LEARNING PDE SOLUTION OPERATORS VIA DIFFEOMORPHIC MAPPINGS: APPLICATIONS IN FLUID DYNAMICS

by

Shiyi Chen

A dissertation submitted to Johns Hopkins University in conformity with the requirements for the degree of Master of Science in Engineering

Baltimore, Maryland

December 2024

Abstract

Approximating the solution operator for partial differential equations (PDEs) is critical across numerous scientific and engineering disciplines. Neural operators have emerged as a powerful tool for predicting these solution operators when trained on high-quality ground truth data, such as results from numerical simulations. However, their ability to generalize to unseen spatial domains is often hindered by the need for extensive datasets encompassing diverse geometric configurations, which may be infeasible to generate in many scenarios. To address this challenge, we propose training a latent neural operator using solution fields that are diffeomorphically mapped from varying spatial domains to a unified reference configuration. The efficacy of this approach hinges on preserving the differential operator's properties during the mapping process, which enhances the regularity of the solution fields and reduces the data requirements for training accurate models. Through two numerical experiments, we validate our framework: (1) leveraging the conformal invariance of the Laplacian in 2D doubly-connected domains, we demonstrate that conformal mappings significantly improve learning efficiency compared to other standardization methods; and (2) extending to 3D nonlinear flows around spheroids of varying aspect ratios, we accurately predict surface shear stress distributions using reduced training data. These results underscore the potential of this framework for applications in computational fluid dynamics, including scenarios such as biomedical flows, where data collection is constrained by practical limitations.

Primary Reader and Advisor: Natalia A. Trayanova

Secondary Reader: Rajat Mittal

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my parents, Xiaoyu and Xianlin, for their unconditional and long-term support of my education and their respect for my decisions. Their unwavering support made my journey at Johns Hopkins possible.

I am profoundly grateful to everyone at the Computational Cardiology Lab, especially Natalia, who provided me with the opportunity to develop machine learning frameworks for PDE prediction. Special thanks to Zan for introducing me to the lab, teaching me about operator learning, and showing me the fascinating world of operators learning on shapes. I am also indebted to Minglang, Mauro, and Yashil for their invaluable discussions and contributions that were instrumental to this project's success.

The Mechanical Engineering Department has been a source of constant support and intellectual stimulation. The Friday morning bagel discussions were a highlight of my time here, and I am particularly thankful to Yifan, Tianrui, Martin, Hanna, Lexie, Sahaj, and Yue for their mental and emotional support during some of my most challenging times at Hopkins.

I would be remiss not to acknowledge Charlie, whose influence extends far beyond his teachings in Math, Physics, and Biology. His encouragement to follow my passion and his continued support from my undergraduate years to the present have been invaluable to my academic journey.

This work would not have been possible without the collective support of all these individuals who have contributed to my growth both as a researcher and as a person.

To my father, Xianlin and my mother Xiaoyu

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Numerical simulations of physical systems governed by partial differential equations (PDEs) are a crucial aspect of many problems in science and engineering. However, traditional numerical methods often demand significant computational resources, particularly for complex systems or high-dimensional parameter spaces. Recent advancements in scientific machine learning include *neural operators*, which can efficiently predict PDE solutions after training on high-fidelity simulation data, as a promising alternative to traditional approaches [12]. A major challenge in applying neural operators to real-world data lies in the variability of geometric and parametric domains on which these PDEs are defined. Learning solution fields across these varying domains is difficult for typical neural operator frameworks, such as Deep Operator Networks (DeepONet) [13] and Fourier Neural Operators (FNO) [17] which either require a fixed grid or a regular domain on which the Fourier transform is available, respectively. To address this, some variants of these methods transform numerical solutions from their original domain to a latent space where training can be unified. This process often involves mapping solutions through integral transforms [16, 3, 15] or spatial transformations [14, 24, 22]. Spatial transformations, in particular, standardize the domain by quotienting the geometric variability during training. While effective, these approaches require plentiful expressive/geometrically varying training samples to obtain generalizable model. This raises questions about how the choice of transformation influences learning outcomes. The literature has yet to fully address specifically how the nature of the *spatial* transformation of the solution domains and the representation of solutions in the transformed (reference/latent) space impact the learning process.

In this work, we focus on the problem of learning a neural operator on latent solution

representations of the PDE after applying an invertible spatial transformation to a reference configuration. Using the 2D Laplace equation on doubly-connected domains as our first test case, we analyze how different types of mappings affect the performance of neural operators and effectively demonstrate that using mappings which preserve mathematical properties of the PDE in the latent space facilitate more accurate and data-efficient learning. Additionally, we extend our investigation to a more complex, three-dimensional case of flow past spheroids at moderate Reynolds numbers. This nonlinear problem serves to demonstrate the broader applicability of our framework beyond simple linear PDEs and lower-dimensional domains. By accurately predicting surface shear stress distributions on spheroids of varying aspect ratios, we show that our method can effectively handle the complex fluid dynamics characteristic of bluff body flows. Through numerical experimentation in both cases, we compare different mappings of the solutions to canonical domains with varying geometry to numerically elucidate the relationship between the choice of mapping, the transformed representation of the PDE solutions, and the strong influence of these factors on the resultant learning efficiency within this framework. Furthermore, this study aims to provide empirical evidence to motivate the design of mappings (or algorithms that can approximate such mappings) that minimize computational and data requirements for high-fidelity numerical simulations and neural operator training.

Looking ahead, we envision applying this framework to predict hemodynamic indices associated with thrombus formation risk in the left atrial appendage (LAA), a critical factor in thromboembolic stroke assessment[6]. Our successful prediction of surface stress fields in the spheroid case provides encouraging evidence for the framework's potential application to such complex biological flows. The ability to accurately capture surface stresses and other flow features in the controlled environment of rigid body flow suggests that our approach could be adapted to handle the highly nonlinear phenomena characteristic of cardiovascular flows, potentially enabling more efficient risk assessment tools for clinical applications.

Chapter 2

Problem setups

2.1 Operator Learning Frameworks for Geometries

Neural operators are a family of neural network frameworks designed to approximate nonlinear, continuous operators, such as the solution operators of partial differential equations (PDEs). Unlike traditional neural networks, which learn mappings between finite-dimensional vector spaces, neural operators learn mappings between infinite-dimensional function spaces, enabling them to model the relationships between input functions (initial and boundary conditons) and PDE solutions directly.

Formally, a neural operator $\mathcal{N}_\theta$ is defined as a mapping from an input function space $\mathcal{V}(\Omega_\alpha; \mathbb{R}^{d_v})$ to an output function space $\mathcal{U}(\Omega_\alpha; \mathbb{R}^{d_u})$, where $\Omega_\alpha \subset \mathbb{R}^d$ is a domain parameterized by $\alpha \in \mathcal{A}$. This mapping can be expressed as:

$$\mathcal{N}_\theta : \mathcal{V}(\Omega_\alpha; \mathbb{R}^{d_v}) \to \mathcal{U}(\Omega_\alpha; \mathbb{R}^{d_u}),$$

where $\theta \in \Theta$ denotes the parameters of the neural operator. The spaces $\mathcal{V}(\Omega_\alpha; \mathbb{R}^{d_v})$ and $\mathcal{U}(\Omega_\alpha; \mathbb{R}^{d_u})$ are subspaces of function spaces such as Sobolev spaces $H^s(\Omega_\alpha)$ or spaces of continuous functions $C(\Omega_\alpha)$. Here:

- $\mathcal{V}(\Omega_\alpha; \mathbb{R}^{d_v})$ represents the space of input functions, such as coefficients, forcing terms, or boundary conditions of the PDE.
- $\mathcal{U}(\Omega_\alpha; \mathbb{R}^{d_u})$ corresponds to the space of PDE solutions.

For an input $(v_1, \ldots, v_m) \in \mathcal{V}(\Omega_\alpha; \mathbb{R}^{d_v})$, the true solution operator $\mathcal{G}$ maps these inputs

to the PDE solution $u \in \mathcal{U}(\Omega_\alpha; \mathbb{R}^{d_u})$, satisfying:

$$u = \mathcal{G}(v_1, \ldots, v_m).$$

The neural operator $\mathcal{N}_\theta$ approximates the solution operator $\mathcal{G}$ by learning from a set of input-output pairs:

$$\{(v_{1,i}, \ldots, v_{m,i}, u_i)\}_{i=1}^N, \quad \text{where } u_i = \mathcal{G}(v_{1,i}, \ldots, v_{m,i}).$$

Once trained, $\mathcal{N}_\theta$ serves as a surrogate for $\mathcal{G}$, enabling efficient and accurate predictions of u for new inputs $(v_1, \ldots, v_m)$ without the computational cost of solving the PDE numerically. By working directly with function spaces, neural operators offer the flexibility and efficiency needed for tasks involving parametric PDE simulations and high-dimensional domains.

2.1.1 Diffeomorphic Mapping Operator Learning

Let $\mathcal{A}$ denote a compact subset of $\mathbb{R}^p$, parametrizing the domain variations. For each $\alpha \in \mathcal{A}$, the embedding φ_α is a C^2 diffeomorphism from the reference domain Ω_0 to the target domain Ω_α, and the mapping $\mathcal{A} \to C^2(\Omega_0, \mathbb{R}^d)$, given by $\alpha \mapsto \varphi_\alpha$, is continuous in the standard C^2-norm.

The solution operator on each domain Ω_α is denoted by

$$\mathcal{G}_\alpha : \mathcal{V}_1^\alpha(\Omega_\alpha) \times \cdots \times \mathcal{V}_m^\alpha(\Omega_\alpha) \to \mathcal{U}^\alpha(\Omega_\alpha)$$

mapping inputs defined on Ω_α to the PDE solution $u^\alpha(\cdot; \mathbf{v}^\alpha)$. Here, $\mathbf{v}^\alpha = (v_1^\alpha, \ldots, v_m^\alpha)$ represents the input functions.

