PARALLEL MULTIPHASE NAVIER-STOKES SOLVER

A Dissertation

by

FAHAD SALEH A. ALRASHED

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Jean-Luc Guermond
Committee Members,	Vivek Sarin
	Wolfgang Bangerth
	Andrea Bonito
	Regis Vilagines
Head of Department,	Emil Straube

May 2015

Major Subject: Mathematics

ABSTRACT

We study and implement methods to solve the variable density Navier-Stokes equations. More specifically, we study the transport equation with the level set method and the momentum equation using two methods: the projection method and the artificial compressibility method. This is done with the aim of numerically simulating multiphase fluid flow in gravity oil-water-gas separator vessels. The result of the implementation is the parallel ASPEN software framework based on the massively parallel `deal.II` .

For the transport equation, we briefly discuss the theory behind it and several techniques to stabilize it, especially the graph laplacian artificial viscosity with higher order elements. Also, we introduce the level set method to model the multiphase flow and study ways to maintain a sharp surface in between phases.

For the momentum equation, we give an overview of the two methods and discuss a new projection method with variable time stepping that is second order in time. Then we discuss the new third order in time artificial compressiblity method and present variable density version of it. We also provide a stability proof for the discrete implicit varaible density artificial compressibility method.

For all the methods we introduce, we conduct numerical experiments for verification, convergence rates, as well as realistic models.

ACKNOWLEDGEMENTS

I could not have completed a feat of this magnitude alone. There are so many people that, one way or another, helped me get to this point. I thank all of them profusely. I am forever in your debt and I apologies in advance for forgetting to mention you.

First and foremost, I am indebted to my parents who nudged me toward the person I am right now. My wife Mashael and family, their support and love, has been the bedrock that I could rely on to finish my Ph.D. No words can describe my gratitude.

I would like to thank my advisor, J.-L. Guermond, for his patience with a disoriented first year student with little math background. His continued guidance, encouragement, and discussions made me who I am today.

A whole new world opened up to me when W. Bangerth introduced me to `deal.II`. His meticulous work has propelled the quality of my work by orders of magnitude. On top of that, his soft spoken guidance and encouragement were indispensable. The interesting insights, guidance and conversations I had from my committee members V. Sarin, A. Bonito, and R. Vilagines were crucial to my education. All of that was facilitated by a great learning environment provided by the Department of Mathematics at Texas A&M University.

Last but not least, I would like to thank my friends and colleagues who made this journey towards completing my studies enjoyable and making so many ordinary moments, extraordinary.

I would also like to thank Saudi Aramco for the sponsorship and support during my Ph.D.

NOMENCLATURE

$\boldsymbol{u}\|_K$	$\boldsymbol{u}$ restricted to cell K.
$\boldsymbol{u}\|_{\partial\Omega}$	$\boldsymbol{u}$ restricted to the boundary of domain Ω.
$\nu := \ldots$	define the value of ν.
$\nu = \mu$	ν and μ have the same value.
$\nabla\phi$	$:= \left(\frac{\partial\phi}{\partial x_1}, \frac{\partial\phi}{\partial x_2}, \ \ldots\ , \frac{\partial\phi}{\partial x_d}\right)^\top$.
$\mathrm{div}(\mathbf{f})$	$:= \sum_{i=1}^{d} \frac{\partial \mathbf{f}_i}{\partial x_i}$.
$\Delta\phi$	$:= \mathrm{div}(\nabla\phi) = \sum_{i=1}^{d} \frac{\partial^2\phi}{\partial x_i^2}$.
$\nabla^s\mathbf{f}$	$:= \frac{1}{2}\left(\nabla\mathbf{f} + \nabla^\top\mathbf{f}\right)$.
$(v,u)_\Omega$	$:= \int_\Omega vu\ dx$, where $u, v : \Omega \mapsto \mathbb{R}^d$
$L^0(\Omega)$	$\{u$ measurable $\mid \mathrm{meas}\{\lvert u\rvert > \lambda\} < \infty\ \forall\lambda > 0\}$.
$L^p(\Omega)$	$\{u \mid (\int_\Omega \lvert u\rvert^p)^{\frac{1}{p}} < \infty\}$.
$L^\infty(\Omega)$	$\{u \mid \max_{\boldsymbol{x}\in\Omega} \lvert u(x)\rvert < \infty\}$.
$L^p_{\mathrm{loc}}(\Omega)$	$\{u \mid u\rvert_K \in L^p(K)\ \forall K \subset \Omega, K$ compact$\}$.
$D^\alpha f$	$:= \frac{\partial^{\lvert\alpha\rvert} f}{\partial^{\alpha_1} x_1 \ldots\ \partial^{\alpha_n} x_n}$, e.g. $D^{1,2,0} f = \frac{\partial^3 f}{\partial x \partial^2 y}$ if f is a function in 3D.
$W^{s,p}(\Omega)$	$\{u \mid D^\alpha u \in L^p(\Omega)\ \forall\lvert\alpha\rvert \le s, \alpha_i \ge 0\}$.
$W^{s,p}_{\mathrm{loc}}(\Omega)$	$\{u \mid D^\alpha u \in L^p_{\mathrm{loc}}(\Omega)\ \forall\lvert\alpha\rvert \le s, \alpha_i \ge 0\}$.
$H^p(\Omega)$	$W^{1,p}(\Omega)$.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

In life, we strive to understand the world around us. We see water flow in rivers and how beautiful and serene it is. On the other hand, we see wind howling in hurricanes and how fatally dangerous it can become. Both are essentially physical fluid flows like many in nature and – more specifically – fluid flows in industrial plants and pipes in many systems around us and the industry. We are interested in maximizing (or minimizing) a property of these systems so that they run optimally. To achieve that, one needs to ''tinker'' with the physical system and hope that the optimal state is reached. This can be prohibitively expensive, time consuming and simply impractical. The better approach is to build mathematical models of such flows, build software that simulates the systems using these models, match the control variables to the physical system and then be able to predict how such systems behave. Equipped with that, now one can use algorithms to optimize any aspect of the physical systems. The mathematical model that describes such motion of fluids is called the *Navier-Stokes equations*.

1.1 What are the Navier-Stokes Equations?

The *incompressible Navier-Stokes equations* are defined as follows:

$$\partial_t \rho + \mathrm{div}(\rho \boldsymbol{u}) = 0, \qquad \text{in } \Omega \times (0,T], \tag{1.1}$$

$$\rho[\partial_t \boldsymbol{u} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u}] - 2\mu \mathrm{div}(\nabla^s \boldsymbol{u}) + \nabla p = \rho \mathbf{f}, \qquad \text{in } \Omega \times (0,T], \tag{1.2}$$

$$\mathrm{div}(\boldsymbol{u}) = 0, \qquad \text{in } \Omega \times (0,T], \tag{1.3}$$

where $\rho, \boldsymbol{u}, p$ are the density, velocity, and pressure respectively. μ is the dynamic viscosity and $\mathbf{f}$ is the driving force. We will elaborate on (1.1)-(1.3) throughout the upcoming chapters.

The Navier-Stokes equations are *conservation laws*. They conserve *mass, momentum*, and *energy* in a system of partial differential equations (PDE). The equations have been known for quite a long time (Navier 1823, Stokes 1845) but did not have many practical uses except in very narrow cases. With the advent of computers, these equations have risen in importance and a lot of research in the twentieth century was dedicated to them and still continues today.

In this dissertation, we will describe the Navier-Stokes equations in depth and explain two different approaches to solving them. Then, we will propose an extension of the new "Artificial Compressibility" method and investigate its stability. This will be done in three chapters. In the chapter 3, we will introduce the transport equation which handles the mass conservation. After that, we will describe the constant density momentum equation in one chapter and the variable density momentum equation in the next; both conserve the momentum. In all the chapters, we will present the equations, the discretization in both space and time, the linear systems produced and the different numerical schemes to stabilize and solve them. Finally, we present numerical results and figures. A more detailed overview is presented in section 1.4.

1.2 Commercial CFD Software Packages

To motivate the next section, we will give an overview of the current Computational Fluid Dynamics (CFD) software packages. There are many open and commercial software solve the Navier-Stokes equations. Over the past few decades, the market has seen a surge of CFD software packages that has grown in complexity

and utility. This has been accompanied with exceptional progress in researching new and improved numerical methods that address simple and complex flow problems. These software packages offer myriad tools for automatic mesh generation, error control mechanisms, efficient parallel iterative solvers, etc. These are the preferred features for engineering firms and project design institutes. However, those software packages have offered limited extensibility for research environments. Moreover, intellectual property (IP) is paramount to commercial companies and they would require access to the code of CFD software to be able to expand its functionality in house with their own methods or have a contract with CFD companies. However, CFD companies have been reluctant to build software with other's IP and usually refuse such fundamental customizations. Consequently, building custom software has become an attractive option to some commercial CFD users.

1.3 The Saudi Aramco Project

In this dissertation, we will develop the methods used in the CFD software project that was started in Texas A&M Department of Mathematics for Saudi Aramco – the National Oil Company of Saudi Arabia. Aramco needs a customized simulation software that:

- simulates multiphase fluid flow. Specifically, oil; water; and gas mixtures flowing through crude oil production facilities,
- is high performance and massively-parallel (> 1000 processors),
- comes with the source code, and
- is fully customizable and extendable.

As shown in the previous section, these requirements are difficult to achieve by buying an off-the-shelf commercial CFD package.

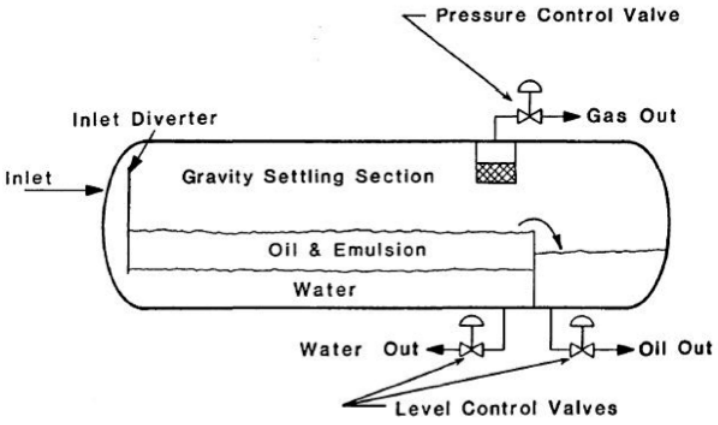


Figure 1.1: Schematic view of a gravity separation vessel found at Gas Oil Separation Plants (GOSP).

The software will be used initially to simulate fluid flows in Gas Oil Separation Plants (GOSP) (see figures 1.1 and 1.2). They are plants that receive feed from the oil wells - which typically contains oil, water, and gas mixed together. Starting at one end of the plant, the mixture is allowed some time to settle and gravity does its work and separates the gas near the beginning of the plant. The oil and water start separating one-thirds of the way in the plant. Two-thirds the way in, a weir (short dam) allows the top layer of oil to spill over to the other side. At the same time, two pipes underneath the plant extract the separated oil and water and a pipe above the GOSP extracts the gas.

1.4 Overview

The main contributions of this dissertation are as follows: built the massively parallel CFD framework ASPEN, extended a second order projection method for the

Figure 1.2: Gas-Oil Separation Plants (GOSP) facility at Murjan, Saudi Arabia. (courtesy: www.poonglim.co.kr).

Navier-Stokes equation to use variable time stepping, extended the constant density artificial compressibility method for the Navier-Stokes equation to variable density, and proved partial stability results using a simplified form of Navier-Stokes equations. There are also numerous contributions to the `deal.II` library for performance enhancement and bug fixing.

There are 4 main chapters in this dissertation, Chapter 2 discusses the Aspen framework and parallelism. Each of the next 3 chapters address one self-contained subset of the Navier-Stokes equations: the transport equation in chapter 3, constant-density momentum equation in chapter 4, and chapter 5 with the variable-density momentum equation.

In chapter 3, we will explore the transport equation and briefly talk about the existence and uniqueness of its solution in the literature. Then, we describe the numerical methods to discretize in space and time and continue on to higher-order

methods. After that, we discuss the need for stabilization in the transport equation and describe several methods, old and new, to achieve that. That leads to the discussion of linear systems constructed based on the equation and how to solve them in parallel. We finally conclude the chapter with numerical results and convergence tests.

Chapter 4 discusses the constant-density momentum equation. Like chapter 2, we present the model and shortly discuss its existence and uniqueness. Then present an issue posed by the incompressibility condition and how to address it. This then leads us to present three time discretization methods: the Uzawa method, Projection method, and Artificial Compressibility method. Then we will discuss the linear systems that surface from the equation and how to solve them in parallel. Also, we conclude the chapter with numerical results and convergence tests.

Finally, chapter 5 discusses the variable-density momentum equation. It will build upon chapter 4 and investigates the stability of the variable-density system of equations. The two methods, the projection and artificial compressibility, in chapter 4 are extended to variable-density. We also prove continuous and implicit discrete stability results for a simplified Navier-Stokes equations. We finally test the methods with numerical results and convergence tests.

2. ASPEN FRAMEWORK

In the next few chapters, we will get into the mathematical details of the incompressible Navier-Stokes equations. In this chapter, however, we will discuss the framework ASPEN - an acronym for Advanced Solver for multiPhase fluid sEparatioN - that we built to solve the incompressible Navier-Stokes equations. This framework was built with the following principles in mind:

- Usability and Extendability: To enhance usability, ASPEN allows plugable geometries, boundary conditions, initial conditions, etc. This extensibility allows others to modify ASPEN with as little change to the core functionality as possible. Also, the core Navier-Stones equations solver is split into separate classes following the best practice of minimizing coupling and increasing cohesion (c.f. Stevens et al. [73]).

- Massively parallel: As we will discuss in the next section, ASPEN is built from the ground up with massive parallelism in mind: scaling to 1000's of processors. This is because nowadays higher performance can only be achieved by running on multiple processors in parallel, each having a portion of the problem. (`deal.II`, the underlying finite element library is capable of handling billions of degrees of freedom [8]).

- Build on other's work: To push the boundaries of our knowledge, we need to "stand on the shoulders of giant." Thus, we built ASPEN on strong foundational libraries such as `deal.II`, `PETSc`, `p4est`(Bangerth et al. [6], Balay et al. [4], Burstedde et al. [12] respectively), etc. These libraries are regularly maintained, well documented and regularly tested. Taking advantage of these libraries is

the sensible approach.

Many of the properties above are borrowed from ASPECT (c.f. Kronbichler et al. [50], Bangerth et al. [7]). The idea behind the choice of principles is to be able to reuse other people's code to push the boundaries of knowledge. To rebuild what has already been built well is a waste of time and effort. A better approach is to modify existing frameworks - preferably by building a reasonably isolated plugin - to achieve the desired goals. However, the barrier to understanding frameworks is quite high, and many resort to building code from scratch. This makes building well designed, usable, and accessible frameworks a necessity in this day and age.

2.1 Motivation

Microprocessor technology started in the late 1970s and dominated the market of computers. This was due to the processors' ability to take advantage of enhancements in intergraded circuit (IC) technology. As a result, single processor performance has increased at a rate of 52% per year up until 2003 (c.f Hennessy and Patterson [45]). However, due to the ''maximum power dissipation'' limit of air cooled processors and a dearth of instruction-level parallelization that can be efficiently found, the performance increase has been less than 22% since 2003. Clock speeds also have hardly changed since 2003 with an average increase of 1% per year. Consequently, Intel, one of the largest manufacturers of microprocessors, abandoned higher performance single processor plans in 2004 and declared, with other companies, that the road ahead in high performance is through multicore chips [45]. As a result, higher performance can only be achieved by explicitly writing software with parallelization in mind.

Parallelization is based on the concept of splitting computations, as much as possible, into smaller tasks that run in parallel. Some of the popular technologies

used to achieve that are MPI [57], OpenMP [20], and Threading Building Blocks (TBB) [65]. They are the basis technology for splitting and aggregating data among the tasks to achieve the needed results. Here, we will mainly concentrate on MPI .

Many of the computations required for our code have already been parallelized efficiently. Our task will be to continue that and build upon the parallelized building blocks and fine-tune the higher level constructs. For example, although many well known solvers and preconditioners are already parallelized efficiently, there are many parameters to tune for the specific matrix being inverted (e.g. Hypre's BoomerAMG, see Henson and Yang [46]). Also, assembling the linear system matrix consumes a sizable portion of execution time and exploiting the processors' cache and the matrix's properties such as symmetry often lead to jumps in parallel performance.

The code developed during this research is based on deal.II (Bangerth et al. [6, 5]), an open source intrinsically parallel finite element framework written in C++. The modular and granular design of deal.II in addition to the strong integration with many high performance parallel packages such as PETSc , Trilinos, and MUMPS provides a fast and reliable base to build our massively parallel Navier-Stokes solver. In addition, access to high quality preconditioners helps the parallel solve converge faster. For example, parallel Incomplete LU decomposition (ILU) preconditioner performs well but the efficiency goes down as the number of processors goes up (see Chan and Van der Vorst [14]). A more efficient parallel preconditioner is Hypre's BoomerAMG, an algebraic multigrid preconditioner. It helps the solver by propagating information faster without knowledge of the domain's geometry. Both parallel ILU and BoomerAMG are accessible through PETSc.

2.2 MPI

`MPI` [57] is a standard interface for communicating among processes. These processes can be on the same computer (with multiple cores per processor) or distributed on a cluster. The advantage of `MPI` is that it does not matter where the process is run; it only matters that the processes can communicate by moving data around with speed and reliability. The underlying system optimizes the communication based on the hardware platform. The `MPI` system launches N processes and gives each process a unique number ''rank'' ranging from $0 \ldots N-1$. Whether the processes are on the same computer or different computers, `MPI` handles all the logistics and optimum communication path.

For many applications such as CFD simulation software, communication among processes is the bottleneck of performance. Many new technologies try to alleviate this with very fast interconnection networks such as ''InfiniBand''. The main characteristics needed for a network interconnect is low latency (the time it takes for a packet of data to reach its destination) and high throughput (the amount of data per second). We will, however, go no further into networks in this dissertation.

2.3 Measuring Parallelization Performance

To analyze parallel implementations, we employ several measurement criteria. Amdahl [2] was the first to discuss the notion of **speedup**, the most common measure of parallel performance. Given a program that spends T_{ser} seconds in the serial part of the code and T_{par} seconds in the parallel part, we calculate a fraction $[0,1] \ni F_p = \frac{T_{par}}{T_{ser}+T_{par}}$ and serial fraction $F_s = 1 - F_p$, then the speedup for p processors

in perfect conditions is:

$$S_p = \text{speedup}(p) = \frac{T_1}{T_p} = \frac{1}{F_s + \frac{F_p}{p}}$$

where T_1 is the time spent in the serial fraction of the program, p is the number of processors, and T_p is the time spent in the parallel part. This is crucial because the best speedup we can hope for when $p \to \infty$ is:

$$\lim_{p \to \infty} S_p = \frac{1}{F_s}$$

which is finite for any $F_s > 0$. If $F_s = 0$, then $\text{speedup}(p) = p$. F_s usually includes serial overhead, communication waiting time (network or storage), etc. This is a disadvantage for communication-heavy software like the one we are building in this dissertation. There are some rare cases where this limit is exceeded due to **cache effects** (e.g. Benzi and Damodaran [10, p. 95]) .

Another metric that is often used is **efficiency**:

$$E_p = \text{efficiency}(p) = \frac{S_p}{p} \tag{2.1}$$

The ideal efficiency is 1 and it can range between $[0, 1]$.

When testing implementation's scalability in practice, two common scalability tests are used: weak scalability and strong scalability.

- In **strong scalability**, the program size is fixed and the number of processors increases. This is usually used to find the optimum number of processors for a given problem size. Usually, it is difficult to achieve good strong scaling because communication will dominate as the number of processors increase. This is

because the processors get ever smaller portions of the problem and, thus, the processors sit mostly idle waiting for communication to complete.

- In **weak scalability**, the problem size assigned to each processor is kept constant while the overall problem size and number of processors increase (i.e. $\frac{\text{problem size}}{p}$ is constant). To calculate speedup, we need to normalize the time with respect the problem size. Namely:

$$S_p = \frac{T_1}{T_p} \times \frac{N_p}{N_1}, \tag{2.2}$$

were N_p is the global problem size for p processors while performing weak scaling. This test reveals how well the program scales up to thousands of processors when the problem size increases in a typical supercomputer setup. The weak scaling efficiency is usually consistent for programs that have local communications with neighbors that is relatively constant when the problem scale is increased. However, weak scaling efficiency decreases when global communication is used often. Moreover, iterative solvers tend to need more iterations when the problem size increases. One way to account for such a change in number of iterations is to normalize the run time by the average number of iterations. However, we argue that weak scaling must reflect the effect of real life ''throwing more processors at a problem'' when facing big problem sizes regardless of how problem size affects the internals of the algorithms.

We ran the 2D setup described in section 5.5.1.1 with different weak and strong scaling parameters. The results are shown in figures 2.1-2.5. We see in figure 2.1 two regions in the weak scaling on the Brazos supercomputer. The first is $p = 1, \cdots, 16$ that loses efficiency quickly and the second region $p = 16, \cdots, 1024$ where efficiency

is not lost as fast. The reason has to do with $p = 16$ being the first setup with two machines with 8 processors each and communicating over InfiniBand (the high-speed network). It is worth mentioning that at $p = 1,024$, ASPEN is solving a problem with 360 million degrees of freedom, a significant problem size.

The weak scaling on the 64 processor Symmetric Multiprocessing machine (SMP) up.math.tamu.edu in figure 2.2 clearly shows the architecture of the machine itself. The machine has 4 AMD 6380 Opteron microprocessors each containing 16 cores. The figure shows that after $p = 16$, communication starts to occur between the microprocessors which is slower than communicating within the microprocessor and, thus, leads to lower efficiency.

Figure 2.3 showcases a typical problem with strong scaling. Because the problem size is constant, the problem can be too big for $p = 1$ and too small for $p = 1,024$. In this case, $p = 1$ has $\approx 300,000$ degrees of freedom per processor and $p = 1,024$ has ≈ 300 degrees of freedom per processor, a significant difference. This resulted in the terrible efficiency one can see in figure 2.3. To remedy that, figure 2.4 starts from $p = 16$ instead and solve a problem that has $1,443,328$ degrees of freedom per processor. It reaches $p = 1,024$ with $22,552$ degrees of freedom per processor. As a result, we notice a less severe decrease in efficiency as the number of processors reach $p = 1,024$. We also see efficiencies that are above 1 at $p = 32, 64$. This is a side effect of choosing the initial state on $p = 16$. Also, this may indicate that the problem size per processor at $p = 64$ is optimal for the problem at hand where each processor has $360,832$ degrees of freedom. Similarly, in figure 2.5, we see the same behavior on up.math.tamu.edu as we observed on figure 2.3. This also has to do with the ratio of degrees of freedom between the lowest and highest number of processors.

From all the scaling tests discussed, we can conclude that the communication-heavy nature of CFD software is clearly observed as an overall significant decrease

in efficiency as the problem size grows. This can be alleviated by using ever faster and higher bandwidth communication technology. Also, although we only scale up to 1,024 processors, `deal.II` (which ASPEN is built upon) is capable of handling massively parallel computations at 16,384 processors (c.f. Heister [44]).

2.4 Direct and Iterative Solvers

The result of finite element method system assembly is a linear system: a matrix A and a right hand side vector b. The next step is to solve $Ax = b$ to get the unknowns x. There are two types of methods to achieve that: direct methods and iterative methods.

1. **Iterative methods** are based on successive projections onto Krylov subspaces (c.f. Saad [66]). Examples of such methods include Conjugate Gradient Method (CG), Biconjugate Gradient Stabilized (BiCGSTAB), and Generalized Minimum Residual Method (GMRES). They solve $x \approx A^{-1}b$ approximately up to a certain tolerance. This is done by iteratively generating a sequence of approximations that converge to the solution. There are several criteria that have to be met for an iterative method to be convergent (c.f. Saad and Schultz [67] and references therein). In ASPEN, we use CG, BiCGSTAB, and GMRES depending on the enabled options.

   For PDEs that arise in CFD, the linear systems produced converge slowly and requires ‘‘preconditioning.’’ The preconditioner is applied to the linear system $Ax = b$ to transform it in such a way that iterative method converges much faster. ASPEN uses many preconditioners such as IL and Multigrid BoomerAMG.

2. **Direct methods** such as Gaussian Elimination method, LU factorization, and

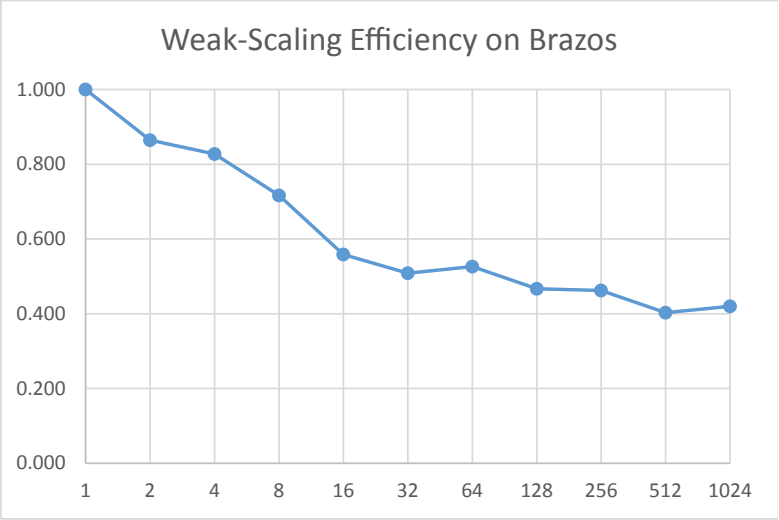


Figure 2.1: Efficiency of weak scaling on Brazos supercomputer [77]. Note that the problem size per processor for $\{p = 4^k, k = 0, \cdots, 5\}$ is $\approx 300,000$ and for $\{p = 2(4^k), k = 0, \cdots, 4\}$ is $\approx 600,000$. Efficiency was calculated using (2.1) and (2.2).

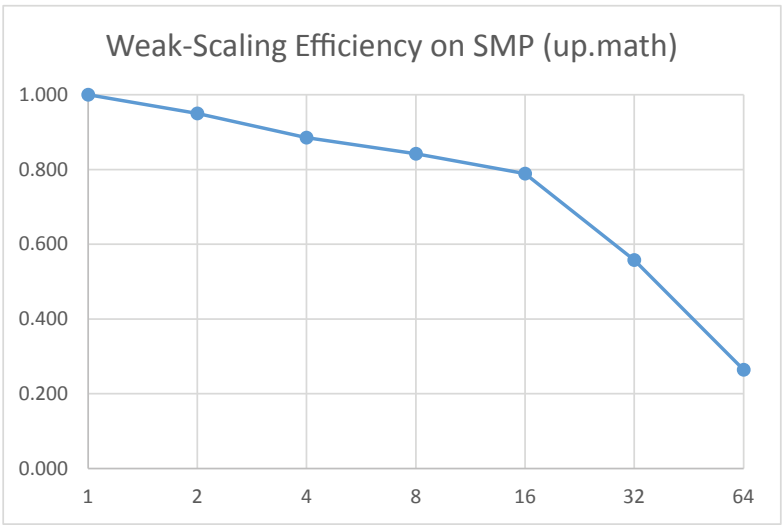


Figure 2.2: Efficiency of weak scaling on the Symmetric Multiprocessor (SMP) machine up.tamu.math.edu. The dip after $p = 16$ is where the communication happens between microprocessors. Efficiency was calculated using (2.1) and (2.2).

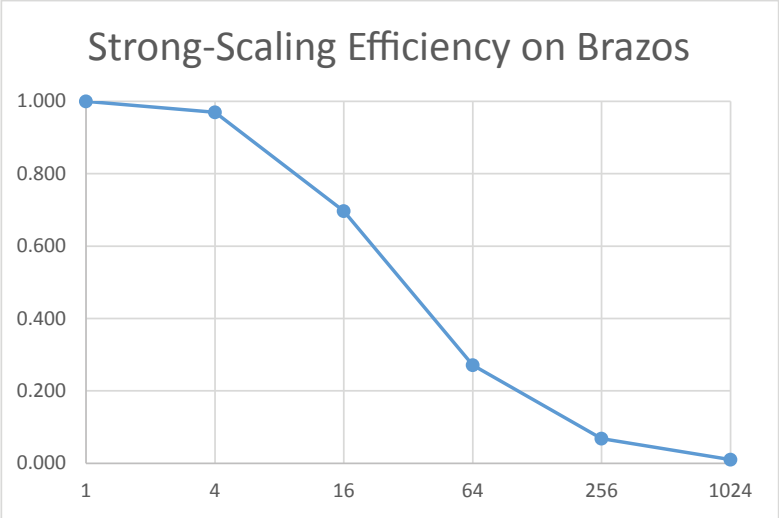


Figure 2.3: Efficiency of strong scaling on Brazos supercomputer [77]. Notice the low efficiency with $p = 1,024$ which comes from the fact that huge gap between number of degrees of freedom: $\approx 300,000$ per processor when $p = 1$ and ≈ 300 per processor when $p = 1,024$.

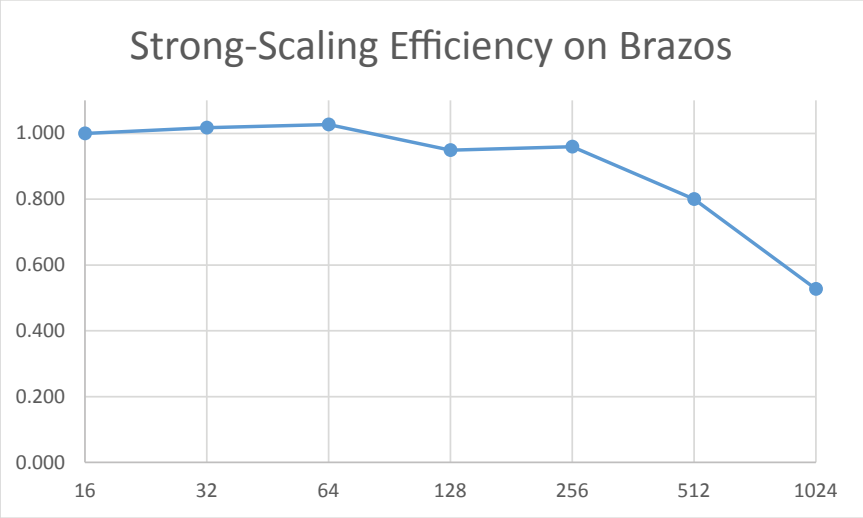


Figure 2.4: Efficiency of strong scaling on Brazos supercomputer [77] when based on $p = 16$. Compared to figure 2.3, we see much better efficiency since $p = 16$ is two machines with 8 processors each communicating over the high-speed network, a reasonable setup to compare $p = 32, \cdots$ to.

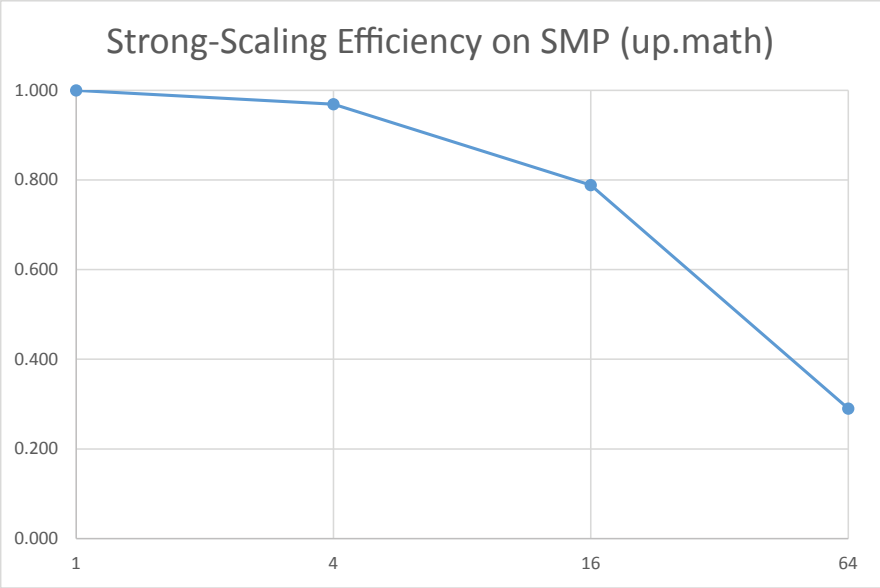


Figure 2.5: Efficiency of strong scaling on the Symmetric Multiprocessor (SMP) machine up.tamu.math.edu.

Cholesky factorization solves the system in a single step exactly. They are robust and work reliably for all invertible matrices. They also do not need preconditioners but need much more memory than iterative methods.

For small problems ($<$ 100,000 dofs), direct solvers are usually faster than iterative solvers. On the other hand, large problems with billions of degrees of freedom can only be feasibly solved with iterative solvers. For a band matrix (a sparse matrix which has the non-zero elements form a band around the diagonal) in 2D, a direct solver works $\mathcal{O}(nb^2)$ where b is the width of the band but an iterative solver can work only $\mathcal{O}(n)$ (or much worse if not properly preconditioned). With clever reordering of rows and columns, one can minimize the matrix's fill-in of non-zero entries in the direct solver's LU factors resulting in an improved $\mathcal{O}(n^{\frac{3}{2}})$ work (c.f. Poulson [62]).

