ADVANCES IN NUMERICAL PARTIAL DIFFERENTIAL EQUATIONS: FROM DISCRETIZATION-BASED SOLVERS TO NEURAL OPERATORS

By

ZEYUAN SONG

Bachelor of Science in Statistics
Bachelor of Arts in Law
Shandong University of Science and Technology
Qingdao, China
2019

Master of Science in Mathematics
University of Macau
Taipa, Macau
2022

Master of Science in Industrial Engineering and Management
Oklahoma State University
Stillwater, OK
2025

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
May, 2026

ADVANCES IN NUMERICAL PARTIAL DIFFERENTIAL EQUATIONS: FROM DISCRETIZATION-BASED SOLVERS TO NEURAL OPERATORS

Dissertation Approved:

Dr. Zheyu Jiang

Dissertation Advisor

Dr. Yu Feng

Dr. Hong Je Cho

Dr. Akash Deep

ACKNOWLEDGMENTS

I would like to express my deepest gratitude to my advisor, Dr. Zheyu Jiang, for his invaluable guidance and unwavering support throughout my doctoral studies. He has always been open to my ideas, even when they were diverse or unconventional, providing me with the freedom to explore while keeping me grounded. Beyond the technical research, I have learned a tremendous amount from him regarding academic writing, effective presentation, and professional integrity. His rigorous attitude toward science and his wisdom regarding life will continue to inspire me in my future career.

I would also like to extend my sincere thanks to my committee members, Dr. Yu Feng, Dr. Hong Je Cho, and Dr. Akash Deep. I am grateful for the time they took to review this dissertation and for their insightful comments and constructive feedback, which have significantly improved the quality of this work.

Moreover, I acknowledge the financial support provided by the Oklahoma State University and the National Science Foundation (under Award # 2442806). I am also grateful to the staff in the School of Chemical Engineering for their administrative assistance and kindness throughout my time here. I also want to thank Dr. Ieng Tak Leong from University of Macau, who first introduced me to the topic of adaptive Fourier decomposition.

Lastly, words cannot express my gratitude to my family. Thank you for your patience during the late nights and busy weekends. Your unwavering belief in me made this journey possible. Most importantly, I owe a debt of gratitude to my girlfriend. Your love, understanding, and sacrifice have been my greatest strength. Thank you for believing in me.

Acknowledgments reflect the views of the author and are not endorsed by committee members or Oklahoma State University.

Name: Zeyuan Song

Date of Degree: May, 2026

Title of Study: ADVANCES IN NUMERICAL PARTIAL DIFFERENTIAL EQUATIONS

Major Field: Chemical Engineering

Abstract:

Accurate numerical solutions of partial differential equations (PDEs) are crucial for numerous science and engineering applications, from precision agriculture and soil moisture monitoring to fluid dynamics and inverse problems. This dissertation presents comprehensive advances in numerical PDE solution methods by bridging traditional discretization-based approaches with modern machine learning techniques. We introduce the Message Passing Finite Volume Method (MP-FVM), a novel hybrid algorithm that holistically integrates adaptive fixed-point iteration, encoder-decoder neural networks, Sobolev training, and message passing within a finite volume framework to solve the highly nonlinear Richards equation governing water flow in unsaturated soils. We further introduce AFDONet, the first neural PDE solver whose architecture is fully guided by adaptive Fourier decomposition theory, enabling accurate solution representation on arbitrary Riemannian manifolds through adaptively selected poles parameterizing rational orthogonal bases in reproducing kernel Hilbert spaces. We extend AFDONet to inverse problems in Banach spaces through AFDONet-inv for handling parameters with sparse or discontinuous structures, and develop Adaptive Fourier Mamba Operators (AFMO) for efficient solution on irregular meshes with linear-time complexity. Finally, we propose a four-agent Large Language Model pipeline for automated end-to-end neural operator design, transforming neural architecture design from an art requiring rare interdisciplinary expertise into a systematic, science-based process.

Through extensive case studies from one-dimensional to three-dimensional problems, we demonstrate that MP-FVM achieves superior accuracy compared to state-of-the-art solvers including finite difference methods, physics-informed neural networks, and commercial HYDRUS software, with mass balance consistently exceeding ninety-five percent. AFDONet significantly outperforms existing neural operators including Fourier Neural Operator, Wavelet Neural Operator, and DeepONet on benchmark problems involving Helmholtz, Navier-Stokes, and Poisson equations, thanks to its deep connections with adaptive Fourier decomposition theory. The LLM-assisted design framework consistently outperforms human-designed baselines across diverse PDE benchmarks while requiring significantly less human effort. Overall, this work presents a new paradigm for designing explainable neural operator frameworks by systematically translating established mathematical theories into neural network components, providing practical tools for science and engineering.

TABLE OF CONTENTS

Chapter Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Figure Page

CHAPTER I

INTRODUCTION

1.1 Motivation and Background

A wide range of scientific and engineering phenomena can be characterized and modeled by partial differential equations (PDEs). These physiomechanical and physicochemical phenomena range from fluid dynamics to heat and mass transfer, to structural mechanics, and to quantum mechanics. Given the ubiquity of PDEs, most of them do not have analytical solutions and need to be solved numerically. However, traditional discretization-based numerical solvers, such as finite element methods and finite difference methods, can become quite slow, inefficient, and unstable, especially for large-scale problems or complex geometries (Hittinger & Banks, 2013; Sokic et al., 2011; Carey et al., 1993). The computational cost of traditional solvers grows substantially with finer resolutions, as well as for parametric studies where solutions must be computed for many different parameter configurations. Therefore, developing accurate, computationally efficient, and scalable PDE solution techniques is key to addressing critical challenges in sustainability, digital agriculture, Industry 4.0, and beyond.

For instance, understanding how soil moisture distributes in the root zone of a large crop field under different weather, topographic, and soil conditions is critical to designing a water-efficient irrigation scheduling system; and it is typically achieved by solving an agro-hydrological model that describes the movement of water through unsaturated soils. Most existing agro-hydrological models are based on the Richards equation (Richards, 1931), a highly nonlinear PDE. Solving the Richards equation numerically faces several fundamental challenges. The equation exhibits strong nonlinearity through the soil moisture diffusivity

and hydraulic conductivity functions, which can vary by several orders of magnitude within a single simulation. Additionally, the resulting discrete systems are often stiff and sparse, requiring sophisticated linearization and iterative solution techniques. Traditional finite difference methods (Celia et al., 1990) are straightforward to implement for regular grids. However, they may not guarantee stability for highly nonlinear problems (Meerschaert & Tadjeran, 2004) and cannot ensure local mass conservation. Finite element (Bergamaschi & Putti, 1999) and finite volume methods (Song & Jiang, 2023b) offer better conservation properties and can handle complex geometries, but require careful treatment of nonlinearities through linearization schemes. Among the iterative approaches, the fixed-point iteration scheme has gained attention for its convergence properties. Static fixed-point iterations use a fixed linearization parameter for all iterations, time steps, and discretized cells (Bergamaschi & Putti, 1999; Zeidler, 1986), while adaptive fixed-point iterations allow the parameter to vary, potentially offering better convergence behavior (Song & Jiang, 2023b). However, selecting the appropriate linearization parameter remains a challenge that often requires problem-specific tuning and expertise.

Meanwhile, data-driven methods, such as neural PDE solvers, can directly learn the trajectory of the family of equations from solution data, and thus can be orders of magnitude faster than traditional solvers once trained (Li et al., 2020b). Neural PDE solvers can amortize the computational cost across multiple parameter configurations, enabling rapid solution prediction for new parameters after the training phase. Recent advances in operator learning have led to several notable neural PDE solver architectures. DeepONet (Lu et al., 2019, 2021), inspired by the universal approximation theorem for nonlinear operators, learns mappings between infinite-dimensional function spaces through a branch-trunk architecture. The Fourier Neural Operator (FNO) (Li et al., 2020b, 2023b) performs convolution in the frequency domain to capture global spatial dependencies efficiently, achieving mesh-independent approximation of PDE solutions. Both paradigms have led to numerous variants, including Factorized FNO (Tran et al., 2021), Decomposed FNO (Li & Ye, 2025),

Wavelet Neural Operator (Tripura & Chakraborty, 2023), and Multiwavelet Neural Operator (Gupta et al., 2021), each targeting specific challenges such as computational efficiency, multi-scale phenomena, or adaptation to different geometries.

However, existing neural PDE solvers face several critical challenges. First, most existing approaches are designed for regular Euclidean domains, while many real-world applications involve PDEs defined on non-Euclidean manifolds or complex geometries. Most classical numerical approaches to solve PDEs on manifolds rely on parameterization (Lui et al., 2005), collocation (Chen & Ling, 2020), or spectral methods (Yan et al., 2023). Although researchers have begun to explore manifold-aware neural architectures that can learn directly from point clouds (He et al., 2024; Liang et al., 2024) or graphs (Bronstein et al., 2017), they cannot easily be generalized to different manifolds. The challenge lies in developing neural architectures that can handle the intrinsic geometry of manifolds while maintaining computational efficiency and solution accuracy.

Second, the design of exact neural architectures in many neural PDE solvers has been "more of an art than a science" (Sanderse et al., 2025), typically done in a bottom-up approach that involves significant intuition, expert experience, and trial-and-error experimentation. Neural architecture design often requires extensive hyperparameter tuning, selection of activation functions, choice of normalization techniques, and determination of network depth and width. Although neural operators such as FNO and DeepONet are grounded in theoretical insights, considerable manual effort and domain expertise are still required to adapt these architectures to specific PDE problems. Rigorous mathematical basis and explainability have been lacking in guiding the design of these neural architectures, making it difficult to understand why certain architectural choices lead to better performance or to predict which architectures will work well for new problems.

Third, extending neural operators to inverse problems, which are generally ill-posed, remains challenging, especially when parameters lie in sparse domains that are better modeled as Banach spaces rather than Hilbert spaces. Inverse problems for PDEs aim to identify

unknown parameters of a physical system from observations of its output. A large class of inverse problems are only well-defined as mappings from operators to functions. Traditional approaches to inverse problems include variational methods in Hilbert spaces (Engl, 2007), Bayesian inference techniques (Stuart, 2010), and optimization-based methods. However, it has been shown that a Banach space setting for the parameter space would be closer to reality for a wide range of problems, particularly when the parameters exhibit sparse or discontinuous structures (Grasmair et al., 2011; Clason et al., 2021). Existing operator learning frameworks either do not explicitly account for the underlying operator space or solve the inverse problems in a Hilbert space, limiting their applicability to problems with sparse parameter domains.

1.2 Research Objectives and Contributions

This dissertation addresses these challenges through a comprehensive research program that bridges traditional numerical methods with modern neural operator learning. The overarching goal is to advance the state of the art in numerical solutions of partial differential equations by developing novel algorithms and frameworks that combine the mathematical rigor and physical consistency of traditional methods with the computational efficiency and flexibility of data-driven approaches.

The first research objective is to develop robust, convergent numerical solvers for nonlinear, stiff PDEs by introducing hybrid algorithms that couple classical finite volume discretization with machine learning, specifically for solving the Richards equations. This objective recognizes that while traditional numerical methods provide strong theoretical guarantees and physical consistency, they can benefit significantly from data-driven components that adapt to the nonlinear characteristics of specific problems. We propose the Message Passing Finite Volume Method (MP-FVM) that integrates adaptive fixed-point iteration schemes with encoder-decoder neural networks and message passing mechanisms to enhance convergence and preserve mass conservation. The hybrid solver is both accurate and efficient for

highly nonlinear agro-hydrological problems.

The second research objective is to develop mathematically grounded neural operator architectures for solving PDEs on arbitrary manifolds, with rigorous theoretical foundations and explainable design principles. This objective addresses the fundamental limitation that most existing neural PDE solvers lack systematic design principles and are restricted to Euclidean domains. By adopting a top-down approach guided by established mathematical frameworks, specifically adaptive Fourier decomposition theory, we aim to create neural architectures whose every component has clear mathematical interpretation and justification. This approach not only enables solutions on arbitrary Riemannian manifolds but also provides convergence guarantees and performance bounds rooted in approximation theory.

The third research objective is to extend neural operator frameworks to inverse problems in Banach spaces, addressing ill-posed parameter estimation problems with sparse parameter domains. Inverse problems are ubiquitous in science and engineering, arising whenever we need to infer system parameters or properties from observed data. However, these problems are often ill-posed and require careful regularization. By developing operator learning frameworks that explicitly account for Banach space structures rather than restricting to Hilbert spaces, we aim to handle inverse problems where parameters naturally exhibit sparse or discontinuous characteristics, such as identifying piecewise constant material properties or localized sources.

The fourth research objective is to explore automated neural operator design using large language models, transforming the design process from an art into a science. The manual design of neural architectures for specific PDE problems requires substantial expertise in both the mathematical properties of the PDEs and the capabilities of neural network components. By developing systematic pipelines that leverage the reasoning capabilities of large language models, we aim to automate the process of translating mathematical theories into implementable neural architectures, making theory-guided neural operator design accessible to a broader community and accelerating the development of problem-specific solvers.

The key contributions of this dissertation, which span from hybrid numerical methods for soil moisture modeling to theory-guided neural operators and automated design frameworks, are summarized as follows:

1.2.1 Hybrid Data-Driven Numerical Methods

We introduce the Message Passing Finite Volume Method (MP-FVM), a novel solution algorithm that holistically integrates adaptive fixed-point iteration scheme, encoder-decoder neural network architecture, Sobolev training, and message passing mechanism in a finite volume discretization framework. The MP-FVM algorithm addresses the fundamental challenge of solving the highly nonlinear Richards equation by combining the strengths of classical numerical methods with modern machine learning techniques. At its core, the algorithm employs a finite volume discretization to ensure local mass conservation, which is critical for accurate long-term predictions of soil moisture dynamics. Unlike conventional finite volume methods that convert the discretized equation into a large, stiff matrix equation which can be challenging to solve, the MP-FVM algorithm adopts an adaptive fixed-point iteration scheme that solves the discretized Richards equations iteratively, providing a robust solution procedure with controllable convergence properties.

A key innovation of the MP-FVM algorithm is its integration of encoder-decoder neural network architecture with the message passing mechanism. The encoder-decoder architecture learns the complex nonlinear relationships between pressure head solutions obtained from different numerical solvers, capturing both the sensitivity to different parameter choices and the distinct topological features of solution spaces. The encoder maps pressure head solutions to a latent space, while the decoder reconstructs solutions from this latent representation, ensuring that essential topological features are accurately captured. The message passing mechanism, implemented within the latent space through a processor that operates iteratively, enhances the convergence and numerical stability of the algorithm. By defining latent variables that are solved iteratively using the adaptive fixed-point iteration scheme, the MP-

FVM algorithm enables the message passing mechanism to preserve physical consistency and mass conservation while achieving superior solution accuracy.

The MP-FVM algorithm incorporates Sobolev training in the loss functions for both encoder and decoder neural networks, adding regularization terms that enforce consistency not only at the function value level but also across derivatives. This ensures compatibility and stability across the solution space, preventing small perturbations in solutions at initial conditions or previous time steps from leading to slow convergence or inaccurate solutions at the final time step. The algorithm provides guaranteed convergence under reasonable assumptions, which we rigorously prove by showing that the iterative scheme is contractive, with the error decreasing geometrically at each iteration. This theoretical foundation, combined with the ability to leverage pre-trained models for transfer learning across different boundary and initial conditions, makes the MP-FVM algorithm both accurate and computationally efficient. The algorithm achieves fine-scale accuracy using coarse-grid training data through a coarse-to-fine approach, bypassing the need for computationally expensive high-resolution training datasets while maintaining excellent solution quality.

1.2.2 Theory-Guided Neural Operators

We introduce AFDONet (Adaptive Fourier Deep Operator Network), the first neural PDE solver whose architectural and component design is fully guided by adaptive Fourier decomposition (AFD) theory (Qian, 2010; Qian et al., 2012). AFD is a signal decomposition technique that leverages the Takenaka-Malmquist system and adaptive orthogonal bases to sparsely represent functions in reproducing kernel Hilbert spaces (RKHS). Unlike classical Fourier methods that use fixed global basis functions, AFD adaptively selects poles that parameterize rational orthogonal bases according to a maximal selection principle, enabling accurate representation of functions with localized features, sharp gradients, or non-periodic structures. By replicating the AFD framework in a neural architecture, we create a solver that exhibits exceptional mathematical explainability and groundness, where each compo-

nent of the network has clear interpretation in terms of the underlying approximation theory.

The AFDONet architecture consists of three main components designed following AFD principles: an encoder based on variational autoencoder (VAE) framework that maps PDE inputs to a latent space, a latent-to-RKHS network that projects latent representations to their nearest reproducing kernel Hilbert space where AFD operations are defined, and an AFD-type dynamic convolutional kernel network (CKN) decoder that reconstructs solutions through adaptive basis selection. The use of VAE as the backbone is motivated by the observation that many PDE solution fields lie on low-dimensional manifolds in high-dimensional function space, and the variational inference in VAE aligns well with the maximal selection principle in AFD (Chen et al., 2020a). The latent-to-RKHS network extends previous latent-to-kernel approaches (Lu et al., 2020a) by explicitly constraining the functional space to be an RKHS through feature maps that perform orthogonal projection, ensuring that the reproducing property is satisfied and enabling rigorous theoretical analysis.

AFDONet achieves outstanding solution accuracy on arbitrary Riemannian manifolds, significantly outperforming existing neural operators such as FNO, DeepONet, and Wavelet Neural Operator across diverse benchmark problems including the Helmholtz equation on planar manifolds, Navier-Stokes equation on tori, and Poisson equation on quarter-cylindrical surfaces. The superior performance stems from AFDONet's ability to adapt its basis functions to the specific geometry and solution characteristics of each problem. While FNO and its variants rely on fast Fourier transforms that are inherently defined on Euclidean domains and struggle with non-periodic boundaries, AFDONet uses adaptive rational bases parameterized by poles that are learned from input data, allowing the bases to locally adapt to sharp gradients, discontinuities, and complex geometries. Furthermore, AFDONet shows superior performance on datasets with sharp gradients due to its connection with holomorphic function theory. We extend Sobolev training (Czarnecki et al., 2017a) to the complex domain through a holomorphic training loss that enforces consistency between predicted and true solutions at both the function value level and across all orders of derivatives, capturing

the inherent smoothness and analytic structure of the target function.

The mathematical groundness of AFDONet enables us to provide rigorous convergence guarantees and theoretical foundations. We prove three main theoretical results: First, we bound the generalization error of AFDONet in terms of the number of training samples, network depth and width, and the smoothness of the target function, showing that with appropriate network scaling, the expected error decays polynomially in the number of samples. Second, we prove the existence of the RKHS constructed by our latent-to-RKHS network by extending results from approximation theory (Caragea et al., 2022), showing that for any function in a Hilbert space and any tolerance, there exists a neural network that maps the function to an RKHS with controlled approximation error. Third, we prove convergence of the dynamic CKN decoder by leveraging the convergence mechanism of AFD, establishing conditions on layer width, depth, and kernel complexity that ensure the reconstructed solutions converge to the true solutions. These theoretical results distinguish AFDONet from existing neural operators that lack such rigorous foundations.

We extend AFDONet to inverse problems (AFDONet-inv), operating in reproducing kernel Banach spaces (RKBS) rather than Hilbert spaces, addressing the challenge of sparse parameter domains in inverse problems. Inverse problems for PDEs aim to identify unknown parameters from observations of system outputs and are typically ill-posed, requiring careful regularization. While most existing operator learning frameworks assume parameters lie in Hilbert spaces, many real-world inverse problems involve parameters with sparse or discontinuous structures that are better modeled in Banach spaces, particularly L^1 or bounded variation spaces (Grasmair et al., 2011; Clason et al., 2021). AFDONet-inv extends the AFD framework from RKHS to RKBS by constructing appropriate reproducing kernels for Banach spaces and modifying the orthogonalization procedure to account for the duality structure of Banach spaces. The architecture explicitly represents the mapping from operator spaces to parameter spaces, enabling solution of inverse problems where both inputs and outputs are functional objects. We demonstrate that AFDONet-inv achieves superior accuracy and

stability compared to Hilbert space approaches on benchmark inverse problems involving parameter identification for elliptic PDEs with sparse coefficient fields.

1.2.3 Advanced Neural Operator Architectures

We develop Adaptive Fourier Mamba Operators (AFMO), which integrate reproducing kernels for state-space models with Takenaka-Malmquist systems, enabling accurate solutions on diverse geometries and meshes. Frequency-based neural operators such as FNO are attractive for their ability to capture global dependencies through spectral representations, but they face significant challenges when dealing with irregular geometries and non-uniform meshes. Traditional Fourier transforms require regular grids and periodic boundary conditions, limiting their applicability to complex real-world domains. To address these limitations, AFMO builds upon recent advances in state-space models, particularly the Mamba architecture, which has shown remarkable efficiency in sequence modeling tasks through selective state-space representations.

The key innovation in AFMO is the integration of reproducing kernel theory with state-space models to create a neural operator that can handle irregular geometries while maintaining the computational efficiency of frequency-based approaches. We construct reproducing kernels that are compatible with the state-space model's hidden state dynamics, allowing the network to learn representations that respect the geometry of the problem domain. The Takenaka-Malmquist system provides the theoretical foundation for adaptively selecting basis functions that can accurately represent solutions on irregular domains. By parameterizing the state-space model's matrices using these adaptive bases, AFMO can selectively focus on important spatial and temporal features while efficiently propagating information across the domain.

AFMO demonstrates superior performance on problems involving irregular geometries, non-uniform meshes, and complex boundary conditions. Unlike FNO which requires interpolation or padding to handle irregular domains, potentially introducing artifacts and reducing

accuracy, AFMO operates directly on point clouds or unstructured meshes. The state-space formulation enables linear-time complexity in sequence length, making AFMO particularly efficient for high-resolution simulations and long-time integration. We validate AFMO on benchmark problems including flow past obstacles with complex geometries, heat diffusion on irregular domains, and wave propagation in heterogeneous media, demonstrating improved accuracy and reduced computational cost compared to existing neural operators.

1.2.4 Automated Neural Operator Design

We propose a four-agent Large Language Model (LLM) pipeline consisting of specialized agents (Theorist, Programmer, Critic, Refiner) that designs mathematically grounded neural operators end-to-end. The design of neural operators for specific PDE problems currently requires substantial expertise in both the mathematical properties of the equations and the architectural patterns of neural networks. Domain experts must understand the structure of the PDE, identify appropriate functional spaces, select suitable basis representations, and translate these insights into implementable neural architectures through extensive trial and error. This process is time-consuming, requires rare interdisciplinary expertise, and often results in suboptimal designs due to the vast space of possible architectural choices.

Our LLM-assisted framework automates this design process while maintaining mathematical rigor and grounding. The framework consists of four specialized agents, each responsible for a distinct phase of the design process. The Theorist agent takes as input a description of the PDE problem and relevant mathematical theories, then reasons about the key mathematical structures that should be reflected in the neural architecture. Drawing on its broad knowledge of mathematical theories, approximation methods, and operator theory, the Theorist identifies suitable function spaces, proposes appropriate basis representations, and outlines the mathematical framework that should guide the architecture design. The Programmer agent translates the Theorist's mathematical blueprint into executable code, making specific choices about network layers, activation functions, training procedures, and

implementation details while remaining faithful to the mathematical principles identified by the Theorist.

The Critic agent evaluates the designed architecture both theoretically and empirically. It checks whether the implementation correctly reflects the intended mathematical structures, identifies potential issues such as numerical instabilities or violations of physical constraints, and suggests improvements based on mathematical analysis. The Critic performs both static analysis of the code and dynamic analysis of training behavior, checking for issues like gradient pathologies, inappropriate initialization, or insufficient expressiveness. Finally, the Refiner agent iteratively improves the architecture based on feedback from the Critic, making adjustments to address identified issues while preserving the core mathematical framework. This refinement process continues until the architecture meets specified quality criteria in terms of both mathematical soundness and empirical performance.

This LLM-assisted framework consistently outperforms human-designed baselines across diverse PDE benchmarks spanning different equation types, domain geometries, and physical phenomena. We evaluate the framework on benchmark problems including advection-diffusion equations, Burgers' equation, Navier-Stokes equations, and various elliptic and parabolic PDEs. The automatically designed architectures achieve comparable or superior accuracy to carefully hand-crafted baseline methods while requiring significantly less human effort. Moreover, the framework demonstrates good generalization, producing effective architectures for problems that differ from those seen during the development of the pipeline, suggesting that the LLMs have learned general principles of neural operator design rather than memorizing specific patterns.

The framework transforms neural operator design from an art requiring rare expertise into a more systematic, science-based process. By explicitly grounding the design in mathematical theory and automating the translation from theory to implementation, we make theory-guided neural operator design accessible to researchers who may have deep understanding of their specific PDE problems but limited expertise in neural architecture design.

The framework also enables rapid prototyping and exploration of different theoretical frameworks, accelerating the development of problem-specific solvers. We demonstrate that the framework is reliable across most mathematical theories commonly used in PDE analysis, including spectral methods, finite element methods, kernel methods, and operator splitting techniques, showing that it can effectively leverage diverse mathematical tools to create specialized neural architectures. This work opens new directions for theory-aware automated scientific machine learning, where mathematical insights systematically guide the construction of data-driven models.

1.3 Organization of Dissertation

This dissertation is organized into eight chapters that progressively build from traditional numerical methods to advanced neural operator frameworks and automated design techniques.

Chapter 2 presents the Message Passing Finite Volume Method (MP-FVM) for solving the Richards equation. We begin by formulating the Richards equation in a finite volume discretization framework and derive the adaptive fixed-point iteration scheme that provides a robust iterative solution procedure. We introduce a novel adaptive rule for updating the linearization parameter, where the parameter adjusts dynamically with respect to space, time, and iteration count based on monitoring the condition number of the resulting system matrix and controlling the relative error between successive iterations. This ensures the numerical scheme is well-posed and reaches convergence within the specified number of iterations. The chapter then presents the integration of encoder-decoder neural network architecture with the message passing mechanism. We discuss dataset preparation and data augmentation strategies, including the use of Gaussian noise to enhance generalization performance, and introduce Sobolev training to ensure compatibility and stability across the solution space. The message passing process operates in the latent space defined by the encoder-decoder networks, solving for latent variables iteratively using the adaptive fixed-

point iteration scheme. We rigorously prove convergence guarantees for the MP-FVM algorithm under reasonable assumptions, showing that the iterative scheme is contractive. The effectiveness of the algorithm is demonstrated through comprehensive case studies in one, two, and three dimensions, including benchmark problems with known analytical solutions, layered soil problems with discontinuous properties, and realistic irrigation scenarios. Comparisons with state-of-the-art solvers including finite difference methods, physics-informed neural networks, and commercial HYDRUS software demonstrate that MP-FVM achieves superior accuracy, better preserves mass conservation and underlying physical relationships, and maintains excellent computational efficiency.

Chapter 3 introduces AFDONet, a theory-guided neural operator for solving PDEs on smooth manifolds. We begin with preliminaries on adaptive Fourier decomposition theory, explaining the Takenaka-Malmquist system, reproducing kernel Hilbert spaces, and the maximal selection principle for adaptive pole selection. The chapter then presents the AFDONet architecture, systematically deriving each component from AFD theory. We explain the design of the VAE-based encoder, the latent-to-RKHS network that projects latent representations to their nearest reproducing kernel Hilbert space, and the AFD-type dynamic convolutional kernel network decoder that reconstructs solutions through adaptive basis selection. Each architectural choice is justified by its connection to AFD theory, demonstrating the top-down, theory-guided design approach. We present three main theoretical results: bounds on the generalization error of AFDONet, proof of existence of the RKHS constructed by the latent-to-RKHS network, and convergence guarantees for the dynamic CKN decoder. Extensive experimental validation is provided on benchmark problems including the Helmholtz equation on planar manifolds with perfectly matched layers, incompressible Navier-Stokes equations on tori, and Poisson equations on quarter-cylindrical surfaces. Comprehensive ablation studies demonstrate the necessity of each component, and comparisons with FNO, DeepONet, and Wavelet Neural Operator show AFDONet's superior performance on manifolds and datasets with sharp gradients.

Chapter 4 presents Adaptive Fourier Mamba Operators (AFMO) for handling irregular geometries and non-uniform meshes. We explain the integration of reproducing kernel theory with state-space models, showing how the Mamba architecture's selective state-space representation can be adapted to learn PDE solution operators on irregular domains. The theoretical foundations connecting Takenaka-Malmquist systems with state-space models are presented, along with algorithmic details for implementing AFMO efficiently. Experiments on problems with complex geometries, including flow past obstacles and diffusion on irregular domains, demonstrate AFMO's superior performance compared to methods requiring regular grids or interpolation.

Chapter 5 extends AFDONet to inverse problems in Banach spaces, introducing AFDONet-inv. We begin by discussing the limitations of Hilbert space formulations for inverse problems with sparse parameters and motivate the need for Banach space frameworks. The chapter develops the theoretical foundation for adaptive Fourier decomposition in reproducing kernel Banach spaces, extending key concepts from RKHS theory including reproducing properties, orthogonalization procedures, and approximation theorems. We present the AFDONet-inv architecture, which explicitly represents mappings from operator spaces to parameter spaces in Banach spaces, handling the duality structure appropriately. The architecture incorporates sparsity-promoting regularization through appropriate choice of Banach space norms, typically L^1 or bounded variation spaces. We derive convergence and stability results for AFDONet-inv, showing that the learned inverse operators are robust to noise in the observations. The chapter demonstrates superior performance on benchmark inverse problems including coefficient identification for elliptic PDEs with sparse or discontinuous parameters, source identification problems, and initial condition reconstruction. Comparisons with traditional variational methods, Bayesian inversion techniques, and Hilbert space operator learning approaches demonstrate the advantages of the Banach space formulation for problems with inherently sparse structures.

Chapter 6 explores automated neural operator design using large language models. We

begin by analyzing the challenges in manual neural operator design and motivating the need for automated approaches. The chapter presents the four-agent LLM pipeline in detail, describing the role and implementation of each agent: the Theorist that reasons about mathematical structures, the Programmer that translates theory to code, the Critic that evaluates designs, and the Refiner that iteratively improves architectures. We explain how the agents communicate through structured interfaces, how mathematical theories are represented and processed by the LLMs, and how the iterative refinement process is controlled. Extensive experiments demonstrate the framework's effectiveness across diverse PDE benchmarks including hyperbolic, parabolic, and elliptic equations on various domain geometries. We analyze the architectures designed by the framework, showing that they incorporate appropriate mathematical structures and often discover novel architectural patterns not present in existing literature. Ablation studies examine the contribution of each agent and the importance of theory-grounding. The chapter also discusses limitations of the current framework, including cases where LLMs struggle with highly specialized mathematical theories or produce architectures with implementation issues, and proposes directions for improvement.

Chapter 7 concludes the dissertation with a comprehensive summary of contributions, discussion of limitations, and directions for future research. We synthesize the key insights from the hybrid numerical methods, theory-guided neural operators, and automated design frameworks, discussing how they collectively advance the field of numerical PDE solutions. Limitations of each approach are honestly assessed, including computational costs, applicability to specific problem classes, and theoretical gaps that remain. We identify several promising directions for future research, including extension of the methods to time-dependent PDEs on evolving manifolds, development of uncertainty quantification frameworks for neural operators, integration with multi-fidelity modeling approaches, and application to frontier problems in computational science and engineering. The chapter concludes with reflections on the broader impact of this work in bridging traditional numerical analysis with modern machine learning, and the potential for these methods to accelerate scientific

discovery and engineering innovation.

CHAPTER II

MASSAGE-PASSING FINITE VOLUME METHOD FOR THE RICHARDS EQUATION

The spatiotemporal dynamics of root zone (e.g., top 1 m of soil) soil moisture from precipitation and surface soil moisture information can generally be modeled by the Richards equation Richards (1931), which captures irrigation, precipitation, evapotranspiration, runoff, and drainage dynamics in soil:

$$\begin{aligned} \partial_t \theta(\psi) + \nabla \cdot \mathbf{q} &= -S(\psi), \\ \mathbf{q} &= -K(\theta(\psi))\nabla(\psi + z). \end{aligned} \tag{2.0.1}$$

Here, ψ stands for pressure head (in, e.g., m), $\mathbf{q}$ represents the water flux (in, e.g., $\mathrm{m}^3/\mathrm{m}^2 \cdot \mathrm{s}$), S is the sink term associated with root water uptake (in, e.g., s^{-1}), θ denotes the soil moisture content (in, e.g., $\mathrm{m}^3/\mathrm{m}^3$), K is unsaturated hydraulic water conductivity (in, e.g., m/s), $t \in [0, T]$ denotes the time (in, e.g., s), and z corresponds to the vertical depth (in, e.g., m). The Richards equation is a nonlinear convection-diffusion equation Caputo & Stepanyants (2008), in which the convection term is due to gravity, and the diffusive term comes from Darcy's law Smith et al. (2002). For unsaturated flow, both θ and K are highly nonlinear functions of pressure head ψ and soil properties, making Equation (2.0.1) challenging to solve numerically. Specifically, $\theta(\psi)$ and $K(\psi)$ (or $K(\theta)$, depending on the model) are commonly referred to as the water retention curve (WRC) and hydraulic conductivity function (HCF), respectively. Several of the most widely used empirical models for WRC and HCF are summarized in Table 1.

Due to the highly nonlinear nature of WRC and HCF, analytical solutions to the Richards equation do not exist in general Farthing & Ogden (2017). Thus, the Richards equation is

Table 1: Some of the widely used HCF and WRC models. In these models, A, γ, α, β, n, θ_s, and θ_r are soil-specific parameters and have been tabulated for major soil types.

