Investigation of Computational and Visualization Methods for the Incompressible Navier-Stokes Equations

in pursuit of

M.S. Thesis in Scientific Computing

by

M. Harper Langston

Denis Zorin, Advisor

Advisor: ____

Second Reader: ____

(Yu Chen)

May 2003

Department of Mathematics and Computer Science

New York University Courant Institute and

New York University Media Research Lab

Abstract

Investigation of Computational and Visualization Methods for the Incompressible Navier-Stokes Equations

in pursuit of

M.S. Thesis in Scientific Computing

by M. Harper Langston

May 2003

The incompressible Navier-Stokes equations (NSE) are investigated from their derivation to numerical solution methods to visualization methods. We provide the derivation in order to better understand the development of analysis techniques and numerical solutions. Specifically, we study the projection methods based on the Chorin-Temam scheme as well as provide a general guideline for the time and space discretization of the incompressible NSE. We investigate Line Integral Convolution as a visualization method for fluid flows, and then a technique for visualizing the interactions of 3-D rigid bodies in highly viscous flows. Finally, we discuss directions, in which we are taking this work as well as current developments in the study of fluid flows at the New York University Media Research Lab.

Table of Contents

List of Figures

Acknowledgments

I would like to thank Denis Zorin, George Biros and Lexing Ying for their help and guidance and allowing me to constantly bother them and borrow their books. I would also like to thanks Michael Shelley for his encouragement and providing much of the material and references for Chapter 1. Further, I would like to thank Yu Chen for sparking much of my interest in numerical analysis. And of course, I thank Victoria for putting up with my late nights.

Chapter 1

Introduction to Incompressible Navier-Stokes

The Navier-Stokes equations have a long history of being studied and evaluated as a model for the simulation of viscous fluids in two and three dimensions for both compressible and incompressible fluids. Additionally, they have provided years of numerical analysis and simulation. Before delving into an analysis of these equations, it is important to understand their origins. We provide a brief derivation of the incompressible version of the Navier-Stokes equations. There are several sources for a much more in-depth derivation of these equations such as [3], [30], and [38] for example, but here we provide an overview of these for the casual reader.

1.1 Conservation of Mass and the Material Derivative

Consider a fluid volume, $\mathbf{V} \subseteq \Re^3$, and at every point, $\mathbf{x} \in \mathbf{V}$, the fluid has density, $\rho(\mathbf{x}, t)$, and velocity, $\mathbf{u}(\mathbf{x}, t)$, for $\mathbf{x} = (x_1, x_2, x_3)$ and $\mathbf{u} = (u_1, u_2, u_3)$. Also consider a fixed subvolume, $\mathbf{V}_0 \subseteq \mathbf{V}$ with outward normal, $\mathbf{n}$, such that the mass, M, within $\mathbf{V}_0$ is:

$$M[\mathbf{V}_0, t] = \int_{\mathbf{V}_0} \rho(\mathbf{x}, t)\, d\mathbf{V}.$$

Then, the rate of change of $M[\mathbf{V}_0, t]$ is conserved by the flux of mass through $\partial\mathbf{V}_0$:

$$\frac{dM}{dt} = \frac{d}{dt}\int_{\mathbf{V}_0} \rho(\mathbf{x}, t)\, d\mathbf{V} = -\int_{\partial\mathbf{V}_0} [\rho(\mathbf{x}, t)\mathbf{u}(\mathbf{x}, t)] \cdot \mathbf{n}(\mathbf{x})\, d\mathbf{A}.$$

Applying the divergence theorem from the Appendix to the right-hand side, we obtain 1.1.

$$\frac{d}{dt}\int_{\mathbf{V}_0} \rho(\mathbf{x}, t) d\mathbf{V} = -\int_{\mathbf{V}_0} \nabla \cdot (\rho\mathbf{u}) d\mathbf{V} \tag{1.1}$$

The fact that $\mathbf{V}_0$ is arbitrary implies

$$\frac{d\rho}{dt} = -\nabla \cdot (\rho\mathbf{u}).$$

This equation, the differential form of the mass conservation equation, provides us with a PDE for describing the evolution of ρ in the 'Eulerian frame', where ρ and $\mathbf{u}$ are evaluated at a fixed $\mathbf{x}$. Instead, we would like to evaluate our quantities in the 'Lagrangian frame'. That is, we evaluate ρ and $\mathbf{u}$ in a moving frame which travels with the fluid. For example, let $\boldsymbol{x}(t)$ satisfy:

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{u}(\boldsymbol{x}(t), t).$$

$$\boldsymbol{x}(0) = \boldsymbol{x_0}.$$

We call $\boldsymbol{x}(t)$ is a Lagrangian material path (figure 1.1), and $\rho(\boldsymbol{x}(t), t)$ is the fluid density along the Lagrangian path, $\boldsymbol{x}(t)$.

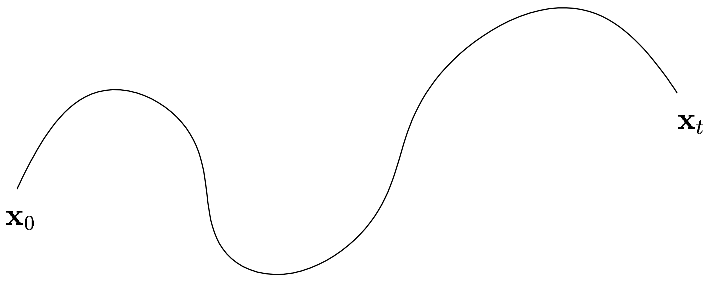


Figure 1.1: A fluid particle path, tracing a particle from $\mathbf{x}_0$ to $\mathbf{x}_t$

We define the *material or substantial derivative* [3] as:

$$\frac{D}{dt} = \frac{\partial}{\partial t} + u \cdot \nabla. \tag{1.2}$$

This derivative considers that fluid particles travel in time and allows us to take the derivative of some variable as it 'follows the fluid'. For example, for some function $\mathbf{f}$:

$$\frac{D\mathbf{f}}{dt} = \frac{\partial \mathbf{f}}{\partial t} + u \cdot \nabla \mathbf{f}.$$

Hence, equation 1.2 becomes:

$$\frac{D\rho}{dt} = -\rho \nabla \cdot \mathbf{u}.$$

Several properties hold for the material derivative, but most importantly, if $\frac{D\mathbf{f}}{dt} = 0$, then for some particle path, $\boldsymbol{x}(t)$, the value of $\mathbf{f}$ remains unchanged. That is, $\mathbf{f}(\mathbf{x}(t), t) = \mathbf{f}(\mathbf{x_0}, 0)$ for such an $\mathbf{f}$. Hence, for an incompressible flow we have our first real development! That is, for $\rho = constant$, $\frac{D\rho}{dt} = 0$ which gives our *incompressibility condition*:

$$\nabla \cdot \mathbf{u} = 0.$$

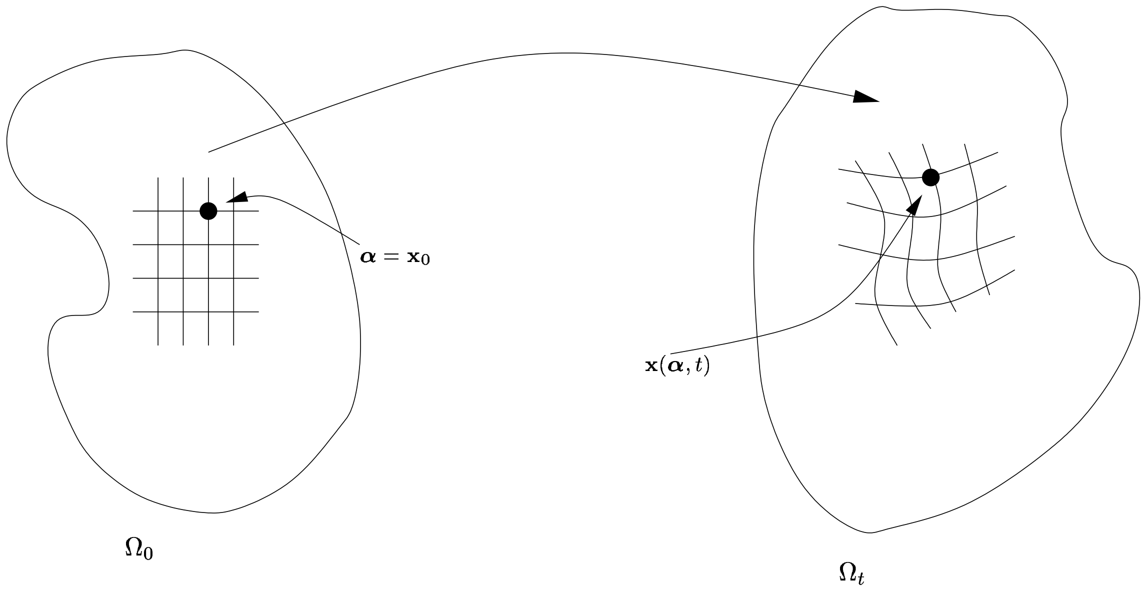
1.2 The Lagrangian Formulation

Let $\Omega_t \subseteq \mathbf{V}$ be our time-dependent volume, moved by the fluid motion from the initial volume, Ω_0. That is, if we were to solve $\frac{d\mathbf{x}}{dt} = \mathbf{u}(\mathbf{x}(t), t)$; $\mathbf{x}(0) = \mathbf{x}_0$ for all $\mathbf{x}_0 \subseteq \Omega_0$, then Ω_t is the set of all $\mathbf{x}(t, \mathbf{x}_0)$. Ω_t is called a *Lagrangian* *volume* and a *Lagrangian variable* [3] is a variable that remains constant along a fluid particle path, as seen in the last section for density, ρ in an incompressible flow.

To construct our flow map, we use the set of initial coordinates, $\mathbf{x}_0 \subseteq \Omega_0$ as Lagrangian variables and transform our spatial variables, where $\boldsymbol{x}(\boldsymbol{\alpha}, t)$, our flow map and $\boldsymbol{\alpha}$, our Lagrangian variable, are constructed by solving:

$$\frac{d\boldsymbol{x}}{dt} = \boldsymbol{u}(\boldsymbol{x}(\boldsymbol{\alpha}, t), t).$$
$$\boldsymbol{x}(\boldsymbol{\alpha}, 0) = \boldsymbol{\alpha}.$$

At time t, the coordinates $(x_1, \ldots, x_n)$ are parameterized by $(\alpha_1, \ldots, \alpha_n)$ where $\boldsymbol{\alpha}$, is the initial condition of $\mathbf{x}$, flowed back to $t = 0$. Figure 1.2 shows an example of the Lagrangian flow map.



$\boldsymbol{\alpha} = \mathbf{x}_0$
$\mathbf{x}(\boldsymbol{\alpha}, t)$
Ω_0
Ω_t

$\mathbf{x}(\boldsymbol{\alpha}, t)$ is the Lagrangian (flow) map.
$\boldsymbol{\alpha}$ is the Lagrangian variable.

Figure 1.2: The Lagrangian Map from Ω_0 to Ω_t

1.3 Reynolds Transport Theorem

We define $J(\boldsymbol{\alpha}, t)$as the Jacobian of the flow map $\boldsymbol{x}(\boldsymbol{\alpha}, t)$. That is,

$$J(\boldsymbol{\alpha}, t) = det \left[\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\alpha}} \right] = det \begin{bmatrix} \frac{\partial x_1}{\partial \alpha_1} & \frac{\partial x_1}{\partial \alpha_2} & \frac{\partial x_1}{\partial \alpha_3} \\ \frac{\partial x_2}{\partial \alpha_1} & \frac{\partial x_2}{\partial \alpha_2} & \frac{\partial x_2}{\partial \alpha_3} \\ \frac{\partial x_3}{\partial \alpha_1} & \frac{\partial x_3}{\partial \alpha_2} & \frac{\partial x_3}{\partial \alpha_3} \end{bmatrix}.$$

where $J(\boldsymbol{\alpha}, 0) = 1$. Further, if we want to differentiate $J(\boldsymbol{\alpha}, t)$, we can use Lioville's Formula:

Lemma 1 (Lioville)

$$\tfrac{\partial}{\partial t} J(\boldsymbol{\alpha}, t) = (\nabla \cdot \boldsymbol{u})(\boldsymbol{x}(\boldsymbol{\alpha}, t), t) \cdot J(\boldsymbol{\alpha}, t)$$

A proof of Lioville's Formula is found in [12].

Using our new understanding of J, we express conservation of mass from Section 1.1 as:

$$J(\boldsymbol{\alpha}, t) = \rho(\boldsymbol{x}(\boldsymbol{\alpha}, t), t) = \rho(\boldsymbol{\alpha}, 0).$$

Intuitively, this means that as J increases, ρ decreases and vice versa. Further, for an incompressible fluid, where $\rho = constant$, then $J = 1$, an additional form of our *incompressibility condition*, from Equation (1.8). Using Lioville's Formula, we state the important theorem, which relates the rates of change of volume integrals over a portion of fluid, most eloquently shown in [1].

Theorem 1.3.1 (Reynolds Transport Theorem) *For a function, f(*$\mathbf{x}, t$*) and* $\Omega_t \subseteq \mathbf{V}$

$$\frac{d}{dt} \int_{\Omega_t} \rho \mathbf{f} d\mathbf{V} = \int_{\Omega_t} \rho \frac{D\mathbf{f}}{Dt} d\mathbf{V}.$$

Using our new form of conservation of mass, we can easily prove this theorem:

Proof of Reynolds's Transport Theorem

$$\begin{aligned} \frac{d}{dt} \int_{\Omega_t} \mathbf{f} \rho d\mathbf{V} &= \frac{d}{dt} \int_{\Omega_0} (J\rho) \mathbf{f} d\mathbf{V} \\ &= \int_{\Omega_0} \frac{\partial}{\partial t} [(J\rho)\mathbf{f}] d\mathbf{V} \\ &= \int_{\Omega_0} (J\rho) \frac{D\mathbf{f}}{Dt} (\mathbf{x}(\boldsymbol{\alpha}, t), t) d\mathbf{V} \\ &= \int_{\Omega_t} \rho \frac{D\mathbf{f}}{Dt} d\mathbf{V} \end{aligned}$$

1.4 Balance of Momentum

If $\mathbf{x}(t)$ is a fluid particle path with velocity $\mathbf{u}(\mathbf{x}(t), t)$, then we give the acceleration of a particle in the fluid by:

$$\begin{aligned} \mathbf{a}(t) &= \frac{d^2}{dt^2} \mathbf{x}(t) \\ &= \frac{d}{dt} \mathbf{u}(\mathbf{x}(t), t) \\ &= \frac{\partial \mathbf{u}}{\partial t} + \frac{\partial \mathbf{u}}{\partial \mathbf{x}} \cdot \dot{x} \\ &= \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} \\ &= \frac{D\mathbf{u}}{Dt} \end{aligned}$$

We temporarily abandon the acceleration of a particle while we look at the forces present in a fluid:

1. Body Forces. Body forces are comprised of externally imposed force fields (i.e., gravity or electromagnetics), which exert a force per unit mass. So, let $\mathbf{F}(\mathbf{x}, t)$ be our body force. Then, the total body forces, $\mathbf{F}_{body}$ exerted upon Ω_t is:

$$\mathbf{F}_{body} = \int_{\Omega_t} \rho \mathbf{F} d\mathbf{V}.$$

2. Forces of Stress. We define *stress* as the force per unit area that a fluid exerts. Often, we say that given a surface with normal, $\mathbf{n}$, the stress is a force exerted by the fluid in the direction of $\mathbf{n}$, acting on the fluid in the $-\mathbf{n}$ direction [3]. So, the forces of stress are those forces which arise from the contact of a fluid volume, Ω_t, across its boundary, $\partial\Omega_t$ with the rest of the fluid. Further, we denote this stress force as $\mathbf{S}$, and we have an important theorem, *Cauchy's Theorem* which states that $\mathbf{S}$ is a linear function of our outward surface normal, $\mathbf{n}$ ([12] and [38], proof: [33]).

Theorem 1.4.1 (Cauchy's Theorem)

$$\mathbf{S} = \sigma \mathbf{n}$$

where σ is a second-rank stress tensor. For most fluid flows, σ is symmetric, and in fact, we make several assumptions about it, succinctly stated in [12]:

1. *σ depends linearly on* $\nabla \mathbf{u}$.
2. *σ does not change under rigid-body rotation* .
3. *σ is symmetric* .

For many fluid flows, a large portion of the modeling is related to the way in which the stress tensor, σ is specified. For example, ideal fluids, viscous fluids, polymeric fluids, etc. all use different modelings of σ. We explore this stress tensor in more detail in the next section for Newtonian flows.

For now, however, we simply note that the forces of stress, $\mathbf{F}_{stress}$ on Ω_t is written as:

$$\mathbf{F}_{stress} = \int_{\partial\Omega_t} (\sigma \cdot \mathbf{n}) d\mathbf{A}.$$

Knowing the fact that the momentum of a material volume of fluid can be evaluated as $\int_{\Omega_t} \rho \mathbf{u} d\mathbf{V}$, then, using Newton's Second Law, we write an equation for the total forces on our fluid as seen in the following theorem. [3]

Theorem 1.4.1 (Newton's Second Law)

$$\frac{d}{dt} \int_{\Omega_t} \rho \mathbf{u} d\mathbf{V} = \mathbf{F}_{body} + \mathbf{F}_{stress}.$$

which we write out as:

$$\frac{d}{dt} \int_{\Omega_t} \rho \mathbf{u} d\mathbf{V} = \int_{\Omega_t} \rho \mathbf{F} d\mathbf{V} + \int_{\partial\Omega_t} (\sigma \cdot \mathbf{n}) d\mathbf{A}.$$

Using Reynolds's Transport Theorem and our derivation of the acceleration of a particle, we rewrite this first part as:

$$\frac{d}{dt}\int_{\Omega_t} \rho \mathbf{u} d\mathbf{V} = \rho\frac{D\mathbf{u}}{Dt}.$$

And, using the Divergence Theorem, we rewrite the last part, describing the total forces of stress as:

$$\mathbf{F}_{stress} = \int_{\partial\Omega_t} (\sigma \cdot \mathbf{n}) d\mathbf{A} = \nabla \cdot \sigma.$$

So, we now have the *Differential Form of the Momentum Equation*:

$$\rho\frac{D\mathbf{u}}{Dt} = \rho\mathbf{F} + \nabla \cdot \sigma.$$

1.5 Translation, Deformation and Rotation

In the simplest case, we have a fluid at rest, that is an ideal fluid for which $\mathbf{S} = -P\mathbf{n}$ for normal, $\mathbf{n}$, and pressure, P, the fluid pressure at rest. In this case, $\sigma = -PI$, [3] and hence, we get Euler's Equation:

Theorem 1.5.1 (Euler's Equation, 1755)

$$\rho\frac{D\mathbf{u}}{Dt} = -\nabla p + \rho\mathbf{F}.$$

Along with our equation for conservation of mass, $\frac{D\mathbf{u}}{Dt} = -\rho\nabla \cdot \mathbf{u}$, we have a closed system. Further, for an incompressible fluid, we remember that Conservation of Mass is equivalent to $\nabla \cdot \mathbf{u} = 0$.

The above describes a static fluid system, however, we are more interested in fluids in motion. In order to do this, we look at σ more closely and use several results about a Stokesian or Newtonian fluid. Firstly, since we know that σ is a second-rank tensor, where $S_i = \Sigma_j \sigma_{ij} n_j$, the each σ_{ij} is just the stress in the i-direction on a piece of surface $\partial\mathbf{S}$ taken with the j-component of the normal, $\mathbf{n}$ [1]. Further, from the previous section, we assumed that σ relies linearly in $\nabla\mathbf{u}$, so we look more closely at $\nabla\mathbf{u}$.

Fix $\mathbf{x}$ and consider a new point, $\mathbf{x} + \mathbf{r}$ where $|\mathbf{r}| << 1$ as in figure 1.3. Then, using Taylor's Expansion, we write

$$\mathbf{u}(\mathbf{x} + \mathbf{r}) = \mathbf{u}(\mathbf{x}) + \nabla\mathbf{u} \cdot \mathbf{r} + O(|r|^2).$$

Note that we rewrite the velocity gradient as

$$\nabla\mathbf{u} = \frac{1}{2}[\nabla\mathbf{u} + \nabla\mathbf{u}^T] + \frac{1}{2}[\nabla\mathbf{u} - \nabla\mathbf{u}^T].$$

where

$$E = \frac{1}{2}[\nabla\mathbf{u} + \nabla\mathbf{u}^T].$$

is a symmetric matrix and

$$R = \frac{1}{2}[\nabla\mathbf{u} - \nabla\mathbf{u}^T].$$

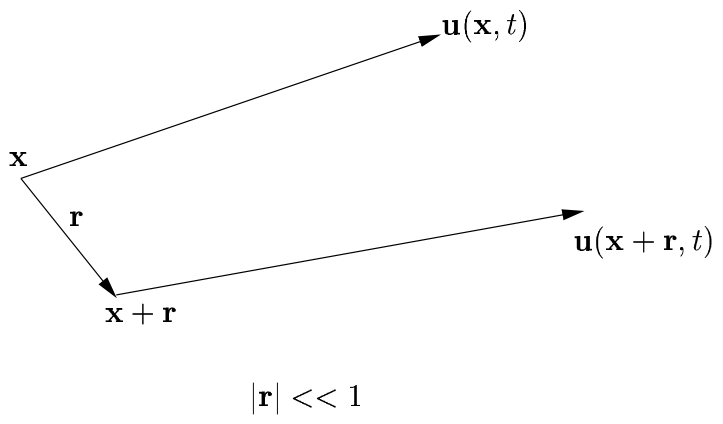


Figure 1.3: Moving our point an arbitrarily small amount

is an anti-symmetric matrix.