2.5 Building Aspen

We spent a year learning `deal.II` and developing the basic Navier-Stokes equations to be solved in parallel from the start. Coming from a background of Computer Science, learning Finite Element and parallel `deal.II` together was not an easy feat. We started with the fully parallel heat equation. Then we verified the convergence rate in time and space for manufactured solutions. Using manufactured solutions that are part of the discrete space $\boldsymbol{u}|_K \in \mathbb{Q}_k^d$ and to expect exactness as a result was of great help (exactness meaning that the approximation error is *machine epsilon*: 1.19209e-07 for float, 2.22045e-16 for double, 1.0842e-19 for long double). It pointed to small problems (e.g. boundary conditions enforcement location within the code) that did not necessarily alter the solution very much but made the code unreliable. After the heat equation, we proceeded to implement the stokes equation then introduced the projection part of the algorithm and finally the nonlinear term. The gradual building of the code was quite illuminating and helps bridge the gap between one's understanding of the finite element method and algorithm actual implementation. Building the code in parallel from the beginning is crucial. It is not a trivial task to convert a sequential code to a parallel one. Also, many data structures in parallel are tricky to implement and difficult to debug. Moreover, during the course of building Aspen, many performance enhancements and new functionality was contributed back to `deal.II` code base.

2.6 Aspen's Modular Design

Aspen uses a modular design that can be expanded by adding new plugins. The main modules are the `NSSolver` classes: `NSProjectionSolver`, `NSACSolver`, and `NSVDACSolver` classes for the Projection, Artificial Compressibility and Variable

Density Artificial Compressibility algorithms respectively. These modules use the following classes:

- `EquationHandler` classes solve particular equations of the Navier-Stokes equation. The classes that inherit from `EquationHandler` are: `ContinuityHandler`, `MomentumHandler`, and `PressureHandler`. For each algorithm, different subsets are used by different algorithms:
  - The ‘‘Projection’’ algorithm uses: `SSPRK3Solver` for continuity (or `BDF2Solver`), `MomentumProjectionSolver`, and `PressureProjectionSolver`.
  - The ‘‘Constant Density Artificial Compressibility ’’ algorithm uses: `MomentumACSolver`, and `PressureACSolver`.
  - The ‘‘Variable Density Artificial Compressibility ’’ algorithm uses: `ContinuityVDACSolver`, `MomentumVDACSolver`, and `PressureACSolver`.
- `GeometryModel` is the parent class for all classes that provide the geometry to be used by `NSSolver` and mesh refinement functions. Examples include: `Box`, `ColoredBox`, `Channel`, and `DamBreakingGeometry`.
- `BoundaryCondition` provides the boundary conditions for geometries. For example: `SetNoBoundaryCondition`, `SetInletBoundaryCondition`, and `SetZeroBoundaryCondition`.
- `AspenFunction` provides the initial conditions and is used for error analysis. Examples include: `RhoZero`, `UConstantNE`, and `MomentumSourceTermJLGSinCosVarDens`.

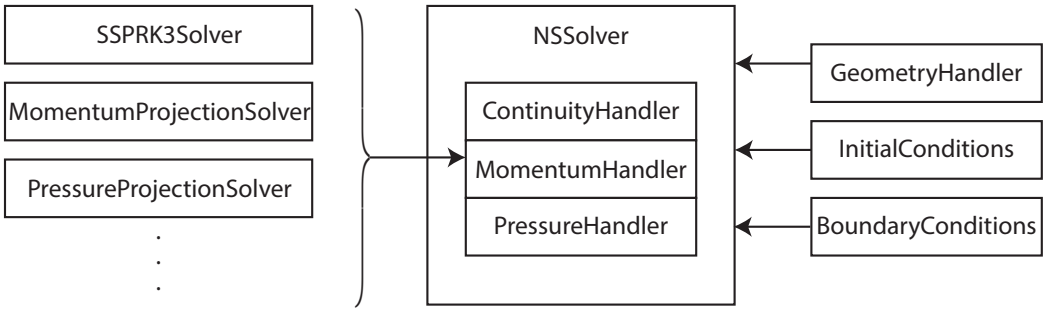


Figure 2.6: Overview diagram of ASPEN.

The `GeometryModel`, `BoundaryCondition`, and `AspenFunction` initial conditions are all plugins chosen using a parameter file. For example, the dam breaking problem discussed in section 5.5.2.2 uses the following parameter file:

```
...
subsection Initial Conditions
  set ExactRho = RhoDamBreaking
  set ExactU   = UZero
  set ExactP   = PZero
  set MomentumSourceTerm   = GravitySource
  set ContinuitySourceTerm   = SourceContinuityZero
end

subsection Geometry Model
  set GeometryModel = DamBreakingGeometry
end

subsection Boundary Conditions
  set ContinuityBoundaryConditions = SetNoBoundaryCondition
  set MomentumBoundaryConditions = DamBreakingBoundaryCondition
  set ProjectionBoundaryConditions = SetZeroBoundaryCondition
end
...
```

A sample of the output for the parameter file is:

```
started 2nd-order method
Cycle 1:
Unable to open points.txt. Assuming no point evaluations.
   Number of active cells:       2107392
   Number of degrees of freedom U: 51295107
   Number of degrees of freedom P: 2167425
   Number of degrees of freedom Rho: 17098369
   Number of degrees of freedom per processor: 1670704
The current time is Wed Jan  7 09:48:44 2015

Time step 1 at t_tryg=0.01 at t=0.01
   Phi_star^1 solved in 5 iterations.
   Recreating the continuity SSPRK3 preconditioner ...
PreconditionBlockJacobi done.
   SolverCG:   Rho^1 solved in 0 iterations.
   Phi_star^2 solved in 0 iterations.
   SolverCG:   Rho^2 solved in 0 iterations.
   Phi_star^3 solved in 0 iterations.
   SolverCG:   Rho^3 solved in 0 iterations.
   started assemble_system ... done
   Recreating the momentum 2ndorder preconditioner ...
PreconditionBlockJacobi done.
   SolverBicgstab:   U solved in 9 iterations.
   cfl = 0.020423
   suggested new time_step = 0.01 t_ty = 0.01 cfl = 0.020423
   Recreating the pressure projection dpsi preconditioner ...
PreconditionNone done.
   SolverCG:   Dpsi solved in 851 iterations.
   Recreating the pressure projection dq preconditioner ...
PreconditionBlockJacobi done.
   SolverCG:   Q solved in 20 iterations.
   OUTPUTING VTU PLOT
   Saving all vectors to vectors.dat
Time for n-1=0 time steps = 26208.1s. (inf s/dof/step)
Outputed norms at t=0.01
Outputed point values at t=0.01

The current time is Wed Jan  7 09:56:37 2015

Time step 2 at t_tryg=0.02 at t=0.02
   Phi_star^1 solved in 0 iterations.
   SolverCG:   Rho^1 solved in 26 iterations.
   Phi_star^2 solved in 0 iterations.
   SolverCG:   Rho^2 solved in 24 iterations.
   Phi_star^3 solved in 0 iterations.
```

```
   SolverCG:   Rho^3 solved in 24 iterations.
   started assemble_system ... done
   SolverBicgstab:   U solved in 11 iterations.
   cfl = 0.235749
   suggested new time_step = 0.01 t_ty = 0.02 cfl = 0.235749
   SolverCG:   Dpsi solved in 885 iterations.
   SolverCG:   Q solved in 19 iterations.
   OUTPUTING VTU PLOT
   Saving all vectors to vectors.dat
Time for n-1=1 time steps = 18827.6s. (0.000176082 s/dof/step)
Outputed norms at t=0.02
Outputed point values at t=0.02

The current time is Wed Jan  7 10:03:12 2015

Time step 3 at t_tryg=0.03 at t=0.03
   Phi_star^1 solved in 0 iterations.
...
```

Figure 2.6 shows an overview of how the different modules are connected.

This is the output of a simulation that has a total of 70,500,000 degrees of freedom taking 450 GB of RAM running on `top.math.tamu.edu` and 64 processors. This corresponds to 6850 bytes per degree of freedom. This is high but acceptable for a project in progress. With optimization, this number will significantly go down and allow for simulation running in the 100's of million of degrees of freedom. More specifically, ordering the allocation of memory for vectors and matrices and calculating the preconditioners in such a way that the maximum memory used at any given time does not exceed the machine's memory. This may reduce the memory per degree of freedom by 20-30% or more.

3. THE TRANSPORT EQUATION

The transport equation describes the *advection* or *transport* of an incompressible fluid along a velocity field. It essentially describes the shifting of the infinitesimal particles of a fluid in the direction of the velocity field.

Consider a simple example in 1D: $\partial_t \rho + 1\partial_x \rho = 0$. This describes the movement of $\rho(x,t)$ in the positive direction with speed 1 per time t. Let $\rho(x,0) = x$ which is a line with slope 1, intersecting the x-axis at 0. Let us approximate the time derivative with the following $\frac{\rho(x,1)-\rho(x,0)}{1} + \partial_x \rho(x,0) = 0 \implies \rho(x,1) = \rho(x,0) - 1 = x - 1$. This is a line with slope 1 intersecting the x-axis at 1. Using the terminology of this section, we transported the function $\rho(x,0) = x$ to the right by 1 at time $t = 1$ and thus becomes $\rho(x,1) = x - 1$. If the speed is c instead of 1, then the PDE becomes $\partial_t \rho + c\partial_x \rho = 0$ and the solution $\rho(x,t) = x - ct$ at time t. These results can also be obtained by using the well known ‘‘method of characteristics.’’

The example above becomes more complicated when working with complex functions and higher dimensions but follows the same advection principle. Moreover, the transport equation becomes unstable if not discretized properly. This is going to be discussed in detail in sections 3.2.1 and 3.2.2. In the next section, we will discuss the mathematical model of the transport equation. Then, we discuss the temporal and spatial discretization of the equation. After that, we will show specific techniques to stabilize the transport equation and compression techniques to counteract the diffusive effects of stabilization.

3.1 The Mathematical Model

The transport equation is a hyperbolic, n-dimensional *Initial Value Problem* scalar conservation partial differential equation (PDE) that has the following form:

$$\frac{\partial}{\partial t}\rho + \operatorname{div}(\boldsymbol{u}\rho) = 0, \qquad \text{in } \Omega, \tag{3.1}$$

$$\rho(\boldsymbol{x}, t) = \rho_{\partial\Omega}, \qquad \text{in } \partial\Omega_-, \tag{3.2}$$

$$\rho(\boldsymbol{x}, 0) = \rho_0, \qquad \rho_0 > 0, \tag{3.3}$$

where $\Omega \subset \mathbb{R}^d, d = \{2, 3\}$ a open domain with a ''smooth enough boundary''. The density $\rho(\boldsymbol{x}, t) : \Omega \times \mathbb{R}_+ \mapsto \mathbb{R}$, and the velocity field $\boldsymbol{u}(\boldsymbol{x}, t) : \Omega \times \mathbb{R}_+ \mapsto \mathbb{R}^d$. We mainly work with incompressible fluids where $\operatorname{div} u = 0$ and, thus, (3.1) becomes:

$$\frac{\partial}{\partial t}\rho + \boldsymbol{u} \cdot \nabla\rho = 0, \qquad \text{in } \Omega. \tag{3.4}$$

Since this is an advection equation, we need to impose the boundary conditions where the *inlets* are. Intuitively, when you have an area adjacent to the boundary and mass is moved inside the domain, something has to be put in its place. Specifically, $\partial\Omega_-$ (the inlet) is where $\boldsymbol{u}\cdot\boldsymbol{n} < 0$ where $\boldsymbol{n}$ is the outward unit vector at the boundaries. This essentially means that the velocity field at the boundary projected to the normal of that boundary must be negative when ''material'' is coming into the domain. ρ_0 is the initial density with no vacuum (hence the strict inequality $\rho_0 > 0$).

Since we are using the finite element method, we take the variational form of the above equations and find the weak solutions to the problem. The problem becomes:

Find $\rho(\boldsymbol{x}, t) \in V(\Omega)$ such that:

$$\int_{\Omega} v\left(\frac{\partial}{\partial t}\rho + \boldsymbol{u} \cdot \nabla\rho\right)\mathrm{dx} = 0, \qquad \forall v \in V(\Omega), \tag{3.5}$$

$$\rho(\boldsymbol{x}, t) = \rho_{\partial\Omega}, \qquad \text{in } \partial\Omega_{-}, \tag{3.6}$$

$$\rho(\boldsymbol{x}, 0) = \rho_0, \qquad \rho_0 > 0, \tag{3.7}$$

where V is an appropriate space for the transport equation with Dirichlet boundary conditions.

For the existence and uniqueness of the solution of (3.5), we quote the following theorem.

Theorem 3.1.1. (DiPerna and Lions [22, Theorem II.3])
For $d = \{2, 3\}$, If $\boldsymbol{u} \in L^1(0, T; W^{1,1}_{loc}(\mathbb{R}^d))$, $\mathrm{div}\,(\boldsymbol{u}) \in L^1(0, T; L^{\infty}(\mathbb{R}^d))$ and $\rho_0 \in L^0(\mathbb{R}^d)$, then there exists a unique renormalized solution of (3.5) in $L^{\infty}(0, T; L^0(\mathbb{R}^d))$.

Theorem 3.1.1 shows that the transport equation has a unique solution in $\mathbb{R}^d$ given the conditions on ρ_0 and $\boldsymbol{u}$. In this dissertation, however, we will deal with bounded domains Ω which have more complex existence and uniqueness conditions than shown in DiPerna and Lions [22], Lions [55] which is beyond the scope of this dissertation. Also, the notion of ‘‘renormalization’’ is discussed in DiPerna and Lions [21].

3.1.1 The Level Set Model

The fluid mixture we are interested in modeling with the transport equations has three phases: oil, water, and gas. Each has a different density value. Since they do not mix, it is important that each phase must be distinct when modeled and the volume of each phase in Ω be conserved. Otherwise, the incompressibility condition $\mathrm{div}\,(u) = 0$ will be violated. As a consequence, when solving the approximation of

the transport equation, one needs to make sure the interface between two phases is tracked with enough accuracy. There are many methods to achieve such accuracy, which can be divided into two classes. In the first one, the interface is implicitly tracked by a function defined on the whole domain. Such methods include the level set method, and volume of fluid method. In the second class, the interface is explicitly tracked with front-tracking methods. We will concentrate in this dissertation on the level set method between two phases.

The level set method was first introduced by Osher and Sethian [61] to evolve the interface with speeds depending on the curvature of a given velocity field. The interface is tracked with a function $\Phi(\boldsymbol{x})$ to represent the $n-1$ dimensional interface $\Gamma \subset \Omega$ separating Ω into two phases Ω_1 and Ω_2. There are many ways to define Γ but we are going to discuss two: the signed distance function with the interface at $\Phi(\boldsymbol{x}) = 0$, and tanh function with the interface at $\Phi(\boldsymbol{x}) = 0.5$. The signed distant function is defined as:

$$\Phi(\boldsymbol{x}) := d(\boldsymbol{x}) = \begin{cases} \min\limits_{\boldsymbol{x}_\Gamma \in \Gamma} \|\boldsymbol{x} - \boldsymbol{x}_\Gamma\|, & \boldsymbol{x} \in \Omega_1, \\ -\min\limits_{\boldsymbol{x}_\Gamma \in \Gamma} \|\boldsymbol{x} - \boldsymbol{x}_\Gamma\|, & \boldsymbol{x} \in \Omega_2. \end{cases}$$

Note that $\|.\|$ can be any norm. The tanh function is defined as:

$$\Phi(\boldsymbol{x}) := \frac{1}{2}\left(1 + \tanh\left(\frac{d(\boldsymbol{x})}{\alpha}\right)\right),$$

where α controls how steep the interface is. From now on, we will use the tanh function to define the level set. In order to describe the evolution of an interface that

is transported along with a fluid, we can use Φ is used instead of ρ in (3.5):

$$\int_\Omega v\left(\frac{\partial}{\partial t}\Phi + \boldsymbol{u}\cdot\nabla\Phi\right)\mathrm{dx} = 0, \qquad \forall v \in V(\Omega), \tag{3.8}$$

$$\Phi(\boldsymbol{x},t) = \Phi_{\partial\Omega}, \qquad \text{in } \partial\Omega_-, \tag{3.9}$$

$$\Phi(\boldsymbol{x},0) = \Phi_0, \qquad \Phi_0 > 0. \tag{3.10}$$

This essentially transports the Φ function instead of the density ρ. To reconstruct ρ from Φ, we use the function $H(\Phi)$:

$$H(\Phi(\boldsymbol{x})) = \begin{cases} \rho_1, & \Phi(\boldsymbol{x}) < 0.5, \\ \rho_2, & \Phi(\boldsymbol{x}) \geq 0.5, \end{cases} \tag{3.11}$$

where ρ_1, ρ_2 are the densities of the fluids in Ω_1 and Ω_2 respectively ($\rho_1 < \rho_2$). However, $H(\Phi)$ will produce density fields that have discontinuous transitions between phases which are undesirable when dealing with PDEs that expect smooth enough functions. There are many functions that create smoother transitions such as:

$$H(\Phi(\boldsymbol{x})) = \frac{\rho_2 - \rho_1}{2} + \frac{\rho_2 + \rho_1}{2}\tanh\left(\frac{\Phi(\boldsymbol{x})}{\alpha}\right),$$

where α controls how steep the transition between the two densities is. The advantage of this reconstruction is that it produces the closest density field close to (3.11) with some retained smoothness. Another candidate H function is:

$$H(\Phi(\boldsymbol{x})) = (\rho_2 - \rho_1)\Phi(\boldsymbol{x}) + \rho_1, \tag{3.12}$$

which is a linear scaling of the level set to the densities in Ω. It is quite robust but it transfers the undesirable oscillations that extends beyond $\Phi(\boldsymbol{x}) > 1$ or $\Phi(\boldsymbol{x}) < 0$.

This may be solved by clipping the reconstruction at a certain radius α around 0.5 $(0 \leq \alpha \leq 0.5)$:

$$H(\Phi(\boldsymbol{x})) = \begin{cases} \rho_1, & \Phi(\boldsymbol{x}) \leq 0.5 - \alpha, \\ \rho_2, & \Phi(\boldsymbol{x}) \leq 0.5 + \alpha, \\ \big(\Phi(\boldsymbol{x}) - (0.5 - \alpha)\big)\dfrac{\rho_2 - \rho_1}{2\alpha} + \rho_1, & \text{otherwise.} \end{cases} \tag{3.13}$$

However, this reconstruction introduces relatively sharp changes in the density gradient and affects the stability of simulation runs. Finally, the last reconstruction we are going to introduce has the property of having slope 0 at the $0.5 \pm \alpha$ points and being a transition polynomial of third degree (again $0 \leq \alpha \leq 0.5$):

$$H(\Phi(\boldsymbol{x})) = \begin{cases} \rho_1, & \Phi(\boldsymbol{x}) \leq 0.5 - \alpha, \\ \rho_2, & \Phi(\boldsymbol{x}) \leq 0.5 + \alpha, \\ \dfrac{(4\alpha - 2\Phi(\boldsymbol{x}) + 1)(2\alpha + 2\Phi(\boldsymbol{x}) - 1)^2}{32\alpha^3}(\rho_2 - \rho_1) + \rho_1, & \text{otherwise.} \end{cases}$$

Compared to the clipped reconstruction (3.13), the above has smooth gradient transitions. Figure 3.1 shows how the transitions look in 1D.

3.2 Numerical Methods

Now we discuss numerical methods to solve equation (3.5). Because computers are of finite capacity, we need to discretize the continuous form (3.5) in both space and time into finite blocks. This is going to be discussed in sections 3.2.1 and 3.2.2. Then sections 3.3.1 and 3.3.3 will elaborate on methods to stabilize discretized variational forms. After that, we will discuss the level set compression technique in 3.3.4.

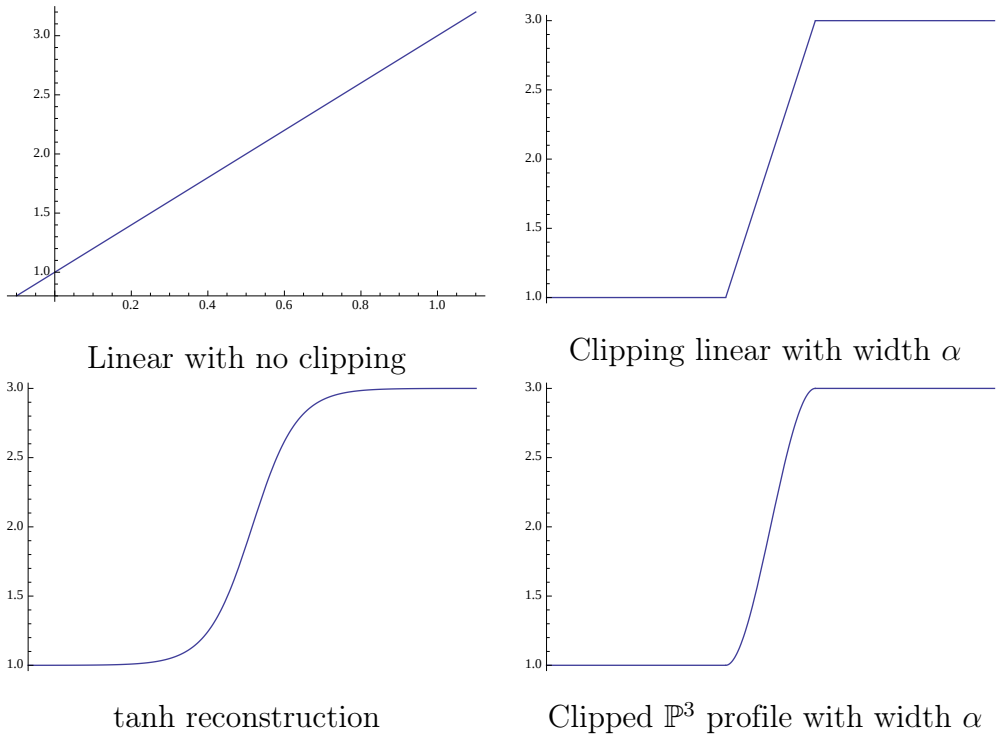


Figure 3.1: The various formulas for reconstruction of the level set to the density field. $\rho_1 = 1, \rho_2 = 3, \alpha = 0.1$.

3.2.1 Time Stepping

Since we are solving the transport equation numerically, we need to discretize the transport equation in time which means we get the values of $\rho(\boldsymbol{x}, t)$ at times $0 = t_0 < t_1 < \cdots < t_n = T$. There are many methods to discretize the transport equation in time. Each method has many advantages and disadvantages. For example, Forward Euler is simple to implement but is first order and requires stabilization (this will be explained in section 3.3.1). Runge-Kutta, on the other hand, is higher order and but requires multiple steps and higher memory requirements.

3.2.1.1 Forward Euler Method

Forward Euler is a first order *explicit* Ordinary Differential Equation solver (also called integrator). It can be derived from expanding a Taylor series around the $n+1$ step and truncating after the second order term.

Consider the following Ordinary Differential Equation (ODE):

$$\frac{\partial y}{\partial t} = f(t, y), \tag{3.14}$$

$$y(0) = y_0. \tag{3.15}$$

We want to find the solution of $y(t)$ at discrete points $\{t^k\}_{k=1}^{K}$ where $t^k = k\Delta t$ using Forward Euler Method as follows:

$$y^{k+1} = y^k + \Delta t f(t^k, y^k), \tag{3.16}$$

which has an error of $O(\Delta t)$. For the transport equation, we have the following:

$$\frac{\partial \rho}{\partial t} = -\boldsymbol{u} \cdot \nabla \rho. \tag{3.17}$$

Consequently, putting (3.16) and (3.17) together is:

$$\rho^{k+1} = \rho^k + \Delta t(-\boldsymbol{u}^k \cdot \nabla \rho^k). \tag{3.18}$$

Forward Euler is stable if the time step is small enough for ODEs (and stable for PDEs when some space stabilization, e.g. artificial viscosity, is added). Moreover, because Forward Euler stepping is essentially a linear combination of the previous time steps, if the proper conditions are met, it will be ‘‘maximum principle preserving’’. This

can be shown by looking at (3.18) and with the proper space discretization discussed in 3.2.2:

$$\rho_i^{k+1} = \rho_i^k + \Delta t \sum_{j \in \text{supp}(\rho_i^k)} (-\boldsymbol{u}_j^k \cdot \nabla \rho_j^k), \tag{3.19}$$

where the superscript is the time index, the subscript is the degree of freedom (dof) index, and $\text{supp}(\rho_i^k)$ is the set of all dofs that are in the closure of the support of the shape function for ρ_i. This can be put more generally as:

$$\rho_i^{k+1} = \sum_{j \in \text{supp}(\rho_i^k)} a_{ij} \rho_j^k. \tag{3.20}$$

Then, we can have the following maximum principle preservation lemma:

Lemma 3.2.1. Let $|\rho_j^k| < \infty, j = 1 \dots N$, if (3.18) is a convex combination of ρ^k. i.e. $\Delta t, h$ such that $\forall j, \sum_i a_{ij} = 1, a_{ij} \geq 0$ in (3.20) then:

$$\min_i \rho_i^k \leq \min_i \rho_i^{k+1} \leq \max_i \rho_i^{k+1} \leq \max_i \rho_i^k.$$

Proof. Assume not. Without loss of generality, $\exists\ i$ such that $\rho_i^{k+1} < \min_j \rho_j^k$. Since $\rho_i^{k+1} = \sum_j a_{ij} \rho_j^k$, this implies that there are some $a_i < 0$ which is a contradiction. Also, if $\rho_i^{k+1} > \max_j \rho_j^k$, then $\sum_j a_{ij} > 1$ which is also a contradiction. □

Forward Euler is stable for ODEs only when the Courant-Friedrichs-Lewy (CFL) condition is satisfied (c.f. Courant et al. [18]). The CFL condition for the transport equation is $|\boldsymbol{u}|_{L^\infty(\Omega)} \frac{\Delta t}{h} \leq C$ where h is the mesh cell edge size and is heuristically $C = 0.5$ (with 1D analysis using uniform mesh). The definition of h will be clarified in the spatial discretization section 3.2.2.

3.2.1.2 Backward Euler Method

The Backward Euler (or *implicit*) method is a slight modification on the Forward Euler method. Backwards Euler is also referred to in the literature as *Backward Differentiation Formula 1* (BDF1). As before, to get $y(t)$ at the discrete points $\{t^k\}_{k=1}^K$:

$$y^{k+1} = y^k + \Delta t f(t^{k+1}, y^{k+1}), \tag{3.21}$$

which also has an error of $O(\Delta t)$. Applying it to the transport equation means putting (3.17) and (3.21) together and we get:

$$\rho^{k+1} - \Delta t(-\boldsymbol{u}^{k+1} \cdot \nabla \rho^{k+1}) = \rho^k.$$

The difference between Forward and Backward Euler methods is that the Backwards Euler method is unconditionally stable (albeit more computationally expensive). For a more thorough discussion of finite difference methods, refer to Thomas [78].

3.2.1.3 Crank-Nicolson Method

The *Crank-Nicolson* method is an *implicit* method combining both Forward and Backward Euler. The averaging of Forward and Backward Euler steps turns to be unconditionally stable and second order in time (Crank and Nicolson [19], Thomas [78]). The order of convergence can be proven by taking a Taylor's expansion around $t^{k+\frac{1}{2}}$. To get the discrete points of $y(t)$ at $\{t^k\}_{k=1}^K$:

$$y^{k+1} = y^k + \frac{\Delta t}{2}\Big[f(t^k, y^k) + f(t^{k+1}, y^{k+1})\Big]. \tag{3.22}$$

The resulting transport equation time step update comes from (3.17) and (3.22)

as follows:

$$\rho^{k+1} - \frac{\Delta t}{2}(-\boldsymbol{u}^{k+1} \cdot \nabla \rho^{k+1}) = \rho^k + \frac{\Delta t}{2}(-\boldsymbol{u}^k \cdot \nabla \rho^k).$$

Both the Backward Euler and Crank-Nicolson schemes are implicit and require the value of an unknown $\boldsymbol{u}^{k+1}$ which is calculated next when we solve the Navier-Stokes momentum equation. To overcome this issue, we calculate a second order extrapolation of such a value: namely $\boldsymbol{u}^{k+1} = 2\boldsymbol{u}^k - \boldsymbol{u}^{k-1}$.

3.2.1.4 Runge-Kutta Method

Runge-Kutta Methods (RK) are a family of ODE integrators that are generated in a common way. Specifically, RK uses the information from slope $\partial y/\partial t = f(t, y)$ at several locations in between t^k and t^{k+1} then perform a weighted averaging of the values. The result is a solution with higher accuracy and better stability property.

The most commonly used RK method is RK of 4th order errors $O(\Delta t^4)$ and is as follows:

$$\begin{aligned}
k_1 &= \Delta t f(t^k, y^k), \\
k_2 &= \Delta t f(t^k + \frac{\Delta t}{2}, y^k + \frac{1}{2}k_1), \\
k_3 &= \Delta t f(t^k + \frac{\Delta t}{2}, y^k + \frac{1}{2}k_2), \\
k_4 &= \Delta t f(t^{k+1}, y^k + k_3), \\
y^{k+1} &= y^k + \frac{1}{6}\big(k_1 + 2k_2 + 2k_3 + k_4\big).
\end{aligned}$$

For the transport equation, we will use Strong Stability Preserving Runge-Kutta with 3 stages and is of order 3 (abbreviated as SSPRK(3,3)) as described by Gottlieb [28]. The strong stability preserving property is $\|y^{k+1}\| \leq \|y^k\|$ which is what makes

it attractive in the transport equation case. The SSPRK(3,3) steps are:

$$y^{(1)} = y^k + \Delta t f(t^k, y^k), \tag{3.23}$$

$$y^{(2)} = \frac{1}{4}\left(3y^k + y^{(1)} + \Delta t f(t^{k+1}, y^{(1)})\right), \tag{3.24}$$

$$y^{k+1} = \frac{1}{3}\left(y^k + 2y^{(2)} + 2\Delta t f(t^{k+\frac{1}{2}}, y^{(2)})\right). \tag{3.25}$$

The SSP property comes from the maximum principle preserving property of the Forward Euler method. Given that Forward Euler is maximum principle preserving, we can derive SSPRK(3,3). First:

$$y^{(1)} = y^k + \Delta t f(t^k, y^k). \tag{3.26}$$

is maximum principle preserving by assumption. So $\min y^k \leq \min y^{(1)} \leq \max y^{(1)} \leq \max y^k$. Note that $y^{(1)}$ is at time $k+1$. Then calculate:

$$y^\star = y^{(1)} + \Delta t f(t^{k+1}, y^{(1)}). \tag{3.27}$$

Again, $\min y^{(1)} \leq \min y^\star \leq \max y^\star \leq \max y^{(1)}$ and $y^\star$ is at time $k+2$. Now, define:

$$y^{(2)} = \frac{3}{4}y^k + \frac{1}{4}y^\star. \tag{3.28}$$

Clearly, since $\frac{3}{4} + \frac{1}{4} = 1$, then $\min y^\star \leq \min y^{(2)} \leq \max y^{(2)} \leq \max y^\star$ and the time of $y^{(2)}$ is $\frac{3}{4}k + \frac{1}{4}(k+2) = k + \frac{1}{2}$. We take one more Forward Euler:

$$\tilde{y} = y^{(2)} + \Delta t f(t^{k+\frac{1}{2}}, y^{(2)}). \tag{3.29}$$

Once more, $\min y^{(2)} \leq \min \tilde{y} \leq \max \tilde{y} \leq \max y^{(2)}$ and $\tilde{y}$ is one time step later at time

$k + \frac{3}{2}$. Finally, define:

$$y^{k+1} = \frac{1}{3}y^k + \frac{2}{3}\tilde{y}. \tag{3.30}$$

by substituting (3.29) in (3.30) and (3.27) in (3.28), we get (3.23)-(3.25)

Like before, we use (3.17) and define $L(\boldsymbol{u}, \rho) = \frac{\partial \rho}{\partial t} = -\boldsymbol{u} \cdot \nabla\rho$+stabilization.

$$\rho^{(1)} = \rho^k + \Delta t L^k(\boldsymbol{u}, \rho^k), \tag{3.31}$$

$$\rho^{(2)} = \frac{1}{4}\left(3\rho^k + \rho^{(1)} + \Delta t L^{k+1}(2\boldsymbol{u}^k - \boldsymbol{u}^{k-1}, \rho^{(1)})\right), \tag{3.32}$$

$$\rho^{k+1} = \frac{1}{3}\left(\rho^k + 2\rho^{(2)} + 2\Delta t L^{k+\frac{1}{2}}\left(\frac{1}{2}\left[3\boldsymbol{u}^k - \boldsymbol{u}^{k-1}\right], \rho^{(2)}\right)\right). \tag{3.33}$$

Now, we can claim the following:

Lemma 3.2.2. The SSPRK (3,3) steps (3.31)-(3.33) satisfy:

$$\min_i \rho_i^k \leq \min_i \rho_i^{k+1} \leq \max_i \rho_i^{k+1} \leq \max_i \rho_i^k.$$

given the CFL condition is met.

