FACULTY OF SCIENCE AND TECHNOLOGY

MASTER THESIS

Study programme / specialisation: Master of Science /Computational Engineering	The spring semester, 2022 Open
Author: Rúnar Hoydal	Rúnar Hoydal (signature author)
Supervisor: Tore Halsne Flåtten	
Thesis title: Large Time Step schemes for Hyperbolic Conservation Laws in 2 Space Dimensions	
Credits (ECTS): 30	
Keywords: Hyperbolic Conservation Laws Large Time Step schemes Partial Differential Equations	Pages: 53 Stavanger, 15.6/2022 date/year

Acknowledgments

I wanted to thank Tore Flåtten for being my supervisor on this thesis. His guidance, enthusiasm and encouragement helped me stay motivated and left me with a feeling of playful wonder as we were working out different ways of tackling this thesis topic.

Abstract

Numerical solutions to partial differential equations (PDEs) will create varying amounts of error depending on different factors such as the numerical scheme and how fine the grid size is. In this thesis, we explored two different methods of discretizing a two dimensional domain in order to reduce this error. We compared the numerical error created from a rectangular- and a hexagonal-grid by using a specific type of PDE namely hyperbolic conservation laws. To this end, we used an explicit finite volume method not limited by the Courant-Friedrichs-Lewy (CFL) condition. This method is referred to as a Large Time Step (LTS) method and was proposed by LeVeque back in the 1980's. Since the two different grids are both what is called a structured grid, we were able to pair up the LTS method with a method called dimension splitting in order to both decrease the overall simulation time and increase the accuracy of the simulation. Combining those two methods we tested against two separate test cases for the flux. One was a linear flux while the other was a non-linear flux, namely a rotational flux around a center point. We found that for the linear flux, LTS was not strictly necessary in order to achieve an exact solution, however by utilizing LTS, an exact solution would be achievable for increasing amount of flux directions for both the rectangular and hexagonal grid. For the non-linear flux, no exact solution was found for any value of the Courant number (C), however we found that there exists an optimal value for C depending on the grid size which would minimize the numerical error. This value was clearly larger than 1 meaning that LTS is needed in order to achieve minimal numerical error on a rotational flux. The results showed that a rectangular grid was better suited in order to handle linear flux as it is able to capture a wider range of flux directions with an exact solution for all values of C. As for the non-linear flux, the two grids seems to be about even in terms of numerical accuracy.

Contents

1 Introduction

1.1 Scalar Conservation Law

A scalar conservation law in one space dimension is a first order partial differential equation (PDE) of the form

$$u_t + f(u)_x = 0. \tag{1.1}$$

Here $u = u(x,t)$ is called the conserved quantity, while f is the flux of that conserved quantity. The variable x denotes the one dimensional space position, while the variable t denotes the time. Equations of the form (1.1) often describe transport phenomena and are useful for modeling water- and air- flow which for instance can help guide the architectural design for a safer living and working environment. Another non-fluid example could be the density of cars on the road which (1.1) also applies to. Integrating (1.1) over a given interval $[a,b]$ you get

$$\int_a^b u_t(x,t)dx + \int_a^b f(u(x,t))_x dx = 0 \tag{1.2}$$

or

$$\int_a^b u_t(x,t)dx = f(u(a,t)) - f(u(b,t)) = [\text{inflow at } a] - [\text{inflow at } b]. \tag{1.3}$$

In other words, the total amount contained within an interval will only change due to the flow of u across the boundary points. Using the chain rule, (1.1) can be written in its quasilinear form

$$u_t + a(u)u_x = 0 \tag{1.4}$$

where $a(u)$ is the derivative of the flux f with respect to the space variable x. Given a continuous smooth solution, (1.1) and (1.4) are equivalent, however if there exists a discontinuity for u then u_x will be undefined at the discontinuity.

Example 1 (Bressan, 2013):

Let $\rho(x,t)$ be the density of cars on a highway, at the point x at time t. As an example, u may be the number of cars per kilometer. Assuming that ρ is continuous and the velocity of cars v depends only on their density

$$v = v(\rho), \quad \tfrac{dv}{d\rho} < 0 \ . \tag{1.5}$$

Given any two points a, b on the highway, the number of cars between a and b therefore varies according to (1.3)

$$\begin{array}{c} \int_a^b \rho_t(x,t)dx = [\text{inflow at } a] - [\text{inflow at } b] \\ = v(\rho(a,t))\rho(a,t) - v(\rho(b,t))\rho(b,t) = \int_a^b [v(\rho)\rho]_x dx \end{array} . \tag{1.6}$$

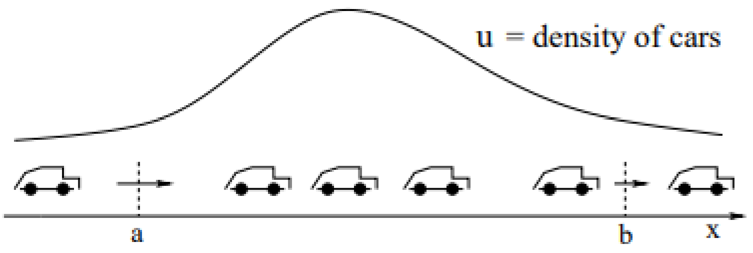


Figure 1.1: (Bressan, 2013)
The density of cars can be described by a conservation law

Since (1.6) holds for all points a, b this leads to the conservation law

$$\rho_t + [v(\rho)\rho]_x = 0 \tag{1.7}$$

where ρ is the conserved quantity and $f(\rho) = v(\rho)\rho$ is the flux function.

1.2 Hyperbolic Systems

In this thesis, we will focus on solving a single conservation law. However, if multiple conservation laws are needed in order to describe the system then $a(u)$ turns into a matrix called the Jacobian matrix. In that case in order for (1.4) to be strictly hyperbolic, the eigenvalues of $a(u)$ needs to be real and distinct. In the case of just a single conservation law, the value of $a(u)$ just need to be real in order for it to be strictly hyperbolic.

Example 2 (Dam Break Problem):

A classic problem which requires a system of conservation laws is the dam break problem, where one tries to model the water flow after the wall separating two different water heights are removed. One set of equations that can be used are called the Shallow Water Equations which in one dimension look like

$$h_t + (hu)_x = 0 \tag{1.8a}$$

$$(hu)_t + (hu^2 + \frac{1}{2}gh^2)_x = 0 \tag{1.8b}$$

where written in the quasilinear form now looks like

$$q_t + a(q)q_x = 0 \tag{1.9}$$

where $a(q)$ is the Jacobian matrix representing the velocity along the horizontal direction, and is written like

$$a(q) = \begin{bmatrix} 0 & 1 \\ -v^2 + gh & 2v \end{bmatrix} \tag{1.10}$$

For further details on the shallow water equations and the jump from (1.8) to (1.9) and the derivation of (1.10), see (LeVeque, 2002).

1.3 Previous Work

Finding new ways to solve hyperbolic conservation laws numerically or refining old methods is something which is still being worked on in order to improve simulation time and accuracy. In the 1980s LeVeque proposed the first LTS methods in a series of papers(LeVeque, 1982, 1985; LeVeque and mathematics, 1984) which could increase the timestep size past the usual CFL criterion and allow for faster simulations while using an explicit method of discretization. Since then, others have taken this idea and tried to figure out where one can reasonably apply the method. In one paper(Morales-Hernandez et al., 2012), it was used together with the upwind explicit scheme for the 1D shallow water equations. Others tried to apply the LTS method on the two dimensional Navier-Stokes equations and tested the results on transonic inviscid flows over the NACA0012 airfoil and ONERA M6 swept wing(Dong and Liu, 2018; Qian and Lee, 2011). Working in two dimensions, the choice of grid structure can also help increase the numerical accuracy because of how some discretization schemes leave a lot of false diffusion. This can be reduced by an increase in the number of grid points, however that in turn increases the computational time. Therefore, others have tried other ways of discretizing the grid in order to increase simulation accuracy such as using a triangular grid(Morales-Hernández et al., 2017) or hexagonal grid(Fabero et al., 2001; Karaa, 2006; Lee et al., 2014). Hamilton and Bilbao also did a comparison between the rectangular and hexagonal grid for the two dimensional wave equation(Hamilton and Bilbao, 2013). However, where previous work ends, and new work beings is in the combined use of a hexagonal grid together with the LTS method of discretization. We will test out whether or not there is any advantage in changing over to a hexagonal grid rather than a rectangular grid by comparing test results from both a linear and non-linear flux example.

1.4 Outline of Thesis

Exact solutions to PDEs are generally really hard to find or outright impossible due to the complexity of some PDEs. Here is where the numerical schemes help by discretizing the PDEs and iterating until an acceptable convergent solution has been reached. However, the numerical solution depends on the discretization and for a two dimensional system there exists infinitely many ways of discretizing the domain. One may categorize these discretization methods into two cases, structured and unstructured, where the name refers to the grid cells that make up the domain where a structured grid implies some symmetry between the cells, while an unstructured grid implies no symmetry. The numerical inaccuracy of a solution depends in part by the choice of discretization, where the most common discretization method is a structured grid composed of equally sized rectangles.

What this paper aims to look at is to compare the numerical results resulting from two different choices of discretization. The two grids under comparison are a rectangular- and hexagonal grid, and both are categorized as structured, which imply some symmetry that we can take advantage of. LTS-Roe scheme (Lindqvist et al., 2016) will be used and a comparison on the numerical accuracy given an increasing Courant number for a linear- and non- linear conservation law will be made. Given the structure of the hexagonal grid, our hypothesis is that the hexagonal grid will perform better in the case of a rotational flux due to it naturally forming more of a circular shape compared to the rectangular grid, and that the rectangular grid will perform better in the case of a linear flux for the same reason.

In section 2, we classify what hyperbolic conservation laws are and solution strategies to solve a Riemann problem in one dimension. In section 3, we go over different discretization schemes, among them is the Roe scheme, which we will be using in order to simulate the results shown in section 8. Section 4 we define the LTS method which will be used in order to decrease the computational time of the simulations. In section 5 and 6 we transition over to two dimensions and go over how the discretized schemes changes from one dimension to two dimensions as well as going over the two different grids we will be using. Lastly, in section 7 and 8, we go over the simulation setup and show the simulation results.

2 Hyperbolic Conservation Laws

Conservation laws are derived by considering conserved quantities of a system. These quantities can be anything from density (Bressan, 2013), momentum (Jenkins, 1992) or energy (McLachlan and Quispel, 2013) just to mention a few. The change of a conserved quantity u inside the control volume CV with surface ∂CV is given by

$$\frac{\partial}{\partial t}\int\limits_{CV} udV + \int\limits_{\partial CV} f(u)\hat{n}dA = 0. \tag{2.1}$$

Here, $f(u)$ is the flux of u, dV is the differential volume, dA is the differential area and $\hat{n}$ is the normal vector to the flux. Using the Gauss theorem, (2.1) can be changed to a partial differential equation of the form

$$\frac{\partial}{\partial t}\int\limits_{CV} udV + \int\limits_{CV} \nabla f(u)dV = 0 \tag{2.2}$$

and using the fundamental theorem of calculus (2.2) can be rewritten as a partial differential equation of the form

$$\frac{\partial u}{\partial t} + \nabla f(u) = 0. \tag{2.3}$$

Note however, that (2.3) assumes u to be a continuous function, so if there develops discontinuities (2.3) no longer applies.

2.1 Characteristics

Given a linear system of N conservation laws in one spatial dimension, the quasilinear form of (2.3) can be written as (1.4) where u is now a vector of size N and a is given by

$$a(u) \equiv \frac{\partial f(u)}{\partial u} \tag{2.4}$$

and is the Jacobian matrix of the system. We say that a system in the form (1.4) is hyperbolic if the matrix a is diagonalizable

$$a(u) = TDT^{-1} \tag{2.5}$$

where $D = \text{diag}(\lambda^1(u), \lambda^2(u), ..., \lambda^N(u))$ is a diagonal matrix consisting of the real eigenvalues of a. Multiplying (1.4) with T^{-1} from the left as well as substituting for $a(u)$ by using (2.5) we have

$$\frac{\partial v}{\partial t} + D\frac{\partial v}{\partial x} = 0 \tag{2.6}$$

where

$$dv = T^{-1}du \tag{2.7}$$

is the characteristic variable of the system. Since D is diagonalizable (2.6) can be rewritten as

$$\frac{\partial v^i}{\partial t} + \lambda^i(u)\frac{\partial v^i}{\partial x} = 0. \tag{2.8}$$

The implications of (2.8) is that the time evolution of each characteristic variable can be described independently with the propagating speeds being the corresponding eigenvalues.

