NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

BSc THESIS

Creating CFD simulations from scratch using the Finite Volume Method

Panagiotis G. Iatrou

Supervisors: **Ioannis Emiris,** Professor NKUA
Nektarios Vlahakis, Professor NKUA

ATHENS

MAY 2025

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Δημιουργία ΥΔΡ προσομοιώσεων από το μηδέν χρησιμοποιώντας την Μέθοδο Πεπερασμένων Όγκων

Παναγιώτης Γ. Ιατρού

Επιβλέποντες: **Ιωάννης Εμίρης,** Καθηγητής ΕΚΠΑ
Νεκτάριος Βλαχάκης, Καθηγητής ΕΚΠΑ

ΑΘΗΝΑ

ΜΑΪΟΣ 2025

BSc THESIS

Creating CFD simulations from scratch using the Finite Volume Method

Panagiotis G. Iatrou

S.N.: 1115201900065

SUPERVISORS: **Ioannis Emiris,** Professor NKUA
Nektarios Vlahakis, Professor NKUA

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Δημιουργία ΥΔΡ προσομοιώσεων από το μηδέν χρησιμοποιώντας την Μέθοδο Πεπερασμένων Όγκων

Παναγιώτης Γ. Ιατρού

Α.Μ.: 1115201900065

ΕΠΙΒΛΕΠΟΝΤΕΣ: **Ιωάννης Εμίρης,** Καθηγητής ΕΚΠΑ
Νεκτάριος Βλαχάκης, Καθηγητής ΕΚΠΑ

ABSTRACT

Computational Fluid Dynamics (CFD) software packages are very commonly used to tackle real-world CFD problems. Most of the times, they are being used without proper knowledge of how they work behind the scenes. However, this is an important aspect of being a professional CFD engineer, since it leads to a deeper understanding of how and why certain issues arise during the modeling and simulation of a problem. This guide aims to do just that: provide a detailed guide that helps the user to understand exactly how these software packages work. After reading this guide, the user should feel comfortable in making a CFD software package from scratch. This is neither an easy nor a short task. However, this guide aims to make the task very intuitive, by providing every step necessary.

SUBJECT AREA: Computational Fluid Dynamics (CFD)

KEYWORDS: cfd, fvm, navier-stokes, simulation, guide

ΠΕΡΙΛΗΨΗ

Τα Υπολογιστικά πακέτα Υπολογιστικής Δυναμικής Ρευστών (ΥΔΡ) χρησιμοποιούνται για την επίλυση πραγματικών προβλημάτων. Τις περισσότερες φορές, χρησιμοποιούνται χωρίς την γνώση ως προς το πως λειτουργούν. Παρ'όλα αυτά, η γνώση αυτή είναι ένα σημαντικό στοιχείο ενός Μηχανικού ΥΔΡ, καθώς οδηγεί σε βαθιά κατανόηση του πως και γιατί μερικά θέματα προκύπτουν κατά την μοντελοποίηση και την προσομοίωση προβλημάτων. Αυτός ο οδηγός έχει φτιαχτεί για να βοηθήσει τον χρήστη να καταλάβει πως ακριβώς λειτουργούν τέτοιου είδους υπολογιστικά πακέτα. Κατόπιν της ανάγνωσης, ο χρήστης θα είναι πλήρως προετοιμασμένος για να φτιάξει ένα τέτοιο πακέτο. Η διαδικασία ενδέχεται να μην είναι εύκολη ή σύντομη, αλλά ο οδηγός αυτός σκοπεύει να δώσει μια προσέγγιση η οποία είναι εύκολα κατανοητή, προσφέροντας τις απαραίτητες γνώσεις, βήμα προς βήμα.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Υπολογιστική Δυναμική Ρευστών (ΥΔΡ)

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: υδρ, μπο, νάβιερ-στόουκς, προσομοίωση, οδηγός

ACKNOWLEDGEMENTS

Θα ήθελα να ευχαριστήσω των επιβλέποντα Ιωάννη Εμίρη, καθηγητή Πληροφορικής στο ΕΚΠΑ για την καθοδήγηση που παρείχε στην διεκπεραίωση της τρέχουσας πτυχιακής εργασίας. Παράλληλα, θα ήθελα να ευχαριστήσω τον Νεκτάριο Βλαχάκη, καθηγητή Φυσικής στο ΕΚΠΑ, καθώς και την Ομάδα Φυσικής Ωκεανογραφίας και Αριθμητικών Μοντέλων του τμήματος Φυσικής στο ΕΚΠΑ, για την αποφασιστή συμβολή τους καθ'όλη την διάρκεια της εκπόνησης της.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 What is CFD?

Computational fluid dynamics (or CFD) [1] is a branch of fluid mechanics that leverages the power of computers to simulate fluid flow problems. It has seen a rapid rise in use in recent years as computational resources are constantly growing.

1.2 Why Even Bother With CFD?

A fair question to ask is why do we even need to worry about solving fluid flow problems within a computer [49]. Indeed, physically simulating the problem sounds like a natural idea at first, especially when considering the main drawback of computer simulations: accuracy. Physically simulating the problem yields the maximum possible accuracy. Physical simulations, though, quickly become practically impossible as the scale and complexity of the problems increase. An example of an impractical problem to simulate physically would be the aerodynamics of an airfoil. Let's unfold its biggest drawbacks:

- It is very expensive to produce an airfoil, let alone to make it fit a specific type of plane. Even then, no one can guarantee that the airfoil will not split in half during a simulation, putting a huge toll on the cost. All these are just in the test stage of manufacturing.
- Further modifications are really hard and time-consuming. When designing airfoils, it is imperative to check variations in the shape of the airfoil in order to find the optimal one.

Designing an airfoil is just an example, and it is easy to find similar impractical fluid flow problems, rendering physically simulating fluid flow problems for most real problems impractical. So, this time, let's unfold CFD's main advantages:

- Even though accuracy is inferior compared to physical simulations, we can achieve very high accuracy by leveraging supercomputers, which is usually more than enough.
- Modifications to the simulation environment can be done within a short amount of time, saving both human resources and cost, as well as time.
- CFD is suitable for practically all fluid flow problems, as the simulation environment is easily configurable.
- It's fun!

Considering the above, it becomes quite clear why using CFD is currently our best bet for simulating fluid flows for most problems.

1.3 Background and Related Work

There is an abundance of resources and textbooks on CFD on the internet. So, a fair question is, why publish another one? While writing this thesis, I found it immensely hard to get into the field of CFD. There are many reasons for this, and they are the main driving factors that led me to do it all.

- CFD is a very complex and deep field of study.
- There are not a lot of beginner guides that cover all of the necessary topics, so jumping between resources can be frustrating. This guide acts as a complete introduction to the field of CFD.
- Most of the resources available are aimed at advanced CFD or people with a solid and thorough background and understanding of Fluid Mechanics or Computer Science. Looking back, through all the time I spent making this thesis, it was clear that none of these prerequisites were necessary. Of course, having a solid background in those can immensely help and is truly what separates the amateur from the professional, but in no case should it not be accessible.
- Reading through resources, one can easily get overwhelmed by the lingo people use in the field. Intentional or not, it poses significant discomfort to the learner. This guide is focused on making a straightforward introduction to the field of CFD.

1.4 Prerequisites and Focus of This Guide

As mentioned, this guide focuses on making a straightforward introduction to the field of CFD. This also includes making it as accessible as possible to various study backgrounds. Naturally, though, a Physics or Computer Science background will make things easier. Before following this guide, though, some things should be considered necessary:

- Knowledge of at least one programming language. Popular choices are Python and C++. The reader should be relatively comfortable in terms of programming.
- Multivariable calculus, while not absolutely necessary, will make the math look a lot less scary. By design, Fluid Mechanics contains lots of math, and avoiding it all is hard. Most of the math or physics included in this guide will be explained, though if the reader wants to focus on just making a CFD software, it should be feasible to skip some now and then (though not recommended).

2. THEORETICAL BACKGROUND

2.1 General info

2.1.1 Scalar and Vector Fields

During this guide, we will constantly be dealing with scalar and vector fields. Let's take a moment to clarify those terms.

A **scalar field** can be thought of as a function that maps a set of variables to a single value. An example of a scalar field is the pressure field. In 2D, one can think of it as two variables (x and y) mapping to a single value (the value of the pressure p on the point (x, y)). For instance, figure 2.1 shows a scalar field that represents the value of each point using color.

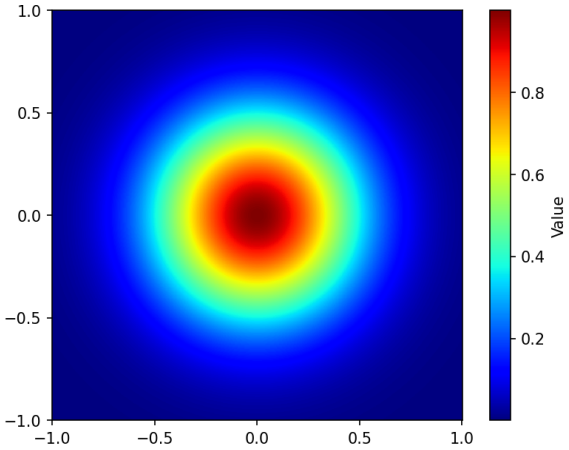


Figure 2.1: Example of a Scalar field | $F(x, y) = -4 \cdot (x^2 + y^2)$

A **vector field** can be thought of as a function that maps a set of variables to a set of values. A vector field example is the velocity field $\vec{v} = (u, v)$. In 2D, one can think of it as two variables (x and y) mapping to 2 values (the value of the x-component u and y-component v on the point (x, y)). Since each point in space has two values, a helpful way to represent it is with arrows, as shown in Figure 2.2

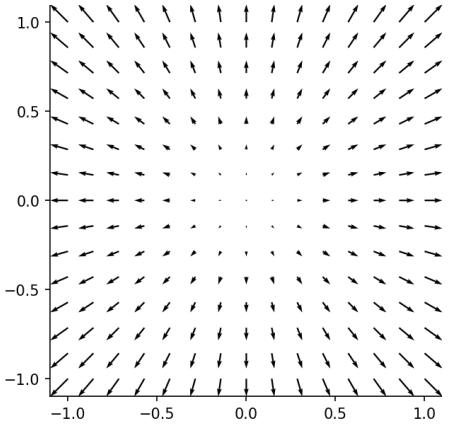


Figure 2.2: Example of a Vector field | $F(x, y) = (x, y)$

2.1.2 Eulerian specification

According to the Eulerian specification of the flow field, the properties of the flow field are functions of space and time. For example, the x-component of velocity u is written as $u(x, y, t)$. Most of the time, though, it is simply written just as u, so as not to bloat the equations when they get bigger.

2.1.3 Incompressibility

This guide is focused on simulating incompressible fluids. The term incompressibility is used to describe fluids that can't be compressed. In the context of physics, it means that the density ρ of the fluid remains constant. In reality, all fluids are compressible, but it is generally considered an acceptable approximation to think of fluids as incompressible.

It should be noted, however, that this approximation can only be justified under certain circumstances. Compressibility usually doesn't have significant effects on the flow of the fluid when the velocity of the flow is not high enough. A rule of thumb is that if the velocity of the fluid is greater than $0.3 \cdot Mach$, then a compressible approach should be considered as the effects of the compressibility start to increase. Otherwise, when the velocity of the fluid is smaller than $0.3 \cdot Mach$, it is a good approximation to follow an incompressible approach.

Compressible flows are significantly more complex to simulate and are more costly in terms of computational power. Thus, in this guide, we will assume the fluids to be incompressible. In addition, one could argue that for most everyday simulations, the incompressible assumption is good enough since most of them do not exhibit such high magnitudes of velocity.

For instance, in a scenario in which we want to simulate the aerodynamics of a car, the limit of $0.3 \cdot Mach$ that we set earlier accounts for speeds not greater than $\approx 103 m/s$ or $\approx 370 km/h$. Most cars produced (when making this guide) don't even come close to this limit.

2.1.4 Laminar and turbulent flow

Flow can be categorized as either laminar or turbulent [9].

- **Laminar** flow is smooth and predictable, without abrupt fluctuations.
- **Turbulent** flow is the opposite, chaotic, and unpredictable. It is sensitive to disturbances and often results in swirls and vortices.

Predicting whether a flow will reach a laminar or a turbulent regime is difficult, as different flows behave differently depending on various factors. A way often used to predict the flow regime is the **Reynolds number**, which we will discuss in further detail in 2.2.3.

2.1.5 Steady and unsteady state simulations

A distinction that will arise later is the difference between steady and unsteady state simulations.

- **Unsteady state** simulations are the most intuitive. We start with an initial condition (initial timestep) and progress through time, calculating the next timestep using the information from the latest timestep. Therefore, the result of an unsteady state simulation consists of multiple timesteps that can be thought of as instances in time or frames in an animation. This type of simulation considers the effects of time inside the equations themselves. They are by far the most common type of simulations.
- **Steady state** simulations do not take into consideration the effects of time in the equations and try to skip the whole process by calculating the final state of the fluid flow, as it would be after running an unsteady simulation for a really long time (as time approaches infinity). In other words, they try to find the steady state of the fluid flow in which progressing time would have no effect at all. This simulation type is not as common since it has a big flaw. The vast majority of real fluid flow problems are turbulent by nature and therefore do not reach a steady fluid flow state after any amount of time. Some of them are oscillatory. These fluid flow problems cannot be tackled using steady-state simulations

2.2 Fluids

2.2.1 Fluid Properties

When talking about fluids, we consider the following fundamental properties

- **Density** - It measures how tightly the particles are packed in a fluid. Specifically, when talking about incompressible fluids, the density is considered constant throughout the fluid domain. In SI, it is measured in kilograms per cubic meter (kg/m^3).
- **Viscosity** - It is a measure of the internal friction caused by the fluid molecules [16]. Inherently, it smooths out the fluid flow, effectively restricting its motion. Different fluids have different values of viscosity. For example, honey has higher viscosity compared to water. Therefore, it is more viscous and flows in a relatively restricted manner.

  Fluids with constant viscosity throughout their domain are considered **Newtonian**. Examples of Newtonian fluids are water, air, and honey. Fluids whose viscosity varies are considered **Non-Newtonian**. Examples of non-Newtonian fluids are ketchup and toothpaste.

  For this guide, we will be dealing exclusively with Newtonian fluids.

  Viscosity (or **dynamic viscosity**) μ is measured in SI in newton-seconds per square meters ($N \cdot s/m^2$). **Kinematic viscosity** ν is defined as the dynamic viscosity divided by density and is measured in SI in square meters per second (m^2/s).

2.2.2 Flow properties

When talking about flows, we consider the following fundamental properties.

- **Velocity** - Describes the speed and direction of the fluid. Different points in the fluid domain may have different velocity values. In SI, its magnitude is measured in meters per second (m/s).

- **Pressure** - Describes the pressure of the fluid. Different points in the fluid domain may have different pressure values. In SI, it is measured in pascal (Pa).

2.2.3 Reynolds number

The Reynolds number [11] is one of the most important numbers when it comes to Fluid Mechanics. It is a dimensionless quantity, meaning it has no measurement unit. Its value defines the behavior of the flow. It can be calculated using the formula

$$Re = \frac{\rho u L}{\mu} \tag{2.1}$$

where L is the characteristic length, u the characteristic velocity, ρ the density and μ the dynamic viscosity. The term *characteristic* is used to describe the value of a variable that is considered representative. It might seem a bit vague at first, but it will make more sense further down the line when we start talking about real flow problems.

Lower values of the Reynolds number are associated with laminar flow, while higher values are associated with turbulent flow. It should be noted, though, that finding the turning point in the Reynolds number at which the flow turns from laminar to turbulent varies significantly from problem to problem, and there is no good way to predict it, other than empirically.

2.3 The Incompressible Navier-Stokes Equations

The Navier-Stokes equations [23] describe the motion of fluids. The **incompressible Navier-Stokes** equations are a special case of the Navier-Stokes equations where fluids are considered incompressible. The term incompressible is described in the previous section 2.1.3. It is a set (or system) of partial differential equations that can be described as

$$\begin{aligned} \nabla \cdot \vec{v} &= 0 \\ \rho\left[\frac{\partial \vec{v}}{\partial t} + (\vec{v} \cdot \nabla)\vec{v}\right] &= -\nabla p + \mu \nabla^2 \vec{v} \end{aligned} \tag{2.2}$$

It should be noted that this is the differential form of the equations. There is also an integral form, which is usually not used when dealing with CFD.

These are the equations we will be dealing with for the rest of this guide, so it is important to understand them before diving deep into solving them. At first, they might seem daunting, so let's break them down into parts. At the same time, we will expand those in a way that makes it easier to solve them numerically later on.

2.4 The Continuity Equation

Taking the first equation of (2.2), we have

$$\nabla \cdot \vec{v} = 0 \tag{2.3}$$

This small equation ensures that no mass is created or destroyed. Mathematically, this equation states that the **Divergence** [5] of the velocity field is equal to 0. For instance, let's consider what different values of divergence at a specific point could mean. In the context of the figure 2.3, the divergence is positive on the left, while on the right, it is negative.

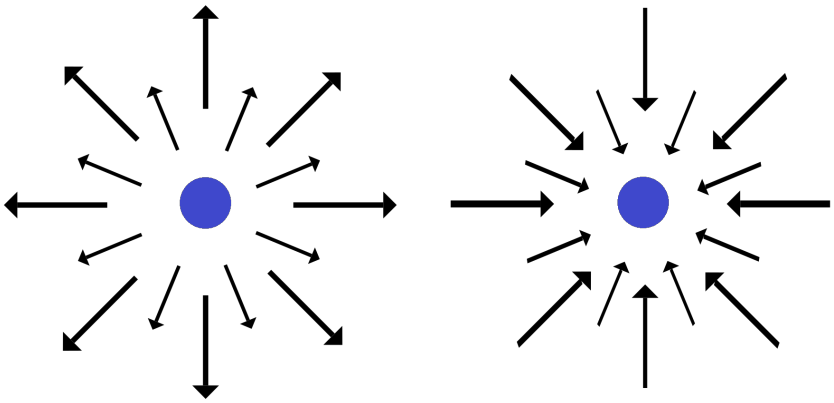


Figure 2.3: Positive divergence (left) and negative divergence (right)

Holding the assumption of incompressible flows, in the context of the figure 2.3, considering the arrows represent the velocity of the flow, we can conclude that mass is created when the divergence is positive. When it is negative, then mass is destroyed. It can sometimes be useful to think of positive divergence as a source and negative divergence as a sink. It is therefore intuitive to think of the divergence being 0 as mass neither being created nor destroyed, so mass is conserved.

It turns out that the form 2.3 of the equation is unsuitable when solving numerically. We will expand it in 2D, as follows.

$$\begin{aligned} \nabla \cdot \vec{v} &= 0 \\ \implies \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} &= 0 \end{aligned} \tag{2.4}$$

The equation 2.4 is the version of the continuity equation we will use later to solve for the flow fields numerically.

2.5 The momentum equation

Taking the second equation of (2.2), we have

$$\rho \left[\frac{\partial \vec{v}}{\partial t} + (\vec{v} \cdot \nabla)\vec{v} \right] = -\nabla p + \mu \nabla^2 \vec{v} \tag{2.5}$$

$$\implies \rho \frac{\partial (\vec{v})}{\partial t} + \rho \cdot (\vec{v} \cdot \nabla)(\vec{v}) = -\nabla p + \mu \nabla^2 \vec{v} \tag{2.6}$$

This equation ensures that momentum is conserved. It is significantly more complex than the continuity equation in 2.4. There are 2 equations hidden in this single equation (in 2D). To see how, we will first expand each term individually, the same way we did for the continuity equation.

2.5.1 Diffusion

$$\mu \nabla^2 \vec{v} \tag{2.7}$$

Notice how this is part of the equation (2.6). It takes into consideration the **viscosity** μ, which is a property of the fluid as described in 2.2.

As with the mass continuity equation, (2.7) in this form is unsuitable for solving numerically. We will expand it in 2D, as follows.

$$\begin{aligned}
\mu \nabla^2 \vec{v} &= \begin{bmatrix} \mu \nabla^2 u \\ \mu \nabla^2 v \end{bmatrix} = \begin{bmatrix} \mu \nabla \cdot \nabla u \\ \mu \nabla \cdot \nabla v \end{bmatrix} = \begin{bmatrix} \mu \nabla \cdot \left(\dfrac{\partial u}{\partial x}, \dfrac{\partial u}{\partial y} \right) \\ \mu \nabla \cdot \left(\dfrac{\partial v}{\partial x}, \dfrac{\partial v}{\partial y} \right) \end{bmatrix} \\
&= \begin{bmatrix} \mu \left(\dfrac{\partial}{\partial x} \left(\dfrac{\partial u}{\partial x} \right) + \dfrac{\partial}{\partial y} \left(\dfrac{\partial u}{\partial y} \right) \right) \\ \mu \left(\dfrac{\partial}{\partial x} \left(\dfrac{\partial v}{\partial x} \right) + \dfrac{\partial}{\partial y} \left(\dfrac{\partial v}{\partial y} \right) \right) \end{bmatrix} = \begin{bmatrix} \mu \dfrac{\partial}{\partial x} \left(\dfrac{\partial u}{\partial x} \right) + \mu \dfrac{\partial}{\partial y} \left(\dfrac{\partial u}{\partial y} \right) \\ \mu \dfrac{\partial}{\partial x} \left(\dfrac{\partial v}{\partial x} \right) + \mu \dfrac{\partial}{\partial y} \left(\dfrac{\partial v}{\partial y} \right) \end{bmatrix}
\end{aligned} \tag{2.8}$$

Notice how the diffusion term of the momentum equation consists of 2 components, an x-component and a y-component, one for each momentum equation. Always remember that the velocity vector field $\vec{v}$ has two components: the scalar fields u and v.

2.5.2 Convection

$$(\vec{v} \cdot \nabla)\vec{v} \tag{2.9}$$

Also known as advection. Notice how this is part of the equation (2.6). It physically transports the velocity field by the velocity itself. For example, imagine the velocity vector field of any flow at an instance in time. In the next time instance, the convection term will update the velocity field using the previous value of the velocity field, effectively ”moving” the flow field. Mathematically, $\vec{v} \cdot \nabla$ is the convection operator which is applied to the velocity field $\vec{v}$ itself .

As with diffusion, (2.9) in this form is unsuitable for solving numerically. We will expand it in 2D, as we did for diffusion, as follows.

$$(\vec{v} \cdot \nabla)\vec{v} = \begin{bmatrix} (\vec{v} \cdot \nabla)u \\ (\vec{v} \cdot \nabla)v \end{bmatrix} = \begin{bmatrix} \left(u \dfrac{\partial}{\partial x} + v \dfrac{\partial}{\partial y} \right) u \\ \left(u \dfrac{\partial}{\partial x} + v \dfrac{\partial}{\partial y} \right) v \end{bmatrix} = \begin{bmatrix} u \dfrac{\partial u}{\partial x} + v \dfrac{\partial u}{\partial y} \\ u \dfrac{\partial v}{\partial x} + v \dfrac{\partial v}{\partial y} \end{bmatrix} \tag{2.10}$$

Now, we are going to make use of the expanded continuity equation (2.4). Since the left hand side is equal to 0, we can add $u\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right)$ to the x-component of (2.10) and $v\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right)$ to the y-component of (2.10).

$$(2.10) = \begin{bmatrix} u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}+u\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right) \\ u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}+v\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right) \end{bmatrix} = \begin{bmatrix} u\frac{\partial u}{\partial x}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}+u\frac{\partial v}{\partial y} \\ u\frac{\partial v}{\partial x}+v\frac{\partial u}{\partial x}+v\frac{\partial v}{\partial y}+v\frac{\partial v}{\partial y} \end{bmatrix} \tag{2.11}$$

By making use of the product rule, we arrive at

$$(2.11) = \begin{bmatrix} \frac{\partial(uu)}{\partial x}+\frac{\partial(vu)}{\partial y} \\ \frac{\partial(uv)}{\partial x}+\frac{\partial(vv)}{\partial y} \end{bmatrix} \tag{2.12}$$

The convection term on the momentum equations is also multiplied by density.

$$\rho\cdot(\vec{v}\cdot\nabla)\vec{v} = \begin{bmatrix} \rho\left(\frac{\partial(uu)}{\partial x}+\frac{\partial(vu)}{\partial y}\right) \\ \rho\left(\frac{\partial(uv)}{\partial x}+\frac{\partial(vv)}{\partial y}\right) \end{bmatrix} \tag{2.13}$$

Notice how, similar to the diffusion term, the convection term of the momentum equation consists of 2 components, an x-component and a y-component, one for each momentum equation.

2.5.3 Pressure

$$-\nabla p \tag{2.14}$$

Notice how this is part of the equation (2.6). It expresses the movement of the fluid due to pressure changes. Fluids naturally tend to flow from high-pressure to low-pressure zones.

Once again, (2.14) in this form is not suitable for solving numerically. We will expand it in 2D, the same way we did for diffusion and convection, as follows.

$$-\nabla p = \begin{bmatrix} -\frac{\partial p}{\partial x} \\ -\frac{\partial p}{\partial y} \end{bmatrix} \tag{2.15}$$

Notice how, similarly to the diffusion and convection terms, the pressure term of the momentum equation consists of 2 components, an x-component and a y-component, one for each momentum equation.

2.5.4 Time

$$\frac{\partial \vec{v}}{\partial t} \tag{2.16}$$

Notice how this is part of the equation (2.6). It expresses the change in velocity with respect to time. It only takes effect when considering unsteady flows. In steady flows, this term is equal to 0.

As with all the other terms of the momentum equation (2.6), it hides 2 components.

$$\frac{\partial \vec{v}}{\partial t} = \begin{bmatrix} \frac{\partial u}{\partial t} \\ \frac{\partial v}{\partial t} \end{bmatrix} \tag{2.17}$$

The time term that appears in the momentum equations is also multiplied by density.

$$\rho\frac{\partial \vec{v}}{\partial t} = \begin{bmatrix} \rho\frac{\partial u}{\partial t} \\ \rho\frac{\partial v}{\partial t} \end{bmatrix} \tag{2.18}$$

2.6 Putting everything together

After manipulating both continuity and momentum equations, we are ready to combine them into a set of three equations: a continuity equation, an x-momentum equation, and a y-momentum equation.