If we first look at E, then we see that since E is symmetric, it has real eigenvalues, λ_i, and mutually orthogonal eigenvectors, ψ_i. E is commonly known as the *symmetric rate-of-strain tensor* [3] where the eigenvalues are the principal rates of strain and the eigenvectors are the principle directions of strain. If we look at the effect of $E\mathbf{r} = (\Psi^{-1}\Lambda\Psi)\mathbf{r}$ graphically represented in figure 1.4, we see that E compresses and expands a ball centered around $\mathbf{r}$ into an ellipsoid whose axes are the principal directions of strain. For an incompressible fluid, the resulting ellipsoid maintains the same volume as the original ball and the diagonal elements of E are zero [3].

For our anti-symmetric part, we have

$$R = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}$$

where

$$\boldsymbol{\omega} = \nabla \times \mathbf{u}.$$

is the *vorticity* of the fluid or *curl* of $\mathbf{u}$.

The effect of

$$R\mathbf{r} = \frac{1}{2}\boldsymbol{\omega} \times \mathbf{r}.$$

is a divergence-free rigid-body rotation with angular velocity $\frac{1}{2}\boldsymbol{\omega}$. Hence, we have

$$\mathbf{u}(\mathbf{x}+\mathbf{r}) = \mathbf{u}(\mathbf{x}) + E\mathbf{r} + R\mathbf{r} + O(|r|^2).$$

where $\mathbf{u}(\mathbf{x})$ represents our translation, $E\mathbf{r}$ represents our deformation and $R\mathbf{r}$ represents our rotation.

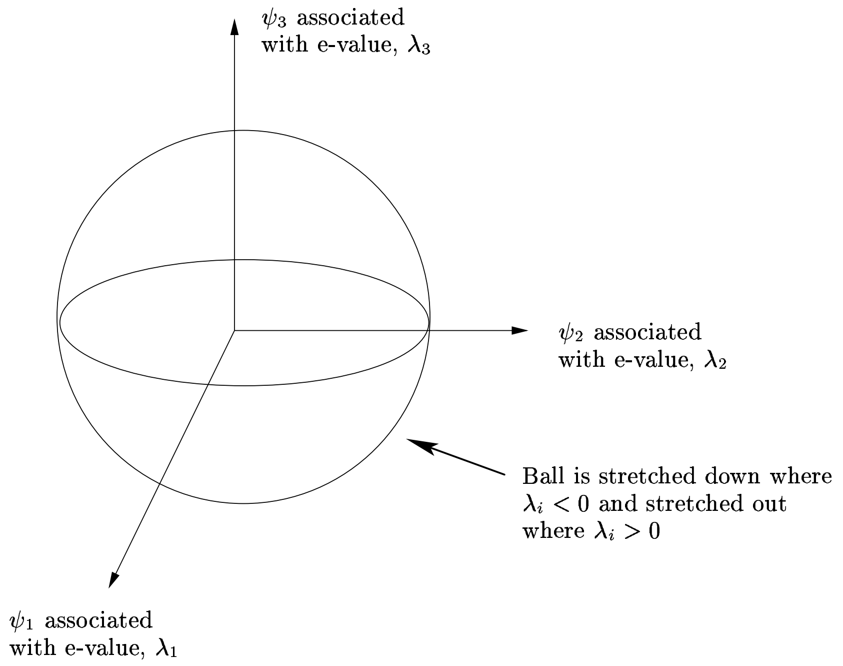

ψ_3 associated with e-value, λ_3
ψ_2 associated with e-value, λ_2
Ball is stretched down where $\lambda_i < 0$ and stretched out where $\lambda_i > 0$
ψ_1 associated with e-value, λ_1

Figure 1.4: Graphical representation of the rate of strain tensor

1.6 *Evaluating the Stress Tensor for Incompressible Navier-Stokes*

Now that we know how to represent $\nabla \mathbf{u}$, and since we know that σ is symmetric, then we know that σ depends linearly on the symmetric part of $\nabla \mathbf{u}$ only. Since we know that for a fluid at rest, $\sigma = -PI$ for the static pressure, then for a fluid in motion

$$\sigma = -PI + D.$$

where D is the *deviatoric stress tensor*. Since $D = 0$ for a fluid at rest, $-PI$ represents the *isotropic* portion of our fluid while D is the *anisotropic* portion such that $trace(D) = 0$ [3].

Clearly there is a relation between the deviatoric stress tensor and rate-of-strain tensor in a Newtonian or Stokesian fluid. Through analysis in [3] and [38], we obtain that for a Newtonian viscous fluid (ignoring our incompressibility constraint for now):

$$D = 2\mu E + \eta(\nabla \cdot \mathbf{u})I.$$

for viscosities, μ and η and E, our symmetric rate-of-strain tensor. μ is the *kinematic viscosity* while η is the *dynamic viscosity* for which measured values is found [30]. Taking the trace of D,

$$trace(D) = 2\mu + 3\eta = 0.$$

such that $\eta = -\frac{2}{3}\mu$, so that now:

$$\sigma = -PI + 2\mu E - \frac{2}{3}(\nabla \cdot \mathbf{u})I.$$

.

Plugging into our equation of momentum, we obtain the general Navier-Stokes equation:

$$\rho \frac{D\mathbf{u}}{Dt} = -\nabla p + \nabla[2\mu[E - \frac{1}{3}(\nabla \cdot \mathbf{u})I]] + \rho \mathbf{F}.$$

Introducing our incompressibility constraint, $\nabla \cdot \mathbf{u} = 0$, defining $\nu = \mu/\rho$, scaling P by ρ such that we say the pressure is $p = P/\rho$, and noticing that $\nabla(2\mu E) = \mu \Delta \mathbf{u}$, we write the complete Navier-Stokes equations for an incompressible fluid as:

$$\frac{D\mathbf{u}}{Dt} + \nabla p - \nu \Delta \mathbf{u} = \mathbf{F}. \tag{1.3}$$

$$\nabla \cdot \mathbf{u} = 0. \tag{1.4}$$

Chapter 2

Analysis of the Incompressible Navier-Stokes Equations

In order to be able to successfully solve the Incompressible Navier-Stokes equations, it is important to have a deeper understanding of the many aspects of these equations, and what is needed in different circumstances. For example, we need to analyze what effect the viscosity has on our approach toward solving the equations. Additionally, we have not discussed the need for boundary conditions or how pressure comes into effect in the incompressible case. We look at all of these aspects and discuss them here.

2.1 Diffusion and Convective Terms

Expanding the result from Chapter 1, write the incompressible Navier-Stokes equations as:

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla p - \nu \Delta \mathbf{u} + (\nabla \cdot \mathbf{u})\mathbf{u} = \mathbf{F}. \tag{2.1}$$

$$\nabla \cdot \mathbf{u} = 0. \tag{2.2}$$

where $\nu\Delta\mathbf{u}$ is the diffusion term and $(\nabla \cdot \mathbf{u})\mathbf{u}$ is the convective term.

Through these two terms, the Navier-Stokes equations dictate how the velocity, $\mathbf{u}$ of fluid particles are convected and diffused concurrently at rates based partially on forces of pressure [12]. The interaction and relative rates between the two depends on which term dominates, and the domination of one term influences solution techniques. Fortunately, we have a way of judging which term dominates in the form of *Reynolds number*.

2.2 Reynolds Number and the Stokes Equations

The *Reynolds Number*, Re, is a quantity determined by

$$Re = \frac{UL}{\nu}.$$

where U is a typical flow velocity and L is a characteristic flow length. For example, we could have a fluid flow in a pipe of width, L, and typical speed, U as seen in figure 2.1.

Other examples are found in [1] and [12] where, for example, the characteristic length may be the diameter or radius of a cylinder, around which a fluid is traveling. We now set $T = \frac{L}{U}$ and scale $\mathbf{x}$, $\mathbf{u}$ and t by U, L and T as $\hat{\mathbf{x}} = \frac{\mathbf{x}}{U}$, $\hat{\mathbf{u}} = \frac{\mathbf{u}}{L}$ and $\hat{t} = \frac{t}{T}$. Using these to do a change of variables in the Navier-Stokes equation, shown in [12], we get:

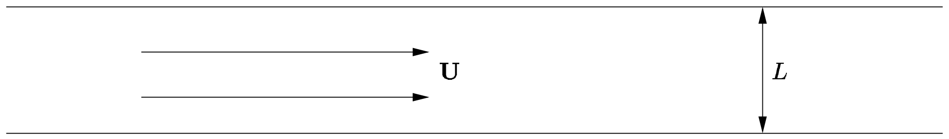


Figure 2.1: Simple example for determination of Reynolds Number based on L and $\mathbf{U}$ in 2D

$$\frac{\partial \hat{\mathbf{u}}}{\partial t} + \nabla p - \frac{1}{Re}\Delta \hat{\mathbf{u}} + (\nabla \cdot \hat{\mathbf{u}})\hat{\mathbf{u}} = \mathbf{F}.$$

Based on this derivation, the Reynolds number shows that it is the only parameter which differentiates incompressible fluid flows. Equivalently, we can use the Reynolds number to model similar flows. For example, given two geometrically similar flows, $\mathbf{u}_1$ and $\mathbf{u}_2$ with Reynolds numbers, Re_1 and Re_2, respectively, if $Re_1 = Re_2$, then $\mathbf{u}_1$ and $\mathbf{u}_2$ are *similar flows*. That is, even if $L_1 \neq L_2$, $U_1 \neq U_2$ or $\nu_1 \neq \nu_2$, when $Re_1 = Re_2$, then $\mathbf{u}_1$ and $\mathbf{u}_2$ are equivalent scaled solutions to the Navier-Stokes equations. Hence, large fluid problems can be designed and studied in smaller-scale and provide similar results [12].

Other than helping in actually design situations, the Reynolds number can also help determine whether more emphasis needs to be placed on the solving of the diffusion term, $\nu\Delta\mathbf{u}$, or convective term, $(\nabla \cdot \mathbf{u})\mathbf{u}$. For example, for fluid flows with very little viscosity (ν very small), $Re >> 1$. Hence, in the face of the convective term, the diffusion term is considered negligible. These flows are often considered unstable ([1],[38]) and are often of more interest in compressible flows [12], [3]. However, it is important to notice that this brings us back to Euler's Equations for a perfect fluid from Chapter 1. This makes sense since a perfect fluid is viscosity free.

On the other hand, when $Re << 1$, the convective term is negligible compared to the diffusion term. This is more typical in liquids in small areas for example. In this case, the Navier-Stokes equations are approximated to high accuracy by the simpler *Stokes* equations.

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla p - \nu\Delta\mathbf{u} = \mathbf{F}. \tag{2.3}$$

$$\nabla \cdot \mathbf{u} = 0. \tag{2.4}$$

The Stokes equations are studied briefly in the next chapter and visualized in Chapter 4.

2.3 Boundary Conditions

In order to fully describe the Navier-Stokes equations, and to be able to solve them numerically, we need to provide conditions at the boundaries. For Euler's Equations we need to say that no fluid can physically move through the boundary. Think of the simple example of a 2-dimensional box of size $[0,1]^2$ where fluid cannot pass through the edges for example. That is, we set $\mathbf{u} = 0$ outside of the boundaries as n figure 2.2.

Since the boundary has outward normal, $\mathbf{n}$, we must impose the condition

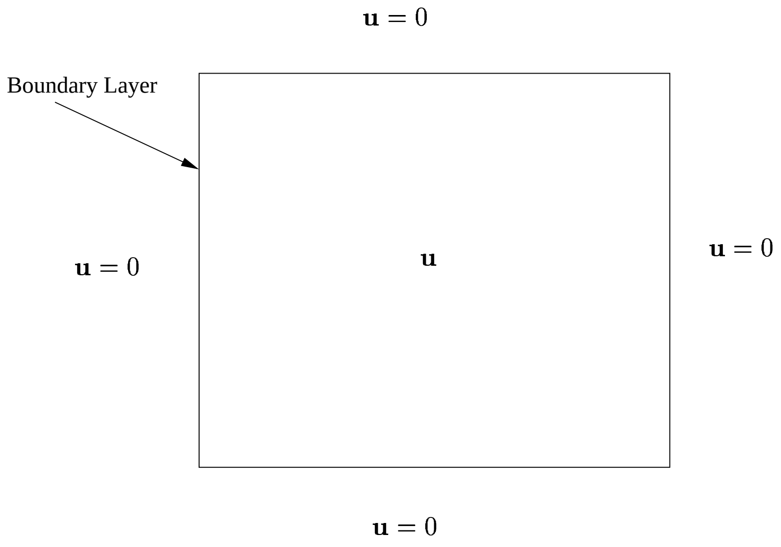


Figure 2.2: 2-d box with fluid velocities equal to zero on the outside

$$\mathbf{u} \cdot \mathbf{n} = 0.$$

However, the Navier-Stokes equations includes the diffusion term, $\Delta\mathbf{u}$, while Euler's Equation for ideal flow only involves the convective term. As a result, we need an extra boundary condition [12]. This comes in the form of what is typically known as the 'no-slip' condition. That is, we require that the velocity at the boundaries is zero.

$$\mathbf{u} = 0.$$

So, figure 2.2 would require that a particle approaching a wall has velocity $\mathbf{u} \to 0$. But would this mean that a particle gets stuck at the boundary? Obviously not. In fact, the 'no-slip' condition actually guarantees that a particle's tangential-velocity at a boundary is zero. For example, in two dimensions, a fluid particle approaching a boundary at some x-value, $x = x_0$, has its velocity in the x-direction $\to 0$, and hence the particle is forced either in the $-y$ or $+y$ direction. Modifying our figure above, say that our boundaries are at $x = 0$, $y = 0$, $x = 1$ and $y = 1$. Our new boundary conditions is seen in figure 2.3.

This type of boundary condition is commonly known as a Dirichlet boundary condition. However, we may be interested in setting a boundary condition which dictates the change in the value of a variable across the boundary. Such a boundary condition is known as a Neumann boundary condition. In fact, we see such a condition in Chapter 3 for the pressure, in which case the boundary condition is $\frac{\partial p}{\partial \mathbf{n}} = 0$ on $\partial\Omega$ where we have surface normal, $\mathbf{n}$. That is, we want the change in the pressure across the boundary to remain constant (zero flux). An example of this is seen in figure 2.3 for 2-D.

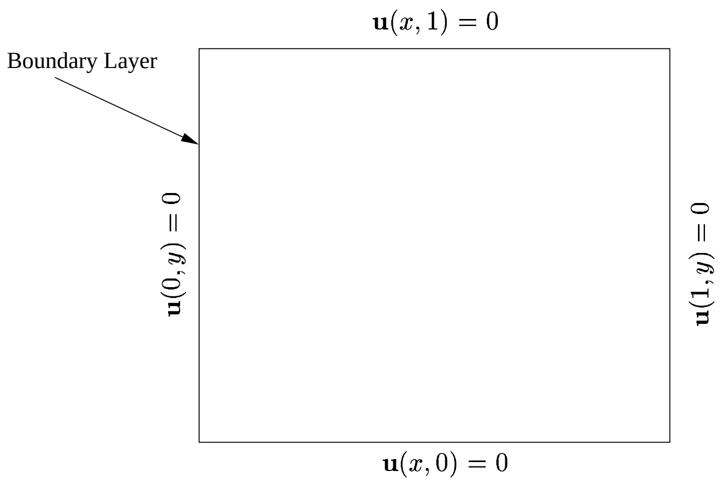


Figure 2.3: 2-d box with tangential fluid velocities equal to zero at the boundaries, $x = 0$, $y = 0$, $x = 1$ and $y = 1$

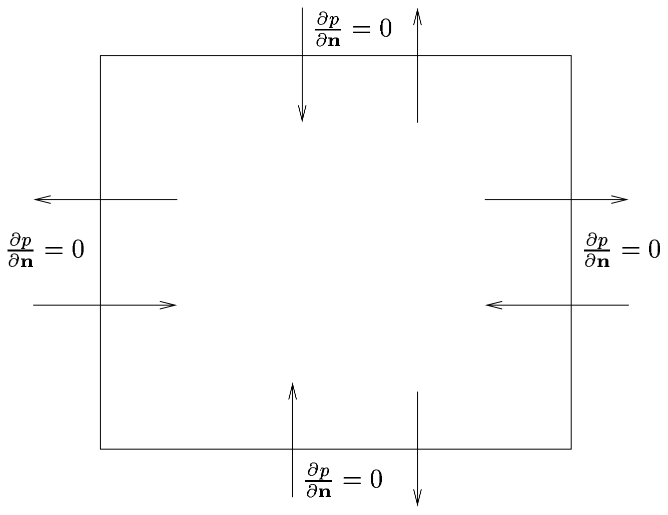

$\frac{\partial p}{\partial \mathbf{n}} = 0$
$\frac{\partial p}{\partial \mathbf{n}} = 0$
$\frac{\partial p}{\partial \mathbf{n}} = 0$
$\frac{\partial p}{\partial \mathbf{n}} = 0$

Figure 2.4: 2-d box with Neumann boundary conditions designating zero flux of the pressure across the boundary

Boundary values are necessary for numerical simulations of most PDEs, determining the values at interior points. Furthermore, from a mathematical standpoint, the use of boundary values has allowed several difficult problems to be proved as *well-posed.* That is, given a problem and its boundary values, is a solution, $\mathbf{u}(\mathbf{x}, t)$ unique and smooth for all time t [12]. In fact, a lot of work has gone into proving such results for the incompressible Navier-Stokes equations, most significantly in [29] for showing uniqueness of solutions in three dimensions for *finite periods* of time. In two dimensions, well-posedness results are more readily available for all time t (for examples of proofs, see [27]).

2.4 The Role of Pressure

In compressible flows, we stated that the pressure is a function of the density. However, it is often difficult to see how pressure is derived in incompressible flows since the density, ρ is constant. In fact, the pressure must be determined by the incompressibility constraint, $\nabla \cdot \mathbf{u} = 0$. Density and pressure again rely on each other; however, since ρ is constant, p must depend on $\nabla \cdot \mathbf{u}$. And, since $\nabla \cdot \mathbf{u}$ is one of our equations, it alone must determine the pressure, p [12]. Rather than seeing what determines the pressure, it is actually more informative to see what the pressure affects. To do so, we turn to using the *Helmholtz Decomposition.* This principle is seen in many forms, being called the Ladyzhenskaya Theorem [39], Helmholtz Principle, or Helmholtz-Hodge Decomposition Theorem [12]. It involves decomposing a vector field into its divergence-free and gradient components. This is be the basis of numerical solutions via projection methods in the next chapter, and it is often essential to the basic understanding of least squares approximation problems [45]. Here, we defer to [12] and refer to it as the Helmholtz-Hodge Decomposition Theorem (for proofs, see [22] and [12]):

Theorem 2.4.1 (Helmholtz-Hodge Decomposition) *Given a (2-d or 3-d) volume Ω with boundary, $\partial\Omega$ and outward normal, $\mathbf{n}$, let $\mathbf{v}$ be a vector function with $\mathbf{v} \in \Omega$. Then, $\mathbf{v}$ is uniquely decomposed as*

$$\mathbf{v} = \mathbf{u} + \nabla p.$$

with $\mathbf{u} \in \Omega$ is divergence-free, $\nabla \cdot \mathbf{u} = 0$ and $\mathbf{u} \cdot \mathbf{n} = 0$ on $\partial\Omega$.

To see how this is helpful, let Φ be an orthogonal projector which projects any vector onto its divergence-free part. In 2-d, this is graphically represented as in Figure 2.5 (an obvious extension to 3-D is seen in [12]).

If $\nabla \cdot \mathbf{u} = 0$, then

$$\Psi\mathbf{u} = \mathbf{u}.$$

And, for some vector, $\mathbf{v}$ with $\nabla \cdot \mathbf{v} \neq 0$, then

$$\mathbf{v} = \Psi\mathbf{v} + \nabla p.$$

Further, for scalar, a,

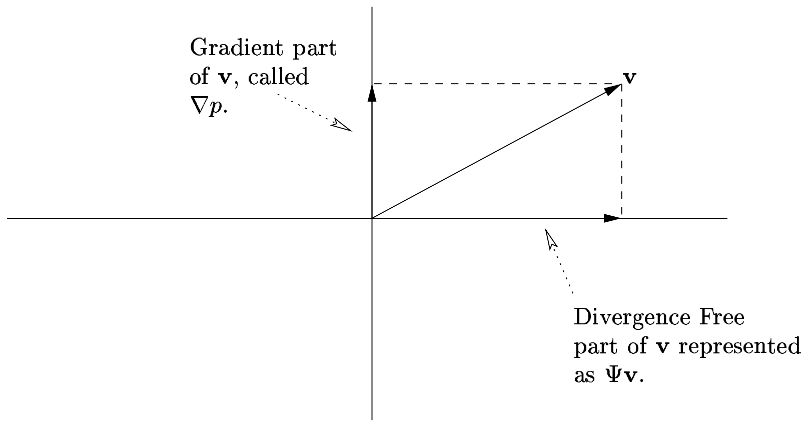

Gradient part of $\mathbf{v}$, called ∇p.
$\mathbf{v}$
Divergence Free part of $\mathbf{v}$ represented as $\Psi\mathbf{v}$.

Figure 2.5: Graphical example of using the Helmholtz-Hodge Decomposition in 2-d

$$\Psi(\nabla a) = 0.$$

If we apply Ψ to the incompressible Navier-Stokes equations (letting $\mathbf{F} = 0$ arbitrarily for simplicity), we get:

$$\Psi\left(\frac{\partial \mathbf{u}}{\partial t} + \nabla p - \nu\Delta\mathbf{u} + (\nabla \cdot \mathbf{u})\mathbf{u}\right) = 0.$$

However, from the principles of Ψ from above, we have, $\Phi(\frac{\partial \mathbf{u}}{\partial t}) = \frac{\partial \mathbf{u}}{\partial t}$ since $\nabla \cdot \mathbf{u} = 0$, and $\Psi(\nabla p) = 0$. Note that $\Psi(\Delta\mathbf{u})$ is not necessarily zero [12]. As a result, we see that

$$\frac{\partial \mathbf{u}}{\partial t} + \Psi(-\nu\Delta\mathbf{u} + (\nabla \cdot \mathbf{u})\mathbf{u}) = 0.$$

allows us to study $\mathbf{u}$ without knowing the pressure. Conversely, if we know $\mathbf{u}$, we can find p since it must merely be the gradient part of $\Psi(-\nu\Delta\mathbf{u} + (\nabla \cdot \mathbf{u})\mathbf{u})$.

