Numerical Simulation of Blood flow in Arteries

A Project Report Submitted
in Partial Fulfilment of the Requirements
for the Degree of

MASTER OF SCIENCE

in

MATHEMATICS

by

S Soundarya
(Roll No. IPHD20025)

to

SCHOOL OF MATHEMATICS
INDIAN INSTITUTE OF SCIENCE EDUCATION AND RESEARCH
THIRUVANANTHAPURAM - 695 551, INDIA

May 2023

DECLARATION

I, **S Soundarya (Roll No: IPHD20025)**, hereby declare that, this report entitled **"Numerical Simulation of Blood Flow in Arteries"** submitted to Indian Institute of Science Education and Research Thiruvananthapuram towards the partial requirement of **Master of Science** in **Mathematics**, is an original work carried out by me under the supervision of **Dr. Nagaiah Chamakuri** and has not formed the basis for the award of any degree or diploma, in this or any other institution or university. I have sincerely tried to uphold academic ethics and honesty. Whenever a piece of external information or statement or result is used then, that has been duly acknowledged and cited.

Thiruvananthapuram - 695 551 **S Soundarya**

May 2023

CERTIFICATE

This is to certify that the work contained in this project report entitled **'Numerical Simulation of Blood Flow in Arteries'** submitted by **S Soundarya** (**Roll No: IPHD20025**) to Indian Institute of Science Education and Research, Thiruvananthapuram towards the partial requirement of **Master of Science** in **Mathematics** has been carried out by her under my supervision and that it has not been submitted elsewhere for the award of any degree.

Thiruvananthapuram - 695 551

May 2023

Dr Nagaiah Chamakuri

Project Supervisor

ACKNOWLEDGEMENT

I express my profound gratitude and deep regard to Dr Nagaiah Chamakuri, IISER TVM, for his guidance, support, and constant encouragement throughout the project. I am grateful to Dr Aswin V S for his immense help in learning the DUNE software. I would like to express my sincere thanks to Mr Nishant Ranwan and all my other lab members for their valuable suggestions that have helped me all along my work. I would like to thank my parents, whose love and guidance are with me in whatever I pursue. Most importantly, my deepest appreciation towards my close friends Mr Sharang Iyer and Mr Aakash Gupta for their pivotal care and well wishes and for encouraging me every step of the way. I am also grateful to the Indian Institute of Science Education and Research Thiruvananthapuram for providing the necessary resources and facilities to complete this project to the best of my ability.

Thiruvananthapuram - 695 551 **S Soundarya**

May 2023

ABSTRACT

In the medium and large-sized arteries, we assume blood to be a homogeneous incompressible Newtonian fluid and can be modeled using the time-dependent Navier-Stokes equation. The convective term in the Navier-Stokes equation gives rise to non-linearity, making it difficult to solve analytically and numerically. Solving the Navier-Stokes equation through Finite Element Method(FEM) demands using a non-linear solver like the Newton solver due to the convective term. The Taylor-Hood finite element method is used for the spatial discretizations of the Navier-Stokes equations. The Theta schemes, particularly the Crank Nicholson scheme, have been used for the temporal discretization on a test example with a candy cane geometry [1] in two and three [2] dimensions using DUNE [2]. We have incorporated various boundary conditions in our numerical experiments, including i) zero do-nothing boundary conditions, ii) Navier-slip boundary conditions (for axi-parallel boundary) and iii) grad-div stabilization.

Raissi [3] introduced a general framework of Physics Informed Neural Networks (PINNs), neural networks that are trained to solve supervised learning tasks while respecting any given law of physics described by general nonlinear partial differential equations. The second part of my thesis is devoted to investigating a mixed-variable PINNs formulation as proposed in [4] for simulation of blood flow using the Navier Stokes Equation in a circular arch. The loss function given by the residuals of governing equation, along with the initial and boundary condition residuals, is minimized using the ADAM and LBFGS Optimization Algorithms. Numerical results using PINNs are compared qualitatively with the solution obtained through FEM. Though training with PINNs is computationally expensive and longer than the traditional FEM, testing it for various test cases takes less time. We have observed that the PINNs provide a promising mesh-free technique with results similar to those obtained using FEM.

Contents

Notations and Results

Let $\Omega \subset R^d$, d is the dimension, be an open,bounded Lipschitz domain with boundary Γ.

- Let $-\infty \leq a < b \leq \infty$,X a Banach space normed by $||\cdot||_X$.
  $L^p(a,b;X) := \{f : (a,b) \longrightarrow X| \quad t \longrightarrow ||f(t)||_X$ is strongly measurable $\}$
  $||f||_{L^P(a,b;X)} = (\int_\Omega ||f||_X^p)^{\frac{1}{p}} < \infty$, $1 \leq p < \infty$
  $||f||_{L^\infty(a,b;X)} = \sup\limits_{t\in(a,b)} \text{ess } ||f(t)||_X < \infty$, if $p = \infty$.
- $L_0^2(\Omega) = \{q : q \in L^2(\Omega) \quad \int_\Omega q(x)dx = 0\}$
- $H(div,\Omega) = \{\mathbf{v} \in L^2(\Omega) : \nabla \cdot \mathbf{v} \in L^2(\Omega)\}$
  $||\mathbf{v}||^2_{H(div,\Omega)} = ||\mathbf{v}||^2_{L^2(\Omega)} + ||\nabla \cdot \mathbf{v}||^2_{L^2(\Omega)}$
- If for $\mathbf{v} \in L^p(\Omega)$ with $p \geq 1$ there exists a function $\theta \in L^1_{loc}(\Omega)$ such that

$$\int_\Omega \nabla\psi \cdot \mathbf{v} d\mathbf{x} = \int_\Omega \psi\theta d\mathbf{x} \quad \forall\psi \in D(\Omega)$$

  then θ is called the weak divergence of $\mathbf{v}$.

- $\mathbf{v} \in L^p(\Omega)$ is called to be weakly divergence-free if $\int_\Omega \nabla\psi \cdot \mathbf{v} = \mathbf{0} \quad \forall\psi \in D(\Omega)$.
- Trace theoram Let Ω be of class $C^{m+1}$and $\Gamma = \partial\Omega$.Then there exists maps $\gamma_0, \gamma_1, ..., \gamma_{m-1}$ from $H^m(\Omega)$ into $L^2(\Gamma)$ such that
  (i) If $\mathbf{v} \in H^m(\Omega)$ is sufficiently smooth, then

$$\gamma_0(\mathbf{v}) = \mathbf{v}|_{\partial\Omega}, \gamma_1(\mathbf{v}) = \frac{\partial\mathbf{v}}{\partial\nu}|_{\partial\Omega}, ..., \gamma_{m-1}(\mathbf{v}) = \frac{\partial^{m-1}\mathbf{v}}{\partial\nu^{m-1}}|_{\partial\Omega}$$

where ν is the outward normal on $\partial\Omega$.

(ii) The range of the map $(\gamma_0, \gamma_1, ..., \gamma_{m-1})$ is $\prod_{j=0}^{m-1} H^{m-j-\frac{1}{2}}(\partial\Omega)$.

(iii)The kernel of the map $(\gamma_0, \gamma_1, .., \gamma_{m-1})$ is the space $H_0^m(\Omega)$.

- For $s > 0$, $H^s(\Omega) := \{\mathbf{v} \in L^2(\Omega) : ||(1 + |\zeta|^2)^{3/2}\hat{v}(\zeta)||_{L^2(\Omega)} < \infty\}$
  $||v||_{H^s(\Omega)} := ||(1 + |\zeta|^2)^{3/2}\hat{v}(\zeta)||_{L^2(\Omega)}$.

- Let $m \geq 1$ and $p \in R$ with $1 \leq p \leq \infty$.Then the following imbeddings hold:

$$W^{m,p}(\Omega) \subset \begin{cases} L^q(\Omega), & \frac{1}{q} = \frac{1}{p} - \frac{m}{n} > 0, \\ L^q_{loc}(\Omega) & \forall q \text{ with } 1 \leq p < \infty \text{ provided } \frac{1}{p} = \frac{m}{n}, \\ C^0(\overline{\Omega}), & \text{provided } \frac{1}{p} < \frac{m}{n}. \end{cases}$$

- A Gelfand Triplet is a pair of imbeddings $V \subset H = H^{'} \subset V^{'}$, where V is a seperable and reflexive Hilbert space, H is a seperable Hilbert space and V is dense in H and continuously imbedded in H. $H = H^{'}$ to be understood in the sense that dual space of H can be identified with H.
  A typical example is $V = H_0^1(\Omega) \quad H = L^2(\Omega) \quad V^{'} = H^{-1}(\Omega)$.

- $W(0,T) \equiv W(0,T;V,V') = \{v \in L^2(0,T;V) : \frac{dv}{dt} \in L^2(0,T;V')\}$ with the norm $||v||_{W(0,T)} = (\int_0^T ||v(t)||_V^2 + ||\frac{dv(t)}{dt}||_{V'}^2 dt)^{1/2}$.

- If $t \longmapsto u(t)$ is a vector-valued function on $[0,T]$ and if $\tilde{u}(t) = u(t)$ on $[0,T]$, $\quad \tilde{u}(t) = 0$ elsewhere, then, the Fourier transform $\tau \longmapsto \hat{\mathrm{u}}(\tau)$ is defined on $\mathbf{R}$ by

$$\hat{\mathrm{u}}(\tau) = \int_{\mathbb{R}} e^{-2i\pi t\tau} \tilde{u}(t) dt$$

- For $\gamma \in \mathbb{R}^+$,

$$\mathcal{H}^{\gamma}(0,T;\mathbf{V},\mathrm{H}) := \left\{ \mathrm{u} \in \mathrm{L}^2(0,\ \mathrm{T};\mathrm{V}); \tau \longmapsto |\tau|^{\gamma} \hat{\mathrm{u}}(\tau) \in \mathrm{L}^2(\mathbf{R};\mathrm{H}) \right\}$$

$$\|u\|_{\mathcal{H}^{\gamma}} = \left\{ \int_0^T \|u(t)\|^2 dt + \int_R |\tau|^{2\gamma} |\hat{u}(\tau)|^2 d\tau \right\}^{1/2}$$

- Grownwall's Inequality: m be an integrable a.e. positive function on $(0,T)$, $C \geq 0$ be a constant, $\phi \in C^0([0,T])$ satisfy

$$0 \leq \phi(t) \leq C + \int_0^t m(s)\phi(s) ds \quad \forall t \in [0,T]$$

  Then

$$\phi(t) \leq C \exp(\int_0^t m(s) ds) \quad \forall t \in [0,T]$$

  .

- Weak Sequential Compactness: Let E be a measurable set and $1 < p < \infty$. Then every bounded sequence in $L^p(E)$ has a subsequence that converges weakly in $L^p(E)$ to a function in $L^p(E)$.

- Helley's Theorem: Let X be a separable normed linear space and T_n be a sequence in the dual space X' that is bounded. Then there exists a subsequence T_{nk} of T_n in X' for which

$$\lim_{k\to\infty} T_{nk}(f) = T(f) \quad \forall \quad f \in X.$$

Chapter 1

Introduction

The cardiovascular system is a major system in the human body, consisting of the heart and blood vessels, whose purpose is to move blood through the body. The heart muscle acts as a pump in the system, expanding and contracting to receive and send blood, respectively. The blood vessels, including arteries, arterioles, capillaries, venules, and veins, are the piping components that allow for the distribution of blood through the body. Two major circulation paths make up the cardiovascular system (i) Pulmonary circulation, which carries blood between the heart and lungs, (ii) and systemic circulation, which carries blood from the heart through the body and back again. The purpose of the pulmonary circulation system is to take deoxygenated blood from the heart to the lungs, where carbon dioxide is released, oxygen is added, and return the blood to the heart. The systemic circulatory system begins in the left atrium, which receives oxygenated blood from the pulmonary veins. The blood then moves from the left atrium to the left ventricle, pumping it into the aorta, the largest artery in the body. From the aorta, the blood travels through the arteries

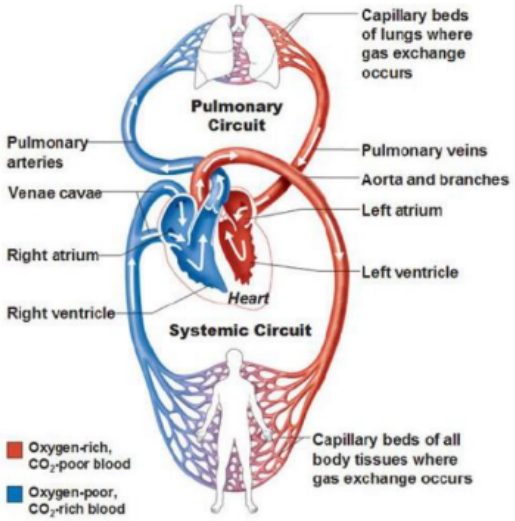


Figure 1.1: Cardiovascular System

into the capillaries and returns to the right atrium through the veins. The heart is continually expanding or contracting. This motion creates pulsatile blood flow, which moves the blood as pressure waves through the body. One complete cardiac cycle occurs during every heartbeat. Two important terms describe the motion of the heart chambers during the cycle – systole and diastole. Systole is the contraction of a chamber, while diastole is the relaxation of a chamber.

1.1 Blood Flow in Arteries

Blood is a concentrated suspension of several formed cellular elements, red blood cells, white blood cells, and platelets, in an aqueous polymeric and ionic solution, the plasma, composed of 93% water, and 3% particles like electrolytes, organic molecules,

numerous proteins, and waste products. The mechanical properties of blood should be studied by considering a fluid containing a suspension of particles. A fluid is said to be Newtonian if it satisfies Newton's law of viscosity (the shear stress is proportional to the rate of shear, and the viscosity is the constant of proportionality). Blood plasma, which consists mostly of water, is a Newtonian fluid. However, the whole blood has complex mechanical properties, which become particularly significant when the particle size is much larger, or at least comparable, with the lumen size. In this case, which happens at the microcirculation level (in the tiny arterioles and capillaries), blood cannot be modeled has a homogeneous fluid, and it is essential to consider it as a suspension of blood cells (specially RBCs) in plasma. Here we assume that all macroscopic length and time scales are sufficiently large compared to length and time scales at the level of the individual erythrocyte so that the continuum hypothesis holds.

1.1.1 Conservation of Mass Equation

Conservation of Mass Equation derived from the continuity equation is given by,

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

The material derivative is given by

$$\frac{D\rho}{Dt} = \frac{\partial \rho}{\partial t} + \nabla \rho \cdot \mathbf{u}$$

In an incompressible flow, there is no change in density over time (else that would imply the fluid had either compressed or expanded) which implies that the material

derivative of the density vanishes. So, the conservation of mass equation reduces to

$$\nabla \cdot \mathbf{u} = 0 \tag{1.1}$$

1.1.2 Conservation of Momentum Equation

The body force on a fluid parcel is due to two components of fluid stresses which are represented as the divergence of the stress tensor because the divergence is the extent to which the tensor acts like a sink or source, and other external forces like gravity. By Newton's Second Law,

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

The stress tensor σ is often divided into two terms of interest, firstly, the volumetric stress tensor, representing the force that tends to change the volume of the body (namely, the pressure forces), and secondly, the stress deviator tensor, representing the forces which determine body deformation and movement (the shear stress tensor).

$$\sigma = \begin{pmatrix} \sigma_{xx} & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \sigma_{yy} & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma_{zz} \end{pmatrix} = - \begin{pmatrix} p & 0 & 0 \\ 0 & p & 0 \\ 0 & 0 & p \end{pmatrix} + \begin{pmatrix} \sigma_{xx} + p & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \sigma_{yy} + p & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma_{zz} + p \end{pmatrix}$$

Substituting,

$$\sigma = -pI + T$$

We get the most general form of the Navier-Stokes Equation into the equation obtained from Newton's Second Law.

$$\rho\frac{D\mathbf{u}}{Dt} = -\nabla p + \nabla \cdot T + \mathbf{f}$$

For a Newtonian Fluid, the stress deviator tensor T is proportional to the symmetric part of the velocity gradient,

$$T_{ij} = \mu(\frac{\partial u_i}{\partial u_j} + \frac{\partial u_j}{\partial u_i}) \qquad T = 2\mu D$$

where μ is the molecular viscosity of the fluid and $D = \frac{(\nabla \mathbf{u} + \nabla \mathbf{u^T})}{2}$. Hence after substituting the above to the general form, we get the Navier Stokes Equation for incompressible Newtonian Fluids.

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u}\mathbf{u^T}) = \nabla \cdot \mathbb{S} + \mathbf{f} \qquad \mathbb{S} = \mu(\nabla \mathbf{u} + \nabla \mathbf{u^T}) - pI, \tag{1.2}$$

$$\begin{aligned} \rho\frac{\partial \mathbf{u}}{\partial t} + \rho(\nabla \mathbf{u})\mathbf{u} &= -\nabla p + \mu\Delta \mathbf{u} + \mathbf{f} \\ \frac{\partial \mathbf{u}}{\partial t} - \nu\Delta \mathbf{u} + (\mathbf{u} \cdot \nabla)\mathbf{u} + \nabla p &= \mathbf{f} \end{aligned} \tag{1.3}$$

where $\nu := \frac{\mu}{\rho}$ is the kinematic viscosity and the constant density is absorbed into the pressure. All the above forms are equivalent. The conservation of mass equation and the conservation of momentum equation constitute the Navier-Stokes equation. The term $(\mathbf{u} \cdot \nabla)\mathbf{u}$ describes the process of convective transport, $(\nu\Delta \mathbf{u})$ represents the molecular diffusion.Also $\nabla \cdot S = \nu\Delta \mathbf{u} - \nabla p = \nabla \cdot (\nu\nabla \mathbf{u} - pI)$.

Chapter 2

FEM for Navier-Stokes Equation

2.1 Stokes Equation

If $f \in \mathbf{H^{-1}}(\Omega)$ the Stokes problem is given by

$$-\nu\Delta\mathbf{u} + \nabla p = \mathbf{f} \text{ in } \Omega$$

$$\nabla \cdot \mathbf{u} = 0 \text{ in } \Omega$$

If $V = \mathbf{H^1_{0,\Gamma_D}}(\mathbf{\Omega})$ the divergence operator is defined as div $: V \longrightarrow$ range(div), $\mathbf{v} \longrightarrow \nabla \cdot \mathbf{v}$ If $\mathbf{v} \in V$ then integrating by parts we get $\int_\Omega \nabla \cdot \mathbf{v} = 0 \quad \mathbf{v} \in V$ such that $\nabla \cdot \mathbf{v} \in L^2_0(\Omega)$, range(div) $\subset Q = Q'$

The Gradient Operator defined on $Q = L^2_0(\Omega)$ is given by, grad $: Q \longrightarrow$ range(grad) $q \longrightarrow \nabla q$. Since gradient of a function from $L^2(\Omega)$ is in $\mathbf{H^{-1}}(\mathbf{\Omega})$ one obtains that

the range(grad)$\subset V'$.

Isomorphism of the Gradient and divergence operator

Let $V_{div} = \{\mathbf{v} \in V : (\nabla \cdot \mathbf{v}, q) = 0 \quad \forall q \in Q\}$. If $\mathbf{f} \in V'$ satisfies $< \mathbf{f}, \mathbf{v} >_{V',V} = 0 \quad \forall \mathbf{v} \in V_{div}$, then there exists a unique $q \in Q$ such that $\mathbf{f} = \text{grad}(q)$. That means the range of the gradient operator consists of functionals in V' that vanish on V_{div},

$$\tilde{V}' = \{\mathbf{f} \in V' :< \mathbf{f}, \mathbf{v} >_{V',V} = 0 \quad \forall \mathbf{v} \in V_{div}\}.$$

Hence gradient operator is an isomorphism from Q onto $\tilde{V}'$. Also, the divergence operator is an isomorphism from $V_{div}^{\perp}$ onto Q'.

From the theory of Linear Saddle point problems, because of the above isomorphism, we can say that the inf-sup conditions hold for the Stokes problem given by

$$\underset{0 \neq q \in Q}{inf} \underset{0 \neq \mathbf{v} \in V}{sup} \frac{(\nabla \cdot \mathbf{v}, q)}{||\nabla \mathbf{v}||_V ||q||_Q} \geq \beta_{is}$$

for some $\beta_{is} > 0$. Also, the coercivity condition holds $\forall v \in V$.

The coercivity and the *inf-sup* condition is necessary and sufficient conditions for the well-posedness of the linear saddle point problem. Hence Stokes's problem is well-posed.

2.1.1 Navier Stokes Equation

Lemma 1. Let X be a given Banach space with dual X' and let u, g be two functions $\in L^1(a, b; X)$. Then the following three conditions are equivalent:

(i)u is a.e. equal to a primitive function of g,

$$u(t) = \zeta + \int_a^t g(s)ds \text{a.e. } t \in (a,b)$$

(ii)For each test function $\phi \in D(a,b)$, we have,

$$\int_a^b u(t)\phi^{'}(t)dt = -\int_a^b g(t)\phi(t)dt$$

(iii)For each $\eta \in X'$,

$$\frac{d}{dt} < u, \eta >_X = < g, \eta >_X \text{ in } D'((a,b))$$

If one of the above items is satisfied, then u, in particular, is a.e. equal to a continuous function from [a, b] into X.

The function g of the above lemma is called the weak derivative of u and is usually denoted by $g = u' = \frac{du}{dt}$.

Lemma 2. $W(0,T)$ is a Hilbert space. If $v \in W(0,T)$ then it coincides almost everywhere with a function of $C^0([0,T];V')$ by the Lemma 1. In fact, it coincides, a.e. with a function in $C^0([0,T];H)$. Also the following Green's theorem holds for all $u, v \in W(0,T)$:

$$\int_0^T \left\{ < \frac{du}{dt}(t), v(t) > + < u(t), \frac{dv}{dt}(t) > \right\} dt = (u(T), v(T)) - (u(0), v(0))$$

Lemma 3. If the imbedding of V into H is compact, then the canonical imbedding of $\mathcal{H}^\gamma(0,T;V,H)$ into $L^2(0,T;H)$ is also compact.

The Navier-Stokes equations for an incompressible Newtonian fluid with Dirichlet Boundary Condition are given by

$$\mathbf{Q}\begin{cases} \frac{\partial \mathbf{u}}{\partial t} - \nu\Delta\mathbf{u} + \sum_{j=1}^{n} u_j \frac{\partial \mathbf{u}}{\partial x_j} - \nabla p = \mathbf{f} \\ \nabla \cdot \mathbf{u} = 0 \quad (x,t) \in \Omega \text{ x } R^+ \\ \mathbf{u} = \mathbf{0} \quad (x,t) \in \Gamma \text{ x } R^+ \\ \mathbf{u(0)} = \mathbf{u_o} \quad x \in \Omega \end{cases}$$

where $\mathbf{f}, \mathbf{u_o}$ are two prescribed functions. Consider the function spaces

$$\mathcal{V} = \{\mathbf{v} \in (D(\Omega)) : \nabla \cdot \mathbf{v} = \mathbf{0}\}$$
$$V = \{\mathbf{v} \in \mathbf{H_0^1(\Omega)} : \nabla \cdot \mathbf{v} = \mathbf{0}\}$$
$$H = \{\mathbf{v} \in (L^2(\Omega)) : \nabla \cdot \mathbf{v} = \mathbf{0}, \gamma_o(\mathbf{v}) = \mathbf{0}\}$$

$\mathcal{V}$ is dense in H, V and the spaces H, V are examples of Gelfand triplets.

If $\mathbf{u}, \mathbf{v}, \mathbf{w} \in H^1(\Omega)^n$ $a_o(\mathbf{u}, \mathbf{v}) := (\nabla u, \nabla v)$ $\quad a_1 := (\mathbf{w}, \mathbf{u}, \mathbf{v}) = \sum_{i,j=1}^{n} \int_\Omega w_j \frac{\partial u_i}{\partial x_j} v_i$
$a(\mathbf{w}; \mathbf{u}, \mathbf{v}) = \nu a_o(\mathbf{u}, \mathbf{v}) + a_1(\mathbf{w}; \mathbf{u}, \mathbf{v})$
If $\mathbf{u}, \mathbf{w} \in L^2(0, T; V)$ $A_o : L^2(0, T; V) \longrightarrow L^2(0, T; V')$

$$< A_o\mathbf{u}(t), \mathbf{v} >:= a_o(\mathbf{u}(t), \mathbf{v}) \quad \forall \mathbf{v} \in V$$

$t \longmapsto A_1(\mathbf{w(t)}, \mathbf{u}(t))$ defined a.e. on [0,T] by

$$A_1(\mathbf{w(t)}, \mathbf{u}(t)) \in V', < A_1(\mathbf{w(t)}, \mathbf{u}(t)), \mathbf{v}(t) >= a_1(\mathbf{w(t)}; \mathbf{u}(t), \mathbf{v}(t)) \quad \forall \mathbf{v} \in V.$$

The function $t \longmapsto A_1(\mathbf{w}(\mathbf{t}), \mathbf{u}(t))$ is measurable on $(0, T)$.

