Analysis of AMR and Other Mesh-Refinement Techniques for Inviscid Fluid Problems

Oda Marit Ølmheim

Master Thesis in Applied and Computational Mathematics,

Department of Mathematics,

University of Bergen,

June 2021

Abstract

In this thesis we will look at a numerical method for solving PDEs called the Adaptive Mesh Refinement method. This method is based on the idea that the grid should adapt to the problem during the computations, and increase or reduce the mesh spacing between the numerical points depending on the error of the approximated solution. We want to analyze the mesh refinement used in the method, and how this refinement affects the numerical solution. We study different ways to do this refinement and introduce a new approach in addition to the two presented in the articles [BO84] and [BC89] by Marsha Berger, Joseph Oliger and Phillip Colella. Later, simulations with the advection equation and the Euler equations are done, where we discover that in either case, refining the grid leads to a decreasing accuracy and convergence rate. Another experiment with a radial explosion, shows that when locating the boundary of the fine grid somewhere in the domain where the solution only has small changes, the error improves. We also work with another grid structure called a Staircase boundary, where we try to approximate a sloping boundary with a uniform Cartesian grid. This leads to a staircase formed numerical boundary approximating the sloping edge. Such an approximation have a similar outcome as the refinement, it reduces the convergence rate to 1.

Acknowledgements

I would like to thank my supervisor Magnus Svärd for introducing me to the field, and for making this past year a great and rewarding experience. Thank you for taking the time to discuss ideas and results with me and helping me out with the problems I have faced.

I also want to thank my family for all the support, and for always being there for me to cheer me up when I need it.

Finally, I want to thank all my friends for making the past five years a great time. To my friends at the mathematics department, thank you for being supportive and helping me out, and for all the discussions we have had about math and everything else.

Contents

Chapter 1

Introduction

Fluid dynamics is the study of the mathematics and physics of fluid motion. A fluid is a substance that deforms when subjected to shear stress, and the term fluid is used for gasses and liquids [Kun15]. Fluid dynamics covers a lot of different research fields such as aerodynamics and ocean simulations to mention some. The Navier-Stokes equations model all fluid motion. They are derived from the conservation laws of mass, momentum and energy. Additionally the equations contains a viscous term, which is a property that every fluid has to some degree. The Euler equations are also much applied for this purpose. They describe inviscid flow, as they lack the viscous term from the Navier-Stokes equations, and are often used when the viscosity of the fluid is small enough to be neglected. This makes the flow easier to model. Still, one has to be careful when neglecting viscosity, as even the smallest amount might have a great impact on the fluid in some cases, for example in boundary layers [Ach91]. There exists no proof of well-posedness for the Navier-Stokes and Euler equations, meaning that there is no official proof of the existence of a unique solution or stability for the equations. When wanting to solve one of these equations, we have to approximate it numerically by using Computational Fluid Dynamics (CFD).

Computational fluid dynamics is the study of numerical approximation methods for Fluid dynamic problems. When in need of simulating the motion of a fluid, these numerical methods are used to compute an approximated solution of the PDEs describing the fluid motion. Even though the field is still limited by the complexity of the problems, it is constantly growing, and through simulations one can achieve accurate approximations to a degree that has not been achieved earlier [Whi11].

There are many ways to discretize and approximate a PDE. Depending on the type of PDE and the purpose of the calculations, the methods all have their advantages. For hyperbolic PDEs the finite difference and the finite volume methods are popular choices. The hyperbolic PDEs does not change globally all over the domain during a time step, but the change happens within a local set of points. The finite difference and finite volume methods are similar when used on uniform Cartesian grids, which we will look at in this thesis. The two methods differs in that the finite volume method is based on calculating the flow through the control volumes around the grid points, while the finite difference method approximates each point from the surrounding points. The Euler equations are hyperbolic, and so is two of the three Navier-Stokes equations.

These problems become complicated quite fast. The simulations of such problems are therefore compromised, and there are always a need to prioritize the different parts of the simulation. In many cases, the accuracy is the most important, and the most accurate method is used. However, time is also an issue, and for some simulations, one can be satisfied with less precision if it means that time and costs are reduced. The Adaptive Mesh Refinement method which we will be studying in this thesis, revolves around this problem. Trying to enforce both accuracy and time, it refines only the parts of the grid that are in need of more precision, avoiding time consuming refinement everywhere on the computational domain. As an example, when simulating boundary layers and turbulence, the precision is crucial. For such computations, a high amount of accuracy is necessary, and would be supplied by a fine grid. The Adaptive Mesh Refinement method was introduced by Marsha Berger and Joseph Oliger in [BO84], and later by Marsha Berger and Phillip Colella [BC89]. The method is used in different fields such as engineering, physics, data science and computational mathematics.

1.1 Thesis outline

In the next chapters we will go through some theory. Chapter 2 contains preliminaries with PDE theory and numerical analysis, Chapter 3 contains some theory from fluid dynamics including a presentation of advection equation and the Euler equations, and in Chapter 4 we present the AMR method. In this chapter we go into details about how the method is recreated in the simulations, and we will also introduce an alternative way of refining meshes which is constructed in order to achieve stability in calculations. In the end of this chapter we present the numerical experiments from the mesh refinement theory, and discuss the results. In Chapter 5 we look into Staircase Boundaries, which is a way to approximate a sloping edge of a domain with a Cartesian grid. At the end of this chapter the results of some simulations with the Euler equations are presented and discussed. Chapter 6 concludes the work done in the thesis.

Chapter 2

Preliminaries

We start by going through some PDE theory and numerical analysis.

There are certain properties we look for in PDEs and in numerical schemes. When these are satisfied, we know more about what to expect when doing numerical approximations of fluid dynamic problems. Three of these properties are Well-posedness, stability and consistency, which we will explain here. The theory is mostly explained with finite difference schemes. We will be using finite volumes schemes, but for the cases in this thesis the schemes can be proven equal, and so the finite difference theory can be applied. The explanations and definitions in this chapter are obtained from Chapter 2 in [Gus07], along with some of the notation.

2.1 Well-posedness

Consider the following initial-boundary value problem:

$$\begin{aligned} u_t &= Pu + F,\ \ 0 \leq t \\ Bu &= g \\ u &= f, t = 0 \end{aligned} \tag{2.1}$$

where P is a differential operator, B is a boundary operator and F is a forcing function.

Definition 2.1.1 (Well-posedness)**.** The problem (2.1) is well-posed if there exists a unique solution, and the following estimate can be achieved:

$$||u||_I \leq K(||f||_{II} + ||g||_{III} + ||F||_{IV})$$

where $||\cdot||_I, ||\cdot||_{II}, ||\cdot||_{III}, ||\cdot||_{IV}$ are chosen norms, often decided to be the L^2-norm, and K is independent of g, f and F.

The estimate makes sure that small changes to the applied data only leads to small changes in the solution. This can be observed by considering a small change in the input data δf, δg and δF and inserting this into the problem in Equation (2.1), so that it becomes:

$$\begin{aligned} \partial_t v &= Pv + F + \delta F \\ Bv &= g + \delta g \\ v &= f + \delta f,\ t = 0 \end{aligned}$$

Subtracting this from (2.1), we get the following problem:

$$\begin{aligned} \partial_t w &= Pw + \delta F \\ Bw &= \delta g \\ w &= \delta f,\ t = 0 \end{aligned}$$

where $w = u - v$, because $Pu - Pv = P(u - v) = Pw$ and $Bu - Bv = B(u - v) = Bw$ for linear operators P and B. Thus from the definition of well-posedness, if the PDE is stable, we get the estimate:

$$||w||_I \leq K(||\delta f||_{II} + ||\delta g||_{III} + ||\delta F||_{IV})$$

Which means that the changes in the solution is bounded by the small changes in the initial data times a constant K.

2.2 Stability

When solving a PDE numerically, we need to discretize the domain so that it is divided into finitely many points on which we can calculate the solution. There are several ways to do this. Different coordinate systems can be used, such as Cartesian, polar or cylindrical, or the domain can be divided into triangles, or other geometrical shapes. For a cartesian grid in one dimension, we partition the domain into m points $x_0, x_1, ..., x_m$. Let $h_i = x_{i+1} - x_i$ be the distance between the points. If $h_i = h$ for all i, the grid is uniform. We call h the mesh spacing. In two dimensions we discretize in both x- and y-direction to get a two-dimensional grid. Discretization in time works similarly to the discretization of one dimensional space, we divide the time interval into n points: $t_n = kn$.

When approximating a solution of a PDE-problem with a difference operator, we need to know that the discrete problem is stable. This means, as with the continuous case, that the system will converge to the solution even when there are small disturbances in the input data. The stability definition for the discrete problem is similar to the continuous case.

Consider the semi-discrete problem:

$$\begin{aligned} \partial_t u_i &= Qu_i + F_i \\ Bu &= g(t) \\ u &= f_i \\ i &= 1, 2, ..., m \end{aligned} \tag{2.2}$$

where Q is a finite volumes operator or difference operator, and B is a boundary operator. From [Gus07] we have the following definition:

Definition 2.2.1 (Stability)**.** The problem (2.2) is strongly stable if there exists a unique solution satisfying

$$||u(t)||_h^2 \leq Ke^{\alpha t}\Big(||f||_h^2 + \int_0^t ||F(\cdot, \tau)||_h^2 + |g|^2 dt\Big) \tag{2.3}$$

where K and α are independent of g,f and F.

Here $||u||_h^2 = \sum_{i=0}^m u^2 h$ is the discrete norm.

2.3 Consistency

When using a numerical scheme to approximate a PDE, the scheme needs to satisfy one more property in order to converge, and that is consistency. To inspect the consistency of the scheme, we must look at the truncation error. We will continue to write in one dimension, and the extension to two dimensions is similar. Consider the initial-boundary value problem in Equation (2.1) and the semi-discrete problem in Equation (2.2). Inserting the exact solution from Equation (2.1) into the scheme in Equation (2.2), we get the following:

$$\begin{aligned} u(x_i, t_{n+1}) &= Qu(x_i) + kT(x_i, t_n) \\ u(x_i, 0) &= f_i + \phi_i \\ u(0, t_n) &= g_n + \psi_n \end{aligned} \tag{2.4}$$

where $T(x_i, t_n)$ is the truncation error, and ϕ_i and ψ_n are the errors in the initial and boundary data. The truncation error is found by Taylor-expanding the scheme around the point (x_i, t_n). By doing this we get the equation $u(x_i, t_{n+1}) = Qu(x_i, t_n) + kT(x_i, t_n)$, where the truncation error is multiplied by the factor k. In order for the scheme to converge towards the solution $u(x_i, t_n)$, we must make sure that the error $T(x_i, t_n)$ goes to zero as $h, k \to 0$. For this to happen, the error must depend on h and k:

Definition 2.3.1 (Order of accuracy)**.** Let $T(x_i, t_n)$ be the truncation error of the scheme in Equation (2.2), and let

$$|T(x_i, t_n)| \leq K(h^p + k^q) \tag{2.5}$$

where K is independent of h and k. Then (p, q) is the order of accuracy.

Definition 2.3.2 (Consistency)**.** The scheme in (2.2) is consistent if in Equation (2.5), $p \geq 0$ and $q \geq 0$.

Together with stability, consistency gives a convergent scheme, which is stated in the following theorem from [RM67]:

Theorem 2.3.1 (Lax's equivalence theorem)**.** If the PDE problem in Equation (2.1) is well-posed and the scheme in Equation (2.3) is consistent, then the numerical approximation converges to the PDE problem if and only if it is stable.

2.4 The Finite volume method

In this thesis we will be looking at hyperbolic PDEs, and the functions we will use are conservative. The Finite Volume method is designed to have this property, and so it is a suitable choice. We start with the following conservation law:

$$u_t + f_x(u) + g_y(u) = 0 \tag{2.6}$$

Consider a uniformly discretized, two-dimensional Cartesian domain with the discretization $x_i = ih, y_j = jh$. To begin with, the domain is divided into control volumes, and the solution is calculated inside each of these volumes before they are summed up. There are two ways to do this. In the first case, the volumes are located around the nodes of the grid, and the solution is calculated at these nodes in the center of the volume. This is illustrated in Figure 2.1. In this case, the volumes and the grid lines are not coinciding. The second way is to calculate the solution inside the cells on the grid, making the center of the volume a point in the middle of the grid-cell, instead of at the node. Here, the control volumes coincides with the grid lines. This is shown in figure 2.2. It is slightly more difficult to calculate the boundary values in this case, as the boundary points and the cell centers does not overlap. The last nodes at the edges of the grids are shifted from the boundary of the grid a distance of $\frac{h}{2}$. In this explanation we will use the first approach, the one illustrated in Figure 2.1, but the two versions of the FVM are similar, and gives the same results. We are following the explanation from Chapter 12.4 in [Gus07].

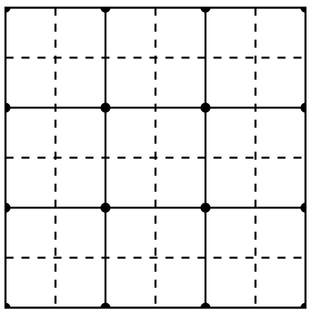
Figure 2.1: Illustration of node-centered finite volumes. The cells are drawn around the nodes of the grid.

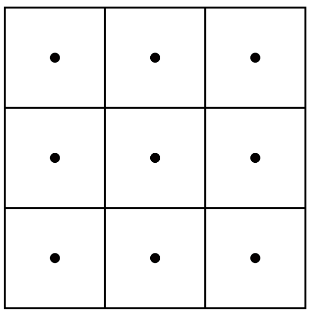
Figure 2.2: Illustration of cell-centered finite volumes. The volumes are the grid-cells, and nodes are drawn inside the volumes to mark the center of the cell.

Since the grid is uniform, the area of the control volumes is $|V_{i,j}| = h^2$. For each point (x_i, y_j) on the grid, the control volume $V_{i,j}$ is defined by the two intervals $I_i = [x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}]$ and $I_j = [y_{j-\frac{1}{2}}, y_{j+\frac{1}{2}}]$. We calculate the average of the solution u in each volume by evaluating the flux that flows in and out of the volume. Thus we are approximating Equation (2.6) on integral form inside the control volume $V_{i,j}$:

$$\iint_{V_{i,j}} u_t \, dx \; dy = \iint_{V_{i,j}} (f_x + g_y) \, dx \; dy$$

by Green's Theorem $\iint_{V_{i,j}} (f_x + g_y) \, dx \; = \int_{I_j} f \, dy - \int_{I_i} g \, dx$, and the equation becomes:

$$\int_{V_{i,j}} u_t \, dx \; dy = \int_{I_j} f \, dy - \int_{I_i} g \, dx \tag{2.7}$$

We approximate the integral of u_t over $V_{i,j}$ by multiplying with the area of the volume: $|V_{i,j}| u_t$. As for the right hand side, the flux through the volume in x-direction is given by:

$$(f_{i+\frac{1}{2}} - f_{i-\frac{1}{2}}) \; h \tag{2.8}$$

where $f_{i+\frac{1}{2}} = f(u(x_{i+\frac{1}{2}}))$ is the flux over the edge at $x_{i+\frac{1}{2}}$ and $f_{i-\frac{1}{2}} = f(u(x_{i-\frac{1}{2}}))$ is the flux over the edge at $x_{i-\frac{1}{2}}$, and both are multiplied by the length of the edge they are crossing. The y-direction is similar:

$$(g_{j+\frac{1}{2}} - g_{j-\frac{1}{2}})\ h$$

Since the numerical solution is not known at point $x_{i+\frac{1}{2}}$, we approximate the solution by using the values from the nodes on each side of $x_{i+\frac{1}{2}}$, that is, in x_{i+1} and x_{i-1}. This is done by using the following average:

$$f_{i+\frac{1}{2}} = \frac{f_{i+1} + f_i}{2}$$

Inserting this into (2.8), we get:

$$\Big(\frac{f_{i+1} + f_i}{2} - \frac{f_i + f_{i-1}}{2}\Big)h = \Big(\frac{f_{i+1} - f_{i-1}}{2}\Big)h$$

And the same for the g-flux. With this we get the following approximation of Equation (2.7):

$$|V_{i,j}|\ u_t = \Big(\frac{f_{i+1} - f_{i-1}}{2}\Big)h + \Big(\frac{g_{j+1} - g_{j-1}}{2}\Big)h$$

Dividing by the area of the control volume, we arrive at the final scheme:

$$u_t = \frac{f_{i+1} - f_{i-1}}{2h} + \frac{g_{j+1} - g_{j-1}}{2h} \tag{2.9}$$

2.5 Artificial diffusion

In some cases when the instabilities of the approximation becomes too big, it might help to modify the problem by adding artificial diffusion. In this section we will look at how this is done and what effect it gives to the solution. The following is obtained from Chapter 11 of [Gus07].

Consider the conservation law:

$$u_t + f_x(u) = 0 \tag{2.10}$$

When working with conservative PDEs, shocks (see Section 3.2.1) often occur. In such cases the approximation will fail, because the solution becomes discontinuous. One way to avoid this is to modify (2.10) by adding a small diffusion term

λu_{xx} to the left side of the equation, so that the equation becomes:

$$u_t + f_x(u) = \lambda u_{xx}$$

With this term, the conservation law is no longer conservative because the diffusion term creates a loss of energy. The conservation law is however approximated by letting $\lambda \to 0$. When choosing a sufficiently small diffusion coefficient, the approximation achieves some smoothing but not enough to change the solution entirely.

The diffusion term can be approximated numerically. To do this we write the flux in the following way:

$$f_{i+\frac{1}{2}} = \frac{f_{i+1} + f_i}{2} - \lambda \frac{u_{i+1} - u_i}{2} \tag{2.11}$$

When λ is constant, the scheme at a point x_i becomes

$$\begin{aligned} f_{i+\frac{1}{2}} - f_{i-\frac{1}{2}} =& \Big(\frac{f_{i+1} + f_i}{2} - \lambda \frac{u_{i+1} - u_i}{2}\Big) \\ &- \Big(\frac{f_i + f_{i-1}}{2} - \lambda \frac{u_i - u_{i-1}}{2}\Big) \\ =& \frac{f_{i+1} + f_i}{2} - \frac{f_i + f_{i-1}}{2} \\ &- \Big(\lambda \frac{u_{i+1} - u_i}{2} - \lambda \frac{u_i - u_{i-1}}{2}\Big) \\ =& \frac{f_{i+1} - f_{i-1}}{2} - \lambda \frac{u_{i+1} - 2u_i + u_{i-1}}{2} \end{aligned} \tag{2.12}$$

Here we can see that the term $-\lambda \frac{u_{i+1} - 2u_i + u_{i-1}}{2}$ approximates the second derivative $-\lambda u_{xx}$. This diffusion term is called artificial diffusion (or artificial viscosity).

Sometimes when faced with an unstable numerical method, artificial viscosity can be added to achieve the same smoothing effect as for the shock and this can lead to stability for the method. We will return to this later in the thesis.

2.6 Runge-Kutta

The theory in this section is obtained from [KW93] and Chapter 5.2 in [Gus07]. In this thesis we have used 4th order Runge-Kutta to discretize in time. Runge-Kutta is an ODE-solver, and the general method is expressed the following way:

$$\begin{aligned} u(t+k) &= L(kQ)u(t) + kG \\ u(0) &= f \end{aligned} \tag{2.13}$$

where $L(kQ) = \sum_{j=0}^{q} \alpha_j \frac{(kQ)^j}{j!}$. If we write out the method in Equation (2.13) with $q = 4$, we get the fourth order Runge-Kutta which is the following:

$$\begin{aligned} k_1 &= F(t_n, u^n) \\ k_2 &= F(t_n + \frac{1}{2}k, u^n + \frac{1}{2}kk_1) \\ k_3 &= F(t_n + \frac{1}{2}k, u^n + \frac{1}{2}kk_2) \\ k_4 &= F(t_n + k, u^n + kk_3) \\ u^{n+1} &= u^n + k(\frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4) \end{aligned}$$

We will mostly analyze semi-discrete schemes in this thesis, but we will now show that when using 4th order Runge-Kutta in time, the stability will follow when the semi-discrete scheme is stable.

