0 = 0.1$ to the final time $T = 5$, step sizes $\Delta x = \Delta t$ and $D = 1$

Δx	Number of Cells (N)	L_1 – error	L_2 – error	L_∞ – error
5.0000e-01	8.0000e+01	7.8707e-03	1.1131e-02	6.5597e-04
2.5000e-01	1.6000e+02	1.9078e-03	3.8156e-03	1.5411e-04
1.6667e-01	2.4000e+02	8.4336e-04	2.0658e-03	6.8016e-05
1.2500e-01	3.2000e+02	4.7352e-04	1.3393e-03	3.8118e-05
1.0000e-01	4.0000e+02	3.0280e-04	9.5753e-04	2.4342e-05
8.3333e-02	4.8000e+02	2.1018e-04	7.2809e-04	1.6898e-05
7.1429e-02	5.6000e+02	1.5438e-04	5.7763e-04	1.2408e-05
6.2500e-02	6.4000e+02	1.1817e-04	4.7270e-04	9.4949e-06

Table 2.5: Estimated error for the diffusion equation 2.3.1, numerical simulation has been done by using Backward Euler scheme from initial time $t_0 = 0.1$ to the final time $T = 5$, step sizes $\Delta x = \Delta t$ and $D = 1$

Δx	Number of Cells (N)	L_1 – error	L_2 – error	L_∞ – error
5.0000e-01	8.0000e+01	1.0425e-02	1.4743e-02	1.2371e-03
2.5000e-01	1.6000e+02	2.5318e-03	5.0636e-03	2.9380e-04
1.6667e-01	2.4000e+02	1.1194e-03	2.7421e-03	1.2936e-04
1.2500e-01	3.2000e+02	6.2850e-04	1.7777e-03	7.2528e-05
1.0000e-01	4.0000e+02	4.0187e-04	1.2708e-03	4.6348e-05
8.3333e-02	4.8000e+02	2.7894e-04	9.6626e-04	3.2160e-05
7.1429e-02	5.6000e+02	2.0487e-04	7.6655e-04	2.3616e-05
6.2500e-02	6.4000e+02	1.5682e-04	6.2728e-04	1.8075e-05

Table 2.6: Estimated error for the diffusion equation 2.3.1, numerical simulation has been done by using Crank-Nicolson scheme from initial time $t_0 = 0.1$ to the final time $T = 5$, step sizes $\Delta x = \Delta t$ and $D = 1$

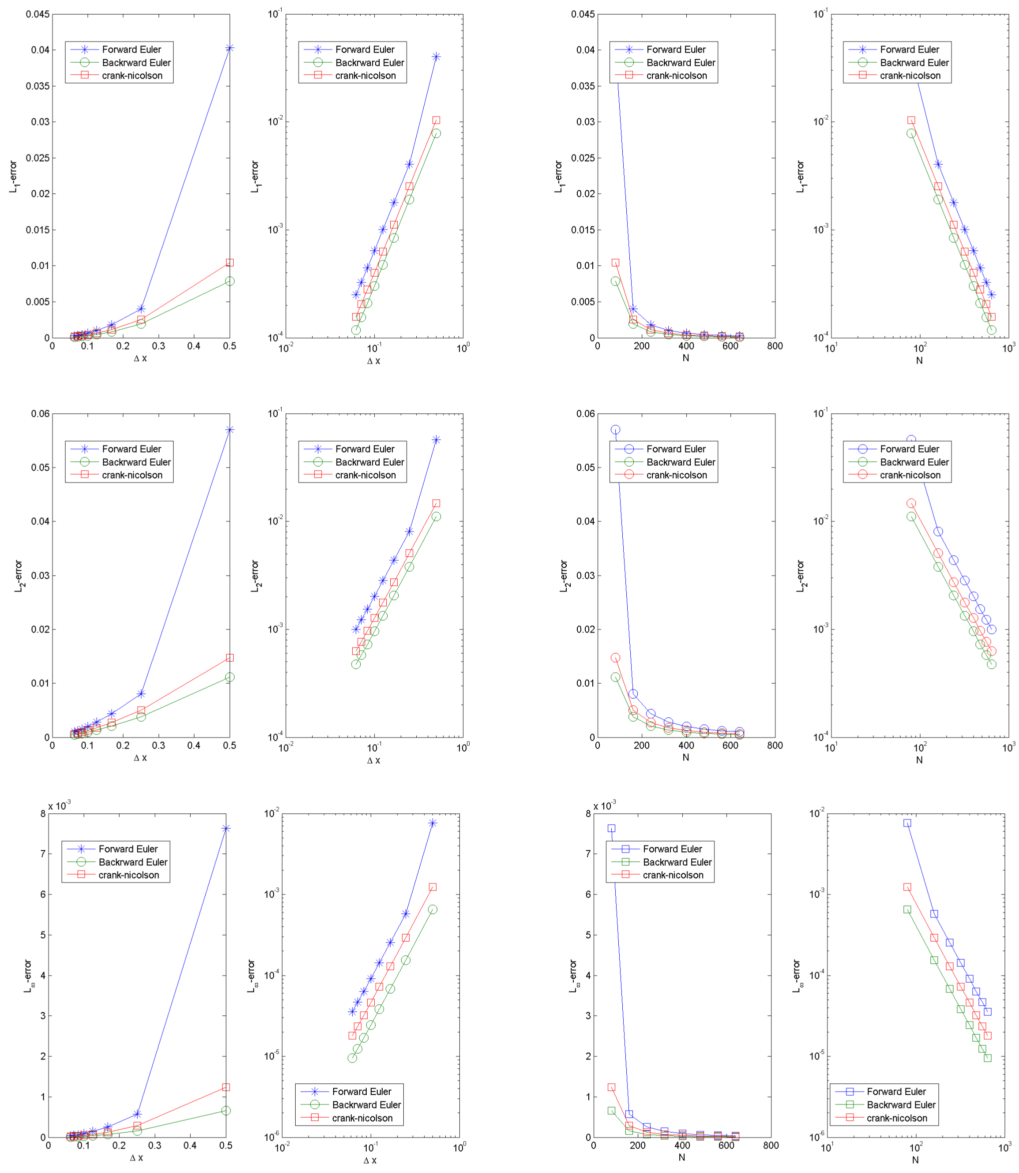

Figure 2.8: From upper-left to bottom-right, comparison of numerical error at time $T = 5$ in L_1, L_2 and L_∞-norm, obtained by Forward Euler scheme, Backward Euler Scheme and Crank-Nicolson scheme with the space and time step, $\Delta x = \Delta t$ for the equation 2.3.1, N-represents the number of cells.

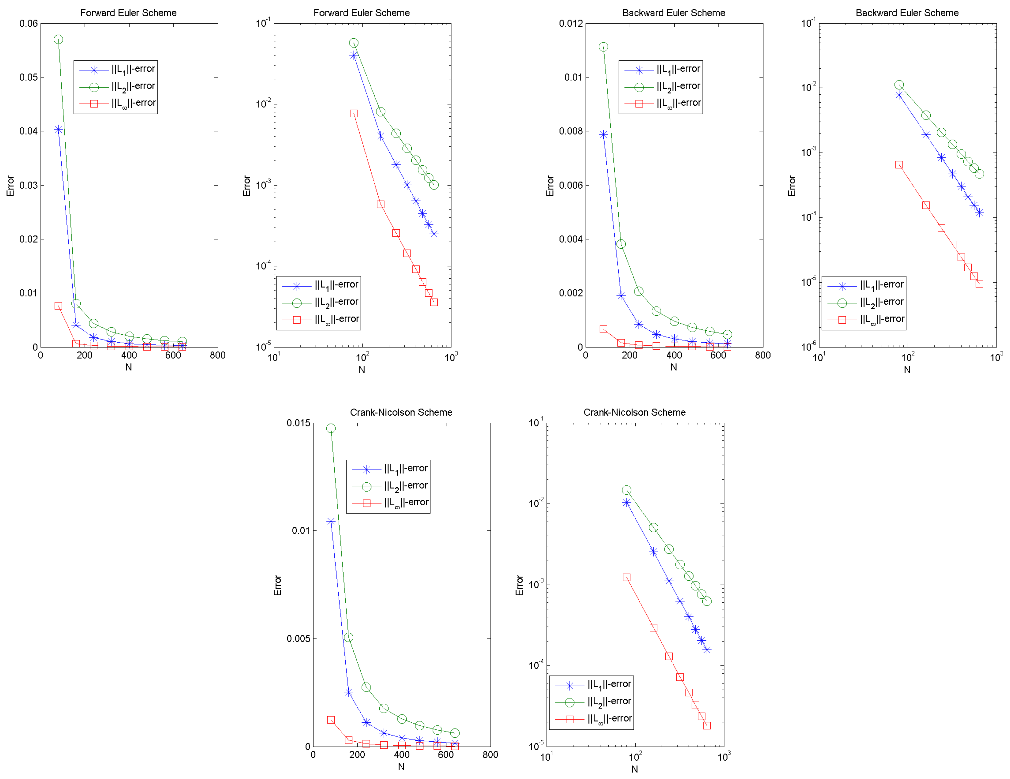

Figure 2.9: From upper-left to bottom-right, comparison of numerical error at time $T = 5$ in L_1, L_2 and L_∞-norm, obtained by Forward Euler scheme (Upper=left), Backward Euler Scheme (Upper-right) and Crank-Nicolson scheme (bottom) with the space and time step, $\Delta x = \Delta t$ for the equation 2.3.1, N-represents the number of cells.