2.2 Analytical Solution to Linear Systems

In order to understand better the spacial development of the system over time, the time derivative of (2.3) is taken. The first term on the left hand side of the equation equals

$$\frac{\partial}{\partial t}(u) = \frac{\partial}{\partial t}(u(x(t), t)) = u_x\frac{\partial x}{\partial t} + u_t\frac{\partial t}{\partial t}. \tag{2.9}$$

Using $\frac{\partial x}{\partial t} = f'(u(x(t), t))$ and noting that $\frac{\partial t}{\partial t} = 1$, (2.9) equals

$$\frac{\partial}{\partial t}(u) = u_x f'(u(x(t), t)) + u_t. \tag{2.10}$$

Assuming u to be smooth, (2.10) can be simplified further to

$$\frac{\partial}{\partial t}u(x(t), t) = f(u)_x + u_t \tag{2.11}$$

where the right hand side is by definition equal 0 from (2.3). From this result, one can conclude that

$$u(x(t), t) = \text{constant} = u(x(t = 0), t = 0) = u_0(x_0), \tag{2.12}$$

which is the analytical representation of (2.3) for linear conservation laws in 1 dimension. Using the substitution from earlier $\frac{\partial x}{\partial t} = f'(u(x(t), t))$, the path of

$x(t)$ can be determined by

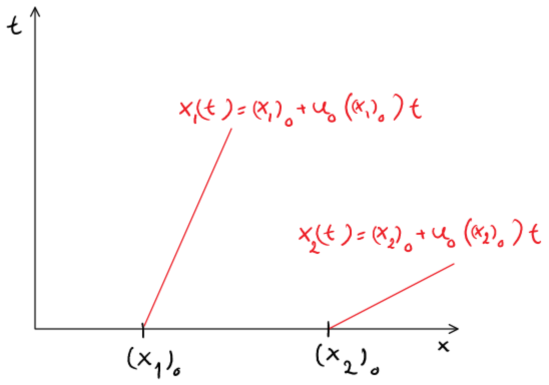
$$\frac{d}{dt}x(t) = f'(u(x(t), t)) = f'(u_0(x_0)) = \text{constant} \tag{2.13}$$

or

$$x(t) = x_0 + f'(u_0(x_0))t. \tag{2.14}$$

Equation (2.14) may indicate two different behaviors for the time evolution of the conserved variable. These two behaviors are called a rarefaction wave and a shock wave and will be discussed in section 2.3 and 2.4 respectively. In order to visualize the results of (2.14) and how it can be used to determine the properties of a rarefaction- and shock- wave, an example will be given for each case. In both examples, the Burgers' equation will be used to represent the flux, which is characterized by the flux function $f(u) = \frac{1}{2}u^2$.

2.3 Rarefaction wave



t
$x_1(t) = (x_1)_0 + u_0((x_1)_0)t$
$x_2(t) = (x_2)_0 + u_0((x_2)_0)t$
$(x_1)_0$
$(x_2)_0$
x

Figure 2.1:
Illustration of spreading characteristics

Suppose now that the initial condition is given in such a way that given two points x_1, x_2 where $x_1 < x_2$, the conserved variable at those two points is ordered such that $u_0(x_1) < u_0(x_2)$. Constructing a picture of $x_1(t), x_2(t)$ in x-t space for given $x_1 = 1$, $x_2 = 2$, $u_0(x_1) = 1$ and $u_0(x_2) = 2$. the space evolution of x_1 and x_2 is given by (2.14) and can be written as

$$x_1(t) = (x_1)_0 + u_0((x_1)_0)t \tag{2.15a}$$
$$x_2(t) = (x_2)_0 + u_0((x_2)_0)t \tag{2.15b}$$

or

$$x_1(t) = 1 + t \tag{2.16a}$$
$$x_2(t) = 2 + 2t \tag{2.16b}$$

From Figure 2.1, we see that in x-t space, the two equations never cross paths, which is the indication of a rarefaction wave.

2.4 Shock wave

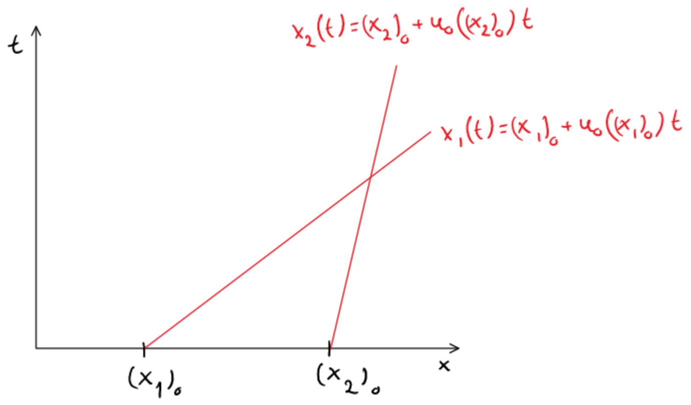

t
$x_2(t)=(x_2)_0+u_0((x_2)_0)t$
$x_1(t)=(x_1)_0+u_0((x_1)_0)t$
$(x_1)_0$
$(x_2)_0$
x

Figure 2.2:
Illustration of crossing characteristics

Imposing similar initial conditions, where we have two points x_1, x_2 where $x_1 < x_2$, however this time the initial conditions for the conserved variable at those two points are swapped $u_0(x_1) > u_0(x_2)$. Given $x_1 = 1$, $x_2 = 2$, $u_0(x_1) = 2$ and $u_0(x_2) = 1$. the space evolution of x_1 and x_2 is given by (2.15) and can be written as

$$x_1(t) = (x_1)_0 + u_0(x_1)t \tag{2.17a}$$

$$x_2(t) = (x_2)_0 + u_0(x_2)t \tag{2.17b}$$

or

$$x_1(t) = 1 + 2t \tag{2.18a}$$

$$x_2(t) = 2 + t \tag{2.18b}$$

From Figure 2.2, we see that in x-t space, the two equations cross paths, which is the indication of a shock wave. Note that in order to avoid ambiguity, the time evolution of the conserved variable is only valid until the time T_b (breaking time) where the characteristic equations intersect.

2.5 Riemann Problem

A Riemann problem is characterized by a discontinuity in the initial condition of the conserved variable.

$$u_t + f(u)_x = 0 \tag{2.19a}$$

$$u(x,0) = \begin{cases} u_l & x < 0 \\ u_r & x \geq 0 \end{cases} \tag{2.19b}$$

To understand how to solve Riemann problems, we will take a look at a couple of examples. A Riemann problem can be characterized as a piece-wise function where the solution depends on the relation between $f'(u_l)$ and $f'(u_r)$.

Case 1: $f'(u_l) > f'(u_r)$

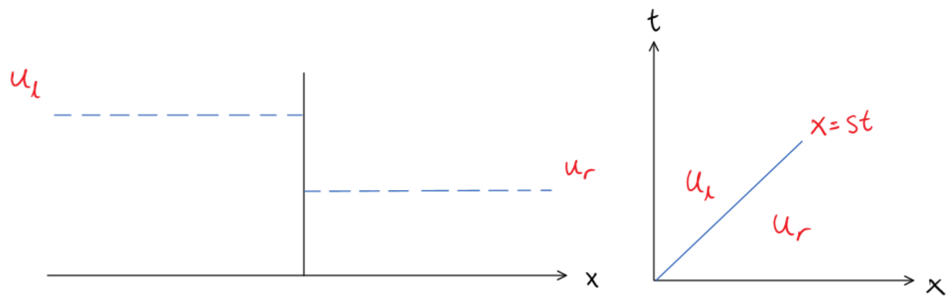


Figure 2.3:
(Left) Illustration of the discontinuity in space
(Right) Representation of the solution to x between u_l and u_r.

Notice that we now have a shock wave and the solution strategy is to find a discontinuous solution of (2.19a) where the jump satisfies the Rankine-Hugoniot condition (Lin, 2000)

$$s[u] = [f(u)]. \tag{2.20}$$

Here $s = \frac{dx}{dt}$ is the shock speed,

$$[u] = u(x(t)_r, t) - u(x(t)_l, t) \tag{2.21a}$$
$$[f(u)] = f(u(x(t)_r, t)) - f(u(x(t)_l, t)) \tag{2.21b}$$

stand for the jump of u and $f(u)$ across the shock. In the case of scalar conservation laws, (2.20) can be simplified to

$$s = \frac{f(u_l) - f(u_r)}{u_l - u_r} \tag{2.22}$$

turning the solution for the time evolution of the conserved variable equal to

$$u(x, 0) = \begin{cases} u_l & x < st \\ u_r & x \geq st \end{cases}. \tag{2.23}$$

Case 2: $f'(u_l) < f'(u_r)$

Now we have a rarefaction wave forming and the solution strategy is to find a continuous solution where u_l and u_r are connected by a rarefaction wave. Using

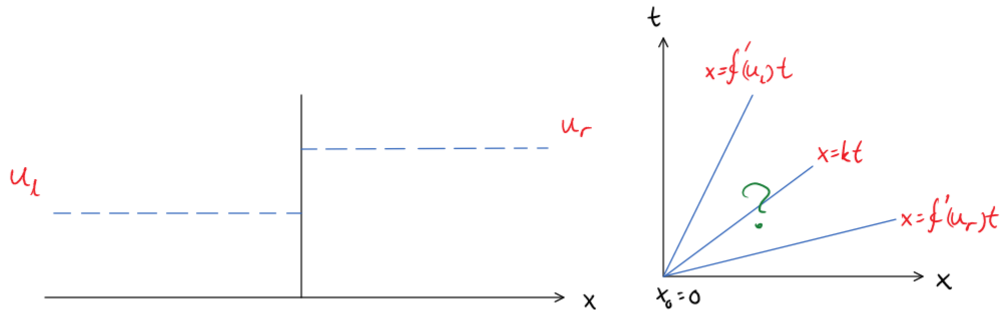


Figure 2.4:
Left) Illustration of the discontinuity in space
(Right) Representation of a speed k in between $f'(u_l)$ and $f'(u_r)$ which x may travel with.

the similarity solution

$$u(x,t) = v(\frac{x}{t}) \tag{2.24}$$

and

$$f'(u_l) < \frac{x}{t} < f'(u_r), \tag{2.25}$$

the time evolution for u can be written as

$$u(x,t) = \begin{cases} u_l & \frac{x}{t} \leq f'(u_l) \\ (f')^{-1}(\frac{x}{t}) & f'(u_l) < \frac{x}{t} < f'(u_r) \\ u_r & \frac{x}{t} \geq f'(u_r) \end{cases}. \tag{2.26}$$

Proof:

Using (2.24) the time derivative of u can be written in terms of v as follows

$$u_t = \frac{\partial}{\partial t} v(\frac{x}{t}) = v'(\frac{x}{t}) \frac{\partial}{\partial t}(\frac{x}{t}) = v'(\frac{x}{t})(\frac{-x}{t^2}). \tag{2.27}$$

Similarly for the space derivative we get

$$u_x = \frac{\partial}{\partial x} v(\frac{x}{t}) = v'(\frac{x}{t}) \frac{\partial}{\partial x}(\frac{x}{t}) = v'(\frac{x}{t})(\frac{1}{t}). \tag{2.28}$$

From here we have

$$f(u)_x = f'(u)u_x = f'(v)v'(\frac{x}{t})\frac{1}{t} \tag{2.29}$$

and substituting (2.27) and (2.29) into (2.19a) we have

$$-\tfrac{x}{t^2}v'(\tfrac{x}{t}) + f'(v)v'(\tfrac{x}{t})\tfrac{1}{t} = 0, \quad t > 0 \tag{2.30}$$

or

$$[f'(v) - \frac{x}{t}]v'(\frac{x}{t}) = 0. \tag{2.31}$$

Finding the zeros of (2.31), the two options are;

Case 1: $v'(\frac{x}{t}) = 0$

If the derivative of v is 0, then $v(\frac{x}{t}) =$ constant which is only possible if $u_l = u_r$, meaning that u_l and u_r are already connected.

Case 2: $f'(v) - \frac{x}{t} = 0$

Solving for v leaves us with

$$v(\frac{x}{t}) = (f')^{-1}(\frac{x}{t}) \tag{2.32}$$

under the constraint that $(f')^{-1}$ exists.

Given the combination of (2.14) and (2.32) the value of x at u_l and u_r can be evaluated as

$$x = f(u_l)t, \qquad \text{for } u_l \tag{2.33a}$$

$$x = f(u_r)t, \qquad \text{for } u_r \tag{2.33b}$$

and a formulation for $v(\frac{x}{t})$ can be expressed in terms of u. For the left hand side, $v(\frac{x}{t})$ turns into

$$v(\frac{x}{t}) = (f')^{-1}(\frac{x}{t}) = (f')^{-1}(f(u_l)) = u_l \tag{2.34}$$

similarly for the right hand side we get

$$v(\frac{x}{t}) = (f')^{-1}(\frac{x}{t}) = (f')^{-1}(f(u_r)) = u_r. \tag{2.35}$$

For the middle section where (2.25) applies the formulation of $v(\frac{x}{t})$ cannot be evaluated further without further knowledge of the flux function itself. Combining these three cases for $\frac{x}{t}$, in particular $\frac{x}{t} \leq f'(u_l)$, $f'(u_l) < \frac{x}{t} < f'(u_r)$ and $\frac{x}{t} \geq f'(u_r)$ we get exactly the formulation as required.

Since the solutions to the Riemann problem is essential to the computations and their results shown later on in the thesis, ending with a concrete example showcasing both the formation of a rarefaction wave and a shock wave will be done.

Example 1

Consider (2.19) where the flux function equals

$$f(u) = \frac{1}{2}u^2, \tag{2.36}$$

and the initial condition is given as

$$u(x, t = 0) = \begin{cases} 0 & x \leq 0 \\ 1 & x > 0 \end{cases}. \tag{2.37}$$

From (2.36), we have $f(u) = u$ and $(f')^{-1}(u) = u$. The first thing we need to find is if the behavior of the characteristics represent that of a shock- or rarefaction- wave. Evaluating $f'(u)$ at u_l and u_r we have

$$f'(u_l) = u_l = 0 \tag{2.38a}$$

$$f'(u_r) = u_r = 1 \tag{2.38b}$$

and $f'(u_l) < f'(u_r)$ which equate to spreading characteristics. Using (2.38) we get

$$u(x,t) = \begin{cases} 0 & \frac{x}{t} \leq 0 \\ \frac{x}{t} & 0 < \frac{x}{t} < 1 \\ 1 & \frac{x}{t} \geq 1 \end{cases} \tag{2.39}$$

substituting the corresponding values in. Figure 2.5 illustrates the time evolution of the conserved variable at times $t = \{1, 2, 4\}$.