By breaking up the velocity and pressure, we see how the pressure actually affects our solution. In fact, this theorem is the basis of many numerical solvers for the incompressible Navier-Stokes equations as seen in [43], [42], [9], [39], and we discuss how these and others are used in the next chapter.

Chapter 3

Numerical Solutions for the Navier-Stokes Equations

As solutions are difficult to find analytically, we seek numerical solutions to the incompressible Navier-Stokes equations by discretizing in time and space. In the following chapter we discuss how we do this, using the Chorin-Temam based projection methods. We confine the majority of our discussion to the 2D case; however, all of these results extend to 3D.

3.1 Introduction to the Numerical Problem and Breaking the NSE into Components

The standard problem studied in numerical linear algebra is how to solve the system

$$A\mathbf{u} = \mathbf{f}.$$

for $\mathbf{u}$ when given the matrix, A, and vector, $\mathbf{f}$. To solve for $\mathbf{u}$, one must invert A using simple numerical techniques such as such as L-U Decomposition, described in [13] and [45], or more complex ones such as the Multigrid solution method, [14]. There are many sources for studying such methods, and [13] provides many references.

In this chapter, we focus on how to construct A using time and space discretization methods to approximate our space and solve for $\mathbf{u}$, the velocity vector describing the solutions to the incompressible Navier-Stokes equations. The vector, $\mathbf{f}$, consists of all boundary conditions described by the vector, $\mathbf{b}$, and any outside forcing terms. For example, we expand the 2-dimensional incompressible NSE in equations 3.1 and 3.2 on the domain, Ω, boundary, $\partial\Omega$, and boundary conditions, $\mathbf{u}|_{\partial\Omega} = \mathbf{b}$. We want to solve 3.1 for x-component, u_x, and 3.2 for y-component, u_y.

$$\frac{\partial u_x}{\partial t} + u_x\frac{\partial u_x}{\partial x} + u_y\frac{\partial u_x}{\partial y} + \frac{\partial p}{\partial x} - \nu\left(\frac{\partial^2 u_x}{\partial x^2} + \frac{\partial^2 u_y}{\partial y^2}\right) = f_x + b_x. \tag{3.1}$$

$$\frac{\partial u_y}{\partial t} + u_y\frac{\partial u_y}{\partial y} + u_x\frac{\partial u_y}{\partial x} + \frac{\partial p}{\partial y} - \nu\left(\frac{\partial^2 u_x}{\partial x^2} + \frac{\partial^2 u_y}{\partial y^2}\right) = f_y + b_y. \tag{3.2}$$

The incompressibility constraint applies to both of our components and is seen as such in 3.3.

$$\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} = 0. \tag{3.3}$$

In the next section, we discuss time discretization techniques for the 2-dimensional incompressible NSE before we look at space-discretization techniques.

3.2 Time Discretization

Instead of solving one system of equations, we solve an evolving system of equations as we are interested in a solution, $\mathbf{u}$, which depends on position, $\mathbf{x}$, as well as time, t. We do this at the $n+1$st time-step, t^{n+1}, by solving the modified equation from before as

$$A\mathbf{u}^{n+1} = \mathbf{f}^{n+1}. \tag{3.4}$$

where we incorporate the previous time step's known solution, $\mathbf{u}^n$, into solving 3.4.

In order to evaluate the evolution of a PDE over time, we need a way of numerically approximating the problem. There are many schemes used for numerical approximations that are either explicit, implicit or semi-implicit. Some methods are more helpful in situations with linear systems and others with nonlinear systems, and here we assume that the reader has a basic awareness of such schemes (for example, backward Euler's, Newton-Raphson, Crank-Nicholson, etc.) such as described in [25] for understanding how time discretizations of PDEs are derived.

Developing a time-discretization of the Navier-Stokes equations can be difficult, mainly due to deciding how to handle the nonlinear convective term. Fortunately, there are many time-discretizations that have been developed and analyzed by others for these equations which also use some linearization techniques for the convective term [39], and we look at a few here.

Three strategies discussed by [39] are the *Euler-Newton*, *Euler semi-implicit* and *Crank-Nicholson/Adams-Bashforth* methods. The first uses backward Euler's method and linearizing with Newton's method, producing

$$\frac{1}{\Delta t}\{\mathbf{u}^{n+1} - \mathbf{u}^n\} + \nabla p^{n+1} - \nu\mathbf{u}^{n+1} + (\mathbf{u}^n \cdot \nabla)\mathbf{u}^{n+1} + (\mathbf{u}^{n+1} \cdot \nabla)\mathbf{u}^n - (\mathbf{u}^n \cdot \nabla)\mathbf{u}^n = \mathbf{f}^{n+1}. \tag{3.5}$$

The semi-implicit Euler method simply assumes that $(\mathbf{u}^n \cdot \nabla)\mathbf{u}^n$ and $(\mathbf{u}^{n+1} \cdot \nabla)\mathbf{u}^n$ approximate each other, so those two terms cancel out in 3.5, producing

$$\frac{1}{\Delta t}\{\mathbf{u}^{n+1} - \mathbf{u}^n\} + \nabla p^{n+1} - \nu\mathbf{u}^{n+1} + (\mathbf{u}^n \cdot \nabla)\mathbf{u}^{n+1} = \mathbf{f}^{n+1}.$$

The problem with the two methods above is that they involve $\mathbf{u}^{n+1}$ in the convective term. For our interests, this adds unnecessary complexity. Instead, it is simpler to use the trapezoidal rule, also known as Crank-Nicholson [25] for the linear part of the Navier-Stokes equations and use the Adams-Bashforth [25] method to approximate the convective term. Adams-Bashforth relies on the previous two time steps for $\mathbf{u}$ but not on $\mathbf{u}^{n+1}$. Additionally, as a result of Crank-Nicholson, we compute the pressure, p, and function, $\mathbf{f}$, at half-time steps. That is, we want $p^{n+\frac{1}{2}}$ and $\mathbf{f}^{n+\frac{1}{2}}$, which we compute from [25] that we have

$$p^{n+\frac{1}{2}} \approx \frac{1}{2}\{p^{n+1} + p^n\}.$$

$$\mathbf{f}^{n+\frac{1}{2}} \approx \frac{1}{2}\{\mathbf{f}^{n+1} + \mathbf{f}^n\}.$$

Computing $[(\mathbf{u}\cdot)\mathbf{u}]^{n+\frac{1}{2}}$ is more complicated due to linearization. However, the resulting time-discretization is seen in 3.6 [26].

$$\frac{1}{\Delta t}\{\mathbf{u}^{n+1}-\mathbf{u}^{n}\}-\frac{\nu}{2}\{\Delta\mathbf{u}^{n+1}+\Delta\mathbf{u}^{n}\}+\frac{1}{2}\{\nabla p^{n+1}+\nabla p^{n}\}+\frac{1}{2}\{3(\mathbf{u}^{n}\cdot\nabla)\mathbf{u}^{n}-(\mathbf{u}^{n-1}\cdot\nabla)\mathbf{u}^{n-1}\}=\frac{1}{2}\{\mathbf{f}^{n+1}+\mathbf{f}^{n}\}. \quad (3.6)$$

At time $n+1$, we know everything from the nth time-step as well as the value $\mathbf{f}^{n+1}$, so we simplify 3.6 by collecting everything we already know onto the right-hand side, add it up, and call it a new function, $\mathbf{F}$. Namely,

$$\mathbf{F}^{n-1,n,n+1}=\frac{1}{\Delta t}\{\mathbf{u}^{n}\}+\frac{1}{2}\{\mathbf{f}^{n+1}+\mathbf{f}^{n}\}-\frac{1}{2}\{3(\mathbf{u}^{n}\cdot\nabla)\mathbf{u}^{n}-(\mathbf{u}^{n-1}\cdot\nabla)\mathbf{u}^{n-1}\}-\frac{1}{2}\{\nabla p^{n}\}. \quad (3.7)$$

We rewrite 3.6 for our unknowns as

$$\frac{2}{\Delta t}\{\mathbf{u}^{n+1}\}-\nu\Delta\mathbf{u}^{n+1}+\nabla p^{n+1}=2\mathbf{F}^{n-1,n,n+1}. \quad (3.8)$$

Hence, the matrix, A, for 3.4 simply involves a discretization of the Laplacian, Δ, plus the terms for $\frac{2}{\Delta t}\{\mathbf{u}^{n+1}\}$ and ∇p^{n+1}. We will see that we may easily break this problem into several simpler problems, using the Helmholtz-Hodge Decomposition in Section 3.4,

It turns at that 3.8 represents a second-order (in time) method once we add our space-discretization in the next section and some other modifications. In fact, such modifications are plentiful, the most popular of which is seen in [4] and [26]. Additionally, [7] discusses the second-order accuracy of these methods and others. However, to start with, it may be good to look at an explicit method which also does not use $\mathbf{u}^{n+1}$ in the convective term. A method using explicit Euler methods for the convective term is seen as

$$\frac{1}{\Delta t}\{\mathbf{u}^{n+1}-\mathbf{u}^{n}\}-\nu\Delta\mathbf{u}^{n+1}+\nabla p^{n+1}+(\mathbf{u}^{n}\cdot\nabla)\mathbf{u}^{n}=\mathbf{f}^{n+1}. \quad (3.9)$$

This explicit method is first-order in time, and it is a simpler method to implement and study before adding the extra half-step required by 3.8. Further, as in 3.8, we combine all of the convective terms to the right-hand side, simplifying the work we have to do, and the first-order method is still second-order in space.

To complete the time-discretization of the Navier-Stokes equations, we must obviously discretize our incompressibility constraint in order to produce a well-posed and closed system:

$$\nabla\cdot\mathbf{u}^{n+1}=0.$$

Finally, we note that for the Stokes equations, the discretization is greatly simplified by not having to handle the nonlinear convective term, taking for example the semi-implicit Euler method form, 3.7, without this term as our time-discretized system.

3.3 Space Discretization

Here, we use a finite-difference approach in space-discretizing our terms from 3.8. The finite element can be used as discussed in [39]; however, we do not discuss it here.

In most of the literature, for the space-discretization method, a staggered grid is suggested in which pressures are calculated at cell centers and velocities at the edges for a 2-D solution (3-D extends this to voxel centers/edges) as in figure 3.1 for 2-D. This is often called the Marker-Cell grid [39] or simply MAC grid [48]. By sampling values at different points on a grid, we achieve higher accuracy, an analysis of which is available in [23]. Further, as we discuss later, it is important to have points outside of the boundary for calculating the Neumann boundary condition that appears later as a result of using the projection methods in Section 3.4.

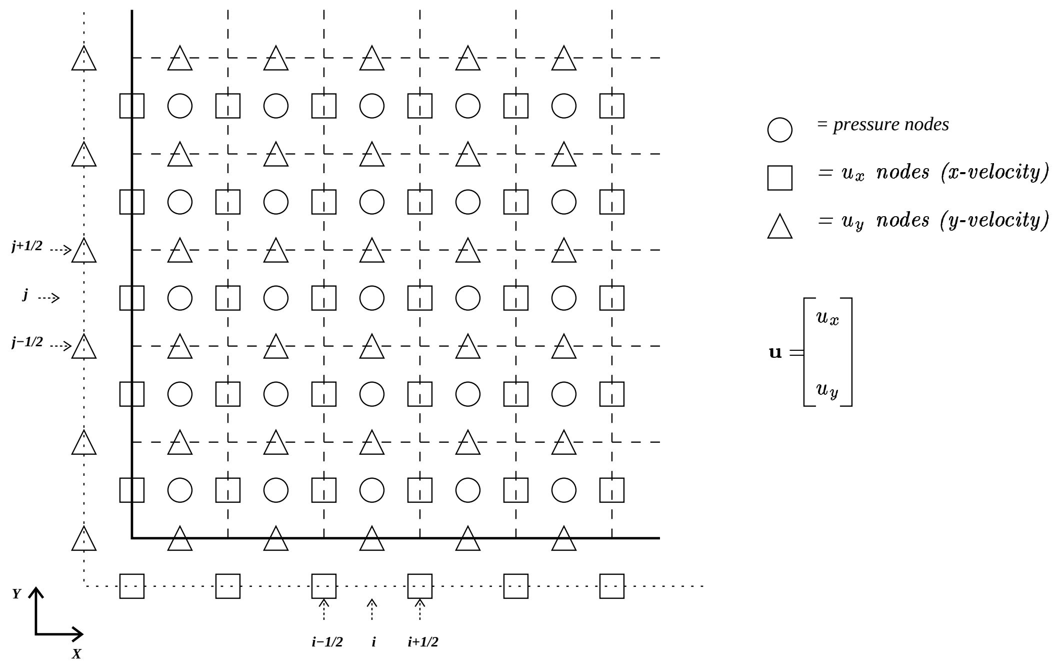

= pressure nodes
= u_x nodes (x-velocity)
= u_y nodes (y-velocity)
$\mathbf{u} = \begin{bmatrix} u_x \\ u_y \end{bmatrix}$
j+1/2
j
j−1/2
i−1/2
i
i+1/2
Y
X

Figure 3.1: The 2-D staggered MAC grid, showing how we label points for our space-discretization and boundary calculations

3.3.1 Space Discretizing the Laplacian

To calculate $\Delta\mathbf{u}$, we use a second-order five-point stencil scheme as detailed in [44] and [25]. Let u_x be the x-component and u_y be the y-component of $\mathbf{u}$. Then,

$$\Delta\mathbf{u} = \begin{bmatrix} \Delta u_x \\ \Delta u_y \end{bmatrix}.$$

To begin, we must calculate the first partial derivatives and then the second. Without loss of generality, we begin with u_x and do a first-order finite-difference approximation in the x-direction of the first partial derivative at grid-points (i,j)in 3.10 and $(i+1,j)$ in 3.11.

$$\frac{\partial u_x}{\partial x}|_{x=i\cdot\Delta x, y=j\cdot\Delta y} \approx \frac{u_{x_{i+\frac{1}{2},j}} - u_{x_{i-\frac{1}{2},j}}}{\Delta x}. \tag{3.10}$$

$$\frac{\partial u_x}{\partial x}|_{x=(i+1)\cdot\Delta x, y=j\cdot\Delta y} \approx \frac{u_{x_{i+\frac{3}{2},j}} - u_{x_{i+\frac{1}{2},j}}}{\Delta x}. \tag{3.11}$$

Combining, we get an approximation for the second partial derivative with respect to x (taking the negative) and grid-point $(i+\frac{1}{2},j)$.

$$-\frac{\partial^2 u_x}{\partial x^2}|_{x=(i+\frac{1}{2}\cdot\Delta x), y=j\cdot\Delta y} \approx \frac{-u_{x_{i+\frac{3}{2},j}} + 2u_{x_{i+\frac{1}{2},j}} - u_{x_{i-\frac{1}{2},j}}}{\Delta x^2}. \tag{3.12}$$

Doing the same for the second partial derivative in the y-direction, we get the second partial derivative with respect to y at grid-point, $(i+\frac{1}{2},j)$ as

$$-\frac{\partial^2 u_x}{\partial y^2}|_{x=(i+\frac{1}{2}\cdot\Delta x), y=j\cdot\Delta y} \approx \frac{-u_{x_{i+\frac{1}{2},j+1}} + 2u_{x_{i+\frac{1}{2},j}} - u_{x_{i+\frac{1}{2},j-1}}}{\Delta y^2}. \tag{3.13}$$

Combining 3.12 and 3.13, we get our second-order approximation to Δu_x at the $(i+\frac{1}{2},j)$th point on our staggered grid.

$$-\Delta u_{x_{i+\frac{1}{2},j}} \approx \frac{-u_{x_{i+\frac{3}{2},j}} + 2u_{x_{i+\frac{1}{2},j}} - u_{x_{i-\frac{1}{2},j}}}{\Delta x^2} + \frac{-u_{x_{i+\frac{1}{2},j+1}} + 2u_{x_{i+\frac{1}{2},j}} - u_{x_{i+\frac{1}{2},j-1}}}{\Delta y^2}. \tag{3.14}$$

This is often seen in forms where $\Delta x = \Delta y = h$ on an evenly distributed grid, in which case 3.14 becomes

$$-\Delta u_{x_{i+\frac{1}{2},j}} \approx \frac{4u_{x_{i+\frac{1}{2},j}} - u_{x_{i+\frac{3}{2},j}} - u_{x_{i-\frac{1}{2},j}} - u_{x_{i+\frac{1}{2},j+1}} - u_{x_{i+\frac{1}{2},j-1}}}{h^2}. \tag{3.15}$$

Similarly, for u_y, we analyze Δu_y at $(i,j+\frac{1}{2})$

$$-\Delta u_{y_{i,j+\frac{1}{2}}} \approx \frac{-u_{y_{i-1,j+\frac{1}{2}}} + 2u_{y_{i,j+\frac{1}{2}}} - u_{y_{i+1,j+\frac{1}{2}}}}{\Delta x^2} + \frac{-u_{y_{i,j+\frac{3}{2}}} + 2u_{y_{i,j+\frac{1}{2}}} - u_{y_{i,j-\frac{1}{2}}}}{\Delta y^2}. \tag{3.16}$$

It is immediately apparent why this is called a five-point stencil finite-difference scheme. Further, this approximation for $\Delta\mathbf{u}$ is second-order [13], and while higher-order nine-point schemes are available [25], the extra computation would be a wasted effort since our other approximations of the space would negate any benefit in the development of an overall first or second order incompressible Navier-Stokes solver. As a final note, in our use of the projection methods in Section 3.4, we will also need to compute the application of the Laplacian to the scalar pressure field, p which is easily computed, using our second-order five point difference scheme, at the (i,j)th grid-point in 2-D as

$$\Delta p_{i,j} \approx \frac{p_{i-1,j} - 2p_{i,j} + p_{i+1,j}}{\Delta x^2} + \frac{p_{i,j+1} - 2p_{i,j} + p_{i,j-1}}{\Delta y^2}. \tag{3.17}$$

3.3.2 Space Discretizing the Convective Term

We turn our attention to discretizing the convective term, $(\mathbf{u}\cdot\nabla)\mathbf{u}$ in space. In fact, from the Appendix, we represent this in matrix form for 2-D as

$$(\mathbf{u}\cdot\nabla)\mathbf{u} = \begin{bmatrix} u_x\frac{\partial u_x}{\partial x} + u_y\frac{\partial u_x}{\partial y} \\ u_y\frac{\partial u_y}{\partial y} + u_x\frac{\partial u_y}{\partial x} \end{bmatrix}. \tag{3.18}$$

We focus on the first component of this equation. We compute $\frac{\partial u_x}{\partial x}$ and $\frac{\partial u_x}{\partial y}$ as we did before, but if we want to compute these at the point $u_{x_{i+\frac{1}{2},j}}$ we have to increase spatial step-size. That is, we obtain a second-order approximation by computing these partial derivatives as

$$\frac{\partial u_{x_{i+\frac{1}{2},j}}}{\partial x} \approx \frac{u_{x_{i+\frac{3}{2},j}} - u_{x_{i-\frac{1}{2},j}}}{2\Delta x}. \tag{3.19}$$

$$\frac{\partial u_{x_{i+\frac{1}{2},j}}}{\partial y} \approx \frac{u_{x_{i+\frac{1}{2},j+1}} - u_{x_{i+\frac{1}{2},j-1}}}{2\Delta y}. \tag{3.20}$$

For the first component of 3.18 we obtain

$$\left(u_x\frac{\partial u_x}{\partial x} + u_y\frac{\partial u_x}{\partial y}\right)_{i+\frac{1}{2},j} \approx u_{x_{i+\frac{1}{2},j}}\frac{u_{x_{i+\frac{3}{2},j}} - u_{x_{i-\frac{1}{2},j}}}{2\Delta x} + u_{y_{i+\frac{1}{2},j}}\frac{u_{x_{i+\frac{1}{2},j+1}} - u_{x_{i+\frac{1}{2},j-1}}}{2\Delta y}. \tag{3.21}$$

From figure 3.1, we have no value for $u_{y_{i+\frac{1}{2},j}}$ as needed in 3.21. However, we can obtain a second-order approximation by taking the average of the values of u_y which surround the, $(i+\frac{1}{2},j)$th grid point by computing the following

$$u_{y_{i+\frac{1}{2},j}} \approx \frac{1}{4}\left(u_{y_{i,j+\frac{1}{2}}} + u_{y_{i+1,j+\frac{1}{2}}} + u_{y_{i,j-\frac{1}{2}}} + u_{y_{i+1,j-\frac{1}{2}}}\right). \tag{3.22}$$

We combine 3.21 and 3.22 to compute the first component of the convective term. Additionally, we can write down a similar approximation for the second term of 3.18 at the $(i,j+\frac{1}{2})$th grid point as

$$\left(u_y\frac{\partial u_y}{\partial y} + u_x\frac{\partial u_y}{\partial x}\right)_{i,j+\frac{1}{2}} \approx u_{x_{i,j+\frac{1}{2}}}\frac{u_{y_{i+1,j+\frac{1}{2}}} - u_{y_{i-1,j+\frac{1}{2}}}}{2\Delta x} + u_{y_{i,j+\frac{1}{2}}}\frac{u_{y_{i,j+\frac{3}{2}}} - u_{y_{i,j-\frac{1}{2}}}}{2\Delta y}. \tag{3.23}$$

where

$$u_{x_{i,j+\frac{1}{2}}} \approx \frac{1}{4}\left(u_{x_{i-\frac{1}{2},j+1}} + u_{x_{i+\frac{1}{2},j+1}} + u_{x_{i+\frac{1}{2},j}} + u_{x_{i-\frac{1}{2},j}}\right). \tag{3.24}$$

Knowing what we do from last section, if we already have the solution at time n as $\mathbf{u}^n$, using 3.18 and 3.21-3.24, we are able to compute $(\mathbf{u}^n\cdot\nabla)\mathbf{u}^n$, our convective term.