Model	HCF ($K(\psi)$ or $K(\theta)$)	WRC ($\theta(\psi)$)
Haverkamp et al. (1977)	$K_s \frac{A}{A+\lvert\psi\rvert^{\gamma}}$	$\theta_r + \frac{\alpha(\theta_s-\theta_r)}{\alpha+\lvert\psi\rvert^{\beta}}$
Mualem (1976); Van Genuchten (1980)	$K_s\sqrt{\frac{\theta-\theta_r}{\theta_s-\theta_r}}\left\{1-\left[1-\left(\frac{\theta-\theta_r}{\theta_s-\theta_r}\right)^{\frac{l}{l-1}}\right]^{\frac{l-1}{l}}\right\}^2$	$\theta_r + \frac{\theta_s-\theta_r}{\left[1+(\alpha\lvert\psi\rvert)^n\right]^{\frac{n-1}{n}}}$
Gardner (1958)	$K_s e^{\alpha\psi}$	$\theta_r + (\theta_s-\theta_r)e^{\alpha\psi}$

typically solved numerically in some discretized form. Consider the discretized version of Equation (2.0.1), whose control volume $V \subset \mathbb{R}^d$ $(d = 1, 2, 3)$ is discretized into N small cells $V_1, \ldots, V_N$. Using implicit Euler method on the time domain with a time step size of Δt, the discretized Richards equation at time step $m = 0, 1, \ldots, \lceil \frac{T}{\Delta t} \rceil - 1$ can be expressed as:

$$\begin{cases} \theta(\psi_i^{m+1}) - \theta(\psi_i^m) - \Delta t \nabla \cdot \left[K\left(\theta(\psi_i^{m+1})\right) \nabla\left(\psi_i^{m+1} + z\right) \right] + \Delta t S(\psi_i^{m+1}) = 0, \\ \text{Dirichlet boundary condition: } \psi_j(\cdot) = 0 \quad \text{for all } V_j \subset \partial V, \\ \text{Initial condition: } \psi(0, \cdot) = \psi_0(\cdot), \end{cases} \tag{2.0.2}$$

where ψ_i^m is the pressure head in cell V_i and time step m, and $\psi_0(\cdot)$ denotes the initial condition at $t = 0$.

The performance of a numerical PDE solver depends theoretically on the well-posedness of the PDE Sizikov et al. (2011), which is an essential property that certifies the accuracy and reliability of numerical solutions to the PDE. A PDE is said to be well-posed if its weak solution exists, is unique, and depends continuously on the problem's initial conditions Sizikov et al. (2011); Evans (2010). Here, we consider an FVM discretization with a discrete space $Q_h \subset L^2(V)$ of piecewise constants, where h denotes the maximum dimension of any cell in its mesh. With this, we define the space of piecewise constant functions on the set of meshes $\mathcal{T}_h = \{V_1, V_2, \ldots, V_N\}$ as $Q_h(V) = \{v \in L^2(V) : v|_{V_i} \text{ is constant for all } V_i \in \mathcal{T}_h\}$. Then, we introduce the discrete gradient operator Hyman & Shashkov (1997), GRAD_h, which maps a cell-based function in Q_h to a face-based function that approximates the gradient.

Note that ψ_i^m in Equation (2.0.2) denotes the pressure head in cell V_i and time step m, which is the value of ψ^m in the cell V_i. To study the pressure head solution in function space $Q_h(V)$, we focus on ψ^m rather than ψ_i^m. With this, the discrete solution for the FVM-discretized Richards equation can be defined as follows:

Definition 2.0.1 *Given $\psi^m \in Q_h$, if for any $v \in Q_h$,*

$$\left\langle \theta(\psi^{m+1}) - \theta(\psi^m), v \right\rangle_V \tag{2.0.3}$$

$$+ \Delta t \Big\langle K(\theta(\psi^{m+1}))\,\mathrm{GRAD}_h(\psi^{m+1} + z), \mathrm{GRAD}_h(v) \Big\rangle_{\mathcal{E}_h} \tag{2.0.4}$$

$$+ \Delta t \left\langle S(\psi^{m+1}), v \right\rangle_V = 0 \tag{2.0.5}$$

holds, where $\mathcal{E}_h$ denotes the set of all faces that make up the mesh $\mathcal{T}_h$, then ψ^{m+1} is a discrete solution of the FVM-discretized Richards equation.

Following Definition 2.0.1, for the discrete function space Q_h, an inner product over a cell V_i is defined for piecewise constant functions $f, g \in Q_h$ as $\langle f, g\rangle_{V_i} \coloneqq \int_{V_i} fg\,\mathrm{d}V$. In this case, by denoting f_i and g_i as the function values of f and g respectively on V_i (i.e., $f_i = f|_{V_i}$ and $g_i = g|_{V_i}$, both of which are constants), we have $\int_{V_i} fg\,\mathrm{d}V = f_i g_i \mathrm{vol}(V_i)$. The global inner product over the entire domain V is then $\langle f, g\rangle_V \coloneqq \sum_{i=1}^N \langle f, g\rangle_{V_i}$. We remark that the existence and uniqueness of the weak solution of the Richards equation have been rigorously established and carefully studied Merz & Rybka (2010); Misiats & Lipnikov (2013); Abdellatif et al. (2018), setting up the theoretical foundation for developing an efficient solution algorithm to solve the discretized Richards equation numerically.

2.1 Adaptive fixed-point iteration scheme of Discretized Richards Equation

In this section, we will formally introduce the adaptive fixed-point iteration scheme formulation of the FVM-discretized Richards equation. We will also derive sufficient conditions for parameter τ to ensure convergence. We will also analyze the convergence behavior of the resulting sequence of solutions $\{\psi_i^{m+1,s}\}_s$, where s is the iteration count ($s = 1, 2, \ldots, S$).

2.1.1 Adaptive fixed-point iteration scheme for the Richards Equation

To discretize the Richards equation via FVM, we first integrate both sides of Equation (2.0.1) over V:

$$\int_V \left[\partial_t \theta(\psi) + S(\psi)\right] \mathrm{d}V = \int_V \nabla \cdot \left[K(\theta)\nabla(\psi + z)\right] \mathrm{d}V. \tag{2.1.1}$$

Next, we apply the divergence theorem to Equation (2.1.1), which converts the volume integral on the RHS into a surface integral:

$$\left[\partial_t \theta(\hat{\psi}) + S(\hat{\psi})\right]_{\hat{\psi} \in V} \mathrm{vol}(V) = \oint_{S_V} K(\theta)\nabla(\psi + z) \cdot \mathbf{n} \, \mathrm{d}S_V, \tag{2.1.2}$$

where $\mathrm{vol}(V)$ is the volume of V, S_V is the surface of V and $\mathbf{n}$ is the outward pointing unit normal to the boundary ∂V. The common surface shared by cell V_i and cell V_j is denoted as $\omega_{i,j}$. With this, we can rewrite the operator $K(\cdot)\nabla(\cdot)$ and the outward pointing unit normal vector $\mathbf{n}$ on $\omega_{i,j}$ as $\left[K(\cdot)\nabla(\cdot)\right]_{\omega_{i,j}}$ and $\mathbf{n}_{\omega_{i,j}}$, respectively. After FVM discretization, we obtain the discretized version of Equation (2.1.2) as:

$$\partial_t \theta_i \mathrm{vol}(V_i) + S(\psi_i)\mathrm{vol}(V_i) = \sum_{j \in \mathcal{N}_i} \left[K(\theta)\nabla(\psi + z)\right]_{\omega_{i,j}} \cdot \mathbf{n}_{\omega_{i,j}} A_{\omega_{i,j}} \qquad \forall i = 1, \ldots, N, \tag{2.1.3}$$

where $\partial_t \theta_i$ refers to the time derivative $\partial_t \theta(\psi_i)$ in cell V_i, $\mathcal{N}_i$ denotes the index set of all the neighboring cells sharing a common surface with V_i, and $A_{\omega_{i,j}}$ is the area of surface $\omega_{i,j}$.

In static fixed-point iteration scheme, for each cell V_i and at each time step $m+1$, one would add the term $\frac{1}{\tau}(\psi_i^{m+1,s+1} - \psi_i^{m+1,s})$ to either side of Equation (2.1.3), so that the Richards equation can be solved in an iterative manner. The fixed-point pressure head solution of this iterative procedure is denoted as ψ_i^m. Since τ is a static constant, a trial-and-error procedure is typically required to obtain an appropriate τ value that avoids convergence issues. Not only is this search procedure tedious to implement, the solutions obtained are also less accurate most of the time as we will show in Section 2.3.1. Thus, inspired by previous works Amrein (2019); Zhu et al. (2019), we propose an adaptive fixed-point iteration scheme that replaces the static τ with $\tau_i^{m+1,s}$, which adjusts itself for each specific discretized cell,

time step, and iteration count. We then introduce the term $\frac{1}{\tau_i^{m+1,s}}(\psi_i^{m+1,s+1} - \psi_i^{m+1,s})$ to the LHS of Equation (2.1.3), which leads to:

$$\begin{aligned}\psi_i^{m+1,s+1} =& \psi_i^{m+1,s} + \tau_i^{m+1,s} \sum_{j \in \mathcal{N}_i} \left[K(\theta)\nabla(\psi + z)\right]_{\omega_{i,j}}^{m+1,s} \cdot \mathbf{n}_{\omega_{i,j}} A_{\omega_{i,j}} \\ & - \tau_i^{m+1,s} \left[\partial_t \theta_i^{m+1,s} + S(\psi_i^{m+1,s})\right] \mathrm{vol}(V_i),\end{aligned} \tag{2.1.4}$$

By discretizing $\partial_t \theta_i^{m+1,s}$ using implicit Euler scheme as $\frac{\theta(\psi_i^{m+1,s}) - \theta(\psi_i^m)}{\Delta t}$, we can obtain the adaptive fixed-point iteration scheme of the FVM-discretized Richards equation:

$$\begin{aligned}\psi_i^{m+1,s+1} = & \psi_i^{m+1,s} + \tau_i^{m+1,s} \sum_{j \in \mathcal{N}_i} K_{\omega_{i,j}}^{m+1,s} \frac{(\psi + z)_j^{m+1,s} - (\psi + z)_i^{m+1,s}}{\mathrm{dist}(V_j, V_i)} \mathbf{e} \cdot \mathbf{n}_{\omega_{i,j}} A_{\omega_{i,j}} \\ & - \tau_i^{m+1,s} \left[\frac{\theta(\psi_i^{m+1,s}) - \theta(\psi_i^m)}{\Delta t} + S(\psi_i^{m+1,s})\right] \mathrm{vol}(V_i),\end{aligned} \tag{2.1.5}$$

where $\mathbf{e} = (1, 1, 1)$ for the standard 3-D Cartesian coordinate system, and $\mathrm{dist}(\cdot, \cdot)$ represents the Euclidean distance function.

2.1.2 Choice of Adaptive Linearization Parameter

In adaptive fixed-point iteration scheme, we observe that $\tau_i^{m+1,s}$ needs to be sufficiently small because otherwise, the RHS of Equation (2.1.5) could approach infinity, which affects the convergence of the scheme. To prevent $\frac{1}{\tau_i^{m+1,s}}$ from being too large, we impose a user-specified global upper bound τ_0:

$$\tau_i^{m+1,s} \leq \tau_0.$$

In addition, the choice of $\tau_i^{m+1,s}$ can impact the accuracy of solutions. In other words, the term $\left|\frac{\psi_i^{m+1,s+1} - \psi_i^{m+1,s}}{\psi_i^{m+1,s}}\right|$ should be no greater than a prespecified tolerance ρ. Thus, we have:

$$\left|\frac{\psi_i^{m+1,s+1} - \psi_i^{m+1,s}}{\psi_i^{m+1,s}}\right| = \frac{\tau_i^{m+1,s} |g_i^{m+1,s}|}{|\psi_i^{m+1,s}|} \leq \rho \qquad \forall s = 1, \ldots, S,$$

where S is the user-specified total number of iterations for convergence, ρ should be no less

than the overall tolerance of convergence ϵ (to be discussed in Section 2.1.3), and:

$$g_i^{m+1,s} = \sum_{j \in \mathcal{N}_i} K_{\omega_{i,j}}^{m+1,s} \frac{(\psi + z)_j^{m+1,s} - (\psi + z)_i^{m+1,s}}{\text{dist}(V_j, V_i)} \mathbf{e} \cdot \mathbf{n}_{\omega_{i,j}} A_{\omega_{i,j}} - \frac{\theta(\psi_i^{m+1,s}) - \theta(\psi_i^m)}{\Delta t} \text{vol}(V_i) - S(\psi_i^{m+1,s}) \text{vol}(V_i).$$

This implies that:

$$\tau_i^{m+1,s} \leq \frac{\rho |\psi_i^{m+1,s}|}{(1+\rho)|g_i^{m+1,s}|} \qquad \forall s = 1, \ldots, S, \tag{2.1.6}$$

whose RHS can be explicitly determined from the results of the previous iteration. Note that, in actual implementation, we select $\tau_i^{m+1,s}$ based on:

$$\tau_i^{m+1,s} = \min \left\{ \tau_0, \frac{\rho |\psi_i^{m+1,s}|}{(1+\rho)|g_i^{m+1,s}|} \right\} \qquad \forall s = 1, \ldots, S. \tag{2.1.7}$$

Meanwhile, we can monitor the sensitivity of solutions obtained by our adaptive fixed-point iteration scheme and make sure that the solutions do not change drastically with respect to small perturbations. To achieve this, following Zarba (1988) and Celia & Zarba (1988), we explicitly write down Equation (2.1.5) for all discretized cells in the form of a matrix equation:

$$\mathbf{A}\mathbf{x}^{m+1,s} = \mathbf{b}, \tag{2.1.8}$$

where the ith element of vector $\mathbf{x}^{m+1,s}$ is $x_i^{m+1,s} = \psi_i^{m+1,s+1} - \psi_i^{m+1,s}$, which corresponds to cell V_i. Here, it is worth mentioning that Equation (2.1.8) is not used for solving Equations (2.1.5) as it is an explicit numerical scheme. Rather, it is used for analyzing the properties of the scheme after $x_i^{m+1,s}$ solutions are obtained by solving Equation (2.1.5). For example, to evaluate the choice of $\tau_i^{m+1,s}$, we can calculate the condition number of $\mathbf{A}$ based on the solutions obtained from the chosen $\tau_i^{m+1,s}$. If the condition number is larger than a user-specified threshold, we will update τ_0 in Equation (2.1.7) so that the condition number drops below the threshold. For 1-D problems, Zarba (1988) showed that $\mathbf{A}$ is a $N \times N$ asymmetric tridiagonal matrix. In this case, the condition number of $\mathbf{A}$ can be determined by calculating its eigenvalues. On the other hand, for 2-D and 3-D problems, $\mathbf{A}$ is a rectangular matrix, so that singular value decomposition will be used to determine its condition number.

2.1.3 Convergence of Adaptive Fixed-Point Iteration Scheme

We now study the convergence behavior of our adaptive fixed-point iteration scheme, which is formalized in Theorem 2.1.1. Recall that functions ψ^m and $\psi^{m+1,s}$ are considered to study the convergence. To show this, the idea is to leverage Definition 2.0.1 and find $\psi^{m+1,s+1} \in Q_h(V)$ given ψ^m, $\psi^{m+1,s} \in Q_h(V)$ such that:

$$\begin{aligned}\left\langle \theta(\psi^{m+1,s+1}) - \theta(\psi^m), v \right\rangle_V + \frac{\Delta t}{\tau^{m+1,s}} \left\langle \psi^{m+1,s+1} - \psi^{m+1,s}, v \right\rangle_V + \left\langle S(\psi^{m+1,s+1}), v \right\rangle_V \\ = -\Delta t \left\langle K\left(\theta(\psi^{m+1})\right) \mathrm{GRAD}_h(\psi^{m+1,s+1} + z), \mathrm{GRAD}_h(v) \right\rangle_{\mathcal{E}_h}\end{aligned} \tag{2.1.9}$$

holds for any $v \in Q_h(V)$. We remark that, unlike previous proofs (e.g., Amrein (2019)) that are based on several restrictive assumptions, our convergence proof follows a different approach that is intuitive and flexible, as it does not involve any additional assumptions other than the properties listed below.

Theorem 2.1.1 *The sequence $\{\psi^{m+1,s}\}_s$ converges to a unique solution $\psi^{m+1} \in Q_h(V)$ for $m = 0, 1, \ldots, \lceil \frac{T}{\Delta t} \rceil - 1$.*

Proof. First, we state two key properties used in the proof:

1. The Cauchy-Schwarz inequality holds for the discrete L^2 inner product: for any $u, w \in Q_h$, we have $\langle u, w \rangle_V \leq \|u\|_{L^2} \|w\|_{L^2}$.

2. $\dot{\theta}(\psi) = \frac{\mathrm{d}\theta}{\mathrm{d}\psi}|_{\psi^{m+1,s}} \geq c_0 > 0$, which is valid in most WRC models (see Table 1). Similarly, $\dot{S}(\psi) = \frac{\mathrm{d}S}{\mathrm{d}\psi}|_{\psi^{m+1,s}} \geq 0$ in the region between the start and optimal root water extraction.

First, we subtract Equation (2.0.3) from Equation (2.1.9) to obtain the error equation. Let $e^s \coloneqq \psi^{m+1,s} - \psi^{m+1}$, we have:

$$\begin{aligned}&\left\langle \theta(\psi^{m+1,s+1}) - \theta(\psi^{m+1}), v \right\rangle_V + \frac{\Delta t}{\tau^{m+1,s}} \left\langle e^{s+1} - e^s, v \right\rangle_V \\ &+ \left\langle S(\psi^{m+1,s+1}) - S(\psi^{m+1}), v \right\rangle_V = -\Delta t \left\langle K(\cdot) \mathrm{GRAD}_h(e^{s+1}), \mathrm{GRAD}_h(v) \right\rangle_{\mathcal{E}_h}.\end{aligned} \tag{2.1.10}$$

Let the test function $v = e^{s+1} = \psi^{m+1,s+1} - \psi^{m+1}$. This is a valid choice as $e^{s+1} \in Q_h$. By applying the mean value theorem to the θ and S terms, and using Observation 2, we have:

$$\begin{aligned} \left\langle \theta(\psi^{m+1,s+1}) - \theta(\psi^{m+1}), e^{s+1} \right\rangle_V &= \left\langle \dot{\theta}(\xi_\theta) e^{s+1}, e^{s+1} \right\rangle_V \geq c_0 \|e^{s+1}\|_{L^2}^2, \\ \left\langle S(\psi^{m+1,s+1}) - S(\psi^{m+1}), e^{s+1} \right\rangle_V &= \left\langle \dot{S}(\xi_S) e^{s+1}, e^{s+1} \right\rangle_V \geq 0 \end{aligned} \tag{2.1.11}$$

for some ξ_θ, ξ_S between $\psi^{m+1,s+1}$ and ψ^{m+1}. The flux term on the RHS of Equation (2.1.10) is also non-negative:

$$-\Delta t \left\langle K(\cdot)\, \mathrm{GRAD}_h(e^{s+1}), \mathrm{GRAD}_h(e^{s+1}) \right\rangle_{\mathcal{E}_h} = -\Delta t \|e^{s+1}\|_h^2 \leq 0, \tag{2.1.12}$$

where $\|\cdot\|_h$ is the discrete energy semi-norm. Substituting Equation (2.1.11) and Equation (2.1.12) into Equation (2.1.10) gives:

$$c_0 \|e^{s+1}\|_{L^2}^2 + 0 + \frac{\Delta t}{\tau^{m+1,s}} \left\langle e^{s+1} - e^s, e^{s+1} \right\rangle_V \leq 0 \tag{2.1.13}$$

By applying Observation 1, Equation (2.1.13) leads to:

$$c_0 \|e^{s+1}\|_{L^2}^2 + \frac{\Delta t}{\tau^{m+1,s}} \left(\|e^{s+1}\|_{L^2}^2 - \langle e^s, e^{s+1} \rangle_V \right) \leq 0. \tag{2.1.14}$$

Then, we have:

$$\left(c_0 + \frac{\Delta t}{\tau^{m+1,s}} \right) \|e^{s+1}\|_{L^2}^2 \leq \frac{\Delta t}{\tau^{m+1,s}} \langle e^s, e^{s+1} \rangle_V \leq \frac{\Delta t}{\tau^{m+1,s}} \|e^s\|_{L^2} \|e^{s+1}\|_{L^2}. \tag{2.1.15}$$

If $e^{s+1} = 0$, we complete the proof. If $e^{s+1} \neq 0$, we can divide Equation (2.1.15) by $\|e^{s+1}\|_{L^2}$:

$$\left(c_0 + \frac{\Delta t}{\tau^{m+1,s}} \right) \|e^{s+1}\|_{L^2} \leq \frac{\Delta t}{\tau^{m+1,s}} \|e^s\|_{L^2}, \tag{2.1.16}$$

which yields the contraction:

$$\|e^{s+1}\|_{L^2} \leq \underbrace{\left(\frac{\frac{\Delta t}{\tau^{m+1,s}}}{c_0 + \frac{\Delta t}{\tau^{m+1,s}}} \right)}_{=:\gamma_s} \|e^s\|_{L^2}. \tag{2.1.17}$$

Since $c_0 > 0$, the contraction factor γ_s is strictly less than 1. Therefore, the sequence is a contraction mapping on the discrete space $Q_h(V)$ equipped with the L^2 norm. By the Banach fixed-point theorem, the sequence $\{\psi^{m+1,s}\}$ converges to a unique solution $\psi^{m+1} \in Q_h(V)$. This completes the proof. ■

2.2 Message Passing Finite Volume Method (MP-FVM)

Once the adaptive fixed-point iteration scheme for the FVM-discretized Richards equation is established, we incorporate it in our MP-FVM algorithm to enhance the solver accuracy and ability to retain underlying physics (e.g., mass conservation). As discussed previously, the message passing neural PDE solver proposed by Brandstetter et al. (2022) comprises three main components: an encoder, a processor, and a decoder. The message passing mechanism is implemented within the processor that operates in the latent space. However, it has not been extended to discretized PDEs. In this work, we introduce the message passing mechanism for the discretized Richards equation by defining a latent variable $\mu_i^{m,s}$ as the processor. Therefore, by leveraging our adaptive fixed-point iteration scheme, we can now solve the latent variable iteratively to enhance the convergence and numerical stability of the message passing mechanism. Specifically, this integrative algorithm, MP-FVM, adopts one neural network (encoder) $\hat{f}_{\text{NN}}$ to learn the map $\psi_i^{m,s} \mapsto \mu_i^{m,s}$ and another neural network (decoder) $\hat{f}_{\text{NN}}^{-1}$ to learn the inverse map $\mu_i^{m,s} \mapsto \psi_i^{m,s}$. Overall, our MP-FVM algorithm involves offline training (dataset preparation and encoder-decoder training) and solution (message passing) process, which are summarized in the flowchart of Figure 1.

2.2.1 Dataset Preparation and Data Augmentation

The dataset used to train the encoder and decoder neural networks comes from two different sources/solvers. Specifically, for each cell V_i and time step m, we approximate the latent variable solution $\mu_i^{m,S}$ from a finite difference solver (e.g., Ireson et al. (2023)). Here, S is the user-specified total iteration number. The corresponding $\psi_i^{m,S}$ solution is obtained separately from the fixed-point iteration scheme of Equation (2.1.5) using a static parameter τ. The resulting set of solution pairs, $\left\{\left(\psi_i^{m,S}, \mu_i^{m,S}\right)\right\}_{i,m}$, form a set of original "reference solutions". In actual implementation, we obtain multiple sets of original reference solutions by selecting multiple total iteration numbers $(S_1, \ldots, S_p)$ and/or fixed-point parameters $(\tau_1, \ldots, \tau_r)$ that

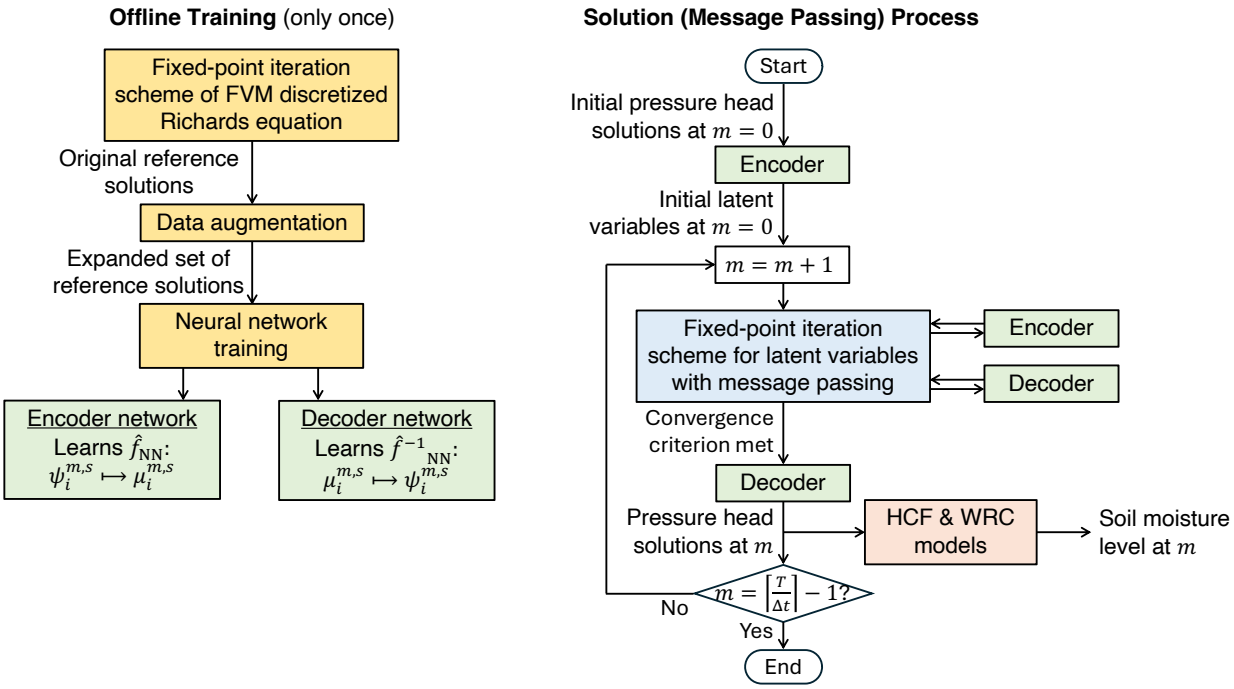


Figure 1: Flowchart of our proposed algorithm to solve the FVM-discretized Richards equation using a message passing mechanism.

cover their ranges expected during the actual solution process. These sets of original reference solutions, which are $\left\{\left(\psi_i^{m,S_1}, \mu_i^{m,S_1}\right)|_{\tau_1}\right\}_{i,m}, \ldots, \left\{\left(\psi_i^{m,S_p}, \mu_i^{m,S_p}\right)|_{\tau_r}\right\}_{i,m}$, are combined to form a larger set to perform data augmentation.

Next, to apply data augmentation, we introduce Gaussian noise $Z_q \sim \mathcal{N}(0, \sigma_q^2)$ with different variances $\sigma_1^2, \ldots, \sigma_Q^2$ to each and every element in the reference solution set obtained previously. After data augmentation, the resulting expanded set of reference solutions, $\left\{(\psi_i^{m,S_1} + Z_p, \mu_i^{m,S_1} + Z_q)|_{\tau_1}\right\}_{i,m,q}, \ldots, \left\{(\psi_i^{m,S_p} + Z_q, \mu_i^{m,S_p} + Z_q)|_{\tau_r}\right\}_{i,m,q}$, is denoted as $\mathcal{S}$ and will be used for neural network training. This data augmentation step not only increases the size of the training dataset, but also reflects the characteristics of actual soil sensing data, which are subject to various measurement uncertainties. Furthermore, In Section 2.3.1, we will show that introducing Gaussian noise can greatly reduce the biases of reference solutions and enhance generalization performance Da Silva & Adeodato (2011), thereby significantly improving the accuracy of numerical solutions.

2.2.2 Neural Network Training

A neural network is capable of approximating any function provided that it contains enough neurons Hornik (1991); Pinkus (1999). In the actual implementation, depending on the problem settings, the desired choices of optimal optimizer, number of hidden layers, and activation functions can vary. Based on our extensive research and hyperparameter tuning, we find that a simple three-layer neural network with 256 neurons in each layer achieves the best performance for most 1-D through 3-D problems compared to other more complex neural network architectures (e.g., LSTM). Also, we find that stochastic gradient decent (SGD) optimizer often outperforms others (e.g., Adam or RMSProp). The learning rate is set to be 0.001. This simple neural network structure makes our MP-FVM algorithm training much less computationally expensive compared to state-of-the-art neural PDE solvers (e.g., Lu et al. (2020b); Brandstetter et al. (2022)).

In terms of loss function design, we note that the solution of the Richards equation at a given time step depends on the pressure head solution at the initial condition and previous time steps. A small perturbation in these solutions can lead to slow convergence or inaccurate solutions at the final time step. To account for this, we introduce Sobolev training Czarnecki et al. (2017b) for both neural networks $\hat{f}_{\text{NN}}$ and $\hat{f}_{\text{NN}}^{-1}$ to ensure compatibility and stability in the same solution space. We implement Sobolev training by adding a Sobolev regularization term to the standard Mean Squared Error (MSE) in the loss functions for $\hat{f}_{\text{NN}}$ and $\hat{f}_{\text{NN}}^{-1}$:

$$\mathcal{L}_{\hat{f}_{\text{NN}}} = \underbrace{\frac{1}{|\mathcal{S}|}\sum_{(\psi,\mu)\in\mathcal{S}}\left(\mu - \hat{f}_{\text{NN}}(\psi)\right)^2}_{\text{MSE term}} + \lambda_{\hat{f}_{\text{NN}}} \cdot \underbrace{\frac{1}{|\mathcal{S}|}\sum_{(\psi,\mu)\in\mathcal{S}}\left(\left\|\nabla\left(\mu - \hat{f}_{\text{NN}}(\psi)\right)\right\|_{L^2}^2\right)}_{\text{Sobolev regularization term}}, \tag{2.2.1}$$

and

$$\mathcal{L}_{\hat{f}_{\text{NN}}^{-1}} = \underbrace{\frac{1}{|\mathcal{S}|}\sum_{(\psi,\mu)\in\mathcal{S}}\left(\psi - \hat{f}_{\text{NN}}^{-1}(\mu)\right)^2}_{\text{MSE term}} + \lambda_{\hat{f}_{\text{NN}}^{-1}} \cdot \underbrace{\frac{1}{|\mathcal{S}|}\sum_{(\psi,\mu)\in\mathcal{S}}\left(\left\|\nabla\left(\psi - \hat{f}_{\text{NN}}^{-1}(\mu)\right)\right\|_{L^2}^2\right)}_{\text{Sobolev regularization term}}, \tag{2.2.2}$$

where $\lambda_{\hat{f}_{\text{NN}}}$ and $\lambda_{\hat{f}_{\text{NN}}^{-1}}$ are user-specified regularization parameters for the neural networks $\lambda_{\hat{f}_{\text{NN}}^{-1}}$ and $\hat{f}_{\text{NN}}^{-1}$, respectively. Here, we use the Leaky ReLU activation function, as it has been shown

that there exists a single hidden-layer neural network with ReLU (or Leaky ReLU) activation function that can approximate any function in a Sobolev space Czarnecki et al. (2017b). Overall, this combined loss function ensures that the model not only produces accurate predictions but also generates smooth and regular outputs by matching the gradients of the true function.

2.2.3 Message Passing Process

When neural network training is complete, the trained encoder $\hat{f}_{\text{NN}}$ and decoder $\hat{f}_{\text{NN}}^{-1}$ can then be incorporated into Equation (2.1.5) to derive the following fixed-point iterative scheme for the latent variables with message passing mechanism:

$$\mu_i^{m+1,s+1} = \mu_i^{m+1,s} + \tau_i^{m+1,s} \sum_{j \in \mathcal{N}_i} K_{\omega_{i,j}}^{m+1,s} \mathbf{e} \cdot \mathbf{n}_{\omega_{i,j}} \frac{\mu_j^{m+1,s} - \mu_i^{m+1,s}}{\text{dist}(V_j, V_i)} A_{\omega_{i,j}} + \hat{f}_{\text{NN}}(J), \tag{2.2.3}$$

where $J = \tau_i^{m+1,s} \sum_{j \in \mathcal{N}_i} K_{\omega_{i,j}}^{m+1,s} \mathbf{e} \cdot \mathbf{n}_{\omega_{i,j}} \frac{z_j - z_i}{\text{dist}(V_j, V_i)} A_{\omega_{i,j}} - \tau_i^{m+1,s} \left(\frac{\theta_i^{m+1,s} - \theta_i^m}{\Delta t} + S(\psi_i^{m+1,s}) \right) \text{vol}(V_i)$. To solve Equation (2.2.3), we will adopt a similar strategy as in Equation (2.1.7) to adaptively select the linearization parameter $\tau_i^{m+1,s}$. To start the message passing process, we obtain the initial pressure head solutions in the control volume at $m = 0$ from the initial and boundary conditions. These initial pressure head solutions can be mapped to the latent space via trained encoder network $\hat{f}_{\text{NN}}$. Next, for each new time step $m + 1$, the latent variable for every cell can be iteratively solved by Equation (2.2.3) by utilizing the trained neural networks $\hat{f}_{\text{NN}}$ and $\hat{f}_{\text{NN}}^{-1}$. Note that the iterative usage of $\hat{f}_{\text{NN}}^{-1}$ is implicitly implied in the MP-FVM algorithm, as the term J in Equation (2.2.3) contains $\psi_i^{m+1,s}$ that must be evaluated by applying $\hat{f}_{\text{NN}}^{-1}$ on latent variable $\mu_i^{m+1,s}$. Also, it is worth mentioning that, since ψ and J have different scales, in actual implementation, in addition to $\hat{f}_{\text{NN}}$ for learning $\psi_i^{m,s} \to \mu_i^{m,s}$, we train another neural network named $\hat{f'}_{\text{NN}}$ for mapping J to the latent space in Equation (2.2.3). To monitor convergence of the iterative message passing process, we define the relative error RE_s as:

$$\text{RE}_s := \frac{||\mu^{m+1,s+1} - \mu^{m+1,s}||_{L^2}}{||\mu^{m+1,s+1}||_{L^2}}, \tag{2.2.4}$$

where $\mu^{m+1,s+1} = (\mu_1^{m+1,s+1}, \ldots, \mu_N^{m+1,s+1})^T$ and so on. Once RE_s is below a user-specified tolerance tol (typically in the order of 10^{-6}), we declare convergence of $\{\mu_i^{m+1,s}\}_s$ to μ_i^{m+1}. From there, one can determine the converged ψ_i^{m+1} using $\hat{f}_{\text{NN}}^{-1}$, followed by obtaining other physical quantities such as soil moisture content θ_i^{m+1} and $\mathbf{q}_i^{m+1}$ from the WRC and HCF models (Table 1) and Equation (2.0.1). The entire solution process then repeats itself in the next time step until $m = \lceil \frac{T}{\Delta t} \rceil - 1$.