This family of operators $\{\mathcal{G}_\alpha\}_{\alpha \in \mathcal{A}}$ can be reformulated in terms of a latent operator $\mathcal{F}_0$

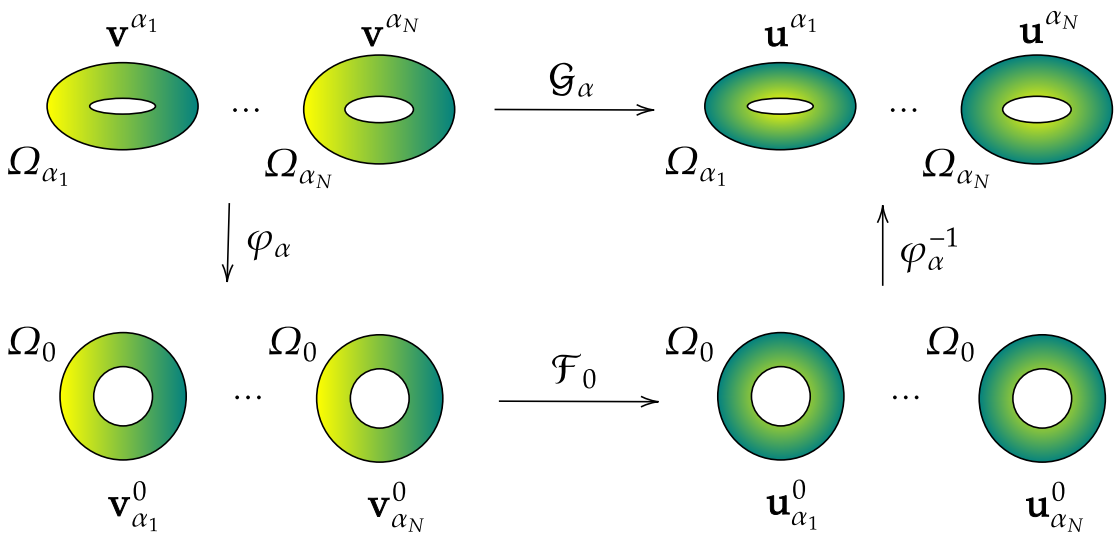

Figure 2.1: Relation of reference and target domains in DIMON

defined on the reference domain Ω_0, as:

$$\begin{aligned} u^\alpha(\mathbf{x}) &= \mathcal{G}_\alpha(v_1^\alpha, \ldots, v_m^\alpha) \\ &= \mathcal{F}_0\left(\alpha, v_1^\alpha \circ \varphi_\alpha, \ldots, v_m^\alpha \circ \varphi_\alpha\right) \circ \varphi_\alpha^{-1}. \end{aligned}$$

Defining the pullback of the inputs as $v^0_{\alpha,k} = v^\alpha_k \circ \varphi_\alpha$ and the latent solution as $u^0_\alpha = \mathcal{F}_0(\alpha, v^0_{\alpha,1}, \ldots, v^0_{\alpha,m})$, the solution on Ω_α can be equivalently written as:

$$u^\alpha(\cdot; \mathbf{v}^\alpha) = u^0_\alpha \circ \varphi_\alpha^{-1}.$$

Here, the operator $\mathcal{F}_0$ captures the relationship between the shape parameter α, the input functions transported to Ω_0, and the corresponding solution on the reference domain. This formulation effectively maps solutions from Ω_α into a fixed domain Ω_0, enabling the neural operator to learn in a geometry-independent space. Of course, the representation of the solution (for a fixed $\alpha \in \mathcal{A}$) in the latent space where a neural operator learns $\mathcal{F}_0$ is dependent on the choice of map and (in most cases) the specific geometries Ω_α and Ω_0. In principle, $\mathcal{F}_0$ may be approximated by any neural operator method for the chosen Ω_0.

In this work, we use a modified Deep Operator Network (DeepONet) architecture similar to MIONet [8] to approximate the solution operator of the form:

$$\mathcal{F}_0 : \mathcal{A} \times \mathcal{V}_1(\Omega_0) \times \cdots \times \mathcal{V}_m(\Omega_0) \to \mathcal{U}(\Omega_0),$$

where $\mathcal{V}_k(\Omega_0)$ and $\mathcal{U}(\Omega_0)$ are function spaces defined on a fixed grid Ω_0, and $\mathcal{A}$ represents the parameter space.

Our architecture consists of two key input encodings:

- A geometry encoder $\mathbf{g}_{\text{geo}}(\alpha) : \mathcal{A} \to \mathbb{R}^q$, which maps the shape parameter α to a low-dimensional representation. This encoding can be computed using principal component analysis (PCA) on the displacement vectors between the target domain Ω_α and the reference domain Ω_0, or by computing a representation of the embedding directly from the flow of the map φ_α. An example of such PCA representation is shown in Figure 2.2 in our Laplace problem.

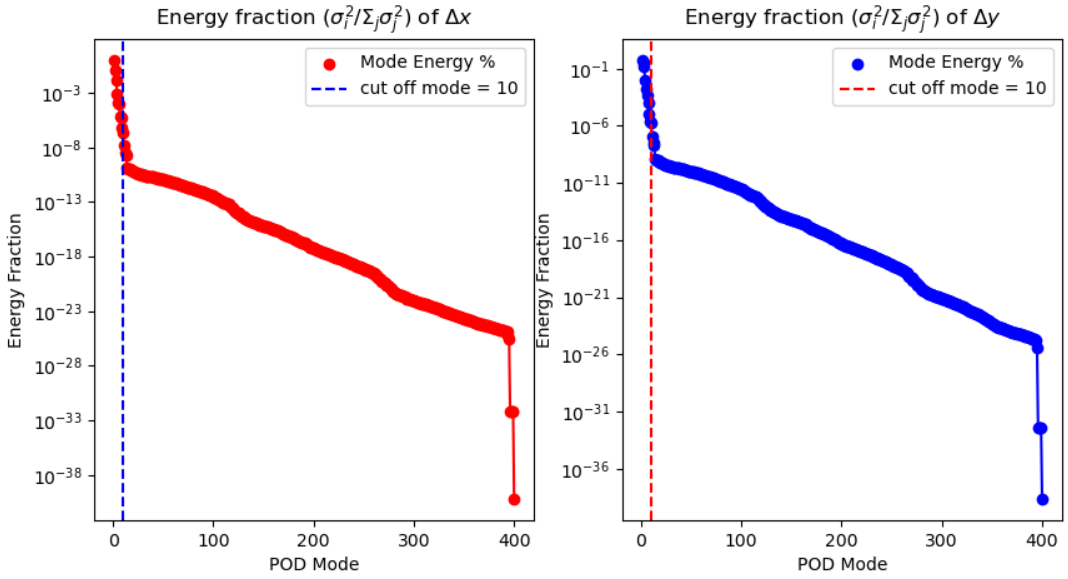

Figure 2.2: POD spectrum, taking first 10 modes for shape parameters in 2D

- A physical condition encoder $\mathbf{g}_{\text{phys}}(\mathbf{v}_\alpha^0) : \mathcal{V}_1(\Omega_0) \times \cdots \times \mathcal{V}_m(\Omega_0) \to \mathbb{R}^r$, which encodes

the initial and boundary conditions transported to Ω_0.

These encodings are concatenated and used as inputs to a neural operator. For spatial dependence, we use a separate network $\mathbf{h}(\mathbf{x}) : \Omega_0 \to \mathbb{R}^s$, which maps spatial coordinates to a feature representation. The final approximation of the solution is given by:

$$u_\alpha^0(\mathbf{x}) \approx \hat{\mathcal{F}}_\theta \left(\sum_{i=1}^{n} \mathbf{w}_i\big(\mathbf{g}_{\text{geo}}(\alpha), \mathbf{g}_{\text{phys}}(\mathbf{v}_\alpha^0)\big) \cdot \mathbf{h}_i(\mathbf{x}) \right),$$

where:

- $\mathbf{g}_{\text{geo}}(\alpha)$ encodes the geometric information for the shape parameter α and θ denotes the network parameters,
- $\mathbf{g}_{\text{phys}}(\mathbf{v}_\alpha^0)$ encodes the transported initial and boundary conditions $\mathbf{v}_\alpha^0$,
- $\mathbf{h}_i(\mathbf{x})$ evaluates the spatial basis functions at $\mathbf{x} \in \Omega_0$,
- $\mathbf{w}_i\big(\mathbf{g}_{\text{geo}}(\alpha), \mathbf{g}_{\text{phys}}(\mathbf{v}_\alpha^0)\big)$ computes the coefficients by combining the outputs of the geometry and physical condition encoders.

This architecture is well-suited to our framework since all solutions and inputs are mapped to the fixed reference domain Ω_0 and fixed grid at that, enabling the network to learn the latent operator $\mathcal{F}_0$ efficiently. By incorporating shape information and input conditions through separate encoders, the network captures the dependence of the solution operator on both geometry and PDE parameters.

2.2 PDE examples

2.2.1 Laplace Equation

In this section, we describe the example we use to illustrate the main ideas of the neural operator approach described in the previous section. Specifically, we construct three different

mappings $\varphi_\alpha : \Omega_\alpha \to \Omega_0$ with different degrees of regularity on the mapped solutions and compare how well and how efficiently each of them allowed the latent neural operator to learn $\mathcal{F}_0$ and, via composition with φ_α^{-1}, $\mathcal{G}_\alpha$

We use the 2D Laplace equation as a test case for our neural operator framework. 2D Laplace equation emerges in many fields of science and engineering. For example, in fluid dynamics, it can represent the solution of potential flow where both vorticity and viscosity are considered negligible. This simplification allows us to model the velocity field as the gradient of a scalar potential field, $u(x, y)$. Under the assumptions of incompressibility, the governing equation for the potential field reduces to the Laplace equation:

$$\nabla^2 u = 0 \quad \text{on } \Omega_\alpha \tag{2.1}$$

$$\mathbf{n} \cdot \nabla u = 0 \quad \text{on } \partial\Omega_\alpha^I \tag{2.2}$$

$$\mathbf{n} \cdot \nabla u = b_\alpha(s) \quad \text{on } \partial\Omega_\alpha^o \tag{2.3}$$

where the boundary $\partial\Omega_\alpha$ is the union of the inner and outer boundaries:

$$\partial\Omega_\alpha = \partial\Omega_\alpha^I \cup \partial\Omega_\alpha^o \tag{2.4}$$

and the $b_\alpha(s)$ in Equation 2.3 represents the Neumann Boundary Condition on the outer side of the disk. $\mathbf{n}$ is the normal vector perpendicular to $\partial\Omega_\alpha$ for both boundaries.

