SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY - INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics: Games Engineering

Differentiable Finite Volume Method

Anil Celik Maral

SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY - INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics: Games Engineering

Differentiable Finite Volume Method

Differenzierbare Finite Volumen Methode

Author:	Anil Celik Maral
Supervisor:	Prof. Dr. Nils Thuerey
Advisor:	Philipp Holl, M.Sc.
Submission Date:	25.07.2024

I confirm that this master's thesis in informatics: games engineering is my own work and I have documented all sources and material used.

Munich, 25.07.2024 Anil Celik Maral

Munich, 25. 07. 2024

Acknowledgments

Firstly, I would like thank to Philipp Holl, M.Sc. for supervising me during my thesis, developing Φ_{Flow} and giving me the opportunity to work on this topic. Additionally, I would like to thank Dr. Aidan Wimshurst for providing informative and free CFD lectures under the Fluid Mechanics 101 channel on Youtube and OpenFOAM for putting a lot of free CFD resources. I would also like to thank my family and friends who have supported me throughout this process.

Abstract

In this thesis, we investigate the numerical solutions to the incompressible Navier-Stokes equations using the finite volume method (FVM). The primary challenges addressed are the discretization process of the incompressible Navier-Stokes equations and pressure-velocity coupling problem inherent in the incompressible formulation of the Navier-Stokes equations [1, Pages 165-167]. The Projection Method introduced by Chorin [2] serves as a foundational approach to decouple these variables, allowing for stable and accurate simulations of fluid flow.

FVM utilizes conservation laws and fluxes across the control volume surfaces to approximate the integrals over the control volumes. These form the bases of the dicretizations done to be able to numerically solve the given PDEs for a variety of CFD problems. The discretization of the Navier-Stokes equations using both explicit and implicit methods is examined, with particular attention given to the backward Euler method for its stability advantages [3]. The concepts of advection and convection are explained, providing clarity on their roles within fluid dynamics simulations.

The projection method involves solving a Poisson equation for the pressure field, derived to ensure the incompressibility constraint is satisfied at each time step. We explore the derivation and implementation of the Poisson equation for pressure in the Projection Method Poisson, highlighting its critical role in maintaining stability.

Furthermore, we discuss the Courant-Friedrichs-Lewy (CFL) condition and its implications for time step selection, ensuring numerical stability.

A key focus of the thesis is the finite volume method's ability to handle complex geometries and non-uniform meshes, which is essential for accurately simulating real-world fluid dynamics problems. The incorporation of obstacles and their influence on the mesh and solution stability is also explored.

By implementing the finite volume method using these methods in a computational framework such as Φ_{Flow} [4], we provide practical examples and validate the numerical solutions against benchmark problems where it is possible. Additionally, solutions generated by the finite volume method are compared, where it is possible to do so, to their counterparts generated by the finite difference method using the same initial conditions. The results demonstrate the efficacy of the projection method and the finite volume method in solving the incompressible Navier-Stokes equations, offering insights into their application for various conditions.

Kurzfassung

In dieser Dissertation untersuchen wir die numerischen Lösungen der inkompressiblen Navier-Stokes-Gleichungen unter Verwendung der Finite-Volumen-Methode (FVM). Die primären Herausforderungen, die behandelt werden, sind der Diskretisierungsprozess der inkompressiblen Navier-Stokes-Gleichungen und das Druck-Geschwindigkeit Kopplungsproblem, das in der inkompressiblen Formulierung der Navier-Stokes-Gleichungen [1, Seiten 165-167] inhärent ist. Die von Chorin eingeführte Projektion-Methode [2] dient als grundlegender Ansatz zur Entkopplung dieser Variablen, wodurch stabile und genaue Simulationen des Fluidstroms ermöglicht werden.

Die FVM nutzt Erhaltungsgesetze und Flüsse über die Flächen der Kontrollvolumina, um die Integrale über die Kontrollvolumina zu approximieren. Diese bilden die Grundlage für die Diskretisierungen, um die gegebenen PDEs für eine Vielzahl von CFD-Problemen numerisch zu lösen. Die Diskretisierung der Navier-Stokes-Gleichungen unter Verwendung sowohl expliziter als auch impliziter Methoden wird untersucht, wobei besonderes Augenmerk auf das Backward-Euler-Verfahren aufgrund seiner Stabilitätsvorteile gelegt wird [3]. Die Konzepte von Advektion und Konvektion werden erklärt, um ihre Rollen innerhalb von Strömungsdynamiksimulationen klarzustellen.

Die Projektion-Methode beinhaltet die Lösung einer Poisson-Gleichung für das Druckfeld, die abgeleitet wird, um sicherzustellen, dass die Inkompressibilitätsbedingung bei jedem Zeitschritt erfüllt ist. Wir untersuchen die Ableitung und Implementierung der Poisson-Gleichung für den Druck in der Projektion-Methode und heben ihre entscheidende Rolle für die Stabilität hervor.

Des Weiteren diskutieren wir die Courant-Friedrichs-Lewy (CFL)-Bedingung und ihre Auswirkungen auf die Auswahl des Zeitschritts, um die numerische Stabilität zu gewährleisten.

Ein Schwerpunkt der Dissertation liegt auf der Fähigkeit der Finite-Volumen-Methode, komplexe Geometrien und nicht-uniforme Netze zu handhaben, was für die genaue Simulation realer Fluiddynamikprobleme entscheidend ist. Die Integration von Hindernissen und deren Einfluss auf das Netz und die Lösungstabilität wird ebenfalls untersucht.

Durch die Implementierung der Finite-Volumen-Methode unter Verwendung dieser Methoden in einem rechnerischen Framework wie Φ_{Flow} [4] bieten wir praktische Beispiele und validieren die numerischen Lösungen gegen Benchmark-Probleme, wo es möglich ist. Darüber hinaus werden Lösungen, die mit der Finite-Volumen-Methode erzeugt wurden, mit ihren Gegenstücken verglichen, wo es möglich ist, die mit der Finite-Differenzen-Methode unter Verwendung der gleichen Anfangsbedingungen erzeugt wurden. Die Ergebnisse zeigen die Wirksamkeit der Projektion-Methode und der Finite-Volumen-Methode bei der Lösung der inkompressiblen Navier-Stokes-Gleichungen und bieten Einblicke in ihre Anwendung unter verschiedenen Bedingungen.

Contents

Acknowledgments **iii**

Abstract **iv**

Kurzfassung **v**

1. Introduction **1**

2. Main Concepts of the Finite Volume Method **3**
2.1. Problem and Solution Overview 3
2.2. Unstructured Meshes Versus Structured Meshes 5
2.3. Components of a Cell 7
2.4. Linearity of Flow Variables in FVM 8
2.5. PDE Discretization 9
2.6. Construction of the Matrices 11
2.7. Concept of Discretization 13

3. Discretization of the Incompressible Flow Navier-Stokes Momentum Equation **15**
3.1. Incompressible Flow 15
3.2. Time Derivative Term Discretization 16
3.2.1. Forward Euler Method 17
3.2.2. Backward Euler Method 18
3.3. Advection Term Discretization 19
3.4. Diffusion Term Discretization 23
3.4.1. Green Gauss Method 29
3.5. Pressure-Velocity Coupling 30

4. Results **32**
4.1. Divergence Evolution Simulation 32
4.2. Implicit FVM Simulation 33
4.3. Implicit FVM Simulation with a Sinusoidal Unstructured Mesh 39
4.4. Implicit FVM Inflow Simulation 46
4.5. Implicit FVM Simulation Using a Large Mesh 53
4.6. Implicit FVM Simulation with Obstacles 63

5. Conclusion **70**

A. Appendix **71**
A.1. Gauss's Divergence Theorem 71
A.2. Linear Interpolation 71
A.3. Divergence of a Point 72

Bibliography **73**

1. Introduction

This thesis explores the application and effectiveness of the Finite Volume Method (FVM) for solving the incompressible Navier-Stokes equations, with a particular focus on addressing the challenges inherent in the discretization process and the Pressure-Velocity Coupling problem. The FVM is renowned for its robustness in handling complex geometries and its ability to ensure conservation properties, making it a powerful tool in fluid dynamics simulations [5].

The Finite Volume Method discretizes partial differential equations (PDEs) by partitioning the computational domain into small control volumes [1]. The key idea is to integrate the governing equations over each control volume, ensuring the conservation of fluxes across the boundaries of these volumes [1]. This approach is particularly well-suited for problems with complex geometries and where conservation laws are critical in fluid dynamics such as heat transfer [5]. These properties make FVM robust for a wide range of applications [5].

In comparison, Finite Difference Methods (FDM) approximate derivatives by replacing them with finite difference quotients on a grid [6]. The computational domain is divided into a grid, and the PDEs are solved at discrete grid points [6]. Common finite difference schemes include explicit, implicit, and Crank-Nicolson methods [6]. FDM is conceptually straightforward and widely used in computational physics and engineering due to its simplicity and ease of implementation [6]. However, it can struggle with complex geometries and ensuring conservation properties [7].

This thesis delves deeply into the details of the FVM, including the discretization of advection and diffusion terms. The concepts of non-orthogonality and why its needed for the diffusion term is explained. For the advection term, why differencing schemes such as upwind scheme are needed is explained. The discretization process for each term is done and described step by step in detail. Additionally, both explicit and implicit methods are examined for the discretization of the Navier-Stokes equations while putting an emphasis on the backward Euler method due to it stability advantages [3]. Each term in the Navier-Stokes equations are explained and handled specially.

A significant challenge in solving the incompressible Navier-Stokes equations is the pressure-velocity coupling problem [1, Pages 165-167]. The pressure field does not have its own evolution equation but is implicitly determined by the incompressibility condition [2]. The projection method, introduced by Chorin [2], decouples the velocity and pressure fields by solving a Poisson equation for the pressure, derived to ensure the velocity field remains divergence-free. The thesis goes over the derivation of the Poisson equation for pressure and explains how it is derived and how Chorin's algorithm makes it possible to satisfy the incompressibility condition.

The Courant-Friedrichs-Lewy (CFL) condition is described regarding its implications for time step selection to ensure numerical stability. Ways to ensure that this condition is satisfied

throughout the simulations are described such as calculating the Courant number for an arbitrary mesh and velocity field **U**.

Since one of the biggest advantages of FVM is its ability to utilize irregular / unstructured polyhedral meshes, it allows for accurate simulations of real-world fluid dynamic problems with complex meshes where conservation of properties are especially important. Therefore, unstructured meshes are described and compared with structured meshes.

The implementation of the FVM within the Φ_{Flow}[4] computational framework includes practical examples and validation against benchmark problems where it is possible. The results are compared, once again where it is possible to do so, to those obtained using the Finite Difference Method under identical initial conditions, highlighting the strengths and weaknesses of each approach.

In conclusion, this thesis aims to advance the understanding and application of the Finite Volume Method for fluid dynamics.

2. Main Concepts of the Finite Volume Method

2.1. Problem and Solution Overview

To solve partial differential equations found in fluid dynamics, we can utilize different techniques. Each method has its strengths and weaknesses depending on a given problem and its specifications such as the scale of the problem, the required accuracy, etc [1, pages 59-61].

The FVM is one of such techniques. In FVM, we discretize and create models of systems that we want to simulate by utilizing control volumes to create models of such physical systems. Control volumes are defined regions of typically stationary space surrounded by a surface through which fluid enters and exits [1, pages 59-61].

To capture the physical realities accurately, we need to conserve properties such as mass, momentum, and energy. By utilizing conservation equations and equating the fluxes from inflow and outflow at surfaces and boundaries and considering any sources present inside the domain, FVM tries to achieve this. Specificially, these equations are applied to each control volume [1, pages 59-61].

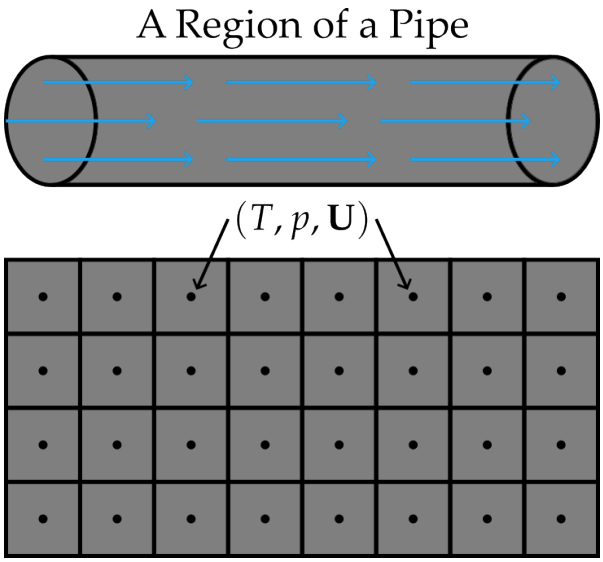

Figure 2.1.: We try to solve a problem such as how a given fluid behaves while going through a pipe. Using FVM, we need to solve for the velocity and the pressure fields to approximate the behavior of the fluid.

To begin any CFD simulation, we need to define the problem and solution domains. The

problem domain can specify a region of space such as a pipe as seen in Figure 2.1. Such problem domains are continuous; therefore, we need to discretize the problem domain itself as well as the governing partial differential equations to be able to solve and simulate them using computers. To do so, we start by creating a mesh to map our problem domain to the solution domain. A mesh consists of cells that map a volume or an area to discrete chunks. An example of such a mapping is shown in Figure 2.1, where we map a pipe to a 2D mesh using square cells. Each cell stores flow variables such as the T (temperature), p (pressure), $\mathbf{U}$ (velocity), etc. at its cell centroid [1, pages 61-63].

Mesh creation can be based on arbitrary requirements and it can be automatic or manual. There are many software programs available to do both. Therefore, mesh generation and its intricacies and details won't be discussed in depth here. Additionally, although later during the implementation we will be exclusively working in 2D, the underlying concepts and equations remain the same and can be easily applied to 3D problems.

In general, for FVM, meshes can contain cells of any irregular polyhedral shape, where each cell can have an unlimited but finite number of faces and each face can similarly have an unlimited but finite number of edges. Except for the boundary faces, which are the outermost faces of the solution domain, any face of a given cell is common to or shared by its neighboring cells, i.e., cells are contiguous and cells can be aligned freely without any restriction regarding the coordinate axes [1, pages 61-63].

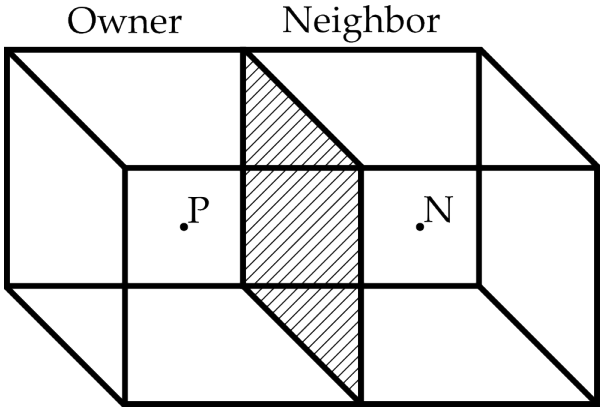

Figure 2.2.: A Shared Face Showing the Owner and Neighbor Cells

Figure 2.2 shows two cells within a mesh with the face that is shared between the two highlighted. The mesh is constructed using points / vertices, faces and polygons. Each face consists of certain vertices and each polygon consists of certain faces. A face is only shared by two cells which are the owner cell and the neighbor cell. The exception to this are the boundary faces which only belong to the owner cells. Boundary faces are tracked and grouped to be able to apply specific boundary conditions when running a simulation. Grouped boundaries are usually called patches. For example, we might track the faces that make up the top, bottom, right and left boundaries. There are different strategies to keep track of the cells and their connectivities in an unstructured mesh such as by assigning a unique index to each cell. Storing the indices of the owner and neighbor cell of each face

allows the connectivity between cells to be represented [1, pages 61-63].

In essence, we need a list of vertices, a list of faces defined by a sequence of vertices / vertex indices, a list of owner cell indices as well as a list of neighbor cell indices associated with faces and lastly a grouping of boundary faces to define a mesh [1, pages 61-63].

2.2. Unstructured Meshes Versus Structured Meshes

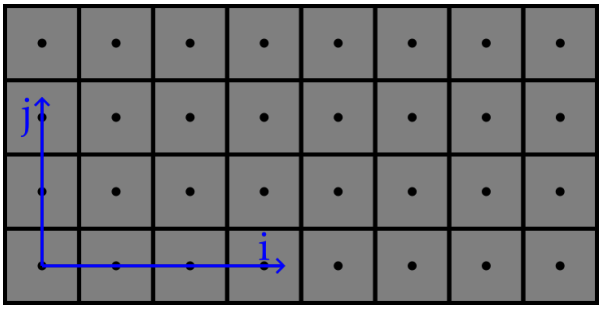

Figure 2.3.: Structured Mesh

Cells of the same shape with a regular pattern, such as when all cells are quadrilaterals or cuboids, are utilized by structured meshes. Since in a structured mesh the cells have a regular pattern, this allows the identification of each cell by an indexing scheme such as an i, j, k index numbering scheme [8, page 7]. Figure 2.3 shows such a structured mesh.

Although most structured meshes consist of rectangles and hexahedra (considering 2D and 3D respectively), the main feature of a structured mesh is its regular pattern and not the shape of the cells that make up the mesh (cells of any arbitrary shape can be used as long as all cells have the same shape). The regular pattern allows us to use the indexing / numbering scheme and that is what separates a structured mesh from an unstructured mesh [8, page 7].

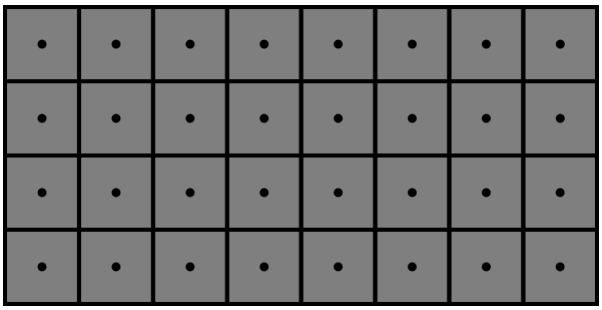

Figure 2.4.: Unstructured Mesh with a Simple Geometry

Despite its ease of use, the restrictive nature of structured meshes result in most of the modern CFD codes using unstructured meshes instead. The main advantage of an unstructured mesh is that cells of any arbitrary shape can be used simultaneously in a given mesh. For example, rectangles, triangles or pentagons can be used in the same mesh. Similarly,

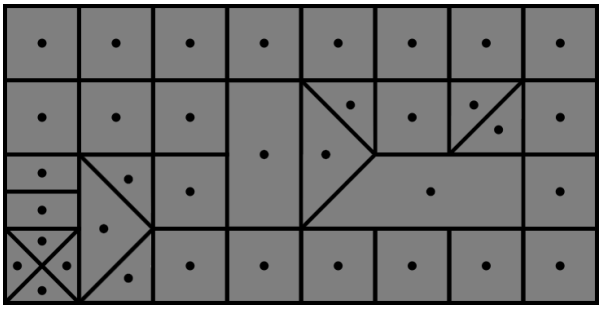

Figure 2.5.: Unstructured Mesh with a Complex Geometry

for 3D, triangular pyramids, hexahedra or cuboids can be used in the same mesh. This means that for a given fluid dynamics problem, the user can choose and utilize the most suitable mesh structure depending on arbitrary criteria such as the desired accuracy etc. [8, page 7].

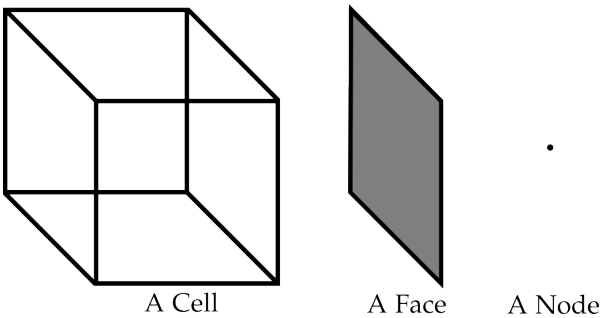

Figure 2.6.: Cells, faces and nodes are treated as separate entities when unstructured meshes are utilized.

In unstructured meshes, cells, faces and nodes are treated as separate entities. As a result, the collection of cells, faces and nodes in an arbitrary manner make up an unstructured mesh. In contrast, cells are defined by their i, j, k indexing and they are treated as complete units in a structured mesh. Although the mesh in Figure 2.4 has the same pattern as the mesh in Figure 2.3, due to the lack of the capability to index, the underlying structure is an unstructured mesh. In comparison, the mesh in Figure 2.5 can only be constructured by utilizing an unstructured mesh due to the irregular pattern which makes it impossible to use indexing [8, page 7].

Even though the underlying formulas and concepts remain the same, the underlying code changes substantialy to utilize unstructured meshes in a CFD code. In comparison to a CFD code that utilizes structured meshes, CFD codes that utilize unstructured meshed are more complex [8, page 7].

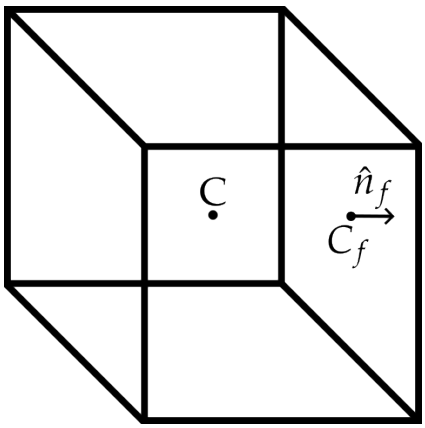

Figure 2.7.: A 3D Cell with the Cell Centroid C, Face Center C_f and a Face Unit Normal Vector $\hat{n}_f$

2.3. Components of a Cell

In FVM, discretized values of fields such as velocity, pressure, temperature etc. are stored at the cell centroids C of each cell within a mesh and these values are then interpolated to the face centers C_f to be used while solving the discretized PDEs such as Navier-Stokes Equations [1, pages 63-65]. Centroid C of a cell is the center of mass for that cell and similarly the face center C_f is the center of mass for that face. These point locations can be seen in Figure 2.7.

While solving the discretized PDEs, cell volume V (for 3D) or area A (for 2D) is used. Additionally, face area A_f (for 3D) or line length L_f (for 2D) is utilized.

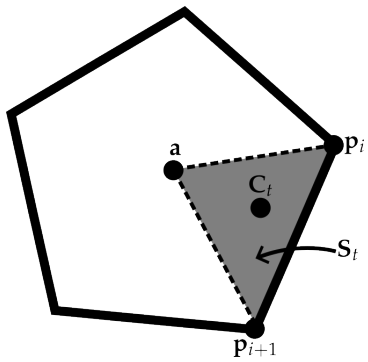

Figure 2.8.: An Irregular Shaped Face, **a** is the average of face vertices also called the apex point, $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ are the other two vertices of the triangle, $\mathbf{S}_t = \hat{\mathbf{n}}_t A_t$ i.e. the triangle area vector, and $\mathbf{C}_t$ is the triangle center.

In order to calculate the cell volume V and cell centroid C, there are multiple ways but one of the most common ways is as follows [1, pages 63-65]:

$$\mathbf{S}_t = (\mathbf{p}_{i+1} - \mathbf{p}_i) \times (\mathbf{a} - \mathbf{p}_i)/2 \tag{1}$$

$$\mathbf{C}_t = (\mathbf{p}_{i+1} + \mathbf{p}_i + \mathbf{a})/3 \tag{2}$$

Firstly, each face is decomposed into triangles by averaging the vertices of the face to calculate the apex point $\mathbf{a}$ and connecting the face vertices to this point. Then, the triangle area vector $\mathbf{S}_t$ and triangle center $\mathbf{C}_t$ are calculated per triangle using equations 1, where "$\times$" symbol stands for the cross product, and 2.

$$\mathbf{S}_f = \sum_{i=1}^{N} \mathbf{S}_t \tag{3}$$

$$\mathbf{C}_f = \frac{1}{|\mathbf{S}_f|} \sum_{i=1}^{N} |\mathbf{S}_t| \mathbf{C}_t \tag{4}$$

$\mathbf{S}_f$ is calculated by the sum of area vectors as in equation 3. The face center $\mathbf{C}_f$ is calculated by the area weighted sum of triangle centres over the N triangles, that the was face decomposed to, as in equation 4.

$$V = \int_V dV = \frac{1}{3} \int_V \nabla \cdot \mathbf{x} dV = \frac{1}{3} \int_S (d\mathbf{S} \cdot \mathbf{x}) \approx \frac{1}{3} \sum_f \mathbf{S}_f \cdot \mathbf{C}_f \tag{5}$$

The cell volume V is calculated using equation 5. In this formula, Gauss's divergence theorem is used to turn the volume integral of the divergence of position to a surface integral of $\mathbf{x}$ dot producted by the infinitesimally small surface area vector. In the formula, surface normal vector is combined with the infinitesimally small surface element as $d\mathbf{S} = \hat{\mathbf{n}} dS$ and the fact that $\nabla \cdot \mathbf{x} \equiv 3$ is used. Since our cells have a finite number of faces, the surface integral can be approximated as the summation of the product area of each face with the face center dot producted with the surface area vector. For the reasoning behind using the face center for the approximation, please check section 2.4.

$$\mathbf{C} = \frac{1}{V} \int_V \mathbf{x} dV = \frac{1}{2V} \int_V \nabla |\mathbf{x}|^2 dV = \frac{1}{2V} \int_S (\mathbf{dS} |\mathbf{x}|^2) \approx \frac{1}{2V} \sum_f \mathbf{S}_f |\mathbf{C}_f|^2 \tag{6}$$

The cell centroid / center is calculated by using a similar logic and formulation while also noting $\nabla |\mathbf{x}|^2 \equiv 2\mathbf{x}$ as in equation 6.

2.4. Linearity of Flow Variables in FVM

In FVM, all flow variables vary linearly across a cell as well as a cell face. The logic and the reasoning can be seen in Figure 2.9 and Figure 2.10. $\int_S d\mathbf{S} \cdot \mathbf{\Psi} = \int_S d\mathbf{S} \cdot \mathbf{\Psi}_P$ says that the result of the integral for integrating a flow variable across a cell is the same if we just integrate value of the flow variable at the cell centroid. $\int_S d\mathbf{S} \cdot \mathbf{\Psi} = \int_S d\mathbf{S} \cdot \mathbf{\Psi}_f$ says that the result of the integral for integrating a flow variable across a face is the same if we just integrate value of the flow variable at the face center [8, pages 9-10].