First, some definitions and assumptions will be presented from [KW93]. We consider the method applied to the scalar ODE $y_t = \lambda y$:

$$u(t+k) = L(k\lambda)u(t)$$

Definition 2.6.1 (Stability domain)**.** We define the stability domain, Ω, for the Runge-Kutta method as

$$\Omega := \{\mu = k\lambda \in \mathbb{C} \mid |L(\mu)| \leq 1\} \tag{2.14}$$

Definition 2.6.2 (Locally Stable)**.** A method is locally stable if there exists a radius $R \in \mathbb{C}$ such that

$$\{\mu \in \mathbb{C} \mid L(\mu) \leq R\} \subset \Omega \tag{2.15}$$

Now, from [KS92] we state Theorem 4.1 about the local stability of 4th order Runge-Kutta:

Theorem 2.6.1. If q = m, then the Runge-Kutta method is unstable for m = 1,2 and stable for m = 3,4.

for the method expressed in Equation (2.13). Thus, we arrive at the following theorem from [KW93]:

Theorem 2.6.2. Let the following points be satisfied:

- The method is locally stable
- Let $\mu \in \mathbb{C}, Re(\mu) = 0, |\alpha| \leq R$ be a simple root of the equation $L(\mu) = e^{i\alpha}$. Then there exists no other number β such that $|\mu| \leq R$ $\mu = i\beta$.
- The semi-discretization is stable in a generalized sense

Then the fully discrete appriximation is stable in the genealized sense if

$$||kQ||_h \leq R_1 < R \tag{2.16}$$

Since we know that the Runge-Kutta 4 is locally stable, we need only to consider the two other points in the theorem. Recall that $L(\mu) = 1 + \mu + \frac{\mu^2}{2} + \frac{\mu^3}{6} + \frac{\mu^4}{24}$ for the 4th order method. For the equation

$$1 + \mu + \frac{\mu^2}{2} + \frac{\mu^3}{6} + \frac{\mu^4}{24} = e^{i\alpha}$$

where $e^{i\alpha} = \cos\alpha + i\sin\alpha$, $\mu = 0$ is a unique simple root [Gus07]. Thus, whenever we present a semi-discrete approximation which is stable in a generalized sense, we know that when using Runge-Kutta 4 to discretize in time, it follows that the fully discrete approximation is also stable in the same manner for $||kQ||_h \leq R_1 < R$.

Chapter 3

Advection equation and Euler equations

In [BO84] Marsha Berger and Joseph Oliger writes about the Adaptive Mesh refinement for hyperbolic PDEs. In this thesis, we will use hyperbolic PDEs, and more specifically they will be conservative. In this chapter we will present the PDEs which we will use and refer to later in the thesis. All the numerical experiments are done in two dimensions, and we will therefore present the equations in 2D-form. If possible, we will show well-posedness.

3.1 The advection equation

The two-dimensional advection equation with constant coefficients is presented in the following equation:

$$\begin{aligned} u_t + au_x + bu_y &= 0 \\ u &\in \Omega \end{aligned} \tag{3.1}$$

the advection equation is also called the transport equation, and it describes the transportation of a substance in a fluid flow. It is one of the simplest conservative PDEs.

3.1.1 Well posedness for Advection equation

We want to show that the initial boundary value problem

$$\begin{aligned} u_t + u_x + u_y &= 0, \quad 0 \le x, y \le 1,\ 0 \le t \\ u([x,0],t) &= g_1(x,t) \\ u([0,y],t) &= g_2(y,t) \\ u([x,y],0) &= f(x,y) \end{aligned} \tag{3.2}$$

is well posed. For simplicity we are showing this for Equation (3.1) with $a = b = 1$, and this is the equation we have used in the numerical experiments as well. First of all we define the L^2-norm:

$$||u||_2 = \left(\iint_\Omega u^2 \, dx \right)^{1/2} \tag{3.3}$$

We use the Energy method to show stability. Starting with the equation

$$u_t + u_x + u_y = 0$$

we multiply the equation with the solution u and integrate over the domain:

$$\iint_\Omega u_t u \; dx\, dy + \iint_\Omega u_x u \; dx\, dy + \iint_\Omega u_y u \; dx\, dy = 0$$

Noticing that $\partial_t(u)u = \partial_t(\frac{u^2}{2})$, we get:

$$\iint_\Omega (\frac{u^2}{2})_t \; dx\, dy + \iint_\Omega (\frac{u^2}{2})_x \; dx\, dy + \iint_\Omega (\frac{u^2}{2})_y \; dx\, dy = 0$$

Integrating the last two terms, this becomes:

$$\begin{aligned} \frac{1}{2}(||u||_2^2)_t + \int_{y=0}^{y=1} \frac{1}{2}(u^2(1,y,t) - u^2(0,y,t)) \; dy \\ + \int_{x=0}^{x=1} \frac{1}{2}(u^2(x,1,t) - u^2(x,0,t)) \; dx = 0 \end{aligned}$$

And we can insert the boundary functions:

$$\begin{aligned}\frac{1}{2}(||u||_2^2)_t + \int_{y=0}^{y=1} \frac{1}{2}(u^2(1,y,t) - g_2^2(y,t))\ dy \\ + \int_{x=0}^{x=1} \frac{1}{2}(u^2(x,1,t) - g_1^2(x,t))\ dx = 0\end{aligned}$$

Arriving at the equation:

$$\begin{aligned}\frac{1}{2}(||u||_2^2)_t = &- \frac{1}{2}\int_{y=0}^{y=1} u^2(1,y,t)\ dy + \frac{1}{2}\int_{y=0}^{y=1} g_2^2(y,t)\ dy \\ &- \frac{1}{2}\int_{x=0}^{x=1} u^2(x,1,t)\ dx + \frac{1}{2}\int_{x=0}^{x=1} g_1^2(x,t)\ dx\end{aligned}$$

Since $-\frac{1}{2}\int_{y=0}^{y=1} u^2(1,y,t)\ dy \leq 0$ and $-\frac{1}{2}\int_{x=0}^{x=1} u^2(x,1,t)\ dx \leq 0$ we get the inequality:

$$(||u||_2^2)_t \leq \int_{y=0}^{y=1} g_2^2(y,t)\ dy + \int_{x=0}^{x=1} g_1^2(x,t)\ dx$$

Integrating in time:

$$\int_0^t (||u||_2^2)_t\ dt \leq \int_0^t \Big(\int_{y=0}^{y=1} g_2^2(y,t)\ dy + \int_{x=0}^{x=1} g_1^2(x,t)\ dx \Big)\ dt$$

$$||u([x,y],t)||_2^2 - ||u([x,y],0)||_2^2 \leq \int_0^t \Big(\int_{y=0}^{y=1} g_2^2(y,t)\ dy + \int_{x=0}^{x=1} g_1^2(x,t)\ dx \Big)\ dt$$

$$||u([x,y],t)||_2^2 \leq ||f(x,y)||_2^2 + \int_0^t \Big(\int_{y=0}^{y=1} g_2^2(y,t)\ dy + \int_{x=0}^{x=1} g_1^2(x,t)\ dx \Big)\ dt$$

From Definition (2.3) of well-posedness we conclude that the 2D advection equation is well-posed.

3.2 The Euler equations

Next, we will look at the two-dimensional Euler equations:

$$\begin{aligned}
&\frac{\partial \rho}{\partial t} + \nabla(\rho \vec{v}) = 0 \\
&\frac{\partial \vec{v}}{\partial t} + \rho \vec{v} \cdot \nabla \vec{v} + \nabla p = 0 \\
&\frac{\partial E}{\partial t} + \nabla((E+p)\vec{v}) = 0
\end{aligned} \tag{3.4}$$

where ρ is the density, $\vec{v} = [v_1, v_2]^T$ are the two velocity components, and E is the energy. We want to derive the first equation which is called the continuity equation, and the following derivation is obtained from [And03]. The continuity equation is derived from the mass conservation law, which states that the time rate of change of mass inside a control volume is equal to the net flow of mass into the control volume. Let $\mathcal{V}$ be a control volume with surface S. Let dS be a elemental surface area on S, and let $d\vec{S} = \vec{n} dS$ where $\vec{n}$ is the normal vector on dS pointing out of the control volume. The mass flow through the elemental area dS is given by:

$$\rho \vec{v} \cdot d\vec{S}$$

and the mass flow into the control volume is given by the integral over the surface:

$$-\iint_S \rho \, \vec{v} \cdot d\vec{S}$$

which is negative because it flows in opposite direction of $\vec{v}$. Let $d\mathcal{V}$ be an elemental volume inside the control volume. The mass inside this volume is given by:

$$\rho \, d\mathcal{V}$$

The time rate of change of the entire control volume is then given by the time rate of change of the integral over $\mathcal{V}$:

$$\frac{\partial}{\partial t} \iiint_{\mathcal{V}} \rho \, d\mathcal{V}$$

Thus the mass conservation law gives us the following continuity equation:

$$\frac{\partial}{\partial t} \iiint_{\mathcal{V}} \rho \, d\mathcal{V} = -\iint_S \rho \, \vec{v} \cdot d\vec{S} \tag{3.5}$$

written on conservation form. Written on the same form as in equation (3.4), we use the divergence theorem and get:

$$\frac{\partial}{\partial t}\iiint_{\mathcal{V}} \rho \, d\mathcal{V} = -\iiint_{\mathcal{V}} \nabla(\rho \, \vec{v}) d\mathcal{V}$$

Since this equation holds for all coltrol volumes in the fluid, it holds for the entire fluid. Integrating over a unit area, we get the equation:

$$\frac{\partial \rho}{\partial t} + \nabla(\rho \, \vec{v}) = 0$$

The other two equations, conservation of momentum and conservation of energy, are derived similarly and a detailed derivation can be found in [And03].

3.2.1 Shock and rarefaction waves

When observing certain PDEs, we can sometimes encounter solutions that are discontinuous. The discontinuity implies that there is an instant change in some material quantity, for example density, and this is called a shock wave. As an example from Chapter 3 in [Ach91], for a sound wave with finite amplitude the velocity is not constant along the wave, resulting in some parts of the wave catching up with the other, creating a discontinuity. Physically, each side of this shock is separated by a thin layer of molecules, in which there is a rapid change in density and velocity, and it is therefore described as discontinuous. This layer exists due to the viscosity in the fluid, which is keeping the wave from breaking down. The two sides of the shock, state 1 and 2, are related by the Rankine-Hugoniot relations from Chapter 15.4 in []:

$$\begin{aligned}
\rho_1 u_1 &= \rho_2 u_2 \\
p_1 - p_2 &= \rho_1 u_1^2 + \rho_2 u_2^2 \\
h_1 + \frac{1}{2}u_1^2 &= h_2 + \frac{1}{2}u_2^2
\end{aligned}$$

where the four variables density ρ, velocity u, pressure p and enthalpy h are given on both side of the shock, in state 1 and 2.

Consider the same sound wave as mentioned above. While in a shock wave, the change in density is positive, $\Delta\rho > 0$, a rarefaction wave is when the density is suddenly reduced, that is $\Delta\rho < 0$ (from [And03]). A rarefaction wave can sometimes occur after a shock wave, as we will see later in one of the numerical experiments, Section 4.5.3.

Mathematically this gives some trouble when modelling the problem with non-viscous theory. When approximating a shock numerically, the numerical solution will break down. In such cases a small amount of artificial diffusion is added to the equation, as explained in Section 2.5, in order to avoid that the solution becomes discontinuous. The mathematical theory of shocks and rarefaction waves solutions of PDEs is complicated and further explanations of how to deal with this mathematically can be found in Chapter 3.4 of [Eva10].

Chapter 4

Adaptive mesh refinement

In this chapter we present the Adaptive Mesh Refinement (AMR) method introduced in [BO84], [BC89] and [Ber86]. The idea behind the AMR method is to create a dynamic computational domain which adapts to the PDE-problem and adds precision to the parts of the domain that needs it. This is done by refining the mesh of these particular parts during the computations. This way the resolution is kept without covering the entire domain with the refined grids, which saves computational time. We will go into more detail about how AMR works and how it has been used in this thesis. We start by introducing mesh refinement, and different ways to do this. Later, we continue to the Adaptiveness, and we will end up with a general description of the AMR method.

4.1 Mesh refinement with finite volumes

The Adaptive Mesh refinement method was first introduced in [BO84], where the finite difference scheme was used. Another approach was presented in [BC89] with the finite volume scheme. In this section we will explain how we have recreated the method based on the explanations from the original article [BO84], but we have used a coupled finite volumes scheme instead of the finite difference scheme. We will call this Method A. We have tried to recreate the method as closely as possible to the way it was written in [BO84]. This has resulted in a method using the node centered finite volume scheme explained in Section 2.4. Later, two alternative mesh refinements are presented, one of which is created with the modifications from [BC89]. We will now go into detail about the different computations and implementations that have been done.

4.1.1 Grid structure

The notation in this section is inherited from [BO84]. We start by discretizing the domain. In this thesis all grids will be uniform Cartesian grids. Uniform means that all the points on the grid are separated by the same distance h, in both x- and y-direction. The method can be generalized to non-uniform meshes. Thus in two dimensions with m_x points in x-direction and m_y points in y-direction we get:

$$\begin{aligned} x_i &= h\,i, \quad i = 0, 1, ..., m_x \\ y_j &= h\,j, \quad j = 0, 1, ..., m_y \end{aligned}$$

Let G_0 denote the set of grids at the initial level with mesh spacing h_0. A grid level indicates the level of refinement. At level 0, the grids have the initial mesh spacing. An initial grid may be covered by one or more patches of refined grids at any subset in need of it. These patches of refined grids might not give enough accuracy, and new refinments of these can be created in the same way. This results in a grid structure of one or more initial grids with patches of refined grids covering parts of the initial grids, which in turn has its own patches of subgrids. A refinement of one of the initial grids in G_0 will belong to the next level of grids, in the set G_1. At this level the mesh spacing is given by $h_1 = \frac{h_0}{r}$, where r is the refinement ratio. This continues, and for each grid in G_i a refinement will belong in G_{i+1} at level $i + 1$, with mesh spacing $h_{i+1} = \frac{h_i}{r^i}$. The different grids in each level are denoted $G_{i,j}$ where j indicates the specific grid in question, so that $G_{i,j} \in G_i$. To keep track of this grid hierarchy in the simulations, we do as in [BO84] and [Ber86] and use linked lists. Inherited from the terminology of linked lists, a parent grid of a grid $G_{i,j}$, is the coarse grid which $G_{i,j}$ is a refinement of. A child of a grid $G_{i,j}$ is then the refinement. Each grid is a node in the list, and contains a link to the underlying coarse grid which it is a refinement of, a link to its children, and a link to any potential siblings which are other grids at the same level. The node

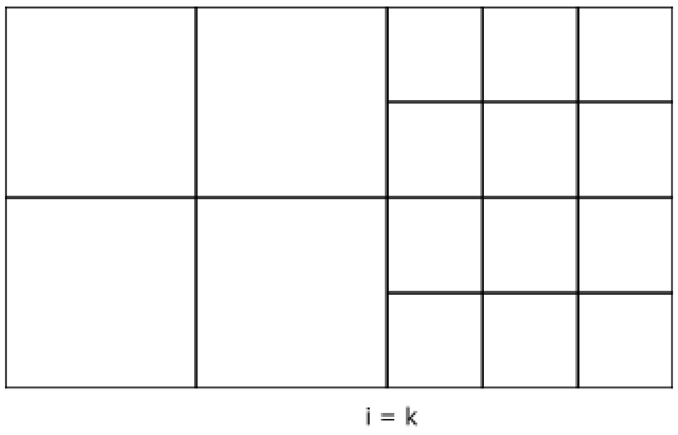

Figure 4.1: Mesh with refinement of ratio $r = 2$ to the right of i.

also contains information about the location of the grid in the domain, the mesh spacing h_i, the time step k_i, the time t to which the grid was last updated, number of points in the grid and the solution vector for the grid. The structure is a clever way to keep track of the grids and to make the method recursive.

4.1.2 Interpolation of boundary points

According to [BO84], the boundaries of the refined grids are calculated by interpolation on the underlying coarse grid points. For simplicity we will assume for the rest of this chapter that we only have one initial grid, and denote it G_0. Let G_0 have a refined subgrid with refinement r. With the node centered finite volume scheme introduced earlier, the boundary of the refined grid contains the same points as the underlying coarse grid, but with $r - 1$ additional points in between two coarse points as shown in Figure 4.1.

Using linear interpolation, the boundary values g^{fine} of the refined grid along the interface $i = k$ are calculated the following way:

$$\begin{aligned} g^{fine}_{k,j} &= u^{coarse}_{k,j} \\ g^{fine}_{k,j+\frac{1}{2}} &= \frac{u^{coarse}_{k,j+1} + u^{coarse}_{k,j}}{2} \end{aligned}$$

And so at the boundary of the refined subgrid, the scheme looks like the following:

$$\begin{aligned} u_{k,j} &= \frac{f(u_{k+1,j}) - f(g_{k,j}^{fine})}{h} \\ u_{k,j+\frac{1}{2}} &= \frac{f(u_{k+1,j}) - f(g_{k,j+\frac{1}{2}}^{fine})}{h} \end{aligned} \tag{4.1}$$

where u is the solution and h is the mesh spacing at the fine grid. When doing so, we end up with a coupled finite volume scheme where each grid has its own scheme, and the schemes are connected through the computations at the boundaries of the refinements.

In the simulations done in this thesis the initial data is known and since there is no adaption, no additional refined grid will appear during the computations. Thus, the fine grids are created from the start and are initialized by the same data as the coarse initial grid. However, if adaption is implemented, the refined grids could need initializing at an arbitrary time and there might not be an exact solution to use for this. In such cases interpolation is again used to fill in the solution vector of the refined grid, and this is done in a similar way as for the boundary conditions.

4.2 Stable mesh refinement

When working with methods such as AMR, where one changes the structure of a uniform mesh, it is natural to question whether the method is stable. It is not easy to show stability for this mesh refinement because of the interpolation. However, in [Ber85] Marsha Berger has been able to show stability for the mesh refinement method by adding artificial diffusion.

The question of stability for the mesh refinement gives motivation to make some small adjustments to the method in order to make it provably stable. This can be done by refining the grid in a different manner. The idea behind this approach is to create proper control volumes around the points at the interfaces to enable the use of a finite volume scheme. Then, by using values from both the fine and coarse grids, we can calculate the flux through the volumes instead of using interpolation.

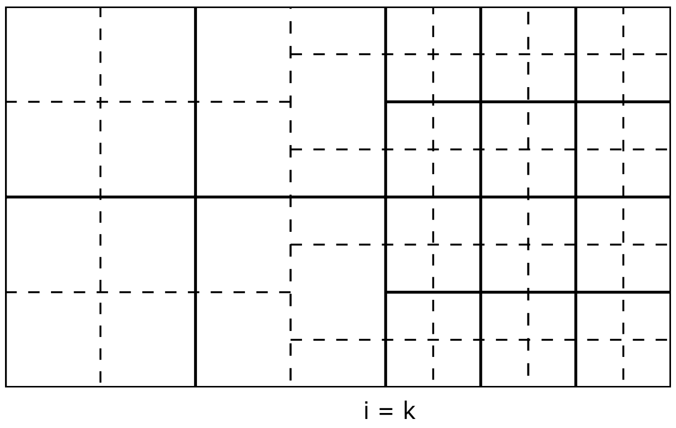

Figure 4.2: Mesh with volumes (dashed lines) and refinement of ratio $r = 2$ to the right of i=k.