Compatibility Condition

For the Laplace equation to possess a valid solution, the specified boundary conditions must satisfy a compatibility condition. The condition is represented as:

$$\int_{\Omega_\alpha} \nabla^2 u \, dA = \int_{\partial\Omega_\alpha^o} \mathbf{n} \cdot \nabla u \, ds = \int_{\partial\Omega_\alpha^o} b_\alpha(s) \, ds = 0 \tag{2.5}$$

where the second equality is derived from the Divergence theorem.

2.2.2 Flow Past Ellipsoid

We consider the incompressible Navier-Stokes equations in a three-dimensional domain exterior to an ellipsoid. The fluid motion is governed by:

$$\rho\frac{\partial \mathbf{u}}{\partial t} + \rho(\mathbf{u}\cdot\nabla)\mathbf{u} = -\nabla p + \mu\nabla^2\mathbf{u} \quad \text{in } \Omega_\alpha \tag{2.6}$$

$$\nabla\cdot\mathbf{u} = 0 \quad \text{in } \Omega_\alpha \tag{2.7}$$

where $\mathbf{u}(\mathbf{x},t)$ represents the velocity field, $p(\mathbf{x},t)$ is the pressure, ρ is the fluid density, and μ is the dynamic viscosity.

The boundary conditions are specified as follows:

$$\mathbf{u} = \mathbf{U}_\infty \quad \text{on } \Gamma_{\text{in}} \tag{2.8}$$

$$p\mathbf{n} - \mu(\nabla\mathbf{u} + \nabla\mathbf{u}^T)\cdot\mathbf{n} = 0 \quad \text{on } \Gamma_{\text{out}} \tag{2.9}$$

$$\boldsymbol{\sigma}\cdot\mathbf{n} = 0 \quad \text{on } \partial\Omega_\alpha^o \setminus (\Gamma_{\text{in}} \cup \Gamma_{\text{out}}) \tag{2.10}$$

$$\mathbf{u} = 0 \quad \text{on } \partial\Omega_\alpha^I \tag{2.11}$$

where $\mathbf{U}_\infty$ is the free-stream velocity, $\boldsymbol{\sigma}$ is the stress tensor, and $\mathbf{n}$ is the unit normal vector.

The inner boundary $\partial\Omega_\alpha^I$ is defined as an ellipsoid given by the quadratic form:

$$\mathbf{x}^T A\mathbf{x} = 1 \tag{2.12}$$

where A is a symmetric positive definite matrix. For a sphere of radius 0.5, $A = 4I$. Two approaches can be used to map between the reference unit sphere domain Ω_0 and the target spheroidal domain Ω_α:

Eigenvalue Decomposition Approach The mapping can be constructed using eigenvalue decomposition of A:

$$A = V\Lambda V^T \tag{2.13}$$

where V contains the eigenvectors and Λ is a diagonal matrix of eigenvalues. The mapping is then:

$$\varphi_\alpha^{-1}(\mathbf{x}) = \sqrt{\Lambda}V^T\mathbf{x} \tag{2.14}$$

This transformation first rotates the coordinate system to align with the principal axes of the ellipsoid (through V^T), and scales along these axes (through $\sqrt{\Lambda}$).

Cholesky Decomposition Approach Alternatively, using Cholesky decomposition:

$$A = \Phi\Phi^T \tag{2.15}$$

where Φ is a lower triangular matrix. The mapping is then simply:

$$\varphi_\alpha^{-1}(\mathbf{x}) = \Phi\mathbf{x} \tag{2.16}$$

We adopt the Cholesky decomposition due to better preservation of spatial locality as it does not involve rotations of the coordinate system.

For a point $\mathbf{x} \in \partial\Omega_0^I$ satisfying $\mathbf{x}^T\mathbf{x} = 1$, both mappings transform it to a point $\mathbf{y} \in \partial\Omega_\alpha^I$ on the spheroid satisfying $\mathbf{y}^T A\mathbf{y} = 1$. While this linear mapping is effective for ellipsoidal geometries, future work will explore extensions to more general shapes using nonlinear transformations such as quasi-conformal mapping or LDDMM.

Chapter 3

Methodology

This chapter presents the methodological framework underlying our project, which encompasses both the numerical methods employed for generating training data and the machine learning architecture of DIMON. To develop and train the DIMON framework, we rely on high-fidelity numerical simulations that provide the necessary training data. The following sections detail the numerical methods utilized, including the finite element formulation and the immersed boundary method, which form the foundation of our simulation approach. Subsequently, we elaborate on the DIMON framework itself, presenting its architectural design and implementation. This chapter aims to provide a comprehensive yet concise overview of both the data generation process and the machine learning methodology, establishing the technical groundwork for our results and subsequent analysis.

3.1 Numerical Methods

To train the DIMON framework, we rely on high-fidelity numerical simulations that provide the necessary training data. Our numerical approaches vary according to the specific problem configurations: both the Laplace equation and the gear-shaped Couette flow problems employ finite element formulations, while the three-dimensional flow past bluff bodies utilizes an immersed boundary method for structural representation. The following sections detail these numerical methods and provide key code in FEniCSx[1].

3.1.1 Finite Element Applied to our Laplace problem

The Laplace equation $\nabla^2 u = 0$ can be solved in weakly by:

$$\int_{\Omega_\alpha} \nabla^2 u \cdot \tilde{u}\, dx = 0 \tag{3.1}$$

We let $u \in U$ be the trial function and $\tilde{u} \in U$ be the test function, where U is the function space defined on the target domain:

$$U = \mathcal{U}(\Omega_\alpha; \mathbb{R}^2) \tag{3.2}$$

Applying the divergence theorem:

$$\begin{aligned} 0 &= \int_{\Omega_\alpha} \nabla^2 u \cdot \tilde{u}\, dx = \int_{\Omega_\alpha} \nabla \cdot (\nabla u) \cdot \tilde{u} dx \\ &= \int_{\partial\Omega_\alpha} \langle \nabla u, \mathbf{n} \rangle \tilde{u} ds - \int_{\Omega_\alpha} \langle \nabla u, \nabla \tilde{u} \rangle dx \end{aligned} \tag{3.3}$$

The variational form of the Laplace equation reads: find $u \in U$ such that

$$a(u, v) = L(\tilde{u}) \quad \forall \tilde{u} \in U \tag{3.4}$$

where the bilinear form $a(u, v)$ and linear form $L(v)$ are defined as:

$$\begin{aligned} a(u, \tilde{u}) &= \int_{\Omega_\alpha} \langle \nabla u, \nabla \tilde{u} \rangle dx \ = \\ L(v) &= \int_{\partial\Omega_\alpha^o} b \cdot v ds - \int_{\partial\Omega_\alpha^I} 0 \cdot \tilde{u} ds \end{aligned} \tag{3.5}$$

In FEniCSx notation, due to use of ufl, we got a high level representation which is very close to our mathematical expressions:

```
# Define function space
V = fem.functionspace(mesh, ("Lagrange", 1))
```

```
# Define trial and test functions
phi = ufl.TrialFunction(V)
v = ufl.TestFunction(V)

# Define variational form
a = inner(grad(phi), grad(v)) * dx
L = inner(g, v) * ds

# Solve the problem
problem = LinearProblem(a, L, bcs=[])
uh = problem.solve()
```

where `g` represents the Neumann boundary condition $\nabla\phi \cdot n = g$ on Γ_o. In the code, we set `bcs=[]`. This means that there is no Dirichlet Boundary condition specified in our problem.

3.1.2 Immersed Boundary Method

The Immersed Boundary Method (IBM) has emerged as a powerful computational approach for simulating fluid-structure interactions, particularly for problems involving rigid bodies immersed in fluid flows [20]. Originally developed by Peskin for studying blood flow in the heart [19], the method has since been extended to handle a wide range of applications, including rigid body dynamics [10][11]. In the context of flow past fixed rigid bodies, two primary implementations have gained prominence: the dual representation approach [11] and the direct forcing method [2][10].

The dual representation approach conceptualizes the rigid body as a collection of Lagrangian points connected by stiff springs to their target positions. These springs enforce the rigidity constraint by generating restoration forces that maintain the body's shape and

position. While this approach naturally extends from the classical IBM formulation for elastic boundaries, the spring constants must be carefully chosen to balance numerical stability with rigid body behavior. Too soft springs may lead to undesirable deformation, while excessively stiff springs can introduce numerical stiffness into the system. In contrast, the direct forcing method calculates the exact force required to maintain the rigid body's position and shape at each timestep. This approach directly enforces the no-slip condition at the fluid-solid interface by computing the force that makes the interpolated fluid velocity match the prescribed body velocity. The direct forcing method eliminates the need for artificial spring constants and provides better numerical stability, making it particularly suitable for applications where we really want to bump up the time step.

For our investigation of flow past a fixed bluff body, we adopt the direct forcing approach implemented within the IBAMR (Immersed Boundary Adaptive Mesh Refinement) framework [7]. This choice is motivated by framework's efficient handling of the coupled fluid-structure system through adaptive mesh refinement strategies. The following subsections provide the mathematical formulation of both approaches, with particular emphasis on the direct forcing method employed in our simulations.

For our study of flow past a fixed rigid body, we employ the direct forcing method to enforce the no-slip boundary condition. In this approach, we compute the force required to maintain zero velocity ($\mathbf{U}_b = \mathbf{0}$) at the immersed boundary points. The method proceeds as follows:

First, we solve the unconstrained fluid equations to obtain a preliminary velocity field $\mathbf{u}^{n+1}$. At the immersed boundary points, we then compute the velocity difference:

$$\Delta \mathbf{U}^{n+1} = -\mathcal{R}\mathbf{u}^{n+1} \tag{3.6}$$

where $\mathcal{R}$ is the velocity interpolation operator. This expression simplifies from the general form since our body is fixed in place with $\mathbf{U}_b = \mathbf{0}$.