3.3.3 Space Discretizing the Gradient

It will turn out that we need to compute the gradient of the pressure field in the use of the projection methods below in Section 3.4. Using the Appendix as a reference, it is straightforward to calculate a second-order approximation to the gradient of our scalar pressure field at the (i, j)th point. However, as will soon become more clear, we want to apply the first component (x-direction) of ∇p to u_x and the second component (y-direction) to u_y. Therefore, we must compute $\frac{\partial p}{\partial x}$ at the $(i+\frac{1}{2}, j)$th grid point, and $\frac{\partial p}{\partial y}$ at the, $(i, j+\frac{1}{2})$th grid point. By looking at figure 3.1, we quickly realize this can be done as shown in 3.25 and 3.26.

$$\frac{\partial p}{\partial x}_{i+\frac{1}{2},j} \approx \frac{p_{i+1,j} - p_{i,j}}{\Delta x}. \tag{3.25}$$

$$\frac{\partial p}{\partial y}_{i,j+\frac{1}{2}} \approx \frac{p_{i,j+1} - p_{i,j}}{\Delta y}. \tag{3.26}$$

Hence, we never really compute $(\nabla p)_{i,j}$, but only at the half-steps. Again, the reason for this will become more clear as we discuss the projection methods.

3.3.4 Space Discretizing the Divergence

Calculating the divergence, $\nabla\cdot$, is straightforward as described in the Appendix. We will need to take the divergence of the vector field, $\mathbf{u}$, in Section 3.4 for the projection methods. Further, we need to do so and apply the results to the scalar field, p at the (i, j)th grid point. We are more interested in obtaining the updated $p_{i,j}$ value which is why we do not calculate the pressure values at the half-step grid points and simply calculate the divergence of the vector field there. The divergence operator takes our vector field and produce a scalar one, so doing so at the (i, j)th grid-point is more sensible, maintaining consistent grid structure. By now it should be clear how we want to calculate the approximate values, so we continue straight to equation 3.27, which shows how we make this second-order approximation.

$$(\nabla \cdot \mathbf{u})_{i,j} \approx \frac{u_{x_{i+\frac{1}{2},j}} - u_{x_{i-\frac{1}{2},j}}}{\Delta x} + \frac{u_{y_{i,j+\frac{1}{2}}} - u_{y_{i,j-\frac{1}{2}}}}{\Delta y}. \tag{3.27}$$

3.3.5 Computing the Neumann Boundary Conditions for the Pressure

As was stated in Chapter 2, we have to compute the Neumann boundary conditions for the pressure. These usually consist of $\frac{\partial p}{\partial \mathbf{n}}$ being equal to either zero or some other value. Two possible forms that are seen are $\frac{\partial p}{\partial \mathbf{n}} = 0$ and $\frac{\partial p}{\partial \mathbf{n}} = \nu(\Delta \mathbf{u}) \cdot \mathbf{n}$. The second condition is obviously more complicated, but with the help of figure 3.2, we show how such an approximation can be calculated.

In order to compute $\frac{\partial p}{\partial \mathbf{n}}$ across our boundary, a pressure point must be evaluated outside of the current pressure boundary in figure 3.1. That is, we compute our partial derivative across the boundary in 3.28 for computing the Neumann condition across the left-hand 'x'-boundary as in figure 3.2.

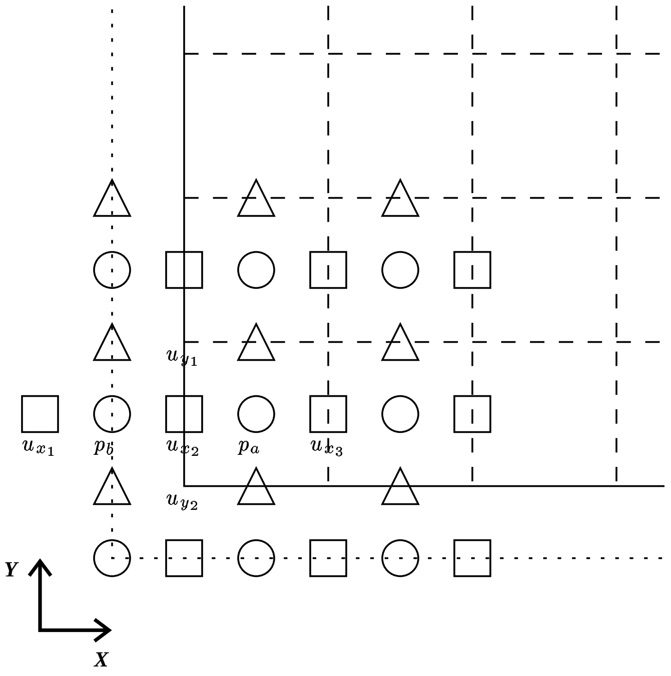


Figure 3.2: Labeled mini-grid used for explaining computations necessary for the Neumann boundary conditions

$$\frac{\partial p}{\partial x} = \frac{p_b - p_a}{\Delta x}. \tag{3.28}$$

We omit the equivalent computation for the Neumann boundary condition, $\frac{\partial p}{\partial y}$. Therefore, by extending the physical boundary for the pressure to a half-step outside of $\partial\Omega$, we compute $\frac{\partial p}{\partial \mathbf{n}} = 0$ as needed for the projection methods.

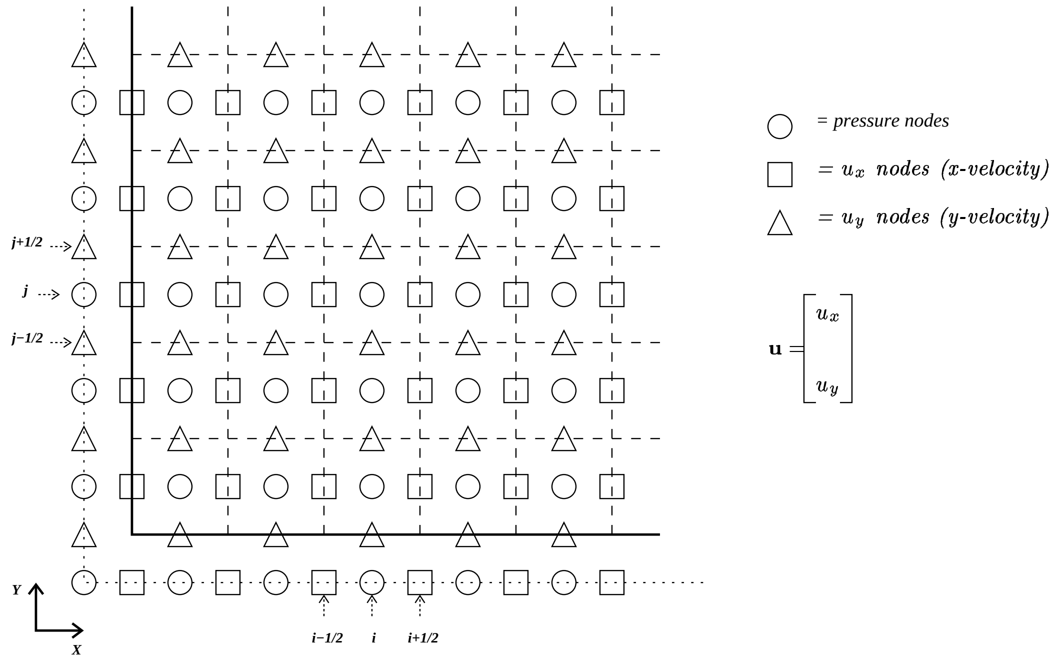
As stated, it is often necessary for some of the projection methods to compute $\frac{\partial p}{\partial \mathbf{n}} = \nu(\Delta \mathbf{u}) \cdot \mathbf{n}$. Normally, in order to compute $\nu(\Delta \mathbf{u}) \cdot \mathbf{n}$ across $\partial\Omega$, we would need an additional values for u_x and u_y outside of our new pressure boundary values. We have illustrated this in figure 3.2 for u_x with the point u_{x_1}. Using u_{x_1}, u_{x_2} and u_{x_3}, again from figure 3.2, 3.29 shows how we compute $\frac{\partial p}{\partial x} = \nu \frac{\partial^2 u_x}{\partial x}$.

$$\frac{p_b - p_a}{\Delta x} = \nu \frac{u_{x_1} + u_{x_3} - 2u_{x_2}}{(\Delta x)^2}. \tag{3.29}$$

The equivalent derivation in the y-direction would introduce new u_y values outside of our boundary; however, we want to introduce as few new points as possible since this degrades the value of our physical boundary conditions. So, we note that our incompressibility constraint, $\nabla \cdot \mathbf{u} = 0$, implies that at the point, u_{x_2}, we have 3.30.

$$\frac{u_{y_1} - u_{y_2}}{\Delta y} + \left(\frac{u_{x_3} + u_{x_2}}{2\Delta x} - \frac{u_{x_1} + u_{x_2}}{2\Delta x} \right) = 0. \tag{3.30}$$

This equation quickly simplifies to 3.31.



= pressure nodes
$= u_x$ nodes (x-velocity)
$= u_y$ nodes (y-velocity)
$\mathbf{u} = \begin{bmatrix} u_x \\ u_y \end{bmatrix}$
j+1/2
j
j−1/2
i−1/2
i
i+1/2
Y
X

Figure 3.3: The new altered 2-D staggered MAC grid, showing how we label points for our space-discretization and boundary calculations with new pressure boundary points one step further out for computation of the Neumann boundary conditions

$$u_{x_3} = \frac{\Delta x}{\Delta y}\left[u_{x_3} + 2\left(u_{y_2} - u_{y_1}\right)\right]. \tag{3.31}$$

The virtual points, u_{y_1} and u_{y_2}, from figure 3.2 still have to be computed; however, these points are easily computed as the average of their neighbors to project values for u_y at these non-grid locations.

Using 3.31, we rewrite 3.29 as 3.32.

$$\frac{p_b - p_a}{\Delta x} = \nu \frac{u_{x_1} - 2u_{x_2} + \frac{\Delta x}{\Delta y}\left[u_{x_3} + 2\left(u_{y_2} - u_{y_1}\right)\right]}{(\Delta x)^2}. \tag{3.32}$$

Computing $\frac{\partial p}{\partial \mathbf{n}}$ also involves computing $\frac{\partial p}{\partial y}$, but as this derivation is similar to computing $\frac{\partial p}{\partial x}$ as above, and we do not use it directly here, it is omitted. It is important to realize, however, that this form of the Neumann condition often appears in higher-order projection methods [4], [7].

Finally, we note that all of our space-discretization methods are clearly second-order, so we know that if our NSE numerical method is first-order in time, we are still able to check the space convergence by halving the time steps as is done at the end of this chapter for our experiments. Knowing how to calculate most of the necessary components as well as providing a glimpse of some of the components necessary for the projection methods, we are prepared begin our discussion of these methods in the following section.

3.4 Projection Methods

One of the more popular solution methods for the incompressible Navier-Stokes equations is the projection method, sometimes referred to as the Chorin-Temam scheme because it was concurrently, yet independently, published by Chorin ([9], [10] and collected in [11]) and Temam ([42] and [43]). Since that time, hundreds of papers have relied on this scheme and dozens of researchers have investigated its accuracy as well as updated the method to higher orders, creating a vast array of projection methods from which to choose [39], [4], [7], [26], [28], [32].

The projection method is based on the Helmholtz-Hodge Decomposition, presented in Section 2.4. The idea is that we apply an orthogonal projector, P, which maps a vector field onto its divergence-free part, to our incompressible Navier-Stokes equations such that $P(\nabla p) = 0$. So, we project away the pressure, and then in subsequent steps we recover it. In fact,we use the projector to compute an intermediate velocity, $\mathbf{u}^*$, and then uses this to compute our pressure and final solution, $\mathbf{u}$.

3.4.1 First-Order (in Time) Projection Method for the NSE

For the explicit time-discretization in 3.9, we assume that we know $\mathbf{u}^n$, and we first want to solve for $\mathbf{u}^{*n+1}$, an intermediate velocity, in 3.33.

$$\frac{\mathbf{u}^{*n+1}}{\Delta t} - \nu \Delta \mathbf{u}^{*n+1} = \mathbf{f}^{n+1} - (\mathbf{u}^n \cdot \nabla)\mathbf{u}^n + \frac{\mathbf{u}^{*n}}{\Delta t}. \tag{3.33}$$

where we have the boundary condition that $\mathbf{u}*^{n+1} = \mathbf{0}$ on $\partial\Omega$. Further, we could treat the convective term differently for the implicit Euler method by computing $(\mathbf{u}^{n+1} \cdot \nabla)\mathbf{u}^{n+1}$ and for the semi-implicit Euler method by computing $(\mathbf{u}^n \cdot \nabla)\mathbf{u}^{n+1}$ [39]. However, for simplicity in computing the matrix, A and the right-hand side of 3.33, we use the explicit Euler formulation here.

We wish to recover the new pressure and solve for $\mathbf{u}^{n+1}$ by computing

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^{*n+1}}{\Delta t} + \nabla p^{n+1} = \mathbf{0}. \tag{3.34}$$

$$\nabla \cdot \mathbf{u}^{n+1} = 0. \tag{3.35}$$

However, we have the Neumann boundary condition that $\mathbf{u}^{n+1} \cdot \mathbf{n} = 0$ on $\partial\Omega$ for surface normal, $\mathbf{n}$. So, by taking the divergence of 3.34 and 3.35, we simplify 3.34 to the form of 3.36 [39].

$$\Delta p^{n+1} = \frac{\nabla \cdot \mathbf{u}^{*n+1}}{\Delta t}. \tag{3.36}$$

where we now have the Neumann boundary condition, $\frac{\partial p^{n+1}}{\partial \mathbf{n}} = 0$ on $\partial\Omega$, as seen in section 3.3.2. We note that this new Neumann boundary condition is born out of the result of applying the Helmholtz-Hodge decomposition [48], [39].

While 3.36 gives us a way of computing the updated pressure values, we need to find the actual solution $\mathbf{u}^{n+1}$. This is easily done in 3.37 by updating our intermediate solution, $\mathbf{u}^{*n+1}$ with p^{n+1}.

$$\mathbf{u}^{n+1} = \mathbf{u}^{*n+1} - \Delta t \nabla p^{n+1}. \tag{3.37}$$

Looking at these equations, it is quickly apparent why we ran through the the space-discretizations of the gradient and divergence functions as we did in section 3.3.2. That is, in 3.36 we computed the divergence of our intermediate solution for the purposes of finding our updated pressure solution, where the results of $\nabla \cdot \mathbf{u}^{*n+1}$ had to be computed at pressure locations in figure 3.1. Further, 3.37 shows how we need to compute the gradient of the pressure at the velocity locations, as stated in section 3.3.2.

3.4.2 Second-Order (in Time) Projection Methods for the NSE

While we will discuss the implementation of a first-order projection method, it is worth noting that there are several second-order projection methods that have emerged in the last two decades ([4], [28], [7], [26]). We will briefly describe the Van Kan scheme [26], based on the discussion in [39], as an introduction to these second-order methods. We start with the time-discretized version of the incompressible Navier-Stokes from 3.6, found via the Crank-Nicholson and Adams-Bashforth schemes. Then, we follow the same guideline in applying an orthogonal projector to our equations as we did for the first-order method above. Aside from our boundary conditions, we also assume that we have an initial velocity, $\mathbf{u}^0$, and we generate an arbitrary initial pressure, p^0. Since we need the nth and $(n+1)$th values for 3.6, we also use a first-order approximation to build an initial estimate for $\mathbf{u}^1$, for which we solve in a more highly discretized space for greater accuracy, and then project back onto our actual space. This increases the computation for only one step. We assume that for all $n > 1$, we have previously solved for $\mathbf{u}^n$, $\mathbf{u}^{n-1}$ and p^n. As such, we use 3.38 to solve for an intermediate velocity, $\mathbf{u}^{*n+1}$.

$$\frac{2\mathbf{u}^{*n+1}}{\Delta t} - \nu \Delta \mathbf{u}^{*n+1} = \frac{2\mathbf{u}^n}{\Delta t} + \mathbf{f}^{n+1} + \mathbf{f}^n + \nu \Delta \mathbf{u}^n + (\mathbf{u}^{n-1} \cdot \nabla)\mathbf{u}^{n-1} - 3(\mathbf{u}^n \cdot \nabla)\mathbf{u}^n. \tag{3.38}$$

where we set $\mathbf{u}^{*n+1} = \mathbf{0}$ on $\partial\Omega$. As before, we use the intermediate velocity, $\mathbf{u}^{*n+1}$, to solve for $\mathbf{u}^{n+1}$ and p^{n+1} in 3.39-3.40.

$$\frac{2\left(\mathbf{u}^{n+1} - \mathbf{u}^{*n+1}\right)}{\Delta t} + \nabla\left(p^{n+1} - p^n\right) = \mathbf{0}. \tag{3.39}$$

$$\nabla \cdot \mathbf{u}^{n+1} = 0. \tag{3.40}$$

with Neumann boundary condition, $\mathbf{u}^{n+1} \cdot \mathbf{n} = 0$ on $\partial\Omega$. Equivalently, we write 3.39-3.40 with boundary condition into the form of 3.41 [39] to solve for p^{n+1}.

$$\Delta p^{n+1} = \Delta p^n + \frac{2\nabla \cdot \mathbf{u}^{*n+1}}{\Delta t}. \tag{3.41}$$

with Neumann boundary condition, $(\nabla p^{n+1} \cdot \mathbf{n} - \nabla p^n \cdot \mathbf{n}) = 0$ on $\partial\Omega$. Finally, we solve for $\mathbf{u}^{n+1}$ with our intermediate velocity and updated pressure in 3.42.

$$\mathbf{u}^{n+1} = \mathbf{u}^{*n+1} - \frac{\Delta t(\nabla p^{n+1} - \nabla p^n)}{2}. \tag{3.42}$$

Discussions on the second-order accuracy of this projection method are found in [26] and [4].

3.5 Implementation Example

We implement a small example to show how the projection method process works. For our example, we choose to implement the explicit first-order projection method. Additionally, we choose 3.43 and 3.44 as our values for $\mathbf{u}$ and p as the boundary values on which to test the projection method for time t.

$$\mathbf{u} = e^{-\frac{2t}{Re}} \begin{bmatrix} -\cos x \sin y \\ \sin x \cos y \end{bmatrix}. \tag{3.43}$$

$$p = \frac{1}{4} e^{-\frac{4t}{Re}} \{\cos(2x) + \sin(2x)\}. \tag{3.44}$$

Further, we start with boundary conditions 3.45.

$$\mathbf{u} = \begin{bmatrix} -\cos x \sin y \\ \sin x \cos y \end{bmatrix}. \tag{3.45}$$

For this problem, we also set $\mathbf{f}^n = 0$ for all $n \geq 0$, which makes the setup discussion simpler in the next subsection. Further, for this problem, if we plug 3.43 and 3.44 into the incompressible Navier-Stokes equations, we find that for all time, t, the boundary conditions are actual solutions to the problem on our whole domain Ω. This is easily verified, and we do not show it here; however, this makes it a perfect test problem since we will be able to check to see if our solution is valid (This problem is a slight variation on the one found in [10]).

3.5.1 Detailed Setup Issues

We solve this problem on the domain $[0, \pi]^2$ and our Reynolds' number, $Re = 100$. The results below performed similarly for larger Reynolds' numbers up to 10000 and smaller Reynold's numbers down to 10. Studying larger numbers was not fruitful since, as discussed, the convective term becomes negligible. For smaller numbers, the pressure solution becomes unstable.

Since we are on a square domain, and for simplicity, we set $\Delta x = \Delta y = h$. Therefore, we rewrite 3.33 with space discretization as 3.46 for $u_x^{*(n+1)}$ and 3.47 for $u_y^{*(n+1)}$ with appropriate boundary conditions where we compute $u_{x_{i,j+\frac{1}{2}}}$ and $u_{y_{i+\frac{1}{2},j}}$ as described earlier in section 3.3.

$$\begin{gathered}\left(4 + \frac{h^2}{\Delta t}\right) u_{x_{i+\frac{1}{2},j}}^{*(n+1)} - \left\{ u_{x_{i+\frac{3}{2},j}}^{*(n+1)} + u_{x_{i-\frac{1}{2},j}}^{*(n+1)} + u_{x_{i+\frac{1}{2},j+1}}^{*(n+1)} + u_{x_{i+\frac{1}{2},j-1}}^{*(n+1)} \right\} = \\ \frac{h^2}{\Delta t} u_{x_{i+\frac{1}{2},j}}^{(n)} - \frac{h}{2} \left\{ u_{x_{i+\frac{1}{2},j}}^{(n)} \left(u_{x_{i+\frac{3}{2},j}}^{(n)} - u_{x_{i-\frac{1}{2},j}}^{(n)} \right) + u_{y_{i+\frac{1}{2},j}}^{(n)} \left(u_{x_{i+\frac{1}{2},j+1}}^{(n)} - u_{x_{i+\frac{1}{2},j-1}}^{(n)} \right) \right\}.\end{gathered} \tag{3.46}$$

$$\begin{gathered}\left(4 + \frac{h^2}{\Delta t}\right) u_{y_{i,j+\frac{1}{2}}}^{*(n+1)} - \left\{ u_{y_{i,j+\frac{3}{2}}}^{*(n+1)} + u_{y_{i,j-\frac{1}{2}}}^{*(n+1)} + u_{y_{i-1,j+\frac{1}{2}}}^{*(n+1)} + u_{y_{i+1,j+\frac{1}{2}}}^{*(n+1)} \right\} = \\ \frac{h^2}{\Delta t} u_{y_{i,j+\frac{1}{2}}}^{(n)} - \frac{h}{2} \left\{ u_{x_{i,j+\frac{1}{2}}}^{(n)} \left(u_{y_{i+1,j+\frac{1}{2}}}^{(n)} - u_{y_{i-1,j+\frac{1}{2}}}^{(n)} \right) + u_{y_{i,j+\frac{1}{2}}}^{(n)} \left(u_{y_{i,j+\frac{3}{2}}}^{(n)} - u_{x_{i,j-\frac{1}{2}}}^{(n)} \right) \right\}.\end{gathered} \tag{3.47}$$

Then, we compute the pressure, p^{n+1} from 3.36 using the space discretization methods in 3.17 and 3.27 to produce 3.48.

$$p^{*(n+1)}_{i-1,j} + p^{*(n+1)}_{i+1,j} + p^{*(n+1)}_{i,j+1} + p^{*(n+1)}_{i,j-1} - 4p^{*(n+1)}_{i,j} = \frac{h}{\Delta t}\left(u^{*(n+1)}_{x_{i+\frac{1}{2},j}} - u^{*(n+1)}_{x_{i-\frac{1}{2},j}} + u^{*(n+1)}_{y_{i,j+\frac{1}{2}}} - u^{*(n+1)}_{y_{i,j-\frac{1}{2}}}\right). \tag{3.48}$$

Finally, space-discretizing 3.37, using 3.25 and 3.26, we compute $\mathbf{u}^{n+1}$ by computing $\mathbf{u}_x^{n+1}$ in 3.49 and $\mathbf{u}_y^{n+1}$ in 3.50

$$u^{n+1}_{x_{i+\frac{1}{2},j}} = u^{*(n+1)}_{x_{i+\frac{1}{2},j}} - \frac{\Delta t}{h}\left(p^{*(n+1)}_{i+1,j} - p^{*(n+1)}_{i,j}\right). \tag{3.49}$$

$$u^{n+1}_{x_{i,j+\frac{1}{2}}} = u^{*(n+1)}_{x_{i,j+\frac{1}{2}}} - \frac{\Delta t}{h}\left(p^{*(n+1)}_{i,j+1} - p^{*(n+1)}_{i,j}\right). \tag{3.50}$$

Since we are less concerned with algorithmic running time, we use an inefficient yet accurate $O(n^3)$ L-U solver with partial pivoting [13].