Furthermore, it is worth mentioning that, when neural network training for a specific problem setting (e.g., boundary condition and initial condition) is complete, the trained neural networks can be saved as a pre-trained model. As we encounter a new problem setting, the pre-trained model provides a strong starting point that can be quickly refined with a small number of epochs (typically no more than 100) before it can be deployed to solve the new problem. The use of pre-trained model is a well-established technique in machine/deep learning for leveraging knowledge learned from (large) datasets, reducing the need for extensive training data and computation, and enabling faster deployment and improved performance in new tasks through fine-tuning.

2.2.4 Convergence of MP-FVM Algorithmn

The convergence of our MP-FVM algorithm, which features the sequence $\{\mu^{m+1,s}\}_s$, can be established by extending Theorem 2.1.1 and investigating the convergence behavior of stochastic gradient descent (SGD) for neural network realizations of $\hat{f}_{\text{NN}}$ and $\hat{f}_{\text{NN}}^{-1}$. Similar to Theorem 2.1.1, we consider functions $\{\mu^{m+1,s}\}_s$ and μ^{m+1} instead of their discretized variants.

Theorem 2.2.1 *The sequence $\{\mu^{m+1,s}\}_s$ converges to μ^{m+1} for $m = 0, 1, \ldots, \lceil \frac{T}{\Delta t} \rceil - 1$.*

To prove Theorem 2.2.1, we first need to introduce the following preliminary assumptions and results from Fontaine et al. (2021) and Berner et al. (2019).

1. The objective function f is L-smooth.

2. There exists a Polish probability space $(Z, \mathcal{Z}, \pi^Z)$ and $\eta \geq 0$ such that one of the following conditions holds:

   (a) There exists a function $H : \mathbb{R}^d \times Z \to \mathbb{R}^d$ such that for any $x \in \mathbb{R}^d$,

   $$\int_Z H(x,z) d\pi^Z(z) = \nabla f(x), \qquad \int_Z \|H(x,z) - \nabla f(x)\|_{L^2}^2 d\pi^Z(z) \leq \eta.$$

   (b) There exists a function $\tilde{f} : \mathbb{R}^d \times Z \to \mathbb{R}$ such that for all $z \in Z$, $\tilde{f}(\cdot, z) \in C^1(\mathbb{R}^d, \mathbb{R})$ is L-smooth. Furthermore, there exists $x^* \in \mathbb{R}^d$ such that, for any $x \in \mathbb{R}^d$,

   $$\int_Z \tilde{f}(x,z) d\pi^Z(z) = f(x), \ \int_Z \nabla \tilde{f}(x,z) d\pi^Z(z) = \nabla f(x), \ \int_Z \|\nabla \tilde{f}(x^*,z)\|_{L^2}^2 d\pi^Z(z) \leq \eta.$$

   In this case, we define $H = \nabla \tilde{f}$.

3. There exists $M \geq 0$ such that for any $x, y \in \mathbb{R}^d$,

$$\|\Sigma(x)^{1/2} - \Sigma(y)^{1/2}\|_{L^2} \leq M \|x - y\|_{L^2}.$$

4. One of the following conditions holds:

   (a) For Assumption 2(a): f is convex, *i.e.*, for any $x, y \in \mathbb{R}^d$,

   $$\langle \nabla f(x) - \nabla f(y), x - y \rangle \geq 0,$$

   and there exists a minimizer $x^* \in \arg\min_{x \in \mathbb{R}^d} f$.

   (b) For Assumption 2(b): For all $z \in Z$, $\tilde{f}(\cdot, z)$ is convex, and there exists a minimizer $x^* \in \arg\min_{x \in \mathbb{R}^d} f$.

Under Assumptions 1 and 2, we introduce the sequence $\{X_n\}_{n \in \mathbb{N}}$ starting from $X_0 \in \mathbb{R}^d$ corresponding to SGD with non-increasing step sizes for any $n \in \mathbb{N}$ by:

$$X_{n+1} = X_n - \gamma (n+1)^{-\alpha} H(X_n, Z_{n+1}),$$

where $\gamma > 0$, $\alpha \in [0, 1]$, and $\{Z_n\}_{n \in \mathbb{N}}$ is a sequence of independent random variables on a probability space $(\Omega, \mathcal{F}, P)$ valued in $(Z, \mathcal{Z})$ such that for any $n \in \mathbb{N}$, Z_n is distributed

according to π^Z. As Fontaine et al. (2021) pointed out, the solution of the following SDE is a continuous counterpart of $\{X_n\}_{n\in\mathbb{N}}$:

$$\mathrm{d}\mathbf{X_t} = -(\gamma+t)^{-\alpha}\nabla f(\mathbf{X_t})\,\mathrm{d}t + \gamma(\gamma+t)^{-2\alpha}\Sigma(\mathbf{X_t})^{1/2}\,\mathrm{d}\mathbf{B_t},$$

where $\gamma_\alpha = \gamma^{1/(1-\alpha)}$ and $(B_t)_{t\geq 0}$ is a d-dimensional Brownian motion.

Given these preliminaries, we now leverage two established results as lemmas:

Lemma 2.2.1 (Theorem 6 of Fontaine et al. (2021)) *Let $\alpha, \gamma \in (0,1)$, for $f \in C^2(\mathbb{R}^d, \mathbb{R})$, there exists $C \geq 0$ such that for any $T \geq 1$,*

$$\mathbb{E}[f(\mathbf{X_T})] - \min_{x\in\mathbb{R}^d} f \leq C\frac{(1+\log(T))^2}{T^{\alpha(1-\alpha)}}.$$

Lemma 2.2.2 (Equation 35 of Berner et al. (2019)) *Suppose $\tilde{f}$ with an at most polynomially growing derivative is the "true" function learned by the neural network. Let $\kappa > 0$ be the polynomial growth rate, there exists $D \geq 0$ such that*

$$\|\tilde{f}(x) - \tilde{f}(y)\|_{L^2} \leq D\left(1 + \|x\|_{L^2}^{\kappa+2} + \|y\|_{L^2}^{\kappa+2}\right)\|x-y\|_{L^2}$$

holds.

With Lemmas 2.2.1 and 2.2.2, we are now ready to give the proof of Theorem 2.2.1 which accounts for the convergence of SGD:

Proof. To start, we have:

$$\begin{aligned}
\|\mu^{m+1,s+1} - \mu^{m+1,s}\|_{L^2} &\leq \mathbb{E}\left[\|\hat{f}_{\mathrm{NN}}(\psi^{m+1,s+1}, \mathbf{X_T}) - \hat{f}_{\mathrm{NN}}(\psi^{m+1,s}, \mathbf{X_T})\|_{L^2}\right] \\
&\leq \mathbb{E}\left[\|\hat{f}_{\mathrm{NN}}(\psi^{m+1,s+1}, \mathbf{X_T}) - \tilde{f}(\psi^{m+1,s+1})\|_{L^2}\right] \\
&\quad + \|\tilde{f}(\psi^{m+1,s+1}) - \tilde{f}(\psi^{m+1,s})\|_{L^2} \\
&\quad + \mathbb{E}\left[\|\tilde{f}(\psi^{m+1,s}) - \hat{f}_{\mathrm{NN}}(\psi^{m+1,s}, \mathbf{X_T})\|_{L^2}\right],
\end{aligned}$$

where $\mathbf{X_T}$ is the weights of $\hat{f}_{\mathrm{NN}}$ optimized by SGD optimizer, whose process is assumed to be well-approximated by the SDE in Lemma 2.2.1, and $\tilde{f}$ is the true function learned by $\hat{f}_{\mathrm{NN}}$.

To bound the first and third terms, we define the objective function for the SGD process for a given input ψ as $f_\psi(x) = \|\hat{f}_{\text{NN}}(\psi, x) - \tilde{f}(\psi)\|_{L^2}$. We assume this function satisfies the conditions for Lemma 2.2.1. The terms we seek to bound are then precisely of the form $\mathbb{E}[f_\psi(\mathbf{X_T})]$. From Lemma 2.2.1, we have:

$$\mathbb{E}[f_\psi(\mathbf{X_T})] \leq C\frac{(1+\log(T))^2}{T^{\alpha(1-\alpha)}} + \min_{x\in\mathbb{R}^d} f_\psi(x).$$

We assume the network is a good approximator, such that for a given $\varepsilon > 0$ and for any relevant ψ, the minimum error satisfies $\min_{x\in\mathbb{R}^d} f_\psi(x) \leq \frac{\varepsilon}{6}$. When the network is trained for a sufficiently large T, we can ensure $C\frac{(1+\log(T))^2}{T^{\alpha(1-\alpha)}} \leq \frac{\varepsilon}{6}$. Thus, for both the first and third terms, which correspond to $\psi = \psi^{m+1,s+1}$ and $\psi = \psi^{m+1,s}$, we have the bound:

$$\mathbb{E}\left[\|\hat{f}_{\text{NN}}(\psi, \mathbf{X_T}) - \tilde{f}(\psi)\|_{L^2}\right] \leq \frac{\varepsilon}{6} + \frac{\varepsilon}{6} = \frac{\varepsilon}{3}.$$

Next, for the term $\|\tilde{f}(\psi^{m+1,s+1}) - \tilde{f}(\psi^{m+1,s})\|_{L^2}$, it can be bounded using Lemma 2.2.2 and the result $\|\psi^{m+1,s+1} - \psi^{m+1,s}\|_{L^2} \leq \frac{\varepsilon}{3D\left(1+\|\psi^{m+1,s+1}\|_{L^2}^{\kappa+2} + \|\psi^{m+1,s}\|_{L^2}^{\kappa+2}\right)}$ obtained from Theorem 2.1.1:

$$\begin{aligned}\|\tilde{f}(\psi^{m+1,s+1}) - \tilde{f}(\psi^{m+1,s})\|_{L^2} \leq D\left(1 + \|\psi^{m+1,s+1}\|_{L^2}^{\kappa+2} + \|\psi^{m+1,s}\|_{L^2}^{\kappa+2}\right) \\ \cdot \|\psi^{m+1,s+1} - \psi^{m+1,s}\|_{L^2} \leq \frac{\varepsilon}{3}.\end{aligned}$$

Therefore, it follows that

$$\|\mu^{m+1,s+1} - \mu^{m+1,s}\|_{L^2} \leq \frac{\varepsilon}{3} + \frac{\varepsilon}{3} + \frac{\varepsilon}{3} = \varepsilon,$$

which completes the proof. ■

2.3 Case Studies

Now that we have introduced the MP-FVM algorithm formulation for the Richards equation, in this section, we evaluate our MP-FVM framework on a series of 1-D through 3-D benchmark problems modified from the literature Celia et al. (1990); Gasiorowski & Kolerski

(2020); Tracy (2006); Berardi et al. (2018); Orouskhani et al. (2023). Specifically, we extensively study the 1-D benchmark problem of Celia et al. (1990) to demonstrate the need and benefits of different components employed in our MP-FVM algorithm, including adaptive fixed-point iteration scheme, encoder-decoder architecture and message passing mechanism, and Sobolev training. Also, using this problem as a benchmark, we demonstrate the accuracy of our solution algorithm with respect to state-of-the-art solvers. In the 1-D layered soil case study proposed by Berardi et al. (2018), we show that our MP-FVM algorithm is capable of handling discontinuities in soil properties and modeling the infiltration process through the interface of two different soils. In the 2-D case study adopted from Gasiorowski & Kolerski (2020), we show that our MP-FVM algorithm can better satisfy the mass balance embedded in the Richards equation. In the 3-D case study adopted from Tracy (2006) in which an analytical solution to the Richards equation exists, we show that our MP-FVM algorithm produces much more accurate solutions compared to conventional FVM solvers. Finally, we study a 3-D problem adopted from Orouskhani et al. (2023) featuring an actual center-pivot system and validate the accuracy and robustness of our MP-FVM algorithm in modeling real-world precipitation and irrigation scenarios for a long period of time.

2.3.1 A 1-D Benchmark Problem

Here, we study the 1-D benchmark problem over a 40 cm deep soil presented by Celia et al. (1990). The HCF and WRC adopt the model of Haverkamp et al. (1977) (see Table 1), whose parameters are listed in Table 2. The initial condition is given by $\psi(z, 0) = -61.5$ cm, whereas the two boundary conditions are $\psi(40 \text{ cm}, t) = -20.7$ cm, $\psi(0, t) = -61.5$ cm, respectively Haverkamp et al. (1977). This benchmark problem ignores the sink term.

Through this 1-D illustrative example, we will highlight the benefits of (a) adopting an adaptive fixed-point iteration scheme as opposed to standard the fixed-point iteration scheme, (b) implementing the MP-FVM algorithm as opposed to the conventional FVM method, and (c) integrating the adaptive fixed-point iteration scheme with encoder-decoder

Soil-specific Parameters	Values	Units
Saturated hydraulic conductivity, K_s	0.00944	cm/s
Saturated soil moisture content, θ_s	0.287	–
Residual soil moisture content, θ_r	0.075	–
α in Haverkamp's model	1.611×10^6	cm
A in Haverkamp's model	1.175×10^6	cm
β in Haverkamp's model	3.96	–
γ in Haverkamp's model	4.74	–
Total time, T	360	s

Table 2: soil-specific parameters and their values used in the 1-D case study of Celia et al. (1990) based on the empirical model developed by Haverkamp et al. (1977).

network and message passing mechanism in a holistic numerical framework.

The Need for Adaptive fixed-point iteration scheme

To illustrate how adaptive fixed-point iteration scheme improves convergence and accuracy of conventional fixed-point iteration schemes, we compare the pressure head solution profiles at $t = T = 360$ seconds obtained by different static fixed-point parameters after (a) $S = 500$ iterations and (b) tol $= 3.2 \times 10^{-5}$. We adopt a spatial grid containing 101 mesh points ($\Delta z = 0.4$ cm) and a temporal grid satisfying the Courant-Friedrichs-Lewy (CFL)-like condition, typically expressed as $\Delta t \leq \frac{\Delta z^2}{2K}$ De Moura & Kubrusly (2013). As shown in Figure 2, when using static fixed-point iteration scheme, the choice of parameter τ and the total number of iterations can impact the solution accuracy and algorithm stability significantly. For example, when the fixed-point parameter is too large (e.g., $\tau = 2$ for this problem), the stability of the static fixed-point iteration scheme can be adversely affected (as illustrated by the zigzag pressure head profile towards $z = 40$ cm). Another key observation

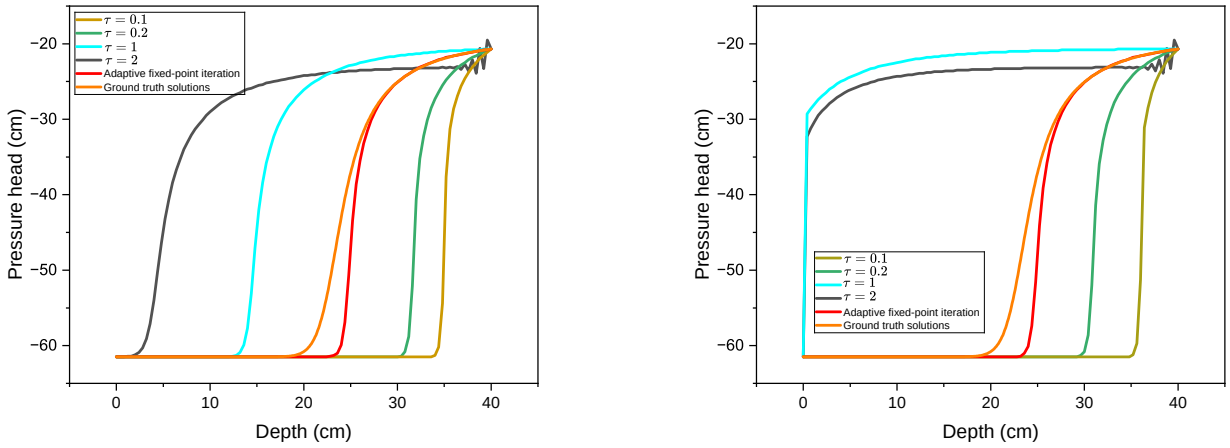


Figure 2: Comparison of pressure head solution profiles at $t = T = 360$ seconds under (a) $S = 500$ iterations and (b) tol $= 3.2 \times 10^{-5}$ for the 1-D benchmark problem Celia et al. (1990) using standard and adaptive fixed-point iteration schemes (Equation (2.1.5)). The solutions obtained from Celia et al. (1990) based on very fine space and time steps are marked as the ground truth solutions.

is that, increasing the total number of iterations sometimes deteriorates solution accuracy of static fixed-point iteration scheme. These observations pose practical challenges for using static fixed-point iteration scheme, especially when the ground truth solutions are absent, as identifying the optimal fixed-point parameter and total number of iterations that would yield accurate solutions will not be possible without referring to ground truth solutions. This motivates us to develop adaptive fixed-point iteration scheme as a robust and reliable numerical scheme that produces solutions that are close to ground truth solutions without trail-and-error parameter tuning. Also, it is worth noting that our adaptive fixed-point iteration scheme successfully bypasses the singularity issue as $\frac{1}{\tau_i^{m+1,s}}$ approaches to 0 and correctly calculates the pressure head solutions for $z \in [0, 20 \text{ cm}]$ where $\dot{\theta}(\psi)$ becomes small.

The Need for Encoder-Decoder Architecture

To generate the reference solutions, we consider a coarse spatial discretization containing 40 cells (i.e., grid size $\Delta z = 1$ cm) and solve for $T = 360$ seconds. The time step size Δt is determined using the CFL condition De Moura & Kubrusly (2013). A set of pressure head solutions ψ is obtained using the finite difference method that incorporates a modified Picard iteration scheme developed by Celia et al. (1990). Meanwhile, another set of pressure head solutions, which essentially becomes the latent variable dataset μ for neural network training, is obtained from the fixed-point iteration scheme of Equation (2.1.5) under 4 different static fixed-point parameter $\tau = 0.25$, 0.24, 0.23, 0.22 and 10 different total iteration counts $S = 1,000$, $2,000$, up to $10,000$.

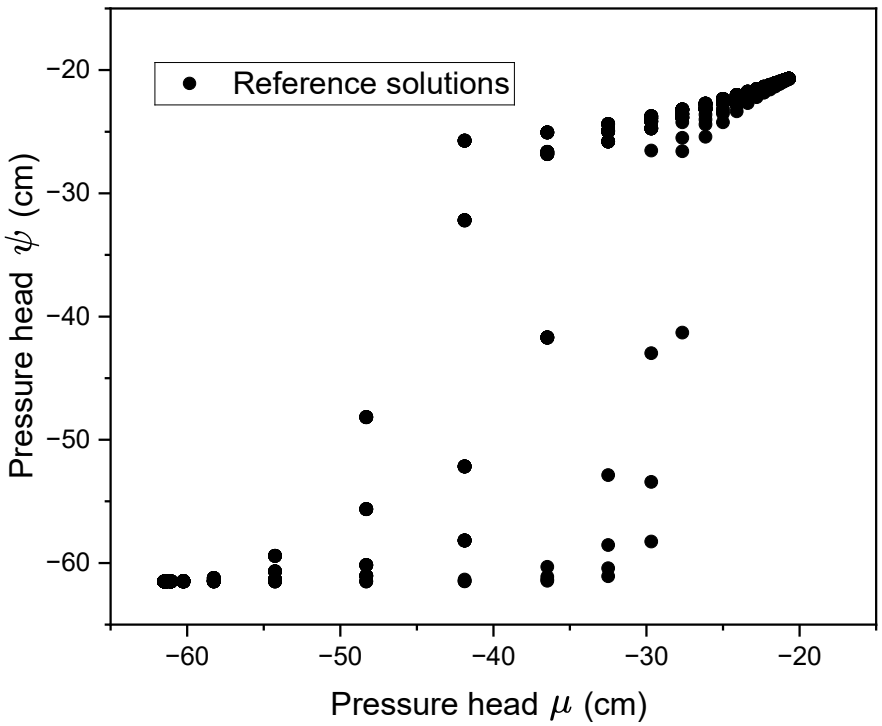


Figure 3: The relationships between 1640 pressure head solutions ψ and μ, which are obtained by two distinct approaches. The resulting nonlinearity present in these reference solutions highlights need for data-driven approach.

As mentioned earlier, reference solutions utilized to train the encoder $\hat{f}_{\text{NN}}$ and decoder

$\hat{f}_{\text{NN}}^{-1}$ come from two different sources. As shown in Figure 3, a highly nonlinear relationship between two sources of pressure head solutions is observed. This is mainly because pressure head solutions from different sources exhibit different sensitivities with respect to different choices of τ and S. Without knowing the ground truth solutions a priori, it is hard to determine which set of pressure head solutions is more accurate. This motivates us to adopt an encoder-decoder architecture to explicitly capture this nonlinear relationship, which encapsulates the sensitivity of solution with respect to different choices of τ and S.

Another motivation for adopting an encoder-decoder architecture in our numerical solver comes from the fact that different sources of pressure head solutions also exhibit different topological features. To see this, we use persistent homology Edelsbrunner & Morozov (2013) as a way to capture the multiscale topology of each source of pressure head solutions. Specifically, we construct a sequence of simplicial complexes and track the "birth" and "death" of topological features across this sequence. Figure 4 shows that the μ solutions exhibit longer-lasting topological components than the ψ solutions, as all points die off much sooner (e.g., ~ 7.4 on the death axis) for the ψ solutions. Therefore, the use of an encoder $\hat{f}_{\text{NN}}$, which maps the pressure head solutions ψ to a latent space where μ solutions lie, can capture the distinct topological structures of two sources of pressure head solutions. Similarly, the decoder $\hat{f}_{\text{NN}}^{-1}$ transforms the latent representation μ back to the original solution space, ensuring that the essential topological features of ψ solutions are accurately captured and reconstructed.

Improving MP-FVM Algorithm Performance via Sobolev Training and Encoder-decoder Architecture

As previously discussed, we perform data augmentation on the reference solutions to increase dataset size and enhance generalization performance. Specifically, after we obtain a set of ψ solutions using the finite difference method developed by Celia et al. (1990), we make multiple copies of it and append each copy to the μ solutions obtained by the fixed-point iteration

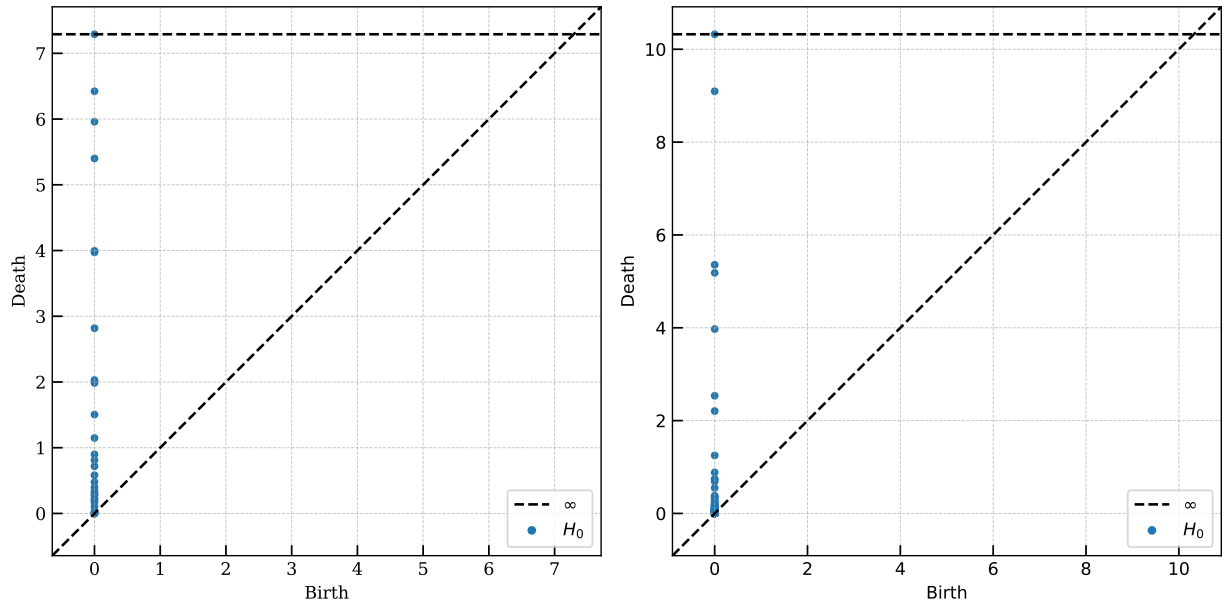


Figure 4: Persistence diagrams Edelsbrunner & Morozov (2013) for pressure head solutions ψ (left) and μ (right). The marked differences in topological features illustrate the need for an encoder to map ψ into the topological space of μ. Here, ∞ refers to infinite lifespan and H_0 are connected components.

scheme of Equation (2.1.5) under different static τ and S values. We then add zero-mean Gaussian noises with standard deviation varying from 0.1 to 0.5 to these augmented reference solutions. Overall, this leads to a total of $17,097$ reference solutions for neural network training and validation. Note that, as previously discussed, the original and augmented reference solutions are generated using a coarse grid ($\Delta z = 1$ cm). Thus, they can be obtained relatively efficiently. On the other hand, in the solution step, we will use a more refined grid containing 101 mesh points ($\Delta z = 0.4$ cm). This "coarse-to-fine" approach can therefore enhance the solution accuracy of our MP-FVM algorithm without requiring a large amount of high-accuracy, fine-mesh training data. Furthermore, when augmented reference solutions are used for training, only 100 additional epochs are needed to retrain neural networks that have already been trained using the original reference solutions. Second, we notice that there is only a slight difference in the final pressure head solution profile when

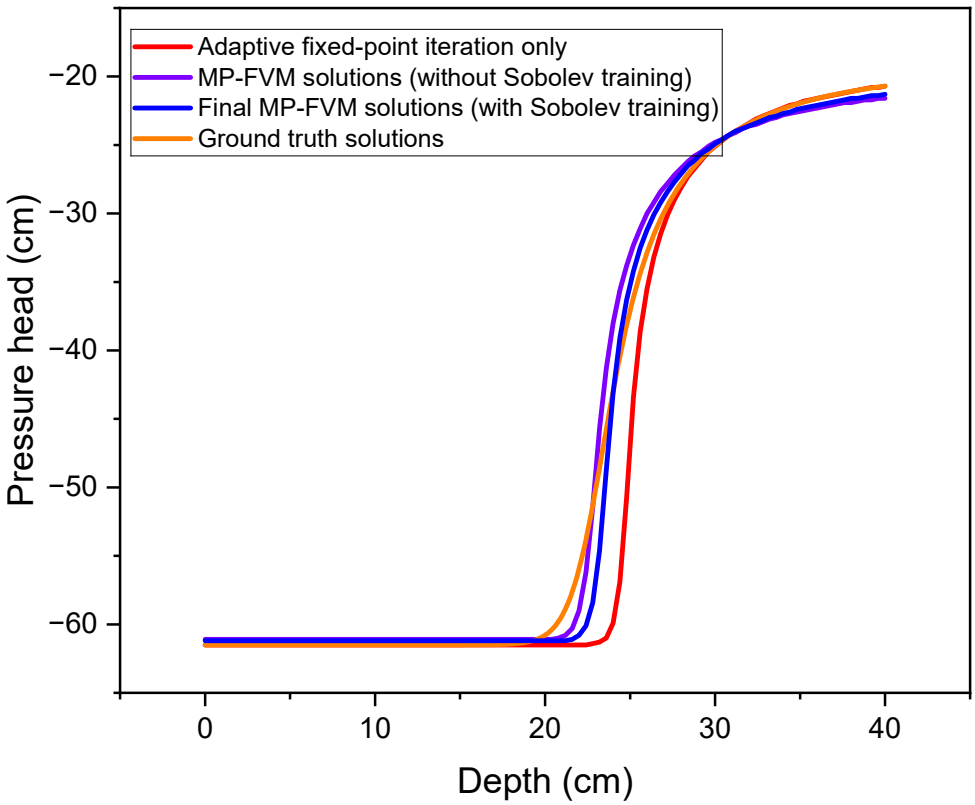


Figure 5: Comparison of pressure head solution profiles at $t = T = 360$ seconds produced from adaptive fixed-point iteration scheme only (Equation (2.1.5)) and from MP-FVM algorithm (Equation (2.2.3)) with and without implementing Sobolev training.

Gaussian noises of different magnitudes are directly added to the original reference solutions without augmenting them together. Third, increasing training data size (from $1,640$ to $17,097$) via data augmentation of original reference solutions is an effective way to improve solution accuracy of our MP-FVM algorithm, as the pressure head profile matches very well with the ground truth solution.

From Figure 5, it is clear that integrating adaptive fixed-point iteration scheme in the MP-FVM framework synergistically improves the overall solution accuracy of the Richards equation, especially in the region where pressure head changes rapidly with respect to depth (i.e., between $z = 20$ to 30 cm). On the other hand, we observe slight discrepancy in pressure

head solution close to $z = 40$ cm when comparing our MP-FVM algorithm with ground-truth solutions, whereas the solution produced by adaptive fixed-point iteration scheme alone matches perfectly with ground-truth solution at $z = 40$ cm, which corresponds to one of the boundary conditions. We believe that this is due to the fact that $\hat{f}_{\text{NN}}$ and $\hat{f}_{\text{NN}}^{-1}$ only approximate the true relationships f and f^{-1}, respectively, and the resulting induced error causes discrepancies in pressure head solutions even at the boundaries. To overcome this limitation, one way is to increase the size of the augmented reference solutions for neural network training. Another approach is to switch from MP-FVM (Equation (2.2.3)) to adaptive fixed-point iteration scheme only (i.e., Equation (2.1.5)) when solving for the boundary conditions. We leave this refinement for future research.

Figure 6 illustrates how Sobolev training affects the solution quality of our MP-FVM algorithm. Specifically, we find that, first, the effectiveness of Sobolev training depends on the choice of hyperparameter λ. Second, larger values of λ (e.g., 10^{-5}) may not lead to improved accuracy in pressure head solution, as in this case, neural network training may prioritize smoothness or derivative agreement over fitting the pressure head solutions. Third, smaller values of λ (e.g., 10^{-9}) could still be useful in improving solution accuracy compared to without Sobolev training (i.e., $\lambda = 0$). Last but not least, we notice that, when pre-trained models are used, the sensitivity of pressure head solution to λ is significantly reduced, especially for $\lambda < 10^{-6}$. We suspect that this is because pre-trained models already capture the relationships between ψ and μ solutions reasonably well, so that the Sobolev loss primarily serves to fine tune the models.

Convergence and Solution Accuracy Comparison

We compare our MP-FVM algorithm with other solvers based on computational performance and solution accuracy under two scenarios. In Scenario 1, we set the error tolerance tol to be 3.2×10^{-5}, whereas in Scenario 2, we set the total number of iterations $S = 500$. For static fixed-point iteration scheme, we use an optimal fixed-point parameter $\tau = \frac{1}{3.5} \approx 0.2857$

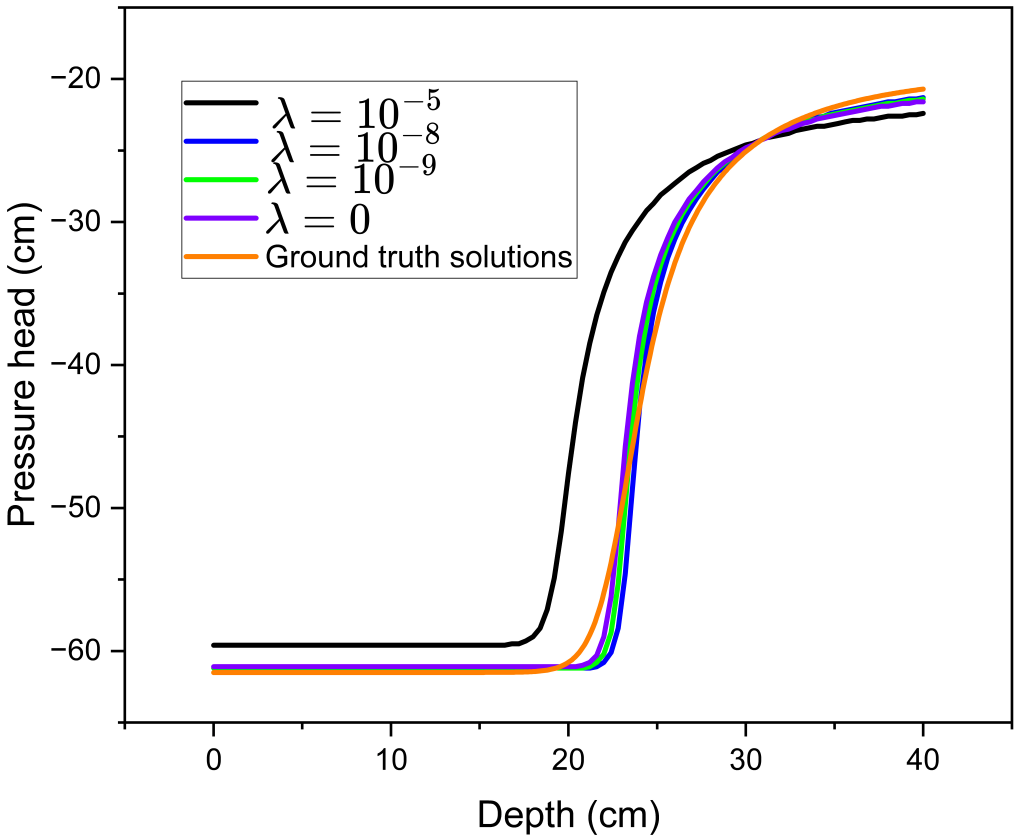


Figure 6: Comparison of pressure head solution profiles at $t = T = 360$ seconds produced from MP-FVM algorithm (Equation (2.2.3)) with implementing Sobolev training with different regularization parameters in Equation (2.2.1) and Equation (2.2.2) at Scenario 2. Here, we use the same $\lambda = \lambda_{\hat{f}_{\text{NN}}} = \lambda_{\hat{f}_{\text{NN}}^{-1}}$ and all neural networks are trained from scratch.

identified by trail-and-error process. In terms of computational performance, we use the condition number of matrix $\mathbf{A}$ defined in Equation (2.1.8), which measures the sensitivity of fixed-point iteration scheme subject to small perturbations, as the metric.