Proof. Since (3.31)-(3.33) are shown to be essentially several applications of Forward Euler and convex combinations of intermediate solutions. Meeting the CFL condition implies that the Forward Euler substeps are maximum principle preserving. We invoke lemma 3.2.1 repeatedly to get the stated result. □

3.2.2 Spatial Discretization

Now that we have the time discretization, we need to discretize the space. In this case, we want to split Ω into disjoint cells and thus create a mesh over Ω. Let $\mathcal{T}_h$ be

a mesh as follows:

$$\mathcal{T}_h = \{K_i,\ i = 1, .., N \mid K_i \subset \bar{\Omega} \text{ with non-empty interior and Lipschitz boundary },$$
$$\cup_i^N K_i = \bar{\Omega},\ N < \infty,\ \mathring{K}_i \cap \mathring{K}_j = \emptyset, K_i \cap K_j = \{\emptyset|\text{vertex}|\text{edge}|\text{face}\}\ \forall i, j \leq N\}.$$

The definition above allows for cell K to be shaped in any way possible as long as either any two cells do not intersect or only intersect at a vertex, edge, or face. However, in practice, it is much easier to deal with polygonal K. Specifically in this dissertation, we will deal with cells that have 4 faces and 4 vertices in 2D (quadrilaterals) and 6 faces, 12 edges and 8 vertices in 3D (hexahedra). The mesh has to be *conforming* as in Ern and Guermond [24, Definition 1.55] which essentially forbids hanging nodes that appear mid edge or face (it is worth mentioning that if adaptive mesh refinement is used, the hanging nodes restriction will be relaxed with constraints (c.f. Bangerth et al. [8])). Moreover, the family of meshes $\{\mathcal{T}_h\}_{h>0}$ need to be *non-degenerate* which means that

$$\max_i \frac{h_i}{l_i} < \beta < \infty.$$

where h_i is defined as the maximum diameter of cell $K_i \in \mathcal{T}_h$ and l_i is the diameter of the largest inscribed circle (or sphere) inside K_i. In the literature, such a family of meshes $\{\mathcal{T}_h\}_{h>0}$ are referred to as ‘‘shape-regular in the sense of Ciarlet’’ (cf. Ciarlet [17]).

Once we have the mesh, we choose the finite elements to use within each cell. There are many finite elements to choose from depending on the PDE. Here, we will

use the Lagrange finite element. Specifically, $\mathbb{Q}_k$ elements.

$$\mathbb{Q}_k = \left\{ \sum_{0 \le i_1 \dots i_d \le k} a_{i_1 \dots i_d} \; x_1^{i_1} \dots x_d^{i_d} \;\middle|\; a_{i_1 \dots i_d} \in \mathbb{R} \right\}.$$

For each node p_i or *degree of freedom* in the local cell, we can construct a *shape function* $\sigma_i(\boldsymbol{x}) \in \mathbb{Q}_k$ such that:

$$\sigma_i(p_j) = \delta_{ij}, \quad \forall i, j.$$

The set of local shape functions $\Sigma_K = \{\sigma_1, \sigma_2, ..., \sigma_N\}$ in cell K form a basis in $\mathbb{Q}_k(K)$. This fundamental property makes it possible to compute approximations of solutions of PDEs with finite number of degrees of freedom. When the above are satisfied, the triplet $\{K, \mathbb{Q}_k, \Sigma_K\}$ is called a *finite element*. If, in addition, any $p \in \mathbb{Q}_k(K)$ is uniquely determined by the degrees of freedom Σ_K, then $\{K, \mathbb{Q}_k, \Sigma_K\}$ is also called *unisolvent*.

Now that we have our finite elements, we want to estimate the error of the approximate solutions that we get using finite elements with piecewise $\mathbb{Q}_k$. We expect that with higher order polynomials, we should get a better fitting function to the solution and, therefore, less errors. Indeed, we get the following estimate for the interpolation error:

Theorem 3.2.1. Consider an affine family of finite elements of type $\mathbb{Q}_k$ on a quasi-uniform triangulation $\mathcal{T}_h$. Let $\Pi_{\mathcal{T}_h} : H^{k+1}(\Omega) \mapsto \mathbb{Q}_{\mathcal{T}_h,k}(\Omega)$, which is defined piecewise on the triangulation $\mathcal{T}_h$ of Ω, be a projector onto piecewise polynomials of degree at most k. Then there exists a constant $c > 0$ such that

$$\|u - \Pi_{\mathcal{T}_h} u\|_{H^r(\Omega)} \le c \; h^{k+1-r} \; |u|_{H^{k+1}(\Omega)}, \qquad \text{for } 0 \le r \le k+1. \tag{3.34}$$

Proof. See Grossmann et al. [29, Theorem 4.28] □

We are now ready to define the discrete transport equation. The spatial semi-discretization of the transport equation becomes: Find $\rho_h \in V_{h,k}(\Omega) \subset V(\Omega)$ such that:

$$\int_\Omega v_h \left(\frac{\partial \rho_h}{\partial t} + \boldsymbol{u}_h \cdot \nabla \rho_h \right) dx = 0, \qquad \forall v_h \in V_{h,k}(\Omega), \tag{3.35}$$

$$\rho_h(\boldsymbol{x}, t) = \rho_{h,\partial\Omega}, \qquad \text{in } \partial\Omega_-, \tag{3.36}$$

$$\rho_h(\boldsymbol{x}, 0) = \rho_{h,0}, \qquad \rho_0 > 0, \tag{3.37}$$

where

$$V_{h,k} = \{v_h \in C(\bar{\Omega}) |\ \forall K \in \mathcal{T}_h,\ v_h|_K \in \mathbb{Q}_k, v_h|_{\partial\Omega_-} = 0\},$$

and the time partial derivative is handled by one of the time-stepping schemes described above. Now that we have the discretization in both space and time, we are ready to construct the linear system $A\boldsymbol{x} = \boldsymbol{b}$ which we elaborate on in section 3.4. But first, we need to address an important issue in the next section.

3.3 Oscillations and Stabilization

Since solving the hyperbolic transport equation is known to introduce ever-increasing oscillations if not discretized properly, we need to stabilize it. Some of the stabilization methods are first-order upwinding, streamline upwinding, Petrov-Galerkin, artificial viscosity, etc. Here, we will study stabilizing by adding artificial diffusion of different accuracies. This is especially needed with certain explicit time ODE integrators such as Forward Euler. An important property is that the viscosity vanishes as the discretization becomes finer. In the next sections, we will discuss linear, nonlinear, and Graph Laplacian artificial viscosities.

3.3.1 Linear Artificial Viscosity

Artificial viscosity has been used for over 60 years (c.f. Von Neumann and Richtmyer [84]) to simulate the hyperbolic wave equations in discretized spaces. It is used only in areas where discontinuities appear or oscillations (e.g. shocks) where the gradient of the velocity field at the shock is large. Linear viscosity is added to dampen the numerical discretization noise using a first-order artificial viscosity (c.f. Ern and Guermond [24]). This is done usually by adding $-\text{div}\,(\nu\nabla\rho)$ to the equation 3.35):

$$\int_\Omega v_h \left(\frac{\partial \rho_h}{\partial t} + \boldsymbol{u}_h \cdot \nabla \rho_h - \text{div}\ (\nu \nabla \rho) \right) \text{dx} = 0, \quad \forall v_h \in V_h(\Omega), \tag{3.38}$$

where $\nu|_K = C_m h_K |\boldsymbol{u}|_{L^\infty(K)}$ for each cell. This would stabilize the equation nicely as it adds enough viscosity per cell depending on the speed of the flow. However, it has a drawback of being first-order and active throughout the mesh making it highly dissipative. Also, choosing first order artificial viscosity in second order or higher numerical schemes is ineffective. Choosing the right coefficient C_m is also not trivial and usually requires tuning on the coarse mesh first. Moreover, the definition of h is ambiguous when using nonuniform meshes. In the next section, we will describe a new linear artificial viscosity that does not need tuning.

3.3.2 Graph Laplacian Artificial Viscosity

In this section, we will present a linear artificial viscosity method by Guermond and Nazarov [32] that preserves the ‘‘maximum-principle.’’ It also has the advantage of having no tunable constant although it is more computationally intensive to setup.

The maximum-principle states that solutions for certain elliptic and parabolic PDEs attain their maximum at the boundary. In the context of the transport equation,

the "local discrete maximum-principle" is defined (by Lax [52, p.190]) for the discrete solution ρ_{n+1} at time $n+1$, assuming the initial condition ρ_0 which has a minimum $\rho_{\min} = \min_{\boldsymbol{x}} \rho_0(\boldsymbol{x})$ and $\rho_{\max}$ defined equivalently, as follows: $\rho_{\min} \leq \min \rho_n \leq \rho_{n+1} \leq \max \rho_n \leq \rho_{\max}$. The upwind-flux scheme, discontinuous Galerkin finite elements and finite volumes with piecewise constants have been known to be maximum-principle preserving since the work of Lax [52] or even before that. Guermond and Nazarov [32] first introduced maximum-principle preserving continuous finite element scheme. Although it is only first-order, it paves the way for the development of higher-order schemes. However, this is outside the scope of this dissertation.

Previously, we identified the stabilization term as ''added viscosity'' but Guermond and Nazarov [32] state that maintaining the maximum-principle has little to do with the physical viscosity. Rather, they show it depends on the velocity, cell geometry, and shape functions only. They, therefore, propose a new *Graph Laplacian* term $b(v_h, \rho_h)$ instead of the regular Laplacian $-\mathrm{div}\,(\nu \nabla \rho_h)$ as follows:

$$\int_{\Omega_h} v_h \left(\frac{\partial \rho_h}{\partial t} + \boldsymbol{u}_h \cdot \nabla \rho_h \right) dx + b(v_h, \rho_h) \ = 0, \tag{3.39}$$

where $b(v_h, \rho_h) = \sum_{K \in \mathcal{T}_h} \nu_K \sum_{i,j \in \mathcal{I}(K)} V_i P_j b_K(\varphi_i, \varphi_j)$, such that $\rho_h = \sum_{i=1}^{N} P_i \varphi_i, v_h = \sum_{i=1}^{N} V_i \varphi_i$, and :

$$b_K(\varphi_i, \varphi_j) := \begin{cases} -\dfrac{1}{n_K - 1}|K|, & \text{if } i \neq j, \quad i,j \in \mathcal{I}(K), \\ |K|, & \text{if } i = j, \quad i,j \in \mathcal{I}(K), \\ 0, & \text{if } i \notin \mathcal{I}(K) \text{ or } j \notin \mathcal{I}(K), \end{cases} \tag{3.40}$$

where $\mathcal{I}(K)$ is the union of the support of all shape functions φ_i in K. i.e. let S_i be the support of φ_i, $|S_i|$ is its measure and $S_{ij} := S_i \cap S_j$, then $\mathcal{I}(K) :=$

$\{j \in \{1,...,N\}; |S_j \cap K| \neq 0\}$ and n_k is the number of vertices in cell K, i.e $n_K := \text{card}(\mathcal{I}(K))$. Now, we are ready to specify ν_K:

$$\nu_K = \max_{i \neq j \in \mathcal{I}(K)} \frac{\left| \int_{S_{ij}} (\boldsymbol{u}_h \cdot \nabla \varphi_j) \varphi_i dx \right|}{-\sum_{T \subset Sij} b_T(\varphi_j, \varphi_i)}. \tag{3.41}$$

The time stepping algorithm becomes:

$$P_i^{k+1} = P_i^k - \Delta t m_i^{-1} \sum_{K \subset S_i} \left(\nu_K^k b_K(\rho_h^k, \varphi_i) + \int_K \left(\boldsymbol{u}_h^k \cdot \nabla \rho_h^k \right) \varphi_i \text{dx} \right), \tag{3.42}$$

where $m_i := \int_{S_i} \varphi_i \text{dx}$ the lumped mass matrix diagonal entries. This is used because using the consistent mass matrix does not guarantee the maximum-principle preservation. The lumped mass matrix, however, is known to introduce dispersive effects in the solution. There are techniques to mitigate them (c.f. Guermond and Pasquetti [34]).

To prove that (3.42) is maximum principle preserving, we start with a few preliminaries. The discrete space V_h must be such that the lumped mass matrix is positive definite and:

$$0 < \mu_k^{\min} := \min_{i \in \mathcal{I}(K)} \frac{1}{|K|} \int_K \varphi_i(\boldsymbol{x}) \text{dx}, \quad \mu_k^{\max} := \max_{i \in \mathcal{I}(K)} \frac{1}{|K|} \int_K |\varphi_i(\boldsymbol{x})| \text{dx}. \tag{3.43}$$

Note that ([32, Lemma 4.1])

$$0 < \mu_K^{\min} |S_i| \leq m_i \leq \mu_K^{\max} |S_i|. \tag{3.44}$$

Since $\mu_k^{\min}, \mu_k^{\max}$, and n_K depend on $\hat{K}$ the reference element and there are a finite

number of reference elements in the mesh family $\{\mathcal{T}_h\}_{h>0}$, we can define:

$$\lambda := \max_{\mathcal{T}_h} \max_{K\in\mathcal{T}_h} \frac{\mu_K^{\max}}{\mu_K^{\min}} < +\infty, \quad \gamma := \min_{\mathcal{T}_h} \min_{K\in\mathcal{T}_h} \frac{1}{n_K - 1} > 0. \tag{3.45}$$

Now we are ready for the main theorem.

Theorem 3.3.1. (Discrete maximum principle [32]) Assume that the CFL number is small enough, i.e., $|\boldsymbol{u}|\Delta t^k h^{-1} \leq 1/(\lambda(1+\gamma^{-1}))$. Then the solution to (3.42) satisfies the local discrete maximum-principle, i.e., $\rho_{\min} \leq \min_{j\in\mathcal{I}(S_i)} P_j^k \leq P_i^{k+1} \leq \max_{j\in\mathcal{I}(S_i)} P_j^k \leq \rho_{\max}$ for all $k \geq 0$

Proof. ([32, Theorem 4.2]) We proceed by induction. Let $k \geq 0$ and assume $\rho_{\min} \leq P_i^k \leq \rho_{\max}$ for all $i = 1, \ldots, N$. Notice that $k = 0$ is discrete local maximum principle preserving by definition. Taking the definition (3.41) together with γ in (3.45), the inequality $\|\nabla\varphi_j\|_{L^\infty(K)} \leq h_k^{-1}$ (c.f. [32, (2.3)]), and the inequality $\int_k |\varphi_j| \mathrm{dx} \leq \mu_k^{\max}|K|$ implies that

$$\nu_K^k \leq |\boldsymbol{u}| \max_{i\neq j\in\mathcal{I}(K)} \frac{\left|\int_{S_{ij}} \|\nabla\varphi_j\|\varphi_i \mathrm{dx}\right|}{\gamma|S_{ij}|} \leq \gamma^{-1}|\boldsymbol{u}|h^{-1}\mu_K^{\max}. \tag{3.46}$$

We recast (3.42) into the following:

$$\begin{aligned} P_i^{k+1} =& P_i^k\left(1 - \Delta t^k m_i^{-1} \sum_{k\subset S_i} \left(\nu_K^k b_k(\varphi_i,\varphi_i) + \int_k (\boldsymbol{u}_h^k \cdot \nabla\varphi_i)\varphi_i \mathrm{dx}\right)\right) \\ & - \Delta t^k m_i^{-1} \sum_{\mathcal{I}(S_i)\ni i\neq j} P_j^k \sum_{k\subset S_{ij}} \left(\nu_K^k b_k(\varphi_j,\varphi_i) + \int_k (\boldsymbol{u}_h^k \cdot \nabla\varphi_j)\varphi_i \mathrm{dx}\right), \end{aligned}$$

that we formally write as $P_i^{k+1} = \sum_{j\in\mathcal{I}(S_i)} a_{ij} P_j^k$. First, observe that

$$\sum_{j\in\mathcal{I}(S_i)} a_{ij} = 1 - \Delta t^k m_i^{-1} \sum_{k\subset S_i} \left(\nu_K^k b_k\left(\sum_{j\in\mathcal{I}(S_i)} \varphi_j, \varphi_i\right) + \int_k \left(\boldsymbol{u}_h^k \cdot \nabla \sum_{j\in\mathcal{I}(S_i)} \varphi_j\right)\varphi_i \mathrm{dx}\right),$$

$= 1.$

Since $\sum_{j\in\mathcal{I}(S_i)}\varphi_j|_{S_i} = 1$ and $b_k(\sum_{j\in\mathcal{I}(S_i)}\varphi_j,\varphi_i) = b_k(\sum_{j\in\mathcal{I}(K)}\varphi_j,\varphi_i) = 0$ by design (3.40). Second, we evaluate a bound from below for a_{ii},

$$\begin{aligned}
a_{ii} :&= 1 - \Delta t^k m_i^{-1}\sum_{k\subset S_i}\Big(\nu_K^k b_k(\varphi_i,\varphi_i) + \int_k (\boldsymbol{u}_h^k\cdot\nabla\varphi_i)\varphi_i \mathrm{dx}\Big),\\
&\geq 1 - \Delta t^k m_i^{-1}\sum_{k\subset S_i}(\gamma^{-1}|\boldsymbol{u}|h^{-1}|K| + |\boldsymbol{u}|h^{-1}|K|)\mu_k^{\max},\\
&\geq 1 - |\boldsymbol{u}|\Delta t^k h^{-1}(1+\gamma^{-1})|S_i|\mu_k^{\max}m_i^{-1} \geq 1 - |\boldsymbol{u}|\Delta t^k h^{-1}(1+\gamma^{-1})\lambda,
\end{aligned}$$

where we used $\mu_K^{\max}|S_i|m_i^{-1} \leq \mu_K^{\max}/\mu_K^{\min} \leq \lambda$ (see 3.44). This implies that $a_{ii} \geq 0$ since $|\boldsymbol{u}|\Delta t^k h^{-1} \leq 1/(\lambda(1+\gamma^{-1}))$. Third, we bound a_{ij} from below where $i \neq j$. Observe that $b_k(\varphi_j,\varphi_i) \leq 0$. The definition (3.41) implies that

$$\begin{aligned}
-\sum_{K\subset S_{ij}}\nu_K^k b_K(\varphi_j,\varphi_i) &\geq -\sum_{K\subset S_{ij}}\frac{\left|\int_{S_{ij}}(\boldsymbol{u}_h\cdot\nabla\varphi_j)\varphi_i dx\right|}{-\sum_{T\subset Sij}b_T(\varphi_j,\varphi_i)}b_K(\varphi_j,\varphi_i),\\
&\geq \left|\int_{S_{ij}}(\boldsymbol{u}_h\cdot\nabla\varphi_j)\varphi_i dx\right|,
\end{aligned}$$

which gives

$$a_{ij} := \Delta t^k m_i^{-1}\sum_{k\subset S_{ij}}\left(\nu_K^k b_k(\varphi_j,\varphi_i) + \int_k(\boldsymbol{u}_h^k\cdot\nabla\varphi_j)\varphi_i\mathrm{dx}\right) \geq 0.$$

The above argument shows that P_i^{k+1} is a convex combination of $\{P_j^k\}_{j\in S_i}$. This proves the local discrete maximum-principle and the induction holds for $k+1$. $\square$

Tables 3.1 and 3.2 show some of the values for λ for $\mathbb{P}_n$ and $\mathbb{Q}_n$. The assumptions above break down when $\lambda < 0$. The tables also shows the condition number of the

consistent M and lumped M_L mass matrices. For a discussion on the optimal and suboptimal CFL, refer to [32, Remark 4.3].

2D $\mathbb{Q}_n$

n	λ	$\text{CFL}_{\text{suboptimal}}$	$\text{CFL}_{\text{optimal}}$	$\kappa(M)$	$\kappa(M_L)$
1	1.000	2.50E-01	1.00E+00	9	1
2	16.000	6.94E-03	6.25E-02	81	16
3	16.901	3.70E-03	5.92E-02	148.608	9
4	41.327	9.68E-04	2.42E-02	366.506	20.898
5	50.097	5.54E-04	2.00E-02	1058.54	15.5817
6	244.886	8.33E-05	4.08E-03		
7	151.943	1.03E-04	6.58E-03		
8	-7.086	-1.74E-03	-1.41E-01		
9	4233.660	2.36E-06	2.36E-04		

Table 3.1: Values based on 2D $\mathbb{Q}_n$ for λ and optimal and suboptimal CFL for the Graph Laplacian artificial viscosity where $\text{CFL}_{\text{optimal}} = 1/\lambda$ and $\text{CFL}_{\text{suboptimal}} = 1/(\lambda(1+\gamma^{-1}))$.

3.3.2.1 Graph Laplacian Viscosity and Higher Order Elements

We will describe here some oscillations observed when using the Graph Laplacian artificial viscosity. Let $\Omega = (0, 0.5) \times (-2, 2)$ and the initial conditions be as follows:

$$\rho(\boldsymbol{x}, 0) = \begin{cases} 3, & \text{if } y > 1.5. \\ 2y, & \text{if } 1.5 \geq y > 0.5. \\ 1, & \text{otherwise.} \end{cases} \tag{3.47}$$

The velocity vector field is a constant $\boldsymbol{u} = (0, -1)$ and the boundary condition for ρ is $\rho(\boldsymbol{x}) = 3$ at $\boldsymbol{x} = (x, 2)$ where $x \in [0, 0.5]$.

2D $\mathbb{P}_n$					
n	λ	$\text{CFL}_{\text{optimal}}$	$\text{CFL}_{\text{optimal}}$	$\kappa(M)$	$\kappa(M_L)$
1	1.000	3.33E-01	1.00E+00	5	1
2	∞	0.00	0.00	21.5333	∞
3	13.500	7.41E-03	7.41E-02	67.6667	13.5
4	-14.781	-4.51E-03	-6.77E-02	114.415	∞
5	26.764	1.78E-03	3.74E-02	239.448	18.1818
6	-4.518	-7.90E-03	-2.21E-01		
7	-10.674	-2.60E-03	-9.37E-02		
8	-2.979	-7.46E-03	-3.36E-01		
9	-2.756	-6.60E-03	-3.63E-01		

Table 3.2: Values based on 2D $\mathbb{P}_n$ for λ and optimal and suboptimal CFL for the Graph Laplacian artificial viscosity where $\text{CFL}_{\text{optimal}} = 1/\lambda$ and $\text{CFL}_{\text{suboptimal}} = 1/(\lambda(1+\gamma^{-1}))$.

Like linear artificial viscosity, solutions obtained with the linear transport equation that are stabilized with the Graph Laplacian artificial viscosity would be completely smooth at each time step but too diffusive as in figure 3.2. One can observe that with $\mathbb{Q}_1$ elements and a time step obeying the CFL condition. However, $\mathbb{Q}_2$ elements and higher show some ''jaggedness'' as shown in figures 3.3, 3.4 and 3.5 that are of order 10^{-3} and diminishes as the polynomial degree increases.

Moreover, the total variation $\int_\Omega |\partial_x \rho| + |\partial_y \rho| dx$ is not 0.5 consistently on all elements. With $\mathbb{Q}_1$ as in figure 3.6, one can see that total variation is consistent. Yet, figures 3.7, 3.8 and 3.9 show that total variation is not consistent for $\mathbb{Q}_2$ and higher. They do seem to converge to more or less 0.5.

To address many of the deficiencies of linear artificial viscosity and to achieve second order or higher stabilization, we need the nonlinear artificial viscosity: Entropy-Viscosity.

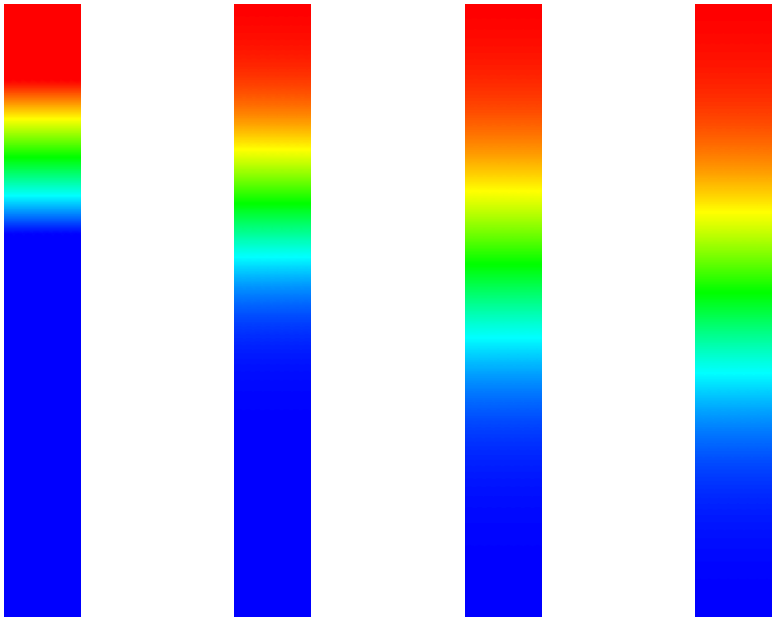


Figure 3.2: This figure shows ρ in linear transport test case #1 as in (3.47). It clearly shows the diffusive nature of the first order artificial viscosity.

3.3.3 Entropy-Viscosity

The Entropy-Viscosity is (at least) a second-order stabilization term introduced by Guermond et al. [42] and Guermond and Pasquetti [33]. It has the advantage of having less diffusive effect on the solution and thus allowing to construct stabilized second order numerical schemes. Using the same transport equation weak form in (3.38) and ν is calculated for each cell separately as follows. Define $E(\phi)$ as convex functions that satisfies the differential inequality:

$$\partial_t E(\phi) + \boldsymbol{u} \cdot \nabla E(\phi) < 0,$$

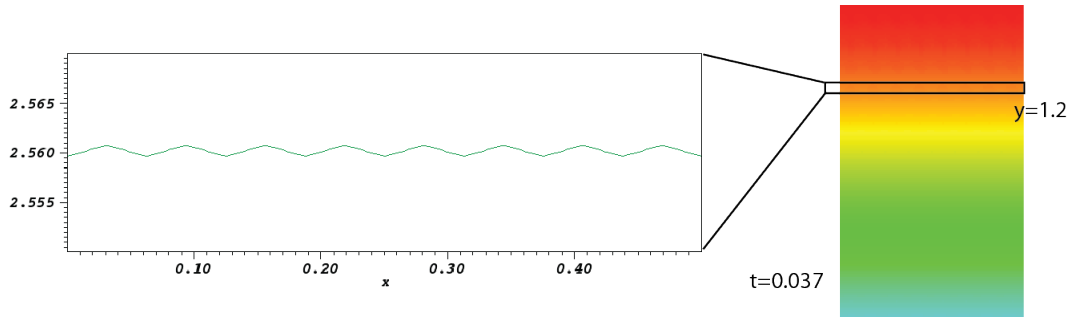


Figure 3.3: The observed jaggedness at the line $y = 1.25$ at $t = 0.037$ when using $\mathbb{Q}_2$ elements, Note that the lines between the degrees of freedom are a side effect of using VisIt visualization software and do not represent the actual cross section from the solution.

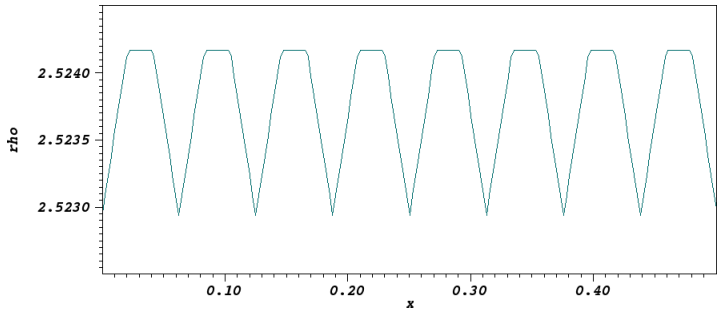


Figure 3.4: The observed jaggedness at the line $y = 1.25$ at $t = 0.037$ when using $\mathbb{Q}_3$ elements. Note that the lines between the degrees of freedom are a side effect of using VisIt visualization software and do not represent the actual cross section from the solution.

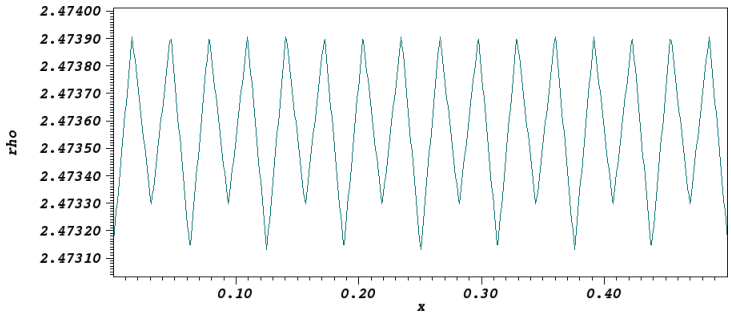


Figure 3.5: The observed jaggedness at the line $y = 1.25$ at $t = 0.037$ when using $\mathbb{Q}_4$ elements. Note that the lines between the degrees of freedom are a side effect of using VisIt visualization software and do not represent the actual cross section from the solution.

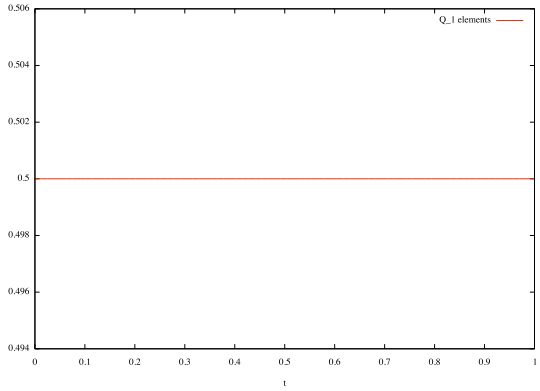


Figure 3.6: The total variation $\int_\Omega |\partial_x \rho| + |\partial_y \rho| dx$ with $\mathbb{Q}_1$ elements with 8×32 cells in 2D and $\Delta t = 0.0002$.

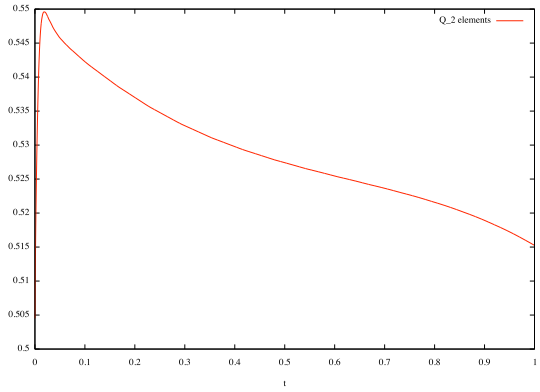


Figure 3.7: The total variation $\int_\Omega |\partial_x \rho| + |\partial_y \rho| dx$ with $\mathbb{Q}_2$ elements with 8×32 cells in 2D and $\Delta t = 0.0002$.

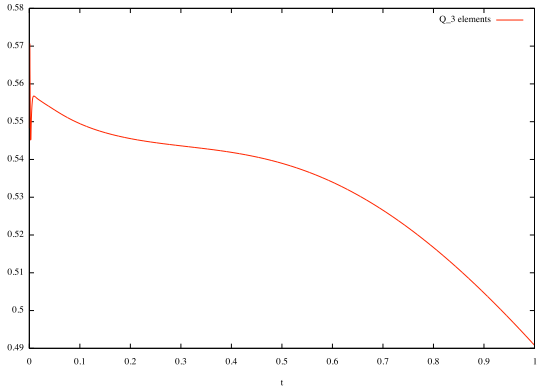


Figure 3.8: The total variation $\int_\Omega |\partial_x \rho| + |\partial_y \rho| dx$ with $\mathbb{Q}_3$ elements with 8×32 cells in 2D and $\Delta t = 0.0002$.

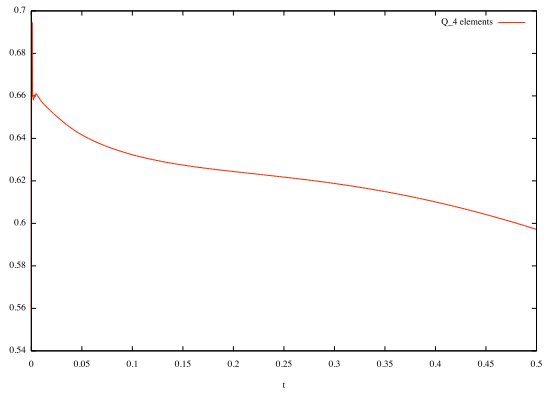


Figure 3.9: The total variation $\int_\Omega |\partial_x \rho| + |\partial_y \rho| dx$ with $\mathbb{Q}_4$ elements with 8×32 cells in 2D and $\Delta t = 0.00005$.

where ϕ is the level set function mentioned in section 3.1.1 and $E(\phi)$ is the entropy function. For examples, one can use:

$$E(\phi) = \begin{cases} \frac{1}{p}(\phi - \frac{1}{2})^p \text{ where } p = 1, 2, \ldots, \\ -\log(|\phi(1-\phi)| + 10^{-14}). \end{cases}$$

In the fully discretized setting, use ϕ^n, ϕ^{n-1} and compute the following values for each quadrature points q_k, q_f in cell k and face f:

$$
\begin{aligned}
R^{n+1/2}(q_k) =& \frac{\Pi_{\mathcal{T}_h} E\left(\phi^n\right) - \Pi_{\mathcal{T}_h} E\left(\phi^{n-1}\right)}{\Delta t} + \\
& \frac{1}{2}\left(\boldsymbol{u}^n \cdot \nabla \Pi_{\mathcal{T}_h} E\left(\phi^n\right) + \boldsymbol{u}^{n-1} \cdot \nabla \Pi_{\mathcal{T}_h} E\left(\phi^{n-1}\right)\right) \\
J^n(q_f) =& u^n \cdot \boldsymbol{n} [\![\partial_n \Pi_{\mathcal{T}_h} E(\phi^n)]\!]|_f .
\end{aligned}
$$

Then get the maximum $R_k^{n+1/2} = \max_{q_k \in k} \left| R^{n+1/2}(q_k) \right|$ and $J_k^n = \max_{f \in k} \max_{q_f \in f} |J^n(q_f)|$. Note that we are using the Crank-Nicolson scheme to calculate R giving us a second order accurate value for R. The viscosity ν_k will then be:

$$
\nu_k = \min\left(C_m h |\mathbf{u}|_{L^\infty}, C_e h^2 \frac{R^{n+1/2} + J_k^n}{\| E(\phi^n) - \overline{E(\phi^n)} \|_{L^\infty(\Omega)}} \right), \tag{3.48}
$$

where $\overline{E(\phi)} = \frac{1}{|\Omega|} \int_\Omega E(\phi)$ and $\| E(\phi^n) - \overline{E(\phi^n)} \|_{L^\infty(\Omega)}$ is a normalization factor. The amount of artificial viscosity is proportional to the entropy production but bounded from above by the linear artificial viscosity. If the solution is smooth and entropy production is very small, little or no artificial viscosity is added. Some disadvantages remain such as coefficients C_e, C_m to tune and the ambiguity of h.