Example 2

Consider the same setup as in example 1, only that the initial condition for u has changed where the values for u_l and u_r are swapped

$$u(x, t = 0) = \begin{cases} 1 & x \leq 0 \\ 0 & x > 0 \end{cases}. \tag{2.40}$$

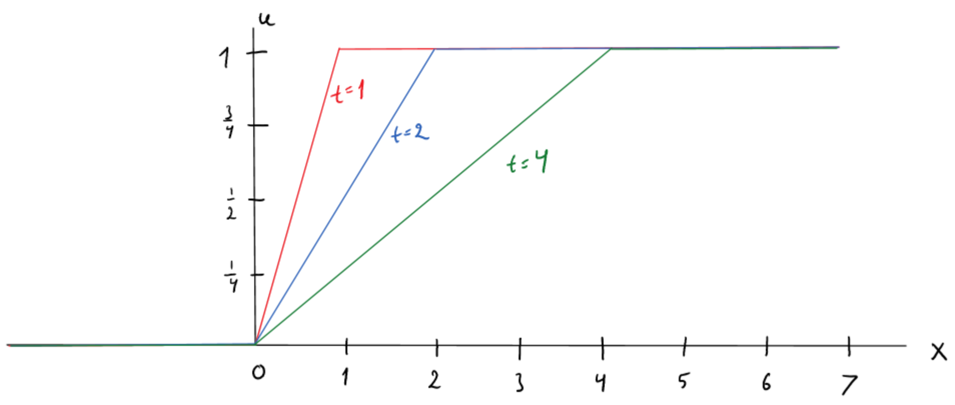


Figure 2.5:
An illustration of the time evolution of the conserved variable at $t = \{1, 2, 4\}$

Evaluating $f'(u)$ at u_l and u_r we have

$$\begin{aligned} f'(u_l) &= u_l = 1 \\ f'(u_r) &= u_r = 0 \end{aligned}, \tag{2.41}$$

and $f'(u_l) > f'(u_r)$ which equate to crossing characteristics. A discontinuous solution that satisfies the Rankine-Hugoniot condition (2.22) is given by

$$u(x,0) = \begin{cases} 1 & x < st \\ 0 & x \geq st \end{cases} \tag{2.42}$$

where

$$s = \frac{\frac{1}{2}1^2 - \frac{1}{2}0^2}{1-0} = \frac{1}{2}. \tag{2.43}$$

Figure 2.6, illustrates the time evolution of the conserved variable at times $t = \{1, 2, 4\}$.

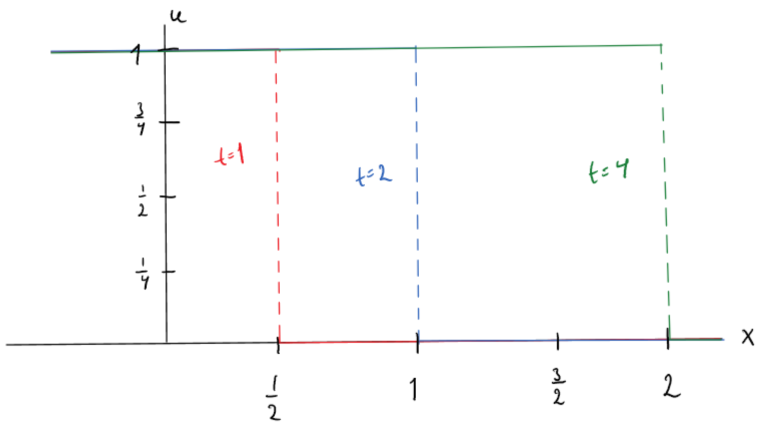


Figure 2.6:
An illustration of the time evolution of the conserved variable at $t = \{1, 2, 4\}$

3 Finite Volume Method

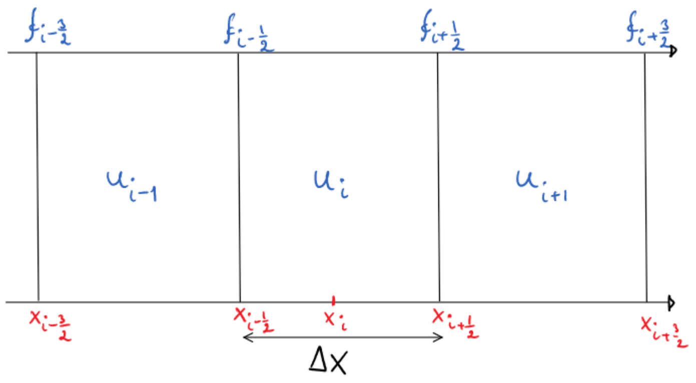


Figure 3.1:
Finite Volume Method

Consider the initial value problem

$$u_t + f(u)_x = 0 \tag{3.1a}$$

$$u(x,0) = u_0(x) \tag{3.1b}$$

In order to do numerical computations with (3.1a), it needs to be changed into a more appropriate form. In the case of a one dimensional domain of size L, one can divide the space into $N \in \mathbb{N}$ intervals that may or may not be evenly spaced. Each area defined by its boundaries can be called a grid cell and in this thesis we will use grid cells which are evenly spaced. We are interested in finding the time evolution of the average value of u in each cell during a small time step Δt. In order to do so we need to discretize each term in (1.1), the first term can be discretized in the following manner

$$u_t = \frac{\partial u}{\partial t} = \frac{u_i^{n+1} - u_i^n}{\Delta t} \tag{3.2}$$

where n indicated the current time step, and $n+1$ indicated the next time step given by $n+1 = t + dt$. However, for the second term there are two ways of handling the evaluation of the flux term called the explicit- and implicit-method. The implicit method is evaluating the flux at the new time step and is highly stable, but is more computationally heavy to solve, and the explicit method evaluates the flux at the current time step and is much simpler to solve, however is now also bound by a stability criterion. In this thesis, we will use the explicit method, meaning that the flux term in (1.1) can be discretized as follows.

$$f(u)_x = \frac{\partial f(u)}{\partial x} = \frac{f^n_{i+\frac{1}{2}} - f^n_{i-\frac{1}{2}}}{\Delta x}. \tag{3.3}$$

Substituting (3.2) and (3.3) into (3.1a) and isolating u_i^{n+1} on the left hand side leaves us with

$$u_i^{n+1} = u_i^n - \frac{\Delta t}{\Delta x}(f^n_{i+\frac{1}{2}} - f^n_{i-\frac{1}{2}}). \tag{3.4}$$

In comparison, the implicit version of (3.4) can be written like

$$u_i^{n+1} = u_i^n - \frac{\Delta t}{\Delta x}(f^{n+1}_{i+\frac{1}{2}} - f^{n+1}_{i-\frac{1}{2}}) \tag{3.5}$$

where further evaluating of (3.4) and (3.7) depends on how the flux at the cell edges are handled.

3.1 Upwind Scheme

When evaluating the flux term at the cell boundaries, what the upwind scheme does is to select one of the cells adjacent to it rather than some combination of the two. The motivation behind this choice is to imagine some kind of flow across the domain, for simplicity let us imagine that the flow is uniform and traveling towards the right in Figure 3.1. Then for the cell edge $f_{i+\frac{1}{2}}$, all information will be coming from the left side and so we use the value of f_i, likewise for flow in the opposite direction then we use the value of f_{i+1} instead. Given a uniform velocity a across the entire domain and that $f_{i+\frac{1}{2}} = au_{i+\frac{1}{2}}$ we are left with two distinct discretized forms depending on the flux direction

$$\begin{cases} u_i^{n+1} = u_i^n - a\frac{\Delta t}{\Delta x}(u^n_{i+1} - u^n_i) & \text{for } a \geq 0 \\ u_i^{n+1} = u_i^n - a\frac{\Delta t}{\Delta x}(u^n_i - u^n_{i-1}) & \text{for } a < 0 \end{cases}. \tag{3.6}$$

Note that the stability criterion for explicit methods uses a constant factor called the Courant number which is defined as

$$C = |a|\frac{\Delta t}{\Delta x}. \tag{3.7}$$

For a 3-point stencil, which is what (3.4) and (3.6) represent, the Courant number cannot exceed a value of 1(Courant et al., 1928)

$$C \leq 1. \tag{3.8}$$

This condition is named the Courant-Friedrich-Lewy condition and the intuitive explanation for this fact is that the Courant number represents how many cells the conserved variable travels through from one time step to the next. Since (3.4) and (3.6) only uses one neighboring cell on each end, the Courant number cannot exceed a value of 1 otherwise some information about the conserved variable would be lost, and the numerical solution would become unstable. In section 3.2 we will go over a variation of the upwind scheme, namely the Roe scheme.

3.2 Roe Scheme

For scalar equations, a 3- point scheme in the form (3.4) can be split into left-going and right-going waves as

$$f^n_{i+\frac{1}{2}} - f_{i-\frac{1}{2}} = a^+_{i-\frac{1}{2}}(u^n_i - u^n_{i-1}) + a^-_{i+\frac{1}{2}}(u^n_{i+1} - u^n_i) \tag{3.9}$$

commonly denoted as the Flux-Difference Splitting (FDS) formulation (LeVeque, 2002). Substituting (3.9) into (3.4) leaves

$$u^{n+1}_i = u^n_i - \frac{\Delta t}{\Delta x}(a^+_{i-\frac{1}{2}}(u^n_i - u^n_{i-1}) + a^-_{i+\frac{1}{2}}(u^n_{i+1} - u^n_i)) \tag{3.10}$$

which is essentially a more compact formulation of (3.6). For scalar equations, the formulation of $a^{\pm}$ is defined as

$$a^{\pm} = \pm\frac{\Delta x}{\Delta t}\max(0, \pm C^*), \tag{3.11}$$

where C^* is the local Courant number defined with the flux velocity at the cell edge which handles the separate cases automatically. The advantage of using the Roe scheme is that it linearizes a non-linear problem which simplifies the equation by transforming (3.4) into a set of linear equations. Note that by doing so, the Roe scheme is not able to distinguish between a rarefaction- or shock-wave formed by discontinuities and treats them all as if they were shock waves.

3.3 Total Variation Diminishing

If compact support or periodic boundary conditions are assumed, weak solution to the scalar initial value problem (3.1) possess the following monotonicity properties (Harten, 1997)

1. No new local extrema in x may be created
2. The value of a local minimum is non-decreasing, and the value of a local maximum is non-increasing.

As a consequence, we have that

$$\frac{d}{dt}\int\left|\frac{\partial u}{\partial x}\right| dx \leq 0. \tag{3.12}$$

In (Harten, 1997), Harten introduced a concept called Total Variation Non-Increasing (TVNI) which later is just referred to as Total Variation Diminishing (TVD) which respect (3.12) on the discrete level. The total variation at time step n is defined as

$$\mathrm{TV}^n = \sum_i \left|u_{i+1}^n - u_i^n\right| \tag{3.13}$$

and the numerical scheme (1.1) is said to be TVD if

$$\mathrm{TV}^{n+1} \leq \mathrm{TV}^n. \tag{3.14}$$

4 Large Time Step Methods

Large Time Step methods are explicit finite volume methods that are not limited by the standard CFL condition $C \leq 1$. In this chapter, we will highlight how one can handle the flux function in such a case when we relax the CFL criterion and expand the time step length allowed for each iteration.

4.1 Large Time Step Methods

We saw in Section 3.1 that the stability of a 3 point stencil was tied to the CFL condition. There we explored the idea where the Courant number represented how far the conserved variable could travel for each iteration. We noticed that the scheme (3.6) only took into account a single neighbor on each side of the cell in question, and so the Courant number could not exceed the value 1 as that would mean that some information would be lost, and the numerical scheme became unstable. However now we want to explore what happens if we do let the conserved variable travel further, what would need to be done in order to keep the scheme stable and are there any assumptions that need to be made in order for it to work.

Starting with the CFL condition, we would now like to increase the maximum Courant number before the scheme becomes unstable. As such, a value k will replace the number 1 in (3.8)

$$C \leq k \tag{4.1}$$

where $k \in \mathbb{N}$. In order for (4.1) to have a chance at stability, the definitions defined in (3.9), (3.10) and (3.11) needs to be updated. To begin with, the previously 3 point stencil definition can be written like a $(2(1)+1)$- point stencil and comparing (3.8) to (4.1) we need a $(2k+1)$ - point stencil. The extension to the FDS formulation can be written like (Jameson and Lax, 1986)

$$u_i^{n+1} = u_i^n - \frac{\Delta t}{\Delta x} \sum_{m=0}^{\infty} (a_{i-\frac{1}{2}-m}^{m+} \Delta_{i-\frac{1}{2}-m} + a_{i+\frac{1}{2}+m}^{m-} \Delta_{i+\frac{1}{2}+m}) \tag{4.2}$$

where

$$\Delta_{i-\frac{1}{2}} = u_i - u_{i-1} \tag{4.3}$$

and

$$a^{m\pm} = 0, \quad m \geq k \ . \tag{4.4}$$

Secondly, when limiting the Courant number to be less than one, we did not need to consider what would happen if multiple discontinuities interacted with each other. However, now that the conserved variable is allowed to travel further than one cell for each iteration, two discontinuities might interact with each other given there exists a difference in propagating speed.