The constraint force is then calculated as:

$$\mathbf{F}_c^{n+1} = \frac{\rho}{\Delta t}\Delta \mathbf{U}^{n+1} \tag{3.7}$$

This force is subsequently spread back to the Eulerian grid using the spreading operator $\mathcal{S}$ to correct the fluid velocity field:

$$\mathbf{f}^{n+1} = \mathcal{S}\Delta \mathbf{F}_c^{n+1} \tag{3.8}$$

ensuring that the no-slip condition is satisfied at the immersed boundary. This approach directly enforces the desired boundary condition without introducing artificial spring constants or feedback parameters, making it particularly suitable for rigid, fixed bodies. This $\mathbf{f}^{n+1}$ is then taken as the source term of the Naiver Stoke Momentum equation.

3.2 Latent Solution Representations

The choice of spatial transformation $\varphi_\alpha : \Omega_\alpha \to \Omega_0$ plays a pivotal role in the representation of the latent solution u_α^0 on the reference domain Ω_0 and the form of the *diffeomorphic latent operator* $\mathcal{F}_0$. While it is not always possible to find a map that perfectly preserves the differential operator $\mathcal{L}_\alpha$ of a PDE, one can aim to construct maps that minimize deviations from this ideal. Such maps improve the regularity of the latent solution representation and enable a more accurate approximation of the true solution operator. They also decrease the sensitivity of the mapped solution fields to changes in the shape parameter α of the original domain Ω_α and lower the required dimensionality of the shape encoding “geometry branch" in the DeepONet for efficiently approximating $\mathcal{F}_0$.

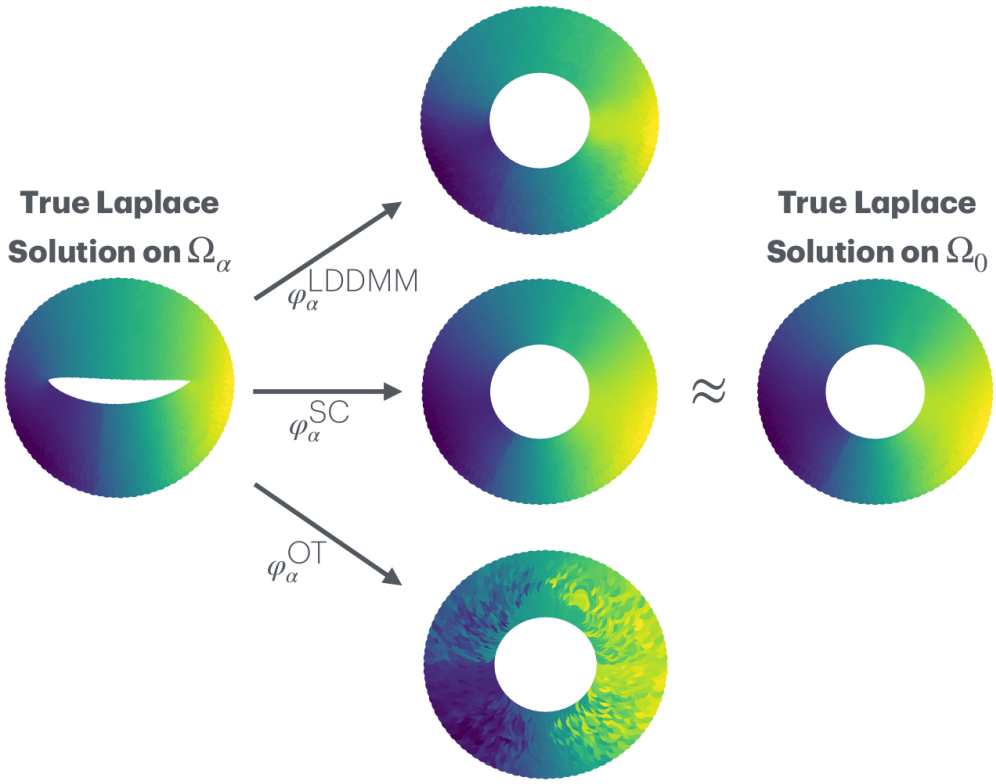

Figure 3.1: Visual comparison of mappings $\varphi_\alpha : \Omega_\alpha \to \Omega_0$ for different approaches—LDDMM, Schwarz-Christoffel (SC) map, and discrete optimal transport (OT)—and their impact on the Laplace solution.

3.2.1 Schwarz Christoffel Mapping for the Laplace Example

Conformal mapping is a transformation that preserves angles and the local shapes of structures. In the context of our application, conformal mapping has a particularly useful property: if a function g is harmonic, i.e., $\nabla^2 g = 0$, and φ is a conformal mapping on the 2D plane, then the transformed function $\varphi \circ g$ is also harmonic [18]. This property, known as the *conformal invariance of the Laplacian*, makes conformal mappings especially valuable when working with harmonic PDEs such as the Laplace equation.

Our objective is to find a conformal mapping from the target domains $\{\Omega_\alpha\}$ to a fixed reference domain Ω_0. Conformal mappings are generally difficult to compute analytically, especially for arbitrary shapes. To overcome this, we compute these mappings numerically using the MATLAB Schwarz–Christoffel Toolbox [5], developed by Toby Driscoll. The

Schwarz–Christoffel (SC) mapping is a powerful method for finding conformal maps from the unit disk or upper half-plane to polygonal domains (Figure 3.2) [4].

The Schwarz–Christoffel mapping is expressed as:

$$\varphi(z) = C + \int^{z} \prod_{k=1}^{n} (w - z_k)^{\omega_k - 1}\, dw, \tag{3.9}$$

where:

- z_k are the pre-images of the polygon's vertices on the unit disk or upper half-plane,
- ω_k are the interior angles at the vertices, scaled relative to π,
- C is an additive constant of integration.

This integral describes a holomorphic function that maps the source domain (e.g., the unit disk) to a polygonal target domain, preserving angles at the vertices. For doubly connected domains, SC mappings require extensions to account for inner and outer boundaries, which are handled by numerical tools such as the Schwarz–Christoffel Toolbox.

Let $\partial\Omega^I$ denote the inner boundary and $\partial\Omega^o$ denotes the outer boundary. Since the radius in our reference domain r is fixed, we must generate conformally equivalent shapes in our numerical experiment. After generating $\partial\Omega^I$ using the Jowkowsky approach, We utilize `extermap` in sc-toolbox to approximate a mapping from a unit disk to the exterior of $\partial\Omega^I$. Figure 3.2 shows the boundaries of original and mapped domains. From Ω_0 to Ω_α, we first invert the inner and outer radius by $h(z) = \frac{1}{z}$ Since $\partial\Omega_0^I$ is a unit circle, it is invariant under $h(z)$, and $\partial\Omega_0^I = \partial\Omega_0'^I$. Based on the shape of the target inner boundary $\partial\Omega_\alpha^I$, we use `extermap` to calculate a Schwarz–Christoffel mapping g_α that maps a unit circle to the exterior of $\partial\Omega_\alpha^I$. Then the outer boundary $\partial\Omega_0^O$ is mapped passively by the composition $\varphi_\alpha^{-1} = g_\alpha \circ h$:

$$\partial\Omega_\alpha^o = \varphi_\alpha^{-1}(\partial\Omega_0^o) = g_\alpha \circ h(\partial\Omega_0^o) \tag{3.10}$$

We leave the outer domain free to deform under f to make sure that the domains we generated are conformally equivalent to each other.

By leveraging conformal mappings, we transform solutions $u^\alpha \in \mathcal{U}(\Omega_\alpha)$ defined on doubly connected target domains to a fixed reference annulus $\Omega_0 = \mathbb{A}_r$. The mappings are computed numerically and used to standardize the input and solution spaces. Specifically, for a given target domain Ω_α, we compute the pullback of the input and solution functions via the conformal mapping φ_α:

$$v^0_{\alpha,k} = v^\alpha_k \circ \varphi^{-1}_\alpha, \quad u^0_\alpha = u^\alpha \circ \varphi^{-1}_\alpha. \tag{3.11}$$

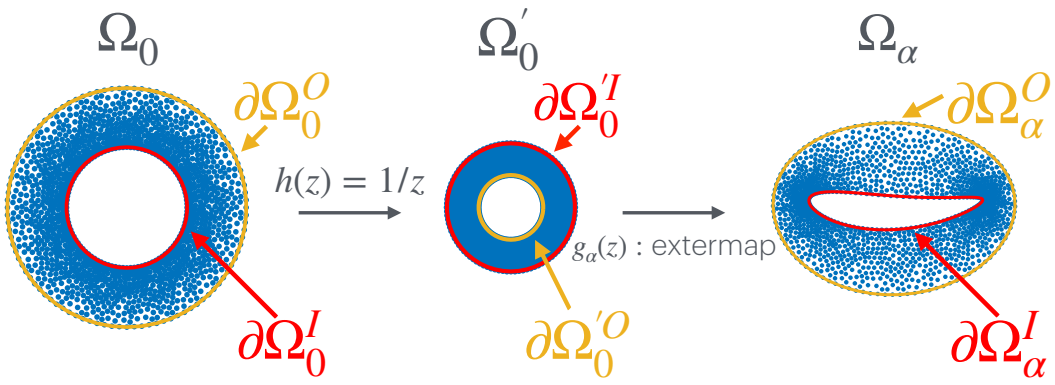

Figure 3.2: Schwarz-Christoffel (SC) composition diagram

This ensures that the latent neural operator for approximating $\mathcal{F}_0$ learns solutions in a geometry-independent space while preserving the harmonic structure of the transformed solutions due to the conformal invariance of the Laplacian. In theory, since the geometry is completely factored out, the only relationship we need to learn is between the boundary condition $f(s)$ and the solution representation in $\mathcal{F}_0$.