Lemma 4. When $n = 2$, all elements $\phi \in H_0^1(\Omega)$ satisfy

$$||\phi||_{L^4(\Omega)} < 2^{1/4} |\phi|^{1/2}_{H^1(\Omega)} ||\phi||^{1/2}_{L^2(\Omega)}$$

Lemma 5. When $\mathbf{w}, \mathbf{u}$ both belong to $L^2(0, T; V) \cap L^\infty(0, T; H)$ then

$$A_1(\mathbf{w}, \mathbf{u}) \in \begin{cases} L^2(0, T; V') & n = 2 \\ L^{4/3}(0, T, V') & n = 3 \end{cases}$$

If $u \in L^2(0, T; V) \cap L^\infty(0, T; H)$ then

$$||A_1(\mathbf{u}, \mathbf{u})||_{L^{4/3}(0,T;V')} \leq C ||\mathbf{u}||^{1/2}_{L^\infty(0,T;H)} ||\mathbf{u}||^{3/2}_{L^2(0,T;V)} \quad n = 3$$

$$||A_1(\mathbf{u}, \mathbf{u})||_{L^2(0,T;V')} \leq C ||\mathbf{u}||_{L^\infty(0,T;H)} ||\mathbf{u}||_{L^2(0,T;V)} \quad n = 2$$

Variational Problem (P) Multiplying the first equation in problem **Q** by a test function $\mathbf{v} \in V$ and integrating, along with the divergence condition in the second equation of **Q**, we get

For a given function $\mathbf{f} \in L^2(0, T; (H^{-1}(\Omega)))$ $\quad \mathbf{u_o} \in H$, find

$\mathbf{u} \in L^2(0, T; V) \cap L^\infty(0, T; H)$ such that

$$\mathbf{(P)} \begin{cases} \frac{d}{dt}(\mathbf{u}(t), \mathbf{v}) + a(\mathbf{u}(t); \mathbf{u}(t), \mathbf{v}) =< \mathbf{f}(\mathbf{t}), \mathbf{v} >, \quad \forall \mathbf{v} \in V \end{cases}$$

with initial condition $\mathbf{u}(\mathbf{0}) = \mathbf{u_o}$.

When $n = 2$ $\mathbf{u} \in W(0,T)$ by Lemma 2, $\mathbf{u} \in C^o([0,T];H)$ and $n = 3$, by Lemma 1, $\mathbf{u} \in C^o(0,T;V)$. So $\mathbf{u}(t)$ is continuous at $t = 0$ and we can prescribe its value at $t = 0$.

Restating the problem **P** For a given function $\mathbf{f} \in L^2(0,T;(H^{-1}(\Omega))), \mathbf{u_o} \in H$, find $\mathbf{u} \in L^2(0,T;V) \cap L^\infty(0,T;H)$ and conditions in Lemma 6 such that

$$\mathbf{(P)} \left\{ \frac{d\mathbf{u}}{dt} + \nu A_o(\mathbf{u}) + A_1(\mathbf{u},\mathbf{u}) = \mathbf{f} \right.$$

with initial condition $\mathbf{u}(\mathbf{0}) = \mathbf{u_o}$.

Equivalence of problem P and Q.

Clearly, if $(\mathbf{u}, p)$ is a solution of **Q**, then it is a solution of **P**.

Conversely, let $\mathbf{u} \in L^2(0,T;V) \cap L^\infty(0,T;H)$ be a solution of (P) and consider the mapping defined on $H_0^1(\Omega)$ by

$$L(\mathbf{v},t) : \mathbf{v} \longmapsto \int_0^t < \mathbf{f}(s), \mathbf{v} > -a(\mathbf{u}(s), \mathbf{u}(s), \mathbf{v})ds - (\mathbf{u}(t), \mathbf{v}) + (\mathbf{u}_o, \mathbf{v})$$

For each t, L is a linear functional on $H_0^1(\Omega)$ that vanishes on V. Hence, by isomorphism, for each $t \in R^+$ and each $L(t)$ there exists exactly one function $P(t) \in L_0^2(\Omega)$, such that

$$L(\mathbf{v},t) = - < \nabla P(t), \mathbf{v} > = < P(t), \nabla \cdot \mathbf{v} > \quad \forall \mathbf{v} \in H_0^1(\Omega)$$

In other words,

$$(P(t), \nabla \cdot \mathbf{v}) = \int_0^t < \mathbf{f}(s), \mathbf{v} > -a(\mathbf{u}(s), \mathbf{u}(s), \mathbf{v})ds - (\mathbf{u}(t), \mathbf{v}) + (\mathbf{u}_o, \mathbf{v})$$

By Lemma 5, $P \in C^o([0,T]; L_0^2(\Omega))$. By differentiating, we get,

$$< \frac{dP}{dt}(t), \nabla \cdot \mathbf{v} >=< \mathbf{f}(\mathbf{t}), \mathbf{v} > -a(\mathbf{u}(t); \mathbf{u}(t), \mathbf{v}) - < \frac{d\mathbf{u}(t)}{dt}, \mathbf{v} > \quad \forall \mathbf{v} \in H_0^1(\Omega)$$

Therefore, set the pressure as $p = \frac{dP}{dt}$, we get the form $\mathbf{Q}$.

Existence and Uniqueness of the solution

In order to prove that problem P has a solution, we propose to construct first a sequence (P_m) of semi-discrete problems, each of which has a unique solution. Then, by means of a priori estimates, we show that some subsequence of these solutions converges towards a function that satisfies (P).

Since V is seperable, we can find a basis $< \mathbf{w}_m >_{m \geq 1}$ of V such that its span is dense in V and denote by V_m the space spanned by the set $\{\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3, .., \mathbf{w}_m\}$.Replace the problem $\mathbf{P}$ by the following problem in V_mx$[0,T]$.

Find a function $\mathbf{u}_m(t)$ of the form $\mathbf{u}_m(t) = \sum_{j=1}^m g_{jm}(t)\mathbf{w}_j$ satisfying the initial value of the following differential equation:

$$(\mathbf{P_m}) \begin{cases} \frac{d}{dt}(\mathbf{u}_m(t), \mathbf{w}_i) + a(\mathbf{u}_m(t); \mathbf{u}_m(t), \mathbf{w}_i) =< \mathbf{f}, \mathbf{w}_i > \quad 1 \leq i \leq m \\ \mathbf{u}_m(0) = \mathbf{u}_{0m} \in V_m. \end{cases}$$

The starting value u_{0m} is chosen such that $lim_{m \to \infty} \mathbf{u}_{0m} = \mathbf{u}_0$ in H.

Overview of the steps:

Step 1. Problem $(\mathbf{P_m})$ has a unique solution $\mathbf{u}_m \in L^\infty(0,T;H) \cap L^2(0,T;V)$. Moreover, $||\mathbf{u}_m||_{L^\infty(0,T;H)} + ||\mathbf{u}_m||_{L^2(0,T;V)} \leq C$ where C is a constant independent of m.

The problem $(\mathbf{P_m})$ in terms of its unknowns $g_{jm}(t)$:

$$\sum_{j=1}^{m}(\mathbf{w}_j, \mathbf{w}_i)\frac{d}{dt}g_{jm}(t) + \sum_{j=1}^{m}\sum_{k=1}^{m} a(\mathbf{w}_j; \mathbf{w}_k, \mathbf{w}_i)g_{jm}(t)g_{km}(t) =< \mathbf{f(t)}, \mathbf{w}_i > \quad 1 \leq i \leq m$$

with the starting value $g_{jm}(0) = g_{jm}^0 \quad \text{for} \quad 1 \leq j \leq m$, where g_{jm}^o are the coefficients of $\vec{u}_{om}$. Inverting the nonsingular matrix with elements $(\mathbf{w}_i, \mathbf{w}_j), 1 \leq i, j \leq m$ we get,

$$\frac{d}{dt}g_{im}(t) = \phi_i(t; g_{1m}(t), \ldots, g_{mm(t)}) \quad 1 \leq i \leq m$$

Now, according to Carathéodory's theorem, the above system has a local maximal solution $\mathbf{u}_m(t)$ in an interval $(0, t_m)$.If $t_m < T$ then $\lim_{t \to t_{m^-}} |\mathbf{u}_m(t)| = \infty$. If we show $\mathbf{u}_m(t)$ is bounded independent of m, t then $t_m = T$.

Step 2. Taking $\mathbf{u}, \mathbf{v}, \mathbf{w} = \mathbf{u}_m(t)$ we get $a_1(u_m(t); u_m(t), u_m(t)) = 0$. Hence the weak formulation becomes

$$(\frac{d}{dt}\mathbf{u}_{m(t)}, \mathbf{u}_m(t)) + \nu||\mathbf{u}_m(t)||^2 =< f(t), \mathbf{u}_m(t) >$$

.Integrate both sides we get

$$\frac{1}{2}|\mathbf{u}_m(t)|^2 - \frac{1}{2}|\mathbf{u}_m(0)|^2 + \nu\int_0^t ||\mathbf{u}_m(s)||^2 ds = \int_0^t < \mathbf{f(s)}, \mathbf{u}_m(s) > ds$$

By Cauchy's inequality, we have

$$|\int_0^t < \mathbf{f}(\mathbf{s}), \mathbf{u}_m(s) > ds| \leq \frac{\epsilon}{2}\int_0^t ||\mathbf{u}_m(s)||_V^2 + \frac{1}{2\epsilon}\int_0^t ||\mathbf{f}(\mathbf{s})||_{V'}^2 ds$$

Choose $\epsilon = 2\nu$ we get

$$|\mathbf{u}_m(t)|^2 \leq |\mathbf{u}_{om}|^2 + \frac{1}{2\nu}\int_0^t ||\mathbf{f}(\mathbf{s})||_{V'}^2 ds$$

which is bounded independent of m, t. Hence $\mathbf{u}_m$ is bounded in $L^\infty(0,T;H)$.
Choose $\epsilon = \nu, t = T$ we get

$$|\mathbf{u}_m(T)|^2 + \nu\int_0^T ||\mathbf{u}_m(t)||^2 dt \leq |\mathbf{u}_{om}|^2 + \frac{1}{\nu}\int_0^T ||\mathbf{f}(\mathbf{t})||_{V'}^2 dt$$

Hence $\mathbf{u}_m$ is bounded in $L^2(0,T,V)$. This gives that the sequence $\mathbf{u}_m$ is bounded in $L^\infty(0,T;H) \cap L^2(0,T;V)$.
The uniqueness of $\mathbf{u}_m$ can also be proved. Hence $\mathbf{u}_m$ is a solution of the problem P_m.

Step 3. The sequence $(\mathbf{u}_m)$ is bounded in $H^\gamma(0,T;V,H)$ for $0 < \gamma < \frac{1}{4}$. Hence there exists a subsequence $(\mathbf{u}_\mu)$ of $(\mathbf{u}_m)$ such that

$$\lim_{\mu\to\infty} \mathbf{u}_\mu = \mathbf{u} \text{ weakly in } L^2(0,T;V)$$

$$\lim_{\mu\to\infty} \mathbf{u}_\mu = \mathbf{u} \text{ weak star in } L^\infty(0,T;H)$$

by the Weak Sequential Compactness and Helly's theorem. The above statements mean that

$$\int_0^T (\mathbf{u}_\mu, \mathbf{v}(t))dt \longrightarrow \int_0^T (\mathbf{u}, \mathbf{v}(t))dt \quad \forall \mathbf{v} \in L^2(0,T,V')$$

$$\int_0^T (\mathbf{u}_\mu, \mathbf{v}(t))dt \longrightarrow \int_0^T (\mathbf{u}, \mathbf{v}(t))dt \quad \forall \mathbf{v} \in L^1(0,T,H)$$

The first convergence also holds for all $\mathbf{v} \in L^2(0,T,H)$. Hence $\mathbf{u} \in L^2(0,T,V) \cap L^\infty(0,T,H)$.

Since the subsequence is bounded in $H^\gamma(0,T;V,H)$ for $0 < \gamma < \frac{1}{4}$, by Lemma 3 we have

$$\lim_{\mu\to\infty} \mathbf{u}_\mu = \mathbf{u}$$

in $L^2(0,T;H)$.

Step 4. Take $\psi \in C^1([0,T])$ with $\psi(T) = 0$ multiply in P_m and integrate over $[0,T]$ we get

$$-\int_0^T (\mathbf{u}_m, \mathbf{w}_i)\psi'(t)dt + \int_0^T a(\mathbf{u}_m(t); \mathbf{u}_m(t), \mathbf{w}_i)\psi(t)dt = \int_0^T < \mathbf{f}(\mathbf{t}), \mathbf{v} > \psi(t)dt - (\mathbf{u}_{0\mu}, \mathbf{v})\psi(0)$$

Fix an arbitrary integer k and let $\mathbf{v} \in V_k$. Then the above holds for all $\mathbf{v} \in V_k$. By the convergences in Step 3, the following limits hold

$$lim_{\mu\to\infty} \int_0^T (\mathbf{u}_\mu(t), \mathbf{v})\psi'(t)dt = \int_0^T (\mathbf{u}(t), \mathbf{v})\psi'(t)dt$$

$$lim_{\mu\to\infty} \int_0^T a_o(\mathbf{u}_\mu(t), \mathbf{v})\psi(t)dt = \int_0^T a_o(\mathbf{u}(t), \mathbf{v})\psi(t)dt$$

Convergence of the convective term

$$\int_0^T a_1(\mathbf{u}_\mu(t); \mathbf{u}_\mu(t), \mathbf{v})\psi(t)dt = -\int_0^T a_1(\mathbf{u}_\mu(t); \mathbf{v}, \mathbf{u}_\mu(t))\psi(t)dt$$

$$= -\int_0^T [\sum_{i,j=1}^{n} \int_\Omega (\mathbf{u}_\mu)_j \frac{\partial \mathbf{v}_i}{\partial x_j}(\mathbf{u}_\mu)_i \psi(t)] dx dt$$

The weak convergence and strong convergence from step 3 in $L^2(0,T,V'), L^2(0,T,H)$ respectively assert that

$$\lim_{\mu\to\infty} \int_0^T a_1(\mathbf{u}_\mu(t); \mathbf{u}_\mu(t), \mathbf{v})\psi(t)dt = \int_0^T a_1(\mathbf{u}(t); \mathbf{u}(t), \mathbf{v})\psi(t)dt$$

The above convergences will enable us to pass the limit in the problem $(\mathbf{P}_\mu)$ we get $\mathbf{u}$ to be the solution and restricting ψ to $D(0,T)$, $\mathbf{u}$ satisfies the problem $\mathbf{P}$ in the distributional sense with the initial condition satisfied.

Uniqueness of the solution in Two Dimensions

There is exactly one weak solution of the Navier-Stokes equations in two dimensions. Proof: Let $\mathbf{u}_1, \mathbf{u}_2$ be two solutions of problem $\mathbf{Q}$. Substituting these in the second energy inequality from step 2, we get

$$\frac{1}{2}\frac{d}{dt}||\mathbf{u}_1 - \mathbf{u}_2||^2_{L^2(\Omega)} + \nu||\nabla(\mathbf{u}_1 - \mathbf{u}_2)||^2_{L^2(\Omega)} = a_1(\mathbf{u}_2, \mathbf{u}_2, \mathbf{u}_1 - \mathbf{u}_2) - a_1(\mathbf{u}_1, \mathbf{u}_1, \mathbf{u}_1 - \mathbf{u}_2)$$

We have the sobolev embedding theoram, if $mp < d$,$W^{m,p}(\Omega) \to L^q(\Omega)$ for $1 \le q \le \frac{dp}{d-mp}$. Hence with $d = 2, m = \frac{1}{2}, p = 2$, we have the embedding, $H^{\frac{1}{2}}(\Omega) \to L^4(\Omega)$ and the Young's inequality with $p = \frac{4}{3}, q = 4$ yields,

$$a_1(\mathbf{u}_2, \mathbf{u}_2, \mathbf{u}_1 - \mathbf{u}_2) - a_1(\mathbf{u}_1, \mathbf{u}_1, \mathbf{u}_1 - \mathbf{u}_2) \le \frac{\nu}{2}||\nabla(\mathbf{u}_1 - \mathbf{u}_2)||^2 + C||\mathbf{u}_2||^4_{L^4(\Omega)}||\mathbf{u}_1 - \mathbf{u}_2||^2_{L^2(\Omega)}$$

Hence we have,

$$\frac{1}{2}\frac{d}{dt}||\mathbf{u}_1 - \mathbf{u}_2||^2_{L^2(\Omega)} + \nu||\nabla(\mathbf{u}_1 - \mathbf{u}_2)||^2_{L^2(\Omega)} \leq C||\mathbf{u}_2||^4_{L^4(\Omega)}||\mathbf{u}_1 - \mathbf{u}_2||^2_{L^2(\Omega)}$$

The second term of the left can be estimated with zero from below, and Grownwall's lemma yields for all times,

$$||(\mathbf{u}_1 - \mathbf{u}_2)(t)||^2_{L^2_\Omega} \leq exp(C||\mathbf{u}_2||^4_{L^4(0,T;L^4(\Omega))})||(\mathbf{u}_1 - \mathbf{u}_2)(0)||^2_{L^2(\Omega)} = 0$$

Since both solutions satisfy the same initial conditions,$\mathbf{u}_1 = \mathbf{u}_2$ in $L^2(0,T;V) \cap L^\infty(0,T;H)$.

Uniqueness in three dimensions In three dimensions, one can prove that there exists a unique solution of the Navier–Stokes equations with higher regularity and in a time interval $(0;T'), T' > 0$, if the data $\mathbf{u}_0$ and $\mathbf{f}$ are sufficiently small. The time T' depends on the size of the data. In case of steady state Navier Stokes we require that $\frac{N}{\nu}^2||\mathbf{f}||_{V^*} < 1$, where $N = \sup_{\mathbf{u},\mathbf{v},\mathbf{w}\in V} \frac{|a_1(\mathbf{w};\mathbf{u},\mathbf{v})|}{|\mathbf{u}|_{H^1}|\mathbf{v}|_{H^1}|\mathbf{w}|_{H^1}}$.

2.1.2 Energy balance and Different Boundary Condition

Weak formulation of Navier Stokes: A weak formulation of the Navier Stokes Equation can be derived by multiplying by test functions $(\mathbf{v}, q)$ and integrating it over Ω

$$\int_\Omega \frac{\partial \mathbf{u}}{\partial t} \cdot \mathbf{v} d\Omega - \int_\Omega \nu\Delta\mathbf{u} \cdot \mathbf{v} d\Omega + \int_\Omega (\mathbf{u} \cdot \nabla)\mathbf{u} \cdot \mathbf{v} d\Omega + \int_\Omega \nabla p \cdot \mathbf{v} d\Omega = \int_\Omega \mathbf{f} \cdot \mathbf{v} d\Omega \,.$$

Using Green's theorem, we obtain,

$$\int_\Omega \frac{\partial \mathbf{u}}{\partial t}\cdot\mathbf{v}d\Omega+\int_\Omega \nu\nabla\mathbf{u}\cdot\nabla\mathbf{v}d\Omega+\int_\Omega(\mathbf{u}\cdot\nabla)\mathbf{u}\cdot\mathbf{v}d\Omega-\int_\Omega p\nabla\cdot\mathbf{v}d\Omega=\int_\Omega \mathbf{f}\cdot\mathbf{v}d\Omega+\int_{d\Omega}(\nu\frac{\partial\mathbf{u}}{\partial n}-p\mathbf{n})\cdot\mathbf{v}d\gamma$$

$$\int_\Omega q\nabla\cdot\mathbf{u}d\Omega=0 \qquad \forall \quad (\mathbf{v},q)$$

Pressure space for different Boundary Conditions

If $\partial\Omega=\Gamma_D$,*i.e.*, we are imposing essential boundary condition on all of $\partial\Omega$ and $\mathbf{u}=\mathbf{g}$ on $\partial\Omega$, with the test function $q=1$ in the continuity equation we get,

$$0=\int_\Omega(\nabla\cdot\mathbf{u})=\int_{\Gamma_D}\mathbf{u}\hat{n}dS=\int_{\Gamma_D}\mathbf{g}\hat{n}dS$$

Also, there is no condition on the pressure term. Hence, if p is a solution, then $p+c$ is also a solution. To avoid such indeterminacy, one could fix a priori the value of p at one given point x_o in Ω that is set $p(x_o)=p_o$, or alternatively, to have null average, that is, $p\in L^2_0(\Omega)$.

Energy Balance: If the solution to the above variational problem exists and is regular enough, we perform an energy balance. We multiply the momentum equation by $\mathbf{u}$ and perform integration by parts to obtain:

$$\rho\int_\Omega\frac{\partial\mathbf{u}}{\partial t}\cdot\mathbf{u}+\rho\int_\Omega(\mathbf{u}\cdot\nabla\mathbf{u})\cdot\mathbf{u}+\nu\int_\Omega|\nabla\mathbf{u}|^2+\text{ boundary terms}=0$$

which can rewritten as

$$\rho \int_\Omega \frac{1}{2}\frac{\partial |\mathbf{u}|^2}{\partial t} + \rho \int_\Omega (\mathbf{u}\cdot\nabla\mathbf{u})\cdot\mathbf{u} + \nu ||\nabla \mathbf{u}||^2_{L^2(\Omega)} + \text{boundary terms} = 0$$

Since the domain is rigid, it does not depend on time, and thus,

$$\int_\Omega \frac{1}{2}\frac{\partial |\mathbf{u}|^2}{\partial t} = \frac{1}{2}\frac{d}{dt}\int_\Omega |\mathbf{u}|^2 = \frac{1}{2}\frac{d}{dt}||\mathbf{u}(\cdot,t)||^2_{L^2(\Omega)}$$

Moreover, we integrate by parts the convective term

$$\int_\Omega (\mathbf{u}\cdot\nabla\mathbf{u})\cdot\mathbf{u} = \int_\Omega (\sum_i \mathbf{u}_i\partial_i\mathbf{u})\cdot\mathbf{u} = \int_\Omega \sum_j \mathbf{u}_j \sum_i \mathbf{u}_i\partial_i\mathbf{u}_j = \frac{1}{2}\int_\Omega \sum_i \mathbf{u}_i\partial_i|\mathbf{u}|^2 = \int_\Omega \mathbf{u}\cdot\nabla\frac{|\mathbf{u}|^2}{2}$$

$$\int_\Omega (\mathbf{u}\cdot\nabla\mathbf{u})\cdot\mathbf{u} = \int_{\partial\Omega} \frac{|\mathbf{u}|^2}{2}\mathbf{u}\cdot\mathbf{n} - \int_\Omega \frac{|\mathbf{u}|^2}{2}\nabla\cdot\mathbf{u}$$

Since the velocity divergence is zero, denoting $E(t) = \frac{\rho}{2}||\mathbf{u}(\cdot,t)||^2_{L^2(\Omega)}$,the fluid kinetic energy, we have,

$$\frac{d}{dt}E(t) = -\rho\int_{\partial\Omega}\frac{|\mathbf{u}|^2}{2}\mathbf{u}\cdot\mathbf{n} - \nu||\nabla\mathbf{u}||^2_{L^2(\Omega)} + \text{boundary terms}$$

The terms of the right-hand side of the above equation represent the flux of kinetic energy entering/exiting and the rate of variation of kinetic energy through the power dissipated by the fluid viscosity.

Essential Boundary Conditions In the case when $\partial\Omega = \Gamma_D$, it remains only,

$$\frac{d}{dt}E(t) = -\nu||\nabla\mathbf{u}||^2_{L^2(\Omega)}$$

Hence, in this case, the kinetic energy flux is zero, which allows us to get an energy balance. Integrating on $(0, T)$ gives,

$$E(T) = E(0) - \nu \int_0^T ||\nabla \mathbf{u}||^2_{L^2(\Omega)}$$

which gives that $E(t)$ is bounded over $(0, T)$ hence, $\mathbf{u} \in L^\infty(0, T; L^2(\Omega))$. Since the dissipated energy is bounded over $(0, T)$, we have $\mathbf{u} \in L^2(0, T; H^1_0(\Omega))$. Thus, $\mathbf{u} \in L^\infty(0, T; L^2(\Omega)) \cap L^2(0, T; H^1_0(\Omega))$. In a 2-dimensional evolutional case, the uniqueness of a weak solution on any time interval $[0, T]$ yields for the Navier–Stokes system with Dirichlet's boundary data. In a 3-dimensional evolutional case, the existence of a unique strong solution is known only for sufficiently small data, e.g., $||\nabla \mathbf{u}_o||_{L^2(\Omega)}$ small enough (global-in-time unique solution) or on sufficiently short intervals of time,$0 \leq t \leq T$. However, there are weak solutions on $(0, T)$, for all T, but the uniqueness of these solutions is still an open problem.

Natural Boundary condition Many practical problems in fluid dynamics are studied and conceptualized in unbounded domains. Then, these domains have to be truncated to allow the computation of the flow field. Free Outflow boundary condition occurs when homogeneous Neumann Condition occurs on all the boundaries, that is, $p\mathbf{n} - \nu \nabla \mathbf{u} \cdot \mathbf{n} = 0$ on Γ and so the boundary terms on the Energy balance equation disappear and no condition is imposed on the unknown velocity. We can impose a pressure force on the artificial borders that close the domain, $\nu \nabla \mathbf{u} \cdot \mathbf{n} - p\mathbf{n} = -p_\alpha \mathbf{n}$ on Γ_α, $\alpha =$in, out. We use a constant pressure p_α on all Γ_α.This problem is called the pressure drop problem.

If we replace $\int_\Gamma (p\mathbf{n} - \nu\nabla\cdot\mathbf{n})\cdot\mathbf{v}$ by following forms on the right hand side:

$$l_\alpha : \mathbf{H}^1_{0,\Gamma_D}(\Omega) \longrightarrow \mathbf{R}$$

$$\mathbf{v} \longrightarrow l_\alpha(\mathbf{v}) = -\int_{\Gamma_\alpha} p_\alpha \mathbf{v}\cdot\mathbf{n}$$

with $\alpha = $ in,out and $p_\alpha \in L^2(0,T)$.

Given u_0, find $\mathbf{u} \in L^2(0,T;\mathbf{H^1(\Omega)})$ and $p \in L^2(0,T;L^2(\Omega))$ such that $\forall \mathbf{v} \in \mathbf{H^1_{0,\Gamma_D}(\Omega)}$ and $\forall q \in L^2(\Omega)$ and $\forall t \geq 0$

$$\begin{cases} \rho(\frac{\partial \mathbf{u}}{\partial t}, \mathbf{v}) + \nu a_o(\mathbf{u},\mathbf{v}) + a_1(\mathbf{u},\mathbf{u},\mathbf{v}) - (p, \nabla\cdot\mathbf{v}) - l_{out}(\mathbf{v}) + l_{in}(\mathbf{v}) \\ (p, \nabla\cdot\mathbf{u}) = 0 \end{cases} \tag{2.1}$$

with $\mathbf{u}|_{t=0} = \mathbf{u}_0$.