Using the finite volumes scheme in Equation (2.7) on the uniform grids as before, we need to make some modifications at the interface from the way it is calculated in Method A. In the following, we will refine by a factor of two, so $r = 2$. Consider a 2D grid with a refinement at $i = k$ as shown in Figure 4.2 with the volumes drawn around the nodes. Let h be the mesh spacing at the fine grid. For volumes from the coarse grid, the mesh spacing then becomes $2h$. For Method A the interface is calculated as shown in Equation (4.1). We must modify the scheme at the interface, along $i = k$, and at the points to the left of the interface, along $i = k-1$, to connect the flux through the volumes at the coarse and fine grid. The changes are explained using Figure 4.2 which has refinement in x-direction so that only the f-flux is affected. For refinement in the y-direction the same changes are done to the g-flux. There are three cases in which the scheme needs to be changed. The first one is for all points at the coarse side of the interface, at position $i = k - 1$. The semi-discretization of these points should be given by:

$$\begin{aligned}(u_{k,j})_t + \frac{(\frac{1}{2}f_{k-\frac{1}{2},j} + \frac{1}{4}f_{k-\frac{1}{2},j+\frac{1}{2}} + \frac{1}{4}f_{k-\frac{1}{2},j-\frac{1}{2}}) - f_{k-\frac{3}{2},j}}{2h} \\ + \frac{g_{k-1,j+\frac{1}{2}} - g_{k-1,j-\frac{1}{2}}}{2h} = 0\end{aligned} \tag{4.2}$$

The flux going into the refinement is really going into three different volumes as shown in Figure 4.2. Thus all three volumes from the fine grid needs to be taken into the computations.

The second change to the scheme is for the points located at the interface $i = k$ and overlapping a coarse grid point. This volume abuts a coarse volume to the left and a fine volume to the right. Thus the fluxes are the same as for the uniform parts of the grid, but the change in mesh size makes the volume a $\frac{3}{2}h \times h$ rectangle, and the mesh size in the scheme must be changed to:

$$(u_{i,j})_t + \frac{(f_{k+\frac{1}{2},j} - f_{k-\frac{1}{2},j})}{\frac{3}{2}h} + \frac{(g_{k,j+\frac{1}{2}} - g_{k,j-\frac{1}{2}})}{h} = 0 \tag{4.3}$$

Finally, for the points at $i = k$ lying in between the coarse grid points, at the refinement, the left edge of the volume borders to two different volumes from the coarse grid and the scheme changes to:

$$(u_{k,j})_t + \frac{f_{k,j} - \frac{1}{2}(f_{k,j+\frac{1}{2}} + f_{k,j-\frac{1}{2}})}{h} + \frac{g_{k,j+\frac{1}{2}} - g_{k,j-\frac{1}{2}}}{h} = 0 \tag{4.4}$$

With these modifications, there is no need for interpolation, as the "boundary points" are calculated through the fluxes instead, and we get a slightly different mesh refinement method. We call this Method B. The method does loose some of its simplicity as the subgrids needs a different integration method than the initial grids. Still, since these computations are only done at the boundaries of the fine grid, they do not require a lot of extra time compared to Method A.

4.2.1 Stability of Method A for advection equation

The modifications done to the mesh refinement method was made in order to achieve stability through calculations. Shown in this section is the stability proof of the discrete problem by the energy method.

The stability is only shown for the grid in Figure 4.2 where half of the grid is refined, and the interface is only along the y-direction, at $i = k$. The calculations for all other edges of the fine grid is similar. We have not included any boundaries in the calculations, but used infinite sums and assumed that it tends to zero. This is to make the computations as simple as possible, and to make the interface the main focus. Stability at the boundaries can be achieved as well, and in the experiments presented later we specify the particular boundary data used.

We start by summing over the domain. In y-direction there is no changes, but in x-direction the scheme is different for the four parts $i < k - 1$, $i = k - 1$, $i = k$ and $i > k$. They can be seen in Equations (4.5)-(4.8), and we consider these four parts of the domain separately:

- (4.5): the sum over all points for which $i < k - 1$
- (4.6): the sum of all points along the line $i = k - 1$
- (4.7): the sum of all points along the line $i = k$
- (4.8): the sum of all points for which $i > k - 1$

$$\sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty}\partial_t(\frac{u^2}{2})|V_{i,j}|$$

$$+\sum_{i=-\infty}^{k-2}\sum_{j=-\infty}^{\infty}\Big(a(u_{i+1,j}-u_{i-1,j})\frac{h}{2}+b(u_{i,j+1}-u_{i,j-1})\frac{h}{2}\Big)u_{i,j} \tag{4.5}$$

$$+\sum_{j=-\infty}^{\infty}\Big(a(\frac{1}{2}u_{k,j}+\frac{1}{4}u_{k,j+\frac{1}{2}}+\frac{1}{4}u_{k,j-\frac{1}{2}}-u_{k-2,j})\frac{h}{2} \tag{4.6}$$

$$+b(u_{k-1,j+1}-u_{k-1,j-1})\frac{h}{2}\Big)u_{k-1,j}$$

$$+\sum_{j=-\infty}^{\infty}\Big(\Big(a(u_{k+1,j}-u_{k-1,j})\frac{h}{4}+b(u_{k,j+\frac{1}{2}}-u_{k,j-\frac{1}{2}})\frac{3}{4}\frac{h}{2}\Big)u_{k,j} \tag{4.7}$$

$$\Big(a\big(u_{k+1,j+\frac{1}{2}}-\frac{1}{2}(u_{k-1,j}+u_{k-1,j+1})\big)\frac{h}{4}+b(u_{k,j+1}-u_{k,j})\frac{3}{4}\frac{h}{2}\Big)u_{k,j+\frac{1}{2}}\Big)$$

$$+\sum_{i=k+1}^{\infty}\sum_{j=-\infty}^{\infty}\Big(\Big(a(u_{i+1,j}-u_{i-1,j})\frac{h}{4}+b(u_{i,j+\frac{1}{2}}-u_{i,j-\frac{1}{2}})\frac{h}{4}\Big)u_{i,j} \tag{4.8}$$

$$\Big(a(u_{i+1,j+\frac{1}{2}}-u_{i-1,j+\frac{1}{2}})\frac{h}{4}+b(u_{i,j+1}-u_{i,j})\frac{h}{4}\Big)u_{i,j+\frac{1}{2}}\Big)$$

$$=0$$

First, for Equation (4.5) all points where $i < k-1$, all terms in y-direction will be cancelled, and the same for all terms in x-direction, except for the ones along $i = k-2$:

$$\sum_{i=-\infty}^{k-2}\sum_{j=-\infty}^{\infty}\left(a\left(u_{i+1,j}-u_{i-1,j}\right)\frac{h}{2}+b\left(u_{i,j+1}-u_{i,j-1}\right)\frac{h}{2}\right)u_{i,j}$$

$$=\sum_{i=-\infty}^{k-2}\sum_{j=-\infty}^{\infty}\Big(a\frac{h}{2}\,u_{i+1,j}\,u_{i,j}-a\frac{h}{2}\,u_{i-1,j}\,u_{i,j}+b\frac{h}{2}\,u_{i,j+1}\,u_{i,j}-b\frac{h}{2}\,u_{i,j-1}\,u_{i,j}\Big) \tag{4.9}$$

$$=\sum_{j=-\infty}^{\infty}a\frac{h}{2}\,u_{k-1,j}\,u_{k-2,j}$$

The same happens to the sum over the fine grid in Equation (4.8). All points for which $i > k$ disappears except for the terms along the line $i = k+1$ next to the interface:

$$\sum_{i=k+1}^{\infty}\sum_{j=-\infty}^{\infty}\Bigg(\Big(a(u_{i+1,j}-u_{i-1,j})\frac{h}{4}+b(u_{i,j+\frac{1}{2}}-u_{i,j-\frac{1}{2}})\frac{h}{4}\Big)u_{i,j}$$
$$\Big(a(u_{i+1,j+\frac{1}{2}}-u_{i-1,j+\frac{1}{2}})\frac{h}{4}+b(u_{i,j+1}-u_{i,j})\frac{h}{4}\Big)u_{i,j+\frac{1}{2}}\Bigg) \qquad (4.10)$$
$$=\sum_{j=-\infty}^{\infty}\Big(-a\frac{h}{4}\,u_{k,j}\,u_{k+1,j}-a\frac{h}{4}\,u_{k,j+\frac{1}{2}}\,u_{k+1,j+\frac{1}{2}}\Big)$$

Now for the remaining two sums along $i=k-1$ and $i=k$ in Equations (4.6) and (4.7), we get the following:

$$\sum_{j=-\infty}^{\infty}\Big(a(\frac{1}{2}u_{k,j}+\frac{1}{4}u_{k,j+\frac{1}{2}}+\frac{1}{4}u_{k,j-\frac{1}{2}}-u_{k-2,j})\frac{h}{2}+$$
$$b(u_{k-1,j+1}-u_{k-1,j-1})\frac{h}{2}\Big)u_{k-1,j}$$
$$+\sum_{j=-\infty}^{\infty}\Bigg(\Big(a(u_{k+1,j}-u_{k-1,j})\frac{h}{4}+b(u_{k,j+\frac{1}{2}}-u_{k,j-\frac{1}{2}})\frac{3}{4}\frac{h}{2}\Big)u_{k,j}$$
$$\Big(a\big(u_{k+1,j+\frac{1}{2}}-\frac{1}{2}(u_{k-1,j}+u_{k-1,j+1})\big)\frac{h}{4}+b(u_{k,j+1}-u_{k,j})\frac{3}{4}\frac{h}{2}\Big)u_{k,j+\frac{1}{2}}\Bigg)$$
$$=\sum_{j=-\infty}^{\infty}\Big(a\,\frac{1}{2}u_{k,j}u_{k-1,j}\frac{h}{2}+a\,\frac{1}{4}u_{k,j+\frac{1}{2}}u_{k-1,j}\frac{h}{2}+a\,\frac{1}{4}u_{k,j-\frac{1}{2}}u_{k-1,j}\frac{h}{2}$$
$$-a\,u_{k-2,j}u_{k-1,j}\frac{h}{2}+b\,u_{k-1,j+1}u_{k-1,j}\frac{h}{2}-b\,u_{k-1,j-1}u_{k-1,j}\frac{h}{2}\Big)$$
$$+\sum_{j=-\infty}^{\infty}\Bigg(\Big(a\,u_{k+1,j}u_{k,j}\frac{h}{4}-a\,u_{k-1,j}u_{k,j}\frac{h}{4}+b\;u_{k,j+\frac{1}{2}}u_{k,j}\frac{3}{4}\frac{h}{2}-b\;u_{k,j-\frac{1}{2}}u_{k,j}\frac{3}{4}\frac{h}{2}\Big)$$
$$+\Big(a\,u_{k+1,j+\frac{1}{2}}u_{k,j+\frac{1}{2}}\frac{h}{4}-a\,\frac{1}{2}u_{k-1,j}u_{k,j+\frac{1}{2}}\frac{h}{4}-a\,\frac{1}{2}u_{k-1,j+1}u_{k,j+\frac{1}{2}}\frac{h}{4}$$
$$+b\,u_{k,j+1}u_{k,j+\frac{1}{2}}\frac{3}{4}\frac{h}{2}-b\,u_{k,j}u_{k,j+\frac{1}{2}}\frac{3}{4}\frac{h}{2}\Big)\Bigg)$$

The b-terms are from the g-flux in y-direction, and they cancel when summing over the j component as we do here. This leads to:

$$\begin{aligned}
&\sum_{j=-\infty}^{\infty} \Big(a\,\frac{h}{4}\,u_{k,j}\,u_{k-1,j} + a\,\frac{h}{8}\,u_{k,j+\frac{1}{2}}\,u_{k-1,j} + a\,\frac{h}{8}\,u_{k,j-\frac{1}{2}}\,u_{k-1,j} \\
&\qquad - a\,\frac{h}{2}\,u_{k-2,j}\,u_{k-1,j}\Big) \\
&+ \sum_{j=-\infty}^{\infty} \Bigg(a\,\frac{h}{4}\,u_{k+1,j}\,u_{k,j} - a\,\frac{h}{4}\,u_{k-1,j}\,u_{k,j} + a\,\frac{h}{4}\,u_{k+1,j+\frac{1}{2}}\,u_{k,j+\frac{1}{2}} \\
&\qquad - a\,\frac{h}{8}\,u_{k-1,j}\,u_{k,j+\frac{1}{2}} - a\,\frac{h}{8}\,u_{k-1,j+1}\,u_{k,j+\frac{1}{2}}\Bigg) \\
&= \sum_{j=-\infty}^{\infty} \Big(a\,\frac{h}{8}\,u_{k,j-\frac{1}{2}}\,u_{k-1,j} - a\,\frac{h}{2}\,u_{k-2,j}\,u_{k-1,j}\Big) \\
&+ \sum_{j=-\infty}^{\infty} \Bigg(a\,\frac{h}{4}\,u_{k+1,j}\,u_{k,j} + a\,\frac{h}{4}\,u_{k+1,j+\frac{1}{2}}\,u_{k,j+\frac{1}{2}} - a\,\frac{h}{8}\,u_{k-1,j+1}\,u_{k,j+\frac{1}{2}}\Bigg)
\end{aligned}$$

The first and last term, $a\,\frac{h}{8}\,u_{(k,j-\frac{1}{2})}\,u_{(k-1,j)}$ and $-a\,\frac{h}{8}\,u_{(k-1,j+1)}\,u_{(k,j+\frac{1}{2})}$, will cancel as we sum along j, and we are left with:

$$\begin{aligned}
&\sum_{j=-\infty}^{\infty} \Big(-a\,\frac{h}{2}\,u_{k-2,j}\,u_{k-1,j}\Big) \\
&+ \sum_{j=-\infty}^{\infty} \Bigg(a\,\frac{h}{4}\,u_{k+1,j}\,u_{k,j} + a\,\frac{h}{4}\,u_{k+1,j+\frac{1}{2}}\,u_{k,j+\frac{1}{2}}\Bigg)
\end{aligned} \tag{4.11}$$

Collecting the resulting sums (4.9), (4.10) and (4.11) we observe that:

$$\begin{aligned}
&\sum_{j=-\infty}^{\infty} a\frac{h}{2}\,u_{k-1,j}\,u_{k-2,j} \\
&+ \sum_{j=-\infty}^{\infty} \Big(-a\,\frac{h}{2}\,u_{k-2,j}\,u_{k-1,j}\Big) \\
&+ \sum_{j=-\infty}^{\infty} \Bigg(a\,\frac{h}{4}\,u_{k+1,j}\,u_{k,j} + a\,\frac{h}{4}\,u_{k+1,j+\frac{1}{2}}\,u_{k,j+\frac{1}{2}}\Bigg) \\
&+ \sum_{j=-\infty}^{\infty} \Big(-a\frac{h}{4}\,u_{k,j}\,u_{k+1,j} - a\frac{h}{4}\,u_{k,j+\frac{1}{2}}\,u_{k+1,j+\frac{1}{2}}\Big) \\
&= 0
\end{aligned} \tag{4.12}$$

In conclusion, when replacing Equations (4.5)-(4.8) with (4.12) we get:

$$\sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty}\partial_t\Big(\frac{u^2}{2}\Big)|V_{i,j}| = \partial_t\frac{1}{2}||u||^2 = 0$$

which means that the energy is bounded and by Definition 2.2.1 the method is stable.

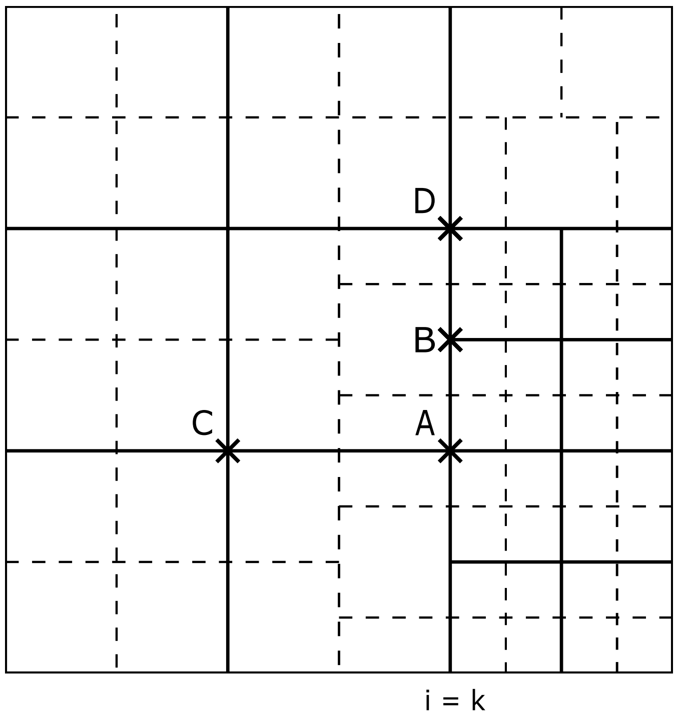

Figure 4.3: Grid with refinement to the right of $i = k$ and corner point.

4.2.2 Truncation error on interface at mesh refinement

We analyze the stable mesh refinement method further, and calculate the truncation error at the interface between the coarse and fine grid. There are four different points at the interface which we want to calculate, and these are marked as point A, B, C and D in Figure 4.3. It is important to notice while calculating the truncation error at the interface, that the mesh spacing changes on each side of the point. Starting with point A for example, the scheme looks like this:

$$(u_{i,j})_t + a\,\frac{u_{i+1,j}-u_{i-1,j}}{3h} + b\,\frac{u_{i,j+1}-u_{i,j-1}}{2h} = 0 \tag{4.13}$$

and the points x_{i+1} and x_i are separated by a length of h, while the points x_i and x_{i-1} by a length of $2\,h$ as seen in Figure 4.3. Taylor expanding the numerical solution around (x_i, y_j) for point A we get:

$$\begin{aligned}
&(u_{i,j})_t \\
&+ \frac{a}{3h}\Bigg(\Big(u + u_x(x_{i+1}-x_i) + \frac{1}{2}\,u_{xx}(x_{i+1}-x_i)^2 + \frac{1}{6}\,u_{xxx}(\zeta_1,\eta_1)(x_{i+1}-x_i)^3\Big) \\
&\qquad - \Big(u + u_x(x_{i-1}-x_i) + \frac{1}{2}\,u_{xx}(x_{i-1}-x_i)^2 + \frac{1}{6}\,u_{xxx}(\zeta_2,\eta_2)(x_{i-1}-x_i)^3\Big)\Bigg) \\
&+ \frac{b}{2h}\Bigg(\Big(u + u_y(y_{j+1}-y_j) + \frac{1}{2}\,u_{yy}(y_{j+1}-y_j)^2 + \frac{1}{6}\,u_{yyy}(\theta_1,\xi_1)(y_{j+1}-y_j)^3\Big) \\
&\qquad - \Big(u + u_y(y_{j-1}-y_j) + \frac{1}{2}\,u_{yy}(y_{j-1}-y_j)^2 + \frac{1}{6}\,u_{yyy}(\theta_2,\xi_2)(y_{j-1}-y_j)^3\Big)\Bigg) \\
&= 0
\end{aligned} \tag{4.14}$$

Next, since $(x_{i+1}-x_i) = h$ and $(x_{i-1}-x_i) = -2\,h$, and $(y_{j+1}-y_j) = h = -(y_{j-1}-y_j)$ in y-direction, the equation becomes:

$$\begin{aligned}
(u_i^j)_t &+ \frac{a}{3h}\Big(u + u_x\,h + \frac{1}{2}\,u_{xx}\,h^2 - u + 2\,u_x\,h - \frac{4}{2}\,u_{xx}\,h^2 + O(h^3)\Big)\Big) \\
&+ \frac{b}{2h}\Big(u + u_y\,h + \frac{1}{2}\,u_{yy}\,h^2 - u + u_y\,h - \frac{1}{2}\,u_{yy}\,h^2 + O(h^3)\Big)\Big) = 0
\end{aligned}$$

$$\begin{aligned}
(u_i^j)_t &+ \frac{a}{3h}\Big(3\,u_x\,h - \frac{3}{2}\,u_{xx}\,h^2 + O(h^3)\Big) \\
&+ \frac{b}{2h}\Big(2\,u_y\,h + O(h^3)\Big) = 0
\end{aligned}$$

$$(u_i^j)_t + a\,u_x + b\,u_y - \frac{a}{2}\,u_{xx}\,h + O(h^2) = 0$$

The computations shows that the truncation error is $\tau(x_i, y_j) = -\frac{a}{2}\, u_{xx}\, h + O(h^2)$ and that $|\tau(x_i, y_j)| \leq K(h)$ for a constant K, which means that the order of accuracy for point A is 1.