3.3.4 Compression for the Level Set

For the level set method to work, the shape of the level set over the boundary must be maintained to prevent adding non-physical effect to the model. The stabilization viscosity diffuses the level set interface as the simulation marches in time. Consequently, with the presence of the diffusion term, we add the compression term $\operatorname{div}\left(C_K \frac{\nu}{h} (1 - \phi_h) \phi_h \frac{\nabla \phi_h}{|\nabla \phi_h|} \right)$ to the transport equation (3.5) as described in Olsson and Kreiss [60]:

$$
\int_{\Omega \times [0,T]} \frac{\partial}{\partial t} \phi_h + \boldsymbol{u} \cdot \nabla \phi_h - \operatorname{div}\left(\nu \nabla \phi_h - C_K \frac{\nu}{h} \phi_h (1 - \phi_h) \frac{\nabla \phi_h}{|\nabla \phi_h|} \right) dx dt = 0, \tag{3.49}
$$

where the level set $\phi \in [0,1]$ and defined at $\phi = 0.5$ In practice, it has been observed that the compression term in (3.49) induces ''fingering'' effect in simulations. It is the result of perturbations in the initial level set that the compression term gradually propagates resulting in the level set extending like fingers. To mitigate that, a smoothed out ϕ_h^* is used in the normal of the compression front $\frac{\nabla \phi_h}{|\nabla \phi_h|}$ where ϕ_h^* is the solution to $\phi_h^* - h^2 \Delta \phi_h^* = \phi, \nabla \phi_h^* \cdot n = 0$ on $\partial\Omega$. We will denote S as the operator that maps ϕ to the corresponding ϕ^* (i.e. $S\phi_h = \phi_h^*$).

Let us detail the algorithm for solving (3.49):

1. Initialize the level set by normalizing the initial density scalar field.

$$\phi_h^0 = \frac{\rho_h^0 - \rho_{\min}}{\rho_{\max} - \rho_{\min}},$$

2. For each of the SSPRK(3,3) steps below, we need to solve the following:

$$L^n(\boldsymbol{u}_h, \phi_h, \phi_h^*) = -\boldsymbol{u}_h \cdot \nabla \phi_h - \operatorname{div}\left(\nu \nabla \phi_h - C_k \frac{\nu}{h} \phi_h (1 - \phi_h) \frac{\nabla \phi_h^*}{|\nabla \phi_h^*|}\right), \quad (3.50)$$

when solved for each of the three steps below, the values are

$$\phi_h^{(1)} = \phi_h^n + \Delta t L^n(\boldsymbol{u}_h^n, \phi_h^n, S\phi_h^n), \quad (3.51)$$

$$\phi_h^{(2)} = \frac{1}{4}\left(3\phi_h^n + \phi_h^{(1)} + \Delta t L^{n+1}(2\boldsymbol{u}_h^n - \boldsymbol{u}_h^{n-1}, \phi_h^{(1)}, S\phi_h^{(1)})\right), \quad (3.52)$$

$$\phi_h^{n+1} = \frac{1}{3}\left(\phi_h^n + 2\phi_h^{(2)} + 2\Delta t L^{n+\frac{1}{2}}\left(\frac{1}{2}\left[3\boldsymbol{u}_h^n - \boldsymbol{u}_h^{n-1}\right], \phi_h^{(2)}, S\phi_h^{(2)}\right)\right). \quad (3.53)$$

3. Lastly, we ''denormalize'' the level set with a reconstruction function such as:

$$\rho_h^{n+1} = H(\phi_h)(\rho_{\max} - \rho_{\min}) + \rho_{\min}. \quad (3.54)$$

It is worth mentioning that when the entropy-viscosity vanishes in well resolved regions of the solution, the compression stops working and the sharpness of the level set interface is lost. This may be remedied by using some ''antivanish'' viscosity $\nu_{\text{antivanish}} = \nu + \nu_\epsilon$ where ν_ϵ is a small positive amount of viscosity that maintains the balance between diffusion and compression and, thus, maintains the slope of the level set.

3.4 Linear Systems and Linear Solvers

After discretizing the PDE in space and time, the complete linear system using SSPRK(3,3) become a series of linear systems $Mx = b$ where M is the mass matrix. Also, the calculation of the smoothed ϕ_h^* is also done with the linear system $(M + h^2 A)x = b$. The sum of the matrices behaves well in the sense that it has a condition number that remain reasonable when $h \to 0$. In the following linear systems, $\{\varphi_i\}$ signify the shape functions (or basis) of $V_{h,k}$.

For the algorithm described in the last section, the linear systems are as follows:

- The smoothed level set ϕ_h^* is obtained by solving:

$$(M + h^2 A)\phi_h^* = \phi_h,$$

  where M is the mass matrix and A is the stiffness matrix.

- The three ϕ_h's solutions in equations (3.51)-(3.53) is obtained by solving $(\varphi_i, k) = (\varphi_i, L^n(\boldsymbol{u}_h, \phi^n, S\phi^n))$. The weak form with proper integration by parts becomes:

$$Mk_h = \int_\Omega \nabla\varphi_i \left(\nu\nabla\phi_h - C_k \frac{\nu}{h}\phi_h(1-\phi_h)\frac{\nabla\phi_h^*}{|\nabla\phi^*|} \right) - \varphi_i(\boldsymbol{u}_h \cdot \nabla\phi_h)\mathrm{dx},$$

where the boundary terms are removed with appropriate boundary conditions and M is the mass matrix. After that, add the vectors:

$$\phi_h^{(1)} = \phi_h^n + \Delta t k_h.$$

The next SSPRK(3,3) steps follow in a similar fashion.

3.5 Numerical Results

Now that we have a parallel, working implementation, the natural next step is to validate the code. We do that in two ways: conforming ($\rho \in V_h$) and nonconforming ($\rho \notin V_h$) manufactured solutions.

3.5.1 Validation

Implementing numerical schemes is tricky. There are many interdependent components of the implementation - both programming and mathematical in nature - that need to be done correctly so that it works with every configuration. To this end, it is important to test the code against *manufactured solutions*. We take advantage of the error estimates (e.g. (3.34)) and use it validate our implementation. If we use $\mathbb{Q}_2$ piecewise elements, we expect that we can ''approximate'' polynomials of degree 2 exactly; meaning that the error should be machine epsilon. Also, the time stepping method is accurate to a certain order; e.g. SSPRK(3,3) is third order and, therefore, would exactly represent polynomials of third degree in time. Let us start by using the following manufactured solution in the 2D domain $\Omega = (0,1) \times (0,1)$ going to

$T = 0.2$:

$$\begin{aligned} \rho(\boldsymbol{x}, t) &= (t+1)\left(x^2 - 2y^2\right), \\ \boldsymbol{u}(\boldsymbol{x}, t) &= (1,1)^\top, \end{aligned} \tag{3.55}$$

which is a polynomial of second degree in space and first degree in time. As expected, when using SSPRK(3,3) and $\mathbb{Q}_3$ elements, table 3.3 shows the error is ''machine epsilon'' away from zero.

cells	ρ_{dof}	Δt	$\|e_\rho\|_{L^2}$
4	49	0.06	8.06E-16
16	169	0.03	1.16E-15
64	625	0.015	1.93E-15
256	2401	0.0075	3.19E-15

Table 3.3: The error when using $\mathbb{Q}_3$ elements with the manufactured solution (3.55).

Remark. To get exact results as table 3.3 shows, we need to be careful with when to enforce boundary conditions. For example, when time stepping using SSPRK(3,3), we solve three $L(\boldsymbol{u}, \rho)$ equations. Then we add the result to get $\rho^{(1)}, \rho^{(2)}$, and ρ^{k+1}. We have to enforce the boundary conditions in the *last step* only as suggested in Carpenter et al. [13] to achieve exactness. Otherwise, we do not get the expected third-order convergence in time. However, when using SSPRK(3,3) with the real problem instead of the manufactured solution, Carpenter et al. [13] suggests enforcing the boundary condition in the intermediate steps as well as this would help with increasing the CFL number.

Remark. For the mass conservation equation specifically, one needs to take care when constructing a manufactured ρ. For example, figure 3.10 shows Mathematica

```
In[1]:= SSPRK3Error = Function[ρ,
          (* L[t,r] = f[t] - u.r where f[t] = ∂t ρ[t]+u.ρ[t]*)
          L[time_, r_] := D[ρ[time], {t}] + D[ρ[time], {x}] + D[ρ[time], {y}] -
            D[r, {x}] - D[r, {y}];
          (*Do one SSPRK(3,3) step*)
          ρ1 = ρ[t] + τ L[t, ρ[t]] ;
          ρ2 = (3 ρ[t] + ρ1 + τ L[t + τ, ρ1] ) / 4;
          ρ3 = (ρ[t] + 2 ρ2 + 2 τ L[t + τ / 2, ρ2] ) / 3;

          (* Find the SSPRK(3,3) error for the given ρ *)
          Simplify[ρ3 - ρ[t + τ]]]

In[2]:= ρ[t_] := (x - 2 y) (t + 2)^2;
        SSPRK3Error[ρ]

Out[3]= 0

In[4]:= ρ[t_] := (x - 2 y)^2 (t + 2)^2;
        SSPRK3Error[ρ]

Out[5]= -τ^4/3
```

Figure 3.10: Mathematica code to check how much error SSPRK(3,3) introduces for a given ρ.

code that evaluates the local error that SSPRK(3,3) introduces in a single time step. In the first example, $\rho(t) = (x - 2y)(t + 2)^2$ is a second degree polynomial in time and is exactly reproduced by SSPRK(3,3) as shown in the figure. However, $\rho(t) = (x - 2y)^2(t + 2)^2$ is also second degree in time but gives a local error of $\frac{-\Delta t^4}{3}$ for a global error of $\mathcal{O}(\Delta t^3)$. This depends on how L is constructed based on the exact ρ chosen. Just because the time-stepping scheme is $\mathcal{O}(\Delta t^3)$ does not mean that it reproduces 3rd degree polynomials exactly.

3.5.2 Convergence Analysis

Now we use manufactured solutions that are nonconforming and will therefore show the expected convergence rates for the mass conservation equation. Johnson et al. [48] shows that the standard continuous Galerkin finite element method using

$\mathbb{Q}_k$ elements has an error estimate of $\mathcal{O}(h^k)$ with no stabilization (With streamline diffusion stabilization method, the error improves to $\mathcal{O}(h^{k+\frac{1}{2}})$). We will calculate convergence rates in two problems: the **channel** problem setting and the **circular** problem setting.

3.5.2.1 Channel problem

The **channel** problem has a 2D domain $(0, 0.5) \times (-2, 2)$ uniform mesh with a constant downwards velocity $\boldsymbol{u} = (0, -1)^\top$. We will use the algorithm (3.51)-(3.53) with a level set function $\phi \in [0, 1]$ with the linear reconstruction (3.12) such that $\rho_{\min} = 1,\ \rho_{\max} = 3$. The boundary condition is set at the top edge $y = 2$ with the value of $\rho_h((x, 2)^\top, t)$ as given in (3.56). The discrete space for the density $\rho_h \in \{\mathbb{Q}_k \mid k = 1 \dots 4\}$ and velocity field $\boldsymbol{u}_h \in \mathbb{Q}_2$. Also, Gaussian quadrature is used with $\max\{\deg(\rho_h), \deg(\boldsymbol{u}_h)\} + 1$ number of points. No stabilization is added to the algorithm (i.e. $C_m = C_e = 0$) and the simulation is run until $T = 1$ so that the simulations can finish in a reasonable time even for small Δt. The shape of the initial density is :

$$\rho_h(\boldsymbol{x}) = \frac{\rho_{\max} + \rho_{\min}}{2} + \frac{\rho_{\max} - \rho_{\min}}{2} \tanh \frac{y + \beta}{\alpha}, \tag{3.56}$$

where $\alpha = 0.1$ and $\beta = -0.5$. The time step is chosen small enough to show the convergence rate in space. This is achieved by taking a time step $\Delta t \ll h^{\frac{k+1}{3}}$. If the CFL exceeds 0.3 then the time step is adjusted accordingly. Note that we can get the shape of the exact solution at time t by having $\beta = t - 0.5$ in (3.56) because $|\boldsymbol{u}| = 1$.

Table 3.4 shows the convergence rates with respect to space. Notice that all the Lagrange finite elements $\mathbb{Q}_k$ with k odd have a convergence rate of $\mathcal{O}(h^{k+1})$ but when k is even, the rate is $\mathcal{O}(h^k)$ (note that $\mathcal{O}(h^{k+1})$ is the best error one can get with Lagrange finite elements approximation space according to theorem 3.2.1).

FE	cells	ρ_{dofs}	Δt	h	$\|e_\rho\|_{L2}$	rate	CFL_{max}
$\mathbb{Q}_1$	128	165	1.25E-03	2.20E-01	7.41E-02	-	0.01
	512	585	5.00E-04	1.17E-01	1.76E-02	2.27	0.008
	2048	2193	1.25E-03	6.04E-02	2.42E-03	3.00	0.04
	8192	8481	5.00E-04	3.07E-02	4.79E-04	2.40	0.032
	32768	33345	1.25E-03	1.55E-02	1.18E-04	2.04	0.16
	131072	132225	5.00E-04	7.78E-03	2.94E-05	2.02	0.128
$\mathbb{Q}_2$	128	585	6.25E-04	1.17E-01	7.09E-03	-	0.01
	512	2193	2.50E-04	6.04E-02	1.41E-03	2.44	0.008
	2048	8481	6.25E-04	3.07E-02	3.62E-04	2.01	0.04
	8192	33345	2.50E-04	1.55E-02	9.23E-05	2.00	0.032
	32768	132225	6.25E-04	7.78E-03	1.93E-05	2.27	0.16
	131072	526593	2.50E-04	3.90E-03	4.99E-06	1.96	0.128
$\mathbb{Q}_3$	128	1261	5.00E-05	2.82E-02	8.48E-04	-	0.0012
	512	4825	1.25E-05	1.44E-02	5.46E-05	4.09	0.0006
	2048	18865	2.50E-05	7.28E-03	2.38E-06	4.59	0.0024
	8192	74593	5.00E-06	3.66E-03	1.46E-07	4.07	0.001
	32768	296641	6.25E-06	1.84E-03	8.65E-09	4.09	0.0024
	131072	1183105	1.25E-06	9.19E-04	5.40E-10	4.01	0.001
$\mathbb{Q}_4$	128	2193	5.00E-05	2.14E-02	8.07E-05	-	0.0016
	512	8481	1.25E-05	1.09E-02	4.12E-06	4.40	0.0008
	2048	33345	5.00E-05	5.48E-03	3.18E-07	3.74	0.0064
	8192	132225	1.25E-05	2.75E-03	1.94E-08	4.06	0.0032

Table 3.4: The convergence rates with respect to space for the **channel** problem setting. The time step was calculated to be small enough ($\Delta t \ll h^{\frac{k+1}{3}}$) so that we can get the space error. Some time steps are repeated because the time step is small enough for the fine mesh.

We conjecture that this phenomena may be attributed to symmetries found in odd $\mathbb{Q}$ finite elements that captures solutions with higher accuracy as oppose to even numbered finite elements. Figure **??** shows the density and the error of one of the solutions calculated. One can see some oscillations developing in the red part of the density field. This could be eliminated by adding some stabilizing viscosity. However, this would affect the convergence rate calculations which is the aim of this exercise.

3.5.2.2 Circular problem

The **circular** problem has a 2D unit square domain $(0,0) \times (1,1)$ with a velocity field $\boldsymbol{u}(\boldsymbol{x},t) = (\sin(2\pi t)\sin(\pi x)\cos(\pi y), -\sin(2\pi t)\cos(\pi x)\sin(\pi y))^\top$. This field has the property of having zero normal velocity on the boundaries and thus boundary conditions need not be enforced there. Also, at times $t = 0, 1, 2, \ldots$, the density field returns to the initial shape. Similar to the channel problem, we will use the algorithm (3.51)-(3.53) with a level set function $\phi \in [0,1]$ with the linear reconstruction (3.12) such that $\rho_{\min} = 1$, $\rho_{\max} = 3$. Also, the discrete space for the density $\rho_h \in \{\mathbb{Q}_k \mid k = 1 \ldots 4\}$ and velocity field $\boldsymbol{u}_h \in \mathbb{Q}_2$. Gaussian quadrature is used with $\max\{\deg(\rho_h), \deg(\boldsymbol{u}_h)\} + 1$ number of points. No stabilization is added to the algorithm and the simulation is run until $T = 1$. The initial density is (3.56) with $\alpha = 0.1$ and $\beta = 0$.

Table 3.5 shows that we get the convergence rate $\mathcal{O}(h^{k+1})$ which is higher than what we got with the channel problem. It also does not exhibit the same odd/even elements behavior that we see with the channel problem. Figure 3.12 shows the density solution at the beginning, middle, and end of the circular simulation. It also shows the error compared to a stationary solution. Notice that at time $t = 1$ the maximum error is *lower* than the error in the first time step.

The convergence rate of SSPRK(3,3) with respect to time is validated in table 3.6

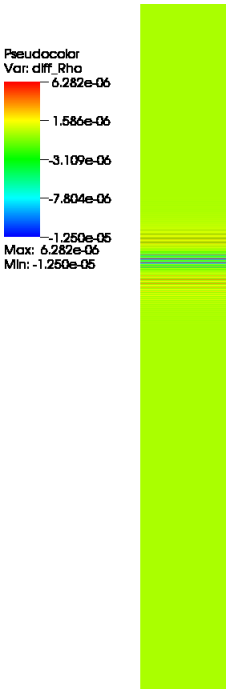
FE	cells	ρ_{dofs}	Δt	h	$\|e_\rho\|_{L2}$	rate	CFL_{max}
$\mathbb{Q}_1$	16	25	1.25E-02	2.00E-01	1.66E-01	-	0.0495
	64	81	5.00E-03	1.11E-01	4.34E-02	2.28	0.0399
	256	289	1.25E-02	5.88E-02	1.04E-02	2.24	0.1999
	1024	1089	5.00E-03	3.03E-02	2.65E-03	2.07	0.16
	4096	4225	2.50E-03	1.54E-02	6.64E-04	2.04	0.16
	16384	16641	1.25E-03	7.75E-03	1.66E-04	2.02	0.16
$\mathbb{Q}_2$	16	81	6.25E-03	1.11E-01	1.72E-02	-	0.0495
	64	289	2.50E-03	5.88E-02	2.61E-03	2.97	0.0399
	256	1089	5.00E-03	3.03E-02	6.18E-04	2.17	0.1599
	1024	4225	2.50E-03	1.54E-02	8.13E-05	2.99	0.16
	4096	16641	1.25E-03	7.75E-03	1.41E-05	2.56	0.16
	16384	66049	5.00E-04	3.89E-03	1.36E-06	3.39	0.128
$\mathbb{Q}_3$	16	169	5.00E-03	7.69E-02	5.29E-03	-	0.0594
	64	625	1.25E-03	4.00E-02	9.02E-04	2.70	0.0299
	256	2401	2.50E-03	2.04E-02	5.17E-05	4.25	0.1199
	1024	9409	5.00E-04	1.03E-02	3.12E-06	4.11	0.048
	4096	37249	6.25E-05	5.18E-03	1.95E-07	4.03	0.012
$\mathbb{Q}_4$	16	289	5.00E-03	5.88E-02	2.47E-03	-	0.0792
	64	1089	1.25E-03	3.03E-02	8.43E-05	5.09	0.0399
	256	4225	5.00E-04	1.54E-02	3.53E-06	4.68	0.032
	1024	16641	1.25E-04	7.75E-03	1.51E-07	4.60	0.016

Table 3.5: The convergence rates with respect to space for the **circular** problem setting. The time step was calculated to be small enough ($\Delta t \ll h^{\frac{k+1}{3}}$) so that we can get the space error. Notice the rates that suggest $\mathcal{O}(h^{k+1})$.

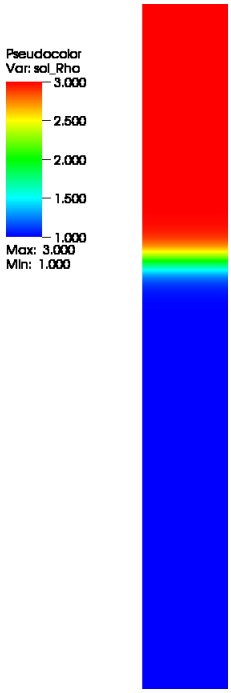
with a small h. The expected $\mathcal{O}(\Delta t^3)$ is clearly evident in the table.

cells	ρ_{dofs}	Δt	h	$\|e_\rho\|_{L2}$	rate
128	2193	2.00E-02	2.50E-02	2.74E-03	-
512	8481	1.00E-02	1.25E-02	3.75E-04	2.87
2048	33345	5.00E-03	6.25E-03	4.79E-05	2.97

Table 3.6: Convergence rate with respect to time using $\mathbb{Q}_5$ element in the channel problem. Note that the CFL_{max} is 0.64.

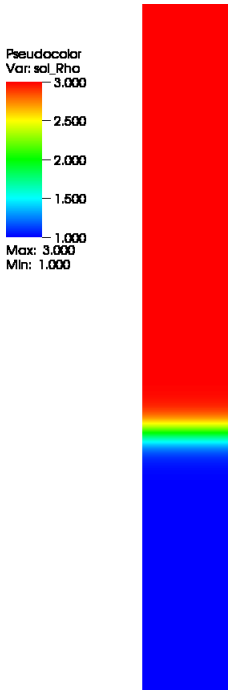


$\rho - \rho_h$ at $\Delta t = 1.25\text{E-}3$



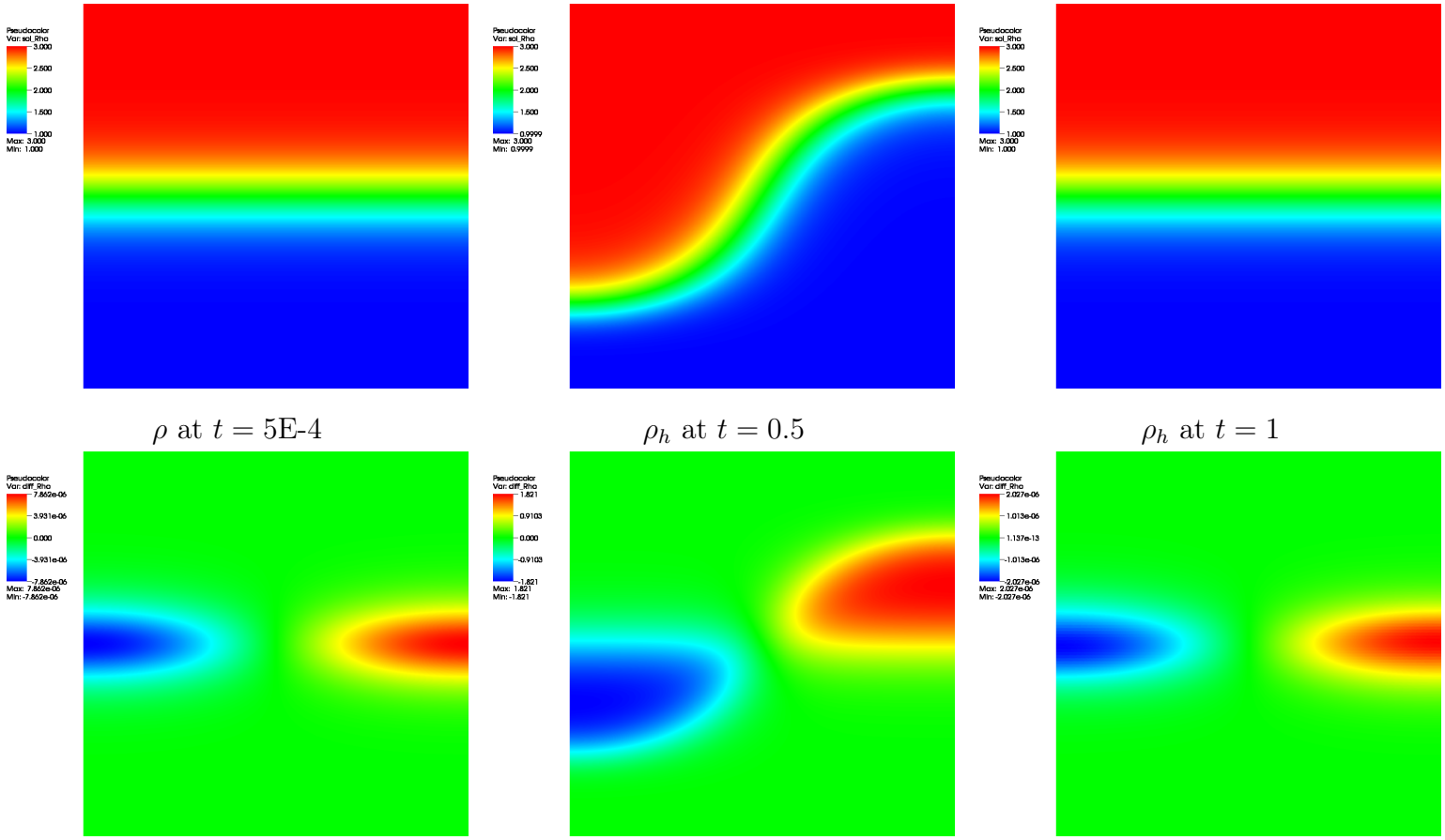
ρ_h at $\Delta t = 1.25\text{E-}3$

$\rho - \rho_h$ at $t = 1$



ρ_h at $t = 1$

Figure 3.11: The channel problem with 32768 $\mathbb{Q}_2$ cells, 132225 dofs, $\Delta t = 1.25\text{E-}3$, and run until $t = 1$. Notice that the error $\rho - \rho_h$ at $t = 1$ has a maximum error of 1E-4 compared to the circular problem where the error becomes lower at $t = 1$.



ρ at $t = 5\text{E-}4$ $\quad$ ρ_h at $t = 0.5$ $\quad$ ρ_h at $t = 1$

$\rho - \rho_h$ at $t = 5\text{E-}4$ $\quad$ $\rho - \rho_h$ at $t = 0.5$ $\quad$ $\rho - \rho_h$ at $t = 1$

Figure 3.12: The circular problem with 16384 $\mathbb{Q}_2$ cells, 66049 dofs, Δt =5E-4, and run until $t = 1$. Notice that the error $\rho - \rho_h$ at $t = 1$ has a maximum error of 2E-6 which is *lower* than the maximum error of 7E-6 at $t = 5\text{E-}4$.

4. THE CONSTANT-DENSITY NAVIER-STOKES EQUATION

In this chapter, we consider the simplified constant-density momentum equation. This simplification allows us to analyze the momentum equation characteristics without introducing too many unnecessary technicalities. In the next chapter, we will address the complete and more complicated momentum equation in the variable density Navier-Stokes equation.

4.1 The Mathematical Model

The flow of a viscous incompressible fluid is described by the *incompressible Navier-Stokes equations* — under certain assumptions — defined as follows:

$$\partial_t \boldsymbol{u} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} - 2\nu \mathrm{div}\,(\nabla^s \boldsymbol{u}) + \nabla p = \mathbf{f}, \qquad \text{in } \Omega \times (0, T] \tag{4.1}$$

$$\mathrm{div}\,(\boldsymbol{u}) = 0, \qquad \text{in } \Omega \times (0, T] \tag{4.2}$$

Where $\Omega \subset \mathbb{R}^d, d = 2, 3$ and $\partial\Omega$ is its boundary, $\boldsymbol{u}(\boldsymbol{x}, t)$ is the velocity vector field, $p(\boldsymbol{x}, t)$ is the pressure, ν is the kinematic viscosity coefficient, and $\mathbf{f}$ is the driving external force.

The initial and boundary conditions of $\boldsymbol{u}$ for the incompressible Navier-Stokes equations are:

$$\boldsymbol{u}|_{t=0} = \boldsymbol{u}_0, \quad \text{in } \Omega, \tag{4.3}$$

$$\boldsymbol{u}|_{\partial\Omega} = \boldsymbol{u}_{\partial\Omega}, \quad \text{in } [0, T], \tag{4.4}$$

for Dirichlet boundary conditions where $\boldsymbol{u}_{\partial\Omega}$ is the boundary condition of $\boldsymbol{u}$ and $\boldsymbol{u}_0$ is the initial condition.

We will discuss what is meant by ''solution'' for the Navier-Stokes equations. Before the 1930's, it was believed that solutions had to have two continuous derivatives for the velocity and one for the pressure as required by the equations. Such solutions are call ''classical'' solutions:

$$\boldsymbol{u} \in C^1([0,T], C^2(\Omega) \cap C^0(\overline{\Omega})), \qquad p \in C^0([0,T], C^1(\Omega)).$$

This setting is too restrictive and impractical. This gave rise to the need of weak solutions. Weak solutions come from solving the weak formulation of (4.1). Define $\mathcal{D}(\Omega)$ as the set of C^∞ functions that are compactly supported in Ω and $\mathcal{V} = \{\boldsymbol{v} \in \mathcal{D}(\Omega) \mid \operatorname{div}(\boldsymbol{v}) = 0\}$ the restriction of $\mathcal{D}(\Omega)$ to solenoidal $\boldsymbol{u}$. The weak formulation becomes:

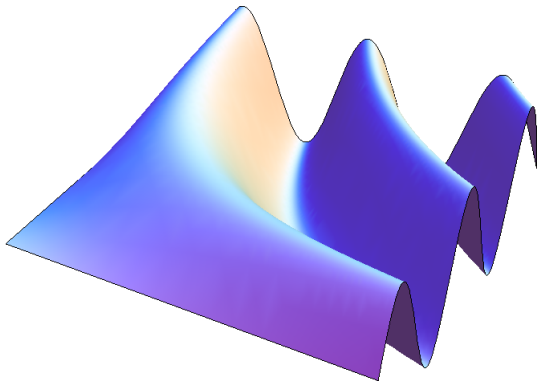
Find $(\boldsymbol{u}, p) \in (H, L^2(\Omega))$ such that

$$\int_\Omega \boldsymbol{v}\Big(\partial_t \boldsymbol{u} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u} - 2\nu \operatorname{div}(\nabla^s \boldsymbol{u}) + \nabla p\Big)\mathrm{dx} = \langle \boldsymbol{v}, \mathbf{f}\rangle, \qquad \forall \boldsymbol{v} \in V, \tag{4.5}$$

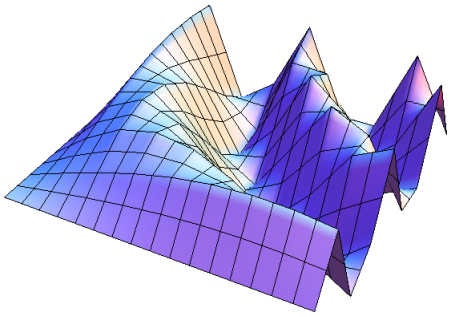
where V is the closure of $\mathcal{V}$ in the $L^2(\Omega)$ norm and H is the closure of $\mathcal{V}$ in the $H_0^1(\Omega)$. Figure 4.1 shows a visual of the difference between some classical and weak solutions.

4.1.1 Existence and Uniqueness

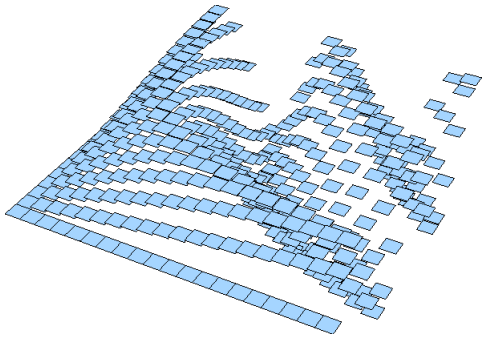
The existence and uniqueness theory of the Navier-Stokes equations is non-trivial and incomplete. The main problem is that the equations have elliptic, parabolic and hyperbolic characteristics all tangled together. Leray [54] was the first to prove the existence of weak solutions for (4.5) using sequences of regularized solutions and a compactness argument. Another proof was presented by Lemarie-Rieusset [53] for the



Smooth C^2 solution



Weak H^1 solution



Weak discontinuous Galerkin solution

Figure 4.1: The different types of solutions for the Navier-Stokes equations.

existence of Leray weak solutions in $\mathbb{R}^3$. For bounded domains with no boundaries, Témam [76] presented a proof using spectral decomposition. For further elaboration, the reader is advised to read Lions [55] and Fernández-Cara and Guillén [25]. As for uniqueness, it has been proven for $d = 2$ (c.f. Ladyzhenskaya and Silverman [51]). The case for $d = 3$ is unsolved and is one of the Clay millennium prize problems which has the value of one million dollars.

Efficient approximation of the Navier-Stokes equations is also difficult (c.f. Liu and Walkington [56] among others). For the discretization of (4.1)-(4.2) to be stable, it is important for the two spaces for velocity and pressure be "compatible." This means that the Ladyženskaja-Babuška-Brezzi (LBB) condition or the inf-sup condition must

be satisfied (c.f. Girault and Raviart [26] and Ern and Guermond [24, p.100]):

$$\exists \beta_h > 0, \qquad \inf_{\boldsymbol{u}_h \in M_h} \sup_{p_h \in X_h} \frac{\int_\Omega \operatorname{div}(\boldsymbol{u}_h) p_h \, \mathrm{dx}}{\|\boldsymbol{u}_h\|_{M_h} \|p_h\|_{X_h}} > \beta_h, \tag{4.6}$$

where $X_h \subset H, M_h \subset L^2(\Omega)$ are finite dimensional subspaces. This is both a necessary and sufficient condition for saddle point problems to be well posed when using Galerkin discretization.