3.5.2 Testing our Software and Results

Before we display pictures of solutions from our problem, we start by verifying that this method is indeed first-order in time. In figure 3.4, we fix our grid size at 32 by 32 ($h = \frac{\pi}{32} \approx 0.09817$) and run tests for $t = \frac{1}{10}$ to $t = \frac{1}{320}$. For smaller time-discretizations, it is necessary to increase the grid size. Further, in order to test the convergence, we pick a single point at which to calculate the norm of the difference of the actual solution to our numerical solution.

We see that as we halve our time-steps, we indeed see first-order convergence, as anticipated. In the following pages, we show example solutions for this problem at different space and time steps.

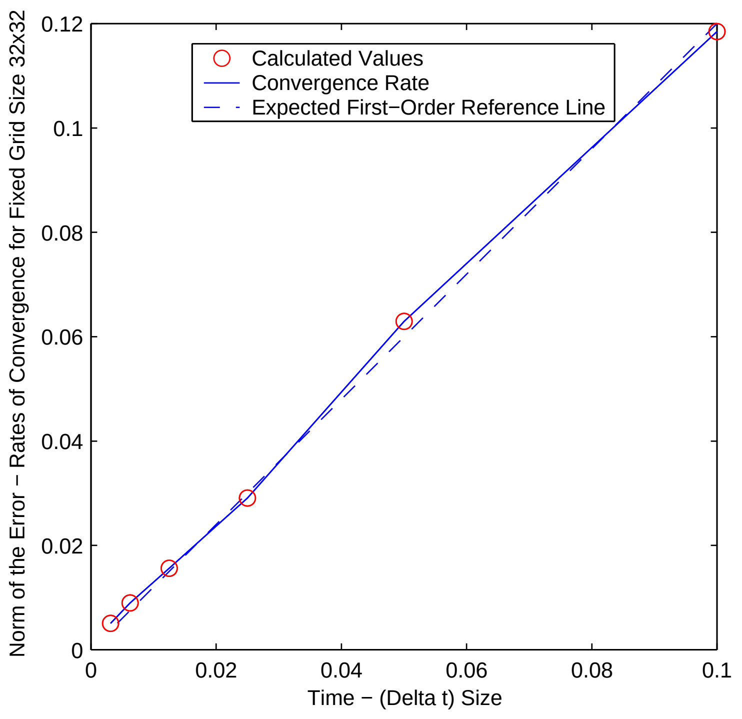


Figure 3.4: Plot of the errors of our numerical solver for changing time step and fixed grid size, verifying first-order (in time) convergence of the explicit Euler method for solver the incompressible NSE

Our attention is now turned to measuring the order of convergence for space. In order to test this on a time-dependent problem with our first-order (in time) method, we have to decrease our time step by $\frac{1}{4}$ every time we halve the size of h. We start by setting $h = \frac{\pi}{8}$ (an 8-by-8 grid) and testing it up to $h = \frac{\pi}{128}$ (a 128-by-128 grid) and $\Delta t = \frac{1}{10}$ to $\frac{1}{2560}$. Since the solutions to u_x and u_y are similar, we study u_x solely for order of convergence in space. Additionally, since the pressure, p, is part of the solution of u_x, we do not look at how it converges in detail, except to note again that for small viscosities, the pressure solution behaves poorly, and hence, $\mathbf{u}$, becomes unstable.

Quickly, we show some of our output for solutions to u_x in 3.5-3.9 where 3.9 is the sole color figure for easier viewing.

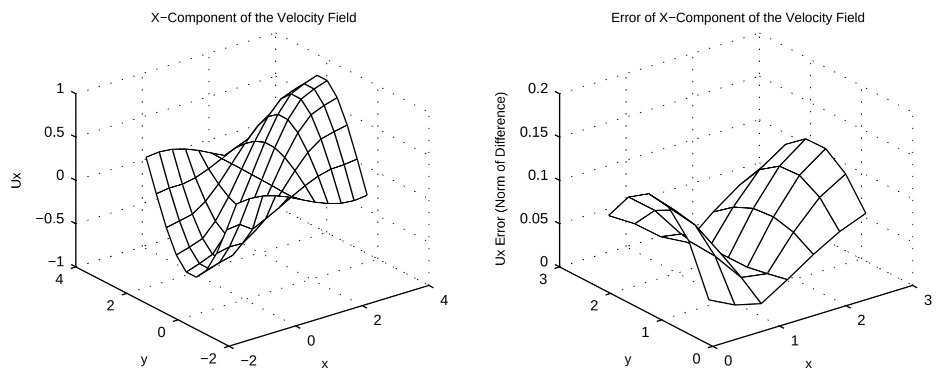


Figure 3.5: Solution for u_x, the x-component of the velocity on an 8-by-8 grid ($h = \frac{\pi}{8}$) along with the norm of the difference between our numerical solution and the actual solution

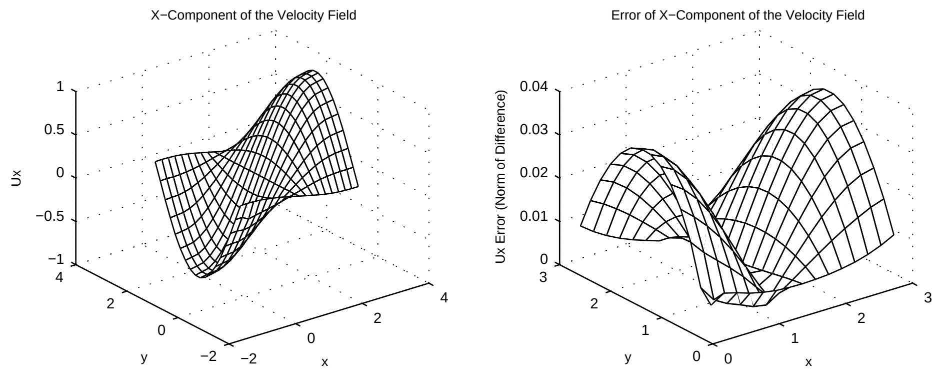


Figure 3.6: Solution for u_x, the x-component of the velocity on an 16-by-16 grid ($h = \frac{\pi}{16}$) along with the norm of the difference between our numerical solution and the actual solution

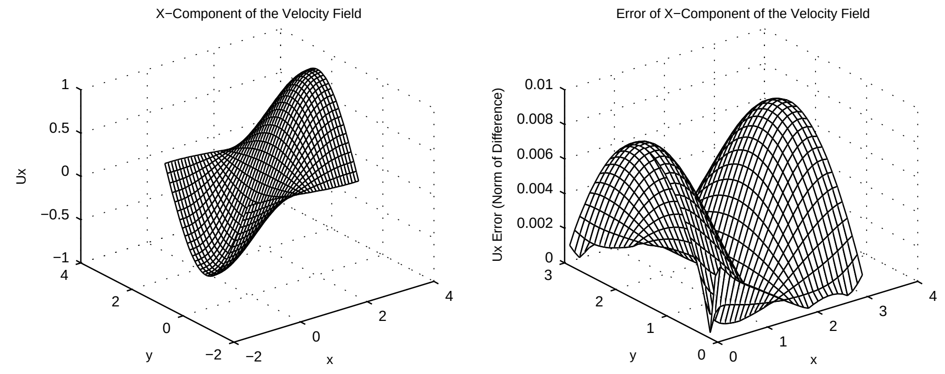


Figure 3.7: Solution for u_x, the x-component of the velocity on an 32-by-32 grid ($h = \frac{\pi}{32}$) along with the norm of the difference between our numerical solution and the actual solution

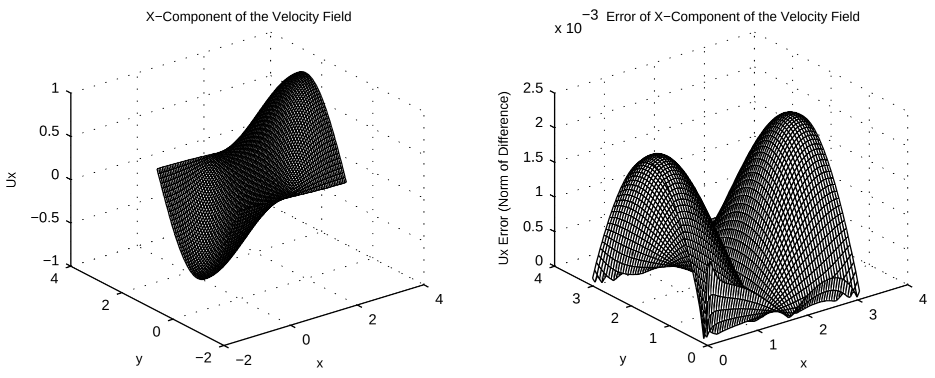


Figure 3.8: Solution for u_x, the x-component of the velocity on an 64-by-64 grid ($h = \frac{\pi}{64}$) along with the norm of the difference between our numerical solution and the actual solution

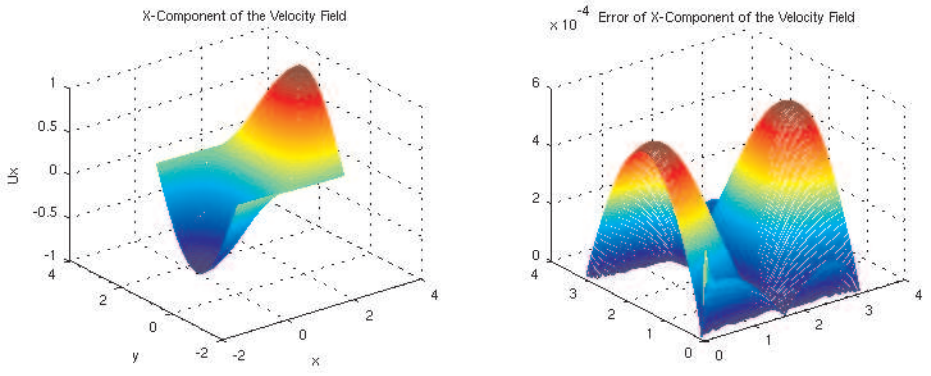


Figure 3.9: Solution for u_x, the x-component of the velocity on an 128-by-128 grid ($h = \frac{\pi}{128}$) along with the norm of the difference between our numerical solution and the actual solution

Finally, we choose an interior point and check to see how it converges as we increase our mesh-size. In 3.10, we plot the desired convergence as before where we take the value at our coarsest mesh and calculate the desired second-order convergence rate. Again, we see that our space-discretization methods perform as desired.

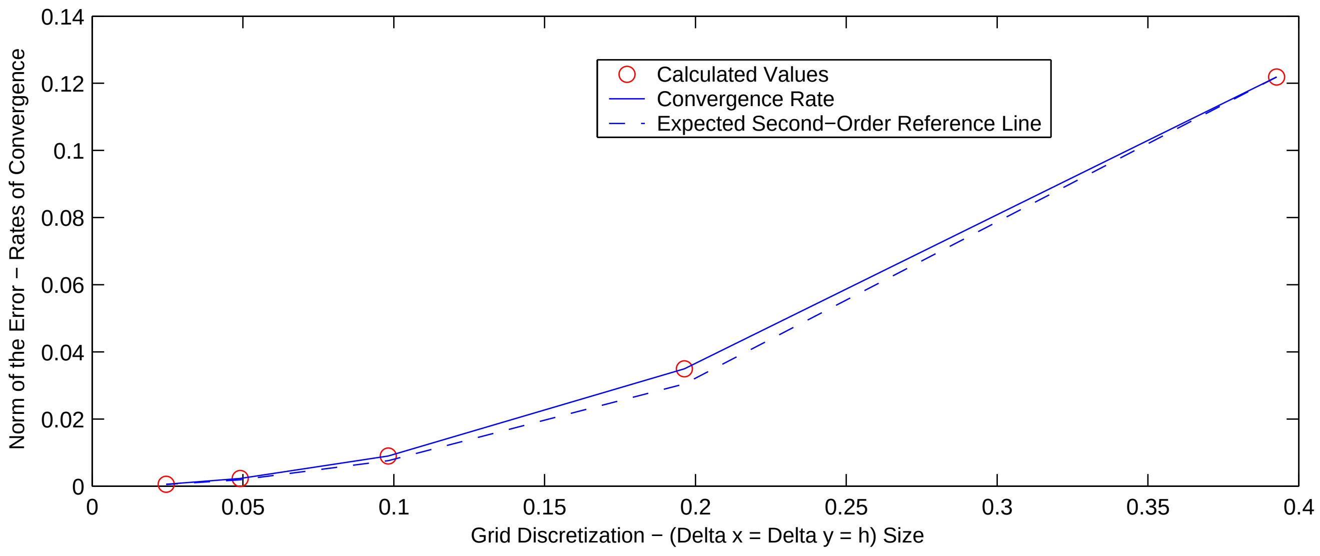


Figure 3.10: Plot of the errors of our numerical solver for changing grid size, verifying second-order (in time) convergence of the explicit Euler method for solver the incompressible NSE as we move from a coarse to fine mesh

Having verified the approach of the projection method as valid for first-order convergence, we are currently in the process of implementing a method for the incompressible Navier-Stokes such as the Van Kan method that is second-order in time as well as in space. Additionally, we hope to run this in parallel, using PETSC and the MPI-interface and a more efficient solver for this system of linear equations.

Chapter 4

Visualization of Incompressible 2-D Navier-Stokes and 3-D Stokes Flows

Given ways to solve the Navier-Stokes equations, we turn to ways in which solutions are visualized. Visualization provides a way to verify results or gain insight into further aspects of a problem. For example, we can verify the location vorticies, study the interaction of solid objects in a fluid or investigate how one fluid is advected and dispersed in another fluid of different viscosity. One simple yet powerful technique for visualizing fluids is Line Integral Convolution [8] which we look at in detail in 2-D for the Navier-Stokes equations and is easily extended to 3-D. Additionally, the modeling of fluids has become an increasingly popular area in the field of Computer Graphics and Animation as evidenced by several recent publications [20, 19, 18, 16, 35]. So, for 3-D, we discuss simple techniques used to visualize the interaction of 3-D rigid bodies in a highly viscous fluid from data provided by George Biros, Lexing Ying and Denis Zorin from [5],[50].

4.1 Introduction to Line Integral Convolution

Line Integral Convolution (LIC) [8] uses filtering techniques to blur local texture objects (pixels in 2-D and voxels in 3-D) along a streamline, found through a vector field provided manually or through a numerical solver for a PDE. For fluid visualization purposes, we use a randomly generated texture as input along with our vector field. The following discussion involves generating an LIC filtered image in 2-D using the output from a 2-D Navier-Stokes solver.

Given a 2-D vector field of size (N, M), assume that each interior point of the domain Ω as well as out boundary $\partial\Omega$ represents a single pixel, p. Then, we approximate the influence of a vector field on a pixel location $p = (x, y)$ by incorporating the influence of advecting in the forward and negative directions for L distance through the vector field, V, creating a parametric curve, $P(p, s)$. Define $F(p)$ to be the texture value of pixel p. Then, using a convolution/filter kernel, $k(s)$, the output pixel value, $F'(p)$ is given by

$$F'(p) = \frac{\int_{-L}^{L} F(P(p,s))k(s)ds}{\int_{-L}^{L} k(s)ds}. \tag{4.1}$$

Figure 4.1 shows a small example of how, starting at the shaded pixel value, we follow the vector field from pixel to pixel, summing the pixel value influences, convolving this sum and normalizing via a convolution function, $k(s)$.

The authors of [8] discuss how they choose their particular convolution filter, deciding upon a Hanning filter because of its empirical abilities to blur a texture in the direction of a vector field. In particular, they choose the filter:

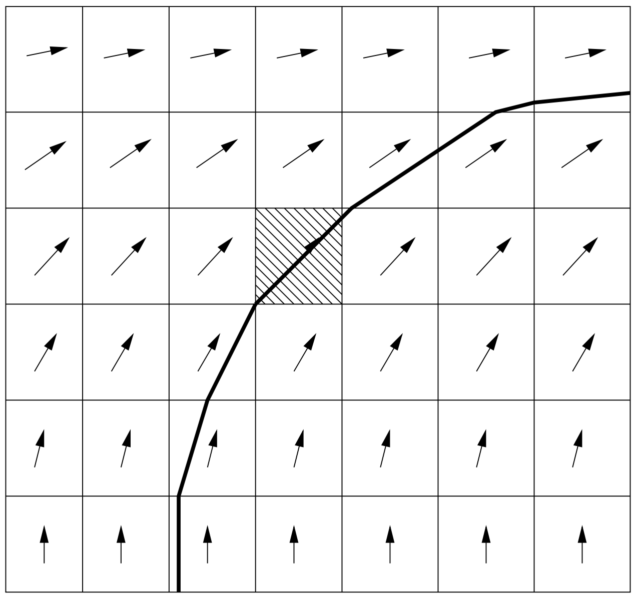


Figure 4.1: Example of vector field and generation of the streamline for the LIC algorithm

$$k(w) = \frac{1 + \cos(w + \beta)}{2}. \tag{4.2}$$

where β is a simple phase-shifting coefficient. One of the problems with 4.2 is that it has infinite influence; however, the authors of the LIC algorithm require a convolution kernel which which operated locally, so they window the Hanning filter with with a Gaussian, producing

$$k(w) = \frac{(1 + \cos(cw))(1 + \cos(dw + \beta)}{4}. \tag{4.3}$$

The constants c and d affect the dilation of the Gaussian window and the original Hanning filter. Empirical study with these constants is discussed more in the next section. For graphical examples of the Hanning filter and window functions, see [8].

4.2 The LIC Algorithm

Computing LIC is fairly straightforward. For a particular pixel, $p_0 = (x_0, y_0)$, let $P_0 = (x_0 + \frac{1}{2}, y_0 + \frac{1}{2})$ and $V(P_0) = V(\lfloor P_{x0} \rfloor), \lfloor P_{y0} \rfloor)$. Let P_i be the ith element along P_0's streamline, with vector field value, V_i. Then, we advect along a streamline, moving from pixel to pixel as in figure 4.1, in the forward direction by computing

$$P_i = P_{i-1} + \frac{V_{i-1}}{|V_{i-1}|} \Delta s_{i-1}.$$

and in the negative direction ($i \leq 0$) by computing

$$P_{i-1} = P_i - \frac{V_i}{|V_i|}\Delta s_i.$$

In the above, Δs_i represents the smallest positive (or negative for backward advection) distance along the advected streamline to the border of a pixel cell as in figure 4.1. That is, label the sides of a pixel cell as *top*, *bottom*, *left*, and *right*. Then,

$$\Delta s_i = min(s_{top}, s_{bottom}, s_{left}, s_{right}).$$

where

$$s_e = \frac{\lfloor P_c \rfloor - P_c}{V_c}.$$

and the e and c in the above equation refer to the following possible pairs of (e, c)

$$(e,c) \in \left\{ \begin{array}{c} (top, y) \\ (bottom, y) \\ (left, x) \\ (right, x) \end{array} \right\}.$$

Note, however, there are two circumstances in which s_e must be altered. First, if $\frac{\lfloor P_c \rfloor - P_c}{V_c} < 0$, then let $s_e = 0$. Second, if $V_c = 0$ (or close to 0), then let $s_e = \infty$. Also, if advection leads us to the boundaries of our vector field, or if we reach a pixel cell whose vector represents a 'sink', we terminate the advection, or simply let $\Delta s_i = 0$.

Once we know our values of P_i from the equations above for $-L < i < L$ and our chosen length, L, we numerically evaluate 4.1 as

$$F'(p) = \frac{\sum\limits_{i=l_{neg}}^{l_{pos}} F_i h_i}{\sum\limits_{i=l_{neg}}^{l_{pos}} h_i}. \tag{4.4}$$

where

$$F_i = F(\lfloor P_i \rfloor).$$

and h_i from 4.4 refers to the i-th component of the integral of our convolution kernel, $k(w)$:

$$h_i = \int_{s_i}^{s_{i+1}} k(w)dw. \tag{4.5}$$

where $s_0 = 0$ and $s_i = s_{i-1} + \Delta s_{i-1}$ for forward advection and $s_{i_1} = s_i - \Delta s_i$ for backward advection.

Further, for 4.4, $l_{pos} = i$ for $s_i \leq L < s_{i+1}$ and $l_{neg} = i$ for $s_{i-1} > L \geq s_i$.