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{2.19}$$

$$\rho\frac{\partial u}{\partial t} + \rho\left[\frac{\partial (uu)}{\partial x} + \frac{\partial (vu)}{\partial y}\right] = -\frac{\partial p}{\partial x} + \mu\frac{\partial}{\partial x}\left(\frac{\partial u}{\partial x}\right) + \mu\frac{\partial}{\partial y}\left(\frac{\partial u}{\partial y}\right) \tag{2.20}$$

$$\rho\frac{\partial v}{\partial t} + \rho\left[\frac{\partial (uv)}{\partial x} + \frac{\partial (vv)}{\partial y}\right] = -\frac{\partial p}{\partial y} + \mu\frac{\partial}{\partial x}\left(\frac{\partial v}{\partial x}\right) + \mu\frac{\partial}{\partial y}\left(\frac{\partial v}{\partial y}\right) \tag{2.21}$$

These 3 equations are all we need to continue our journey into simulating 2D flows of incompressible fluids.

The expanded form of the equations makes it clear that what we need to solve for is the x-component u of the velocity field, the y-component v of the velocity field, and the pressure field p. That is 3 equations with 3 unknowns at each point. It should be noted that u, v, and p are not individual values but scalar fields. That means they have a value assigned at each point (x, y) in space, so effectively, we are solving for 3 values in each point in space, as seen in figure 2.4.

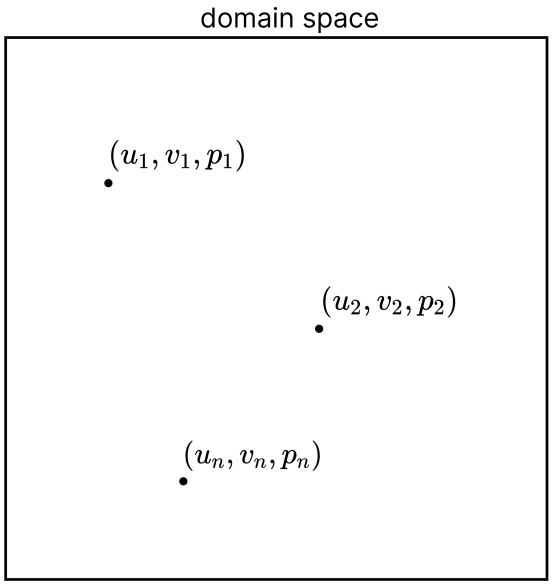


Figure 2.4: Unknown scalar fields u, v, p

Note that the domain mathematically has infinite points that cover it as a whole.

3. GRIDS

3.1 From the Equations to the Grid

A fair question is why we can't immediately start solving the equations we derived at 2.6. To understand why, it is essential to realize that computers are constrained by their nature: limited memory, processing power, etc. They can only handle a specific amount of work. On the other hand, mathematics doesn't have this flaw. The three equations we derived have three scalar fields as the unknowns we solve for, and each has an infinite number of points scattered throughout its domain. Therefore, it is impossible for a computer to find the exact solution to these three scalar fields, since this would imply that it can handle an infinite number of points.

As a result, to solve those equations, it is necessary to move from an analytical approach to a numerical one. This is where the field of Numerical Analysis [10] comes into play. There are a lot of methods that can be used to achieve this. Throughout this guide, we will use the Finite Volume Method or simply FVM [6] [24]. The essence of this method is to divide and fill the domain that we want to simulate into small cells. Effectively, we are creating a grid of cells (also called nodes). We can arrive at many grid types depending on the rules we set.

3.2 The Cartesian Grid

This is the most straightforward and intuitive type of grid we can create, in terms of ease of understanding and implementation. In this type of grid, considering a 2D scenario in which the domain is a rectangle, its cells appear as small squares. Figure 3.1 shows a 10×10 cell Cartesian grid.

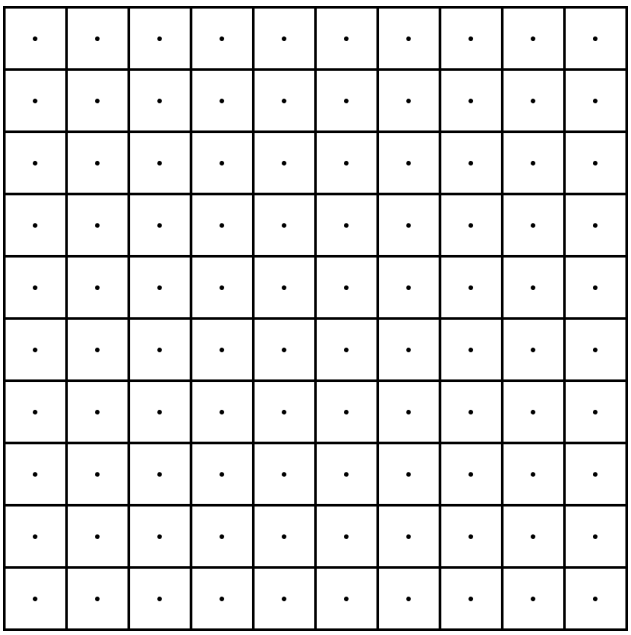


Figure 3.1: 2D 10×10 Cartesian Grid

In later chapters, for the sake of simplicity, we might start from a 1D grid and then move to a 2D grid. A 1D grid could look something like Figure 3.2.

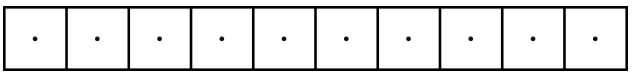


Figure 3.2: 1D 10-cell Cartesian Grid

Note that because this is in 1D (only x-axis in this case), the cell faces technically don't have a height, and this is how we should treat them when we start introducing some mathematics in the next chapter. Presenting them as cells, though, helps us visualize them better in the context of FVM.

3.3 Grid Arrangements

In both figures 3.1 and 3.2, you have probably noticed the dots in the center of the cells. They represent the **centroids** of each cell. This is where the value of each variable is stored. This is known as the **colocated** grid arrangement, since all the variables are stored in the cell centroids.

There are also other types of arrangements, such as the staggered grid arrangement, but it poses many disadvantages that make it impractical for real CFD problems. Most commercial CFD software packages use the colocated grid arrangement, which is the one we will be using for the scope of this guide.

3.4 Grid resolution

Let's take the pressure field, for example. In the context of the 2D case 3.1 we saw earlier, since this is a 10×10 cell grid, the total number of values of the pressure field is $10 \cdot 10 = 100$. This goes in contrast to the mathematical sense of the pressure field, which would contain infinite values, but this is a limitation that we have to accept. It is in our freedom, though, to increase the resolution of the grid to produce more accurate results, getting closer and closer to the "real" pressure field. However, this is going to increase computational cost. There is no such thing as the perfect grid resolution. After thoroughly considering the situation, the user should choose the right balance between accuracy and computational cost. Is there a time limit for the computation? How much accuracy is needed or would be acceptable? These are questions that should constantly cross a CFD engineer's mind.

3.5 Limitations of the Cartesian Grid

Before considering grid types other than Cartesian, one should understand their limitations. So far, we have modeled a rectangular domain. What if the domain we want to model, or even objects or walls inside the domain, are not rectangular or at an angle? In this case, it is obvious that a Cartesian grid cannot describe the domain properly and might not be the best option available.

For example, let's look at figure 3.3. Suppose we have an obstacle inside our domain (grid on the left). Trying to model it using Cartesian grid cells becomes really difficult due to the object's shape being ellipsoid and rotated (grid on the right), resulting in an unphysical interpretation. Again, this can be tackled by increasing the resolution of the grid, but it comes at a computational cost.

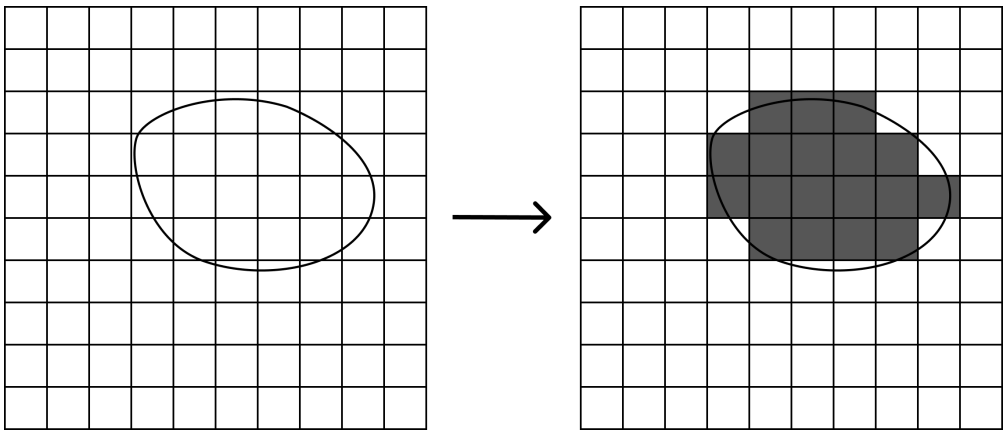
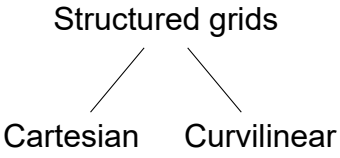


Figure 3.3: Modeling of a non-Cartesian object

This limitation encourages using different grid types and leads us directly to 3.6.

3.6 Other Grid Types

Cartesian grids belong to a broader category, the **Structured** grids [40]. Cartesian grids abide by the principle that the grid lines are always parallel to the coordinate axes. This is not necessarily the case with all structured grids. The structured grids that don't abide by this principle fall into the subcategory of Curvilinear grids. These simplify modeling complex geometries but are not as straightforward to implement.



Another common type of grids is **Unstructured** grids [40]. These give the user the maximum amount of freedom when it comes to modeling: cells can have a varying number of faces, and grid lines are not necessarily parallel. Most commercial CFD software packages use unstructured grids. However, these are by far the hardest to implement.

A less frequent category is **Block-structured** grids [40]. These types of grids allow subdividing the domain into subgrids and then modeling each one individually.

During this guide, we will exclusively be using the Cartesian grid type. It is by far the easiest to implement. It is excellent for understanding the underlying ideas, which can then be generalized into other types of grids, such as unstructured grids. Jumping into more complex grid types is not recommended before implementing the Cartesian grid.

3.7 Unknown Variables in CFD

As we saw in Figure 2.4, when solving the Incompressible Navier-Stokes, we have three unknown scalar fields: u, v, and p. Since, for this guide, we will be dealing with colocated grids, each scalar field will have a value in each cell centroid.

4. SOLVING SYSTEMS OF LINEAR EQUATIONS

4.1 Why is it important?

In the next chapter and throughout this guide, we will find out that to solve the Navier-Stokes equations numerically, we will have to solve various systems of linear equations [13]. Thus, learning how to do it right beforehand is crucial and will pay off in the long run.

4.2 Direct vs Iterative Methods

A vast variety of methods exist to solve linear equations. They are categorized into **direct** and **iterative**. Both categories pose advantages and disadvantages [51].

To capture the essence of each category, direct methods require a specific, predefined number of steps to solve the system. On the other hand, iterative methods do not need a certain number of steps to obtain a solution. Instead, a procedure is performed iteratively, and the iteration stops when a certain amount of accuracy is reached. When that happens, we say the method has **converged** [42].

For this guide, we will be using the **Gauss-Seidel** iterative method. The main reason is that it is relatively easy to implement and serves as the basis for more complex methods used in the industry, such as **multigrid** methods. Keep in mind, however, that most commercial CFD packages offer a range of both iterative and direct solvers, since the choice is problem dependent [33].

4.3 The Gauss-Seidel Method

4.3.1 Steps of the method

Suppose we have a system of n linear equations with n unknowns that we want to solve (4.1). Here, $x_1, \ldots, x_n$ are the variables that we want to solve for, while and $a_{11}, \ldots, a_{nn}$ and and $b_1, \ldots, b_n$ are the variable coefficients.

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n \end{cases} \tag{4.1}$$

The variables are unknown, while the variable coefficients are known beforehand. This means that throughout the solution process of Gauss-Seidel, $a_{11}, \ldots, a_{nn}$ and $b_1, \ldots, b_n$ will remain constant.

To solve the system (4.1) using the Gauss-Seidel method, we follow the steps:

Step 1. Manipulate the system of equations

We will manipulate the system (4.1) so that each equation is solved for an unknown. Since we have n equations for n unknowns, every equation solves for a different one. Typically, each equation is solved for its diagonal variable. (equation 1 for x_1, equation 2 for x_2 and so on).

$$(4.1) \implies \begin{cases} a_{11}x_1 = b_1 - (a_{12}x_2 + \ldots + a_{1n}x_n) \\ a_{22}x_2 = b_2 - (a_{21}x_1 + a_{23}x_3 + \ldots + a_{2n}x_n) \\ \vdots \\ a_{nn}x_n = b_n - (a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn-1}x_{n-1}) \end{cases} \tag{4.2}$$

Each equation in the system (4.2) can be written in the general form.

$$a_{ii}x_i = b_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} (a_{ij}x_j) \tag{4.3}$$

where, i is the equation number (from 1 to n). Then, we divide by the coefficient of the corresponding variable

$$(4.2) \implies \begin{cases} x_1 = \dfrac{b_1 - (a_{12}x_2 + \ldots + a_{1n}x_n)}{a_{11}} \\ x_2 = \dfrac{b_2 - (a_{21}x_1 + a_{23}x_3 + \ldots + a_{2n}x_n)}{a_{22}} \\ \vdots \\ x_n = \dfrac{b_n - (a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn-1}x_{n-1})}{a_{nn}} \end{cases} \tag{4.4}$$

Each equation of the system (4.4) can be written in the general form.

$$x_i = \frac{1}{a_{ii}} \left(b_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} (a_{ij}x_j) \right) \tag{4.5}$$

Step 2. Choose an initial guess for all the variables

The closer the initial guess is to the solution, the faster it will converge. A common initial guess is 0 for every variable, which is usually the default choice if we don't have more information regarding the problem we try to solve.

Step 3. Perform iterations until the solution converges

In each iteration, the new value of every variable is sequentially calculated using the corresponding equation (4.5) from the system. It should be noted that during an iteration, each equation uses the latest values of the variables, even if they

were calculated in the previous equation and not the variable values of the previous iteration. This is the difference between the Gauss-Seidel and the **Jacobi** Method [8].

The number of iterations needed for the solution to converge cannot be predicted. As discussed in 4.2, the solution is converged when a certain amount of accuracy is reached. How to asses convergence is explained on 4.3.2.

4.3.2 Convergence

A fair question to ask is what the criteria should be for convergence. There are a lot of options. A reliable one is to calculate the **absolute imbalance** of the system [20].

The absolute imbalance of an equation is equal to the difference between the left-hand side and the right-hand side of the equation (4.3) [25]

$$\left| a_{ii}x_i - b_i + \sum_{\substack{j=0 \\ j\neq i}}^{n} (a_{ij}x_j) \right| \tag{4.6}$$

where, i is the equation number (from 1 to n). After doing so for every equation, at the end of a single Gauss-Seidel iteration, we sum up those equation imbalances, and we arrive at the absolute imbalance of the system

$$\sum_{i=1}^{n} \left| a_{ii}x_i - b_i + \sum_{\substack{j=0 \\ j\neq i}}^{n} (a_{ij}x_j) \right| \tag{4.7}$$

Formula (4.7) is considered a great tool to assess convergence. We check for this value at the end of every Gauss-Seidel iteration. When the value of this quantity gets small enough (i.e., smaller than a **tolerance** we have defined), the solution has converged and the algorithm stops.

The absolute imbalance of the system is a type of **residual**. There are also other ways to calculate the residual of a system, but this will do for now.

4.3.3 Example

Suppose we want to solve the following system of linear equations with three equations and three unknowns.

$$\begin{cases} 2x_1 + x_2 + x_3 = 5 \\ 3x_1 + 5x_2 + 2x_3 = 15 \\ 2x_1 + x_2 + 4x_3 = 8 \end{cases} \tag{4.8}$$

Step 1: Manipulate the system of equations

Here, we obtain an equation for each variable

$$(4.8) \implies \begin{cases} 2x_1 = 5 - (x_2 + x_3) \\ 5x_2 = 15 - (3x_1 + 2x_3) \\ 4x_3 = 8 - (2x_1 + x_2) \end{cases} \tag{4.9}$$

$$\implies \begin{cases} x_1 = \dfrac{5 - (x_2 + x_3)}{2} \\ x_2 = \dfrac{15 - (3x_1 + 2x_3)}{5} \\ x_3 = \dfrac{8 - (2x_1 + x_2)}{4} \end{cases} \tag{4.10}$$

Step 2: Choose an initial guess for all the variables

Since we don't have any other info regarding this system, we will initialize all the variables with 0

$$x_1 = x_2 = x_3 = 0$$

Step 3: Perform iterations until the solution converges

Now, we start iterating until we reach an absolute system imbalance of 0.001. This is our tolerance. Any residual lower than that will be considered sufficient for this example. Note that this value was picked arbitrarily. If we needed more accuracy, we would lower this number, and if we were to be satisfied with lower accuracy, we would increase it.

Iteration 1

Calculate the new variable values. Notice that the new value of x_1 is used to calculate the new values of x_2 and x_3, and the new value of x_2 is used to calculate the new value of x_3. This is because Gauss-Seidel always uses the latest values of each variable.

$$x_1 = \frac{5 - (0 + 0)}{2} = 2.5$$

$$x_2 = \frac{15 - (3 \cdot 2.5 + 2 \cdot 0)}{5} = 1.5$$

$$x_3 = \frac{8 - (2 \cdot 2.5 + 1.5)}{4} = 0.375$$

Calculate the absolute system imbalance using formula (4.7)

$$\begin{aligned} Residual = & 2 \cdot 2.5 - 5 + 1.5 + 0.375 \\ & + 5 \cdot 1.5 - 15 + 3 \cdot 2.5 + 2 \cdot 0.375 \\ & + 4 \cdot 0.375 - 8 + 2 \cdot 2.5 + 1.5 \\ = & 2.625 \end{aligned}$$

Residual 2.625 is bigger than our tolerance of 0.001, so we proceed on the second iteration.

Iteration 2

$$x_1 = \frac{5 - (1.5 + 0.375)}{2} = 1.5625$$

$$x_2 = \frac{15 - (3 \cdot 1.5625 + 2 \cdot 0.375)}{5} = 1.9125$$

$$x_3 = \frac{8 - (2 \cdot 1.5625 + 1.9125)}{4} = 0.740625$$

$$\begin{aligned} Residual =& 2 \cdot 1.5625 - 5 + 1.9125 + 0.740625 \\ &+ 5 \cdot 1.9125 - 15 + 3 \cdot 1.5625 + 2 \cdot 0.740625 \\ &+ 4 \cdot 0.740625 - 8 + 2 \cdot 1.5625 + 1.9125 \\ =& 1.50938 \end{aligned}$$

Residual 1.50938 is bigger than our tolerance of 0.001, so we proceed on the third iteration. Notice, though, how it decreased in comparison to the first iteration.

Iteration ...

Iteration 8

$x_1 = 0.999572$

$x_2 = 2.00017$

$x_3 = 1.00017$

$Residual = 0.000579087$

This is the first iteration where the residual 0.000579087 is smaller than our tolerance of 0.001, so Gauss-Seidel has converged within eight iterations.

Since Gauss-Seidel converged, the latest values x_1, x_2, and x_3 constitute the solution to the system (4.8).

4.4 Scarborough Criteria

The Gauss-Seidel method poses a great drawback. Depending on the system, the solution might diverge instead of converging. This means that the residual will get higher as the iterations progress instead of getting lower, essentially ”blowing up”, as we say.

Luckily, there is a way to test whether or not Gauss-Seidel will converge or diverge by using the **Scarborough Criteria** [12]. Essentially, it resembles **diagonal dominance**, but also allows a few more cases. For the Scarborough Criteria to hold, the following must be true for the coefficients of the variables of the system of equations.

$$\frac{\sum_{\substack{j=0 \\ j \neq i}}^{n} |a_{ij}|}{|a_{ii}|} \leq 1 \text{ at all rows and } < 1 \text{ at one row at least} \tag{4.11}$$

where a_{ij} is the coefficient of the variable x_j at row i.

This means that for every row, the absolute value of the diagonal coefficient must be bigger than or equal to the sum of the absolute values of all the other row coefficients multiplied by their coefficients. **At least one** row must be **strictly bigger** though.

When the Scarborough Criteria hold, the Gauss-Seidel method is guaranteed to converge. This is going to be really important later on when we start solving the Navier-Stokes equations.

For example, take system (4.8).

In row 1, $\frac{|1|+|1|}{|2|} = 1 \leq 1$

In row 2, $\frac{|3|+|2|}{|5|} = 1 \leq 1$

In row 3, $\frac{|2|+|1|}{|4|} = 0.75 < 1$

Since the fraction is always less than or equal to 1 and is strictly less than 1 for at least one row, the Scarborough Criteria hold, and Gauss-Seidel is guaranteed to converge.

4.5 Implicit Relaxation

A concept that we will need later on in chapter 8 is **relaxation**. It is used to tackle cases where the Scarborough Criteria we discussed at 4.4 are not met or in cases where some approximations introduce instabilities. In those cases, solving the linear system of equations would result in divergence. There are two common types of relaxation, **explicit** and **implicit**. We will focus on implicit relaxation.

Implicit relaxation works by dividing all the diagonal coefficients by a constant number α, before even starting to solve the system. There are 3 cases, according to the value of α:

- **Under-relaxation** when $0 < \alpha < 1$: Effectively, dividing all the diagonal coefficients by a number higher than 0 and lower than 1 makes them bigger and thus results in stronger diagonal dominance. This stabilizes the system of linear equations, making it less prone to divergence, but has a negative effect on speed, often requiring more iterations to obtain a solution
- **No relaxation** when $\alpha = 1$: In this case, division by 1 does not affect the coefficients and is therefore equivalent to not applying any relaxation
- **Over-relaxation** when $\alpha > 1$: This is the exact opposite of under-relaxation. The solution is obtained faster, requiring fewer iterations. However, it often leads to divergence and less stability, so it should be used with care

For most CFD applications, applying under-relaxation to some systems is necessary to avoid divergence and will be our choice, as we will see later in chapter 8.

5. THE DIFFUSION EQUATION

5.1 Why Even Bother with the Diffusion Equation?

A fair question to ask is why even bother with solving this equation. Indeed, by itself, it will not help us to solve any fluid flow problem. However, diffusion is a part of the momentum equations of the Navier-Stokes equations, as seen at 2.5.1, so solving the momentum equation includes solving diffusion itself. Thus, the reader must be comfortable with it before moving on to later chapters. In addition, solving the rest of the Navier-Stokes equations follows a similar procedure. Note that while the momentum equations we will solve at the end are in 2D, we will first solve the diffusion equation in 1D and later solve it in 2D. This will make things a lot simpler: moving from 1D to 2D is as straightforward as moving from 2D to 3D, as long as the procedure is understood, enabling the reader to expand the solver to 3D in case it is desirable.

5.2 The 1D Steady state Diffusion Equation

The steady state 1D diffusion equation for a general variable ϕ and constant diffusion coefficient Γ can be written as

$$\Gamma \frac{\partial}{\partial x}\left(\frac{\partial \phi}{\partial x}\right) = 0 \tag{5.1}$$

where Γ is a constant value and ϕ is a scalar field. The left-hand side is equivalent to the term $\Gamma \nabla^2 \phi$, if we expand it in 1D. At this point, it is worth noting that the general variable ϕ in the diffusion equation could be any other quantity, such as velocity (in which case the diffusion coefficient would be viscosity μ), temperature (in which case the diffusion coefficient would be thermal diffusivity α), or any other quantity. Provided this is the only equation we solve, it doesn't really matter what we choose, as the result will be the same, and the only thing that changes is the interpretation. Also, notice how the equation becomes part of the momentum equations in the case of velocity u and viscosity μ.

A question might arise: why don't we factor out and remove Γ from the equation? Indeed, this can be done since Γ is constant; either way, it will not affect the results. We will keep it, though, as it will be necessary later on when more terms get added to the equation, in which case it will not be possible to factor it out.

5.2.1 Discretizing the equation using the central-differencing scheme

We have just arrived at one of the most critical steps when solving any equation in CFD: discretizing it. This is where we will turn the mathematical equation into an equation the computer can solve numerically. As discussed in 3.1, we will use the Finite Volume Method (FVM) for that. What we need to do is obtain an equation for every cell on the grid.

Let's imagine an arbitrary cell on the 1D grid (called P), as well as its neighbors, W (West) and E (East). The left cell face is denoted by w (west), while the east cell face is denoted by e (east). For the rest of this guide, we will abide by the convention that big letters

represent cell centroids, while small letters represent cell faces. All the grid points are separated by a distance of Δx. (Figure 5.1)

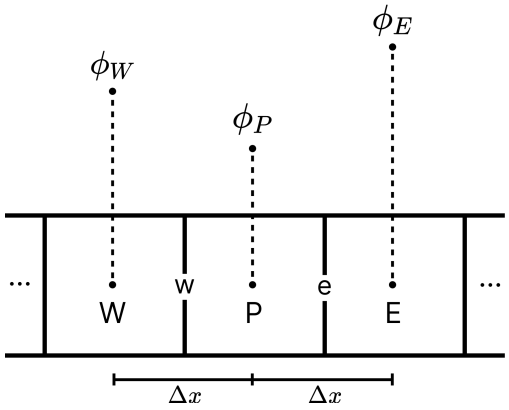


Figure 5.1: 3-cell 1D stencil

Notice that each grid point has a value of the variable ϕ assigned to it. We will integrate the equation for the cell P, only on the x-axis, since it is 1D, from the west face w to the east face e.

$$\int_{x=w}^{e} \Gamma \frac{\partial}{\partial x}\left(\frac{\partial \phi}{\partial x}\right) dx = 0 \tag{5.2}$$

$$\implies \Gamma\left(\frac{\partial \phi}{\partial x}\right)_e - \Gamma\left(\frac{\partial \phi}{\partial x}\right)_w = 0 \tag{5.3}$$

Here, a minor issue arises. How do we evaluate the first derivative of the variable ϕ on the cell faces? Our goal is to arrive at a linear equation with only values of ϕ as the unknowns. A **discretization scheme** should be introduced. We will assume that neighboring cell centroids vary linearly. This is called the **central-differencing scheme** or, equivalently, linear interpolation.