CHAPTER 3

NUMERICAL SIMULATION OF POROUS MEDIA EQUATION

The aim of this chapter is to study numerical approximation of the nonlinear diffusion equation

$$\rho_t = \Delta(\rho^\gamma), \quad \gamma > 1, \tag{3.0.1}$$

and usually it is called the porous medium equation (PME), with due attention paid to its closest relatives. The default settings are:$\rho = \rho(x,t)$ is a non-negative scalar function of space $x \in \mathbb{R}^d$ and time $t \in \mathbb{R}$, the space dimension is $d \geq 1$, and γ is a constant larger than 1 and Δ represents the Laplacian form. The equation can be posed for all $x \in \mathbb{R}^d$ and $0 < t < \infty$, and then initial conditions are needed to determine the solutions; but it is quite often posed, especially in practical problems, in a bounded sub-domain $\Omega \in \mathbb{R}^d$ for $0 < t < T$, and then determination of a unique solution asks for boundary conditions as well as initial conditions.

This equation is one of the simplest examples of a nonlinear evolution equation of parabolic type. It appears in the description of different natural phenomena, and its theory and properties depart strongly from the heat equation, $u_t = \Delta u$, its most famous relative. Hence the interest of its analytical and numerical study is an important task for both for the pure mathematicians and the applied scientists.

3.1 Self-similar solution of PME

For the equation $(3.0.1)$, a solution of self-similar form can be recovered in a similar way as for the linear diffusion equation. We therefore look for a solution of the form in one dimension,

$$\rho(x,t) = t^{-\beta}\hat{\rho}(xt^{-\alpha}), \quad x \in \mathbb{R}, \quad t > 0 \tag{3.1.1}$$

to the one-dimensional PME

$$\rho_t = P(\rho)_{xx}, \tag{3.1.2}$$

where, $P(\rho) = \rho^\gamma, \quad \gamma > 1$.

First of all, the conservation of the total mass easily implies

$$\int_{\mathbb{R}} \rho(x,t)dx = t^{-\beta} \int_{\mathbb{R}} \hat{\rho}(xt^{-\alpha})dx = t^{-\beta+\alpha} \int_{\mathbb{R}} \hat{\rho}(\zeta)d\zeta$$

and therefore we have, $\beta = \alpha$.
In the new variables $\hat{\rho}$ and $\zeta = xt^{-\alpha}$, we obtain,

$$\alpha t^{-\alpha-1}(\zeta\hat{\rho})_\zeta + t^{-\alpha(\gamma+2)}(\zeta\hat{\rho^\gamma})_{\zeta\zeta} = 0,$$

which requires the choice

$$\alpha = \frac{1}{\gamma+1},$$

in order to get rid of the time variable. The above can be written as

$$\partial_x(\hat{\rho}\partial_x(\frac{\gamma}{\gamma-1}\hat{\rho^{\gamma-1}} + \frac{\zeta^2}{2(\gamma+1)})) = 0. \tag{3.1.3}$$

In the domain where $\hat{\rho} > 0$, we can impose

$$\hat{\rho^{\gamma-1}}(\zeta) = (C - \frac{\gamma-1}{2\gamma(\gamma+1)}\zeta^2),$$

where, C is any arbitrary constant. Clearly, the above gives problems in case, $\hat{\rho}$ becomes negative. Since, $\hat{\rho} = 0$ on an interval solves the equation for $\hat{\rho}$, we can introduce the following solution

$$\hat{\rho}(\zeta) = [C - \frac{\gamma-1}{2\gamma(\gamma+1)}\zeta^2]_+^{\frac{1}{\gamma-1}},$$

which in the original variables reads

$$\rho(x,t) = B(x,t) = t^{\frac{-1}{\gamma+1}}[C - \frac{\gamma-1}{2\gamma(\gamma+1)}\frac{x^2}{t^{2/\gamma+1}}]_+^{\frac{1}{\gamma-1}}, \tag{3.1.4}$$

where, $\rho_+ = max(\rho, 0)$ and equation (3.1.4) is called Barenblatt solution. Such a solution has compact support on the interval

$$-t^{\frac{-1}{\gamma+1}}\sqrt{\frac{2\gamma C(\gamma+1)}{\gamma-1}} \leq x \leq t^{\frac{-1}{\gamma+1}}\sqrt{\frac{2\gamma C(\gamma+1)}{\gamma-1}},$$

which grows as t grows. Moreover, it can be easily proven that $B(.;0)$ is a multiple of the Dirac delta distribution, exactly as it is the case for the Gaussian solution of the linear diffusion equation. Analyzing the differences with the linear diffusion case, we clearly see that the Barenblatt solution is not smooth, opposite to the Gaussian solution (G) to the linear diffusion equation, which is C^∞. More precisely, the Barenblatt

solution has possible lack of smoothness at the boundary of its support, where it eventually features discontinuities in a space derivative of a certain order depending on the adiabatic exponent γ. At any other points, B is C^{∞}. A key difference between G and B is that the support of G becomes unbounded immediately after $t = 0$ (in finite speed of propagation) whereas the support of B is compact for all times, it travels with finite speed.
Figure: 3.1 represents the equation $(3.1.2)$ for the different exponent $\gamma = 1, 2, 4, 6, 8$. For $\gamma = 1$, equation $(3.1.2)$ represents just an one-dimensional diffusion equation, so, for this we take initial value as a Gaussian solution at time, $t = 0.05$ and time, $t = 10$ shows the result for final time for the Gaussian solution, but for the rest of exponents (γ), we take initial value as a Barenblatt solution at time, $t = 0.05$ and time, $t = 10$ shows the result for final time for the Barenblatt solution. For all γ, we take boundary condition as a homogenous Neumann boundary condition $\frac{\partial\rho}{\partial x} = 0$

3.2 Numerical Schemes

In this section we are going to present numerical schemes based on finite difference method. In particular, we consider two different kind of the implicit method (pure implicit scheme and Crank-Nicolson implicit-explicit scheme) on an uniform grid with the mesh size $\Delta x = h$ and time step $\Delta t = k$. In one dimension on an interval $[0, L] \times [0, T]$ we define nodes $x_i, i = 1; \dots; s$ such that $x_1 = 0$ and $x_s = L$. We denote by ρ_i^n an approximation of a function $\rho(x, t)$ at node x_i at time t^n, where, $t \in [0, T]$ and for any function $f(\rho)$ we denote $f(\rho_i^n) = f_i^n$ for simplicity. This notation is going to be used throughout the chapter. Since we know, in $\theta-$ scheme, where, $\theta = 1$, scheme is called backward Euler scheme which is fully implicit scheme and $\theta = 1/2$, scheme is called Crank-Nicolson scheme which is implicit-explicit scheme. We want to compare numerical error between theae two schemes and for this reason, first we are going to present $\theta-$ scheme for PME.