From Tables 3 and Table 4, we see that implementing adaptive fixed-point iteration scheme significantly improves the stability of conventional FVM and our MP-FVM algorithms, as matrix $\mathbf{A}$ is well-conditioned. These observations suggest that adaptive fixed-point iteration scheme outperforms static fixed-point iteration scheme in enhancing the convergence behavior of discretization-based solvers.

In terms of solution accuracy, we consider two metrics. The first metric is the discrepancy

Algorithm	Average condition number of $\mathbf{A}$ obtained from Zarba (1988) (Scenario 1)	
	Static fixed-point iteration scheme	Adaptive fixed-point iteration scheme
FVM	1.7668	1.0064
MP-FVM	1.7419	1.0075

Table 3: Comparison of average condition number under Scenario 1 across all time steps (as Equation (2.1.8) already considers all discretized cells) for conventional FVM and our MP-FVM algorithms that implement static or adaptive fixed-point iteration scheme.

Algorithm	Average condition number of $\mathbf{A}$ obtained from Zarba (1988) (Scenario 2)	
	Static fixed-point iteration scheme	Adaptive fixed-point iteration scheme
FVM	1.7206	1.0064
MP-FVM	1.7113	1.0071

Table 4: Comparison of average condition number under Scenario 2 across all time steps for conventional FVM and our MP-FVM algorithms that implement static or adaptive fixed-point iteration scheme.

from the ground truth solutions of Celia et al. (1990). The comparison results are illustrated in Figure 7. The second metric is the solver's performance in preserving the mass (moisture) balance, which is quantified by the mass balance measure MB defined in Celia et al. (1990):

$$\text{MB} = \frac{\text{total additional mass in the domain}}{\text{total water flux into the domain}}. \tag{2.3.1}$$

In Figure 7, we compare the pressure head profiles obtained from our MP-FVM algorithm (which implements adaptive fixed-point iteration scheme and Sobolev training), the conventional FVM algorithm (that implements adaptive fixed-point iteration scheme), and a state-of-the-art physics-informed neural network (PINN) solver based on Bandai & Ghezzehei (2021), against the ground truth solution Celia et al. (1990). Clearly, in both scenarios, compared with the MP-FVM solutions, PINN and FVM solutions are further apart from

ground truth solutions.

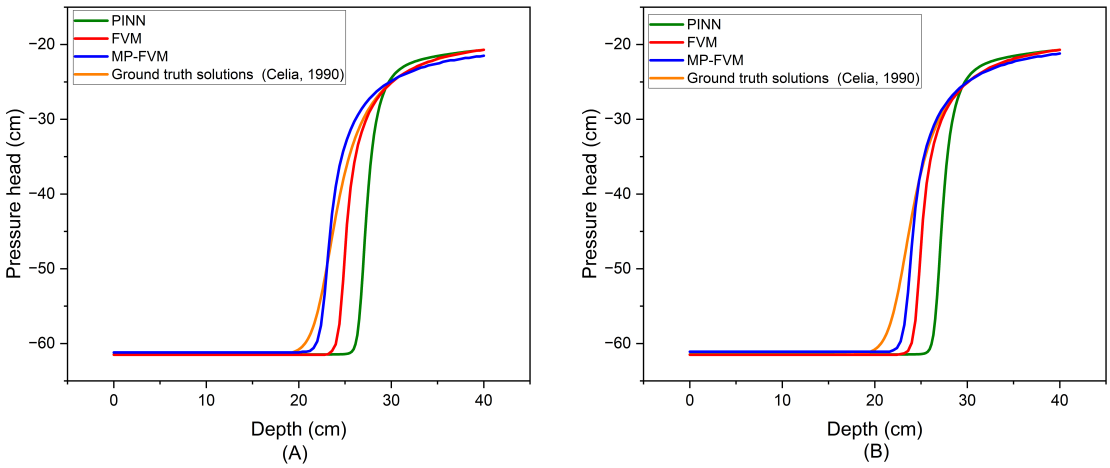


Figure 7: Pressure head profiles at $t = T = 360$ sec obtained by different algorithms under (left) Scenario 1, and (right) Scenario 2. Both conventional FVM and our MP-FVM algorithm incorporate adaptive fixed-point iteration scheme. Note the PINN solver is not an iterative method, thus the solution profile is the same under both scenarios.

From Tables 5 and 6, we observe that, in both Scenarios 1 and 2, our MP-FVM algorithm achieves the best MB values when using either coarse time steps suggested by the CFL condition De Moura & Kubrusly (2013) or a fixed time step. Considering that using coarse time steps reduces solution time without affecting solution quality, adopting a CFL-like condition is desired.

Remark on Computational Efficiency

Although our MP-FVM framework does involve neural network training which will take some additional time, there are several well-established strategies widely used in the machine/deep learning community to reduce the overall computational time and costs. For example, as previously discussed, one can leverage the previously trained neural network from a different problem setting as a good starting point to train with new dataset for the new problem setting in just a small number of epochs. To see this, we run the 1-D benchmark problem of Celia

Method used	Scenario	Average Δt (sec)	MB
FVM algorithm	1	18.90	96.13%
MP-FVM algorithm	1	18.68	**100.23%**
FVM algorithm	2	17.62	86.04%
MP-FVM algorithm	2	18.35	**97.29%**
Celia et al. (1990)	N/A	10	95.00%

Table 5: MB results of different numerical methods. Note that here, Δt is the determined for each method by the CFL condition De Moura & Kubrusly (2013) and we take the average across all iterations.

Method used	Scenario	MB ($\Delta t = 15$ sec)
FVM algorithm	1	98.87%
MP-FVM algorithm	1	**100.72%**
FVM algorithm	2	96.79%
MP-FVM algorithm	2	**97.81%**
Celia et al. (1990)	N/A	95.00%

Table 6: MB results of different numerical methods, in which a common $\Delta t = 10$ seconds is used for all numerical methods.

et al. (1990) in a Dell Precision 7920 Tower equipped with Intel Xeon Gold 6246R CPU and NVIDIA Quadro RTX 6000 GPU (with 24GB GGDR6 memory). The MP-FVM algorithm is implemented in Python 3.10.5. The total computational time for solving the Celia problem from scratch with $S = 500$ is 181.43 seconds, in which the neural network training step costs 127.58 seconds. On the other hand, when using a pre-trained model, the time for neural network training step and the total computational time are reduced by 89.79% and 63.21% down to 13.01 and 66.76 seconds, respectively. Meanwhile, the computational time for a

direct solver is 43.75 seconds. While our MP-FVM algorithm still takes more time than the direct solver, it is still an attractive numerical framework as: 1) it gives more accurate solutions; 2) its data-driven nature makes it suitable for seamless integration between physics-based modeling and in situ soil sensing technologies; 3) for large-scale and/or more complex problem settings, the neural network training time will become less significant compared to the actual solution time; and 4) our MP-FVM algorithm consumes less computational time compared to many neural PDE solvers Lu et al. (2020b); Brandstetter et al. (2022).

2.3.2 A 1-D Layered Soil Benchmark Problem

To investigate the robustness of our MP-FVM algorithm in handling realistic problems, we study the classic Hills' problem Hills et al. (1989) that involves the 1-D water infiltration into two layers of very dry soil, each having a depth of 30 cm. The top layer (layer 1) corresponds to Berino loamy fine sand and the bottom layer (layer 2) corresponds to Gledale clay loam. The WRC and HCF follow the Mualem-van Genutchen model. The soil-specific parameters are extracted from Hills et al. (1989) and are listed in Table 7. This benchmark problem also ignores the sink term.

As pointed out by Berardi et al. (2018), the dry condition is the most challenging physical case to model from a numerical point of view. The presence of discontinuous interface across the two soil layers presents another complication to this problem. We simulate the problem for up to 7.5 minutes. For neural network training, we generate a total of 30,500 reference solutions using conventional FVM solver (which implements the static fixed-point iteration scheme of Equation (2.1.5) with an optimal $\tau = 0.04$ identified by a trial-and-error procedure).

Figure 8 illustrates the soil moisture profile at three different times obtained using our MP-FVM algorithm, conventional FVM algorithm, as well as the Transversal Method of Lines (TMOL) solver Berardi et al. (2018) (which is considered the current state-of-the-art algorithm for this problem). All three approaches adopt the same discretized temporal

Soil	θ_r	θ_s	α	n	K_s
Berino loamy fine sand	0.029	0.366	0.028	2.239	541.0
Gledale clay loam	0.106	0.469	0.010	1.395	13.10

Table 7: Soil-specific parameters and constants used in the layered soil problem of Hills et al. (1989).

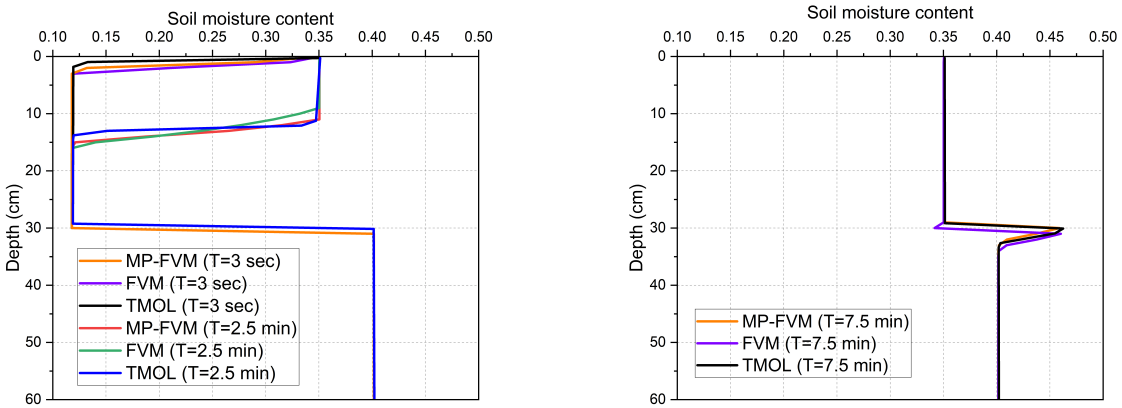


Figure 8: Comparison of soil moisture content profile obtained different methods with $\Delta z = 1$ cm under (left) MP-FVM, FVM and TMOL at $t = T = 3$ sec and $t = T = 2.5$ min and (right) MP-FVM, FVM and TMOL at $t = T = 7.5$ min. Note that TMOL by Berardi et al. (2018) is not an iterative method. FVM and MP-FVM are implemented for 500 iterations at every time step.

($\Delta t = 1$ second) and spatial steps ($\Delta z = 1$ cm). We set $\text{RE}_s = 1 \times 10^{-5}$ as the common stopping criterion. From Figure 8, we observe that our MP-FVM algorithm is capable of successfully simulating this challenging problem with discontinuities in soil properties at the interface. The soil moisture solutions obtained by our MP-FVM algorithm are also consistent with existing solvers. In fact, compared to the FVM solver, the solutions produced by our MP-FVM algorithm are closer to the state-of-the-art TMOL solutions.

2.3.3 A 2-D Benchmark Problem

In the second example, we study the 2-D Richards equation for an infiltration process in a 1m $\times$ 1m loam soil field Gasiorowski & Kolerski (2020). The spatial steps in both horizontal (Δx) and vertical (Δz) directions are set to be 0.02 m, and the time step used for this comparison study is $\Delta t = 10$ seconds. The Mualem-van Genuchten model (see Table 1) was used in this case study. The soil-specific parameters, given by Carsel & Parrish (1988), are listed in Table 8. This problem also ignores the sink term.

Property	Symbol	Value	Units
Saturated hydraulic conductivity	K_s	2.89×10^{-6}	m/s
Saturated water content	θ_s	0.43	–
Residual water content	θ_r	0.078	–
van Genuchten Constant	α	3.6	m^{-1}
van Genuchten Constant	n	1.56	–
Total time	T	1.26×10^4	s

Table 8: Soil-specific parameters and constants used in 2-D case study.

The initial and boundary conditions of this case study are given by:

$$\text{Initial condition: } \psi(x, z, t = 0 \text{ s}) = \begin{cases} 0\text{m}, & x \in [0.46, 0.54]\text{m},\ z = 0\text{m}, \\ -10\text{m}, & \text{otherwise.} \end{cases}$$

Boundary condition: $\psi(x \in [0.46, 0.54]\text{m}, z = 0, t) = 0\text{m}$, no slip conditions for other boundaries.

Note that the initial and boundary conditions are symmetric along $x = 0.5$m. We first obtain 9 sets of original reference solutions (ψ, μ), where each ψ or μ is a 51×51 array. Here, ψ solutions are obtained from the conventional 2-D FVM solver (which implements the static fixed-point iteration scheme) that uses a spatial step of 0.02 m under three different

fixed-point parameters $\tau = 2$, 2.22 and 2.5 and three total iteration counts $S = 300$, 400 and 500. Then, we apply data augmentation by adding Gaussian noises with σ^2 values ranging from 0.01 to 0.05 to generate a total of 400 reference solutions (which also contain the original reference solutions). Meanwhile, μ solutions are obtained from the HYDRUS software Šimůnek et al. (2016). These reference solutions are used to train the encoder-decoder networks for our MP-FVM algorithm. Each neural network contains 3 hidden layers and 256 neurons in each layer. ReLU activation function is adopted in each layer, and each neural network is trained by Adam optimizer for 100 epochs. We set the total iteration number to be $S = 500$. The total computational time for our MP-FVM algorithm to run from scratch with $S = 500$ is 1473.5 seconds, whereas the FVM solver takes 876.6 seconds under the same S.

Meanwhile, we also simulate this 2-D problem using HYDRUS software Šimůnek et al. (2016) and compare the pressure head results at $t = T = 1.26 \times 10^4$ sec with our MP-FVM algorithm and the FVM solver (the fixed-point parameter identified to be 1 by trial-and-error). From Figure 9, we can draw two observations. First, the pressure head solution profiles for both FVM and MP-FVM algorithms appear to be symmetric along $x = 0.5$ m, whereas HYDRUS 2D shows a clear asymmetric profile. As pointed out earlier, since the initial and boundary conditions are symmetric along $x = 0.5$ m, symmetry in the pressure head solutions is expected. This suggests that both FVM and MP-FVM based solvers can capture some degree of underlying physics of the original problem. Second, despite the assymetric behavior in pressure head profile, the size of isolines for the HYDRUS 2D simulation result is more similar to our MP-FVM solution than to the FVM solver solution. This observation is also consistent with the information presented in Figure 11a. In fact, both observations can also be carried over to the soil moisture profile, as shown in Figures 10 and 11b. Finally, in terms of mass conservation, our MP-FVM algorithm achieves significantly higher MB value compared to other benchmark solvers (see Table 9).

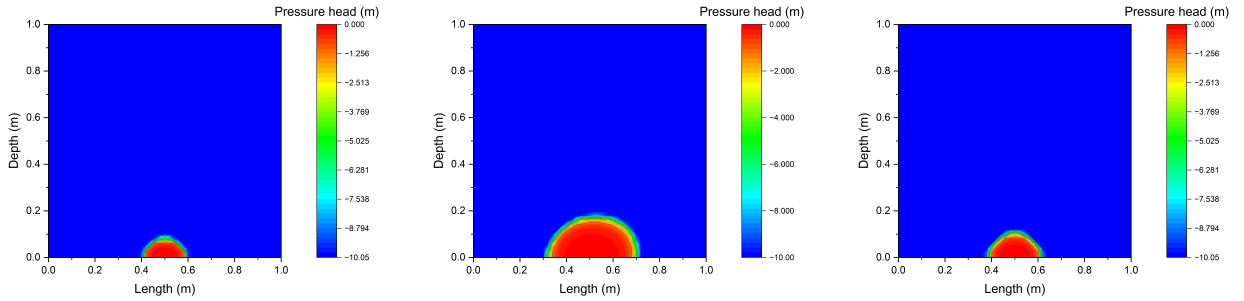


Figure 9: Pressure head solution profile obtained from three numerical methods: (left) FVM solver (fixed-point parameter $\tau = 1$); (middle) HYDRUS 2D software; (right) our MP-FVM algorithm.

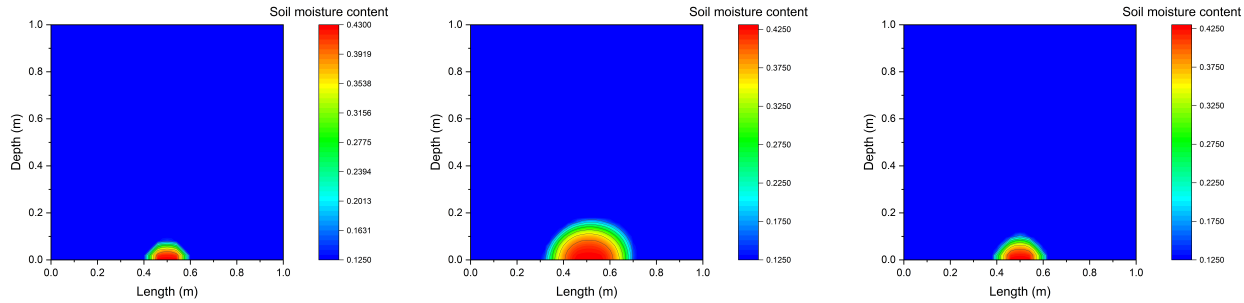


Figure 10: Soil moisture solution profile obtained from three numerical methods: (left) FVM solver (fixed-point parameter $\tau = 1$); (middle) HYDRUS 2D software; (right) our MP-FVM algorithm.

2.3.4 A 3-D Benchmark Problem with Analytical Solutions

Lastly, we consider a 3-D water infiltration example, in which the analytical solution exists Tracy (2006). In this example, V is a 3-D cuboid $[0, a] \times [0, b] \times [0, c]$. The hydraulic conductivity function follows the Gardner's model Gardner (1958) (see Table 1). The initial condition is given by:

$$\psi(x, y, z, t = 0) = h_r,$$

where h_r is a constant. The boundary condition is given by:

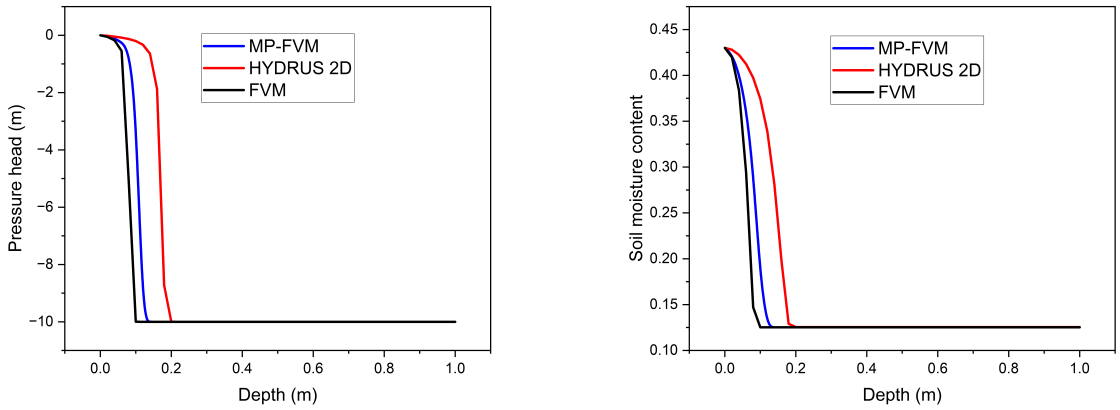


Figure 11: Cross-sectional view ($x = 0.5$m) of: (left) the pressure head profile; (right) soil moisture profile.

Method	MB ($\Delta t = 10$ sec)
FVM algorithm	63.12%
HYDRUS 2D simulation	62.45%
MP-FVM algorithm	**71.74%**

Table 9: MB results of three methods at $x = 0.5$ m.

$$\psi(x, y, z = c, t) = \frac{1}{\alpha} \ln \left[\exp(\alpha h_r) + \overline{h}_0 \sin \frac{\pi x}{a} \sin \frac{\pi y}{b} \right],$$

where $\overline{h}_0 = 1 - \exp(\alpha h_r)$. Ignoring the sink term, the pressure head solution for this problem was derived in Tracy (2006) as:

$$\psi = \frac{1}{\alpha} \ln \left\{ \exp(\alpha h_r) + \overline{h}_0 \sin \frac{\pi x}{a} \sin \frac{\pi y}{b} \exp\left(\frac{\alpha(c-z)}{2} \right) \left[\frac{\sinh \beta z}{\sinh \beta c} + \frac{2}{zd} \sum_{k=1}^{\infty} (-1)^k \frac{\lambda_k}{\gamma} \sin(\lambda_k z) \exp(-rt) \right] \right\}, \tag{2.3.3}$$

where $d = \frac{\alpha(\theta_s - \theta_r)}{K_s}$, $\lambda_k = \frac{k\pi}{c}$, $\gamma = \frac{\lambda_k^2 + \beta^2}{c}$ and $\beta = \sqrt{\frac{\alpha^2}{4} + (\frac{\pi}{a})^2 + (\frac{\pi}{b})^2}$.

The infinite series in Equation (2.3.4) is convergent by the alternating series test, and we consider the first $1,000$ terms of this series. Note from Equation (2.3.4) that the ana-

lytical solution depends only on the saturated (θ_s) and residual soil moisture content (θ_s). The Mualem-van Genuchten correlation Mualem (1976); Van Genuchten (1980) tabulated in Table 1 was used for the water retention curve $\theta(\psi)$. The constants and parameters used in this case study are listed in Table 10.

Property	Symbol	Value	Units
Saturated hydraulic conductivity	K_s	1.1	m/s
Saturated soil moisture	θ_s	0.5	–
Residual soil moisture	θ_r	0	–
Parameter in Gardner's model	α	0.1	m^{-1}
Parameter in intial and boundary conditions	h_r	-15.24	m
Length of V	a	2	m
Width of V	b	2	m
Depth of V	c	2	m
Total time	T	86,400	sec

Table 10: Soil-specific parameters and constants used in the 3-D case study.

Our goal is to compare the accuracy of our MP-FVM algorithm with FVM solvers using this analytical solution as the benchmark. We use our own in-house 3-D FVM solver, which implements the static fixed-point iteration scheme of the FVM-discretized 3-D Richards equation, to obtain $1,734$ original reference solutions using a coarse grid of $\Delta x = \Delta y = \Delta z = 0.4$ m under two fixed-point parameters $\tau = 1$ and 2 and five total iteration counts $S = 100, 200, \ldots, 500$, while excluding any NaN values. Then, data augmentation is applied by introducing Gaussian noise, resulting in a total of $8,820$ data points (which include the original reference solutions) for neural network training. For both FVM and MP-FVM algorithms, we set the tolerance to 1×10^{-9}, which can be achieved in less than 500 iterations for each time step.

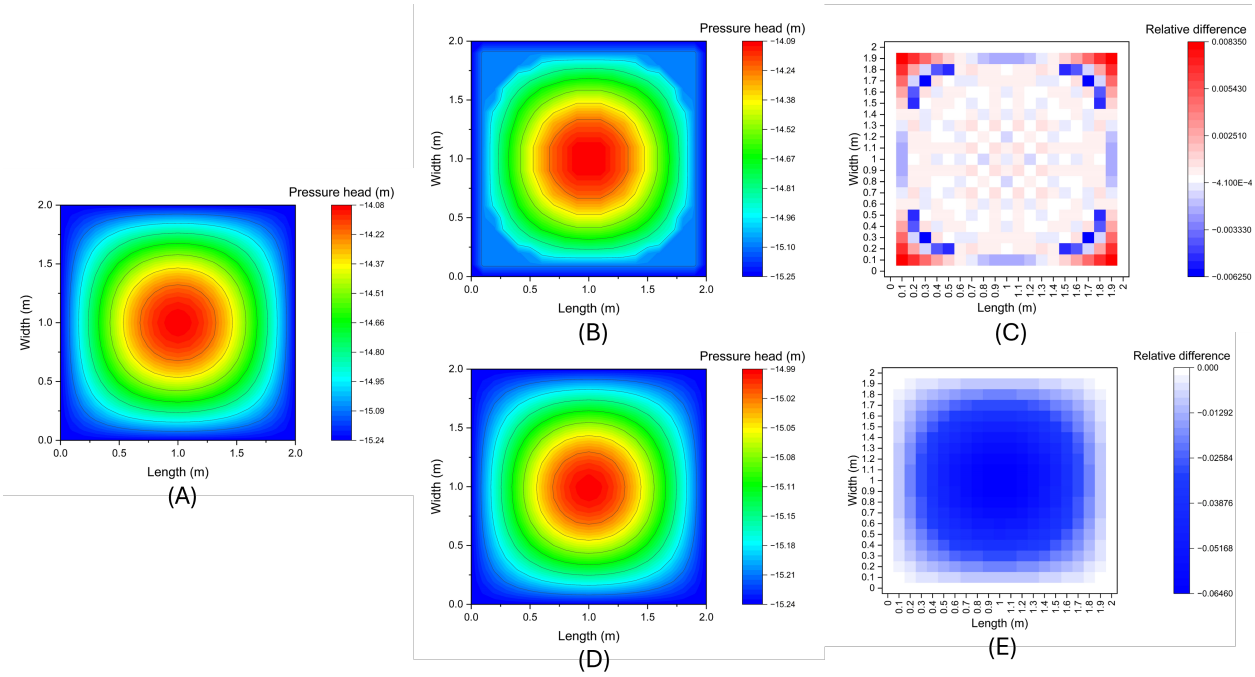


Figure 12: Pressure head solution at $z = 0.5$ m of different methods: (A) analytical solution, (B) MP-FVM algorithm, (C) the relative difference between analytical and MP-FVM solutions, (D) conventional FVM solver (which implements static fixed-point iteration scheme with an optimal $\tau = 2$) and (E) the relative difference between analytical solution and FVM solution.

We examine and compare the pressure head solutions at $z = 0.5$ and 1 m, which are shown in Figure 12 and 13, respectively. We quantify the relative difference between the numerical and analytical solutions by $\frac{\psi_{\text{analytical}} - \psi_{\text{numerical}}}{\psi_{\text{analytical}}}$. From the relative difference heat map of Figure 12c,e and 13c,e, we observe that, first, the magnitude of relative difference of our MP-FVM algorithm is significantly lower than that of the conventional FVM solver. Second, the largest relative difference of our MP-FVM pressure head solution occurs around the four corners of the x-y domain, whereas the largest relative difference of FVM solution occurs in the center of the x-y domain. Furthermore, in each cell, the relative difference of FVM based pressure head solution is always non-positive, whereas that of MP-FVM based solution can be positive or negative.

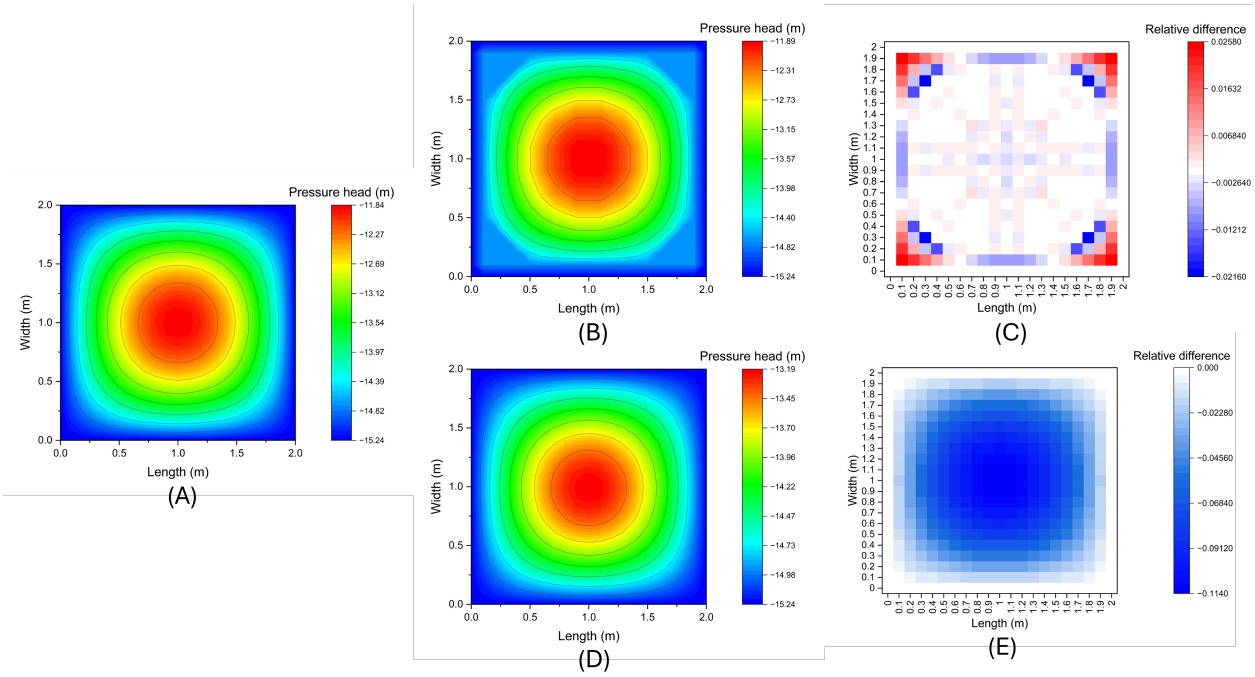


Figure 13: Pressure head solution at $z = 1.0$ m of different methods: (A) analytical solution, (B) MP-FVM algorithm, (C) the relative difference between analytical and MP-FVM solutions, (D) conventional FVM solver (which implements static fixed-point iteration scheme with an optimal $\tau = 2$) and (E) the relative difference between analytical solution and FVM solution.

Here, we provide some justifications for these observations. First, for conventional FVM solver that embeds the static fixed-point iteration scheme, we observe from Equations (2.1.5) that:

$$\psi_{\text{analytical}} - \psi_{\text{numerical}} \propto \left\{ \sum_{j \in \mathcal{N}_i} \left[K(\psi) \nabla (\psi + z) \right]_{\omega_{i,j}}^{m+1,s} \cdot \mathbf{n}_{\omega_{i,j}} A_{\omega_{i,j}} - \partial_t \theta_i^{m+1} \text{vol}(V_i) \right\},$$

for any s, discretized cell V_i, and discretized time step m. Since the hydraulic conductivity function is positive and symmetric along $x = 1$ m and $y = 1$ m, and $\nabla \psi \big|_{\omega^+ := [0,1] \times [0,1] \times z} = -\nabla \psi \big|_{\omega^- := [1,2] \times [1,2] \times z}$, we have $\sum_{j \in \mathcal{N}_i} \left[K(\theta(\psi)) \nabla (\psi + z) \right]_{\omega_{i,j}}^{m+1,s} \cdot \mathbf{n}_{\omega_{i,j}} A_{\omega_{i,j}} > 0$. Meanwhile, $\partial_t \theta_i^{m+1}(\psi) \text{vol}(V_i)$ is typically small due to the slow dynamics of water infiltration in soil and the fact that $\text{vol}(V_i)$ is small. Thus, we have $\psi_{\text{analytical}} - \psi_{\text{numerical}} > 0$ for the FVM solution, which explains why the relative difference is non-positive. On the other hand, for

our MP-FVM algorithm, the use of neural networks to approximate f and f^{-1} complicates the behavior (including the sign) of the relative difference.

Regarding the distribution of the magnitude of relative difference in the FVM solver, since hydraulic conductivity function is an increasing function of ψ, and ψ is at its maximum at the center of the x–y plain, it is expected that $\sum_{j\in\mathcal{N}_i}\left[K(\psi)\nabla(\psi+z)\right]_{\omega_{i,j}}^{m+1,s}\cdot\mathbf{n}_{\omega_{i,j}}A_{\omega_{i,j}}$ (hence the relative difference) is maximized at and around the center of the x-y plane. However, for MP-FVM based pressure head solution, we suspect that the higher relative difference at the four corners may be attributed to the slight decrease in accuracy of neural networks in approximating f and f^{-1} near the domain boundaries.

Finally, we evaluate the Mean Absolute Error (MAE) by averaging the absolute errors between numerical and analytical pressure head solutions across all cells on two vertical planes, $z = 0.5$ m and $z = 1$ m. For $z = 0.5$ m, $\text{MAE}_{\text{MP-FVM}}$ and MAE_{FVM} are calculated to be 0.0146 and 0.3444, respectively. For $z = 1$ m, $\text{MAE}_{\text{MP-FVM}}$ and MAE_{FVM} are 0.0375 and 0.5653, respectively. This indicates that the MAE of the FVM solutions is typically 1 to 2 orders of magnitude higher than the MP-FVM solutions, highlighting the accuracy of our MP-FVM algorithm.