Chapter 3

Space and Time Discretization

The nonlinearity in the Navier–Stokes equations can be written in several ways, which are equivalent in the continuum formulation (since $\nabla \cdot \mathbf{u}$), but which lead to different discrete forms. The free-divergence equation is only weakly enforced in a discrete framework, so we do not have an exact discrete free-divergence velocity. Moreover, the divergence of the discrete velocity may grow large enough and cause significant differences between different schemes.

3.1 Space Discretization

Let T_h be a family of quasi-uniform triangulation $T_h = K$ of Ω with the mesh size h. For a given positive integer r, we introduce the finite element space

$$\mathbf{V_h} = \{\mathbf{v} \in \mathbf{V} : \mathbf{v}|_K \in P_r(K) \quad \forall K \in T_h\} \subset \mathbf{V} = \mathbf{H^1_{0,\Gamma_D}}(\mathbf{\Omega})$$

which is the space of continuous piecewise polynomial functions of degree r, and then an approximation of V. Let $\mathbf{u}_h \in \mathbf{V}_h$ be the discretized-in-space function.

Stokes System

The Stokes problem for incompressible flow is given as

$$\begin{aligned} -\nu\nabla^2\mathbf{u} + \nabla p &= \mathbf{f} \quad \text{on } \Omega \\ \nabla \cdot \mathbf{u} &= 0 \quad \text{on } \Omega \\ \mathbf{u} &= \mathbf{g} \quad \text{on } \partial\Omega_D \\ p \cdot \mathbf{n} - \nu\nabla\mathbf{u} \cdot \mathbf{n} &= \mathbf{s} \quad \text{on } \partial\Omega_N \end{aligned}$$

The weak formulation of the Stokes system is

$$\begin{aligned} \int_\Omega \mathbf{v} \cdot (-\nabla^2\mathbf{u} + \nabla p) &= \int_\Omega \mathbf{f} \cdot \mathbf{v} \\ \int_\Omega q\nabla \cdot \mathbf{u} &= 0 \end{aligned}$$

for all $(\mathbf{v}, q)$ in suitably chosen spaces of test functions. Using Green's theorem,

$$\begin{aligned} \int_\Omega \nabla\mathbf{u} : \nabla\mathbf{v} - \int_\Omega p\nabla \cdot \mathbf{v} &= \int_\Omega \mathbf{f} \cdot \mathbf{v} + \int_{d\Omega} (\nu\frac{\partial\mathbf{u}}{\partial\mathbf{n}} - p \cdot \mathbf{n}) \cdot \mathbf{v} \\ \int_\Omega q\nabla \cdot \mathbf{u} &= 0 \end{aligned}$$

Here $\nabla\mathbf{u} : \nabla\mathbf{v}$ represents the componentwise scalar product, for example, in two dimensions $\nabla u_x \cdot \nabla v_x + \nabla u_y \cdot \nabla v_y$.

$$\begin{aligned} \mathbf{H}^1_g &:= \{\mathbf{u} \in \mathbf{H}^1(\Omega) | \mathbf{u} = \mathbf{g} \text{ on } \partial\Omega_D\} \\ \mathbf{H}^1_0 &:= \{\mathbf{v} \in \mathbf{H}^1(\Omega) | \mathbf{u} = \mathbf{0} \text{ on } \partial\Omega_D\} \end{aligned}$$

Find $(\mathbf{u}, p) \in \mathbf{H}_g^1(\Omega)$ x $L_0^2(\Omega)$ such that

$$\int_\Omega \nabla \mathbf{u} : \nabla \mathbf{v} - \int_\Omega p \nabla \cdot \mathbf{v} = \int_\Omega \mathbf{f} \cdot \mathbf{v} + \int_{\partial \Omega_N} \mathbf{s} \cdot \mathbf{v}$$
$$\int_\Omega q \cdot \nabla \cdot \mathbf{u} = 0$$

for all $(\mathbf{v}, q) \in \mathbf{H}_0^1(\Omega)$ x $L_0^2(\Omega)$.

We consider the following finite-dimensional spaces for an integer $r \geq 0$

$\mathbf{V_h} := \{\mathbf{v} \in \mathbf{V} : \mathbf{v}|_K \in \mathbf{P_{r+1}}(K) \quad \forall K \in T_h\} \subset \mathbf{V} = \mathbf{H_0^1(\Omega)}$

$\mathbf{U_h} := \{\mathbf{u} = \mathbf{u}_g + \mathbf{v} : \mathbf{u}_g = \mathbf{g} \text{ on } \Gamma_D, \mathbf{v} \in \mathbf{V_h} \text{ and } \mathbf{v}|_K \in \mathbf{P_{r+1}}(K) \quad \forall K \in T_h\} \subset \mathbf{U} = \mathbf{H_g^1(\Omega)}$

$M_h := \{p \in M : p|_K \in P_r(K) \quad \forall K \in T_h\} \subset M = L_0^2(\Omega)$

Find $(\mathbf{u}_h, p_h) \in U_h$ x M_h such that

$$\int_\Omega \nabla \mathbf{u}_h : \nabla \mathbf{v}_h - \int_\Omega p_h \nabla \cdot \mathbf{v}_h = \int_\Omega \mathbf{f} \cdot \mathbf{v}_h + \int_{\partial \Omega_N} \mathbf{s} \cdot \mathbf{v}_h$$
$$\int_\Omega q_h \cdot \nabla \cdot \mathbf{u}_h = 0$$

for all $(\mathbf{v}_h, q_h) \in V_h$ x M_h

We introduce a set of vector-valued (velocity) basis functions $\{\phi_\mathbf{j}\}$. For example, in two dimensions, we have

$$\{\boldsymbol{\phi_1}, \boldsymbol{\phi_2}, \cdots, \boldsymbol{\phi_{2n}}\} := \{(\phi_1, 0)^t, \cdots, (\phi_n, 0)^t, (0, \phi_1)^t, \cdots, (0, \phi_n)^t\}$$

We assume here that each basis function is of Lagrangian type, that is, each basis function ϕ_j has a node $x_j \in \Omega$ associated with it such that

$$\phi_j(x_j) = 1, \quad \phi_j(x_i) = 0 \quad \text{for all nodes } x_i \neq x_j$$

Similarly, we introduce a set of scalar(pressure) basis function $\{\psi_k\}$. To identify the corresponding linear algebra problem, we set the solution $(\mathbf{u}_h, p_h)$ as

$$\mathbf{u}_h = \sum_{j=1}^{n_u} \mathbf{u}_j \boldsymbol{\phi}_j + \sum_{j=n_u+1}^{n_u+n_\delta} \mathbf{u}_j \boldsymbol{\phi}_j$$

$$p_h = \sum_{k=1}^{n_p} p_k \psi_k$$

with $\sum_{j=1}^{n_u} \mathbf{u}_j \boldsymbol{\phi}_j \in V_h$. We fix the coefficients $u_j : j = n_u + 1, \ldots, n_u + n_\partial$ so that the second term interpolates the boundary data on $\partial\Omega_D$, that is, for $x_j \in \partial\Omega_D$, $\mathbf{u}_h(x_j) = \mathbf{u}_j = \mathbf{g}_j$.

Symmetric and Non symmetric Tensor

In a continuous framework, the following holds

1. $\nabla \cdot \mathbf{u} = 0$
2. $\nabla \cdot (\nabla \mathbf{u})^t = \nabla(\nabla \cdot \mathbf{u})$ and $\nabla \cdot (\nabla \mathbf{u}) = \Delta \mathbf{u}$

$$2 \cdot \nabla \cdot (\frac{\nabla \mathbf{u} + (\nabla \mathbf{u})^t}{2}) = 2 \cdot (\frac{\nabla \cdot (\nabla \mathbf{u}) + \nabla(\nabla \cdot \mathbf{u})}{2}) = \Delta \mathbf{u}$$

Hence in the continuous framework, using the divergence-free condition, the symmetric tensor $2\nabla \cdot (\frac{\nabla \mathbf{u} + (\nabla \mathbf{u})^t}{2})$ is equal to the nonsymmetric tensor $\Delta \mathbf{u}$. By us-

ing the non-symmetric tensor, the matrix system in n dimensions specifically with $\mathbf{u} := ([\mathbf{u}_x]_1, \ldots, [\mathbf{u}_x]_n, [\mathbf{u}_y]_1, \ldots, [\mathbf{u}_y]_n)^T$

$$\begin{pmatrix} A & O & B_x^T \\ O & A & B_y^T \\ B_x & B_y & O \end{pmatrix} \begin{pmatrix} \mathbf{u}_x \\ \mathbf{u}_y \\ p \end{pmatrix} = \begin{pmatrix} \mathbf{f}_x \\ \mathbf{f}_y \\ \mathbf{g} \end{pmatrix}$$

where the n x n matrix A is the scalar Laplacian matrix and the n_p x n matrices B_x and B_y represent weak derivatives in the x and y directions:
$A = [a_{ij}], a_{ij} = \int_\Omega \nabla\phi_i \cdot \nabla\phi_j,$
$B_x = [b_{x,ki}], b_{x,ki} = -\int_\Omega \psi_k \frac{\partial\phi_i}{\partial x},$
$B_y = [b_{x,ki}], b_{y,kj} = -\int_\Omega \psi_k \frac{\partial\phi_j}{\partial y},$

The stiffness matrix is block diagonal, but using the symmetrized tensor, the matrix will no longer be block-diagonal. At the discrete level,$\nabla \cdot \mathbf{u}_h$ is not exactly equal to 0. In particular, the differences were stronger when one used a coarse mesh since the derivative computations are less accurate and the difference is stronger for the flows that contain more rotational structures in turbulent flows where the Reynolds number is very high.
The unique solvability of the system is obtained by looking at the homogeneous part

$$\mathbf{A}\mathbf{u} + B^T p = 0$$
$$B\mathbf{u} = 0$$

Multiplying first and second equation by $\mathbf{u}^T, p^T$ respectively, $\mathbf{u}_x^T \mathbf{A}\mathbf{u}_x + \mathbf{u}_y^T \mathbf{A}\mathbf{u}_y = 0$. Since $\mathbf{A}$ is positive definite, we get unique solvability w.r.t. the velocity $\mathbf{u}$ but $B^T p = 0$ implies the pressure is unique only upto the nullspace of the matrix B^T.

$< B^T p, \mathbf{v} >= 0 \quad \forall \mathbf{v} \in V_h$ implies

$$\int_\Omega p_h \nabla \cdot \mathbf{v}_h = \int_\Omega (\nabla p_h)\mathbf{v}_h = 0 \quad \forall \mathbf{v} \in V_h$$

which gives p_h = constant, null(B^T)=1. Any nonconstant pressure functions in the null space of B^T are referred to as the spurious pressure nodes.

Necessary Condition for the existence of a solution Consider the Stokes system for dimension 1. In this case, the matrix system becomes,

$$\begin{pmatrix} A & B^T \\ B & O \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}$$

where $A \in R^{n_v \text{ x } n_v}$, $B \in R^{n_q \text{ x } n_v}$. Since A is positive definite, A has full rank n_v, and the existence of a solution for the system depends upon the rank of matrix B. If $n_q > n_v$, then there are n_q number of n_v dimensional vectors, which causes them to be linearly dependent. So we get, rank(B) $< n_q$. So for the existence of a unique solution for the system, it is necessary that $n_q < n_v$. Also, the inf-sup condition holds iff the rank of B is n_q.

3.2 Stable Unstable pairs of Finite Element Spaces

Non Inheritance of the *inf-sup* condition: Let V_h, Q_h be finite-dimensional subspaces of V, Q,then

$$\underset{0 \neq \mathbf{v}^h \in V^h}{sup} \frac{(\nabla \cdot \mathbf{v}^h, q)}{||\nabla \mathbf{v}^h||_{V^h} ||q||_Q} \leq \underset{0 \neq \mathbf{v} \in V}{sup} \frac{(\nabla \cdot \mathbf{v}, q)}{||\nabla \mathbf{v}||_V ||q||_Q}$$

Hence we cannot ensure a lower bound for the lhs. Taking infimum over Q^h, one is taking infimum over a smaller set so that

$$\underset{0\neq q^h\in Q^h}{inf}\ \underset{0\neq \mathbf{v}^h\in V^h}{sup}\ \frac{(\nabla\cdot\mathbf{v}^h, q)}{||\nabla\mathbf{v}^h||_{V^h}||q^h||_Q}$$

becomes positive.

Appropriate choice for finite element spaces: The discrete *inf-sup* conditions suggest that the velocity space should be sufficiently large and the pressure space should be small.

Unstable Pairs of Finite element Spaces

The violation of the discrete inf-sup condition: If we can find a non-trivial $q_h \in Q^h$ such that $b^h(\mathbf{v}^h, q^h) = 0 \quad \forall\mathbf{v}^h \in V^h$. Such non-trivial $q^h \in Q^h$ are called spurious pressure nodes.

(i)The P_1/P_1 pair of Finite element. Let K be a mesh cell and $q^h_{1,K}, q^h_{2,K}, q^h_{3,K}$ be the values of the pressure in the vertices of K. Then, the integral of q^h on K can be evaluated exactly by a quadrature rule $\int_K q^h(x)dx = \frac{|K|}{3}(q^h_{1,K} + q^h_{2,K} + q^h_{3,K})$. Hence the integral mean value condition for the finite element pressure space reads as

$$0 = \int_\Omega q^h(x)dx = \sum_{K\in T^h}\int_K q^h(x)dx = \frac{|K|}{3}(q^h_{1,K} + q^h_{2,K} + q^h_{3,K})$$

On each mesh cell K, it follows that

$$\nabla\cdot\mathbf{v}^h|_K(x) = c_K$$

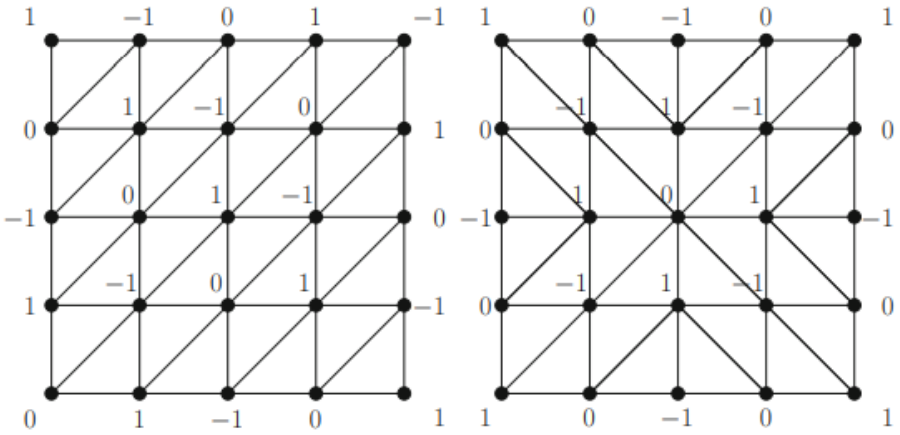


Figure 3.1: Checkerboard instabilities for the P_1/P_1 finite element

for some constant c_K, since $\mathbf{v}^h \in P_1$. And since for each mesh cell K, one can construct a non-trivial function q^h with $q^h_{1,K} + q^h_{2,K} + q^h_{3,K} = 0$ for all $K \in T^h$, we can have ,

$$\int_K (\nabla \cdot \mathbf{v}^h) q^h = \sum_{K \in T^h} c_K \int_K q^h = -\frac{|K|}{3} \sum_{K \in T^h} c_K (q^h_{1,K} + q^h_{2,K} + q^h_{3,K})$$

Hence P_1/P_1 pair of Finite Element Spaces is not stable and these spurious nodes lead to a checkerboard kind of instabilities.

(ii)The P_1/P_0 pair of Finite Element. For this pair it holds that $\nabla \cdot P_1 \subset P_0$. But this pair violates the discrete inf-sup condition.Consider the domain $\Omega = (0,1)^2$ and a grid on it as presented in figure with $2n^2$ mesh cells with $n > 1$. Then the dimension of V^h corresponds to twice the number of interior vertices, $\dim(V^h) = 2(n-1)^2$. The dimension of Q^h corresponds to the number of mesh cells reduced by one(because of

the integral mean value condition), $\dim(Q^h) = 2n^2 - 1$. Hence,

$$dim(V^h) - dim(Q^h) = 2n^2 - 4n + 2 - 2n^2 + 1 = 3 - 4n < 0$$

for $n \geq 1$. So the necessary condition for the existence of a unique solution $n_q < n_v$ is violated.

3.3 Time Discretization

Let $0 = t_0 < t_1 < \cdots < t_N = I$ be a uniform decomposition of the considered time interval $[0, I]$ and $\delta_t = t^n - t^{n-1}, 1 \leq n \leq N$, be the time step size. Using the Taylor expansion at $t^* = \theta t^n + (1-\theta)t^{n-1}$ for $0 \leq \theta \leq 1$,

$$\frac{u(t^n) - u(t^{n-1})}{\delta_t} = u'(t^*) + \frac{(1-2\theta)}{2}\delta_t u''(t^*) + O(\delta_t^2)$$

Consider the following finite dimensional subspaces for an integer $r \geq 0$

$\mathbf{V_h} := \{\mathbf{v} \in \mathbf{V} : \mathbf{v}|_K \in \mathbf{P_{r+1}}(K) \quad \forall K \in T_h\} \subset \mathbf{V} = \mathbf{H_0^1}(\mathbf{\Omega})$

$\mathbf{U_h} := \{\mathbf{u} = \mathbf{u}_g + \mathbf{v} : \mathbf{u}_g = \mathbf{g} \text{ on } \Gamma_D, \mathbf{v} \in \mathbf{V_h} \text{ and } \mathbf{v}|_K \in \mathbf{P_{r+1}}(K) \quad \forall K \in T_h\} \subset \mathbf{U} = \mathbf{H_g^1}(\mathbf{\Omega})$

$M_h := \{p \in M : p|_K \in P_r(K) \quad \forall K \in T_h\} \subset M = L_0^2(\Omega)$

The weak formulation of the test problem (4) discretized in space leads to the semi-discrete form

For each $t \in [0,T]$, find $(\mathbf{u}_h(\cdot,t), p_h(\cdot,t)) \in \mathbf{U_h}$ x M_h such that

$$\begin{cases} (\frac{d}{dt}\mathbf{u}_h(t), \mathbf{v}_h) + a_1(\mathbf{u}_h(t); \mathbf{u}_h(t), \mathbf{v}_h) + \nu a_o(\mathbf{u}_h(t), \mathbf{v}_h) - (\nabla \cdot \mathbf{v}_h, p_h(t)) =< \mathbf{f}, \mathbf{v} > \\ (\nabla \cdot \mathbf{u}_h, q_h) = 0 \\ \mathbf{u}_h(0) = \mathbf{u}_{0,h} \qquad \forall (\mathbf{v}_h, q_h) \in V_h \text{ x } M_h \end{cases} \tag{3.1}$$

With $r = 1$, we have a $\mathbf{P_2}$ space for velocity and P_1 space for pressure.

If $t^* \in [t^{n-1}, t^n]$, $\mathbf{u}_h(t^*) = \theta \mathbf{u}_h(t^n) + (1-\theta)\mathbf{u}_h(t^{n-1})$,
$p_h(t^*) = \theta p_h(t^n) + (1-\theta)p_h(t^{n-1})$,
$\mathbf{f}_h(t^*) = \mathbf{f}(\theta t^n + (1-\theta)t^{n-1})$
The system of equations is given as

$$\begin{cases} \frac{1}{\Delta t}(\mathbf{u}^{n+1}, \mathbf{v}) - \frac{\rho}{\Delta t}(\mathbf{u}^n, \mathbf{v}) + \nu(\nabla \mathbf{u}_\theta^{n+1}, \nabla \mathbf{v}) + ((\mathbf{u}_\theta^{n+1} \cdot \nabla)\mathbf{u}_\theta^{n+1}, \mathbf{v}) - (\nabla \cdot \mathbf{v}, p_\theta^{n+1}) =< \mathbf{f_\theta^{n+1}}, \mathbf{v} > \\ \nabla \cdot \mathbf{u}^{n+1} q = 0 \end{cases}$$

The system of equations with the symmetric tensor $D(\mathbf{u}) = \frac{\nabla \mathbf{u} + \nabla \mathbf{u}^T}{2}$ is given as

$$\begin{cases} \frac{1}{\Delta t}(\mathbf{u}^{n+1}, \mathbf{v}_h) - \frac{1}{\Delta t}(\mathbf{u}^n, \mathbf{v}_h) + 2\nu(D(\mathbf{u}_\theta^{n+1}), \nabla \mathbf{v}_h) + ((\mathbf{u}_\theta^{n+1} \cdot \nabla)\mathbf{u}_\theta^{n+1}, \mathbf{v}_h) \\ -(\nabla \cdot \mathbf{v}_h, p_\theta^{n+1}) =< \mathbf{f_\theta^{n+1}}, \mathbf{v}_h > \\ (\nabla \cdot \mathbf{u}^{n+1}, q_h) = 0 \qquad \forall (\mathbf{v}_h, q_h) \in V_h \text{ x } M_h \end{cases}$$

Theta Scheme

$$\begin{cases} (\mathbf{u}^{n+1}, \mathbf{v}_h) + \theta\Delta t[2\nu D(\mathbf{u}^{n+1}), \nabla\mathbf{v}_h) + ((\mathbf{u}^{n+1} \cdot \nabla)\mathbf{u}^{n+1}, \mathbf{v}_h)] - \Delta t\theta(\nabla \cdot \mathbf{v}_h, p^{n+1}) \\ = (\mathbf{u}^n, \mathbf{v}_h) - (1-\theta)\Delta t[2\nu(D(\mathbf{u}^n), \nabla\mathbf{v}_h) + ((\mathbf{u}^n \cdot \nabla)\mathbf{u}^n, \mathbf{v}_h)] + \Delta t(1-\theta)(\nabla \cdot \mathbf{v}_h, p^n) \\ +(1-\theta)\Delta t < \mathbf{f^n}, \mathbf{v}_h > +\theta\Delta t < \mathbf{f^{n+1}}, \mathbf{v}_h > \\ (\nabla \cdot \mathbf{u}^{n+1}, q_h) = 0 \qquad \forall(\mathbf{v}_h, q_h) \in V_h \text{ x } M_h \end{cases}$$

In two dimensions, finite element velocity and pressure basis functions

$$\{\boldsymbol{\phi_1}, \boldsymbol{\phi_2}, \cdots, \boldsymbol{\phi_{n_u}}\} := \{(\phi_1, 0)^t, \cdots, (\phi_{n_u}, 0)^t, (0, \phi_1)^t, \cdots, (0, \phi_{n_u})^t\}$$

$$\{\psi_1, \psi_2, \cdots, \psi_{n_p}\}$$

With

$$\mathbf{u}_h = \begin{pmatrix} \sum_{j=1}^{n_u+n_\delta} u_j^x \phi_j \\ \sum_{j=1}^{n_u+n_\delta} u_j^y \phi_j \end{pmatrix} = \sum_{j=1}^{n_u} \mathbf{u}_j \boldsymbol{\phi}_j + \sum_{j=n_u+1}^{n_u+n_\delta} \mathbf{u}_j \boldsymbol{\phi}_j$$

$$p_h = \sum_{k=1}^{n_p} p_k \psi_k$$

Explicit Euler

We get the Explicit Euler scheme by setting $\theta = 0$ both for velocity and pressure variables. The local truncation error for the Explicit Euler Scheme is $O(\delta t^2)$. Since the global truncation error is proportional to δt, it is a first-order technique. The system obtained through the Explicit Euler method is a linear system.

$$\begin{pmatrix} M & 0 & 0 \\ 0 & M & 0 \\ B_x & B_y & 0 \end{pmatrix} \begin{pmatrix} u_x^{n+1} \\ u_y^{n+1} \\ p^{n+1} \end{pmatrix} = \begin{pmatrix} h_x(u_x^n, p^n, f_x^n) \\ h_y(u_y^n, p^n, f_y^n) \\ 0 \end{pmatrix}$$

$M \in R^{n_u \text{ x } n_u}$ is the mass matrix $m_{ij} = \int_\Omega \phi_\mathbf{i} \phi_\mathbf{j} d\Omega$.

$B_x, B_y \in R^{np \text{ x } n_u}$ $b_{xi,xj} = -\int_\Omega \psi_i \frac{\partial \phi_j}{\partial x} d\Omega$ $b_{yi,yj} = -\int_\Omega \psi_i \frac{\partial \phi_j}{\partial y} d\Omega$

This system is overdetermined for $\mathbf{u}^{n+1}$ and does not allow the determination of p^{n+1}.

Semi Explicit Euler

We obtain the Semi Explicit Euler by taking $\theta = 0$ for velocity variables and $\theta = 1$ for pressure variables. Here too, the system is linear.

$$\begin{pmatrix} M & 0 & \Delta t B_x^T \\ 0 & M & \Delta t B_y^T \\ B_x & B_y & 0 \end{pmatrix} \begin{pmatrix} u_x^{n+1} \\ u_y^{n+1} \\ p^{n+1} \end{pmatrix} = \begin{pmatrix} h_x(u_x^n, f_x^n) \\ h_y(u_y^n, f_y^n) \\ 0 \end{pmatrix}$$

$M \in R^{n_u \text{ x } n_u}$ is the mass matrix $m_{ij} = \int_\Omega \phi_\mathbf{i} \phi_\mathbf{j} d\Omega$.