3.2.2 Other Mapping Methods: LDDMM and OT

We present another two other approaches for mapping as comparison with SC mapping we have discussed above: Large Deformation Diffeomorphic Metric Mapping (LDDMM)[23] and Discrete Optimal Transport (OT) Mapping[21]. Each method offers distinct characteristics

for domain standardization, with different trade-offs between smoothness, computational efficiency, and geometric preservation.

LDDMM provides a framework for constructing smooth, invertible transformations (diffeomorphisms) between shapes or domains. It models these transformations as flows induced by time-dependent vector fields, effectively creating a continuous morphing process between domains. The key advantage of LDDMM lies in its ability to preserve the topological properties and smoothness of the original domains, ensuring that geometric features are maintained throughout the transformation.

Let Ω_0 denote our reference domain and $\{\Omega_\alpha\}_{\alpha\in\mathcal{A}}$ represent our family of target domains. LDDMM constructs transformations $\varphi_\alpha : \Omega_\alpha \to \Omega_0$ that map each target domain to the reference domain while minimizing a deformation energy. These transformations enable us to represent input functions and solutions consistently in the reference domain through the mappings:

$$v^0_{\alpha,k} = v^\alpha_k \circ \varphi_\alpha,$$
$$u^0_\alpha = u^\alpha \circ \varphi_\alpha.$$

In contrast, discrete optimal transport (OT) offers a different approach by computing pointwise bijections between discretized representations of the domains. Given point clouds representing the reference domain Ω_0 and a target domain Ω_α, OT finds a mapping that minimizes the total transport cost between points. While this approach may not guarantee the smoothness properties of LDDMM, it provides computational efficiency and ensures uniform mass transport under a specified cost function.

In this project, we use a simplified version of the OT map where we assume that the probability measures on the point clouds are uniform and there is no mass splitting from the points when mapped. Since there is no mass splitting and the constraint on each point cloud has an equal number of points, T is a bijection and we have:

$$v^0_{\alpha,k} = v^\alpha_k \circ T^{-1},$$
$$u^0_\alpha = u^\alpha \circ T^{-1}.$$

LDDMM generates smooth, topology-preserving transformations but may require more computational resources. Discrete OT offers computational efficiency and guaranteed bijective mappings but may introduce discontinuities that could affect the spatial structure of solutions. In practice, both approaches enable the standardization of geometric variations, allowing neural operators to work in a unified latent space. We will show the accuracy comparison in the results section below.

Chapter 4

Results and Discussion

4.1 Laplace Equation

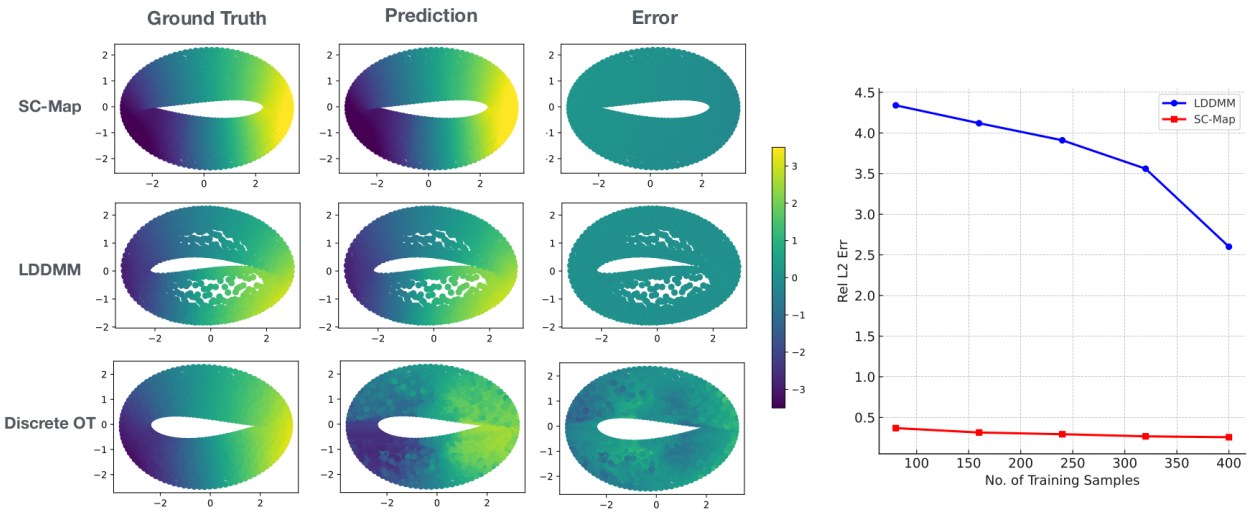

Figure 4.1: Comparison of mapping approaches (SC-Map, LDDMM, and Discrete OT) for the Laplace solutions on doubly connected domains.

In Figure 4.1, the first three columns illustrate the ground truth solution, predicted solution, and the error (ground truth - prediction) for each mapping approach. The rightmost panel shows the relative L_2 error as a function of the number of training samples, highlighting the superior performance and efficiency of SC-Map compared to LDDMM. These results highlight how the choice of mapping significantly impacts the prediction accuracy and error distribution.

The mapped solutions for each approach are visualized in Figure 3.1, the true Laplace solution on the original domain Ω_α is mapped to the reference domain Ω_0 using each mapping method. The SC map preserves the Laplace operator, resulting in an exact solution on Ω_0,

while LDDMM introduces minor distortions, and discrete OT introduces significant artifacts due to its non-diffeomorphic nature. The rightmost solution represents the true Laplace solution on the annulus Ω_0, illustrating the fidelity of each mapping.. Our main observations are summarized below:

- Conformal maps preserve the Laplace equation entirely, resulting in mapped solutions on the annulus that match the true Laplace solution with the same original boundary conditions. This property eliminates the dependence on the original domain geometry Ω_α, allowing the neural operator to learn a much simpler mapping.

- While diffeomorphic, the LDDMM mapping introduces slight distortions in the mapped solutions, deviating from the true Laplace solution on Ω_0.

- As expected for a non-diffeomorphic mapping, discrete OT introduces significant noise, leading to irregularities in the mapped solutions.

These observations confirm that the regularity of the mapped solutions strongly depends on the properties of the mapping. The closer the mapped solutions remain to being true PDE solutions in Ω_0, the more efficient and accurate the operator training becomes.

Table 4.1 summarizes the relative L_2 error, training epochs, and the number of PCA modes used for the geometry branch in the neural operator. For conformal maps, no geometry branch was required (0 PCA modes), as the mapping completely factors out the geometry variability and preserves the PDE. Conversely, for LDDMM and discrete OT, we used 10 PCA modes of the displacement vector field between Ω_α and Ω_0.

For training, the operator network utilized a branch network for the boundary conditions of the outer domain and a geometry branch for the original domain encoding (except in the conformal case). The conformal map's complete preservation of the Laplace operator reduced the problem's dimensionality, enabling the network to efficiently learn the relationship between the boundary condition and the mapped solution.

Figure 4.1 (right panel) shows how training sample size affects performance. Even with only 80 training samples, the network trained on conformally mapped solutions achieved relative L_2 errors an order of magnitude lower than LDDMM with 400 samples. As the training samples decreased for LDDMM, performance degradation was more pronounced, likely due to insufficient sampling of geometries to represent the displacement fields effectively.

$\varphi_\alpha : \Omega_\alpha \to \Omega_0$	PCA Modes	Epochs	Rel L_2 Error
Conformal Map	0	$1,000$	0.26%
LDDMM	10	$10,000$	2.56%
Discrete OT	10	$50,000$	22.4%

Table 4.1: Comparison of different mapping approaches based on PCA modes, training epochs, and relative L_2 error.

It is clear that the further away the mapped solutions are in Ω_0 from being solutions of the original PDE problem that was solved on Ω_α as if it were solved on Ω_0, the less effective training is. Most notably, Figure 4.1 shows that even with a fifth of the training data (80 samples), the network trained on the conformally mapped solutions is still an order of magnitude better than the results on the 400 training samples mapped to Ω_0 via LDDMM. We also see that as the training samples decrease for LDDMM, the performance degrades at a greater rate, presumably since not enough geometries were sampled to be fully expressive.

While the differences between the solutions mapped to the reference domain Ω_0 using LDDMM and conformal maps are subtle, the respective relative L_2 errors are an order of magnitude apart, emphasizing the impact of the mapping's construction on the data efficiency of the neural operator. The LDDMM map introduces minor distortions to the solution field, yet these deviations from the true Laplace solution on Ω_0 significantly degrade the neural operator's ability to generalize. This finding strongly suggests that even small adjustments to the mapping, particularly in a direction that better preserves the governing differential operator, can yield substantial improvements in the efficiency and accuracy of the latent neural operator $\mathcal{F}_0$.

For large-scale, higher-dimensional problems, identifying the optimal mapping φ_α from an uncountably infinite family of potential transformations is highly non-trivial. A promising avenue for exploration lies in the development of parameterized constructions of φ_α that explicitly enforce conservation laws and physical constraints of the solution field when mapping between Ω_α and Ω_0.

Integrating these conservation-driven mappings like conformal mapping into the neural operator framework requires careful consideration of how the resulting regularity in Ω_0 affects the training dynamics and parameterization of the neural network. Understanding the relationship between geometry, physics, and data efficiency is a critical research direction for advancing operator learning methods, especially in data-scarce settings.

4.2 Shear Stress Prediction

4.2.1 CFD Simulation result

For our CFD simulations, we utilize an adaptive mesh refinement (AMR) approach as shown in Figure 4.2 and Figure 4.3. The computational domain is set according to the following dimensions:

Re	dx_fluid	ds_solid	refinement ratio	patch levels	Lx	Ly	Lz
50	0.0312	0.037	4	3	[-6,10]	[-4,4]	[-4,4]

Table 4.2: Simulation parameters for flow past spheroid

We simulate flow past solid shells with finite thickness, investigating the convergence of results as the shell thickness increases. As demonstrated in Figure 4.4, increasing the number of Lagrangian layers improves accuracy, with our results converging toward literature values[9] for the sphere case. This validation study confirms the reliability of our numerical approach.