For example, let's take the cell face w. Since cell centroids W and P vary linearly, the slope (or derivative) in each point in between (including w) is constant, as all the lines have constant slope. To find the slope, we can use the formula $\dfrac{y_1 - y_0}{x_1 - x_0}$

(5.3) $$\implies \Gamma\left(\frac{\phi_E - \phi_P}{\Delta x}\right) - \Gamma\left(\frac{\phi_P - \phi_W}{\Delta x}\right) = 0 \tag{5.4}$$

$$\implies \frac{\Gamma}{\Delta x}\phi_E - \frac{\Gamma}{\Delta x}\phi_P - \frac{\Gamma}{\Delta x}\phi_P + \frac{\Gamma}{\Delta x}\phi_W = 0 \tag{5.5}$$

$$\implies \left(\frac{\Gamma}{\Delta x} + \frac{\Gamma}{\Delta x}\right)\phi_P = \left(\frac{\Gamma}{\Delta x}\right)\phi_W + \left(\frac{\Gamma}{\Delta x}\right)\phi_E \tag{5.6}$$

$$\implies a_P \phi_P = a_W \phi_W + a_E \phi_E \tag{5.7}$$

$$\implies a_P \phi_P = \sum_{nb} \left(a_{nb} \phi_{nb}\right) + S \tag{5.8}$$

where

a_W	$\frac{\Gamma}{\Delta x}$
a_E	$\frac{\Gamma}{\Delta x}$
a_P	$\frac{\Gamma}{\Delta x} + \frac{\Gamma}{\Delta x}$
S	0

Table 5.1: 1D steady state Diffusion coefficients

Notice how equation 5.6 is a linear equation. Its unknown variables are ϕ_W, ϕ_P and ϕ_E while the coefficients are a_W, a_P, a_E and S and are known beforehand. S is called the source term and is analogous to the b term we saw in chapter 4 when we learned how to solve systems of linear equations numerically. We introduced it (even if it is 0) for generality purposes, since later on there will arise cases where it will not be 0. Again, the coefficients and the source term remain constant throughout the Gauss-Seidel process.

If we apply the above procedure to every single grid cell on the domain, assuming the grid has N cells, we will obtain N linear equations with N unknowns in total (one unknown for every cell). This then becomes a linear system that the computer can solve numerically. In fact, we just learned how to do it in chapter 4. By solving every linear equation for its ϕ_P variable, we obtain one equation for every variable.

Equation 5.8 is really important. It is the general form that every equation reaches after discretization using the FVM. We will be aiming for this form for the rest of this guide. The subscript nb means neighbor and refers to the neighboring cells.

5.2.2 Boundary conditions

An important observation to make is that the equations that we derived by discretization during 5.2.1 only refer to cases where the cell of interest has both a west and an east cell neighbor (notice how variable ϕ of those neighbors are included in the equation 5.7). What happens when one of them is missing? We should be able to handle such cases since they occur at the domain boundaries. Figure 5.2 shows one such case, illustrating the left boundary of a 1D domain.

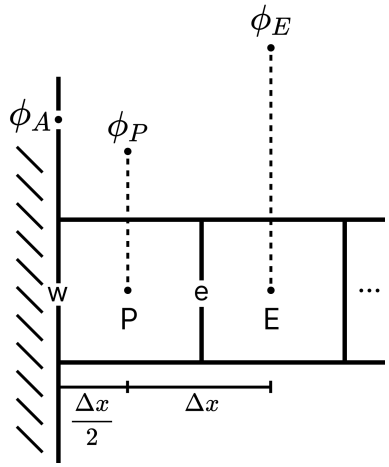


Figure 5.2: 1D left-boundary stencil

The question is, how to create a discretized equation for ϕ_P? There are many choices regarding how to handle this. Later, we will realize that it is problem-dependent and is a crucial part of CFD problem modeling. Almost all of those fall into two main categories: **Neumann** or **Dirichlet**. Essentially, for every unknown variable (ϕ in this case), we either define a fixed value (Dirichlet) or a gradient value (Neumann) on the cell face boundary. We will discuss various boundary conditions based on these principles in detail in chapter 10.

During this chapter, we will look at the simplest boundary condition of them all, the one in which we set a fixed value on the variable. In this boundary condition, ϕ at the boundary faces is set to a fixed value we choose (it is of Dirichlet type for the above reasons).

On all 1D cases, we will need two boundary conditions for ϕ, one for the left-most face and one for the right-most face at the boundaries. Let's see what happens on the west boundary in figure 5.2. Suppose that for the cell face, w, we set a fixed value boundary condition with a value of ϕ_A. Then, just like we did previously with an arbitrary cell, we integrate

$$\int_{x=w}^{e} \Gamma \frac{\partial}{\partial x}\left(\frac{\partial \phi}{\partial x}\right) dx = 0 \tag{5.9}$$

$$\implies \Gamma\left(\frac{\partial \phi}{\partial x}\right)_e - \Gamma\left(\frac{\partial \phi}{\partial x}\right)_w = 0 \tag{5.10}$$

The first issue pops up here. We cannot evaluate the derivative of ϕ at the cell face w the same way we did previously. That is because there is no cell towards the west. Still, we will make use of the slope formula $\frac{y_1 - y_0}{x_1 - x_0}$, but instead, we are going to use the values from the west boundary till the centroid P instead of the west cell centroid till the centroid P (because it doesn't exist). Again, it doesn't matter which point of the line we pick: the slope is the same (even at the end of the line segment). Notice how the distance is also halved. The evaluation of the derivative at the east face e remains unchanged.

$$(5.10) \implies \Gamma\left(\frac{\phi_E - \phi_P}{\Delta x}\right) - \Gamma\left(\frac{\phi_P - \phi_A}{\Delta x/2}\right) = 0 \tag{5.11}$$

$$\implies \frac{\Gamma}{\Delta x}\phi_E - \frac{\Gamma}{\Delta x}\phi_P - \frac{2\Gamma}{\Delta x}\phi_P + \frac{2\Gamma}{\Delta x}\phi_A = 0 \tag{5.12}$$

$$\implies \left(\frac{2\Gamma}{\Delta x} + \frac{\Gamma}{\Delta x}\right)\phi_P = \left(\frac{\Gamma}{\Delta x}\right)\phi_E + \frac{2\Gamma}{\Delta x}\phi_A \tag{5.13}$$

$$\implies a_P\phi_P = a_E\phi_E + S \tag{5.14}$$

$$\implies a_P\phi_P = \sum_{nb}\left(a_{nb}\phi_{nb}\right) + S \tag{5.15}$$

where

a_W	0
a_E	$\frac{\Gamma}{\Delta x}$
a_P	$\frac{2\Gamma}{\Delta x} + \frac{\Gamma}{\Delta x}$
S	$\frac{2\Gamma}{\Delta x}\phi_A$

Table 5.2: 1D steady state west boundary Diffusion coefficients

The term $\frac{2\Gamma}{\Delta x}\phi_A$ was added to the source since it is a known value.

The procedure is entirely symmetric when dealing with the eastern boundary of the grid. We set a fixed value boundary condition with a value of ϕ_B. Then, just like we did previously, we arrive at equation (5.15) with the coefficient values shown in table 5.3.

a_W	$\frac{\Gamma}{\Delta x}$
a_E	0
a_P	$\frac{\Gamma}{\Delta x} + \frac{2\Gamma}{\Delta x}$
S	$\frac{2\Gamma}{\Delta x}\phi_B$

Table 5.3: 1D steady state east boundary Diffusion coefficients

Our 1D grid has three types of equations in total: one for the left-most boundary cell, one for the right-most boundary cell, and one for the rest of the interior cells.

5.2.3 Example

Let's define a domain with a size of 1, with $N = 10$ cells and constant diffusion coefficient 0.01. It is not essential to set the metric units, as they could be anything, without affecting the nature of the solution, and are therefore left free for the user to choose. In this case, the total number of unknown variables will be N. We also need to choose some values for the boundaries, for instance, $\phi_A = 0$ and $\phi_B = 10$. The domain could look something like Figure 5.3.

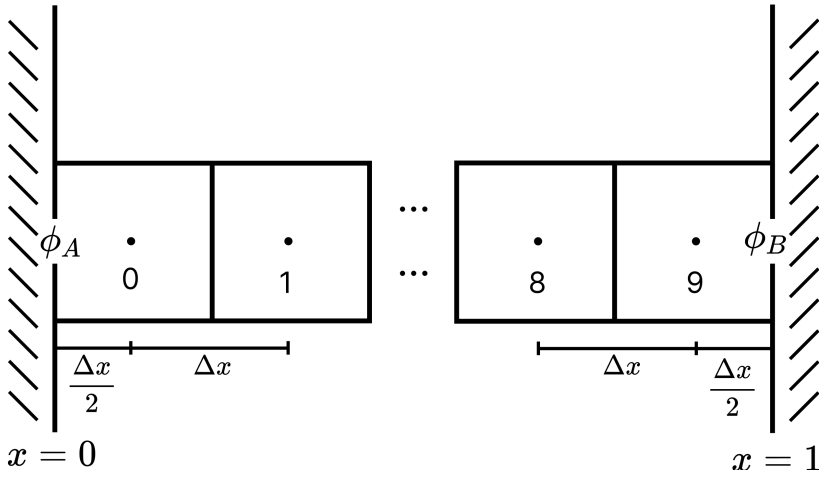


Figure 5.3: 1D domain stencil

The cells are indexed from 0 to 9. For practical reasons, cells $2 - 7$ are not displayed, but in reality, they do exist. Using the procedure we explained at 5.2.1 and 5.2.2, we obtain 10 linear equations in total, 1 for the left-most cell 0, 1 for the right-most cell 9, and 8 for the interior cells $1 - 8$.

We then follow the steps provided in the diagram 5.4 below. To solve the system of linear equations, we use the procedure explained in chapter 4.

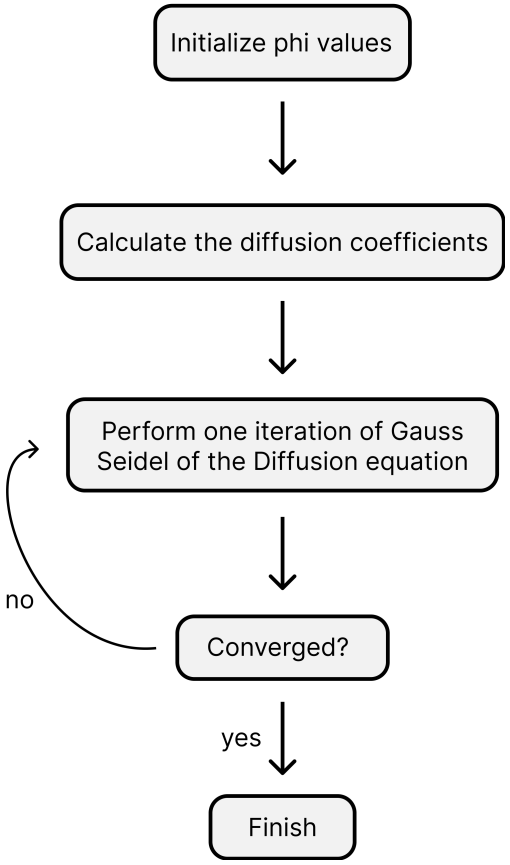


Figure 5.4: 1D steady state Diffusion steps

and obtain the solution of the system: 10 ϕ values, one for each cell, as demonstrated below in 5.5

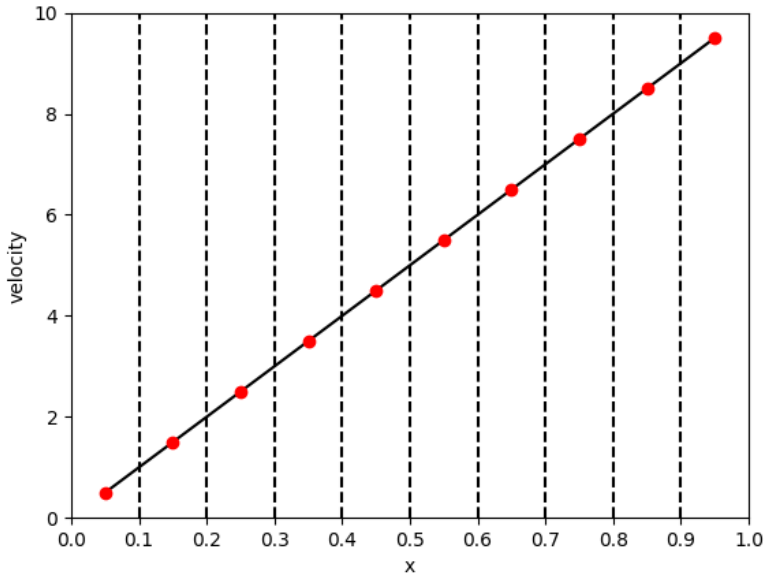


Figure 5.5: 1D Diffusion solution

The dotted lines represent the faces of the cells, and the red dots represent the solution values on the cell centroids. As we can see, the solution shows a linear profile, which

is as expected, should one solve the equation analytically by hand. Also, notice how the diffusion coefficient Γ doesn't affect the result, since it is constant and can be factored out of the equation. It would, however, in case we were dealing with an unsteady state case or if we added more terms into the equation, both of which we will cover later.

5.3 The 2D Steady state Equation

The 2D steady state diffusion equation [47] for a general variable ϕ and constant diffusion coefficient Γ can be written as

$$\Gamma \frac{\partial}{\partial x}\left(\frac{\partial \phi}{\partial x}\right) + \Gamma \frac{\partial}{\partial y}\left(\frac{\partial \phi}{\partial y}\right) = 0 \tag{5.16}$$

The left-hand side is equivalent to the term $\Gamma \nabla^2 \phi$, if we expand it in 2D. Again, if we use velocity u or v as the variable and viscosity μ as the diffusion coefficient, the equation appears as part of the momentum equations shown at 2.6. In fact, it is precisely what we will use later on when we solve the Navier-Stokes equations.

5.3.1 Discretizing the equation using the central-differencing scheme

As we did for the 1D case, we will obtain a linear equation for every cell on the grid. Let's imagine an arbitrary cell on the 2D grid (called P), as well as its neighbors, W (West), E (East), S (South), and N (North). The left cell face is denoted by w (west), the east cell face by e (east), the south cell face by s (south), and the north cell face by n (north). All the cell centroids are separated by a distance of Δx on the x-axis and by Δy on the y-axis. This is shown in Figure 5.6.

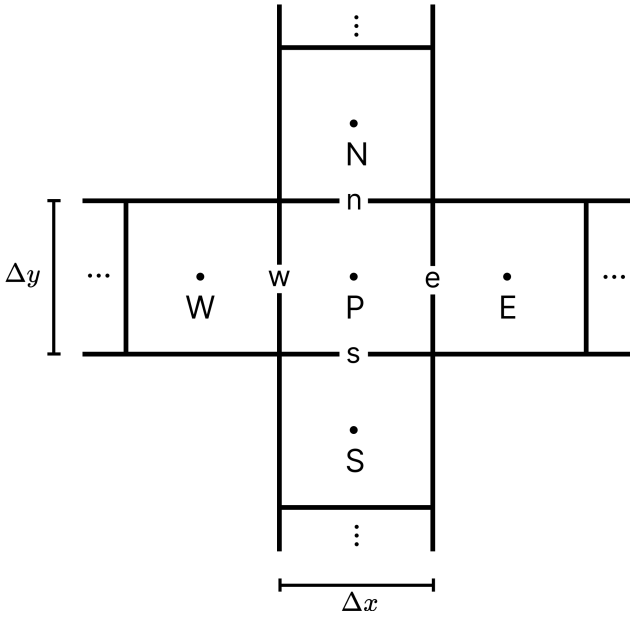


Figure 5.6: 5-cell 2D stencil

Since this is in 2D, we will integrate the equation for the cell P on the x-axis from the west face w to the east face e and on the y-axis from the south face s to the north face n.

$$\int_{y=s}^{n}\int_{x=w}^{e}\left[\Gamma\frac{\partial}{\partial x}\left(\frac{\partial\phi}{\partial x}\right)+\Gamma\frac{\partial}{\partial y}\left(\frac{\partial\phi}{\partial y}\right)\right]dxdy=0 \tag{5.17}$$

$$\implies \int_{y=s}^{n}\int_{x=w}^{e}\Gamma\frac{\partial}{\partial x}\left(\frac{\partial\phi}{\partial x}\right)dxdy+\int_{x=w}^{e}\int_{y=s}^{n}\Gamma\frac{\partial}{\partial y}\left(\frac{\partial\phi}{\partial y}\right)dydx=0 \tag{5.18}$$

$$\implies \int_{y=s}^{n}\left[\Gamma\left(\frac{\partial\phi}{\partial x}\right)_e-\Gamma\left(\frac{\partial\phi}{\partial x}\right)_w\right]dy+\int_{x=w}^{e}\left[\Gamma\left(\frac{\partial\phi}{\partial y}\right)_n-\Gamma\left(\frac{\partial\phi}{\partial y}\right)_s\right]dx=0 \tag{5.19}$$

In order to proceed, we should make a decision regarding the way the integrands vary throughout the cell face. We will assume that the integrands are constant throughout the cell faces.

$$(5.19)\implies\left[\Gamma\left(\frac{\partial\phi}{\partial x}\right)_e-\Gamma\left(\frac{\partial\phi}{\partial x}\right)_w\right]\Delta y+\left[\Gamma\left(\frac{\partial\phi}{\partial y}\right)_n-\Gamma\left(\frac{\partial\phi}{\partial y}\right)_s\right]\Delta x=0 \tag{5.20}$$

$$\implies\Gamma\Delta y\left(\frac{\partial\phi}{\partial x}\right)_e-\Gamma\Delta y\left(\frac{\partial\phi}{\partial x}\right)_w+\Gamma\Delta x\left(\frac{\partial\phi}{\partial y}\right)_n-\Gamma\Delta x\left(\frac{\partial\phi}{\partial y}\right)_s=0 \tag{5.21}$$

We will evaluate the derivative of ϕ with respect to x the same way we did for the 1D case. As for the derivative of ϕ with respect to y, we will follow a similar procedure. Instead of using neighboring nodes on the x-axis (west and east faces), we will use neighboring cells on the y-axis as follows.

$$(5.21)\implies\Gamma\Delta y\left(\frac{\phi_E-\phi_P}{\Delta x}\right)-\Gamma\Delta y\left(\frac{\phi_P-\phi_W}{\Delta x}\right)+\Gamma\Delta x\left(\frac{\phi_N-\phi_P}{\Delta y}\right)-\Gamma\Delta x\left(\frac{\phi_P-\phi_S}{\Delta y}\right)=0 \tag{5.22}$$

Doing some algebra and factorization results in the following

$$(5.22)\implies\left(\frac{\Gamma\Delta y}{\Delta x}+\frac{\Gamma\Delta y}{\Delta x}+\frac{\Gamma\Delta x}{\Delta y}+\frac{\Gamma\Delta x}{\Delta y}\right)\phi_P=\left(\frac{\Gamma\Delta y}{\Delta x}\right)\phi_W+\left(\frac{\Gamma\Delta y}{\Delta x}\right)\phi_E+\left(\frac{\Gamma\Delta x}{\Delta y}\right)\phi_S+\left(\frac{\Gamma\Delta x}{\Delta y}\right)\phi_N \tag{5.23}$$

$$\implies a_P\phi_P=a_W\phi_W+a_E\phi_E+a_S\phi_S+a_N\phi_N \tag{5.24}$$

$$\implies a_P\phi_P=\sum_{nb}\left(a_{nb}\phi_{nb}\right)+S \tag{5.25}$$

In equation (5.25), we arrived at the same general formula as we did for the 1D version, where

a_W	$\frac{\Gamma \Delta y}{\Delta x}$
a_E	$\frac{\Gamma \Delta y}{\Delta x}$
a_S	$\frac{\Gamma \Delta x}{\Delta y}$
a_N	$\frac{\Gamma \Delta x}{\Delta y}$
a_P	$\frac{\Gamma \Delta y}{\Delta x} + \frac{\Gamma \Delta y}{\Delta x} + \frac{\Gamma \Delta x}{\Delta y} + \frac{\Gamma \Delta x}{\Delta y}$
S	0

Table 5.4: 2D steady state Diffusion coefficients

However, these values only correspond to the linear equations for the interior, not boundary cells.

5.3.2 Boundary conditions

As in 1D, attention should also be given when dealing with boundary conditions in 2D. In 1D, we only had two boundary cells, regardless of the number of cells on the domain, so it made sense to deal with each one individually. 2D cases are more complex because the number of boundary cells depends on the grid size. For example, if we have a 20×10 grid, the number of boundary cells will be 56. Most of those boundary cells will only have one direction in which a boundary face exists. However, exactly 4 of them (all the corner cells) will have boundaries in two directions. It is clear that trying to figure out what happens on each cell is impractical, and another strategy towards handling boundary cases should be adopted. On that note, handling each boundary cell individually becomes a nightmare in 3D.

The key to understanding how to handle any boundary case is to generalize the process for each cell, whether it has boundary faces or not. Thus, it is crucial to realize what happens when a boundary face exists in a specific direction (west, east, south, or north) and how it affects the equation coefficients. This way, we can handle an arbitrary number of boundary faces on each cell.

We will start with equation (5.21). Each of the four terms can be interpreted as the diffusive flux of a face in a cell (west, east, south, north). Note that this is the same equation whether we have any boundary faces or not, since boundaries only affect the equation during the next step, in which we evaluate the derivative of ϕ on the cell faces.

Suppose that for an arbitrary cell, we have only one boundary face, for example, on the west. We will compare this to the case with no boundary faces (as seen in 5.3.1). Once again, we will use the central-differencing scheme. The ϕ derivative on the west face will be handled in the same way as in 5.2.2.

$$(5.21) \implies \Gamma\Delta y\left(\frac{\phi_E-\phi_P}{\Delta x}\right)-\Gamma\Delta y\left(\frac{\phi_P-\phi_A}{\Delta x/2}\right)+\Gamma\Delta x\left(\frac{\phi_N-\phi_P}{\Delta y}\right)-\Gamma\Delta x\left(\frac{\phi_P-\phi_S}{\Delta y}\right)=0 \tag{5.26}$$

During this discretization, we notice a few things. Comparing equations (5.22) and (5.26), the only diffusive flux that had a change was the one on the west (the one we assumed to be a boundary). All the other ones remained the same. Thus, all the coefficient changes (compared to the case with no boundaries) emerged from the west face flux. In other words, introducing a boundary condition on an individual cell face only affects the diffusive flux of that face. As we will see, this is a critical observation to make.

Let's extract the differences by isolating the west diffusive flux in equation (5.21).

$$\Gamma\Delta y\left(\frac{\partial\phi}{\partial x}\right)_w=0 \tag{5.27}$$

If the west face is not a boundary

$$(5.27) \implies \Gamma\Delta y\left(\frac{\phi_P-\phi_W}{\Delta x}\right)=0 \implies \left(\frac{\Gamma\Delta y}{\Delta x}\right)\phi_P=\left(\frac{\Gamma\Delta y}{\Delta x}\right)\phi_W \tag{5.28}$$

Therefore, the west diffusive flux will have the following effects on the diffusion equation coefficients:

- Add $\frac{\Gamma\Delta y}{\Delta x}$ to the center coefficient
- Add $\frac{\Gamma\Delta y}{\Delta x}$ to the west coefficient

If the west face is a boundary

$$(5.27) \implies \Gamma\Delta y\left(\frac{\phi_P-\phi_A}{\Delta x/2}\right)=0 \implies \left(\frac{2\Gamma\Delta y}{\Delta x}\right)\phi_P=\frac{2\Gamma\Delta y\phi_A}{\Delta x} \tag{5.29}$$

Therefore, the west diffusive flux will have the following effects on the diffusion equation coefficients:

- Add $\frac{2\Gamma\Delta y}{\Delta x}$ to the center coefficient
- Add $\frac{2\Gamma\Delta y\phi_A}{\Delta x}$ to source S.

This was the procedure for the west face of a cell. We can do the same thing for all the faces of a cell and arrive at the following general steps for an arbitrary cell face f.

Let $A = \frac{\Delta y}{\Delta x}$ if f is a west/east face or $A = \frac{\Delta x}{\Delta y}$ if f is a south/north face.

If f is not a boundary face

- Add ΓA to the center coefficient
- Add ΓA to the coefficient in the direction of f

If f is a boundary face

- Add $2\Gamma A$ to the center coefficient
- Add $2\Gamma A \phi_f$ to source S

Figure 5.7: 2D Diffusive flux effects (Central differencing)

All we have to do for every cell to create the discretized equation and obtain the equation coefficients is to follow the steps found in 5.7 for each one of the four faces of the cell. This procedure is more straightforward than determining exactly what happens on each individual cell and face. Additionally, it gives a lot of freedom regarding the number of boundary faces of each cell: it doesn't matter anymore if one individual cell has no boundary faces, only has one, or has four. Every face in every cell will be handled the same way.

5.3.3 Example

Let's define a domain with a size of 1×1, with $N \times M = 30 \times 30$ cells and constant diffusion coefficient 0.01. The unknown variables will be $N \cdot M = 900$ in this case. We also need to choose some values for the boundaries. We have the freedom to do so for every single boundary cell face on the domain. For this example, since the domain has four sides, we will set a constant value for each side. This means setting the value of ϕ for every cell boundary face on a specific domain side to a fixed value. The domain could look something like in Figure 5.8.

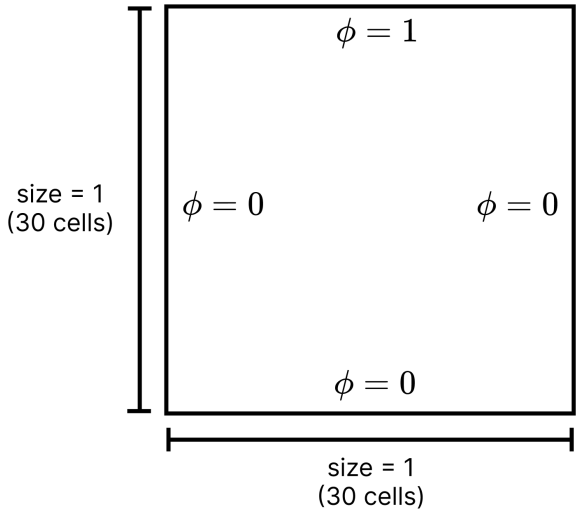


Figure 5.8: 2D domain

Using the procedure we explained at 5.3.1 and 5.3.2, we obtain 900 linear equations in total, one for every cell, with 900 unknowns.