$B_x, B_y \in R^{np \text{ x } n_u}$ $\quad b_{xi,xj} = -\int_\Omega \psi_i \frac{\partial \phi_j}{\partial x} d\Omega \qquad b_{yi,yj} = -\int_\Omega \psi_i \frac{\partial \phi_j}{\partial y} d\Omega.$

Rewriting in a compact form

$$\frac{1}{\Delta t} M\mathbf{u}^{n+1} + B^T p^{n+1} = \mathbf{H}$$
$$B\mathbf{u^{n+1}} = \mathbf{0}$$

M is symmetric and positive definite $BM^{-1}B^T p^{n+1} = BM^{-1}H$ is non singular if $\ker(B^T)$=0.

Crank Nicholson

We get the Crank Nicholson scheme by setting $\theta = 0.5$ for velocity and pressure variables. This scheme is second-order accurate in time. The system obtained is non-linear due to the presence of a non-zero convective term.

$$\begin{cases} 2(\mathbf{u}_{k+1}^{n+1}, \mathbf{v}) + \Delta t[\nu(D(\mathbf{u}_{k+1}^{n+1}), \nabla\mathbf{v}) + ((\mathbf{u}_{k+1}^{n+1} \cdot \nabla)\mathbf{u}_{k+1}^{n+1}, \mathbf{v})] - \Delta t(\nabla \cdot \mathbf{v}, p_{k+1}^{n+1}) = \\ (\mathbf{u}_{k+1}^{n}, \mathbf{v}) - \Delta t[\nu(D(\mathbf{u}_{k+1}^{n}), \nabla\mathbf{v}) + ((\mathbf{u}_{k+1}^{n} \cdot \nabla)\mathbf{u}_{k+1}^{n}, \mathbf{v})] + \Delta t(\nabla \cdot \mathbf{v}, p_{k+1}^{n}) \\ +\Delta t < \mathbf{f_{k+1}^{n}}, \mathbf{v} > +\Delta t < \mathbf{f_{k+1}^{n+1}}, \mathbf{v} > \\ (\nabla \cdot \mathbf{u}_{k+1}^{n+1}, q) = 0 \end{cases}$$

The two-dimensional matrix system for the Crank Nicholson scheme is given as

$$\begin{pmatrix} M + A + S_{ax} & S_{ay} & B_x^T \\ S_{bx} & M + A + S_{by} & B_y^T \\ B_x & B_y & 0 \end{pmatrix} \begin{pmatrix} u_{x,k+1}^{n+1} \\ u_{y,k+1}^{n+1} \\ p_{k+1}^{n+1} \end{pmatrix} + \begin{pmatrix} N_x \\ N_y \\ 0 \end{pmatrix} = \begin{pmatrix} h_x(u_x^n, p^n, f_x^{n,n+1}) \\ h_y(u_y^n, p^n, f_y^{n,n+1}) \\ 0 \end{pmatrix}$$

where $M, A, S_{ax}, S_{ay}, S_{bx}, S_{by} \in R^{n_u \text{ x } n_u}$, $B_x, B_y \in R^{np \text{ x } n_u}$ and $N_x, N_y \in R^{n_u \text{ x } 1}$ where

$[m]_{ij} = \frac{2}{\Delta t}\int_\Omega \phi_i\phi_j d\Omega \quad , [a]_{ij} = \nu\int_\Omega \nabla\phi_i\nabla\phi_j d\Omega \quad , [s_{ax}]_{ij} = \nu\int_\Omega \frac{\partial\phi_j}{\partial x}\frac{\partial\phi_i}{\partial x}d\Omega$

$[s_{ay}]_{ij} = \nu\int_\Omega \frac{\partial\phi_j}{\partial y}\frac{\partial\phi_i}{\partial x}d\Omega \quad , [s_{bx}]_{ij} = \nu\int_\Omega \frac{\partial\phi_j}{\partial x}\frac{\partial\phi_i}{\partial y}d\Omega$

$[s_{by}]_{ij} = \nu\int_\Omega \frac{\partial\phi_j}{\partial y}\frac{\partial\phi_i}{\partial y}d\Omega, \quad [n_x]_i = \int_\Omega(\nabla u^{n+1}_{x,k+1})u^{n+1}_{x,k+1}\phi_i d\Omega$

$[n_y]_i = \int_\Omega(\nabla u^{n+1}_{y,k+1})u^{n+1}_{y,k+1}\phi_i d\Omega, \quad [b_x]_{ij} = -\int_\Omega \psi_i\frac{\partial\phi_j}{\partial x}d\Omega, \quad [b_y]_{ij} = -\int_\Omega \psi_i\frac{\partial\phi_j}{\partial y}d\Omega$

M is the mass matrix, $A, S_{ax}, S_{ay}, S_{bx}, S_{by}$ are the matrices arising out of the symmetric tensor, N_x, N_y are non linear vectors. So the system is not linearly solvable. For notational convenience, we represent the above system by the residual $R(\mathbf{u}^{n+1}_{k+1}, p^{n+1}_{k+1})$ as

$$R(\mathbf{u}^{n+1}_{k+1}, p^{n+1}_{k+1}) := [K][\mathbf{u}^{n+1}_{k+1}\ p^{n+1}_{k+1}]^T + N(\mathbf{u}^{n+1}_{k+1}, p^{n+1}_{k+1}) - H(\mathbf{u}^{n}_{k+1}, p^{n}_{k+1}, f^n) \tag{3.2}$$

The goal is to find $(\mathbf{u}^{n+1}_{k+1}, p^{n+1}_{k+1})$ such that the Residual $R(\mathbf{u}^{n+1}_{k+1}, p^{n+1}_{k+1})$ is minimum.

3.3.1 Newton's Method

Newton's Method is a root-finding algorithm for a function f that uses the first few terms of the Taylor series(up to the first derivative) to produce a sequence of iterates that converges to the root of the given function quadratically in the vicinity of a suspected root. The regular Newton-Raphson method is initialized with a starting point x_0 and then we iterate

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

In the higher dimension, the derivative is given by the Jacobian matrix J

$$\mathbf{x}_{n+1} = \mathbf{x}_n - (J(\mathbf{x}_n))^{-1} F(\mathbf{x}_n)$$

Calculating the inverse of the Jacobian matrix is computationally expensive, so we first solve the linear system

$$J(\mathbf{x}_n)\mathbf{y}_n = F(\mathbf{x}_n)$$

and update the next iteration as

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \mathbf{y}_n$$

We use Newton's method to find an approximate root $(\mathbf{u}_{k+1}^{n+1}, p_{k+1}^{n+1})$ of the residual R, the $k+1$ iterate at time $n+1$. The Jacobian matrix is computed by using numerical differentiation. Let $z = (\mathbf{u}_k^{n+1}, p_k^{n+1})$, then

$$[J]_{ij}(z) = \frac{\partial R_i}{\partial z_j}(\mathbf{u}_k^{n+1}, p_k^{n+1})$$

$$\frac{\partial R_i}{\partial z_j}(\mathbf{u}_k^{n+1}, p_k^{n+1}) = \frac{R_i(z_1, \ldots, z_j + \Delta z_j, \ldots, z_l)|_z - R_i(z_1, \ldots, z_j, \ldots, z_l)|_z}{\Delta z_j}$$

Newton's Iteration for solving the Navier Stokes system

Choose the initial guess $(\mathbf{u}^0, p^0)$ at time $t = 0$. The initial guess for the solution at the $(n+1)^{th}$ time step $(\mathbf{u}_0^{n+1}, p_0^{n+1})$ can be taken to be the solution from the previous time step $(\mathbf{u}^n, p^n)$.

Algorithm 1 Newton's Iteration

Input tolerance tol, initial guess $(\mathbf{u}_0^{n+1}, p_0^{n+1})$

$k = 0$

while True **do**

  Compute $J(\mathbf{u}_k^{n+1}, p_k^{n+1})$ by numerical differentiation.

  Solve for $(\mathbf{w}_k^{n+1}, l_k^{n+1})^T$ by using a suitable linear solver.

$$J(\mathbf{u}_k^{n+1}, p_k^{n+1}) \begin{pmatrix} \mathbf{w}_k^{n+1} \\ l_k^{n+1} \end{pmatrix} = R(\mathbf{u}_k^{n+1}, p_k^{n+1})$$

  if $(||w_k^{n+1}||^2 + ||l_k^{n+1}||^2)^{\frac{1}{2}} < tol$ **then**

    break

  end if

  Update

$$\begin{pmatrix} \mathbf{u}_{k+1}^{n+1} \\ p_{k+1}^{n+1} \end{pmatrix} = \begin{pmatrix} \mathbf{u}_k^{n+1} \\ p_k^{n+1} \end{pmatrix} - \begin{pmatrix} \mathbf{w}_k^{n+1} \\ l_k^{n+1} \end{pmatrix}$$

  $k = k + 1$

end while

3.3.2 Biconjugate Gradient Method with ILU Preconditioner

Conjugate Gradient Method

The conjugate gradient method is a popular iterative algorithm used to solve linear systems of equations. It is beneficial for solving large, sparse, symmetric, and positive-definite systems of linear equations. The algorithm is based on minimizing a quadratic function by finding a sequence of directions that are conjugate to each other. Let $\mathbf{x}^*$ be solution of the system $A\mathbf{x} = b$. $\mathbf{x}^*$ is also the unique minimizer of

the following function

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} - \mathbf{x}^T\mathbf{b}$$

The existence of a unique minimizer $\mathbf{x}^*$ is apparent as the Hessian matrix of second derivatives is symmetric positive-definite

$$\mathbf{H}(f(\mathbf{x})) = \mathbf{A}$$

$$\nabla f(\mathbf{x}) = \mathbf{A}\mathbf{x} - \mathbf{b}$$

At each iteration k the algorithm finds a new search direction $\mathbf{p}_k$ which is conjugate to all the previous search directions $\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_{k-1}$, that is, $\mathbf{p}_k^T\mathbf{A}\mathbf{p}_j = 0$ for all $j = 0, 1, \ldots, k-1$. At each iteration k, the residual $\mathbf{r}_k$ is defined as $\mathbf{r}_k = \mathbf{b} - \mathbf{A}\mathbf{x}_k$. $\mathbf{r}_k$ is the negative gradient of f at $\mathbf{x}_k$, so the gradient descent is required to move in the direction $\mathbf{r}_k$. To enforce the condition that $\mathbf{p}_k$ must be conjugate to all the previous search directions, we can use the Gram-Schmidt orthonormalization to compute $\mathbf{p}_k$ as

$$\mathbf{p}_k = \mathbf{r}_k - \sum_{i<k} \frac{\mathbf{p}_i^T\mathbf{A}\mathbf{r}_k}{\mathbf{p}_i^T\mathbf{A}\mathbf{p}_i}\mathbf{p}_i$$

β is a scalar coefficient that determines the weighting of the previous search direction. If β is small, then the new search direction will be close to orthogonal to the previous search direction and if it's large, then the new search direction will be close to the previous search direction. α is a scalar coefficient determining the step size along the search direction.

Algorithm 2 Conjugate Gradient Method

Input tolerance tol, positive definite matrix $\mathbf{A}$, $\mathbf{b}$
$\mathbf{x}_0 = \mathbf{0}$
$\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$
$\mathbf{p}_0 := \mathbf{r}_0$
$k = 0$
while True **do**
  $\alpha_k := \frac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{p}_k^T \mathbf{A} \mathbf{p}_k}$
  $\mathbf{x}_{k+1} := \mathbf{x}_k + \alpha_k \mathbf{p}_k$
  $\mathbf{r}_{k+1} := \mathbf{r}_k - \alpha_k \mathbf{A} \mathbf{p}_k$
  if $\mathbf{r}_{k+1} < tol$ **then**
    $break$
  end if
  $\beta_k := \frac{\mathbf{r}_{k+1}^T \mathbf{r}_{k+1}}{\mathbf{r}_k^T \mathbf{r}_k}$
  $\mathbf{p}_{k+1} := \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k$
  $k := k + 1$
end while

Biconjugate Gradient Stabilized Method

Biconjugate Gradient Stabilized (BiCGSTAB) is an iterative method for solving linear systems of equations Ax = b, where A is a non-symmetric matrix. BiCGSTAB is similar to the Conjugate Gradient (CG) method but is adapted to nonsymmetric matrices by working with two different sequences of search directions. The basic idea of the BiCGSTAB algorithm is to generate two sequences of search directions $\mathbf{p}_k$ and $\mathbf{q}_k$ that are mutually biorthogonal w.r.t matrix $\mathbf{A}$, that is, $\mathbf{p}_k^T \mathbf{A} \mathbf{q}_j = 0$ for $j \neq k$. The BiCGSTAB algorithm combines these two sequences of search directions to accelerate convergence by using information from both the residual and the aux-

iliary residual to compute the step size for each iteration. If the matrix is symmetric positive-definite, CG is generally faster and more efficient than BiCGSTAB, but if A is nonsymmetric, BiCGSTAB is the only option.

Incomplete LU (ILU) Preconditioning

Iterative methods such as Conjugate Gradient (CG) and Biconjugate Gradient Stabilized (BiCGSTAB) rely on the computation of matrix-vector products, which can be computationally expensive for large matrices. Preconditioning is a technique that modifies the original matrix to make it easier to solve while preserving the solution to the original problem. A standard approach is to use a non-singular matrix M and rewrite the system as

$$M^{-1}A\mathbf{x} = M^{-1}\mathbf{b}$$

In incomplete LU factorization we take $M = LU$ where L, U are lower and upper triangular matrices such that $A \sim LU$.

In the context of sparse matrix factorization, fill-in refers to the nonzero elements that are introduced in the factors of a matrix during the factorization process but were not present in the original matrix. When a sparse matrix is factored, certain operations can cause additional nonzero entries to appear in the factorization, leading to more dense factors. These additional nonzero entries are called fill-in.

One way to control fill-in is by using a fill-in parameter p that limits the fill-in allowed during the factorization. A larger value of p allows more fill-in, which results in more efficient factorization but less effective preconditioning and more computational cost. Conversely, a smaller value of p results in less fill-in and better preconditioning but slower factorization and lesser computational cost.

Algorithm 3 ILU factorization with fill-in parameter p

Input A, fill-in parameter p

n is the size of matrix A

Set $L = U = [0]_{n \times n}$ ▷ Initialize matrices to zero

for $k = 1$ to n **do**

$U[k,k] = A[k,k] - \sum_{j=1}^{k-1} L[k,j] \cdot U[j,k]$ ▷ Diagonal entry of U

Compute non-zero entries above the diagonal of U:

for $j = k+1$ to $\min(k+p, n)$ **do**

$U[k,j] = \frac{1}{U[k,k]} \left(A[k,j] - \sum_{i=1}^{k-1} L[k,i] \cdot U[i,j] \right)$

end for

$L[k,k] = 1$ ▷ Diagonal entry of L

Compute non-zero entries below the diagonal of L:

for $i = k+1$ to $\min(k+p, n)$ **do**

$L[i,k] = \frac{1}{U[k,k]} \left(A[i,k] - \sum_{j=1}^{k-1} L[i,j] \cdot U[j,k] \right)$

end for

end for

Return L and U

Algorithm 4 BiCGSTAB with ILU Preconditioning

Input A, $\mathbf{b}$, tolerance tol, max_iter, fill-in parameter p

Compute ILU factorization of A: L, $U = \text{ILU}(A, p)$.

Initialize $k = 0$, $\mathbf{x}_0 = 0$, $\mathbf{r}_0 = b$, $\tilde{\mathbf{r}}_0 = b$, $\mathbf{p}_0 = 0$, $\tilde{\mathbf{p}}_0 = 0$

$\tilde{\mathbf{z}}_0 := U^{-1}\tilde{\mathbf{r}}_0$

$\mathbf{z}_0 := L^{-1}\mathbf{r}_0$

$\rho_0 := \tilde{\mathbf{z}}_0^T \mathbf{r}_0$

while True **do**

$k = k + 1$

$\alpha_k = \frac{\rho_{k-1}}{(\tilde{\mathbf{p}}_{k-1}^T A \mathbf{p}_{k-1})}$.

$\mathbf{x}_k = \mathbf{x}_{k-1} + \alpha_k \mathbf{p}_{k-1}$

$\mathbf{r}_k = \mathbf{r}_{k-1} - \alpha_k A \mathbf{p}_{k-1}$

if $\mathbf{r}_k <$tol **then**

break

end if

$\tilde{\mathbf{r}}_k = \tilde{\mathbf{r}}_{k-1} - \alpha_k A^T \tilde{\mathbf{p}}_{k-1}$

$\mathbf{z}_k = L^{-1}\mathbf{r}_k$

$\tilde{\mathbf{z}}_k = U^{-1}\tilde{\mathbf{r}}_k$

$\rho_k = \tilde{\mathbf{z}}_k^T \mathbf{r}_k$

$\beta_k = \frac{\rho_k}{\rho_{k-1}}$.

$\mathbf{p}_k = \mathbf{z}_k + \beta_k \mathbf{p}_{k-1}$

$\tilde{\mathbf{p}}_k = \tilde{\mathbf{z}}_k + \beta_k \tilde{\mathbf{p}}_{k-1}$

end while

Chapter 4

Flow in Circular Domains

Any internal flow is strongly affected by the geometry of its conduit, and here we seek to identify characteristic features of the relation between flow and vessel geometry which in turn characterizes the nature of the fluid flow.

4.1 Types of fluid flow

Streamline in a flow field is an imaginary curve traced by a fluid particle for a fixed instant of time, tangent to which gives the instantaneous velocity at that point. Based on the trajectories of streamlines, we can identify the flow to be the following:

- Laminar Flow: is characterized by fluid particles following smooth paths in layers, with each layer moving smoothly past the adjacent layers with little or no mixing. This type of flow is also referred to as streamline flow because it

is characterized by non-crossing streamlines. That means every fluid particle follows the same trajectory as its preceding particle.

- Steady Flow: is one in which all parameters(flow velocity, pressure) along any streamline remains constant with respect to time.

- Fully developed flow occurs when the velocity profile does not change along the direction of flow (axial direction), i.e., two different axial locations in the conduit have the same velocity profile. In order for this to occur, the fluid must travel through a length of a straight pipe.

- Turbulent flow refers to an irregular flow in which eddies, swirls, and flow instabilities occur. It is characterized by the crossing of streamlines in the flow.

4.2 Flow in a straight, circular pipe

Reynolds Number is a dimensionless number that depends upon the velocity of the fluid and is given as

$$Re = \frac{\rho|\mathbf{u}|d}{\mu} = \frac{\text{inertial forces}}{\text{viscous forces}}$$

where ρ is the density of fluid $\mathbf{u}$ is the velocity, μ is the dynamic viscosity,d is the diameter of the pipe.

4.2.1 Transition from laminar to turbulent flow

Flow in a straight, circular domain can become turbulent due to the crossing of streamlines arising from increased Reynolds numbers. When Re is smaller, the viscous forces dominate over the inertial forces(the fluid is flowing slower), the forces are sufficient to keep all the fluid particles in line, and the flow is laminar.

Hence in a domain of zero curvature, at low values of the Reynolds number, the flow is laminar. When the Reynolds number exceeds a threshold value, semi-developed turbulence occurs in the flow. This regime is usually referred to as the "transition regime" and occurs for a specific range of Reynolds numbers. Beyond that range, the flow becomes fully turbulent. The mean value of the Reynolds number in the transition regime is usually named "Critical Reynolds number." It is the threshold between the laminar and the turbulent flow.

The Reynolds number describes the global behavior of flow, not its local behavior; in large domains, it is possible to have localized turbulent regions that do not extend to the whole domain.

4.3 Flow in a curved pipe

Flow in a curved tube experiences a secondary flow in the plane perpendicular to the axial direction. The curved tube was analyzed by Dean(1972) in toroidal coordinates. When a fluid moves in a curved path, each particle of the fluid is acted by a centrifugal force. Centrifugal force is directly proportional to the square of the

body's velocity and inversely proportional to the radius of the curvature of its path. The definition of a fluid particle, as used here, is a tiny element of mass of the fluid, which displays the general characteristic properties of a large sample of fluid. Under the action of this centrifugal force, each of the particles is forced toward the outside of the curved path.

4.3.1 Vortex generation

Since the velocity of the fluid particles close to the axis is higher compared to the one away from the edges, and the centrifugal force acting on the particles closer to the axis is greater, the fluid particles closer to the center of the pipe will be forced harder under the action of the centrifugal force to move toward the outside of the bend than the particles which are located in the region closer to the wall, i.e., the lower velocity region. So the higher velocity particles move toward the outside of the bend, they displace the lower velocity particles near the outer wall. Since the cross-section of the pipe is closed and conditions within the cross-section must be satisfied, the lower velocity particles along the wall of the pipe will be forced toward the inside of the bend. This motion set up by the outward and inward movement of the particles constitutes the secondary flow.

4.3.2 Dean Number

Amongst the first pioneering works on curved pipes, Eustice(1910) was able to prove experimentally the existence of secondary motion in curved pipes by injecting colored

dye in the flow. Dean(1927) studied the flow analytically and found a solution to the incompressible Navier-Stokes equations valid in the small-curvature assumption. In this way, a mathematical proof of the existence of the secondary motion in the form of two symmetric counter-rotating vortices named after Dean was given. Moreover, Dean showed that there is only one governing parameter in the flow, named Dean number after him:

$$De = Re\sqrt{\delta} = \frac{\sqrt{(\text{inertial force})(\text{centrifugal force})}}{\text{viscous forces}}$$

where $\delta = \frac{a}{R}$ is the curvature, a : the radius of cross-section of the pipe, R : the radius of curvature at the pipe centerline.
The Dean number can be regarded as a measure of the intensity of the secondary flow. The analytical solution proposed by Dean has the further limit of being valid only for $De << 37.94$.

4.3.3 Flow transitions in Straight and curved pipes

Taylor(1929) and White(1929) showed experimentally that the Reynolds number at which laminar-turbulent transition occurs in helical pipes increases with the curvature. Indeed, the flow in curved pipes is much more stable than that in a straight pipe, and the critical Reynolds number maybe even twice (or more) as large (Taylor, 1929, Berger et al., 1983). This result was confirmed by the work by Sreenivasan and Strykowski (1983), who observed that the flow in the straight section downstream of the helix remains laminar for values of the Reynolds number higher than the critical one for straight pipes.

4.4 Test Examples

4.4.1 Example 1

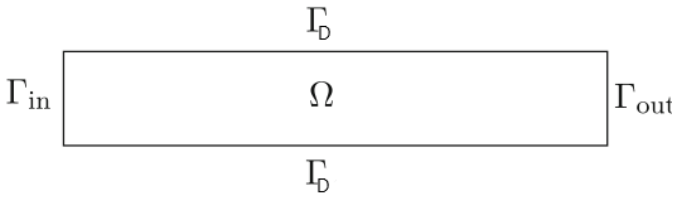
Numerical Simulation of flow in a rectangle in the DUNE [5] Framework is presented. The geometry is created using GMSH [6] and an Unstructured grid is used for meshing. The computational domain is a rectangle in two dimensions of length $2.2cm$ and breadth $0.41cm$. The viscosity $\nu = 0.001gs^{-1}cm^{-1}$. The velocity is of degree two, and pressure is considered degree one. The number of elements in the mesh is 2821. The number of edges in the mesh is 8220. The total number of vertices in the mesh is 24903. The number of constrained degrees of freedom is 842. Crank Nicholson Scheme is used for discretization. The nonlinear solver used is Newton. Bi-Conjugate Gradient Stabilized with ILU Preconditioner is used as the linear solver. The simulation is done for $T = 4$ seconds with the time step $\Delta t = 0.05$ seconds. The CPU time taken for the simulation is 104.81 seconds.

Boundary Conditions

$$\begin{aligned} \mathbf{u} &= 0 && \text{on } \Gamma_D \\ \mathbf{u} &= \begin{pmatrix} 3 \cdot x_2 \frac{0.41 - x_2}{0.41^2} \cdot sin(\frac{\pi t}{6}) \\ 0 \end{pmatrix} && \text{on } \Gamma_{in} \\ \nu \nabla \mathbf{u} \mathbf{n} - p\mathbf{n} &= 0 && \text{on } \Gamma_{out} \end{aligned}$$

Initial conditions $(\mathbf{u}^0, p^0) = (\mathbf{0}, 0)$

Domain



Observations

- The Reynolds number of the flow is 1650, and the flow is laminar.