2.4 A Realistic Case Study

Finally, we consider a real-world case study adopted from Orouskhani et al. (2023), where infiltration, irrigation, and root water extraction take place in circular agricultural field, equipped with a center-pivot irrigation system with a radius of 50 m, located in Lethbridge, Alberta. Soil moisture sensors are inserted at a depth of 25 cm across 20 different locations in this field to collect soil moisture data every 30 min from June 19 to August 13, 2019. To validate our MP-FVM algorithm in solving real-world 3-D applications, we select one of the 20 locations where the Mualem-van Genuchten WRC and HCF model parameters are identified and given in Orouskhani et al. (2023). We consider a cylindrical control volume V with a radius of 0.1 m and a depth of 25 cm. We discretize V into 6, 40 and 22 nodes in

the radial, azimuthal, and axial directions, respectively. The time step size Δt is determined using the heuristic formula. Thus, we reformulate Equation (2.2.3) in cylindrical coordinate system as:

$$\mu_i^{m+1,s+1} = \mu_i^{m+1,s} + \tau_i^{m+1,s} \sum_{j \in \mathcal{N}_i} K_{\omega_{i,j}}^{m+1,s} \hat{\mathbf{e}}_j \cdot \mathbf{n}_{\omega_{i,j}} \frac{\mu_j^{m+1,s} - \mu_i^{m+1,s}}{\text{dist}(V_j, V_i)} A_{\omega_{i,j}} + f^{-1}(J),$$

where $\hat{\mathbf{e}}_j = (1, \frac{1}{r_j^2}, 1)^T$ and

$$\begin{aligned} J =& \tau_i^{m+1,s} \sum_{j \in \mathcal{N}_i} K_{\omega_{i,j}}^{m+1,s} \hat{\mathbf{e}}_j \cdot \mathbf{n}_{\omega_{i,j}} \frac{z_j - z_i}{\text{dist}(V_j, V_i)} A_{\omega_{i,j}} - \tau_i^{m+1,s} \frac{\theta_i^{m+1,s} - \theta_i^m}{\Delta t} \text{vol}(V_i) \\ & - \tau_i^{m+1,s} S(\psi_i^{m+1,s}) \text{vol}(V_i). \end{aligned} \tag{2.4.1}$$

Here, the sink term in S follows the Feddes model Feddes & Zaradny (1978):

$$S = \sigma(\psi) S_{\max}, \tag{2.4.2}$$

where $S_{\max}$ is the maximum possible root extraction rate and σ denotes a dimensionless water stress reduction factor (see Agyeman et al. (2020) for the detailed formulation).

The boundary conditions are given by:

$$\begin{aligned} &\frac{\partial \psi(r, \omega, z)}{\partial r} = 0 \quad \text{at } r = 0\text{m}, \\ &\frac{\partial \psi(r, \omega, z)}{\partial r} = 0 \quad \text{at } r = 0.1\text{m}, \\ &\frac{\partial \psi(r, \omega, z)}{\partial z} = 0 \quad \text{at } z = 0 \text{ cm}, \\ &\frac{\partial \psi(r, \omega, z)}{\partial z} = -1 - \frac{u_{\text{irr}}}{K(\psi)} \quad \text{at } z = 25 \text{ cm}, \\ &\psi(r, \omega = 0, z) = \psi(r, \omega = 2\pi, z), \end{aligned}$$

where u_{irr} is the irrigation rate (in m/s). The initial condition is simply:

$$\psi(x, y, z, t = 0) = h_r,$$

where h_r is the starting pressure head recording.

Note that the boundary conditions are time dependent due to u_{irr}. This poses a potential computational challenge, as the neural networks typically need to be retrained whenever

initial and/or boundary conditions change Mattey & Ghosh (2022); Brecht et al. (2023). To overcome this practical challenge, we adopt a new approach of training the two neural networks with 3,000 epochs based on the boundary conditions for June 19, 2019 (no irrigation) when data collection began. Then, the trained weights within these two neural networks serve as the starting point for retraining when a new set of boundary conditions is adopted. This way, only 500 epochs are sufficient to retrain the neural networks. For each set of boundary conditions, we obtain the training set containing $84,480$ reference solutions. In addition, the dataset provided by Orouskhani et al. (2023), after performing data augmentation by introducing Gaussian noises, is also included in our training dataset. Each neural netowrk, which has 5 hidden layers with 256 neurons in each layer, is trained using SGD optimizer with a learning rate of 0.001. We set the stopping criterion to be 1×10^{-9}, which can be achieved well within 500 iterations.

For this problem, we simulate the pressure head from 1:00 am on June 19, 2019 to 5:00 pm on July 28, 2019. As mentioned in Orouskhani et al. (2023), there are two irrigation instances between this time frame, one is on July 4 (the 15th day, 1.81 mm) and the other is on July 18 (the 30th day, 1.58 mm). Figure 14 shows the pressure head solution profile obtained by our MP-FVM algorithms compared to the experimental measurements provided by Orouskhani et al. (2023) over the course of 35 days. We observe that, most of the time, the MP-FVM solutions match with the experimental measurements very well. The only major mismatch between experimental measurements and MP-FVM solutions occurs on the 30th day, which corresponds to the time when the irrigation takes place. We believe that the mismatch is due to our simplifying assumption regarding the irrigation schedule. Due to the limited information we have on the exact irrigation schedule and intensity, we have to assume that the irrigation instances occurred throughout the day. Thus, we simply divide the irrigation amount by $86,400$ seconds to obtain u_{irr}. However, in reality, the irrigation could end in less than 24 hours. With more accurate u_{irr} model, our MP-FVM algorithm is expected to produce highly accurate solutions that match more closely with experimental

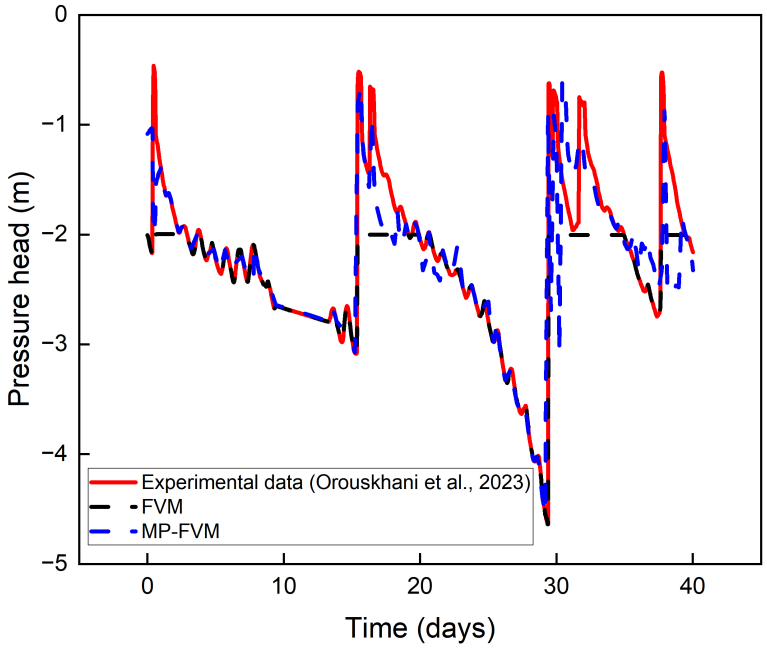


Figure 14: Comparison of pressure head profile at $z = 25$ cm in a selected 0.1-m radius region (averaged for all $6 \times 40 = 240$ cells at $z = 25$ cm) in the field. Note that the standard FVM solver becomes highly inaccurate when the boundary condition changes (15th day, 30th day, etc.). The flattening of true peaks of pressure head solutions represents a nonphysical smoothing of the true solution Miller et al. (1998), which we suspect to come from the numerical dispersion and inherent discrete maximum principle (DMP)-type peak clipping behavior observed in standard FVM schemes Njifenjou (2025).

measurements at all times. This makes our MP-FVM algorithm an accurate and scalable numerical framework to solve Richards equation over a long period of time.

CHAPTER III

ADAPTIVE FOURIER DECOMPOSITION-GUIDED NEURAL OPERATORS

3.1 Problem Statement

We consider a PDE defined on a spatial domain $\Omega \subset \mathbb{R}^d$ and a time interval $(0, T]$:

$$\mathcal{L}_\alpha[u(x,t)] = f(x,t), \quad \forall (x,t) \in \Omega \times (0,T], \tag{3.1.1}$$

where $\mathcal{L}$ denotes the differential operator, $f(x,t)$ is the source/sink term, and the parameter function $\alpha \in \mathcal{A}$ specifies the physical parameters and the initial and boundary conditions. Our goal is to learn a neural operator $G : \mathcal{A} \to \mathcal{F}(D \times [0,T])$, which maps the parameter function α from its parameter space $\mathcal{A}$ to the corresponding solution $u(x,t) \in \mathcal{F}$. In this work, we focus on two types of tasks: (i) the static task, which solves a PDE for one set of physical parameters α and a fixed final time T (i.e., $u(x,T)$); and (ii) the autoregressive task, which forecasts the PDE solution at time step $t+1$ (i.e., $u(x,t+1)$) based on the solution at the previous time step t (i.e., $u(x,t)$).

3.2 Related Work

Classic Fourier-based methods, such as Fourier transform approaches Negero (2014), Fourier series expansions Asmar (2016), and Fourier spectral methods Alali & Albin (2020), have been extensively used to solve PDEs numerically. Classic Fourier-based methods offer accurate and efficient representations of smooth, periodic functions by transforming differential operators into simple algebraic operations in the frequency domain. However, the use

of global basis functions produces oscillations when approximating functions with discontinuities or sharp transitions Gottlieb & Shu (1997). Furthermore, the fixed basis structure in these methods lacks adaptability to signals with time-localized, transient, or nonperiodic features. In addition, these methods are typically defined on simple, regular domains, making them difficult to apply directly to manifolds.

Operator learning aims to directly learn the mapping between infinite-dimensional function spaces (e.g., from input functions to solutions) to enable fast, mesh-independent approximation of PDE solutions across various input conditions, including source and/or sink term, physical parameters, and initial and boundary conditions. Among existing operator learning-based PDE solvers, two notable ones backed by the approximation theory are DeepONet Lu et al. (2019, 2021), which is inspired by the universal approximation theorem for nonlinear operators, and the Fourier Neural Operator (FNO) Li et al. (2020b, 2023b), which performs convolution in the frequency domain to capture global spatial dependencies efficiently. Both operator learning paradigms have led to several new variants. Some of the recently developed network architectures He et al. (2023); Goswami et al. (2022); He et al. (2024); Li et al. (2023a) built upon DeepONet provide enhancements such as physics-informed structure, parameterized geometry and phase-field modeling. Some of the new variants of FNO include Factorized FNO (F-FNO) Tran et al. (2021), Decomposed FNO (D-FNO) Li & Ye (2025), Spherical FNO Bonev et al. (2023), Domain Agnostic FNO (DAFNO) Liu et al. (2023), Wavelet Neural Operator (WNO) Tripura & Chakraborty (2023), Multiwavelet Neural Operator (MWT) Gupta et al. (2021), Coupled Multiwavelet Neural Operator (CMWNO) Xiao et al. (2025), and Adaptive Fourier Neural Operator (AFNO) Guibas et al. (2021).

Physics-informed representation learning and variational autoencoder (VAE). Another avenue for solving PDEs is to directly incorporate physical knowledge and constraints derived from the PDE into a neural architecture. One of the popular frameworks

is the Physics-Informed Neural Network (PINN) Raissi et al. (2019, 2017), where the PDE itself is embedded in the loss function as a regularization term. Another approach is to introduce variational autoencoders (VAEs) Tait & Damoulas (2020); Kingma et al. (2013) in a physics-informed architecture. This provides a structured latent space and a probabilistic framework for integrating physics, leading to more stable and generalizable representation learning. Several physics-informed VAE models have recently been proposed, including Glyn-Davies et al. (2024); Zhong & Meidani (2023); Takeishi & Kalousis (2021); Lu et al. (2020a). Specifically, Lu et al. (2020a) used a dynamics encoder and a propagating decoder to extract interpretable physical parameters from PDEs. Later, Takeishi & Kalousis (2021) proposed a physics-informed VAE model by introducing physics-based models to augment latent variables, encoder, and decoder. However, these methods lack rigorous theoretical justifications for the design of their neural architectures that ensure convergence and performance guarantees.

3.3 Preliminaries to Adaptive Fourier Decomposition (AFD)

AFD is a novel signal decomposition technique that leverages the Takenaka-Malmquist system and adaptive orthogonal bases Qian (2010); Qian et al. (2012). It is established as a new approximation theorem in a reproducing kernel Hilbert space (RKHS) sparsely in a given domain Ω as $s = \sum_{i=1}^{\infty} \langle s, \mathscr{B}_i \rangle \mathscr{B}_i$ for the chosen orthonormal bases $\mathscr{B}_i$ Saitoh et al. (2016). An RKHS is a Hilbert space of functions where evaluation at any point is continuous with respect to the inner product $\langle \cdot, \cdot \rangle$, and each point on the domain corresponds to a unique kernel function. For AFD in RKHS, the sparse bases $\{\mathscr{B}_i\}_i$ are made orthonormal to each other by applying Gram-Schmidt orthogonalization to the normalized reproducing kernels associated with a set of adaptively selected "poles" $\{a_i\}_i$, which are complex numbers used to parameterize the sparse bases. Specifically, to decompose signals in a Hardy space (i.e., a Hilbert space consisting of holomorphic functions defined on the unit disk), which can be further relaxed to an RKHS Song & Sun (2022), the orthonormal basis functions $\mathscr{B}_i$ can be

derived as:

$$\mathscr{B}_i(z) = \frac{\sqrt{1-|a_i|^2}}{1-\overline{a_i}z}\prod_{j=1}^{i-1}\frac{z-a_j}{1-\overline{a_j}z}, \quad a_i \in \mathbb{D}, \tag{3.3.1}$$

where $\mathbb{D} = \{z \in \mathbb{C} : |z| < 1\}$. To adaptively select the sequence of poles such that convergence of AFD approximation is ensured, one shall follow the so-called "maximal selection principle", such that the resulting $|\langle s, \mathscr{B}_i\rangle|$ is as large as possible. That is, to select the next pole a_i given $i-1$ already selected poles, $a_1, \ldots, a_{i-1}$ (hence bases $\mathscr{B}_1, \ldots, \mathscr{B}_{i-1}$), the corresponding orthonormal basis $\mathscr{B}_i$ needs to satisfy:

$$|\langle s, \mathscr{B}_i\rangle| \geq \rho_i \sup\left\{|\langle s, \mathscr{B}_i'\rangle| b_i \in \Omega \backslash \{a_1, \ldots, a_{i-1}\}\right\}, \tag{3.3.2}$$

where $0 < \rho_0 \leq \rho_i < 1$, $\mathscr{B}_1' = \frac{k_{b_1}}{\|k_{b_1}\|_{H(\Omega)}}$ and $\mathscr{B}_i' = \frac{k_{b_i} - \sum_{j=1}^{i-1}\langle k_{b_i}, \mathscr{B}_j\rangle \mathscr{B}_j}{\|k_{b_i} - \sum_{j=1}^{i-1}\langle k_{b_i}, \mathscr{B}_j\rangle \mathscr{B}_j\|_{H(\Omega)}}$. Here, k_{b_i} is the reproducing kernel (e.g., Gaussian or Bergman kernel) at b_i. In classic AFD theory, the algorithmic procedure of pole selection, which is discussed in Song & Sun (2022), is computationally expensive. Therefore, integrating the classical AFD with neural operators is a promising approach to enable fast and accurate solution of PDEs through the use of adaptive orthonormal basis functions.

3.4 AFDONet Architecture

Guided by the AFD theory, we design AFDONet to approximate PDE solution spaces on any smooth manifold. The AFDONet architecture shown in Figure 15 consists of an encoder, a latent-to-RKHS network, and an AFD-type dynamic convolutional kernel network (CKN). These components work synergistically to enhance the performance of the AFDONet solver. After the encoder, AFDONet identifies the closest RKHS where the latent variables reside using a latent-to-RKHS network. Subsequently, AFDONet reconstructs the PDE solutions by replicating the AFD operation and adaptively selecting the poles using a specially designed decoder network. For static tasks, the training dataset is denoted as $\{u(x,T)\}_{\{\alpha\}}$ for different sets of physical parameters α, while for autoregressive tasks, the training dataset is denoted as $\{u(x,t), u(x,t+1)\}_{t=0}^{T}$.

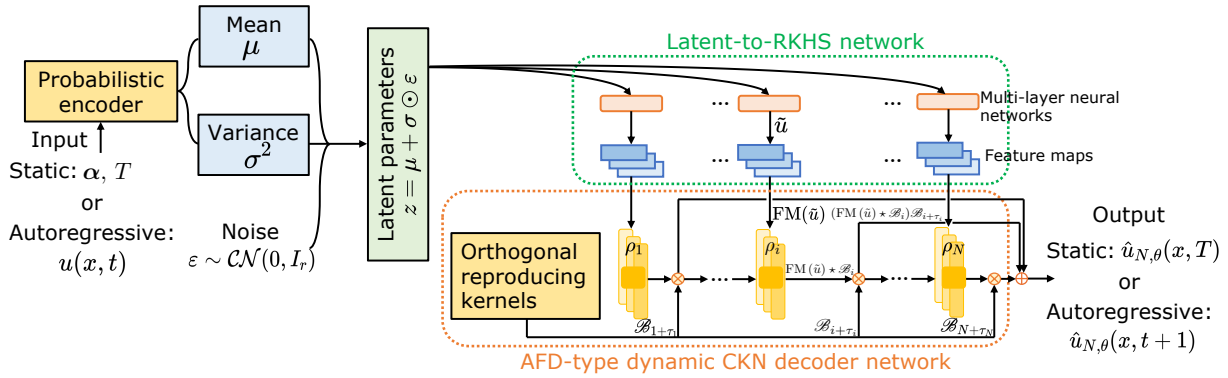


Figure 15: Our proposed AFDONet framework, which adopts VAE as the backbone, introduces a latent-to-RKHS network and a dynamic CKN decoder to reproduce the AFD setting and operation.

The use of VAE as architecture backbone is motivated from both methodological and experimental perspectives. From a methodological perspective, the use of VAE architecture as the backbone for our AFDONet is motivated by several reasons. First, many PDE solution fields lie on low-dimensional manifolds in high-dimensional function space. VAE-based neural operators can learn a probabilistic latent representation of these manifolds, mapping high-dimensional inputs to a compact latent space while capturing variation in solution behavior. This reduces the complexity of learning and enables generalization across parametric inputs, as shown in many prior successes in VAE-based neural operators Zhong & Meidani (2023); Rafiq et al. (2025); Lu et al. (2020a); Takeishi & Kalousis (2021). Second, VAE is inherently connected to AFD theory in several ways. For instance, VAEs benefit from frequency transformations Li et al. (2024), which are the foundation of bases used in AFD. Also, the maximal selection principle of basis functions in AFD aligns well with the variational inference of VAE Chen et al. (2020a).

From an experimental perspective, we will show in Section 3.10 that the use of VAE and its holistic integration with other components in the AFDONet architecture help significantly improve the accuracy of PDE solutions on manifolds.

The encoder network maps the inputs α or $u(x,t)$ to a latent space in the complex domain $\mathbb{C}^{2r}$ using a standard probabilistic encoder network based on the VAE framework. For the static task, this means:

$$\big(\mu(\alpha),\, \log\sigma^2(\alpha)\big) \;=\; A_2\big(\Phi\big(A_1\alpha\big)\big), \quad z \;=\; \mu(\alpha) + \sigma(\alpha)\odot\varepsilon, \quad \varepsilon\sim\mathcal{CN}(0, I_r), \tag{3.4.1}$$

where $A_1 \in \mathbb{C}^{W_\mathrm{e}\times d}$ and $A_2 \in \mathbb{C}^{2r\times W_\mathrm{e}}$ are the weight matrices (where $W_\mathrm{e} = \mathcal{O}(r)$), $\Phi(\cdot)$ is the activation function, the latent mean is $\mu(\alpha) \in \mathbb{C}^r$, the log-variance is $\log\sigma^2(\alpha) \in \mathbb{C}^r$, and z is the latent parameter vector.

For the autoregressive task, the input $u_t = u(x,t)$ lies on the Hilbert space $H(\mathcal{M})$ of manifold $\mathcal{M}$. Therefore, $u_t = u(x,t)$ must be projected from $H(\mathcal{M})$ into an appropriate complex domain. Let $\{\phi_k\}_{k=0}^{\infty}$ be an orthonormal Fourier basis. Then, we define a linear projection:

$$\Pi_K u_t := \big(\langle u_t, \phi_0\rangle,\, \ldots,\, \langle u_t, \phi_{K-1}\rangle\big) \in \mathbb{C}^K, \tag{3.4.2}$$

which retains the first K modes of the field. This leads to the following encoder structure:

$$\big(\mu_t,\, \log\sigma_t^2\big) \;=\; A_2\Big(\Phi\big(A_1\,\Pi_K u_t\big)\Big), \quad z_t \;=\; \mu_t + \sigma_t\odot\varepsilon_t, \quad \varepsilon_t\sim\mathcal{CN}(0, I_r), \tag{3.4.3}$$

where $A_1 \in \mathbb{C}^{W_\mathrm{e}\times K}$ and $A_2 \in \mathbb{C}^{2r\times W_\mathrm{e}}$ are the weight matrices (where $W_\mathrm{e} = \mathcal{O}(r)$), $\Phi(\cdot)$ is the activation function. In both tasks, the encoder network has a depth $L_\mathrm{e} = 2$ and width $W_\mathrm{e} = \mathcal{O}(r)$.

The latent-to-RKHS network maps the latent parameters to convolutional kernels while constraining the corresponding functional space to be an RKHS, where the AFD operations are defined. This extends the latent-to-kernel network proposed by Lu et al. (2020a) by explicitly accounting for the fact that the kernels are constructed in a Hilbert space. Our latent-to-RKHS network consists of multi-layer fully-connected feedforward (MLP) networks and feature maps. The MLP networks will first take the latent parameter vector z obtained from the encoder network to generate $\tilde{u}(x,\cdot)$ on $H(\mathcal{M})$. Then, feature maps $\mathrm{FM}(\cdot)$ will map $\tilde{u}(x,\cdot)$ to its nearest RKHS $\mathcal{H}(\mathcal{M})$ via orthogonal projection. This way, the latent-to-RKHS

network learns the feature maps from $H(\mathcal{M})$ to its nearest RKHS $\mathcal{H}(\mathcal{M})$, in which the reproducing kernel k_a can be obtained by:

$$k_a(\xi) \;=\; \sum_{i=1}^{N'} \nu_i(a)\, e^{2\pi j \phi\cdot(\xi - y_i)}, \qquad \forall a, \xi \in \mathcal{M} \tag{3.4.4}$$

where $j^2 = -1$ and ϕ is the fundamental frequency. Here, weights $\nu_i \in \mathbb{C}$ and parameters $y_i \in \mathcal{M}$ are learnable from the latent-to-RKHS network. Essentially, a feature map applies a fast Fourier transform (FFT) to its input, multiplies the top N' low-frequency components by learnable complex weights while discarding the high-frequency components, and then performs an inverse FFT. Note that this is different from Fourier layers in FNO because we only perform one-sided (positive-frequency) operations, whereas FNO performs both positive- and negative-frequency operations. This is because, in AFD, negative frequencies are redundant, as they can be determined by the positive ones via complex conjugation.

We also point out that, since Fourier basis kernel $e^{2\pi j\phi\cdot(\xi - y_i(a))}$ lies in $\mathcal{H}(\mathcal{M})$, which is closed under finite linear combinations, the reproducing kernel $k_a(\xi)$ is guaranteed to lie in $\mathcal{H}(\mathcal{M})$ as well. In addition, although Fourier basis kernels are orthogonal to each other, the reproducing kernels are not. Thus, orthogonalization is still needed.

Orthogonal reproducing kernels. Like AFD, in AFDONet, a set of reproducing kernels in Equation 3.4.4, each corresponding to one of the N distinct poles $a_1, \ldots, a_N \in \mathcal{M}$, need to be first orthogonalized via Gram-Schmidt orthogonalization:

$$\mathscr{B}_1 = \frac{k_{a_1}(\xi)}{\|k_{a_1}(\xi)\|_{\mathcal{H}(\mathcal{M})}}; \quad \mathscr{B}_i = \frac{k_{a_i}(\xi) - \sum_{j=1}^{i-1}\langle k_{a_i}(\xi), \mathscr{B}_j\rangle \mathscr{B}_j}{\left\| k_{a_i}(\xi) - \sum_{j=1}^{i-1}\langle k_{a_i}(\xi), \mathscr{B}_j\rangle \mathscr{B}_j \right\|_{\mathcal{H}(\mathcal{M})}} \quad \text{for } i = 2, \ldots, N. \tag{3.4.5}$$

To adaptively select the poles, we develop a maximum selection principle that is analogous to Equation 3.3.2 in AFD theory as:

$$\left|\mathrm{FM}\left(\tilde{u}(x,\cdot)\right) \star \mathscr{B}_i\right| \geq \rho_i \sup\left\{\left|\mathrm{FM}\left(\tilde{u}(x,\cdot)\right) \star \mathscr{B}'_i\right| : b_i \in \mathcal{M}\backslash\{a_1, \ldots, a_{i-1}\}\right\}, \tag{3.4.6}$$

where $\mathscr{B}'_1 = \frac{k_{b_1}(\xi)}{\|k_{b_1}(\xi)\|_{\mathcal{H}(\mathcal{M})}}$, $\mathscr{B}'_i = \frac{k_{b_i}(\xi) - \sum_{j=1}^{i-1}\langle k_{b_i}(\xi), \mathscr{B}_j\rangle \mathscr{B}_j}{\left\| k_{b_i}(\xi) - \sum_{j=1}^{i-1}\langle k_{b_i}(\xi), \mathscr{B}_j\rangle \mathscr{B}_j \right\|_{\mathcal{H}(\mathcal{M})}}$ for $i = 2, \ldots, N$, and k_{b_i} is the reproducing kernel at b_i.

The AFD-type decoder network reconstructs PDE solutions from $\mathrm{FM}\left(\tilde{u}(x,\cdot)\right)$ once the RKHS and its reproducing kernel are established. The decoder adopts a dynamic convolutional kernel network (CKN) Mairal et al. (2014); Chen et al. (2020b), which (i) performs cross-correlation between $\mathrm{FM}\left(\tilde{u}(x,\cdot)\right)$ and the orthogonal reproducing kernels $\mathscr{B}_i$, (ii) assigns a multiplier $0 < \rho_0 \leq \rho_i < 1$ to the output of each convolutional layer, and (iii) incorporates skip connections for each convolutional layer. With this, the output of the dynamic CKN with N convolutional layers (each pole is associated with a layer) replicates the AFD operation and reconstructs the PDE solution as:

$$\hat{u}_{N,\theta}(x,\cdot) = \sum_{i=1}^{N} \langle \mathrm{FM}\left(\tilde{u}(x,\cdot)\right), \mathscr{B}_{i+\tau_i} \rangle \mathscr{B}_{i+\tau_i} = \sum_{i=1}^{N} \left(\mathrm{FM}\left(\tilde{u}(x,\cdot)\right) \star \mathscr{B}_i\right) \mathscr{B}_{i+\tau_i}, \tag{3.4.7}$$

where $\star$ is the cross-correlation defined as $f \star g(\tau_i) = \int_{\mathcal{M}} \bar{f}(z) g(z+\tau_i) \mathrm{d}z$ and τ_i can choose between 0 and $N-i$ for convolutional layer i.

Training. Overall, our AFDONet model is trained end-to-end by minimizing the loss function:

$$\begin{aligned}\mathcal{L}(\theta) = \; & \underbrace{\left|\left|u(x,\cdot) - \hat{u}_{N,\theta}(x,\cdot)\right|\right|^2_{\mathcal{H}(\mathcal{M})}}_{\text{reconstruction loss in RKHS}} + \underbrace{\left|\left|\tilde{u}(x,\cdot) - \mathrm{FM}(\tilde{u}(x,\cdot))\right|\right|^2_{H(\mathcal{M})}}_{\text{feature map loss}} \\ & + \underbrace{\omega \; \mathcal{D}_{\mathrm{KL}}\Big(\mathcal{CN}(\mu,\sigma^2) \;\Big\|\; \mathcal{CN}(0,I_r)\Big)}_{\text{latent space regularization}} + \underbrace{\sum_{i=0}^{k} w_i \; \left|\left|\nabla^i \hat{u}_{N,\theta}(x,\cdot) - \nabla^i u(x,\cdot)\right|\right|^2_{L^2(\mathcal{M})}}_{\text{holomorphic training loss}},\end{aligned} \tag{3.4.8}$$

where $\nabla^i u$ denotes the i-th covariant derivative defined on manifold $\mathcal{M}$. Notice that here, we extend the idea of Sobolev training Czarnecki et al. (2017a) to the complex domain and introduce a holomorphic training loss to enforce consistency with the ground truth solutions both at the function value level and across all orders of derivatives. This enables AFDONet to better capture the inherent smoothness and analytic structure of the target function.

3.5 Properties of AFDONet

The design of AFDONet architecture is fully guided by the AFD theory, making it mathematically interpretable in several aspects. Here, we list three important properties of AFDONet:

1. Under the loss function of Equation 3.4.8, we can rigorously bound the error of AFDONet in Theorem 3.5.1.

2. By extending the work of Caragea et al. (2022), we can rigorously prove the existence of RKHS $\mathcal{H}(\mathcal{M})$ through the construction of feature map $\mathrm{FM}(\cdot)$ in the latent-to-RKHS network in Theorem 3.5.2.

3. To ensure convergence of AFDONet, we leverage the convergence mechanism of AFD to design a convergent dynamic CKN decoder by regulating the layer width, depth, and kernel complexity based on the number of samples and the intrinsic smoothness of the target function.

3.5.1 Main theorems

Theorem 3.5.1 *Let $\mathcal{P} \subset \mathbb{R}^d$ be compact and $\{(p_i, u_i)\}_{i=1}^Z$ be Z i.i.d. samples with $u_i = F(p_i) + \xi_i$, $\xi_i \sim SubGaussian(\mathcal{H}(\mathcal{M}))$, and $\mathbb{E}[\xi_i] = 0$, where $F : \mathcal{P} \to \mathcal{H}(\mathcal{M})$ is holomorphic, and $\mathcal{H}(\mathcal{M})$ is an RKHS with a kernel k_m whose eigenvalues decay polynomially with rate k. Suppose $L_d = \mathcal{O}(\log Z)$ and $W_d = \mathcal{O}(Z^{\frac{1}{2(k+1)}})$ in the decoder network. For the minimizer $\hat{\theta}$ of the loss function $\mathcal{L}(\theta)$ in Equation 3.4.8, there exists a constant $C > 0$ such that:*

$$\mathbb{E}\left[\left\|\hat{u}_{N,\hat{\theta}} - F\right\|^2_{\mathcal{H}(\mathcal{M})}\right] \leq C Z^{-\frac{2k+1}{2(k+1)}} (\log Z)^2.$$

Theorem 3.5.2 *Let H be a Hilbert space on a manifold $\mathcal{M}$. Fix $d, n \in \mathbb{N}$, then for any $\widetilde{x} \in H(\mathcal{M})$ and any $\varepsilon > 0$, there exist a convolutional kernel K defining an RKHS $\mathcal{H}(\mathcal{M})$ and a complex-valued modReLU neural network $\mathrm{FM}_{\theta'}$ with at most $C \ln(2/\varepsilon)$ layers, $C\eta^{-2d/n} \ln^2(2/\varepsilon)$ weights, and weights bounded by $C\varepsilon^{-44d}$ such that*

$$\mathrm{FM}_{\theta'}(\widetilde{x}) \in \mathcal{H}(\mathcal{M}) \quad \textit{and} \quad \|\widetilde{x} - \mathrm{FM}_{\theta'}(\widetilde{x})\|_{H(\mathcal{M})} \leq \inf_{\theta} \|\widetilde{x} - \mathrm{FM}_{\theta}(\widetilde{x})\|_{H(\mathcal{M})} + \varepsilon,$$

where $C = C(d, n) > 0$ depends only on the dimension d and the smoothness parameter n.

Theorem 3.5.3 *Let L_d, W_d, and N denote the depth, width, and number of layers of dynamic CKN decoder network satisfying Equation 3.4.6. For any $\varepsilon > 0$, there exist $L_d = \mathcal{O}\big(\log \frac{1}{\varepsilon}\big), W_d = \mathcal{O}\big(\varepsilon^{-\frac{1}{k+1}}\big), N = \mathcal{O}\big(\log \frac{1}{\varepsilon}\big)$ and $\theta \in \mathcal{N}_{L_d, W_d, N}$ such that*

$$\sup_{p \in \mathcal{P}} \|\hat{u}_{N,\theta} - F(p)\|_{\mathcal{H}(\mathcal{M})} \;\leq\; \varepsilon,$$

where $\mathcal{N}_{L_d, W_d, N}$ is the class of complex-analytic networks with depth L_d and width W_d.

3.6 Proof of Theorem 3.5.1

We introduce and prove a few lemmas before proving Theorem 3.5.1. We assume that the neural network f_θ is Lipschitz continuous with respect to hyperparameters θ (i.e., $\|f_\theta - f_{\theta'}\|_{\mathcal{H}} \leq L_f \|\theta - \theta'\|_2$).

Lemma 3.6.1 *For any $0 < \delta < 1$, for the class of complex-analytic networks with depth L_d and width W_d, denoted as $\mathcal{N}_{L_d, W_d, N}$, there exists $\dot{C} > 0$ such that:*

$$\log \mathcal{N}\left(\delta, \mathcal{N}_{L_d, W_d, N}, \|\cdot\|_{\mathcal{H}}\right) \leq \dot{C} W_d L_d \log\left(\frac{W_d L_d}{\delta}\right),$$

where $\mathcal{N}\left(\delta, \mathcal{N}_{L_d, W_d, N}, \|\cdot\|_{\mathcal{H}}\right)$ means the δ-covering number of $(\mathcal{N}_{L_d, W_d, N}, \|\cdot\|_{\mathcal{H}})$.