5 2D Implementation

Expanding to the 2 dimensional plane, the flux is now a vector of two velocity components

$$f = \begin{bmatrix} v_x \\ v_y \end{bmatrix}. \tag{5.1}$$

Expanding (2.3) into two dimensions will now look like

$$\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} + \frac{\partial f(u)}{\partial y} = 0 \tag{5.2}$$

for the simplest way of discretizing a two dimensional domain, namely the rectangular grid.

5.1 Finite Volume Method

In order to better compare between the numerical results of the rectangular- and hexagonal- grid, we will utilize the finite volume implementation which describes how much the flowrate fills or empties a cell volume based on the flow through the cell edges. As such, we will modify Δx in (2.3) to reflect that. Rather than being defined as the distance from cell center to cell center along the x-axis, Δx is now replaced by

$$\Delta x \to \frac{A}{L}. \tag{5.3}$$

where A is the total area of a cell, and L is the length of the cell edge. Note that for a rectangular grid, there would be no change as $L = \Delta x_{\text{rectangle}}$ and $A = L^2 = (\Delta x_{\text{rectangle}})^2$ resulting in (5.3) evaluating to

$$\frac{A}{L} = \frac{(\Delta x_{\text{rectangle}})^2}{\Delta x_{\text{rectangle}}} = \Delta x_{\text{rectangle}} \tag{5.4}$$

while it would differ for the hexagonal grid where $L = \frac{1}{\sqrt{3}}\Delta x_{\text{hexagon}}$ and $A = \frac{\sqrt{3}}{2}(\Delta x_{\text{hexagon}})^2$ resulting in (5.3) evaluating to

$$\frac{A}{L} = \frac{\frac{\sqrt{3}}{2}(\Delta x_{\text{hexagon}})^2}{\frac{1}{\sqrt{3}}\Delta x_{\text{hexagon}}} = \frac{3}{2}\Delta x_{\text{hexagon}}. \tag{5.5}$$

This conversion shifts the perspective from Δx describing the distance from cell center to cell center, to one which describes how the volume of a cell changes based on the flux along the cell edges.

Using the same discretization strategy as for the 1 dimensional case with finite volume method on (5.2) it can be written as

$$u_{i,j}^{n+1} = u_{i,j}^{n} - \frac{L\Delta t}{A}(f_{i+\frac{1}{2},j}^{n} - f_{i-\frac{1}{2},j}^{n} + f_{i,j+\frac{1}{2}}^{n} - f_{i,j-\frac{1}{2}}^{n}) \tag{5.6}$$

where each f represent its own cell edge (North, South, East, West).

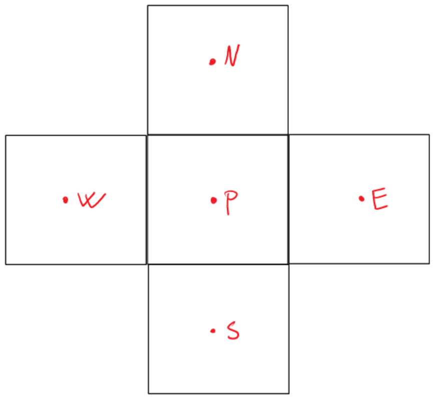


Figure 5.1:
Rectangular neigbors to a point P

Using the same method as for (4.2), (5.6) can be written as

$$\begin{aligned} u_{i,j}^{n+1} = u_{i,j}^{n} - \frac{L\Delta t}{A}\sum_{m=0}^{\infty}(&a_{i,j-\frac{1}{2}-m}^{m+}\Delta_{i,j-\frac{1}{2}-m} + a_{i,j+\frac{1}{2}+m}^{m-}\Delta_{i,j+\frac{1}{2}+m} \\ &+a_{i-\frac{1}{2}-m,j}^{m+}\Delta_{i-\frac{1}{2}-m,j} + a_{i+\frac{1}{2}+m,j}^{m-}\Delta_{i+\frac{1}{2}+m,j}) \end{aligned} \tag{5.7}$$

Currently, this scheme is not fully suited to capture flux which are not flowing normal to a cell edge. We will explore an alternative form of (5.7) in section 5.2 and explain why it is more suited in section 6.1 when we explore visually which cells in (5.7) are being used on the grid.

In the case for the hexagonal grid structure, the flux is no longer flowing along the x- and y- direction, and so a modification to (5.1) and (5.7) is needed. Suppose the flux flow can be represented by the flux direction perpendicular to the cell edges of the hexagon

$$f = \begin{bmatrix} v_i \\ v_j \\ v_k \end{bmatrix} \tag{5.8}$$

where

$$\begin{aligned} v_i &= [1,0] \\ v_j &= \tfrac{1}{2}[1,\sqrt{3}] \quad . \\ v_k &= \tfrac{1}{2}[-1,\sqrt{3}] \end{aligned} \tag{5.9}$$

Given that any two dimensional flux is always written in the form (5.1), so the corresponding velocities $v_{i,j,k}$ need to be calculated. Given that we know the flux velocities (5.9), (2.3) can now be represented by

$$\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial i} + \frac{\partial f(u)}{\partial j} + \frac{\partial f(u)}{\partial k} = 0 \tag{5.10}$$

where the discretization (5.6) is now

$$u^{n+1}_{i,j,k} = u^n_{i,j,k} - \frac{L\Delta t}{A}(f^n_{i+\frac{1}{2},j,k} - f^n_{i-\frac{1}{2},j,k} + f^n_{i,j+\frac{1}{2},k} - f^n_{i,j-\frac{1}{2},k} + f^n_{i,j,k+\frac{1}{2}} - f^n_{i,j,k-\frac{1}{2}}) \tag{5.11}$$

where each f represent its own cell edge (East, West, North-East, North-West, South-East, South-West).

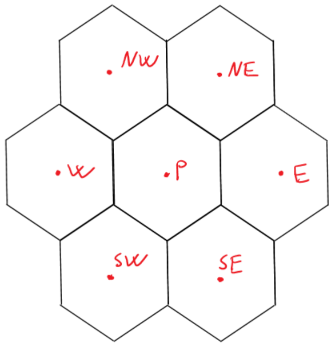


Figure 5.2:
Hexagonal neigbors to a point P

Lastly, to complete the discretization in 2 dimensions for both the rectangular- and hexagonal- grid. The (5.2) equivalent version for the hexagonal grid can be written as

$$
\begin{aligned}
u_{i,j}^{n+1} = u_{i,j}^{n} - \frac{L\Delta t}{A}\sum_{m=0}^{\infty}(&a_{i+\frac{1}{2}+m,j,k}^{m-}\Delta_{i+\frac{1}{2}+m,j,k} + a_{i-\frac{1}{2}-m,j,k}^{m+}\Delta_{i-\frac{1}{2}-m,j,k} \\
&+a_{i,j+\frac{1}{2}+m,k}^{m-}\Delta_{i,j+\frac{1}{2}+m,k} + a_{i,j-\frac{1}{2}-m,k}^{m+}\Delta_{i,j-\frac{1}{2}-m,k} \\
&+a_{i,j,k+\frac{1}{2}+m}^{m-}\Delta_{i,j,k+\frac{1}{2}+m} + a_{i,j,k-\frac{1}{2}-m}^{m+}\Delta_{i,j,k-\frac{1}{2}-m})
\end{aligned} \tag{5.12}
$$

which, as with (5.7) we will explore further in sections 5.2 and 6.1.

5.2 Directional Splitting

A popular method used by multiple authors in order to simplify the two dimensional problem into a series of one dimensional problems is called the Dimension Splitting method, see (LeVeque, 1982; Morales-Hernandez et al., 2014; Qian and Lee, 2011) for full details. The essential idea behind this method is that the components of the flux function are orthogonal to each other, and so the discretized equation (5.7) can be broken into multiple 1 dimensional equations where the change in the conserved variable only depend on flow in one direction at a time

$$
u_{i,j}^{n+1} = u_{i,j}^{n} - \frac{L\Delta t}{A}\sum_{m=0}^{\infty}(a_{i,j-\frac{1}{2}-m}^{m+}\Delta_{i,j-\frac{1}{2}-m} + a_{i,j+\frac{1}{2}+m}^{m-}\Delta_{i,j+\frac{1}{2}+m}) \tag{5.13a}
$$

$$
u_{i,j}^{n+1} = u_{i,j}^{n} - \frac{L\Delta t}{A}\sum_{m=0}^{\infty}(a_{i-\frac{1}{2}-m,j}^{m+}\Delta_{i-\frac{1}{2}-m,j} + a_{i+\frac{1}{2}+m,j}^{m-}\Delta_{i+\frac{1}{2}+m,j}) \tag{5.13b}
$$

This formulation also alleviate some of the numerical diffusion caused when the flux direction is not pointing directly orthogonal to a cell edge. In order to compare the results, we need something similar to (5.13) for the hexagonal grid. So even if the components of the flux are no longer orthogonal to each other, we will explore the case when (5.12) can be written as a series of 1 dimensional equations, where each equation takes care of one of the flux directions

$$u^{n+1}_{i,j,k} = u^n_{i,j,k} - \frac{L\Delta t}{A}\sum_{m=0}^{\infty}(a^{m-}_{i+\frac{1}{2}+m,j,k}\Delta_{i+\frac{1}{2}+m,j,k} + a^{m+}_{i-\frac{1}{2}-m,j,k}\Delta_{i-\frac{1}{2}-m,j,k}) \tag{5.14a}$$

$$u^{n+1}_{i,j,k} = u^n_{i,j,k} - \frac{L\Delta t}{A}\sum_{m=0}^{\infty}(a^{m-}_{i,j+\frac{1}{2}+m,k}\Delta_{i,j+\frac{1}{2}+m,k} + a^{m+}_{i,j-\frac{1}{2}-m,k}\Delta_{i,j-\frac{1}{2}-m,k}) \tag{5.14b}$$

$$u^{n+1}_{i,j,k} = u^n_{i,j,k} - \frac{L\Delta t}{A}\sum_{m=0}^{\infty}(a^{m-}_{i,j,k+\frac{1}{2}+m}\Delta_{i,j,k+\frac{1}{2}+m} + a^{m+}_{i,j,k-\frac{1}{2}-m}\Delta_{i,j,k-\frac{1}{2}-m}) \tag{5.14c}$$

Since the separated equations in (5.14) does not represent a unique dimension, a more fitting name for such an implementation will be called directional splitting from here on.

6 Grid Analysis

Before heading into the core part of this thesis which is the comparison between simulation results, some details about the different grids needs to be discussed first. For any grid structure, each cell in the domain needs a unique index to identify the cell. For a 5x5 grid, the initial setup we chose as indexing for each cell is shown in Figure 6.1.

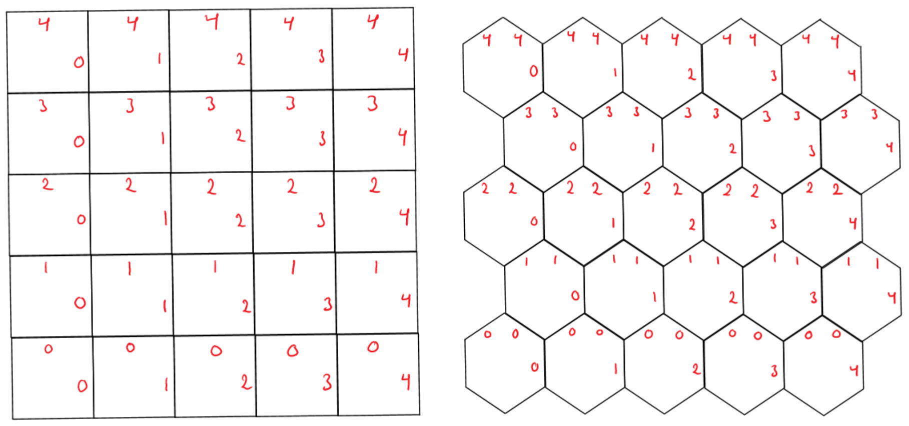


Figure 6.1:
(Left) Cell mapping for a rectangular grid
(Right) Cell mapping for a hexagonal grid

Here we start in the bottom left corner and set each index corresponding to a flux direction equal to 0, and then we incremented the index depending on which flux direction you are moving along. This method is straight forward for the rectangular grid, however for the hexagonal grid, there is no way to move in the k- th direction from the bottom left cell. As a consequence, we chose the j- and k- th index for the entire bottom row to have index 0 and then followed the pattern described earlier. Noting that for the hexagonal grid, the j- and k- th index are equal for all rows, so some final simplifications were done which is showcased in Figure 6.2, where those two indexes were merged together into one index representation.

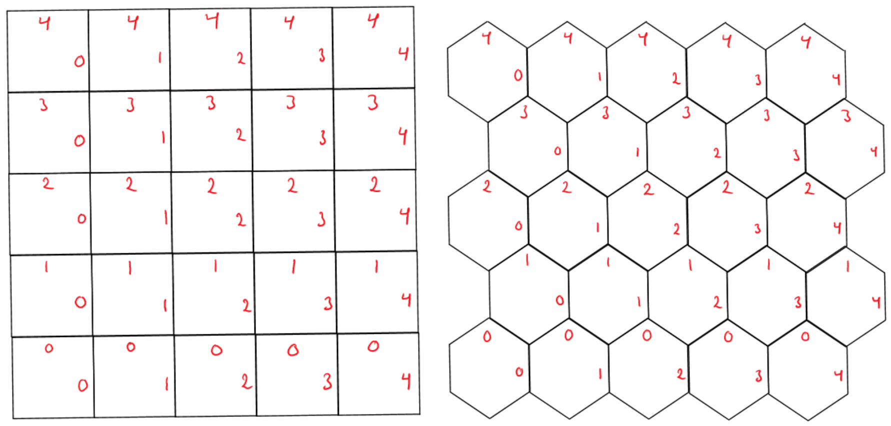


Figure 6.2:
(Left) Cell mapping for a rectangular grid
(Right) Simplified cell mapping for a hexagonal grid

6.1 Visualization of the Numerical Scheme

Now that the indexing has been taken care of, next we need an illustration of what (5.7) and (5.12) would look like, and how using directional splitting changes the scheme.