3.2.1 $\theta-$ scheme

Consider the PME equation $(3.1.2)$

$$\begin{aligned}
\rho_t &= P(\rho)_{xx}, \quad P(\rho) = \rho^{\gamma}, \quad x \in [0, L], \quad \gamma > 1 \\
(x, 0) &= \rho_0(x), \quad x \in [0, L] \\
\frac{\partial\rho(x, t)}{\partial x} &= 0, \quad (x, t) \in [0, L] \times [0, T]
\end{aligned} \tag{3.2.1}$$

We have to notice that in this case the presence of nonlinear diffusion complicated the scheme. The system of linear equations in the previous case is replaced by nonlinear one. It has to be solved at each time iteration, which increases computational cost. We are going to use Newton method. It requires calculation of a Jacobian for a

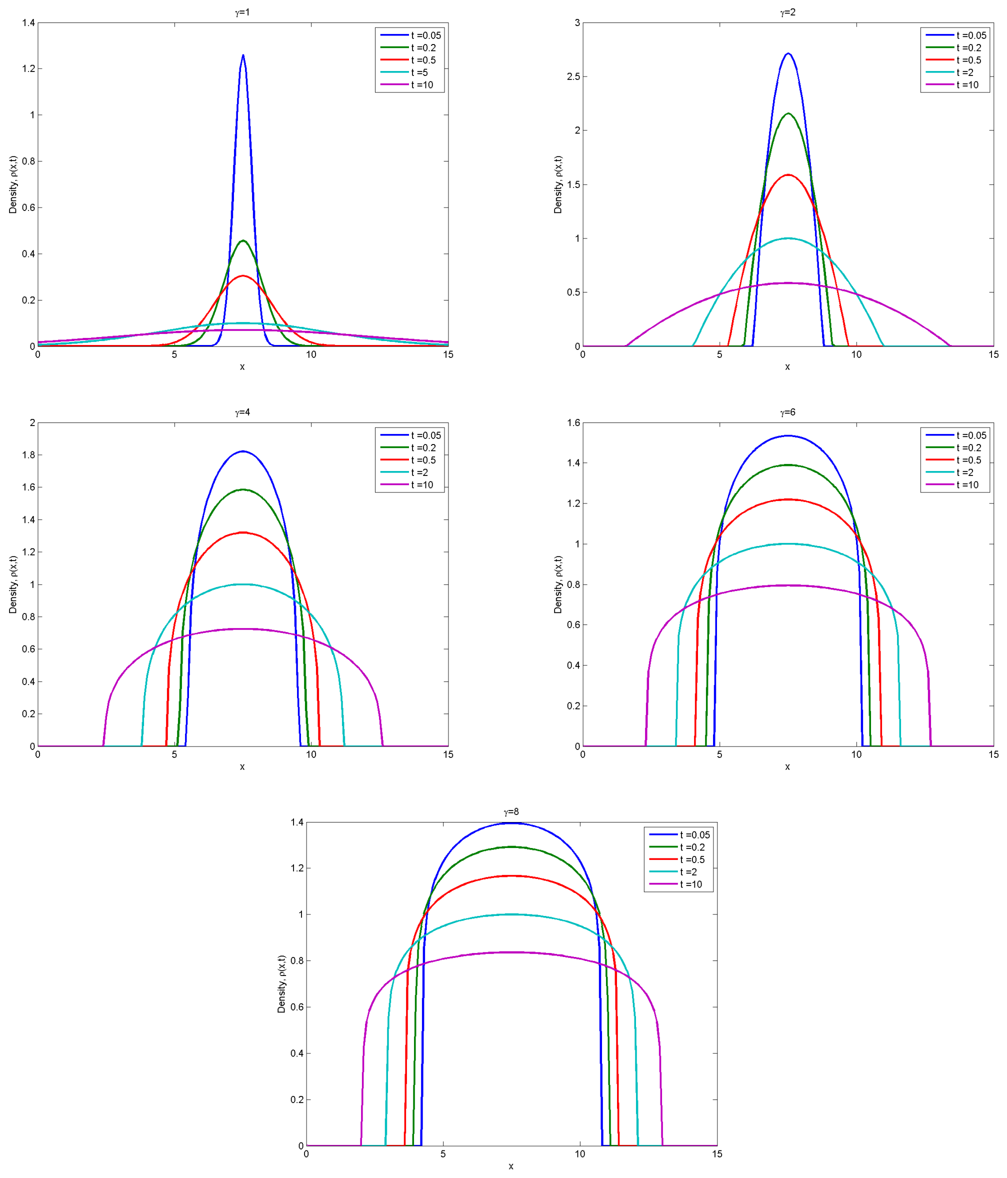

Figure 3.1: Comparison between Gaussian solution(when $\gamma = 1$) and Barenblatt solution (when $\gamma = 2, 4, 6, 8$) in time with $\Delta x = \Delta t = 0.1$ and $C = 1$. When the parameter γ increases, the Barenblatt solution tends to vary more slowly than Gaussian Solution inside its support, and it tends to be steeper near the interface of the support.

$s-2$-dimensional vector not only at each time iteration but also at each iteration inside Newton method until reaching the desired accuracy. To construct the scheme for (3.2.1) we need at first space discretization of the diffusion terms. We are going to use conservative, second order, centered discretizations. The approximation for diffusion is given at node x_i by Using a forward difference in time and a central difference in space for the equation $(3.2.1)$,

$$P(\rho)_{xx}|_{x_i} \approx \frac{1}{h^2}(P(\rho_{i-1}) - 2P(\rho_i) + P(\rho_{i+1})). \tag{3.2.2}$$

Introducing vectors $W \in \mathbb{R}^s$ such that

$$W_i(\rho) = \frac{1}{h^2}(P(\rho_{i-1}) - 2P(\rho_i) + P(\rho_{i+1}))$$

$\theta -$ scheme takes the form,

$$\rho_i^{n+1} = \rho_i^n + \theta k W^{n+1} + (1-\theta)kW^n\,. \tag{3.2.3}$$

The above results in a nonlinear system of simultaneous equations, As mentioned before the presence of non linear stiff terms requires at each time step the solution of a system of nonlinear equations. Let us define for every vector $w \in \mathbb{R}^{s-2}$ a function

$$F(w) := w - \theta k W^{n+1} - f(\rho^n), \tag{3.2.4}$$

where,

$$f(\rho^n) := \rho^n + (1-\theta)kW^n. \tag{3.2.5}$$

which corresponds to finding ρ^{n+1} at internal nodes, we use Newton's method initialized with $w = \rho^n$ and with tolerance Δx. In the case of Neumann boundary conditions, $\frac{\partial\rho(x,t)}{\partial x} = 0$, we assume, J_F is $(s-2)\times(s-2)$ matrix, which can be decomposed in the following way

$$J_F = \begin{pmatrix} J_{1,1} & \cdots & \cdots & \cdots & \cdots & \cdots & J_{1,s-2} \\ \vdots & & & & & & \vdots \\ \vdots & & & J^{\text{int}} & & & \vdots \\ \vdots & & & & & & \vdots \\ J_{s-2,1} & \cdots & \cdots & \cdots & \cdots & \cdots & J_{s-2,s-2} \end{pmatrix}, \tag{3.2.6}$$

where, $J^{\text{int}} \in M_{(s-4)\times(s-2)}$ and by J_1, J_{s-2} we denote $(s-2)$-dimensional vectors being the first and last row of the matrix J_F . The elements of the internal part J^{int} equal

$$\begin{aligned} J^{\text{int}}_{i,i-1} &= -\theta\lambda_x P'(\rho_{i-1}) \\ J^{\text{int}}_{i,i} &= 1 + 2\theta\lambda_x P'(\rho_i) \\ J^{\text{int}}_{i,i+1} &= -\theta\lambda_x P'(\rho_{i+1}) \end{aligned}$$

The vectors J_1, J_{s-2} contain the information about the homogeneous Neumann boundary conditions.

3.3 Simulation

In the numerical point of view, solving non-linear PDE is based on iterative process. There are several methods can apply to solve non-linear PDE, such as, bisection method, line search technique, gradient search technique, Newton-Raphshon technique etc. For our simulation we choose Newton-Raphshon method. In Newton method initial guessing is an important task to have faster convergence that we have described in the first chapter. We study this case numerically for simple linear and non-linear case. First we consider a linear function

$$f(x) = 10x - 1$$

The root of this function is $1/10$. Figure 3.2 represents the importance of initial guess for the Newton-Rapshon method. For this function, we see, if we choose initial guess except $-4 < x_0 < 4$ with respect to tolerance $\epsilon = 1e - 15$, it takes one iteration. On the other hand if we choose initial guess in between $-4 \quad to \quad 4$ i.e, $-4 < x_0 < 4$, say, $x_0 = -3$, then it will take 0 iteration.

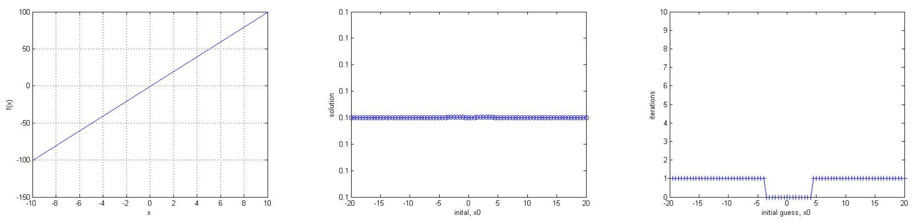

Figure 3.2: Graph of the function, $f(x) = 10x - 1$ (on the left), numerical solution (at middle) and expected iteration for the convergence with respect to the initial guess (on the right) for the Newton-Rapshon method

Now we consider a non-linear function

$$f(x) = x^2 - 2sin(x)$$

The roots of this function are $x \approx 0$ and $x \approx 1.40441$. table 3.1 represents the 6 numerical results and the number of iteration for the convergence to the root for the function, $f(x) = x^2 - 2sin(x)$.