The simulation employs adaptive time stepping with a constant CFL number. The inflow

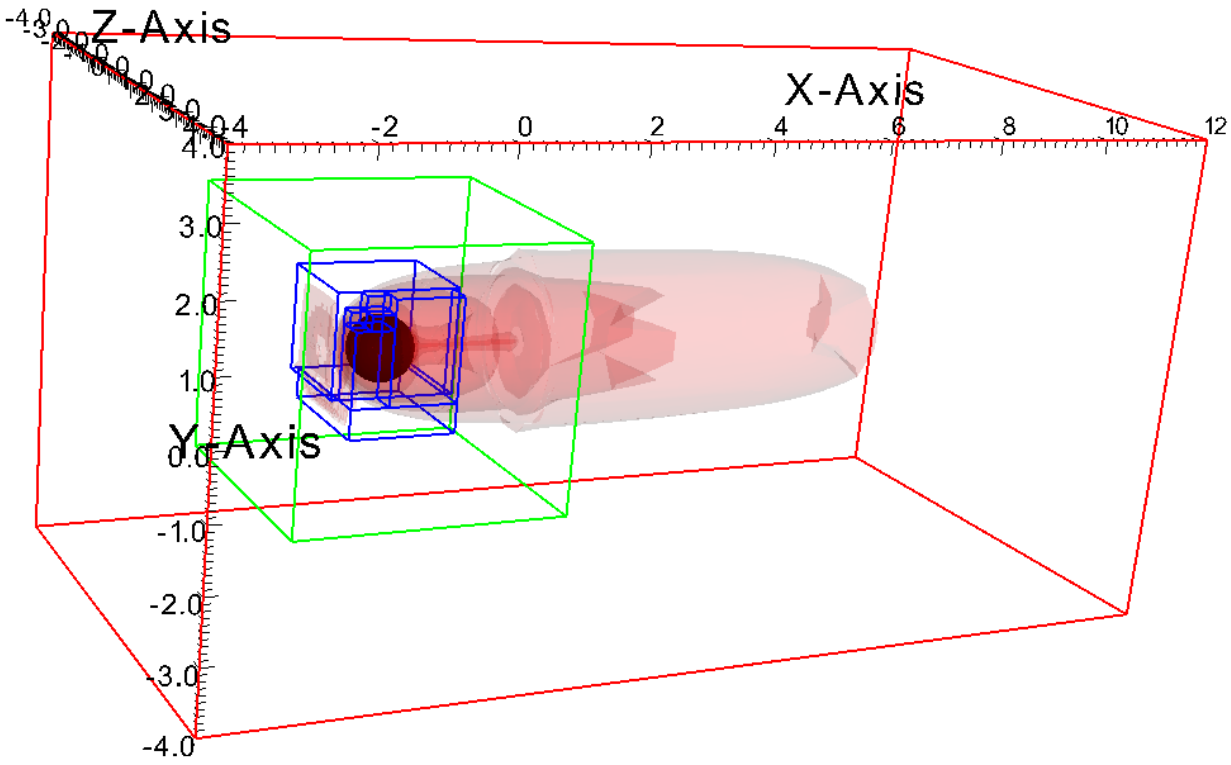

Figure 4.2: AMR boxes and vorticity magnitude contour

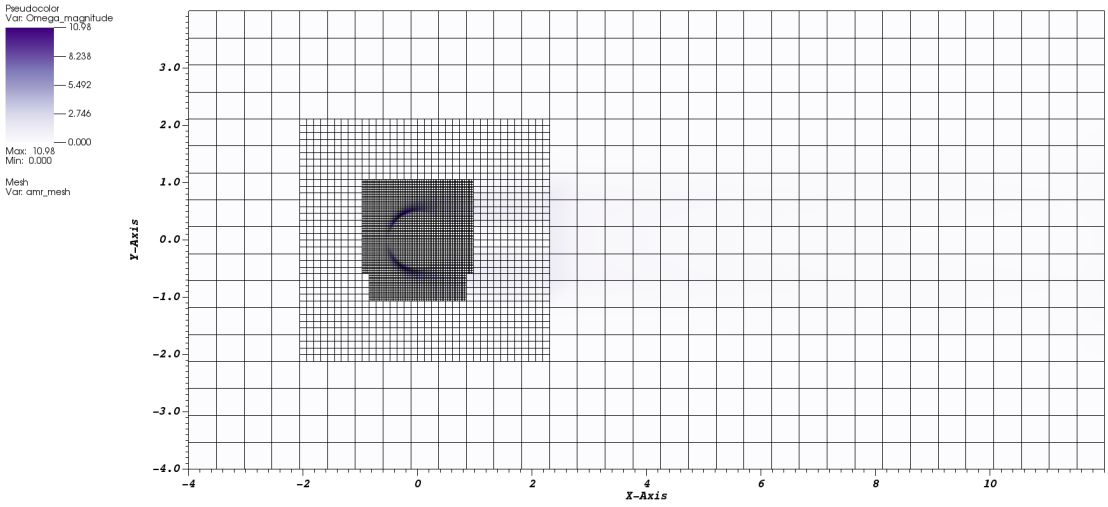

Figure 4.3: AMR mesh slice in the x-y plane at the midpoint for CFD computation

velocity gradually increases following a smooth ramping function:

$$g(t) = \frac{\tanh((t - t_{half})/\tau) + \tanh(t_{half}/\tau)}{1 + \tanh(t_{half}/\tau)} \tag{4.1}$$

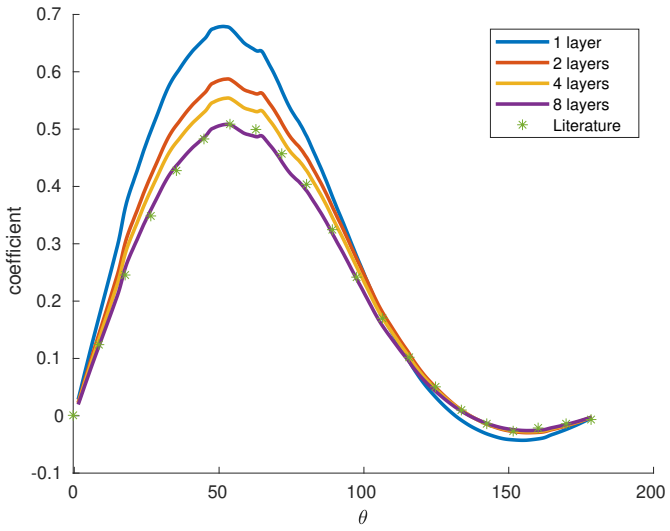

Figure 4.4: Relationship of number of layers and sphere surface shear stress compared with literature (Re=50)

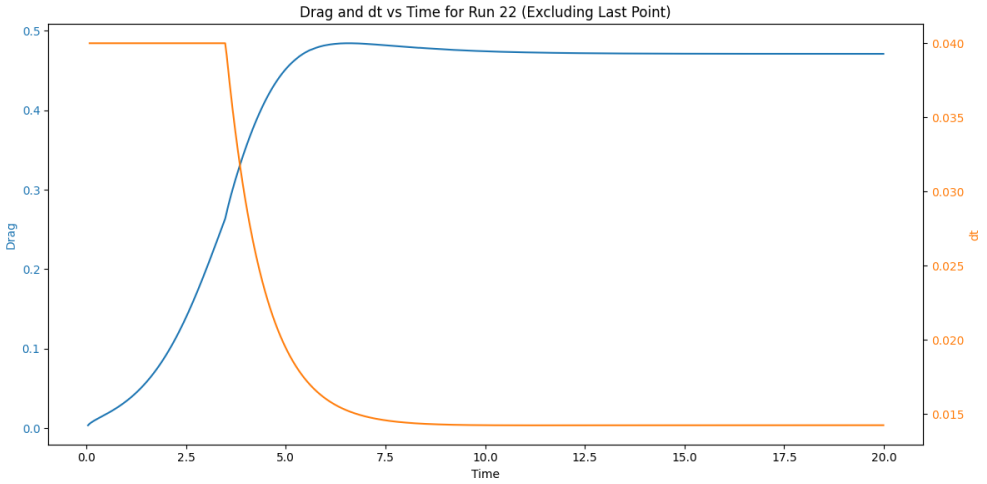

Figure 4.5: Drag history and time step size for one example of spheroid

The drag history shown in Figure 4.5 demonstrates stable behavior without oscillations, reaching a steady-state plateau. This confirms our choice of maximum simulation time (T_{max}) is sufficient for achieving steady-state conditions. For the sphere case, Figure 4.6 illustrates the surface shear stress distribution and corresponding streamlines, where θ is the angle made with $-x$ direction and ϕ is the angle in the y-z plane. The flow separation angle