Proof. Let us consider the p-dimensional ℓ_2-unit ball $\mathcal{B}^p(1) = \{x \in \mathbb{R}^p : \|x\|_2 \leq 1\}$. Results for covering $\mathcal{B}^p$ Wainwright (2019) concludes:

$$\log \mathcal{N}(\delta, \mathcal{B}^p(1), \|\cdot\|_2) \leq p \log\left(1 + \frac{2}{\delta}\right) \leq p \log\left(\frac{3}{\delta}\right). \tag{3.6.1}$$

Extending this result to a ℓ_2-ball of radius R, Equation 3.6.1 becomes:

$$\log \mathcal{N}(\delta, \mathcal{B}^p(R), \|\cdot\|_2) \leq p \log\left(1 + \frac{2R}{\delta}\right) \leq p \log\left(\frac{3R}{\delta}\right) \tag{3.6.2}$$

by rescaling δ in the RHS of Equation 3.6.1 with δ/R. Furthermore, by letting $p = 2W_d L_d$, Equation 3.6.2 becomes:

$$\log \mathcal{N}(\delta, \mathcal{B}^{2W_d L_d}(R), \|\cdot\|_2) \leq 2W_d L_d \log\left(\frac{3R}{\delta}\right). \tag{3.6.3}$$

From the Lipschitz property and the fact that the parameter space of $\mathcal{N}_{L_d,W_d,N}$ can be controlled by $\mathcal{B}^{2W_dL_d}(R)$, we have:

$$\log \mathcal{N}\left(\delta, \mathcal{N}_{L_d,W_d,N}, \|\cdot\|_{\mathcal{H}}\right) \leq \log \mathcal{N}\left(\frac{\delta}{L_f}, \mathcal{B}^{2W_dL_d}(R), \|\cdot\|_2\right) \leq 2W_dL_d \log\left(\frac{3L_fR}{\delta}\right), \quad (3.6.4)$$

where L_f is the Lipschitz constant. With $R = \mathcal{O}(W_dL_d)$, Equation 3.6.4 leads to:

$$\log \mathcal{N}\left(\delta, \mathcal{N}_{L_d,W_d,N}, \|\cdot\|_{\mathcal{H}}\right) \leq \dot{C}W_dL_d \log\left(\frac{W_dL_d}{\delta}\right), \quad (3.6.5)$$

which completes the proof. ■

Lemma 3.6.2 *For $a > 1$ and $0 < r \leq \min(a, e)$ where e is the base of the natural logarithm, there exists $b > 0$ that satisfies the following inequality:*

$$r\sqrt{\log\left(\frac{a}{r}\right)} \leq \sqrt{b}\sqrt{r \log a}.$$

Proof. For the case $1 < r \leq \min(a, e)$, we may choose $b = e$. Squaring both sides of the inequality and rearranging lead to $(r - e)\log a \leq r \log r$. Suppose $r = e$, the inequality is automatically satisfied for any $a > 1$. Suppose $r < e$, since $a \geq r$, we have: $(r - e)\log a \leq (r - e)\log r$. Thus, it suffices to show $(r - e)\log r \leq r \log r$, which is equivalent to showing $e \log r \geq 0$. This is automatically satisfied because $0 < \log r \leq 1$.

For the case $0 < r \leq 1$, we rearrange the inequality and obtain $b \geq \frac{r(\log a - \log r)}{\log a} > 0$. Furthermore, $\frac{r(\log a - \log r)}{\log a}$ reaches its maximum, $\frac{a}{e \log a}$, at $r = \frac{a}{e}$. Thus, suppose $a \leq e$, then we may choose $b \geq \frac{a}{e \log a}$ and the inequality is satisfied. Suppose $a \geq e$, then $\max \frac{r(\log a - \log r)}{\log a} = 1$ within $0 < r \leq 1$. Thus, we may choose $b \geq 1$ and the inequality is satisfied. ■

Lemma 3.6.3 *There exists $\widetilde{C} > 0$ such that:*

$$\mathbb{E}_\epsilon\left[\sup_{f \in \mathscr{F}}\left|\frac{1}{Z}\sum_{i=1}^{Z}\epsilon_i f_\theta(p_i)\right|\right] \leq \widetilde{C}\sqrt{\frac{rW_dL_d \log(W_dL_d)}{Z}},$$

where ϵ_i are i.i.d. Rademacher variables and $\mathscr{F}$ is a function class for a radius $0 < r \leq e$ defined as $\{f \in \mathcal{N}_{L_d,W_d,N} : \|f - F\|_{\mathcal{H}} \leq r\}$.

Proof. From Dudley's entropy integral bound Wainwright (2019), we have:

$$\mathbb{E}_\epsilon\left[\sup_{f\in\mathscr{F}}\left|\frac{1}{Z}\sum_{i=1}^{Z}\epsilon_i f(p_i)\right|\right] \leq \frac{24}{\sqrt{Z}}\int_\varepsilon^{2r}\sqrt{\log\mathcal{N}\left(t,\mathscr{F},\|\cdot\|_{\mathcal{H}}\right)}dt. \tag{3.6.6}$$

Since $\mathcal{N}(\delta,\mathscr{F},\|\cdot\|_{\mathcal{H}}) \leq \mathcal{N}(\delta,\mathcal{N}_{L_d,W_d,N},\|\cdot\|_{\mathcal{H}})$ and according to Lemma 3.6.1, Equation 3.6.6 becomes:

$$\begin{aligned}\mathbb{E}_\epsilon\left[\sup_{f\in\mathscr{F}}\left|\frac{1}{Z}\sum_{i=1}^{Z}\epsilon_i f(p_i)\right|\right] &\leq \frac{24}{\sqrt{Z}}\int_\varepsilon^{2r}\sqrt{\log\mathcal{N}\left(t,\mathcal{N}_{L_d,W_d,N},\|\cdot\|_{\mathcal{H}}\right)}\,dt \\ &\leq \frac{24}{\sqrt{Z}}\int_\varepsilon^{2r}\sqrt{\dot{C}W_dL_d\log\left(\frac{W_dL_d}{t}\right)}\,dt.\end{aligned} \tag{3.6.7}$$

To evaluate the integral on the RHS of Equation 3.6.7, we apply the change of variables technique by defining $u = \log\left(\frac{W_dL_d}{t}\right)$ (and thus $dt = -W_dL_de^{-u}du$):

$$\begin{aligned}\int_\varepsilon^{2r}\sqrt{\log\left(\frac{W_dL_d}{t}\right)}dt &= \int_{\log(\frac{W_dL_d}{2r})}^{\log(\frac{W_dL_d}{\varepsilon})}\sqrt{u}\cdot W_dL_de^{-u}\,du \\ &= W_dL_d\left[\Gamma\left(\frac{3}{2},\log\left(\frac{W_dL_d}{2r}\right)\right)-\Gamma\left(\frac{3}{2},\log\left(\frac{W_dL_d}{\varepsilon}\right)\right)\right] \\ &= 2r\sqrt{\log\left(\frac{W_dL_d}{2r}\right)}+\mathcal{O}\left(\frac{r}{\log(\frac{W_dL_d}{2r})}\right),\end{aligned} \tag{3.6.8}$$

where $\Gamma(s,x)=\int_x^\infty t^{s-1}e^{-t}dt$ is the upper incomplete gamma function.

Substituting Equation 3.6.8 into Equation 3.6.7 and applying Lemma 3.6.2 lead to:

$$\begin{aligned}\mathbb{E}_\epsilon\left[\sup_{f\in\mathscr{F}}\left|\frac{1}{Z}\sum_{i=1}^{Z}\epsilon_i f(p_i)\right|\right] &\leq 24\cdot 2r\sqrt{\frac{\dot{C}W_dL_d\log\left(\frac{W_dL_d}{2r}\right)}{Z}} \\ &\leq 24\sqrt{b}\sqrt{\frac{2r\dot{C}W_dL_d\log\left(W_dL_d\right)}{Z}} \\ &\leq \widetilde{C}\sqrt{\frac{rW_dL_d\log\left(W_dL_d\right)}{Z}},\end{aligned} \tag{3.6.9}$$

where $\widetilde{C}\geq 24\sqrt{2b\dot{C}}$. ■

Lemma 3.6.4 *Let $\hat{\theta}$ minimize the loss function $\mathcal{L}$ in Equation 3.4.8. With probability at least $1-e^{-t}$ for all $t\geq 0$,*

$$\mathcal{L}(\hat{\theta}) \leq \inf_\theta \mathcal{L}(\theta) + \hat{C}\frac{W_dL_d\log(W_dL_d)+t}{Z}$$

holds for some $\hat{C}$.

Proof. From the symmetrization inequality Boucheron et al. (2012), we have:

$$\mathbb{E}\left[\mathcal{L}(\hat{\theta}) - \mathcal{L}(\theta)\right] \leq 2\mathbb{E}\left[\sup_{f\in\mathscr{F}} \frac{1}{Z}\sum_{i=1}^{Z} \epsilon_i f(p_i)\right], \tag{3.6.10}$$

where ϵ_i are i.i.d. Rademacher variables.

Let us define the centered process:

$$\mathscr{Z} = \sup_{f\in\mathscr{F}} \sum_{i=1}^{Z} \left(f(p_i) - \mathbb{E}[f(p_i)]\right) \tag{3.6.11}$$

under the assumptions that there exists $\mathscr{Z}'_k$ such that: (i) $\mathscr{Z}'_k \leq \mathscr{Z} - \mathscr{Z}_k \leq 1$ almost surely; (ii) $\mathbb{E}^k[\mathscr{Z}'_k] \geq 0$, where $\mathbb{E}^k$ is the expectation taken conditionally to the sigma field generated by $(p_1, \ldots, p_{k-1}, p_{k+1}, \ldots p_Z)$; and (iii) there exists $q > 0$ such that $\mathscr{Z}'_k \leq q$ almost surely. Here, $\mathscr{Z}_k = \sup_{f\in\mathscr{F}} \sum_{i\neq k} \left(f(p_i) - \mathbb{E}[f(p_i)]\right)$.

Applying Bennett concentration inequality Bousquet (2002) to the process $\mathscr{Z}$ leads to:

$$\mathbb{P}\left(\mathscr{Z} \geq \mathbb{E}[\mathscr{Z}] + \sqrt{2vt} + \frac{t}{3}\right) \leq e^{-t}, \tag{3.6.12}$$

where $v = (1+q)\mathbb{E}[\mathscr{Z}] + Z\sigma^2$ and $\sigma^2 \geq \frac{1}{\mathscr{Z}} \sum_{k=1}^{Z} \mathbb{E}^k\left[(\mathscr{Z}'_k)^2\right]$.

Combining Equations 3.6.10, 3.6.12 and 3.6.12 with probability at least $1 - e^{-t}$, we have:

$$\mathcal{L}(\hat{\theta}) - \mathcal{L}(\theta) \leq 2\mathbb{E}\left[\sup_{f\in\mathscr{F}} \frac{1}{Z}\sum_{i=1}^{Z} \epsilon_i f(p_i)\right] + \frac{1}{Z}\left(\sqrt{2vt} + \frac{t}{3}\right). \tag{3.6.13}$$

Moreover, by putting $\mathbb{E}_\epsilon\left[\sup_{f\in\mathscr{F}} \left|\frac{1}{Z}\sum_{i=1}^{Z} \epsilon_i f(p_i)\right|\right] \asymp r$ for Lemma 3.6.3 ($\asymp$ stands for asymptotic equivalence), we obtain:

$$r \asymp \frac{W_d L_d \log(W_d L_d)}{Z}. \tag{3.6.14}$$

Extending the result of Equation 3.6.10 to $\mathscr{Z}$ defined in Equation 3.6.11 leads to:

$$\begin{aligned} \mathbb{E}[\mathscr{Z}] &\leq 2Z\mathbb{E}\left[\sup_{f\in\mathscr{F}} \frac{1}{Z}\sum_{i=1}^{Z} \epsilon_i f(p_i)\right] \\ &\leq 2Z\widetilde{C}\sqrt{\frac{rW_d L_d \log(W_d L_d)}{Z}} \asymp 2\widetilde{C} W_d L_d \log(W_d L_d), \end{aligned} \tag{3.6.15}$$

where the second inequality and last asymptotic equivalence come from Lemma 3.6.3 and Equation 3.6.14, respectively.

According to Efron-Stein inequality Boucheron et al. (2012), there exists $\mathscr{Z}'_k = \mathscr{Z} - \mathscr{Z}_k$, such that:

$$\sigma^2 \leq \sum_{k=1}^{Z} \mathbb{E}\left[(\mathscr{Z} - \mathbb{E}[\mathscr{Z} \mid p_k])^2\right] \leq \mathbb{E}^k[(\mathscr{Z}'_k)^2], \tag{3.6.16}$$

where $\mathscr{Z} \mid p_k$ excludes p_k from $\mathscr{Z}$. Thus, to derive an upper bound on $\mathbb{E}^k[(\mathscr{Z}'_k)^2]$, we write:

$$(\mathscr{Z}'_k)^2 \leq \left(\sup_{f\in\mathscr{F}} |f(p_k) - \mathbb{E}[f(p_k)]|\right)^2 \leq 2\left(\sup_{f\in\mathscr{F}} f(p_k)^2 + \mathbb{E}[f(p_k)]^2\right) \leq 4 \sup_{f\in\mathscr{F}} f(p_k)^2, \tag{3.6.17}$$

where the second inequality comes from $(a-b)^2 \leq 2(a^2+b^2)$ and the last inequality holds by Jensen's inequality ($\mathbb{E}[f(p_k)]^2 \leq \mathbb{E}[f(p_k)^2]$). Then, for $f \in \mathscr{F}$ and a bounded function F, it follows:

$$\mathbb{E}[f(p_k)^2] \leq 2(\|f - F\|^2_{\mathcal{H}} + \|F\|^2_{\mathcal{H}}) \leq 2(r^2 + \|F\|^2_{\mathcal{H}}). \tag{3.6.18}$$

Substituting the result of Equation 3.6.18 into Equation 3.6.17 and combining it with Equation 3.6.16 give:

$$\sigma^2 \leq Dr^2 \asymp \left(\frac{W_d L_d \log(W_d L_d)}{Z}\right)^2, \tag{3.6.19}$$

for some $D > 0$.

Substituting Equations 3.6.19 and 3.6.15 into 3.6.12 gives:

$$\begin{aligned} v &= (1+q)\mathbb{E}[\mathscr{Z}] + Z\sigma^2 \leq C'(1+q)W_d L_d \log(W_d L_d) + \frac{(W_d L_d \log(W_d L_d))^2}{Z} \\ &\leq \left(C'(1+q) + \frac{1}{Z}\right)(W_d L_d \log(W_d L_d))^2. \end{aligned} \tag{3.6.20}$$

Substituting Equations 3.6.20 and 3.6.14 into 3.6.13 gives:

$$\begin{aligned} \mathcal{L}(\hat{\theta}) &\leq \mathcal{L}(\theta) + 2\widetilde{C}\frac{W_d L_d \log(W_d L_d)}{Z} + \sqrt{2\left[C'(1+q) + \frac{1}{Z}\right]t}\frac{W_d L_d \log(W_d L_d)}{Z} + \frac{t}{3Z} \\ &\leq \mathcal{L}(\theta) + \hat{C}\frac{W_d L_d \log(W_d L_d) + t}{Z} \end{aligned} \tag{3.6.21}$$

holds for any θ, where $\hat{C} = \max\left\{2\widetilde{C}, \sqrt{2\left[C'(1+q) + \frac{1}{Z}\right]t}, \frac{1}{3}\right\}$. Thus, we conclude that $\mathcal{L}(\hat{\theta}) \leq \inf_\theta \mathcal{L}(\theta) + \hat{C}\frac{W_d L_d \log(W_d L_d)+t}{Z}$. ■

Proof of Theorem 3.5.1

Proof. From Lemma 3.6.4, we know that with probability at least $1 - e^{-t}$ for all $t \geq 0$ and some $\hat{C}$,

$$\mathcal{L}(\hat{\theta}) \leq \inf_{\theta} \mathcal{L}(\theta) + \hat{C}\frac{W_d L_d \log(W_d L_d) + t}{Z}. \tag{3.6.22}$$

Realizing $\mathcal{L}(\theta) \asymp \|\hat{u}_{N,\theta} - F\|_{\mathcal{H}}^2$, then for $s_0 = \inf_\theta \mathcal{L}(\theta) + \hat{C}\frac{W_d L_d \log(W_d L_D) + t_0}{Z}$, it holds that:

$$\begin{aligned}
\mathbb{E}[\mathcal{L}(\hat{\theta})] &\leq \int_0^\infty \mathbb{P}(\mathcal{L}(\hat{\theta}) \geq s) ds \\
&= \int_0^{s_0} \mathbb{P}(\mathcal{L}(\hat{\theta}) \geq s) ds + \int_{s_0}^\infty \mathbb{P}(\mathcal{L}(\hat{\theta}) \geq s) ds \\
&\leq s_0 + M \cdot e^{-t_0} \\
&= s_0 + \frac{M}{Z},
\end{aligned} \tag{3.6.23}$$

where $t_0 = \log Z$ and we assume that $\mathcal{L} \leq M$ for $t > t_0$.

Since $L_d = \mathcal{O}(\log Z)$ and $W_d = \mathcal{O}(Z^{\frac{1}{2(k+1)}})$, we have:

$$\begin{aligned}
\frac{W_d L_d \log(W_d L_d)}{Z} &\asymp \frac{Z^{\frac{1}{2(k+1)}} \cdot \log Z \cdot \log(Z^{\frac{1}{2(k+1)}} \log Z)}{Z} \\
&= \frac{Z^{\frac{1}{2(k+1)}} \cdot \log Z \cdot \left(\frac{1}{2(k+1)} \log Z + \log\log Z\right)}{Z} \\
&\asymp Z^{\frac{1}{2(k+1)} - 1} \cdot \log Z \cdot \log Z \\
&= Z^{-\frac{2k+1}{2(k+1)}} (\log Z)^2.
\end{aligned} \tag{3.6.24}$$

Combining Equations 3.6.22, 3.6.23 and 3.6.24 leads to the final result:

$$\mathbb{E}\left[\left\|\hat{u}_{N,\hat{\theta}} - F\right\|_{\mathcal{H}}^2\right] \leq C Z^{-\frac{2k+1}{2(k+1)}} (\log Z)^2 + \mathcal{O}(Z^{-1}), \tag{3.6.25}$$

where $C > 0$ is a constant and the term $\mathcal{O}(Z^{-1})$ vanishes for a large Z. ∎

3.7 Proof of Theorem 3.5.2

Proof. First, we show that $\mathcal{H}(\mathcal{M})$ exists by introducing a map $\Phi : H(\mathcal{M}) \to \mathcal{H}(\mathcal{M})$ and the reproducing kernel is defined as $K(x, x') = \langle \Phi(x), \Phi(x') \rangle_{\mathcal{H}(\mathcal{M})}$. Specifically, the map $\Phi(x)$ corresponding to a convolutional kernel K can be represented as $\mathcal{A}_L \circ \mathcal{M}_L \circ \mathcal{P}_L \cdots \mathcal{A}_1 \circ \mathcal{M}_1 \circ \mathcal{P}_1 x$

where L is the depth of the kernel and $\mathcal{A}_l, \mathcal{M}_l$ and $\mathcal{P}_l$ are the linear operators related to pooling, kernel mapping and patch extraction, respectively Bietti (2022). Without loss of generality, we assume that $\mathcal{H}(\mathcal{M}) \subset H(\mathcal{M})$. Next, we point out that $\mathcal{H}(\mathcal{M})$ is convex by showing that, for any two functions $f, g \in \mathcal{H}(\mathcal{M})$:

$$\begin{aligned} \alpha f + (1-\alpha)g = \alpha \langle f, \mathcal{A}_L \circ \mathcal{M}_L \circ \mathcal{P}_L \cdots \mathcal{A}_1 \circ \mathcal{M}_1 \circ \mathcal{P}_1 x \rangle_{\mathcal{H}(\mathcal{M})} + (1-\alpha) \\ \langle g, \mathcal{A}_L \circ \mathcal{M}_L \circ \mathcal{P}_L \cdots \mathcal{A}_1 \circ \mathcal{M}_1 \circ \mathcal{P}_1 x \rangle_{\mathcal{H}(\mathcal{M})} \\ = \langle \alpha f + (1-\alpha)g, \mathcal{A}_L \circ \mathcal{M}_L \circ \mathcal{P}_L \cdots \mathcal{A}_1 \circ \mathcal{M}_1 \circ \mathcal{P}_1 x \rangle_{\mathcal{H}(\mathcal{M})} \end{aligned} \tag{3.7.1}$$

for $\alpha \in [0,1]$. Thus, $\mathcal{H}(\mathcal{M})$ is closed due to the closedness of manifold $\mathcal{M}$ and the completeness of Hilbert space $\mathcal{H}$.

Next, from the Hilbert projection theorem, for $\widetilde{x} \in H(\mathcal{M})$, there exists a unique $y \in \mathcal{H}(\mathcal{M})$ such that, for any $\widetilde{y} \in \mathcal{H}(\mathcal{M})$, $||\widetilde{x} - y||_{H(\mathcal{M})} \leq ||\widetilde{x} - \widetilde{y}||_{H(\mathcal{M})}$. Let us denote y as $\Psi(\widetilde{x})$, where Ψ is a map from $H(\mathcal{M})$ to $\mathcal{H}(\mathcal{M})$. Following the main result of Caragea et al. (2022), for any $\widetilde{y} \in \mathcal{H}(\mathcal{M})$ and any $\varepsilon > 0$, there exists a complex-valued modReLU neural network with hyperparameters θ, FM_θ, containing no more than $C\ln(2/\varepsilon)$ layers, $C\eta^{-2d/n}\ln^2(2/\varepsilon)$ weights (all weights bounded by $C\varepsilon^{-44d}$), such that $||\widetilde{y} - \mathrm{FM}_\theta(\widetilde{x})||_{H(\mathcal{M})} < \frac{\varepsilon}{2}$. In addition, there also exists another complex-valued modReLU neural network with hyperparameters θ', $\mathrm{FM}_{\theta'}$, such that $||\Psi(\widetilde{x}) - \mathrm{FM}_{\theta'}(\widetilde{x})||_{H(\mathcal{M})} < \frac{\varepsilon}{2}$. Thus, we have:

$$\begin{aligned} ||\widetilde{x} - \mathrm{FM}_{\theta'}(\widetilde{x})||_{H(\mathcal{M})} &= ||\widetilde{x} - \Psi(\widetilde{x}) + \Psi(\widetilde{x}) - \mathrm{FM}_{\theta'}(\widetilde{x})||_{H(\mathcal{M})} \\ &\leq ||\widetilde{x} - \Psi(\widetilde{x})||_{H(\mathcal{M})} + ||\Psi(\widetilde{x}) - \mathrm{FM}_{\theta'}(\widetilde{x})||_{H(\mathcal{M})} \\ &\leq ||\widetilde{x} - \widetilde{y}||_{H(\mathcal{M})} + \frac{\varepsilon}{2} \\ &= ||\widetilde{x} - \widetilde{y} + \mathrm{FM}_\theta(\widetilde{x}) - \mathrm{FM}_\theta(\widetilde{x})||_{H(\mathcal{M})} + \frac{\varepsilon}{2} \\ &\leq ||\widetilde{x} - \mathrm{FM}_\theta(\widetilde{x})||_{H(\mathcal{M})} + ||\mathrm{FM}_\theta(\widetilde{x}) - \widetilde{y}||_{H(\mathcal{M})} + \frac{\varepsilon}{2} \\ &\leq ||\widetilde{x} - \mathrm{FM}_\theta(\widetilde{x})||_{H(\mathcal{M})} + \varepsilon. \end{aligned} \tag{3.7.2}$$

This completes the proof. ■

3.8 Proof of Theorem 3.5.3

To prove Theorem 3.5.3, we first introduce and/or prove a few lemmas.

Lemma 3.8.1 (Yarotsky (2017)) *For any dimension n, smoothness parameter $k+1$, and error tolerance $\varepsilon \in (0,1)$, there exists a ReLU neural network architecture such that it can approximate any function f with accuracy ε, i.e., with approximation error at most ε. The network has depth at most $c(\ln(1/\varepsilon)+1)$, and uses at most $c\varepsilon^{-\frac{d}{n}}(\ln(1/\varepsilon)+1)$ weights and computation units, where $c = c(d,n)$ is a constant depending only on d and n.*

Lemma 3.8.2 *Let $f \in C^k([0,1]^d)$ or $W^{k+1,\infty}([0,1]^d)$, for $\varepsilon > 0$, there exists a ReLU network f_θ with width $W_d = \mathcal{O}\left(\varepsilon^{-\frac{d}{k+1}}\right)$ such that $\|f - f_\theta\|_{L^\infty} \leq \varepsilon$.*

Proof. The result follows from Lemma 3.8.1, which states that for any $d \in \mathbb{N}$, $n \in \mathbb{N}$, and $\varepsilon \in (0,1)$, there exists a ReLU neural network of depth $\mathcal{O}(\log(1/\varepsilon))$ and size $\mathcal{O}(\varepsilon^{-\frac{d}{n}}\log(1/\varepsilon))$ that can uniformly approximate any function in the class $F_{d,n}$, which includes functions in $W^{n,\infty}([0,1]^d)$ with bounded norm. By setting $n = k+1$, it holds that $f \in W^{k+1,\infty}([0,1]^d)$, with the network width scaling as $\mathcal{O}(\varepsilon^{-\frac{d}{k+1}})$, up to a logarithmic factor. Note that any $f \in C^k([0,1]^d)$ with bounded derivatives up to order k also belongs to $W^{k,\infty}([0,1]^d)$ and can be embedded into $W^{k+1,\infty}$. Thus, Lemma 3.8.2 holds for any $f \in C^k([0,1]^d)$. ■

Remark 3.8.1 *The result of Lemma 3.8.2 is nearly optimal. (Yarotsky, 2017, Theorem 5) shows that there exist functions $f \in W^{n,\infty}([0,1]^d)$ for which the complexity $N(f,\varepsilon)$ is not $o(\varepsilon^{-\frac{d}{9n}})$ as $\varepsilon \to 0$. This implies that no network architecture can uniformly approximate all such functions with significantly better scaling in ε.*

Lemma 3.8.3 *Let $\mathcal{H}$ be a separable Hilbert space and $f \in \mathcal{H}$ belong to a class of functions with k-th order smoothness. For $\varepsilon > 0$, there exists a ReLU network f_θ with width $W_d = \mathcal{O}\left(\varepsilon^{-\frac{d}{k+1}}\right)$ such that $\|f - f_\theta\|_{\mathcal{H}} \leq \varepsilon$.*

Proof. Assume $f \in \mathrm{dom}(A^{-k})$ with respect to its operator A with input dimension d. Let $\{e_j\}_{j=1}^{\infty}$ be an orthonormal basis of $\mathcal{H}$ with associated eigenvalues $\lambda_j \asymp j^{2\alpha}$ (assuming that

$\alpha \geq \frac{k+1}{2dk}$) of A. Then, we have $\|A^k f\|_{\mathcal{H}}^2 = \sum_{j=1}^{\infty} \lambda_j^{2k} |\langle f, e_j \rangle|^2 < \infty$. We can define the eigenexpansion of f as $P_N f = \sum_{j=1}^{N} \langle f, e_j \rangle e_j$ and $\|f - P_N f\|_{\mathcal{H}} \leq C N^{-(k+\frac{1}{2})\alpha} \leq \varepsilon/2$ holds for $N = \lceil \varepsilon^{-\frac{1}{2\alpha k+\alpha}} \rceil \asymp \varepsilon^{-\frac{1}{2k\alpha}}$. In the finite-dimensional subspace $\text{span}\{e_1, ..., e_N\} \cong \mathbb{R}^N$, each coordinate function $f_j = \langle f, e_j \rangle$ inherits C^k regularity and can be approximated by a ReLU network $\tilde{f}_j$ with $|\tilde{f}_j(x) - f_j(x)| \leq \frac{\varepsilon}{2\sqrt{N}}$ using width $\mathcal{O}(\varepsilon^{-\frac{d}{k+1}})$ per coordinate from Lemma 3.8.2. The RELU network $f_\theta = \sum_{j=1}^{N} \tilde{f}_j e_j$ then satisfies $\|f - f_\theta\|_{\mathcal{H}} \leq \|f - P_N f\|_{\mathcal{H}} + \sqrt{\sum_j \|\tilde{f}_j - f_j\|_{L^\infty}^2} \leq \varepsilon$. The total width $W_d = \mathcal{O}(N \cdot \varepsilon^{-\frac{d}{k+1}}) = \mathcal{O}(\varepsilon^{-\frac{d}{k+1}})$ ■

Proof of Theorem 3.5.3

Proof. First, we show that, for a sufficiently large N and any $\varepsilon > 0$,

$$\|\hat{u}_{N,\theta} - \text{FM}\,(\tilde{u})\,\|_{\mathcal{H}(\mathcal{M})} \leq \frac{\varepsilon}{4} \tag{3.8.1}$$

holds. From Equation 3.4.7, we have $\hat{u}_{N,\theta} = \sum_{i=1}^{N} \langle \text{FM}\,(\tilde{u})\,, \mathscr{B}_{i+\tau_i} \rangle \mathscr{B}_{i+\tau_i}$. Here, we prove by contradiction. Suppose $\|\hat{u}_{N,\theta} - \text{FM}\,(\tilde{u})\,\|_{\mathcal{H}(\mathcal{M})} > \frac{\varepsilon}{4}$, then there exists an open ball $\mathcal{B}$ and $C > 0$ such that:

$$\left\| \text{FM}\,(\tilde{u}(x,\cdot)) - \sum_{i=1}^{N} \langle \text{FM}\,(\tilde{u}(x,\cdot))\,, \mathscr{B}_{i+\tau_i} \rangle \mathscr{B}_{i+\tau_i} \right\|_{\mathcal{H}(\mathcal{M})} = C \max_{m,\xi} (\|k_m(\xi)\|) > \frac{\varepsilon}{4}, \tag{3.8.2}$$

for $(x, \cdot) \in \mathcal{B} \subset \mathcal{M}$. Furthermore, since the term $\sum_{i=1}^{N} \|\langle \text{FM}\,(\tilde{u}(x,\cdot))\,, \mathscr{B}_{i+\tau_i} \rangle\|_{\mathcal{H}(\mathcal{M})}^2 < \infty$ is finite, there exists N_0 such that for any $n \geq N_0$, we have:

$$\sum_{i=n}^{N} \|\langle \text{FM}\,(\tilde{u}(x,\cdot))\,, \mathscr{B}_{i+\tau_i} \rangle\|_{\mathcal{H}(\mathcal{M})}^2 < \left(\frac{\rho_0 C}{2} \right)^2. \tag{3.8.3}$$

Next, we examine the term $\|\langle u_n, \frac{k_b}{\|k_b\|} \rangle\|_{\mathcal{H}(\mathcal{M})}$, where $(x, b) \in \mathcal{B}$ and

$$\begin{aligned} u_n &= \text{FM}\,(\tilde{u}(x,\cdot)) - \sum_{i=1}^{n-1} \langle \text{FM}\,(\tilde{u}(x,\cdot))\,, \mathscr{B}_{i+\tau_i} \rangle \mathscr{B}_{i+\tau_i} \\ &= \text{FM}\,(\tilde{u}(x,\cdot)) - \sum_{i=1}^{N} \langle \text{FM}\,(\tilde{u}(x,\cdot))\,, \mathscr{B}_{i+\tau_i} \rangle \mathscr{B}_{i+\tau_i} + \sum_{i=n}^{N} \langle \text{FM}\,(\tilde{u}(x,\cdot))\,, \mathscr{B}_{i+\tau_i} \rangle \mathscr{B}_{i+\tau_i}. \end{aligned} \tag{3.8.4}$$

Therefore, we have:

$$
\begin{aligned}
&\left\|\langle u_n, \frac{k_b}{\|k_b\|}\rangle\right\|_{\mathcal{H}(\mathcal{M})} \\
&= \left\|\left\langle \mathrm{FM}\left(\tilde{u}\right) - \sum_{i=1}^{N}\langle \mathrm{FM}\left(\tilde{u}\right), \mathscr{B}_{i+\tau_i}\rangle \mathscr{B}_{i+\tau_i} + \sum_{i=n}^{N}\langle \mathrm{FM}\left(\tilde{u}\right), \mathscr{B}_{i+\tau_i}\rangle \mathscr{B}_{i+\tau_i}, \frac{k_b}{\|k_b\|}\right\rangle\right\|_{\mathcal{H}(\mathcal{M})} \\
&\geq \left\|\left\langle \mathrm{FM}\left(\tilde{u}\right) - \sum_{i=1}^{N}\langle \mathrm{FM}\left(\tilde{u}\right), \mathscr{B}_{i+\tau_i}\rangle \mathscr{B}_{i+\tau_i}, \frac{k_b}{\|k_b\|}\right\rangle\right\|_{\mathcal{H}(\mathcal{M})} \\
&- \left\|\left\langle \sum_{i=n}^{N}\langle \mathrm{FM}\left(\tilde{u}\right), \mathscr{B}_{i+\tau_i}\rangle \mathscr{B}_{i+\tau_i}, \frac{k_b}{\|k_b\|}\right\rangle\right\|_{\mathcal{H}(\mathcal{M})} \\
&\geq \left\|\frac{\left(\mathrm{FM}\left(\tilde{u}\right) - \sum_{i=1}^{N}\langle \mathrm{FM}\left(\tilde{u}\right), \mathscr{B}_{i+\tau_i}\rangle \mathscr{B}_{i+\tau_i}\right)\Big|_b}{\|k_b\|}\right\|_{\mathcal{H}(\mathcal{M})} \\
&- \sqrt{\sum_{i=n}^{N}\left\|\langle \mathrm{FM}\left(\tilde{u}(x,\cdot)\right), \mathscr{B}_{i+\tau_i}\rangle\right\|_{\mathcal{H}(\mathcal{M})}^2} \\
&\geq C - \frac{C}{2} = \frac{C}{2},
\end{aligned}
\tag{3.8.5}
$$

where the third inequality holds due to the reproducing property of RKHS: $\langle f, k_m\rangle = f(m)$.

Meanwhile, there exists $\gamma > 0$ satisfying Equation 3.4.6 such that:

$$
\begin{aligned}
\left\|\langle u_n, \frac{k_b}{\|k_b\|}\rangle\right\|_{\mathcal{H}(\mathcal{M})} &= \frac{\left\|\langle u_n, k_b - \sum_{i=1}^{n-1}\langle k_b, \mathscr{B}_{i+\tau_i}\rangle \mathscr{B}_{i+\tau_i}\rangle\right\|_{\mathcal{H}(\mathcal{M})}}{\|k_b\|} \\
&\leq \frac{\left\|\langle u_n, k_b - \sum_{i=1}^{n-1}\langle k_b, \mathscr{B}_{i+\tau_i}\rangle \mathscr{B}_{i+\tau_i}\rangle\right\|_{\mathcal{H}(\mathcal{M})}}{\left\|k_b - \sum_{i=1}^{n-1}\langle k_b, \mathscr{B}_{i+\tau_i}\rangle \mathscr{B}_{i+\tau_i}\right\|_{\mathcal{H}(\mathcal{M})}} \\
&= \left\|\langle u_n, \mathscr{B}^b_{n+\tau_n}\rangle\right\|_{\mathcal{H}(\mathcal{M})} \\
&\leq \left\|\frac{1}{\rho_0}\langle u_n, \mathscr{B}_{n+\tau_n}\rangle - \frac{\gamma}{\rho_0}\right\|_{\mathcal{H}(\mathcal{M})} \\
&\leq \frac{1}{\rho_0}\cdot\frac{\rho_0 C}{2} - \frac{\gamma}{\rho_0} \\
&< \frac{C}{2}.
\end{aligned}
\tag{3.8.6}
$$

Hence, Equations 3.8.5 and 3.8.6 lead to a contradiction. Therefore, Equation 3.8.1 must hold.