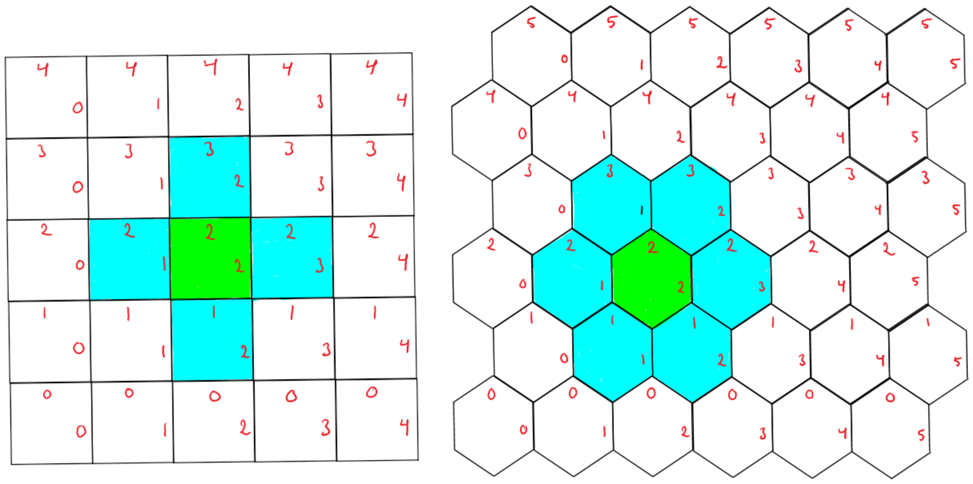


Figure 6.3:
(Left) Rectangular cell and its neighbors
(Right) Hexagonal cell and its neighbors

In Figure 6.3, we see that for Courant numbers up to 1, (5.7) will be unsuited given a diagonal flux while (5.12) will continue to be stable for all flux

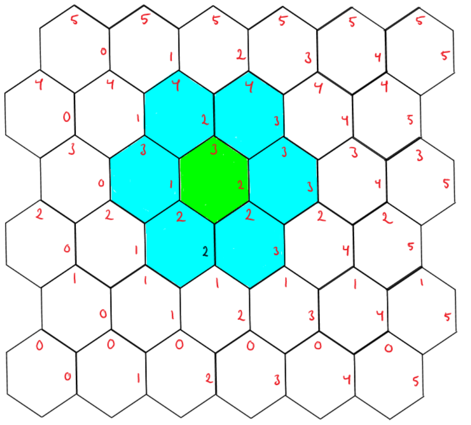
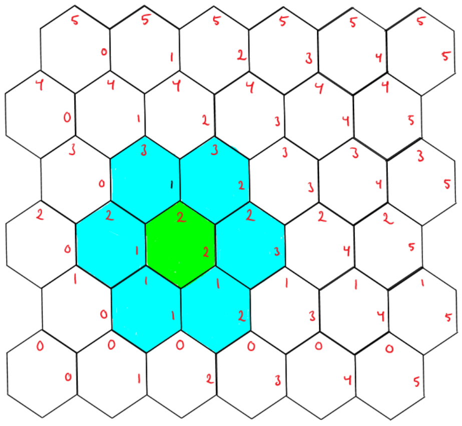



Figure 6.4:
(Left) Odd row value
(Right) Even row value

directions. Another thing to note for the hexagonal grid is that the indexing for the neighbors changes depending on if the row is even or odd.

If we now increasing to larger values of C, the rectangular grid will be more and more limited to which flux directions will continue to be stable, and the hexagonal grid will start to have the same problem the rectangular grid has, just not nearly as bad, as shown in Figure 6.5.

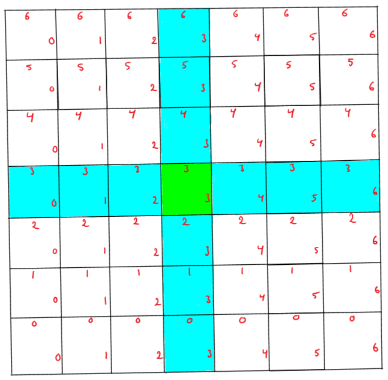
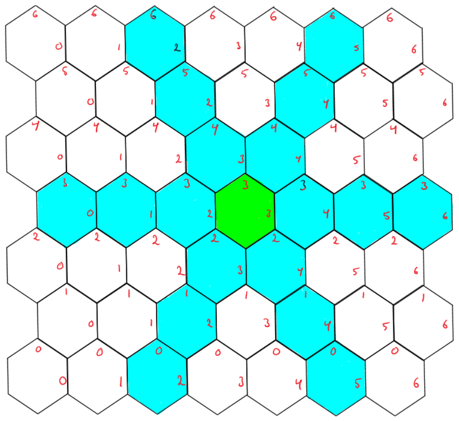



Figure 6.5: C = 3
(Left) Rectangular cell and its neighbors
(Right) Hexagonal cell and its neighbors

From Figure 6.3 and 6.5 we can see that (5.7) and (5.12) will poorly capture any flux which flows towards a white cell. Since the discretized scheme is only using cells which are highlighted in blue. As an example, if the flux were to move diagonally to the top right corner with a flux

$$f = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \tag{6.1}$$

then the rectangular grid will have some of the conserved variable move east and some move north which creates diffusion. Using the same flux, we can showcase the steps of directional splitting for the rectangular grid in Figure 6.6.

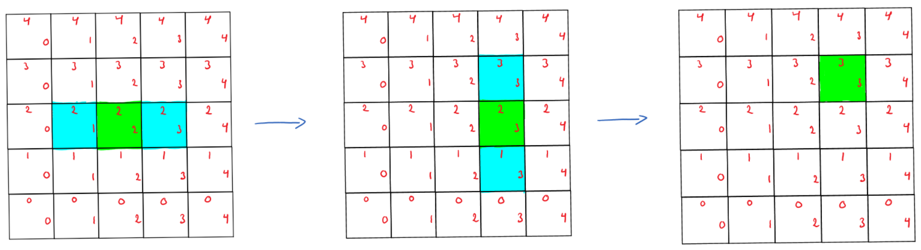


Figure 6.6:
Directional Splitting on a rectangular grid
In each step, the position of cell which is being updated (green) and the neighboring cells which contribute to the position updating (blue) are shown

It is a two-step procedure where we arbitrarily choose a flux direction to start, and update the conserved variable based on only the flux along that direction. Once the conserved variable is update, the second flux direction is used to update the conserved variable from that updated position which lands it in its desired position. In Figure 6.3 (Left) a flux equal to (6.1) would result in an unstable scheme without directional splitting.

Directional splitting on a hexagonal grid will be a three step process shown in Figure 6.7.

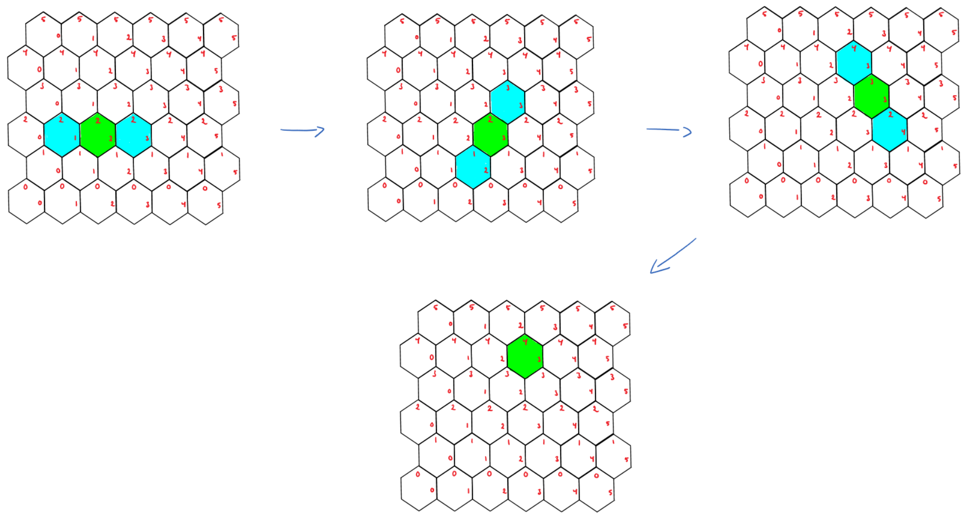


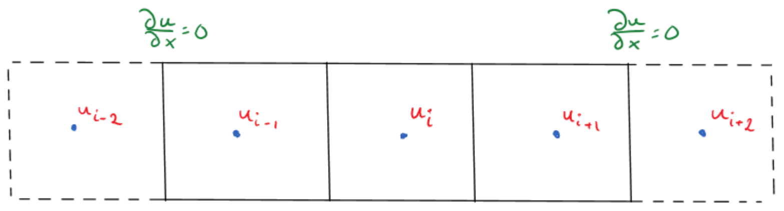
Figure 6.7:
Directional Splitting on a hexagonal grid
In each step, the position of cell which is being updated (green) and the neighboring cells which contribute to the position updating (blue) are shown

Similarly to Figure 6.6, the order of which direction will update the conserved variable is chosen arbitrarily. Each flux direction will update the conserved variable step-wise until it reaches its updated position.

6.2 Ghost Cells

As a finishing topic to discuss around the grid and how to update the conserved variable, we need to mention how the boundaries of the domain will be handled. The method we will be using to handle boundary conditions is a method which utilizes so called "ghost cells". Ghost cells are additional cells added to the perimeter of the domain which help with boundary cell updating. Figure 6.8 illustrate the one dimensional case with a particular boundary condition

$$\frac{\partial u}{\partial x} = 0. \tag{6.2}$$



$\frac{\partial u}{\partial x}=0$
$\frac{\partial u}{\partial x}=0$
u_{i-2}
u_{i-1}
u_i
u_{i+1}
u_{i+2}

Figure 6.8:
Ghost cells on a 1 dimensional grid

Given the boundary condition (6.2), the respective ghost cell values can be calculated. Discretizing equation (6.2) we have

$$\frac{u_{i+1}-u_i}{\Delta x}=0 \tag{6.3}$$

or

$$u_{i+1}=u_i. \tag{6.4}$$

Using (6.4), the left and right boundary cells of Figure 6.8 are the following

$$\begin{aligned} u_{i-2} &= u_{i-1} \\ u_{i+2} &= u_{i+1} \end{aligned} . \tag{6.5}$$

Extending to a two dimensional grid, the same calculations are done in order to determine the values for the ghost cells. When using LTS, the number of ghost cells needed at each boundary edge equals k in a $(2k+1)$ stencil system, where reflective boundary conditions are used. The LTS extension to (6.2), for a grid with N number of cells, is depicted in Figure 6.9 and is equal to

$$\begin{aligned} u_{k-m} &= u_{k+1+m}, \quad & m \in \{0,1,2,...,k-1\} \\ u_{N-k-1-m} &= u_{N-k+m}, \quad & m \in \{0,1,2,...,k-1\} \end{aligned} . \tag{6.6}$$

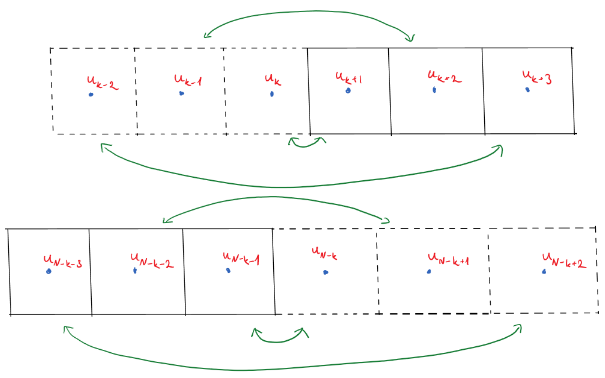


Figure 6.9:
Reflective boundary conditions using multiple ghost cells
(Above) Left boundary
(Below) Right boundary

7 Simulation Setup

In this thesis, we will consider both the case when the conservation law is linear and when it is non-linear. As such, we will go over a brief overview of the simulation setup for each case before showing the simulation results.

7.1 Computational Domain

The computational domain will differ slightly in size for the rectangular and hexagonal grid, the reason for this is that we prioritized having equally many cells for both grid types rather than keeping the domain shape equal. This will only result in a vertical stretch for the hexagonal domain compared to the rectangular domain. As for the rectangular grid, the computational domain used will be a square with side lengths equal

$$L = 50. \tag{7.1}$$

The domain will then be divided into N cells along the horizontal and vertical direction creating a NxN grid where $N \in \mathbb{N}$, where the width and height of the cell dx, dy equaling

$$dx = dy = \frac{L}{N} \tag{7.2}$$

which is shown in Figure 7.1.