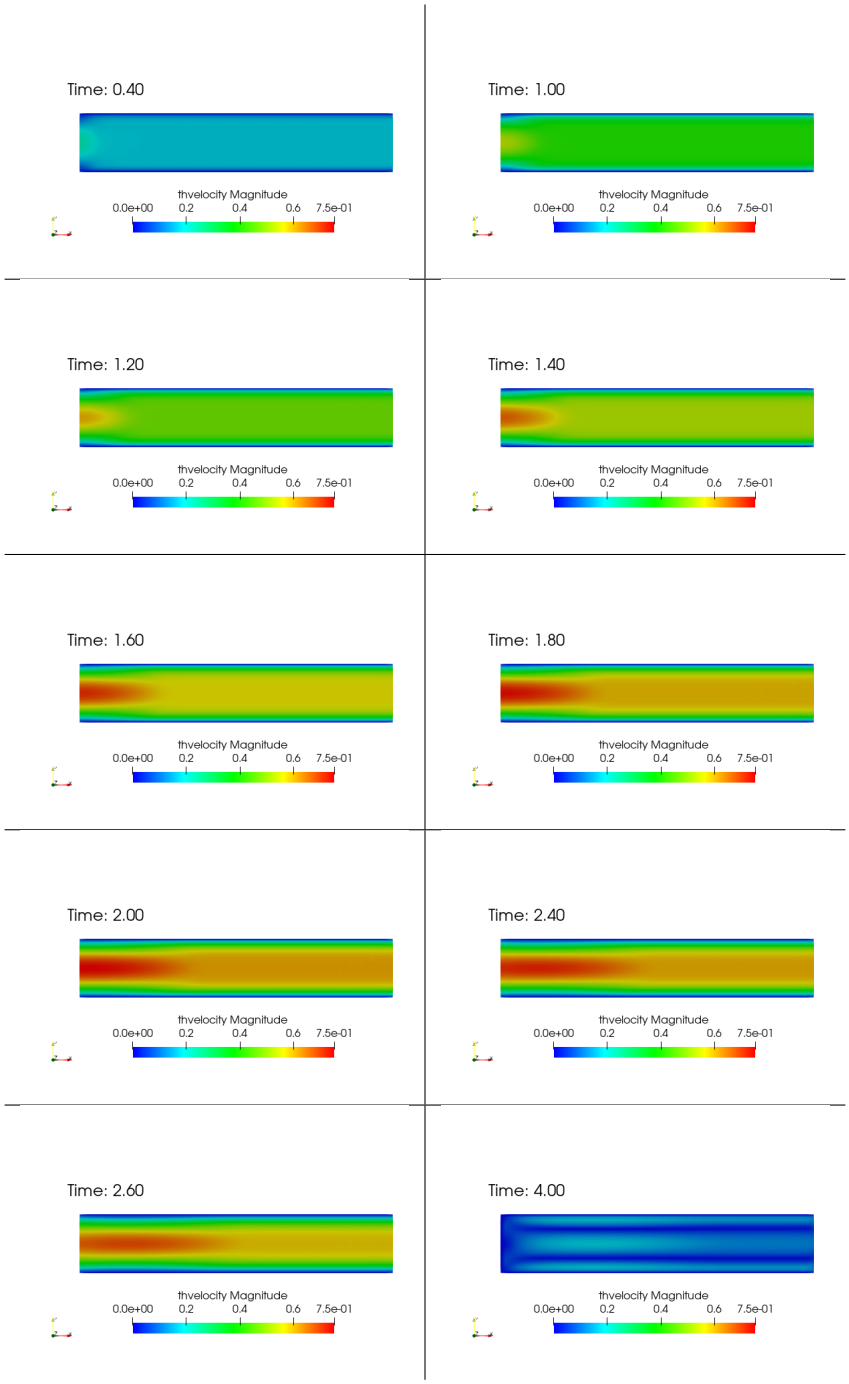


Table 4.1: Magnitude of Velocity at different time steps

4.4.2 Example 2

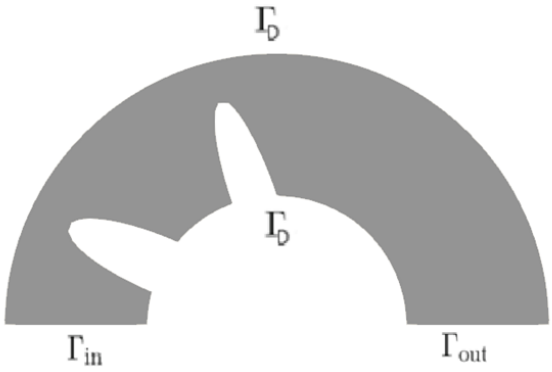
Numerical Simulation of fluid flow in a semicircular domain [1] in the DUNE [5] Framework is presented. The geometry is created using GMSH [6] and an Unstructured grid is used for meshing. The computational domain is a semicircular pipe with smooth disturbances in two dimensions with cross-sectional radius $a = 1.6cm$ and curvature radius $R = 2.9cm$. The kinematic viscosity $\nu = 0.4\ gs^{-1}cm^{-1}$. The velocity is of degree two, and pressure is considered degree one. The number of elements in the mesh is 1936. The number of edges in the mesh is 3002. The total number of vertices in the mesh is 1067. The number of constrained degrees of freedom is 739. Crank Nicholson Scheme is used for discretization. The nonlinear solver used is Newton. Bi-Conjugate Gradient Stabilized with ILU Preconditioner is used as the linear solver. The simulation is done for $T = 6$ seconds with the time step $\Delta t = 0.01$ seconds. The CPU time taken for the simulation is 247.824 seconds.

Boundary Conditions

$$\begin{aligned} \mathbf{u} &= 0 && \text{on } \Gamma_D \\ \mathbf{u} &= \begin{pmatrix} 0 \\ 3 \cdot x_1 \cdot \frac{(3.2-x_1)}{3.2 \cdot 3.2} \cdot (sin(\frac{\pi t}{6.0})) \end{pmatrix} && \text{on } \Gamma_{in} \\ \nu \nabla \mathbf{u} \mathbf{n} - p\mathbf{n} &= 0 && \text{on } \Gamma_{out} \end{aligned}$$

Initial conditions $(\mathbf{u}^0, p^0) = (\mathbf{0}, 0)$

Domain



Observations

- The Reynolds number of the flow is 22.875, 228.75 for viscosity 0.4 and 0.04 respectively.
- The Dean number for the viscosity $\nu = 0.4\ gs^{-1}cm^{-1}$ is 18.894. The flow is completely unidirectional for low Dean numbers ($De < 40 \sim 60$) and a similar flow is observed in the simulation.
- The Dean number for viscosity $\nu = 0.04gs^{-1}cm^{-1}$ is 188.94. A secondary instability appears for $De > 75 \sim 200$ and in our case, at time $T = 4$ seconds, there is a vortex formation leading to backflow instability.

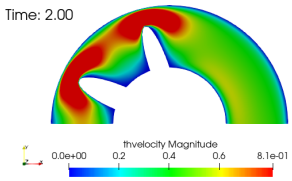
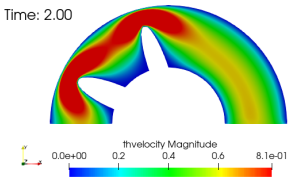
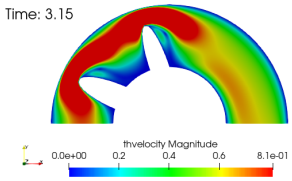
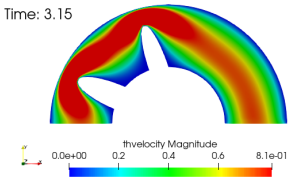
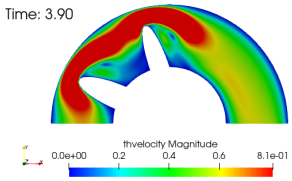
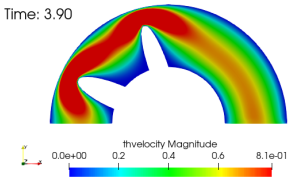
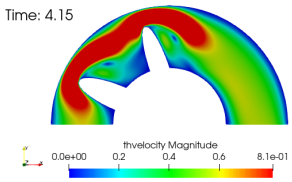
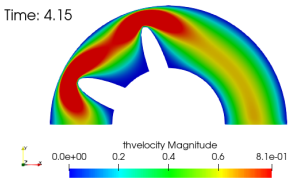
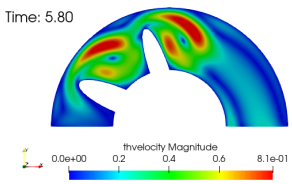
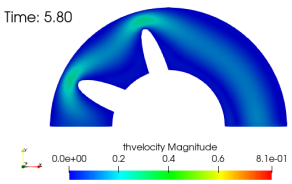
viscosity=0.04	viscosity=0.4
	
	
	
	
	

Table 4.2: Magnitude of Velocity at different time steps

4.4.3 Example 3

Numerical Simulation of fluid flow in a cylindrical domain [2] in the DUNE [5] Framework is presented. The geometry is created using GMSH [6] and an Unstructured grid is used for meshing. The computational domain is a cylindrical channel in three dimensions with cross-sectional radius $R = 1$ cm and length $L = 4$ cm. The dynamic viscosity $\mu = 0.4\ gcm^{-1}s^{-1}$, density $\rho = 1.06\ gcm^{-3}$, blood pressure drop $\Delta P = 1Pa$. The velocity is of degree three and pressure is considered degree two. The number of elements in the mesh is 1570. The total number of vertices in the mesh is 296. The number of constrained degrees of freedom is 4168. Crank Nicholson Scheme is used for discretization. The nonlinear solver used is Newton. Bi-Conjugate Gradient Stabilized with ILU Preconditioner is used as the linear solver. The total time taken for the simulation is 2 seconds with the time step $\Delta t = 0.01$ seconds. The CPU time taken for the simulation is 3214.88 seconds.

Boundary Conditions

$$\mathbf{u} = 0 \qquad \text{on } \Gamma_D$$

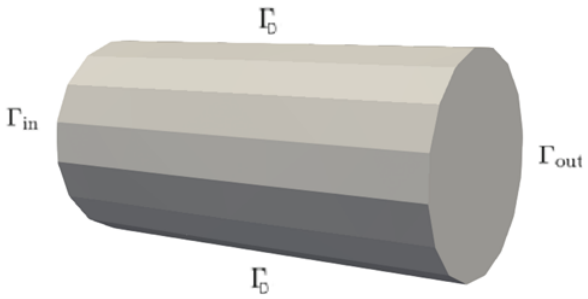
$$\mathbf{u} = \begin{pmatrix} (1 - (\frac{r}{R})^2) \cdot a \cdot Q \\ 0 \\ 0 \end{pmatrix} \qquad \text{on } \Gamma_{in}$$

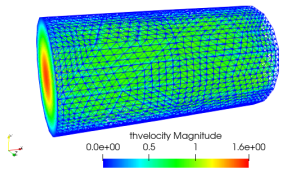
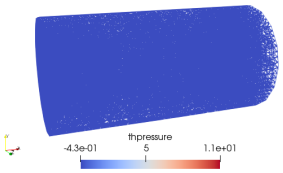
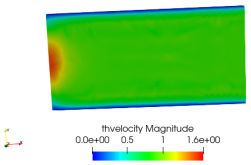
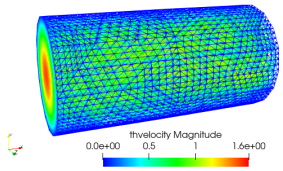
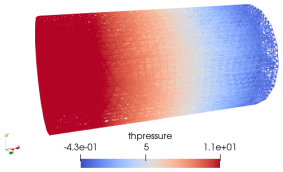
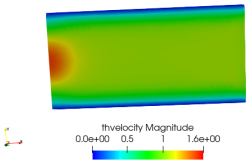
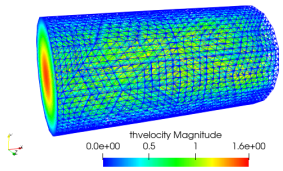
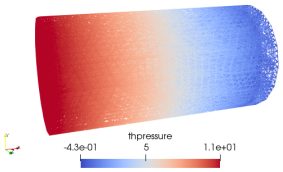
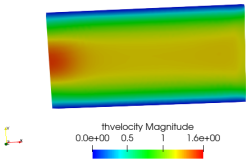
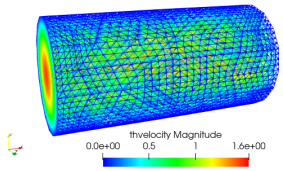
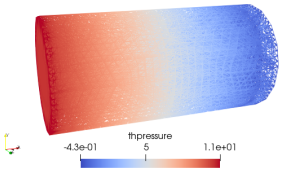
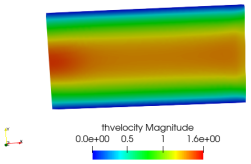
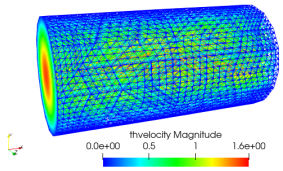
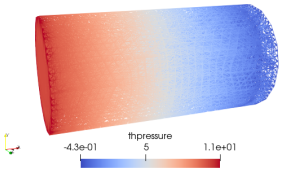
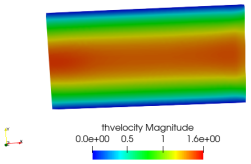
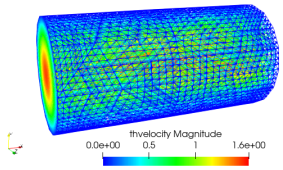
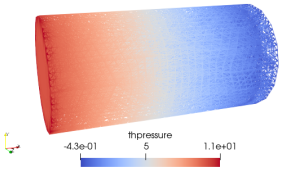
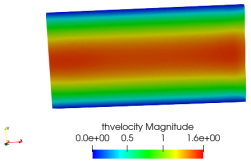
$$\nu \nabla \mathbf{u}\mathbf{n} - p\mathbf{n} = 0, \quad p = 0 \qquad \text{on } \Gamma_{out}$$

where, $Q := \frac{\pi R^4 \Delta P}{8\mu L} = 2.183\ cm^3 s^{-1}$ is the volumetric inflow rate and $a := \frac{2}{\pi R^2}$.
$\mathbf{n}$ denotes the unit outer normal vector to the domain. The Reynolds number of the flow is 21.83.

Initial conditions $(\mathbf{u}^0, p^0) = (\mathbf{0}, 0)$

Domain



velocity	pressure	clip view
Time: 0.01	Time: 0.01	Time: 0.01
		
Time: 0.05	Time: 0.05	Time: 0.05
		
Time: 0.10	Time: 0.10	Time: 0.10
		
Time: 0.15	Time: 0.15	Time: 0.15
		
Time: 0.20	Time: 0.20	Time: 0.20
		
Time: 0.25	Time: 0.25	Time: 0.25
		

4.4.4 Example 4

Numerical Simulation of fluid flow in a semi-circular domain [2] in the DUNE [5] Framework is presented. The geometry is created using GMSH [6] and an Unstructured grid is used for meshing. The computational domain is a semi-circular domain with three dimensions: cross-sectional radius $R = 1.6\ cm$, curvature radius 2.9 cm, and domain length $L = 12.2\ cm$. The dynamic viscosity $\mu = 0.4\ gcm^{-1}s^{-1}$, density $\rho = 1.06\ gcm^{-3}$, blood pressure drop $\Delta P = 1Pa$.

The velocity is of degree three and pressure is considered degree two. The number of elements in the mesh is 1114. The total number of vertices in the mesh is 227. The number of constrained degrees of freedom is 5383. Crank Nicholson Scheme is used for discretization. The nonlinear solver used is Newton. Bi-Conjugate Gradient Stabilized with ILU Preconditioner is used as the linear solver. The simulation is done for $T = 2$ seconds with time $\Delta t = 0.01$ seconds. The CPU time taken for the simulation is 2739.57 seconds.

Boundary Conditions

$$\mathbf{u} = 0 \qquad \text{on } \Gamma_D$$

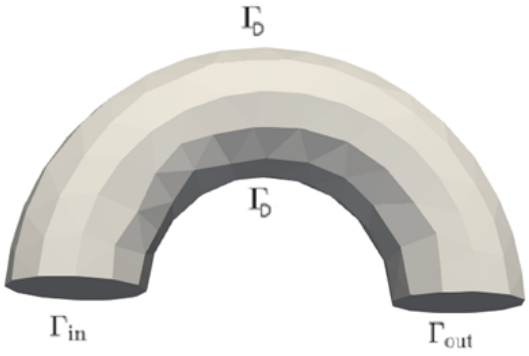
$$\mathbf{u} = \begin{pmatrix} 0 \\ (1 - (\frac{r}{R})^2) \cdot a \cdot Q \\ 0 \end{pmatrix} \qquad \text{on } \Gamma_{in}$$

$$\nu \nabla \mathbf{u}\mathbf{n} - p\mathbf{n} = 0, \quad p = 0 \qquad \text{on } \Gamma_{out}$$

where, $Q := \frac{\pi R^4 \Delta P}{8\mu L} = 5.2703\ cm^3 s^{-1}$ is the volumetric inflow rate and $a := \frac{2}{\pi R^2}$.
$\mathbf{n}$ denotes the unit outer normal vector to the domain.

Initial conditions $(\mathbf{u}^0, p^0) = (\mathbf{0}, 0)$

Domain



Observations

- The Reynolds number of the flow is 161.
- The Dean number of the flow is 33.19 and the flow is laminar as expected.
- The maximal axial speed is pushed toward the outer bend of the pipe due to centrifugal force arising as a result of the curvature. This can be clearly seen if we compare the flow at $T = 0.40$ second and $T = 1.25$ second.

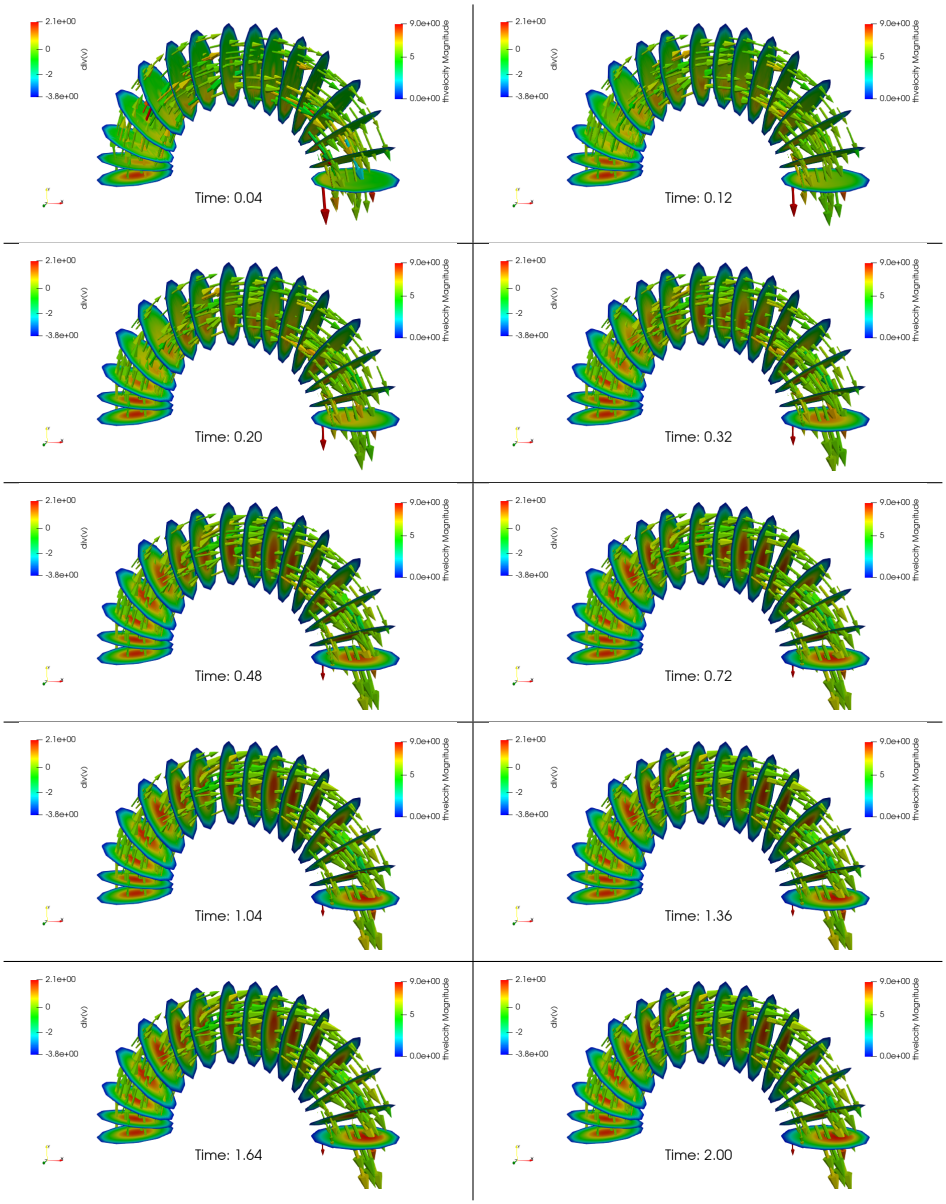


Table 4.3: Magnitude of Velocity at different time steps

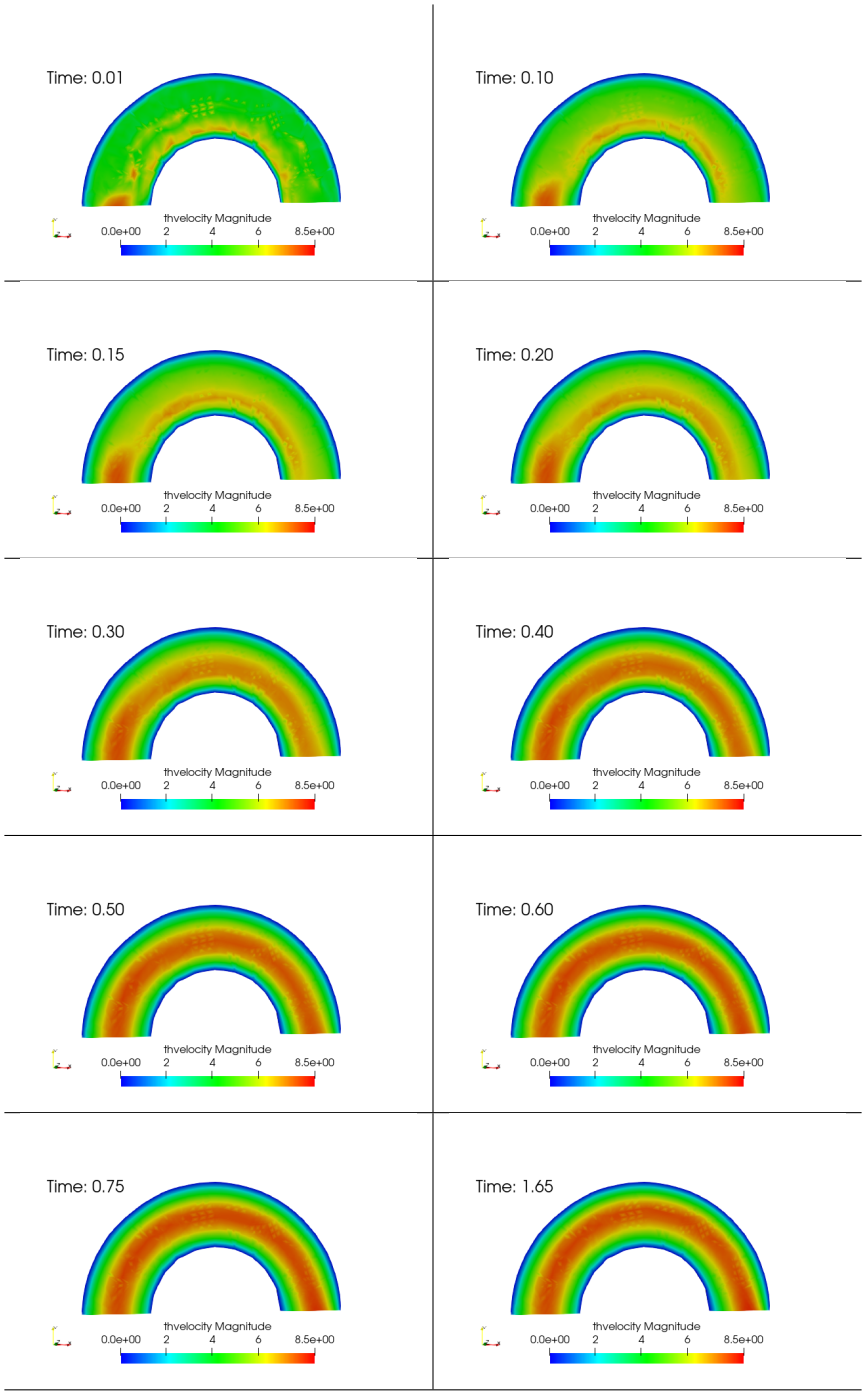


Table 4.4: Clip view of velocity magnitude

Chapter 5

Neural Networks

5.1 Artificial Neural Networks

Neural networks were proposed in 1944 by Warren McCullough and Walter Pitts, two University of Chicago researchers. Neural nets are a machine learning method in which a computer learns to perform some task by analyzing training examples.

The general idea of a Neural Network is to find a model that, based on a set of N samples,

$$D = [x_1, ..x_N], [y_1, .., y_N]$$

will approximate a unknown function f, with

$$f(x_i) = y_i$$

as well as possible. Neural Network consists of the following:

- Layers: The model always consists of one input layer, an output layer, and at least one hidden layer. Each layer consists of several units, the neurons. The amount of units in each layer defines the dimension of the layer.

- Weights: The layers are connected with weighted transitions that can be modeled by a product of a weight matrix with a vector that stands for the previous layer's output.

- Activation function: The activation functions in the neurons model a threshold that decides whether the information a neuron got will be relevant for further calculations. If the information is relevant, it will be passed on to the next layer until the output layer is reached.

During the so-called training phase, the model is given a sample data set D where the matrix

$$[x_1, ..., x_N]$$

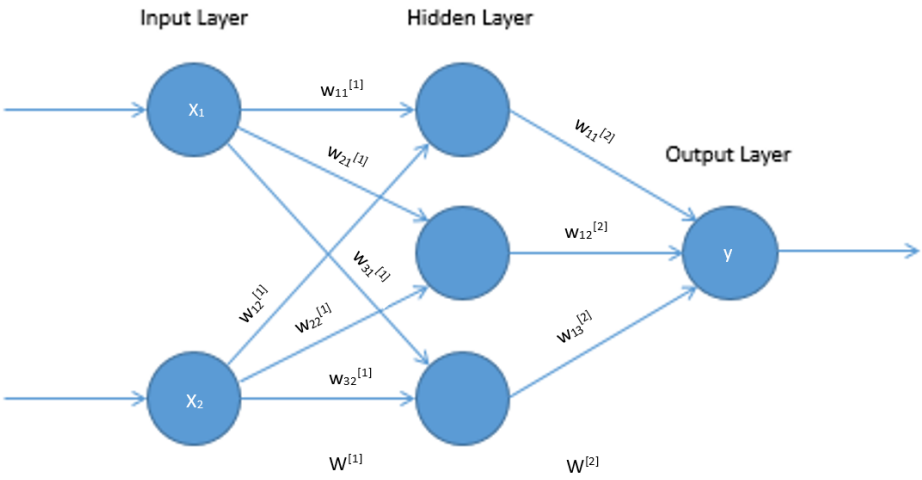
is the input and

$$[y_1, \ldots, y_N]$$

the output matrix. After processing each sample $(x_i, y_i) \in D$, the distance of the output that the network produces the actual desired output y_i, which is measured by some loss function, for example, the L^2norm or the Euclidean norm. According to the loss, the weights of the transitions are modified to achieve a better result in the next iteration.