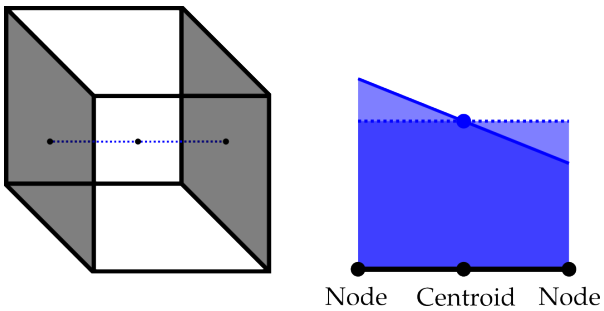

Figure 2.9.: $\int_S d\mathbf{S} \cdot \mathbf{\Psi} = \int_S d\mathbf{S} \cdot \mathbf{\Psi}_P$

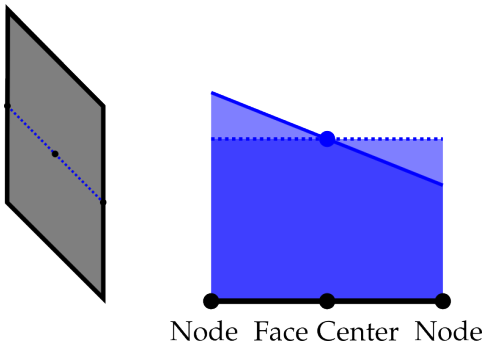

Figure 2.10.: $\int_S d\mathbf{S} \cdot \mathbf{\Psi} = \int_S d\mathbf{S} \cdot \mathbf{\Psi}_f$

2.5. PDE Discretization

For continuous fields such as velocity $\mathbf{U}$ or pressure p, discretization of associated partial differential equations turns these continuous fields into sets of linear equations for discrete fields [1, pages 65-67].

As previously mentioned, flow variables such as the velocity $\mathbf{U}$ are stored at the cell centroids of each cell within the mesh such as in Figure 2.4. These fields are usually represented by an array of values where $\mathbf{\Psi}_i$ represents the value stored at the cell centroid of the cell indexed by $i = 1, 2, 3, ..., N$ (N is the number of cells in the mesh).

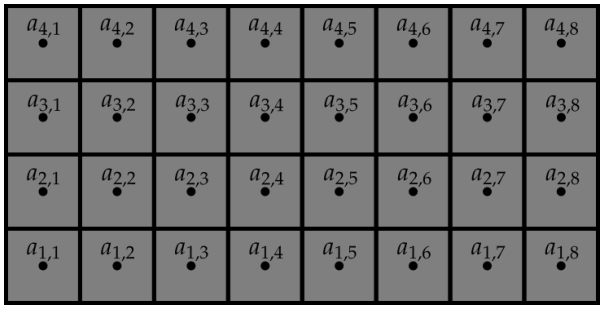

Figure 2.11.: Each index in the field corresponds to a field value stored at the cell centroid of a cell.

The process of partial differential equation discretization creates a linear equation for each cell in the mesh. For example, for cell with the index $3,4$ in Figure 2.11, the equation might have the following form for the velocity $\mathbf{U}$:

$$a_{(3,4),(3,3)}\mathbf{U_{3,3}} + a_{(3,4),(4,4)}\mathbf{U_{4,4}} + a_{(3,4),(3,5)}\mathbf{U_{3,5}} + a_{(3,4),(2,4)}\mathbf{U_{2,4}} + a_{(3,4),(3,4)}\mathbf{U_{3,4}} = b_{3,4} \tag{7}$$

In equation 7, the coefficients $a_{i,j}$ and b_i correspond to cell indices i, j. For example, in $a_{(3,4),(3,3)}\mathbf{U_{3,3}}$ $i = (3,4)$ and $j = (3,3)$. $a_{(3,4),(3,3)}$ in $a_{(3,4),(3,3)}\mathbf{U_{3,3}}$ symbolizes the coefficient of the contribution of the velocity of the cell $(3,3)$ to cell $(3,4)$. These contributions usually come when the values stored at the cell centroids of the neighboring cells affect the value at the current cell's cell centroid [1, pages 65-67]. One such example is when the advection affects the face center value of one of the faces of the current cell due to the direction of the flow. This will be later explained in detail.

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,N} \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots & a_{2,N} \\ a_{3,1} & a_{3,2} & a_{3,3} & \cdots & a_{3,N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{N,1} & a_{N,2} & a_{N,3} & \cdots & a_{N,N} \end{bmatrix} \begin{bmatrix} \Psi_1 \\ \Psi_2 \\ \Psi_3 \\ \vdots \\ \Psi_N \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_N \end{bmatrix} \tag{8}$$

The set of linear equations for all cells in a mesh for a given field $\mathbf{\Psi}$ can be written as a matrix equation as shown in equation 8 [1, pages 65-67].

In the matrix equation, each row correspond to the linear equation for the cell with index i. For example, the first row in equation 8 corresponds to the linear equation for the cell with index $1,1$. In the matrix of coefficients, diagonal entries $a_{i,i}$ are the coefficients of the respective cells for which this row corresponds to the linear equation to be solved. This means that in each row only the diagonal coefficients and the near-neighbors of the respective cell are non-zero. In a way, the matrix of coefficients kind of corresponds to the currently considered cell's connectivity. As a result, the resulting matrices are very sparse [1, pages 65-67].

To simplify the notation, the matrix equations can be represented as $[A][\Psi] = [b]$. $[A]$ stands for the matrix of coefficients, $[b]$ stands for the source coefficients (such as gravity) and $\mathbf{\Psi}$ stands for the discretized field for the given flow variable such as the velocity $\mathbf{U}$ or the pressure p [1, pages 65-67].

In CFD simulations, a set of equations for different flow variables are solved and FVM is no different. As mentioned before, in the FVM, each physical equation for each flow variable is discretized separately to form individual matrix equations corresponding to the flow variable that we want to solve for. So, instead of creating a single matrix equation to represent all of the equations for all of the flow variables, we separate each matrix equation and solve them individually [1, pages 65-67].

In separate steps, we solve for one flow variable, such as velocity $\mathbf{U}$, pressure p or temperature T etc., at a time using the segregated matrix equations. In our case, the matrix equations for the velocity $\mathbf{U}$ and pressure p will be considered in detail and solved accordingly. Each of

the flow variables that is being solved is either a vector or a tensor. For example, considering that the velocity **U** is a vector, we need to assemble a matrix equation per component such as one for the x component, y component and z component. In the end, we will have to solve four total matrix equations to simulate flow in our CFD simulations. These matrix equations will have the following forms [1, pages 65-67]:

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,N} \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots & a_{2,N} \\ a_{3,1} & a_{3,2} & a_{3,3} & \cdots & a_{3,N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{N,1} & a_{N,2} & a_{N,3} & \cdots & a_{N,N} \end{bmatrix} \begin{bmatrix} U_{x1} \\ U_{x2} \\ U_{x3} \\ \vdots \\ U_{xN} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_N \end{bmatrix} \tag{9}$$

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,N} \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots & a_{2,N} \\ a_{3,1} & a_{3,2} & a_{3,3} & \cdots & a_{3,N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{N,1} & a_{N,2} & a_{N,3} & \cdots & a_{N,N} \end{bmatrix} \begin{bmatrix} U_{y1} \\ U_{y2} \\ U_{y3} \\ \vdots \\ U_{yN} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_N \end{bmatrix} \tag{10}$$

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,N} \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots & a_{2,N} \\ a_{3,1} & a_{3,2} & a_{3,3} & \cdots & a_{3,N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{N,1} & a_{N,2} & a_{N,3} & \cdots & a_{N,N} \end{bmatrix} \begin{bmatrix} U_{z1} \\ U_{z2} \\ U_{z3} \\ \vdots \\ U_{zN} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_N \end{bmatrix} \tag{11}$$

$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,N} \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots & a_{2,N} \\ a_{3,1} & a_{3,2} & a_{3,3} & \cdots & a_{3,N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{N,1} & a_{N,2} & a_{N,3} & \cdots & a_{N,N} \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ \vdots \\ p_N \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_N \end{bmatrix} \tag{12}$$

Each matrix equation is solved using an iterative process. That means for a component like U_x, the iterative process incorporates current values of other variables such as U_y, U_z and p into the source vector b.

2.6. Construction of the Matrices

$$\frac{\partial \mathbf{\Psi}}{\partial t} + \nabla \cdot (\mathbf{U}\mathbf{\Psi}) - \nabla \cdot (\Gamma \nabla \mathbf{\Psi}) = S_{\mathbf{\Psi}} \tag{13}$$

Consider equation 13 which has a temporal, advection, diffusion and a source term $S_{\mathbf{\Psi}}$. The corresponding matrix equation for the variable $\mathbf{\Psi}$ is constructed by determining the matrix and source coefficients of the matrix of coefficients A and source vector b respectively. As previously hinted, these coefficients come from the linear equations, such as equation 7, that are constructed for each cell in the mesh. These linear equations are the result of compiling

the effects of different terms of the equation such as the diffusion, advection, temporal or source terms on a given cell i.e. coefficients are calculated for each individual term, using the discretization methods later described, and added up for a respective cell. Additionally, boundary conditions affect the cells at the domain boundary due to their effects through the advection and diffusion terms [1, pages 67-69].

$$\frac{\partial \mathbf{\Psi}}{\partial t} + \nabla \cdot (\mathbf{U}\mathbf{\Psi}) - \nabla \cdot (\Gamma \nabla \mathbf{\Psi}) = S_\Psi$$

$$\begin{bmatrix} * & * & & * \\ * & * & * & \\ & * & * & * \\ * & & * & * \end{bmatrix} \begin{bmatrix} \mathbf{\Psi} \end{bmatrix} = \begin{bmatrix} * \\ * \\ * \\ * \end{bmatrix}$$

Figure 2.12.: Term contributions to the matrix equation.

$$\mathbf{\Psi} = \mathbf{\Psi}_b$$

$$\begin{bmatrix} * & * & & * \\ * & \circledast & * & \\ & * & * & * \\ * & & * & * \end{bmatrix} \begin{bmatrix} \mathbf{\Psi} \end{bmatrix} = \begin{bmatrix} * \\ \circledast \\ * \\ * \end{bmatrix}$$

Figure 2.13.: Boundary condition's contribution to a cell at the domain boundary.

A visualization of the terms of the equations and boundary conditions affecting the coefficients and as a result the linear equations of the cells in the mesh can be seen in Figure 2.12 and Figure 2.13. Although it depends on the discretization scheme used, most terms contribute to both the matrix A's and source vector b's coefficients [1, pages 67-69].

Discretization of a term can result in explicit or implicit contributions to the matrix equation $[A][\mathbf{\Psi}] = [b]$. The contribution is implicit if the discretized term contributes to the coefficients of the matrix A and the contribution is explicit if the discretized term contributes to the coefficient of the source vector b. In an implicit contribution, the term contributes to A by treating $\mathbf{\Psi}$ as the solved field $[\mathbf{\Psi}]$ and in an explicit contribution, the term contributes to b because it uses current values of the fields [1, pages 67-69].

When solving for the field $\mathbf{\Psi}$, derivatives without $\mathbf{\Psi}$ should be treated explicitly. Terms with $\mathbf{\Psi}$ are usually treated implicitly but they can also be treated explicitly by using the current known values of $\mathbf{\Psi}$ but this is usually not done or kept to a minimum as it otherwise leads to instability in the solutions. Terms such as the time derivative $\frac{\partial \mathbf{\Psi}}{\partial t}$, advection $\nabla \cdot (\mathbf{U}\mathbf{\Psi})$ and

diffusion $\nabla \cdot (\Gamma \nabla \mathbf{\Psi})$[1] are usually treated implicitly. If linear terms such as $c\mathbf{\Phi}$, where c is a scalar, appear in an equation, they are usually treated implicitly as well. Examples of terms that are treated explicitly are the gradient $\nabla \mathbf{\Psi}$ and the divergence $\nabla \cdot \mathbf{\Psi}$ [1, pages 67-69].

2.7. Concept of Discretization

As previously mentioned, the finite volume method utilizes the concept of control volumes to be able to solve for and simulate the behavior of fluids. These control volumes correspond to cells in our mesh whose cell centroids store the respective flow variables $\mathbf{\Psi}$. The concept of constructing linear equations per cell in our mesh and then compiling them to matrix equations by discretizing each term in a given equation was also mentioned. Now, lastly, we will talk about how discretization as a concept works before going on to discretize each term in the Navier-Stokes momentum equations for unsteady incompressible flows in the next chapter 3 [1, pages 69-71].

The discretization starts with integrating each term in a given equation over a control volume such as a cell and then we turn those volume integrals to integrals over the surface of the control volume. The concept remains the same regardless of the given structures such as irregular polyhedral meshes etc [1, pages 69-71].

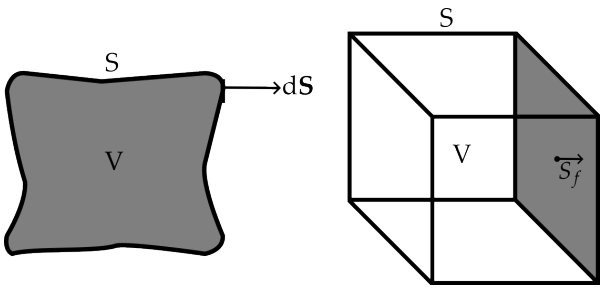

Figure 2.14.: Surface integrals $\int_S$ are approximated by summation over the face of the cell.

The surface integral appears as a result of converting the integration of a flow variable $\mathbf{\Psi}$ over the cell volume to an equal integration over the surface of the cell, which gives the total resulting flux of the variable $\mathbf{\Psi}$ from inflow and outflow at the surfaces and boundaries and considering any sources present inside the domain. The key idea is to later approximate this surface integral as a sum over the finite number of faces of the cell. Surface integral that represent derivatives such as $\nabla \cdot \mathbf{\Psi}$ are approximated by using this concept of summation of the fluxes of the flow variable over the cell faces f of the cell as shown in equation 14 [1, pages 69-71].

$$\int_S (d\mathbf{S} \cdot \mathbf{\Psi}) \rightarrow \sum_f \mathbf{S}_f \cdot \mathbf{\Psi}_f \tag{14}$$

[1]There are exceptions for the diffusion term such as when a given mesh has non-orthogonality [1].

To calculate the the flux $\mathbf{S}_f \cdot \mathbf{\Psi}_f$ of $\mathbf{\Psi}$, the value of $\mathbf{\Psi}$ at the face center of each corresponding face $\mathbf{\Psi}_f$ must be calculated. This value is calculated by interpolating $\mathbf{\Psi}$ to face center by using the current cell's and the neighboring cell's stored $\mathbf{\Psi}$ values at their cell centroids. The neighboring cell is the cell that is across the shared face i.e. the face shared between the current cell and this neighboring cell. This interpolation is usually done using linear interpolation which also considers the orthogonality of the face during the interpolation. This form of the linear interpolation is explained in the appendix, section A.2 [1, pages 69-71].

Lastly, after the discretizations are completed, the resulting expressions describe the derivatives i.e. the terms at a point such as in equation 15 [1, pages 69-71]. The concept is explained in the appendix, section A.3. The reasoning behind this is because in FVM, we store the flow variables at the cell centroids.

$$\nabla \cdot (\mathbf{U}\mathbf{\Psi}) \rightarrow \frac{1}{V} \sum_f \phi_f \mathbf{\Psi}_f \tag{15}$$

3. Discretization of the Incompressible Flow Navier-Stokes Momentum Equation

3.1. Incompressible Flow

The law of conservation of mass for a fluid, where ρ is the mass density of the fluid, is shown in equation 16 [1, pages 13-15].

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{U}) = 0 \tag{16}$$

Material derivative of a fixed mass of moving material, where $\mathbf{\Psi}$ denotes an example tensor of any rank, is given by equation 17 [1, pages 15-17].

$$\frac{D\mathbf{\Psi}}{Dt} = \frac{\partial \mathbf{\Psi}}{\partial t} + \mathbf{U} \cdot \nabla \mathbf{\Psi} \tag{17}$$

A homogenous and incompressible material has the property $\rho = constant$ uniformly observed throughout the entire fluid. Incompressible flow is a flow where this property is obeyed i.e. a flow where the density ρ does not change with time or space.

As an incompressible fluid has the property $\rho = constant$ over time, then for such a moving volume of fluid $\frac{D\rho}{Dt} = 0$. Combining the conservation of mass equation and the material derivative i.e. equations 16 and 17 for an incompressible fluid gives the incompressibility condition which is also called the continuity equation [1, pages 31-33]. This is shown in equation 18.

$$\nabla \cdot \mathbf{U} = 0 \tag{18}$$

This equation states that the divergence of the velocity field $\mathbf{U}$ is zero, implying that the flow is divergence-free. For our fluid in our simulation to be considered incompressible, the velocity field $\mathbf{U}$ must obey this rule.

The full form of the unsteady incompressible Navier-Stokes momentum equation for a homogenous Newtonian fluid is given by equation 19. This form of the Navier-Stokes momentum equation is derived using the law of conservation of momentum, the material derivative and other properties and identities [1, pages 31-33]. This equation is derived in a similar manner to the continuity equation but since the derivation is fairly involved, it is not shown here.

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot (\mathbf{U}\mathbf{U}) - \nabla \cdot (\nu \nabla \mathbf{U}) - (\nabla \nu) \cdot (\nabla \mathbf{U}) = -\nabla p + \mathbf{b} \tag{19}$$

The individual terms in equation 19 are:

- $\frac{\partial \mathbf{U}}{\partial t}$ (20), the time derivative term;
- $\nabla \cdot (\mathbf{UU})$ (21), the advection term;
- $-\nabla \cdot (\nu \nabla \mathbf{U}) - (\nabla \nu) \cdot (\nabla \mathbf{U})$ (22), the diffusion term;
- $-\nabla p$ (23), the pressure gradient term;
- $\mathbf{b}$ (24), the body force term;

The $\mathbf{b}$ symbol denotes any body force per unit mass. Body forces encompass forces that act throughout the volume rather than on a boundary surface, such as gravitational forces that operate over a distance [1, pages 19-21]. As previously mentioned, the ρ symbol is the fluid density. The ν symbol denotes the kinematic viscosity which is $\nu = \frac{\mu}{\rho}$, where μ is the dynamic pressure, and finally, the p symbol denotes the kinematic pressure, where dynamic (or absolute) pressure is divided by ρ [1, pages 31-33].

If $\mathbf{b} = 0$, and $\nu = constant$, then equation 19 simplifies and becomes equation 25 [1, pages 33-35].

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot (\mathbf{UU}) - \nabla \cdot (\nu \nabla \mathbf{U}) = -\nabla p \tag{25}$$

Equation 25 is the form of the Navier-Stokes momentum equation that we get when the fluid is homogeneous and incompressible, the diffusivity is constant and there is no body force acting on the fluid. This is the form that we will be discretizing and numerically solving.

3.2. Time Derivative Term Discretization

$$\frac{\partial \mathbf{\Psi}}{\partial t} \rightarrow \frac{\mathbf{\Psi} - \mathbf{\Psi^o}}{\Delta t} \tag{26}$$

The local time derivative $\frac{\partial \mathbf{\Psi}}{\partial t}$ in equation 25 or in other equations can be discretized using a finite difference in time. The idea is to express time t in discrete intervals / steps with a duration of of Δt such that $t^{n+1} = t^n + \Delta t$. As shown in equation 26, the time derivative is calculated using the field from the current time $\mathbf{\Psi}$ and the previous / old time $\mathbf{\Psi^o}$ [1, pages 91-93].

There are multiple ways to apply the discretization using different schemes but the ones we will focus on are the Euler methods, more specificially the forward Euler method (sometimes also called the forward difference), and the backward Euler method (sometimes also called the implicit Euler method). These two methods differ in the nature of how they calculate the next time step. Forward Euler method is an explicit method whereas backward Euler method is an implicit method [9] [3].

Forward Euler method is explicit because the next time step's velocity field is computed directly from the known quantities at the current time step. In more mathematical terms, the solution y_{n+1} is an explicit function of y_i for $i \leq n$ as seen in the form of the equation to be solved $y_{n+1} = y_n + hf(t_n, y_n)$ [9].

Backward Euler method is implicit because the next time step's velocity field is computed using information from the next time step itself. This involves solving a system of equations because the next time step's value appears on both sides of the equation. In more mathematical terms, the new approximation y_{n+1} appears on both sides of the equation. That is the reason why we must solve an algebraic equation for the unknown y_{n+1}. The equation to be solved is in the form of $y_{n+1} = y_n + hf(t_{n+1}, y_{n+1})$ [3].

3.2.1. Forward Euler Method

We start from equation 25 and isolate the time derivative on the left side by doing some algebraic manipulation. This gives us equation 27.

$$\frac{\partial \mathbf{U}}{\partial t} = -\nabla \cdot (\mathbf{U}\mathbf{U}) + \nabla \cdot (\nu \nabla \mathbf{U}) - \nabla p \tag{27}$$

We discretize the time derivative as a finite difference in time. This results in equation 28.

$$\frac{\mathbf{U}^{n+1} - \mathbf{U}^n}{\Delta t} = -\nabla \cdot (\mathbf{U}^n \mathbf{U}^n) + \nabla \cdot (\nu \nabla \mathbf{U}^n) - \nabla p^n \tag{28}$$

Since we already know the value of the velocity field from current time step, we isolate the velocity from the next time step on the left to get its value using the values we already know. This gives us the form of the explicit formulation that we can use which is equation 29.

$$\mathbf{U}^{n+1} = \mathbf{U}^n + \Delta t(-\nabla \cdot (\mathbf{U}^n \mathbf{U}^n) + \nabla \cdot (\nu \nabla \mathbf{U}^n) - \nabla p^n) \tag{29}$$

After discretizing other terms, equation 29 allows us to easily compute the next step. Although it is a simple and computationally cheap method, it can have stability issues if the time step used is large. As a result of this, it is important to monitor the Courant number for a given solution / simulation and adjust the time step that is used in the calculations accordingly [1, pages 91-93].

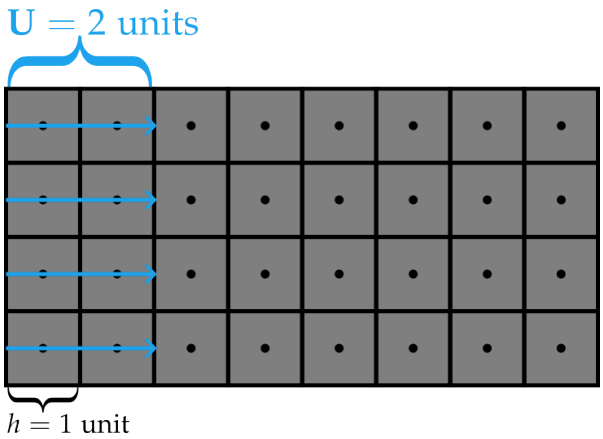

Figure 3.1.: A 2D mesh with axis aligned square cells where $\mathbf{U} = 2$ units and $h = 1$ unit.

In essence, Courant number is a measure of how much distance information travels ($\mathbf{U}$) through a computational grid cell (Δh) within a given time step (Δt). If the information travels more than one cell at time step of Δt, that means we miss the point at which the information was at the skipped cell. This obviously causes problems in our solution. Therefore, we would like to avoid this by choosing a Δt such that at every step we accurately get the correct information from each cell. To clarify how such a problem might occur, let's consider a solid example. Imagine a 2D grid with square cells all of which are axis aligned and has a size of 1 unit per side. Also imagine a fluid with a speed of 2 units per second in the direction of $x+$. If we were to use a Δh and Δt of 1, then per time step, the fluid would move 2 cells but that would mean we would lose resolution as we would skip a cell per time step Δt as seen in Figure 3.1. What we instead want is to be able to get information from every cell per each time step Δt. This is the reasoning behind the CFL condition $C_o \leq 1$ [1, pages 91-93] [10].

$$C_o = \Delta t(\sum_{i=1}^{n} \frac{u_i}{\Delta x_i}) \leq C_{max} \tag{30}$$

For mesh with axis aligned cells, the generalized CFL condition formula can be written as in equation 30. The value of C_{max} varies based on the method employed to solve the discretized equation, particularly depending on whether the approach is explicit or implicit. For explicit solvers, C_{max} is typically 1. Implicit solvers, on the other hand, tend to be more resilient to numerical instability, allowing for higher values of C_{max} [11].

For cells of any irregular polyhedral shape, the formula can be generalized as in equation 31, where $\phi^+ = \mathbf{S}_f \cdot \mathbf{U}_f$, ($\mathbf{S}_f = \hat{\mathbf{n}}_f A_f$) [1, pages 91-93].

$$C_o = \frac{\Delta t}{V} \sum_f \phi^+ \tag{31}$$

3.2.2. Backward Euler Method

We start from equation 25 and isolate the time derivative on the left side by doing some algebraic manipulation. This gives us equation 32.

$$\frac{\partial \mathbf{U}}{\partial t} = -\nabla \cdot (\mathbf{U}\mathbf{U}) + \nabla \cdot (\nu \nabla \mathbf{U}) - \nabla p \tag{32}$$

We discretize the time derivative as a finite difference in time. This results in equation 33.

$$\frac{\mathbf{U}^{n+1} - \mathbf{U}^n}{\Delta t} = -\nabla \cdot (\mathbf{U}^{n+1}\mathbf{U}^{n+1}) + \nabla \cdot (\nu \nabla \mathbf{U}^{n+1}) - \nabla p^{n+1} \tag{33}$$

We isolate the velocity from the next time step on the left to get its value by solving the system of equations. This gives us the form of the implicit formulation that we can use which is equation 34.

$$\mathbf{U}^{n+1} = \mathbf{U}^n + \Delta t(-\nabla \cdot (\mathbf{U}^{n+1}\mathbf{U}^{n+1}) + \nabla \cdot (\nu \nabla \mathbf{U}^{n+1}) - \nabla p^{n+1}) \tag{34}$$

$$\frac{\partial \mathbf{\Psi}}{\partial t} \rightarrow \frac{\mathbf{\Psi} - \mathbf{\Psi^o}}{\Delta t}$$

$$\frac{1}{\Delta t} \qquad \frac{\mathbf{\Psi^o}}{\Delta t}$$

$$\begin{bmatrix} * & * & & * \\ * & \circledast & * & \\ & * & * & * \\ * & & * & * \end{bmatrix} \begin{bmatrix} \mathbf{\Psi} \end{bmatrix} = \begin{bmatrix} * \\ \circledast \\ * \\ * \end{bmatrix}$$

Figure 3.2.: The velocity field at the current time $\mathbf{\Psi}$ only contributes to the diagonal entries in the matrix of coefficients $[A]$ while the velocity field at the previous time $\mathbf{\Psi}^o$ contributes to the source vector $[b]$!