We then follow the steps found in diagram 5.4 and obtain the solution of the system: 900 ϕ values, one for each cell, as demonstrated below in Figure 5.9

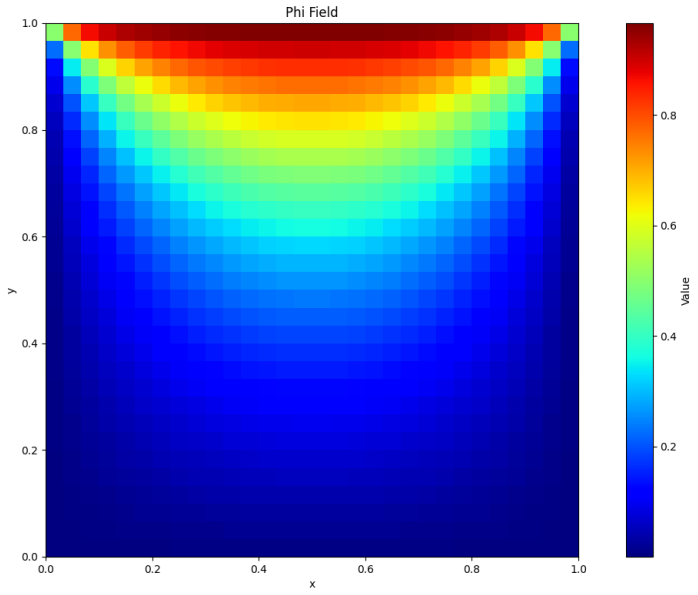


Figure 5.9: 2D Diffusion solution

Variable ϕ values are displayed using a colormap, meaning each color represents a value. This is a very common way to plot 2D scalar fields. Also, notice how the plot consists of small squares. Each one of them represents a cell on the grid.

To make sense of this result, think of the general variable ϕ as temperature T and the domain as a square plate. The top side of the plate is fixed to a high temperature, while the other sides are fixed to a low temperature. In this case, the results show what temperature the plate would obtain at each point of its domain. Since this is a steady state simulation,

it shows the state of the plate after an arbitrary amount of time, after which the marching of time doesn't affect the solution anymore; hence the steady state.

5.4 The 2D Unsteady state Diffusion Equation

The 2D unsteady state diffusion equation [47] for a general variable ϕ and constant diffusion coefficient Γ can be written as

$$\frac{\partial \phi}{\partial t} = \Gamma \frac{\partial}{\partial x}\left(\frac{\partial \phi}{\partial x}\right) + \Gamma \frac{\partial}{\partial y}\left(\frac{\partial \phi}{\partial y}\right) \tag{5.30}$$

This is very similar to the 2D steady state diffusion equation defined at 5.3. The only difference is that an extra term has been added to the equation that accounts for the changes to the scalar field ϕ due to time. More regarding the difference between steady and unsteady state simulations was discussed earlier at 2.1.5.

5.4.1 Discretizing the time term of the equation

By isolating the left-hand side of the equation, we can discretize it and focus on the coefficients that it produces.

$$\frac{\partial \phi}{\partial t} = 0 \tag{5.31}$$

We will integrate on both spatial dimensions (x and y) the same way we did for 2D steady state diffusion. However, since we are also considering the effects of time, we will integrate from an arbitrary time t to a time $t + \Delta t$ into the future. Δt is the difference in time between each timestep. Therefore, timestep t is known to us beforehand.

$$\int_{y=s}^{n} \int_{x=w}^{e} \int_{t=t}^{t+\Delta t} \frac{\partial \phi}{\partial t} dt dx dy = 0 \tag{5.32}$$

To evaluate the time integral, we will abide by the convention that old values during t are denoted by a $^\circ$ superscript, while new values during $t + \Delta t$ are not denoted with any superscript.

$$(5.32) \implies \int_{y=s}^{n} \int_{x=w}^{e} (\phi_P - \phi_P{}^\circ) dx dy = 0 \tag{5.33}$$

$$\implies (\phi_P - \phi_P{}^\circ)\Delta x \Delta y = 0 \tag{5.34}$$

$$\implies \Big(\Delta x \Delta y\Big)\phi_P = \phi_P{}^\circ \Delta x \Delta y \tag{5.35}$$

$$\implies a_P \phi_P = S \tag{5.36}$$

$$\implies a_P \phi_P = \sum_{nb} \left(a_{nb} \phi_{nb} \right) + S \tag{5.37}$$

where

a_W	0
a_E	0
a_S	0
a_N	0
a_P	$\Delta x \Delta y$
S	${\phi_P}^{\circ} \Delta x \Delta y$

Table 5.5: 2D time derivative coefficients

5.4.2 Discretizing the diffusion part of the equation

By isolating the right-hand side of the equation (5.30), we can discretize it and focus on the coefficients that it produces.

$$\Gamma \frac{\partial}{\partial x} \left(\frac{\partial \phi}{\partial x} \right) + \Gamma \frac{\partial}{\partial y} \left(\frac{\partial \phi}{\partial y} \right) = 0 \tag{5.38}$$

This should look very familiar. It is the 2D steady state diffusion equation we discretized on 5.3. The discretization process will be very similar. The only difference is that we will also integrate with respect to time.

$$\int_{t=t}^{t+\Delta t} \int_{y=s}^{n} \int_{x=w}^{e} \left[\Gamma \frac{\partial}{\partial x} \left(\frac{\partial \phi}{\partial x} \right) + \Gamma \frac{\partial}{\partial y} \left(\frac{\partial \phi}{\partial y} \right) \right] dx dy dt = 0 \tag{5.39}$$

$$\implies \int_{t=t}^{t+\Delta t} \int_{y=s}^{n} \int_{x=w}^{e} \Gamma \frac{\partial}{\partial x} \left(\frac{\partial \phi}{\partial x} \right) dx dy dt + \int_{t=t}^{t+\Delta t} \int_{x=w}^{e} \int_{y=s}^{n} \Gamma \frac{\partial}{\partial y} \left(\frac{\partial \phi}{\partial y} \right) dy dx dt = 0 \tag{5.40}$$

The same procedure explained in 5.3.1 is followed to evaluate the first two spatial integrals.

$$(5.40) \implies \int_{t=t}^{t+\Delta t} \Delta y \left[\Gamma \left(\frac{\partial \phi}{\partial x} \right)_e - \Gamma \left(\frac{\partial \phi}{\partial x} \right)_w \right] dt + \int_{t=t}^{t+\Delta t} \Delta x \left[\Gamma \left(\frac{\partial \phi}{\partial y} \right)_e - \Gamma \left(\frac{\partial \phi}{\partial y} \right)_w \right] dt = 0 \tag{5.41}$$

To solve the time integral, we should decide how the integrands vary through time, from t to $t + \Delta t$. A time scheme should be introduced. There are a lot of options. The most straightforward and robust (though not the most accurate) is the **implicit scheme**. It assumes that the value of the integrands originates only from the future timestep $t + \Delta t$. The opposite is the explicit scheme, which assumes that the integrands originate only from the timestep t. Other time schemes use both timesteps [37].

Throughout this guide, we will use the implicit scheme. No superscript is needed since it only takes values from timestep $t + \Delta t$.

$$(5.41) \implies \left[\Gamma \left(\frac{\partial \phi}{\partial x} \right)_e - \Gamma \left(\frac{\partial \phi}{\partial x} \right)_w \right] \Delta y \Delta t + \left[\Gamma \left(\frac{\partial \phi}{\partial y} \right)_e - \Gamma \left(\frac{\partial \phi}{\partial y} \right)_w \right] \Delta x \Delta t = 0 \tag{5.42}$$

$$\implies \Gamma \Delta y \Delta t \left(\frac{\partial \phi}{\partial x} \right)_e - \Gamma \Delta y \Delta t \left(\frac{\partial \phi}{\partial x} \right)_w + \Gamma \Delta x \Delta t \left(\frac{\partial \phi}{\partial y} \right)_e - \Gamma \Delta x \Delta t \left(\frac{\partial \phi}{\partial y} \right)_w = 0 \tag{5.43}$$

This is the flux equation for the 2D unsteady-state diffusion part. Notice that it is very similar to equation (5.21), which is the flux equation for the 2D steady state diffusion part. The only difference between the two is the multiplication of every flux by Δt. This is a very convenient and valuable outcome of the implicit time scheme: Adding a time integral only results in the multiplication of the fluxes by Δt.

This means that to construct the 2D diffusion part coefficients, we can follow a very similar procedure to figure 5.7 for every face on the cell. Only this time, we multiply A by Δt. Therefore, $A = \dfrac{\Delta y \Delta t}{\Delta x}$ if f is a west/east face or $A = \dfrac{\Delta x \Delta t}{\Delta y}$ if f is a south/north face.

5.4.3 Discretizing the Diffusion equation

We have discretized both parts of the 2D unsteady state diffusion equation. We obtain the final coefficients by combining the coefficients from both parts of the equation through addition. This includes the time coefficients in table 5.5 and the diffusion coefficients that can be obtained by following the steps in figure 5.7 for every cell face (with the inclusion of Δt in A).

5.4.4 Example

We will use the domain presented in the example for the 2D steady state diffusion equation in 5.3.3. This time, we will simulate the unsteady state version of it. We will calculate 1000 timesteps, with $\Delta t = 0.01$. Thus, the total simulation time will be $1000 \cdot 0.01 = 10$.

We follow the steps found in diagram 5.10

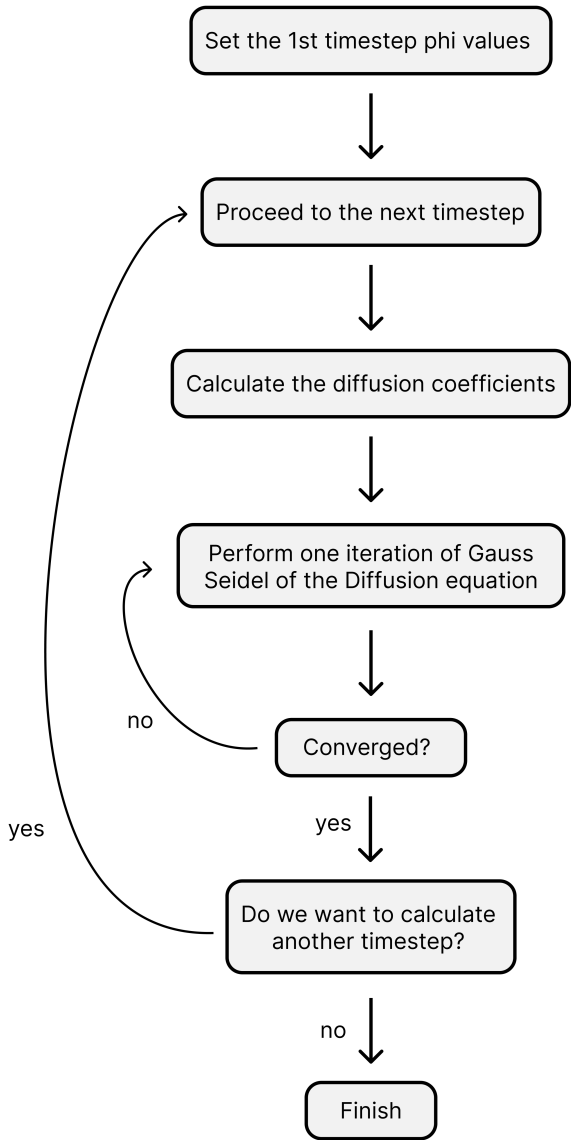
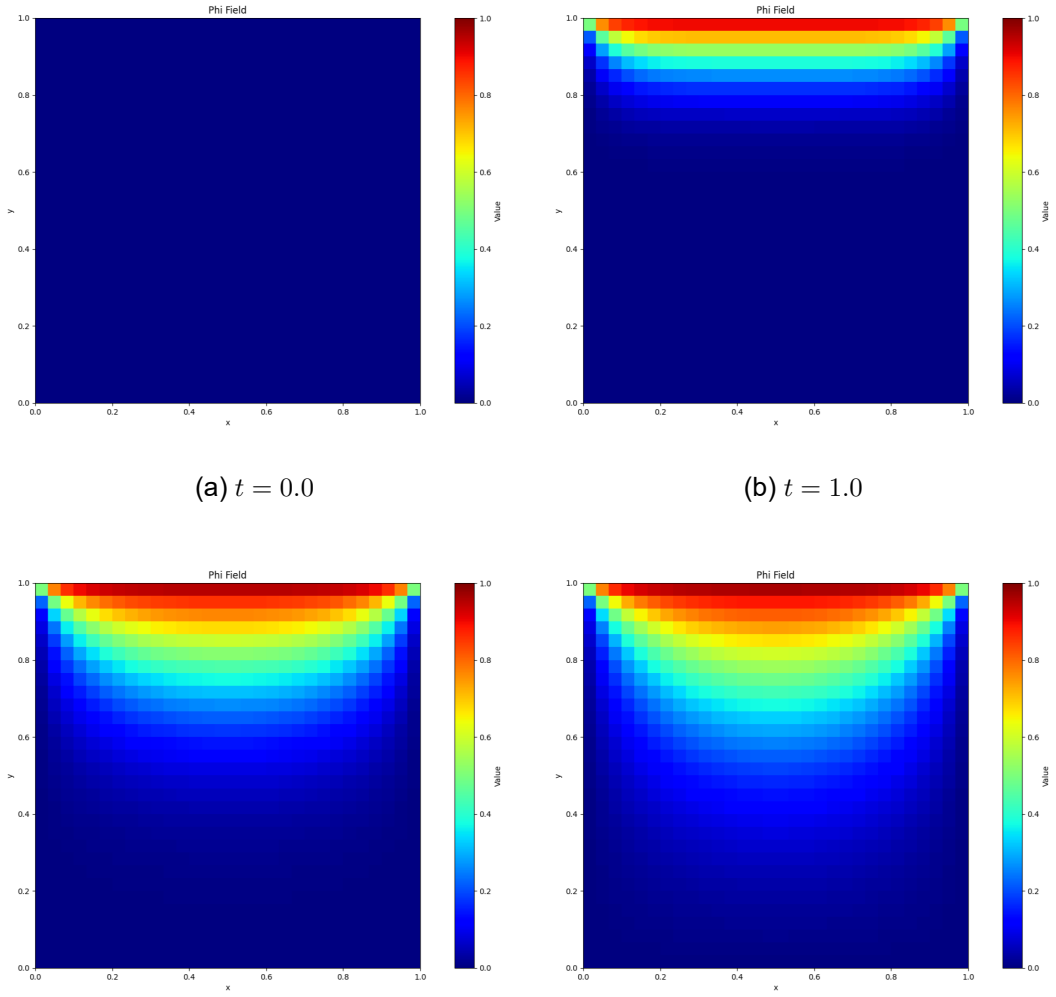


Figure 5.10: 2D unsteady state Diffusion steps

and obtain the solution of the system: 1000 timesteps with 900 ϕ values each. It is quite obvious that compared to the steady state case, the unsteady state case takes considerably more time for the calculation and space for the ϕ values. Below in Figure 5.11, we pick four timesteps out of all of them at different times, to showcase the effect of time marching on the solution. On that note, these 1000 timesteps could alternatively be used to create a video animation as well, which is another form of showcasing unsteady state simulations.



(a) $t = 0.0$ (b) $t = 1.0$

(c) $t = 5.0$ (d) $t = 10.0$

Figure 5.11: Time marching of 2D unsteady state Diffusion

It can be seen that time has a greater effect at the start of the simulation and decreases as time marches, effectively reaching the steady state (if it exists) after some time. In our case, at $t = 10.0$, the solution appears to be very close to the results we obtained when solving the 2D steady state Diffusion case at 5.3.3. In fact, it will keep getting closer as time marches.

6. THE CONVECTION-DIFFUSION EQUATION

6.1 From Diffusion to the Convection-Diffusion Equation

The convection-diffusion equation is the next logical step in solving the Navier-Stokes equations. In comparison to the diffusion equation, it includes one extra term: convection. Therefore, before moving on, it is crucial that the reader is already comfortable with solving the diffusion equation since it is a part of the convection-diffusion equation, let alone the Navier-Stokes equations.

6.2 The 2D Steady State Convection-Diffusion Equation

The 2D steady-state convection-diffusion equation for a general variable ϕ and constant diffusion coefficient Γ can be written as

$$\frac{\partial(u\phi)}{\partial x} + \frac{\partial(v\phi)}{\partial y} = \Gamma\frac{\partial}{\partial x}\left(\frac{\partial\phi}{\partial x}\right) + \Gamma\frac{\partial}{\partial y}\left(\frac{\partial\phi}{\partial y}\right) \tag{6.1}$$

Again, if we use velocity u or v as the variable and viscosity μ as the diffusion coefficient, the equation appears as part of the momentum equations shown at 2.6.

Note that when we use ϕ as the variable, velocities u and v are considered constant scalar fields. This means they are defined beforehand and are not changed throughout the simulation. They can be thought of as a fixed vector field by which ϕ gets convected.

6.2.1 Discretizing the convection part of the equation using the upwind scheme

Considering the equation (6.1), bearing in mind the diffusion equation (5.16), we can express it like $Convection = Diffusion$. Thus, it is important to observe that it consists of 2 parts, one of which (diffusion) we already know how to discretize. Therefore, it makes sense to pay attention to the left-hand side of the equation, which expresses convection, extract the coefficients through the discretization process, and then merge them with the diffusion coefficients we obtained in the previous chapter through addition to come up with the coefficients for the convection-diffusion equation. This will also make the discretization process easier as the equations would otherwise become too long and prone to error.

Therefore, we are going to discretize this part of the equation, effectively eliminating diffusion.

$$\frac{\partial(u\phi)}{\partial x} + \frac{\partial(v\phi)}{\partial y} = 0 \tag{6.2}$$

We will approach this the same way we have done so far. Since this is a steady state, we will only integrate over the two spatial dimensions.

$$\int_{y=s}^{n}\int_{x=w}^{e}\left[\frac{\partial(u\phi)}{\partial x}+\frac{\partial(v\phi)}{\partial y}\right]dxdy=0 \tag{6.3}$$

$$\implies \int_{y=s}^{n}\int_{x=w}^{e}\frac{\partial(u\phi)}{\partial x}dxdy+\int_{x=w}^{e}\int_{y=s}^{n}\frac{\partial(v\phi)}{\partial y}dydx=0 \tag{6.4}$$

$$\implies \int_{y=s}^{n}(u_e\phi_e-u_w\phi_w)dy+\int_{x=w}^{e}(v_n\phi_n-v_s\phi_s)dx=0 \tag{6.5}$$

$$\implies u_e\phi_e\Delta y-u_w\phi_w\Delta y+v_n\phi_n\Delta x-v_s\phi_s\Delta x=0 \tag{6.6}$$

Equation (6.6) is the convection flux equation. It is very similar to the diffusion flux equation (5.21), in the sense that it is the sum of the fluxes in the faces of a cell. As we did for diffusion, we will isolate one face flux to see how it behaves. For example, the flux on face w.

$$u_w\phi_w\Delta y=0 \tag{6.7}$$

If the west face is not a boundary

We must find a way to evaluate the values of u, v, and ϕ at the cell faces.

u and v face values will be calculated using linear interpolation. Since node velocity values are known, face values are also known. For example, $u_w=\frac{u_W+u_P}{2}$, since the cell face w is in between points W and P, while keeping the same distance with both.

For ϕ face values, a discretization scheme should be used. An option would be to use central differencing, as we did for ϕ values at the diffusion term. However, it turns out that it is not considered a good option for the convection term since it can lead to instabilities and divergence [19]. A more robust scheme for convection is the **upwind scheme** [15], which is unconditionally stable. It should be noted that we indeed have the freedom to choose different discretization schemes for different terms of the equations we discretize. For example, in this case, we use central differencing for the diffusion term and upwind for the convection term.

Suppose we have the following case in figure 6.1 and we want to evaluate ϕ at the cell face f using the upwind scheme.

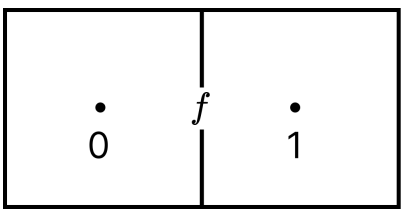


Figure 6.1: 2 - cell grid

The upwind scheme defines ϕ_f as follows

$$\phi_f = \begin{cases} \phi_0, & u_f > 0 \\ \phi_1, & u_f < 0 \end{cases} \tag{6.8}$$

Mathematically, we can use a small trick that allows us to write it inline, which becomes convenient in the case where we only want one final generalized equation and not different cases depending on face velocity values

$$u_f \phi_f = max(u_f, 0)\phi_0 - max(-u_f, 0)\phi_1 \tag{6.9}$$

Take a moment to realize that both sides of the equation (6.9) are equivalent for different values of u_f.

This trick allows us to continue our discretization of the west face flux

$$(6.7) \implies \Big[max(u_w, 0)\phi_W - max(-u_w, 0)\phi_P\Big]\Delta y = 0 \tag{6.10}$$

$$\implies \Big[max(-u_w, 0)\Delta y\Big]\phi_P = \Big[max(u_w, 0)\Delta y\Big]\phi_W \tag{6.11}$$

Therefore, the west convective flux will have the following effects on the convection equation coefficients:

- Add $max(-u_w, 0)\Delta y$ to the center coefficient
- Add $max(u_w, 0)\Delta y$ to the west coefficient

If the west face is a boundary

The values of u and v at the cell face boundary will be extrapolated from the nearest node P, for example, $u_w = u_P$. As for ϕ values at the cell face boundary, we have a fixed value boundary condition with $\phi_w = \phi_A$, so all values are known.

Therefore, the west convective flux will have the following effects on the Convection equation coefficients:

- Add $u_w \phi_A \Delta y$ to source S

This was the procedure for the west face of a cell. We can do the same thing for all of the faces of a cell and arrive at the following general steps for an arbitrary cell face f.

Define the values A, B, V

- $A = \Delta y$ if f is a west/east face or $A = \Delta x$ if f is a south/north face
- $B = 1$ if f is a west/south face or $B = -1$ if f is an east/north face
- $V = u_f$ if f is a west/east face or $V = v_f$ if f is a south/north face

If f is not a boundary face

- Add $A \cdot max(-BV, 0)$ to the center coefficient
- Add $A \cdot max(BV, 0)$ to the coefficient in the direction of f

If f is a boundary face

- Add $BAV\phi_f$ to source S

Figure 6.2: 2D Convective flux effects (Upwind)

All we have to do for every cell to create the discretized equation and obtain the equation coefficients is to follow the steps found in 6.2 for each one of the four faces of the cell.

6.2.2 Discretizing the Convection-Diffusion equation

We have discretized both parts of the 2D steady-state convection-diffusion equation. By combining the coefficients from both parts of the equation, through addition, we obtain the final coefficients. This includes the diffusion and the convection part coefficients that can be obtained by following the steps from figures 5.7 and 6.2, respectively, for every cell face.

6.2.3 Example

We will use the domain presented in the example for the 2D steady state diffusion equation in 5.3.3. This time, we will simulate the steady-state convection-diffusion equation. What we need to add is a known velocity vector field. That consists of 2 constant scalar fields: velocity u and v. We will choose the uniform fields $u(x, y) = 0.1$ and $v(x, y) = -0.1$. This velocity vector field can be visualized as having all its vectors pointing southeast.

We follow the steps found in diagram 6.3

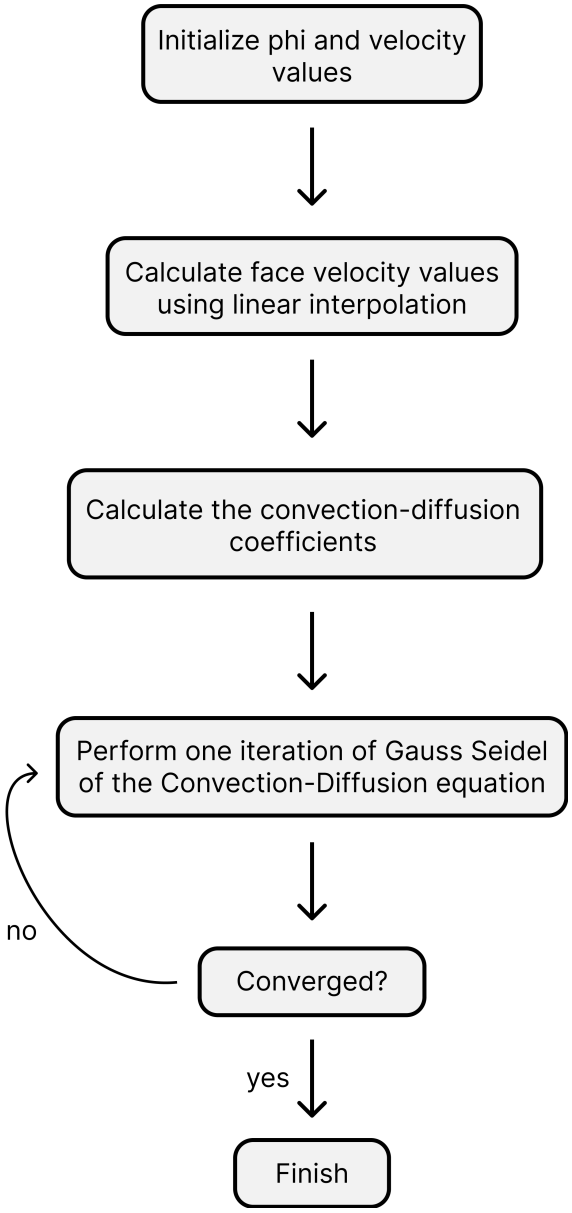


Figure 6.3: 2D steady state Convection-Diffusion steps

and obtain the solution of the system, as demonstrated below in Figure 6.4

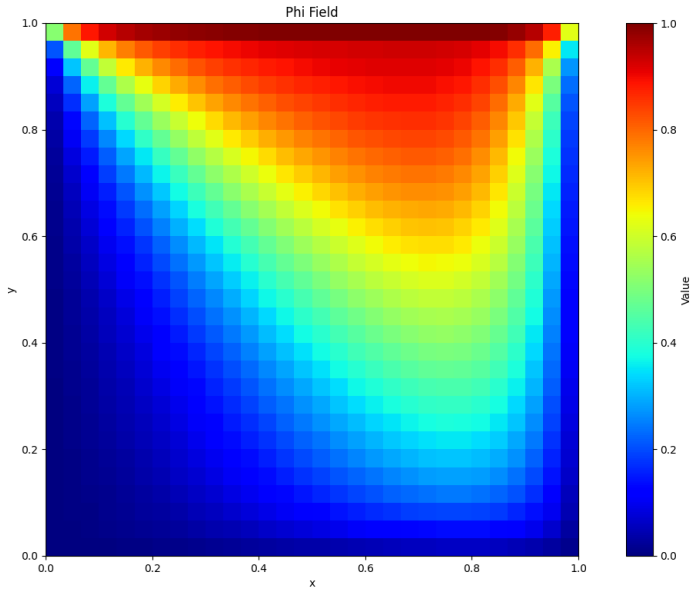


Figure 6.4: 2D Convection-Diffusion solution

Notice how, compared to the 2D Diffusion case found in 5.9, the velocity field has an immediate effect on the ϕ field, slanting it towards the southeast direction of the velocity field as expected.