5.1.1 Example of a Neural Network

Let's take an example of a neural network with two input features, one hidden layer, and one output node.



$w_{ij}^{[l]}$ is the weight associated with a node i of layer l when it receives an input from node j of the previous layer and the bias associated with the node i of layer l. The weight matrix associated with layer l is represented by $W^{[l]}$, and the dimension of this matrix is the number of nodes in layer l x number of nodes in layer $l-1$. For instance, the weight matrix associated with layer 1 is:

$$W^{[1]} = \begin{pmatrix} w_{11}^{[1]} & w_{12}^{[1]} \\ w_{21}^{[1]} & w_{22}^{[1]} \\ w_{31}^{[1]} & w_{32}^{[1]} \end{pmatrix}$$

and the bias matrix in layer 1 is

$$\begin{pmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \end{pmatrix}$$

The output at layer l is given by the matrix $A^{[l]}$ given by applying the activation function σ to the matrix obtained by multiplying the weights with the input from the previous layer,

$$A^{[l]} = \sigma(W^{[l]}A^{[l-1]} + b^{[l]})$$

with $A^{[0]}$ given by the input features. In this example, $A^{[0]} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$.

For the given network, we have the output y to be

$$y = A^{[2]} = \sigma(W^{[2]}A^{[1]} + b^{[2]}) = \sigma(W^{[2]}(\sigma(W^{[1]}A^{[0]} + b^{[1]})) + b^{[2]})$$

This procedure of calculating the output y from the input training data is known as the forward pass. The loss function is given by $C(w, b)$. The weights are then updated by using the backpropagation algorithm.

5.2 Backpropagation Algorithm

For a layer l let $Z^{[l]} = W^{[l]}A^{[l-1]} + b^{[l]}$, hence $A^{[l]} = \sigma(Z^{[l]})$.

$$\frac{\partial C}{\partial Z^{[l]}} = \frac{\partial C}{\partial A^{[l]}} \frac{\partial A^{[l]}}{\partial Z^{[l]}} = \sigma^{'}(Z^{[l]}) \frac{\partial C}{\partial A^{[l]}}$$

$$\frac{\partial C}{\partial W^{[l]}} = \frac{\partial C}{\partial Z^{[l]}} \frac{\partial Z^{[l]}}{\partial W^{[l]}} = \frac{\partial C}{\partial Z^{[l]}} (A^{[l-1]})^T$$

$$\frac{\partial C}{\partial b^{[l]}} = \frac{\partial C}{\partial Z^{[l]}} \frac{\partial Z^{[l]}}{\partial b^{[l]}} = \frac{\partial C}{\partial Z^{[l]}}$$

$$\frac{\partial C}{\partial W^{[l]}} = \frac{\partial C}{\partial Z^{[l]}} \frac{\partial Z^{[l]}}{\partial W^{[l]}} = \frac{\partial C}{\partial Z^{[l]}} (A^{[l-1]})^T$$

$$\frac{\partial C}{\partial A^{[l-1]}} = \frac{\partial C}{\partial Z^{[l]}} \frac{\partial Z^{[l]}}{\partial A^{[l-1]}} = (W^{[l]})^T \frac{\partial C}{\partial Z^{[l]}}$$

Let l be the outermost layer, $\frac{\partial C}{\partial A^{[l]}}$ can be computed. The algorithm is as follows.

Algorithm 5 Backpropagation Algorithm

1: **while** $l \geq 1$ **do**

2: $\frac{\partial C}{\partial Z^{[l]}} = \sigma'(Z^{[l]}) \frac{\partial C}{\partial A^{[l]}}$

3: $\frac{\partial C}{\partial W^{[l]}} = \frac{\partial C}{\partial Z^{[l]}} (A^{[l-1]})^T$

4: $\frac{\partial C}{\partial b^{[l]}} = \frac{\partial C}{\partial Z^{[l]}}$

5: $W^{[l]} = W^{[l]} - \alpha \frac{\partial C}{\partial W^{[l]}}$

6: $b^{[l]} = b^{[l]} - \alpha \frac{\partial C}{\partial b^{[l]}}$

7: $\frac{\partial C}{\partial A^{[l-1]}} = (W^{[l]})^T \frac{\partial C}{\partial Z[l]}$

8: $l = l - 1$

9: **end while**

The weights at each layer are updated in lines 5 and 6 using the gradient descent algorithm to minimize the cost function C. α is the learning rate.
Hence at each epoch, forward feed takes place to compute the output and then the backpropagation algorithm is implemented to update the weights in each layer.

5.3 Optimization Algorithms

Gradient descent involves updating the weights by taking the sum of the gradient of the Cost function with respect to all the training examples.

$$w = w - \frac{\alpha}{n} \sum_{i=1}^{n} \frac{\partial C_i}{\partial w}$$

where C_i is the loss associated with the training example i. When the size of the training set is large, evaluating the sums of gradients becomes computationally very expensive. This problem can be addressed by the mini-batch Stochastic Gradient Descent(SGD) Algorithm. SGD replaces the actual gradient (calculated from the entire data set) with an estimate thereof (calculated from a randomly selected subset of the data).

Algorithm 6 mini-batch SGD Algorithm

randomly initialize weights W for every layer and initial learning rate α_0
set epoch $t = 0$ and update $n = 0$
while not converged **do**
  randomly shuffles train data into mini-batches
  for each mini-batch B and each layer l **do**
    for each $\mathbf{x} \in B$ **do**
      forward pass $\mathbf{x} \longrightarrow \mathbf{y}$
      backward pass $\{\mathbf{x}, \mathbf{y}\} \longrightarrow \frac{\partial C(W_n^{[l]};\mathbf{x})}{\partial W_n^{[l]}}$
    end for
    update model: $W_{n+1}^{[l]} = W_n^{[l]} - \frac{\alpha_t}{|B|} \sum_{x \in B} \frac{\partial C(W_n^{[l]};\mathbf{x})}{\partial W^{[l]}}$
    n=n+1
  end for
  $\alpha_t \longrightarrow \alpha_{t+1}$
  $t = t + 1$
end while

Initialization of Weights: In practice, it is empirically found that random initialization works well for neural networks. At the beginning of the Algorithm, all network parameters are randomly set according to a uniform or Gaussian distribution centered at zero.

In mini-batch SGD, the gradient estimates are more noisy at each model update. If the batch size is very small, these noises may fluctuate in the learning process and eventually slow down the convergence of learning. But these fluctuations may be beneficial for the learning process to escape from poor initialization, saddle points, or even bad local optimums. We can parallelize the forward/backward passes of all samples within each mini-batch using bigger mini-batches. Hence choosing an

optimal batch size is important.

5.3.1 Gradient Descent using Momentum

If α is too small, then convergence is slow, and if it's too large, then we risk divergence or slow convergence due to oscillation. As the network becomes deep, we can find that $\frac{\partial C}{\partial W^{[l]}}$ may be substantially different from the gradient of the loss with respect to the weights in the first layer.

$$\begin{aligned}
\frac{\partial C}{\partial W^{[l]}} &= \frac{\partial C}{\partial Z^{[l]}}(A^{[l-1]})^T \\
\frac{\partial C}{\partial Z^{[1]}} &= \frac{\partial A^{[1]}}{\partial Z^{[1]}} \cdot \frac{\partial Z^{[2]}}{\partial A^{[1]}} \cdots \frac{\partial A^{[l-1]}}{\partial Z^{[l-1]}} \cdot \frac{\partial Z^{[l]}}{\partial A^{[1-1]}} \cdot \frac{\partial A^{[l]}}{\partial Z^{[l]}} \cdot \frac{\partial C}{\partial A^{[l]}} \\
&= \frac{\partial A^{[1]}}{\partial Z^{[1]}} \cdot W^{[2]} \cdots \frac{\partial A^{[l-1]}}{\partial Z^{[l-1]}} \cdot W^{[l]} \cdot \frac{\partial A^{[l]}}{\partial Z^{[l]}} \cdot \frac{\partial C}{\partial A^{[l]}}
\end{aligned} \tag{5.1}$$

This can be seen from 6.1 where the output gradient is multiplied by all the weight matrices of the network and is fed back through the derivatives of the activation function. This can lead to a problem of exploding or vanishing gradients, in which the back-propagated gradient is much too big or small to be used in an update rule with the same step size. So, we'll consider having an independent step-size parameter for each weight and updating it based on a local view of how the gradient updates have been going.

Running Averages It is a computational strategy for estimating a possibly weighted average of a sequence of data. If the sequence of data is $\alpha_1, \alpha_2, \ldots$ then the sequence

of running averages $A_0, A_1, A_2, \ldots$ can be defined as

$$A_0 = 0$$
$$A_t = \gamma_t A_{t-1} + (1 - \gamma_t)\alpha_t$$

where $\gamma_t \in (0, 1)$. If γ_t is a constant, then this is a moving average in which,

$$A_T = \sum_{i=1}^{T} \gamma^{T-t}(1 - \gamma)\alpha_t$$

So the inputs α_t closer to the end of the sequence T have more effect on A_T than early inputs.

Gradient descent with momentum

GD with momentum for t^{th} update is given as

$$M_t = \gamma M_{t-1} + (1 - \gamma)\frac{\partial C}{\partial W_{t-1}^{[l]}}$$
$$W_t^{[l]} = W_{t-1}^{[l]} - \alpha M_t$$

where $M_0 = 0$.

5.3.2 ADAM Optimization Algorithm

The Adaptive Moment estimation(ADAM) Algorithm is a method for stochastic optimization that only requires first-order gradients and uses the estimations of the first and second moments of the gradient to adapt the learning rate for each weight of the neural network.

Algorithm 7 ADAM Algorithm

randomly initialize weights W for every layer and initial learning rate α_0

set epoch $t = 0$, $n = 0$ and $\mathbf{u}_0 = \mathbf{v}_0 = 0$

while not converged **do**

randomly shuffles train data into mini-batches

for each mini-batch B and each layer l **do**

for each $\mathbf{x} \in B$ **do**

forward pass $\mathbf{x} \longrightarrow \mathbf{y}$

backward pass $\{\mathbf{x}, \mathbf{y}\} \longrightarrow \frac{\partial C(W_n^{[l]};\mathbf{x})}{\partial W^{[l]}}$

end for

$\mathbf{g_n} = \frac{\alpha_t}{|B|} \sum_{x \in B} \frac{\partial C(W_n^{[l]};\mathbf{x})}{\partial W^{[l]}}$

$\mathbf{u}_{n+1} = \alpha \mathbf{u}_n + (1 - \alpha)\mathbf{g_n}$

$\mathbf{v}_{n+1} = \beta \mathbf{v}_n + (1 - \beta)\mathbf{g_n} \odot \mathbf{g_n}$

$\hat{\mathbf{u}}_{n+1} = \frac{\mathbf{u}_{n+1}}{1-\alpha^{n+1}}$ and $\hat{\mathbf{v}}_{n+1} = \frac{\mathbf{v}_{n+1}}{1-\beta^{n+1}}$

update model: $W_{n+1}^{[l]} = W_n^{[l]} - \eta \hat{\mathbf{u}}_{n+1} \odot ((\hat{\mathbf{v}}_{n+1} + \epsilon^2)^{-\frac{1}{2}})$

n=n+1

end for

$\alpha_t \longrightarrow \alpha_{t+1}$

t=t+1

end while

Here $\odot$ is element-wise matrix multiplication. $\mathbf{g_n}$ denotes the averaged gradients over a mini-batch. $\mathbf{u}_{n+1}, \mathbf{v}_{n+1}$ denote the exponential moving averages of the first and second moments of the gradients over time. $\hat{\mathbf{u}}_{n+1}$ and $\hat{\mathbf{v}}_{n+1}$ are the unbiased estimators of $\mathbf{u}_{n+1}$ and $\mathbf{v}_{n+1}$. This is required as initially both $\mathbf{u}_0, \mathbf{v}_0$ are set to zero. Since α, β is closer to one, $\hat{\mathbf{u}}_{n+1}$, $\hat{\mathbf{v}}_{n+1}$ tend to be more biased towards zero, so dividing by $1 - \alpha^{n+1}$ resolves it. The i^{th} element of the unbiased estimators is given

by

$$\hat{u}_{n+1}(i) = \frac{u_{n+1}(i)}{1-\alpha^{n+1}} = \frac{g_n(i) + \alpha \cdot g_{n-1}(i) + \alpha^2 \cdot g_{n-2}(i) + \ldots}{1 + \alpha + \alpha^2 + \ldots}$$
$$\hat{v}_{n+1}(i) = \frac{v_{n+1}(i)}{1-\beta^{n+1}} = \frac{g_n^2(i) + \alpha \cdot g_{n-1}^2(i) + \alpha^2 \cdot g_{n-2}^2(i) + \ldots}{1 + \beta + \beta^2 + \ldots}$$

The averaged gradients $g_n(i)$ are slowly changing over n. Hence, $\mathbb{E}[g_{n-k}(i)] = \mathbb{E}[g_n(i)]$ for all k which gives

$$\mathbb{E}[\hat{u}_{n+1}(i)] = \mathbb{E}[g_n(i)] \qquad \mathbb{E}[\hat{v}_{n+1}(i)] = \mathbb{E}[{g_n}^2(i)]$$

The ϵ in the weight updation step is added to ensure numerical stability when $\hat{v}_{n+1}(i)$ becomes extremely small. The weight updation for the i^{th} parameter is

$$\Delta \mathbb{W}_i = \eta \frac{\hat{u}_{n+1}(i)}{\sqrt{\hat{v}_{n+1}(i)}}$$

By using $\mathbb{E}[x^2] = (\mathbb{E}[x])^2 + var[x]$ we get

$$||\Delta W_i||^2 \simeq \eta^2 \frac{(\mathbb{E}[\hat{u}_{n+1}(i)])^2}{\mathbb{E}[\hat{v}_{n+1}(i)]} = \frac{\eta^2 (\mathbb{E}[g_n(i)])^2}{(\mathbb{E}[g_n(i)])^2 + var[g_n(i)]}$$

Suppose the i^{th} parameter fluctuates around an optimum. In that case, its gradients are alternatively positive and negative, i.e., $\mathbb{E}[g_n(i)] \longrightarrow 0$ and $var[g_n(i)]$ is large. The ADAM Algorithm will automatically reduce the update for i^{th} parameter as $||\Delta W_i||^2 \longrightarrow 0$. On the other hand, if the i^{th} parameter is still far away from the optimum, all gradients are either positive or negative, so $(E[g_n(i)])^2$ tend to be large and the $var[g_n(i)]$ is small. Hence $||\Delta W_i||^2$ is large in this case and the update for this parameter is relatively large as well.

5.3.3 BFGS Optimization Algorithm

Quasi Newton Methods: Quasi-Newton methods are a class of optimization algorithms that require only the gradient of the objective function to be supplied at each iterate. By measuring the changes in gradients, they determine the descent direction by preconditioning the gradient with curvature information and constructing a model of the objective function that is good enough to produce superlinear convergence. The most popular quasi-Newton Method is the BFGS method, named after its discoverers Broyden, Fletcher, Goldfarb, and Shanno.

The BFGS Method:

Quadratic model of the objective function at current iterate x_k is given as

$$m_k(p) = f_k + \nabla f_k^T + \frac{1}{2} p^T B_k p$$

Here B_k is a n x n symmetric positive definite matrix updated at each iteration. Value and gradient of this model at $p = 0$ match f_k and ∇f_k. The minimizer of p_k of the quadratic model is p_k is

$$p_k = -B_k^{-1} \nabla f_k$$

and is used as the search direction, and the new iterate is

$$x_{k+1} = x_k + \alpha_k p_k$$

Instead of constructing B_k afresh every iteration, Davidon proposed to update it simply to account for the curvature measured during the most recent step. The

Taylor series of $f(x)$ around an iterate is

$$f(x_{k+1}) = f(x_k + \Delta x) \sim f(x_k) + \nabla f(x_k)^T \Delta x + \frac{1}{2}\Delta x^T B \Delta x$$

Taking the gradient of this approximation gives

$$\nabla f_{k+1} \sim \nabla f_k + B\Delta x$$

If $s_k = x_{k+1} - x_k = \alpha_k p_k$ $y_k = \nabla f_{k+1} - \nabla f_k$ then at every iterate the Hessian approximation is chosen to satisfy

$$B_{k+1} s_k = y_k$$

which is the secant equation. The curvature $s_k^T y_k > 0$ must be satisfied for the symmetric positive definite matrix B_{k+1} to map s_k into y_k. When f is strongly convex, the inequality will be satisfied for any two points x_k and x_{k+1}. But if the function is non-convex, then we need to enforce the inequality explicitly by imposing restrictions on the line search procedure that chooses the step length α. In fact, the inequality is satisfied if we impose the Wolfe condition on the line search. A step length α_k is said to satisfy the Wolfe condition, restricted to the direction p_k if

$$f(\mathbf{x_k} + \alpha_k \mathbf{p_k}) \leq f(\mathbf{x_k}) + c_1 \alpha_k \mathbf{p_k} \nabla f(x_k)$$

$$-\mathbf{p_k}^T \nabla f(\mathbf{x} + \alpha_k p_k) \leq -c_2 \mathbf{p_k}^T \nabla f(\mathbf{x_k})$$

with $0 < c_1 < c_2 < 1$, usually taken to be 10^{-4} and 0.9 respectively. So when the curvature condition is satisfied, the secant equation always has a solution B_{k+1}.

But it is an under-determined system since the $n(n+1)/2$ unknown degrees of a symmetric matrix exceed the n equations. To determine B_{k+1} uniquely, we impose an additional condition that B_{k+1} must be chosen closest to the current matrix B_k. In other words, we solve the optimization problem

$$\min_B ||B - B_k||$$

$$\text{subject to } B = B^T, \quad Bs_k = y_k$$

A norm that allows an easy solution to the minimization problem is the weighted Frobenius norm $||A||_W = ||W^{\frac{1}{2}} A W^{\frac{1}{2}}||_F$, with the weight matrix W begin chosen as any matrix satisfying the secant equation. With this, we get a unique solution as

$$B_{k+1} = (I - \frac{y_k s_k^T}{y_k^T s_k}) B_k (I - \frac{s_k y_k^T}{y_k^T s_k}) + \frac{y_k y_k^T}{y_k^T s_k}$$

The inverse of B_k, denoted by $H_k = B_k^{-1}$ must be symmetric and positive definite and must satisfy the secant equation $H_{k+1} y_k = s_k$. The condition of closeness to H_k can be imposed, and H_{k+1} can be found by solving the following optimization problem analogous to B_k.

$$\min_H ||H - H_k||$$

$$\text{subject to } H = H^T, \quad Hy_k = s_k$$

The norm is again the weighted Frobenius norm where the weight matrix W is now any matrix satisfying $Ws_k = y_k$ and the unique solution H_{k+1} is given by

$$H_{k+1} = (I - \frac{s_k y_k^T}{y_k^T s_k}) H_k (I - \frac{y_k s_k^T}{y_k^T s_k}) + \frac{s_k s_k^T}{y_k^T s_k}$$

The initial Hessian H_0 can be taken as the Hessian at the initial point calculated by finite differences at x_0 or can simply be set to the identity matrix.

Algorithm 8 BFGS Algorithm

1: Given starting point x_0, tolerance ϵ, inverse Hessian approximation H_0
2: set $k = 0$
3: **while** $||\nabla f_k|| > \epsilon$ **do**
4: Compute the search direction $p_k = -H_k \nabla f_k$
5: Compute α_k from a line search procedure to satisfy the Wolfe conditions.
6: Set $x_{k+1} = x_k + \alpha_k p_k$
7: Define $s_k = x_{k+1} - x_k$ and $y_k = \nabla f_{k+1} - \nabla f_k$
8: Compute $H_{k+1} = (I - \frac{s_k y_k^T}{y_k^T s_k}) H_k (I - \frac{y_k s_k^T}{y_k^T s_k}) + \frac{s_k s_k^T}{y_k^T s_k}$
9: $k = k + 1$
10: **end while**

Each iteration can be performed at a cost of $O(n^2)$ and the rate of convergence is superlinear. Even though Newton Method converges more rapidly, its cost per iteration is higher because of its need to compute the second derivatives.

5.4 Physics Informed Neural Network

A deep Learning Model can be constructed in either a data-driven or physics-informed manner. In a data-driven framework, the Deep learning model is constructed as a black box to learn a surrogate mapping from the formatted input $x \in R^m$ to the output $y \in R^n$. Physics-Informed Neural Networks (PINNs) are a scientific machine learning technique used to solve problems involving Partial Differ-

ential Equations(PDE). PINNs approximate PDE solutions by training a neural network to minimize a loss function; it includes terms reflecting the initial and boundary conditions along the space-time domain's boundary and the PDE residual at selected points in the domain (called collocation points). The basic concept behind PINN training is that it can be considered an unsupervised strategy that does not require labeled data, such as results from prior simulations or experiments. The PINN algorithm is essentially a mesh-free technique that finds PDE solutions by converting the problem of directly solving the governing equations into a loss function optimization problem. Many PDE constraints fit the following abstraction:

$$F(u(x,t)) = 0 \quad x \in \Omega \subset \mathbb{R}^d \quad t \in [0,T]$$

where F is a differential operator representing the PDE, $u(x,t)$ is the parameter of interest, Ω is the spatial domain.

5.5 Auto-differentiation

Automatic differentiation [7] is a set of techniques for evaluating derivatives (gradients) numerically. The method uses symbolic rules for differentiation, which are more accurate than finite difference approximations. Unlike a purely symbolic approach, automatic differentiation evaluates expressions numerically early in the computations rather than carrying out large symbolic computations. In other words, automatic differentiation evaluates derivatives at particular numeric values; it does not construct symbolic expressions for derivatives. Auto-differentiation can be carried through forward mode or backward mode.

5.5.1 Forward mode

Forward automatic mode differentiation [8] evaluates a numerical derivative by performing elementary derivative operations concurrently with the operations of evaluating the function itself. Consider the problem of evaluating the function $f(x_1, x_2) = ln(x_1) + x_1x_2 - sin(x_2)$ and its gradients at $x_1 = 2$ and $x_2 = 5$. For computing the derivative of f with respect to x_1 we start by associating with each intermediate variable v_i a derivative

$$\dot{v}_i = \frac{\partial v_i}{\partial x_1}$$

Forward Primal Trace	Forward Tangent Trace
$v_{-1} = x_1 \quad = 2$	$\dot{v}_{-1} = \dot{x_1} \quad = 1$
$v_0 = x_2 \quad = 5$	$\dot{v}_0 = \dot{x_2} \quad = 0$
$v_1 = ln(v_{-1}) \quad = ln(2)$	$\dot{v_1} = \frac{\dot{v}_{-1}}{v_{-1}} \quad = \frac{1}{2}$
$v_2 = v_{-1} \cdot v_0 \quad = 2 \cdot 5$	$\dot{v}_2 = \dot{v}_{-1} \cdot v_0 + \dot{v}_0 \cdot v_{-1} \quad = 1 \cdot 5 + 0 \cdot 2$
$v_3 = sin(v_0) \quad = sin(5)$	$\dot{v}_3 = \dot{v}_0 \cdot cos(v_0) \quad = 0 \cdot cos(5)$
$v_4 = v_1 + v_2 \quad = 0.693 + 10$	$\dot{v}_4 = \dot{v}_1 + \dot{v}_2 \quad = 0.5 + 5$
$v_5 = v_4 - v_3 \quad = 10.693 + 0.959$	$\dot{v}_5 = \dot{v}_4 - \dot{v}_3 \quad = 5.5 - 0$
$y = v_5 \quad = 11.652$	$\dot{y} = \dot{v}_5 \quad = 5.5$

To compute the partial derivative with respect to x_2, one has to traverse a similar computational graph. Therefore, when we have to compute the gradient of the function, the number of graph traversals is the same as the number of variables. This process can be slow for many applications when the objective function or nonlinear constraints depend on many variables.

5.5.2 Reverse mode

Reverse mode automatic differentiation [7] uses an extension of the forward mode computational graph to enable the computation of a gradient by a reverse traversal of the graph. As the software runs the code to compute the function and its derivative, it records operations in a trace data structure.