In order to evaluate 4.5, we integrate 4.3 (done in [8]) as

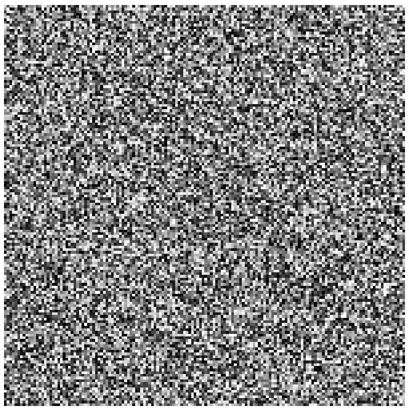


Figure 4.2: Random input texture for 2-D LIC

$$\int_{s_i}^{s_j} k(w)dw = \frac{1}{4}\left(\begin{array}{c} s_j - s_i + \\ \frac{\sin(s_j c)-\sin(s_i c)}{c} + \\ \frac{\sin(s_j d+\beta)-\sin(s_i d+\beta)}{d} + \\ \frac{\sin(s_j(c-d)-\beta)-\sin(s_i(c-d)-\beta)}{2(c-d)} + \\ \frac{\sin(s_j(c+d)+\beta)-\sin(s_i(c+d)+\beta)}{2(c+d)} \end{array}\right).$$

Choosing the constants c and d involves some experimentation, causing blurring, sharpening, or more or less apparent motion to be viewed. The authors of [8] suggest that the Hanning filter function should have a frequency of about 3 cycles per period of the Gaussian window, However, often playing with these values for specific problems is the best test. For example, in the next section, we end up using values of $c = \frac{1}{10}$, $d = \frac{1}{3}$ and $\beta = \frac{\pi}{4}$.

Knowing how to apply the LIC algorithm, and given a vector field, we need to have an original texture to sample. Sensibly, we choose an input of random pixel values, allowing us to sample several random values over any given streamline to convolve for our output. In particular, figure 4.2 is a perfect input texture. This texture is used in the next section to show example outputs for visualizing the incompressible Navier-Stokes equations in 2-D.

The above derivation was done with a 2-D LIC computation in mind. But, clearly the problem is easily extended to 3-D by extending the problem to voxels and voxel boundaries when pixels and pixel boundaries are discussed. In fact, our code is already capable of doing the 3-D LIC computation and we have generated several such results. The major difficulty in representing these solutions is how to visualize the resulting 3-D scalar fields. These problems have been studied in depth by various authors such as [40], [24] and [17], using volume rendering techniques such as those in [15], [41] and [46], all of which we are currently looking at in order to develop a good 3-D LIC representation.

4.3 LIC Results

As stated at the end of the last section, it is often necessary to play with the values of c, d and β for the LIC convolution filter, k. The following pictures use values of $c = \frac{1}{10}$, $d = \frac{1}{3}$ and $\beta = \frac{\pi}{4}$ for the following problem. For the Navier-Stokes equations, let the boundary conditions be defined as in figure 4.3. That is, the velocities at all of the boundaries are 0 except at the top of our 2-D box, the velocity in the x-direction is 1. This is known as the 2-D driven cavity problem.

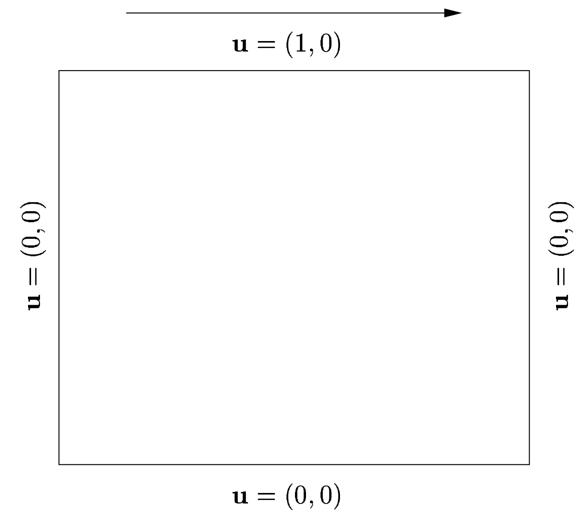


Figure 4.3: Boundary values for 2-D driven cavity problem for the NSE

Numerically solving this results in a vector field which we plug into our LIC algorithm. The following pictures were generated with low viscosity, so that interesting vorticities appear along the borders as time evolves. In particular, for this example, the Reynolds number, $Re = 10000$ is used on a 2-D box of size $[0,1]^2$. Only four time steps are shown, but for this example, we have computed up to 1000 time steps, and generated a movie which can be seen at [31] along with several other such movies and pictures. Additionally, a simple color is added to give the sense of this being a fluid, but along with investigating 3-D, we are also looking into using color to indicate the relative velocity at each point as well as possibly using color to indicate pressure variations.

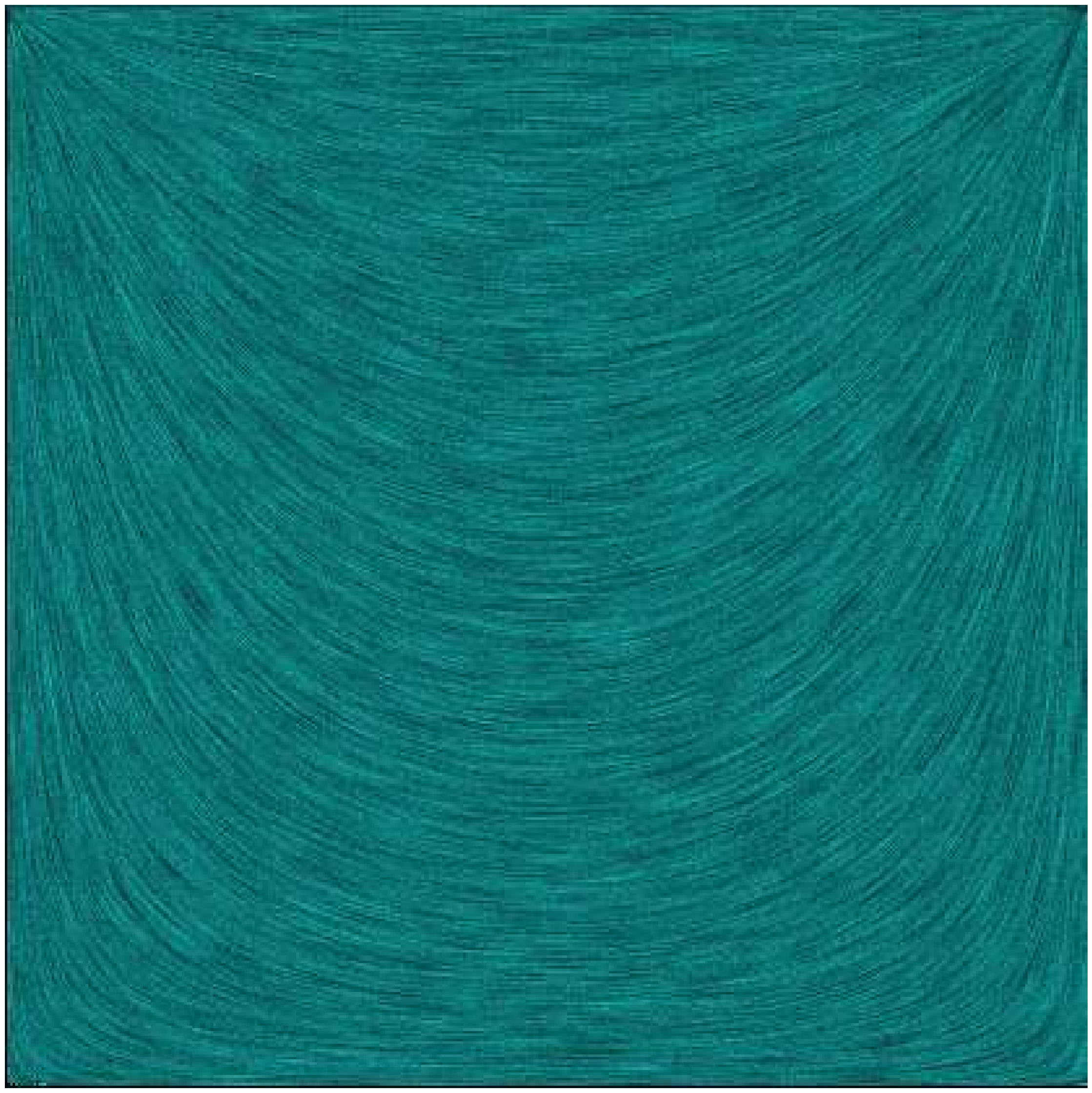


Figure 4.4: LIC visualization for 2-D cavity problem example 1

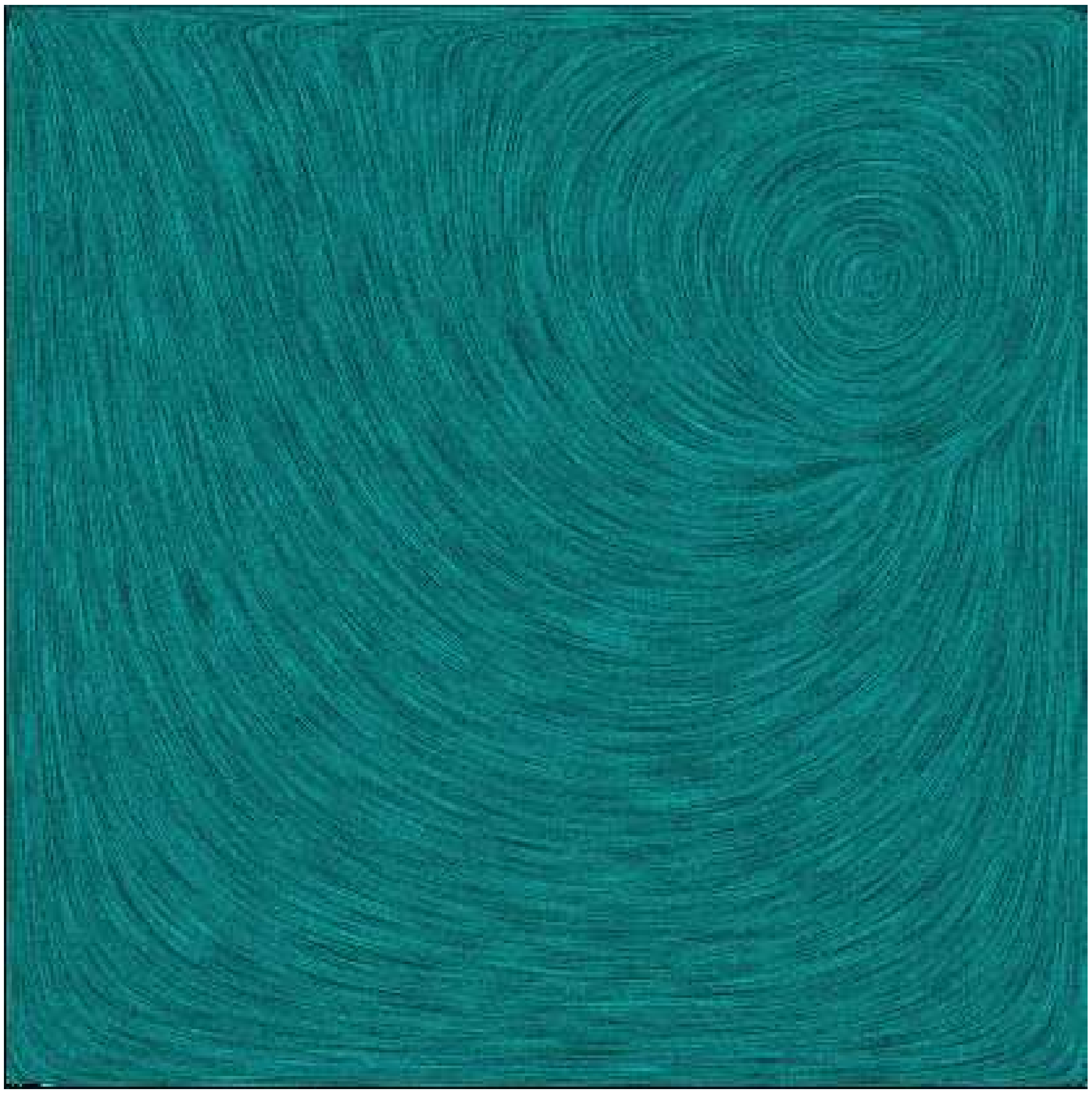


Figure 4.5: LIC visualization for 2-D cavity problem example 2

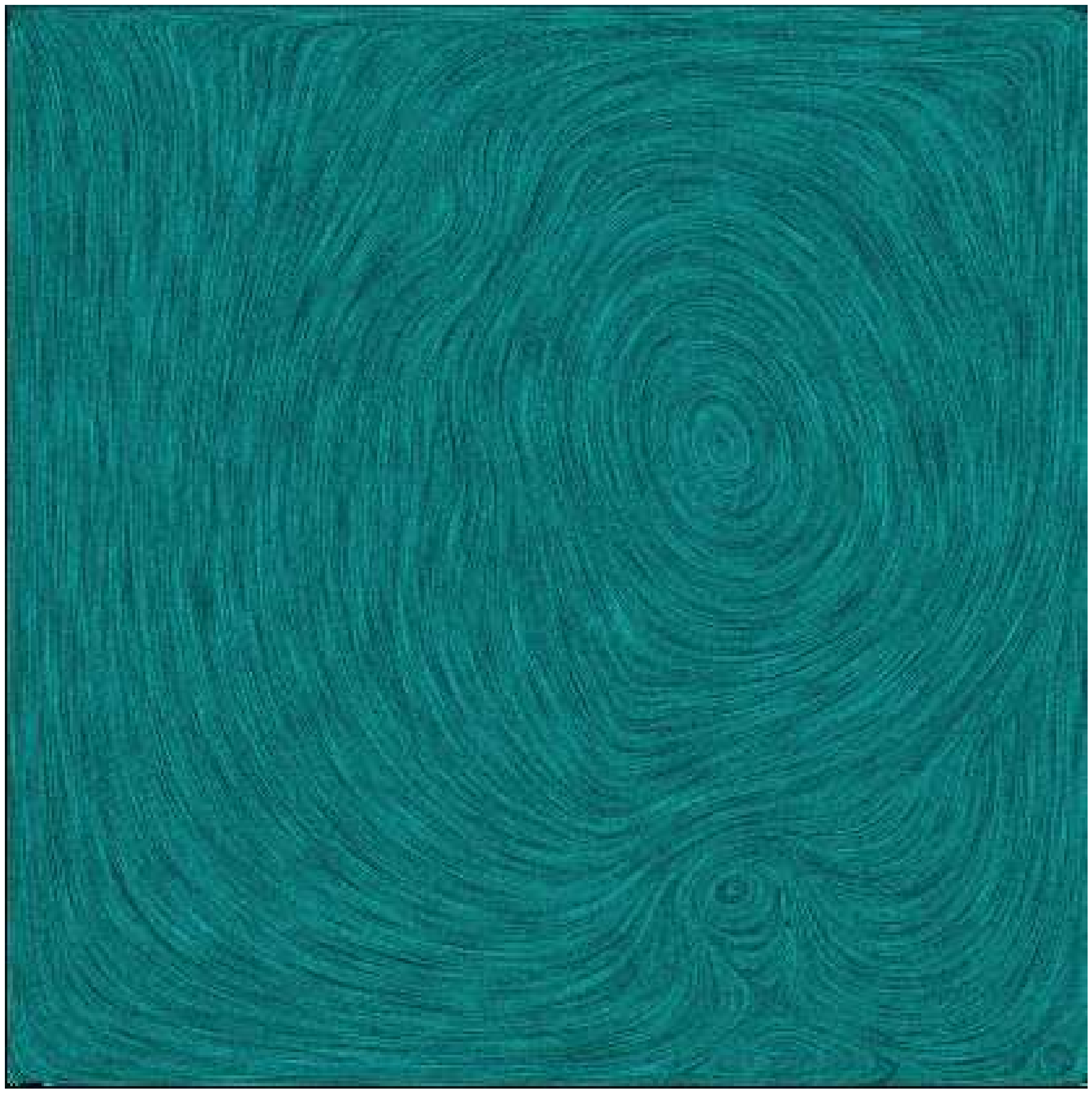


Figure 4.6: LIC visualization for 2-D cavity problem example 3

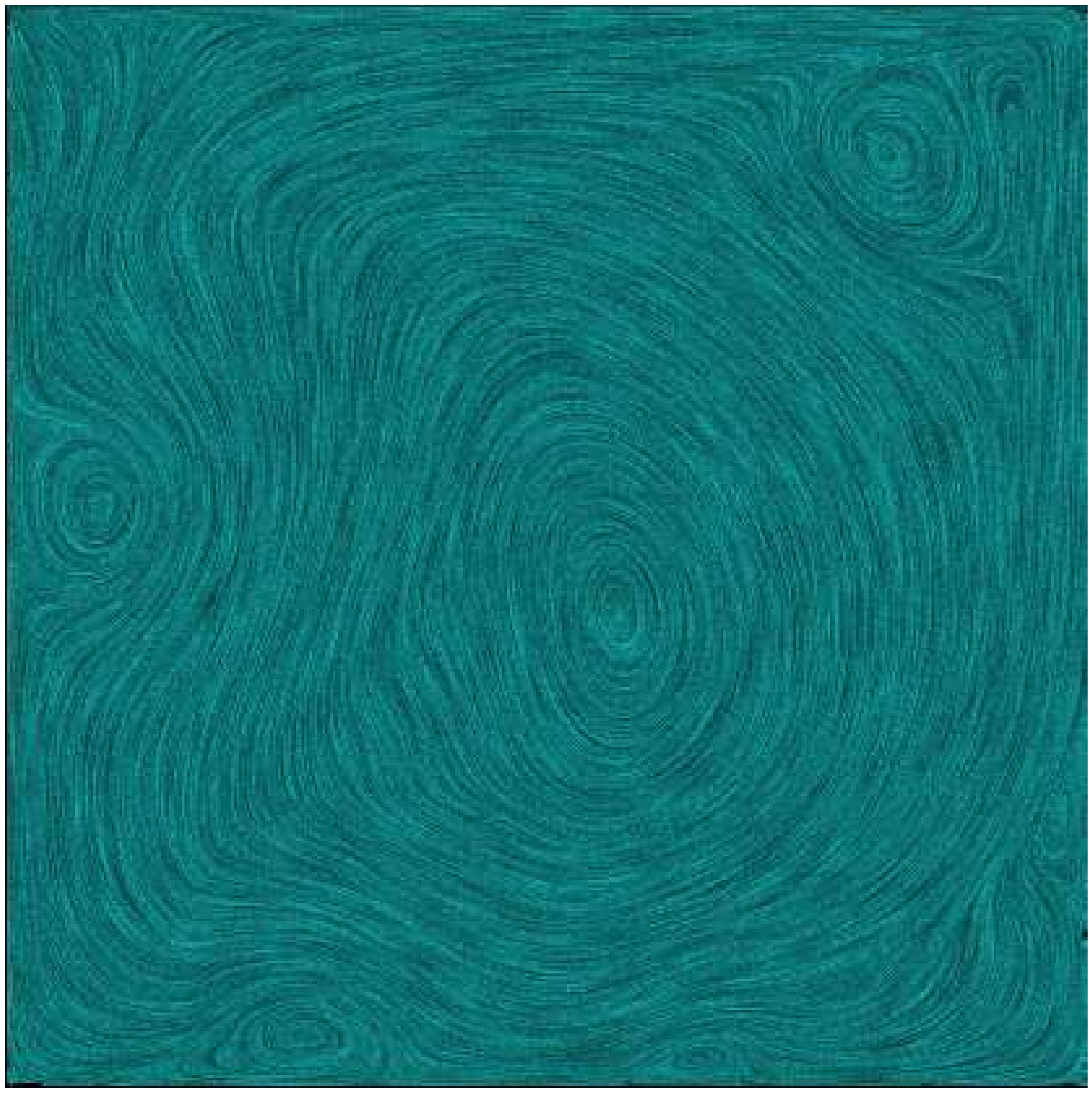


Figure 4.7: LIC visualization for 2-D cavity problem example 4

4.4 Visualizing Interaction of 3-D Rigid Bodies in a Stokesian Flow

While visualizing vector fields provides insight into the flow properties of a fluid, visualizing the interaction of solid objects in a flow provides real-world insight. Working with data provided by George Biros, Lexing Ying and Denis Zorin from their work in developing a fast parallel solver for the interaction of 3-D rigid bodies in a highly viscous flow (negligible convective term) through the work in [6] and [50], we have developed some visualizations of their 3-D data with some simple techniques.

First, we treat each rigid body as a simple skeleton. The motivation for this was for the possible inclusion of animals such as fish into a 3-D flow. For the examples generated thus far, however, these bodies were simple and 'one-jointed', such as spheres or a propeller which do not allow articulated movement. Then, the data was transformed into a Biovision format file as in [47]. That is, each joint is treated as being offset from its relative root or parent joint where there is one root of the full figure (such as a hip for a human figure), so that as the root translates, each connected child joint translates accordingly. Similarly, if a joint is the parent of another joint, and it is rotated, the rotations travel down to each child joint (i.e., elbow to wrist) and so on. For our purposes, the root joint is given six degrees of freedom (three translation and three rotation) while each successive joint is given three rotations. Each translation or rotation is treated as a channel which is given a numeric value at each frame in our animation. For a specific example of what this file would look like for a multi-jointed animal with a hip, chest, head and tail joint, producing two frames of animation at intervals of $\frac{1}{2}$ second each (1 second of total animation), see figure 4.8.

Treating each of our rigid bodies as an individual skeleton (non-articulated for 1-jointed bodies examples below) in a 3-D flow, we transform the data provided into biovision format, which is then easily imported into Maya by using Maya's MEL scripting capabilities. Currently, we are working to automate the entire process, writing a plug-in using Maya's C++ API.