6.3 The 2D Unsteady State Convection-Diffusion Equation

The 2D unsteady state convection-diffusion equation for a general variable ϕ and constant diffusion coefficient Γ can be written as

$$\frac{\partial \phi}{\partial t} + \frac{\partial (u\phi)}{\partial x} + \frac{\partial (v\phi)}{\partial y} = \Gamma \frac{\partial}{\partial x}\left(\frac{\partial \phi}{\partial x}\right) + \Gamma \frac{\partial}{\partial y}\left(\frac{\partial \phi}{\partial y}\right) \tag{6.12}$$

It looks exactly the same as the 2D steady-state convection-diffusion (6.1), with the addition of the time derivative term.

6.3.1 Discretizing the equation

During previous chapters, we have discretized all three parts of the equation 6.12. By combining the coefficients from all parts of the equation through addition, we obtain the final coefficients. This includes the time coefficients in table 5.5 and the diffusion/convection coefficients that can be obtained by following the steps in 5.7 and 6.2, respectively, for every cell face (with the inclusion of Δt in A).

6.3.2 Example

We will use the domain presented in the example for the 2D steady state diffusion equation in 5.3.3. We will simulate 1000 timesteps with $\Delta t = 0.01$. This time, we will simulate the unsteady state convection-diffusion equation. For the velocity fields, we will use the same as in the steady state case, $u(x, y) = 0.1$ and $v(x, y) = -0.1$.

We follow the steps found in diagram 6.5

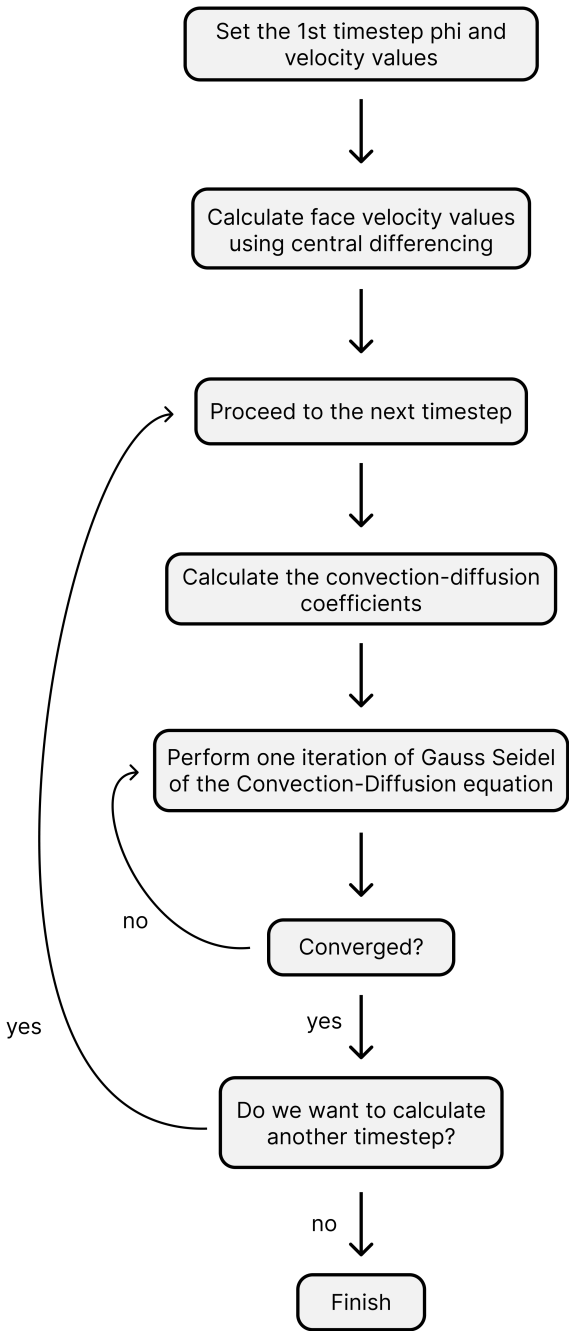
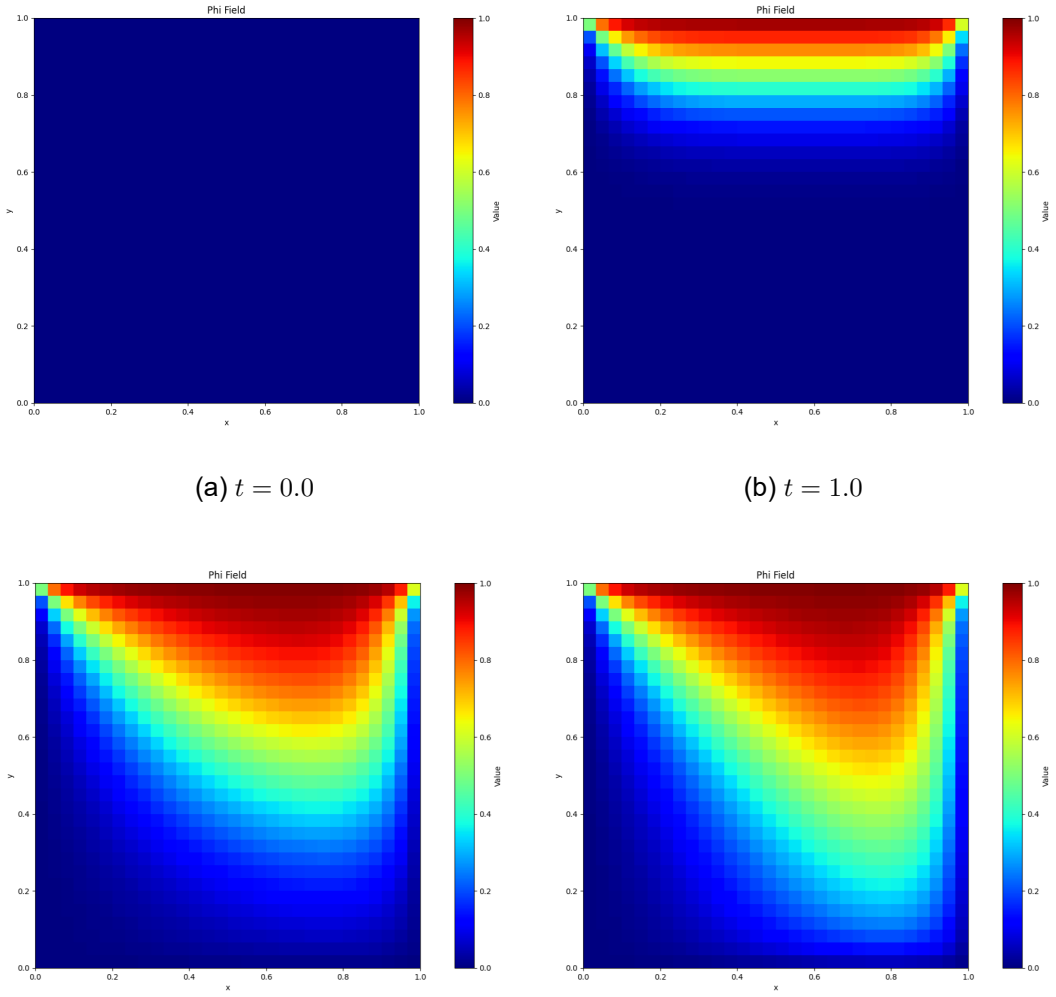


Figure 6.5: 2D unsteady state Convection-Diffusion steps

and obtain the solution of the system. Below in Figure 6.6, we pick four timesteps out of them at different times, to showcase the effect of time marching on the solution.



(a) $t = 0.0$ (b) $t = 1.0$

(c) $t = 5.0$ (d) $t = 10.0$

Figure 6.6: Time marching of 2D unsteady state Convection-Diffusion

After some time, the steady state is reached. In our case, at $t = 10.0$, the solution appears to be very close to the results we obtained when solving the 2D steady state convection-diffusion case at 6.2.3. In fact, it will keep getting closer as time marches.

7. COUPLED CONVECTION-DIFFUSION EQUATIONS

In the previous chapter 6, we solved a single convection-diffusion equation of a general variable ϕ. In this chapter, we will get a bit closer to the Navier-Stokes equations by solving a set of coupled convection-diffusion equations.

7.1 Steady State Coupled Convection-Diffusion Equations

We will use the convection-diffusion equation (6.1) and create 2 of them. Instead of ϕ, the first will have velocity u and the second will have velocity v as the variable. Both of them will have viscosity μ as the diffusion coefficient.

$$\frac{\partial(uu)}{\partial x} + \frac{\partial(vu)}{\partial y} = \mu\frac{\partial}{\partial x}\left(\frac{\partial u}{\partial x}\right) + \mu\frac{\partial}{\partial y}\left(\frac{\partial u}{\partial y}\right) \tag{7.1}$$

$$\frac{\partial(uv)}{\partial x} + \frac{\partial(vv)}{\partial y} = \mu\frac{\partial}{\partial x}\left(\frac{\partial v}{\partial x}\right) + \mu\frac{\partial}{\partial y}\left(\frac{\partial v}{\partial y}\right) \tag{7.2}$$

Our goal is to solve both equations and obtain the scalar fields u and v. The difference with equation (6.1) is that now velocities u and v are not constant. In fact, they are the variables we solve for. This poses two new problems.

- The equations are **non-linear**. This means that they contain non-linear terms such as uv,uu, or vv (multiplication of dependent variables). This is an issue because the Gauss-Seidel method we are using is not meant to be used on nonlinear equations. Note that only the convection part of the equation contains non-linear terms.
- The equations are **coupled**. This means that the solution of one equation depends on the solution of the other. In this case, both equations depend on variables u and v. This complicates things since we can't just solve the equations sequentially, one after the other.

In practice, we will use equation (7.1) to obtain the u scalar field, while (7.2) to obtain the v scalar field.

7.1.1 Discretizing the equations

Our goal here remains the same as with previous chapters: integrate over an arbitrary cell and obtain a linear equation. It used to be straightforward when dealing with equations with one variable, but if we follow the same procedure in equations (7.1) or (7.2), we will quickly realize that we arrive at a non-linear equation. This arises from the fact that the discretization scheme used for convection (upwind, for example) will be applied to both parts of the non-linear terms uu, vv, or uv.

To tackle this issue, a common practice is to linearize the nonlinear terms beforehand. This would mean assuming one of the variables in the non-linear terms uu, vv, or uv to be constant while letting the other be variable.

It is important to note that while these changes are enough to obtain linear equations through discretization, the linearization is an approximation of the equations. This, together with the fact that the equations are coupled, can be tackled with an iterative process that will be explained later.

As far as the discretization goes, we will describe the process briefly, since it is very similar to the diffusion and convection terms we discretized in 5.3.2 and 6.2.1, respectively. We will discretize the convection equation (7.1) for variable u, but the process should be very similar for equation (7.2) as well.

Diffusion term

By discretizing the diffusion term of equation (7.1), the same way we did at 5.3.1, we arrive at the flux equation for diffusion (7.4).

$$\int_{y=s}^{n}\int_{x=w}^{e}\left[\mu\frac{\partial}{\partial x}\left(\frac{\partial u}{\partial x}\right)+\mu\frac{\partial}{\partial y}\left(\frac{\partial u}{\partial y}\right)\right]dxdy=0 \tag{7.3}$$

$$\implies \mu\Delta y\left(\frac{\partial u}{\partial x}\right)_e-\mu\Delta y\left(\frac{\partial u}{\partial x}\right)_w+\mu\Delta x\left(\frac{\partial u}{\partial y}\right)_n-\mu\Delta x\left(\frac{\partial u}{\partial y}\right)_s=0 \tag{7.4}$$

We then isolate one of the four flux terms of equation (7.4), extract the differences between boundary and interior faces, and generalize the process for every cell face. By doing this, we arrive at 7.1, which is very similar to the one in 5.7.

Let $A=\dfrac{\Delta y}{\Delta x}$ if f is a west/east face or $A=\dfrac{\Delta x}{\Delta y}$ if f is a south/north face

If f is not a boundary face

- Add μA to the center coefficient
- Add μA to the coefficient in the direction of f

If f is a boundary face

- Add $2\mu A$ to the center coefficient
- Add $2\mu A u_f$ to source S

Figure 7.1: 2D Diffusive flux effects (Central differencing)

Comparing this to 5.7, the only difference is the term added to the source term if the face f is on the boundary. In this case, since our variable is u, ϕ_f got changed to u_f.

Convection term

This is the point at which we will apply the linearization. We will assume the first u of the term uu and v of the term vu to be constant. This way, at least one variable u remains in all of the terms, while all of the v components are linearized. We will use an overbar $\bar{x}$ for the linearized terms that will be considered constant. By discretizing the convection term of equation (7.1), we arrive at the flux equation for convection

7.6.

$$\int_{y=s}^{n}\int_{x=w}^{e}\left[\frac{\partial(\bar{u}u)}{\partial x}+\frac{\partial(\bar{v}u)}{\partial y}\right]dxdy=0 \tag{7.5}$$

$$\implies \overline{u_e}u_e\Delta y-\overline{u_w}u_w\Delta y+\overline{v_n}u_n\Delta x-\overline{v_s}u_s\Delta x=0 \tag{7.6}$$

Looking back at the convection flux equation for a general variable ϕ (6.6), we notice a lot of similarities to the equation (7.6). For example, $\overline{u_e}$ from equation (7.6) is exactly the same quantity as u_e from equation (6.6), and this is also true for all the other cell faces. This is indeed the case, since both of them are constant. On (6.6) it is constant because this is what we assumed from the start, and on (7.6) it is constant due to the linearization we applied. This leaves the change from variable ϕ to variable u as the only difference between the equations. This is an important note to make, since it allows us to continue the discretization the same way we did in 6.2.1. By doing so, we arrive at 7.2.

Define the values A, B, V

- $A = \Delta y$ if f is a west/east face or $A = \Delta x$ if f is a south/north face
- $B = 1$ if f is a west/south face or $B = -1$ if f is an east/north face
- $V = u_f$ if f is a west/east face or $V = v_f$ if f is a south/north face

If f is not a boundary face

- Add $A \cdot max(-BV, 0)$ to the center coefficient
- Add $A \cdot max(BV, 0)$ to the coefficient in the direction of f

If f is a boundary face

- Add $BAVu_f$ to source S

Figure 7.2: 2D Convective flux effects (Upwind)

Comparing this to figure 6.2, the only difference is the term added to the source term in case the face f is on the boundary. In this case, since our variable is u, ϕ_f got changed to u_f.

Following the procedure explained for diffusion and convection in figures 7.1 and 7.2 respectively, for every cell face, we obtain the coefficients for the convection-diffusion equation for variable u. The same process can be used to obtain the coefficients for the convection-diffusion equation for variable v. The only difference will be the change from u_f to v_f on the source term if the face f is on the boundary.

7.1.2 Example

Let's define a domain very similar to the one we had for diffusion and convection-diffusion at 5.3.3, with the only difference being the values in the boundaries. Also, note that since

we are solving for two variables, we should set boundary values for both u and v. Viscosity is equal to 1 throughout the domain.

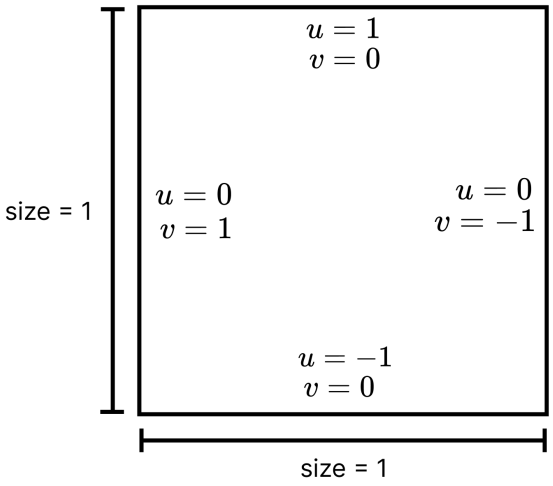


Figure 7.3: 2D domain

However, we still don't know how to tackle the issue that appeared due to the linearization of the equations, together with the fact that both equations are coupled.

The key to understanding how is to introduce the concept of **partial convergence** of an equation. Instead of fully solving each equation, one after the other, which would lead to the solution of the linearized equations, we partially solve them. This means solving the equations with lower accuracy, essentially spending less time. Typically, we aim for a reduction in the value of the residual of the equations of 1 or 2 orders of magnitude. This gives us time to repeat the process all over again. This time, however, the equation coefficients we re-calculate only reflect the new values of the velocities u and v. This way, the result of one equation is propagated to the other and vice versa. We repeat this process enough times until both equations have converged.

The procedure above is explained in steps in diagram 7.4.

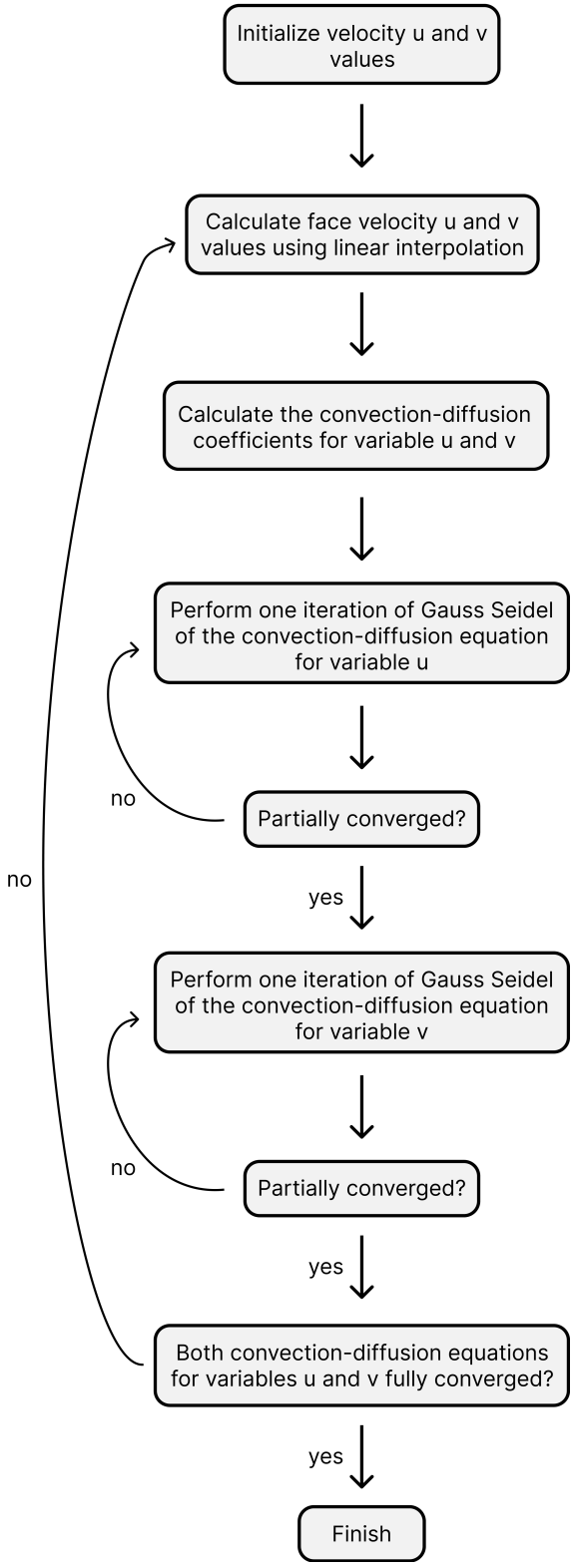


Figure 7.4: 2D steady state Coupled Convection-Diffusion steps

and obtain the solution of the two systems of equations, one for variable u and one for variable v, as demonstrated below in Figure 7.5.

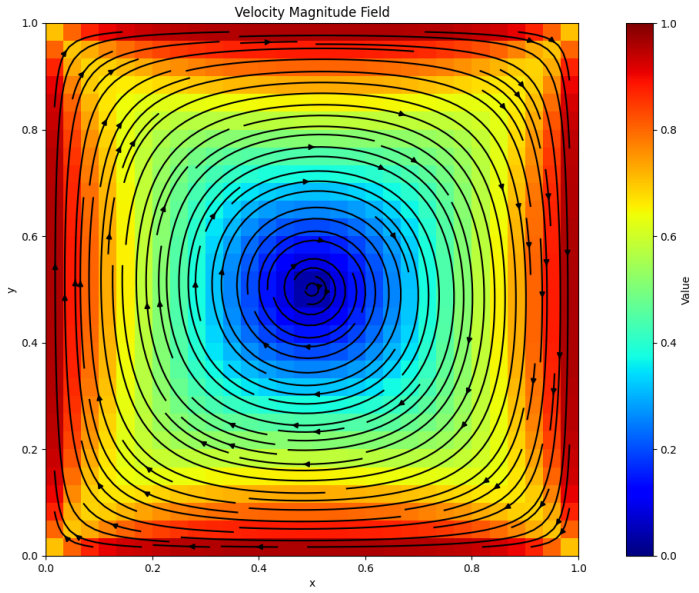


Figure 7.5: 2D Coupled Convection-Diffusion solution

Let's try to figure out what this result means. First of all, compared to previous results from diffusion and convection-diffusion simulations, we have two scalar fields (u and v) that we need to display instead of just one (ϕ). Therefore, in every cell, we calculate the velocity magnitude of those two quantities and display that in terms of color (since the velocity magnitude is also a scalar field). Furthermore, since every cell has both a velocity u and v value, we can think of that as a vector indicating direction. We can plot this in terms of streamlines. To sum up, the color indicates the speed, and the streamlines indicate the flow direction.

7.2 Unsteady State Coupled Convection-Diffusion Equations

By adding the time derivative to both equations 7.1 and 7.2, we arrive at the following system of equations

$$\frac{\partial \phi}{\partial t} + \frac{\partial (uu)}{\partial x} + \frac{\partial (vu)}{\partial y} = \mu \frac{\partial}{\partial x}\left(\frac{\partial u}{\partial x}\right) + \mu \frac{\partial}{\partial y}\left(\frac{\partial u}{\partial y}\right) \tag{7.7}$$

$$\frac{\partial \phi}{\partial t} + \frac{\partial (uv)}{\partial x} + \frac{\partial (vv)}{\partial y} = \mu \frac{\partial}{\partial x}\left(\frac{\partial v}{\partial x}\right) + \mu \frac{\partial}{\partial y}\left(\frac{\partial v}{\partial y}\right) \tag{7.8}$$

7.2.1 Discretizing the equations

The discretization process should be very similar to the process described for the steady state equations in 7.1.1. We will discretize each term of the equation separately. The following is true for the equation (7.7) for variable u, but the process should be almost identical for the equation (7.8) for variable v.

Diffusion term

We follow the procedure described in 7.1 for each cell face. The only difference is that we will multiply quantity A by Δt, since we are dealing with an unsteady state case.

Convection term

We follow the procedure described in Figure 7.2 for each cell face. The only difference, once again, is that we will multiply quantity A by Δt, since we are dealing with an unsteady state case.

Time term

We will follow a similar procedure to the one explained in 5.4.1. The only difference is the variable of the equation (u instead of ϕ).

$$\int_{y=s}^{n}\int_{x=w}^{e}\int_{t=t}^{t+\Delta t}\frac{\partial u}{\partial t}dtdxdy = 0 \tag{7.9}$$

$$\implies \Big(\Delta x\Delta y\Big)u_P = {u_P}^{\circ}\Delta x\Delta y \tag{7.10}$$

$$\implies a_P u_P = \sum_{nb}\big(a_{nb}u_{nb}\big) + S \tag{7.11}$$

where

a_W	0
a_E	0
a_S	0
a_N	0
a_P	$\Delta x\Delta y$
S	${u_P}^{\circ}\Delta x\Delta y$

Table 7.1: 2D time derivative coefficients

Following the procedure explained for diffusion and convection, for every cell face and table 7.1 for the time term, we obtain the coefficients for the unsteady state convection-diffusion equation for variable u. The same process can be followed to obtain the coefficients for the unsteady state convection-diffusion equation for variable v.

7.2.2 Example

We will use the same domain described in example 7.1.2. We will simulate 1000 timesteps with $\Delta t = 0.0001$. Therefore, the total simulation time will be $1000 \cdot 0.0001 = 0.1$.

We then follow the steps below in diagram 7.6.