The volume around point A is easier to deal with then for the other points, as it only borders to one volume in each direction. The volume around the fine grid-point B however, borders to two different coarse volumes, as seen in Figure 4.3. Thus we must take into account that two different volumes contributes to the flux over the left edge. Writing out the scheme at point B we get:

$$(u_{i,j})_t + a\,\frac{u_{i+1,j} - \frac{1}{2}\left(u_{i-1,j+1} + u_{i-1,j-1}\right)}{3h} + b\,\frac{u_{i,j+1} - u_{i,j-1}}{2h} = 0 \tag{4.15}$$

For the g-flux in y-direction the scheme is the same as for point A, and the truncation error would be the same for B since it is not affected by the flux in x-direction. We will leave this part out of the calculations, and Taylor expand only in x-direction. This gives:

$$\begin{aligned}
\frac{a}{3h}\Bigg(&\Big(u + u_x(x_{i+1} - x_i) + \frac{1}{2}\,u_{xx}(x_{i+1} - x_i)^2 + \frac{1}{6}\,u_{xxx}(\zeta_1, \eta_1)(x_{i+1} - x_i)^3\Big) \\
&-\frac{1}{2}\Big(u + u_x(x_{i-1} - x_i) + u_y(y_{j+1} - y_j) \\
&\quad + \frac{1}{2}\left(u_{xx}(x_{i-1} - x_i)^2 + 2u_{xy}(x_{i-1} - x_i)(y_{j+1} - y_j) + u_{yy}(y_{j+1} - y_j)^2\right) \\
&\quad + \frac{1}{6}\left(u_{xxx}(\zeta_2, \eta_2)(x_{i-1} - x_i)^3 + 3u_{xxy}(\zeta_2, \eta_2)(x_{i-1} - x_i)^2(y_{j+1} - y_j)\right. \\
&\qquad \left. + 3u_{xyy}(\zeta_2, \eta_2)(x_{i-1} - x_i)(y_{j+1} - y_j)^2 + u_{yyy}(\zeta_2, \eta_2)(y_{j+1} - y_j)^3\right) \\
&\quad + u + u_x(x_{i-1} - x_i) + u_y(y_{j-1} - y_j) \\
&\quad + \frac{1}{2}\left(u_{xx}(x_{i-1} - x_i)^2 + 2u_{xy}(x_{i-1} - x_i)(y_{j-1} - y_j) + u_{yy}(y_{j-1} - y_j)^2\right) \\
&\quad + \frac{1}{6}\left(u_{xxx}(\zeta_3, \eta_3)(x_{i-1} - x_i)^3 + 3u_{xxy}(\zeta_3, \eta_3)(x_{i-1} - x_i)^2(y_{j-1} - y_j)\right. \\
&\qquad \left. + 3u_{xyy}(\zeta_3, \eta_3)(x_{i-1} - x_i)(y_{j-1} - y_j)^2 + u_{yyy}(\zeta_3, \eta_3)(y_{j-1} - y_j)^3\right)\Big)\Bigg)
\end{aligned} \tag{4.16}$$

As before, the refinement causes a change in the mesh spacing to the left and right of the point so that $(x_{i+1}-x_i)=h$, $(x_{i-1}-x_i)=-2h$ and $(y_{i+1}-y_i)=h=-(y_{i-1}-y_i)$, and we get:

$$\begin{aligned}&\frac{a}{3h}\Bigg(\Big(u+u_xh+\frac{1}{2}\,u_{xx}h^2+O(h^3)\Big)\\ &\quad-\frac{1}{2}\Big(u-u_x2h+u_yh+\frac{1}{2}\,(u_{xx}4h^2-2u_{xy}2h^2+u_{yy}h^2)+O(h^3)\\ &\qquad+u-u_x2h-u_yh+\frac{1}{2}\,(u_{xx}4h^2+2u_{xy}2h^2+u_{yy}h^2)+O(h^3)\Big)\Bigg)\end{aligned} \tag{4.17}$$

Which in turn becomes:

$$\begin{aligned}&\frac{a}{3h}\Bigg(u+u_xh+\frac{1}{2}\,u_{xx}h^2+O(h^3)\\ &\qquad-u+u_x2h-\frac{4}{2}\,u_{xx}h^2+u_{yy}h^2+O(h^3)\Bigg)\\ &=\frac{a}{3h}\Bigg(3u_xh-\frac{3}{2}\,u_{xx}h^2+u_{yy}h^2+O(h^3)\Bigg)\\ &=au_x-\frac{a}{2}\,u_{xx}h+\frac{a}{3}\,u_{yy}h+O(h^2)\end{aligned} \tag{4.18}$$

So we end up with the complete estimate:

$$(u_i^j)_t+au_x+b\,u_y-\frac{a}{2}\,u_{xx}h+u_{yy}h+O(h^2)=0$$

We observe that the truncation error is $\tau(x_i,y_j)=-\frac{a}{2}\,u_{xx}h+\frac{a}{2}\,u_{xxx}h^2+u_{yy}h-\frac{a}{3}\,u_{xyy}h^2+\frac{b}{6}\,u_{yyy}\,h^2+O(h^3)$, which means that the order of accuracy is 1 for this point as well.

Next we will look at point C. This point belong to the coarse grid, and the volume borders to one coarse volume to the left and three fine volumes to the right. The scheme is shown in the following equation:

$$(u_{i,j})_t+a\frac{(\frac{1}{2}\,u_{i+1,j}+\frac{1}{4}\,u_{i+1,j+\frac{1}{2}}+\frac{1}{4}\,u_{i+1,j-\frac{1}{2}})-u_{i-1,j}}{2h}+b\frac{u_{i,j+1}-u_{i,j-1}}{2h}$$

This point has the same distance to all its surrounding points, and we denote it by $2h$ as it is twice the size of h, the mesh spacing of the fine grid. For the g-flux, we get the same result as before. For the f-flux we get:

$$
\begin{aligned}
\frac{a}{2(2h)}\Bigg(&\frac{1}{2}\Big(u+u_x(x_{i+1}-x_i)+\frac{1}{2}\,u_{xx}(x_{i+1}-x_i)^2+\frac{1}{6}\,u_{xxx}(\zeta_1,\eta_1)(x_{i+1}-x_i)^3\Big)\\
&+\frac{1}{4}\Big(u+u_x(x_{i+1}-x_i)+u_y(y_{j+\frac{1}{2}}-y_j)\\
&\quad+\frac{1}{2}\,(u_{xx}(x_{i+1}-x_i)^2+2u_{xy}(x_{i+1}-x_i)(y_{j+\frac{1}{2}}-y_j)+u_{yy}(y_{j+\frac{1}{2}}-y_j)^2)\\
&\quad+\frac{1}{6}\,(u_{xxx}(\zeta_2,\eta_2)(x_{i+1}-x_i)^3+3u_{xxy}(\zeta_2,\eta_2)(x_{i+1}-x_i)^2(y_{j+\frac{1}{2}}-y_j)\\
&\qquad+3u_{xyy}(\zeta_2,\eta_2)(x_{i+1}-x_i)(y_{j+\frac{1}{2}}-y_j)^2+u_{yyy}(\zeta_2,\eta_2)(y_{j+\frac{1}{2}}-y_j)^3)\Big)\\
&+\frac{1}{4}\Big(u+u_x(x_{i+1}-x_i)+u_y(y_{j-\frac{1}{2}}-y_j)\\
&\quad+\frac{1}{2}\,(u_{xx}(x_{i+1}-x_i)^2+2u_{xy}(x_{i+1}-x_i)(y_{j-\frac{1}{2}}-y_j)+u_{yy}(y_{j-\frac{1}{2}}-y_j)^2)\\
&\quad+\frac{1}{6}\,(u_{xxx}(\zeta_3,\eta_3)(x_{i+1}-x_i)^3+3u_{xxy}(\zeta_3,\eta_3)(x_{i+1}-x_i)^2(y_{j-\frac{1}{2}}-y_j)\\
&\qquad+3u_{xyy}(\zeta_3,\eta_3)(x_{i+1}-x_i)(y_{j-\frac{1}{2}}-y_j)^2+u_{yyy}(\zeta_3,\eta_3)(y_{j-\frac{1}{2}}-y_j)^3)\Big)\\
&-\Big(u+u_x(x_{i-1}-x_i)+\frac{1}{2}\,u_{xx}(x_{i-1}-x_i)^2+\frac{1}{6}\,u_{xxx}(\zeta_4,\eta_4)(x_{i-1}-x_i)^3\Big)\Bigg)
\end{aligned}
$$

Since $(x_{i+1}-x_i)=2h$, $(x_{i-1}-x_i)=-2h$, $(y_{j+\frac{1}{2}}-y_j)=h$ and $(y_{j-\frac{1}{2}}-y_j)=-h$, we proceed as before and arrive at:

$$
\begin{aligned}
\frac{a}{4h}\Bigg(&\Big(u+2u_xh+(u_yh-u_yh)+2\,u_{xx}h^2+\frac{1}{8}\,(4u_{xy}h^2-4u_{xy}h^2)\\
&\quad+\frac{1}{8}\,(u_{yy}h^2+u_{yy}h^2)+O(h^3)\Big)\\
&-\Big(u-2u_xh+2\,u_{xx}h^2+O(h^3)\Big)\Bigg)\\
&=\frac{a}{4h}\Big(4u_xh+\frac{1}{4}\,u_{yy}h^2+O(h^3)\Big)\\
&=au_x+\frac{a}{16}\,u_{yy}h+O(h^2)
\end{aligned}
$$

For this point, the truncation error is $\tau(x_i,y_j)=au_x+\frac{a}{16}\,u_{yy}h+O(h^2)\leq K(h)$, which means that for this point too, the order of accuracy is 1.

Thus, we now know that the scheme presented for Method B is stable, but these computations shows that the order of accuracy is 1 for the scheme at the interfaces between the fine and coarse grids. Thus the, accuracy is reduced when using Method B for as a mesh refinement method.

The scheme changes when adding artificial diffusion, as explained in 2.5. We will go through the same points A, B and C, and compute the truncation error of the diffusion. The diffusion is added to the scheme at the given point, and so the truncation error for the entire scheme is given when adding the two truncation errors together. Starting with point A, the artificial diffusion looks like the following:

$$
\begin{aligned}
&-\lambda\Big(\frac{u_{i+1,j}-2u_{i,j}+u_{i-1,j}}{3h}\Big)\\
&-\lambda\Big(\frac{u_{i,j+1}-2u_{i,j}+u_{i,j-1}}{2h}\Big)
\end{aligned}
$$

The Taylor expansion of the artificial diffusion at this point is:

$$
\begin{aligned}
-\frac{\lambda}{3h}\Big(&u+u_x(x_{i+1}-x_i)+\frac{1}{2}u_{xx}(x_{i+1}-x_i)^2+\frac{1}{6}u_{xxx}(\zeta_1,\eta_1)(x_{i+1}-x_i)^3\\
&-2u\\
&+u+u_x(x_{i-1}-x_i)+\frac{1}{2}u_{xx}(x_{i-1}-x_i)^2+\frac{1}{6}u_{xxx}(\zeta_2,\eta_2)(x_{i-1}-x_i)^3\Big)\\
-\frac{\lambda}{2h}\Big(&u+u_y(y_{j+1}-y_j)+\frac{1}{2}u_{yy}(y_{j+1}-y_j)^2+\frac{1}{6}u_{yyy}(\theta_1,\xi_1)(y_{j+1}-y_j)^3\\
&-2u\\
&+u+u_y(y_{j-1}-y_j)+\frac{1}{2}u_{yy}(y_{j-1}-y_j)^2+\frac{1}{6}u_{yyy}(\theta_2,\xi_2)(y_{j-1}-y_j)^3\Big)
\end{aligned}
$$

Substituting $(x_{i+1}-x_i)=h=-(x_{i-1}-x_i)$, and the same for y-direction, we get:

$$
\begin{aligned}
&-\frac{\lambda}{3h}\Big(u+u_xh+\frac{1}{2}u_{xx}h^2-2u+u-u_x2h+\frac{1}{2}u_{xx}4h^2+O(h^3)\Big)\\
&-\frac{\lambda}{2h}\Big(u+u_yh+\frac{1}{2}u_{yy}h^2-2u+u-u_yh+\frac{1}{2}u_{yy}h^2+O(h^3)\Big)\\
&=-\frac{\lambda}{3h}\Big(-u_xh+\frac{5}{2}u_{xx}h^2+O(h^3)\Big)-\frac{\lambda}{2h}\Big(u_{yy}h^2+O(h^3)\Big)\\
&=-\frac{\lambda}{3}u_x-\frac{5\lambda}{6}u_{xx}h-\frac{\lambda}{2}u_{yy}h+O(h^2)
\end{aligned}
$$

We observe that the truncation error has a term of order 0. From Section 2.3 we know that the scheme is inconsistent at this point. This means, by Theorem 2.3.1, that we are not guaranteed convergence. Still, since it only applies to some of the points on the grid, the method might still converge as we will see in the next section.

The next point we will observe is point B. The artificial diffusion at this point is given by:

$$\begin{aligned}&-\lambda\Big(\frac{u_{i+1,j}-2u_{i,j}+\frac{1}{2}(u_{i-1,j+1}+u_{i-1,j-1})}{3h}\Big)\\ &-\lambda\Big(\frac{u_{i,j+1}-2u_{i,j}+u_{i,j-1}}{2h}\Big)\end{aligned}$$

and the Taylor expansion for the x-direction is:

$$\begin{aligned}-\frac{\lambda}{3h}\Big(&u+u_x(x_{i+1}-x_i)+\frac{1}{2}u_{xx}(x_{i+1}-x_i)^2+\frac{1}{6}\,u_{xxx}(\zeta_1,\eta_1)(x_{i+1}-x_i)^3\\ &-2u\\ &+\frac{1}{2}\big(u+u_x(x_{i-1}-x_i)+u_y(y_{j+1}-y_j)\\ &\quad+\frac{1}{2}(u_{xx}(x_{i-1}-x_i)^2+2u_{xy}(x_{i-1}-x_i)(y_{j+1}-y_j))+u_{yy}(y_{j+1}-y_j)^2\\ &\quad+\frac{1}{6}(u_{xxx}(\zeta_2,\eta_2)(x_{i-1}-x_i)^3+3u_{xxy}(\zeta_2,\eta_2)(x_{i-1}-x_i)^2(y_{j+1}-y_j)\\ &\qquad+3u_{xyy}(\zeta_2,\eta_2)(x_{i-1}-x_i)(y_{j+1}-y_j)^2+u_{yyy}(\zeta_2,\eta_2)(y_{j+1}-y_j)^3)\big)\\ &+\frac{1}{2}\big(u+u_x(x_{i-1}-x_i)+u_y(y_{j-1}-y_j)\\ &\quad+\frac{1}{2}(u_{xx}(x_{i-1}-x_i)^2+2u_{xy}(x_{i-1}-x_i)(y_{j-1}-y_j)+u_{yy}(y_{j-1}-y_j)^2)\\ &\quad+\frac{1}{6}(u_{xxx}(\zeta_3,\eta_3)(x_{i-1}-x_i)^3+3u_{xxy}(\zeta_3,\eta_3)(x_{i-1}-x_i)^2(y_{j-1}-y_j)\\ &\qquad+3u_{xyy}(\zeta_3,\eta_3)(x_{i-1}-x_i)(y_{j-1}-y_j)^2+u_{yyy}(\zeta_3,\eta_3)(y_{j-1}-y_j)^3)\big)\Big)\end{aligned}$$

As before we substitute with h:

$$\begin{aligned}
&-\frac{\lambda}{3h}\Big(u + u_x h + \frac{1}{2}u_{xx}h^2 + O(h^3) - 2u + \frac{1}{2}\big(u - u_x 2h + u_y h \\
&\qquad + \frac{1}{2}(u_{xx}4h^2 - 2u_{xy}2h^2 + u_{yy}h^2) + O(h^3)\big) + \frac{1}{2}\big(u - u_x 2h - u_y h \\
&\qquad + \frac{1}{2}(u_{xx}4h^2 + 2u_{xy}2h^2 + u_{yy}h^2) + O(h^3)\big) \\
&= \frac{\lambda}{3}u_x - \frac{5\lambda}{6}u_{xx}h - \frac{8\lambda}{12}u_{yy}h + O(h^2)
\end{aligned}$$

For point B, we also have an inconsistent scheme due to the order 0 term $\frac{\lambda}{3}u_x$. Thus, the scheme is inconsistent for all points along the line $i = k$.

Finally, we look at point C. At this point in x-direction, we have the scheme:

$$-\lambda\left(\frac{\frac{1}{2}u_{i+1,j} + \frac{1}{4}u_{i+1,j+\frac{1}{2}} + \frac{1}{4}u_{i+1,j-\frac{1}{2}} - 2u_{i,j} + u_{i-1,j}}{2h}\right)$$

and the Taylor expansion is:

$$\begin{aligned}
&-\frac{\lambda}{2h}\Big(\frac{1}{2}\big(u + u_x(x_{i+1}-x_i) + \frac{1}{2}u_{xx}(x_{i+1}-x_i)^2 + \frac{1}{6}\,u_{xxx}(\zeta_1,\eta_1)(x_{i+1}-x_i)^3\big) \\
&\quad + \frac{1}{4}\big(u + u_x(x_{i+1}-x_i) + u_y(y_{j+\frac{1}{2}}-y_j) \\
&\qquad + \frac{1}{2}(u_{xx}(x_{i+1}-x_i)^2 + 2u_{xy}(x_{i+1}-x_i)(y_{j+\frac{1}{2}}-y_j)) + u_{yy}(y_{j+\frac{1}{2}}-y_j)^2) \\
&\qquad + \frac{1}{6}(u_{xxx}(\zeta_2,\eta_2)(x_{i+1}-x_i)^3 + 3u_{xxy}(\zeta_2,\eta_2)(x_{i+1}-x_i)^2(y_{j+\frac{1}{2}}-y_j) \\
&\qquad\quad + 3u_{xyy}(\zeta_2,\eta_2)(x_{i+1}-x_i)(y_{j+\frac{1}{2}}-y_j)^2 + u_{yyy}(\zeta_2,\eta_2)(y_{j+\frac{1}{2}}-y_j)^3)\big) \\
&\quad + \frac{1}{4}\big(u + u_x(x_{i+1}-x_i) + u_y(y_{j-\frac{1}{2}}-y_j) \\
&\qquad + \frac{1}{2}(u_{xx}(x_{i+1}-x_i)^2 + 2u_{xy}(x_{i+1}-x_i)(y_{j-\frac{1}{2}}-y_j)) + u_{yy}(y_{j-\frac{1}{2}}-y_j)^2) \\
&\qquad + \frac{1}{2}(u_{xxx}(\zeta_3,\eta_3)(x_{i+1}-x_i)^3 + 3u_{xxy}(\zeta_3,\eta_3)(x_{i+1}-x_i)^2(y_{j-\frac{1}{2}}-y_j) \\
&\qquad\quad + 3u_{xyy}(\zeta_3,\eta_3)(x_{i+1}-x_i)(y_{j-\frac{1}{2}}-y_j)^2 + u_{yyy}(\zeta_3,\eta_3)(y_{j-\frac{1}{2}}-y_j)^3)\big) \\
&\quad -2u \\
&\quad +u + u_x(x_{i+1}-x_i) + \frac{1}{2}u_{xx}(x_{i+1}-x_i)^2 + O(h^3)\Big)
\end{aligned}$$

As before we substitute with h:

$$
\begin{aligned}
&-\frac{\lambda}{2h}\Big(\frac{1}{2}(u + u_x h + \frac{1}{2}u_{xx}h^2 + O(h^3))\\
&\quad +\frac{1}{4}\big(u - u_x h + u_y\frac{h}{2} + \frac{1}{2}(u_{xx}h^2 - 2u_{xy}\frac{h^2}{2} + u_{yy}\frac{h^2}{4}) + O(h^3)\big)\\
&\quad +\frac{1}{4}\big(u + u_x h - u_y\frac{h}{2} + \frac{1}{2}(u_{xx}h^2 + 2u_{xy}\frac{h^2}{2} + u_{yy}\frac{h^2}{4}) + O(h^3)\big)\\
&\quad -2u\\
&\quad\quad +u - u_x h + \frac{1}{2}u_{xx}h^2 + O(h^3)\Big)\\
&= -\frac{\lambda}{2h}\Big(u + u_x h + \frac{1}{2}u_{xx}h^2 + \frac{1}{2}u_{yy}\frac{h^2}{4} + O(h^3))\\
&\quad -2u\\
&\quad\quad +u - u_x h + \frac{1}{2}u_{xx}h^2 + O(h^3)\Big)\\
&= -\frac{\lambda}{2}u_{xx}h - \frac{\lambda}{16}u_{yy}h + O(h^3)
\end{aligned}
$$

For this point, the scheme is consistent, and we get a truncation error with an order of 1. From this we observe that the change in mesh refinement is the problem for the artificial diffusion term of the scheme, and is what makes it inconsistent.