This process was done for several examples, movies of which are located at [31]. Below are several examples of our results for the interaction of nine spheres in a highly viscous flow (figure 4.9) followed by the interaction of a sphere and a propeller in a highly viscous flow (figure 4.10).

```
HIERARCHY
ROOT Hips
{
    OFFSET    0.00    0.00    0.00
    CHANNELS 6 Xposition Yposition Zposition Zrotation Xrotation Yrotation
    JOINT Chest
    {
        OFFSET    0.00    5.21    0.00
        CHANNELS 3 Zrotation Xrotation Yrotation
        JOINT Head
        {
           OFFSET    0.00    18.65     0.00
           CHANNELS 3 Zrotation Xrotation Yrotation
           End Site
           {
              OFFSET    0.00    3.05    0.00
            }
         }
     }
     JOINT Tail
     {
        OFFSET    0.00    −10.00    0.00
        CHANNELS 3 Zrotation Xrotation Yrotation
        End Site
        {
            OFFSET    0.00    −3.05    0.00
         }
     }
 }
 MOTION
 Frames: 2
 Frame Time: 0.5
 8.00 35.00 88.00 −3.40 14.50 −164.0 13.00 40.50 −24.60 7.90 0.00 −3.60
 7.50 35.00 86.00 −3.90 12.50 −166.0 12.90 42.50 −22.90 7.80 0.00 −4.80
```

Figure 4.8: Example Biovision (BVH) file format

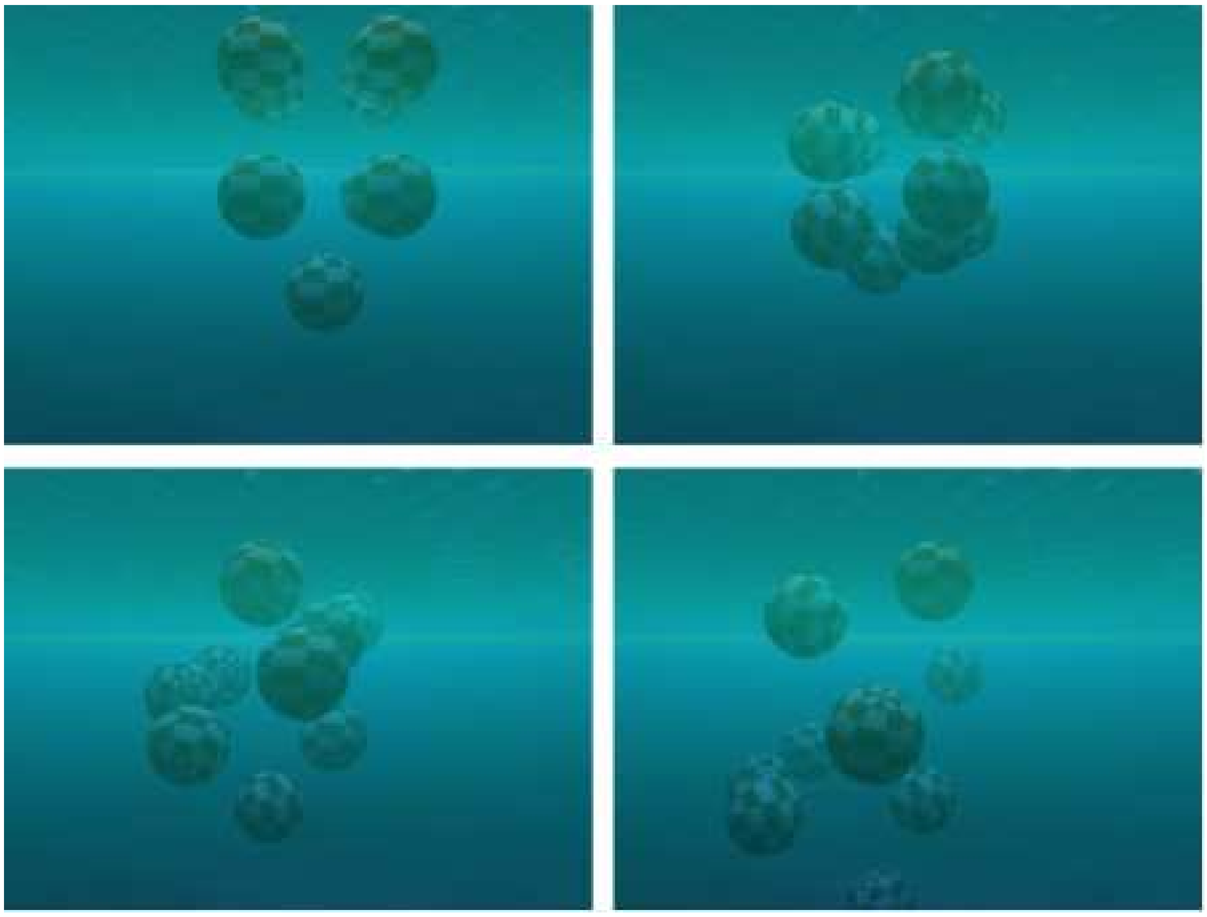


Figure 4.9: Simulation of nine spheres interacting in a Stokesian flow

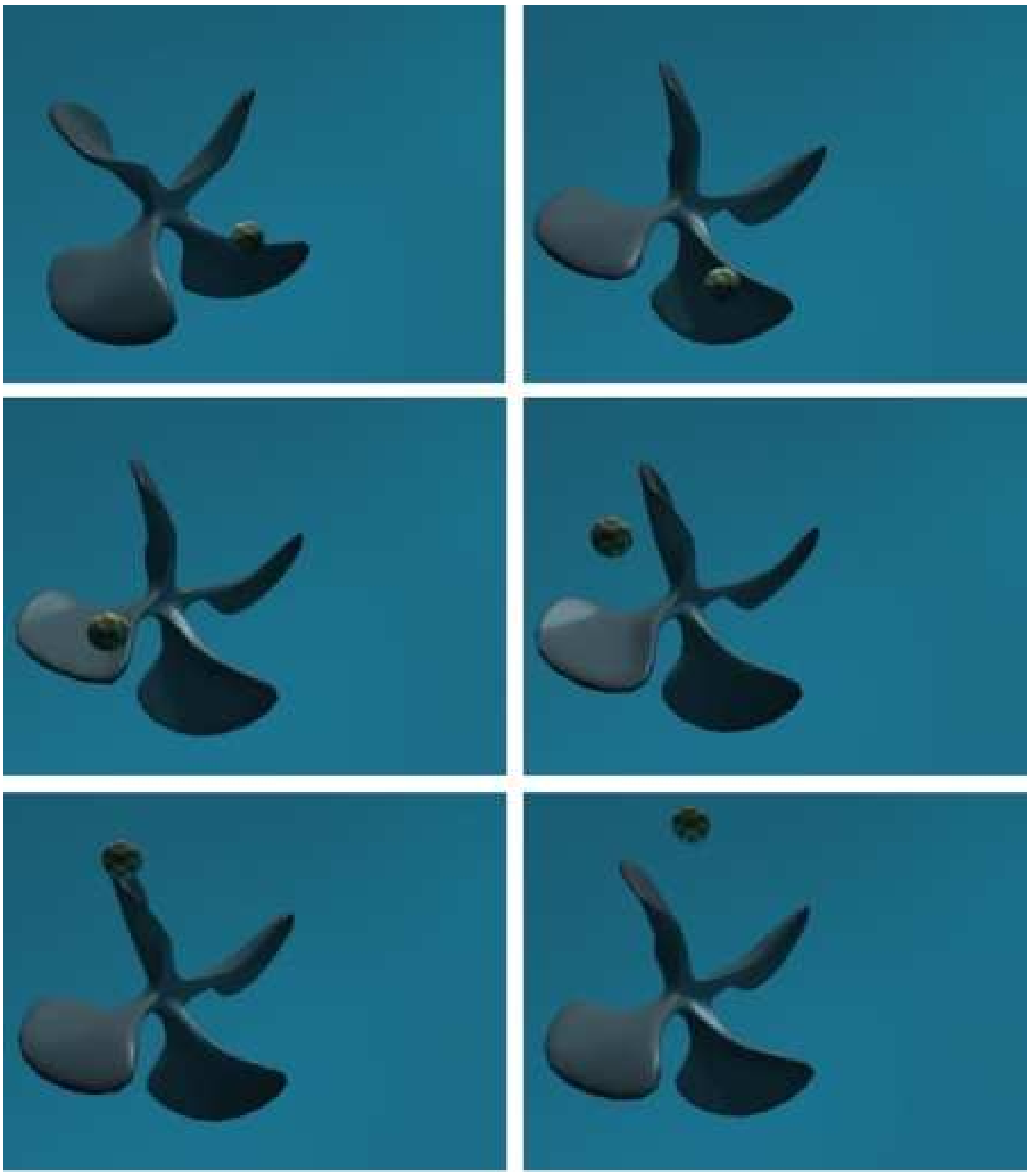


Figure 4.10: Simulation of a propeller and sphere interacting in a Stokesian flow

Chapter 5

Future Work and Conclusion

5.1 *Ongoing and Future Work*

Throughout, we have discussed ways in which we are continuing to work with the incompressible Navier-Stokes fluid equations. For example, implementing a parallel second-order solver for these equations is one current pursuit as in the work of [50] at the NYU Media Research Lab. Additionally, we are building support for 3-D Line Integral Convolution and visualization. Finally, we mention a visualization system that is being developed at the Media Research Lab at New York University in the next subsection, which we hope to integrate with our ongoing work in fluid dynamics.

5.1.1 The Responsive Workbench

One way in which we are attempting to visualize fluid models is a stereoscopic interactive workbench, called the Responsive Workbench (RWB).

We have been working to build a RWB, based on a similar system at Stanford [2], [21]. In a nutshell, the RWB is an interactive 3D virtual environment and workspace. A desired scene is rendered for the right and left-eye's respective viewpoints, and each of these two images is projected through circularly polarized lenses, one clockwise, the other counter-clockwise, and reflected off of an angled mirror onto the bottom of a plastic tabletop (see figure 5.1 for a picture of our recently built table). The viewer wears similarly polarized glasses (see figure 5.2) such that each eye is able to view the left or right-eye image in order to create the desired stereoscopic effect. A 6 Degree of Freedom (6DOF) tracking system tracks (see figure 5.3) the location of the glasses so the viewer always sees the scene from the correct point of view. A 6DOF mouse (see figure 5.2) and a virtual toolbox are also used to allow the user to select, translate, rotate, scale and change the transparency of objects in the virtual environment.

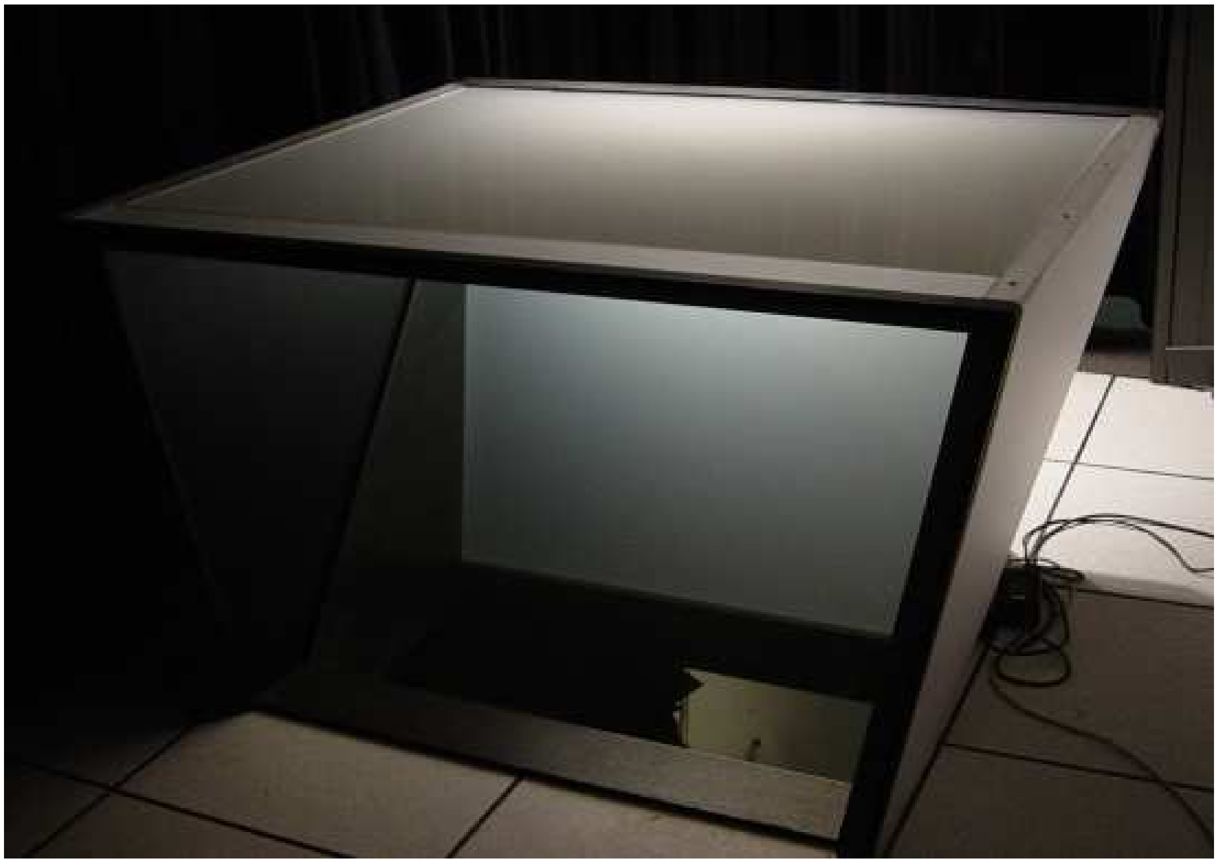


Figure 5.1: The Responsive Workbench (RWB)

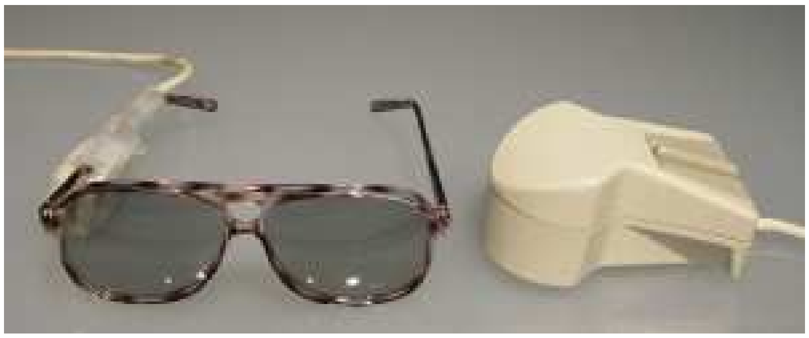


Figure 5.2: Circularly polarized lenses with a tracker on the side as well as a 6DOF mouse with three input buttons

Figure 5.3: Two Flock of Birds trackers (one for eyes and the other for the 6DOF mouse)

Among the many demonstration files we have running on the system is the model of a skeleton as in figure 5.4, which exemplifies some of the best aspects of the system as the user is able to remove, translate, rotate and zoom in on the entire scene or single objects such as bones. Currently, through code of Lexing Ying and the work of Jason Lee, we are working to implement a stereoscopic subdivision model viewing scheme such that the user can interactively subdivide any object in a scene. For example, we have done so for a heart model (see figure 5.5).

By building such a system, the hope is to eventually visualize fluid motions interactively and stereoscopically such as the Peskin's Immersed Boundary Fluids [36], [37], specifically the simulation of the heart which is partially why we are interested in visualizing the subdividing heart model. Along these lines, we are building an additional tool for the RWB's virtual toolbox which will allow us to take cross-slices of an object, potentially being able to view fluids as they move through a rigid boundary.

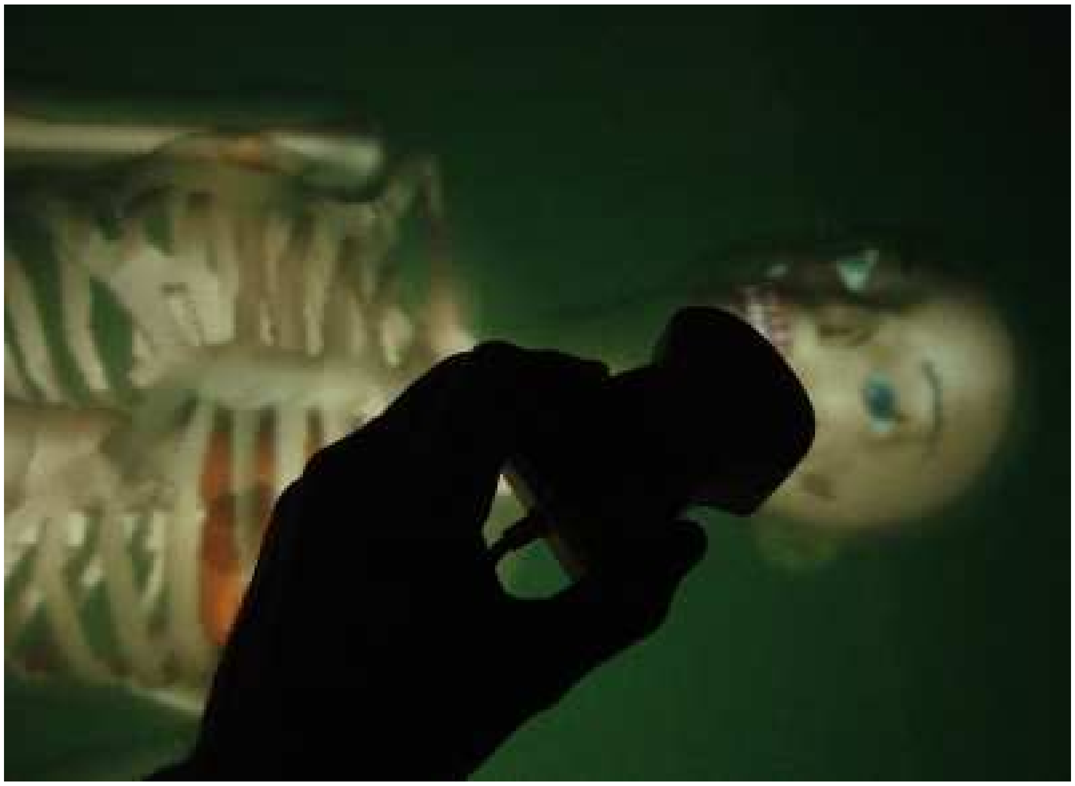


Figure 5.4: Example visualization of skeleton head on the RWB with 6DOF interacting

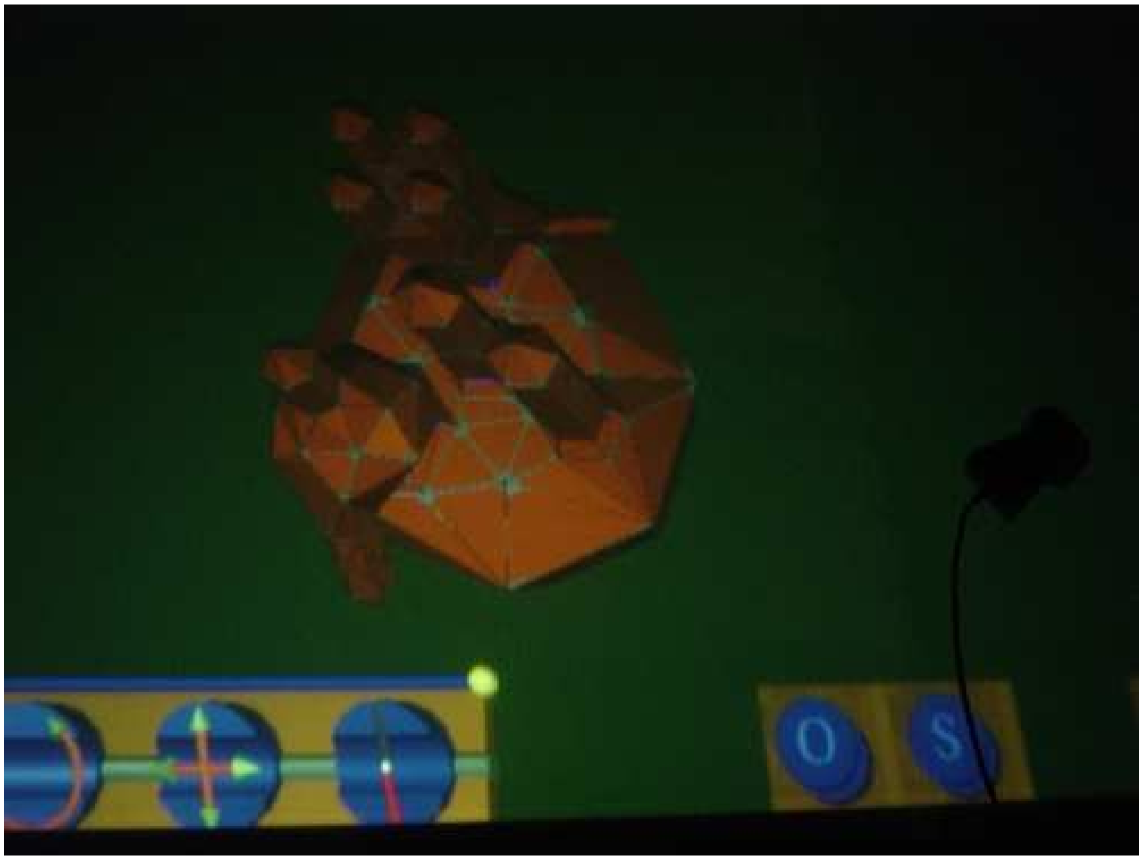


Figure 5.5: L. Ying's subdivided heart model and code as implemented by J. Lee for the RWB

5.2 Conclusion

We have discussed a classical derivation of the incompressible Navier-Stokes equations as well as analyzed several aspects of these equations such as the role of different components (the diffusion and convective terms) and how we define well-posedness for them specifically. We then discussed the Helmholtz-Hodge Decomposition, and how this applies to the Chorin-Temam scheme and other projection-based methods. Further, we showed how the incompressible Navier-Stokes equations are discretized in time and space for first-order and second-order (in time) methods using finite-difference, Euler, Crank-Nicholson and additional numerical schemes. We also discussed different visualization techniques such as the Line Integral Convolution algorithm and a simple way of modeling the interaction of 3-D rigid bodies as well as ways in which the Media Research Lab at New York University is approaching visualization and animation of fluids methods.