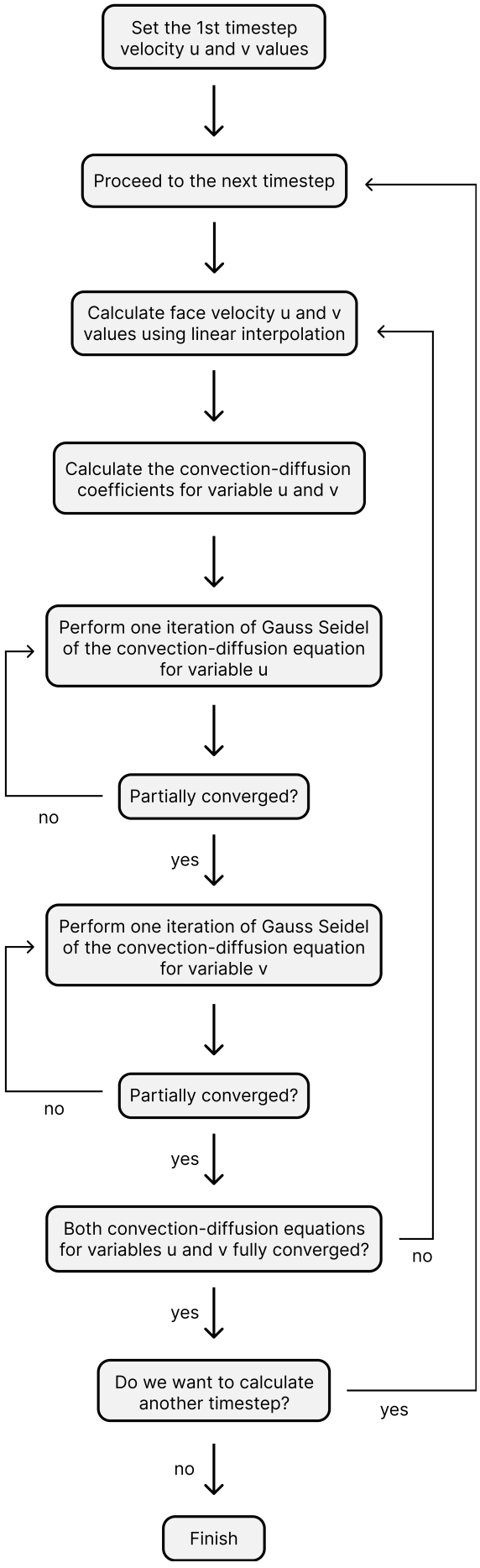
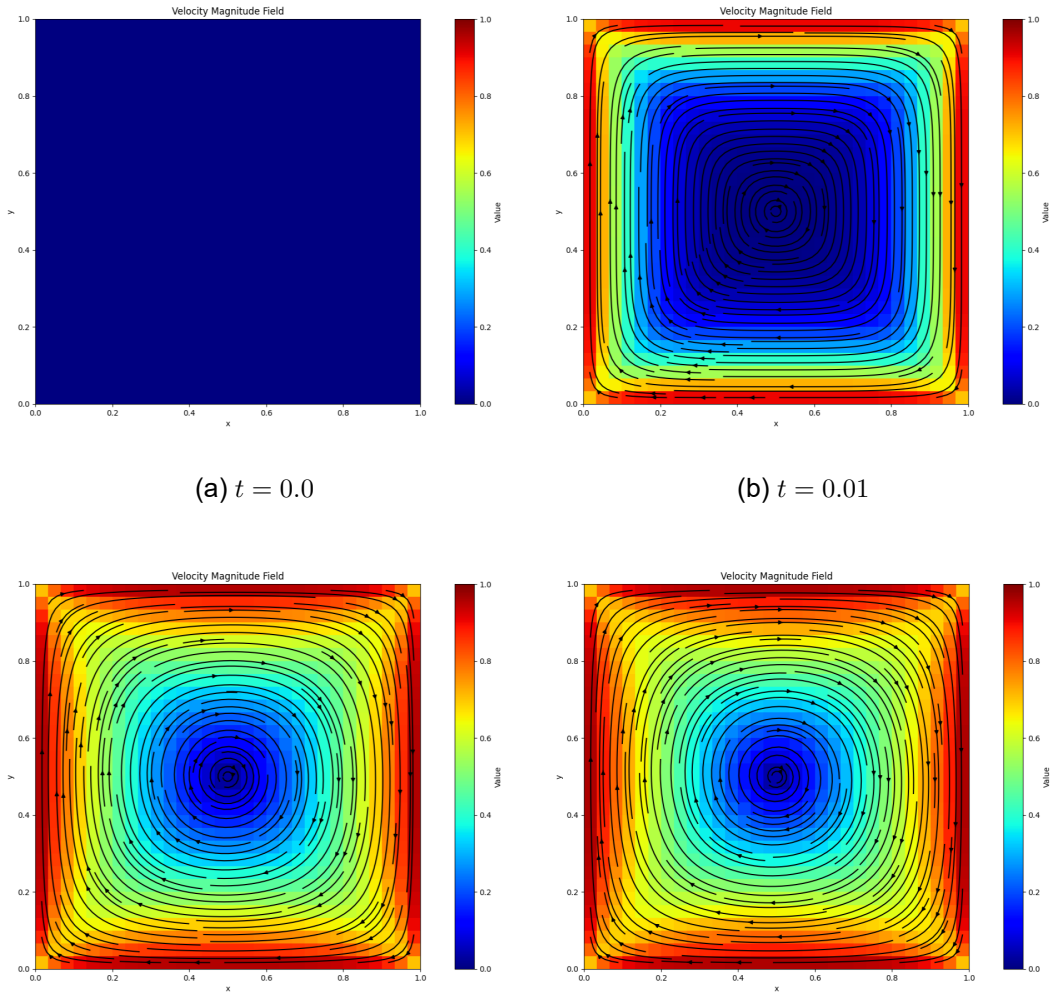


Figure 7.6: 2D unsteady state Coupled Convection-Diffusion steps

And obtain the solution of the system. Below in Figure 7.7, we pick four timesteps out of them at different times, to showcase the effect of time marching on the solution.



(a) $t = 0.0$ (b) $t = 0.01$

(c) $t = 0.05$ (d) $t = 0.1$

Figure 7.7: Time marching of 2D unsteady state Coupled Convection-Diffusion

After some time, the steady state is reached. In our case, at $t = 0.1$, the solution appears to be very close to the results we obtained when solving the 2D steady state convection-diffusion case at 7.1.2. In fact, it will keep getting closer as time marches.

8. THE NAVIER-STOKES EQUATIONS

8.1 From the Convection-Diffusion to the Navier-Stokes Equations

After solving coupled convection-diffusion equations, we are ready to jump right into the main topic of this guide: Solving the Navier-Stokes equations. As we saw in 2.6, the Navier-Stokes equations consist of 3 equations, an x-momentum, a y-momentum, and a continuity equation. A significant difference is that now we will have 3 variables instead of 2. We already have velocities u and v. The extra variable will be pressure p.

The two coupled convection-diffusion equations, which we solved in chapter 7, will be used as a basis for building up the momentum equations. Once again, they will be used to solve for the x and y velocity variables. A question we will have to answer later is how to solve for the pressure p. This is a notoriously difficult question to answer since variable p is coupled to variables u and v.

8.2 The Navier-Stokes Equations

8.2.1 Steady state

The steady state Navier-Stokes equations can be written as:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{8.1}$$

$$\rho\left[\frac{\partial(uu)}{\partial x} + \frac{\partial(vu)}{\partial y}\right] = -\frac{\partial p}{\partial x} + \mu\frac{\partial}{\partial x}\left(\frac{\partial u}{\partial x}\right) + \mu\frac{\partial}{\partial y}\left(\frac{\partial u}{\partial y}\right) \tag{8.2}$$

$$\rho\left[\frac{\partial(uv)}{\partial x} + \frac{\partial(vv)}{\partial y}\right] = -\frac{\partial p}{\partial y} + \mu\frac{\partial}{\partial x}\left(\frac{\partial v}{\partial x}\right) + \mu\frac{\partial}{\partial y}\left(\frac{\partial v}{\partial y}\right) \tag{8.3}$$

Let's compare them to the steady state coupled convection-diffusion equations (7.1) and (7.2) we solved in chapter 7. We can discern two differences. Firstly, the convection term is multiplied by the constant density ρ. Additionally, we notice the addition of the pressure derivative term $-\frac{\partial p}{\partial x}$ and $-\frac{\partial p}{\partial y}$ in the momentum x and y equations, respectively.

8.2.2 Unsteady state

Similarly, by adding the time derivative (multiplied by density ρ) to the momentum equations, we arrive at the unsteady state Navier-Stokes equations.

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{8.4}$$

$$\rho\frac{\partial u}{\partial t} + \rho\left[\frac{\partial(uu)}{\partial x} + \frac{\partial(vu)}{\partial y}\right] = -\frac{\partial p}{\partial x} + \mu\frac{\partial}{\partial x}\left(\frac{\partial u}{\partial x}\right) + \mu\frac{\partial}{\partial y}\left(\frac{\partial u}{\partial y}\right) \tag{8.5}$$

$$\rho\frac{\partial v}{\partial t}+\rho\left[\frac{\partial(uv)}{\partial x}+\frac{\partial(vv)}{\partial y}\right]=-\frac{\partial p}{\partial y}+\mu\frac{\partial}{\partial x}\left(\frac{\partial v}{\partial x}\right)+\mu\frac{\partial}{\partial y}\left(\frac{\partial v}{\partial y}\right) \tag{8.6}$$

Comparing them to the unsteady state coupled convection-diffusion equations (7.7) and (7.8), we notice 3 differences. The convection and the time terms are multiplied by density ρ, and the pressure derivative term is added to the equations.

8.3 Boundary Conditions

When solving the Navier-Stokes equations, we deal with three variables, u, v, and p. Therefore, it is necessary that we set boundary conditions for every one of those variables. Setting a boundary condition for all the variables is referred to as a **boundary condition type** (or **boundary condition set**).

In this chapter, we will be dealing with two of the most common boundary condition types. They both refer to walls and are used when fluid is not meant to pass through them, essentially acting as objects. We can model them by setting a fixed value boundary condition for velocity and a fixed gradient value for pressure. Figure 8.1 shows a cell P with an arbitrary boundary face f.

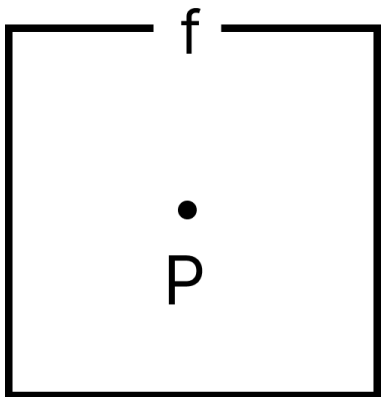


Figure 8.1: Single cell

- **Stationary Wall** Think of a 2D box or a container. We could model all of its sides by setting the boundary condition type to the stationary wall type. For boundary face f, this would mean setting $u_f = 0, v_f = 0$ and $p = p_P$
- **Moving wall** Think of a 2D box or a container and pick one of its four sides to set a moving wall boundary condition type. We set the following values assuming face f is part of that boundary side.
  - If f is a west/east face, then $u = 0, v = v_f$ and $p = p_P$
  - If f is a south/north face, then $u = u_f, v = 0$ and $p = p_P$

  That way, the wall is moving perpendicular to its surface. For instance, for face f in Figure 8.1, the wall would move to the left if $u_f < 0$ and to the right if $u_f > 0$. If $u_f = 0$, then the wall becomes stationary.

8.4 Discretizing the momentum equations

We will discretize both the steady and unsteady state x-momentum equations (8.2) and (8.5). The process is very similar for the y-momentum equations (8.3) and (8.6) as well.

8.4.1 Discretizing all the individual parts

As we have done for every equation so far, we discretize the equations by integrating over an arbitrary cell and obtain a linear equation. Both momentum equations are coupled and non-linear, so a linearization of the convective term, combined with an iterative process similar to the one explained in chapter 7, is necessary. We will discretize every part of the equation individually.

Diffusion term

Nothing changes compared to the process described in Figure 7.1.

Convection term

The term that we have to discretize is

$$\rho\left[\frac{\partial(uu)}{\partial x}+\frac{\partial(vu)}{\partial y}\right] \tag{8.7}$$

It is essentially the same as the convection term of the coupled convection-diffusion equation (7.1) for variable u, with the only difference being that it is now multiplied by the density ρ. It can be seen that during the discretization process, this results in all of the coefficients being multiplied by ρ. We reach the following steps in 8.2.

Define the values A, B, V

- $A = \Delta y$ if f is a west/east face or $A = \Delta x$ if f is a south/north face
- $B = 1$ if f is a west/south face or $B = -1$ if f is an east/north face
- $V = u_f$ if f is a west/east face or $V = v_f$ if f is a south/north face

If f **is not a boundary face**

- Add $\rho A \cdot max(-BV, 0)$ to the center coefficient
- Add $\rho A \cdot max(BV, 0)$ to the coefficient in the direction of f

If f **is a boundary face**

- Add $\rho BAVu_f$ to source S

Figure 8.2: 2D Convective flux effects (Upwind)

Notice how, in comparison to figure 7.2, the only difference is the multiplication of the coefficients by ρ.

Pressure term

By isolating and discretizing the pressure term, we arrive at

$$\int_{y=s}^{n}\int_{x=w}^{e} -\frac{\partial p}{\partial x}dxdy = 0 \tag{8.8}$$

$$\implies \int_{y=s}^{n} -\big(p_e - p_w\big) = 0 \tag{8.9}$$

$$\implies \big(p_w - p_e\big)\Delta y = 0 \tag{8.10}$$

If we were to do the same thing for the pressure term of the y-momentum equation, we would arrive at

$$\big(p_s - p_n\big)\Delta x = 0 \tag{8.11}$$

At this point, it is important to note that although pressure p is a variable that we have to solve for, as far as solving the momentum equations go, it is also linearized, the same way some velocities were during the discretization of the convection term. This is necessary so that we can obtain a linear equation with only one variable, velocity u (or v for y-momentum). This means that the pressure values at the cell faces are known beforehand and can be calculated using linear interpolation (the same way as linearized velocities). For example, for internal cells, $p_w = \frac{p_W + p_P}{2}$. If the cell is on the boundary, in case of a fixed value boundary condition, the value on the boundary is used, while in case of a fixed gradient value, the cell value is used. Therefore, both equations (8.10) and (8.11) can be written in the form

$$a_P\phi_P = \sum_{nb}\big(a_{nb}\phi_{nb}\big) + S \tag{8.12}$$

where

	Momentum X	Momentum Y
a_W	0	0
a_E	0	0
a_S	0	0
a_N	0	0
a_P	0	0
S	$\big(p_w - p_e\big)\Delta y$	$\big(p_s - p_n\big)\Delta x$

Table 8.1: 2D pressure term coefficients

Time term

The time term that we want to discretize is

$$\rho \frac{\partial u}{\partial t} \tag{8.13}$$

It is essentially the same as the time term of the coupled convection-diffusion equation (7.1) for variable u, with the only difference that it is now multiplied by the density ρ. It can be seen that during the discretization process, this results in all of the coefficients being multiplied by ρ. We reach the following table 8.2.

	Momentum X	Momentum Y
a_W	0	0
a_E	0	0
a_S	0	0
a_N	0	0
a_P	$\rho \Delta x \Delta y$	$\rho \Delta x \Delta y$
S	$u_P{}^\circ \rho \Delta x \Delta y$	$v_P{}^\circ \rho \Delta x \Delta y$

Table 8.2: 2D time term coefficients

8.4.2 Obtaining the steady state momentum coefficients

Following the procedure explained for diffusion and convection, for every cell face and table 8.1 for the pressure term, we add the coefficients and obtain the coefficients for the steady state x-momentum equation for variable u. Using a similar procedure, we can obtain the coefficients for the y-momentum equation for variable v.

8.4.3 Obtaining the unsteady state momentum coefficients

Following the procedure explained for diffusion and convection, for every cell face, the table 8.1 for the pressure term, and the table 8.2 for the time term, we add the coefficients and obtain the coefficients for the unsteady state x-momentum equation for variable u. Using a similar procedure, we can obtain the coefficients for the y-momentum equation for variable v.

Note that, since we are dealing with the unsteady state case, the convection, diffusion, and pressure coefficients will also have to be multiplied by Δt. This is a result of the implicit time scheme.

8.5 The SIMPLE Algorithm

We have already obtained equations for solving for variables u and v (x-momentum and y-momentum, respectively). However, we don't have any equation to solve for the pressure p variable. There is one equation left that we haven't discretized yet, the continuity equation, but at first glance, it doesn't contain any pressure variables and thus, can't be directly used to solve for variable p.

The SIMPLE algorithm [45] tackles this by transforming the continuity equation into a pressure-correction equation that corrects the pressure values. However, since p, u, and v are coupled, changing any one of the three results in a change in the other two, so an iterative procedure is yet again unavoidable.

8.5.1 Rhie-Chow interpolation

Rhie-Chow interpolation [48] refers to how internal face velocities are calculated. So far, we have been using a mean formula between the adjacent cell values to obtain internal face velocities. However, when using the SIMPLE algorithm, it is necessary to change the calculation formula, or else the solution might oscillate between different values. This can happen for reasons beyond the scope of this guide. Imagine we are trying to calculate the velocity of face f, between cells 0 and 1, shown in figure 8.3.

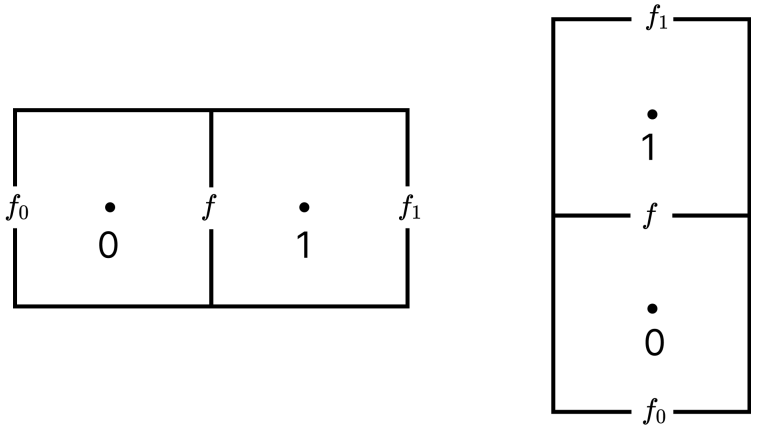


Figure 8.3: 2-cell grid (x-axis on the left and y-axis on the right)

Normally, we would use the linear interpolation formula

$$u_f = \frac{1}{2}(u_0 + u_1) \tag{8.14}$$

However, when implementing the SIMPLE algorithm we will be using the following formula where $A = \Delta y$ if f is a face on the x-axis (see left grid of Figure 8.3) and $A = \Delta x$ if f is a face on the y-axis (see right grid of Figure 8.3).

$$u_f = \frac{1}{2}(u_0 + u_1) + \frac{1}{2}A\Delta t\left[\frac{1}{a_{P_0}}(p_f - p_{f_0}) + \frac{1}{a_{P_1}}(p_{f_1} - p_f) - \left(\frac{1}{a_{P_0}} + \frac{1}{a_{P_1}}\right)(p_1 - p_0)\right] \tag{8.15}$$

Notice the values a_{P_0} and a_{P_1}. They refer to the center coefficients of the momentum equation of cells 0 and 1, respectively. Specifically, to the momentum-x coefficients, if f is a face on the x-axis, and to the momentum-y coefficients if f is a face on the y-axis. Therefore, it is logical to infer that calculating face velocities using Rhie-Chow interpolation can only be done after obtaining the momentum x and y coefficients.

Similar to formula (8.14), (8.15) only works for calculating internal face f velocities and is valid for both x-axis and y-axis cases shown in grids in Figure 8.3. Boundary face velocities are calculated based on the boundary conditions.

8.5.2 Obtaining the pressure-correction equation

As mentioned in 8.5, we can obtain the pressure-correction equation from the continuity equation. Our goal here is to construct an equation that we can use to solve for the pressure correction p', instead of pressure p, such that $p_{new} = p_{old} + p'$.

We first discretize the continuity equation

$$\int_{y=s}^{n}\int_{x=w}^{e}\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right)dxdy = 0 \tag{8.16}$$

$$\implies \int_{y=s}^{n}\int_{x=w}^{e}\frac{\partial u}{\partial x}dxdy + \int_{x=w}^{e}\int_{y=s}^{n}\frac{\partial v}{\partial y}dydx = 0 \tag{8.17}$$

$$\implies \int_{y=s}^{n}\big(u_e - u_w\big)dy + \int_{x=w}^{e}\big(v_n - v_s\big)dx = 0 \tag{8.18}$$

$$\implies \Delta y\big(u_e - u_w\big) + \Delta x\big(v_n - v_s\big) = 0 \tag{8.19}$$

$$\implies u_e\Delta y - u_w\Delta y + v_n\Delta x - v_s\Delta x = 0 \tag{8.20}$$

Equation (8.20) is called the continuity flux equation. Normally, we would apply a discretization scheme to u and v face values, but doing so will result in an equation with two variables and no pressure correction variables. We will use the following formula (8.21) [41], in order to arrive at an equation with variable p'. Imagine a grid, exactly like the one in figure 8.3.

$$\hat{u_f} = u_f + \frac{1}{2}\left(\frac{1}{a_{P_0}}+\frac{1}{a_{P_1}}\right)A\Delta t\big(p'_0 - p'_1\big) \tag{8.21}$$

Here, u_f is a known value that refers to the face velocity at the time during the generation of the pressure correction equation coefficients. $A = \Delta y$ if f is a face on the x-axis and $A = \Delta x$ if f is a face on the y-axis. Values a_{P_0} and a_{P_1} refer to the center coefficients of the momentum equation of cells 0 and 1, respectively. Specifically, the momentum-x coefficients if f is a face on the x-axis, and the momentum-y coefficients if f is a face on the y-axis. This formula can also be used for v_f.

Just like we have done with diffusion or convection parts, we will isolate one of the four fluxes of the continuity flux equation and see how it behaves. For example, let's select the flux on the west face w.

$$u_w \Delta y = 0 \tag{8.22}$$

If the west face is not a boundary

By applying formula (8.21) to the west flux equation (8.22), we obtain

$$(8.22) \implies \Delta y \left[u_w + \frac{1}{2} \left(\frac{1}{a_{P_0}} + \frac{1}{a_{P_1}} \right) \Delta y \Delta t \left(p'_W - p'_P \right) \right] = 0 \tag{8.23}$$

$$\implies \frac{1}{2} \left(\frac{1}{a_{P_0}} + \frac{1}{a_{P_1}} \right) \Delta y^2 \Delta t \left(p'_W - p'_P \right) - u_w \Delta y = 0 \tag{8.24}$$

$$\implies \left[\frac{1}{2} \left(\frac{1}{a_{P_0}} + \frac{1}{a_{P_1}} \right) \Delta y^2 \Delta t \right] p'_W - \left[\frac{1}{2} \left(\frac{1}{a_{P_0}} + \frac{1}{a_{P_1}} \right) \Delta y^2 \Delta t \right] p'_P - u_w \Delta y = 0 \tag{8.25}$$

$$\implies \left[\frac{1}{2} \left(\frac{1}{a_{P_0}} + \frac{1}{a_{P_1}} \right) \Delta y^2 \Delta t \right] p'_P = \left[\frac{1}{2} \left(\frac{1}{a_{P_0}} + \frac{1}{a_{P_1}} \right) \Delta y^2 \Delta t \right] p'_W - u_w \Delta y \tag{8.26}$$

Therefore, the west continuity flux will have the following effects on the pressure-correction equation coefficients:

- Add $\frac{1}{2} \left(\frac{1}{a_{P_0}} + \frac{1}{a_{P_1}} \right) \Delta y^2 \Delta t$ to the center coefficient
- Add $\frac{1}{2} \left(\frac{1}{a_{P_0}} + \frac{1}{a_{P_1}} \right) \Delta y^2 \Delta t$ to the west coefficient
- Subtract $u_w \Delta y$ from the source S

If the west face is a boundary

Since for the rest of this chapter we will only be using wall boundary conditions, the value of velocity u and v at the boundary faces will be constant and known beforehand. Therefore, the west continuity flux will have the following effects on the pressure-correction equation coefficients:

- Subtract $u_w \Delta y$ from the source S

This was the procedure for the west face of a cell. We can do the same thing for all the faces of a cell and arrive at the following general steps for an arbitrary cell face f.

Define the values A, B, V

- $A = \Delta y$ if f is a west/east face or $A = \Delta x$ if f is a south/north face
- $B = 1$ if f is a west/south face or $B = -1$ if f is an east/north face
- $V = u_f$ if f is a west/east face or $V = v_f$ if f is a south/north face

If f is not a boundary face

- Add $\frac{1}{2}\left(\frac{1}{a_{P_0}} + \frac{1}{a_{P_1}}\right)A^2\Delta t$ to the center coefficient
- Add $\frac{1}{2}\left(\frac{1}{a_{P_0}} + \frac{1}{a_{P_1}}\right)A^2\Delta t$ to the coefficient in the direction of f
- Add BVA to the source S

If f is a boundary face

- Add BVA to the source S

Figure 8.4: 2D Continuity flux effects

All we have to do for every cell in order to create the discretized equation and obtain the equation coefficients is to follow the steps found in 8.4 for each one of the four faces of the cell and add their coefficients.

8.5.3 The SIMPLE algorithm

We are now ready to dive into the algorithm responsible for combining everything we have learned so far and solving the Navier-Stokes equations. Effectively, the SIMPLE algorithm solves the problem of velocity-pressure coupling [41] mentioned in 8.5 and consists of an iterative algorithm. We define the steps below:

Step 1: Initialization

Initialize the cell velocity u, v, and pressure p values. Right after, calculate the face velocity u, v, and pressure p values using linear interpolation. Note that we don't use Rhie-Chow interpolation for face velocities yet, since that requires the momentum coefficients, which we haven't obtained yet.

Step 2: Perform an iteration of the SIMPLE algorithm

Step 2.1: Calculate the momentum x and y coefficients

Calculate the momentum x and y coefficients, as explained in 8.4.2.

Afterwards, implicit under-relaxation is applied to the momentum coefficients. It is important since otherwise, the algorithm would most likely diverge. How to do that is explained in 4.5. Essentially, we divide the center coefficient of every cell by a constant number α, where $0 < \alpha < 1$. The value of α is not the same

for every simulation. Typically, we start with a value of 0.7 for both momentum x and y and change it as needed. As mentioned in 4.5, smaller values lead to more stability but lower speed, while bigger values lead to less stability but greater speed.

It should be noted that the order in which we calculate the momentum x and y coefficients doesn't matter, since both steps do not depend on each other.

Step 2.2: Partially solve the momentum x and y equations

For both equations, we perform Gauss-Seidel iterations until they partially converge. This way, we get a partial solution for variable velocities u and v.

Once again, it doesn't matter what equation we solve first, or even if we solve them in parallel, since they do not depend on each other. This is true as long as both momentum x and y coefficients were calculated beforehand.

Step 2.3: Calculate face velocity u and v values using Rhie-Chow interpolation

We recalculate the face velocities. However, this time, since we have already obtained the momentum coefficients, we use Rhie-Chow interpolation instead of linear interpolation.

Step 2.4: Calculate the pressure correction equation coefficients

Calculate the pressure correction equation coefficients, as explained in 8.5.2. Note that during the calculation, we use the implicitly relaxed momentum x and y coefficients we previously calculated in **Step 2.1**.

Step 2.5: Partially solve the pressure correction equation

We perform Gauss-Seidel iterations until the equation partially converges. This way, we get a partial solution for pressure correction p'.

Step 2.6: Calculate the face pressure correction p' values using linear interpolation

Step 2.7: Correct pressure values

Firstly, we have to correct the cell pressure values using the following formula

$$p_{new} = p_{old} + \alpha \cdot p' \tag{8.27}$$

where α is the pressure relaxation factor. Once again, smaller values lead to more stability but lower speed, while bigger values lead to less stability but greater speed. Typically, we start with a value of 0.3 and adjust it as needed, depending on the simulation.

Afterwards, we update the face pressure values using linear interpolation, the same way we did in **Step 1**.

Step 2.8: Correct velocity values

Since velocity and pressure are coupled, a change in pressure results in a change in velocity.