For point D, the corner of the fine grid, the truncation error is very similar as the truncation error along $i = k$. In both x- and y- direction there will be an inconsistent term: $\frac{\lambda}{3}u_x$ and $\frac{\lambda}{3}u_y$.

4.2.3 Error estimate of mesh refinement method

It is possible to determine an error bound representing the worst case error for the method based on the truncation error. Let D be the finite volumes operator such that

$$Du = -a\frac{u_{i+1,j} - u_{i-1,j}}{2h} - b\frac{u_{i,j+1} - u_{i,j-1}}{2h}$$

for the initial-boundary value problem in Equation (2.1). This is the same as the central difference operator. Consider the following semi-discretization of the advection equation:

$$\begin{aligned}(u_{i,j})_t &= Du_{i,j}\\ u_{i,j}(0) &= f_{i,j}\\ u_{0,j} &= g_j(t)\\ u_{i,0} &= g_i(t)\end{aligned}$$

$u_{i,j}$ is the numerical approximation at point (x_i, y_j), and $u(x_i, y_j)$ is the exact solution of (3.1) at the same point. If T is the truncation error then we have that:

$$\begin{aligned}u_t(x_i, y_j, t) &= Du_{i,j} + T(x_i, y_j, t)\\ u(x_i, y_j, 0) &= f_{i,j} + \phi_{i,j}\\ u(0, y_j, t) &= g_j(t) + \psi_y\\ u(x_i, 0, t) &= g_i(t) + \psi_x\end{aligned}$$

Where $\phi_{i,j}$ is the error in the initial data, and ψ is the error in the boundary data. We subtract these two functions $e_{i,j} = u_{i,j} - u(x_i, y_j)$, and obtain the error equation:

$$\begin{aligned}e_t &= T(x_i, y_j)\\ e_{i,j}(0) &= \phi_{i,j}\\ e_{0,j}(t) &= \psi_y\\ e_{0,j}(t) &= \psi_x\end{aligned}$$

Applying the energy method on this error equation we get:

$$\begin{aligned}\sum_{i,j} e_t e &= \sum_{i,j} T(x_i, y_j)e\\ \frac{||e^2||}{2} &= \sum_{i,j} T(x_i, y_j)e\end{aligned}$$

Using the equality $\frac{||e^2||}{2} = ||e||||e||_t$ for the left side and the Cauchy-Schwartz inequality on the right hand side, we get:

$$\begin{aligned}||e||||e||_t &\leq ||T(x_i, y_j)||||e||\\ ||e||_t &\leq ||T(x_i, y_j)||\end{aligned}$$

Proposition (3.4) from [Nor08], tells us that the error $||e||$ is bounded by the truncation error:

$$||e|| \leq ||T(x_i, y_j)|| \tag{4.19}$$

Say we have one initial grid G_0 with a subgrid G_1 located somewhere in the middle of the initial grid. From the computations of truncation errors above, we know that the coarse points outside the boundary of G_1 have a truncation error of $O(h)$, and the points at the interface have a truncation error of $O(1)$. This means that the amount of points with error $O(1)$ and $O(h)$ is bounded by $4m$. This leads to the following estimate:

$$\begin{aligned} ||T(x_i, y_j)|| &= \sqrt{\sum_{i=0}^{m}\sum_{j=0}^{m} h^2 T_{i,j}^2} \\ &\leq \sqrt{h^2 m^2 h^4 + h^2 4mO(h)^2 + h^2 4mO(1)^2} \\ &\leq \sqrt{h^4 + 4hO(h)^2 + 4hO(1)^2} \\ &\leq h^2 + 2h^{\frac{3}{2}} + 2h^{\frac{1}{2}} \leq K(h^{\frac{1}{2}}) \end{aligned} \tag{4.20}$$

Thus, the error is bounded below by $K(h^{\frac{1}{2}})$ which means that the worst order of convergence the method can achieve is 0.5.

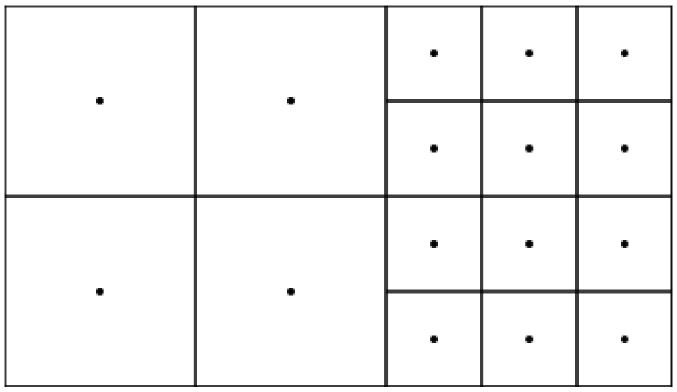

Figure 4.4: Grid with cell-centered nodes and refinement of 2.

4.3 Another approach at the Mesh Refinement method with finite volumes

In [BC89] Marsha Berger and Joseph Colella present a different approach to the Adaptive Mesh Refinement method than in [BO84]. They describe how to approach the method when using a cell-centered finite volume scheme. The changes done to the method in [BC89] will be explained in this section and result in a third method, which we will refer to as Method C.

The cell-centered mesh refinement approach differs from Method A at a few points, one of which is the way we formulate the finite volume scheme. The approximated point is inside each cell on the computational domain rather than at the grid intersections, as explained in Section 2.4. This can be seen in Figure 4.4. For this method, a refinement will look slightly different than for the previous ones. In the figure, we observe that a cell-centered method will lead to a refinement of the cells. The cell centers however, are no longer aligned on the coarse and fine grids. In the previous methods, the refined grid contained the same points as the coarse grid with new points in between. For the cell-centered method, none of the points on the refinement are aligned with the points on the coarse grid, as we can see in Figure 4.4. This does not effect the inner points of the refinement, which have the same structure and mesh spacing as in the other two methods. The solution at these points is calculated the same way as before, separated from

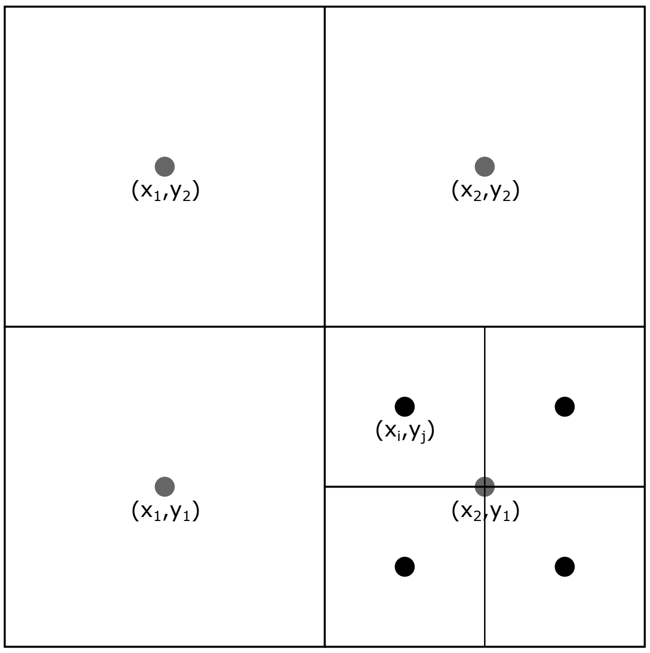

Figure 4.5: Interface between coarse and refined grid for UMR where the solution at point (x_i, y_j) is to be interpolated by the four coarse points surrounding it

the coarse grid points. The boundary points however, needs to be calculated in a different manner.

4.3.1 Boundary values for refinements with cell-centered finite volume scheme

As before, we are using interpolation to approximate the boundary points of the refined subgrid. Earlier we used linear interpolation for this, but since the boundary points for the fine grid are not aligned with the coarse grid points in either x- or y-direction, we will use bilinear interpolation, as in [BC89].

Let (x_i, y_j) be a refined boundary point to be interpolated. The coarse grid points around (x_i, y_j) are given by $(x_1, y_1), (x_1, y_2), (x_2, y_1)$ and (x_2, y_2), illustrated in Figure 4.5. The bilinear interpolation is done the following way (from [Pre92]):

$$
\begin{aligned}
u(x_i, y_1) &\approx \frac{x_2 - x_i}{x_2 - x_1} u(x_1, y_1) + \frac{x_i - x_1}{x_2 - x_1} u(x_2, y_1) \\
u(x_i, y_2) &\approx \frac{x_2 - x_i}{x_2 - x_1} u(x_1, y_2) + \frac{x_i - x_1}{x_2 - x_1} u(x_2, y_2) \\
u(x_i, y_j) &\approx \frac{y_2 - y_j}{y_2 - y_1} u(x_i, y_1) + \frac{y_j - y_1}{y_2 - y_1} u(x_i, y_2)
\end{aligned}
$$

For a boundary at a refined grid with refinement 2, we have that $(x_2 - x_1) = (y_2 - y_1) = h$, $(x_2 - x_i) = (y_j - y_1) = \frac{3}{4}h$ and $(x_i - x_1) = (y_2 - y_j) = \frac{1}{4}h$. Thus the interpolation becomes:

$$
\begin{aligned}
u(x_i, y_1) &\approx \frac{3}{4} u(x_1, y_1) + \frac{1}{4} u(x_2, y_1) \\
u(x_i, y_2) &\approx \frac{3}{4} u(x_1, y_2) + \frac{1}{4} u(x_2, y_2) \\
u(x_i, y_j) &\approx \frac{1}{4} u(x_i, y_1) + \frac{3}{4} u(x_i, y_2)
\end{aligned}
$$

This is how the solution at the boundary of the refined grid is calculated. The next question is how to transfer the updated values from the fine subgrid back to the coarse grid.

4.3.2 Updating a coarse grid

The challenge with non-aligned grid points reappears when it is time to update the coarse grid with the values from its subgrids. This process is similar to the boundary calculations, but reversed. In this case one takes the average of the refined volumes inside the coarse volume as in Figure 4.6. Say once again that we have a refinement of 2. When updating the solution u_{coarse} at (x_{coarse}, y_{coarse}), the following calculations are done:

$$
u_{coarse} = \frac{1}{r^2} \sum_{i=1}^{2} \sum_{j=1}^{2} u(x_i, y_j)
$$

This is done for all the coarse volumes covered by fine volumes from a subgrid.

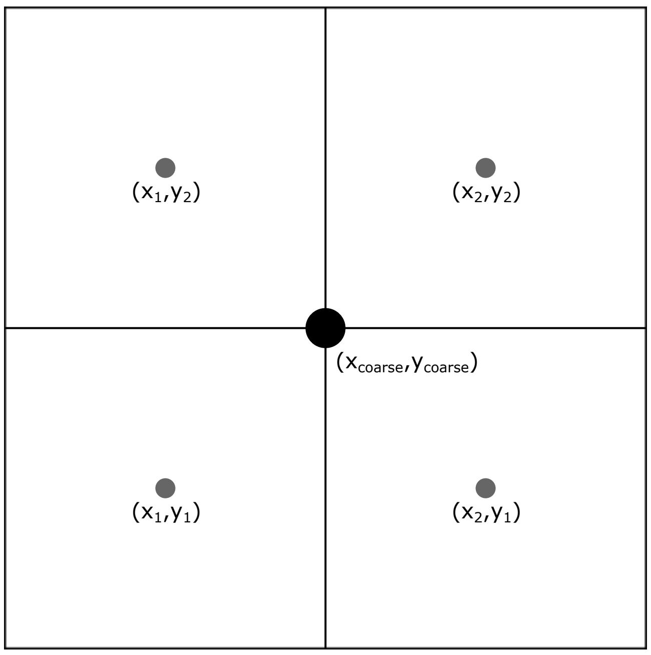

Figure 4.6: Coarse grid cell refined by a factor of $r = 2$

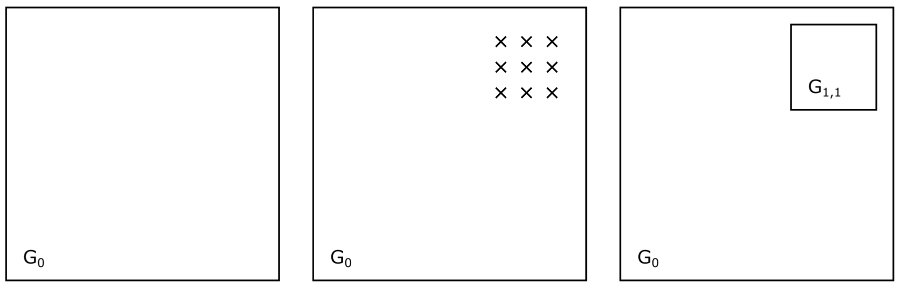

Figure 4.7: (From the left) Image 1: Initial grid G_0. Image 2: Grid G_0 with flagged points where the error is too big. Image 3: Grid G_0 with refinement $G_{1,1}$ where all flagged points are interior to the new refined grid.

4.4 Adaptive Method

Having explained what mesh refinement is and how it is done, we return to the rest of the Adaptive Mesh Refinement method. As mentioned in the introduction, the AMR method uses mesh refinement as a tool to improve the accuracy of the simulation.

The method begins with a Cartesian grid G_0 covering the computational domain. The solution u is first calculated at grid G_0. When using the AMR method, one wants to calculate an approximation with a given accuracy. To determine when or where the grid is in need of refinement, an error estimate is calculated. The error estimate is examined, and all grid points where the estimated solution is not accurate enough are flagged. Next, a new finer grid $G_{1,1}$ is created, covering the flagged points. This is illustrated in Figure 4.7. Now the grid consists of the initial grid G_0, and a refined grid $G_{1,1}$ at the next grid level with the mesh spacing $h_1 = \frac{h_0}{r}$. This continues, and grids are recursively created or destroyed depending on whether or not they are needed. An example of this grid structure is illustrated in Figure 4.8. The recursive process of grid creation is explained in more detail in the next section.

Each grid is kept independently of the others, and the solution is calculated at each of the grids and saved in their own solution vector. This leads to unnecessary storage consumption, but keeping the grids separate makes the programming of

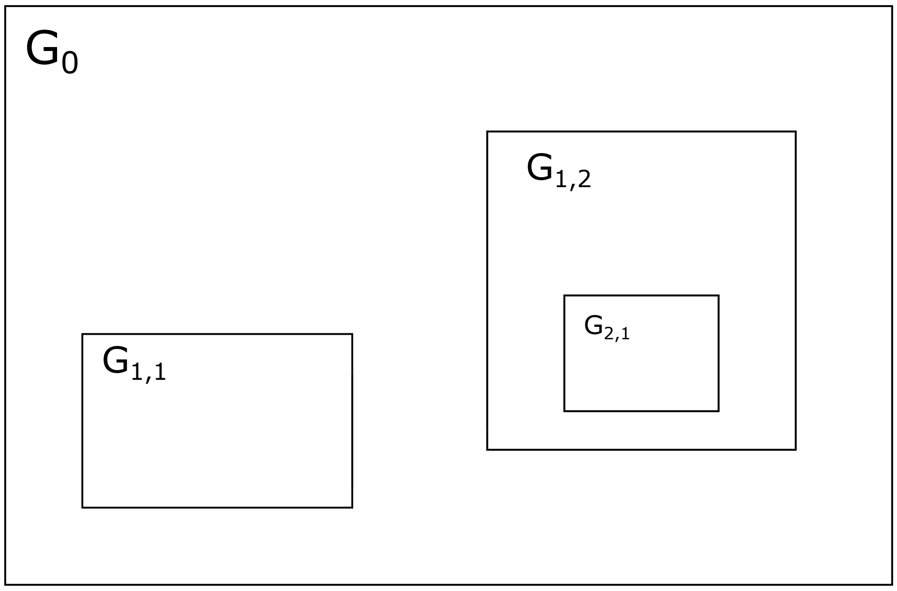

Figure 4.8: Grid G_0 with two refinements $G_{1,1}$ and $G_{1,2}$, where $G_{1,2}$ is refined again with the grid $G_{2,1}$

the solution easier, and saves a lot of time.

If a grid is in need of refinement, a clustering algorithm is used to determine how the new grids should be created. It is not trivial how to best cover a set of flagged points on a domain by different fine grids. Fewest possible grids must be created in order to avoid unnecessary computations, but at the same time one wishes to cover as little as possible of the underlying domain to avoid more precision where it is not needed. Further details of the algorithm is described in [BO84].

4.4.1 Recursive grid computation

The approximation of the numerical solution is done recursively on all the grids, level by level. The AMR method can refine in time as well as in space by the same ratio. This makes it easier to always satisfy the CFL-condition, as the ratio between the space step and the time step is preserved.

The algorithm starts by updating the initial grid G_0 to time $t_0 + k$. Next, the error algorithm is run to check if the grid is in need of refinement anywhere. If so, it flags the points, and creates one or more new, refined girds as explained in the previous subsection. The same procedure is performed on these subgrids, and so on. After updating all grids in G_i, the error algorithm is applied to add new subgrids or delete existing ones if this is necessary. All grids at level i are updated before the algorithm moves on to the next grid level. If there exists a level $i+1$, the procedure is repeated for these grids. They are updated, and the error algorithm is applied. The solution on a coarse parent-grid must be saved before the grid is updated, so that the solution is available when the boundary values on the subgrid are calculated. When all subgrids $G_{i,j}$ at level i are calculated, and there are no further refinements of these grids, the algorithm goes back to the coarser level $i-1$, and the values of the parent grids are updated with the values from its refined subgrids. This is explained in the next paragraph. In the end all grids are at time t_{n+1}, and all parent grids are updated with values from its subgrids. Then the algorithm moves on to the next time step. If the algorithm refines in time too, a subgrid has smaller timesteps than its parent grid and is updated several times before it reaches t_n. A pseudocode is shown in Algorithm 1.

Algorithm 1: Recursive grid computation

```
computeSol($G_i$)
    while $G_i$ not at time $t_{n+1}$ do
        for grid $G_{i,j}$ in $G_i$ do
            update $G_{i,j}$ to $t_i + \frac{k}{r^{i+1}}$ run error algorithm for $G_{i,j}$
        end
        if level $i+1$ exists then
            computeSol($G_{i+1}$)
            update $G_{i,j}$ with values from $G_{i+1,j}$
    end
```

After a grid and its subgrid is updated to the same time t_{n+1}, the coarse grid needs to be updated with the values from its refinement. This is done by simply injecting the values from the fine grid onto the coarse grid. If the coarse grid points are not contained in the fine grid, an average is calculated from the fine grid instead.