After discretizing other terms, equation 34 allows us to compute the next step. As seen in Figure 3.2, the value of the velocity field at current step / time, which is symbolized with $\mathbf{\Psi}$ / $\mathbf{U}^{n+1}$, only contributes with $\frac{1}{\Delta t}$ to the diagonal entries of the matrix of coefficients $[A]$ i.e. to the coefficient of the cell for whose velocity we are solving. On the other hand, the velocity field at the previous time $\mathbf{\Psi}^o$ / $\mathbf{U}^n$ contributes to the source vector $[b]$ with $\frac{\mathbf{\Psi}^o}{\Delta t}$ [1, pages 91-93].

Although it is more stable than the explicit method, it can still have stability issues if the time step used doesn't fulfill the CFL condition. As a result of this, it is important to monitor the Courant number for the implicit solution and adjust the time step that is used in the calculations accordingly as well [1, pages 91-93].

3.3. Advection Term Discretization

Advection typically refers to the transport of a substance or quantity (e.g., heat, mass, momentum) by the bulk motion of the fluid. The advection term in the incompressible Navier-Stokes equation 25 represents how the velocity field $\mathbf{U}$ advects itself through the domain.

The discretization of the advection term starts at equation 25. To get the effect of each term on a given cell, we can integrate the Navier-Stokes momentum equation across the cell which gives us equation 35.

$$\int_V (\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot (\mathbf{U}\mathbf{U}) - \nabla \cdot (\nu \nabla \mathbf{U}))dV = \int_V (-\nabla p)dV \tag{35}$$

Since integration and addition are commutative operations, we can split up the various terms and integrate them one by one. This gives us equation 36.

$$\int_V (\frac{\partial \mathbf{U}}{\partial t}) dV + \int_V (\nabla \cdot (\mathbf{U}\mathbf{U})) dV - \int_V (\nabla \cdot (\nu \nabla \mathbf{U})) dV = \int_V (-\nabla p) dV \tag{36}$$

This way, we can consider each term in isolation and discretize accordingly. For this section, we will only consider the advection term as seen in equation 37.

$$\int_V (\nabla \cdot (\mathbf{U}\mathbf{U})) dV \tag{37}$$

To continue with our discretization, the trick we use is Gauss' divergence theorem which explained in the appendix, section A.1. Gauss' divergence theorem tells us that the integral of the divergence of any general vector field $\mathbf{\Psi}$ over a volume can be expressed equivalently as a surface integral as seen in equation 73 [12].

As a result, we can rewrite the volume integrals in the Navier-Stokes momentum equation as surface integrals. The surface integrals contain the dot product of the vector field $\mathbf{\Psi}$ and the unit normal vector of the surface $\hat{\mathbf{n}}$ that we are currently considering. This means that instead of integrating over the volume of the cell, we integrate over the entire surface of our cell. Applying Gauss' divergence theorem on equation 37 gives us equation 38 [12].

$$\int_V (\nabla \cdot (\mathbf{U}\mathbf{U})) dV = \int_S (\mathbf{U}(\mathbf{U} \cdot \hat{\mathbf{n}})) dS \tag{38}$$

In equation 38, $(\mathbf{U} \cdot \hat{\mathbf{n}}) dS$ is the volume flow rate out of the surface. The other velocity field $\mathbf{U}$ outside the inner paranthesis is the unknown velocity field we are solving for. That means $(\mathbf{U} \cdot \hat{\mathbf{n}}) dS$ is an explicit value that is calculated using the known values i.e. the value of the velocity field from the last time step is used. As a result, this term represents the coefficient contribution of this term to the matrix of coefficients $[A]$ [12].

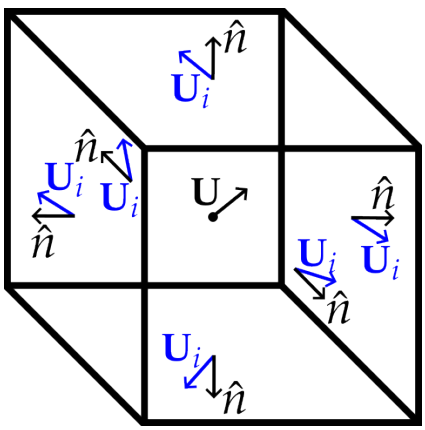

Figure 3.3.: Splitting the surface integral into a sum of surface integrals over individual faces.

Although we've written the volume integral in the form of a surface integral, we need to make further simplifications because we need to linearize the convection term. The trick we use is to remember that we have a cell with a finite number of faces. We can assume we have an arbitrary number of faces symbolized by M. This holds true for any irregular polyhedral

cell. As a result, we can replace the surface integral by the summation of the surface integral over all of the individual faces of our cell as shown in equation 39. A visual representation of this equation can be seen in Figure 3.3 [12].

$$\int_S (\mathbf{U}(\mathbf{U} \cdot \hat{\mathbf{n}}))dS = \sum_{i=1}^{M} \int_S (\mathbf{U}_i(\mathbf{U}_i \cdot \hat{\mathbf{n}}_i))dS_i \tag{39}$$

As previously explain in section 2.4, flow variables vary linearly across the cells and faces. Therefore, we can make a further simplification to equation 39. We can replace the surface integral across the face by using the velocity value at the center of the face multiplied by the volume flow rate out of the face. This is shown in equation 40 [12].

$$\sum_{i=1}^{M} \int_S (\mathbf{U}_i(\mathbf{U}_i \cdot \hat{\mathbf{n}}_i))\, dS_i \approx \sum_{i=1}^{M} \mathbf{U}_{i_f}(\mathbf{U}_{i_f} \cdot \hat{\mathbf{n}}_{i_f})S_i \tag{40}$$

We can see that the integral is now gone and all we have is a summation of terms where the summation is over each of the faces of the cell. For example, if we had a hexahedra, the summation would be a summation of six faces. In essence, what we are doing is we are summing the product of the volume flow rate out of that face center and the velocity at that face center [12].

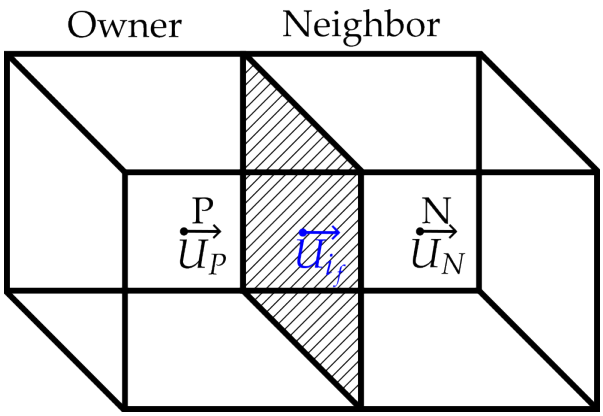

Figure 3.4.: Although we need the face center values $\mathbf{U}_{i_f}$, we only have the cell centroid values. Therefore, we need to use linear interpolation to get $\mathbf{U}_{i_f}$.

$$\sum_{i=1}^{M} \mathbf{U}_{i_f}(\mathbf{U}_{i_f} \cdot \hat{\mathbf{n}}_{i_f})S_i = \sum_{i=1}^{M} \mathbf{U}_{i_f}\phi_{i_f} \tag{41}$$

Equation 40 is usually written as in equation 41 in the literature. ϕ_{i_f} is $(\mathbf{U}_{i_f} \cdot \hat{\mathbf{n}}_{i_f})S_i$ which is called the volumetric flux (we previously also called it the volume flow rate out of the face). While equation 41 is the discretized form of the advection term for the velocity field $\mathbf{U}$, other properties such as temparature can be advected by velocity. Additionally, as previously explained in section 2.7 the resulting discretized expressions describe the derivatives i.e. the

terms at a point. Therefore, a more general form of the discretized advection term is shown in equation 42 [1, pages 75-77].

$$\nabla \cdot (\mathbf{U}\mathbf{\Psi}) = \lim_{\Delta V \to 0} \frac{1}{\Delta V} \int_S (\hat{\mathbf{n}} \cdot \mathbf{U}\mathbf{\Psi}) dS \to \frac{1}{V} \sum_{i=1}^{M} \hat{\mathbf{n}}_{i_f} \cdot \mathbf{U}_{i_f} \mathbf{\Psi}_{i_f} S_{i_f} \to \frac{1}{V} \sum_{i=1}^{M} \phi_{i_f} \mathbf{\Psi}_{i_f} \tag{42}$$

As a result, the final form of the discretized advection term for the momentum equation is shown in equation 43.

$$\frac{1}{V} \sum_{i=1}^{M} \phi_{i_f} \mathbf{U}_{i_f} \tag{43}$$

Equation 43 requires the values at the face centers $\mathbf{U}_{i_f}$ to be known; however, in FVM, flow variables such as the velocity field $\mathbf{U}$ is only stored at the cell centroids.

The velocity $\mathbf{U}_{i_f}$ in the volumetric flux ϕ_{i_f}, which uses the known velocity field values, is calculated by interpolating $\mathbf{U}$ to face center by using the current cell's and the neighboring cell's stored $\mathbf{U}$ values at their cell centroids, which are $\mathbf{U}_P$ and $\mathbf{U}_N$ respectively. The neighboring cell is the cell that is across the shared face i.e. the face shared between the current cell and this neighboring cell. This interpolation is usually done using linear interpolation which also considers the orthogonality of the face during the interpolation. This form of the linear interpolation is explained in the appendix, section A.2 [1, pages 69-71].

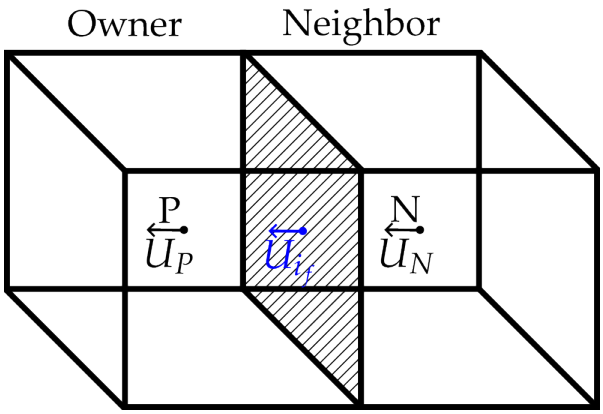

Figure 3.5.: Due to the flow direction, $\mathbf{U}_{i_f}$ is assigned the value of $\mathbf{U}_N$

On the other hand, the interpolation of the current unknown velocity $\mathbf{U}_{i_f}$ in equation 43 and the advected flow property in equation 42 to the face centers involves the flow direction because they are advected i.e. transported in the direction of the flow velocity $\mathbf{U}$. To clarify why this matters, let's take a look at figures 3.4 and 3.5. In figure 3.4, the flow direction is from left to right. Therefore, the advected property will also move in that direction. In contrast, in figure 3.5, the flow direction is from right to left [1, pages 75-79].

There are a variety of face interpolation schemes to deal with this case but we will use use the upwind interpolation scheme. In the upwind scheme, the face center value $\mathbf{\Psi}_{i_f}$ is assigned the value of the centroid value in the upwind cell $\mathbf{\Psi}_U$. Going back to figures 3.4 and

3.5, that would mean in figure 3.4 $\mathbf{U}_{i_f}$ is assigned the value of $\mathbf{U}_P$ whereas in figure 3.5 $\mathbf{U}_{i_f}$ is assigned the value of $\mathbf{U}_N$ due to the flow directions [1, pages 75-79].

Although this may seem straightforward to do visually, in complex flows and irregular meshes determining the flow direction is not so straightforward. So, we need a solid mathematical basis to be able to always determine the correct flow direction and assign the correct upwind value to the face center.

$$\phi_{i_f} = (\mathbf{U}_{i_f} \cdot \hat{\mathbf{n}}_{i_f}) S_i \tag{44}$$

$$\mathbf{\Psi}_{i_f} = \begin{cases} \mathbf{\Psi}_P & \text{if } \phi_{i_f} > 0 \quad \text{(Flow Out)} \\ \mathbf{\Psi}_N & \text{if } \phi_{i_f} < 0 \quad \text{(Flow In)} \end{cases} \tag{45}$$

The key to determining whether there is a flow in or out of the cell through a specific face is to check the sign of the volumetric flux through that face. As previously mentioned, volumetric flux uses the known values of the velocity field $\mathbf{U}_{i_f}$. Therefore, in equation 44, we use the previous / known value of the velocity field $\mathbf{U}_{i_f}$, which as also previously mentioned is interpolated to the face centers using the linear interpolation scheme explained in appendix section A.2.

After calculating the volumetric flux of a face, if the sign of the volumetric flux ϕ_{i_f} is positive, than for the face center value of the advected property $\mathbf{\Psi}_{i_f}$, the value at the owner cell centroid $\mathbf{\Psi}_P$ is used. Instead, if the sign of the volumetric flux ϕ_{i_f} is negative, than for the face center value of the advected property $\mathbf{\Psi}_{i_f}$, the value at the neighbor cell centroid $\mathbf{\Psi}_N$ is used. This is summarized and shown in equation 45 [13].

Thus, for the discretized advection term of the incompressible Navier-Stokes momentum equation shown in equation 43, if the sign of the volumetric flux ϕ_{i_f} is positive, than for the face center value of the advected property $\mathbf{\Psi}_{i_f}$, which is the velocity $\mathbf{U}_{i_f}$ itself in this case, the velocity at the owner cell centroid $\mathbf{U}_P$ is used. Instead, if the sign of the volumetric flux ϕ_{i_f} is negative, than for the face center value of the advected property $\mathbf{\Psi}_{i_f}$, which is the velocity $\mathbf{U}_{i_f}$ itself in this case, the velocity at the neighbor cell centroid $\mathbf{U}_N$ is used [13].

Once the proper values are assigned to the face center values in equation 43, the term can be used to add the coefficients to the matrix of coefficients $[A]$. Specificially, the term influences the coefficients of the owner cell and its neighbors in the row for the given cell.

3.4. Diffusion Term Discretization

This term describes the diffusion occurring across the surface of the fluid element, which in our case is the cell in consideration. Diffusion typically refers to the transfer of a fluid property, momentum in the case of incompressible Navier-Stokes momentum equation, resulting from the fluctuating motions that are not accounted for by the overall bulk motion represented by the continuum velocity $\mathbf{U}$. This is the process by which momentum (or other quantities such as heat or mass) is spread or smoothed out in the fluid due to molecular interactions and viscosity [1, pages 33-35].

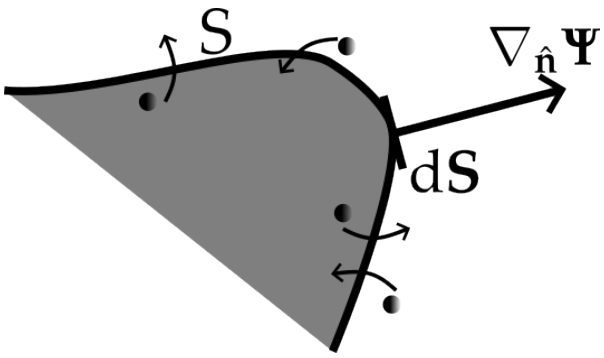

Figure 3.6.: Particles with higher $\mathbf{\Psi}$ migrate to areas where particles have lower $\mathbf{\Psi}$, and the opposite also occurs.

Fluctuations encompass any random movement of the particle components of matter, such as molecules and turbulent structures. These movements enable particles to cross surface boundaries, thereby transporting the property $\mathbf{\Psi}$ via its gradient [1, pages 33-35].

Particles with higher $\mathbf{\Psi}$ migrate to areas where particles have lower $\mathbf{\Psi}$, and the opposite also occurs. Through collisions between particles, regions with high $\mathbf{\Psi}$ values tend to decrease, while those with low values tend to increase. A visual representation of this is shown in figure 3.6, where S is total surface area of the cell, dS is the infinitesimally small surface element and $\nabla_{\hat{\mathbf{n}}}\mathbf{\Psi}$ is the surface normal gradient. The surface normal gradient is indicated by the operator $\nabla_{\hat{\mathbf{n}}}$ where $\nabla_{\hat{\mathbf{n}}}\mathbf{\Psi} \equiv \hat{\mathbf{n}} \cdot \nabla\mathbf{\Psi}$ [1, pages 33-35]. The surface normal gradient will be explained in detail as we discretize the diffusion term.

The discretization of the diffusion term also starts at equation 25. We once again integrate the Navier-Stokes momentum equation across the cell which gives us equation 35. Then, we split the various terms and this brings us back to equation 36. As before, doing so allows us to consider each term in isolation and discretize accordingly. For this section, we will only consider the diffusion term as seen in equation 46.

$$\int_V (\nabla \cdot (\nu \nabla \mathbf{U}))dV \tag{46}$$

To continue with our discretization, we utilize Gauss' divergence theorem once again. Applying Gauss' divergence theorem on equation 46 gives us equation 47 [14].

$$\int_V (\nabla \cdot (\nu \nabla \mathbf{U}))dV = \int_S ((\nu \nabla \mathbf{U}) \cdot \hat{\mathbf{n}})dS \tag{47}$$

To further linearize the diffusion term, we remember that we have a cell with a finite number of faces symbolized by M. Then, we can replace the surface integral by the summation of the surface integral over all of the individual faces of our cell as shown in equation 48. A visual representation of this equation can be seen in Figure 3.3 [14].

$$\int_S ((\nu \nabla \mathbf{U}) \cdot \hat{\mathbf{n}})dS = \sum_{i=1}^{M} \int_S ((\nu \nabla \mathbf{U}_i) \cdot \hat{\mathbf{n}}_i)dS_i \tag{48}$$

Since flow variables vary linearly across the cells and faces, we can make a further simplification to equation 48. We can replace the surface integral across the face by using the velocity surface gradient value at the center of the face multiplied by the kinematic viscosity ν and the surface area S_i all of which are dot producted with the surface normal vector $\hat{\mathbf{n}}_{\mathbf{i_f}}$. This is shown in equation 49 [14].

$$\sum_{i=1}^{M} \int_S ((\nu \nabla \mathbf{U}_i) \cdot \hat{\mathbf{n}}_i) dS_i \approx \sum_{i=1}^{M} ((\nu \nabla \mathbf{U}_{i_f}) \cdot \hat{\mathbf{n}}_{i_f}) S_i \tag{49}$$

We can see that the integral is now gone and all we have is a summation of terms where the summation is over each of the faces of the cell. While equation 49 is the discretized form of the diffusion term for the velocity field $\mathbf{U}$, other properties such as temperature is also subject to diffusion. Additionally, as previously explained in section 2.7 the resulting discretized expressions describe the derivatives i.e. the terms at a point. Therefore, a more general form of the discretized diffusion term is shown in equation 50, where Γ represents the diffusivity. In our case, we plug in the kinematic viscosity ν in diffusivity Γ's place [1, pages 71-75].

$$\nabla \cdot (\Gamma \nabla \mathbf{\Psi}) = \lim_{\Delta V \to 0} \frac{1}{\Delta V} \int_S (dS \Gamma \nabla_{\hat{\mathbf{n}}} \mathbf{\Psi}) \to \frac{1}{V} \sum_{i=1}^{M} |S_{i_f}| \Gamma_{i_f} \nabla_{\hat{\mathbf{n}}} \mathbf{\Psi}_{i_f} \tag{50}$$

As a result, the final form of the discretized diffusion term for the momentum equation is shown in equation 51.

$$\frac{1}{V} \sum_{i=1}^{M} |S_{i_f}| \nu_{i_f} \nabla_{\hat{\mathbf{n}}} \mathbf{U}_{\mathbf{i_f}} \tag{51}$$

In equation 51, $\nabla_{\hat{\mathbf{n}}} \mathbf{U}_{\mathbf{i_f}} = \hat{\mathbf{n}}_{\mathbf{i}} \cdot \nabla \mathbf{U}_{\mathbf{i_f}}$. Calculating $\hat{\mathbf{n}}_{\mathbf{i}} \cdot \nabla \mathbf{U}_{\mathbf{i_f}}$ is not straightforward because when discretizing $\nabla \mathbf{U}_{\mathbf{i_f}}$, we use the finite differencing and that is only accurate when cells in a mesh are orthogonal [1, pages 71-75].

To be able to understand orthogonality and its counterpart non-orthogonality, let's take a look at figures 3.7 and 3.8. In both figures, the cell centroids are connected by a distance vector symbolized by $\mathbf{d}$. Additionally, the face shared between both cells has a unit normal vector symbolized by $\hat{\mathbf{n}}$. The unit normal vector of a face is perpendicular to the shared face by definition. The angle between these two vectors is symbolized by the angle θ. In figure 3.7, we can clearly see that $\mathbf{d}$ and $\hat{\mathbf{n}}$ are aligned with each other and the angle θ between them is $0°$. Therefore, these two cells are orthogonal to each other. In constrast, in figure 3.8, we can clearly see that $\mathbf{d}$ and $\hat{\mathbf{n}}$ are not aligned with each other and the angle θ between them is $30°$. As a result, two cells are non-orthogonal to each other [14].

$$\theta = \cos^{-1} \left(\frac{\mathbf{d} \cdot \hat{\mathbf{n}}}{|\mathbf{d}||\hat{\mathbf{n}}|} \right) \tag{52}$$

The angle between $\mathbf{d}$ and $\hat{\mathbf{n}}$ can be calculated using equation 52. It is used to determine if two faces are orthogonal or not so that different interpolations may be chosen depending on the result. Additionally, some existing CFD codes such as OpenFOAM utilize this angle to refine meshes when it is above a certain value [14].

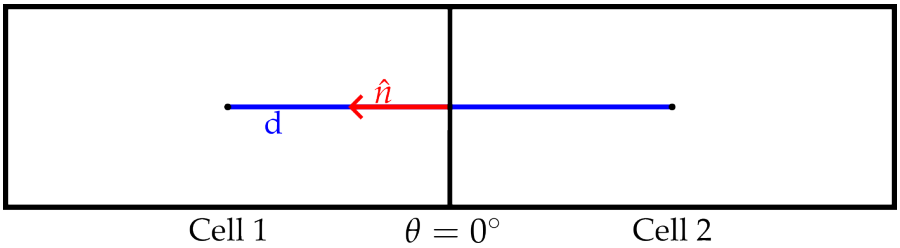

Figure 3.7.: These two cells are orthogonal because the distance vector **d** between the cell centroids and the unit normal vector $\hat{\mathbf{n}}$ of the shared face are aligned.

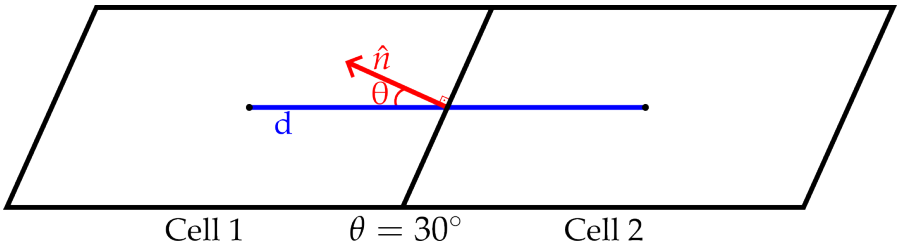

Figure 3.8.: These two cells are non-orthogonal because the distance vector **d** between the cell centroids and the unit normal vector $\hat{\mathbf{n}}$ of the shared face are not aligned.

It should be mentioned this the definition of non-orthogonality angle and non-orthogonality in general can lead to some diffuculties in certain situations. Such a situation is shown in figure 3.9. We can clearly see that cells 1 and 2 are not aligned with each other. However, when we connect the two cell centroids with the vector **d** as we did before, it seems to look aligned with the face unit normal vector $\hat{\mathbf{n}}$, which is pointing normal to the shared face. Therefore, using equation 52 might result in the wrong conclusion that the two cells are aligned. To detect this case and correctly measure the non-zero non-orthogonality angle, we need to use a different definition for the non-orthogonality angle between the two cells [14].

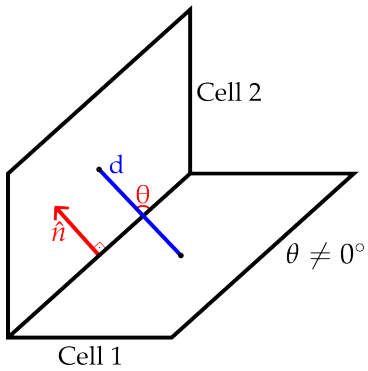

Figure 3.9.: In certain cases, using the vector **d** can lead to not correctly determining whether two cells are aligned or not. Here, using equation 52 would return a degree 0° incorrectly leading us to assume the cells are aligned.

An alternative to using vector **d** would be to use the vector from the cell centroid of the owner cell to the shared face's center. Using this vector that connects the cell centroid of cell 1 to the shared face rather than the cell centroid of the neighboring cell leads to the correct detection of the non-orthogonality. This new vector is called vector **c** as seen in figure 3.10. The alternative formula to calculate the non-orthogonality angle θ can be seen in equation 53 [14].

$$\theta = \cos^{-1}\left(\frac{\mathbf{c}\cdot\hat{\mathbf{n}}}{|\mathbf{c}||\hat{\mathbf{n}}|}\right) \tag{53}$$

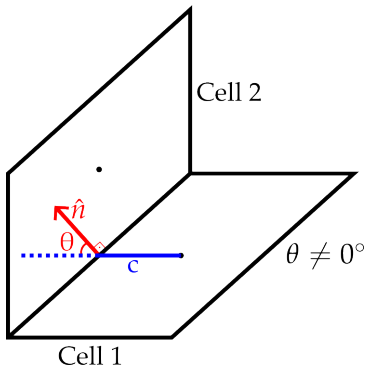

Figure 3.10.: In certain cases, using the vector **d** can lead to not correctly determining whether two cells are aligned or not. Solution is to use the vector **c** that connects the owner cell's centroid to the shared face's center.

Going back to equation 51, ν_{i_f} can be calculated using the linear interpolation scheme explained in the appendix but doing the same is incorrect for the surface normal gradient $\nabla_{\hat{\mathbf{n}}}\mathbf{U}_{\mathbf{i_f}}$ when cells are non-orthogonal i.e. not aligned.