4.2 Numerical Methods

Here, we will talk about the spatial discretization then three methods for time discretization and handling of the saddle point problem.

4.2.1 Spatial Discretization

There is no special inf-sup compatibility requirements for the spaces of density ρ and velocity field $\boldsymbol{u}$. The spaces can be chosen as needed. The compatibility for velocity and pressure spaces still holds as described in section 4.1.

Let us ignore the nonlinear term for the time being. To discretize the momentum equation (4.1)-(4.2), we need to define the discretization spaces. There are many standard ways to address it, but we will follow the notation in Ern and Guermond [24, p.208]. Consider the following discrete problem:

Find $(u_h, p_h) \in (X_h, M_h)$ such that:

$$\begin{cases} a(u_h, v_h) + b(v_h, p_h) = f(v_h), & \forall v_h \in X_h, \\ b(u_h, q_h) = 0, & \forall q_h \in M_h. \end{cases} \tag{4.7}$$

Let N_u, N_p be the dimensions of the subspaces $X_h \subset H$ and $M_h \subset L^2(\Omega)$ respectively. Let $\{v_h^i\}_{1 \le i \le N_u}$ be the basis of X_h and $\{q_h^i\}_{1 \le i \le N_p}$ be the basis of M_h. In the

context of finite elements, the basis are the global shape function. For every discrete $u_h = \sum_{i=1}^{N_u} u_i v_h^i$ in X_h and $p_h = \sum_{i=1}^{N_p} p_i q_h^i$ in X_h, define $U = \{u^1, \ \dots \ , u^{N_u}\}^\top$ in $\mathbb{R}^{N_u}$ and $P = \{p^1, \ \dots \ , p^{N_p}\}^\top$ in $\mathbb{R}^{N_p}$. Note that the map between u_h and U and between p_h and P is a bijection because $\{v_h^i\}_{1 \leq i \leq N_u}$ and $\{q_h^i\}_{1 \leq i \leq N_p}$ are bases. Putting the expansions of u_h and p_h in (4.7) and choosing the test functions as part of X_h and M_h, we get the following linear system:

$$\begin{bmatrix} A & B^\top \\ B & 0 \end{bmatrix} \begin{bmatrix} U \\ P \end{bmatrix} = \begin{bmatrix} F \\ 0 \end{bmatrix}, \tag{4.8}$$

where the block matrices $A \in \mathbb{R}^{N_u, N_u}$ and $B \in \mathbb{R}^{N_p, N_u}$ are defined as $A_{ij} = a(v_h^j, v_h^i)$ and $B_{ij} = b(v_h^j, q_h^i)$ and the vectors $F \in \mathbb{R}^{N_u}$ such that $F_i = f(v_h^i)$. The definitions of A and B will be elaborated in the next section.

This linear system is difficult to solve because the matrix is indefinite due to the saddle point structure. The saddle point comes from the fact that the velocity is constrained by the incompressibility condition $\operatorname{div}(\boldsymbol{u}) = 0$. The pressure acts like a Lagrange multiplier constraining the velocity to solenoidal fields.

Many methods exist to address this problem. One simple approach is to use the Uzawa method. Another more efficient approach for approximating the solution is by discretizing using fractional time stepping or projection methods algorithms. Projection methods split the viscous part and the incompressibility constraint of the equations as first done by Chorin [16] and Témam [75]. The last method we will address is the "Artificial Compressibility" method which will be discussed in section 4.2.4.

Remark. It is worth mentioning that the methods described above encompass steps to handle both the time stepping and the saddle point problem. The weak form (4.7)

only describes the space discretization to illustrate the saddle point problem alone. Each of the projection method and the artificial compressibility method addresses the saddle point problem and time stepping together in a different ways, all of which will be elaborated on in the next sections.

4.2.2 Uzawa Iteration Method

Uzawa [81] was the first to propose converting the saddle point problem (4.8) to the following block matrix:

$$\begin{bmatrix} A & B^\top \\ 0 & S \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ p \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ BA^{-1}\mathbf{f} \end{bmatrix}, \tag{4.9}$$

where $A_{ij} = \int_\Omega v_h^j v_h^i + \nabla v_h^j : \nabla v_h^i, B_{ij} = -\int_\Omega q_h^i \mathrm{div}\, v_h^j$. If A is symmetric positive definite then $S := BA^{-1}B^\top$ is also symmetric positive definite (S is also known as the *Schur complement* of A). Now, we can apply the gradient descent algorithm or conjugate gradient algorithm (c.f. Girault and Raviart [26]).

This algorithm is known to converge slowly because of the condition number of S. To address this issue, one can either precondition S properly (Bramble et al. [11]) or split (4.8) to solve for velocity first then pressure (Ern and Guermond [24, p. 306]). In the next section, we will attempt to do the latter.

4.2.3 Projection Method

As described in the previous section, one way to manage the complexity of velocity and pressure coupling in the Navier-Stokes system is to split solving for the velocity and pressure. The unique splitting turns out to be the Helmholtz decomposition of the L^2 space:

Theorem 4.2.1. (Helmholtz Decomposition) Let $\Omega \subset \mathbb{R}^d$ be a domain with a

Lipschitz boundary. Then the following orthogonal decomposition holds

$$L^2(\Omega) = H_{div=0}(\Omega) \oplus \nabla H^1(\Omega),$$

where

$$H_{div=0}(\Omega) := \{\boldsymbol{u} \in L^2(\Omega) \mid \operatorname{div}(\boldsymbol{u}) = 0, \boldsymbol{u} \cdot \boldsymbol{n} = 0\}, \tag{4.10}$$

$$\nabla H^1(\Omega) := \{\boldsymbol{u} \in L^2(\Omega) \mid \exists q \in H^1(\Omega) : \boldsymbol{u} = \nabla q\}. \tag{4.11}$$

Proof. Refer to Témam [76, Theorem 1.4]. □

The idea is to solve the velocity without the incompressibility constraint then ''project'' the solution onto $H_{div=0}(\Omega)$. The common part of the projection method consists of solving a Poisson equation, a scalar variable Φ (usually the pressure) at time $n+1$ using an equation that looks like:

$$-\operatorname{div}(\nabla\Phi) = f, \quad \partial_n \Phi|_{\partial\Omega} = 0.$$

which has the disadvantage of inducing a linear system that has a condition number that scales $\mathcal{O}(h^{-2})$. When the mesh is very small, this would become a bottleneck.

In this section, we will only present pressure correction projection schemes. We will mention in passing that there are other classes of projection schemes: velocity correction projection scheme and consistent projection scheme (c.f. Guermond et al. [40]).

Remark. Please note that, in the next algorithms, we will switch between the Laplacian $-\Delta\boldsymbol{u}$ and the divergence of the skew-symmetric tensor $-2\operatorname{div}(\nabla^s\boldsymbol{u})$ in the viscous term in the momentum equation. They are identical when $\operatorname{div}\boldsymbol{u} = 0$ but imply

different natural boundary conditions.

The first projection scheme was proposed by Chorin/Temam (1968/1969) which goes as follows:

1. (Viscous) Prediction:

$$\begin{cases} \frac{1}{\Delta t}(\tilde{\boldsymbol{u}}^{k+1} - \boldsymbol{u}^k) - \nu\Delta\tilde{\boldsymbol{u}}^{k+1} = \boldsymbol{f}^{k+1}, \\ \tilde{\boldsymbol{u}}^{k+1}|_{\partial\Omega} = 0. \end{cases}$$

2. Projection:

$$\begin{cases} \frac{1}{\Delta t}(\boldsymbol{u}^{k+1} - \tilde{\boldsymbol{u}}^{k+1}) - \nabla\phi^{k+1} = 0, \\ \operatorname{div}(\boldsymbol{u}^{k+1}) = 0, \quad \boldsymbol{u}^{k+1}\cdot\boldsymbol{n}|_{\partial\Omega} = 0. \end{cases}$$

3. Pressure Correction: $p^{k+1} = \phi^{k+1}$.

where $\tilde{\boldsymbol{u}}$ is an intermediate unconstrained velocity field. This is a simple and popular algorithm. It is of order $\mathcal{O}(\Delta t)$ in the L^2 norm but only of $\mathcal{O}(\Delta t^{\frac{1}{2}})$ in the H^1 norm (c.f. Rannacher [64], Shen [68] for proofs). This stems from the fact that we enforce the extra (artificial) boundary condition in the projection step. Also, because velocity is ''split'' from pressure, it has an irreducible splitting error of $\mathcal{O}(\Delta t)$ (Guermond and Quartapelle [36]). To overcome this limitation, we use the *incremental* projection method.

The incremental projection uses the the pressure explicitly in the viscous prediction step and then correct it appropriately afterwards (cf. Goda [27], van Kan [83]). This reduces the split between velocity and pressure. The algorithm goes as follows:

1. Prediction:

$$\begin{cases} \frac{1}{2\Delta t}(3\tilde{\boldsymbol{u}}^{k+1} - 4\boldsymbol{u}^k + \boldsymbol{u}^{k-1}) - \nu\Delta\tilde{\boldsymbol{u}}^{k+1} + \nabla p^k = \boldsymbol{f}^{k+1}, \\ \tilde{\boldsymbol{u}}^{k+1}|_{\partial\Omega} = 0. \end{cases}$$

2. Projection:

$$\begin{cases} \frac{1}{2\Delta t}(3\boldsymbol{u}^{k+1} - 3\tilde{\boldsymbol{u}}^{k+1}) + \nabla\phi^{k+1} = 0, \\ \operatorname{div}(\boldsymbol{u}^{k+1}) = 0, \quad \boldsymbol{u}^{k+1}\cdot\boldsymbol{n}|_{\partial\Omega} = 0. \end{cases}$$

3. Pressure Correction: $p^{k+1} = p^k + \phi^{k+1}$.

The incremental projection method has the improved error of $\mathcal{O}(\Delta t^2)$ in the L^2 norm and $\mathcal{O}(\Delta t)$ in the H^1 norm. Shen [71] has proved the semi-discrete case and Guermond [30] proved it for the fully discrete case in general domains. We also have an improved but still irreducible splitting error of $\mathcal{O}(\Delta t^2)$ (Guermond and Quartapelle [36]). However, we still enforce the artificial boundary condition. This means that it is not worth using a time stepping scheme better than second order.

Lastly, we will describe the *incremental projection method in rotational form* first proposed by Timmermans et al. [79]. It uses the identity $\Delta\boldsymbol{u} = \nabla\operatorname{div}\boldsymbol{u} - \nabla\times\nabla\times\boldsymbol{u}$ to implicitly impose a consistent boundary condition on the pressure. This modification was shown by Guermond and Shen [38] to produce the best error so far for a projection scheme: $\mathcal{O}(\Delta t^2)$ in the L^2 norm and $\mathcal{O}(\Delta t^{\frac{3}{2}})$ in the H^1 norm.

We will show the projection method in rotational form presented in Guermond et al. [43] where we substitute the intermediate unconstrained velocity field $\tilde{\boldsymbol{u}}$ inside the prediction step. Initialize the new variable $\varphi^0 = p^0, q^0 = 0$ and denote the incremental steps for φ, q as $\delta\varphi, \delta q$ respectively such that $\sum \delta q^k = q^k$ and for φ also.

1. Prediction:

$$
\begin{aligned}
&p^* = p^n + \frac{1}{3}\Big(4\delta\psi^n - \delta\psi^{n-1}\Big),\\
&\begin{cases}
\dfrac{3\boldsymbol{u}^{n+1} - 4\boldsymbol{u}^n + \boldsymbol{u}^{n-1}}{2\Delta t} - \mu\Delta\boldsymbol{u}^{n+1} + \nabla p^\star = \boldsymbol{f}^{\,n+1},\\
\boldsymbol{u}^{n+1}\big|_{\partial\Omega} = 0,
\end{cases}
\end{aligned}
\tag{4.12}
$$

2. Projection:

$$\Delta\delta\psi^{n+1} = \frac{3}{2\Delta t}\mathrm{div}\,(\boldsymbol{u}^{k+1}), \quad \partial_n\delta\psi^{n+1} = 0, \tag{4.13}$$

$$\delta q^{n+1} = -\mathrm{div}\,(\boldsymbol{u}^{n+1}), \tag{4.14}$$

3. Pressure Correction: $p^{n+1} = \psi^{n+1} - \mu q^{n+1}$.

The difference, you will notice, is the addition of $-\mu q^{n+1}$ in the pressure correction step. This gives the rotational form its higher accuracy. However, there is a penalty for doing so; namely, the tangential component of $\boldsymbol{u}$ is not correct and we get the suboptimality in the H^1 norm convergence rate. This is the scheme that we implemented in our code.

4.2.4 Artificial Compressibility Method

The last method that we will describe in this section is the ‘‘Artificial Compressibility'' method proposed by Chorin [16]. Instead of the incompressibility constraint, he used the equation of dynamic pressure at low Mach numbers (derived from the compressible Navier-Stokes energy equation):

$$\frac{Dp}{Dt} + \rho c^2\mathrm{div}\,(\boldsymbol{u}) \;=\; 0,$$

where c is the speed of sound (Drikakis and Rider [23, (2.46)]).

This leads to the Chorin's formulation:

$$\begin{cases} \partial_t \boldsymbol{u}_\epsilon + (\boldsymbol{u}_\epsilon \cdot \nabla)\boldsymbol{u}_\epsilon - \Delta \boldsymbol{u}_\epsilon + \nabla p_\epsilon = \mathbf{f}, \\ \epsilon \partial_t p_\epsilon + \operatorname{div} \boldsymbol{u}_\epsilon = 0, \\ \boldsymbol{u}|_{\partial\Omega} = 0, & \text{in } (0, T], \\ \boldsymbol{u}|_{t=0} = \boldsymbol{u}_0, \quad p(0) = p_0, & \text{in } \Omega. \end{cases} \tag{4.15}$$

where $\epsilon \partial_t p_\epsilon$ is a perturbation of the incompressibility condition that disappears at the limit of $\epsilon \to 0$. It is worth noting that ϵ is not necessarily related to the speed of sound c. The idea is to draw inspiration from the equation of dynamic pressure at low Mach numbers and replace the difficult incompressibility constraint with a more manageable one that converges to the correct constraint in the limit. This perturbation suggests the following first-order approximation (the rest of the section relies heavily on the algorithm buildup and setup in Guermond and Minev [31]):

$$\begin{cases} \dfrac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^n}{\Delta t} - \Delta \boldsymbol{u}^{n+1} + \nabla p^{n+1} = \mathbf{f}^{n+1} - \mathbf{nl}_0^n, \quad \boldsymbol{u}^{n+1}|_{\partial\Omega} = 0, \\ \dfrac{\epsilon}{\Delta t}\left(p^{n+1} - p^n\right) + \operatorname{div} \boldsymbol{u}^{n+1} = 0, \end{cases} \tag{4.16}$$

where $\mathbf{nl}_0^n = (\boldsymbol{u}^n \cdot \nabla)\boldsymbol{u}^n$ is the nonlinearity handled explicitly. To disentangle the two equations, we substitute the value of the second equation in the first:

$$\begin{cases} \dfrac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^n}{\Delta t} - \Delta \boldsymbol{u}^{n+1} + \nabla \left(p^n - \dfrac{\Delta t}{\epsilon} \operatorname{div} \boldsymbol{u}^{n+1} \right) = \mathbf{f}^{n+1} - \mathbf{nl}_0^n, \quad \boldsymbol{u}^{n+1}|_{\partial\Omega} = 0, \\ p^{n+1} = p^n - \dfrac{\Delta t}{\epsilon} \operatorname{div} \boldsymbol{u}^{n+1}, \end{cases} \tag{4.17}$$

which has been shown in Shen [70, proposition 5.1] to be first order in both H^1-norm

and L^2-norm when $\epsilon \sim \Delta t$. This method has a big advantage over splitting schemes discussed in section 4.2.3 because there is no splitting anymore; thus the error can be improved beyond the irreducible error limits induced by splitting the velocity and pressure. Moreover, the complexity of solving an operator like $\boldsymbol{v} - \Delta t(\nabla \operatorname{div} \boldsymbol{v} + \Delta \boldsymbol{v}) = \boldsymbol{b}$ is $\mathcal{O}(\Delta t h^{-2})$. This is good because it is comparable to solving a parabolic equation implicitly.

4.2.4.1 Higher-Order Artificial Compressibility Schemes

We can extend the first order scheme further by adding a second order perturbation in:

$$\begin{cases} \partial_t \boldsymbol{u}_\epsilon + (\boldsymbol{u}_\epsilon \cdot \nabla)\boldsymbol{u}_\epsilon - \Delta \boldsymbol{u}_\epsilon + \nabla p = \mathbf{f}, & \text{in } \Omega \times (0,T], \\ \epsilon \partial_{tt} p + \operatorname{div} \boldsymbol{u}_\epsilon = 0, & \text{in } \Omega \times (0,T], \\ \boldsymbol{u}_\epsilon|_{\partial\Omega} = 0, & \text{in } (0,T], \\ \boldsymbol{u}_\epsilon|_{t=0} = \boldsymbol{u}_0, \quad p(0) = p_0, \quad \partial_t p|_{t=0} = \partial p(0), & \text{in } \Omega. \end{cases} \tag{4.18}$$

Shen [69] has showed that the limit of the continuous version (4.18) is unstable but Guermond and Minev [31] proved the discrete version can be stabilized if ϵ is small enough (e.g.$\mathcal{O}(\Delta t^3)$). Thus, (4.18) becomes:

$$\begin{cases} \dfrac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^n}{\Delta t} - \Delta \boldsymbol{u}^{n+1} + \nabla p^{n+1} = \mathbf{f}^{n+1}, \boldsymbol{u}^{n+1}|_{\partial\Omega} = 0, \\ \dfrac{\epsilon}{\Delta t^2}(p^{n+1} - 2p^n + p^{n-1}) + \operatorname{div} \boldsymbol{u}^{n+1} = 0. \end{cases} \tag{4.19}$$

This will create a system that is officially $\mathcal{O}(\Delta t^2)$ in time. However, if we choose $\epsilon \sim \Delta t^3$, solving (4.19) would be prohibitively expensive because of the linear system induced by an elliptic problem like $\boldsymbol{v} - \nabla \operatorname{div} \boldsymbol{v} - \Delta t \Delta \boldsymbol{v} = \boldsymbol{b}$ behaves like $\mathcal{O}(h^{-2})$.

This is just as bad as a Poisson operator as explained in the previous section.

To overcome this complexity, Guermond and Minev [31] proposed a bootstrapping technique. The idea is as follows; Let $(\boldsymbol{u}, p)$ be a solution to (4.1)-(4.2) and let r be an approximation of p (both smooth in time). Then consider the following perturbation of the artificial compressibility:

$$\begin{cases} \partial_t \boldsymbol{w} - 2\mathrm{div}\,(\nabla^s \boldsymbol{w}) + \nabla s = f, \qquad \boldsymbol{w}|_\Gamma = 0, \quad \boldsymbol{w}|_{t=0} = \boldsymbol{u}_0, \\ \epsilon \partial_t (s - r) + \mathrm{div}\,\boldsymbol{w} = 0, \qquad\qquad\qquad\qquad\qquad\quad s|_{t=0} = p_0. \end{cases} \tag{4.20}$$

Denote $\boldsymbol{e} := \boldsymbol{u} - \boldsymbol{w}$ and $\delta := p - s$ then subtract (4.20) from (4.1)-(4.2) then add $\epsilon \partial_t p$ to get:

$$\begin{cases} \partial_t \boldsymbol{e} - 2\mathrm{div}\,(\nabla^s \boldsymbol{e}) + \nabla \delta = 0, \qquad \boldsymbol{e}|_\Gamma = 0, \quad \boldsymbol{e}|_{t=0} = \boldsymbol{u}_0, \\ \epsilon \partial_t \delta + \mathrm{div}\,\boldsymbol{e} = \epsilon \partial_t (p - r), \qquad\qquad\qquad\qquad\quad \delta|_{t=0} = 0. \end{cases} \tag{4.21}$$

Notice that if r is an $\mathcal{O}(\epsilon^k)$ approximation of p, then $\epsilon \partial_t (p - r)$ is of order $\mathcal{O}(\epsilon^{k+1})$. Moreover, if the mass equation in (4.20) is stable when perturbations are introduced, then $\boldsymbol{e} = \mathcal{O}(\epsilon^{k+1})$ and $\delta = \mathcal{O}(\epsilon^{k+1})$ as well. This means that the accuracy of the pair $(\boldsymbol{w}, s)$ increased by one order in ϵ (c.f. Guermond and Minev [31] for stability analysis).

Guermond and Minev [31] show how to use BDF1 to reach a sequence of solutions $(\boldsymbol{w}_1, s_1) = \mathcal{O}(\epsilon)$ then BDF2 to get a more accurate sequence of solutions $(\boldsymbol{w}_2, s_2) = \mathcal{O}(\epsilon^2)$ which is as follows:

$$\text{for } n \geq 0 \begin{cases} \dfrac{\boldsymbol{w}_1^{n+1} - \boldsymbol{w}_1^n}{\Delta t} - 2\mathrm{div}\,(\nabla^s \boldsymbol{w}_1^{n+1}) + \nabla s_1^{n+1} = \mathbf{f}^{n+1}, \\ s_1^{n+1} - s_1^n + \lambda \mathrm{div}\,\boldsymbol{w}_1^{n+1} = 0, \end{cases} \tag{4.22}$$

$$\text{for } n \geq 1 \begin{cases} \dfrac{3\boldsymbol{w}_2^{n+1} - 4\boldsymbol{w}_2^n + \boldsymbol{w}_2^{n-1}}{2\Delta t} - 2\text{div}\,(\nabla^s \boldsymbol{w}_2^{n+1}) + \nabla s_2^{n+1} = \mathbf{f}^{n+1}, \\ (s_2^{n+1} - s_2^n) - (s_1^{n+1} - s_1^n) + \lambda \text{div}\, \boldsymbol{w}_1^{n+1} = 0, \end{cases} \tag{4.23}$$

with $\boldsymbol{w}_1^0 = \boldsymbol{u}_0, s_1^0 = p_0$ and $\boldsymbol{w}_2^0 = \boldsymbol{u}_0, \boldsymbol{w}_2^1 = \boldsymbol{w}_1^1, s_2^1 = s_1^1$.

We will, however, focus here on a method called ''deferred correction technique'' discussed in the same reference above and Kress and Gustafsson [49]. The idea is to construct an approximation of u^n of order $\mathcal{O}(\Delta t^k)$ as follows: $u(t^n) = u_0^n + \Delta t u_1^n + \Delta t^2 u_2^n + \cdots + \Delta t^k u_k^n + \mathcal{O}(\Delta t^{k+1})$ where $u_0^n, u_1^n, \ldots, u_k^n$ are successive corrections computed in sequence. For example, consider for example the following simple ODE:

$$\partial_t u(t) = f(t), \quad u(0) = u_0, \tag{4.24}$$

where f is smooth. We will use Euler method for simplicity and robustness. We will also use implicit Euler for the stiff part of the ODE and explicit for the remainder (also known as implicit-explicit time stepping method (IMEX) as shown in Ascher et al. [3] and others). For $k = 3$, the following is a Taylor expansion:

$$u(t^n) = u(t^{n+1}) - \Delta t \partial_t u(t^{n+1}) + \frac{\Delta t^2}{2} \partial_{tt} u(t^{n+1}) - \frac{\Delta t^3}{6} \partial_{ttt} u(t^{n+1}) + \mathcal{O}(\Delta t^4), \tag{4.25}$$

or

$$\frac{u(t^{n+1}) - u(t^n)}{\Delta t} = f(t^{n+1}) - \frac{\Delta t}{2} \partial_{tt} u(t^{n+1}) + \frac{\Delta t^2}{6} \partial_{ttt} u(t^{n+1}) + \mathcal{O}(\Delta t^3). \tag{4.26}$$

Inserting $u_0^n + \Delta t u_1^n + \Delta t^2 u_2^n + \mathcal{O}(\Delta t^3)$ for $u(t^n)$ and $u(t^{n+1})$ and regrouping the terms

with the same order, we get:

$$\frac{u_0^{n+1} - u_0^n}{\Delta t} = f(t^{n+1}), \tag{4.27}$$

$$\frac{u_1^{n+1} - u_1^n}{\Delta t} = -\frac{1}{2}\partial_{tt} u_0^{n+1}, \tag{4.28}$$

$$\frac{u_2^{n+1} - u_2^n}{\Delta t} = -\frac{1}{2}\partial_{tt} u_1^{n+1} + \frac{1}{6}\partial_{ttt} u_0^{n+1}. \tag{4.29}$$

Lastly, we approximate the high order derivatives with simple divided differences. This choice makes approximating derivatives easier but introduces a technicality; higher order corrections are delayed by one stage. This is illustrated as follows:

$$\text{for } n \geq 0 \begin{cases} \dfrac{u_0^{n+1} - u_0^n}{\Delta t} = f^{n+1}, \\ du_0^{n+1} = \dfrac{u_0^{n+1} - u_0^n}{\Delta t}, \end{cases} \tag{4.30}$$

$$\text{for } n \geq 1 \begin{cases} d^2 u_0^{n+1} = \dfrac{du_0^{n+1} - du_0^n}{\Delta t}, \\ \dfrac{u_1^n - u_1^{n-1}}{\Delta t} = -\dfrac{1}{2} d^2 u_0^{n+1}, \\ du_1^n = \dfrac{u_1^n - u_1^{n-1}}{\Delta t}, \end{cases} \tag{4.31}$$

$$\text{for } n \geq 2 \begin{cases} d^2 u_1^n = \dfrac{du_1^n - du_1^{n-1}}{\Delta t}, \quad d^3 u_0^{n+1} = \dfrac{d^2 u_0^{n+1} - d^2 u_0^n}{\Delta t}, \\ \dfrac{u_2^{n-1} - u_2^{n-2}}{\Delta t} = -\dfrac{1}{2} d^2 u_1^n + \dfrac{1}{6} d^3 u_0^{n+1}, \\ u^{n-1} = u_0^{n-1} + \Delta t u_1^{n-1} + \Delta t^2 u_2^{n-1}, \end{cases} \tag{4.32}$$

Each stage is staggered in time, u_0^{n+1} is computed before u_1^n to be able to calculate $d^2 u_0^{n+1}$ and so on (a helpful diagram is shown in Guermond and Minev [31] in section 5.2).

Now, we extend the same technique the nonlinear PDEs. Consider the following:

$$\partial_t u + Au = f + Bu, \quad u(0) = u_0, \tag{4.33}$$

where $A : D(T) \subset X \to X$ is densely defined closed unbounded linear operator on a Banach space X and B is a well defined nonlinear operator on D(T). Let us further assume that A is maximal and monotone. This, with Hille-Yosida Theorem, allows A to generate contracting semi groups. Assume also that B still makes the above problem well posed. Then we can generalize (4.30)-(4.32) as follows:

$$\text{for } n \geq 0 \begin{cases} \mathrm{nl}_0^{n+1} = Bu_0^n, \\ \dfrac{u_0^{n+1} - u_0^n}{\Delta t} + Au_0^{n+1} = f^{n+1} - \mathrm{nl}_0^{n+1}, \\ du_0^{n+1} = \dfrac{u_0^{n+1} - u_0^n}{\Delta t}, \end{cases} \tag{4.34}$$

$$\text{for } n \geq 1 \begin{cases} d^2 u_0^{n+1} = \dfrac{du_0^{n+1} - du_0^n}{\Delta t}, \\ \mathrm{nl}_1^n = B(u_0^n + \Delta t u_1^{n-1}), \\ \dfrac{u_1^n - u_1^{n-1}}{\Delta t} + Au_1^n = -\dfrac{1}{2} d^2 u_0^{n+1} - \dfrac{\mathrm{nl}_1^n - \mathrm{nl}_0^n}{\Delta t}, \\ du_1^n = \dfrac{u_1^n - u_1^{n-1}}{\Delta t}, \end{cases} \tag{4.35}$$

$$\text{for } n \geq 2 \begin{cases} d^2 u_1^n = \dfrac{du_1^n - du_1^{n-1}}{\Delta t}, \quad d^3 u_0^{n+1} = \dfrac{d^2 u_0^{n+1} - d^2 u_0^n}{\Delta t}, \\ \mathrm{nl}_2^{n-1} = B(u_0^{n-1} + \Delta t u_1^{n-1} + \Delta t^2 u_2^{n-2}) \\ \dfrac{u_2^{n-1} - u_2^{n-2}}{\Delta t} + Au_2^{n-1} = -\dfrac{1}{2} d^2 u_1^n + \dfrac{1}{6} d^3 u_0^{n+1} - \dfrac{\mathrm{nl}_2^{n-1} - \mathrm{nl}_1^{n-1}}{\Delta t^2}, \\ u^{n-1} = u_0^{n-1} + \Delta t u_1^{n-1} + \Delta t^2 u_2^{n-1}, \end{cases} \tag{4.36}$$

Finally, we extend the algorithm developed above to the Navier-Stokes equations. Let us first focus on handling the bootstrapping for the pressure. Let us restrict

ourselves to the Stokes system for the time being. Denote $\boldsymbol{u}_0,\ \boldsymbol{u}_1$ and $p_0,\ p_1$ the first two corrections on the velocity and pressure respectively. Set $\boldsymbol{w}_1 := \boldsymbol{u}_0,\ s_1 := p_0,\ \boldsymbol{w}_2 = \boldsymbol{u}_0 + \Delta t \boldsymbol{u}_1,\ s_2 := p_0 + \Delta t p_1$. Then we get:

$$\begin{cases} \dfrac{\boldsymbol{w}_1^{n+1} - \boldsymbol{w}_1^n}{\Delta t} + A\boldsymbol{w}_1^{n+1} + \nabla s_1^{n+1} = \mathbf{f}^{n+1}, \\ s_1^{n+1} - s_1^n + \lambda \mathrm{div}\, \boldsymbol{w}_1^{n+1} = 0, \end{cases} \tag{4.37}$$

$$\begin{cases} \dfrac{\boldsymbol{w}_2^{n+1} - \boldsymbol{w}_2^n}{\Delta t} + A\boldsymbol{w}_2^{n+1} + \nabla s_2^{n+1} = \mathbf{f}^{n+1}, \\ (s_2^{n+1} - s_2^n) - (s_1^{n+1} - s_1^n) + \lambda \mathrm{div}\, \boldsymbol{w}_1^{n+1} = 0. \end{cases} \tag{4.38}$$

Subtract (4.37 from (4.38) and divide by Δt, we get the system to be solved to get the second set of corrections:

$$\begin{cases} \dfrac{\boldsymbol{u}_1^{n+1} - \boldsymbol{u}_1^n}{\Delta t} + A\boldsymbol{u}_1^{n+1} + \nabla p_1^{n+1} = \mathbf{f}^{n+1}, \\ (p_1^{n+1} - p_1^n) - \Delta t^{-1}(p_0^{n+1} - p_0^n) + \lambda \mathrm{div}\, \boldsymbol{u}_1^{n+1} = 0. \end{cases} \tag{4.39}$$

The structure of the mass conservation is the same at each level. At level k, the mass conservation will be $(p_k^{n+1} - p_k^n) - \Delta t^{-1}(p_{k-1}^{n+1} - p_{k-1}^n) + \lambda \mathrm{div}\, \boldsymbol{u}_k^{n+1} = 0$.