The AMR method is complex, and each of the introduced parts are carefully developed. The error estimation and clustering algorithm forms the adaptive part of the method. This is the part of the algorithm that automatically finds out where to create new grids and places them where they belong. We have decided to focus on the mesh refinement in the method. The method can be expanded to be adaptive, but in this thesis we have left this part out, and will not go into more detail about it. Instead we focus on studying the convergence, stability and accuracy of the mesh refinements that are used in the AMR-method.

This concludes the Adaptive mesh refinement method. We will now proceed with the results from different numerical experiments and compare the different mesh refinements methods that we have introduced.

4.5 Numerical results

We have done two different simulations comparing the three mesh refinement methods. First we approximate a 2-dimensional sine wave using the advection equation. Later we approximate a vortex with the Euler equations, and at the end of the chapter we have simulated a radial explosion with the Euler equations.

4.5.1 Simulations of 2-dimensional sine wave with the advection equation

We have approximated the following PDE problem:

$$\begin{aligned} u_t + u_x + u_y &= 0, \quad x, y \in [0,1]^2, \quad 0 \le t \le 1 \\ u(x,0,t) &= \sin(x-t) + \sin(t) \\ u(0,y,t) &= \sin(t) + \sin(y-t) \\ u(x,y,0) &= \sin(x) + \sin(y) \end{aligned} \tag{4.21}$$

The PDE is approximated on the domain $[0,1] \times [0,1]$ which is discretized by an initial grid G_0. This grid has a refined subgrid G_1 located at $[0.3, 0.7]^2$ with a refinement of $r = 2$. Thus, if the mesh spacing of G_0 is given by h_0, then the mesh spacing of G_1 becomes $h_1 = \frac{h_0}{2}$. When presenting a result we will always refer to the gridpoints of the initial grid G_0 when specifying the amount of gridpoints $m \times m$. There is no refinement in time, and the CFL condition is $CFL = \frac{\Delta t}{\Delta x} = 2r$, where r is the ratio of refinement. The simulations are run from $t = 0$ to $t = 1$. In Figure 4.9 and 4.10, the initial function of the problem is plotted on a 10×10 grid with refinement. For this grid, the 10×10 grid is the initial grid with mesh spacing $h_0 = \frac{1}{9}$, which means that the subgrid G_1 has mesh spacing $h_0 = \frac{1}{18}$. Figure 4.9 shows the grid refinement for mesh refinement Method A and B, and 4.10 shows the cell-centered mesh refinement method, Method C. In the plots we can see that the refined grid on the cell-centered Method C has no coinciding points with the coarse grid, whereas the original mesh refinement, Method A and B, does.

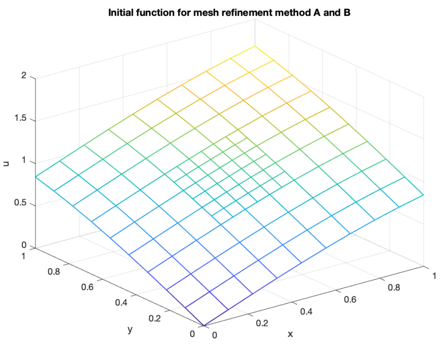

Figure 4.9: Initial function for a sine wave with mesh refinement (Method A and B) on a 10×10 grid.

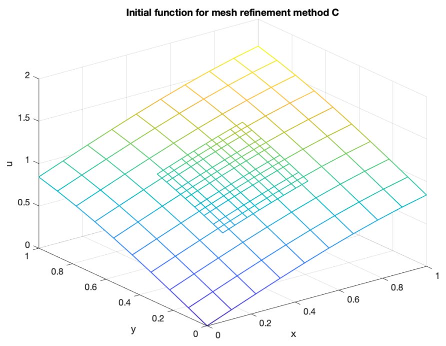

Figure 4.10: Initial function for a sine wave with cell-centered mesh refinement (Method C) on a 10×10 grid.

The first thing to notice about the results is that Method A is not stable for this simulation. The interface between the coarse and the fine grid seems to be the reason. When imposing exact boundary conditions onto the interface instead of interpolating the coarse grid values, the method achieves a convergence rate of 2. This implies two things: First of all the interpolation of the coarse grid points at the boundary of the fine grid is indeed the cause of the instabilities. Second, it gives a stronger assurance that there are no programming errors in the code causing the instabilities. To test this further, we interpolated the exact solution at the fine boundary, rather than the points from the coarse grid. This also resulted in a convergence rate of 2. Since the finite volume scheme is stable, small disturbances to the boundary conditions should not result in instability of the method. This supports the conclusion that the method is unstable, as it shows that the interpolation is done correctly.

In order to make Method A stable, it is necessary to add artificial diffusion. This is what was done by Marsha Berger when proving stability for the AMR method in [Ber85]. When this is done, the scheme looks like in Equation (2.12). Trying to make the results as accurate as possible, the lowest diffusion constant we can choose is $\lambda = 0.1$. Mesh refinement Method B confirms the stability calculations from earlier, and is not in need of any artificial diffusion. Mesh refinement Method C with the cell-centered finite volumes scheme was also implemented. This method is stable, which means that the bilinear interpolation done at the interface between coarse and fine grid does not lead to an instability as the interpolation in Method A does.

The convergence of the three different methods are shown in Table 4.1, together with the method without mesh refinement. Keep in mind that the simulations with mesh refinement Method A are the only ones where artificial diffusion has been added. The results shows that one cannot expect the same convergence when the mesh is refined. It is clear that the method looses some of its accuracy when refinement is added. In light of the truncation errors calculated for mesh refinement Method B in Section 4.2.2, it makes sense that a refinement creates more error in the calculations as the increased error on the refinement does affect the convergence rate.

When comparing mesh refinement Method A to Method B, we see that the extra calculations done to Method B was necessary in order to get stability. Still, mesh refinement Method C is very similar to the stable method, both in error and convergence. It is also interesting to notice that the most accurate method is mesh refinement Method A, even though we have added artificial diffusion to this method. The solution of the PDE is smooth, and thus a good example of where

	Error			
m	Method A	Method B	Method C	no mr.
50	2.278×10^{-3}	3.180×10^{-3}	3.177×10^{-3}	1.091×10^{-4}
100	1.104×10^{-3}	1.505×10^{-3}	1.557×10^{-3}	2.648×10^{-5}
200	5.563×10^{-4}	7.509×10^{-4}	7.756×10^{-4}	6.526×10^{-6}
400	2.788×10^{-4}	3.750×10^{-4}	3.859×10^{-4}	1.621×10^{-6}
800	1.396×10^{-4}	1.874×10^{-4}	1.925×10^{-4}	4.039×10^{-7}

	Convergence			
m	Method A	Method B	Method C	no mr.
50	-	-	-	-
100	1.087	1.139	1.029	2.043
200	1.000	1.025	1.006	2.021
400	0.996	1.009	1.007	2.010
800	0.998	1.004	1.003	2.005

Table 4.1: Error and convergence for mesh refinement Method A, B, C and no mesh refinement for advection equation

an interface would be placed when running the adaptive mesh refinement.

4.5.2 Simulation of an inviscid vortex with the Euler equations

Next, we proceed to the Euler equations presented in Chapter 3 and expressed in Equation 3.4. We have simulated an inviscid vortex, given by the following data obtained from [Svä21]:

Variables in the equations:

Freestream pressure:	ρ_∞	$=$	1
Freestream temperature:	T_∞	$=$	273.15
Vortex radius:	R_v	$=$	0.1
Vortex strength:	β	$=$	1
Center of vortex:	(x_c, y_c)	$=$	$(0.5, 0.5)$
Angle of vortex:	α	$=$	0
Mach number:	Ma	$=$	0.01

These variables gives us the following constants:

Ideal gas constant:	R_g	$=$	287.15
Adiabatic exponent:	γ	$=$	1.4
Speed of sound:	c	$=$	331

Vortex functions:

$$\begin{aligned} f(x,y) &= \frac{(x-x_c)^2 + (y-y_c)^2}{{R_v}^2} \\ du_1(x,y) &= -\beta\, \frac{y-y_c}{R_v}\, e^{\frac{-f(x,y)}{2}} \\ du_2(x,y) &= -\beta\, \frac{x-x_c}{R_v}\, e^{\frac{-f(x,y)}{2}} \\ dT(x,y) &= \frac{1}{2}\, \beta^2\, \frac{e^{-f(x,y)}}{c_p} \end{aligned}$$

Field variables:

$$\begin{aligned} \rho &= \rho_\infty \Bigg(\frac{T_\infty - dT(x,y)}{T_\infty} \Bigg)^{\frac{1}{\gamma-1}} \\ \rho u_1 &= \rho\Big(u_\infty \cos(\alpha) + du_1(x,y) \Big) \\ \rho u_2 &= \rho\Big(u_\infty \sin(\alpha) + du_2(x,y) \Big) \\ E &= \Bigg(\frac{R_g \rho (T_\infty - dT(x,y))}{\gamma - 1} + \frac{1}{2} \frac{(\rho u_1)^2 + (\rho u_2)^2}{\rho} \Bigg) \end{aligned}$$

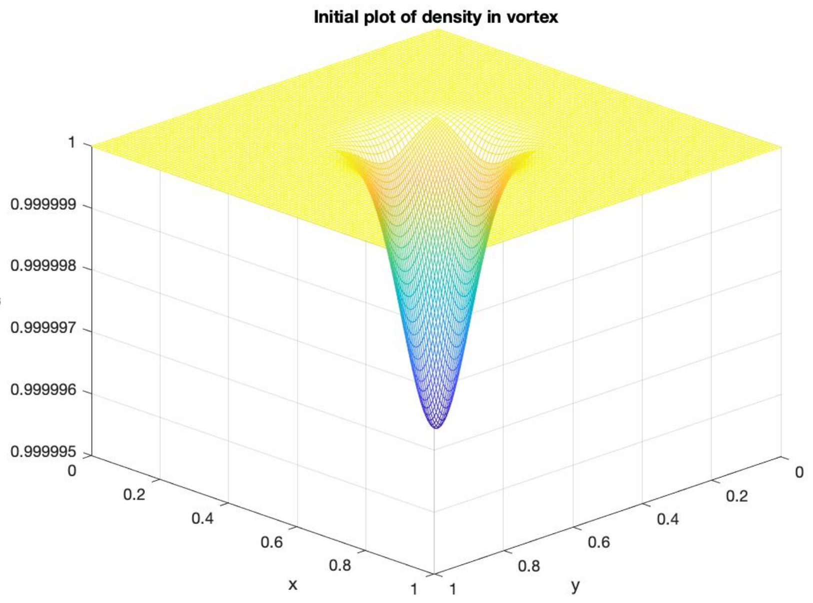

Figure 4.11: Initial density for vortex plotted on 100x100 grid with a refinement of $r = 2$ on $[0.35, 0.65]^2$ seen from below.

The solution is calculated in the time interval $[0, 0.15]$. The CFL-condition is given by $\frac{\Delta x}{\Delta t} = (c + max(\sqrt{u_1^2 + u_2^2}))r$. The refinement is located at the square $[0.35, 0.65]^2$. Initially, the vortex is located in the middle of the domain, inside the refinement. This can be seen in Figure 4.11 of the initial density, where all the points from the coarse and the fine grid is plotted. The figure shows the density from below, and the fine grid can be seen around the middle of the vortex where the mesh spacing is smaller than elsewhere. The vortex moves in the positive x-direction towards the boundary at $x = 1$. Thus, the vortex moves across the interface between the refined and coarse grid. The simulations are done with all three mesh refinement methods presented earlier. We have also run the simulations without mesh refinement and will compare this to the other methods. Figures 4.12, 4.13, 4.14 and 4.15 show the contour plots of the density for the three mesh refinement methods and for the method without mesh refinement, with 400^2 points at time $t = 0.07$. The convergence of all the methods can be seen in Tables 4.2 - 4.5.

Even though we proved stability for mesh refinement Method B with the ad-

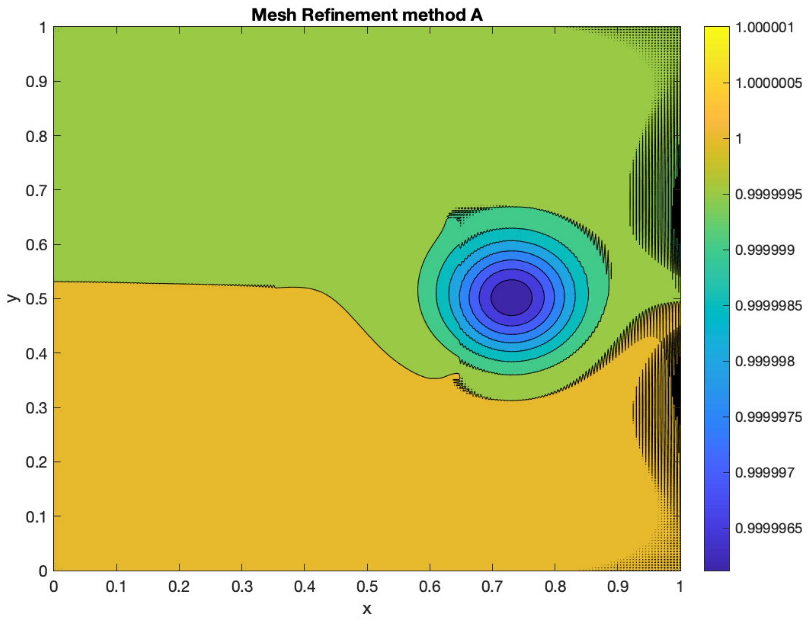

Figure 4.12: Contour plot of density in vortex moving in x-direction with mesh refinement Method A, 400^2 points at time t = 0.07 with artificial diffusion coefficient $\lambda = 6.5$

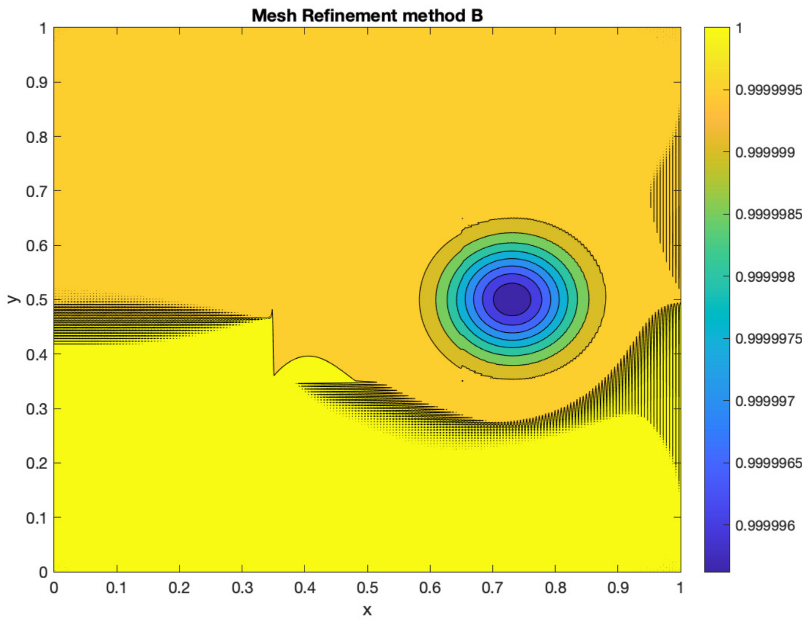

Figure 4.13: Contour plot of density in vortex moving in x-direction with stable mesh refinement, 400^2 points at time t = 0.07 with artificial diffusion coefficient $\lambda = 1.5$

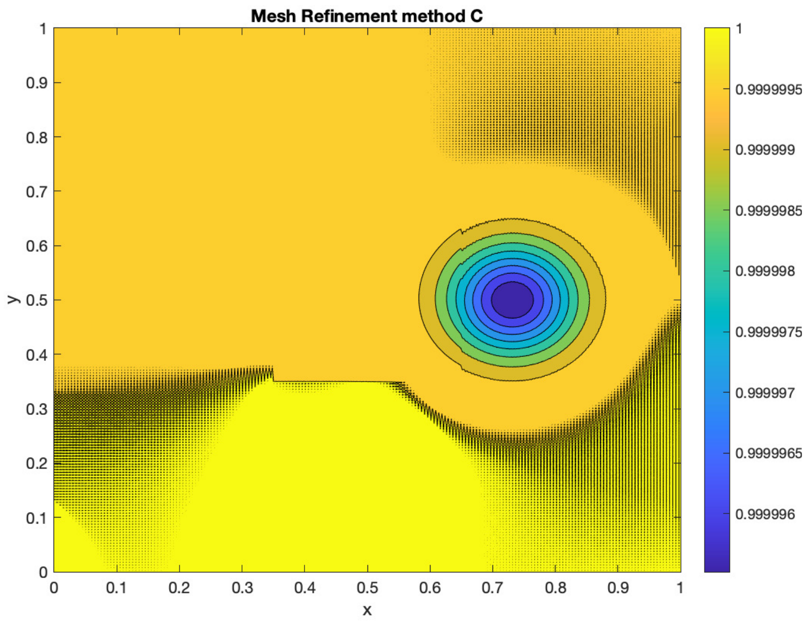

Figure 4.14: Contour plot of density in vortex moving in x-direction with cell-centered mesh refinement method, 400^2 points at time t = 0.07 with artificial diffusion coefficient $\lambda = 0.5$

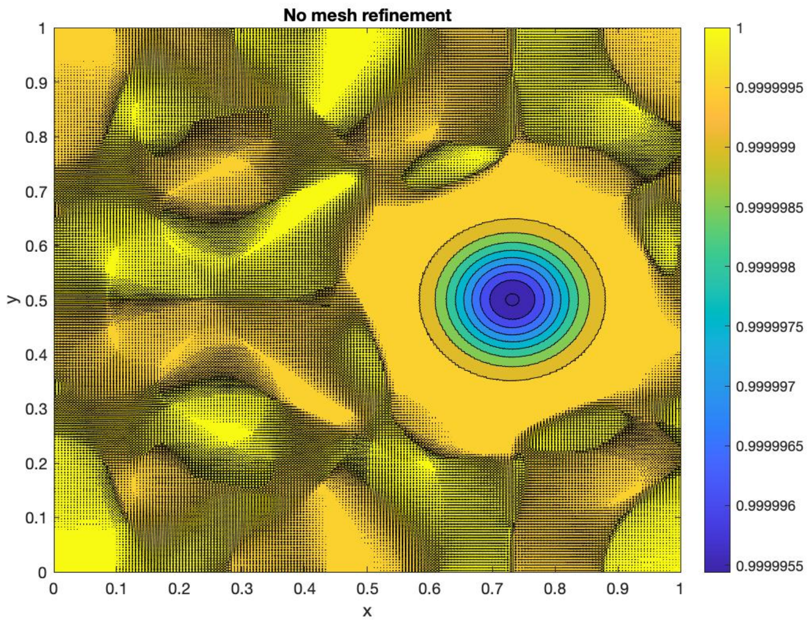

Figure 4.15: Contour plot of density votex moving in x-direction with no mesh refinement, 400^2 points at time t = 0.07 no artificial diffusion

Mesh refinement Method A				
Error				
m	ρ	v_1	v_2	E
100	2.377×10^{-6}	2.692×10^{-2}	4.827×10^{-2}	6.785×10^{-1}
150	1.713×10^{-6}	2.103×10^{-2}	3.696×10^{-2}	4.893×10^{-1}
200	1.345×10^{-6}	1.715×10^{-2}	2.981×10^{-2}	3.844×10^{-1}
250	1.099×10^{-6}	1.451×10^{-2}	2.484×10^{-2}	3.144×10^{-1}
300	9.274×10^{-7}	1.257×10^{-2}	2.133×10^{-2}	2.655×10^{-1}
350	7.977×10^{-7}	1.112×10^{-2}	1.864×10^{-2}	2.286×10^{-1}
400	7.002×10^{-7}	9.943×10^{-3}	1.657×10^{-2}	2.009×10^{-1}

Convergence				
m	ρ	v_1	v_2	E
100	-	-	-	-
150	0.808	0.609	0.658	0.806
200	0.840	0.709	0.747	0.839
250	0.904	0.748	0.818	0.901
300	0.932	0.791	0.836	0.927
350	0.978	0.791	0.875	0.970
400	0.976	0.839	0.882	0.970

Table 4.2: Convergence and error for simulations of vortex moving in positive x-direction with mesh refinement Method A

vection equation, the method is not stable for the Euler vortex. In fact, it is not stable for any of the three mesh refinement methods. Thus we must add artificial diffusion for these methods as well, and we get the scheme in Equation (2.12). Here we also notice an important difference between the mesh refinement Method A and B. Method B needs significantly less diffusion than Method A. While Method A needs about $\lambda = 6.5$ the stable method needs only $\lambda = 1.5$. The latter is clearly more stable which makes sense since we have been able to prove stability for the case in Section 4.2, and we have also seen it earlier when simulating the advection equation. This causes Method A to lose accuracy as we can see in the errors in

Mesh refinement Method B				
Error				
m	ρ	v_1	v_2	E
100	7.700×10^{-7}	1.122×10^{-2}	1.658×10^{-2}	2.289×10^{-1}
150	5.376×10^{-7}	7.115×10^{-3}	1.107×10^{-2}	1.568×10^{-1}
200	4.091×10^{-7}	5.277×10^{-3}	8.362×10^{-3}	1.184×10^{-1}
250	3.253×10^{-7}	4.158×10^{-3}	6.666×10^{-3}	9.383×10^{-2}
300	2.716×10^{-7}	3.460×10^{-3}	5.566×10^{-3}	7.811×10^{-2}
350	2.324×10^{-7}	2.940×10^{-3}	4.748×10^{-3}	6.666×10^{-2}
400	2.030×10^{-7}	2.575×10^{-3}	4.159×10^{-3}	5.815×10^{-2}

Mesh refinement Method B				
Convergence				
m	ρ	v_1	v_2	E
100	-	-	-	-
150	0.886	1.123	0.996	0.934
200	0.949	1.039	0.976	0.976
250	1.027	1.068	1.016	1.043
300	0.991	1.007	0.989	1.006
350	1.010	1.057	1.032	1.028
400	1.014	0.993	0.992	1.023

Table 4.3: Convergence and error for simulations of vortex moving in positive x-direction with Mesh refinement Method B

Table 4.2. Mesh refinement Method C however, proves to be more stable than the other two. For this method we have $\lambda = 0.5$. This is interesting because the only difference between this method and Method A is the way the finite volume scheme is implemented, and the interpolation method.