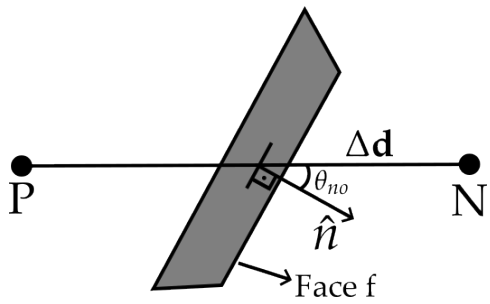

Figure 3.11.: When calculating the surface normal gradient $\nabla_{\hat{\mathbf{n}}}\mathbf{U}_{\mathbf{i_f}}$, it is important to keep in mind the non-orthogonality.

This means that we need to consider the two cases which are when two cells are orthogonal / aligned and when two cells are non-orthogonal / non-aligned. Calculation of $\nabla_{\hat{\mathbf{n}}}\mathbf{U}_{\mathbf{i_f}}$ when two cells are orthogonal is straightforward as the surface normal gradient can be discretized using the finite difference of cell centroid values $\mathbf{U}_N$ and $\mathbf{U}_P$. The resulting equation can be

seen in equation 54. The formula for any flow variable instead can be seen in equation 55 [1, Pages 71-75].

$$\nabla_{\hat{\mathbf{n}}}\mathbf{U}_{\mathbf{i_f}} = \frac{(\mathbf{U}_N - \mathbf{U}_P)}{\Delta d} = C_\Delta(\mathbf{U}_N - \mathbf{U}_P) \tag{54}$$

$$\nabla_{\hat{\mathbf{n}}}\mathbf{\Psi}_{i_f} = \frac{(\mathbf{\Psi}_N - \mathbf{\Psi}_P)}{\Delta d} = C_\Delta(\mathbf{\Psi}_N - \mathbf{\Psi}_P) \tag{55}$$

When two cells are non-orthogonal, the discretization of $\nabla_{\hat{\mathbf{n}}}\mathbf{U}_{\mathbf{i_f}}$ is achieved by the vector sum of the orthogonal scheme $\nabla_{\hat{\mathbf{n}}}\mathbf{U}^{\mathbf{orth}}_{\mathbf{i_f}}$ and the non-orthogonal correction / explicit correction $\nabla_{\hat{\mathbf{n}}}\mathbf{U}^{\mathbf{corr}}_{\mathbf{i_f}}$. The non-orthogonal correction is also called the explicit correction because it is calculated using the full gradient $\nabla\mathbf{U}$ in adjacent cells by interpolating to the face centers to get $\nabla\mathbf{U}_{\mathbf{i_f}}$ [1, Pages 71-75].

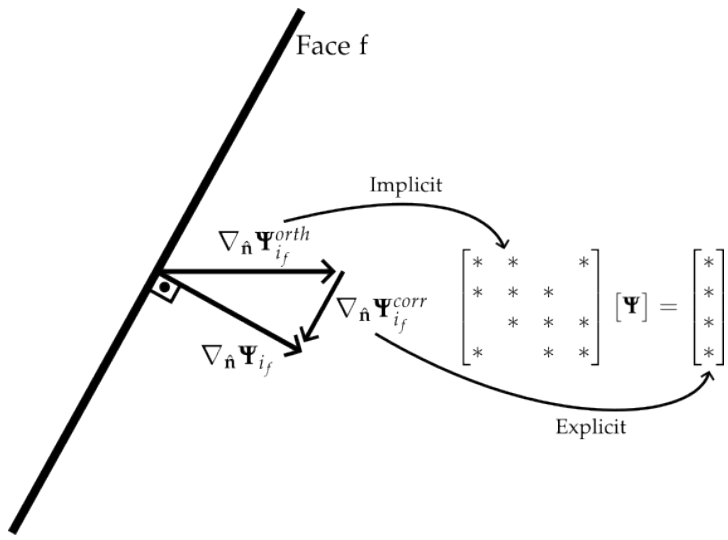

Figure 3.12.: We seperate the $\nabla_{\hat{\mathbf{n}}}\mathbf{U}_{\mathbf{i_f}}$ to its orthogonal and non-orthogonal components. The orthogonal contribution is an implicit one whereas the non-orthogonal contribution is an explicit one.

To better understand what the non-orthogonal correction represents, we can take a look at figure 3.12. Since $\nabla_{\hat{\mathbf{n}}}\mathbf{U}_{\mathbf{i_f}}$ is not aligned with the distance vector $\mathbf{d}$, we separate it two components. One these component is the orthogonal component $\nabla_{\hat{\mathbf{n}}}\mathbf{U}^{\mathbf{orth}}_{\mathbf{i_f}}$ and the other one is the non-orthogonal component $\nabla_{\hat{\mathbf{n}}}\mathbf{U}^{\mathbf{corr}}_{\mathbf{i_f}}$. In a way, we can think of this whole operation as projecting the unit normal vector of the face $\hat{\mathbf{n}}$ to the distance vector $\mathbf{d}$ and then subtracting this projection from $\hat{\mathbf{n}}$ to get the non-orthogonal component of the unit normal vector. Instead, in our formulation, the unit normal vector and the gradient are combined in the surface normal gradient term [1, Pages 71-75].

$$\hat{\mathbf{n}} - C^{corr}_\Delta \Delta\mathbf{d} \tag{56}$$

$$C_{\Delta}^{corr} = \frac{1}{\hat{\mathbf{n}} \cdot \Delta \mathbf{d}} = \frac{1}{|\Delta \mathbf{d}| \cos \theta_{no}} \tag{57}$$

The non-orthogonal component of the unit normal vector is found by using equation 56, where C_{Δ}^{corr} is calculated using equation 57 [1, Pages 71-75].

$$(\hat{\mathbf{n}} - C_{\Delta}^{corr} \Delta \mathbf{d}) \cdot \nabla \mathbf{\Psi}_{i_f} \tag{58}$$

$$(\hat{\mathbf{n}} - C_{\Delta}^{corr} \Delta \mathbf{d}) \cdot \nabla \mathbf{U}_{i_f} \tag{59}$$

Therefore, the way to calculate the non-orthogonal correction of a flow variable can be seen in equation 58 and as a result the non-orthogonal correction of the velocity field can found using equation 59. As previously mentioned, the gradient in the non-orthogonal correction component is calculated using the full gradient $\nabla \mathbf{U}$ in adjacent cells by interpolating to the face centers to get $\nabla \mathbf{U}_{\mathbf{i_f}}$. The way this gradient is discretized is explained in subsection 3.4.1 [1, Pages 71-75].

$$C_{\Delta}^{corr} (\mathbf{\Psi}_N - \mathbf{\Psi}_P) \tag{60}$$

$$C_{\Delta}^{corr} (\mathbf{\Psi}_N - \mathbf{\Psi}_P) \tag{61}$$

Instead of using C_{Δ} for while calculating the orthogonal component of the surface normal gradient using finite differencing, in the literature C_{Δ}^{corr} is used in its place as this ensures that the iterative sequence converges when we solve for the matrix equation later. Thus, the way to calculate the orthogonal component of a flow variable is given in equation 60. For the velocity field, this equation becomes equation 61 [1, Pages 71-75].

$$\nabla_{\hat{\mathbf{n}}} \mathbf{U}_{i_f} = \underbrace{C_{\Delta}^{corr} (\mathbf{U}_N - \mathbf{U}_P)}_{\text{orthogonal, implicit}} + \underbrace{(\hat{\mathbf{n}} - C_{\Delta}^{corr} \Delta \mathbf{d}) \cdot \nabla \mathbf{U}_{i_f}}_{\text{correction, explicit}} \tag{62}$$

$$\nabla_{\hat{\mathbf{n}}} \mathbf{\Psi}_{i_f} = \underbrace{C_{\Delta}^{corr} (\mathbf{\Psi}_N - \mathbf{\Psi}_P)}_{\text{orthogonal, implicit}} + \underbrace{(\hat{\mathbf{n}} - C_{\Delta}^{corr} \Delta \mathbf{d}) \cdot \nabla \mathbf{\Psi}_{i_f}}_{\text{correction, explicit}} \tag{63}$$

Combining contributions from the orthogonal and non-orthogonal components, we get equations 62 and 63, where equation 63 is the generalized form for a flow variable surface normal gradient and equation 62 is the form for the velocity surface normal gradient [1, Pages 71-75].

3.4.1. Green Gauss Method

$$\nabla \mathbf{\Psi} = \lim_{\Delta V \to 0} \frac{1}{\Delta V} \int_S (d\mathbf{S}\mathbf{\Psi}) \to \frac{1}{V} \sum_{i=1}^{M} \mathbf{S}_{i_f} \mathbf{\Psi}_{i_f} \equiv \frac{1}{V} \sum_{i=1}^{M} S_{i_f} \hat{\mathbf{n}} \mathbf{\Psi}_{i_f} \tag{64}$$

Equation 64 is used to calculate the gradient of a flow variable $\mathbf{\Psi}$ at the respective cell centroid. As before, the face center values $\mathbf{\Psi}_{i_f}$ of the flow variable $\mathbf{\Psi}$ are calculated using the linear interpolation scheme described in A.2 [1, Pages 87-89].

Equation 64 is derived using Gauss' divergence theorem but the full derivation will not shown here as it logically follows a very similar structure to the discretization of the diffusion and advection terms.

3.5. Pressure-Velocity Coupling

$$\mathbf{U}^{n+1} = \mathbf{U}^{n} + \Delta t(-\frac{1}{V}\sum_{i=1}^{M}\phi_{i_f}\mathbf{U}_{i_f}^{n+1} + \frac{1}{V}\sum_{i=1}^{M}|S_{i_f}|\nu_{i_f}\nabla_{\hat{\mathbf{n}}}\mathbf{U}_{\mathbf{i_f}}^{\mathbf{n+1}} - \nabla p^{n+1}) \tag{65}$$

Now that we are done with the discretization of the advection and diffusion terms, we can plug in equations 43 and 51 in equation 34 i.e. we replace the non-discretized versions of the advection and diffusion terms with their discretized forms. The resulting equation is given in equation 65.

Although we now have discretized the advection and diffusion terms and we could easily do the same for the pressure gradient, we are faced with a problem. The problem is that at each step in the incompressible Navier-Stokes momentum equation shown in equation 65, the formula requires the current step's pressure field to be known in order to calculate the current step's velocity field. This is called the Pressure-Velocity Coupling [1].

The pressure field p does not have a time evolution equation like the velocity field $\mathbf{U}$. Instead, p must be determined implicitly to ensure that the velocity field remains divergence-free i.e. at each step the velocity field must satisfy the continuity equation which is shown in equation 18. The velocity field must be adjusted in such a way that the incompressibility condition is maintained at each time step [1].

Several numerical methods have been developed to handle the coupling between velocity and pressure in the incompressible Navier-Stokes equations. One of these is called The Projection Method which involves a two step process [2]:

1. An intermediate velocity field $\mathbf{U}^{*n+1}$ is computed without considering the pressure term $\mathbf{U}^{*n+1} = \mathbf{U}^{n} + \Delta t\left(-\frac{1}{V}\sum_{i=1}^{M}\phi_{i_f}\mathbf{U}_{i_f}^{*n+1} + \frac{1}{V}\sum_{i=1}^{M}|S_{i_f}|\nu_{i_f}\nabla_{\hat{\mathbf{n}}}\mathbf{U}_{\mathbf{i_f}}^{*\mathbf{n+1}}\right)$. (66)

2. To determine the pressure p^{n+1}, a Poisson equation for pressure is solved $\nabla^2 p^{n+1} = \frac{1}{\Delta t}\nabla\cdot\mathbf{U}^{*n+1}$. (67)

3. The intermediate velocity field $\mathbf{U}^{*n+1}$ is then corrected by the pressure gradient to enforce the incompressibility condition $\mathbf{U}^{n+1} = \mathbf{U}_{i_f}^{*n+1} - \Delta t\nabla p^{n+1}$. (68)

Therefore, we have a loop we can iteratively solve for a given CFD problem. This algorithm will be the basis of the shared solutions in chapter 4. The reasoning behind why this algorithm works is because what we are doing is that we are utilizing an operator splitting approach. Instead of solving for a velocity field such that we consider the advection, diffusion and

pressure gradient terms at the same time as in equation 65, we first consider the advection and diffusion terms and solve for the intermediate velocity field using equation 66 and then we add the contribution of the pressure gradient term in equation 68. Additionally, we tie the incompressibility condition i.e the continuity equation $\nabla \cdot \mathbf{U}$ and the pressure gradient ∇p such that not only we get a way to solve for the new pressure field at the current time step $n+1$, we also ensure that the contribution of the pressure gradient term ∇p^{n+1} to the intermediate velocity field $\mathbf{U}_{i_f}^{*n+1}$ makes $\mathbf{U}^{n+1}$ satisfy the continuity equation [15].

The Poisson equation for the pressure field is derived as follows:

1. We start at the step where we add the contribution of the pressure gradient term to get the correct velocity field $\mathbf{U}^{n+1} = \mathbf{U}_{i_f}^{*n+1} - \Delta t \nabla p^{n+1}$. (69)

2. To ensure that both the left and right sides of equation 69 satisfy the continuity equation, we take divergence of both sides $\nabla \cdot \mathbf{U}^{n+1} = \nabla \cdot \left(\mathbf{U}_{i_f}^{*n+1} - \Delta t \nabla p^{n+1} \right) = 0$. (70)

3. Using the right hand side of equation 70 and simplifying it, we get the following intermediate equation $\nabla \cdot \mathbf{U}_{i_f}^{*n+1} - \Delta t \nabla^2 p^{n+1} = 0$. (71)

4. By further algebraic manipulation of equation 72 to get the pressure on one side and the velocity on the other, we get the Poisson equation for pressure $\nabla^2 p^{n+1} = \frac{1}{\Delta t} \nabla \cdot \mathbf{U}^{*n+1}$. (72)

4. Results

In chapter, results of the implementation of the FVM method explained in previous chapters are shown using different initial conditions. Additionally, comparisons with FDM, which utilizes the explicit method and uses the Semi-Lagrangian method for its advection term, are provided where it is possible to do so.

The implementations[1] were done in Φ_{Flow} [4] and all of the figures come from the simulations done using Φ_{Flow} [4]. Finite difference method simulation shown also come from Φ_{Flow} [4].

Overall, the simulations done using different initial conditions and unstructured mesh set up (irregular, with obstacles etc.) show satisfactory results confirming the usefulness of FVM for CFD simulations.

4.1. Divergence Evolution Simulation

In this simulation, if we look at Figure 4.1, over time $\nabla^2 p - \nabla \cdot \mathbf{U}^n$ shows that the solution for the Poisson equation for pressure goes to 0 as the solution converges and the velocity field $\mathbf{U}$ becomes divergence free to satisfy the continuity equation. Additionally, $\nabla \cdot \mathbf{U}^{n-1}$ shows that the velocity field indeed becomes divergence free as the divergence of the velocity field goes to 0 over time.

This confirms that the projection method by Chorin [2] indeed makes the velocity field divergence free over time as the solution converges and our FVM implementation in Φ_{Flow} [4] works correctly.

Table 4.1.: Simulation generating Figure 4.1 has the shown initial conditions in the table and the velocity field shown in Figure 4.2.

Left Boundary	Right Boundary	Top Boundary	Bottom Boundary
Zero Gradient	Zero Gradient	Zero Gradient	Zero Gradient
Diffusivity	Δt	Number of Steps	
0.1	0.01	1000	

[1]The corresponding code for each simulation can be found at `https://github.com/Popichi/differentiable_finite_volume_method_masters_thesis_code`.

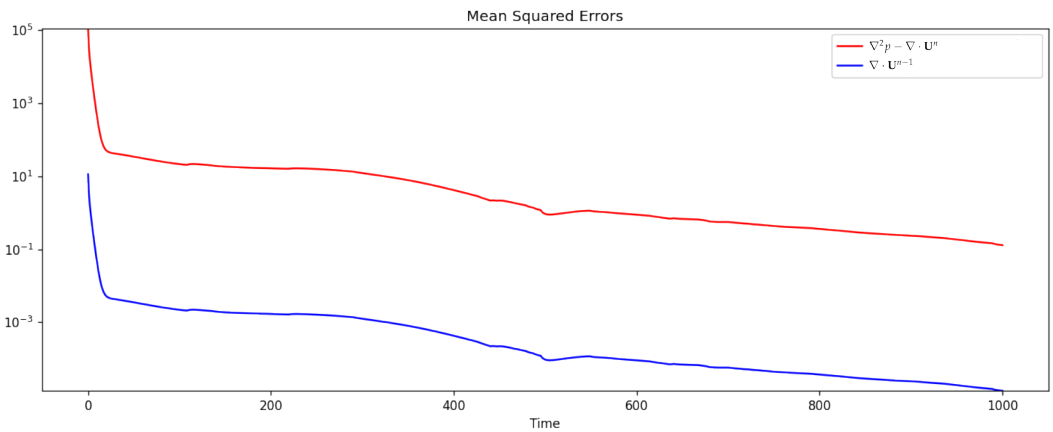

Figure 4.1.: $\nabla^2 p - \nabla \cdot \mathbf{U}^n$ shows that the solution for the Poisson equation for pressure goes to 0 as the solution converges. $\nabla \cdot \mathbf{U}^{n-1}$ shows that the velocity field's divergence goes to 0 over time to satisfy the incompressibility condition as the solution converges.

4.2. Implicit FVM Simulation

In this simulation, a cross shaped wave is initialized as shown in Figure 4.2 using an unstructured mesh for the FVM simulation and as shown in Figure 4.5 using a grid for the explicit FDM simulation. The initial conditions are respectively listed in table 4.2 and table 4.3.

The implicit FVM simulation has the velocity and pressure field evolutions as shown in Figure 4.3 and Figure 4.4 respectively. In comparison, the explicit FDM simulation has the velocity and pressure field evolutions as shown in Figure 4.6 and Figure 4.7 respectively. In addition to using an explicit scheme, the explicit FDM simulation also utilizes the Semi-Lagrangian method for the advection term.

For the given initial conditions, both methods eventually converge towards a similar solution. Differences are to be expected due to the differences in discretization and underlying field structure for unstructured meshes and centered grids. The comparison shows that FVM using an unstructured mesh is indeed a viable option for CFD simulations.

Table 4.2.: Simulation generating Figure 4.3 and Figure 4.4 has the shown initial conditions in the table and the velocity field shown in Figure 4.2.

Left Boundary	Right Boundary	Top Boundary	Bottom Boundary
Zero Gradient	Zero Gradient	Zero Gradient	Zero Gradient
Diffusivity	Δt	Number of Steps	
0.1	0.01	1000	

Table 4.3.: Simulation generating Figure 4.6 and Figure 4.7 has the shown initial conditions in the table and the velocity field shown in Figure 4.5.

Left Boundary Zero Gradient	Right Boundary Zero Gradient	Top Boundary Zero Gradient	Bottom Boundary Zero Gradient
Diffusivity	Δt	Number of Steps	
0.1	0.01	1000	

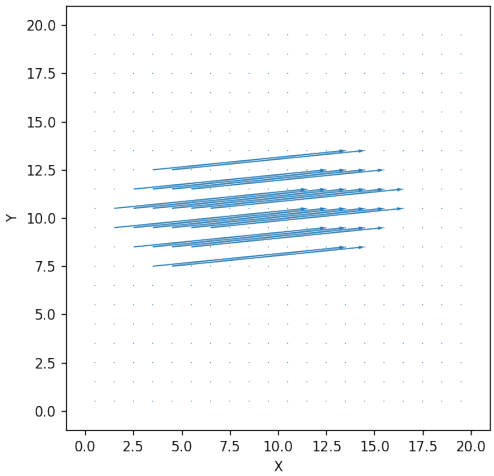

Figure 4.2.: Implicit FVM Simulation Initial Velocity Field

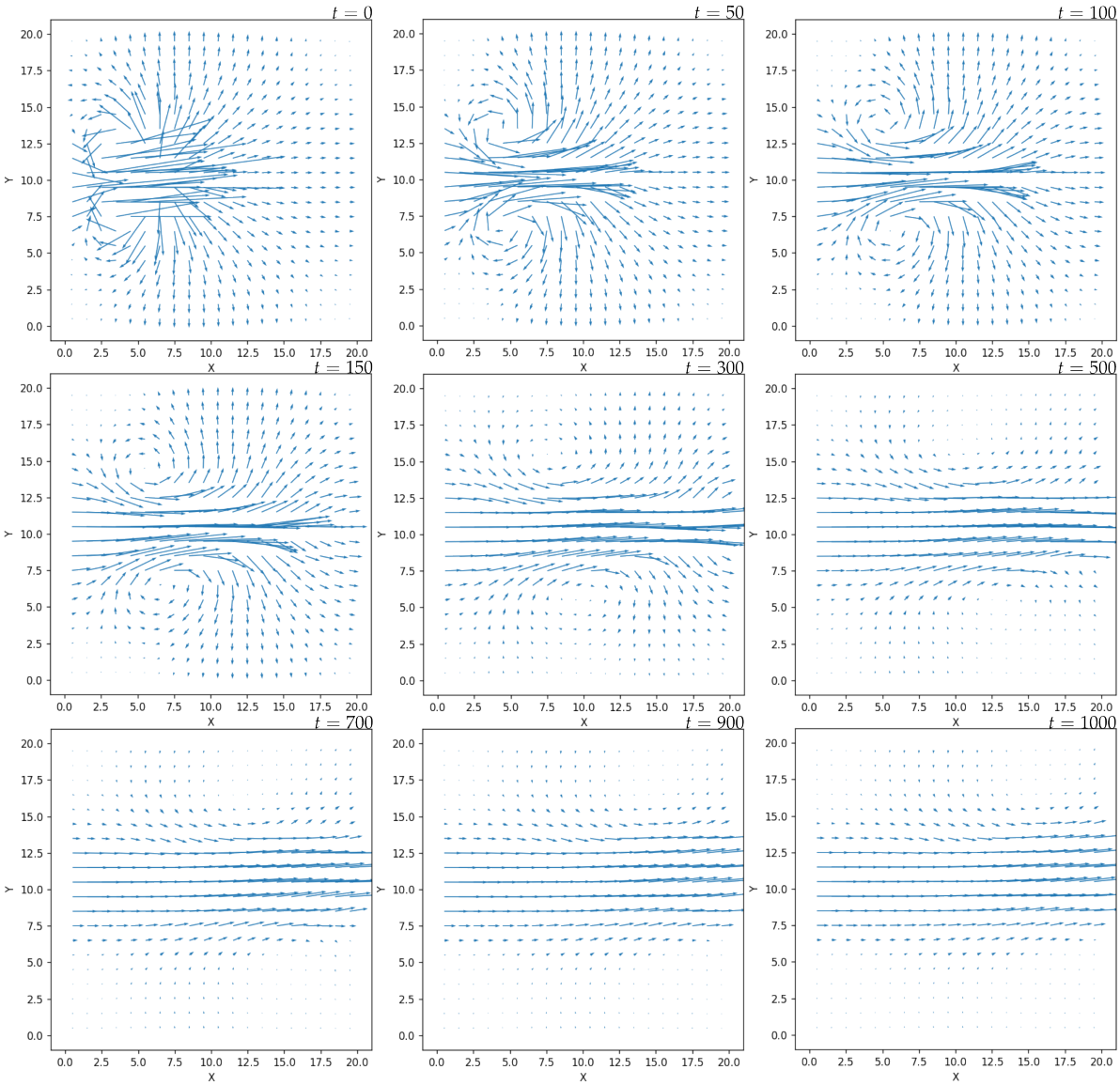

Figure 4.3.: Implicit FVM Simulation Velocity Field Evolution

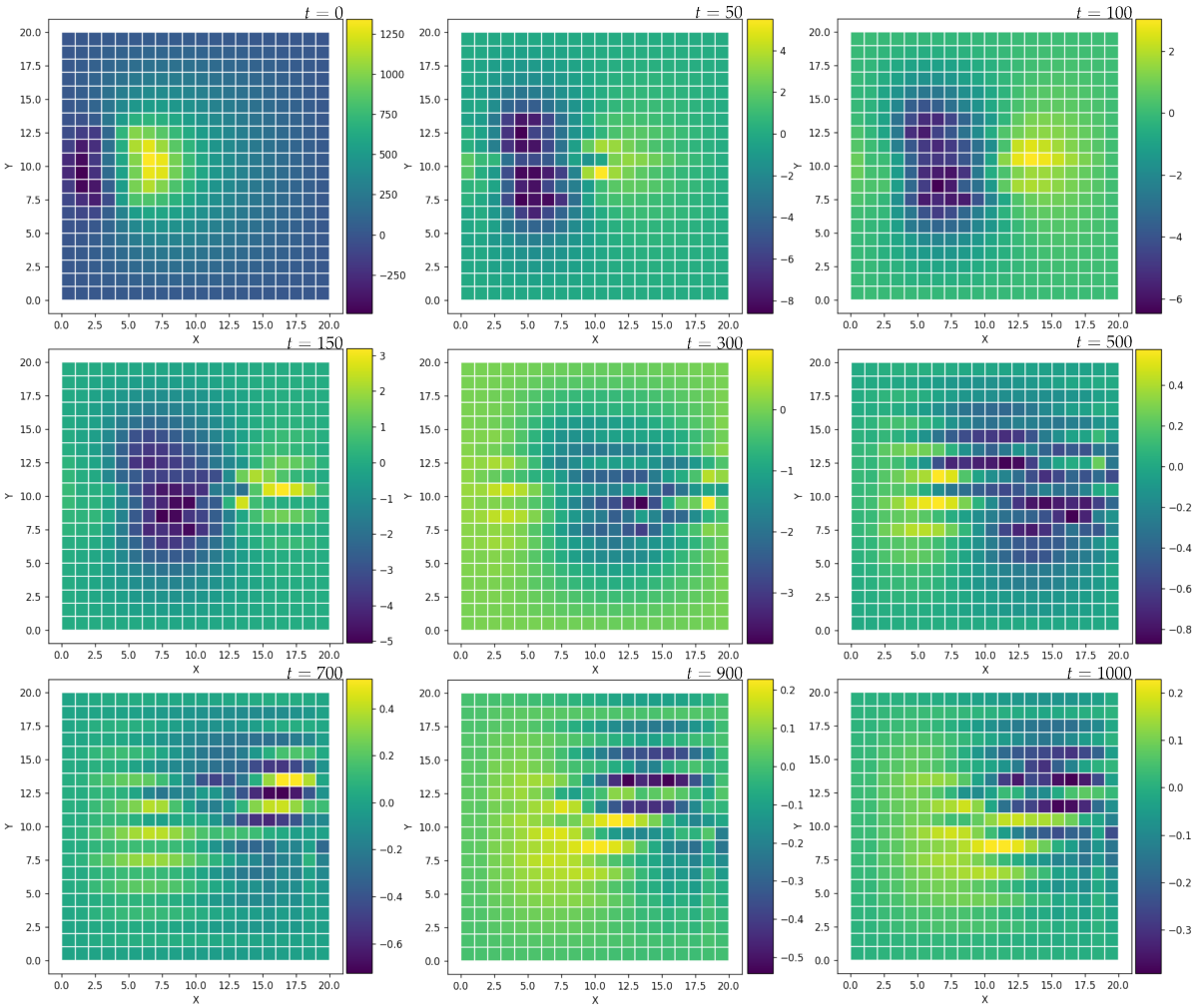

Figure 4.4.: Implicit FVM Simulation Pressure Field Evolution

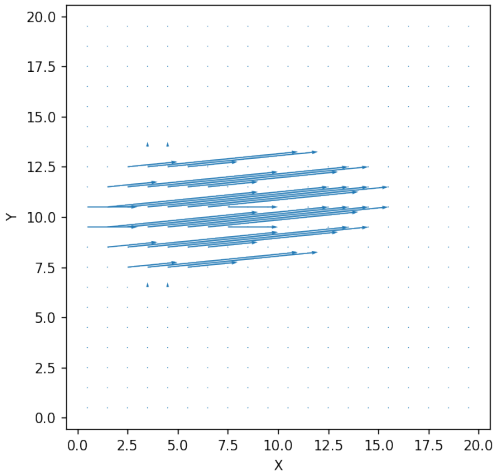

Figure 4.5.: Explicit FDM Simulation using a Grid and Split Step Using Semi-Lagrangian, Initial Velocity Field