Next, from Theorem 3.5.2, there exists a network FM with appropriate hyperparameters

θ' such that:

$$\|\widetilde{u} - \mathrm{FM}_{\theta'}(\widetilde{u})\|_{H(\mathcal{M})} \leq \inf_{\theta} \|\widetilde{u} - \mathrm{FM}_{\theta}(\widetilde{u})\|_{H(\mathcal{M})} + \frac{\varepsilon}{4}. \tag{3.8.7}$$

Let us denote $\mathrm{FM}_{\theta'}$ as FM. Note that $\widetilde{u}$ in Equation 3.8.7 lies in the Hilbert space $H(\mathcal{M})$, not the RKHS $\mathcal{H}(\mathcal{M})$. Furthermore, from Lemma 3.8.1, there exists a set of hyperparameters $\widetilde{\theta}$ such that $\|\widetilde{u} - \mathrm{FM}_{\widetilde{\theta}}(\widetilde{u})\|_{H(\mathcal{M})} \leq \frac{\varepsilon}{4}$. Therefore, Equation 3.8.7 reduces to:

$$\begin{aligned} \|\widetilde{u} - \mathrm{FM}(\widetilde{u})\|_{H(\mathcal{M})} &\leq \inf_{\theta} \|\widetilde{u} - \mathrm{FM}_{\theta}(\widetilde{u})\|_{H(\mathcal{M})} + \frac{\varepsilon}{4} \\ &\leq \|\widetilde{u} - \mathrm{FM}_{\widetilde{\theta}}(\widetilde{u})\|_{H(\mathcal{M})} + \frac{\varepsilon}{4} \\ &\leq \frac{\varepsilon}{4} + \frac{\varepsilon}{4} = \frac{\varepsilon}{2}. \end{aligned} \tag{3.8.8}$$

From Lemma 3.8.3, for $\widetilde{u}$ which is the output of a neural network with width $W_d = \mathcal{O}\left(\varepsilon^{-\frac{d}{k+1}}\right)$, we have:

$$\|\widetilde{u} - F\|_{H(\mathcal{M})} \leq \frac{\varepsilon}{4}. \tag{3.8.9}$$

Putting Equations 3.8.1, 3.8.8 and 3.8.9 together leads to:

$$\begin{aligned} \|\hat{u}_{N,\theta} - F(p)\|_{\mathcal{H}(\mathcal{M})} &\leq \|\hat{u}_{N,\theta} - \mathrm{FM}\left(\tilde{u}\right)\|_{\mathcal{H}(\mathcal{M})} + \|\widetilde{u} - \mathrm{FM}(\widetilde{u})\|_{H(\mathcal{M})} + \|\widetilde{u} - F\|_{H(\mathcal{M})} \\ &\leq \varepsilon \end{aligned} \tag{3.8.10}$$

for any $p \in \mathcal{P}$. Therefore, taking supremum on LHS and RHS of Equation 3.8.10, we have proven Theorem 3.5.3. ■

3.9 Proof that the Helmholtz equation spans an RKHS

Let us consider the Helmholtz equation $\Delta_{\mathcal{M}} u + k^2 u = 0$ without loss of generality. We first introduce some background and preliminaries before proceeding with the proof.

Let $\Delta = \sum_{i=1}^{n} \frac{\partial^2}{\partial x_i^2}$ be the Euclidean Laplace operator acting on the Sobolev space of weakly twice differentiable functions defined on $\mathbb{R}^n$. Let $k > 0$ be a fixed constant. A function u defined on $\mathbb{R}^n$ is called a solution of the Helmholtz equation, if $\Delta u + k^2 u = 0$ on $\mathbb{R}^n$. In other words, u satisfies one of the following:

- $u \in C^2(\mathbb{R}^n)$ is a classical solution of the above equation on $\mathbb{R}^n$; or

- $u \in W^2(\mathbb{R}^n)$ is a solution in the weak L^2-sense, i.e., u is locally square integrable, and satisfies $\int_{\mathbb{R}^n} u(x) \, [\, \Delta\varphi(x) + k^2\varphi(x) \,] \, dx = 0$ for any (test) function $\varphi \in C^\infty(\mathbb{R}^n)$ with compact support.

It follows from Axler et al. (2001) that any solution of homogeneous Helmholtz equation is real analytic on $\mathbb{R}^n$. We define the following space:

$$W_{\text{Helm},k}(\mathbb{R}^n) = \{u \in C^\infty(\mathbb{R}^n) \mid \Delta u + k^2 u = 0 \text{ on } \mathbb{R}^n\}. \tag{3.9.1}$$

Hartman & Wilcox (1961) introduced the concept of Herglotz wave function. The Herglotz wave functions consists of all the entire solutions u of the homogeneous Helmholtz equation $\Delta u + k^2 u = 0$ on $\mathbb{R}^n$ with $k > 0$ such that Herglotz boundedness condition:

$$\lim_{R \to +\infty} \frac{1}{R} \int_{\|x\| < R} |u(x)|^2 \, dx < +\infty \tag{3.9.2}$$

holds. Hartman & Wilcox (1961) characterized the Herglotz wave functions as the entire solutions u of the homogeneous Helmholtz equation with far-field pattern in $L^2(\mathbb{S}^{n-1})$. That is, functions u defined on $\mathbb{R}^n$ can be written as:

$$u(x) = \int_{\mathbb{S}^{n-1}} e^{ik\langle x, \xi \rangle} g(\xi) \, d\sigma(\xi), \tag{3.9.3}$$

for some $g \in L^2(\mathbb{S}^{n-1})$.

With this, let us consider the Helmholtz equation on the standard n-dimensional unit sphere $\mathbb{S}^n = \{x \in \mathbb{R}^{n+1} : \|x\| = 1\}$ in $\mathbb{R}^{n+1}$ with canonical spherical Riemannian metric g. Let $\Delta_{\mathbb{S}^n}$ be the spherical Laplacian acting on the Sobolev space $W^2(\mathbb{S}^n)$ of real-valued, square-integrable, and twice weakly differentiable functions on $\mathbb{S}^n$. Consider the Helmholtz equation on the Riemannian manifold $(\mathbb{S}^{n-1}, g)$ with canonical spherical metric g. Its entire solution can be expressed as:

$$u = W\phi(x) = (2\pi)^{\frac{1-n}{2}} \int_{\mathbb{S}^{n-1}} e^{ikx \cdot \xi} \phi(\xi) d\sigma(\xi), \tag{3.9.4}$$

where W is the Fourier extension operator and $\phi \in L^2(\mathbb{S}^{n-1})$ is Herglotz wave function. It has been shown that W defined in Equation 3.9.4 is an isomorphism of $L^2(\mathbb{S}^{n-1})$ onto the

space W^2 consisting of all solutions of Helmholtz equation with radial and angular derivatives satisfying:

$$||u||^2 = \int_{|x|>1} (|u(x)|^2 + |\frac{\partial u}{\partial r}(x)|^2 + |\frac{\partial u}{\partial \theta}(x)|^2)\frac{dx}{|x|^3} < \infty, \tag{3.9.5}$$

(see Pérez-Esteva & Valenzuela-Díaz (2017)). In this sense, the space W^2 in $\mathbb{R}^2$ is a Hilbert space with reproducing kernel (i.e., RKHS).

Meanwhile, to the best of our knowledge, there exists no such formal analysis on Helmholtz equation on any smooth (Riemannian) manifold $(\mathcal{M}, g)$. In this case, the Laplace-Beltrami operator extends the Laplace operator to Riemannian manifold $(\mathcal{M}, g)$ and is defined as $\Delta_{\mathcal{M}} u := \text{div}_g(\nabla u)$, where ∇u denotes the gradient of u and div_g is the metric-induced divergence. In local coordinates $(x^1, \ldots, x^n)$, the operator takes the form:

$$\Delta_{\mathcal{M}} u = \frac{1}{\sqrt{|g|}} \sum_{i,j=1}^{n} \frac{\partial}{\partial x^i} \left(\sqrt{|g|}\, g^{ij} \frac{\partial u}{\partial x^j} \right), \tag{3.9.6}$$

where g_{ij} is the Riemannian metric tensor, g^{ij} is its inverse, and $|g|$ denotes the determinant of the metric matrix.

For any smooth manifold $(\mathcal{M}, g)$, the Laplace-Beltrami operator $\Delta_{\mathcal{M}}$, defined in Equation 3.9.6, has orthonormal eigenbases on $L^2(\partial\mathcal{M})$ as $\{\psi_\lambda\}_\lambda$ with corresponding eigenvalues $\lambda \geq 0$. For each ψ_λ, let us consider:

$$(\Delta_{\mathcal{M}} + k^2)\phi_\lambda = 0 \text{ in } \mathcal{M}, \quad \phi_\lambda|_{\partial\mathcal{M}} = \psi_\lambda. \tag{3.9.7}$$

By elliptic regularity, $\phi_\lambda \in H^2(\mathcal{M})$. Furthermore, we extend the Fourier extension operator in Equation 3.9.4 to $W_{\mathcal{M}}$ on any smooth manifold $\mathcal{M}$:

$$W_{\mathcal{M}} f(x) = \int_{\partial\mathcal{M}} \Psi(x, \xi) f(\xi) d\sigma(\xi), \quad \text{where } \Psi(x, \xi) = \sum_{\lambda} \phi_\lambda(x)\overline{\psi_\lambda(\xi)}. \tag{3.9.8}$$

Now, we present the main result in Theorem 3.9.1 that $W^2(\mathcal{M})$ is the space of all Herlotz wave functions.

Theorem 3.9.1 *The operator $W_{\mathcal{M}} : L^2(\partial\mathcal{M}) \to W^2(\mathcal{M})$ defined in Equation 3.9.8 is a topological isomorphism, where $W^2(\mathcal{M}) = \{u \in H^2(\mathcal{M}) : (\Delta_{\mathcal{M}} + k^2)u = 0\}$.*

Remark 3.9.1 *Theorem 3.9.1 implies that $W_{\mathcal{M}}$ is an isomorphism between $L^2(\partial\mathcal{M})$ and $W^2(\mathcal{M})$, the space of H^2-solutions to the Helmholtz equation $(\Delta_{\mathcal{M}} + k^2)u = 0$. Such an isomorphism $W_{\mathcal{M}}$ implies that $\mathcal{H}(\mathcal{M})$ inherits a Hilbert space or RKHS structure from $L^2(\partial\mathcal{M})$. In other words, $W^2(\mathcal{M})$ is an RKHS.*

To prove Theorem 3.9.1, we first introduce and prove a lemma.

Lemma 3.9.1 *Let $J_\nu(z)$ be the Bessel function of order $\nu \in \mathbb{R}$. For each eigenfunction ψ_j of $\Delta_{\partial\mathcal{M}}$, define $F_j = W_{\mathcal{M}}\psi_j$. Then:*

1. *$F_j(x) = (2\pi)^{1/2} i^{\nu(j)} r^{-\frac{n-2}{2}} J_{\nu(j)}(kr)\psi_j(\xi)$, where $x = r\xi$ in normal coordinates near $\partial\mathcal{M}$.*

2. *The family $\{F_j\}$ is orthogonal in $W^2(\mathcal{M})$, and*

$$\|F_j\|_{H^2(\mathcal{M})} = \sqrt{2} + \mathcal{O}\left(\frac{1}{\lambda_j}\right).$$

3. *For $f = \sum_j a_j\psi_j \in L^2(\partial\mathcal{M})$ and $u = \sum_j a_j F_j \in W^2(\mathcal{M})$,*

$$\|u\|_{H^2(\mathcal{M})} \sim \|f\|_{L^2(\partial\mathcal{M})},$$

with absolute and uniform convergence on compact subsets of $\mathcal{M}$.

Proof. We prove the three components of Lemma 3.9.1 as follows:

1. Helmholtz equation $(\Delta_{\mathcal{M}} + k^2)\phi_j = 0$ can be written as:

$$\left(\partial_r^2 + \frac{n-1}{r}\partial_r + \frac{1}{r^2}\Delta_{\partial\mathcal{M}} + k^2\right)(r^{-\frac{n-2}{2}} R_j(r)\psi_j(\xi)) = 0. \tag{3.9.9}$$

   Substituting $\phi_j = r^{-\frac{n-2}{2}} R_j(r)\psi_j(\xi)$ into Equation 3.9.9 yields:

$$R_j'' + \frac{1}{r}R_j' + \left(k^2 - \frac{\nu(j)^2}{r^2}\right)R_j = 0, \tag{3.9.10}$$

   whose solution is $R_j(r) = J_{\nu(j)}(kr)$. By the Funk-Hecke formula Xu (2000), we have:

$$F_j(x) = \int_{\partial\mathcal{M}} \Psi(x,\xi)\psi_j(\xi)d\sigma(\xi) = (2\pi)^{1/2} i^{\nu(j)} r^{-\frac{n-2}{2}} J_{\nu(j)}(kr)\psi_j(\xi). \tag{3.9.11}$$

2. Since ψ_j and ψ_k are orthonormal eigenbases, ψ_j and ψ_k are orthogonal on $\partial\mathcal{M}$. Therefore,

$$\langle F_j, F_k\rangle_{H^2(\mathcal{M})} = \int_M \left(\phi_j\overline{\phi_k} + \nabla\phi_j\cdot\overline{\nabla\phi_k}\right) dV_g = 0 \tag{3.9.12}$$

for any $j \neq k$. Using the asymptotic $J_{\nu(j)}(kr) \sim \frac{(kr/2)^{\nu(j)}}{\Gamma(\nu(j)+1)}$ for $r \to 0^+$ and oscillatory decay for $r \to \infty$, we have:

$$\|F_j\|^2_{H^2(\mathcal{M})} = 2 + \mathcal{O}\left(\frac{1}{\lambda_j}\right),$$

where the error term comes from the next-order Bessel asymptotics.

3. From Part 2, the map $f \mapsto u$ is bounded:

$$\|u\|^2_{H^2(\mathcal{M})} = \sum_j |a_j|^2\|F_j\|^2_{H^2(\mathcal{M})} \sim \sum_j |a_j|^2 = \|f\|^2_{L^2(\partial\mathcal{M})}. \tag{3.9.13}$$

Next, we prove $|J_\nu(kr)| \sim \mathcal{O}(\nu^{-1/2})$ uniformly holds on compact subsets $K \subset \mathcal{M}$. According to Watson (1922), we have:

$$\begin{aligned} J_\nu(\nu\sec\beta) \sim \left(\frac{2}{\pi\nu\tan\beta}\right)^{1/2} \Bigg[&\cos\left(\nu\tan\beta - \nu\beta - \frac{\pi}{4}\right)\sum_{m=0}^{\infty}\frac{(-1)^m\Gamma(2m+\frac{1}{2})}{\Gamma(\frac{1}{2})}\cdot \\ &\frac{A_{2m}}{(\frac{1}{2}\nu\tan\beta)^{2m}} + \sin\left(\nu\tan\beta - \nu\beta - \frac{\pi}{4}\right)\sum_{m=0}^{\infty}\frac{(-1)^m\Gamma(2m+\frac{3}{2})}{\Gamma(\frac{1}{2})}\cdot \\ &\frac{A_{2m+1}}{(\frac{1}{2}\nu\tan\beta)^{2m+1}}\Bigg], \end{aligned} \tag{3.9.14}$$

where A_k is defined following $A_0 = 1$, $A_1 = \frac{1}{3} + \frac{5}{24}\cot^2\beta$, $A_2 = \frac{3}{128} + \frac{77}{576}\cot^2\beta + \frac{385}{3456}\cot^4\beta$, and so on.

Let $z = \sec\beta$, which implies $\tan\beta = \sqrt{z^2-1}$ and $\cot\beta = \frac{1}{\sqrt{z^2-1}}$. Moreover, η is defined as $\eta(z) = \tan\beta - \beta = \sqrt{z^2-1} - \sec^{-1} z$. Then, by $\cos\theta = \Re(e^{i\theta})$, $\quad \sin\theta = \Im(e^{i\theta})$, we have:

$$\cos(\nu\eta - \pi/4)\cdot S_0 + \sin(\nu\eta - \pi/4)\cdot S_1 = \Re\left[e^{i(\nu\eta-\pi/4)}(S_0 - iS_1)\right], \tag{3.9.15}$$

where $S_0 = \sum_{m=0}^{\infty}\frac{(-1)^m\Gamma(2m+\frac{1}{2})}{\Gamma(\frac{1}{2})}\cdot\frac{A_{2m}}{(\frac{1}{2}\nu\tan\beta)^{2m}}$ and $S_1 = \sum_{m=0}^{\infty}\frac{(-1)^m\Gamma(2m+\frac{3}{2})}{\Gamma(\frac{1}{2})}\cdot\frac{A_{2m+1}}{(\frac{1}{2}\nu\tan\beta)^{2m+1}}$.

We say that there exists $U_k(p)$ which is a polynomial combination of A_k by comparing $\frac{A_{2m}}{(\nu \tan\beta)^{2m}}$ and $\frac{U_k(p)}{\nu^k}$. By $\tan\beta = \sqrt{z^2-1}$ and $p = \frac{1}{\sqrt{1+z^2}}$, we have:

$$\left(\frac{2}{\pi\nu\tan\beta}\right)^{1/2} = \frac{1}{(1+z^2)^{1/4}} \cdot \frac{1}{\sqrt{2\pi\nu}} \cdot \left(\frac{2z^2}{z^2-1}\right)^{1/4}. \tag{3.9.16}$$

Combining Equation 3.9.14, Equation 3.9.15, and Equation 3.9.16 leads to:

$$J_\nu(\nu z) \sim \frac{\exp\left(\nu\eta - \frac{\pi}{4}\right)}{(1+z^2)^{1/4}\sqrt{2\pi\nu}} \left[\sum_{k=0}^{\infty} \frac{U_k(p)}{\nu^k}\right]. \tag{3.9.17}$$

Next, for $\nu \gg 1$ and $r \in K$ (i.e., $z = \frac{kr}{\nu}$ is bounded), we have:

$$J_\nu(kr) \approx \left(\frac{2}{\pi\nu}\right)^{1/2} \frac{\cos\left(\nu\eta(z) - \frac{\pi}{4}\right)}{(1+z^2)^{1/4}}. \tag{3.9.18}$$

Since $|\cos(\cdot)| \leq 1$ and $(1+z^2)^{1/4}$ has positive lower bound G on K, we have:

$$|J_\nu(kr)| \leq G\left(\frac{2}{\pi\nu}\right)^{1/2} = \mathcal{O}(\nu^{-1/2}). \tag{3.9.19}$$

Finally, substituting Equation 3.9.19 into 3.9.13, we have, for compact subsets $K \subset \mathcal{M}$:

$$\sum_j |a_j||F_j(x)| \leq \left(\sum_j |a_j|^2\right)^{1/2} \left(\sum_j |J_{\nu(j)}(kr)|^2\right)^{1/2} < \infty. \tag{3.9.20}$$

This completes the proof. ■

Proof of Theorem 3.9.1

Proof. For $f = \sum_j a_j\psi_j \in L^2(\partial\mathcal{M})$, let us define:

$$W_{\mathcal{M}} f = \sum_j a_j F_j, \quad \text{where } F_j = W_{\mathcal{M}}\psi_j. \tag{3.9.21}$$

From Part 3 of Lemma 3.9.1, the series converges absolutely and uniformly on compact subsets K as:

$$\sum_j |a_j|\|F_j\|_{L^\infty(K)} \leq C\left(\sum_j |a_j|^2\right)^{1/2} \left(\sum_j \lambda_j^{-1/2}\right)^{1/2} < \infty, \tag{3.9.22}$$

where $\|F_j\|_{L^\infty(K)} \leq C\lambda_j^{-\frac{1}{4}}$ comes from Bessel decay Matviyenko (1993) and $\lambda_j \sim j^{\frac{2}{n-1}}$ comes from Weyl's law Liokumovich et al. (2018).

Then, from Part 2 of Lemma 3.9.1:

$$\|W_{\mathcal{M}} f\|^2_{H^2(\mathcal{M})} = \sum_j |a_j|^2 \|F_j\|^2_{H^2(\mathcal{M})} \sim \sum_j |a_j|^2 = \|f\|^2_{L^2(\partial\mathcal{M})}. \tag{3.9.23}$$

Next, we prove the surjectivity of $W_{\mathcal{M}}$. Let $u \in W^2(\mathcal{M})$. On $\partial\mathcal{M}$, we expand u in eigenfunctions using:

$$u(r,\xi) = \sum_j A_j(r)\psi_j(\xi), \quad A_j(r) = \langle u(r,\cdot), \psi_j \rangle_{L^2(\partial\mathcal{M})}. \tag{3.9.24}$$

This way, the Helmholtz equation $(\Delta_{\mathcal{M}} + k^2)u = 0$ reduces to an ordinary differential equation:

$$A_j'' + \frac{n-1}{r} A_j' + \left(k^2 - \frac{\lambda_j + (\frac{n-2}{2})^2}{r^2} \right) A_j = 0, \tag{3.9.25}$$

whose solution is $A_j(r) = a_j r^{-\frac{n-2}{2}} J_{\nu(j)}(kr)$, where $\nu(j) = \sqrt{\lambda_j + (\frac{n-2}{2})^2}$. Therefore, $u = \sum_j a_j F_j = W_{\mathcal{M}} f$ for $f = \sum_j a_j \psi_j \in L^2(\partial\mathcal{M})$. Finally, the inverse $W_{\mathcal{M}}^{-1} : u \mapsto u|_{\partial\mathcal{M}}$ is bounded by the trace theorem Adams & Fournier (2003):

$$\|W_{\mathcal{M}}^{-1} u\|_{L^2(\partial\mathcal{M})} = \|u|_{\partial\mathcal{M}}\|_{L^2(\partial\mathcal{M})} \leq C\|u\|_{H^2(\mathcal{M})}. \tag{3.9.26}$$

This completes the proof. ■

3.10 Experiments

We evaluate the performance of our proposed model across three different PDEs on different manifolds whose solution spaces are not necessarily an RKHS, and compare it with recent neural PDE solvers including FNO Li et al. (2020b, 2023b), WNO Tripura & Chakraborty (2023), D-FNO Li & Ye (2025), and DeepONet Lu et al. (2019). Then, we present some key results from selected ablation studies to demonstrate the need for each of the core components of our AFDONet framework.

3.10.1 PDE problem settings

Helmholtz equation on planar manifold with boundary. Let $(\mathcal{M}, g)$ be a smooth planar Riemannian manifold with boundary $\mathcal{M} \subset \mathbb{R}^2$ equipped with the Euclidean-induced metric g. We consider the 2-D Helmholtz equation on $\mathcal{M}$ with perfectly-matched layer (PML) absorption on $\partial\mathcal{M}$ as follows:

$$\begin{gathered} \Delta_{\mathcal{M}} u(x,y) \; + \; k^2 n^2(x,y)\, u(x,y) = -\, S(x,y), \quad (x,y) \in \mathcal{M}, \\ \text{PML absorption on } \partial\mathcal{M}, \end{gathered} \tag{3.10.1}$$

where wavenumber k is a positive constant, $n : \mathcal{M} \to \mathbb{C}$ is the complex refractive-index field, and $S : \mathcal{M} \to \mathbb{C}$ is the source density. In our experiment, the planar manifold is constructed following Marchand (2023). Furthermore, one can show that the solutions of the Helmholtz equation naturally span an RKHS (see Section 3.9).

Incompressible Navier-Stokes equation on a torus. Let $(\mathbb{T}^2, g)$ denote a flat two-dimensional torus $\mathbb{T}^2 = \big([0, 2\pi] \times [0, 2\pi]\big)/\!\sim$ obtained by identifying opposite edges of the square and endowed with the Euclidean metric g. For viscosity $\nu > 0$, we study the 2-D incompressible Navier-Stokes system:

$$\begin{aligned} \partial_t \mathbf{u} + (\mathbf{u}\cdot\nabla)\,\mathbf{u} &= -\,\nabla p \; + \; \nu\, \Delta_{\mathbb{T}^2} \mathbf{u}, && (x,y,t) \in \mathbb{T}^2 \times (0,T], \\ \nabla_{\mathbb{T}^2} \cdot\, \mathbf{u} &= 0, && (x,y,t) \in \mathbb{T}^2 \times [0,T], \\ \mathbf{u}(\,\cdot\,, 0) &= \mathbf{u}_0, && x \in \mathbb{T}^2, \end{aligned} \tag{3.10.2}$$

where $\mathbf{u} = (u, v) : \mathbb{T}^2 \times [0,T] \to \mathbb{R}^2$ is the velocity field and $p : \mathbb{T}^2 \times [0,T] \to \mathbb{R}$ is the pressure.

Homogeneous Poisson equation on a quarter-cylindrical surface. Let $(\mathcal{M}, g)$ be a smooth two-dimensional Riemannian manifold $\mathcal{M} = \Big\{(\cos\phi, \sin\phi, z) \in \mathbb{R}^3 \;:\; 0 < \phi < \frac{\pi}{2},\; 0 < z < L\Big\}$, which restricts the lateral surface of the unit cylinder to a single quadrant. The metric g is the Euclidean metric pulled back by the embedding, so that in local coordinates (ϕ, z) one has $\Delta_{\mathcal{M}} = \partial_{\phi\phi} + \partial_{zz}$. We study the 2-D homogeneous Poisson problem with

Dirichlet boundary conditions on $\partial\mathcal{M}$:

$$\begin{aligned} -\Delta_{\mathcal{M}} u(\phi, z) &= f(\phi, z), \qquad (\phi, z) \in (0, \tfrac{\pi}{2}) \times (0, L), \\ u(\phi, z) &= 0, \qquad\qquad (\phi, z) \in \partial\mathcal{M}, \end{aligned} \tag{3.10.3}$$

where the source term $f(\phi, z) = \beta\Big[\big(\frac{\alpha\pi}{L}\big)^2(1-\cos\phi) - \big(\cos\phi + \sin\phi - 4\sin\phi\cos\phi\big)\Big]\sin\Big(\frac{\alpha\pi z}{L}\Big)$ Kamilis (2013).

Since Helmholtz and Poisson equations are stationary, we focus on the static task for both problems. And for the Navier-Stokes equation, we consider both static and autoregressive tasks.

3.10.2 Datasets

Helmholtz equation. We generate the dataset using `helmhurts-python`, a Helmholtz equation solver Marchand (2023). This solver computes the electric field distribution $u(x, y)$ for given $n(x, y)$ and source terms $S(x, y)$, discretized on a uniform grid with resolution $\Delta x = \Delta y = 1\,\text{cm}$. $S(x, y)$ is constructed by assigning a complex-valued excitation $P \cdot e^{i\phi}$ to all pixels marked as sources (RGB (255,0,0)) in the input image, where P is the transmitter power and $\phi = 0$ denotes a uniform phase alignment. Perfectly matched layers (PMLs) of thickness 12 cells absorb outgoing waves to approximate open boundary conditions. We select randomized physical parameters to generate the full dataset, including transmitter power $P \sim \mathcal{U}(0.5, 2.0)$, frequency $f \sim \mathcal{U}(1.5, 3.0)\,\text{GHz}$, and wall properties $\eta \sim \mathcal{U}(1.5, 3.0)$, $\kappa \sim \mathcal{U}(0.05, 0.2)$. The resulting field intensities $|u|$ are log-scaled and normalized to $[0, 1]$.

Navier-Stokes equation. The dataset is generated by numerically solving the 2D incompressible Navier-Stokes equations using a spectral method solver adapted from the `NSsimulation` repository lavenderses (2021) on a torus. The viscosity ν are sampled following $\nu \sim \mathcal{U}(0.001, 0.1)$. For the static task, the dataset contains the value of parameters α and the numerical solutions $\mathbf{u}$. For the autoregressive task, the dataset contains the numerical solutions $\mathbf{u}(x, t)$ and $\mathbf{u}(x, t+1)$.

Poisson equation. Using isogeometric analysis with NURBS basis functions of order $p = 2$ proposed in Kamilis (2013), we generate the dataset for this problem by specifying $\alpha \sim (2, 6)$.

3.10.3 Implementation details

We run all experiments in a Dell Precision 7920 Tower equipped with Intel Xeon Gold 6246R CPU and NVIDIA Quadro RTX 6000 GPU (with 24GB GGDR6 memory).

For FNO-based solvers Li et al. (2020b, 2023b); Li & Ye (2025), the number of Fourier modes considered in the spectral convolutions is an important hyperparameter. We find that no more than 16 Fourier modes are enough to solve the three benchmark PDE problems. In fact, increasing the number of Fourier modes beyond 16 could lead to worse performance. From Figure 16, we plot the average MAE and total computational time of FNO with $8, 12, 16, 32, 64, 128$ Fourier modes. As a result, in our experiments, we set the number of Fourier modes to be 12 for all FNO and D-FNO models. Similar trends happen to other benchmark PDE problems, so we use 12 Fourier modes in all benchmark PDE problems.

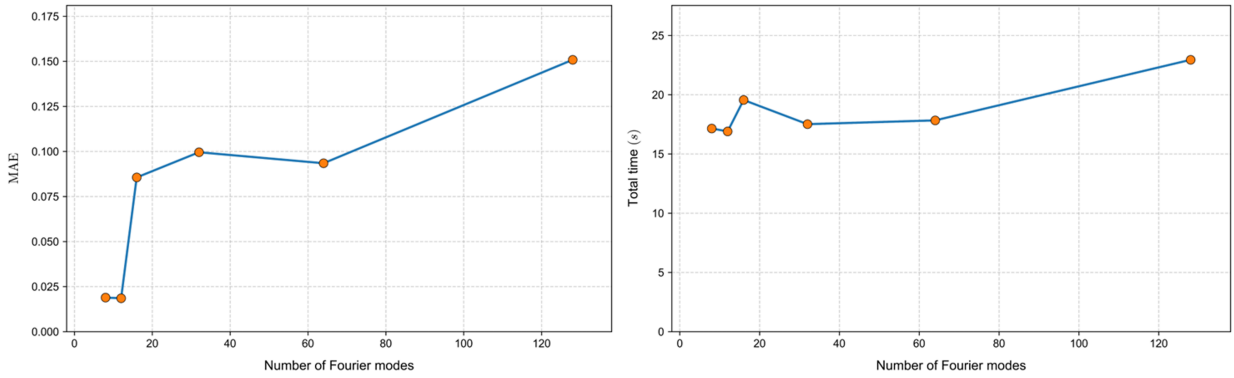


Figure 16: Average MAE and total computational time (in seconds) of FNO solver with respect to number of Fourier modes (averaged over five random seeds) for solving the Helmholtz equation 3.10.1.

In addition, for AFDONet, increasing the dimension of the latent space helps achieve higher accuracy. However, this also comes with an increase in computational costs. This is illustrated in Table 11 below taking Navier-Stokes equation. Therefore, to demonstrate the

effectiveness of our AFDONet solver even in the worst-case scenario, we set the latent space dimension to 10 for all benchmark PDE problems.

Table 11: Average MAE, relative L^2 error, and computational time (in seconds) of AFDONet (averaged over five random seeds) for solving Navier-Stokes equation 3.10.2 (autoregressive task) under different latent space dimensions.

Latent dimension	**MAE**	**Relative L^2 error**	**Time (sec)**
16	6.40E-04 ± 9.90E-05	1.11E-03 ± 1.91E-04	1058.39 ± 19.30
20	5.35E-04 ± 1.36E-04	1.40E-03 ± 1.03E-03	1190.61 ± 15.67
32	3.77E-04 ± 1.28E-04	9.60E-04 ± 8.03E-04	1110.57 ± 18.38
64	4.62E-04 ± 1.35E-04	1.22E-03 ± 8.92E-04	1173.40 ± 17.22
100	4.05E-04 ± 1.09E-04	1.06E-03 ± 9.94E-04	1365.03 ± 21.89
128	3.89E-04 ± 1.26E-04	9.99E-04 ± 8.48E-04	1406.05 ± 23.98
256	5.03E-04 ± 1.98E-04	1.27E-03 ± 1.14E-03	1743.28 ± 27.64

The AFDONet loss function and training specifications are listed in Table 12 below.

Table 12: Specifications of loss function and training for AFDONet solver.

Parameter	**Value**
Training epochs	100
Loss weights (ω)	10^{-5}
Loss weights (w_i)	10^{-8}
Optimizer	Adam
Learning rate	10^{-3}
Batch size	16
Encoder hidden layers dimension	256
Latent space dimension	10

For the benchmark solvers, their detailed architectures are as follows:

- The FNO solver Li et al. (2020b, 2023b) consists of an initial linear projection layer P (width is 32) followed by 5 Fourier layers with 12 Fourier modes and GeLU activation function. A neural network with two fully connected layers Q (the first layer has 128 neurons and the second layer has 2 neurons) is used to project back to the target dimension. The Adam optimizer (learning rate: 10^{-3}) is used to train the FNO solver based on minimizing the MSE loss.

- The D-FNO solver Li & Ye (2025) has a similar architecture as the FNO solver, except that a reduction layer is introduced between the initial linear projection layer P and the 5 Fourier layers to decompose the output of P into a series of two one-dimensional vectors. The reduction layer does not use traditional neurons. Instead, it projects inputs into a rank-16 subspace via factor matrices (see Equation 6 of Li & Ye (2025)). The Fourier layers have 12 Fourier modes (also suggested by Li & Ye (2025)) and use GeLU activation function. After that, an operation called product is used to put the two vectors together. In D-FNO, Q has two layers (the first layer has 128 neurons and the second layer has one neuron). The Adam optimizer (learning rate: 10^{-3}) is used to train the D-FNO solver based on minimizing the MSE loss.