Figure 3.3 represents the plot of the table 3.1, also we see, for this non-linear function, number of iteration iterations are varies on the basis of initial guess for both two roots. If x_0 is any number in (1,5) or in (0, 0.1), Newton-Rapshon method quickly takes us to the root.

After solving the PME using Newton method, we are interested now to check the error in each grid points for the exponent $\gamma > 1$, to get better accuracy first suppose, we know the true Barenblatt solution (B^n) at time $t = t_n$. Let E^n denote the error in the calculation with time and space, that is in each grid point, as computed using

initial guess, x_0	solution	iteration
0.1	$-1.1933e-20$	3
1	1.4044	5
4	1.4044	6
20	1.4044	8
-6	$-1.3629e-19$	6
-20	$-2.5665e-19$	8

Table 3.1: Numerical Results with Newton Rapshon method with respect to the tolerance, $\epsilon = 1e-15$

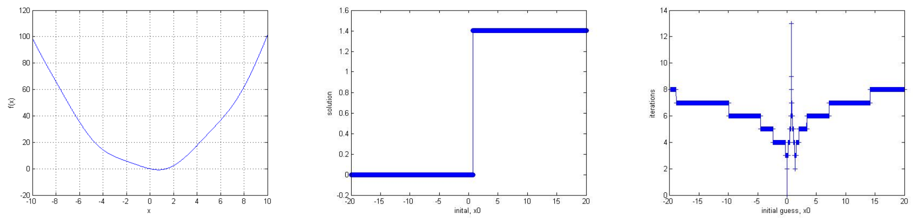

Figure 3.3: Graph of the function,$f(x) = x^2 - 2sin(x)$ (left), numerical solution (middle) and expected iteration for the convergence with respect to the initial guess (right) for the Newton-Rapshon method

the exact solution. We suppose that E^n is a scalar as same as chapter 2 that we did for linear diffusion, typically some norm of the error over the grid points, i.e., $E^n = \left\|\rho^{\text{numerical}} - \rho^{\text{exact}}\right\|$ where $\rho^{\text{numerical}}$ is the numerical solution vector and ρ^{exact} is the true solution evaluated at each grid point by using time steps and space steps. So, At each grid point there will be some error which is important to check and now we are going to check error on the basis of L_1, L_2 and L_∞ norm and we will observe the qualitative behavior of error for the exponent $\gamma > 1$ on each grid point with grid refinement to have better result.

We first present some numerical results to show the effectiveness of the $\theta-$ method. To do that, we begin our simulation for the Barenblatt solution of the PME (3.2.4), where the initial condition is taken as the Barenblatt solution at $t = 1$, and the boundary condition is $u_x(L,t) = 0, x \in [0, 25], \text{for}, t > 1$. We divide the computation domain into $N = 251$ uniform cells, and plot in Figure 3.4, the density profile for different exponents, $\gamma = 2, 3, 4, 5, 6$ using space step $\Delta x = 0.1$ for both schemes. All are compared with the Barenblatt solution. We see that both implicit schemes are able to capture the location of the front correctly without noticeable oscillations near the interface, but need deep insight to check the accuracy for the smooth part of those solutions and decide which one is better for the approximation.

For this region, we choose the parameter of PME $\gamma = 2$ and start refine mesh size

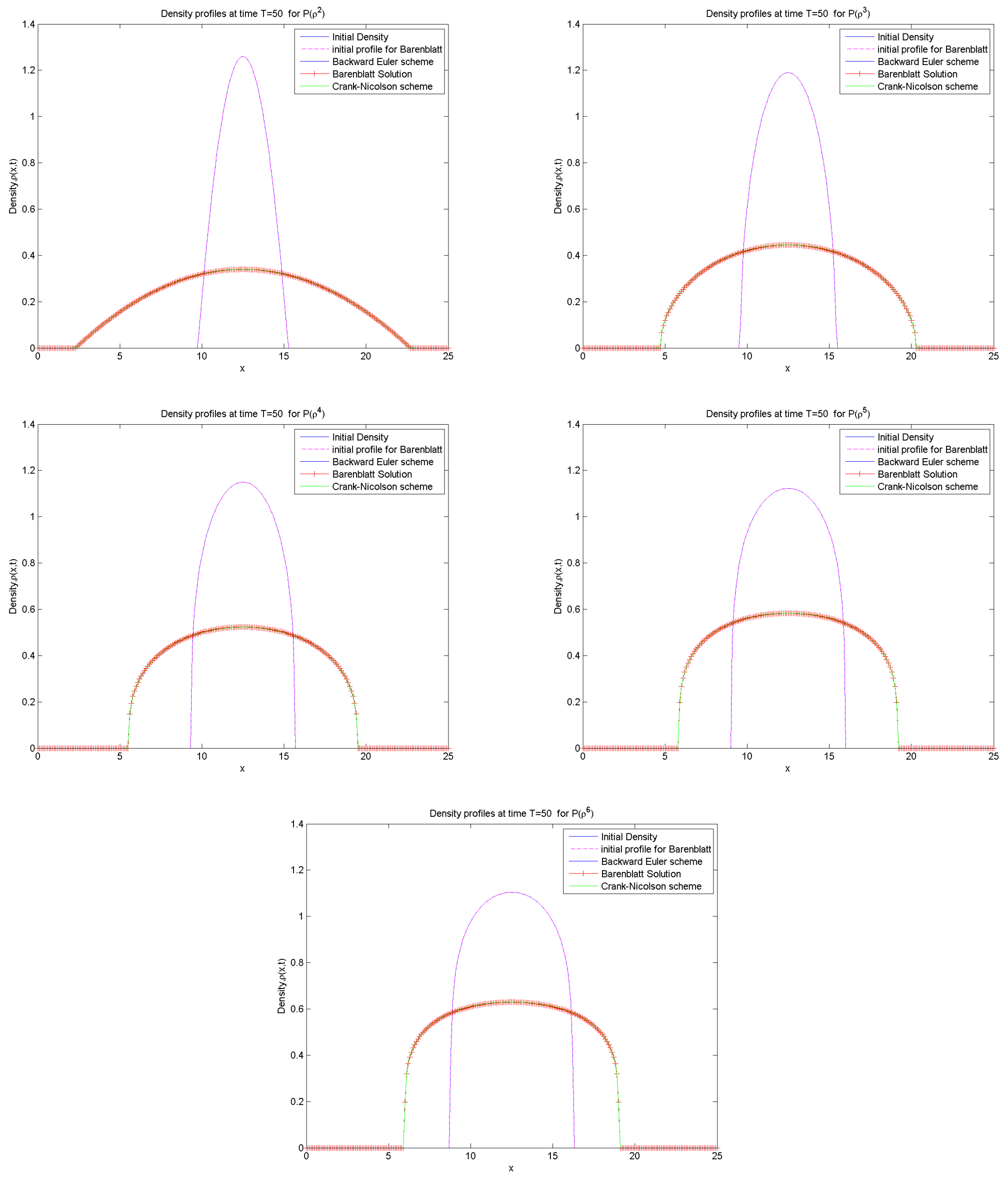

Figure 3.4: Comparison of density profiles at time $T = 50$ obtained by Backward Euler Scheme and Implicit-Explicit schemes with the space and time step, $\Delta x = \Delta t = 0.1$ for the diffusion functions $P(\rho^\gamma)$, where, $\gamma = 2, 3, 4, 5, 6$ from upper-left to bottom. The results are compared with the Barenblatt solution (+) with $C = 1$.

Δx, from 0.25 and end up refining at $\Delta x \approx 0.03$. The accuracy table 3.2 and 3.3 are given for the error between numerical and Barenblatt solution of PME $(3.1.4)$ with exponent parameter $\gamma = 2$ for the Backward Euler scheme and Crank-Nicolson scheme, where we consider space, $x \in [0, 15]$ and time, $t \in [0, 5]$. It shows that the implicit-explicit Crank scheme is better than Backward Eular scheme. And comparison results

Δx	Number of cells (N)	L_1 − error	L_2 − error	L_∞ − error
2.5000e-01	6.1000e+01	1.5416e-02	3.0831e-02	7.1334e-03
1.2500e-01	1.2100e+02	5.1063e-03	1.4443e-02	4.0416e-03
8.3333e-02	1.8100e+02	2.9964e-03	1.0380e-02	2.8521e-03
6.2500e-02	2.4100e+02	2.1197e-03	8.4787e-03	2.2205e-03
5.0000e-02	3.0100e+02	1.6323e-03	7.2997e-03	1.8279e-03
4.1667e-02	3.6100e+02	1.3230e-03	6.4814e-03	1.5598e-03
3.5714e-02	4.2100e+02	1.1119e-03	5.8838e-03	1.3444e-03
3.1250e-02	4.8100e+02	9.6106e-04	5.4366e-03	1.1631e-03

Table 3.2: Estimated error for the PME, numerical simulation has been done by using Backward Euler scheme from initial time $t_0 = 0.2$ to the final time $T = 5$, step sizes $\Delta t = \Delta x$, $C = 1$ for the Barenblatt solution.