The reverse mode uses one forward traversal of a computational graph to set up the trace. Then it computes the entire gradient of the function in one traversal of the graph in the opposite direction. For problems with many variables, this mode is far more efficient. In reverse mode, along with each variable v_i we associate an adjoint variable $\bar{v}_i$

$$\bar{v}_i = \frac{\partial f}{\partial v_i}$$

If the variables v_1 and v_2 depend upon v_{-1} then the associated adjoint equation is

$$\begin{aligned}\frac{\partial f}{\partial v_{-1}} &= \frac{\partial f}{\partial v_1}\frac{\partial v_1}{\partial v_{-1}} + \frac{\partial f}{\partial v_2}\frac{\partial v_2}{\partial v_{-1}} \\ &= \bar{v}_1\frac{\partial v_1}{\partial v_{-1}} + \bar{v}_2\frac{\partial v_2}{\partial v_{-1}}\end{aligned}$$

We have $v_5 = f$

$$\bar{v}_5 = \frac{\partial f}{\partial v_5} = \frac{\partial f}{\partial f} = 1$$

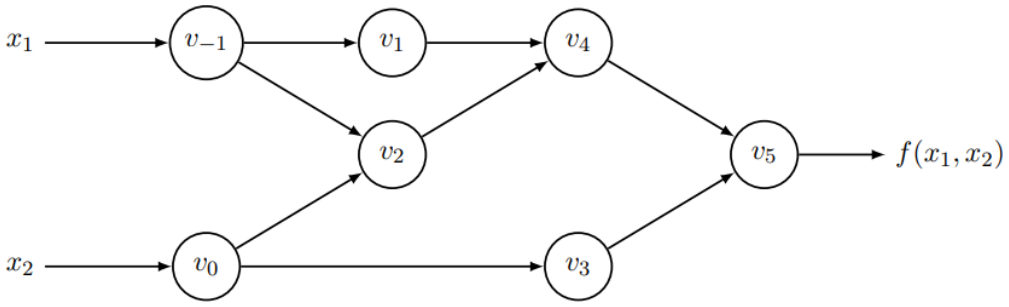
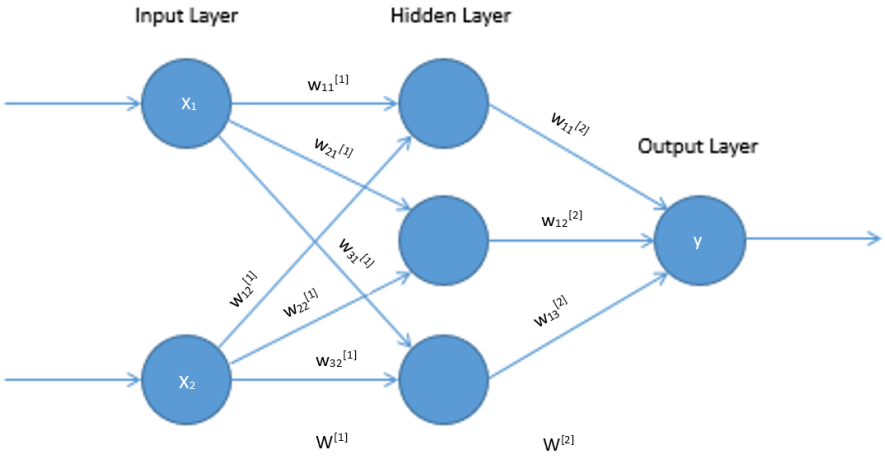


Figure 5.1: Computational graph of the example $f(x_1, x_2) = ln(x_1) + x_1 x_2 - sin(x_2)$

Forward Primal Trace	Reverse Adjoint Trace
$v_{-1} = x_1 \quad = 2$	$\bar{\mathbf{x}}_\mathbf{1} = \bar{\mathbf{v}}_\mathbf{-1} \quad = \mathbf{5.5}$
$v_0 = x_2 \quad = 5$	$\bar{\mathbf{x}}_\mathbf{2} = \bar{\mathbf{v}}_\mathbf{0} \quad = \mathbf{1.716}$
$v_1 = ln(v_{-1}) \quad = ln(2)$	$\bar{v}_{-1} = \bar{v}_2 \frac{\partial v_2}{\partial v_{-1}} + \bar{v}_1 \frac{\partial v_1}{\partial v_{-1}} \quad = \bar{v}_2 \cdot v_0 + \frac{\bar{v}_1}{v_{-1}} \quad = 5.5$
$v_2 = v_{-1} \cdot v_0 \quad = 2 \cdot 5$	$\bar{v}_0 = \bar{v}_3 \frac{\partial v_3}{\partial v_0} + \bar{v}_2 \frac{\partial v_2}{\partial v_0} \quad = \bar{v}_3 \cdot cos(v_0) + \bar{v}_2 \cdot v_{-1} \quad = 1.716$
$v_3 = sin(v_0) \quad = sin(5)$	$\bar{v}_2 = \bar{v}_4 \frac{\partial v_4}{\partial v_2} \quad = \bar{v}_4 \cdot 1 \quad = 1$
$v_4 = v_1 + v_2 \quad = 0.693 + 10$	$\bar{v}_1 = \bar{v}_4 \frac{\partial v_4}{\partial v_1} \quad = \bar{v}_4 \cdot 1 \quad = 1$
$v_5 = v_4 - v_3 \quad = 10.693 + 0.959$	$\bar{v}_3 = \bar{v}_5 \frac{\partial v_5}{\partial v_3} \quad = \bar{v}_5 \cdot (-1) \quad = -1$
	$\bar{v}_4 = \bar{v}_5 \frac{\partial v_5}{\partial v_4} \quad = \bar{v}_5 \cdot 1 \quad = 1$
$f = v_5 \quad = 11.652$	$\bar{v}_5 = \bar{f} \quad = 1$

The final gradient values appear as $\bar{x}_1$, $\bar{x}_2$. Forward mode auto differentiation is efficient and straightforward for functions $f : R \longrightarrow R^m$, as all the derivatives $\frac{\partial f_i}{\partial x}$ can be computed with just one forward pass. Conversely, in the other the extreme of $f : R^n \longrightarrow R$, forward mode Auto differentiation requires n evaluations to compute the gradient, but reverse mode computes in only one forward and one reverse pass.

Example of Auto Differentiation with squared error Loss function:



Let $\hat{y}$ be the actual output and y be the output predicted by the neural network. Let the loss function be given by $L = (y - \hat{y})^2$. If $w_{11}^{[2]}$ is updated as $w_{11}^{[2]} = w_{11}^{[2]} - \alpha \frac{\partial L}{\partial w_{11}^{[2]}}$, we can find $\frac{\partial L}{\partial w_{11}^{[2]}}$ by Auto differentiation. If $(a_1^{[1]}, a_2^{[1]}, a_3^{[1]})$ is the output of the first layer, then the predicted output using the *tanh* activation function is given as

$$y = tanh(w_{11}^{[2]} \cdot a_1^{[1]} + w_{12}^{[2]} \cdot a_2^{[1]} + w_{13}^{[2]} \cdot a_3^{[1]})$$

$$L = (tanh(w_{11}^{[2]} \cdot a_1^{[1]} + w_{12}^{[2]} \cdot a_2^{[1]} + w_{13}^{[2]} \cdot a_3^{[1]}) - \hat{y})^2$$

For computing the derivative of L with respect to $w_{11}^{[2]}$ we start by associating with each intermediate variable v_i a derivative

$$\dot{v}_i = \frac{\partial v_i}{\partial w_{11}^{[2]}}$$

Forward Primal Trace	Forward Tangent Trace
$v_{-2} = w_{11}^{[2]}$	$\dot{v}_{-2} = 1$
$v_{-1} = w_{12}^{[2]}$	$\dot{v}_{-1} = 0$
$v_0 = w_{13}^{[2]}$	$\dot{v}_0 = 0$
$v_1 = a_1^{[1]} \cdot v_{-2}$	$\dot{v}_1 = a_1^{[1]}$
$v_2 = a_2^{[1]} \cdot v_{-1}$	$\dot{v}_2 = 0$
$v_3 = a_3^{[1]} \cdot v_0$	$\dot{v}_3 = 0$
$v_4 = tanh(v_1 + v_2 + v_3)$	$\dot{v}_4 = sech^2(v_1 + v_2 + v_3)(\dot{v}_1 + \dot{v}_2 + \dot{v}_3) = sech^2(v_1 + v_2 + v_3) \cdot a_1^{[1]}$
$v_5 = v_4 - \hat{y}$	$\dot{v}_5 = \dot{v}_4$
$L = v_6 = (v_5)^2$	$\frac{\partial L}{\partial w_{11}^{[2]}} = \frac{\partial v_6}{\partial v_{-2}} = \dot{v}_6 = 2v_5 \cdot \dot{v}_5$

Chapter 6

Navier-Stokes using PINN

The Navier-Stokes equation for an incompressible flow

$$\begin{aligned} \frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v} \cdot \nabla)\mathbf{v} &= -\frac{1}{\rho}\nabla p + \frac{\mu}{\rho}\nabla^2 \mathbf{v} \\ \nabla \cdot \mathbf{v} &= 0 \end{aligned} \tag{6.1}$$

can be written in the formulation

$$\begin{aligned} \frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v} \cdot \nabla)\mathbf{v} &= -\frac{1}{\rho}\nabla \cdot \boldsymbol{\sigma} \\ \boldsymbol{\sigma} &= -pI + \mu(\nabla \mathbf{v} + \nabla \mathbf{v}^T) \end{aligned} \tag{6.2}$$

where $\boldsymbol{\sigma}$ is the Cauchy stress tensor. Taking trace both sides in the second equation of 6.2 and using the divergence-free condition for velocity, we get $p = -tr\boldsymbol{\sigma}/2$.

Stream function: Given a point P and a point A in two dimensions, $\psi(x, y, t)$ can be defined as the integral of the product of the flow velocity (u, v) and the

normal $(dy, -dx)$.

$$\psi = \int_A^P (udy - vdx)$$

The stream function ψ is the volume flux through curve AP. An infinitesimal shift in $\delta P = (\delta x, \delta y)$ of the position P results in a change of the stream function

$$\delta\psi = u\delta y - v\delta x$$

From the exact differential

$$\delta\psi = \frac{\partial\psi}{\partial x}\delta x + \frac{\partial\psi}{\partial y}\delta y$$

the flow velocity components in relation to the stream function ψ have to be

$$u = \frac{\partial\psi}{\partial y} \quad v = -\frac{\partial\psi}{\partial x} \tag{6.3}$$

It can be proved that (u, v) defined this way satisfies the divergence-free condition.

$$\nabla \cdot \boldsymbol{\sigma} = \begin{pmatrix} \frac{\partial}{\partial x}(-p + 2\mu\frac{\partial u}{\partial x}) & \frac{\partial}{\partial y}(\mu\frac{\partial u}{\partial y} + \mu\frac{\partial v}{\partial x}) \\ \frac{\partial}{\partial x}(\mu\frac{\partial u}{\partial y} + \mu\frac{\partial v}{\partial x}) & \frac{\partial}{\partial y}(-p + 2\mu\frac{\partial v}{\partial y}) \end{pmatrix} \tag{6.4}$$

Hence the momentum equation is given by

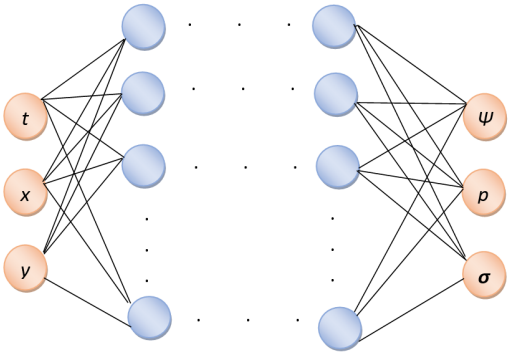
$$\begin{pmatrix} f_u \\ f_v \end{pmatrix} = \begin{pmatrix} \rho\frac{\partial u}{\partial t} + \rho(u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y}) - \frac{\partial}{\partial x}(-p + 2\mu\frac{\partial u}{\partial x}) - \frac{\partial}{\partial y}(\mu\frac{\partial u}{\partial y} + \mu\frac{\partial v}{\partial x}) \\ \rho\frac{\partial v}{\partial t} + \rho(u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y}) - \frac{\partial}{\partial y}(-p + 2\mu\frac{\partial v}{\partial y}) - \frac{\partial}{\partial x}(\mu\frac{\partial u}{\partial y} + \mu\frac{\partial v}{\partial x}) \end{pmatrix} \tag{6.5}$$

and the pressure p is given as

$$p = -tr(\boldsymbol{\sigma})/2 = -\frac{1}{2}((-p + 2\mu\frac{\partial u}{\partial x}) + (-p + 2\mu\frac{\partial v}{\partial y})) \tag{6.6}$$

6.1 Navier Stokes using PINN

For the Navier stokes equation we train a Deep Neural Network(DNN) that maps the spatio-temporal variable $\{t, \mathbf{x}\}^T$ to the mix-variable solution $\{\psi, p_{pred}, \boldsymbol{\sigma}\}$, p_{pred} is the predicted pressure.



The stream function ψ is employed rather than velocity to ensure the divergence-free condition of the flow. The velocity (u, v) can be obtained by auto differentiation of the stream function ψ using 7.3

$$u_{pred} = \frac{\partial \psi}{\partial y} \quad v_{pred} = -\frac{\partial \psi}{\partial x}$$

So (u_{pred}, v_{pred}) obtained this way satisfies the continuity equation.
The output $\boldsymbol{\sigma}$ has three components $\boldsymbol{\sigma} = \{s_{11}, s_{22}, s_{12}\}^T$.

If p_{pred} is the predicted pressure,

$$\begin{pmatrix} f_{upred} \\ f_{vpred} \end{pmatrix} := \begin{pmatrix} \rho \frac{\partial u_{pred}}{\partial t} + \rho(u_{pred} \frac{\partial u_{pred}}{\partial x} + v_{pred} \frac{\partial u_{pred}}{\partial y}) - \frac{\partial s_{11}}{\partial x} - \frac{\partial s_{12}}{\partial y} \\ \rho \frac{\partial v_{pred}}{\partial t} + \rho(u_{pred} \frac{\partial v_{pred}}{\partial x} + v_{pred} \frac{\partial v_{pred}}{\partial y}) - \frac{\partial s_{22}}{\partial y} - \frac{\partial s_{12}}{\partial x} \end{pmatrix} \tag{6.7}$$

On comparing 6.5 and 6.7 we see that u_{pred}, v_{pred} at $\{x, y, t\}$ can satisfy the momentum equation if s_{11}, s_{22} and s_{12}, each of them approximate $(-p_{pred} + 2\mu\frac{\partial u_{pred}}{\partial x})$, $(-p_{pred} + 2\mu\frac{\partial v_{pred}}{\partial y})$ and $(\mu\frac{\partial u_{pred}}{\partial y} + \mu\frac{\partial v_{pred}}{\partial x})$ respectively.

$$\begin{pmatrix} f_{s_{11}} \\ f_{s_{12}} \\ f_{s_{22}} \end{pmatrix} := \begin{pmatrix} (-p_{pred} + 2\mu\frac{\partial u_{pred}}{\partial x}) - s_{11} \\ (\mu\frac{\partial u_{pred}}{\partial y} + \mu\frac{\partial v_{pred}}{\partial x}) - s_{12} \\ (-p_{pred} + 2\mu\frac{\partial v_{pred}}{\partial y}) - s_{22} \end{pmatrix} \tag{6.8}$$

p_{pred} must satisfy 6.6, that is, $p_{pred} = -\frac{1}{2}(s_{11} + s_{22})$.

$$f_{ppred} := p_{pred} + \frac{1}{2}(s_{11} + s_{22}) \tag{6.9}$$

Let r_g denote the residual at $\{x, y, t\}$ obtained from the governing equations and contributes to the physical loss at all the domain points. Then

$$||r_g(x, y, t)||^2 = |f_{upred}|^2 + |f_{vpred}|^2 + |f_{ppred}|^2 + |f_{s_{11}}|^2 + |f_{s_{12}}|^2 + |f_{s_{22}}|^2$$

Let r_i, r_b denote the residuals arising from the initial and boundary conditions and contribute to the physical loss at all the domain points. If $\{x, y, 0\}$ are the domain points at time $t = 0$ then

$$||r_I(x, y, 0)||^2 = ||u(x, y, 0) - u_{pred}(x, y, 0)||^2 + ||v(x, y, 0) - v_{pred}(x, y, 0)||^2 + ||p(x, y, 0) - p_{pred}(x, y, 0)||^2$$

and for any point (x, y) belonging to the boundary at time t

$$||r_b(x, y, t)||^2 := ||u(x, y, t) - u_{pred}(x, y, t)||^2 + ||v(x, y, t) - v_{pred}(x, y, t)||^2 + ||p(x, y, t) - p_{pred}(x, y, t)||^2$$

Loss: The loss function is composed of the physical loss J_g obtained from the governing equations and the initial boundary condition loss $J_{i/bc}$ given by

$$\begin{aligned} J_g &= \frac{1}{N_g} \sum_{i=1}^{N_g} ||r_g(x^i, y^i, t^i)||^2 \\ J_{i/bc} &= \frac{1}{N_I} \sum_{i=1}^{N_I} ||r_I(x^i, y^i, 0)||^2 + \frac{1}{N_b} \sum_{i=1}^{N_b} ||r_b(x^i, y^i, t^i)||^2 \end{aligned} \tag{6.10}$$

where $N_{(\cdot)}$ denotes the number of collocation points.
The total physical loss J_p is defined as

$$J_p = J_g + \beta J_{i/bc} \tag{6.11}$$

where $\beta > 0$ is a user-defined weighting coefficient for the initial and boundary condition loss.
At every epoch, the weights are updated by using various optimization algorithms like Adam, BFGS, or L-BFGS.

6.2 Test Examples

6.2.1 Example 1

Numerical Simulation of flow in a rectangle using DUNE [5] Framework and PINNs is presented. The computational domain is a rectangle in two dimensions of length 1.1 cm and a breadth 0.41 cm. The viscosity $\nu = 0.01 gs^{-1}cm^{-1}$. The time duration

for simulation/modeling is $T = 0.5$ seconds with the time step $\Delta t = 0.01$ seconds.

PINN:

We implement the proposed PINN in TensorFlow [9]. The Neural network has 3 neurons in the input layer, 5 neurons in the output layer, and seven hidden layers with 50 neurons each. The activation function used for each hidden layer is the *tanh* activation function. A total number of 3321 collocation points which include $N_b = 244$ points on the boundary, $N_{in/out} = 81$ on the inlet/outlet boundary, are generated using Latin hypercube sampling (LHS) and train the network. The coefficient β is set to 2. Adam and Limited-memory BFGS (L-BFGS) optimizer is employed to train the Neural Network due to their good convergence speed. The network is first trained with the Adam optimizer for 5000 iterations and then trained with L-BFGS optimizer for 85480 iterations, and the maximum number of iterations to find optimal step length α_k using the Hager-Zhang line search algorithm is 50. The total amount of time taken to train the network is 159045.8855 seconds. A total number of 64561 collocation points which include $N_b = 1124$ points in the boundary $N_{in/out} = 161$ on the inlet/outlet boundary, is used for prediction. The training and prediction is carried out by $\beta = 5$, $\beta = 10$ and the loss is plotted using python.

DUNE:

The geometry is created using GMSH [6] and an Unstructured grid is used for meshing. The velocity is of degree two, and pressure is taken to be degree one. The number of elements in the mesh is 32724. The number of edges in the mesh is 97445. Total number of vertices in the mesh is 64722. The number of constrained degrees of freedom is 2254. Crank Nicholson The scheme is used for discretization. The nonlinear solver used is Newton. Bi-Conjugate Gradient Stabilized with ILU Pre-

conditioner is used as the linear solver. The CPU time taken for the simulation is 3374.51 seconds.

Boundary Conditions

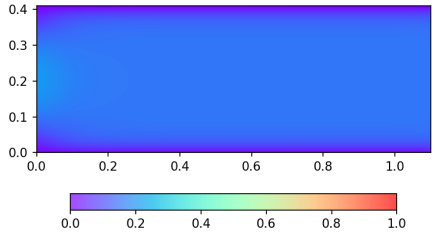
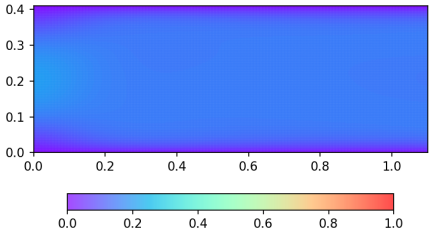
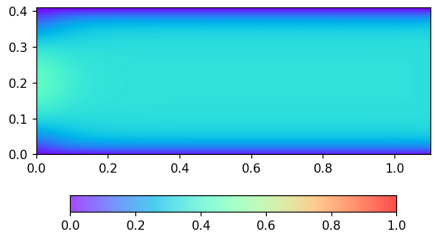
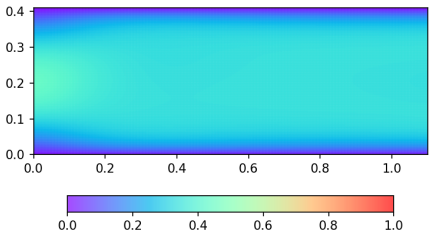
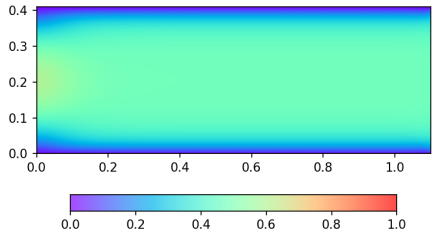
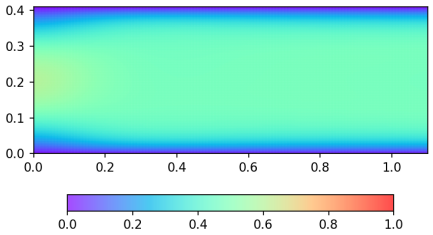
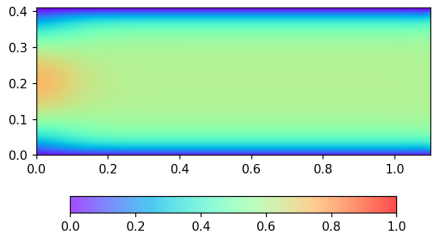
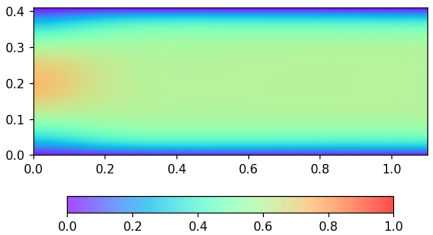
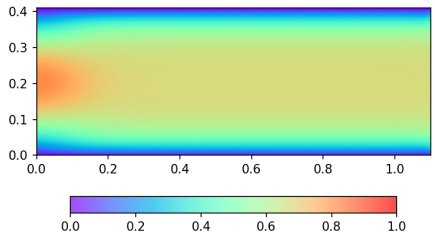
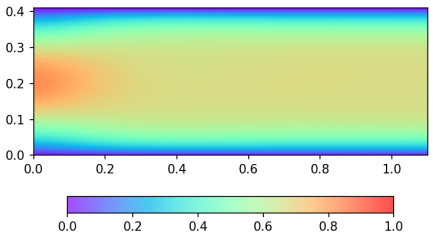
$$\begin{aligned} \mathbf{u} &= 0 && \text{on } \Gamma_D \\ \mathbf{u} &= \begin{pmatrix} 2 \cdot x_2 \cdot \frac{0.41 - x_2}{0.41^2} \cdot (sin(\frac{\pi t}{T} + \frac{3\pi}{2}) + 1) \\ 0 \end{pmatrix} && \text{on } \Gamma_{in} \\ p &= 0 && \text{on } \Gamma_{out} \end{aligned}$$

Initial conditions $(\mathbf{u}^0, p^0) = (\mathbf{0}, 0)$

Observations:

- The results obtained through PINNs and DUNE agree qualitatively.
- The loss produced is 4.024×10^{-5} for $\beta = 2$. The rate of convergence of the L-BFGS optimizer is faster for $\beta = 5$.