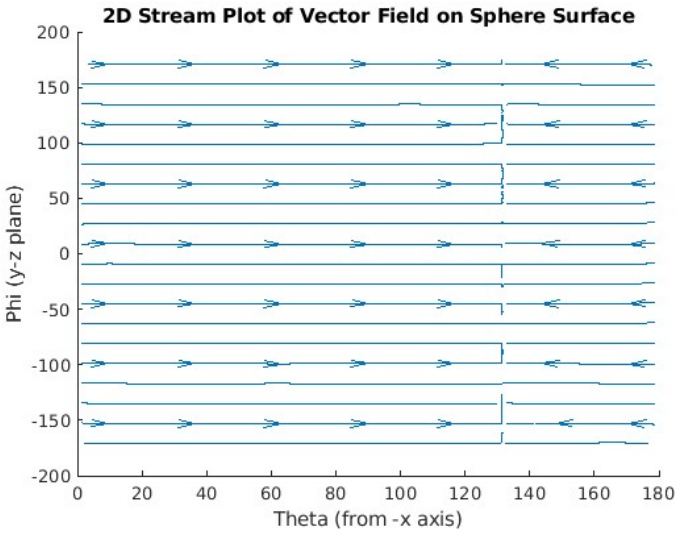

Figure 4.6: Shear Stress on a sphere

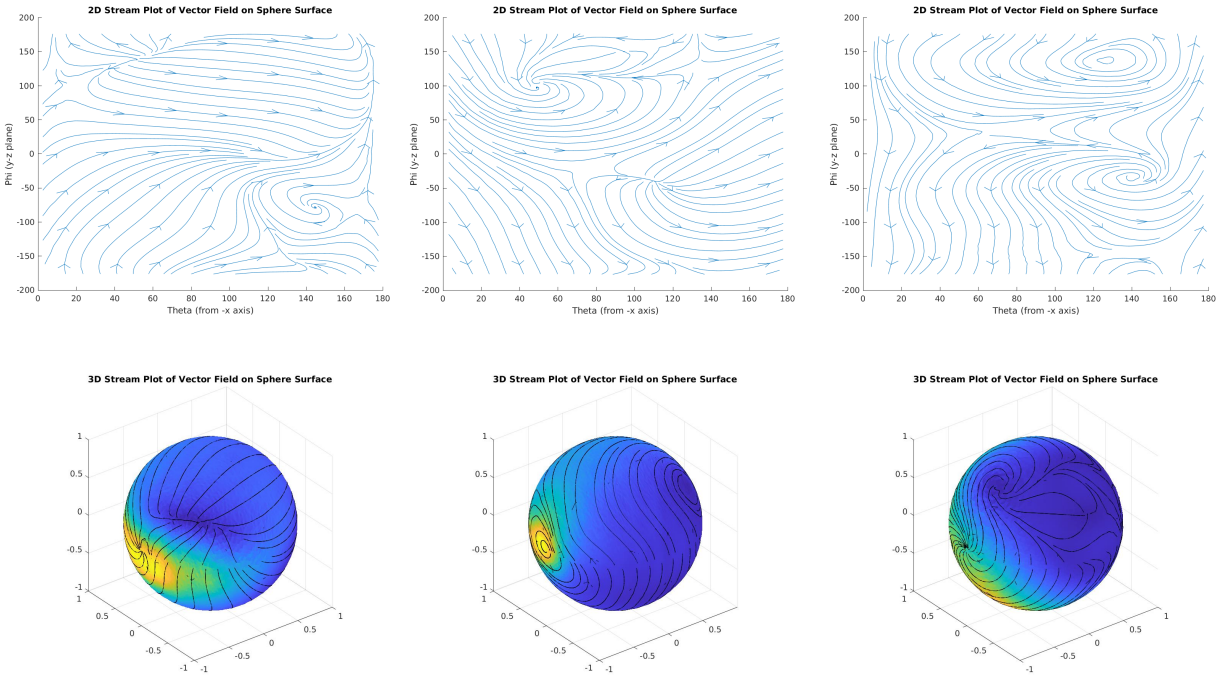

Figure 4.7: Three cases for shear stress streamlines under map induced by eigenvalue decomposition

is approximately 135 degrees, which aligns well with established literature values. Figure 4.7 presents the magnitude of shear stress and their projected streamlines using the eigenvalues

mapping on the sphere surface for different spheroid shapes.

4.2.2 Neural Network Prediction Result

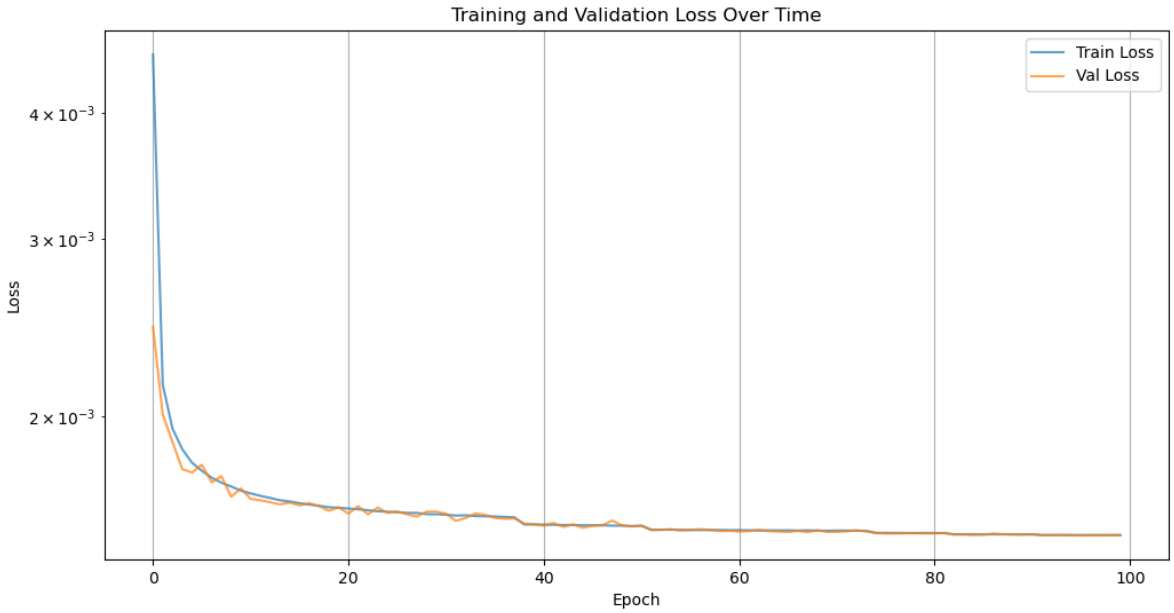

Figure 4.8: Training loss during each epoch

Our training process utilized 600 samples with 120 samples reserved for validation. The neural network architecture consists of two parallel networks - a branch network for processing the geometry information and a trunk network for handling spatial coordinates, a diagram of the network architecture is shown in Figure 4.9.

The model's training progression is illustrated in Figure 4.8. Our model demonstrates strong predictive capabilities, achieving overall mean squared error (MSE) of 0.001271 and mean absolute error (MAE) of 0.020846. The predictions show notably higher accuracy in transverse components (z-direction: MSE 0.000467, MAE 0.013006; y-direction: MSE 0.000487, MAE 0.013424) compared to the streamwise direction (x-direction: MSE 0.002859, MAE 0.036108). This variation in accuracy aligns with the physical characteristics of the flow, where the streamwise component typically exhibits the largest magnitude ($\approx$0.5) and

Algorithm 1: Diffeomorphic Neural Operator Architecture

Input:

$A \in \mathbb{R}^6$ (upper triangular elements of quadric matrix)

$(\theta, \varphi) \in \mathbb{R}^2$ (spherical coordinates)

Branch Network:

$y_{br} = \text{BranchNet}(A): \mathbb{R}^6 \to \mathbb{R}^{128}$

Layers: [6 → 32 → 64 → 128] with LayerNorm and Tanh

Trunk Network:

$y_{tr} = \text{TrunkNet}(\theta, \varphi): \mathbb{R}^2 \to \mathbb{R}^{128}$

Layers: [2 → 16 → 32 → 64] with LayerNorm and Tanh

Output:

$y = \text{OutputLayer}(y_{br} \cdot y_{tr}): \mathbb{R}^{128} \to \mathbb{R}^3$

Figure 4.9: Network Structure

most significant variations.

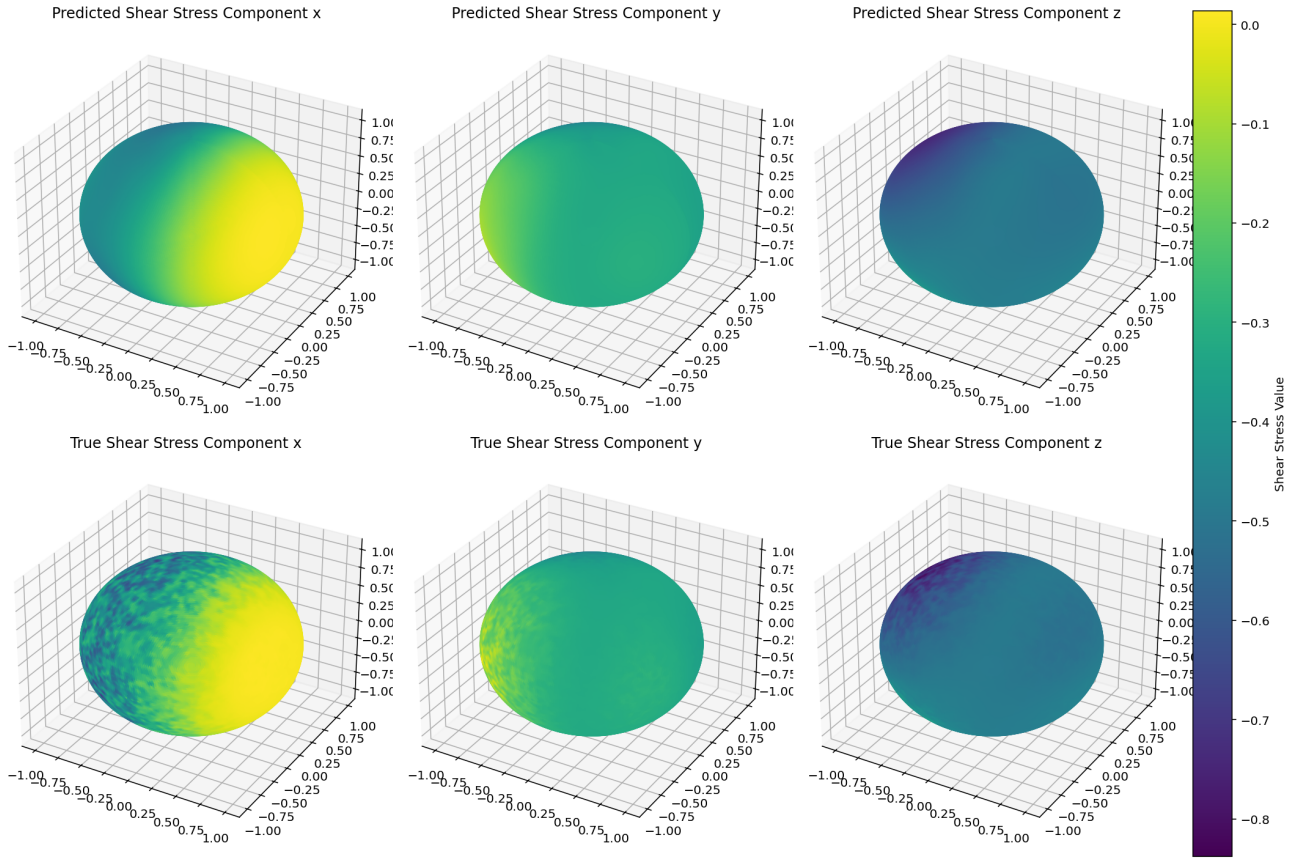

Figure 4.10: Neural Network Prediction for each component in one case

The prediction capabilities are demonstrated in Figure 4.10, showing the predicted com-

ponents of the shear stress field. Figure 4.11 presents a qualitative comparison between predicted and true streamline patterns, demonstrating good agreement in flow characteristics.