Δx	Number of cells (N)	L_1 − error	L_2 − error	L_∞ − error
2.5000e-01	6.1000e+01	1.2723e-02	2.5446e-02	6.6311e-03
1.2500e-01	1.2100e+02	1.7997e-03	5.0902e-03	3.6438e-03
8.3333e-02	1.8100e+02	9.4468e-04	3.2725e-03	2.5251e-03
6.2500e-02	2.4100e+02	6.6255e-04	2.6502e-03	1.9409e-03
5.0000e-02	3.0100e+02	4.5138e-04	2.0186e-03	1.5822e-03
4.1667e-02	3.6100e+02	2.8306e-04	1.3867e-03	1.3395e-03
3.5714e-02	4.2100e+02	1.6583e-04	8.7747e-04	1.1441e-03
3.1250e-02	4.8100e+02	1.3369e-04	7.5625e-04	9.7886e-04

Table 3.3: Estimated error for the PME, numerical simulation has been done by using Crank-Nicolson scheme from initial time $t_0 = 0.2$ to the final time $T = 5$, step sizes $\Delta t = \Delta x$, $C = 1$ for the Barenblatt solution.

are plotted in Figure **??**

$\tilde{U}sing$,L_2 − norm we are going to measure error to know at final time response and see how our simulation is differ from exact solution at each final time. But, we also want the best fit to a specified numerical response in an L_∞ sense rather than the L_2 sense. This minimizes the maximum error of the approximation. If the L_∞ − norm of the error is small, then we are guaranteed that the approximation to our desired solution response will lie within the illustrated bounds.

As mention before, If the L_∞ −norm of the error is small, then we are guaranteed that the approximation to our desired solution response will lie within the illustrated bounds. Figure 3.7 represents the indivisual error for both schemes, where we can see, both schemes satisfy this condition, but Figure **??** represents that Cran-Nicolson scheme even more accurate than Backward Euler implicit scheme for PME.

We now pay more attention to the movement of the numerical interface, to check whether the Crank-Nicolson schemes has the ability to capture the true interface accurately. The position of the numerical interface is detected as follows: we scan the

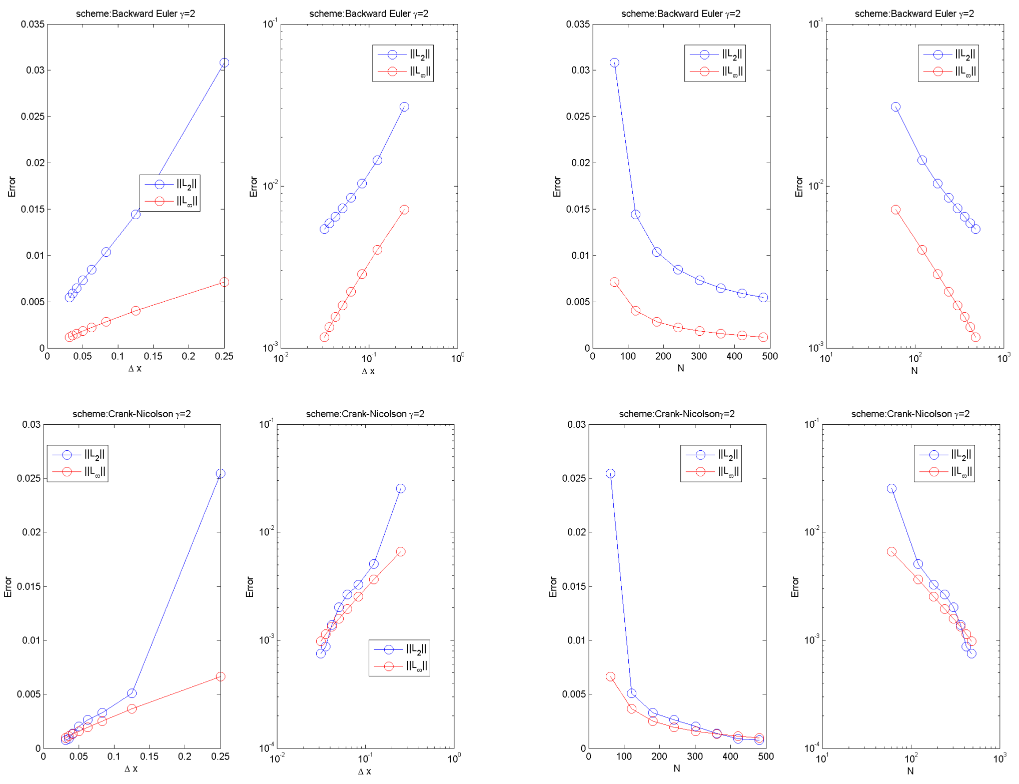

Figure 3.7: Comparison of density error at time $T = 5$ in L_2 and L_∞-norm, obtained by Backward Euler Scheme (upper) and implicit-explicit Crank-Nicolson schemes (bottom) with the space and time step, $\Delta x = \Delta t$ for $P(\rho^2)$, N-represents the number of cells.

numerical solution in the space x from 12.5 to 14, where $x \in [0, 15]$, and find the right endpoint of this cell as the numerical interface. We plot in Figure 3.8 the evolution of the numerical interface and result compare with the Barenblatt solution, for six different parameters $\gamma = 3, 4, 5, 6, 7 \text{and}, 8$, at the final time $T = 10$. Here the solid line is the position of the numerical interface, and the plus $(+)$ is the position of the exact interface evaluated by Barenblatt equation $(3.1.4)$, This figure verifies that the Crank-Nicolson schemes is very accurate at capturing the moving interface.

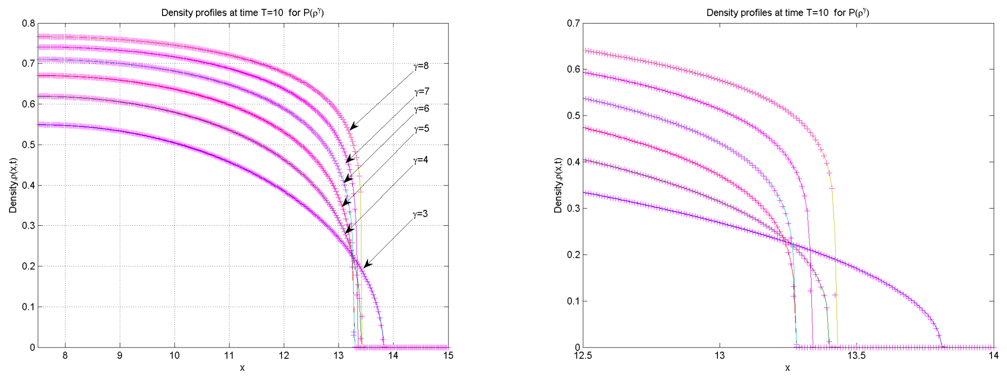

Figure 3.8: At final time $T = 10$, right movement of the numerical interface for $\gamma = 3, 4, 5, 6, 7, \text{and}, 8$ from initial time $t_0 = 0.2$ (left) and spatial scanning look of the left figure (right), numerical results are obtained by implicit-explicit Crank-Nicolson schemes with the space and time step, $\Delta x = \Delta t = 0.01$. Results are compared with Barenblatt solution (+)