Table 6.1: Velocity

DUNE	PINN
Time: 0.15s 	Time: 0.15s 
Time: 0.23s 	Time: 0.23s 
Time: 0.28s 	Time: 0.28s 
Time: 0.33s 	Time: 0.33s 
Time: 0.36s 	Time: 0.36s 

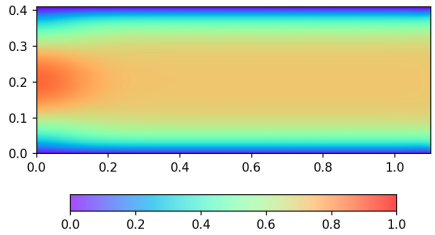
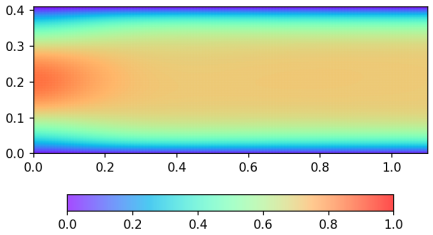
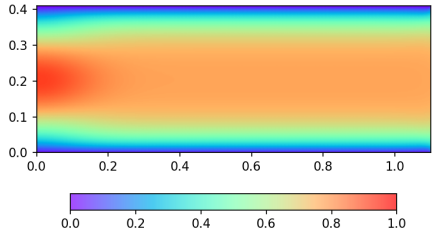
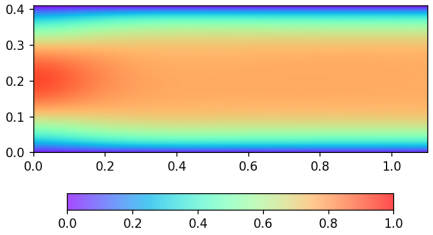
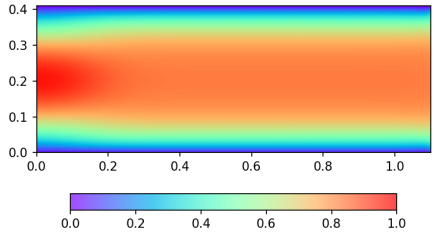
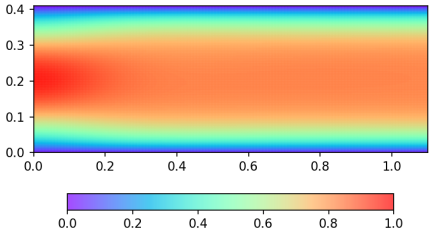
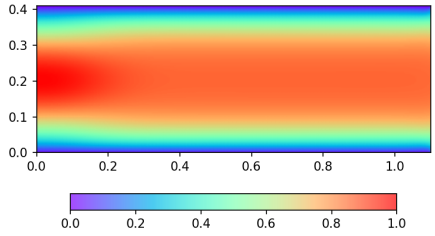
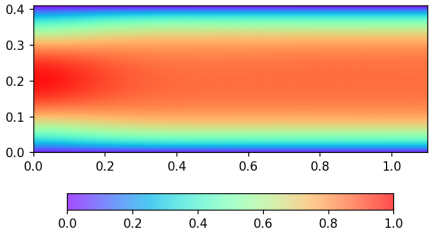
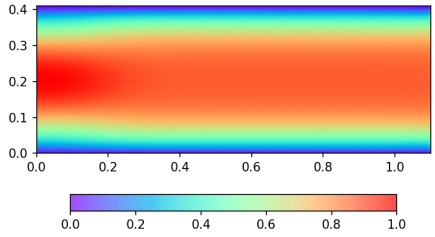
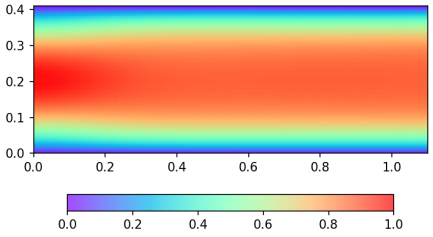
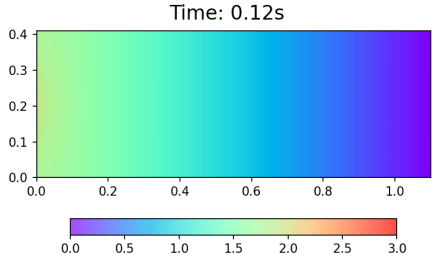
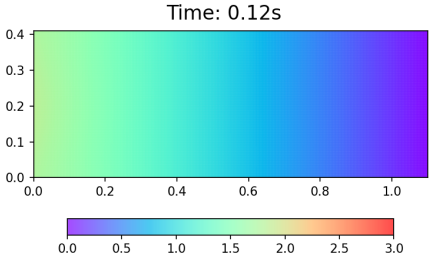
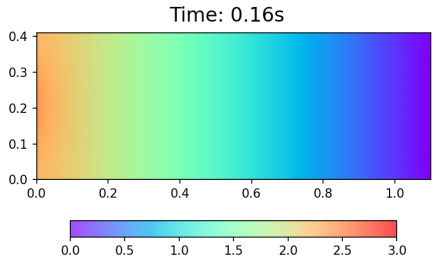
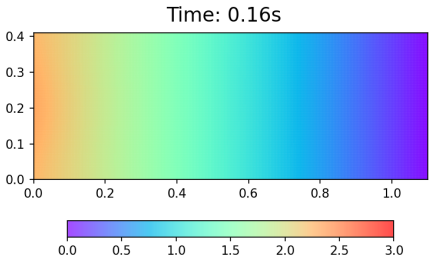
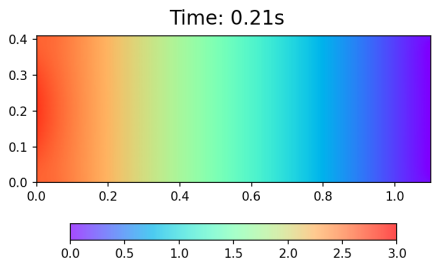
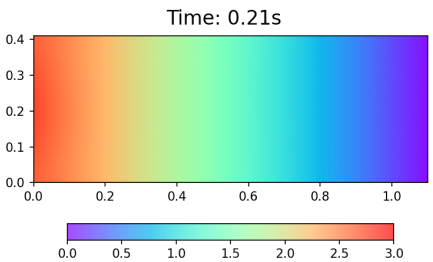
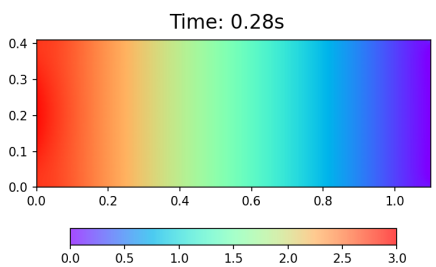
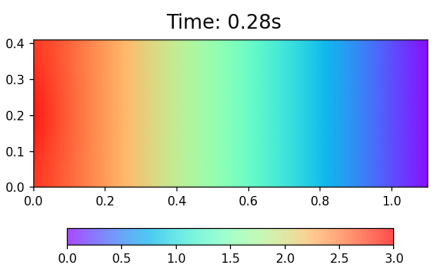
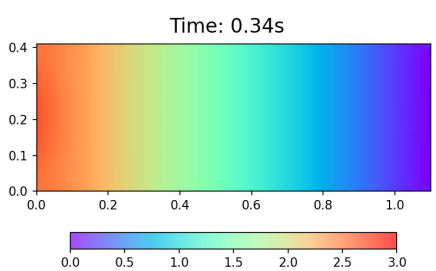
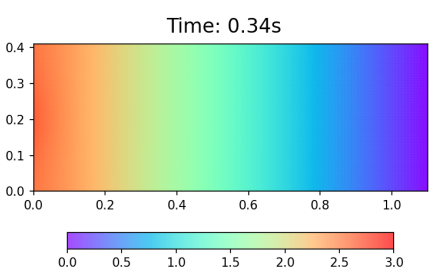
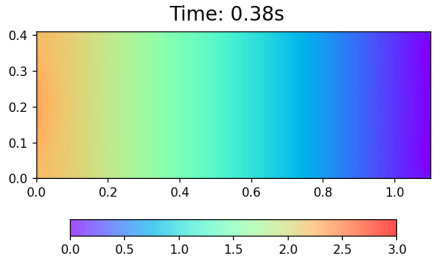
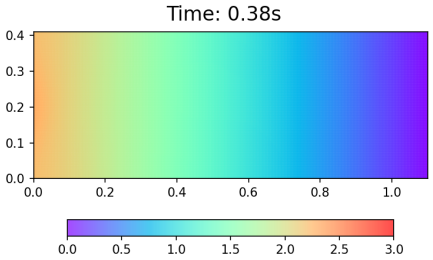
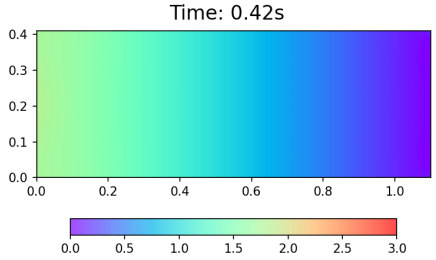
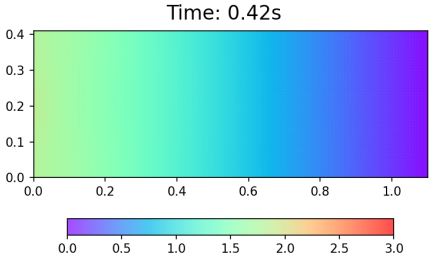
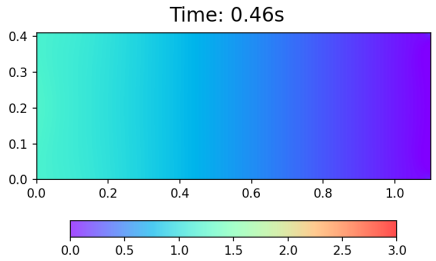
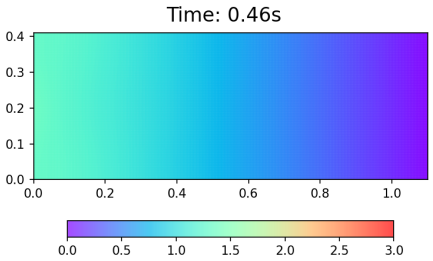
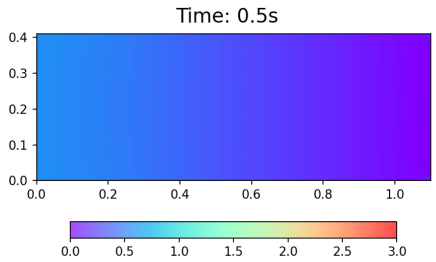
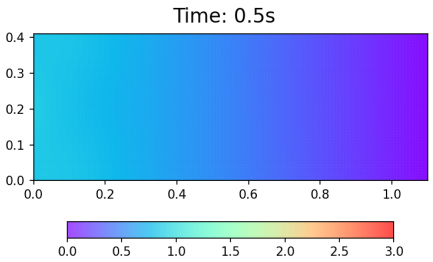
DUNE	PINN
Time: 0.38s	Time: 0.38s
	
Time: 0.41000000000000003s	Time: 0.41000000000000003s
	
Time: 0.45s	Time: 0.45s
	
Time: 0.48s	Time: 0.48s
	
Time: 0.5s	Time: 0.5s
	

Table 6.2: Pressure

DUNE	PINN
	
	
	
	
	

DUNE	PINN
 Time: 0.38s	 Time: 0.38s
 Time: 0.42s	 Time: 0.42s
 Time: 0.46s	 Time: 0.46s
 Time: 0.5s	 Time: 0.5s

Loss

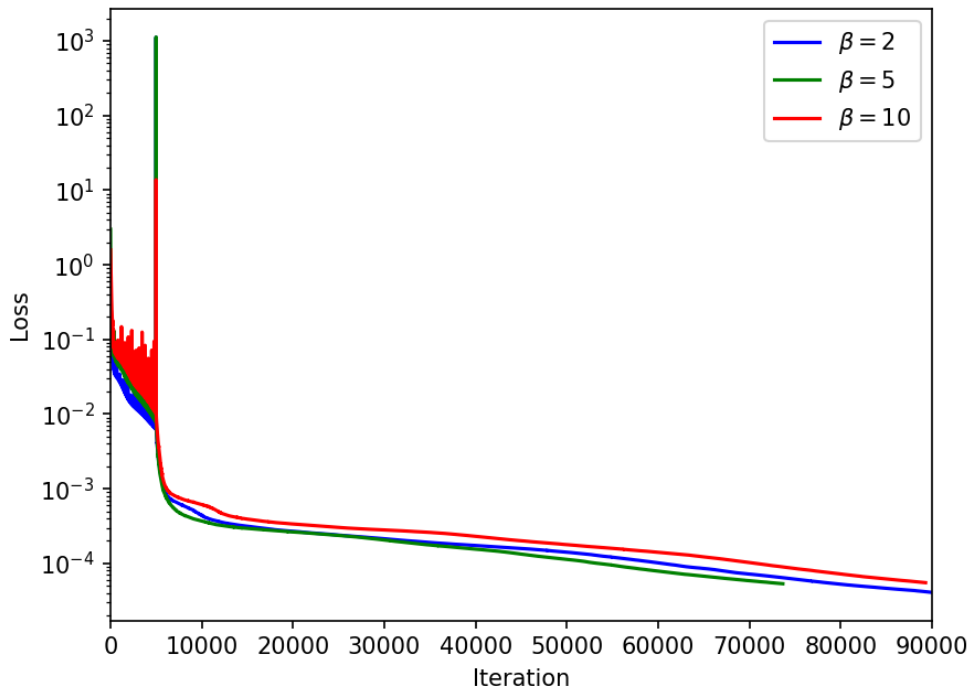


Figure 6.1: Comparison of convergence curves with respect to the coefficient β. Network of 7 x 50 used in all cases. 5000 iterations trained with Adam optimizer followed by 85480 iterations with L-BGFS optimizer

6.2.2 Example 2

Numerical Simulation of flow in a semi-circular domain using DUNE [5] Framework and PINNs is presented. The computational domain is a semicircular pipe with smooth disturbances in two dimensions with cross-sectional radius $a = 1.6cm$ and curvature radius $R = 2.9cm$. The kinematic viscosity $\nu = 0.4\ gs^{-1}cm^{-1}$. The time duration for simulation/modeling is $T = 6$ seconds with the time step $\Delta t = 0.01$ seconds.

PINN:

We implement the proposed PINN in TensorFlow [9]. The Neural network has 3 neurons in the input layer, 5 neurons in the output layer, and seven hidden layers with 50 neurons each. The activation function used for each hidden layer is the $tanh$ activation function. The total number of collocation points is 29760, and a batch of 20000 points is used in every iteration to train the network. The coefficient β is set to 2. Adam and Limited-memory BFGS (L-BFGS) optimizer is employed to train the Neural Network due to their good convergence speed. The network is first trained with the Adam optimizer for 800 iterations and then trained with L-BFGS optimizer for 11344 iterations. The maximum number of iterations to find optimal step length α_k using the Hager-Zhang line search algorithm is 50. The total time taken to train the network is 94102.45567 seconds. A total number of 29760 collocation points is used for prediction.

DUNE:

The geometry is created using GMSH [6] and an Unstructured grid is used for meshing. The velocity is of degree two, and pressure is considered degree one. The number

of elements in the mesh is 1984. The number of edges in the mesh is 3064. Total number of vertices in the mesh is 1081. The number of constrained degrees of freedom is 659. Crank Nicholson The scheme is used for discretization. The non-linear solver used is Newton. Bi-Conjugate Gradient Stabilized with ILU Preconditioner is used as the linear solver. The CPU time taken for the simulation is 131.479 seconds.

Boundary Conditions

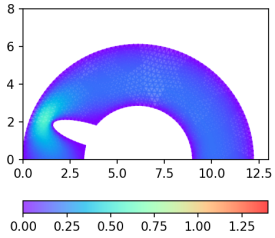
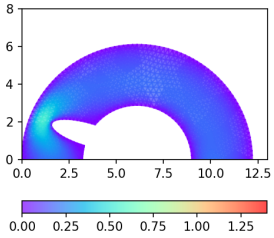
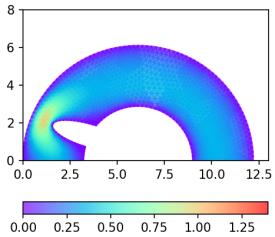
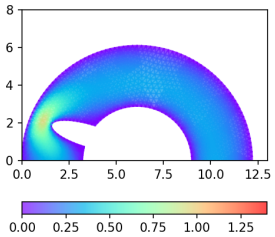
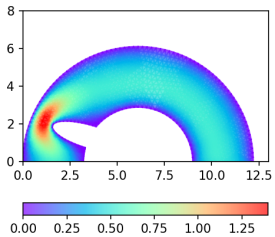
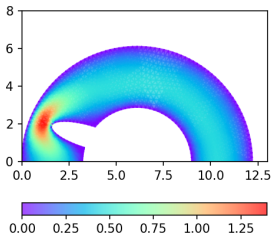
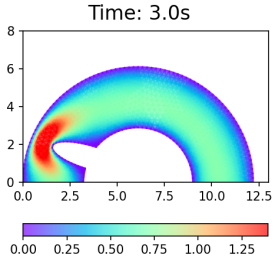
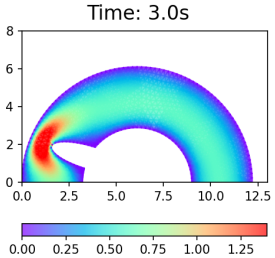
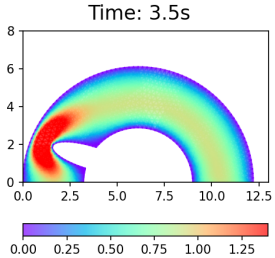
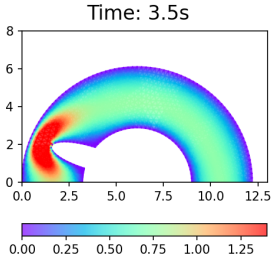
$$
\begin{aligned}
&\mathbf{u} = 0 && \text{on } \Gamma_D \\
&\mathbf{u} = \begin{pmatrix} 0 \\ 3 \cdot x_1 \cdot \frac{3.2 - x_1}{3.2^2} \cdot (sin(\frac{\pi t}{T} + \frac{3\pi}{2}) + 1) \end{pmatrix} && \text{on } \Gamma_{in} \\
&p = 0 && \text{on } \Gamma_{out}
\end{aligned}
$$

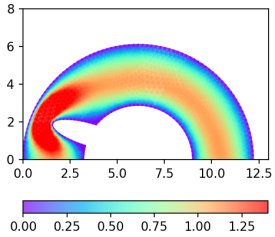
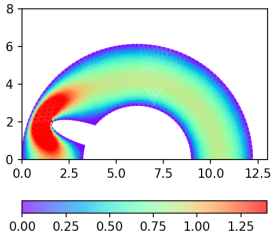
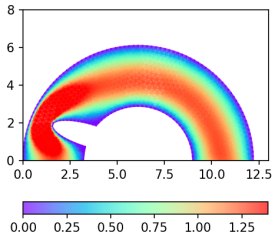
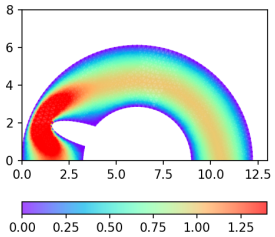
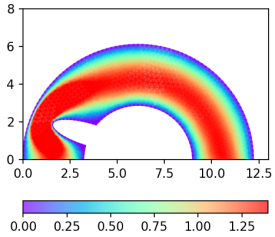
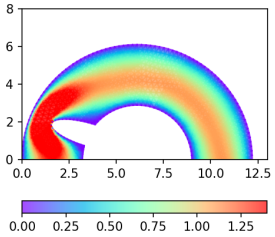
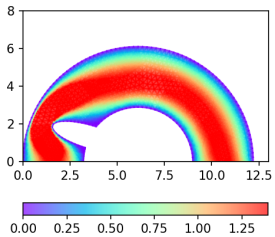
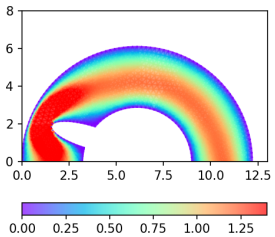
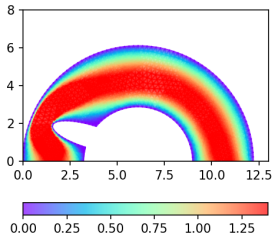
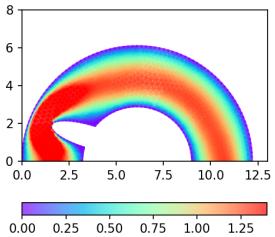
Initial conditions $(\mathbf{u}^0, p^0) = (\mathbf{0}, 0)$

Observations:

- The results obtained through DUNE and PINNs agree qualitatively.
- The loss produced is 0.0003991 for $\beta = 2$.

Table 6.3: Velocity

DUNE	PINN
Time: 1.5s 	Time: 1.5s 
Time: 2.0s 	Time: 2.0s 
Time: 2.5s 	Time: 2.5s 
Time: 3.0s 	Time: 3.0s 
Time: 3.5s 	Time: 3.5s 

DUNE	PINN
Time: 4.0s	Time: 4.0s
	
Time: 4.5s	Time: 4.5s
	
Time: 5.0s	Time: 5.0s
	
Time: 5.5s	Time: 5.5s
	
Time: 6.0s	Time: 6.0s
	

Loss:

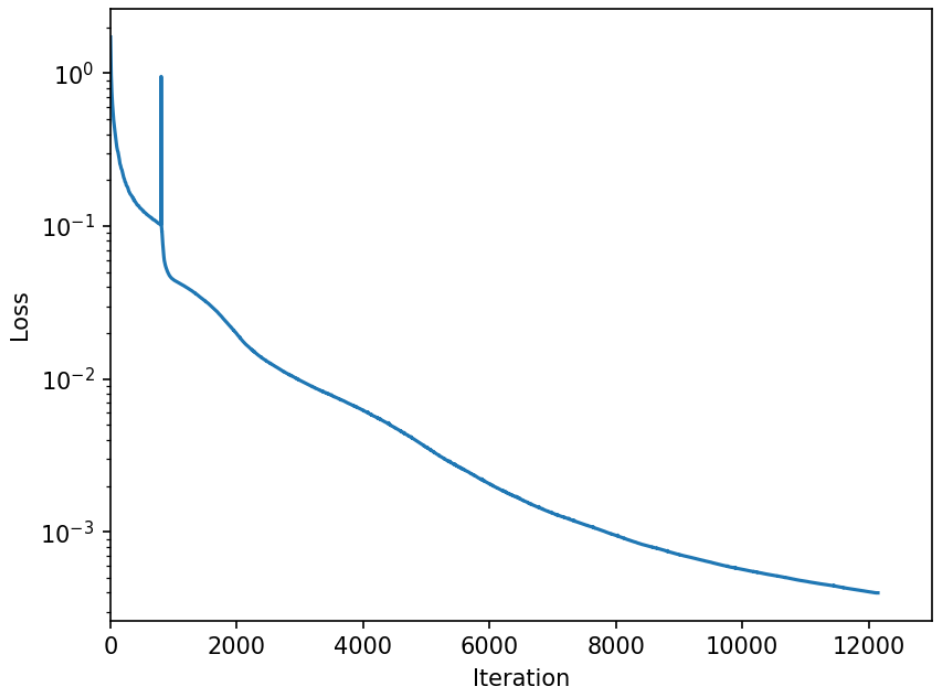


Figure 6.2: Loss curve for example 2. A network of 7 x 50 is used. 800 iterations trained with Adam optimizer followed by 11344 iterations trained with L-BGFS optimizer.

Chapter 7

Conclusion

In this project, we have modeled the blood flow in an open, bounded Lipschitz domain through the Navier-Stokes Equations and studied the existence and uniqueness of a solution in two dimensions through FEM.

In the first part, we have derived a finite element formulation of the Navier-Stokes Equation in appropriate finite-dimensional velocity and pressure spaces, which are the discrete counterparts of continuous spaces $\mathbf{H}_0^1(\Omega)$ and $L^2(\Omega)$. We examined stable and unstable pairs of Finite Element Spaces and proved that the finite element pairs P_1/P_1 pair and P_1/P_0 are unstable. The Taylor Hood finite element pair P_{r+1}/P_r for $r \geq 1$ was found to be stable and was used in the finite-dimensional formulation of the Navier-Stokes Equation.

In the following section, we discussed and simulated the blood flow in circular domains using Finite Element Method in two and three dimensions. The effect of Dean number on flows in a circular domain was demonstrated.

In the last part of the project, the mixed-variable PINNs formulation proposed in [4] has been implemented. The solution got through PINNs and using FEM closely agreed qualitatively on straight domains. The CPU time for training PINNs on a straight domain is 159045.8855 seconds with 3321 collocation points which are computationally expensive compared to the solution obtained in 3374.51 seconds using FEM. The loss is minimized using ADAM Optimization Algorithm for 5000 iterations followed by the LBFGS Optimization Algorithm for 85840 iterations to achieve a loss of 4.024×10^{-5}.

The results on the circular domain obtained through PINNs agreed qualitatively with FEM. The training time for PINNs is 94102.4556 seconds for a batch size of 20000 at every epoch compared to the solution in 131.479 seconds using FEM. The loss is minimized using the ADAM optimization algorithm for 800 iterations followed by the LBFGS optimization algorithm for 11344 iterations to achieve a loss of 3.991×10^{-4}. Parallel programming with GPU can significantly reduce the training time to provide a promising mesh-free technique for predicting blood flow in various domains and has been left for future work.

Bibliography

[1] Auricchio Lefieux. Computational study of aortic hemodynamics: From simplified to patient-specific geometries. *Advances in Computational Fluid-Structure Interaction and Flow Simulation (pp.397-407)*, 2016.

[2] Kratzke and Mang. Tutorial: Aortic blood flow simulation using hiflow. https://emcl-gitlab.iwr.uni-heidelberg.de/hiflow3.org/hiflow3/-/wikis/uploads/ff8030ea189d71d6b546426a50e6a52a/tut_blood_flow.pdf.

[3] Maziar Raissi, Paris Perdikaris, and George E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[4] Chengping Rao, Hao Sun, and Yang Liu. Physics-informed deep learning for incompressible laminar flows. *Theoretical and Applied Mechanics Letters*, 10:207–212, 2020.

[5] Dune numerics. https://www.dune-project.org/.

[6] A three-dimensional finite element mesh generator with built-in pre- and post-processing facilities. https://gmsh.info/.

[7] Automatic differentiation in machine learning: a survey. https://arxiv.org/pdf/1502.05767.pdf.

[8] Automatic differentiation background. https://in.mathworks.com/help/deeplearning/ug/deep-learning-with-automatic-differentiation-in-matlab.html.

[9] Tensorflow. https://www.tensorflow.org/.

[10] V. John. *Finite Element Methods for Incompressible Flow Problems*. Springer Series in Computational Mathematics. Springer International Publishing, 2016.

[11] H.C. Elman, D.J. Silvester, and A.J. Wathen. *Finite Elements and Fast Iterative Solvers:with Applications in Incompressible Fluid Dynamics: with Applications in Incompressible Fluid Dynamics*. Numerical Mathematics and Scientific Computation. OUP Oxford, 2005.

[12] V Girault and P A Raviart. *Finite Element Methods for Incompressible Flow Problems*. Second Edition. Springer, 1989.

[13] R. Temam. *Navier-Stokes Equations: Theory and Numerical Analysis*. Studies in mathematics and its applications. North-Holland, 1984.

[14] L. Formaggia, A. Quarteroni, and A. Veneziani. *Cardiovascular Mathematics: Modeling and simulation of the circulatory system*. MS&A. Springer Milan, 2010.

[15] J. Nocedal and S. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer New York, 2006.

[16] H. Jiang. *Machine Learning Fundamentals*. Cambridge University Press, 2021.

[17] Justine Fouchet-Incaux. Artificial boundaries and formulations for the incompressible navier–stokes equations: applications to air and blood flows. *SeMA*, 2014.

[18] Yogesh Karnam. Multiscale fluid-structure interaction models development and applications to the 3d elements of a human cardiovascular system. *RIT Scholar Works, Rochester Institute of Technology*, 2019.

[19] Laminar flow. https://www.simscale.com/docs/simwiki/cfd-computational-fluid-dynamics/what-is-laminar-flow/.

[20] Physics informed neural networks. https://github.com/maziarraissi/PINNs.

[21] Pinn-laminar-flow. https://github.com/Raocp/PINN-laminar-flow.

[22] BFGS algorithm. https://en.wikipedia.org/wiki/Broyden-Fletcher-Goldfarb-Shanno_algorithm, https://en.wikipedia.org/wiki/Quasi-Newton_method.

[23] Machine learning lecture notes – mit 6.390 fall 2022. https://introml.mit.edu/_static/fall22/LectureNotes/6_390_lecture_notes_fall2022.pdf.