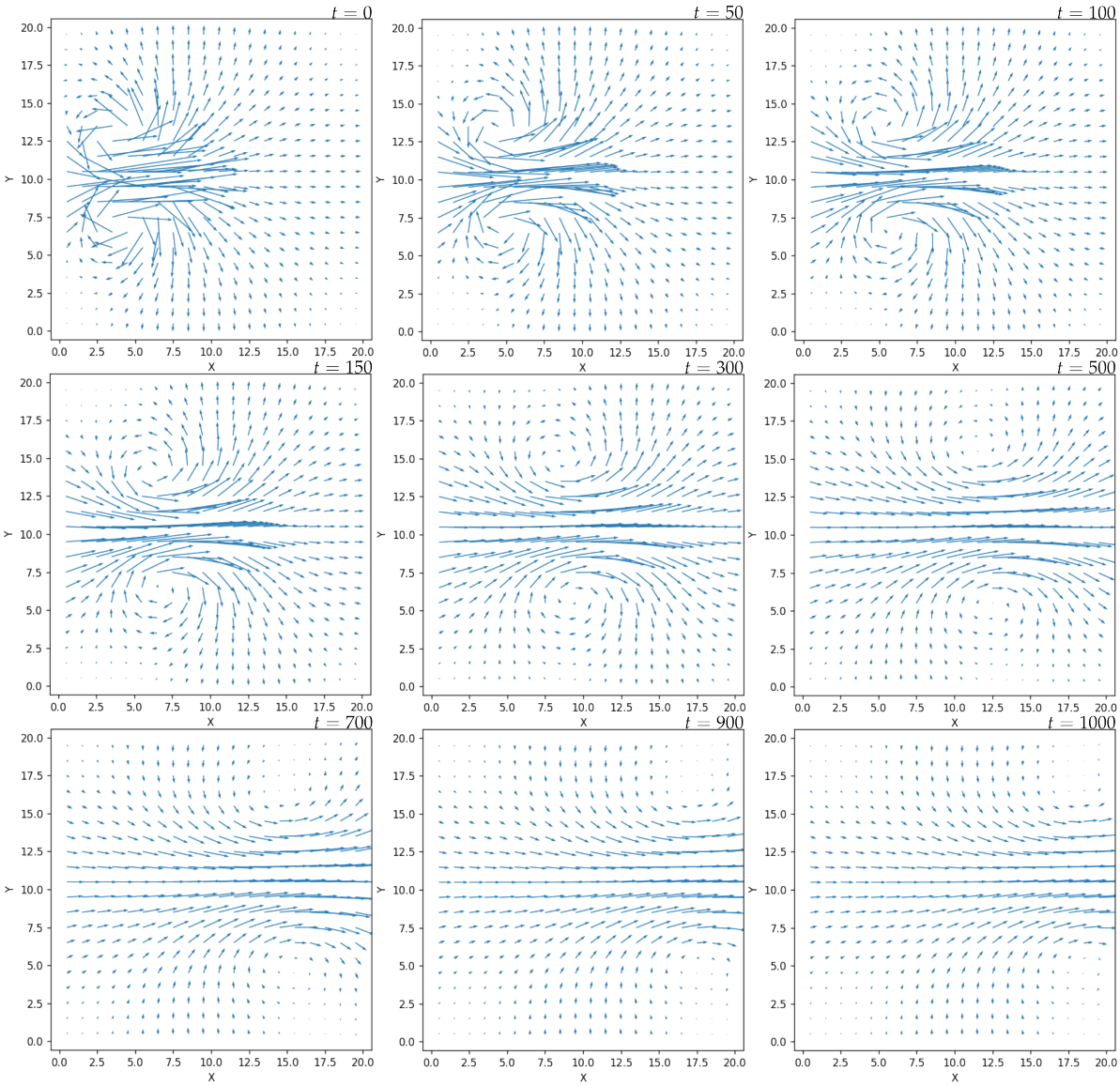

Figure 4.6.: Explicit FDM Simulation using a Grid and Split Step Using Semi-Lagrangian, Velocity Field Evolution

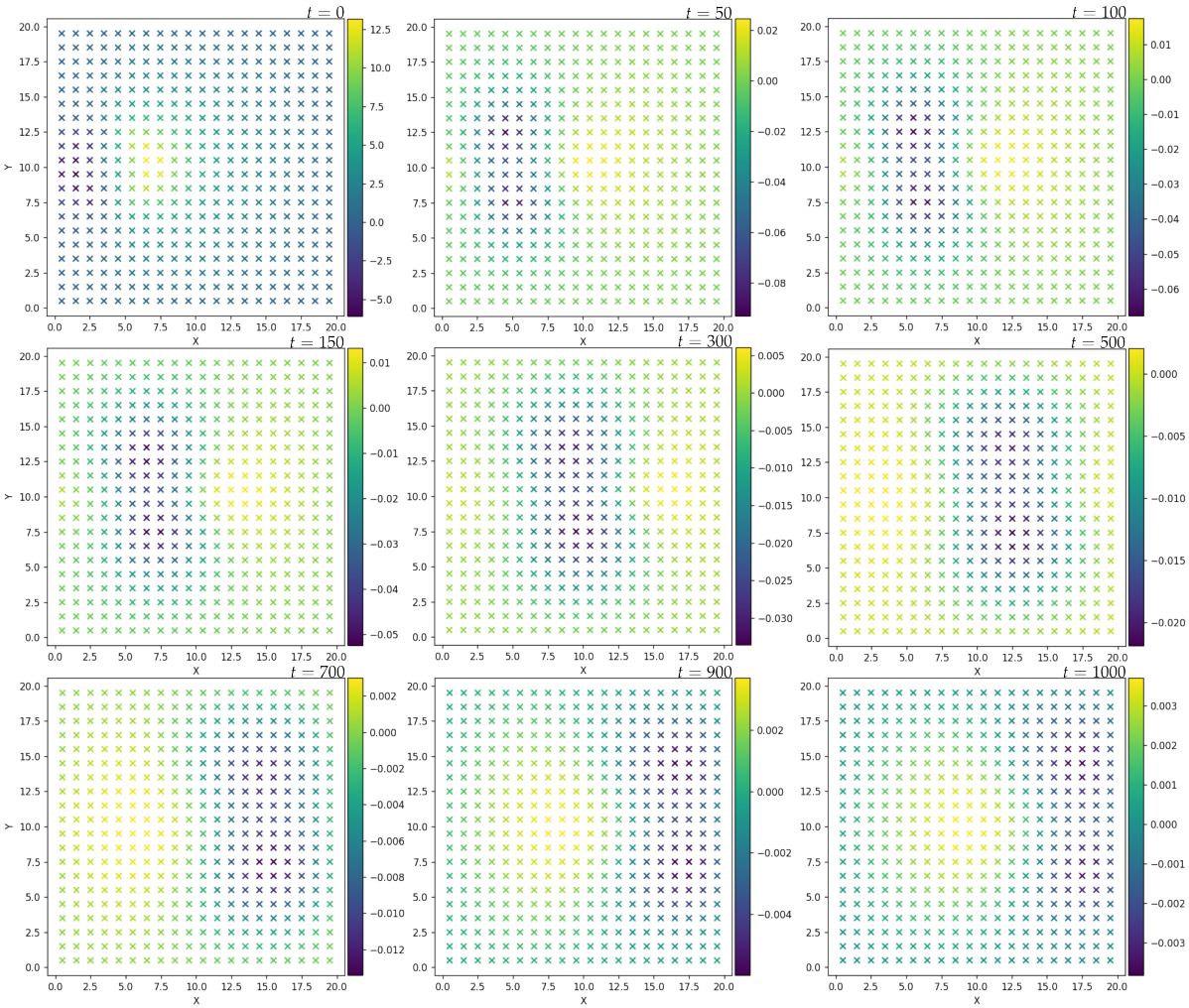

Figure 4.7.: Explicit FDM Simulation using a Grid and Split Step Using Semi-Lagrangian, Pressure Field Evolution

4.3. Implicit FVM Simulation with a Sinusoidal Unstructured Mesh

In this simulation, a cross shaped wave is initialized as shown in Figure 4.8 and Figure 4.11 using an sinusoidal unstructured mesh for the implicit FVM simulation. The initial conditions are respectively listed in table 4.4 and table 4.5. The only difference between the two simulations is the option enabling the non-orthogonal correction for the diffusion term. The implicit FVM simulation without the non-orthogonal correction [2] has the velocity and

[2] In comparison to how the non-orthogonal correction was described in section 3.4, currently Φ_{Flow} uses a simplified version of this correction which is as follows: $\nabla_n \mathbf{U}_{f_i} = \Delta \left(\frac{U_N - U_P}{|\mathbf{d}|} \right) + k \nabla \mathbf{U}_{f_i}$, where $\Delta = \left(\frac{\hat{\mathbf{n}} \cdot \mathbf{d}}{|\mathbf{d}|^2} \right) \mathbf{d}$ and $k = \hat{\mathbf{n}} - \Delta$. Although the given way in section 3.4 is more stable and better, this version is also very effective for the vast majority of cases.

pressure field evolutions as shown in Figure 4.9 and Figure 4.10 respectively.

In comparison, the implicit FVM simulation with the non-orthogonal correction has the velocity and pressure field evolutions as shown in Figure 4.12 and Figure 4.13 respectively.

For the given initial conditions, both methods eventually converge towards a similar solution but we can see that in Figure 4.12 compared to Figure 4.9 the velocity field follows the sinusoidal mesh more closely. This goes to show that in irregular meshes, non-orthogonal correction makes a difference even though it is minimal in this case. In highly irregular meshes, enabling the non-orthogonal correction might be the deciding factor between getting a solution or diverging and not getting a solution.

Table 4.4.: Simulation generating Figure 4.9 and Figure 4.10 has the shown initial conditions in the table and the velocity field shown in Figure 4.8.

Left Boundary	Right Boundary	Top Boundary	Bottom Boundary
Zero Gradient	Zero Gradient	Zero Gradient	Zero Gradient
Diffusivity	Δt	Number of Steps	
0.1	0.01	1000	

Table 4.5.: Simulation generating Figure 4.12 and Figure 4.13 has the shown initial conditions in the table and the velocity field shown in Figure 4.11.

Left Boundary	Right Boundary	Top Boundary	Bottom Boundary
Zero Gradient	Zero Gradient	Zero Gradient	Zero Gradient
Diffusivity	Δt	Number of Steps	
0.1	0.01	1000	

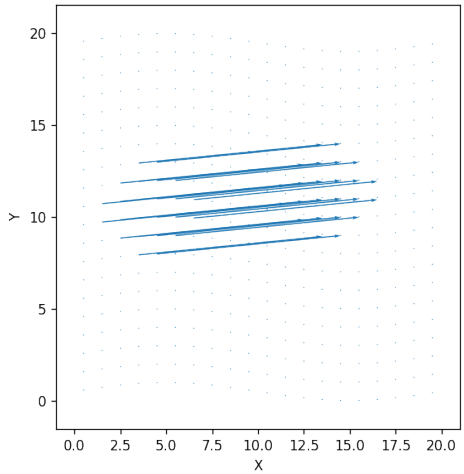

Figure 4.8.: Sinusoidal Mesh Implicit FVM Simulation Initial Velocity Field

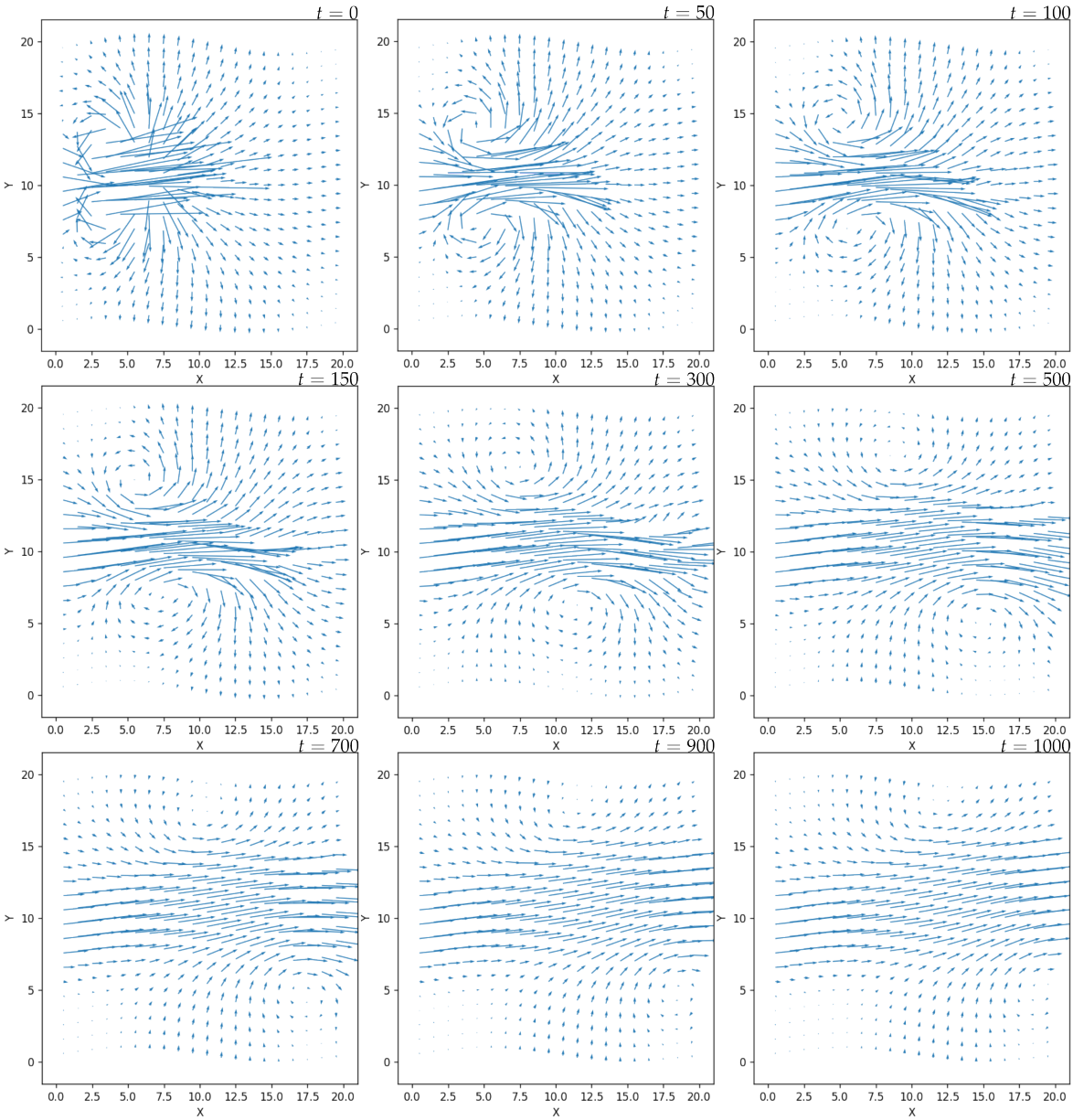

Figure 4.9.: Sinusoidal Mesh Implicit FVM Simulation Velocity Field Evolution

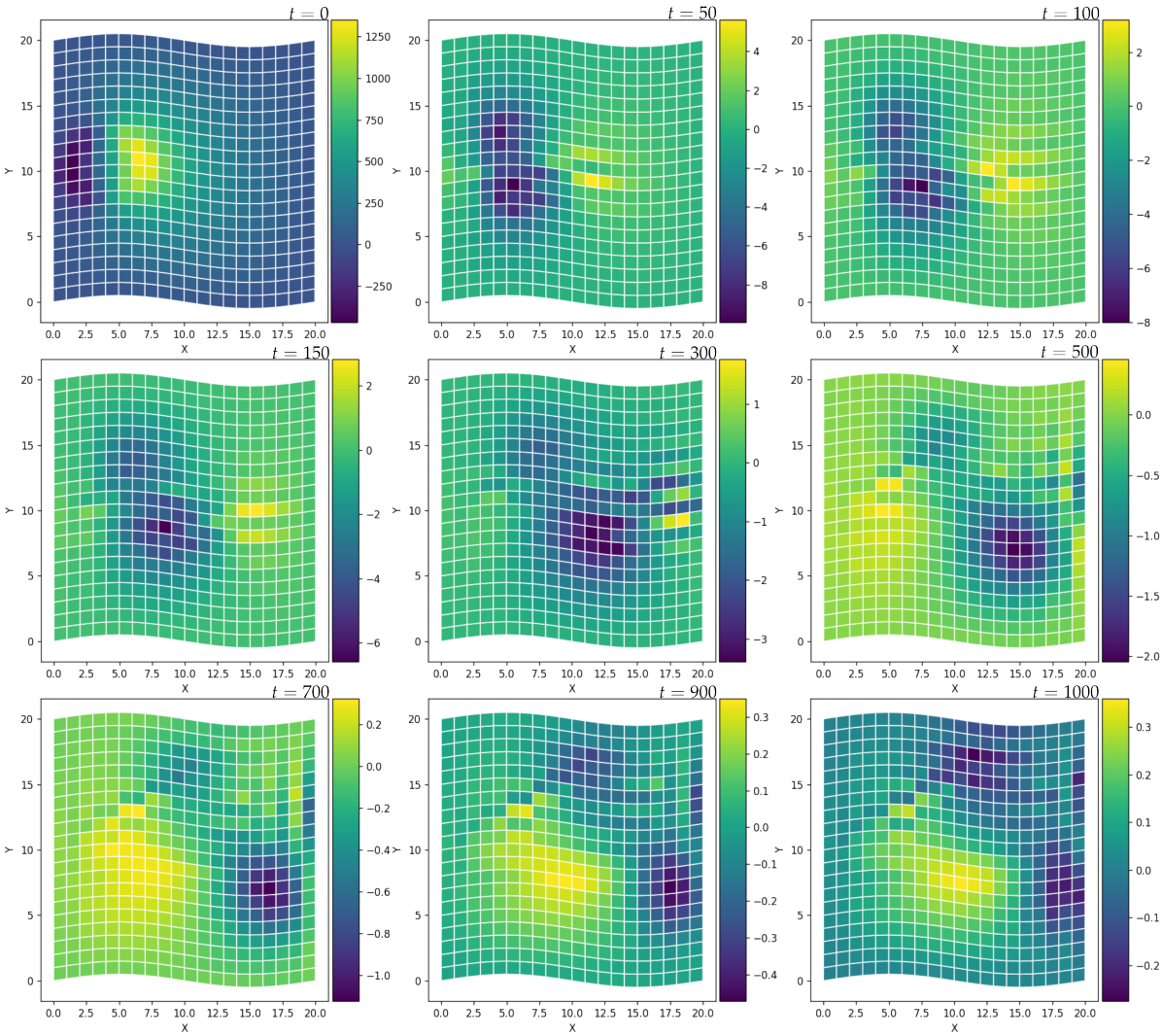

Figure 4.10.: Sinusoidal Mesh Implicit FVM Simulation Pressure Field Evolution

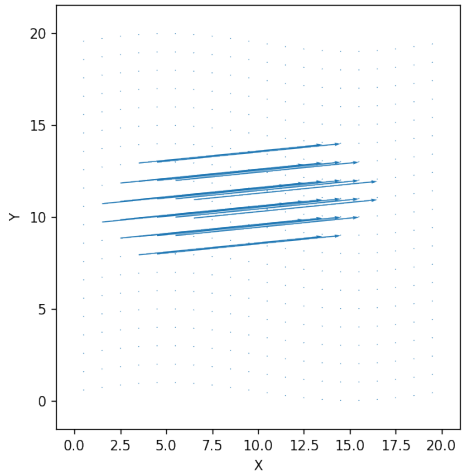

Figure 4.11.: Sinusoidal Mesh with Non-orthogonal Correction Implicit FVM Simulation Initial Velocity Field

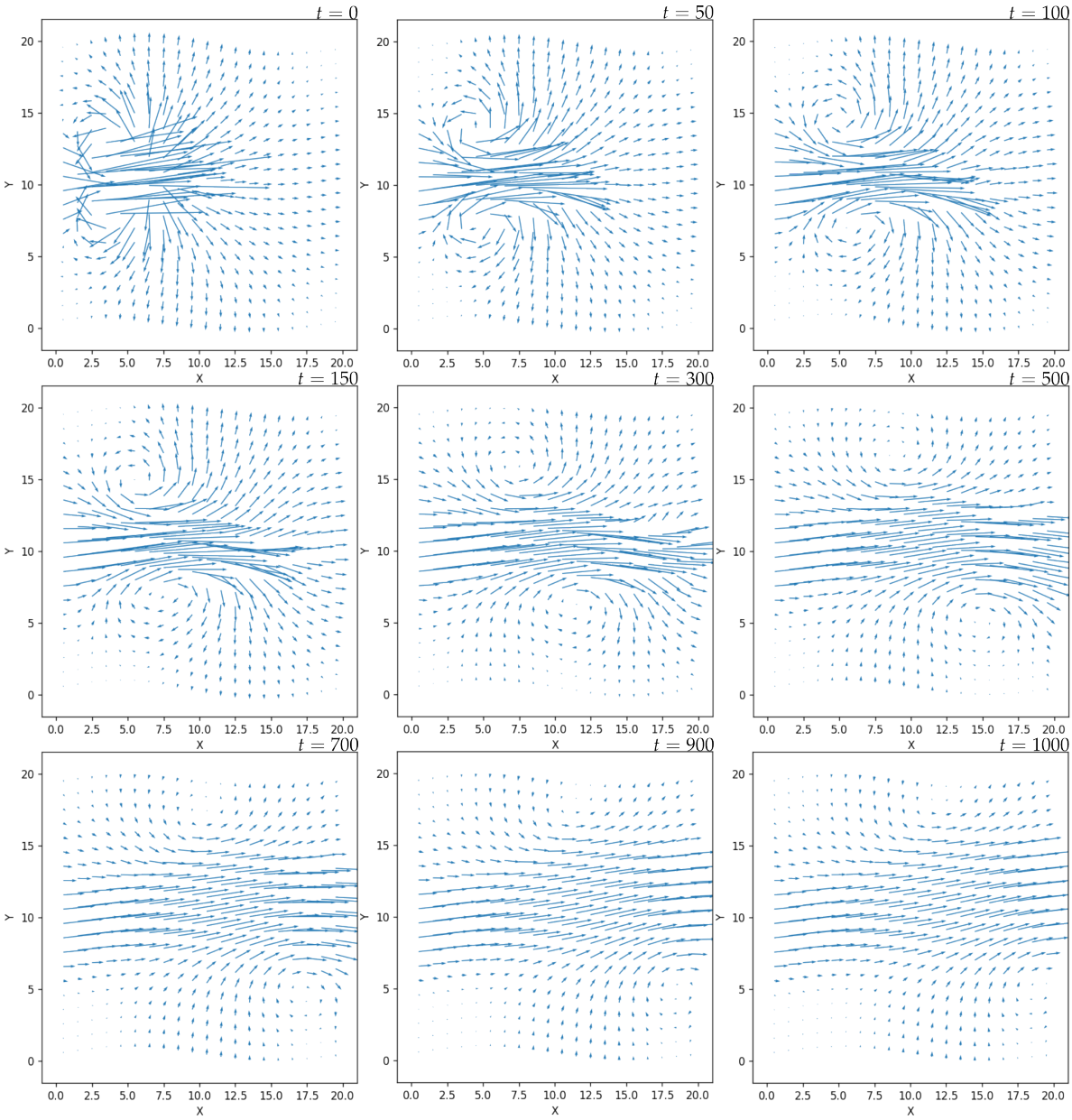

Figure 4.12.: Sinusoidal Mesh with Non-orthogonal Correction Implicit FVM Simulation Velocity Field Evolution

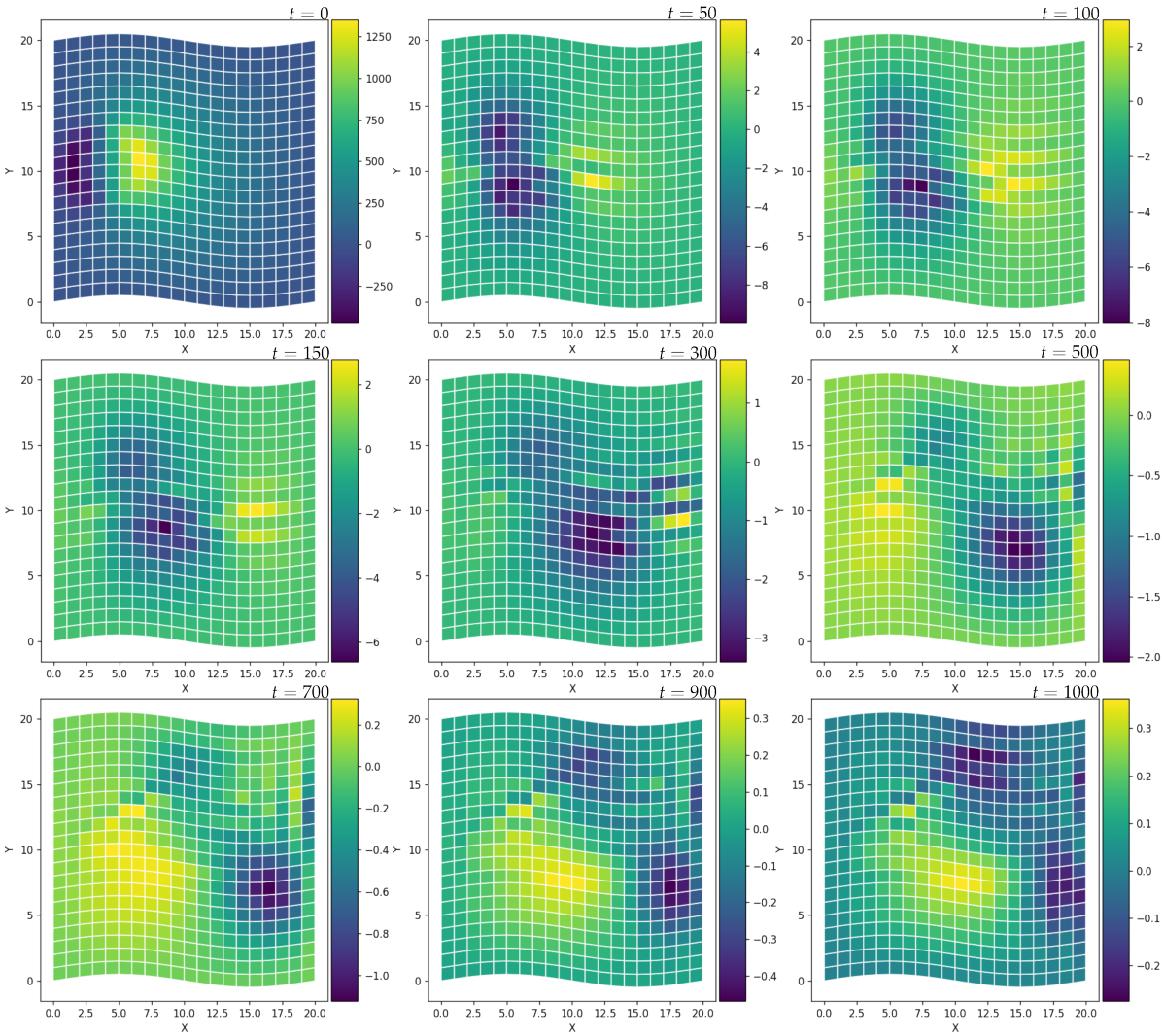

Figure 4.13.: Sinusoidal Mesh with Non-orthogonal Correction Implicit FVM Simulation Pressure Field Evolution

4.4. Implicit FVM Inflow Simulation

In this simulation, a cross shaped wave is initialized as shown in Figure 4.14 and Figure 4.17 using an unstructured mesh for the implicit FVM simulation. The initial conditions are respectively listed in table 4.6 and table 4.7. The only difference between the two simulations is that the latter simulation has a rectangular obstacle from $(x = 10, y = 6)$ to $(x = 10, y = 14)$ and the inflow boundary condition has a lower value as shown in table 4.7 compared to the inflow boundary condition specified in table 4.6. The boundary condition for the obstacle boundary is $(x = 0, y = 0)$ as there shouldn't be an inflow to or outflow from the obstacles.