The simulations of the Euler vortex also show that the results are better when we do not apply the mesh refinement. These simulations achieve a higher accuracy

Mesh refinement Method C				
Error				
m	ρ	v_1	v_2	E
100	2.654×10^{-7}	0.00783	0.00805	0.08898
150	2.548×10^{-7}	0.00469	0.00508	0.07683
200	1.394×10^{-7}	0.00327	0.00372	0.04433
250	1.171×10^{-7}	0.00249	0.00292	0.03632
300	9.824×10^{-8}	0.00200	0.00241	0.03012
350	8.155×10^{-8}	0.00168	0.00203	0.02497
400	7.12589×10^{-8}	0.00144	0.00177	0.02167

Convergence				
m	ρ	v_1	v_2	E
100	-	-	-	-
150	0.101	1.262	1.136	0.362
200	2.096	1.259	1.079	1.912
250	0.783	1.217	1.088	0.893
300	0.962	1.204	1.056	1.027
350	1.208	1.139	1.116	1.216
400	1.010	1.129	1.041	1.061

Table 4.4: Convergence and error for simulations of vortex moving in positive x-direction with Mesh refinement Method C

and we get a convergence of 2, as seen in Table 4.5. The artificial diffusion we added in the mesh refinement methods to make them stable is one of the reasons for the loss of accuracy.

Observing the error for the three mesh refinement methods in Tables 4.2 to 4.4 we see that the cell-centered mesh refinement Method C gives the most accurate results. In Figures 4.16, 4.17, 4.18 and 4.19, we have plotted the error of the density for each of the four methods at $t = 0.07$. In each of the mesh refinement

No mesh refinement				
	Error			
m	ρ	v_1	v_2	E
100	2.33560×10^{-7}	0.00167	0.00391	0.06432
150	1.32997×10^{-7}	7.34094×10^{-4}	0.00168	0.03635
200	6.76252×10^{-8}	4.10647×10^{-4}	9.25629×10^{-4}	0.01849
250	4.19191×10^{-8}	2.61893×10^{-4}	5.86291×10^{-4}	0.01149
300	2.85056×10^{-8}	1.81437×10^{-4}	4.04379×10^{-4}	0.00789
350	2.04871×10^{-8}	1.33078×10^{-4}	2.95661×10^{-4}	0.00562
400	1.54883×10^{-8}	1.01761×10^{-4}	2.25550×10^{-4}	0.00425

	Convergence			
m	ρ	v_1	v_2	E
100	-	-	-	-
150	1.389	2.032	2.089	1.408
200	2.351	2.019	2.061	2.350
250	2.143	2.016	2.046	2.133
300	2.115	2.013	2.037	2.109
350	2.143	2.011	2.031	2.139
400	2.094	2.009	2.027	2.092

Table 4.5: Convergence and error for simulations of vortex moving in positive x-direction with no mesh refinement

methods we can clearly see the contour of the fine-grid interface in the middle of the plot. The sides and the corners of the refined square are prominent. This demonstrates how the refinement creates error. When comparing to Figure 4.19, the error for the method without mesh refinement, we see that the error is spread more evenly throughout the domain, and is generally smaller than in the other plots. There is no diffusion for this simulation which is why the error looks less smooth than for the other plots. Remembering the truncation error that we derived in Section 4.2.2 the computations at the edges and corners are supported by

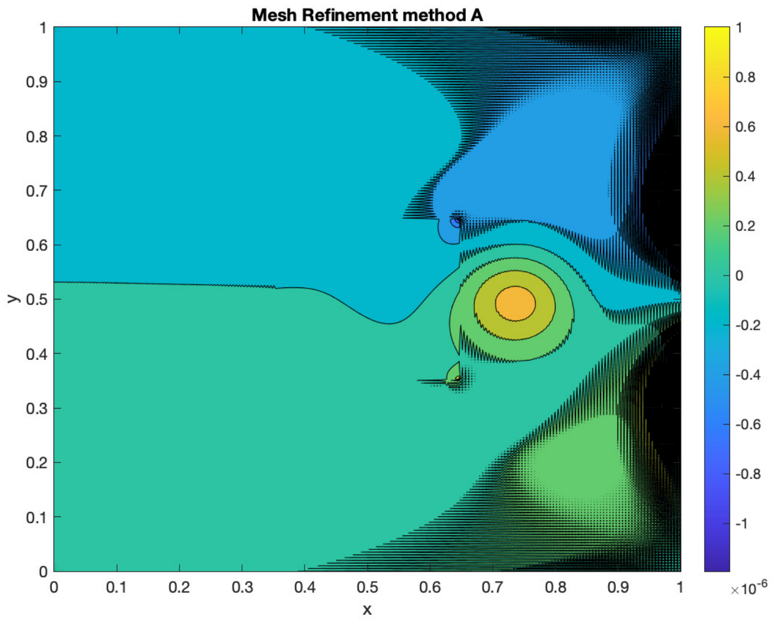

Figure 4.16: Error plot of density in vortex moving in x-direction for 400^2 points at time t = 0.07 with artificial diffusion constant $\lambda = 6.5$, simulated with mesh refinement Method A

these error plots. In the calculations the interface has indeed proven to have less accuracy, and for some points at the edges and the corners, the scheme is even inconsistent.

When it comes to the convergence rates of the three methods we observe that mesh refinement Method B and C is closer to 1 at all times, while the convergence of mesh refinement Method A is strictly less than 1, but climbs slowly up towards 1 as the number of points increases. Thus Method A has the lowest convergence rate of the three and is the least accurate mesh refinement method. We may assume that the stable method has its advantages from the stability which we have shown for the advection equation. Even though the scheme is not consistent in the diffusion at the interface it still converges. The convergence rates of mesh refinement Method B and C both approaches 1, but for the latter one it is even slightly above 1 for the speed components v_1 and v_2. Mesh refinement Method C is altogether the most accurate of the three. However, none of the methods works without diffusion, and the convergence of the non refined method is 2, which means that no

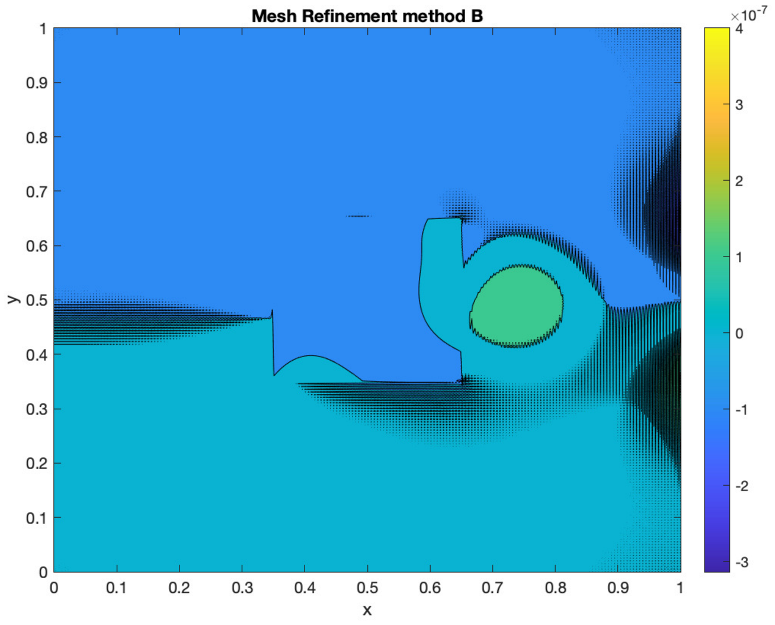

Figure 4.17: Error plot of density in vortex moving in x-direction for 400^2 points at time t = 0.07 with artificial diffusion constant $\lambda = 1.5$, simulated with mesh refinement Method B.

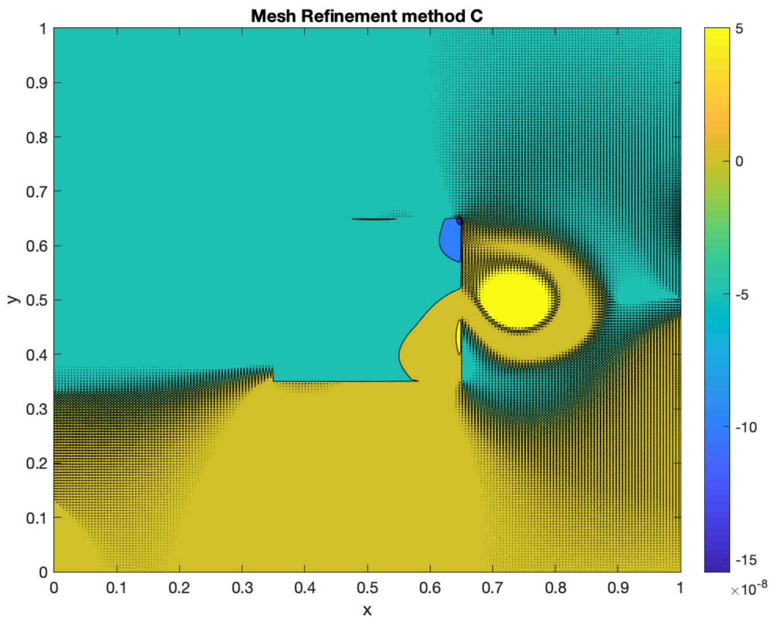

Figure 4.18: Error plot of density in vortex moving in x-direction with 400^2 points at time t = 0.07 with artificial diffusion constant $\lambda = 0.5$, simulated with mesh refinement Method C.

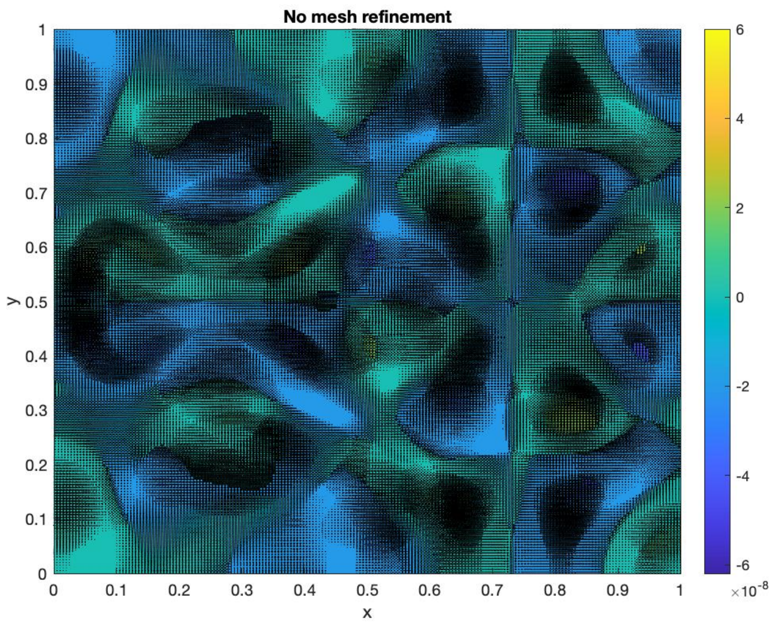

Figure 4.19: Error plot of density in votex moving in x-direction with no mesh refinement, 400^2 points at time t = 0.07 without artificial diffusion.

refinement still is the better option for these simulations when only considering the accuracy.

Still there might be many reasons to use mesh refinement. The computations done in this section are quite simple, concentrating on smaller problems, and there are no discontinuities. When more complex problems are calculated, the computations can take a lot of time, and it might be crucial to reduce this time. By using refinement in time, which we have not tested here, the computational overhead can be much reduced. Then mesh refinement might be a good option. Especially when there are shocks involved, in which case artificial diffusion must be applied anyway and a part of the accuracy is already lost.

4.5.3 Simulation of a radial explosion with the Euler equations

In this section we take the simulations one step further and compute a radial explosion with the Euler equations. In the previous section, we placed the refinement closely around the vortex, and the vortex traveled over the grid interface. This was a way to test the stability of the intersection between the coarse and fine grid. However, the fine meshes are often supplied such that the grid points with more error are interior to the fine grid. This means that the edges are located at smoother areas of the solution, which is not the case for the simulation with the Euler vortex. In this experiment we will try to simulate an explosion, and to test the refinement of the AMR method in a more justified case, the explosion will be located well inside the fine subgrid. With time the explosion is spreading out towards its edges, but stops before the boundaries of the fine grid are reached. We are using wall boundary conditions at the boundaries for these simulations, which seems to be stable for these simulations. A formal proof of stability can be found in [SN08]

Starting with the domain $[0,1] \times [0,1]$, the centre of the domain is $(x_c, y_c) = (0.5, 0.5)$. Given a radius of $R_v = 0.2$, the density within R_v of the center (x_c, y_c) is given by $\rho_{in} = 1$ and outside this radius the density is $\rho_{out} = 0.125$. Similarly the pressure inside is $p_{in} = 1$ and outside it is $p_{out} = 0.1$, so that the energy inside is $E = \frac{p_{in}}{gamma-1}$, and outside $E = \frac{p_{out}}{gamma-1}$. The velocity components are zero. The method is run from time $t_0 = 0$ to time $t_n = 0.065$, and the refinement is located at the square $[0.15, 0.85] \times [0.15, 0.85]$. Since the explosion is a shock, and the initial conditions are discontinuous, we have added artificial diffusion with diffusion coefficient $\lambda = 0.5$. We test the three mesh refinement methods A, B and C, and one simulation is without mesh refinement. All simulations have a 300×300 initial grid. The three mesh refinement methods A, B and C have a refined subgrid in the middle of the domain, as explained above, with refinement ratio 2. The simulations are compared with a 1197×1197 grid, which works as an approximated exact solution. The errors of the simulations are shown in table 4.6, and Figure 4.20 shows the explosion at time $t = 0.065$.

	Error			
variable	Method A	Method B	Method C	no mr.
ρ	6.150×10^{-2}	6.150×10^{-2}	6.325×10^{-2}	1.386×10^{-1}
v_1	5.122×10^{-2}	5.122×10^{-2}	5.293×10^{-2}	1.072×10^{-1}
v_2	5.122×10^{-2}	5.122×10^{-2}	5.293×10^{-2}	1.072×10^{-1}
E	1.836×10^{-1}	1.836×10^{-1}	1.895×10^{-1}	3.962×10^{-1}

Table 4.6: Error for mesh refinement Method A, B, C and no mesh refinement for radial explosion

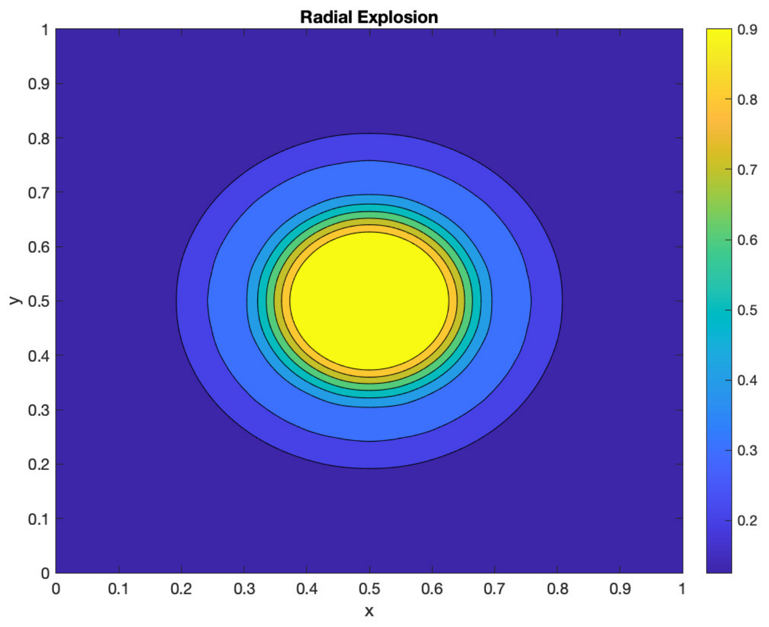

Figure 4.20: Contour of density in a radial explosion on a 300^2 grid at time $t = 0.065$.

We observe that for this experiment, the mesh refinement method works quite well. The challenge we faced earlier with the grid interface between the fine and coarse grid seems to be solved by locating the shock entirely inside the subgrid. While there is a great deal of change inside this patch of subgrid, there is no change outside the explosion before the shock has reached a given point. Thus the flux at this part of the grid, at the interface where the shock has not reached yet, is small or zero. The error of Method A and Method B is the same and the error for method C is similar, which is a result of the location of the fine grid.

This experiment shows that the mesh refinement can work well when applied in a specific way on the right problems. The radial explosion experiment is a good way to demonstrate this because we are able to capture the part of the domain where all of the change happens, while there is nothing happening outside the explosion. Still, something must be done when the shock from the explosion gets close to the wall, which is where the adaptive part of the method usually comes in. This problem is also in need of artificial diffusion due to the shock, and as we saw earlier, this gives the Mesh refinement methods an advantage.

Chapter 5

Staircase boundary

When dealing with Cartesian, uniform grids, there is another situation in which we need to reconsider the grid points. When we are dealing with a domain where the boundary is not aligned with the Cartesian blocks, we must adapt the grid points to this boundary in some sense. In this chapter we want to look at one structure used to solve this problem, which is a staircase boundary.