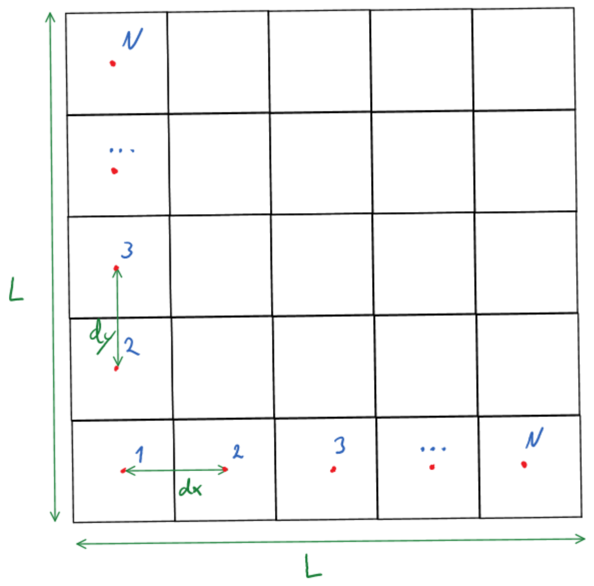


Figure 7.1:
Reflective boundary conditions using multiple ghost cells
Discretized rectangular setup with dx, dy in terms of L and N

As for the hexagonal grid, the computational domain is set to

$$L_x = L \tag{7.3a}$$

$$L_y = L\frac{\sqrt{3}}{2} \tag{7.3b}$$

where L_x, L_y represent the length of the domain along the horizontal and vertical direction respectively. Given this vertical stretch, the hexagonal grid can now fit an equal number of cells as the rectangular grid, which is what was needed. The values for dx, dy can be written as

$$dx = \frac{L_x}{N + \frac{1}{2}} \tag{7.4a}$$

$$dy = dx\frac{\sqrt{3}}{2} \tag{7.4b}$$

showcased in Figure 7.2 below.

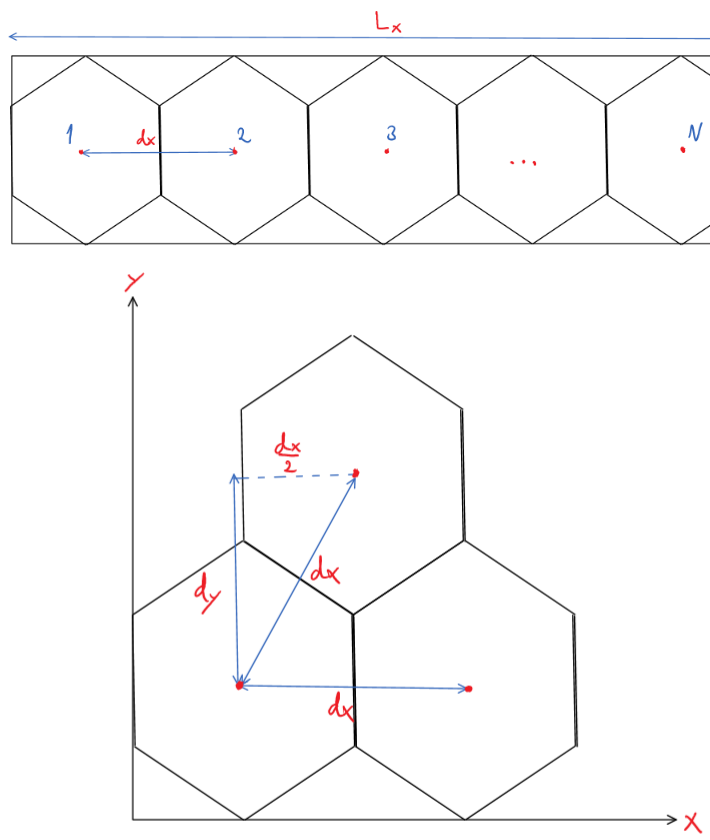


Figure 7.2:
(Above) dx in terms of L_x and N
(Below) dy in terms of dx

7.2 Linear Flux

In the case of the linear flux, the flux field across the computational domain is unidirectional, uniform and non-changing throughout the entire simulation. The domain is centered at the origin and the initial condition of the conserved variable will be a square given

$$|x|, |y| \leq 10 \tag{7.5}$$

where the square has a value of 3 and everything else has a value of 1. Given this initial condition, we will run two different comparison tests simulating

1. the numerical error with flux pointing along the directions dictated by the angle $\theta \in [0, 2\pi)$ with increments of $\frac{\pi}{12}$ (or 15°)

2. the time used during the simulation along the flux direction $[1, 0]$ for varying values of C

In order to keep the entire initial condition still within the computational domain after the simulation, the simulation will start at time 0 and will terminate once it has passed 1. Note however, that in order to keep the time step consistent with the defined Courant number, the total time may severely exceed the value 1 given a large enough C value. This may affect our results since the initial condition will have partially or entirely left the computational domain, so the upper limit for the value for the Courant number will be set to 10% of the value of N, which we found keeps the initial condition within the boundaries after the simulation.

7.3 Non-Linear Flux

For the non-linear flux, the flux field can be represented by

$$F = \begin{bmatrix} -y \\ x \end{bmatrix} \tag{7.6}$$

which creates a circular flux around the origin flowing in the anti-clockwise direction. In this case, the flux will be different for each position of the grid, however it does not change over time. The initial condition will be a quarter circle located in the upper right quadrant where the quarter circle has a value of 3 while the rest has a value of 1. We will be running one type of simulation

- Rotate the initial condition a full revolution 2π around the origin

in order to compare with an analytical solution. The simulation results will be tested for varying values of C and the total time will be measured for each C value to see if we can locate an optimal value in terms of simulation time and error. For our non-linear flux experiment, it is important that the simulation terminates exactly at one full revolution and does not continue past that. Since the analytical solution is static having the simulation terminate at a time not as a multiple of 2π will give inaccurate results as we will be comparing two different things. In order to prevent this, a check on if the next time step will exceed the value of 2π will be done, and if that is the case, the time step will be changed to $2\pi - t$ where t is the current time.

7.4 Boundary Updating

Given the specific examples we will go through, updating the values of the ghost cells in order to reflect a zero gradient boundary reduces to evaluating the ghost cells to be equal to the value 1 since neither of the test cases reaches the boundary.

7.5 Error Calculation

In order to compare simulation results, we are calculating the absolute difference between the analytical solution and numerical results in each cell.

For the linear flux, the analytical solution is given in (2.12) where the value of x is taken to be the point value in the center of a cell. To distinguish between the numerical results and the analytical solution, let us denote the analytical solution in a specific cell as $U_{i,j}$ and the numerical value of the same cell as $u_{i,j}$. Given this, the error calculation can be written as

$$\text{Error} = \sum_{i,j} |(U_{i,j} - u_{i,j}) \cdot A_{i,j}| \tag{7.7}$$

By multiplying with the area of each cell, we are able to compare between results using different grid sizes. For the non-linear flux, the analytical solution will be the initial condition, so one would just replace $U_{i,j}$ by the initial value of the cell.

7.6 Criterion for Exact Solution

In order for (5.13) and (5.14) to achieve analytical solutions, the Courant number along each direction needs to be an integer value. Visually, one can think of it as the conserved variable fully moving from one cell to another in all directions. Following the center cell, Figure 6.6 and 6.7 showcases an example of how an exact solution would look like for the rectangular and hexagonal grid.

8 Simulation Results

Implementing what was described in section 7 we are now ready to test our initial hypothesis.

8.1 Implementation

All necessary code has been written from scratch in Python with standard libraries such as numpy and matplotlib. In addition, since Python is not known for its fast for/while loops, the library called numba was used in order to significantly speed up the computations which was especially needed in the case of the non-linear flux where the flux was unique at all cell edges. In order to create the hexagonal shapes, we used already created github code(Kazakov, 2021) which was a huge help as matplotlib does not easily create non-rectangular grids for plotting.

8.2 Linear Flux

Starting with the linear flux, we will start by calculating the error for different flux directions. Since there will be a total of 24 error plots for both the rectangular and hexagonal grid, we decided to split the graphs into five categories.

1. Normal to a cell face of the rectangle
2. Diagonal to a cell face of the rectangle
3. Normal to a cell face of the hexagon
4. Diagonal to a cell face of the hexagon
5. Remaining angles not present in the other four categories

Note that there are some angles which fit more than one category, in that case they will show up in all places. For these simulations we will use a grid size N of 500 creating a 500x500 grid for both the rectangle and hexagon.

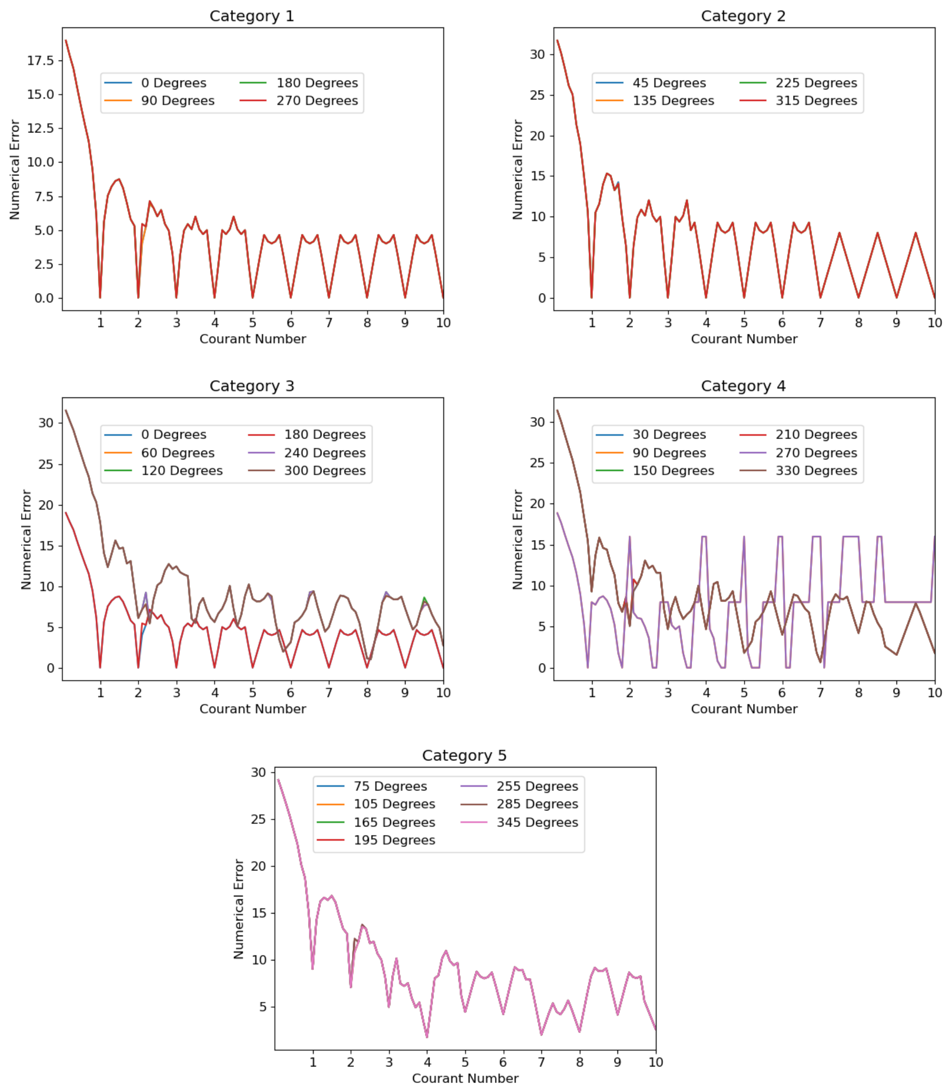


Figure 8.1:
Numerical error based on flux direction for a rectangular grid.
From top left corner and across are the error resulted from the 5 different categories visualized.

Starting with the rectangular grid, we see in Figure 8.1 that both category 1 and 2 give exact results when the Courant number is of an integer value. This indicates that as long as the flux is pointing out from a cell face or through the diagonals, the numerical scheme is able to capture the exact solution. From category 3 and 4, the only lines which give exact results are repeated from category 1 and 2. The new values present in category 3 and 4 are not able to capture the analytical solution exactly given any value of the Courant number, although some do come close to it. For category 5, we see that the scale does not even reach 0, so no value of the Courant number will yield exact results. To summarize, looking at what flux direction a rectangular grid is able to capture, it looks like it is only able to capture flux flowing orthogonal to a flux face or across the diagonals of the rectangle.