The implicit FVM inflow simulation has the velocity and pressure field evolutions as shown

in Figure 4.15 and Figure 4.16 respectively. In comparison, the implicit FVM inflow with obstacles simulation has the velocity and pressure field evolutions as shown in Figure 4.18 and Figure 4.19 respectively.

For the given initial conditions, both methods eventually converge towards a solution. These simulations show that our FVM implementation can handle a variety of different conditions. We are able to accurately simulate an inflow condition with and without an obstacle. The advantage of FVM is that since the obstacles are not meshed, it is much easier for existing methods. There isn't a special handling condition like in FDM with centered grids. We keep using the same implementation as the underlying mesh takes care of whether there is an obstacle or not.

Table 4.6.: Simulation generating Figure 4.15 and Figure 4.16 has the shown initial conditions in the table and the velocity field shown in Figure 4.14.

Left Boundary $(x = 10, y = 0)$	Right Boundary Zero Gradient	Top Boundary $(x = 0, y = 0)$	Bottom Boundary $(x = 0, y = 0)$
Diffusivity 0.1	Δt 0.01	Number of Steps 1000	

Table 4.7.: Simulation generating Figure 4.18 and Figure 4.19 has the shown initial conditions in the table and the velocity field shown in Figure 4.17.

Left Boundary $(x = 3, y = 0)$	Right Boundary Zero Gradient	Top Boundary $(x = 0, y = 0)$	Bottom Boundary $(x = 0, y = 0)$
Obstacle Boundary $(x = o, y = 0)$	Diffusivity 0.1	Δt 0.01	Number of Steps 1000

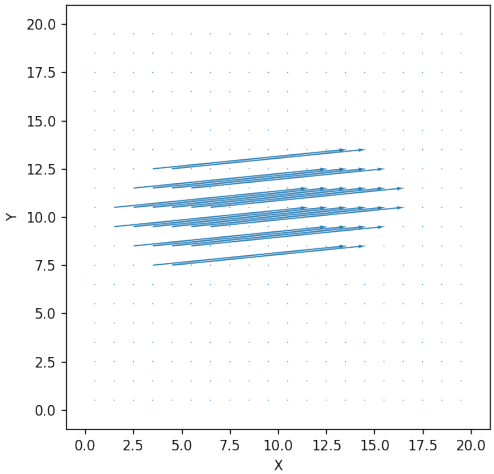

Figure 4.14.: Implicit FVM Inflow Simulation Initial Velocity Field

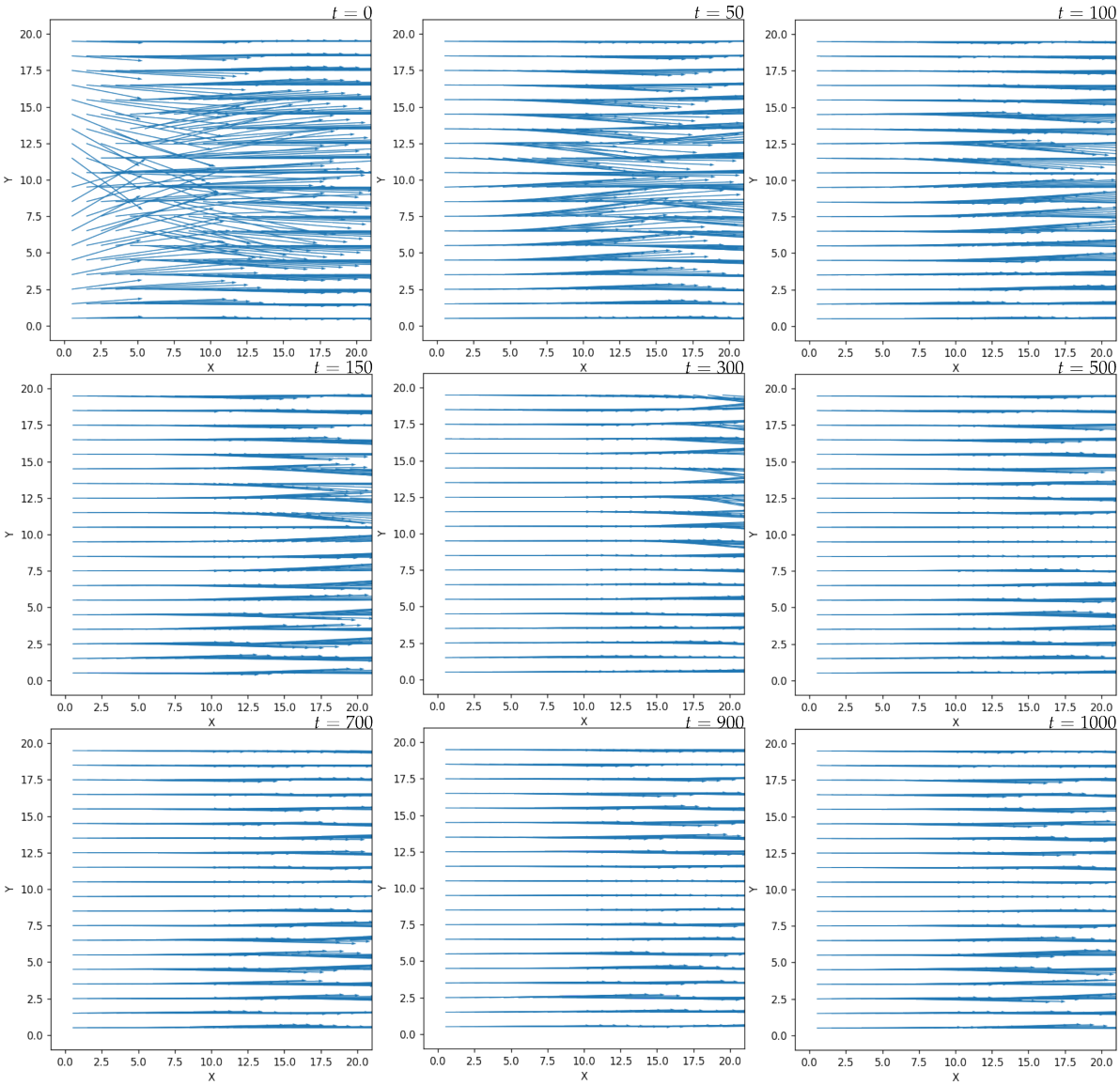

Figure 4.15.: Implicit FVM Inflow Simulation Velocity Field Evolution

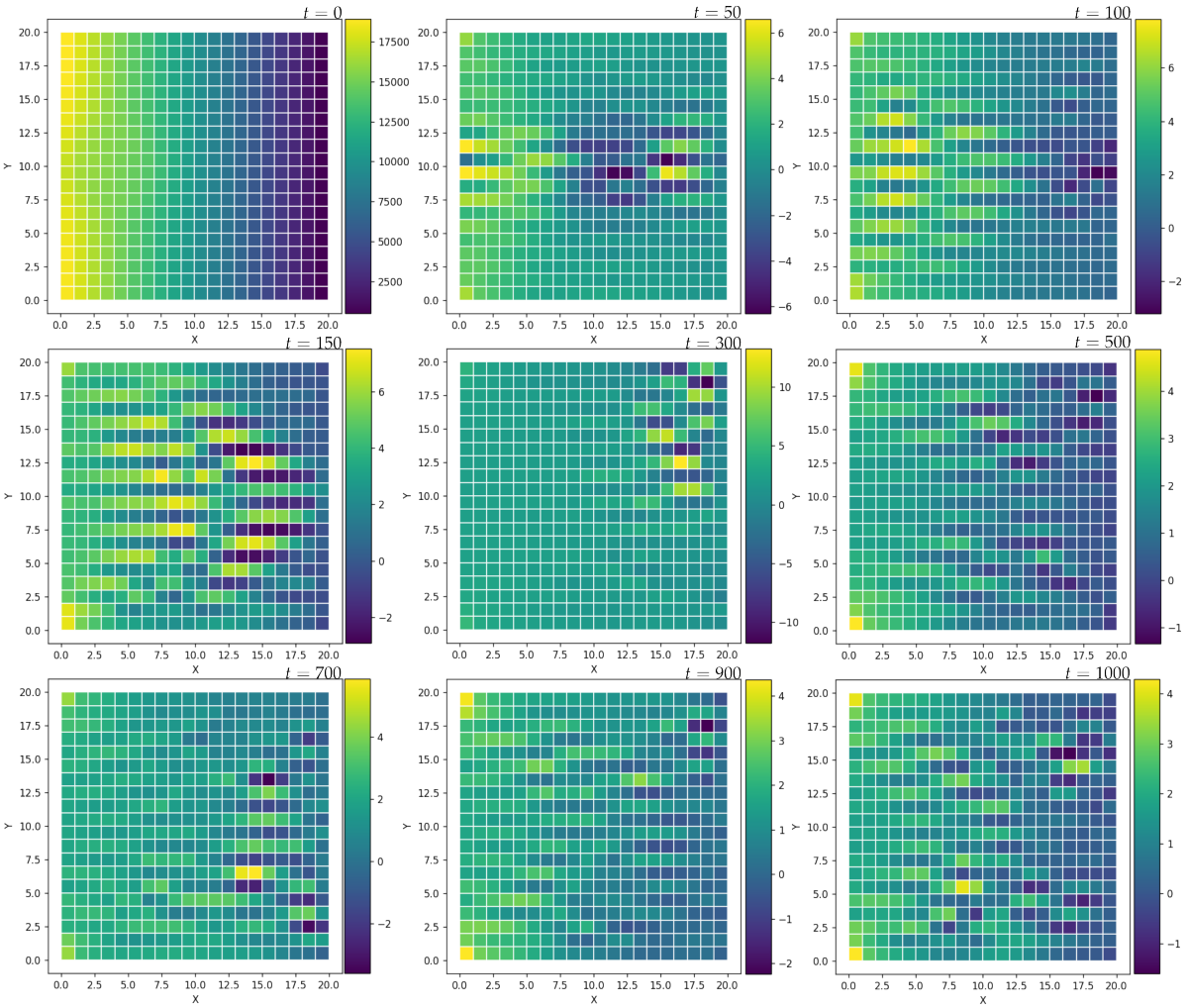

Figure 4.16.: Implicit FVM Inflow Simulation Pressure Field Evolution

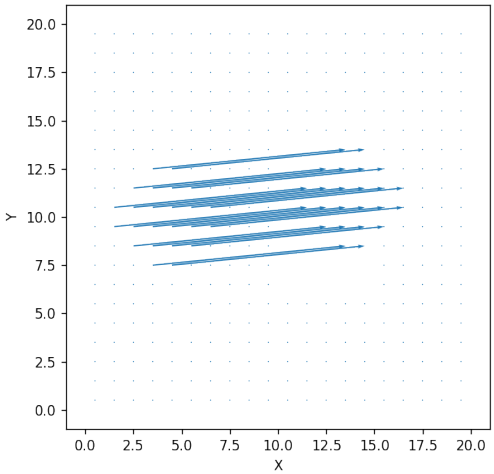

Figure 4.17.: Implicit FVM Inflow with Obstacles Simulation Initial Velocity Field

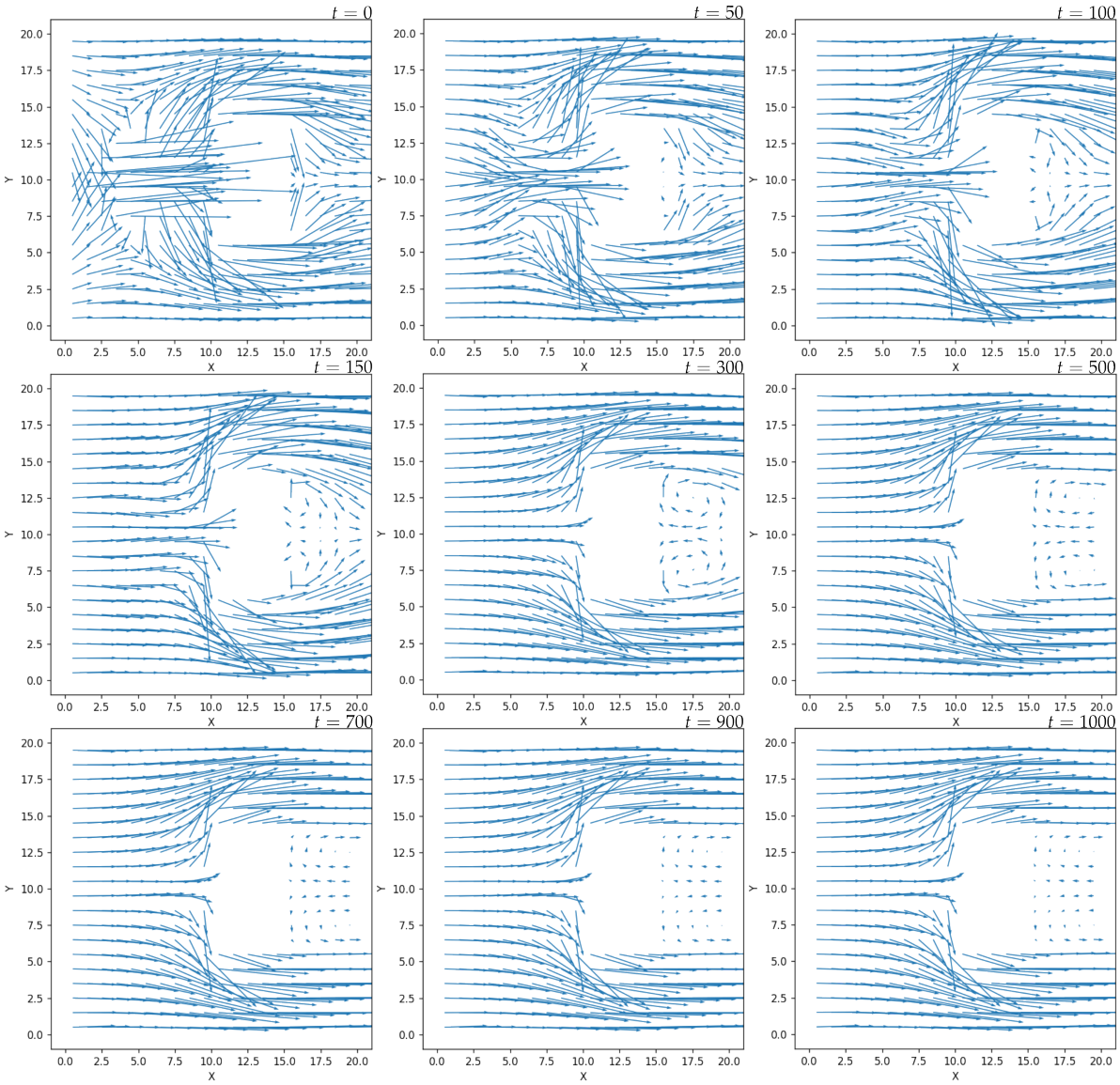

Figure 4.18.: Implicit FVM Inflow with Obstacles Simulation Velocity Field Evolution

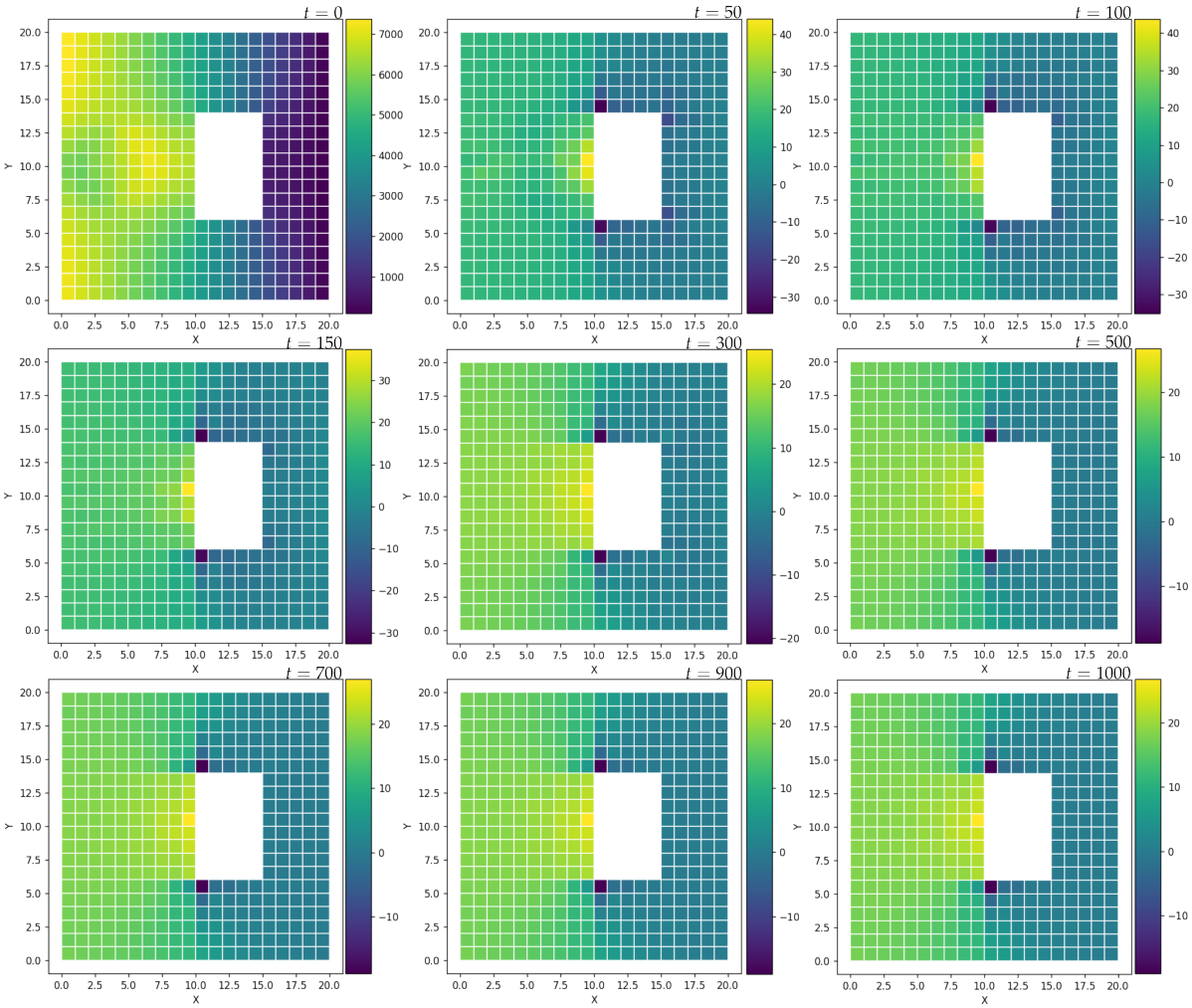

Figure 4.19.: Implicit FVM Inflow with Obstacles Simulation Pressure Field Evolution

4.5. Implicit FVM Simulation Using a Large Mesh

In this simulation, we test our implementation using a large $50x50$ unstructured mesh under different conditions.

The first simulation has the initial velocity field shown in Figure 4.20. There are multiple obstacles within the mesh and an inflow is simulated. Other initial conditions are listed in table 4.8. This simulation has the velocity and pressure field evolutions as shown in Figure 4.21 and Figure 4.22 respectively.

The second simulation has the initial velocity field shown in Figure 4.23. The initial conditions are listed in table 4.9. This simulation has the velocity and pressure field evolutions as shown in Figure 4.24 and Figure 4.25 respectively.

The third simulation has the initial velocity field shown in Figure 4.26. The initial conditions

are listed in table 4.10. This simulation has the velocity and pressure field evolutions as shown in Figure 4.27 and Figure 4.28 respectively.

All of the simulations converge towards a solution and show us that our FVM implementation works well with large meshes under different conditions as well.

Table 4.8.: Simulation generating Figure 4.21 and Figure 4.22 has the shown initial conditions in the table and the velocity field shown in Figure 4.20.

Left Boundary $(x = 3, y = 0)$	Right Boundary Zero Gradient	Top Boundary $(x = 0, y = 0)$	Bottom Boundary $(x = 0, y = 0)$
Obstacle Boundary $(x = o, y = 0)$	Diffusivity 0.1	Δt 0.1	Number of Steps 1000

Table 4.9.: Simulation generating Figure 4.24 and Figure 4.25 has the shown initial conditions in the table and the velocity field shown in Figure 4.23.

Left Boundary Zero Gradient	Right Boundary Zero Gradient	Top Boundary Zero Gradient	Bottom Boundary Zero Gradient
Diffusivity 0.1	Δt 0.1	Number of Steps 1000	

Table 4.10.: Simulation generating Figure 4.27 and Figure 4.28 has the shown initial conditions in the table and the velocity field shown in Figure 4.26.