Firstly, we have to correct the cell velocity values. Suppose we need to calculate u_0 in the following figure 8.5

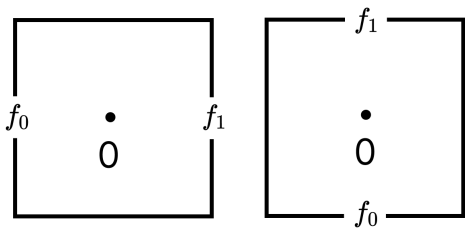


Figure 8.5: 1-cell grid (x-axis on the left and y-axis on the right)

We will use the following formula, which uses the pressure correction face values we calculated in **Step 2.6**.

$$u_0 = A\Delta t \frac{p'_{f_0} - p'_{f_1}}{a_P} \tag{8.28}$$

Afterwards, we correct the face velocity values using formula (8.21).

Note that we correct both u and v velocities.

Step 3: Check for convergence

Right after performing an iteration of the SIMPLE algorithm, we check for convergence. For the momentum x and y equations, this means checking if they have sufficiently converged. For the pressure correction equation, this means checking if the mass imbalance has sufficiently decreased.

If all 3 have converged, then the solution is obtained. If not (i.e., at least one equation has not converged), then we go back to **Step 2** and perform another iteration of the SIMPLE algorithm.

8.6 Example With a Lid-driven Cavity

We will simulate one of the most famous benchmark problems in CFD, the **lid-driven cavity**. This problem, while being interesting on its own, is also very commonly used to check whether a CFD code works or not.

Figure 8.6 shows the domain.

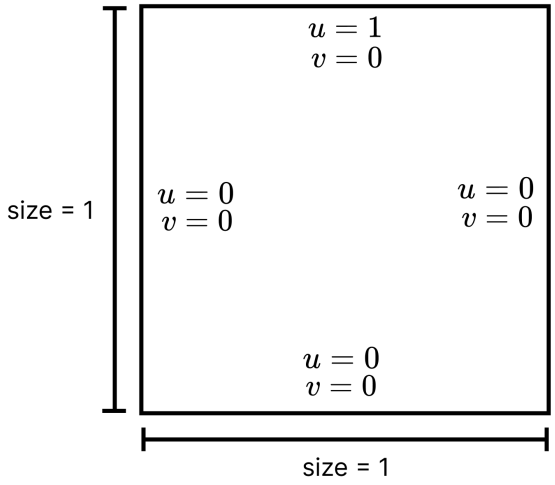


Figure 8.6: 2D Lid-driven cavity domain

Think of a box containing some fluid (cavity). Pick one side of the box (lid), let's say the north side, and make it move parallel to its surface. Specifically in Figure 8.6, the u velocity is positive, so the lid moves horizontally from left to right. Essentially, the boundary conditions types are Stationary walls on the west, east, and south sides and a Moving wall on the north side. We start with uniform velocity x, y, and pressure fields all equal to 0.

The initial velocity of the fluid throughout the domain is 0, which means that there is initially no motion. However, as time marches, we expect that the moving lid of the box will affect the fluid inside, resulting in a swirling pattern called a vortex (also known as an eddy).

For the purposes of this guide, we will conduct the simulation in a 1×1 domain consisting of a 50×50 grid. Viscosity is 0.001, while density is 1. Relaxation factors used for the SIMPLE algorithm are 0.7 for velocity u and v and 0.3 for pressure p.

8.6.1 Steady state

We follow the procedure explained for the SIMPLE algorithm in 8.5.3 and obtain the result shown in Figure 8.7. The magnitude of the velocity field is displayed using color, while streamlines indicating the direction are also plotted.

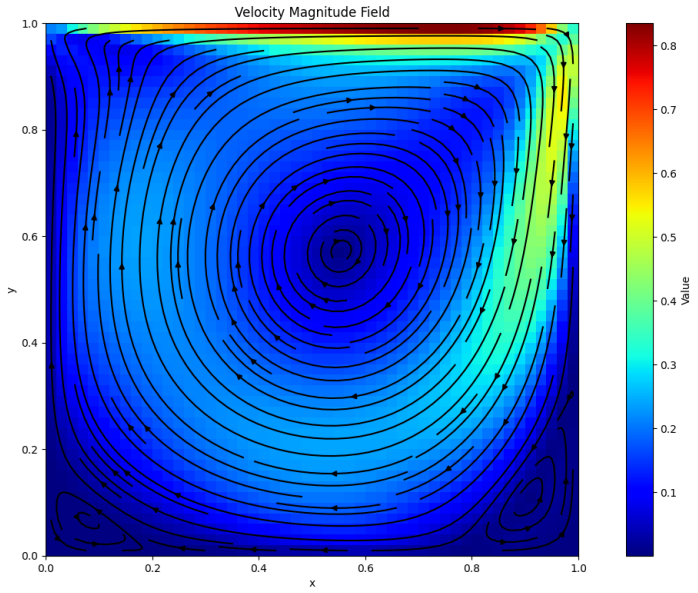
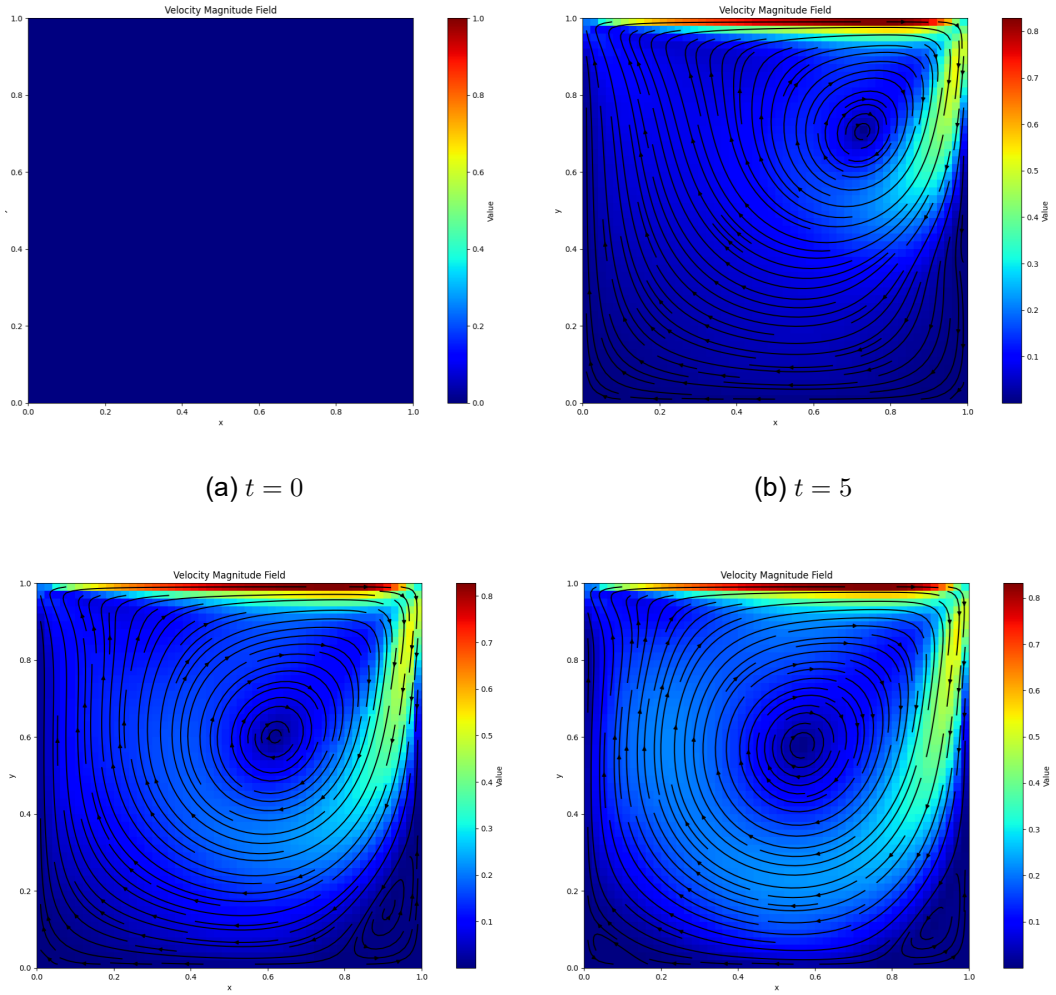


Figure 8.7: Lid-driven cavity steady state solution for $\mathrm{Re} = 1000$

Notice the main vortex covering most of the box, as well as the other 2 smaller vortices forming on both bottom corners.

8.6.2 Unsteady state

We will use the same domain described in the steady case. We will simulate 2000 timesteps with $\Delta t = 0.01$. Therefore, the total simulation time will be $2000 \cdot 0.01 = 20$. We pick four timesteps to showcase the effects of time marching to the solution in Figure 8.8.



(a) $t = 0$ (b) $t = 5$

(c) $t = 10$ (d) $t = 20$

Figure 8.8: Time marching of unsteady state Lid-driven cavity for $\mathrm{Re} = 1000$

After some time, the steady state is reached. In our case, at $t = 20$, the solution appears to be very close to the results we obtained when solving the steady state case at 8.6.1. In fact, it will keep getting closer as time marches.

9. DYE

Sometimes, it is very useful to trace the flow of the fluid to capture its motion clearly. This is especially important in simulations with only one type of fluid since it has a uniform color.

This can be achieved by injecting dye into the fluid and tracing its flow.

Note that while dye injection can be performed in both steady and unsteady state simulations, it only makes sense to do it in unsteady state ones, since what matters to us is the motion of the fluid through time.

There are a lot of ways to implement it. One way is by introducing a new variable scalar field c for the dye concentration [53]. Let $0 \leq c \leq 1$ at every point on the field. A value of 0 means no dye concentration, while a value of 1 means full dye concentration. Intermediate values are also valid. We will formulate an equation that solves for c.

9.1 The 2D Unsteady state Dye Equation

The dye equation is almost identical to the convection-diffusion equation (6.12) we solved in chapter 6. In fact, the only difference is the variable that we solve for: Instead of ϕ, it is now c.

$$\frac{\partial c}{\partial t} + \frac{\partial (uc)}{\partial x} + \frac{\partial (vc)}{\partial y} = \Gamma \frac{\partial}{\partial x}\left(\frac{\partial c}{\partial x}\right) + \Gamma \frac{\partial}{\partial y}\left(\frac{\partial c}{\partial y}\right) \tag{9.1}$$

9.2 Discretizing the Dye Equation

As mentioned in 9.1, the only difference between the dye and the convection-diffusion equation is the variable c that we solve for. Therefore, obtaining the dye equation coefficients should be easy by following the same procedure found in chapter 6.

9.3 Solving the Dye Equation

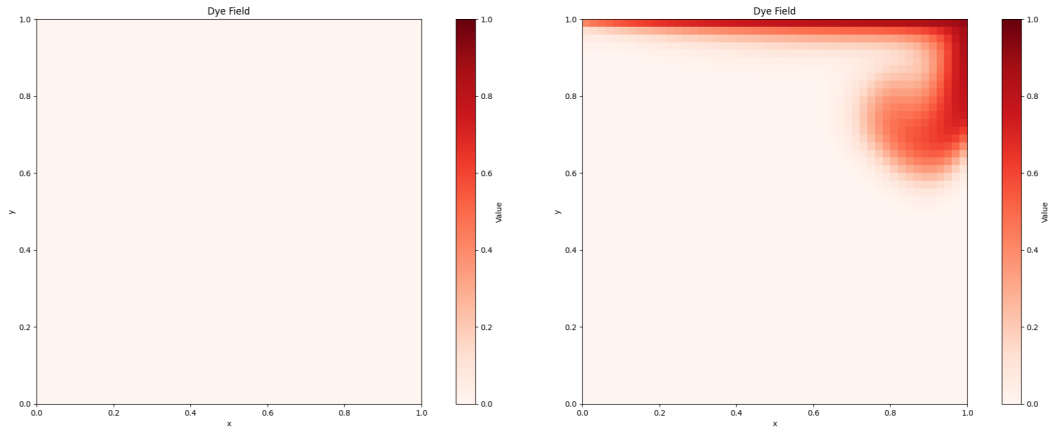
Once again, we are going to simulate the lid-driven cavity problem. This time, however, we will inject some dye on the north side, which will propagate with the flow.

We will use the same domain found in Figure 8.6. We will simulate 1000 timesteps with $\Delta t = 0.01$. Therefore, the total simulation time will be $1000 \cdot 0.01 = 10$. We will start with a uniform c field with a value of 0.

Instead of plotting the velocity field, we will plot the dye field. This is the scalar field of the variable c. We will also use a different colormap, which better showcases the dye concentration. Notice how it is white in areas with no dye and red in areas with dye.

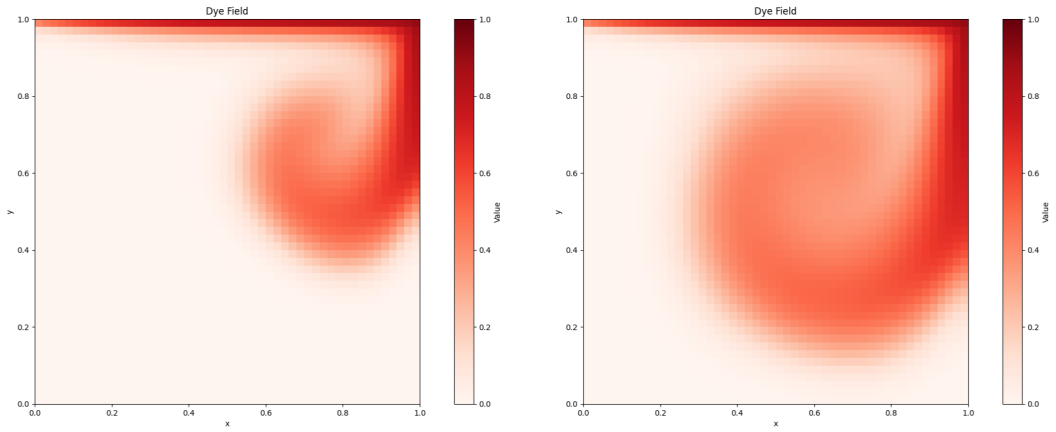
Since c is a variable we solve for, it is necessary to provide boundary conditions. For this specific problem, the north side has a fixed value of $c = 1$, effectively continuously coloring the fluid that makes contact with it red, while the other sides copy the value of their cell, so $c = c_P$.

An important question to ask is when to solve the dye equation. Right after calculating a timestep using the SIMPLE algorithm, the velocity field for this specific timestep is obtained. Therefore, it can be used to calculate the dye equation coefficients. Right after, we solve the equation itself using Gauss-Seidel and obtain the scalar field c for this specific timestep.



(a) $t = 0$

(b) $t = 2.5$



(c) $t = 5$

(d) $t = 10$

Figure 9.1: Time marching of unsteady state Lid-driven cavity for $\mathrm{Re} = 1000$

Note that if we were to continue the simulation for more timesteps, at some point the fluid in the cavity would become completely red, since the north side is continuously coloring the fluid that makes contact with it.

10. BOUNDARY CONDITIONS

In 8.3, we encountered two types of boundary conditions:

- Stationary wall (no-slip)
- Moving wall

These work great when dealing with enclosed regions. However, many problems cannot be solved with just these. In this chapter, we will look into some other boundary condition types.

10.1 Boundary Condition Types

As with stationary and moving walls, each boundary condition type defines the behavior of the variables velocity u, velocity v, and pressure p at a given boundary face f. Note that f can be any of the four sides of the cell.

10.1.1 Inlet

Inlets refer to regions where fluid is being inserted into the domain.

- $u = u_f$, where u_f is a constant
- $v = v_f$, where v_f is a constant
- $p = p_P$

10.1.2 Outlet

Outlets refer to regions where fluid is being removed from the domain.

- $u = u_P$
- $v = v_P$
- $p = p_f$, where p_f is a constant (usually equal to 0)

10.1.3 Stationary wall (slip)

They are very similar to no-slip stationary walls. They prohibit flow perpendicular to the wall but allow flow in parallel with it.

- $u = u_P$ if f is a south/north face or $u = 0$ if f is a west/east face
- $v = 0$ if f is a south/north face or $v = v_P$ if f is a west/east face
- $p = p_P$

10.1.4 Periodic

This type of boundary condition type is used in cases where the given domain is repeated in a specific orientation (horizontal or vertical).

In a way, it is not really a boundary condition type in the sense that it is implemented by simply connecting the opposing grid cells which are periodically connected.

For example, we could have a domain whose left and right sides are periodically connected. In this case, fluid flowing towards one side will come out of the other and vice versa. The cells comprising the domain's left and right sides are considered interior and not boundary cells: The cells on the left side have the cells on the right side as west neighbors. Respectively, the cells on the right side have the cells on the left side as east neighbors. Thus, both sides share the same cell faces. This is important so as not to introduce duplicate faces.

10.1.5 Free

Free boundary conditions are used in regions where the flow has fully developed. They can function as either inlets or outlets (or both).

- $u = u_P$
- $v = v_P$
- $p = p_P$

10.1.6 Others

There are multiple other boundary condition types that are not covered in this guide. However, the ones introduced above can be utilized to solve most fluid flow problems and, therefore, form a strong basis in case the reader wants to expand their knowledge.

11. CONVERGENCE

A common challenge in most CFD simulations is convergence. Sometimes, simulations diverge. Other times, simulations converge, but the result is not the desired one. There are many factors that can affect convergence (or divergence). At first, it can seem daunting, especially to beginners. It is easy to get lost in trying to figure out the cause of a divergence. Therefore, we will try to identify some of the most crucial factors and pitfalls that can affect convergence so that we can avoid them.

11.1 Convergence factors

Some of the most common criteria that can affect convergence are:

- **Mesh quality and Grid size (Δx or Δy)**

  The grid size (i.e., the number of cells on the grid) determines the accuracy of the simulation. The finer the grid (more and smaller cells), the better the accuracy of the simulation. If the grid is not fine enough, the cells sometimes cannot accurately model the geometry of the problem, as we saw in chapter 3. For example, take a look at figure 3.3. This often hinders convergence, since the simulation cannot produce a physically realistic result.

  A solution could be to refine the grid by adding more, smaller cells to better capture the geometry of the problem. However, while it can partially solve the convergence problems, it greatly increases computational time. Ideally, in such cases, another grid type should be used.

- **Choice of Δt**

  This only refers to unsteady state simulations, since steady state ones don't use Δt.

  The choice of Δt is problem-dependent. Sometimes, we require a detailed analysis during a short period of time, so Δt is lower, while other times require the state of the simulation in the long term, in which case Δt could be larger.

  However, in any case, there is an important constraint that we always have to take into consideration. This is the **CFL condition** [2]. It states that the following inequality must be satisfied at any point.

$$\frac{u\Delta t}{\Delta x} \leq 1 \tag{11.1}$$

  where u is the velocity magnitude, Δx is the distance between 2 cell centroids, and Δt is the timestep. Failing to satisfy the CFL condition will more than likely lead to divergence.

  If the condition is unsatisfied (i.e., left-hand side is greater than 1), two solutions exist. We can either decrease Δt or make the grid coarser (i.e., reduce the number of cells in the grid by making the cells bigger). Note that modifying u is impossible since that would alter the nature of the simulation.

  The CFL condition essentially expresses the speed of information propagation. The simulation is stable at any iteration and doesn't diverge, only if information is propagated from a cell to its closest neighbors and not beyond them [31].

- **Discretization schemes**

  The choice of discretization schemes for the different parts of the equations can have a detrimental effect on convergence. The simulation will converge only if the Scarborough criteria discussed at 4.4 are met.

  Luckily, the Upwind scheme for convection, together with the central differencing scheme for diffusion unconditionally satisfy the Scarborough criteria and, therefore, do not hinder convergence. However, the user should be careful in case the need for another discretization scheme arises.

- **Relaxation factors**

  This refers to the relaxation factors that the SIMPLE algorithm uses. If the simulation diverges, the relaxation factors should likely be lowered. As discussed in 8.5.3, a good starting point is a value of 0.7 for velocity u and v and 0.3 for pressure p.

  However, it should be noted that radical changes to the relaxation factors often do not solve the underlying problem and can very quickly slow down the simulation. Therefore, the other factors should be first considered before tuning the relaxation factors.

11.2 Common pitfalls

Some of the most common pitfalls that lead to divergence or an undesired result are:

- **Non-existing steady state**

  When trying to calculate the steady state of a simulation, we can encounter an issue. Not all simulations lead to a steady state, which can happen due to either the existence of turbulence or periodic behavior.

  The higher the Reynolds number, the higher the turbulence. Its existence might cause instabilities that are too significant to result in convergence. One can identify it by observing how the residuals progress through the iterations. Typically, the residuals will keep reducing until a specific amount of accuracy is reached, after which turbulence is evident. After this point, the residuals will keep oscillating, making convergence impossible.

  However, even in cases where turbulence is not evident, a periodic behavior is guaranteed not to result in convergence. For the same reasons, the residuals will keep oscillating.

  Therefore, it is important that whenever possible, the user analyzes the problem beforehand to predict whether the simulation can indeed result in a steady state or not. This is obviously not an issue when we perform unsteady-state simulations.

- **Unrealistic solution**

  Whether we perform a steady or an unsteady simulation, sometimes it will converge with an undesirable result. Maybe it is close to the result we expect, but not quite right, or maybe it is utterly wrong. This can happen for a number of reasons.

  The first reaction should be to check if the modeling of the problem itself is correct. One can refer to 11.1 for possible reasons. Another possible culprit is a mistake in the code implementation itself.

However, sometimes, simply decreasing the tolerance, making it stricter, might solve the issue. This can help because what we may be experiencing is an unfinished simulation that was calculated with lower accuracy than needed.

12. GALLERY

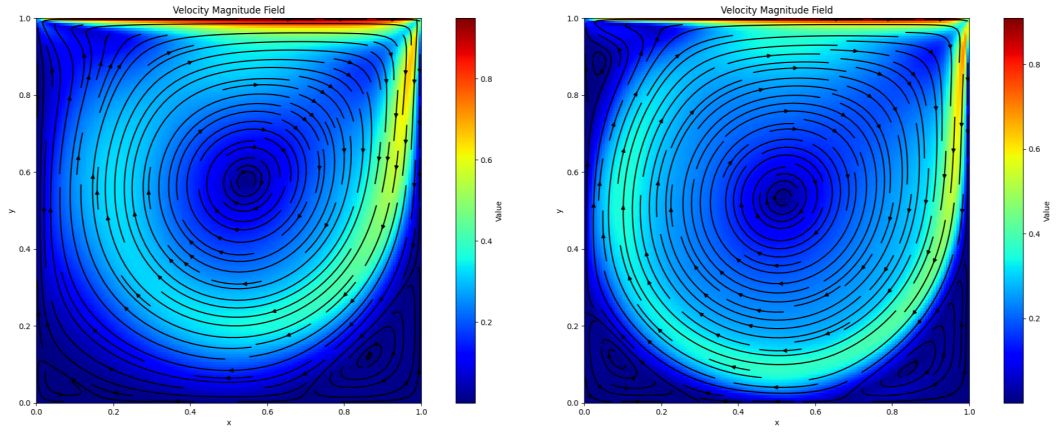
In this chapter, we will run simulations of various well-known problems. These problems can also serve as benchmarks for writing simulators.

Note that we will use Hayase's QUICK scheme [52] for convection instead of Upwind for all of the simulations below. This scheme provides better accuracy while sacrificing computational time.

The simulations below are conducted on a Qualcomm Snapdragon X Plus ARM-based processor.

12.1 Lid-driven Cavity

We will revisit the lid-driven cavity problem that we saw in 8.6. Only this time, we will use a finer grid of 150×150 cells. Velocity and density are set to 1 in both cases, while the domain size is 1×1. Thus, the Reynolds number will depend on the value of viscosity. We will simulate two cases. To achieve a Reynolds number of 1000, the viscosity is set to 0.001, while to achieve a Reynolds number of 3200, the viscosity is set to 0.0003125.



(a) $Re = 1000$ (b) $Re = 3200$

Figure 12.1: Steady state of Lid-driven cavity

Notice how we introduced a third vortex in the top left corner by increasing the Reynolds number.

Each simulation took approximately 4m and 25s to converge, with a tolerance of 1e-4.

12.2 Pipe

It is also interesting to simulate the flow of a fluid through a pipe. The domain will have walls on the north and south faces, while the fluid will flow from west to east. We have a uniform density of 1, the grid size is 400×100, and the domain size is 4×1. The domain is shown below in Figure 12.2.

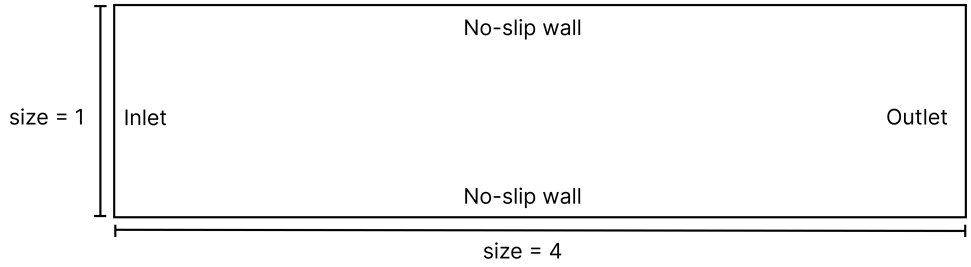


Figure 12.2: Pipe domain

The flow across the inlet will be uniform with a velocity of 1. As for viscosity, we set a value of 0.005, yielding a Reynolds number of 200.