APPENDIX A

MATLAB CODES

A.1 Mass execution

```
%***********************************************
% Solving total mass for linear diffusion equation in
% the case for Periodic, Dirichlet and homoginious
% Nuemann Boundary condition.
%***********************************************
clc;
clear;
L = 1;          % Space Lenth;
tbar =0.05;     % Initial time;
tfinal =10;     %final time to find the Number of time
    steps;
N = 100;        % Number of space steps
dx =L/N;        % Space Step
x = 0:dx:L;
D = 1;          % diffusion constant
dt = dx*dx/D;   % Time Step
theta =1;       % Parameter for the Theta scheme
B = dt/dx*dx;
x0 = L/2;       % Space interval
% Numer of mesh
n = round(length(x/dx)-1);
% Setting initial matrix for numerical solution
u = zeros(N+1,round((tfinal/dt)));
% Setting initial matrix for exact solution
U = zeros(N+1,round((tfinal/dt)));
% Setting initial condition for the Numerical Solution
for i=1:n+1
```

```
    if ((0.4<x(i))&&(x(i)<0.6))
         u(i,:)=5;
     else
         u(i,:)=0;
     end
end
%*********************************************
% Diffusion matrix;
aa = -theta*B;
bb = 1+2.*theta*B;
bbw =1-2.*theta*B;
cc = -theta*B;
aaa = diag(aa*ones(1,n),1);
bbb=diag(bb*ones(1,n+1),0);
bbr = diag(bbw*ones(1,n+1),0);
ccc = diag(cc*ones(1,n),-1);
MMl= aaa+bbb+ccc;
MMr= -aaa+bbr-ccc;
%****************************************************
%-------------Boundary Conditions (B.C)-------------------
%Nueman B.C;
% MMl(1,1)=1;
% MMl(1,2)=MMl(1,1);
% MMl(n+1,1)=1;
% MMl(n+1,n+1)=MMl(n+1,n);
% MMr(1,1)=1;
% MMr(1,2)=MMr(1,1);
% MMr(n+1,:)=1;
% MMr(n+1,n+1)=MMr(n+1,n);
%****************************************************
%Periodic B.C;
% MMl(1,1)=MMl(n+1,n+1);
% MMr(1,1)=MMr(n+1,n+1);
%***************************************************
%Dirichlet B.C;
% MMl(1,1)=0;
% MMl(n+1,n+1)=0;
% MMr(1,1)=0;
% MMr(n+1,n+1)=0;
%***************************************************
% Tracking schemefrom the value of theta;
switch theta
```

```
    case 0                                    % When theta =0;
        scheme=' explicit Euler';
    case 1                                    % When theta =1;
        scheme=' implicit Euler';
    case 1/2                                  % When theta =1/2;
        scheme=' Crank-Nicolson';
end
%**************************************************
for ii= 1:round(tfinal/dt)% Time Evaluation Loop;
% Numerical approximation;
    if theta==1/2
        u(:,ii+1)=MMl\MMr*u(:,ii); % Crank-Nicolson Scheme
            ;
    elseif theta ==1
        u(:,ii+1)=MMl\u(:,ii); % Backward Euler Scheme;
    end
   T = tbar+ii*dt; % Time interval for exact solution
% ---------------Exact solution---------------------------
   U(:,ii+1) = sqrt(tbar/(T))*exp((-(x-x0).^2)/(4*D*T));
%---------------------------------------------------------
%     u(1,1)=u(1,2); u(n,n+1)=u(n+1,n+1); % Nuemann B.C;
%     u(1,ii)=u(n+1,ii); % Periodic B.C;
%      u(1,ii)=0; u(n+1,ii)=0; % Dirichlet B.C;
end
ss =zeros(1,round((tfinal/dt))); %Initialize for the mass
    matrix ;
for j=1:round(tfinal/dt)
   ss(:,j) = sum(abs(u(:,j))*dx);
end
% Ploting Total Mass ;
figure(1);
grid on
plot(ss,'--','LineWidth',1)
ylim([0 2]);
xlabel('time');
ylabel('total mass');
clc;
```

A.2 Error execution for the linear DE equation

By time step refining

****

```
%*****************************************************
% Executing error in different time steps
% for the linear diffusion equation
% in the case of ||L_1||, ||L_2|| and ||L_{\infty}||
% using periodic boundary condition. Initial condition
% has been taken as Gaussian solution at the initial
% time $t_0=0.05$.
%*****************************************************
clc;
clear;
L = 40;              % Maximum lenth
tbar =0.05;          % Initial time
tfinal =10;          % Final time to find the Number of
    time steps
n =100;              % Number of space steps (fixed)
dx =L/n;             % Space step
x = 0:dx:L;          % Grid points in space
D =1;                % diffusion constan
theta =[0 1 1/2];    % Parameter for Theta scheme
CFL =1;
SX =CFL*dx.^2/(D); % Initial mesh
% Refining time steps
dt = [SX/2 SX/4 SX/6 SX/8 SX/10 SX/12 SX/14 SX/16];
% Loop start for executing simulation for each time step
for k=1:8
% Loop for taking three different parameter for the Theta
     scheme
for kk =1:3
B =D*dt(k)./(dx*dx);
x0 = L/2;            % Space interval
%number of time steps for each refinened time step
TT(k)=round((tfinal/dt(k)+1));
% Setting initial matrix for numerical solution
u = zeros(n+1,round((tfinal/dt(k)+1)));
% Setting initial matrix for exact solution
U = zeros(n+1,round((tfinal/dt(k)+1)));
% Loop start for initial condition (I.C)
for i =1:n+1
```

```
u(i,1) = sqrt(1/(4*pi*D*tbar))*exp(-((x(i)-x0).^2)/(4*D*
    tbar));
U(i,1) = sqrt(1/(4*pi*D*tbar))*exp(-((x(i)-x0).^2)/(4*D*
    tbar));
end % end of the loop for I.C

u(1,:)=u(n+1,:);    %Set initial B.C
%************set diffusion matrix*********************
aa = -theta(kk)*B;
ax = B;
bb = (1)+2.*theta(kk)*B;
bx = (1)-2*B;
bbw =(1)-2.*theta(kk)*B;
cc = -theta(kk)*B;
cx = B;
aaa = diag(aa*ones(1,n),1);
bbb=diag(bb*ones(1,n+1),0);
bbr = diag(bbw*ones(1,n+1),0);
ccc = diag(cc*ones(1,n),-1);
aax = diag(ax*ones(1,n),1);
bbx=diag(bx*ones(1,n+1),0);
ccx = diag(cx*ones(1,n),-1);
MMl= aaa+bbb+ccc;
MMr= -aaa+bbr-ccc;
MMx = aax+bbx+ccx;
% set B.C in diffusion matrix
MMl(1,1)=MMl(n,n+1);
MMr(1,1)=MMr(n,n+1);
MMx(1,1)=MMx(n,n+1);
%**************end of diffusion matrix***********
% Tracking scheme from the parameter of theta
switch theta(kk)
case 0 % when theta =0;
scheme=' explicit Euler';
case 1 % when theta =1;
scheme=' implicit Euler';
case 1/2 % when theta =1/2;
scheme=' Crank-Nicolson';
end % end of switch loop
%*****************************************
%***********Numerical solution*****************
for ii= 1:round(tfinal/dt(k))      % Time Evaluation Loop
```

```
    if theta(kk)==1/2
        u(:,ii+1)=MMl\(MMr*u(:,ii));% execute Crank-
           Nicolson schme
    elseif theta(kk) ==1
         u(:,ii+1)=MMl\(u(:,ii));   % execute backward
            Euler schme
    else
         u(:,ii+1)=MMx*(u(:,ii));   % execute forward
            Euler schme
    end
%------------------------------------------------------------
    T(ii) = tbar+(ii)*dt(k); % Time interval for exact
       solution
         % Exact Gaussian solution
    U(:,ii+1) = sqrt(1/(4*pi*D*T(ii)))*exp(-((x-x0).^2)
       /(4*D*T(ii)));
         % Set B.C for
    u(1,ii+1) =u(n+1,ii+1);
    U(1,ii+1) =U(n+1,ii+1);
end

%************Error execution***********************
errorl1 =0; %set initial error for L_1 norm
errorl21=0; %set initial error for L_2 norm
difference=zeros(n,1);
% Loop for error execution
for j=1:n-1
% solution difference in each grid point between exact
   and nuerical solution
difference1 =abs(U(j,round(tfinal/dt(k)))...
-u(j,round(tfinal/dt(k))));
difference(j) =abs(U(j,round(tfinal/dt(k)))...
-u(j,round(tfinal/dt(k))));
errorl1= errorl1+sum(difference1*dx);% Error in L_1 norm
errorl21= errorl21 +...
sqrt(sum(difference1.^2*dx));        % Error in L_2 norm
errorl23 = max(difference);   % Error in L_infinity norm
end %end of loop for error execution

%**********saving all errors***************************
if kk==1 % Error for forward Euler scheme
pr1(k,:)=[dt(k) TT(k) errorl1 errorl21 errorl23];
```

```
elseif kk==2 % Error for backward Euler scheme
pr2(k,:)=[dt(k) TT(k) errorl1 errorl21 errorl23];
else % Error for Crank-Nicolson scheme
pr3(k,:)=[dt(k) TT(k) errorl1 errorl21 errorl23];
end
end % end of the loop for tracking difference scheme
%**************************************************
% Plot the result by using saved data
% for example plot the result for L_2 error with
% respect to each time step step, plot by the following
   code
figure (1)
grid on;
plot(pr2(:,1),pr2(:,4),'-*','MarkerSize',10);
xlabel('\Delta x')
ylabel('L_2-error')
legend('Backward Euler','Location','best')
% to see its log-log scale, plot by the following code
figure (2)
loglog(pr2(:,1),pr2(:,4),'-*','MarkerSize',10);
xlabel('\Delta x')
ylabel('L_2-error')
legend('Backward Euler','Location','best')
end
clc;
```