Left Boundary Zero Gradient	Right Boundary Zero Gradient	Top Boundary Zero Gradient	Bottom Boundary Zero Gradient
Diffusivity 0.1	Δt 0.1	Number of Steps 1000	

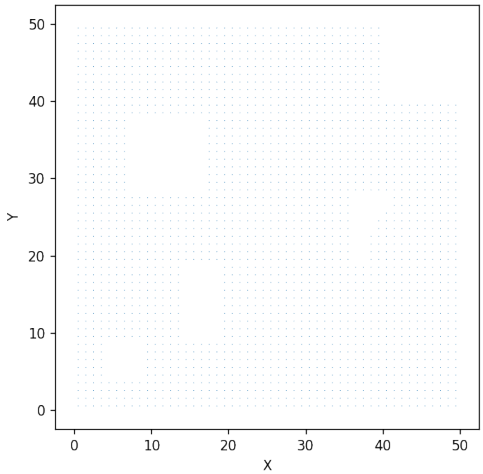

Figure 4.20.: Implicit FVM Inflow with Multiple Obstacles Large Mesh Simulation Initial Velocity Field

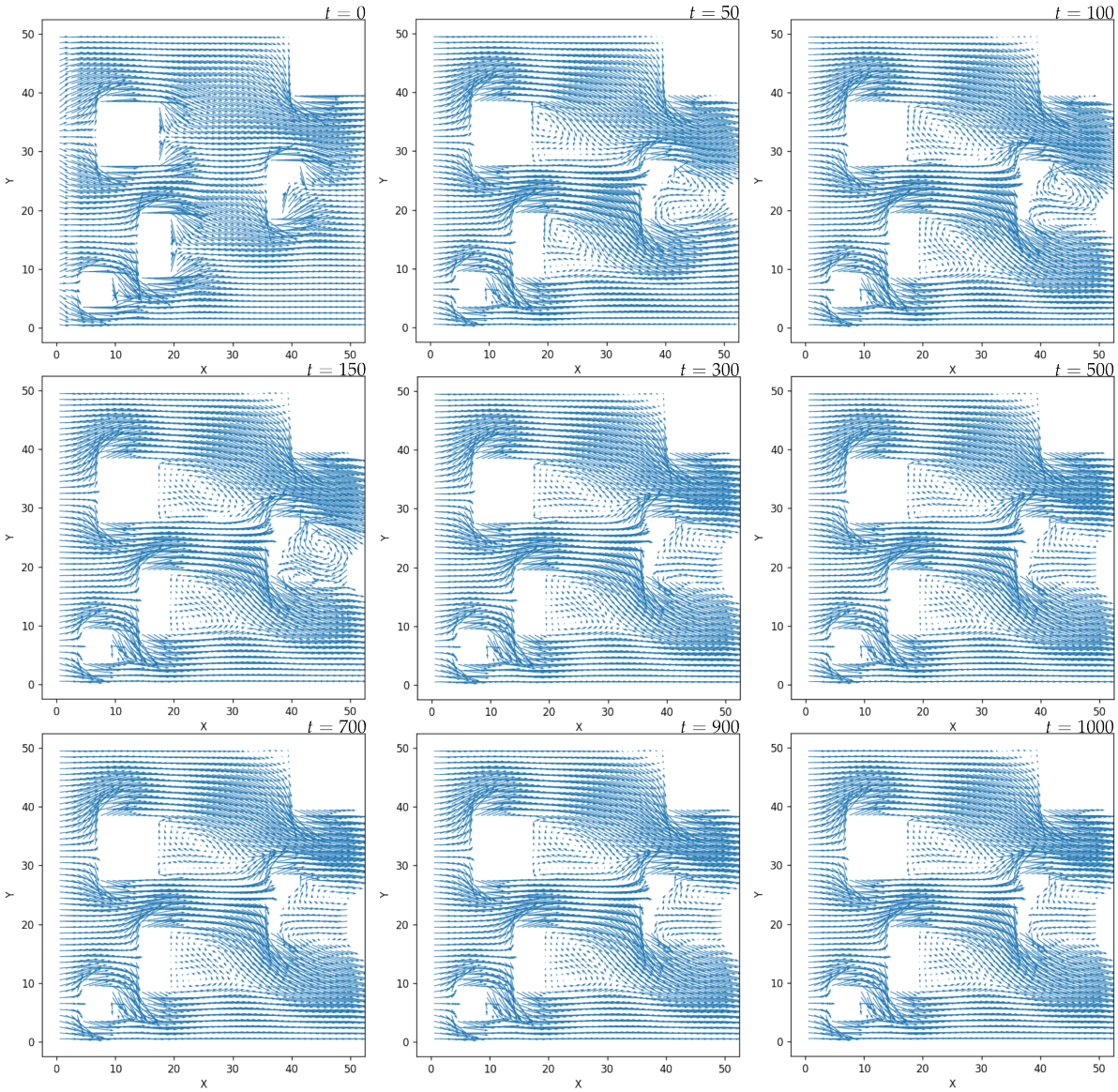

Figure 4.21.: Implicit FVM Inflow with Multiple Obstacles Large Mesh Simulation Velocity Field Evolution

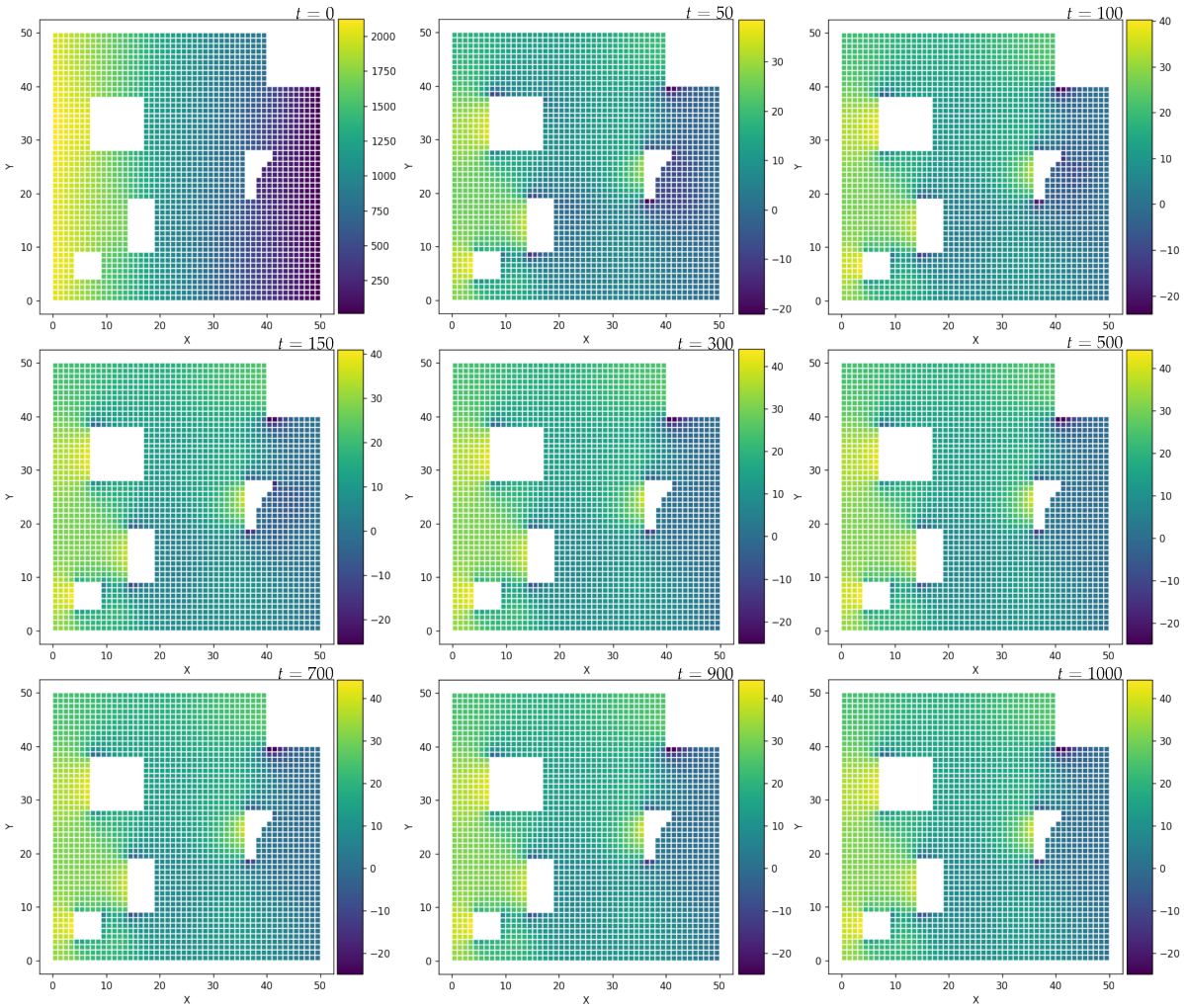

Figure 4.22.: Implicit FVM Inflow with Multiple Obstacles Large Mesh Simulation Pressure Field Evolution

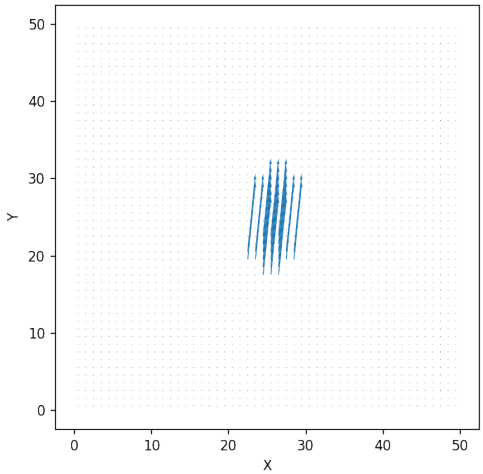

Figure 4.23.: Implicit FVM Small Wave Moving Up in a Large Mesh Simulation Initial Velocity Field

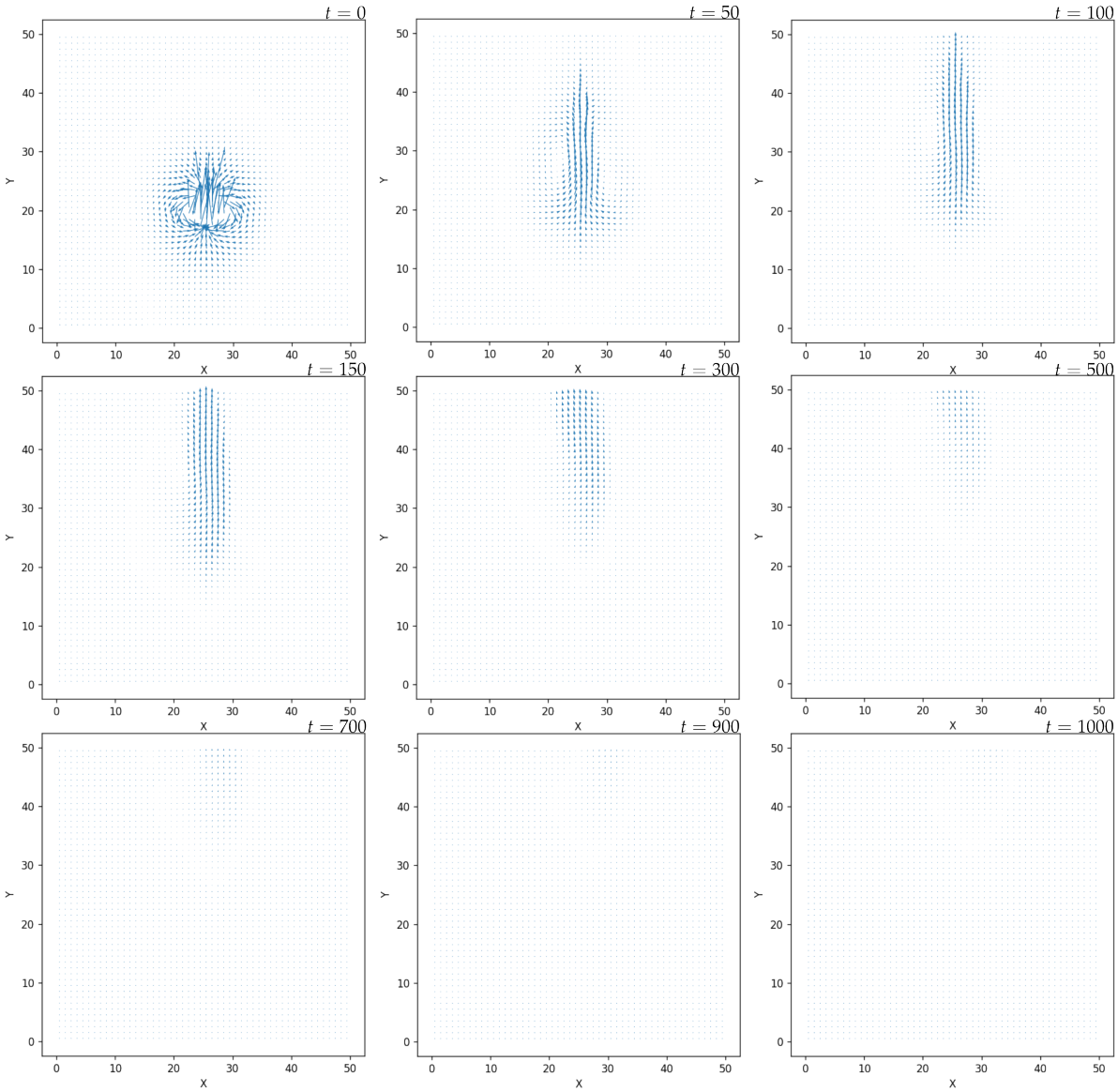

Figure 4.24.: Implicit FVM Small Wave Moving Up in a Large Mesh Simulation Velocity Field Evolution

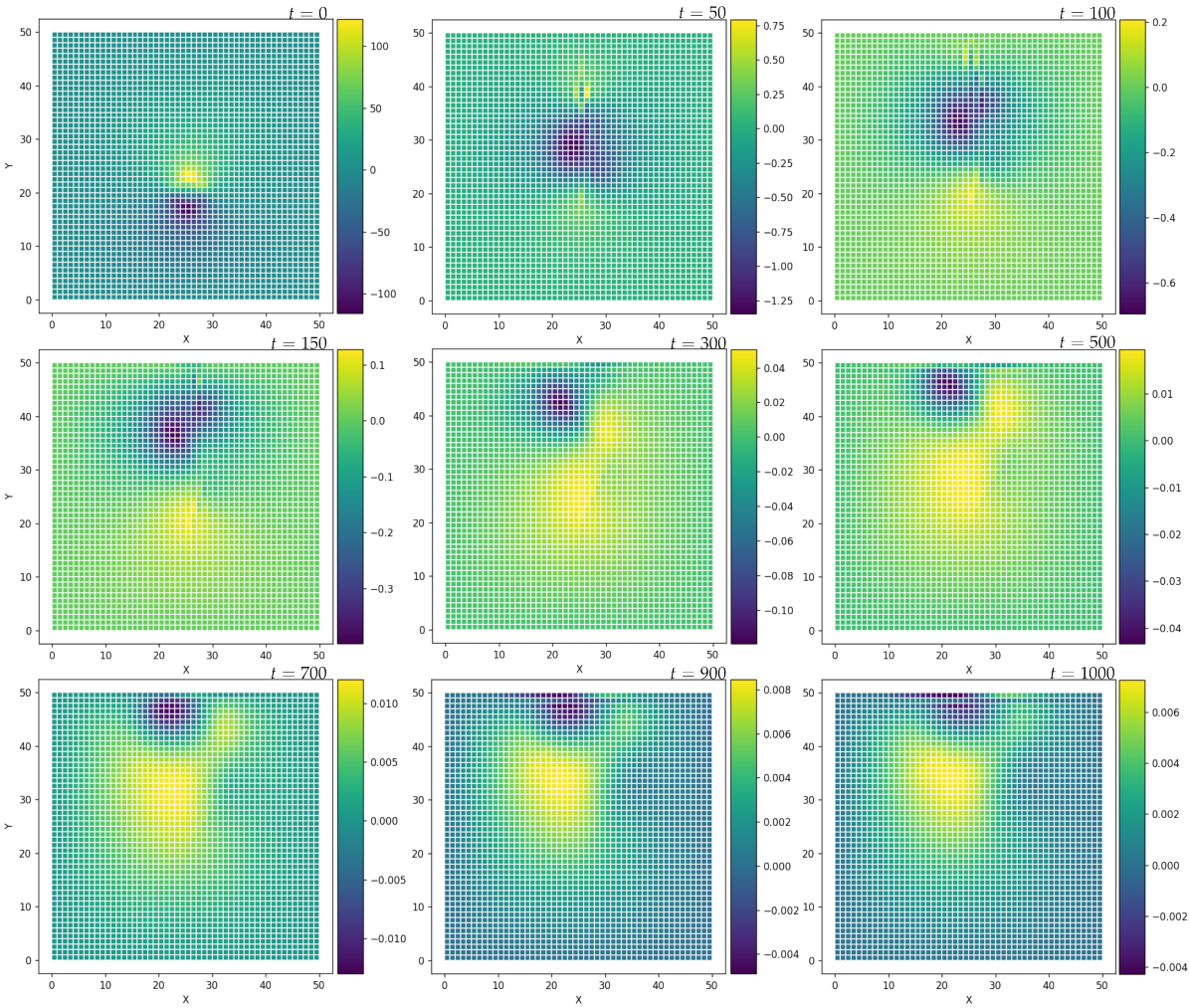

Figure 4.25.: Implicit FVM Small Wave Moving Up in a Large Mesh Simulation Pressure Field Evolution

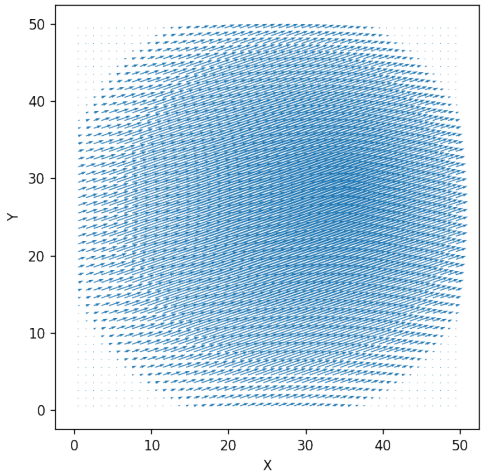

Figure 4.26.: Implicit FVM Decaying Value Wave in a Large Mesh Simulation Initial Velocity Field

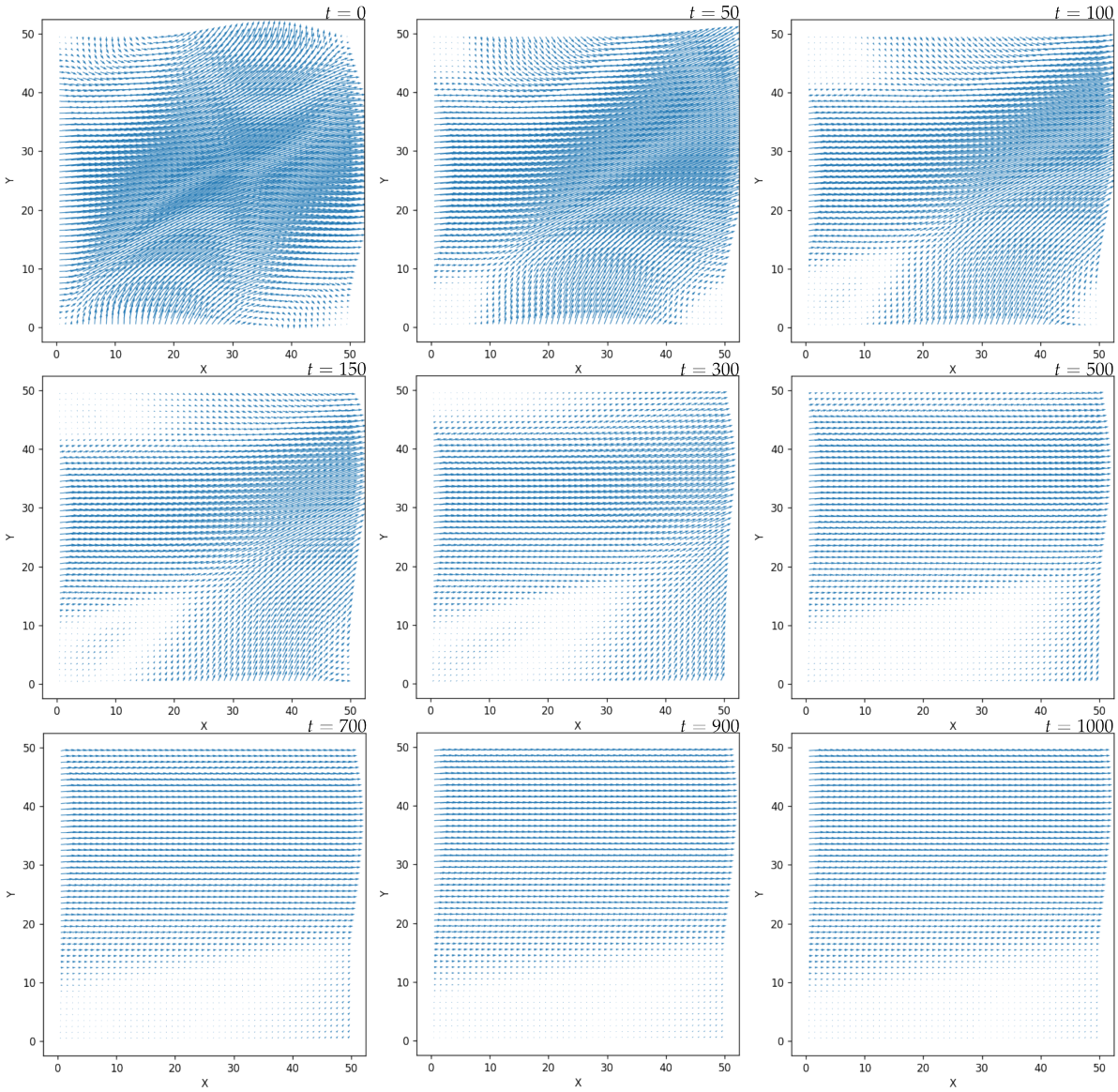

Figure 4.27.: Implicit FVM Decaying Value Wave in a Large Mesh Simulation Velocity Field Evolution

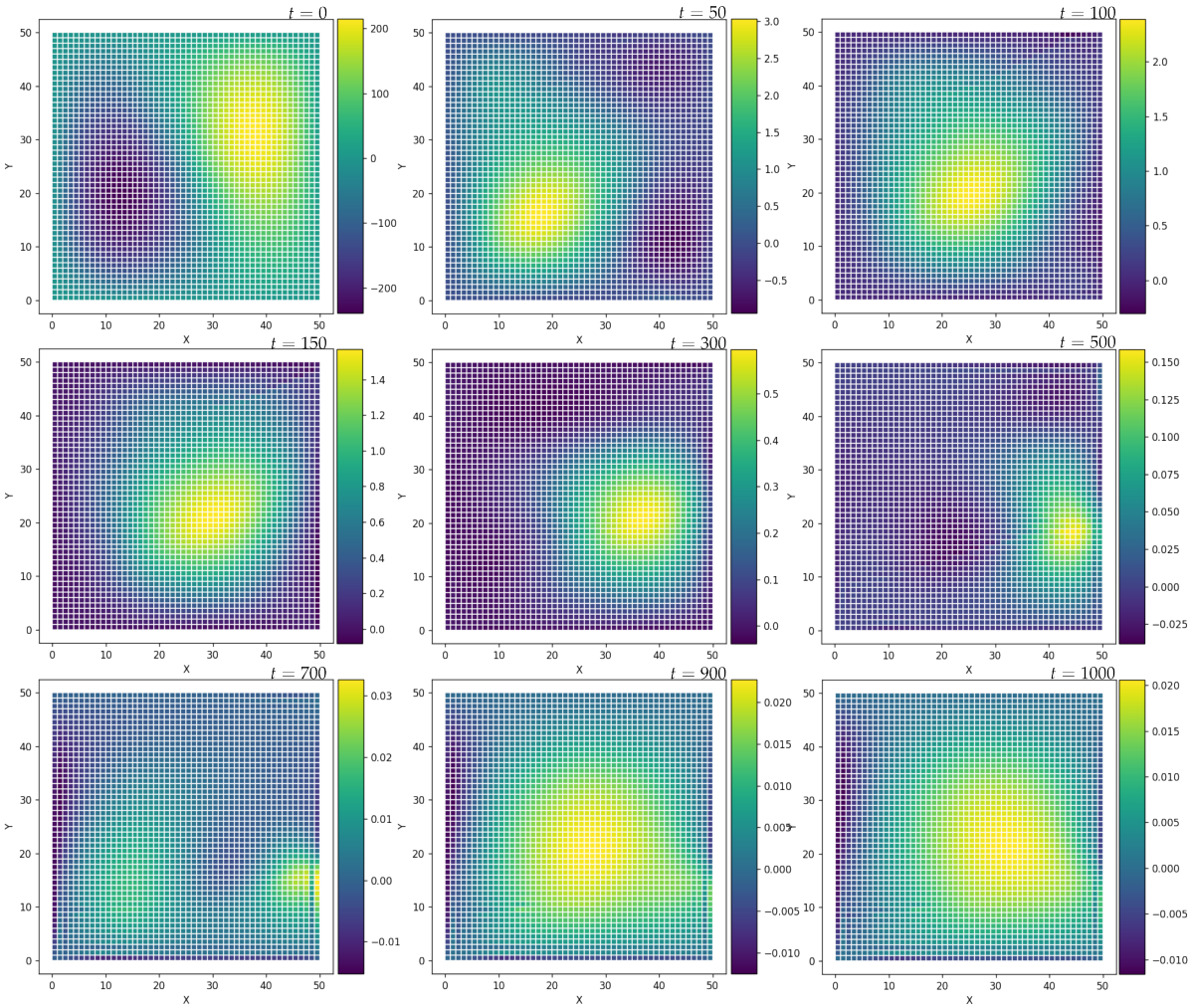

Figure 4.28.: Implicit FVM Decaying Value Wave in a Large Mesh Simulation Pressure Field Evolution

4.6. Implicit FVM Simulation with Obstacles

In this simulation, a cross shaped wave is initialized as shown in Figure 4.29 using an unstructured mesh for the FVM simulation and as shown in Figure 4.32 using a grid for the explicit FDM simulation. The initial conditions are respectively listed in table 4.11 and table 4.12.

The implicit FVM simulation has the velocity and pressure field evolutions as shown in Figure 4.30 and Figure 4.31 respectively. In comparison, the explicit FDM simulation has the velocity and pressure field evolutions as shown in Figure 4.33 and Figure 4.34 respectively. In addition to using an explicit scheme, the explicit FDM simulation also utilizes the Semi-Lagrangian method for the advection term. Both simulations have a rectangular obstacle from

$(x = 10, y = 6)$ to $(x = 10, y = 14)$

For the given initial conditions, both methods eventually converge towards a similar solution. Differences are to be expected due to the differences in discretization and underlying field structure for unstructured meshes and centered grids. The comparison shows that FVM using an unstructured mesh is indeed a viable option for CFD simulations.

Table 4.11.: Simulation generating Figure 4.30 and Figure 4.31 has the shown initial conditions in the table and the velocity field shown in Figure 4.29.

Left Boundary Zero Gradient	Right Boundary Zero Gradient	Top Boundary Zero Gradient	Bottom Boundary Zero Gradient
Obstacle Boundary $(x = 0, y = 0)$	Diffusivity 0.1	Δt 0.01	Number of Steps 1000

Table 4.12.: Simulation generating Figure 4.33 and Figure 4.34 has the shown initial conditions in the table and the velocity field shown in Figure 4.32.