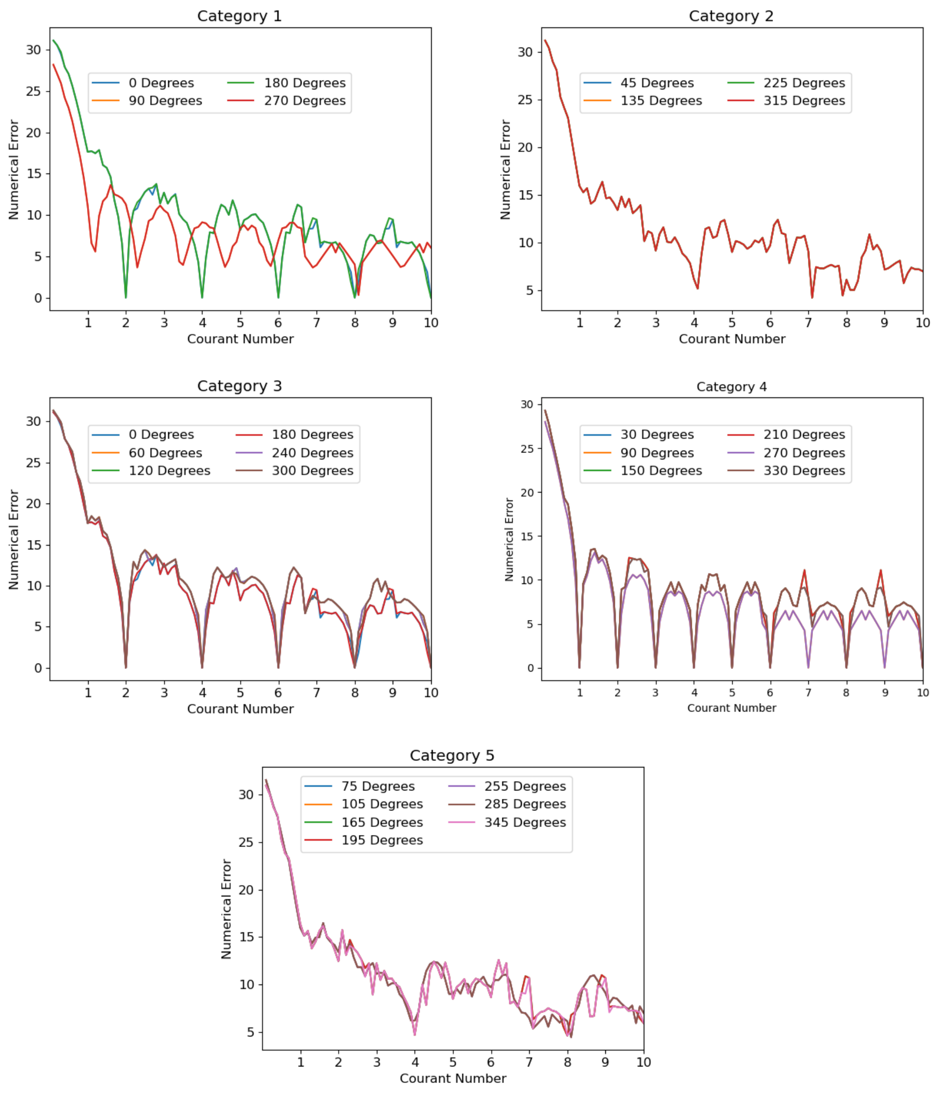


Figure 8.2:
Numerical error based on flux direction for a hexagonal grid.
From top left corner and across are the error resulted from the 5 different categories visualized.

For the hexagonal grid, we notice in Figure 8.2 that category 4 is similar to category 1 and 2 for the rectangular grid where an exact solution occurs for every integer value of the Courant number. However, what differs between them is that given a flux normal to the cell faces, an exact solution is no longer possible without the use of directional splitting since an exact solution only occurs for even values of the Courant number. In order to understand why this is the case we will plot the path the initial condition takes on the hexagonal grid as it is updated along the i-,j- and k- th direction. First let us start with a flux flowing along the i- th direction, namely $f = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and see what happens when the Courant number equals 1.

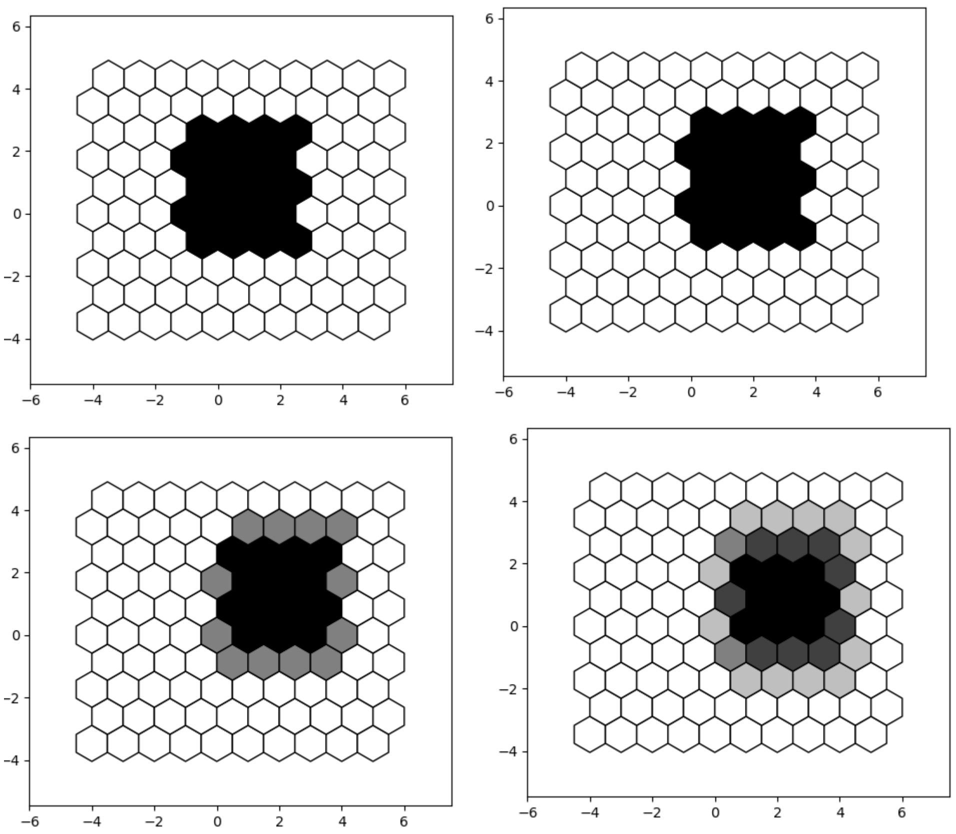


Figure 8.3: $N = 10$
Initial condition is shown in the top left and then it moves in all three directions where the final solution is shown in the bottom right

Here we see that the initial condition will move an entire grid cell along the i-th direction while that is not the case along the j-th and k-th direction meaning we do not get an analytical solution with $C = 1$. Results for $C = 2$ are visualized in Figure 8.4

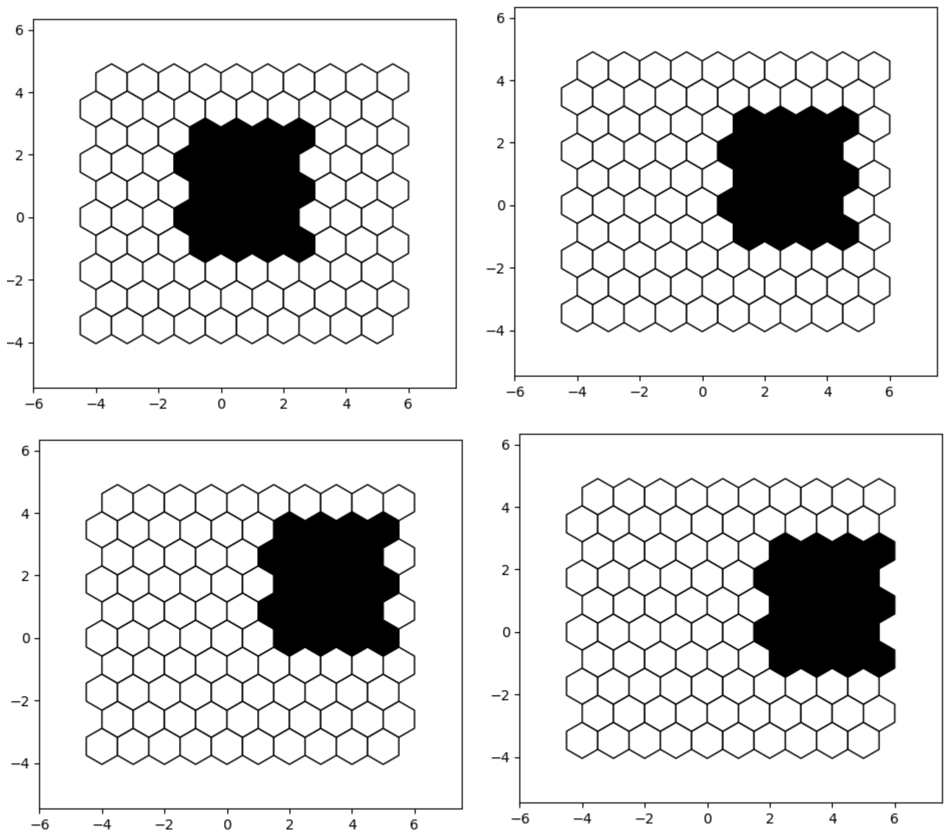


Figure 8.4: $N = 10$
Initial condition is shown in the top left and then it moves in all three directions where the final solution is shown in the bottom right

Here, the initial condition moves two cells along the i-th direction and one cell along the j-th and k-th direction resulting in an exact solution. The theory behind this result lies in how a_i, a_j, a_k was defined to be the magnitude of f along the i, j, k directions. In this specific example, the magnitude of a_j and a_k turns out to be exactly half the magnitude of a_i. This means that when the magnitude of a_i is an even number, the other two values will become an integer which results in an exact solution. Now, changing the Courant number back to 1 and setting the flux to $f = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ which is in category 4, we expect to see an exact solution even though the Courant number is not even.

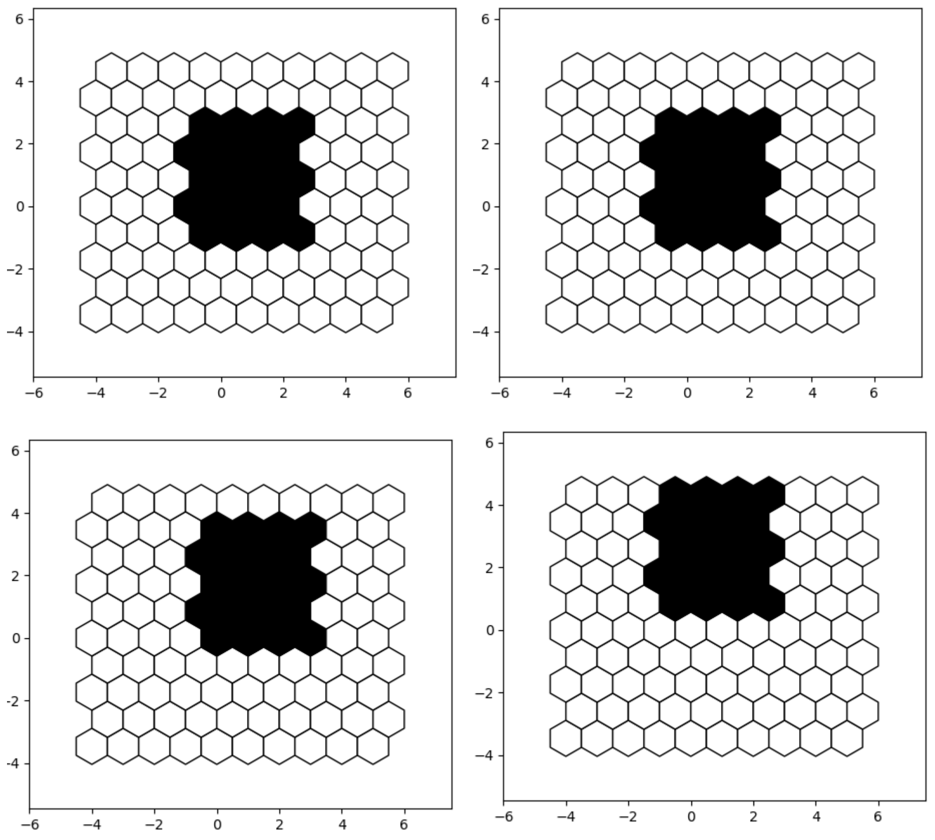


Figure 8.5: $N = 10$
Initial condition is shown in the top left and then it moves in all three directions where the final solution is shown in the bottom right

In Figure 8.5 we now have $a_i = 0$, meaning that as long as the Courant number along the j-th and k-th direction are integers then we have an exact solution. Remember that $a_{i,j,k}$ are calculated using the dot product $a_{i,j,k} = f \cdot n_{i,j,k}$, so the velocities along the j-th and k-th direction equals

$$a_j = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \cdot \frac{1}{2} \begin{bmatrix} 1 \\ \sqrt{3} \end{bmatrix} = \frac{\sqrt{3}}{2} \tag{8.1a}$$

$$a_k = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \cdot \frac{1}{2} \begin{bmatrix} -1 \\ \sqrt{3} \end{bmatrix} = \frac{\sqrt{3}}{2}. \tag{8.1b}$$

Since the time step length is defined by first defining a target Courant number and using the max value between the three velocities $a_{i,j,k}$, an exact solution shown in Figure 8.5 suggests that $a_j = a_k = \max(|a_i|, |a_j|, |a_k|)$ which is indeed the case since $a_i = 0$. For the other categories, we see that category 2 and 5 give similar results which makes sense since the flux is 15° off being normal to a cell face for all angles tested in category 2 and 5. To summarize, given the Courant number is equal to 1, then the hexagonal grid is only able to capture an exact solution along the diagonals and we need to increase the Courant number to be an even number in order to achieve exact solutions at the cell faces.

Next, we will single out a flux direction, namely the flux $f = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ since it is perpendicular to a cell face in both the rectangular and hexagonal grid, and because similar results were found for all other flux directions. Using this flux, we will look at the simulation time required as the Courant number grows larger. Remember that the number of cells required to update a cell grows with the Courant number by the relation $(2k + 1)$ in every direction the discretized scheme was split into. This means that although a larger Courant number will reduce the number of iterations required in order to complete the simulation, each iteration will be more computationally intensive which may slow the overall simulation time more than what was gained by reducing the number of iterations. In this case, we will increase the number of up to 50 which is 10% of the the value N and see if we can find a value for the Courant number which minimizes the simulation time.