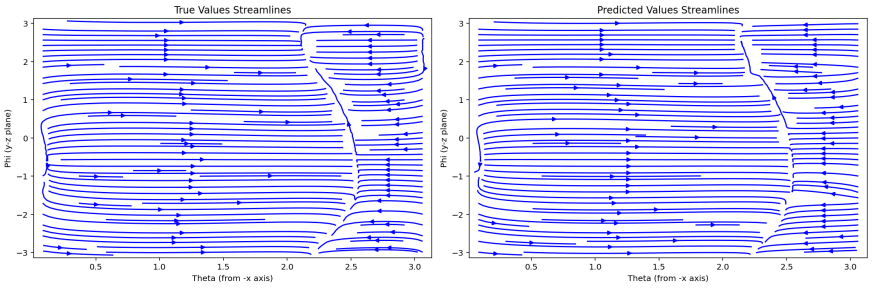

Figure 4.11: Neural Network Prediction v.s. True streamlines under Cholesky transform

While effective for the tested conditions, the model shows limitations in generalizing to higher Reynolds numbers, as it was trained only on steady-state solutions. For instance, while it can accurately predict separation line locations for spheres at the trained Reynolds number, the prediction accuracy decreases as the diameter varies significantly from the training conditions. In terms of computational efficiency, the IBAMR simulations require approximately one core hour per case, while the neural network training on 600 samples takes 2 hours on an NVIDIA Titan RTX (12GB). Once trained, inference requires only seconds, offering significant computational savings for parameter studies.

Future work will focus on extending the framework beyond ellipsoidal geometries to handle more complex shapes, requiring more sophisticated mapping strategies than the current quadratic form transformation. The current framework, while effective for predicting general flow patterns and shear stress distributions, exhibits several limitations and points to important directions for future research. A notable observation from our results, particularly visible in Figure 4.11, is the slight smoothing effect in the neural operator predictions compared to the ground truth solutions. This smoothing, while subtle, could become significant when attempting to generalize the approach to more complex geometries such as patient-specific atrial geometries, where capturing small-scale flow features is crucial for accurate thrombosis risk assessment. The challenge lies in the fundamental trade-off between the

neural network's ability to generalize across different geometries and its capacity to resolve fine-scale structural details in the flow field.

A deeper consideration from a geometric perspective involves our treatment of surface vector fields. Our current approach handles each component of the shear stress vector independently as scalar quantities. However, a more geometrically principled approach would involve properly transforming these vectors under the mapping φ_α between domains. The accuracy of such vector field predictions would inherently depend on the geometric properties preserved by our mapping. For instance, a mapping that better preserves angles between vectors (more conformal-like) would likely yield more accurate predictions of directional quantities. This geometric consideration leads to an important theoretical challenge: unlike the 2D case demonstrated in our Laplace example, conformal mappings between three-dimensional manifolds are not guaranteed to exist. While it is possible to compute quasi-conformal or approximately angle-preserving mappings, such computations often come with significant computational overhead. This presents a practical trade-off between the quality of the geometric mapping and its computational feasibility, particularly in clinical applications where rapid assessment may be required.

These observations suggest several promising directions for future research: developing neural architectures better suited for capturing multi-scale phenomena, investigating more sophisticated vector field transformation approaches, and exploring efficient computational strategies for high-quality geometric mappings in three dimensions. The ultimate goal would be to balance computation time/accuracy with practical applicability in clinical settings.

Chapter 5

Conclusions

This thesis has presented a novel framework for learning PDE solution operators across varying geometries through the use of diffeomorphic mappings to a reference configuration. Through comprehensive numerical experiments on both the 2D Laplace equation and 3D flow past spheroids, we have demonstrated that the choice of mapping significantly impacts both the accuracy and data efficiency of neural operator learning.

Our key findings include:

- For the 2D Laplace equation, conformal mappings that preserve the differential operator enable learning with significantly reduced data requirements - achieving better accuracy with 80 samples than LDDMM achieves with 400 samples. This highlights how preserving mathematical structure through mapping can fundamentally improve learning efficiency.

- In the 3D flow case, we successfully extended the framework to predict surface shear stress distributions on spheroids, achieving mean squared errors of 0.001 across components. This demonstrates the framework's applicability to complex nonlinear PDEs beyond the initial 2D proof-of-concept.

- We identified important limitations and trade-offs in our current approach, particularly the smoothing effect in neural operator predictions and the challenge of properly transforming vector quantities under mappings. For 3D problems, the non-existence of conformal mappings necessitates careful consideration of alternative mapping strategies.

These results suggest several promising directions for future work:

- Development of mapping strategies that better preserve differential operators in 3D
- Neural architectures specifically designed for multi-scale phenomena
- More geometrically principled approaches for handling vector fields
- Extension to more complex geometries beyond ellipsoidal domains
- Uncertainty and probabilistic prediction with confidence interval

The framework's success in predicting both 2D potential flows and 3D surface stresses with reduced data requirements suggests its promising value for applications like patient-specific blood flow modeling, where data acquisition is often limited. By carefully considering the interplay between geometry, physics, and machine learning, this work provides a foundation for more efficient and accurate PDE solution operators.

Bibliography

[1] Igor A Barrata et al. "DOLFINx: The next generation FEniCS problem solving environment". In: (2023).

[2] Amneet Pal Singh Bhalla and Neelesh A Patankar. "A unified constraint formulation of immersed body techniques for coupled fluid-solid motion: continuous equations and numerical algorithms". In: *arXiv preprint arXiv:2402.15161* (2024).

[3] Qianying Cao, Somdatta Goswami, and George Em Karniadakis. "Laplace neural operator for solving differential equations". In: *Nature Machine Intelligence* 6.6 (2024), pp. 631–640.

[4] TA Driscoll. *Schwarz-Christoffel mapping*. Cambridge University Press, 2002.

[5] Tobin A Driscoll. "Algorithm 756: A MATLAB toolbox for Schwarz-Christoffel mapping". In: *ACM Transactions on Mathematical Software (TOMS)* 22.2 (1996), pp. 168–186.

[6] Qi Gao et al. "A deep learning model for efficient end-to-end stratification of thrombotic risk in left atrial appendage". In: *Engineering Applications of Artificial Intelligence* 126 (2023), p. 107187. ISSN: 0952-1976. DOI: https://doi.org/10.1016/j.engappai.2023.107187. URL: https://www.sciencedirect.com/science/article/pii/S0952197623013714.

[7] IBAMR IBAMR. *An adaptive and distributed-memory parallel implementation of the immersed boundary method*. 2014.

[8] Pengzhan Jin, Shuai Meng, and Lu Lu. "MIONet: Learning multiple-input operators via tensor product". In: *SIAM Journal on Scientific Computing* 44.6 (2022), A3490–A3514.

[9] T. A. JOHNSON and V. C. PATEL. "Flow past a sphere up to a Reynolds number of 300". In: *Journal of Fluid Mechanics* 378 (1999), pp. 19–70. DOI: 10.1017/S0022112098003206.

[10] Bakytzhan Kallemov et al. "An immersed boundary method for rigid bodies". In: *Communications in Applied Mathematics and Computational Science* 11.1 (2016), pp. 79–141.

[11] Yongsam Kim and Charles S Peskin. "A penalty immersed boundary method for a rigid body in fluid". In: *Physics of Fluids* 28.3 (2016).

[12] Nikola Kovachki et al. "Neural operator: Learning maps between function spaces with applications to pdes". In: *Journal of Machine Learning Research* 24.89 (2023), pp. 1–97.

[13] Zongyi Li et al. "Fourier neural operator for parametric partial differential equations". In: *arXiv preprint arXiv:2010.08895* (2020).

[14] Zongyi Li et al. "Fourier neural operator with learned deformations for pdes on general geometries". In: *Journal of Machine Learning Research* 24.388 (2023), pp. 1–26.

[15] Miguel Liu-Schiaffini et al. "Neural operators with localized integral and differential kernels". In: *arXiv preprint arXiv:2402.16845* (2024).

[16] Shane E Loeffler et al. "Graph Fourier Neural Kernels (G-FuNK): Learning Solutions of Nonlinear Diffusive Parametric PDEs on Multiple Domains". In: *arXiv preprint arXiv:2410.04655* (2024).

[17] Lu Lu et al. "Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators". In: *Nature machine intelligence* 3.3 (2021), pp. 218–229.

[18] Thomas Parker and Steven Rosenberg. "Invariants of conformal Laplacians". In: *Journal of differential geometry* 25.2 (1987), pp. 199–222.

[19] Charles S Peskin. "Flow patterns around heart valves: a numerical method". In: *Journal of computational physics* 10.2 (1972), pp. 252–271.

[20] Charles S Peskin. "The immersed boundary method". In: *Acta numerica* 11 (2002), pp. 479–517.

[21] Gabriel Peyré, Marco Cuturi, et al. "Computational optimal transport: With applications to data science". In: *Foundations and Trends® in Machine Learning* 11.5-6 (2019), pp. 355–607.

[22] Minglang Yin et al. "Dimon: Learning solution operators of partial differential equations on a diffeomorphic family of domains". In: *arXiv preprint arXiv:2402.07250* (2024).

[23] Laurent Younes. *Shapes and diffeomorphisms*. Vol. 171. Springer, 2010.

[24] Zhiwei Zhao et al. "Diffeomorphism Neural Operator for various domains and parameters of partial differential equations". In: *arXiv preprint arXiv:2402.12475* (2024).