With that, we are ready to extend the algorithm to the Navier-Stokes. Set $B\boldsymbol{u} := \boldsymbol{u} \cdot \nabla \boldsymbol{u}, A\boldsymbol{u} := -2\mathrm{div}\,(\nabla^s \boldsymbol{u})$ and initialize $\boldsymbol{u}_0^0 = \boldsymbol{u}(0),\ p_0^0 = p(0),\ \boldsymbol{u}_1^0 = \boldsymbol{u}_2^0 = 0$. The third order defect correction bootstrapping method can be written as follows:

$$\text{for } n \geq 0 \begin{cases} \mathbf{nl}_0^{n+1} = B\boldsymbol{u}_0^n, \\ \begin{cases} \dfrac{\boldsymbol{u}_0^{n+1} - \boldsymbol{u}_0^n}{\Delta t} + A\boldsymbol{u}_0^{n+1} - \lambda \nabla \mathrm{div}\, \boldsymbol{u}_0^{n+1} + \nabla p_0^n = f^{n+1} - \mathbf{nl}_0^{n+1}, \\ p_0^{n+1} = p_0^n - \lambda \mathrm{div}\, \boldsymbol{u}_0^{n+1}, \end{cases} \\ d\boldsymbol{u}_0^{n+1} = \dfrac{\boldsymbol{u}_0^{n+1} - \boldsymbol{u}_0^n}{\Delta t}, \quad dp_0^{n+1} = \dfrac{p_0^{n+1} - p_0^n}{\Delta t}, \end{cases} \tag{4.40}$$

$$
\text{for } n \geq 1 \begin{cases}
d^2\boldsymbol{u}_0^{n+1} = \dfrac{d\boldsymbol{u}_0^{n+1} - d\boldsymbol{u}_0^n}{\Delta t}, \\
\mathbf{nl}_1^n = B(\boldsymbol{u}_0^n + \Delta t \boldsymbol{u}_1^{n-1}), \\
\begin{cases}
\dfrac{\boldsymbol{u}_1^n - \boldsymbol{u}_1^{n-1}}{\Delta t} + A\boldsymbol{u}_1^n - \lambda\nabla \mathrm{div}\, \boldsymbol{u}_1^n + \nabla(p_1^{n-1} + dp_0^n) \\
\qquad\qquad = -\dfrac{1}{2} d^2\boldsymbol{u}_0^{n+1} - \dfrac{\mathbf{nl}_1^n - \mathbf{nl}_0^n}{\Delta t}, \\
p_1^n = p_1^{n-1} + dp_0^n - \lambda \mathrm{div}\, \boldsymbol{u}_1^n,
\end{cases} \\
d\boldsymbol{u}_1^n = \dfrac{\boldsymbol{u}_1^n - \boldsymbol{u}_1^{n-1}}{\Delta t}, \quad dp_1^n = \dfrac{p_1^n - p_1^{n-1}}{\Delta t},
\end{cases} \tag{4.41}
$$

$$
\text{for } n \geq 2 \begin{cases}
d^2\boldsymbol{u}_1^n = \dfrac{d\boldsymbol{u}_1^n - d\boldsymbol{u}_1^{n-1}}{\Delta t}, \quad d^3\boldsymbol{u}_0^{n+1} = \dfrac{d^2\boldsymbol{u}_0^{n+1} - d^2\boldsymbol{u}_0^n}{\Delta t}, \\
\mathbf{nl}_2^{n-1} = B(\boldsymbol{u}_0^{n-1} + \Delta t \boldsymbol{u}_1^{n-1} + \Delta t^2 \boldsymbol{u}_2^{n-2}), \\
\begin{cases}
\dfrac{\boldsymbol{u}_2^{n-1} - \boldsymbol{u}_2^{n-2}}{\Delta t} + A\boldsymbol{u}_2^{n-1} - \lambda\nabla \mathrm{div}\, \boldsymbol{u}_2^{n-1} + \nabla(p_2^{n-2} + dp^{n-1}) \\
\qquad\qquad = -\dfrac{1}{2} d^2\boldsymbol{u}_1^n + \dfrac{1}{6} d^3\boldsymbol{u}_0^{n+1} - \dfrac{\mathbf{nl}_2^{n-1} - \mathbf{nl}_1^{n-1}}{\Delta t^2}, \\
p_2^{n-1} = p_2^{n-2} + dp_1^{n-1} - \lambda \mathrm{div}\, \boldsymbol{u}_2^{n-1},
\end{cases} \\
u^{n-1} = \boldsymbol{u}_0^{n-1} + \Delta t \boldsymbol{u}_1^{n-1} + \Delta t^2 \boldsymbol{u}_2^{n-1}, \\
p^{n-1} = p_0^{n-1} + \Delta t p_1^{n-1} + \Delta t^2 p_2^{n-1}.
\end{cases} \tag{4.42}
$$

Guermond and Minev [31] proved that the method is unconditionally stable and third order when $B = 0$.

4.3 Stabilization and Turbulence Modeling

The momentum equation is already stabilized by the presence of viscous term. This, however, may not be enough when the Reynolds number is high. In this situation, the momentum equation becomes advection-dominated and the same stability issues we faced in the transport equation are faced here. Even though stabilization appears to be a numerical technique to stabilize hyperbolic equations, ‘‘Turbulence Modeling’’

employees the same techniques to model physical turbulence with rapid changes in pressure and velocity. There are many methods to model turbulence: k-ϵ Model, Large Eddie Simulation (LES), etc. We will concentrate here on LES.

The idea behind LES is to split up the flow into large and small scales. Think, for example, of a hurricane. The large scales can be thought of as the main large vortex. The small scale are the small eddies that appear and disappear during the hurricane. They are small and, during simulation, are usually subgrid but have a significant amount of the energy of the system. For a good overview of LES, see John [47].

In the context of the Navier-Stokes equations, LES is introduced as a cell-wise targeted viscosity $\nu_K \geq 0$ to the term $-2\nu \mathrm{div}\left(\nabla^s \boldsymbol{u}\right)$. The result is a viscosity $\nu + \nu_K$. The classical Smagorinsky model (Smagorinsky [72]) uses:

$$\nu_K := C_s \delta_K^2 \|\nabla^s \boldsymbol{u}\|,$$

where C_s is the Smagorinsky constant, δ_k is the filter width (proportional to h_K). Guermond et al. [41] proposes an Entropy-Viscosity approach:

$$\nu_K := \min\left(C_m h_K |\boldsymbol{u}|, C_e h_K^2 \frac{|D_h(\boldsymbol{x},t)|}{\|\boldsymbol{u}_h^2\|_{L^\infty(\Omega)}}\right),$$

where

$$\begin{aligned} D_h(\boldsymbol{x},t) :=& \partial_t\left(\frac{1}{2}\boldsymbol{u}_h^2\right) + \mathrm{div}\left(\left(\frac{1}{2}\boldsymbol{u}_h^2 + p_h\right)\boldsymbol{u}_h\right) - Re^{-1}\Delta\left(\frac{1}{2}\boldsymbol{u}_h^2\right) \\ &+ Re^{-1}\left(\nabla \boldsymbol{u}_h\right)^2 - \mathbf{f}\cdot\boldsymbol{u}_h, \end{aligned}$$

is the entropy equation, h_K is the local mesh size, $\|\boldsymbol{u}_h^2\|_{L^\infty(\Omega)}$ is a normalizing term, and C_m, C_e are appropriate constants. The first term $C_m h_K |\boldsymbol{u}|$ is the first order artificial

viscosity. When the mesh is fine enough to simulate all the scales, $h_K^2 \, |D_h(\boldsymbol{x}, t)|$ is much smaller than the first-order artificial viscosity. This makes ν_K a consistent viscosity that vanishes when scales of all levels are resolved.

4.4 Linear Systems

Once a numerical method is chosen from the previous section, it is now time to build the linear systems to solve them. One of the main advantages of the Finite Element Method is that the choice and construction of the bases creates matrices that are sparse. This has a huge advantage in memory space and computational performance.

We will concentrate here on the projection and artificial compressibility methods. For Uzawa iterative method, see Ern and Guermond [24, p.212] or Girault and Raviart [26, p.74]. We implemented the projection scheme (4.12)-(4.14). The linear systems to solve it is as follows:

1. Assemble once and solve (4.12) as linear system $AU^{n+1} = b$ where

$$A_{ij} = \int_\Omega 3v_h^j v_h^i + 2\Delta t \nabla v_h^j \nabla v_h^i \mathrm{dx}, \tag{4.43}$$

$$b_i = \int_\Omega v_h^i \big(4u_n^i - u_{n-1}^i + 2\Delta t(f_n - \nabla p_\star)\big) \mathrm{dx}, \tag{4.44}$$

2. Then assemble once and solve the Poisson problem (4.13) $B\Phi^{n+1} = c$ where:

$$B_{ij} = \int_\Omega \nabla v_h^j \nabla v_h^i \mathrm{dx}, \tag{4.45}$$

$$c_i = -\int_\Omega \frac{3}{2\Delta t} v_h^i \mathrm{div}\, u_{n+1} \mathrm{dx}, \tag{4.46}$$

3. After that, assemble once and invert the mass matrix (4.14) $CQ^{n+1} = d$ where:

$$C_{ij} = \int_{\Omega} v_h^j v_h^i \mathrm{dx}, \tag{4.47}$$

$$d_i = \int_{\Omega} v_h^i \mathrm{div}\, u_{n+1} \mathrm{dx}, \tag{4.48}$$

4. Finally, sum $P^{n+1} = \sum_{i=1}^{n+1} \Phi^i - \mu Q^i$.

Notice that step 1 above has a block structure based on the dimension of the velocity field space Ω. For example, when $d = 2$:

$$A = \begin{bmatrix} A_u & A_{uv} \\ A_{vu} & A_v \end{bmatrix}. \tag{4.49}$$

For the equation (4.43), we know that $A_{uv} = A_{uv} = 0$ because the velocity components are uncoupled. If, however, we choose the viscous term $\mathrm{div}\,(\nabla^s \boldsymbol{u}) = \frac{1}{2}(\Delta \boldsymbol{u} + \nabla \mathrm{div}\, \boldsymbol{u})$, then $\nabla \mathrm{div}\, \boldsymbol{u}$ couples the components together. The $\nabla \mathrm{div}\, \boldsymbol{u}$ term is known to control the divergence of the solution especially for high Reynolds number (c.f. Olshanskii et al. [58], Olshanskii and Reusken [59], Heister [44]). The downside of that is that the coupled system takes longer to solve. Specifically, during the tests in section 4.5.1.1, the coupled version of the projection method took 15 seconds to run vs 10 seconds for the uncoupled version, a significant slowdown.

The uncoupled system creates a much simpler linear system. Each matrix block of $N_u/d \times N_u/d$ can be solved individually (if boundary conditions do not couple the blocks; e.g. only allowing tangential flow on boundaries not aligned with the coordinate axes). Therefore, solving d systems that look like $A_{u_i} U = b$ is easier than solving the complete matrix. Also, often, the matrix A_u is identical for each velocity component and requires assembling and preconditioning once for one component then

use it for the other components. The disadvantage of uncoupled systems is that the velocity components cannot be controlled implicitly.

The coupled system solves a $N_u \times N_u$ matrix that solves all the velocity components together. The coupling is often done through the $\nabla \mathrm{div}\, u$. This creates a system that is more difficult to solve because the coupling blocks are nonzero. However, the advantage is gaining the stabilizing effect of $\nabla \mathrm{div}\, u$ especially in high Reynolds numbers.

Also, note that the linear systems discussed above so far need to be assembled once. If, however, we include the nonlinear term implicitly, then the matrix needs to assembled every time step affecting the performance of algorithm.

4.5 Numerical Results

After explaining the schemes in detail in the last sections, we test them in this section and check if the numerical results agree with the theory.

4.5.1 Validation

As we did with the transport equation in the previous chapter, we will validate both the projection and artificial compressibility schemes with conforming manufactured solutions. We expect to get machine epsilon for the solution errors.

4.5.1.1 Projection Scheme

Using $\Omega = (0,1)^d$ domain with a uniform mesh and cell-wise $[\mathbb{Q}_2]^d/\mathbb{Q}_1$ Taylor-Hood continuous finite elements, we introduce the following simple linear polynomial manufactured solution for the momentum equation:

$$\boldsymbol{u}(\boldsymbol{x},t) = (1+t)\begin{pmatrix} x+y \\ x-y \end{pmatrix}, \qquad p(\boldsymbol{x},t) = (1+t)xy, \quad \text{when } d=2, \tag{4.50}$$

$$\boldsymbol{u}(\boldsymbol{x},t) = (1+t)\begin{pmatrix} 1+z \\ 1+x \\ 1+y \end{pmatrix}, \qquad p(\boldsymbol{x},t) = (1+t)xyz \quad \text{when } d=3. \tag{4.51}$$

We solve the equation (4.12) with $\mu = 1$ running until final time $T = 1$. The projection step is disabled, which means that the exact pressure is interpolated from the exact solution to the discrete space every time step. We enforce the following boundary condition $\boldsymbol{u}|_{\partial\Omega} = \boldsymbol{u}(\boldsymbol{x},t)|_{\partial\Omega}$. The source term is modified to reflect the exact solutions. As expected, table 4.1 shows that the error is machine epsilon (~ 0) which means that the algorithm reproduces the conforming manufactured solutions exactly.

	cells	$\boldsymbol{u}_{\text{dofs}}$	Δt	$\|e_{\boldsymbol{u}}\|_{L2}$	$\|e_{\boldsymbol{u}}\|_{H1}$
	16	162	1E-02	8E-16	1E-14
2D	64	578	5E-03	6E-15	4E-14
	256	2178	3E-03	2E-14	1E-13
	8	375	2E-02	1E-15	1E-14
3D	64	2187	1E-02	3E-15	3E-14
	512	14739	5E-03	9E-15	8E-14

Table 4.1: Error values for running conforming manufactured solutions in a unit cube. We get a machine epsilon as expected.

Now, we validate the scheme by running a convergence rate test. We use the same 2D setup as before with the following nonconforming manufactured solutions:

$$\boldsymbol{u}(\boldsymbol{x},t) = \begin{pmatrix} \cos(x) + \cos(y+t) \\ \sin(x) + \sin(y+t) \end{pmatrix}, \qquad p(\boldsymbol{x},t) = \cos(x+y+t). \tag{4.52}$$

We see in table 4.2 that we get the $\mathcal{O}(\Delta t^2)$ in the L^2 norm as expected. The H^1 norms are a bit higher than the expected $\mathcal{O}(\Delta t^{\frac{3}{2}})$.

cells	$\boldsymbol{u}_{\text{dofs}}$	p_{dofs}	Δt	$\|e_{\boldsymbol{u}}\|_{L2}$	rate	$\|e_{\boldsymbol{u}}\|_{H1}$	rate
256	4802	1089	2E-02	1.54E-04	-	1.04E-03	-
1024	18818	4225	1E-02	4.28E-05	1.85	3.11E-04	1.75
4096	74498	16641	5E-03	1.14E-05	1.9	9.01E-05	1.79
16384	296450	66049	2.5E-03	2.98E-06	1.94	2.57E-05	1.81
cells	$\boldsymbol{u}_{\text{dofs}}$	p_{dofs}	Δt	$\|e_p\|_{L2}$	rate	$\|e_p\|_{H1}$	rate
256	4802	1089	2E-02	1.37E-03	-	2.22E-02	-
1024	18818	4225	1E-02	4.10E-04	1.74	8.63E-03	1.36
4096	74498	16641	5E-03	1.18E-04	1.8	3.27E-03	1.4
16384	296450	66049	2.5E-03	3.31E-05	1.83	1.22E-03	1.42

Table 4.2: Convergence rate for the constant density projection method. The $\text{CFL}_{\max}$ is at 0.64.

4.5.1.2 Artificial Compressibility Scheme

We will verify the artificial compressibility method (4.40)-(4.42) in both 2D and 3D using discrete space and time conforming polynomial exact solutions to test for exactness and then a nonconforming exact solution for convergence rates.

Starting with $(0,1) \times (0,1)$ domain Ω, we introduce the following simple linear polynomial manufactured solution for the momentum equation:

$$\boldsymbol{u} = \frac{1}{2\sqrt{2}} \begin{pmatrix} y(1+t) \\ -x(1+t) \end{pmatrix}, \quad p = \frac{1}{2} xy(1+t), \tag{4.53}$$

where the constant coefficients normalize the values of the velocity and pressure such that $\max_{\Omega, 0 \le t \le 1} |\boldsymbol{u}| = 1$ and $\max_{\Omega, 0 \le t \le 1} p = 1$. The reason is that we want the

manufactured solutions in this and next sections to be scaled appropriately compared to each other.

We start with an initial cell refinement of $2^3 \times 2^3$ with $\mathbb{Q}_2/\mathbb{Q}_1$ Taylor-Hood element and $\Delta t = 0.01$ running to $T = 1$. Since (4.53) is linear in all variables, we expect exactness (machine epsilon) similar to what was done in the previous section. However, table 4.3 shows the convergence rate of 3 across the L^2 and H^1 norms and not the expected machine epsilon. One source of error from the non-linearity $\mathbf{nl}_k$ where it is always explicit in the (4.40)-(4.42) equations and an extrapolation from existing information.

Figure 4.2 shows the error versus time for the case with 256 cells. One can observe oscillations in the beginning of the solutions that die out at $t = 0.5$ and reach 2.4E-8 near $t = 1$. By looking at the plot $|\boldsymbol{u}(t) - \boldsymbol{u}^t|$ (not shown here), we observe waves that keep bouncing off the boundaries back and forth across the domain. This explains the oscillations as the waves are superpositioned in phase and out of phase over time until the kinetic viscosity reduces the energy of the waves and get the actual approximation error. It seems that the source of these waves comes from the boundary but we are unable to explain them.

We can get exactness with the simpler (4.30)-(4.32) scheme. Using the same setup above and:

$$\boldsymbol{u} = (1+t)^3 \begin{pmatrix} x \\ -y \end{pmatrix}, \quad \boldsymbol{u} = (1+t)^3 \begin{pmatrix} z \\ x \\ -y \end{pmatrix}, \tag{4.54}$$

for 2D and 3D respectively, we get the expected machine epsilon as shown in table 4.4. Notice that $\boldsymbol{u}$ is a vector with third order polynomials in time and, using a third order scheme, we reproduce the solution almost exactly.

cells	$\boldsymbol{u}_{\text{dofs}}$	p_{dofs}	Δt	$\|e_{\boldsymbol{u}}\|_{L2}$	rate	$\|e_{\boldsymbol{u}}\|_{H1}$	rate
64	578	81	0.01	5.58E-08	-	2.95E-07	-
256	2178	289	0.005	6.99E-09	3.00	3.69E-08	3.00
1024	8450	1089	0.0025	8.74E-10	3.00	4.61E-09	3.00
cells	$\boldsymbol{u}_{\text{dofs}}$	p_{dofs}	Δt	$\|e_p\|_{L2}$	rate	$\|e_p\|_{H1}$	rate
64	578	81	0.01	7.28E-07	-	4.31E-06	-
256	2178	289	0.005	9.08E-08	3.00	5.60E-07	2.95
1024	8450	1089	0.0025	1.13E-08	3.00	7.29E-08	2.94

Table 4.3: Errors for the artificial compressibility method using conforming manufactured solutions using $\mathbb{Q}_2$ elements.

	cells	$\boldsymbol{u}_{\text{dofs}}$	Δt	$\|e_{\boldsymbol{u}}\|_{L2}$	$\|e_{\boldsymbol{u}}\|_{H1}$
	16	338	0.02	2.71E-14	1.64E-12
2D	64	1250	0.01	3.66E-13	4.92E-11
	256	4802	0.005	9.67E-13	2.55E-10
	8	1029	0.02	3.47E-15	6.71E-14
3D	64	6591	0.01	7.45E-15	2.64E-13
	512	46875	0.005	1.64E-14	9.92E-13

Table 4.4: Error values for running conforming manufactured solutions in a unit cube using (4.30)-(4.32) scheme. We get a machine epsilon as expected.

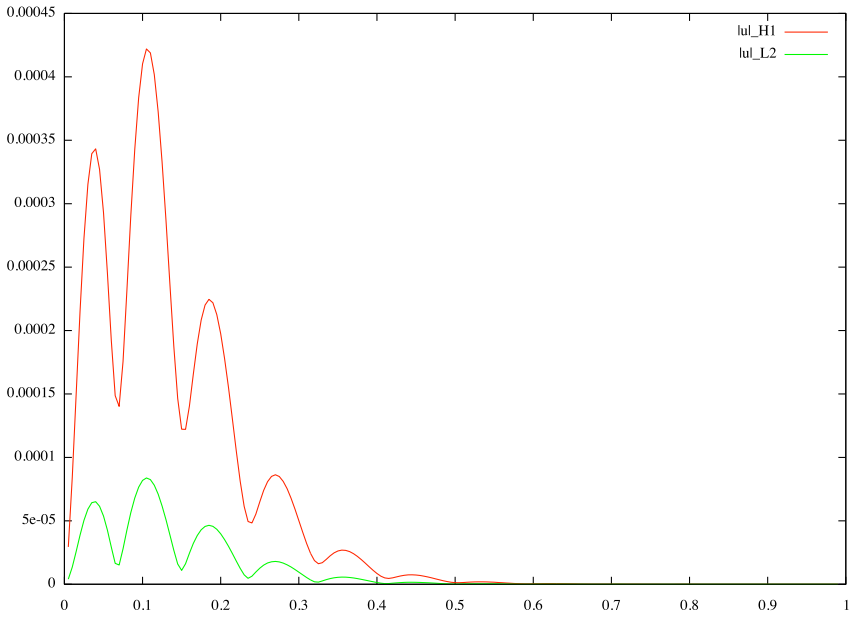


Figure 4.2: The errors $|\boldsymbol{u}(t) - \boldsymbol{u}^t|_{H^1}$ and $\|\boldsymbol{u}(t) - \boldsymbol{u}^t\|_{L^2}$ versus t of the artificial compressibility method with 578 dofs for $\boldsymbol{u}$. One can see the oscillations in the beginning of the scheme.

Now, we verify the convergence rates with nonconforming manufactured solutions. We use the same setup as above but with the nonconforming solutions (4.52). Table 4.5 has the convergence rates with respect to time. It shows a solid convergence rate of ~ 3 as expected.

4.5.2 Realistic Models

We will compare our simulation here with the lid cavity test used in Guermond et al. [39]. The domain is $\Omega = [0,1] \times [0,1] \times [-1,1]$ nonuniform mesh such the location of the mesh points are at:

$$x_i, y_i = \frac{1}{2} + \frac{1}{2}\cos\left(\pi \frac{i-1}{I-1}\right), 1 \leq i \leq I,$$

cells	$\boldsymbol{u}_{\text{dofs}}$	p_{dofs}	Δt	$\|e_{\boldsymbol{u}}\|_{L2}$	rate	$\|e_{\boldsymbol{u}}\|_{H1}$	rate
64	1250	289	0.01	1.99E-06	-	1.54E-05	-
256	4802	1089	0.005	2.32E-07	3.1	1.81E-06	3.09
1024	18818	4225	0.0025	2.80E-08	3.05	2.19E-07	3.05
cells	$\boldsymbol{u}_{\text{dofs}}$	p_{dofs}	Δt	$\|e_p\|_{L2}$	rate	$\|e_p\|_{H1}$	rate
64	1250	289	0.01	3.65E-05	-	8.99E-04	-
256	4802	1089	0.005	4.13E-06	3.14	1.48E-04	2.6
1024	18818	4225	0.0025	4.91E-07	3.07	2.74E-05	2.44

Table 4.5: Errors for the artificial compressibility method using nonconforming manufactured solutions and using $\mathbb{Q}_3$ elements.

$$z_i = \sin\left(\frac{\pi}{2}\frac{j-1}{J-1}\right), 1 \leq j \leq J.$$

We use $39 \times 39 \times 30$ $\mathbb{Q}_2/\mathbb{Q}_1$ tetrahedra elements. The initial condition is $\boldsymbol{u} = 0$ with boundary conditions $\boldsymbol{u} = 0$ everywhere but $x = 1$ where $\boldsymbol{u} = (0, 1, 0)^\top$. Taking advantage of the symmetry of the domain, we only simulate $z \in [0, 1]$ with a symmetric plane at $z = 0$. The Reynolds number used is $Re = 1000$ with time step $\Delta t = 0.005$. Since we work with constant density, we have no gravity driven forces and therefore $\mathbf{f} = 0$. We use the projection method (4.12)-(4.14).

Figure 4.3 shows the values calculated with ASPEN in black as follows: the vertical line is the value of $-\frac{1}{2}\boldsymbol{u}_x$ at $\{x = \frac{1}{2}, y \in [0, 1], z = 0\}$, and the horizontal line is $-\frac{1}{2}\boldsymbol{u}_y$ at $\{x =\in [0, 1], y = \frac{1}{2}, z = 0\}$. The black graph is superimposed on the orange graph obtained from [39] and we can see that it coincides almost exactly.

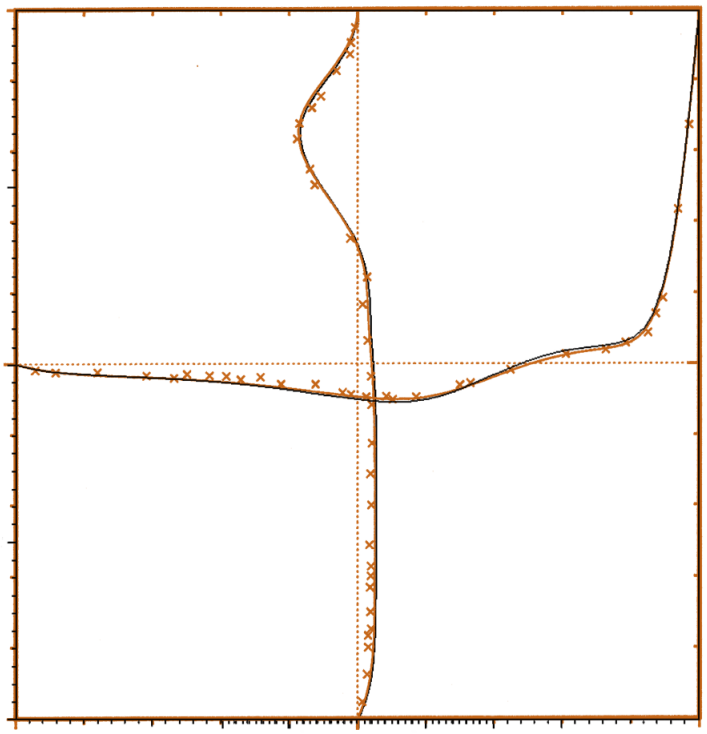


Figure 4.3: The velocity profiles calculated with ASPEN (black) superimposed on the orange results from [39]. The velocity profiles are calculated at the symmetry plane $z = 0$ at time $t = 4$. The vertical line is the values of $-\frac{1}{2}\boldsymbol{u}_x$ and the horizontal line is $-\frac{1}{2}\boldsymbol{u}_y$.

5. THE VARIABLE DENSITY NAVIER-STOKES EQUATION

In chapter one, we discussed the transport equation, which, when solved, results in a density field. Chapter two discusses the constant density Navier-Stokes equations which resolves the velocity field in the case of, well, constant density. In this chapter, we use both models to describe the variable density Navier-Stokes equations. This is needed when modeling two fluids that diffuse into each other (e.g. dye in water, fresh river water in salt water sea) or fluids that have distinct phases and do not mix (e.g. oil/water, gas/liquid mixtures, emulsions).

We will describe the mathematical model and show where the difficulty arises in solving the coupled system. Also, we will elaborate on the implications of adding the density field and how that will impact the linear systems. Then we will describe two methods to solve the variable density Navier-Stokes equations that extend the methods we saw in the last chapter.

5.1 The Mathematical Model

The *variable density incompressible Navier-Stokes equations* are defined as follows:

$$\partial_t \rho + \text{div}(\rho \boldsymbol{u}) = 0, \qquad \text{in } \Omega \times (0, T], \tag{5.1}$$

$$\rho[\partial_t \boldsymbol{u} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u}] - 2\mu \text{div}(\nabla^s \boldsymbol{u}) + \nabla p = \rho \boldsymbol{f}, \qquad \text{in } \Omega \times (0, T], \tag{5.2}$$

$$\text{div}(\boldsymbol{u}) = 0, \qquad \text{in } \Omega \times (0, T], \tag{5.3}$$

where $\Omega \subset \mathbb{R}^{2,3}$ and $\partial\Omega$ is the boundary, $\rho(\boldsymbol{x}, t)$ is the density at $(\boldsymbol{x}, t) \in \Omega \times [0, T]$, $\boldsymbol{u}(\boldsymbol{x}, t)$ is the velocity vector field, μ is the dynamic viscosity, and $p(\boldsymbol{x}, t)$ is the

pressure.

The equations above have the following initial and Dirichlet boundary conditions:

$$\begin{aligned} \boldsymbol{u}|_{t=0} &= \boldsymbol{u}_0, & \rho|_{t=0} &= \rho_0, & &\text{in } \Omega, \\ \boldsymbol{u}|_{\partial\Omega} &= 0, & \rho|_{\partial\Omega} &= \rho_0|_{\partial\Omega}, & &\text{in } [0,T], \end{aligned}$$

where ρ_0 and $\boldsymbol{u}_0$ are the initial conditions for density and velocity respectfully. As we described for the constant density Navier-Stokes equations in the previous chapter, the variable density Navier-Stokes equations is difficult to analyze mathematically because it has elliptic, parabolic and hyperbolic properties. Approximating (5.1)-(5.3) is also a more difficult task. Most approaches extend methods known for the constant density incompressible fluid flows to variable density flows. For example, see Bell and Marcus [9], Almgren et al. [1], Pyo and Shen [63], and Guermond and Quartapelle [35]. Guermond and Quartapelle [35] is the first —to the best of our knowledge —to show the stability for a variable density projection algorithm. The algorithm in Guermond and Quartapelle [35] uses two relatively expensive projections. A less expensive algorithm is presented by Pyo and Shen [63] with a single projection. However, no complete error analysis of the variable density projection methods can be found so far in the literature.

5.2 Numerical Methods

The numerical methods that will be described in this section are built upon the previous two chapters. We will describe the variable density projection and artificial compressibility schemes and contrast them with the constant density schemes.

5.2.1 Variable Density Second Order Projection Method

Last chapter, we explained in length the projection schemes. Here, we will extend only one scheme: incremental projection method in rotational form. This is the algorithm for the projection method that is second order in time and space. It assumes that there exists a $\rho_{\min}$ such that:

$$0 < \rho_{\min} \leq \inf_{\Omega} \rho_0,$$

and for all future time steps $k \geq 0$:

$$\rho_{\min} \leq \inf_{\Omega} \rho^k.$$

The idea for the variable density projection method described below comes from the conservative strong form of the momentum equation:

$$\partial_t(\rho \boldsymbol{u}) + \frac{1}{2}\mathrm{div}\,(\rho \boldsymbol{u} \otimes \boldsymbol{u}) + \cdots = \rho \mathbf{f}. \tag{5.4}$$

Using the product rule and rearranging, we get the mass conservation equation:

$$\underbrace{\boldsymbol{u}\big(\partial_t \rho + \mathrm{div}\,(\rho \boldsymbol{u})\big)}_{\omega=0} + \rho(\partial_t \boldsymbol{u} + \boldsymbol{u} \cdot \nabla \boldsymbol{u}) + \cdots = \rho \mathbf{f}. \tag{5.5}$$

We could remove ω but Guermond and Salgado [37] has used the term $\frac{1}{2}\mathrm{div}\,(\rho \boldsymbol{u})\boldsymbol{u}$ to prove the stability of the scheme below. So, we multiply ω by $\frac{1}{2}$ (it will change nothing as $\omega = 0$) because, when tested with $\boldsymbol{u}$, the expression $(\rho \partial_t \boldsymbol{u} + \frac{1}{2}\boldsymbol{u}\partial_t \rho)\boldsymbol{u} = \partial_t(\frac{1}{2}\rho \boldsymbol{u}^2)$. This term gives the kinetic energy conservation when integrated over space and time.

Now, we are ready to describe the second order variable density projection scheme.

Initialize the algorithm with $\rho^0 = \rho_0, \boldsymbol{u}^0 = \boldsymbol{u}_0, p^0 = p_0, \varphi^0 = q^0 = 0$ then proceed as follows:

1. Setup intermediate variables:

$$\begin{aligned}
\rho^* &= \rho^{n+1} + \frac{1}{6}\mathrm{BDF}_2(\rho^{n+1}), \\
&\quad \text{where } \mathrm{BDF}_2(\phi^{n+1}) = 3\phi^{n+1} - 4\phi^n + \phi^{n-1}, \\
p^* &= p^n + \frac{1}{3}\Big(4\delta\psi^n - \delta\psi^{n-1}\Big), \\
\boldsymbol{u}^* &= 2\boldsymbol{u}^n - \boldsymbol{u}^{n-1}.
\end{aligned}$$

2. Prediction:

$$\begin{aligned}
&\frac{3\rho^*\boldsymbol{u}^{n+1} - 4\rho^{n+1}\boldsymbol{u}^n + \rho^{n+1}\boldsymbol{u}^{n-1}}{2\Delta t} - \rho^{n+1}\boldsymbol{u}^* \cdot \nabla\boldsymbol{u}^{n+1} \\
&\qquad + \frac{1}{2}\mathrm{div}\left(\rho^{k+1}\boldsymbol{u}^*\right)\boldsymbol{u}^{n+1} - \mu\Delta\boldsymbol{u}^{n+1} + \nabla p^\star = \rho^{n+1}\boldsymbol{f}^{\,n+1}, \quad \boldsymbol{u}^{n+1}\big|_{\partial\Omega} = 0,
\end{aligned}$$

3. Projection:

$$\begin{aligned}
\Delta\delta\psi^{n+1} &= \frac{3\rho_{\min}}{2\Delta t}\mathrm{div}\,(\boldsymbol{u}^{k+1}), \quad \partial_n\delta\psi^{n+1} = 0, \\
\delta q^{n+1} &= -\mathrm{div}\,(\boldsymbol{u}^{n+1}),
\end{aligned}$$

4. Pressure correction: $p^{n+1} = \psi^{n+1} - \mu q^{n+1}$.

This variable density projection method still maintains the same errors of its constant density counterpart (4.12)-(4.14) and the stability proof can be found in Guermond and Salgado [37, §5.4]

5.2.2 Variable Time Stepping

We built on the second-order projection method in the last section and introduce the variable density variable time stepping projection method. This step is important to be able to slow down the simulation proportional to the CFL when there is so much movement in the system then speed up when it calms down.