Appendix A

Appendix

A.1 Identities for Vector Fields

Here we list a few vector field identities which are of help in understanding components of the Navier-Stokes equations such as divergence, gradient, the convective term, the Laplacian, etc. We use $\cdot$ to represent the dot-product and $\times$ to represent the cross-product.

For our discussions, we assume that a vector, $\mathbf{u}$ is represented in component form in 3-D as

$$\mathbf{u}(x,y,z) = u_x(x,y,z)\mathbf{i} + u_y(x,y,z)\mathbf{j} + u_z(x,y,z)\mathbf{k} = (u_x, u_y, u_z). \tag{A.1}$$

and for 2-D, we use $\mathbf{v}$ as

$$\mathbf{v}(x,y,z) = v_x(x,y,z)\mathbf{i} + v_y(x,y,z)\mathbf{j} = (v_x, v_y). \tag{A.2}$$

In fact, for these discussions, we can use any vector of size n, but we are only really interested in 2-D and 3-D fluids, so we keep or discussions to those dimensions.

We use the gradient operator, ∇, often referred to as *del* or the *differential operator*, represented as a vector as

$$\nabla = \frac{\partial}{\partial x}\mathbf{i} + \frac{\partial}{\partial y}\mathbf{j} + \frac{\partial}{\partial z}\mathbf{k} = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right).$$

Applying ∇ to A.1, we obtain

$$\nabla\mathbf{u} = \frac{\partial \mathbf{u}}{\partial x}\mathbf{i} + \frac{\partial \mathbf{u}}{\partial y}\mathbf{j} + \frac{\partial \mathbf{u}}{\partial z}\mathbf{k} = \left(\frac{\partial \mathbf{u}}{\partial x}, \frac{\partial \mathbf{u}}{\partial y}, \frac{\partial \mathbf{u}}{\partial y}\right). \tag{A.3}$$

For $\mathbf{v}$ we obtain

$$\nabla\mathbf{v} = \frac{\partial \mathbf{v}}{\partial x}\mathbf{i} + \frac{\partial \mathbf{v}}{\partial y}\mathbf{j} = \left(\frac{\partial \mathbf{v}}{\partial x}, \frac{\partial \mathbf{u}}{\partial y}\right). \tag{A.4}$$

For vectors, $\mathbf{f}, \mathbf{g} \in \Re^n$, $n = 2, 3$, and constant, k, we have the following vector identities:

$$\nabla(k\mathbf{f}) = k(\nabla\mathbf{f}).$$

$$\nabla(\mathbf{f} + \mathbf{g}) = \nabla\mathbf{f} + \nabla\mathbf{g}.$$

$$\nabla(\mathbf{fg}) = \mathbf{f}\nabla\mathbf{g} + \mathbf{g}\nabla\mathbf{f}.$$

For $\mathbf{u}$ and $\mathbf{v}$, we define the divergence, $\nabla\cdot$, as

$$\nabla \cdot \mathbf{u} = \frac{\partial \mathbf{u_x}}{\partial x} + \frac{\partial \mathbf{u_y}}{\partial y} + \frac{\partial \mathbf{u_z}}{\partial z}.$$

$$\nabla \cdot \mathbf{v} = \frac{\partial \mathbf{v_x}}{\partial x} + \frac{\partial \mathbf{v_y}}{\partial y}.$$

Further, for a 3-D scalar variable, p, we define the divergence as

$$\nabla \cdot p = \frac{\partial p}{\partial x} + \frac{\partial p}{\partial y} + \frac{\partial p}{\partial z}.$$

leaving off the last term when calculating the divergence of a 2-D scalar field.

Again, for $\mathbf{f}$ and $\mathbf{g}$, we have the following identities

$$\nabla \cdot (k\mathbf{f}) = k(\nabla \cdot \mathbf{f}).$$

$$\nabla \cdot (\mathbf{f} + \mathbf{g}) = \nabla \cdot \mathbf{f} + \nabla \cdot \mathbf{g}.$$

For a scalar field, $a(x, y, z)$ in 3-D and $a(x, y)$ in 2-D, we also have

$$\nabla \cdot (a\mathbf{f}) = a\nabla \cdot \mathbf{f} + \nabla a \cdot \mathbf{f}.$$

We briefly mention the curl operator, $\nabla\times$, which is helpful in the analysis of fluid flows and their rotations and circulation (for more information, see [3]). For example for an *irrotational* 3-D flow, $\mathbf{u}$, $\nabla \times \mathbf{u} = 0$.

For our 3-D vector, $\mathbf{u}$, we can compute the curl as

$$\nabla \times \mathbf{u} = \left(\frac{\partial u_z}{\partial y} - \frac{\partial u_y}{\partial z}\right)\mathbf{i} + \left(\frac{\partial u_x}{\partial z} - \frac{\partial u_z}{\partial x}\right)\mathbf{j} + \left(\frac{\partial u_y}{\partial x} - \frac{\partial u_x}{\partial y}\right)\mathbf{k} = \begin{bmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ u_x & u_y & u_z \end{bmatrix}.$$

For the 2-D field, $\mathbf{v}$, it is apparent that since $v_z = 0$, $\nabla \times \mathbf{v} = \left(\frac{\partial v_y}{\partial x} - \frac{\partial v_x}{\partial y}\right)\mathbf{k}$.

We have a few identities for the curl as well (again using our scalar field, $a \neq \mathit{constant}$ and $\mathbf{f}$ and $\mathbf{g}$.

$$\nabla \times (k\mathbf{f}) = k(\nabla \times \mathbf{f}).$$

$$\nabla \times (\mathbf{f} + \mathbf{g}) = \nabla \times \mathbf{f} + \nabla \times \mathbf{g}.$$

$$\nabla \times (a\mathbf{f}) = a\nabla \times \mathbf{f} + \nabla a \times \mathbf{u}.$$

Following are a few often helpful identities concerning the curl, divergence and gradient. Again, we use $\mathbf{f}$ and $\mathbf{g}$ to represent vectors both in either $\Re^2$ or $\Re^3$, arbitrarily.

$$\nabla \times (\nabla \mathbf{f}) = \mathbf{0}.$$

$$\nabla \cdot (\nabla \times \mathbf{f}) = \mathbf{0}.$$

$$\nabla \cdot (\mathbf{f} \times \mathbf{g}) = \mathbf{g} \cdot (\nabla \times \mathbf{f}) - \mathbf{f} \cdot (\nabla \times \mathbf{g}).$$

$$\nabla \cdot (\nabla \mathbf{f}) = \nabla^2 \mathbf{f} = \Delta \mathbf{f}. \tag{A.5}$$

$$\nabla \times (\nabla \times \mathbf{f}) = \nabla(\nabla \cdot \mathbf{f}) - \Delta \mathbf{f}.$$

$$\nabla(\nabla \cdot \mathbf{f}) = \nabla \times (\nabla \times \mathbf{f}) + \Delta \mathbf{f}.$$

$$\nabla(\mathbf{f} \cdot \mathbf{g}) = (\mathbf{f} \cdot \nabla)\mathbf{g} + (\mathbf{g} \cdot \nabla)\mathbf{g} + (\mathbf{f} \times (\nabla \times \mathbf{g})) + (\mathbf{g} \times (\nabla \times \mathbf{g})). \tag{A.6}$$

$$\nabla(\mathbf{f} \times \mathbf{g}) = \mathbf{f}(\nabla \cdot \mathbf{g}) - \mathbf{g}(\nabla \cdot \mathbf{f}) + (\mathbf{g} \cdot \nabla)\mathbf{f} - (\mathbf{f} \cdot \nabla)\mathbf{g}. \tag{A.7}$$

We notice that A.5 is the *Laplacian* operator, the divergence of the gradient, which we can expand in 3-D as

$$\nabla^2 = \Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}.$$

with its apparent 2-D equivalent

$$\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}.$$

Also, we notice that A.6 and A.7 have the convective term, $(\mathbf{g} \cdot \nabla)\mathbf{f}$, from the Navier-Stokes equations. We use $\mathbf{u_1}, \mathbf{u_2} \in \Re^3$ for 3-D and $\mathbf{v_1}, \mathbf{v_2} \in \Re^2$ for 2-D to show how to calculate this term. First, we calculate $\mathbf{u_1} \cdot \nabla$ and $\mathbf{v_1} \cdot \nabla$

$$\mathbf{u_1} \cdot \nabla = u_{1_x} \frac{\partial}{\partial x} + u_{1_y} \frac{\partial}{\partial y} + u_{1_z} \frac{\partial}{\partial z}.$$

$$\mathbf{v_1} \cdot \nabla = v_{1_x} \frac{\partial}{\partial x} + v_{1_y} \frac{\partial}{\partial y}.$$

If we have a scalar function, a, and apply last two functions to it, we get

$$(\mathbf{u_1} \cdot \nabla)a = u_{1_x} \frac{\partial a}{\partial x} + u_{1_y} \frac{\partial a}{\partial y} + u_{1_z} \frac{\partial a}{\partial z}.$$

.

$$(\mathbf{v_1} \cdot \nabla)a = v_{1_x} \frac{\partial a}{\partial x} + v_{1_y} \frac{\partial a}{\partial y}.$$

However, if we apply $\mathbf{u_2}$ and $\mathbf{v_2}$, which are non-scalar vector fields, we have to do additional work, resulting in the following equations:

$$(\mathbf{u_1} \cdot \nabla)\mathbf{u_2} = u_{1_x} \left(\frac{\partial u_{2_x}}{\partial x}\mathbf{i} + \frac{\partial u_{2_y}}{\partial x}\mathbf{j} + \frac{\partial u_{2_z}}{\partial x}\mathbf{k} \right) + u_{1_y} \left(\frac{\partial u_{2_x}}{\partial y}\mathbf{i} + \frac{\partial u_{2_y}}{\partial y}\mathbf{j} + \frac{\partial u_{2_z}}{\partial y}\mathbf{k} \right) + u_{1_z} \left(\frac{\partial u_{2_x}}{\partial z}\mathbf{i} + \frac{\partial u_{2_y}}{\partial z}\mathbf{j} + \frac{\partial u_{2_z}}{\partial z}\mathbf{k} \right). \tag{A.8}$$

$$(\mathbf{v_1} \cdot \nabla)\mathbf{v_2} = v_{1_x} \left(\frac{\partial v_{2_x}}{\partial x}\mathbf{i} + \frac{\partial v_{2_y}}{\partial x}\mathbf{j} + \frac{\partial v_{2_z}}{\partial x}\mathbf{k} \right) + v_{1_y} \left(\frac{\partial v_{2_x}}{\partial y}\mathbf{i} + \frac{\partial v_{2_y}}{\partial y}\mathbf{j} + \frac{\partial v_{2_z}}{\partial y}\mathbf{k} \right). \tag{A.9}$$

Now, to be of actual assistance in our analysis of the Navier-Stokes equations, we need to commute this term solely for $\mathbf{u} \in \Re^3$ or $\mathbf{v} \in \Re^2$, depending on whether or not we are solving a 2-D or 3-D problem. So, using A.8 to compute $(\mathbf{u} \cdot \nabla)\mathbf{u}$, we get

$$(\mathbf{u}\cdot\nabla)\mathbf{u} = u_x\left(\frac{\partial u_x}{\partial x}\mathbf{i} + \frac{\partial u_y}{\partial x}\mathbf{j} + \frac{\partial u_z}{\partial x}\mathbf{k}\right) + u_y\left(\frac{\partial u_x}{\partial y}\mathbf{i} + \frac{\partial u_y}{\partial y}\mathbf{j} + \frac{\partial u_z}{\partial y}\mathbf{k}\right) + u_z\left(\frac{\partial u_x}{\partial z}\mathbf{i} + \frac{\partial u_y}{\partial z}\mathbf{j} + \frac{\partial u_z}{\partial z}\mathbf{k}\right). \tag{A.10}$$

Collecting terms, A.10 becomes

$$(\mathbf{u}\cdot\nabla)\mathbf{u} = \left(u_x\frac{\partial u_x}{\partial x} + u_y\frac{\partial u_x}{\partial y} + u_z\frac{\partial u_x}{\partial z}\right)\mathbf{i} + \left(u_x\frac{\partial u_y}{\partial x} + u_y\frac{\partial u_y}{\partial y} + u_z\frac{\partial u_y}{\partial z}\right)\mathbf{j} + \left(u_x\frac{\partial u_z}{\partial x} + u_y\frac{\partial u_z}{\partial y} + u_z\frac{\partial u_z}{\partial z})\mathbf{k}\right). \tag{A.11}$$

Now, using A.9 to compute $(\mathbf{v}\cdot\nabla)\mathbf{v}$, we get

$$(\mathbf{v}\cdot\nabla)\mathbf{v} = v_x\left(\frac{\partial v_x}{\partial x}\mathbf{i} + \frac{\partial v_y}{\partial x}\mathbf{j}\right) + v_y\left(\frac{\partial v_x}{\partial y}\mathbf{i} + \frac{\partial v_y}{\partial y}\mathbf{j}\right). \tag{A.12}$$

Collecting terms, A.12 becomes

$$(\mathbf{v}\cdot\nabla)\mathbf{v} = \left(v_x\frac{\partial v_x}{\partial x} + v_y\frac{\partial v_x}{\partial y}\right)\mathbf{i} + \left(v_x\frac{\partial v_y}{\partial x} + v_y\frac{\partial v_y}{\partial y}\right)\mathbf{j} = \left(v_x\frac{\partial v_x}{\partial x} + v_y\frac{\partial v_x}{\partial y}, v_x\frac{\partial v_y}{\partial x} + v_y\frac{\partial v_y}{\partial y}\right). \tag{A.13}$$

Finally, we present the Divergence Theorem which is helpful in derivations of certain forms of equations (such as the mass conservation equation in Chapter 1).

If V is our volume with boundary ∂V, then we relate the volume integral of the divergence of some vector-valued function, $\mathbf{F}$ and the surface integral of $\mathbf{F}$ by the following theorem.

Theorem A.1.1 (Divergence Theorem)

$$\int_{\partial V}(\mathbf{F}\cdot\mathbf{n})dA = \int_V(\nabla\cdot\mathbf{F})dV$$

Bibliography

[1] D.J. Acheson. *Elementary Fluid Dynamics.* Oxford University Press, 1990.

[2] M. Agrawala, A.C. Beers, B. Frohlich, P. Hanrahan, I. MacDowall, and M. Bolas. The two-user responsive workbench: Support for collaboration through individual views of a shared space. In *Siggraph Proceedings*, 1997.

[3] G.K. Batchelor. *An Introduction to Fluid Dynamics.* Cambridge University Press, 1st edition, 1967.

[4] J.B. Bell, P. Colella, and H.M. Glaz. A second-order projection-method for viscous incompressible flow. *Journal of Computational Physics*, 85:257–283, 1989.

[5] G. Biros, L. Ying, and D. Zorin. An embedded boundary integral solver for the stokes equations. *In Review*, 2002.

[6] G. Biros, L. Ying, and D. Zorin. An embedded boundary integral solver for the unsteady incompressible navier-stokes equations. *In Review*, 2003.

[7] D.L. Brown, R. Cortez, and M.L. Minion. Accurate projection methods for the incompressible navier-stokes equations. *Journal of Computational Physics*, 168:464–499, 2001.

[8] B. Cabral and L. Leedom. Imaging vector fields using line integral convolution. In *Siggraph Proceedings*, pages 263–270, 1993.

[9] A.J. Chorin. A numerical method for solving incompressible viscous flow problems. *Journal of Computational Physics*, 2:12–26, 1967.

[10] A.J. Chorin. Numerical solution of the navier-stokes equations. *Mathematics of Computation*, 22:745–762, 1968.

[11] A.J. Chorin. *Computational Fluid Mechanics Selected Papers.* Academic Press, 1st edition, 1989.

[12] A.J. Chorin and J.E. Marsden. *A Mathematical Introduction to Fluid Dynamics.* Springer-Verlag, 3rd edition, 1993.

[13] J.W. Demmel. *Applied Numerical Linear Algebra.* SIAM, 1st edition, 1997.

[14] A. Dold and B. Eckmann. Multigrid methods. In *Spronger-Verlag Lecture Notes in Mathematics 960, Proceedings, Koln-Porz*, 1981.

[15] R.A. Drebin, L. Carpenter, and P. Hanrahan. Volume rendering. *Computer Graphics*, 22(4):65–74, 1988.

[16] D. Enright, S. Marschner, and R. Fedkiw. Animation and rendering of complex water surfaces. In *SIGGRAPH 2002, Computer Graphics Proceedings*, pages 736–744. ACM Press / ACM SIGGRAPH, 2002.

[17] L.K. Forssell. Visualizing flow over curvilinear grid surfaces using line integral convolution. In *IEEE Visualization*, 1994.

[18] N. Foster and R. Fedkiw. Practical animations of liquids. In *SIGGRAPH 2001, Computer Graphics Proceedings*, pages 23–30. ACM Press / ACM SIGGRAPH, 2001.

[19] N. Foster and D. Metaxas. Realistic animation of liquids. *Graphical models and image processing: GMIP*, 58(5):471–483, 1996.

[20] N. Foster and D. Metaxas. Controlling fluid animation. In *Proceedings CGI '97*, pages 178–188, 1997.

[21] B. Frohlich and P. Hanrahan. Two-handed direct manipulation on the responsive workbench. In *Symposium on Interactive 3D Graphics*, 1997.

[22] V. Girault and P. Raviart. Finite element methods for navier-stokes equations. theory and algorithms. *Springer-Verlag Series in Computational Mathematics no. 5*, 1991.

[23] F. Harlow and F. Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with a free surface. *The Physics of Fluids*, 8:2182–2189, 1965.

[24] V. Interrante and C. Grosch. Strategies for effectively visualizing 3d flow with volume LIC. In *IEEE Visualization*, pages 421–424, 1997.

[25] A. Iserles. *A First Course in the Numerical Analysis of Differential Equations.* Cambridge University Press, 1st edition, 1996.

[26] 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.

[27] T. Kato. On classical solutions of the two-dimensional non-stationary euler equation. *Archive for Rational Mechanics and Analysis*, 25:188–200, 1967.

[28] J. Kim and P. Moin. Application of a fractional-step method to incompressible navier-stokes equations. *Journal of Computational Physics*, 59(2):308–323, 1985.

[29] O.A. Ladyzhenskaya. *The Mathematical Theory of Viscous Incompressible Flow.* Gordon and Breach, 2nd english edition edition, 1969.

[30] L.D. Landau and E.M. Lifshitz. *Fluid Mechanics.* Pergamon, 1968.

[31] M.H. Langston. Lic movie examples. http://mrl.nyu.edu/˜harper.

[32] J.Z. Lou and R.D. Ferraro. A parallel incompressible flow solver package with a parallel multigrid elliptic kernel. In *Proceedings of the 1995 ACM/IEEE conference on Supercomputing (CDROM)*, page 66. ACM Press, 1995.

[33] J.E. Marsden and T.J.R. Hughes. *The Mathematical Foundations of Elasticity.* Dover, 1994.

[34] R.S. Montero and I.M. Llorente. Robust multigrid algorithms for the incompressible navier-stokes equations. NASA/CR-2000-210126, ICASE Report No. 2000-27, http://techreports.larc.nasa.gov/icase/2000/icase-2000-27.pdf, 2000.

[35] D.Q. Nguyen, R. Fedkiw, and H.W. Jensen. Physically based modeling and animation of fire. In *SIGGRAPH 2002, Computer Graphics Proceedings*, pages 721–728. ACM Press / ACM SIGGRAPH, 2002.

[36] C.S. Peskin. Heart simulation by an immersed boundary method with formal second-order accuracy and reduced numerical viscosity. *Proceedings of the International Conference on Theoretical and Applied Mechanics (ICTAM)*, 2001.

[37] C.S. Peskin. The immersed boundary method. *Acta Numerica*, 11:479–517, 2002.

[38] A. Quarteroni and L. Formaggi. Lecture notes de ingenierie mathematique: Calcul scientifique i. Available Online at http://ima.epfl.ch/cmcs/, 1998.

[39] A. Quarteroni, F. Saleri, and A. Veneziani. Factorization methods for the numerical approximation of navier-stokes equations. *Comput. Methods Appl. Mech Engrg.*, 188:505–526, 2000.

[40] C. Rezk-Salama, P. Hastreiter, T. Christian, and T. Ertl. Interactive exploration of volume line integral convolution based on 3D-texture mapping. In David Ebert, Markus Gross, and Bernd Hamann, editors, *IEEE Visualization '99*, pages 233–240, San Francisco, 1999.

[41] P. Sabella. A rendering algorithm for visualizing 3d scalar fields. *Computer Graphics*, 22(4):51–58, 1988.

[42] R. Temam. Sur l'approximation de la solution des equations de navier-stokes par la methode de pas fractionnaires. *Archive for Rational Mechanics and Analysis*, 33:377–385, 1969.

[43] R. Temam. *Numerical Analysis.* D. Reidel Publishing Co., 1st edition, 1973.

[44] J.W. Thomas. *Numerical Partial Differential Equations.* Springer-Verlag New York, Inc, 1999.

[45] L.N. Trefethn and D. Bau. *Numerical Linear Algebra.* SIAM, 1st edition, 1997.

[46] C. Upson and M. Keeler. V-buffer: Visible volume rendering. *Computer Graphics*, 22(4):59–64, 1988.

[47] Various. Biovision bvh explanation. http://www.cs.wisc.edu/graphics/Courses/cs-838-1999/Jeff/BVH.html.

[48] E. Weinan and J. Liu. Projection method i: Convergence and numerical boundary layers. http://www.math.princeton.edu/ weinan/.

[49] E. Weinan and J. Liu. Projection method with spatial discretizations. http://www.math.princeton.edu/ weinan/.

[50] L. Ying, G. Biros, D. Zorin, and H. Langston. A new parallel kernel-independent fast multipole method. *Preprint*, 2003.