Left Boundary Zero Gradient	Right Boundary Zero Gradient	Top Boundary Zero Gradient	Bottom Boundary Zero Gradient
Obstacle Boundary $(x = 0, y = 0)$	Diffusivity 0.1	Δt 0.01	Number of Steps 1000

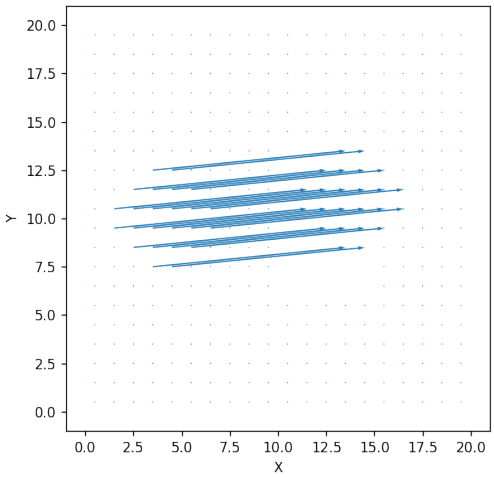

Figure 4.29.: Implicit FVM with Obstacles Simulation Initial Velocity Field

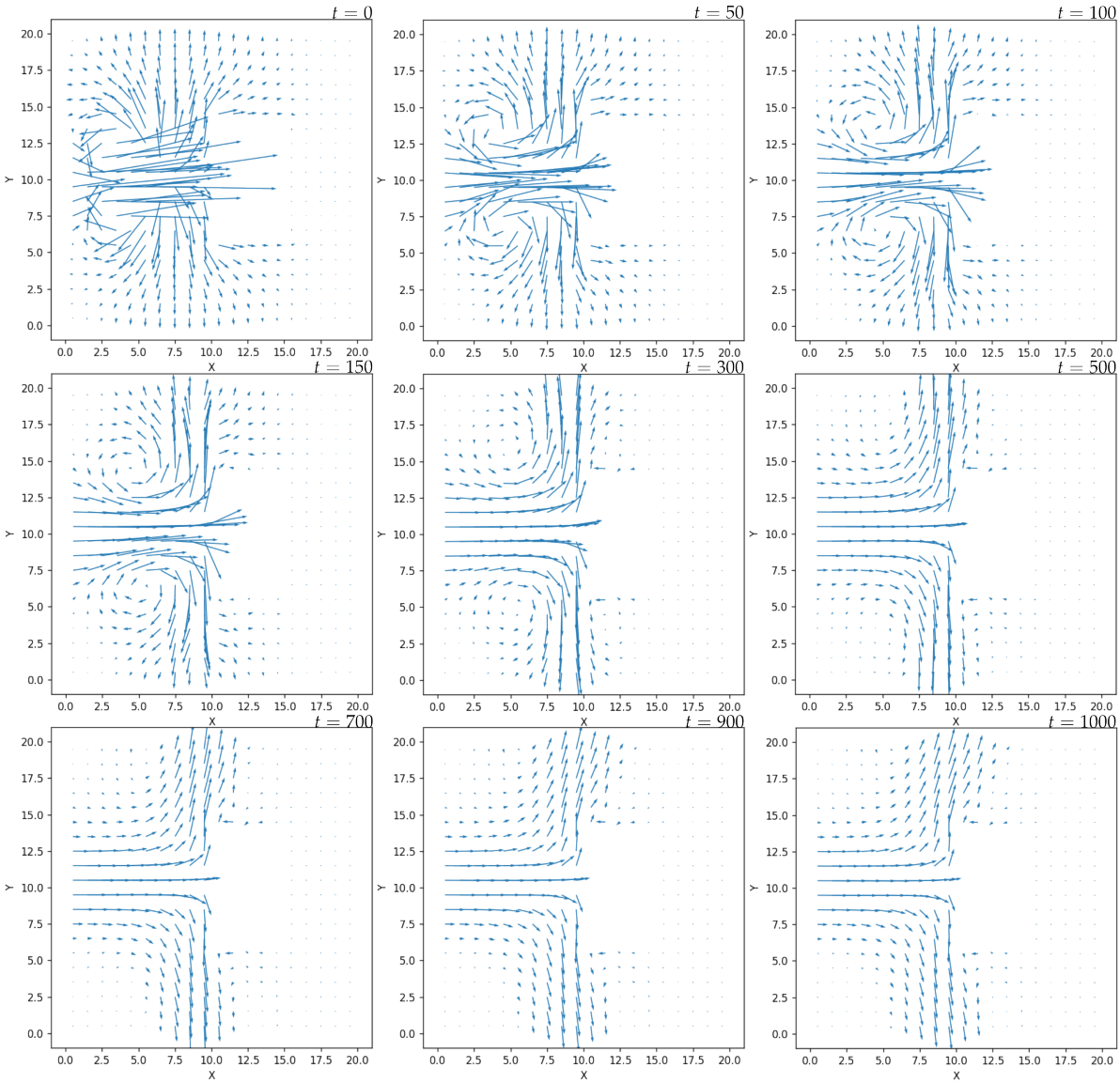

Figure 4.30.: Implicit FVM with Obstacles Simulation Velocity Field Evolution

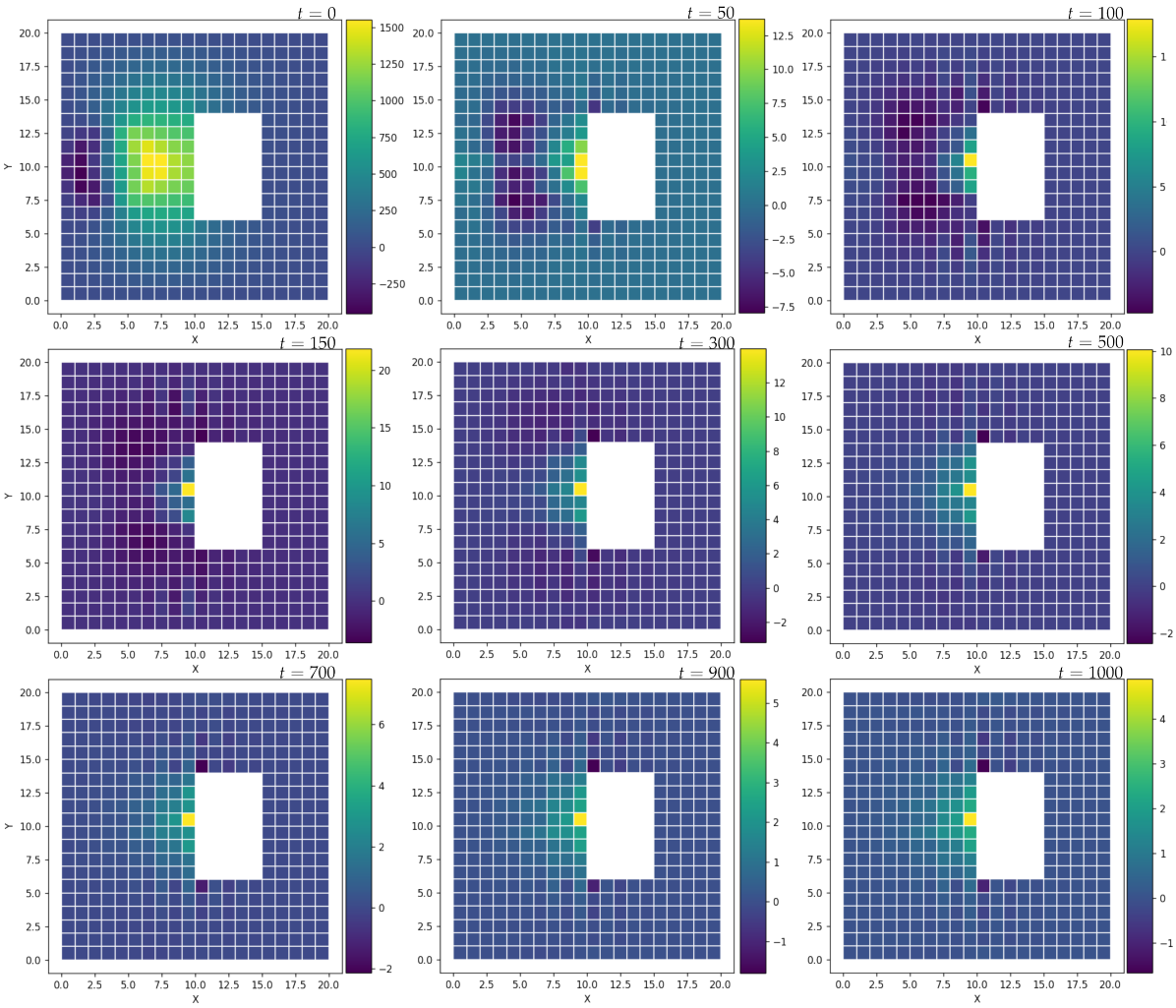

Figure 4.31.: Implicit FVM with Obstacles Simulation Pressure Field Evolution

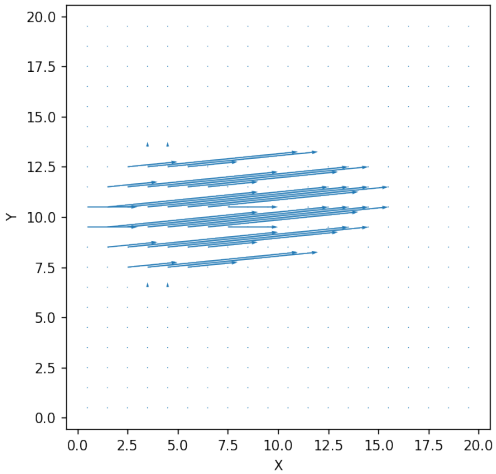

Figure 4.32.: Explicit FDM Simulation with Obstacles using a Grid and Split Step Using Semi-Lagrangian, Initial Velocity Field

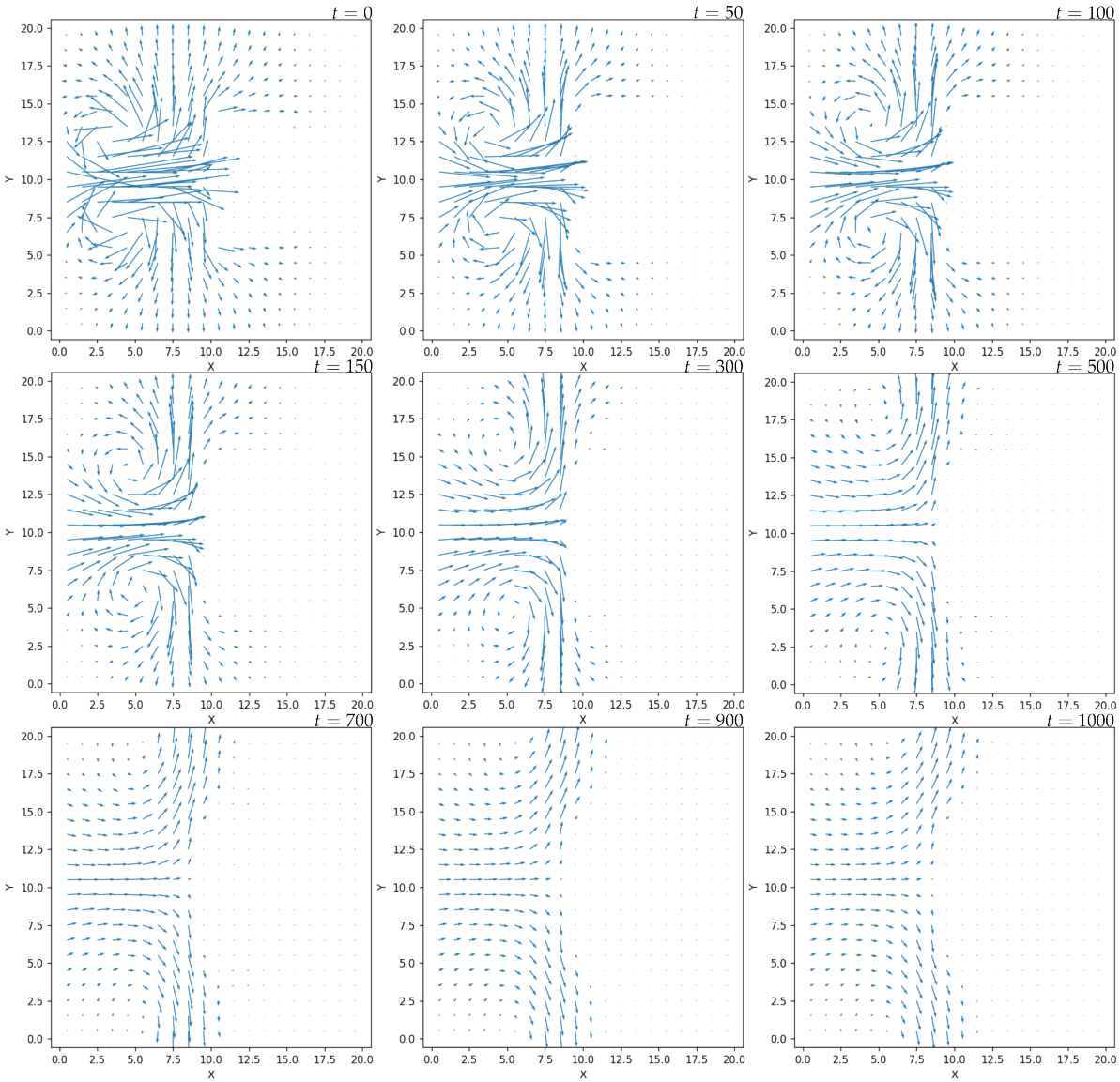

Figure 4.33.: Explicit FDM Simulation with Obstacles using a Grid and Split Step Using Semi-Lagrangian, Velocity Field Evolution

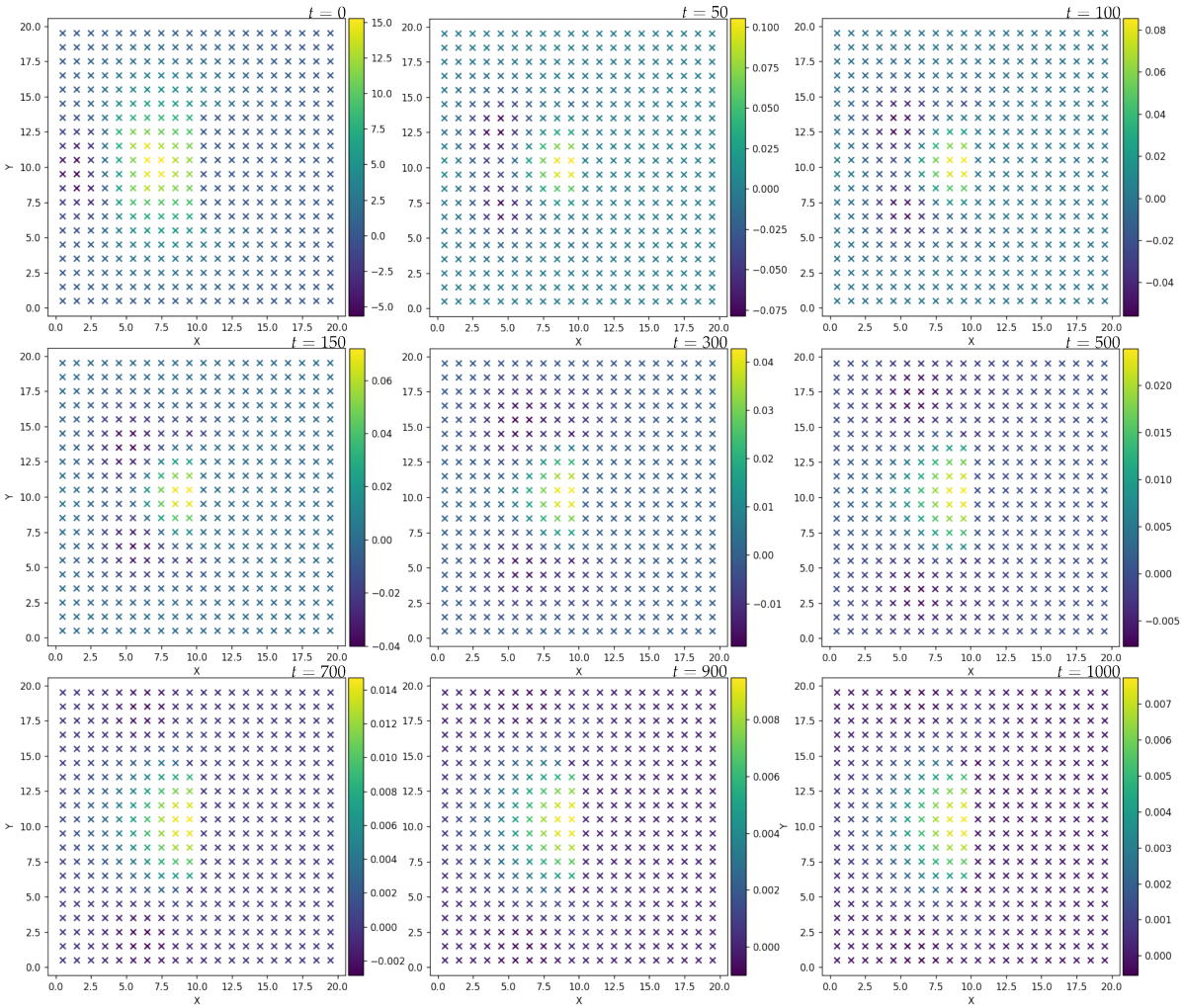

Figure 4.34.: Explicit FDM Simulation with Obstacles using a Grid and Split Step Using Semi-Lagrangian, Pressure Field Evolution

5. Conclusion

This thesis has presented a detailed investigation into the numerical solutions of the incompressible Navier-Stokes equations using the finite volume method. We addressed the significant challenges posed by the discretization process and the pressure-velocity coupling inherent in the incompressible formulation of these equations. The projection method, introduced by Chorin [2], was a pivotal approach in decoupling these variables, thereby enabling stable and accurate fluid flow simulations.

We demonstrated the discretization process for FVM by partitioning the computational domain into small control volumes using the key idea to integrate the governing equations over each control volume in order to ensure the conservation of fluxes across the boundaries of these volumes. The study examined both explicit and implicit methods for the discretization of the Navier-Stokes equations, with a particular emphasis on the backward Euler method. The differentiation between the advection and convection terms was elucidated, clarifying their respective roles in fluid dynamics simulations. Special schemes such as the upwind scheme and non-orthogonal correction, that are needed to discretize the advection and diffusion terms respectively to ensure overall stability, were described in great length.

Through the projection method, we derived and implemented the Poisson equation for the pressure field, ensuring the incompressibility constraint was met at each time step. This derivation underscored the critical role of the Poisson equation in maintaining numerical stability.

Our discussion on the Courant-Friedrichs-Lewy (CFL) condition highlighted its importance in selecting appropriate time steps to ensure numerical stability.

A significant focus of this thesis was the capability of the finite volume method to manage complex geometries and non-uniform meshes, which are crucial for accurately simulating real-world fluid dynamics scenarios. We explored the incorporation of obstacles within the computational domain and their impact on mesh and solution stability.

The implementation of the finite volume method within the Φ_{Flow} [4] computational framework provided practical examples and validation of the numerical solutions against benchmark problems. When it was possible, by comparing solutions generated by the finite volume method to those produced by the finite difference method under identical initial conditions, we demonstrated the effectiveness and robustness of the finite volume method. The results affirm the utility of the projection method and the finite volume method in solving the incompressible Navier-Stokes equations, offering valuable insights into their application for various engineering and scientific challenges.

In conclusion, this thesis contributes to the understanding and implementation of finite volume method for fluid dynamics, providing a solid foundation for future work and improvements in the simulation of incompressible flows.

A. Appendix

A.1. Gauss's Divergence Theorem

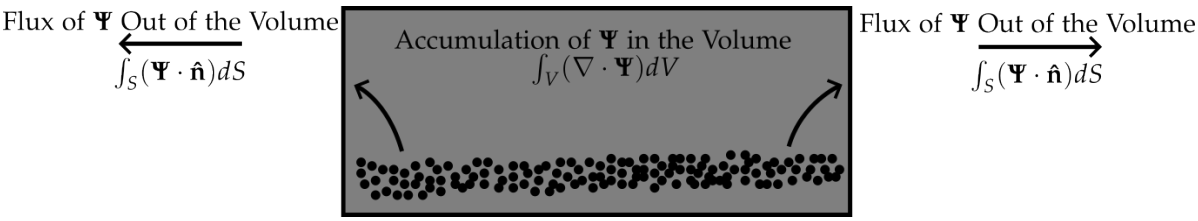

Figure A.1.: A visualization of the Gauss' Divergence Theorem which says that the accumulation of $\mathbf{\Psi}$ in the volume must be equal to the flux of $\mathbf{\Psi}$ across the faces of the volume.

Gauss's divergence theorem says that the conservation laws must be obeyed. If the amount of $\mathbf{\Psi}$ in a cell is increasing, it must mean that either there is $\mathbf{\Psi}$ entering through the faces of the cell or that there is a source of $\mathbf{\Psi}$ in the cell that is creating $\mathbf{\Psi}$ [8, page 9]. Thus, Gauss' divergence theorem states that accumulation of $\mathbf{\Psi}$ in the volume must be equal to the flux of $\mathbf{\Psi}$ out of the volume as seen in equation 73, where $\mathbf{\Psi}$ is a general vector [16].

$$\int_V(\nabla\cdot\mathbf{\Psi})dV = \int_S(\mathbf{\Psi}\cdot\hat{\boldsymbol{n}})dS \tag{73}$$

A.2. Linear Interpolation

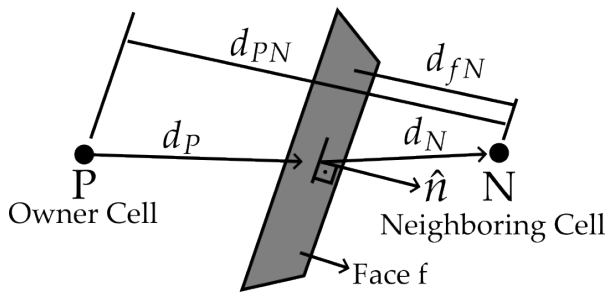

Figure A.2.: $\mathbf{\Psi}_f$ is calculated by linear interpolation using the values $\mathbf{\Psi}_P$ and $\mathbf{\Psi}_N$ at the adjacent cells.

Since in the finite volume method many of discretized terms result in calculating the fluxes at the cell faces to get the contribution of that term to the flow variable $\mathbf{\Psi}$ for the respective cell, interpolation from the cell centroids to cell faces are very common. The most popular method to do is linear interpolation [1, pages 71-73].

The linear interpolation formulation to calculate the cell face center values $\mathbf{\Psi}_f$ of the flow variable $\mathbf{\Psi}$ is done by using the linear variation between the cell centroid values $\mathbf{\Psi}_P$ and $\mathbf{\Psi}_N$ corresponding to the cell values of the owner cell and neighbor cell. The linear interpolation formula is generalized by defining a weights field w for each face using equation 74. The weights are calculated using the distances from the cell face center to the adjacent cell centers in the direction normal to the face i.e. the cell center distance is projected to the unit normal vector of the face as seen in equation 74. The formula used for this interpolation is given by equation 75 [1, pages 71-73].

$$w = \frac{d_{fN}}{d_{PN}} = \frac{\mathbf{n} \cdot \mathbf{d}_N}{\mathbf{n} \cdot (\mathbf{d}_P + \mathbf{d}_N)} \tag{74}$$

$$\mathbf{\Psi}_f = w\mathbf{\Psi}_P + (1 - w)\mathbf{\Psi}_N \tag{75}$$

A.3. Divergence of a Point

$$\nabla \cdot \mathbf{U} = \lim_{\Delta V \to 0} \frac{1}{\Delta V} \int_S (d\mathbf{S} \cdot \mathbf{U}) \tag{76}$$

To define divergence at a specific *point*, we examine the limit as the volume approaches to zero. For a surface S that encloses a volume ΔV, divergence is the flux through the surface per unit volume, as $\Delta V \to 0$ such as in equation 76 [1, pages 13-75].

Bibliography

[1] C. Greenshields and H. Weller. *Notes on Computational Fluid Dynamics: General Principles*. Reading, UK: CFD Direct Ltd, 2022.

[2] A. J. Chorin. "A Numerical Method for Solving Incompressible Viscous Flow Problems". In: *Journal of Computational Physics* 135 (1997). Article No. CP975716, pp. 118–125. DOI: 10.1006/jcph.1997.5716.

[3] Wikipedia contributors. *Backward Euler Method — Wikipedia, The Free Encyclopedia*. Accessed: 2024-07-15]. 2024. URL: https://en.wikipedia.org/wiki/Backward_Euler_method.

[4] P. Holl and N. Thuerey. "PhiFlow: Differentiable Simulations for PyTorch, TensorFlow and Jax". In: *International Conference on Machine Learning*. PMLR. 2024.

[5] R. Eymard, T. Gallouët, and R. Herbin. "Finite Volume Methods". In: *Handbook of Numerical Analysis*. Ed. by P. Ciarlet and J. Lions. Vol. 7. https://hal.science/hal-02100732v2. Reading, UK: Elsevier, 2000, pp. 713–1020. DOI: 10.1016/S1570-8659(00)07005-8.

[6] Wikipedia contributors. *Finite Difference Method — Wikipedia, The Free Encyclopedia*. Accessed: 2024-07-23. 2023. URL: https://en.wikipedia.org/wiki/Finite_difference_method.

[7] S. Bilbao. "Modeling of Complex Geometries and Boundary Conditions in Finite Difference/Finite Volume Time Domain Room Acoustics Simulation". In: *IEEE Transactions on Audio, Speech, and Language Processing* 21.7 (2013), pp. 1524–1533. DOI: 10.1109/TASL.2013.2256897. URL: https://www.research.ed.ac.uk/en/publications/modeling-of-complex-geometries-and-boundary-conditions-in-finite-.

[8] A. Wimshurst. *Fluid Mechanics 101 Fundamentals Course 3*. London, UK, 2022.

[9] W. contributors. *Euler Method — Wikipedia, The Free Encyclopedia*. Accessed: 2024-07-15]. 2024. URL: https://en.wikipedia.org/wiki/Euler_method.

[10] I. Simulations. *Courant Number — Ideal Simulations, CFD Resources*. Accessed: 2024-07-16. 2024. URL: https://www.idealsimulations.com/resources/courant-number-cfd/.

[11] Wikipedia. *Courant–Friedrichs–Lewy Condition — Wikipedia, The Free Encyclopedia*. Accessed: 2024-07-16. 2024. URL: https://en.wikipedia.org/wiki/Courant%E2%80%93Friedrichs%E2%80%93Lewy_condition.

[12] F. M. 101. *The Finite Volume Method in CFD*. https://www.youtube.com/watch?v=E9_kyXjtRHc&list=PLnJ8lIgfDbkp5DtCPtP2rcqEEUJk-PM8N. Accessed: 2024-07-16. 2024.

[13] F. M. 101. *[CFD] What is the difference between Upwind, Linear Upwind and Central Differencing?* https://youtu.be/JVE0fNkc540?si=7gSauLWlmX-RQW_C. Accessed: 2024-07-17. 2024.

[14] F. M. 101. *[CFD] What is Mesh Non-Orthogonality?* https://www.youtube.com/watch?v=Dg2PwX6yxYY&list=PLnJ8lIgfDbkp5DtCPtP2rcqEEUJk-PM8N&index=6. Accessed: 2024-07-16. 2024.

[15] W. contributors. *Projection method (fluid dynamics) — Wikipedia, The Free Encyclopedia*. Accessed: 2024-07-18. 2024. URL: https://en.wikipedia.org/wiki/Projection_method_(fluid_dynamics).

[16] Wikipedia. *Divergence Theorem — Wikipedia, The Free Encyclopedia*. Accessed: 2024-07-17]. 2024. URL: https://en.wikipedia.org/wiki/Divergence_theorem.