By space step refining
****

```
%***************************************************
% Executing error in different space steps
% for the linear diffusion equation
% in the case of ||L_1||, ||L_2|| and ||L_{\infty}||
% using periodic boundary condition. Initial condition
% has been taken as Gaussian solution at the initial
% time $t_0=0.1$.
%***************************************************
clc;
clear all;
L = 40;             % Maximum lenth
tbar =0.1;          % Initial time
tfinal =5;          % Final time to find the
                    % number of time steps
```

```
D =1;                  % Diffusion constan
theta =[0 1 1/2];      % Parameter for the theta scheme
SX =1;                 % Initial space step
cfl=0.5;
dx =[SX/2 SX/4 SX/6 SX/8 SX/10 SX/12 SX/14 SX/16 SX/18...
     SX/20];           % Space refining
for k =1:8             % Simulation loop for refining grids
for kk=1:3             % Loop for the Theta scheme
x=0:dx(k):L;           % Space Lenth
dt(k)=cfl*dx(k)...     % Time steps
      *dx(k);
B = D*dt(k)...
    /(dx(k).^2);
x0 = L/2;              % Space interval
% Numer of mesh
n(k) = round(length(x/dx(k))-1);
% Setting initial matrix for numerical solution
u = zeros(n(k)+1,round((tfinal/dt(k)+1)));
% Setting initial matrix for exact solution
U = zeros(n(k)+1,round((tfinal/dt(k)+1)));
% *********Setting initial condition************
for i =1:n(k)+1
u(i,1) = sqrt(1/(4*pi*D*tbar))*exp(-((x(i)-x0).^2)/(4*D*
    tbar));
U(i,1) = sqrt(1/(4*pi*D*tbar))*exp(-((x(i)-x0).^2)/(4*D*
    tbar));
end
%********************************************

% *********Setting diffusion matrix*************
aa = -theta(kk)*B;
ax = B;
bb = 1+2.*theta(kk)*B;
bx = 1-2*B;
bbw =1-2.*theta(kk)*B;
cc = -theta(kk)*B;
cx = B;
aaa = diag(aa*ones(1,n(k)),1);
bbb=diag(bb*ones(1,n(k)+1),0);
bbr = diag(bbw*ones(1,n(k)+1),0);
ccc = diag(cc*ones(1,n(k)),-1);
aax = diag(ax*ones(1,n(k)),1);
```

```
bbx=diag(bx*ones(1,n(k)+1),0);
ccx = diag(cx*ones(1,n(k)),-1);
MMl= aaa+bbb+ccc;
MMr= -aaa+bbr-ccc;
MMx = aax+bbx+ccx;
MMr(1,1)=MMr(n(k),n(k)+1);
%*********************************************
% Tracking schemefrom the value of theta
switch theta(kk)
case 0
scheme=' Forward Euler';
case 1
scheme=' Backward Euler';
case 1/2
scheme=' Crank-Nicolson';
end

% *************Numerical approximation*********
for ii= 1:round(tfinal/dt(k)+1) % Time Loop
if theta(kk)==1/2 % Execute Crank-Nicolson scheme
u(:,ii+1)=MMl\(MMr*u(:,ii));
elseif theta(kk) ==1 % Execute backward Euler scheme
u(:,ii+1)=MMl\(u(:,ii));
else
u(:,ii+1)=MMx*(u(:,ii)); % Execute forward Euler scheme
end
T = tbar+(ii-1)*dt(k); % Time interval for exact solution
% Exact solution
U(:,ii+1) = sqrt(1/(4*pi*D*T))*exp(-((x-x0).^2)/(4*D*T));
% Setting B.C for numerical solution
u(1,ii+1) =u(n(k)+1,ii+1);
end
%**************************************************
% plot the result for each refining space step at the
    final time
figure(k)
plot(x,U(:,round(tfinal/dt(k)+1)),'r-',x,u(:,round(tfinal
    /dt(k)+1)),'LineWidth',1)     % Result comparision
    with exact solution
xlabel('x')
ylabel('Solution')
legend('exact solution','numerical solution')
```

```
title(strcat('\Delta x =', num2str(dx(k))))

%---------------------Error execution---------------
  errorl1  =0;                  % Initial L_1 error
  errorl21=0;                   % Initial L_2 error
  difference1=zeros(n(k),1);
for j=1:n(k)-1                  % loop for error execution
% Distance between exact and numerical soluion
difference1(j) =abs(U(j,round(tfinal/dt(k)))...
-u(j,round(tfinal/dt(k))));
errorl1=errorl1+sum(abs(U(j,round(tfinal/dt(k)))...
-u(j,round(tfinal/dt(k))))*dx(k));      % L_1 error
errorl23 =max(difference1);             % L_{\infty} error
errorl21 = errorl21+sqrt(sum(abs(U(j,...
round(tfinal/dt(k)))-u(j,...
round(tfinal/dt(k)))).^2*dx(k)));       % L_2 error

%*************************************************
%-----------Save all errors----------------------
if kk==1
pr1(k,:)=[dx(k) n(k) errorl1 errorl21 errorl23];
elseif kk==2
pr2(k,:)=[dx(k) n(k) errorl1 errorl21 errorl23];
else
pr3(k,:)=[dx(k) n(k) errorl1 errorl21 errorl23];
end
end
end
%************************************************
% Plot the result by using saved data
% for example plot the result for L_1 error with
% respect to space step, plot by the following code
figure (1)
grid on;
plot(pr1(:,1),pr1(:,3),'-*','MarkerSize',10);
xlabel('\Delta x')
ylabel('L_1-error')
legend('Forward Euler','Location','best')
% to see its log-log scale, plot by the following code
figure (2)
loglog(pr1(:,1),pr1(:,3),'-*','MarkerSize',10);
xlabel('\Delta x')
```

```
ylabel('L_1-error')
legend('Forward Euler','Location','best')

end
clc;
```