We assume that the density field ρ^{n+1} is already calculated with one of the methods in chapter 1 (e.g. SSPRK(3,3)). We redefine how BDF2 is defined and introduce $\alpha = \frac{\Delta t_2}{\Delta t_1}$ the ratio of the previous two time steps as follows: $\boldsymbol{u}^{n+1}$ happens at time $t + \Delta t_1$, $\boldsymbol{u}^n$ at t, and $\boldsymbol{u}^{n-1}$ at $t - \Delta t_2$. The new BDF2 is:

$$\frac{\partial y}{\partial t} \approx \frac{((1+\alpha)^2 - 1)y^{n+1} - (1+\alpha)^2 y^n + y^{n-1}}{\alpha(1+\alpha)\Delta t}.$$

To avoid clutter, set $a_1 := (1+\alpha)^2 - 1, a_2 := (1+\alpha)^2, a_3 := \alpha(1+\alpha)$. Also, to get a second order velocity extrapolation u^*, we use the variable time step central difference formula:

$$y^{n+1} = \left(1 + \frac{1}{\alpha}\right) y^n - \frac{1}{\alpha} y^{n-1} + \Delta t_1 \frac{\Delta t_1 + \Delta t_2}{2} \partial_{tt} y(x).$$

Then, the variable time projection algorithm becomes:

1. Prediction:

$$\rho^* = \rho^{n+1} + \frac{1}{2a_1} \mathrm{BDF}_{2\alpha}(\rho^{n+1}), \tag{5.6}$$

$$\text{where } \mathrm{BDF}_{2\alpha}(\phi^{n+1}) = a_1 \phi^{n+1} - a_2 \phi^n + \phi^{n-1}, \tag{5.7}$$

$$p^* = p^n + \frac{1}{a_1}\left(a_2 \delta\psi^n - \delta\psi^{n-1}\right), \tag{5.8}$$

$$\boldsymbol{u}^* = \left(1 + \frac{1}{\alpha}\right)\boldsymbol{u}^n - \frac{1}{\alpha}\boldsymbol{u}^{n-1}. \tag{5.9}$$

$$\frac{a_1\rho^*\boldsymbol{u}^{n+1}-a_2\rho^{n+1}\boldsymbol{u}^n+\rho^{n+1}\boldsymbol{u}^{n-1}}{a_3\Delta t_1}-\rho^{n+1}\boldsymbol{u}^*\cdot\nabla\boldsymbol{u}^{n+1}+\nabla p^\star+\frac{1}{2}\mathrm{div}\left(\rho^{k+1}\boldsymbol{u}^*\right)\boldsymbol{u}^{n+1}$$
$$-\mathrm{div}\left(\mu\varepsilon(\boldsymbol{u}^{n+1})\right)-\lambda\nabla\mathrm{div}\left(\boldsymbol{u}^{n+1}\right)=\rho^{n+1}\boldsymbol{f}^{n+1},\quad \boldsymbol{u}^{n+1}\big|_{\partial\Omega}=0. \quad (5.10)$$

2. Projection:

$$\Delta\delta\psi^{n+1}=\frac{a_1\rho_{\min}}{a_3\Delta t_1}\mathrm{div}\left(u^{k+1}\right),\ \text{where}\ \partial_n\delta\psi^{n+1}=0, \quad (5.11)$$

$$\delta q^{n+1}=-\mathrm{div}\left(u^{n+1}\right). \quad (5.12)$$

3. Pressure correction: $p^{n+1}=\psi^{n+1}-\mu q^{n+1}$.

This method is verified in section 5.5.1 and is used to simulate the dam breaking problem in section 5.5.2.2.

5.2.3 Variable Density Artificial Compressibility

We also build upon the third order constant density artificial compressibility (4.40)-(4.42). Here, the mass conservation is also calculated using deferred correction method. The algorithm starts by introducing a first order density solution ρ_0^n in (5.13) then correct it to get a second order density solution $\rho_{O(2)}^n$ in (5.14). The last step uses third order solution $\rho_{O(3)}^n$ in (5.15).

$$
\text{for } n \geq 0 \begin{cases}
\mathrm{adv}_0^{n+1} = -\boldsymbol{u}_0^n \cdot \nabla \rho_0^n, \\
\dfrac{\rho_0^{n+1} - \rho_0^n}{\Delta t} = \mathrm{adv}_0^{n+1}, \\
d\rho_0^{n+1} = \dfrac{\rho_0^{n+1} - \rho_0^n}{\Delta t}, \\
\mathbf{nl}_0^{n+1} = \rho_0^{n+1} B \boldsymbol{u}_0^n, \\
\begin{cases}
\rho_0^{n+1} \dfrac{\boldsymbol{u}_0^{n+1} - \boldsymbol{u}_0^n}{\Delta t} + A\boldsymbol{u}_0^{n+1} - \lambda \nabla \mathrm{div}\, \boldsymbol{u}_0^{n+1} + \nabla p_0^n \\
\qquad\qquad = \rho_0^{n+1} f^{n+1} - \mathbf{nl}_0^{n+1}, \\
p_0^{n+1} = p_0^n - \lambda \mathrm{div}\, \boldsymbol{u}_0^{n+1},
\end{cases} \\
d\boldsymbol{u}_0^{n+1} = \dfrac{\boldsymbol{u}_0^{n+1} - \boldsymbol{u}_0^n}{\Delta t}, \quad dp_0^{n+1} = \dfrac{p_0^{n+1} - p_0^n}{\Delta t},
\end{cases}
\tag{5.13}
$$

$$
\text{for } n \geq 1 \begin{cases}
d^2\rho_0^{n+1} = \dfrac{d\rho_0^{n+1} - d\rho_0^n}{\Delta t}, \\
\mathrm{adv}_1^n = -(\boldsymbol{u}_0^n + \Delta t \boldsymbol{u}_1^{n-1}) \cdot \nabla(\rho_0^n + \Delta t \rho_1^{n-1}), \\
\begin{cases}
\dfrac{\rho_1^n - \rho_1^{n-1}}{\Delta t} = -\dfrac{1}{2} d^2\rho_0^{n+1} + \dfrac{\mathrm{adv}_1^n - \mathrm{adv}_0^n}{\Delta t}\ , \\
d\rho_1^n = \dfrac{\rho_1^n - \rho_1^{n-1}}{\Delta t}, \quad \rho_{O(2)}^n = \rho_0^n + \Delta t \rho_1^n,
\end{cases} \\
d^2\boldsymbol{u}_0^{n+1} = \dfrac{d\boldsymbol{u}_0^{n+1} - d\boldsymbol{u}_0^n}{\Delta t}, \\
\mathbf{nl}_1^n = \rho_{O(2)}^n B(\boldsymbol{u}_0^n + \Delta t \boldsymbol{u}_1^{n-1}), \\
\begin{cases}
\rho_{O(2)}^n \dfrac{\boldsymbol{u}_1^n - \boldsymbol{u}_1^{n-1}}{\Delta t} + A\boldsymbol{u}_1^n - \lambda \nabla \mathrm{div}\, \boldsymbol{u}_1^n + \nabla(p_1^{n-1} + dp_0^n) \\
\qquad\qquad = -\dfrac{1}{2}\rho_{O(2)}^n d^2\boldsymbol{u}_0^{n+1} - \dfrac{\mathbf{nl}_1^n - \mathbf{nl}_0^n}{\Delta t}, \\
p_1^n = p_1^{n-1} + dp_0^n - \lambda \mathrm{div}\, \boldsymbol{u}_1^n,
\end{cases} \\
d\boldsymbol{u}_1^n = \dfrac{\boldsymbol{u}_1^n - \boldsymbol{u}_1^{n-1}}{\Delta t}, \quad dp_1^n = \dfrac{p_1^n - p_1^{n-1}}{\Delta t},
\end{cases}
\tag{5.14}
$$

$$
\text{for } n \geq 2 \begin{cases}
d^2\rho_1^n = \dfrac{d\rho_1^n - d\rho_1^{n-1}}{\Delta t}, \quad d^3\rho_0^{n+1} = \dfrac{d^2\rho_0^{n+1} - d^2\rho_0^n}{\Delta t}, \\
\mathrm{adv}_2^{n-1} = -(\boldsymbol{u}_0^{n-1} + \Delta t \boldsymbol{u}_1^{n-1} + \Delta t^2 \boldsymbol{u}_2^{n-2}) \cdot \\
\qquad\qquad \nabla(\rho_0^{n-1} + \Delta t \rho_1^{n-1} + \Delta t^2 \rho_2^{n-2}), \\
\begin{cases} \dfrac{\rho_2^{n-1} - \rho_2^{n-2}}{\Delta t} = -\dfrac{1}{2} d^2\rho_1^n - \dfrac{1}{6} d^3\rho_1^{n-1} + \dfrac{\mathrm{adv}_2^{n-1} - \mathrm{adv}_1^{n-1}}{\Delta t^2}, \end{cases} \\
d^2\boldsymbol{u}_1^n = \dfrac{d\boldsymbol{u}_1^n - d\boldsymbol{u}_1^{n-1}}{\Delta t}, \quad d^3\boldsymbol{u}_0^{n+1} = \dfrac{d^2\boldsymbol{u}_0^{n+1} - d^2\boldsymbol{u}_0^n}{\Delta t}, \\
\rho_{O(3)}^{n-1} = \rho_0^{n-1} + \Delta t \rho_1^{n-1} + \Delta t^2 \rho_2^{n-1}, \\
\mathbf{nl}_2^{n-1} = \rho_{O(3)}^{n-1} B(\boldsymbol{u}_0^{n-1} + \Delta t \boldsymbol{u}_1^{n-1} + \Delta t^2 \boldsymbol{u}_2^{n-2}), \\
\begin{cases} \rho_{O(3)}^{n-1} \dfrac{\boldsymbol{u}_2^{n-1} - \boldsymbol{u}_2^{n-2}}{\Delta t} + A\boldsymbol{u}_2^{n-1} - \lambda \nabla \mathrm{div}\, \boldsymbol{u}_2^{n-1} + \nabla(p_2^{n-2} + dp^{n-1}) \\ \qquad\qquad = \rho_{O(3)}^{n-1}(-\dfrac{1}{2} d^2\boldsymbol{u}_1^n + \dfrac{1}{6} d^3\boldsymbol{u}_0^{n+1}) - \dfrac{\mathbf{nl}_2^{n-1} - \mathbf{nl}_1^{n-1}}{\Delta t^2}, \\ p_2^{n-1} = p_2^{n-2} + dp_1^{n-1} - \lambda \mathrm{div}\, \boldsymbol{u}_2^{n-1}, \end{cases} \\
u^{n-1} = \boldsymbol{u}_0^{n-1} + \Delta t \boldsymbol{u}_1^{n-1} + \Delta t^2 \boldsymbol{u}_2^{n-1}, \\
p^{n-1} = p_0^{n-1} + \Delta t p_1^{n-1} + \Delta t^2 p_2^{n-1}, \quad \rho^{n-1} = \rho_{O(3)}^{n-1}.
\end{cases}
\tag{5.15}
$$

All the operators used in the above algorithm are simple and work well as explained in section 4.2.4. We have a mass matrix inversion for the density and pressure updates. The only difference is that to solve the momentum equation, we need to invert an operator like $\rho\boldsymbol{v} - \Delta t(\nabla \mathrm{div}\, \boldsymbol{v} + \Delta \boldsymbol{v}) = \boldsymbol{b}$. This means that the $\rho\boldsymbol{v}$ part of the matrix has to be reconstructed every time step three times for a third order method because of the density changes.

5.3 Linear Systems

The algorithms described in the previous section are almost exactly the same as the ones in the previous chapter. The difference is that the linear system associated with the momentum equation has the density ρ implicitly. Thus, the matrix has to be reconstructed at every time step and, with that, the preconditioner. The matrix used by the variable density artificial compressibility method in the last section is symmetric and can be solved using conjugate gradient (CG) method but the variable density projection method is not symmetric and, therefore, requiring an iterative solver like biconjugate gradient stabilized (BiCGSTAB) method (c.f. van der Vorst [82]) or generalized minimal residual (GMRES) method (c.f Saad and Schultz [67]).

5.4 Simplified Variable Density Artificial Compressibility Method

In this section, we will prove the stability of the continuous and discrete forms of a simplified Navier-Stokes equations (5.16)-(5.18). This simplified version removes the space derivatives and keeps the essential features of full Navier-Stokes equations that is relevant to the time discretization. We will first prove the stability of the continuous simplified Navier-Stokes equations. Then prove the simplified implicit discrete Navier-Stokes equation.

5.4.1 Continuous Equations Stability Analysis

Lemma 5.4.1. Let $\boldsymbol{u} \in H^1((0,T);L^2(\Omega))$, $p \in H^1((0,T);L^2(\Omega))$ and $\rho \in H^1((0,T);L^1(\Omega))$ be the solution to:

$$\partial_t \rho + \beta \rho \boldsymbol{u} = 0, \tag{5.16}$$

$$\partial_t(\rho \boldsymbol{u}) + \mu \boldsymbol{u} + p = -\frac{\beta}{2}\rho \boldsymbol{u}^2, \tag{5.17}$$

$$\partial_t p - \lambda \boldsymbol{u} = 0. \tag{5.18}$$

Then (5.16)-(5.18) is stable if $\lambda > 0, \mu > 0$ and $\rho > 0, \forall \rho$. i.e.

$$\left(\rho \frac{\boldsymbol{u}^2}{2}\right)(T) + \frac{p^2}{2\lambda}(T) \leq \left(\rho \frac{\boldsymbol{u}^2}{2}\right)(0) + \frac{p^2}{2\lambda}(0),$$

Proof. Start with (5.17) and multiplying it by $\boldsymbol{u}$:

$$\begin{aligned}
\boldsymbol{u}\partial_t(\rho \boldsymbol{u}) + \mu \boldsymbol{u}^2 + p\boldsymbol{u} &= \frac{\beta}{2}\rho \boldsymbol{u}^2 \boldsymbol{u}, \\
\boldsymbol{u}\rho\partial_t \boldsymbol{u} + \boldsymbol{u}^2\partial_t\rho + \mu \boldsymbol{u}^2 + p\boldsymbol{u} &= \frac{\beta}{2}\rho \boldsymbol{u}^2 \boldsymbol{u}, \\
\rho\partial_t \frac{\boldsymbol{u}^2}{2} + \boldsymbol{u}^2\partial_t\rho + \mu \boldsymbol{u}^2 + p\boldsymbol{u} &= \frac{\beta}{2}\rho \boldsymbol{u}^2 \boldsymbol{u}, \\
\left[\rho\partial_t \frac{\boldsymbol{u}^2}{2} + \frac{\boldsymbol{u}^2}{2}\partial_t\rho\right] + \frac{\boldsymbol{u}^2}{2}\partial_t\rho + \mu \boldsymbol{u}^2 + p\boldsymbol{u} &= \frac{\beta}{2}\rho \boldsymbol{u}^2 \boldsymbol{u}, \\
\partial_t\left(\rho \frac{\boldsymbol{u}^2}{2}\right) + \frac{\boldsymbol{u}^2}{2}\partial_t\rho + \mu \boldsymbol{u}^2 + p\boldsymbol{u} &= \frac{\beta}{2}\rho \boldsymbol{u}^2 \boldsymbol{u}.
\end{aligned}$$

Then using (5.16):

$$\begin{aligned}
\partial_t\left(\rho \frac{\boldsymbol{u}^2}{2}\right) - \frac{\boldsymbol{u}^2}{2}\beta\rho \boldsymbol{u} + \mu \boldsymbol{u}^2 + p\boldsymbol{u} &= -\frac{\beta}{2}\rho \boldsymbol{u}^2 \boldsymbol{u}, \\
\partial_t\left(\rho \frac{\boldsymbol{u}^2}{2}\right) + \mu \boldsymbol{u}^2 + p\boldsymbol{u} &= 0.
\end{aligned}$$

Finally, using (5.18):

$$\begin{aligned}
\partial_t\left(\rho \frac{\boldsymbol{u}^2}{2}\right) + \mu \boldsymbol{u}^2 + \frac{1}{\lambda}p\partial_t p &= 0, \\
\partial_t\left(\rho \frac{\boldsymbol{u}^2}{2}\right) + \mu \boldsymbol{u}^2 + \frac{1}{\lambda}\partial_t \frac{p^2}{2} &= 0,
\end{aligned}$$

Integrating over $t = [0, T]$:

$$\int_0^T \left[\partial_t \left(\rho \frac{\boldsymbol{u}^2}{2}\right) + \mu \boldsymbol{u}^2 + \frac{1}{\lambda}\partial_t \frac{p^2}{2}\right] dt = 0,$$
$$\left(\rho \frac{\boldsymbol{u}^2}{2}\right)(T) + \frac{p^2}{2\lambda}(T) + \int_0^T \mu \boldsymbol{u}^2 dt = \left(\rho \frac{\boldsymbol{u}^2}{2}\right)(0) + \frac{p^2}{2\lambda}(0),$$

Or more specifically:

$$\left(\rho \frac{\boldsymbol{u}^2}{2}\right)(T) + \frac{p^2}{2\lambda}(T) \leq \left(\rho \frac{\boldsymbol{u}^2}{2}\right)(0) + \frac{p^2}{2\lambda}(0),$$

and this completes the proof □

5.4.2 Discrete Implicit Equations Stability Analysis

Lemma 5.4.2. Let $(\boldsymbol{u}^n, p^n, \rho^n)_{n>0}$ be a sequence of solutions to:

$$\frac{\rho^{n+1} - \rho^n}{\Delta t} + \beta \rho^{n+1} \boldsymbol{u}^{n+1} = 0, \tag{5.19}$$
$$\rho^{n+1} \frac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^n}{\Delta t} + (\mu + \lambda)\boldsymbol{u}^{n+1} + p^n = -\frac{\beta}{2}\rho^{n+1}(\boldsymbol{u}^n)^2, \tag{5.20}$$
$$\frac{p^{n+1} - p^n}{\lambda} - \boldsymbol{u}^{n+1} = 0. \tag{5.21}$$

with $(\boldsymbol{u}^0, p^0, \rho^0)$ as initial conditions Then (5.19)-(5.21) is stable if $\lambda > 0, \mu > 0, \Delta t > 0$ and $\rho^n > 0, \forall n$. i.e.

$$\lambda \rho^{k+1}(\boldsymbol{u}^{k+1})^2 + \Delta t (p^{k+1})^2 \leq \lambda \rho^0 (\boldsymbol{u}^0)^2 + \Delta t (p^0)^2.$$

Proof. Start with (5.20) and multiplying it by $\boldsymbol{u}^{n+1}$:

$$\rho^{n+1} \frac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^n}{\Delta t} \boldsymbol{u}^{n+1} + (\mu + \lambda)\left(\boldsymbol{u}^{n+1}\right)^2 + p^n \boldsymbol{u}^{n+1} = -\frac{\beta}{2}\rho^{n+1}(\boldsymbol{u}^n)^2 \boldsymbol{u}^{n+1}.$$

Then using the identity $(a-b)a = \frac{1}{2}[a^2 + (a-b)^2 - b^2]$:

$$\frac{1}{2\Delta t}\rho^{n+1}\left[(\boldsymbol{u}^{n+1})^2 + (\boldsymbol{u}^{n+1} - \boldsymbol{u}^n)^2 - (\boldsymbol{u}^n)^2\right] + (\mu+\lambda)\left(\boldsymbol{u}^{n+1}\right)^2 + p^n\boldsymbol{u}^{n+1} = -\frac{\beta}{2}\rho^{n+1}(\boldsymbol{u}^n)^2\boldsymbol{u}^{n+1},$$

then we remove and add $\frac{\rho^n(\boldsymbol{u}^n)^2}{2\Delta t}$ and regroup:

$$\frac{\rho^{n+1}(\boldsymbol{u}^{n+1})^2 - \rho^n(\boldsymbol{u}^n)^2}{2\Delta t} + \frac{\rho^{n+1}}{\Delta t}(\boldsymbol{u}^{n+1} - \boldsymbol{u}^n)^2 + \frac{\rho^n - \rho^{n+1}}{\Delta t}\frac{(\boldsymbol{u}^n)^2}{2} + (\mu+\lambda)\left(\boldsymbol{u}^{n+1}\right)^2 + p^n\boldsymbol{u}^{n+1} = -\frac{\beta}{2}\rho^{n+1}(\boldsymbol{u}^n)^2\boldsymbol{u}^{n+1},$$

applying (5.19):

$$\frac{\rho^{n+1}(\boldsymbol{u}^{n+1})^2 - \rho^n(\boldsymbol{u}^n)^2}{2\Delta t} + \frac{\rho^{n+1}}{\Delta t}(\boldsymbol{u}^{n+1} - \boldsymbol{u}^n)^2 - \beta\rho^{n+1}\boldsymbol{u}^{n+1}\frac{(\boldsymbol{u}^n)^2}{2} + (\mu+\lambda)\left(\boldsymbol{u}^{n+1}\right)^2 + p^n\boldsymbol{u}^{n+1} = -\frac{\beta}{2}\rho^{n+1}(\boldsymbol{u}^n)^2\boldsymbol{u}^{n+1},$$

$$\frac{\rho^{n+1}(\boldsymbol{u}^{n+1})^2 - \rho^n(\boldsymbol{u}^n)^2}{2\Delta t} + \frac{\rho^{n+1}}{\Delta t}(\boldsymbol{u}^{n+1} - \boldsymbol{u}^n)^2 + (\mu+\lambda)\left(\boldsymbol{u}^{n+1}\right)^2 + p^n\boldsymbol{u}^{n+1} = 0.$$

Take (5.21) multiplied by p^n and use the identity $(a-b)b = \frac{1}{2}[a^2 - (a-b)^2 - b^2]$:

$$\frac{(p^{n+1})^2 - (p^{n+1} - p^n)^2 - (p^n)^2}{2\lambda} - p^n\boldsymbol{u}^{n+1} = 0. \tag{5.22}$$

Applying (5.22) to the previous equation:

$$\frac{\rho^{n+1}(\boldsymbol{u}^{n+1})^2 - \rho^n(\boldsymbol{u}^n)^2}{2\Delta t} + \frac{\rho^{n+1}}{\Delta t}(\boldsymbol{u}^{n+1} - \boldsymbol{u}^n)^2 + (\mu+\lambda)\left(\boldsymbol{u}^{n+1}\right)^2 + \frac{(p^{n+1})^2 - (p^{n+1} - p^n)^2 - (p^n)^2}{2\lambda} = 0.$$

But the term $-(p^{n+1}-p^n)^2$ will need to be dealt with because it is negative. Take (5.21) and square both sides:

$$\left(\frac{p^{n+1}-p^n}{\lambda}\right)^2 = (\boldsymbol{u}^{n+1})^2,$$

Putting it back:

$$\frac{\rho^{n+1}(\boldsymbol{u}^{n+1})^2-\rho^n(\boldsymbol{u}^n)^2}{2\Delta t}+\frac{\rho^{n+1}}{\Delta t}(\boldsymbol{u}^{n+1}-\boldsymbol{u}^n)^2+\mu(\boldsymbol{u}^{n+1})^2+ \\ \frac{(p^{n+1}-p^n)^2}{\lambda}+\frac{(p^{n+1})^2-(p^{n+1}-p^n)^2-(p^n)^2}{2\lambda}=0,$$

$$\frac{\rho^{n+1}(\boldsymbol{u}^{n+1})^2-\rho^n(\boldsymbol{u}^n)^2}{2\Delta t}+\frac{\rho^{n+1}}{\Delta t}(\boldsymbol{u}^{n+1}-\boldsymbol{u}^n)^2+\mu(\boldsymbol{u}^{n+1})^2+ \\ \frac{(p^{n+1}-p^n)^2}{2\lambda}+\frac{(p^{n+1})^2-(p^n)^2}{2\lambda}=0,$$

Finally, summing over $n=0\dots k$:

$$\lambda\left(\rho^{k+1}(\boldsymbol{u}^{k+1})^2-\rho^0(\boldsymbol{u}^0)^2\right)+\Delta t\left((p^{k+1})^2-(p^0)^2\right) \\ +\sum_{n=0}^{k}2\lambda\rho^{n+1}(\boldsymbol{u}^{n+1}-\boldsymbol{u}^n)^2+2\lambda\Delta t\mu(\boldsymbol{u}^{n+1})^2+\Delta t(p^{n+1}-p^n)^2=0,$$

Or more generally:

$$\lambda\rho^{k+1}(\boldsymbol{u}^{k+1})^2+\Delta t(p^{k+1})^2\leq\lambda\rho^0(\boldsymbol{u}^0)^2+\Delta t(p^0)^2,$$

and this completes the proof $\square$

5.5 Numerical Results

As usual, we test the schemes discussed in the previous sections numerically and present them here.

5.5.1 Validation

Here, we will only present the validation of the variable time stepping projection method (5.10) with density $\rho(\boldsymbol{x}, t) = 1$ (this is considered a constant density test. However, we are using the variable time stepping scheme presented in this chapter). Using $\Omega = (0, 1)^d$ domain with a uniform mesh and cell-wise $[\mathbb{Q}_2]^d$ continuous finite elements, we introduce the following simple linear polynomial manufactured solution for the momentum equation:

$$\boldsymbol{u}(\boldsymbol{x}, t) = (1 + t) \begin{pmatrix} x + y \\ x - y \end{pmatrix}, \qquad p(\boldsymbol{x}, t) = (1 + t)xy, \text{ when } d = 2, \tag{5.23}$$

$$\boldsymbol{u}(\boldsymbol{x}, t) = (1 + t) \begin{pmatrix} 1 + z \\ 1 + x \\ 1 + y \end{pmatrix}, \qquad p(\boldsymbol{x}, t) = (1 + t)xyz \text{ when } d = 3. \tag{5.24}$$

We solve the equation (5.10) with $\mu = 1$ running until final time $T = 1$. The projection step is disabled, which means that the exact pressure is interpolated every time step. The boundary condition $\boldsymbol{u}|_{\partial\Omega} = \boldsymbol{u}(\boldsymbol{x}, t)|_{\partial\Omega}$ is enforced. The time step is changed to roughly achieve a CFL of 0.25. As expected, table 5.1 shows that the error is machine epsilon which means that the algorithm reproduces the conforming manufactured solutions exactly.

	cells	$\boldsymbol{u}_{\text{dofs}}$	Δt	$\|e_{\boldsymbol{u}}\|_{L2}$	$\|e_{\boldsymbol{u}}\|_{H1}$	$\text{CFL}_{\max}$
	16	162	8E-03	3E-13	1E-12	0.2621
2D	64	578	4E-03	1E-15	2E-14	0.2606
	256	2178	2E-03	1E-14	8E-14	0.2607
	8	375	3E-02	9E-16	1E-14	0.2601
3D	64	2187	1E-02	5E-15	3E-14	0.2614
	512	14739	7E-03	7E-15	7E-14	0.2613

Table 5.1: Error values for running conforming manufactured solutions in a unit cube. We get the expected value of machine epsilon.

5.5.1.1 Convergence Rate Analysis

We tested the first-order variable density artificial compressibility algorithm (5.13) for the purpose of checking convergence rates. The domain is 2D $\Omega = (0,1) \times (0,1)$ with the following manufactured solutions:

- Manufactured Solutions used in the 2D case:

$$\begin{aligned} \rho &= 2 + \sin^2(x + y + t), \\ \boldsymbol{u} &= (\cos(x) + \cos(y + t), \sin(x) + \sin(y + t))^\top, \\ p &= \cos(x + y + t) \end{aligned}$$

- Manufactured Solutions used in the 3D case:

$$\begin{aligned} \rho &= 2 + \sin^2(x + y + z + t) \\ \boldsymbol{u} &= (\sin(\pi x)\cos(\pi y)\cos(\pi z)\cos(t), \\ &\quad \sin(\pi y)\cos(\pi x)\cos(\pi z)\cos(t), \\ &\quad -2\sin(\pi z)\cos(\pi x)\cos(\pi y)\cos(t))^\top, \end{aligned}$$

$$p \;=\; \cos(x+y+z+t).$$

The mesh is uniform with Taylor-Hood finite elements; i.e. $\mathbb{Q}_2$ approximation for both the density and velocity and $\mathbb{Q}_1$ for the pressure. We add first order artificial viscosity to the transport equation as explained in section 3.3.1 with $C_m = 0.125$. The source term is calculated such that we get the exact solutions above. The simulation is run to $t = 0.5$ then the errors are calculated as shown in table 5.2. The errors are calculated with higher order Gaussian quadrature.

cells	Δt	$\|e_\rho\|_{L^2}$	rate	$\|e_{\boldsymbol{u}}\|_{L^2}$	rate	$\|e_{\boldsymbol{u}}\|_{H^1}$	rate
256	0.01	1.81E-03	-	2.90E-04	-	2.04E-03	-
1024	0.005	8.81E-04	1.04	1.34E-04	1.12	9.75E-04	1.06
4096	0.0025	4.34E-04	1.02	6.47E-05	1.04	4.78E-04	1.03

cells	Δt	$\|e_P\|_{L^2}$	rate	$\|e_P\|_{H^1}$	rate
256	0.01	6.80E-03	-	4.63E-02	-
1024	0.005	3.79E-03	0.84	2.42E-02	0.94
4096	0.0025	1.91E-03	0.99	1.23E-02	0.97

Table 5.2: Convergence rates with respect to time for the first-order variable density artificial compressibility Navier-Stokes equations in 2D (essentially solving only (5.13)). As one can see, the convergence rate is 1 across all solution variables in their associated norms. CFL=0.32.

The convergence rate of interest here is with respect to time by choosing $h \ll \Delta t$. As one can see in table 5.2, the expected asymptotic convergence rate of 1 is reached across all variables and their associated norms.

5.5.2 Realistic Models

In this section, we will study the applications of variable density projection scheme on a more realistic model; the Rayleigh-Taylor instability test. We compare our results with the work of Guermond et al. [42]. Specifically in the early times before turbulent behavior.

5.5.2.1 Rayleigh-Taylor Instability

We now apply the method to a more realistic problem. We use the Rayleigh-Taylor instability test that Tryggvason [80] used. Two fluids are initially at rest in the 2D domain $(-d/2, d/2) \times (-2d, 2d)$ and the heavier fluid is on top. The transition of the phase-field variable ρ is as follows:

$$\rho(x, y, t = 0) = \frac{\rho_{\max} + \rho_{\min}}{2} + \frac{\rho_{\max} - \rho_{\min}}{2} \tanh\left(\frac{y + \mu(x)}{\alpha d}\right), \tag{5.25}$$

where $\alpha \approx 0.04$ and the initial interface is slightly perturbed as follows:

$$\mu(x) = 0.1 \cos(2\pi x/d).$$

The time is also scaled using the Atwood number in Tryggvason as $t_{\text{Tryg}} = t\sqrt{A_t}$

$$A_t = \frac{\rho_0^{\max} - \rho_0^{\min}}{\rho_0^{\max} + \rho_0^{\min}},$$

where $\rho_0^{\max} := \max_{\boldsymbol{x} \in \Omega} \rho_0(\boldsymbol{x})$ and $\rho_0^{\min} := \min_{\boldsymbol{x} \in \Omega} \rho_0(\boldsymbol{x})$. As the system progresses at $t > 0$, the heavy fluid will fall into the lighter fluid as a result of having the momentum equation gravity source term equal $\rho \boldsymbol{g}$

We non-dimensionalize the equations as follows. We divide by: $\rho_0^{\min}$ for the density ρ, d for length, and $d^{1/2}/|\boldsymbol{g}|^{1/2}$ for time. Consequently, $d^{1/2}|\boldsymbol{g}|^{1/2}$ is the

velocity reference and the Reynolds number is $Re = \rho_0^{\min} d^{1/2} |\boldsymbol{g}|^{1/2} d/\mu$. We will restrict ourselves to the domain $(0, d/2) \times (-2d, 2d)$ because we assume that the symmetry of the initial setup continues as time progresses. The top and bottom parts have no-slip boundary conditions and the left and right sides have $\boldsymbol{u} \cdot \boldsymbol{n} = 0$, $(I - \boldsymbol{n} \otimes \boldsymbol{n})\nu \nabla \boldsymbol{u} = 0$ boundary conditions (known as symmetry or free boundary conditions).

Remark. Note that we must integrate the pressure term by parts in the weak form for p to be in L^2. In this experiment, we tested both intergrating by parts and leaving the pressure term as is. This leads to different boundary conditions for each case: $(I - \boldsymbol{n} \otimes \boldsymbol{n})(\nu \nabla \boldsymbol{u} - Ip) = 0$, and $(I - \boldsymbol{n} \otimes \boldsymbol{n})\nu \nabla \boldsymbol{u} = 0$ respectively. In this experiment, both were numerically stable and gave almost exactly the same results when compared to previous papers. By not integrating by parts, p will be in H^1 and we have to answer the question: Is the discrete LBB condition (4.6) satisfied for the space pair H^1, H^1? In this experiment specifically, it seems to be stable but we cannot generalize to all possible cases without a rigours .

As hyperbolic equations need stabilization, we do so with the nonlinear entropy viscosity Guermond et al. [42] using the entropy function $E(x) = -\log|\rho(1-\rho) + 10^{-14}|$. In figure 5.1, the evolution of the density field of ratio 3 at times 1, 1.5, 2, and 2.5 in Tryggvason time scale $t_{\text{Tryg}} = t\sqrt{A_t}$ with $Re = 1000$. The same times are shown in figure 5.2 with density ratio of 100. The are 8484 $\mathbb{Q}_2$ degrees of freedom for ρ with uniform mesh size of 2048 cells. The time stepping is variable and maintains a maximum CFL of 0.4.