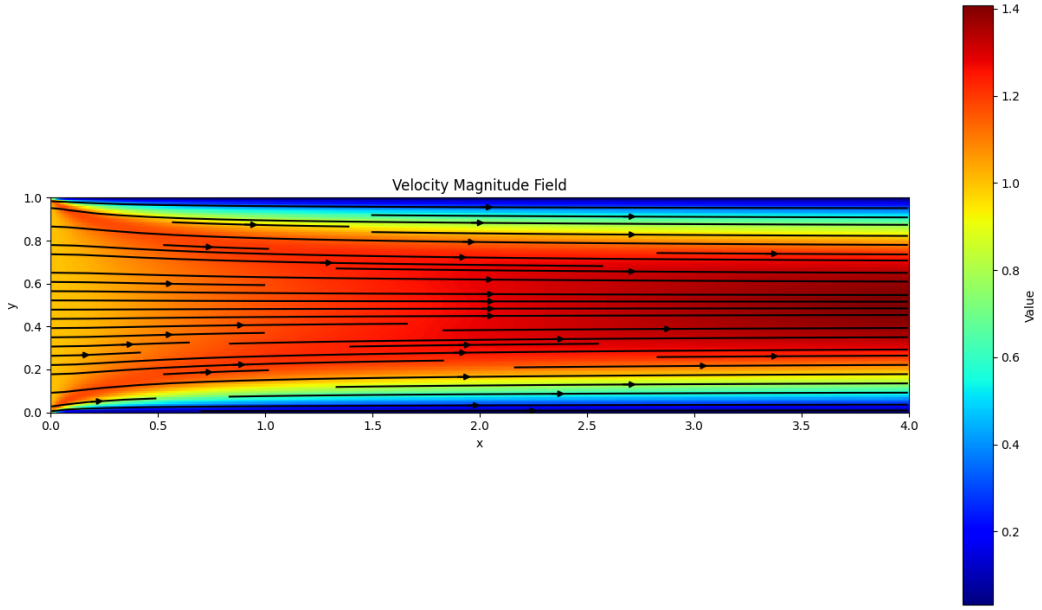


Figure 12.3: Steady state of pipe

Notice how the velocity profile turns from uniform in the inlet to parabolic in the outlet. Additionally, looking at the outlet, the velocity magnitude in the center appears to be 1.4 compared to the input velocity 1. This is a direct result of the continuity equation combined with the no-slip walls on the south and north sides. As the flow develops, since the velocity is 0 at the walls, the velocity loss must be compensated for in order to satisfy the continuity equation. In other words, since the velocity is lower than 1 at some points, it must be higher than 1 at others.

The simulation took approximately 9m 13s to converge, with a tolerance of 1e-4.

12.3 Backward-facing step

A variation of the pipe problem is the backward-facing step. The domain is shown below in Figure 12.4.

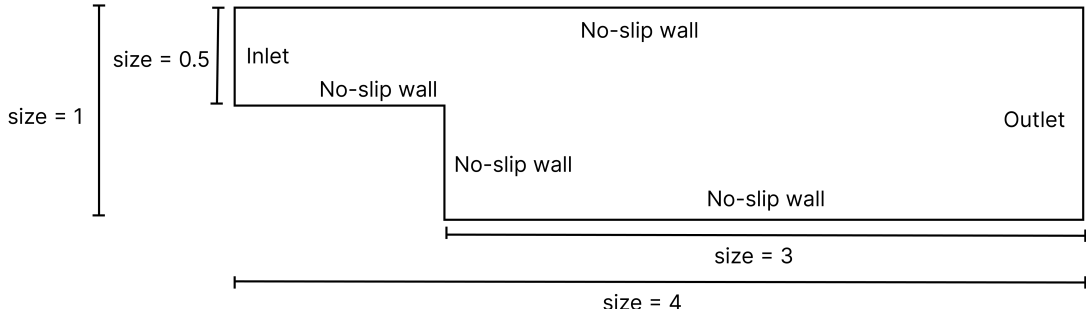


Figure 12.4: Backward-facing step domain

The fluid will flow from west to east. We have a uniform density of 1, the grid size is 400×100, and the domain size is 4×1. Notice that a small area on the south-west is not active and is cut out of the simulation. The flow across the inlet will be uniform with a velocity of 1, while the viscosity is set to a value of 0.0075.

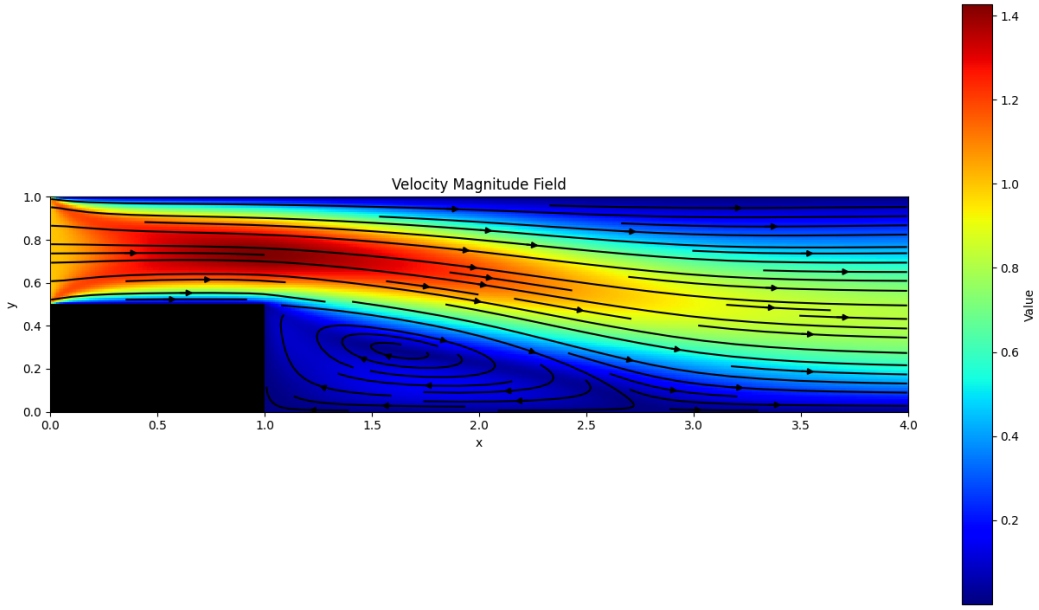


Figure 12.5: Steady state of pipe

Notice how the flow of the fluid in the pipe from $x = 0$ to $x = 1$ causes a vortex to form below, the moment it merges with the main tank. Also, notice that towards the output, the same parabolic profile is created. Only this time, its maximum velocity is 0.7, half of the maximum velocity of the first pipe, which was 1.4, since the pipe has double the diameter and thus mass has to be conserved.

The simulation took approximately 13m 36s to converge, with a tolerance of 1e-4.

12.4 Pipe with obstacles

We can make a cool variation of the pipe problem by introducing some obstacles inside the pipe domain, as shown below in Figure 12.6.

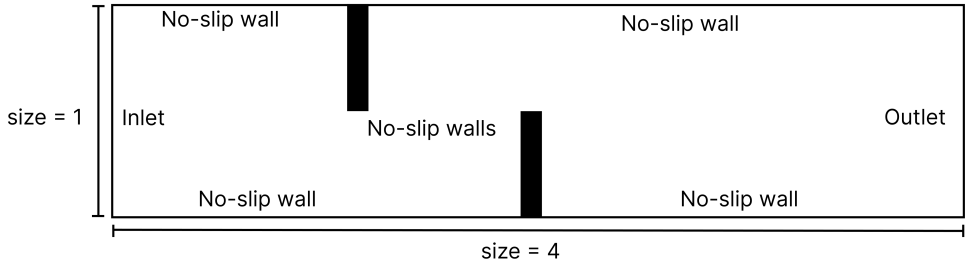


Figure 12.6: Pipe with obstacles domain

Once again, the fluid flows from west to east. We have a uniform density of 1, a grid size of 400×100, and a domain size of 4×1. The flow across the inlet will be uniform with a velocity of 1, while the viscosity is set to 0.025.

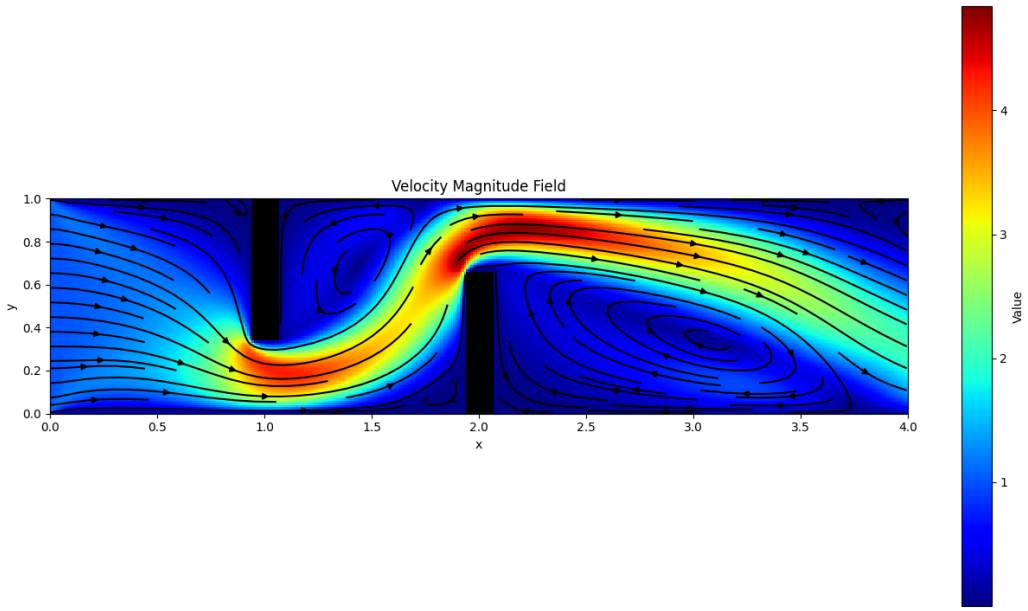


Figure 12.7: Steady state of pipe with obstacles

Notice how the fluid goes around the obstacles. In order to conserve mass (input volume should be equal to output volume), the narrow passages have a higher velocity. Also, notice the two vortices that are being formed.

The simulation took approximately 11m 44s to converge, with a tolerance of 1e-4.

12.5 Von Karman

Perhaps one of the most interesting effects we can simulate in CFD is the Von Karman effect. It is especially famous in meteorology [44]. Its domain is shown below in Figure 12.8.

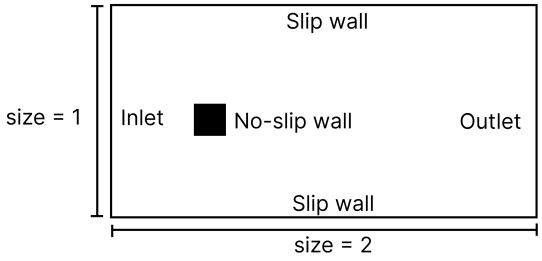


Figure 12.8: Von Karman domain

The fluid will flow from west to east, moving around the small obstacle. We have a uniform density of 1, the grid size is 300×150, and the domain size is 2×1. Notice the small rectangle inside the domain. It is an obstacle that is traditionally a circle. However, since we are using an orthogonal Cartesian grid, using a rectangle avoids the approximations we would have to make, without sacrificing the effect itself. The flow across the inlet will be uniform with a velocity of 1, while the viscosity is set to a value of 0.0005. Assuming the length of the rectangle to be $1/10$ of the domain height results in a Reynolds number of 200.

Note that this effect cannot be simulated in a steady state. It is periodic by nature, so we will conduct an unsteady-state simulation. We will calculate 1000 timesteps with $dt = 0.01$, resulting in a total simulation time of $t = 10$.

Injecting some dye in a small region in the inlet can showcase the effect in greater detail.

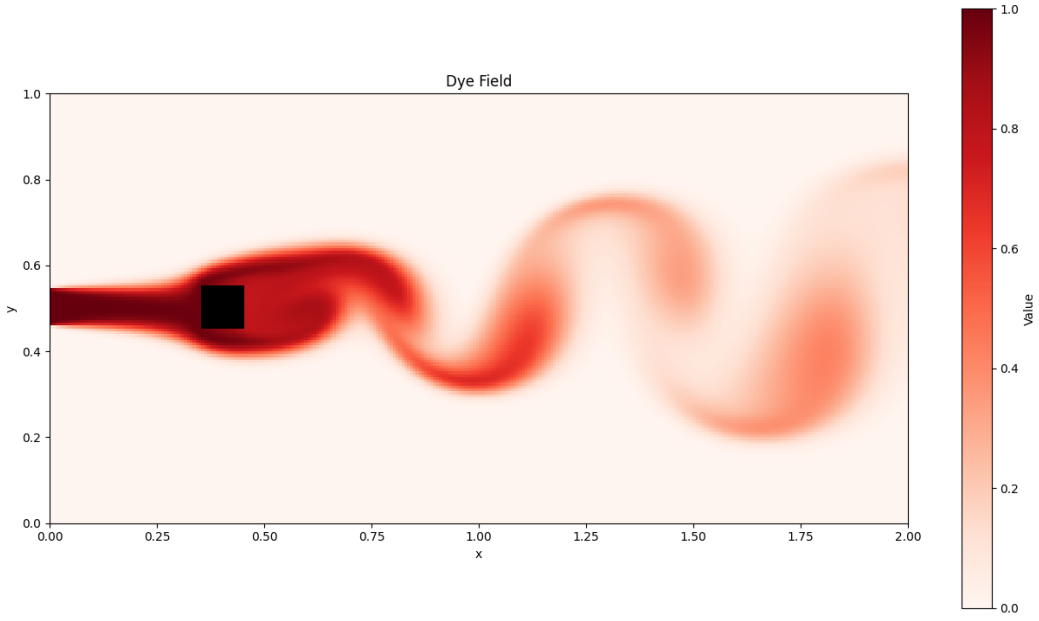


Figure 12.9: Unsteady state of Von Karman at t = 10

It is important to realize that the periodic behavior of this effect will keep oscillating and will in fact never stop.

The simulation took approximately 2h 30m to complete, with a tolerance of 1e-4.

12.6 Kelvin-Helmholtz

Another interesting phenomenon we can simulate in CFD is the Kelvin-Helmholtz. It is also very famous in meteorology [50]. Its domain is shown below in 12.10.

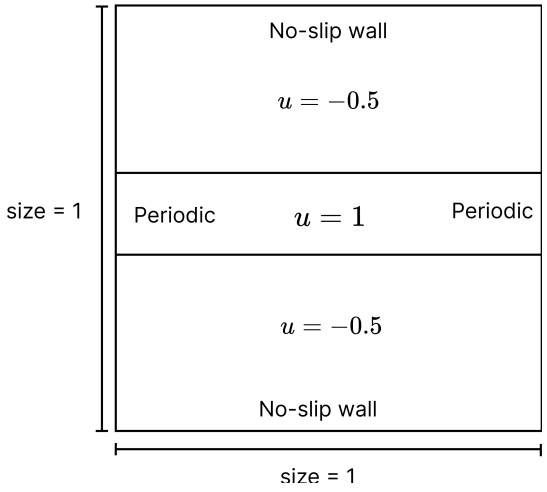


Figure 12.10: Kelvin-Helmholtz domain

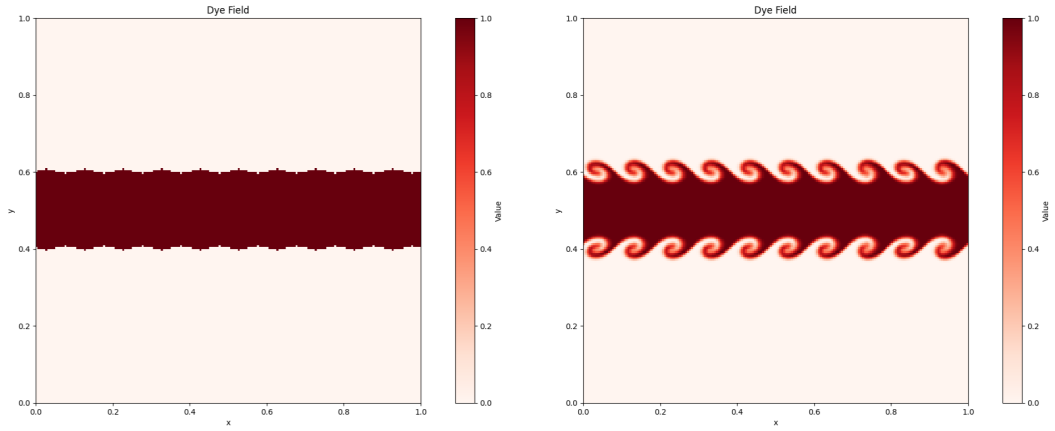
The density is 1, while the viscosity is 0.0001. The grid size is 200×200, and the domain size is 1×1.

As for the initial velocity field, we define three regions. As seen in 12.10, the domain is split into three areas: top, middle, and bottom. The top and bottom regions both have $u = -0.5$, while the middle region has $u = 1$. So, initially, the fluid in the middle area goes from left to right, while in the top and bottom regions, it goes from right to left. Velocity v is set to 0 throughout the domain. We dye the middle area fluid to trace its flow.

The west and right sides are connected through a periodic boundary condition, so fluid going through one side appears on the other.

Again, this effect is characterized by its instability and lack of a steady state. Therefore, we will conduct an unsteady-state simulation. We will calculate 1000 timesteps with $dt = 0.00025$, resulting in a total simulation time of $t = 0.25$.

Additionally, we introduce a minor sinusoidal perturbation as seen below in 12.11, in order to spark the instability.



(a) $t = 0$ (b) $t = 0.1375$

Figure 12.11: Unsteady state of Kelvin-Helmholtz

Soon, the simulation spirals out of control, leading to instability.

The simulation took approximately 23m 44s to complete, with a tolerance of 1e-4.

13. CONCLUSIONS AND FUTURE WORK

Hopefully, by now, the user should be confident in attempting to create a basic CFD software package from scratch. This is not an easy feat by any means, but it can very effectively deepen the understanding of CFD.

If the user has already built a CFD solver, there is a wide variety of features that are not covered in this guide that they could expand upon. Some of these include:

- Parallelizing code execution, either on the CPU or by utilizing the GPU. This can offer an immense speed-up in computational time.
- Simulate real colors. This could be achieved by solving four dye equations for every channel of RGBA, instead of just one.
- Add time-dependent boundary conditions. The need for those might appear, for example, in cases where the inlet stops injecting fluid into the domain (i.e., a faucet).
- Implement different variations of the SIMPLE algorithm. These include SIMPLER and SIMPLEC, which can offer quicker convergence and increased stability.
- Add a compressible fluid solver.
- Add multiphase flow support. This means supporting two or more fluids with different properties inside the same domain.
- Add convergence monitors.

BIBLIOGRAPHY

[1] Computational fluid dynamics. https://en.wikipedia.org/wiki/Computational_fluid_dynamics.

[2] Courant–friedrichs–lewy condition. https://en.wikipedia.org/wiki/Courant%E2%80%93Friedrichs%E2%80%93Lewy_condition.

[3] Derivation of the navier–stokes equations. https://en.wikipedia.org/wiki/Derivation_of_the_Navier-Stokes_equations.

[4] Dimensionless quantity. https://en.wikipedia.org/wiki/Dimensionless_quantity.

[5] Divergence. https://en.wikipedia.org/wiki/Divergence.

[6] Finite volume method. https://en.wikipedia.org/wiki/Finite_volume_method.

[7] Gauss-seidel method. https://math.libretexts.org/Bookshelves/Linear_Algebra/Introduction_to_Matrix_Algebra_(Kaw)/01%3A_Chapters/1.08%3A_Gauss-Seidel_Method.

[8] Gauss–seidel method. https://en.wikipedia.org/wiki/Gauss%E2%80%93Seidel_method.

[9] Laminar and turbulent flow. https://www.geeksforgeeks.org/laminar-and-turbulent-flow/.

[10] Numerical analysis. https://en.wikipedia.org/wiki/Numerical_analysis.

[11] Reynolds number. https://en.wikipedia.org/wiki/Reynolds_number.

[12] Scarborough criterion. https://en.wikipedia.org/wiki/Scarborough_criterion.

[13] System of linear equations. https://en.wikipedia.org/wiki/System_of_linear_equations.

[14] Thermal diffusivity. https://en.wikipedia.org/wiki/Thermal_diffusivity.

[15] Upwind differencing scheme for convection. https://en.wikipedia.org/wiki/Upwind_differencing_scheme_for_convection.

[16] Viscosity. https://en.wikipedia.org/wiki/Viscosity.

[17] Velocity under-relaxation in simple type methods. 2005. https://www.cfd-online.com/Forums/main/9398-velocity-under-relaxation-simple-type-methods.html.

[18] Monitoring residuals. 2009. https://www.afs.enea.it/project/neptunius/docs/fluent/html/ug/node812.htm.

[19] Approximation schemes for convective term - structured grids - common. 2016. https://www.cfd-online.com/Wiki/Approximation_Schemes_for_convective_term_-_structured_grids_-_Common.

[20] Stopping rules for jacobi/gauss-seidel iteration. 2017. https://math.stackexchange.com/questions/2425503/stopping-rules-for-jacobi-gauss-seidel-iteration.

[21] Why is there an under-relaxation factor for density and body forces? 2018. https://www.cfd-online.com/Forums/fluent/210081-why-there-under-relaxation-factor-density-body-forces.html.

[22] Problematic lid-driven cavity using simple on collocated grid. 2020. https://www.cfd-online.com/Forums/main/227393-problematic-lid-driven-cavity-using-simple-collocated-grid.html.

[23] What are the navier-stokes equations? 2023. https://www.simscale.com/docs/simwiki/numerics-background/what-are-the-navier-stokes-equations/.

[24] Fluid Mechanics 101. The finite volume method in cfd. 2019. https://www.youtube.com/watch?v=E9_kyXjtRHc.

[25] Fluid Mechanics 101. Residuals and relaxation in cfd. 2021. https://www.youtube.com/playlist?list=PLnJ8lIgfDbkon37Zbbh4wE_rp2PtX7aiR.

[26] 2BrokeScientists. Computational fluid dynamics | cfd basics. 2019. https://www.youtube.com/watch?v=hXG1uCiIe-U.

[27] Tanmay Agrawal. An introduction to cfd using matlab. 2020. https://www.youtube.com/playlist?list=PLKSR9A4mJH5rvt_Lf007xbmJISRWA0xYS.

[28] Tanmay Agrawal. Simulating fluid flows using python. 2021. https://www.youtube.com/playlist?list=PLKSR9A4mJH5o_fRTxgVciOg7HjZMnHZ1x.

[29] Tanmay Agrawal. Mechanics of fluids 23 playlist. 2023. https://www.youtube.com/playlist?list=PLKSR9A4mJH5pt6TzQ9r-HCfOFeQcq6oj-.

[30] C. W. Richards C. S. Ierotheou and M. Cross. Vectorization of the simple solution procedure for cfd problems-part i: A basic assessment. 1989. https://doi.org/10.1016/0307-904X(89)90062-0.

[31] Guilherme Caminha. The cfl condition and how to choose your timestep size. 2024. https://www.simscale.com/blog/cfl-condition/.

[32] Aerodynamic CFD. Finite difference, finite volume, and finite element methods. 2017. https://www.youtube.com/watch?v=prql73vq9sk.

[33] Comsol. Understanding the fully coupled vs. segregated approach and direct vs. iterative linear solvers. https://www.comsol.com/support/knowledgebase/1258#:~:text=Direct%20versus%20Iterative%20Linear%20System%20Solvers&text=They%20have%20the%20drawback%20of,slowly%20with%20increasing%20model%20size.

[34] Efficient Engineer. Fluid mechanics. 2020. https://www.youtube.com/playlist?list=PLEYqyyrm-hQ09B9JWzypjjTMAITgUItVh.

[35] Joel H. Ferziger. Computational methods for fluid dynamics. 1996.

[36] Joseph Feser. Intro to the finite volume method (the 1hr version!). 2023. https://www.youtube.com/playlist?list=PLu90K6-O5kAw45qBYSZOdpyBuo32qb-CA.

[37] Amit Gupta. Fvm lecture 4: Unsteady heat conduction. 2020. https://www.youtube.com/watch?v=_YlSUmMBpVM.

[38] NPTEL-NOC IITM. Computational fluid dynamics using finite volume method. 2022. https://www.youtube.com/playlist?list=PLyqSpQzTE6M_xbAkneOW2fNDFPWUQLAEz.

[39] Ravi Kant. Computational fluid dynamics. 2021. https://www.youtube.com/playlist?list=PLfPBEyaKPpSLXtzl0FvvwBNcut20mhVTh.

[40] manchestercfd. All there is to know about different mesh types in cfd! 2016. https://www.manchestercfd.co.uk/post/all-there-is-to-know-about-different-mesh-types-in-cfd.

[41] Sandip Mazumder. Cfd lectures on the simple algorithm. 2018. https://www.youtube.com/playlist?list=PLVuuXJfoPgT4gJcBAAFPW7uMwjFKB9aqT.

[42] MIT. Solution methods - iterative techniques - lecture 6. https://ocw.mit.edu/courses/16-920j-numerical-methods-for-partial-differential-equations-sma-5212-spring-2003/2351cb5ce7f15d89fa4cb3fa17eb6f64_lec6_notes.pdf.

[43] Jousef Murad. Computational fluid dynamics playlist. 2020. https://www.youtube.com/playlist?list=PL-K7XxqaUNye0co-PFb-YJLfrFpvbYU33.

[44] Tom Niziol. Whirls, curls, and little swirls: The science behind von karman vortices. 2025. https://www.wunderground.com/cat6/Whirls-Curls-and-Little-Swirls-Science-Behind-Von-Karman-Vortices.

[45] Suhas Patankar. Numerical heat transfer and fluid flow. 1980.

[46] Physics and Science. What is diffusion. 2024. https://www.youtube.com/watch?v=gWkFVSpYcj8.

[47] Hashwanth R. Cfd using matlab - solving the steady and unsteady 2d heat conduction problem. 2022. https://skill-lync.com/student-projects/week-5-1-mid-term-project-solving-the-steady-and-unsteady-2d-heat-conduction-prob

[48] C. M. Rhie and W. L. Chow. Numerical study of the turbulent flow past an airfoil with trailing edge separation. 1983. https://doi.org/10.2514/3.8284.

[49] Snigdha Sarkar. What is computational fluid dynamics (cfd) and why you need it. 2024. https://ketiv.com/what-is-computational-fluid-dynamics-and-why-you-need-it/.

[50] skybrary. Kelvin-helmholtz waves. https://skybrary.aero/articles/kelvin-helmholtz-waves.

[51] StudySession. Direct vs iterative numerical methods | numerical methods. 2020. https://www.youtube.com/watch?v=CHUlnCSkg5k.

[52] R Greif T Hayase, J.A.C Humphrey. A consistently formulated quick scheme for fast and stable convergence using finite-volume iterative calculation procedures. 1990. https://doi.org/10.1016/0021-9991(92)90177-Z.

[53] Ertan Taskin. Modeling of numerical dye washout with scalar transport equation in cfx: A case study of blood flow through aortic arch. 2024. https://blog.ozeninc.com/resources/modeling-of-numerical-dye-washout-with-scalar-transport-equation-in-cfx-a-case-st

[54] Gustav Tschirschnitz and Pierre Sabrowski. Computational fluid dynamics methods explained. 2020. https://www.divecae.com/resources/cfd-methods.

[55] H. K. Versteeg and Weeratunge Malalasekera. An introduction to computational fluid dynamics the finite volume method. 1995.

[56] Nektarios Vlahakis. ΔΥΝΑΜΙΚΗ ΤΩΝ ΡΕΥΣΤΩΝ. 2019.