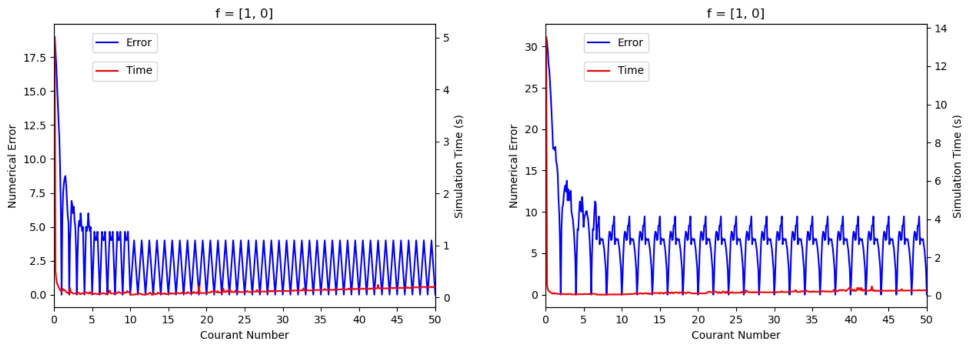


Figure 8.6:
Comparison between simulation time and accuracy given a Courant number
(Left) Rectangular Grid
(Right) Hexagonal Grid

From Figure 8.6, we see a fast decrease in the simulation time up to $C = 0.1$ and then roughly no change for higher Courant numbers. Comparing the numerical error, it seems to give no additional accuracy past a Courant number of 10 for the rectangular grid and 8 for the hexagonal grid. Comparing these numbers with the chosen grid size where $N = 500$ there seems to be no reason to go above a Courant number of 1-2% of N. Similar results were also found for grid sizes $N = \{100, 200, 300, 400\}$.

8.3 Non-Linear Flux

Moving over to the non-linear flux, we are using a grid size N of 500 again and then comparing the numerical error using varying values for the Courant number. Similar to the linear flux, the goal is to find an optimal value for the Courant number which both minimizes the simulation time and numerical error. Figure 8.7 is similar to Figure 8.6 in that both the error and computation time is plotted based on the Courant number.

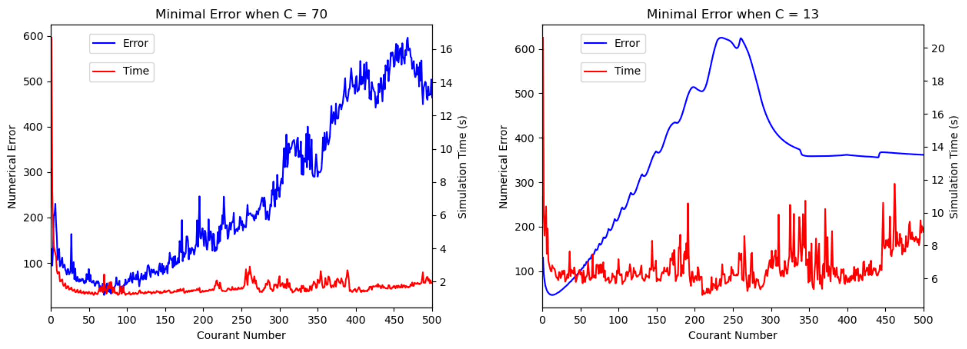


Figure 8.7:
Comparison between simulation time and accuracy given a Courant number
(Left) Rectangular Grid
(Right) Hexagonal Grid

Figure 8.7 shows that there exists some optimal value for the Courant number, of which going above or below it will increase the numerical error. However, after testing different grid sizes of $N = \{100, 200, 300, 400, 500\}$ with the upper bound of the Courant number corresponding to the value of N. The shape of the graph was similar for both the rectangular and hexagonal grids however the optimal Courant number seems to increase linearly with N for the rectangular grid and stay relatively constant for the hexagonal grid. In order to investigate the results of Figure 8.7 further, we will plot the results for a grid size of $N = 100$ using their respective optimal value for C which was 24 and 7 for the rectangular and hexagonal grid.

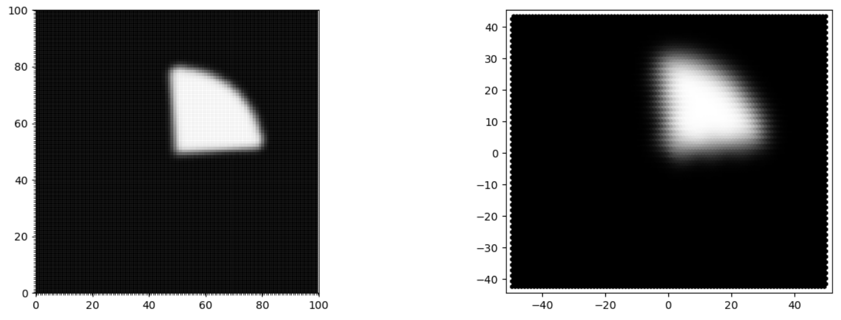


Figure 8.8:
Simulation results after 1 revolution using the optimal Courant number when $N = 100$
(Left) Rectangular Grid
(Right) Hexagonal Grid

Here we see that the rectangular grid has much less diffusion than the hexagonal grid given what the optimal value of the Courant number. Being skeptical of this result for the hexagonal grid and by extension the results in Figure 8.7 for the hexagonal grid we will compare the grids given the same Courant number of 24.

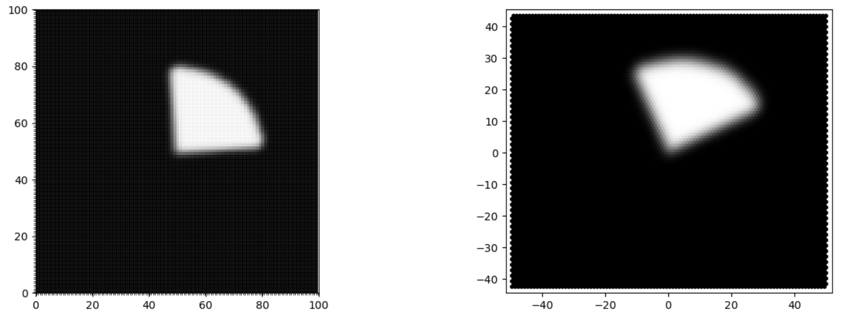


Figure 8.9:
Same as Figure 8.8 with equal value of $C = 24$ for both plots
(Left) Rectangular Grid
(Right) Hexagonal Grid

Here we see that the diffusion from the hexagonal grid has clearly been reduced to be more in line with the rectangular grid, however the reason for a higher numerical error comes from the initial condition having traveled more than 360°.

9 Conclusion

When working with simulations in two dimensions there are multiple ways to discretize the domain. To that end we set out to compare two different ways of discretizing when working with hyperbolic conservation laws, namely the rectangular and hexagonal grid. Both of these grids are structured grids, which meant that we could use strategies such as dimension splitting on the discretized conservation laws in order to treat the problem as a series of one dimensional problems rather than a single two dimensional problem. We found that by using the finite volume method rather than the finite difference method, dimension splitting was not restricted by the normal vectors being normal to each other. Instead, it could be used with normal vectors going in any direction and thus we called it directional splitting rather than dimensional splitting since we thought that was a more fitting name.

9.1 Assessment of Numerical Results

Together with directional splitting we used the LTS method in order to extend the time step length of explicit methods before achieving a divergent solution. This meant that the simulations would require less iterations overall, however each iteration needed to keep track of more cells in order to keep the simulation results stable. We tested the use of directional splitting together with LTS on two examples, one with linear and one with non-linear flux. For the linear flux, we noticed that the rectangular grid gave an analytical solution when the Courant number was an integer value when the flux was normal to the cell edges and along the diagonals. For the hexagonal grid, it did manage to achieve analytical results with integer value of C, meaning that LTS is not required. However, the hexagonal grid only managed those results when the flux was along the diagonals, if the flux was normal to a cell edge, then an analytical solution was only possible for even value of C meaning that LTS is required for analytical solutions along those directions.

Exact Numerical Solutions		
Courant Number	Rectangular Grid	Hexagonal Grid
1	8	6
2	16	12

Table 9.1:
Nr. of angles which result in an exact numerical solution

In table 9.1, we see that the rectangular grid manages to capture more angles of which an exact numerical result can be achieved and as the Courant number increases, the difference between the two numbers will only continue to increase.

Second, we tested a non-linear flux which was a rotation around the center point. Here there was a clear optimal value for the Courant number where the numerical diffusion would be the lowest. For the rectangular grid, the optimal value of the Courant number was somewhere between $15 - 20\%$ of the value for N, however for the optimal Courant number for the hexagonal grid was roughly constant. Being skeptical of that result we tested out why that was the case by plotting the results in Figure 8.8 and 8.9. Those results showed that the hexagonal travels too far given larger Courant numbers which gave rise to larger numerical error. We did not figure out if that was a feature of directional splitting on hexagonal grid or if there were some errors in the code which made it travel further than it was supposed to. If it is the first option then there is not much to be done and one would just have to accept it as is, however if it is the latter than the hexagonal grid has the potential to give better results than the rectangular grid.

9.2 Future Prospects

The main objective of this thesis was to compare between different ways of discretizing a two dimensional domain. In the process we found that the dimension splitting method could be generalized to something we called directional splitting. From this one could explore new ways of discretizing the domain and see what results would arise by using directional splitting. Another thing to explore is to see if the results we got for the hexagonal grid in Figure 8.8 and 8.9 is due to some properties of the hexagonal grid or if it were due to an oversight on our part that we could not figure out.

References

Bressan, A., 2013, Hyperbolic conservation laws: an illustrated tutorial, Modelling and optimisation of flows on networks, Springer, p. 157-245.

Courant, R., K. Friedrichs, and H. J. M. a. Lewy, 1928, Über die partiellen Differenzengleichungen der mathematischen Physik, v. 100, no. 1, p. 32-74.

Dong, H., and F. Liu, 2018, Large time step wave adding scheme for systems of hyperbolic conservation laws: Journal of computational physics, v. 374, p. 331-360, doi: 10.1016/j.jcp.2018.07.016.

Fabero, J., A. Bautista, and L. J. A. M. L. Casasús, 2001, An explicit finite differences scheme over hexagonal tessellation, v. 14, no. 5, p. 593-598.

Hamilton, B., and S. Bilbao, 2013, Hexagonal vs. rectilinear grids for explicit finite difference schemes for the two-dimensional wave equation: Proceedings of Meetings on Acoustics ICA2013, p. 015120.

Harten, A. J. J. o. c. p., 1997, High resolution schemes for hyperbolic conservation laws, v. 135, no. 2, p. 260-278.

Jameson, A., and P. D. J. A. n. m. Lax, 1986, Conditions for the construction of multi-point total variation diminishing difference schemes, v. 2, no. 3-5, p. 335-345.

Jenkins, A. D. J. J. o. P. O., 1992, A quasi-linear eddy-viscosity model for the flux of energy and momentum to wind waves using conservation-law equations in a curvilinear coordinate system, v. 22, no. 8, p. 843-858.

Karaa, S. J. N. M. f. P. D. E. A. I. J., 2006, High-order approximation of 2D convection-diffusion equation on hexagonal grids, v. 22, no. 5, p. 1238-1246.

Kazakov, A., 2021, hexalattice [Source code]. https://github.com/alexkaz2/hexalattice

Lee, D., H. Tien, C. Luo, and H.-N. J. I. J. o. C. M. Luk, 2014, Hexagonal grid methods with applications to partial differential equations, v. 91, no. 9, p. 1986-2009.

LeVeque, R. J., 1982, Large Time Step Shock-Capturing Techniques for Scalar Conservation Laws: SIAM journal on numerical analysis, v. 19, no. 6, p. 1091-1109, doi: 10.1137/0719080.

LeVeque, R. J., 1985, A Large Time Step Generalization of Godunov's Method for Systems of Conservation Laws: SIAM journal on numerical analysis, v. 22, no. 6, p. 1051-1073, doi: 10.1137/0722063.

LeVeque, R. J., 2002, Finite volume methods for hyperbolic problems, v. 31, Cambridge university press.

LeVeque, R. J. J. C. o. p., and a. mathematics, 1984, Convergence of a large time step generalization of Godunov's method for conservation laws, v. 37, no. 4, p. 463-477

Lin, X.-B. J. J. o. D. E., 2000, Generalized Rankine-Hugoniot condition and shock solutions for quasilinear hyperbolic systems, v. 168, no. 2, p. 321-354.

Lindqvist, S., P. Aursand, T. Flåtten, and A. A. J. S. J. o. N. A. Solberg, 2016, Large time step TVD schemes for hyperbolic conservation laws, v. 54, no. 5, p. 2775-2798.

McLachlan, R. I., and G. J. a. p. a. Quispel, 2013, Discrete gradient methods have an energy conservation law.

Morales-Hernandez, M., P. García-Navarro, and J. Murillo, 2012, A large time step 1D upwind explicit scheme (CFL>1): Application to shallow water equations: Journal of computational physics, v. 231, no. 19, p. 6532-6557, doi: 10.1016/j.jcp.2012.06.017.

Morales-Hernandez, M., M. E. Hubbard, and P. J. J. o. c. p. Garcia-Navarro, 2014, A 2D extension of a Large Time Step explicit scheme (CFL> 1) for unsteady problems with wet/dry boundaries, v. 263, p. 303-327.

Morales-Hernández, M., A. Lacasta, J. Murillo, and P. García-Navarro, 2017, A Large Time Step explicit scheme (CFL>1) on unstructured grids for 2D conservation laws: Application to the homogeneous shallow water equations: Applied mathematical modelling, v. 47, p. 294-317, doi: 10.1016/j.apm.2017.02.043.

Qian, Z., and C.-H. J. J. o. C. P. Lee, 2011, A class of large time step Godunov schemes for hyperbolic conservation laws and applications, v. 230, no. 19, p. 7418-7440.