A.3 Error execution for PME

```
%*****************************************************
% Executing error in different space steps
% for the PME in the case of ||L_1||, ||L_2|| and ||L_{\
    infty}||
% using homogeneous Nuemann boundary condition. Initial
    condition
% has been taken as Barenblatt solution at the initial
% time $t_0=0.2$.
%
%*****************************************************
clc;
clear all;
L =15.0;                          % Lenth of Space Domain
t_initial=0.2;                    % Initial time
tfinal =5;                        % Final time to find time
    step
% Input to choose adiabatic exponents
gamma =input('value of \gamma:');
miu =L/2;                         % Space interval
SX =0.5;                          % Initial space step
% Refining space step
dx =[SX/2 SX/4 SX/6 SX/8 SX/10 SX/12 SX/14 SX/16 SX/18 SX
    /20];
theta=[1 1/2];                    % Parameter to choose
    scheme
% Loop start for executing simulation for each time step
for k=1:8
% Loop for taking three different parameter for the Theta
     scheme
for kk =1:2
x =0:dx(k):L;                     % Grid points in space
N(k)=round(length(x/dx(k)));      % Number of mesh
dt(k)=(0.1)*dx(k);                % Time step
```

```
tSteps(k) = round(tfinal/dt(k)); % Final time step
% set up initial mtrix for the numerical solution
rho=zeros(N(k),tSteps(k));
% set up initial matrix for the exact solution
rho_ex=zeros(N(k),tSteps(k));
%set up initial condition
for i=1:N(k)
if abs(x(i)-miu) <= t_initial.^(1/(gamma+1))*(sqrt(2*
    gamma*1*...
(gamma+1)/(gamma-1)));
rho(i,1)=max(t_initial.^(-1/(gamma+1))*(1-((gamma-1)/(2*
    gamma*...
(gamma+1)))*(x(i)-miu).^2/t_initial.^(2/(gamma+1)))...
                .^(1/(gamma-1)),0);
rho_ex(i,1)=rho(i,1);
end
end
% finite difference heta scheme
rx   =dt(k)/(dx(k)*dx(k));
p = theta(kk)*rx;
d = zeros(N(k),1);
J = zeros(N(k),3);
w = zeros(N(k),1);
%********Exact solution********************************
for n=1:tSteps(k)-1 % Time loop
t = t_initial+(n)*dt(k);
for i=1:N(k)          % Spatial loop
if abs(x(i)-miu) <=t.^(1/(gamma+1))*(sqrt(2*gamma*1*...
(gamma+1)/(gamma-1)));
rho_ex(i,n+1)=max(t.^(-1/(gamma+1))*(1-((gamma-1)...
/(2*gamma*...
(gamma+1)))*(x(i)-miu).^2/(t).^(2/(gamma+1)))...
^(1/(gamma-1)),0);
end
end
%**************************************************
% Tracking scheme
switch theta(kk)
case 0                 % When theta =0;
scheme=' Forward Euler scheme';
case 1                 % When theta =1;
scheme=' Backward Euler scheme';
```

```
case 1/2                % When theta =1/2;
scheme=' Crank−Nicolson scheme';
end
% at x = 0 take advantage of symmetry
d(1) = 2.0* rx * ( −rho(1,n)^gamma + rho(2,n)^gamma);
% Set up initial Jacobian matrix
J(1,1) = 1.0;
J(1,2) = 1.0 + 2*gamma * p * rho(1,n)^(gamma−1);
J(1,3) = −gamma * p * rho(2,n)^(gamma−1);

for i=2:N(k)−1  % spatial loop
d(i) = rx * ( rho(i−1,n)^gamma − 2.0*rho(i,n)^gamma +...
rho(i+1,n)^gamma );

% Set up Jacobian matrix
J(i,1) = −p *gamma*rho(i−1,n)^(gamma−1);
J(i,2) = 1.0 + 2*gamma * p * rho(i,n)^(gamma−1);
J(i,3) = −p *gamma*rho(i+1,n)^(gamma−1);
end
% at x = L take advantage of symmetry
d(N(k)) = 2.0*rx*(−rho(N(k),n)^gamma + rho(N(k)−1,n)^
    gamma);
% Set up end boundary condition of Jacobian matrix
J(N(k),1) = −gamma * p * rho(N(k)−1,n)^(gamma−1);
J(N(k),2) = 1.0 + 2*gamma * p * rho(N(k),n)^(gamma−1);
J(N(k),3) = 1.0;
% solve using Newton−Rapshon
w = Newton_solve(J, d);
% update free surface
rho(:,n+1) = rho(:,n) + w;
end
%−−−−−−−−−−−−−−−−−solution storage−−−−−−−−−−−−−−−−−−−−−−−−
if kk==1
rho_im =rho;
else
rho_cr =rho;
end

%−−−−−−−−−−−−Error execution−−−−−−−−−−−−−−−−−−−−−−−−−−−−
errorl1 =0;               % Initial L_1 error
errorl21=0;               % Initial L_2 error
difference1 =zeros(N(k),1);
```

```
for j=1:N(k)             % Spatial loop for error
difference1(j) =abs(rho(j,tSteps(k))...
-rho_ex(j,tSteps(k)));
difference =abs(rho(j,tSteps(k))...
-rho_ex(j,tSteps(k)));
% L_1 error
errorl1=errorl1+sum(difference*dx(k));
% L_2 error
errorl21 = errorl21+sqrt(difference.^2*dx(k));
% L_infinity error
errorl23 = max(difference1);
end
%---------------Saving errors------------------------------
if kk==1
pr1(k,:)=[dx(k) N(k) errorl1 errorl21 errorl23];
else
pr3(k,:)=[dx(k) N(k) errorl1 errorl21 errorl23];
end
end
%-----------------------------------------------------------
% Plot the result by using saved data
% for example plot the result for L_1 error with
% respect to space step, plot by the following code
figure (1)
grid on;
plot(pr1(:,1),pr1(:,3),'-*','MarkerSize',10);
xlabel('\Delta x')
ylabel('L_1-error')
legend('Forward Euler','Location','best')
% to see its log-log scale, plot by the following code
figure (2)
loglog(pr1(:,1),pr1(:,3),'-*','MarkerSize',10);
xlabel('\Delta x')
ylabel('L_1-error')
legend('Forward Euler','Location','best')
end
end
clc;
```

Bibliography

[1] M. Twarogowska. Numerical approximation and analysis of mathematical models arising in cells movement. *PhD thesis, University of L'Aquila, 2010/11*

[2] G. I. Barenblatt. Scaling, self-similarity and intermediate asymptotics. C*ambridge texts in applied mathematics, vol. 14. Cambridge University Press, 1996.*

[3] M. E. Gurtin, R. C. MacCainy, and E. A. Socolovsky. A coordinate transformation for the porous media equation that renders the free-boundary stationary. *Mathematics Research Center, 1983.*

[4] B. F. Knerr. The porous medium equation in one dimension. *Transactions of The American Mathmatical Society, 234(2):381-415, 1977.*

[5] W. L. Kath and D. S. Cohen. Waiting-time behavior in a nonlinear diffusion equation. *Stud. Appl. Math., 67(2):79-105, 1982.*

[6] J. L. Vázquez. The Porous Medium Equation Mathematical Theory. *CLARENDON PRESS . OXFORD :45-65, 2007.*

[7] R. J. LeVeque. Finite Difference Methods for Differential Equations. *Lecture notes for the course of A.Math 585-586, University of Washington, Version of September, 2005.*

[8] M. D. Francesco. Mathematical models in life sciences. *Lecture in Mathematics, University of L'Aquila, Chapter 8:88-99, 2010.*

[9] W. Hundsdorfer and J. Verwer. Numerical solution of time-dependent advection-diffusion-reaction equations, volume 33 of Springer Series in Computational Mathematics. Springer-Verlag, Berlin, 2003.

[10] A. Quarteroni and A. Valli. Numerical Approximation of Partial Differential Equations. *Springer Series in Computational Mathematics, 23:17-25 and 405-435, 1994.*

[11] K. W. Morton and D. F. Mayers. Numerical Solution of Partial Differential Equations. *Cambridge University Press, New York, 2nd edition:86-261, 2005.*

[12] Q. Zhang and Z. Wu. Numerical Simulation for Porous Medium Equation by Local Discontinuous Galerkin Finite Element Method. *Springer J.Sci. Comput. 38: 127-148, 2009.*

[13] F. Huang, R. Pan, and Z. Wang. Convergence to the Barenblatt solution for compressible euler equations with damping. *Archive for Rational Mechanics and Analysis, 200: 665-689, 2011. 10.1007/s 00205-010-0355-1.*

[14] O. A. Oleinik, A. S. Kalasinkov, and Yu-Lin C zou. The Cauchy problem and boundary problems for equations of the type of non-stationary filtration. *Izv. Akad. Nauk SSSR. Ser. Mat., 22:667-704, 1958.*

[15] L. C. Evans. Partial differential equations. *American Mathematical Society, Providence, RI, 1998.*

[16] R. Cavazzoni. Diffusive approximations for a class of nonlinear parabolic equations. *NoDEA Nonlinear Differential Equations Appl., 12(3):275-293, 2005.*

[17] R. Eymard, T. Gallouet, and R. Herbin. Finite volume methods. In Handbook of numerical analysis, Vol. VII. *Handb. Numer. Anal., VII, pages 713-1020. North-Holland, Amsterdam, 2000.*

[18] E. Maitre. Numerical analysis of nonlinear elliptic-parabolic equations. *M2AN Math. Model. Numer. Anal., 36(1):143-153, 2002.*

[19] J. W. Thomas. Numerical partial differential equations, volume 33 of Texts in Applied Mathematics. *Springer-Verlag, New York, 1999.*

[20] J. W. Thomas. Numerical partial differential equations: *finite difference methods, volume 22 of Texts in Applied Mathematics. Springer-Verlag, New York, 1995.*

[21] J. C. Strikwerda. Finite difference schemes and partial differential equations. *Wadsworth Brooks/Cole Advanced Books Software, Pacific Grove, CA, 1989.*

[22] M. Shashkov. Conservative finite-difference methods on general grids. *Symbolic and Numeric Computation Series. CRC*

[23] H.-G. Roos, M. Stynes, and L. Tobiska. Numerical methods for singularly perturbed differential equations. *Springer Series in Computational Mathematics. Springer-Verlag, volume 24, Berlin, 1996.*

[24] K. W. Morton. Numerical solution of convection-diffusion problems. *Applied Mathematics and Mathematical Computation. Chapman Hall, volume 12 London, 1996.*

[25] J. L. Graveleau and P. Jamet. A finite difference approach to some degenerate nonlinear parabolic equations. *SIAM J. Appl. Math. 20, 199-223, 1971.*

[26] E. D. Benedetto and D. Hoff. An interface tracking algorithm for the porous medium equation. *Transactions of The American Mathmatical Society, 284, 463-500, 1984.*