Now, we want to conduct a more challenging test. Specifically, we will test with density ratio 100 to check the robustness of the scheme (see, for example, Sussman et al. [74]). As figure 5.2 shows, the simulation holds nicely. Also, when figure 5.1 is

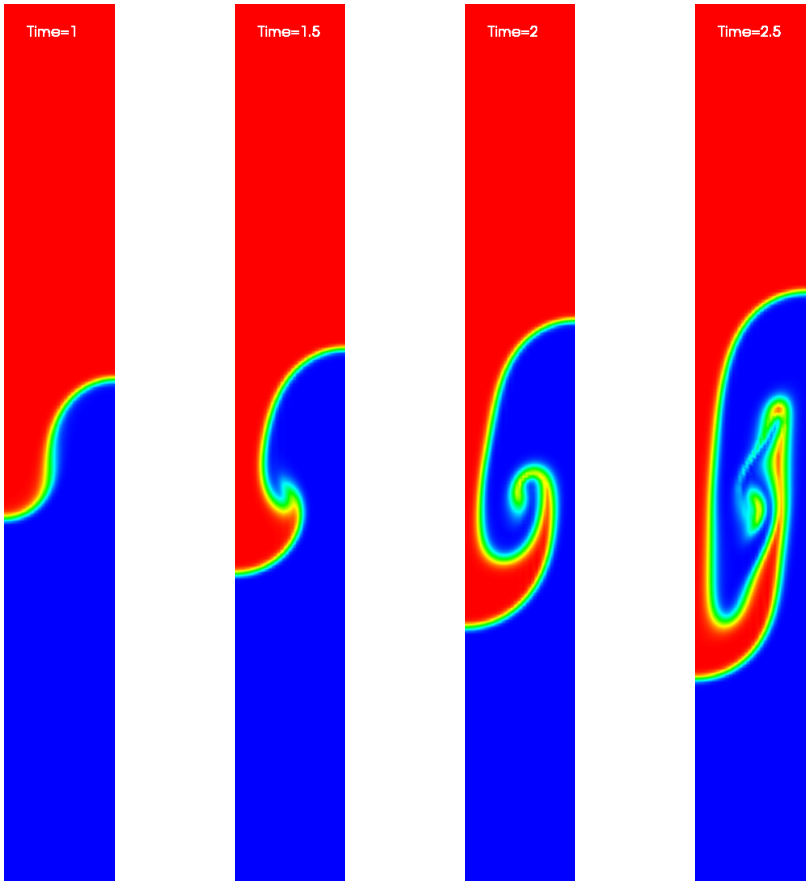


Figure 5.1: The Rayleigh-Taylor instability with density ratio of 3.

visually compared with the results in Guermond et al. [43], the are almost identical.

5.5.2.2 Dam Breaking Simulation

The Dam Breaking problem is setup in domain $\Omega = [0,4] \times [0,3] \times [-1,1]$ with a uniform mesh and 393,216 $\mathbb{Q}_2/\mathbb{Q}_1$ elements. The symmetry plane is placed at $z = 0$ with boundary conditions $\nabla \boldsymbol{u} \cdot \boldsymbol{n} = 0$ and $\boldsymbol{u} = 0$ otherwise. We solve using (5.6)-(5.12). The Reynolds number is 1000 with time stepping variable to maintain a CFL of 0.4. The initial density $\rho(\boldsymbol{x}, t)$ used is (5.25) with $\rho_{\min} = 1$ and $\rho_{\max} = 10$. The initial velocity is 0 and the source term is gravity $\rho \mathbf{g}$. The compression coefficient

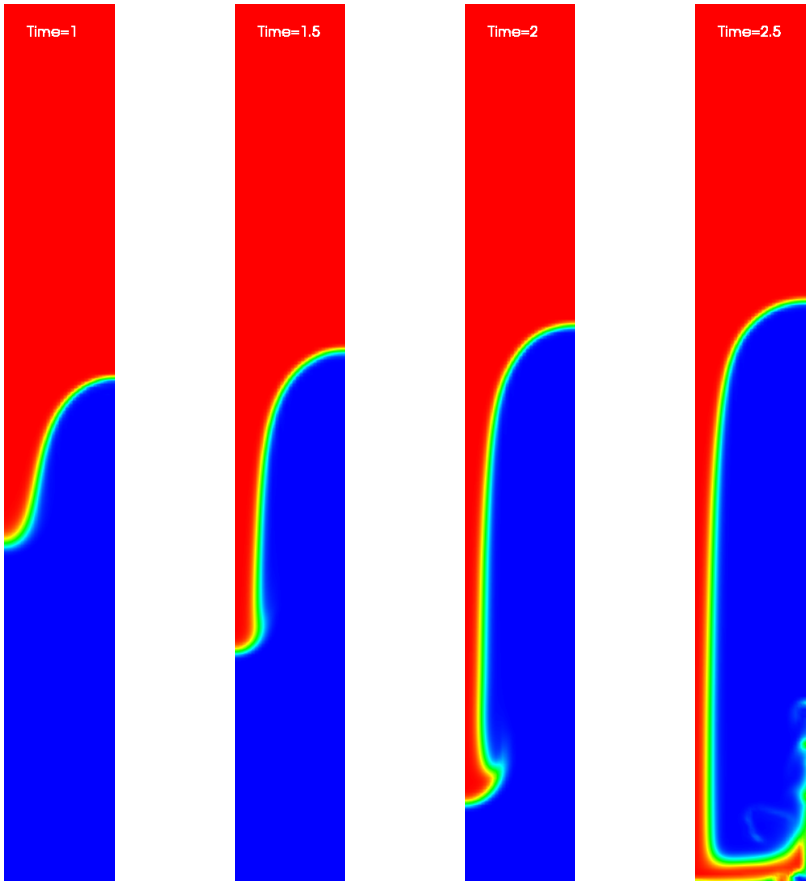


Figure 5.2: The Rayleigh-Taylor instability with density ratio of 100.

$C_k = 1$, Entropy-Viscosity $C_e = 0.2$, and linear viscosity $C_m = 0.125$. The total number of degrees of freedom is 13,309,189.

Figure 5.3 shows the two snapshots of the simulation at times 0.2 and 1.46. The visible band are densities $\rho \in [2, 9]$ which is essentially the transitional part of the level set. You can see how the compression maintained the thickness of the level set.

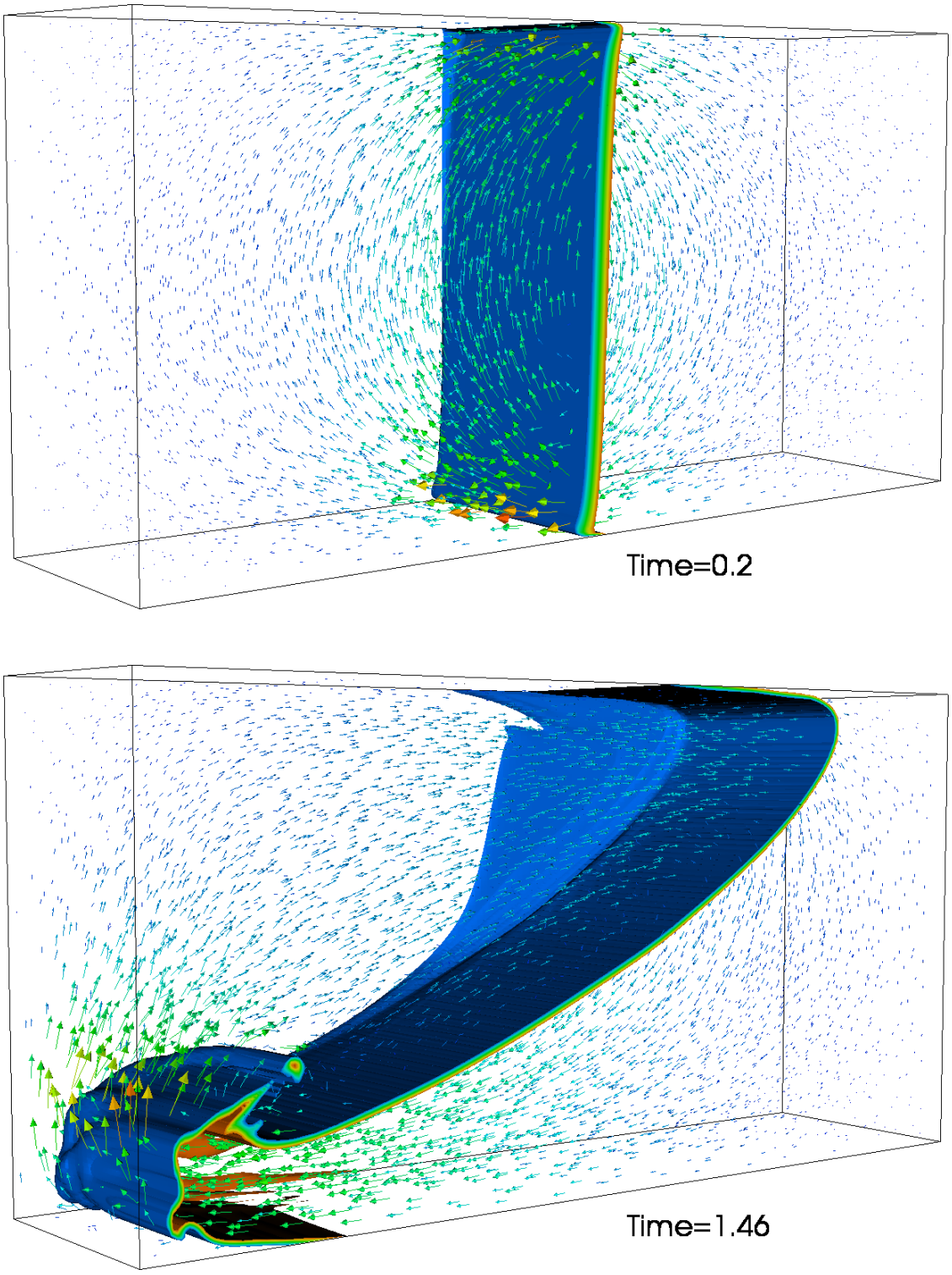


Figure 5.3: The dam breaking simulation with density ratio of 10. Since we use level set to represent the interface, we make the band of densities between $\rho \in [2, 9]$ visible to showcase how the compression of the level set maintains a narrow range of a few cells.

6. CONCLUSION

In this dissertation, we introduced the reason for starting this (potentially massively) parallel Navier-Stokes solver. Then went through explaining the ASPEN framework and the different parts of it. The planned next step for ASPEN is to prepare it to be published as open-source software and thus help other scientist build upon it.

After that, we explained the theory of transport equation and the different time stepping methods that can be followed. We shed light afterwards on stabilization techniques then talked about linear systems and finally validate with numerical results. Hyperbolic equation are tricky to solve and stabilize. Add to that the necessary next step of handling more that two and more phase flow.

Then we explained the constant density Navier-Stokes model and give an overview of its theoretical background. We discussed three methods to handle the saddle point problem that the Navier-Stokes equations pose. The we talked about turbulence and stabilization of the schemes. Finally we validated with numerical results. As the results of lid cavity driven flow show, we need to be careful how the mesh is distributed. If the mesh is not fine enough near the boundaries, the simulation breaks.

In the last chapter, we talked about the variable density Navier-Stokes equations. We offered the variable time stepping technique and a variable density artificial compressibility scheme. We provided stability results for a simplified Navier-Stokes equations in both continuous and implicit discrete setups. We finally provided numerical results at the end. As the dam breaking problem showed, using LES turbulence modeling handles the numerical stability issues that are inevitably faced when the Reynolds number is high.

ASPEN is still in its beginnings. There is a lot of work to be done to clean it up to add more functionality. For example, higher order variable density artificial compressibility is still underdevelopment. Also, artificial compressibility variable time stepping would need to be developed for long running simulations.

All visualizations in this dissertation are done with VisIt visualization software (c.f. Childs et al. [15]).

REFERENCES

[1] A. Almgren, J. Bell, P. Colella, L. Howell, and M. Welcome. A conservative adaptive projection method for the variable density incompressible navier--stokes equations. *Journal of Computational Physics*, 142(1):1 -- 46, 1998. ISSN 0021-9991. doi: http://dx.doi.org/10.1006/jcph.1998.5890. URL `http://www.sciencedirect.com/science/article/pii/S0021999198958909`.

[2] G. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, AFIPS '67 (Spring), pages 483--485, New York, NY, USA, 1967. ACM. doi: 10.1145/1465482.1465560. URL `http://doi.acm.org/10.1145/1465482.1465560`.

[3] U. Ascher, S. Ruuth, and B. Wetton. Implicit-explicit methods for time-dependent partial differential equations. *SIAM Journal on Numerical Analysis*, 32(3):797--823, 1995. doi: 10.1137/0732037. URL `http://dx.doi.org/10.1137/0732037`.

[4] S. Balay, S. Abhyankar, M. Adams, J. Brown, P. Brune, K. Buschelman, V. Eijkhout, W. Gropp, D. Kaushik, M. Knepley, L. C. McInnes, K. Rupp, B. Smith, and H. Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.5, Argonne National Laboratory, 2014. URL `http://www.mcs.anl.gov/petsc`.

[5] W. Bangerth, R. Hartmann, and G. Kanschat. deal.II -- a general purpose object oriented finite element library. *ACM Trans. Math. Softw.*, 33(4):24/1--24/27, 2007.

[6] W. Bangerth, T. Heister, L. Heltai, G. Kanschat, M. Kronbichler, M. Maier,

B. Turcksin, and T. D. Young. The deal.ii library, version 8.1. *arXiv preprint, http://arxiv.org/abs/1312.2266v4*, 2013.

[7] W. Bangerth, T. Heister, et al. *ASPECT: Advanced Solver for Problems in Earth's ConvecTion*, 2014. URL http://aspect.dealii.org/. http://aspect.dealii.org/.

[8] W. Bangerth, C. Burstedde, T. Heister, and M. Kronbichler. Algorithms and data structures for massively parallel generic adaptive finite element codes. *ACM Transactions on Mathematical Software*, 38(2):14:1--14:28, 2011.

[9] J. Bell and D. Marcus. A second-order projection method for variable-density flows. *Journal of Computational Physics*, 101(2):334 -- 348, 1992. ISSN 0021-9991. doi: http://dx.doi.org/10.1016/0021-9991(92)90011-M. URL http://www.sciencedirect.com/science/article/pii/002199919290011M.

[10] J. Benzi and M. Damodaran. Parallel three dimensional direct simulation monte carlo for simulating micro flows. In *Parallel Computational Fluid Dynamics 2007*, volume 67 of *Lecture Notes in Computational Science and Engineering*, pages 91--98. Springer Berlin Heidelberg, 2009. ISBN 978-3-540-92743-3. doi: 10.1007/978-3-540-92744-0_11. URL http://dx.doi.org/10.1007/978-3-540-92744-0_11.

[11] J. Bramble, J. Pasciak, and A. Vassilev. Analysis of the inexact uzawa algorithm for saddle point problems. *SIAM Journal on Numerical Analysis*, 34(3):1072--1092, 1997. doi: 10.1137/S0036142994273343. URL http://dx.doi.org/10.1137/S0036142994273343.

[12] C. Burstedde, L. Wilcox, and O. Ghattas. p4est: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees. *SIAM Journal on Scientific Computing*, 33(3):1103--1133, 2011. doi: 10.1137/100791634.

[13] M. Carpenter, D. Gottlieb, S. Abarbanel, and W. Don. The theoretical accuracy

of runge--kutta time discretizations for the initial boundary value problem: A study of the boundary error. *SIAM Journal on Scientific Computing*, 16(6): 1241--1252, 1995. doi: 10.1137/0916072. URL http://epubs.siam.org/doi/abs/10.1137/0916072.

[14] T. Chan and H. Van der Vorst. Approximate and incomplete factorizations. In D. Keyes, A. Sameh, and V. Venkatakrishnan, editors, *Parallel Numerical Algorithms*, volume 4 of *ICASE/LaRC Interdisciplinary Series in Science and Engineering*, pages 167--202. Springer Netherlands, 1997. ISBN 978-94-010-6277-0. doi: 10.1007/978-94-011-5412-3_6. URL http://dx.doi.org/10.1007/978-94-011-5412-3_6.

[15] H. Childs, E. Brugger, B. Whitlock, J. Meredith, S. Ahern, D. Pugmire, K. Biagas, M. Miller, C. Harrison, G. H. Weber, H. Krishnan, T. Fogal, A. Sanderson, C. Garth, E. W. Bethel, D. Camp, O. Rübel, M. Durant, J. M. Favre, and P. Navrátil. VisIt: An End-User Tool For Visualizing and Analyzing Very Large Data. In *High Performance Visualization--Enabling Extreme-Scale Scientific Insight*, pages 357--372. Oct 2012.

[16] A. J. Chorin. A numerical method for solving incompressible viscous flow problems. *Journal of Computational Physics*, 135(2):118 -- 125, 1997. ISSN 0021-9991. doi: http://dx.doi.org/10.1006/jcph.1997.5716. URL http://www.sciencedirect.com/science/article/pii/S0021999197957168.

[17] P. Ciarlet. *The Finite Element Method for Elliptic Problems*. Studies in Mathematics and its Applications. Elsevier Science, 1978. ISBN 9780080875255. URL http://books.google.com/books?id=TpHfoXnpKvAC.

[18] R. Courant, K. Friedrichs, and H. Lewy. On the partial difference equations of mathematical physics. *IBM J. Res. Dev.*, 11(2):215--234, March 1967. ISSN 0018-8646. doi: 10.1147/rd.112.0215. URL http://dx.doi.org/10.1147/rd.

112.0215.

[19] J. Crank and P. Nicolson. A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. *Advances in Computational Mathematics*, 6(1):207--226, 1996. ISSN 1019-7168. doi: 10.1007/BF02127704. URL http://dx.doi.org/10.1007/BF02127704.

[20] L. Dagum and R. Menon. Openmp: an industry standard api for shared-memory programming. *Computational Science Engineering, IEEE*, 5(1):46--55, Jan 1998. ISSN 1070-9924. doi: 10.1109/99.660313.

[21] R. DiPerna and P.-L. Lions. On the cauchy problem for boltzmann equations: Global existence and weak stability. *Annals of Mathematics*, 130(2):pp. 321--366, 1989. ISSN 0003486X. URL http://www.jstor.org/stable/1971423.

[22] R. DiPerna and P.-L. Lions. Ordinary differential equations, transport theory and sobolev spaces. *Inventiones mathematicae*, 98(3):511--547, 1989. ISSN 0020-9910. doi: 10.1007/BF01393835. URL http://dx.doi.org/10.1007/BF01393835.

[23] D. Drikakis and W. Rider. *High-Resolution Methods for Incompressible and Low-Speed Flows*. Computational Fluid and Solid Mechanics. Springer London, Limited, 2006. ISBN 9783540264545. URL http://books.google.com/books?id=1FVhPGEwJasC.

[24] A. Ern and J.-L. Guermond. *Theory and Practice of Finite Elements*. Number v. 159 in Applied Mathematical Sciences. Springer, 2004. ISBN 9780387205748. URL http://books.google.com/books?id=CCjm79FbJbcC.

[25] E. Fernández-Cara and F. Guillén. Some new existence results for the variable density Navier-Stokes equations. *Ann. Fac. Sci. Toulouse, Math. (6)*, 2(2): 185--204, 1993. ISSN 0240-2963. doi: 10.5802/afst.763.

[26] V. Girault and P. Raviart. *Finite element methods for Navier-Stokes equations: theory and algorithms*. Springer series in computational mathematics. Springer-

Verlag, 1986. ISBN 9783540157960. URL http://books.google.com/books?id=BVfvAAAAMAAJ.

[27] K. Goda. A multistep technique with implicit difference schemes for calculating two- or three-dimensional cavity flows. *Journal of Computational Physics*, 30(1):76 -- 95, 1979. ISSN 0021-9991. doi: http://dx.doi.org/10.1016/0021-9991(79)90088-3. URL http://www.sciencedirect.com/science/article/pii/0021999179900883.

[28] S. Gottlieb. On high order strong stability preserving runge--kutta and multi step time discretizations. *Journal of Scientific Computing*, 25(1):105--128, 2005.

[29] C. Grossmann, H. Roos, and M. Stynes. *Numerical Treatment of Partial Differential Equations*. Universitext (1979). Springer, 2007. ISBN 9783540715825. URL http://books.google.com/books?id=lmtboXIXttAC.

[30] J.-L. Guermond. Un résultat de convergence d'ordre deux en temps pour l'approximation des équations de navier--stokes par une technique de projection incrémentale. *ESAIM: Mathematical Modelling and Numerical Analysis*, 33(01):169--189, 1999.

[31] J.-L. Guermond and P. D. Minev. High-order artificial compressibility for the navier-stokes equations, part I: Theory. *in preparation*, 2015.

[32] J.-L. Guermond and M. Nazarov. A maximum-principle preserving finite element method for scalar conservation equations. *Computer Methods in Applied Mechanics and Engineering*, (0):--, 2014. ISSN 0045-7825. doi: 10.1016/j.cma.2013.12.015. URL http://www.sciencedirect.com/science/article/pii/S0045782514000024.

[33] J.-L. Guermond and R. Pasquetti. Entropy viscosity method for high-order approximations of conservation laws. In J. S. Hesthaven and E. M. Rønquist, editors, *Spectral and High Order Methods for Partial Differential Equa-*

tions, volume 76 of *Lecture Notes in Computational Science and Engineering*, pages 411--418. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-15336-5. doi: 10.1007/978-3-642-15337-2_39. URL http://dx.doi.org/10.1007/978-3-642-15337-2_39.

[34] J.-L. Guermond and R. Pasquetti. A correction technique for the dispersive effects of mass lumping for transport problems. *Computer Methods in Applied Mechanics and Engineering*, 253(0):186 -- 198, 2013. ISSN 0045-7825. doi: http://dx.doi.org/10.1016/j.cma.2012.08.011. URL http://www.sciencedirect.com/science/article/pii/S0045782512002630.

[35] J.-L. Guermond and L. Quartapelle. A projection FEM for variable density incompressible flows. *Journal of Computational Physics*, 165(1):167 -- 188, 2000. ISSN 0021-9991. doi: http://dx.doi.org/10.1006/jcph.2000.6609. URL http://www.sciencedirect.com/science/article/pii/S0021999100966099.

[36] J.-L. Guermond and L. Quartapelle. On incremental projection methods. *Pitman Research Notes in Mathematics Series*, pages 277--288, 1998.

[37] J.-L. Guermond and A. J. Salgado. Error analysis of a fractional time-stepping technique for incompressible flows with variable density. *SIAM J. Numer. Anal.*, 49(3):917--944, May 2011. ISSN 0036-1429. doi: 10.1137/090768758. URL http://dx.doi.org/10.1137/090768758.

[38] J.-L. Guermond and J. Shen. On the error estimates for the rotational pressure-correction projection methods. *Mathematics of Computation*, 73(248):1719--1737, 2004.

[39] J.-L. Guermond, C. Migeon, G. Pineau, and L. Quartapelle. Start-up flows in a three-dimensional rectangular driven cavity of aspect ratio 1:1:2 at Re = 1000. *Journal of Fluid Mechanics*, 450:169--199, 1 2002. ISSN 1469-7645. doi: 10.1017/S0022112001006383. URL http://journals.cambridge.org/article_

S0022112001006383.

[40] J.-L. Guermond, P. Minev, and J. Shen. An overview of projection methods for incompressible flows. *Computer methods in applied mechanics and engineering*, 195(44):6011--6045, 2006.

[41] J.-L. Guermond, R. Pasquetti, and B. Popov. From suitable weak solutions to entropy viscosity. *Journal of Scientific Computing*, 49(1):35--50, 2011. ISSN 0885-7474. doi: 10.1007/s10915-010-9445-3. URL http://dx.doi.org/10.1007/s10915-010-9445-3.

[42] J.-L. Guermond, R. Pasquetti, and B. Popov. Entropy viscosity method for nonlinear conservation laws. *Journal of Computational Physics*, 230(11):4248--4267, 2011.

[43] J.-L. Guermond, A. J. Salgado, and J. Shen. Splitting for variable density flows. *in preparation*, 2015.

[44] T. Heister. *A massively parallel finite element framework with application to incompressible flows*. PhD thesis, Niedersächsische Staats-und Universitätsbibliothek Göttingen, 2011.

[45] J. L. Hennessy and D. A. Patterson. *Computer Architecture, Fifth Edition: A Quantitative Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 5th edition, 2011. ISBN 012383872X, 9780123838728.

[46] V. E. Henson and U. M. Yang. Boomeramg: A parallel algebraic multigrid solver and preconditioner. *Applied Numerical Mathematics*, 41(1):155 -- 177, 2002. ISSN 0168-9274. doi: http://dx.doi.org/10.1016/S0168-9274(01)00115-5. URL http://www.sciencedirect.com/science/article/pii/S0168927401001155. Developments and Trends in Iterative Methods for Large Systems of Equations - in memorium Rudiger Weiss.

[47] V. John. *Large Eddy Simulation of Turbulent Incompressible Flows: Analytical*

and Numerical Results for a Class of Les Models. Lecture Notes in Computational Science and Engineering. Springer Berlin Heidelberg, 2004. ISBN 9783540406433. URL http://books.google.com/books?id=Y-Z0qMEiwdUC.

[48] C. Johnson, U. Nävert, and J. Pitkäranta. Finite element methods for linear hyperbolic problems. *Computer methods in applied mechanics and engineering*, 45(1):285--312, 1984.

[49] W. Kress and B. Gustafsson. Deferred correction methods for initial boundary value problems. *Journal of scientific computing*, 17(1-4):241--251, 2002.

[50] M. Kronbichler, T. Heister, and W. Bangerth. High accuracy mantle convection simulation through modern numerical methods. *Geophysics Journal International*, 191:12--29, 2012.

[51] O. Ladyzhenskaya and R. Silverman. *The mathematical theory of viscous incompressible flow*, volume 76. Gordon and Breach New York, 1969.

[52] P. D. Lax. Weak solutions of nonlinear hyperbolic equations and their numerical computation. *Communications on Pure and Applied Mathematics*, 7(1):159--193, 1954. ISSN 1097-0312. doi: 10.1002/cpa.3160070112. URL http://dx.doi.org/10.1002/cpa.3160070112.

[53] P. Lemarie-Rieusset. *Recent developments in the Navier-Stokes problem*. Chapman & Hall/CRC Research Notes in Mathematics Series. Taylor & Francis, 2002. ISBN 9781420035674. URL http://books.google.com/books?id=Rjn1NqO959sC.

[54] J. Leray. Sur le mouvement d'un liquide visqueux emplissant l'espace. *Acta Math.*, 63:193--248, 1934. ISSN 0001-5962; 1871-2509/e. doi: 10.1007/BF02547354.

[55] P.-L. Lions. *Mathematical Topics in Fluid Mechanics: Volume 1: Incompressible Models*. Mathematical Topics in Fluid Mechanics. Oxford University Press, Incorporated, 1996. ISBN 9780198514879. URL http://books.google.com/

books?id=o6p-SQAACAAJ.

[56] C. Liu and N. J. Walkington. Convergence of numerical approximations of the incompressible navier-stokes equations with variable density and viscosity. *SIAM J. Numer. Anal.*, 45(3):1287--1304, May 2007. ISSN 0036-1429. doi: 10.1137/050629008. URL http://dx.doi.org/10.1137/050629008.

[57] Message Passing Interface Forum. *MPI: A Message-Passing Interface Standard (Version 3.0)*. September 21, 2012.

[58] M. Olshanskii, G. Lube, T. Heister, and J. Löwe. Grad--div stabilization and subgrid pressure models for the incompressible navier--stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 198(49):3975--3988, 2009.

[59] M. A. Olshanskii and A. Reusken. Grad-div stabilization for stokes equations. *Mathematics of Computation*, 73(248):pp. 1699--1718, 2004. ISSN 00255718. URL http://www.jstor.org/stable/4100050.

[60] E. Olsson and G. Kreiss. A conservative level set method for two phase flow. *J. Comput. Phys.*, 210(1):225--246, November 2005. ISSN 0021-9991. doi: 10.1016/j.jcp.2005.04.007. URL http://dx.doi.org/10.1016/j.jcp.2005.04.007.

[61] S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on hamilton-jacobi formulations. *Journal of Computational Physics*, 79(1):12 -- 49, 1988. ISSN 0021-9991. doi: http://dx.doi.org/10.1016/0021-9991(88)90002-2. URL http://www.sciencedirect.com/science/article/pii/0021999188900022.

[62] J. L. Poulson. *Fast parallel solution of heterogeneous 3D time-harmonic wave equations*. PhD thesis, University of Texas at Austin, 2012.

[63] J.-H. Pyo and J. Shen. Gauge--uzawa methods for incompressible flows with variable density. *Journal of Computational Physics*, 221(1):181 -- 197, 2007.

ISSN 0021-9991. doi: http://dx.doi.org/10.1016/j.jcp.2006.06.013. URL http://www.sciencedirect.com/science/article/pii/S0021999106002816.

[64] R. Rannacher. On chorin's projection method for the incompressible navier-stokes equations. In J. Heywood, K. Masuda, R. Rautmann, and V. Solonnikov, editors, *The Navier-Stokes Equations II ÂŮ Theory and Numerical Methods*, volume 1530 of *Lecture Notes in Mathematics*, pages 167--183. Springer Berlin Heidelberg, 1992. ISBN 978-3-540-56261-0. doi: 10.1007/BFb0090341. URL http://dx.doi.org/10.1007/BFb0090341.

[65] J. Reinders. *Intel Threading Building Blocks*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, first edition, 2007. ISBN 9780596514808.

[66] Y. Saad. *Iterative Methods for Sparse Linear Systems: Second Edition*. Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 2003. ISBN 9780898718003. URL http://books.google.com/books?id=h9nwszYPblEC.

[67] Y. Saad and M. Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7(3): 856--869, July 1986. ISSN 0196-5204. doi: 10.1137/0907058. URL http://dx.doi.org/10.1137/0907058.

[68] J. Shen. On error estimates of some higher order projection and penalty-projection methods for navier-stokes equations. *Numerische Mathematik*, 62 (1):49--73, 1992. ISSN 0029-599X. doi: 10.1007/BF01396220. URL http://dx.doi.org/10.1007/BF01396220.

[69] J. Shen. A remark on the projection-3 method. *International Journal for Numerical Methods in Fluids*, 16(3):249--253, 1993. ISSN 1097-0363. doi: 10.1002/fld.1650160308. URL http://dx.doi.org/10.1002/fld.1650160308.

[70] J. Shen. On error estimates of the penalty method for unsteady navier-stokes

equations. *SIAM J. Numer. Anal.*, 32(2):386--403, April 1995. ISSN 0036-1429. doi: 10.1137/0732016. URL http://dx.doi.org/10.1137/0732016.

[71] J. Shen. On error estimates of the projection methods for the navier-stokes equations: Second-order schemes. *Mathematics of Computation*, 65(215):pp. 1039--1065, 1996. ISSN 00255718. URL http://www.jstor.org/stable/2153791.

[72] J. Smagorinsky. General circulation experiments with the primitive equations. *Monthly Weather Review*, 91(3):99--164, 2015/01/06 1963. doi: 10.1175/1520-0493(1963)091<0099:GCEWTP>2.3.CO;2. URL http://dx.doi.org/10.1175/1520-0493(1963)091<0099:GCEWTP>2.3.CO;2.

[73] W. Stevens, G. Myers, and L. Constantine. Classics in software engineering. chapter Structured Design, pages 205--232. Yourdon Press, Upper Saddle River, NJ, USA, 1979. ISBN 0-917072-14-6. URL http://dl.acm.org/citation.cfm?id=1241515.1241533.

[74] M. Sussman, P. Smereka, and S. Osher. A level set approach for computing solutions to incompressible two-phase flow. *Journal of Computational Physics*, 114(1):146 -- 159, 1994. ISSN 0021-9991. doi: http://dx.doi.org/10.1006/jcph.1994.1155. URL http://www.sciencedirect.com/science/article/pii/S0021999184711557.

[75] R. Témam. Sur l'approximation de la solution des équations de navier-stokes par la méthode des pas fractionnaires (ii). *Archive for Rational Mechanics and Analysis*, 33(5):377--385, 1969. ISSN 0003-9527. doi: 10.1007/BF00247696. URL http://dx.doi.org/10.1007/BF00247696.

[76] R. Témam. *Navier-Stokes Equation: Theory and Numerical Analysis*, volume 2. North-Holland, 1977.

[77] Texas A&M University, College Station, Texas. Brazos Computational Resource. Academy for Advanced Telecommunications and Learning Technologies, 2015.

[78] J. Thomas. *Numerical Partial Differential Equations: Finite Difference Methods*. Number v. 1 in Graduate Texts in Mathematics. Springer, 1995. ISBN 9780387979991. URL http://books.google.com/books?id=op5COPwUfX8C.

[79] L. Timmermans, P. Minev, and F. Van De Vosse. An approximate projection scheme for incompressible flow using spectral elements. *International Journal for Numerical Methods in Fluids*, 22(7):673--688, 1996. ISSN 1097-0363. doi: 10.1002/(SICI)1097-0363(19960415)22:7<673::AID-FLD373>3.0.CO;2-O. URL http://dx.doi.org/10.1002/(SICI)1097-0363(19960415)22:7<673::AID-FLD373>3.0.CO;2-O.

[80] G. Tryggvason. Numerical simulations of the rayleigh-taylor instability. *J. Comput. Phys.*, 75(2):253--282, April 1988. ISSN 0021-9991. doi: 10.1016/0021-9991(88)90112-X. URL http://dx.doi.org/10.1016/0021-9991(88)90112-X.

[81] H. Uzawa. Iterative methods for concave programming. In *Preference, production, and capital*, pages 135--148. Cambridge University Press, 1989. ISBN 9780511664496. URL http://dx.doi.org/10.1017/CBO9780511664496.011. Cambridge Books Online.

[82] H. van der Vorst. Bi-cgstab: A fast and smoothly converging variant of bi-cg for the solution of nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 13(2):631--644, March 1992. ISSN 0196-5204. doi: 10.1137/0913035. URL http://dx.doi.org/10.1137/0913035.

[83] J. van Kan. A second-order accurate pressure-correction scheme for viscous incompressible flow. *SIAM Journal on Scientific and Statistical Computing*, 7 (3):870--891, 1986. doi: 10.1137/0907059. URL http://dx.doi.org/10.1137/0907059.

[84] J. Von Neumann and R. D. Richtmyer. A method for the numerical calculation

of hydrodynamic shocks. *Journal of Applied Physics*, 21(3):232--237, 1950. doi: http://dx.doi.org/10.1063/1.1699639. URL http://scitation.aip.org/content/aip/journal/jap/21/3/10.1063/1.1699639.