Consider the domain illustrated in Figure 5.1. The figure shows a squared domain with a sloping boundary in the lower left corner. A discretization of this domain is straightforward on a Cartesian grid except for the sloping edge. On this part of the boundary we need to decide how to represent the boundary through the grid points from the Cartesian grid. This leads a grid boundary that looks like in Figure 5.2, staircase formed. We want to analyze this approach for approximating non-uniform domains with Cartesian grids. First, we explain how to decide the discretization of the boundary.

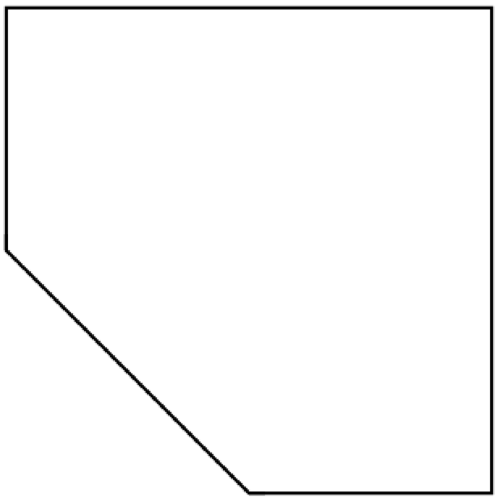
Figure 5.1: Domain with sloping boundary in lower left corner

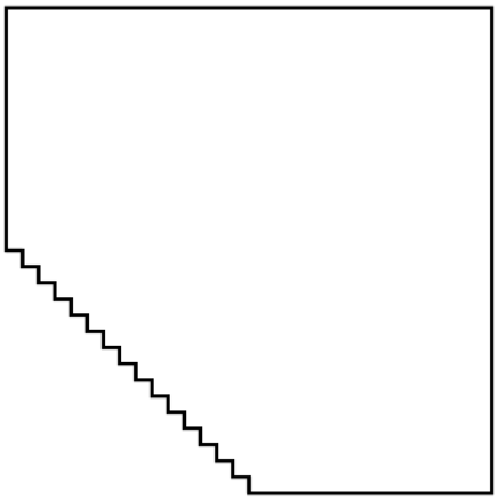
Figure 5.2: Computational domain with staircase boundary in lower left corner

5.1 Boundary calculations

As explained above, we need to approximate the sloping boundary with a staircase formed grid boundary. In these calculations we will assume that the boundary cuts off one corner of the grid as shown in Figure 5.1. To find the numerical boundary points we simply calculate the length from the boundary to the grid points nearby and choose the closest ones. Let the boundary be expressed by the equation

$$y = ax + b$$

By starting with the x-value, we have calculated the corresponding y-value above and below the line and chosen the one closest to the boundary. This results in a boundary looking like in Figure 5.2. The Figures 5.3 and 5.4 shows the result of these computations close up. In Figure 5.3 the boundary has a slope of $a = -1$ and since the grid is uniform Cartesian, the gridpoints at the boundary forms a uniform staircase where each point is separated by the same distance. In this particular case the boundary is located in the middle of two grid points, and we use the grid point lying directly above the boundary. Thus, the computational boundary becomes slightly misplaced. In Figure 5.4 the slope is $0 \leq a \leq -1$. In this case the boundary takes a different staircase form. In this figure we clearly see how the grid points closest to the line are chosen to form the boundary.

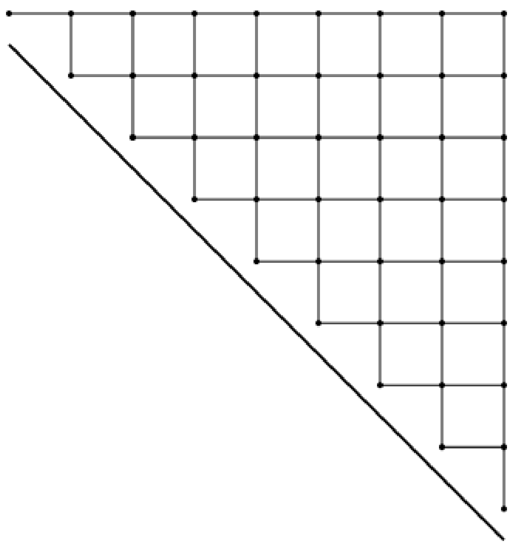
Figure 5.3: Staircase boundary calculated from boundary of a grid expressed by the line $y = ax + b$ with slope $a = -1$

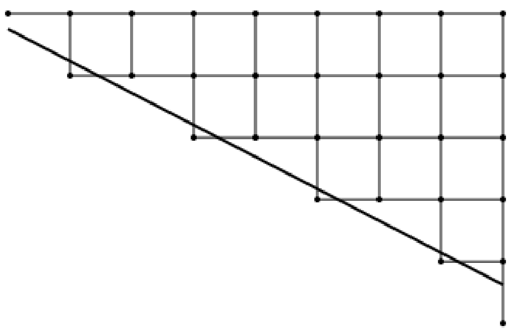
Figure 5.4: Staircase boundary calculated from boundary of a grid expressed by the line $y = ax + b$ with slope $0 \leq a \leq -1$

We present two different ways of calculating the boundary points, and we will refer to the resulting boundaries as Staircase Boundary A and Staircase Boundary B. For the first approach we consider each point at the staircase boundary and decide whether it is a boundary point in x- and y-direction or both, and calculate the flux accordingly. The boundary condition that is imposed in this case is the exact solution at the grid point (x_i, y_j) where the flux is calculated. Let (x_i, y_j) be one such point, chosen to represent the boundary. We assume that both the points (x_{i-1}, y_j) and (x_i, y_{j-1}) are outside the boundary. In this case we calculate the solution at boundary point (x_i, y_j) in the following way:

$$(u_{i,j})_t = -\frac{f(u_{i+1,j}) - f(u_{i,j}^{sol})}{2h} - \frac{g(u_{i,j+1}) - g(u_{i,j}^{sol})}{2h}$$

where $u_{i,j}^{sol}$ is the exact solution at point (x_i, y_j). This is how all the boundary points in Figure 5.3 would be calculated. For the example in Figure 5.4, some points are only boundary points in the y-direction, and they are calculated in the following way:

$$(u_{i,j})_t = -\frac{f(u_{i+1,j}) - f(u_{i-1,j})}{2h} - \frac{g(u_{i,j+1}) - g(u^{sol}_{i,j})}{2h}$$

and correspondingly in y-direction. This is Staircase Boundary A.

In the second case we do the same calculations as for the first case, but the boundary values are different. When approximating a continuous problem, one usually only has information of the boundary values from the exact location of the boundary, and not the exact solution in the entire domain, specifically at the grid boundary points, as in the prevous case. Since we are simulating a boundary with the staircase boundary, we should use the exact values from the line $y = ax + b$. The Staircase Boundary B is calculated this way. The scheme is the same as for Staircase Boundary A, but instead of $u^{sol}_{i,j}$ the exact solution is from the real boundary: $u^{sol}(x, y)$, $(x, y) \in \{(x, y)|y = ax + b\}$. Thus, the boundary values are misplaced with an error of order h. This is the case when doing simulations with the Euler equations where one of the boundaries represents a wall. At the wall, the velocity component normal to the wall is zero.

We want to see how the staircase formed boundary performs compared to a boundary aligned with the $x-$ or $y-$axis, and if this approach works for more complicated domains than squares and rectangles. It is also interesting to see how the change in boundary conditions in the two methods affects the solution.

5.2 Numerical results with staircase boundaries

We have used the Euler equations from Section 3.2 to compute an inviscid vortex. The variables and CFL condition are the same as described there, except for the angle α, the Mach number and the strength β. In these computations we have calculated two cases for the vortex. In the first case the vortex starts in the middle of the domain as before, and moves towards the staircase boundary with a mach number $Ma = -0.01$, an angle $\alpha = \frac{\pi}{4}$ and strength $\beta = 5$. This means that the vortex moves towards the boundary a distance of $0.5\cos(\frac{\pi}{4}) = \frac{\sqrt{2}}{4}$ and passes the boundary with an angle of π radians. The vortex is computed on the domain

$$0 \leq x, y \leq 1$$
$$y \geq -x + 0.5$$

The computational domain is $m \times m$ squared domain with a staircase boundary calculated as described in the previous section for the two approaches. The domain described looks like in Figure 5.1 from the previous section. In the second case the vortex starts at initial position $(x_c, y_c) = (\frac{\sqrt{2}}{4}, \frac{1}{2})$ and travels with Mach number $Ma = -0.01$ on the square domain $0 \leq x, y \leq 1$, without a staircase boundary. Hence, the vortex travels the same distance $\frac{\sqrt{2}}{4}$ to the boundary, but in the first case the boundary is a staircase boundary, and in the second case the boundary is a normal boundary. This is illustrated in Figures 5.5 and 5.6, where the initial values of the density is plotted. The simulations run from time $t = 0$ to $t = 0.12$. The error and convergence of the simulations are shown in Table 5.1 for case A, Table 5.2 for case B and Table 5.3 for the normal boundary.

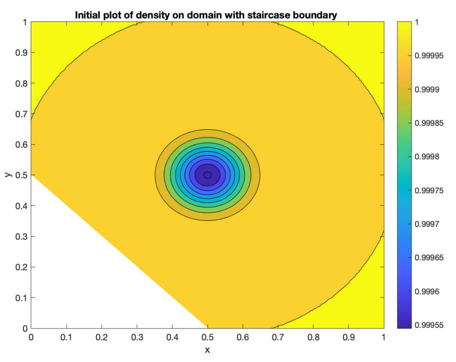

Figure 5.5: Initial plot of density for domain with staircase boundary

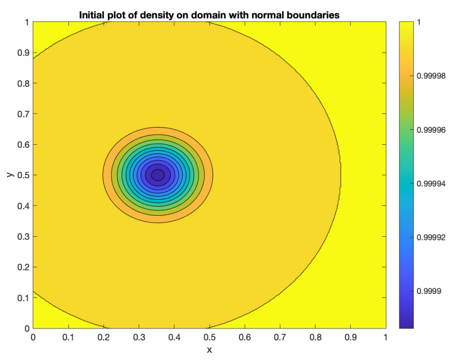

Figure 5.6: Initial plot of density for domain with normal boundaries

The results shows that staircase boundary B is less accurate than Staircase Boundary A and the normal boundary. Figure 5.7 shows a plot of the errors for Staircase Boundary A and B, and for the normal boundary. We observe that boundary A is more accurate than boundary B. Initially the error is similar, but due to the higher convergence, the error of boundary A decreases and reaches the same level as the normal boundary.

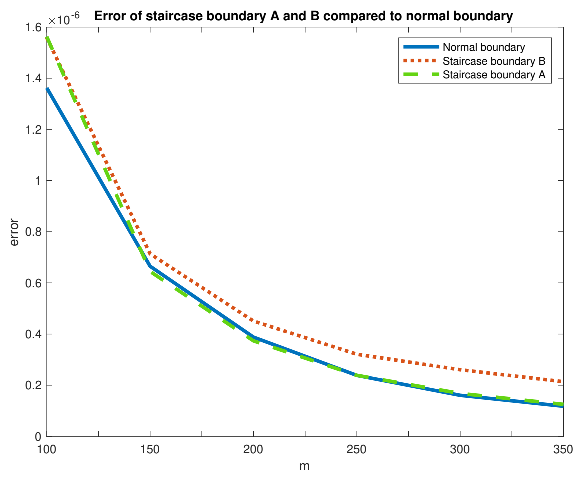

Figure 5.7: Error of Staircase Boundary A, B and normal boundary

The convergence rate for boundary A is 2. For boundary B the velocity converges with a rate of 1, and for the density and energy it starts out higher but decays towards 1 as well. As seen in the tables and the error plot, Staircase Boundary A and the normal boundary have similar accuracy and the same convergence rate of 2. Method B has its faults in that the grid structure does not match the shape of the domain as indicated in the previous section, and that there is an error in the boundary data, leading to a loss of accuracy for the method.

Staircase Boundary A				
Error				
m	ρ	v_1	v_2	E
100	1.562×10^{-6}	1.016×10^{-2}	1.318×10^{-2}	4.246×10^{-1}
150	6.447×10^{-7}	4.398×10^{-3}	5.717×10^{-3}	1.755×10^{-1}
200	3.734×10^{-7}	2.444×10^{-3}	3.182×10^{-3}	1.009×10^{-1}
250	2.386×10^{-7}	1.551×10^{-3}	2.025×10^{-3}	6.405×10^{-2}
300	1.679×10^{-7}	1.070×10^{-3}	1.401×10^{-3}	4.521×10^{-2}
350	1.251×10^{-7}	7.804×10^{-4}	1.027×10^{-3}	3.356×10^{-2}
400	9.454×10^{-8}	5.956×10^{-4}	7.839×10^{-4}	2.546×10^{-2}

Convergence				
m	ρ	v_1	v_2	E
100	-	-	-	-
150	2.183	2.065	2.060	2.179
200	1.898	2.042	2.037	1.926
250	2.007	2.039	2.026	2.034
300	1.929	2.034	2.021	1.911
350	1.906	2.048	2.015	1.932
400	2.100	2.0245	2.0213	2.068

Table 5.1: Error and convergence for simulations of vortex moving towards a staircase boundary calculated with case A

Staircase Boundary B				
Error				
m	ρ	v_1	v_2	E
100	1.559×10^{-6}	3.247×10^{-2}	2.816×10^{-2}	4.360×10^{-1}
150	1.136×10^{-7}	2.211×10^{-2}	2.001×10^{-2}	2.054×10^{-1}
200	4.507×10^{-7}	1.651×10^{-2}	1.534×10^{-2}	1.346×10^{-1}
250	3.210×10^{-7}	1.335×10^{-2}	1.261×10^{-2}	9.858×10^{-2}
300	2.605×10^{-7}	1.105×10^{-2}	1.054×10^{-2}	7.976×10^{-2}
350	2.134×10^{-7}	9.557×10^{-3}	9.192×10^{-3}	6.581×10^{-2}
400	1.860×10^{-7}	8.299×10^{-3}	8.023×10^{-3}	5.716×10^{-2}

Convergence				
m	ρ	v_1	v_2	E
100	-	-	-	-
150	1.927	0.948	0.836	1.857
200	1.597	1.015	0.932	1.468
250	1.521	0.953	0.879	1.396
300	1.147	1.040	0.986	1.161
350	1.284	0.939	0.885	1.248
400	1.040	1.057	1.018	1.055

Table 5.2: Error and convergence for simulations of vortex moving towards a staircase boundary calculated with case B

Normal boundary				
Error				
m	ρ	v_1	v_2	E
100	1.362×10^{-6}	8.260×10^{-3}	1.168×10^{-2}	3.744×10^{-1}
150	6.650×10^{-7}	3.616×10^{-3}	4.989×10^{-3}	1.828×10^{-1}
200	3.879×10^{-7}	2.017×10^{-3}	2.765×10^{-3}	1.066×10^{-1}
250	2.384×10^{-7}	1.285×10^{-3}	1.756×10^{-3}	6.556×10^{-2}
300	1.605×10^{-7}	8.900×10^{-4}	1.213×10^{-3}	4.413×10^{-2}
350	1.178×10^{-7}	6.524×10^{-4}	8.882×10^{-4}	3.237×10^{-2}
400	9.073×10^{-8}	4.986×10^{-4}	6.781×10^{-4}	2.494×10^{-2}

Normal boundary				
Convergence				
m	ρ	v_1	v_2	E
100	-	-	-	-
150	1.768	2.037	2.098	1.769
200	1.874	2.029	2.051	1.873
250	2.181	2.019	2.036	2.180
300	2.170	2.016	2.027	2.171
350	2.009	2.015	2.023	2.010
400	1.953	2.013	2.021	1.954

Table 5.3: Error and convergence for simulations of vortex moving towards a normal boundary in the negative x-direction

Chapter 6

Conclusion

In this thesis we have looked at the Adaptive Mesh Refinement method and studied different ways to refine meshes. Three mesh refinement methods has been considered, two of which were reconstructions of approaches from [BO84] and [BC89], and the last one was constructed in order to satisfy the stability calculations using the Finite Volumes scheme.

The three MR methods have given varying results. The first method presented, Mesh refinement A, was unstable without artificial diffusion in the first two simulations, but still had equally good accuracy for the advection equation. With artificial diffusion it achieved a convergence of 1 for both simulations, as did the other two methods. For the Euler vortex, we saw the biggest difference between the methods. Mesh Refinement Method C gives the best results for these simulations. It requires the least diffusion of all the methods, and generally had the most accurate solution. Mesh refinement Method B is the second best, and we believe some of its success to follow from the stability which was proven in Section 4.2. But even so, neither of the methods are able to achieve the accuracy of the finite volumes method without mesh refinement. There is a great loss of accuracy in such an abrupt refinement of the mesh, which causes all the methods to drop in convergence. All three methods has convergence 1, while the method without any mesh refinement has a convergence of 2. This means that when looking for the most accurate method as a main priority, one might not want to use mesh refinement for this experiment. We have done a second simulation with the Euler equations, which turned out to be more in favour of the AMR method. This was a simulation of a radial explosion, and we have seen that when locating the grid interfaces well outside the explosion where there is little to no change in the solution, we achieve considerably better results for the Mesh Refinement method. Many problems in computational fluid dynamics, such as this one, include shocks or discontinuities, which means that the accuracy is automatically compromised when adding artificial diffusion. Thus, there are different reasons to use the AMR method, depending on the problem.

We have also looked at staircase boundaries as a method for approximating sloping boundaries of a Cartesian grid. This has worked well when supplied with an exact solution, so that the boundary values can be correctly located at the numerical boundary, as done in Staircase Boundary A. However, when approximating the boundary with slightly shifted boundary values, as is often the case for many problems in fluid dynamics, the accuracy and convergence rate are reduced.

Bibliography

[Ach91] David J Acheson. Elementary fluid dynamics, 1991.

[And03] John D Anderson. *Modern compressible flow.* Tata McGraw-Hill Education, 2003.

[BC89] Marsha J Berger and Phillip Colella. Local adaptive mesh refinement for shock hydrodynamics. *Journal of computational Physics*, 82(1):64–84, 1989.

[Ber85] Marsha Berger. Stability of interfaces with mesh refinement. *Mathematics of Computation*, 45:301–318, October 1985.

[Ber86] Marsha J Berger. Data structures for adaptive grid generation. *SIAM Journal on Scientific and Statistical Computing*, 7(3):904–916, 1986.

[BO84] Marsha J Berger and Joseph Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of computational Physics*, 53(3):484–512, 1984.

[Eva10] Lawrence Evans. *Partial differential equations.* American Mathematical Society, Providence, R.I, 2010.

[Gus07] Bertil Gustafsson. *High order difference methods for time dependent PDE*, volume 38. Springer Science & Business Media, 2007.

[KS92] Heinz-Otto Kreiss and Godela Scherer. Method of lines for hyperbolic differential equations. *SIAM Journal on Numerical Analysis*, 29(3):640–646, 1992.

[Kun15] Pijush Kundu. *Fluid mechanics.* Academic Press, Amsterdam, 2015.

[KW93] Heinz O Kreiss and Lixin Wu. On the stability definition of difference approximations for the initial boundary value problem. *Applied Numerical Mathematics*, 12(1-3):213–227, 1993.

[Nor08] Jan Nordström. Error bounded schemes for time-dependent hyperbolic problems. *SIAM Journal on Scientific Computing*, 30(1):46–59, 2008.

[Pre92] William H. Press. *Numerical Recipes in C: The Art of Scientific Computing, Second Edition.* Cambridge University Press, oct 1992.

[RM67] Robert D RICHTMYER and KW MORTON. Difference methods for initial-value problems. Technical report, Interscience Publishers, 1967.

[SN08] Magnus Svärd and Jan Nordström. A stable high-order finite difference scheme for the compressible navier–stokes equations: no-slip wall boundary conditions. *Journal of Computational Physics*, 227(10):4805–4824, 2008.

[Svä21] Magnus Svärd. Entropy stable boundary conditions for the euler equations. *Journal of Computational Physics*, 426:109947, 2021.

[Whi11] Frank White. *Fluid mechanics.* McGraw Hill, New York, N.Y, 2011.