- The WNO solver Tripura & Chakraborty (2023) adopts the FNO architecture by replacing Fourier layers with wavelet integral layers that decompose the inputs using Daubechies wavelets and apply learnable linear transformations to the wavelet coefficients before reconstruction. The structure of Q is the same as that of FNO. GeLU activation function and the Adam optimizer (learning rate: 10^{-3}) are used.

- The DeepONet solver Lu et al. (2019) consists of two subnetworks: a branch network and a trunk network. The branch network which handles the high-dimensional input functions has three fully-connected layers with 64 neurons per layer. The truck network which handles spatial coordinates also has three fully-connected layers with 64 neurons

per layer. Their outputs are combined via a dot product. ReLU activation function is employed in both branch and truch networks. We use the Adam optimizer (learning rate: 10^{-3}) to minimize the MSE loss.

3.10.4 Results and discussions

Comparison with benchmark methods. In Table 13, we report the performance of AFDONet and benchmark methods in terms of average mean absolute error (MAE) and relative L^2 error, as well as their standard deviations ($\pm$) obtained using five random seeds and dataset size of 5000. Synthetic datasets are generated using finite difference and isogeometric methods, and each model is trained on a 60/20/20 split of training, validation, and testing data. We conclude that, given different dataset sizes, our AFDONet solver consistently outperforms FNO-based solvers and DeepONet across all PDE cases on manifolds. Note that FNO, D-FNO, AFNO, and WNO solvers rely on fast Fourier transform and wavelet transform, both of which are inherently defined on Euclidean domain and thus do not generalize well to curved geometries. Meanwhile, DeepONet does not exploit the spectral sparsity of the solution space. In contrast, AFDONet adaptively selects analytic modes and employs pullback operators to ensure accurate, manifold-aware representations.

Scalability of AFDONet. In Figure 17, we show that AFDONet is scalable subject to increasing dataset size for all benchmark PDE problems considered.

Latent-to-RKHS network vs. Latent-to-kernel network. Our decoder operates within an RKHS $\mathcal{H}(\mathcal{M})$, which is constructed via a latent-to-RKHS network. This network maps latent representations to their nearest RKHS within a Hilbert space. To understand the need for function restrictions within an RKHS, we conduct an ablation study and compare the latent-to-RKHS network with the latent-to-kernel network Lu et al. (2020a), which directly maps latent representations to a kernel function that does not necessarily satisfy the reproducing property. By comparing the results in Tables 13 and 14, we observe that latent-

Table 13: Average MAE and relative L^2 errors and their standard deviations for different PDE benchmark solvers obtained using five random seeds. Dataset size is 5000. The best results are bolded. All values in the table have been multiplied by 100.

Equation	**Metric**	**AFDONet (Ours)**	**FNO**	**D-FNO**	**WNO**	**DeepONet**
Helmholtz 3.10.1	MAE	**0.937 ± 0.063**	1.855 ± 0.165	6.085 ± 0.355	11.701 ± 1.429	16.224 ± 1.054
	Rel. L^2	**8.141 ± 1.401**	11.915 ± 0.935	39.191 ± 9.361	69.735 ± 12.675	46.310 ± 10.540
Navier-Stokes (Static) 3.10.2	MAE	**0.332 ± 0.030**	2.908 ± 0.741	0.375 ± 0.103	3.974 ± 0.005	3.189 ± 0.164
	Rel. L^2	**0.882 ± 0.059**	7.567 ± 0.173	0.996 ± 0.263	9.989 ± 0.004	7.251 ± 0.422
Navier-Stokes (Autoreg.) 3.10.2	MAE	**0.068 ± 0.037**	2.386 ± 0.249	0.142 ± 0.009	3.826 ± 0.191	3.168 ± 0.221
	Rel. L^2	**0.170 ± 0.104**	6.288 ± 0.820	0.298 ± 0.060	9.541 ± 0.475	7.071 ± 0.897
Poisson 3.10.3	MAE	**0.158 ± 0.033**	0.777 ± 0.093	0.343 ± 0.066	0.770 ± 0.161	0.531 ± 0.030
	Rel. L^2	**0.472 ± 0.109**	2.567 ± 0.502	0.513 ± 0.242	1.754 ± 0.943	0.483 ± 0.305

to-RKHS network consistently outperforms the latent-to-kernel network. Both MAE and relative L^2 error show at least an order of magnitude reduction for all PDE cases *except* the Helmholtz equation 3.10.1, which only yields a slight performance gain. This is due to the fact that the solution space for the the Helmholtz equation 3.10.1 is already an RKHS (See Section 3.9). This illustrates the need and benefit of restricting the latent representations to their RKHS.

AFD-type decoder vs. other decoder architectures. We conduct ablation studies by replacing our full AFD-type dynamic CKN decoder with three alternatives, namely an MLP decoder, a propagation decoder Lu et al. (2020a); Buchberger et al. (2020), and an AFD-type decoder with a static CNN. As shown in Table 14, full AFD-type dynamic CKN decoder achieves the best performance for all PDE cases. The improvements are especially significant for the Navier-Stokes equation 3.10.2 and Poisson equation 3.10.3, where both the MAE and relative L^2 error are reduced by one to two orders of magnitude compared to the benchmark decoders. Also, we observe that AFD-type decoder with a static CNN performs slightly worse than our AFD-type dynamic CKN decoder since CNN uses stationary kernels

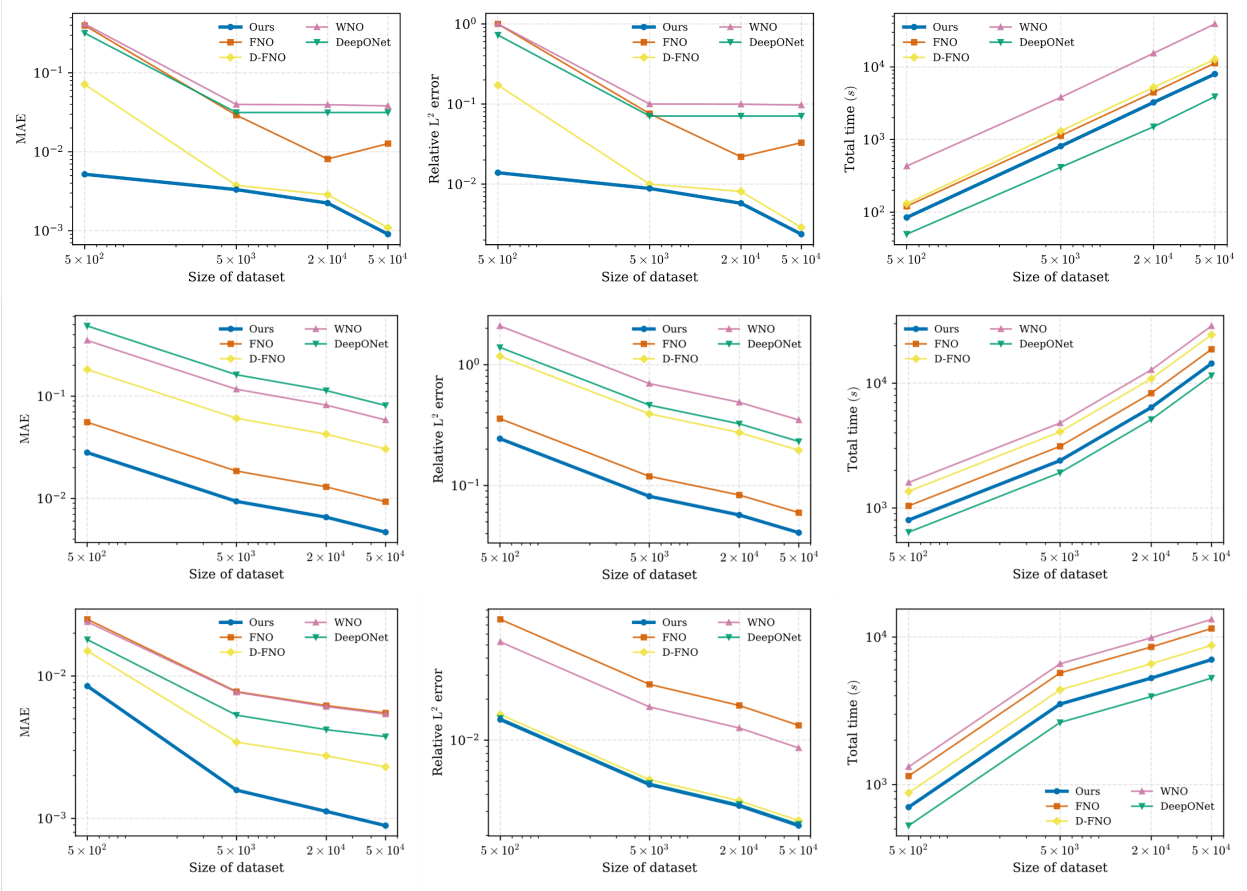


Figure 17: Average MAE, relative L^2 error, and total computational time comparisons with respect to dataset size (averaged over five random seeds) for Navier-Stokes equation (static task) (top row), Helmholtz equation (middle row), and Poisson equation (bottom row).

that lack adaptability to the varying spatiotemporal dynamics in PDE solutions. In contrast, dynamic CKN enables data-driven, non-stationary kernel learning, which can better capture these inherent dynamics, especially for heterogeneous equations such as the Poisson equation 3.10.3 or time-dependent equations like the Navier-Stokes equation 3.10.2.

Need for VAE backbone. We design a new ablation study for the Navier-Stokes example with randomized vortex field dataset. The randomized vortex field dataset exhibits sharp gradients and turbulence-like behavior and includes a phase shift for the v-component. Therefore, the dynamics of this dataset are challenging to learn. Our goal is to determine

Table 14: Ablation studies of our AFDONet architecture show that latent-to-RKHS and AFD-type dynamic CKN decoder work synergistically to improve the solution accuracy. Note that the results for the full architecture are presented in Table 13. The dataset size is 5000.

Equation	**Metric**	**Latent-to-kernel network + AFD-type decoder**	**Latent-to-RKHS network + MLP-type decoder**	**Latent-to-RKHS network + propagation decoder**	**Latent-to-RKHS + AFD-type decoder (static CNN)**	**Latent-to-RKHS network + AFD-type decoder (without Equation 3.4.6)**
Helmholtz 3.10.1	MAE	1.27E-02 ± 1.91E-03	2.11E-01 ± 2.04E-03	1.93E-01 ± 5.11E-02	2.41E-02 ± 1.16E-02	1.81E-01 ± 5.16E-02
	Rel. L^2	8.89E-02 ± 6.90E-03	1.17 ± 1.22E-02	1.07 ± 2.64E-01	1.72E-01 ± 9.13E-02	1.10 ± 2.62E-01
Navier-Stokes (Static) 3.10.2	MAE	8.32E-02 ± 1.46E-02	4.00E-01 ± 4.46E-03	3.98E-01 ± 4.68E-04	7.12E-02 ± 1.20 E-02	1.27E-02 ± 2.03E-03
	Rel. L^2	2.19E-01 ± 3.44E-02	1.00 ± 9.36E-03	1.00 ± 8.30E-06	1.85E-01 ± 3.54E-02	3.71E-02 ± 6.29E-03
Navier-Stokes (Autoreg.) 3.10.2	MAE	6.11E-02 ± 2.92E-03	1.45E-01 ± 2.59E-02	1.48E-01 ± 1.09E-01	8.32E-02 ± 9.28E-03	2.53E-03 ± 8.26E-04
	Rel. L^2	1.58E-01 ± 9.20E-03	3.85E-01 ± 6.84E-02	3.91E-01 ± 2.30E-01	2.16E-01 ± 2.35E-02	7.80E-03 ± 1.10E-03
Poisson 3.10.3	MAE	3.16E-01 ± 8.76E-04	1.71E-02 ± 7.73E-03	1.81E-02 ± 1.84E-03	6.08E-02 ± 6.88E-03	3.53E-02 ± 5.51E-03
	Rel. L^2	9.77E-01 ± 2.31E-03	5.10E-02 ± 2.22E-02	5.61E-02 ± 2.17E-02	1.77E-01 ± 5.16E-03	1.30E-01 ± 1.44E-02

whether the v-component solution profile would visually match with the ground truth solution when the VAE backbone and its components are removed or replaced. From Table 15, it is clear that the synergistic integration of VAE backbone, latent-to-RKHS network, and AFD-type decoder is essential in accurately capturing v-component solution profile in the dataset. Guided by the AFD theory in their design and integration, these components come together to establish the accuracy of our AFDONet solver.

Visualization of solver performance in benchmark PDE problems

In Figures 18 through 20, we plot the ground truth and predicted solutions of AFDONet and baseline methods for the three case studies. The corresponding MAE and relative L^2 error results are listed in Table 13.

AFDONet performance on Navier-Stokes equation problem with randomized vortex dataset

We extend the ablation study shown in Table 14 with a new ablation study for the Navier-Stokes example with randomized vortex field dataset. The initial condition is set by vortex

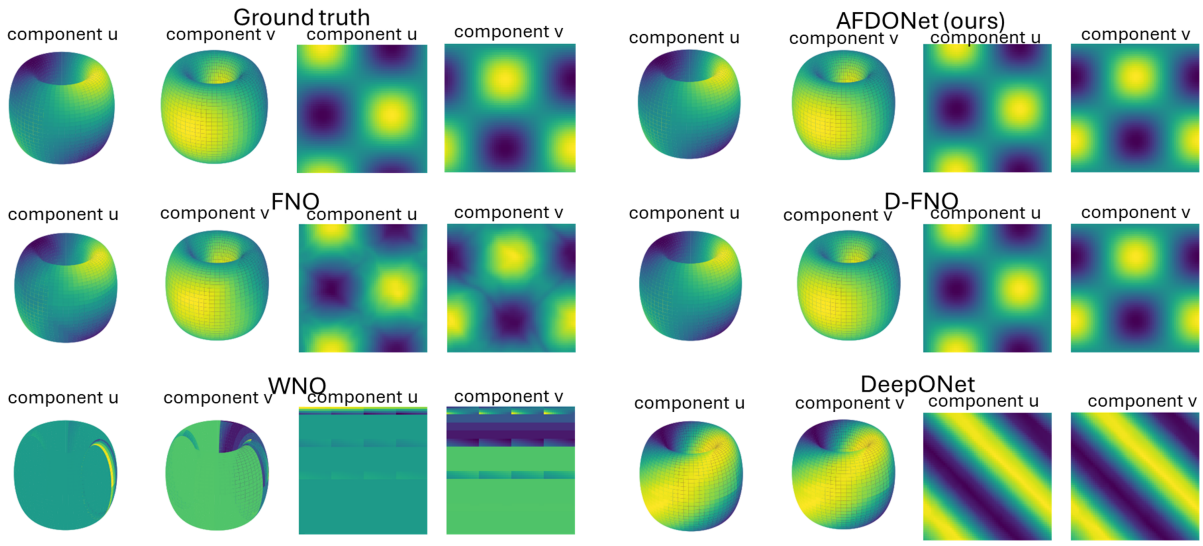


Figure 18: Ground truth and predicted solutions (u, v) of the Navier-Stokes equation (static task) on the torus and heat map.

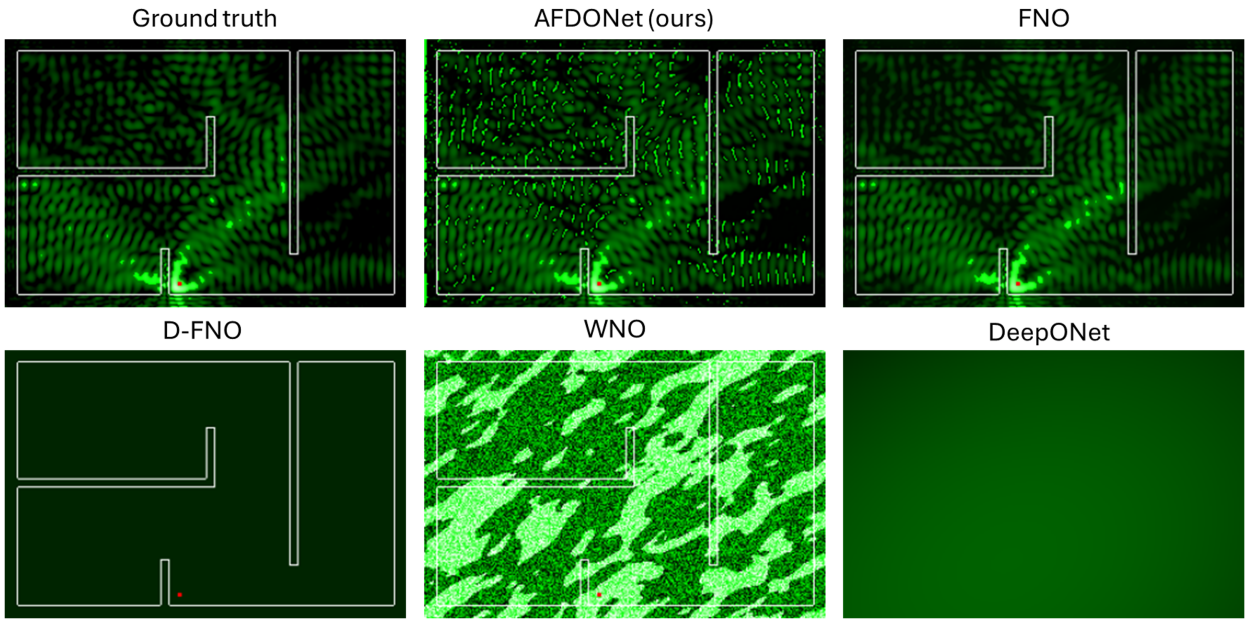


Figure 19: Ground truth and predicted solutions $u(x, y)$ of the Helmholtz equation on the planar manifold.

structures via Gaussian-based stream functions $\psi = A \cdot \exp\left(-\frac{(x-c_x)^2+(y-c_y)^2}{2r^2}\right)$ with randomized parameters vortex centers $(c_x, c_y) \sim \mathcal{U}(1, 5)^2$, radii $r \sim \mathcal{U}(0.5, 2)$, and strengths

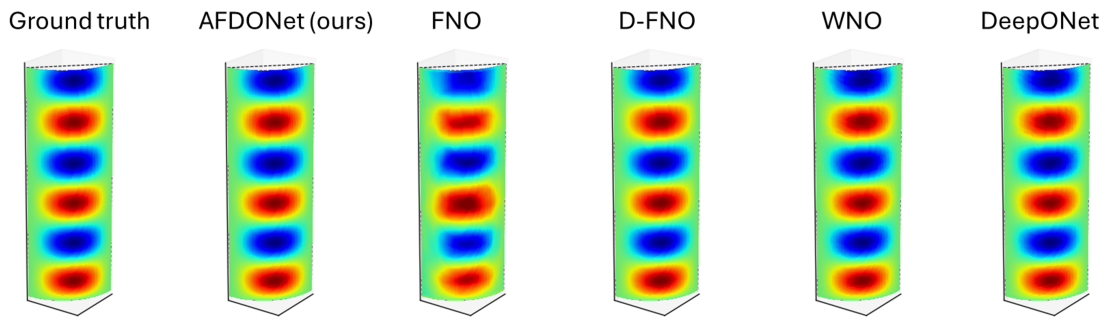


Figure 20: Ground truth and predicted solutions $u(\phi, z)$ of the Poisson equation on the quarter-cylindrical surface.

$A \sim \mathcal{U}(-2, 2)$.

Table 15: Ablation study of replacing VAE with multi-layer fully-connected feedforward (MLP) network as the encoder. Here, ✓: v-component solution dynamics visually matches with the ground truth solution; ✗: v-component solution dynamics does not visually match with the ground truth.

Backbone	**Full AFDONet (latent-to-RHKS network + AFD-type decoder + Equation 3.4.6**	**Latent-to-kernel network + AFD-type decoder**	**Latent-to-RKHS + MLP-type decoder**	**Latent-to-RKHS + propagation decoder**	**Latent-to-RKHS + AFD-type decoder (static CNN)**	**Latent-to-RKHS + AFD-type decoder (without maximal (without Equation 3.4.6)**
VAE	✓	✗	✗	✗	✓	✓
Without VAE (encoder deterministic MLP)	✗	✗	✗	✗	✗	✗

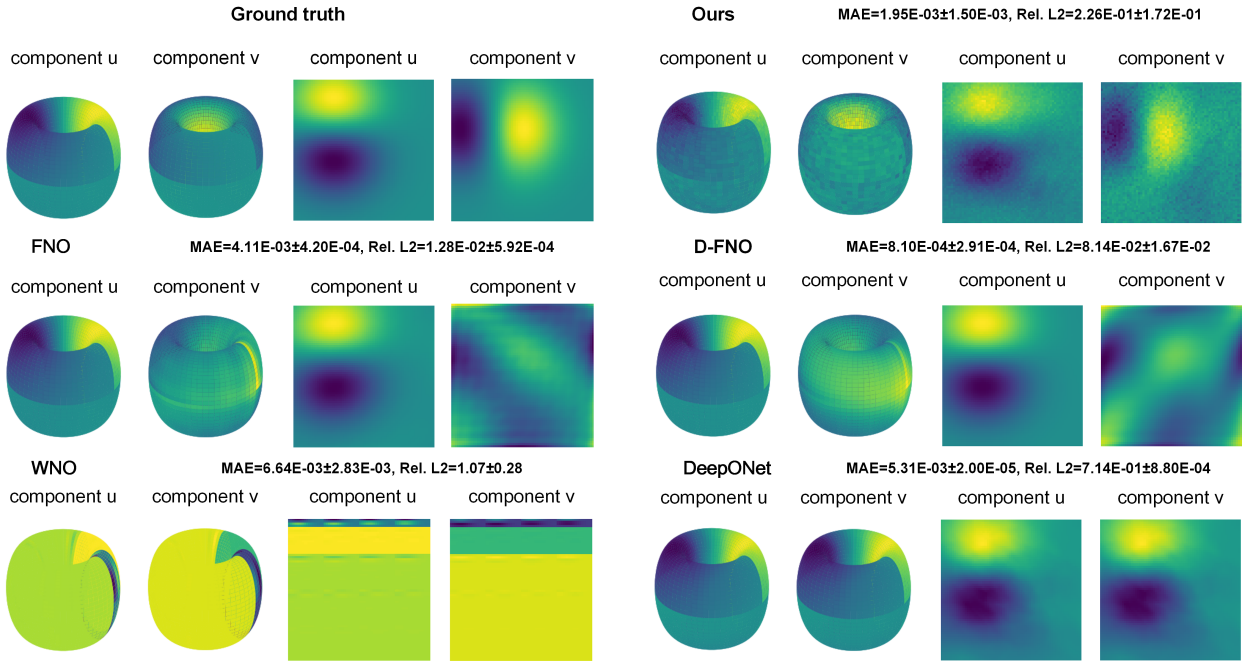


Figure 21: Ground truth and predicted fields (u, v) of the Navier-Stokes equation (for static task) on both the torus $\mathbb{T}^2$ and the heatmap for various solvers. Here, the dataset is generated from Gaussian-based randomized vortex fields (dataset size is 5000) Pedergnana et al. (2020). Average MAE and relative L^2 errors and their standard deviations obtained using five random seeds are also reported.

CHAPTER IV

ADAPTIVE MAMBA NEURAL OPERATORS

4.1 Problem Statement

We frame our task as learning a solution operator for a family of parametric PDEs. In general, consider a PDE defined on a spatial domain $\Omega \subset \mathbb{R}^d$ and a time interval $(0, T]$:

$$\mathcal{L}_a[u(x,t)] = f(x,t), \quad \forall (x,t) \in D \times (0,T], \tag{4.1.1}$$

which is subject to a set of initial and boundary conditions. Here, the parameter function $a \in \mathcal{A}$ specifies the coefficients and initial and boundary conditions of Equation 4.1.1. In operator learning, our goal is to construct an accurate approximation for $\mathcal{G} : \mathcal{A} \to \mathcal{F}(D \times [0,T])$, which maps the parameter function a to the corresponding solution function $u(x,t) \in \mathcal{F}$, via a parametric mapping $\mathcal{G}_\theta$. The aim is to learn θ such that $\mathcal{G}_\theta \approx \mathcal{G}$ from a set of training data $\{(a_j, u_j)\}_j$.

4.2 Related work

Frequency-based neural operators. Early advancements in operator learning exploited spectral decompositions to encode global information efficiently. A notable example is FNO Li et al. (2020b), which parameterizes integral kernels in the Fourier domain to enable resolution-invariance. However, FNO does not generalize well to irregular geometries Li et al. (2020b). Later, Geo-FNO Li et al. (2023b) was proposed to solve PDEs on general geometries. U-FNO Wen et al. (2022) introduced architectural modifications to better capture localized details while maintaining FNO's global properties. Meanwhile, F-FNO Tran

et al. (2021) generalizes the FNO architecture for more efficient spectral layers and deeper architectures. On the other hand, neural operators based on the wavelet transform include WNO Tripura & Chakraborty (2023), MWT Gupta et al. (2021), Padé Gupta et al. (2022), and CMWNO Xiao et al. (2023a). Fourier and wavelet transforms are both special cases of spectral decomposition, and neural operators based on spectral decomposition has recently been proposed Fanaskov & Oseledets (2023).

Attention-based neural operators. Attention mechanisms have been widely studied in neural operator domain. Some of the notable works include orthogonal attention Xiao et al. (2023b), physics-cross-attention Wang & Wang (2024), and nonlocal attention Yu et al. (2024). The Transformer structure is also a promising building block for neural operators. Some of the related works include OFormer Li et al. (2022b), LSM Wu et al. (2023), and Transolver Wu et al. (2024). However, Transformers struggle to capture kernel integral transforms efficiently in complex, high-dimensional continuous PDEs Guibas et al. (2021).

SSM-based neural operators. To address the computational inefficiency of Transformer-based neural operators, SSM and Mamba emerge as promising architectures for neural operator designs Tiwari et al. (2025). Previous studies of SSM-based neural operators Zheng et al. (2024); Cheng et al. (2024); Hu et al. (2024); Tiwari et al. (2025) have been applied to nonlinear PDEs on irregular geometries and dynamical systems. These works incorporate traditional SSMs with different scan strategies without considering the information in the frequency domain. On the other hand, our AFMO considers the frequency information via its explicit kernel and SSMs from a transfer function perspective Parnichkun et al. (2024).

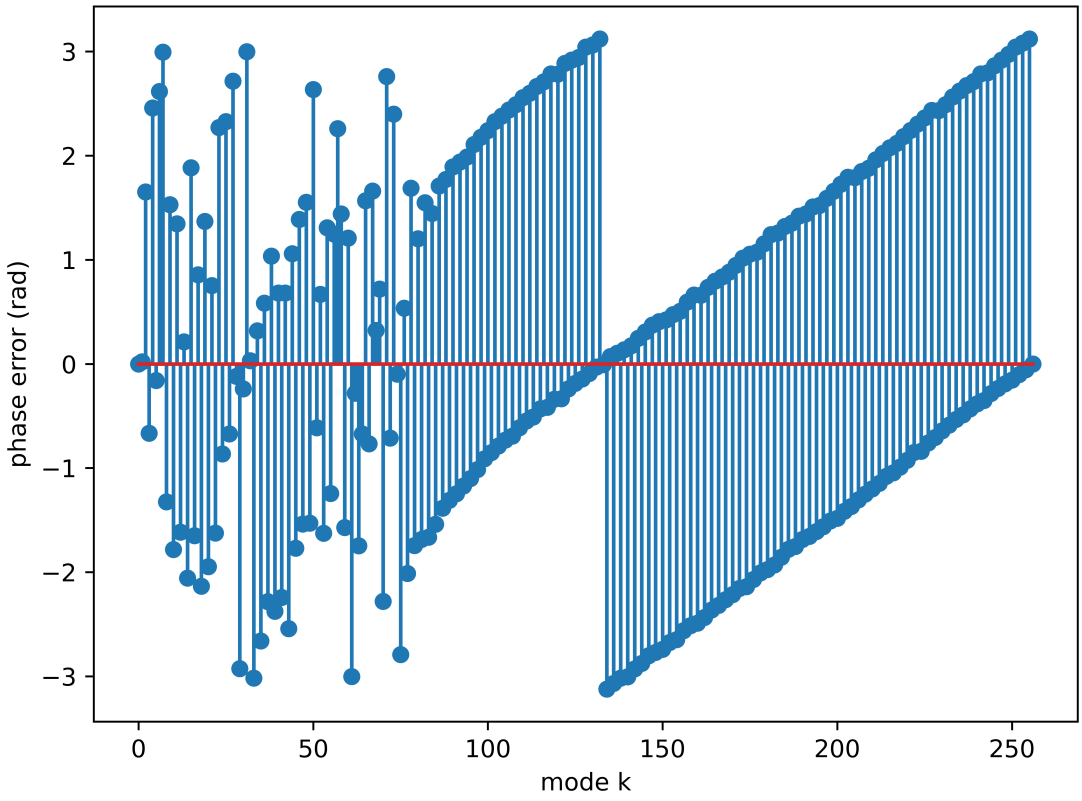


Figure 22: Phase error of solutions predicted by LaMO.

4.3 Illustrative Examples

1-D advection PDE with high-frequency perturbation. We evaluate LaMO on a 1-D linear advection benchmark governed by

$$u_t + c\, u_x = 0 \tag{4.3.1}$$

on a periodic unit interval. Initial conditions $u_0(x)$ are synthesized as smooth Fourier mixtures $\sum_{k=1}^{k_{\max}} a_k \sin(2\pi k x + \phi_k)$ with amplitudes decaying as $a_k \sim (1+k)^{-1}$, to which we add a weak high-frequency spike at wavenumber k_{hi} to probe aliasing and phase accuracy. Trajectories are advanced to time T with a conservative first-order upwind scheme at Courant number $\text{CFL} = c\,\Delta t/\Delta x \leq 0.5$, ensuring stability while preserving sharp phase relationships; the target is the advected field $u(\cdot, T)$.

Figure 22 visualizes the phase error of LaMO's predictions, revealing a pronounced degra-

dation for high-frequency modes (approximately $k \in [140, 250]$). This suggests that LaMO struggles to faithfully capture phase at the upper end of the spectrum.

2-D Darcy flow equation with fractal noise. We construct a challenging 2-D Darcy dataset by solving

$$-\nabla \cdot \big(k(x,y)\nabla u(x,y)\big) = f(x,y) \tag{4.3.2}$$

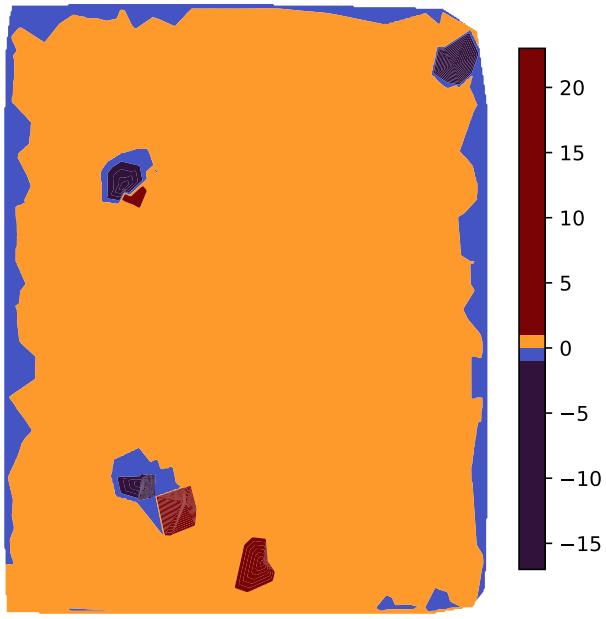
on $[0,1]^2$ with homogeneous Dirichlet boundaries, where the permeability k is positive, highly heterogeneous, and fractal-like. Specifically, k is generated by exponentiating a band-limited fractional Gaussian field (small Hurst parameter for roughness) and then modulating it with narrow channel masks and inclusions to induce strong anisotropy and high contrast. The forcing f combines a weak background term with several randomized Gaussian sources/sinks, which produce near-singular behavior in the solution. The variable-coefficient elliptic problem is discretized on a Cartesian grid using a flux-conservative 5-point stencil with harmonic averaging of k, and solved to tight tolerance via conjugate gradients. For learning, each sample is subsampled irregularly: we draw P points $\{(x_i, y_i)\}$ and record $u(x_i, y_i)$, yielding pairs (XY, U) without exposing k or f.

To visualize and stress singular structures, we show in Figure 23 (a) and (c): (i) contours of the potential u highlighting global flow topology, and (ii) a logarithmic map of the gradient magnitude, $\log|\nabla u|$, computed on a reconstructed dense grid via triangulation. Figure 23 shows LAMO cannot capture the singularities of u and $\log|\nabla u|$. Furthermore, once the complex singularities appear, the performance of LAMO will be affected.

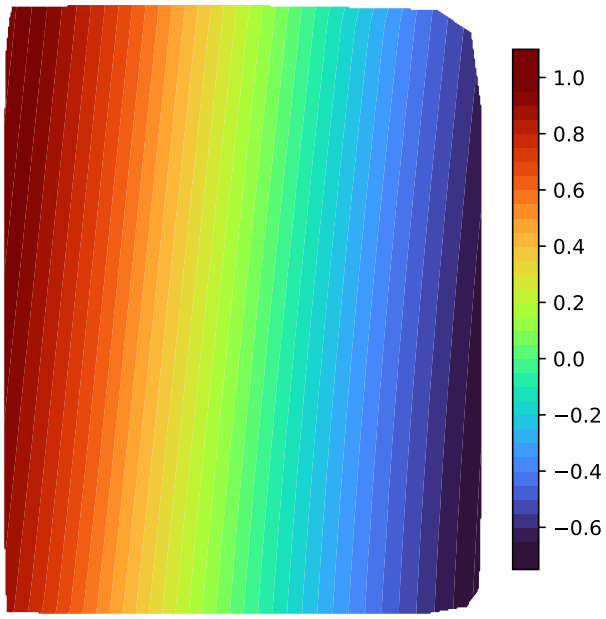
4.4 Adaptive Fourier Mamba operator

4.4.1 AFMO Architecture

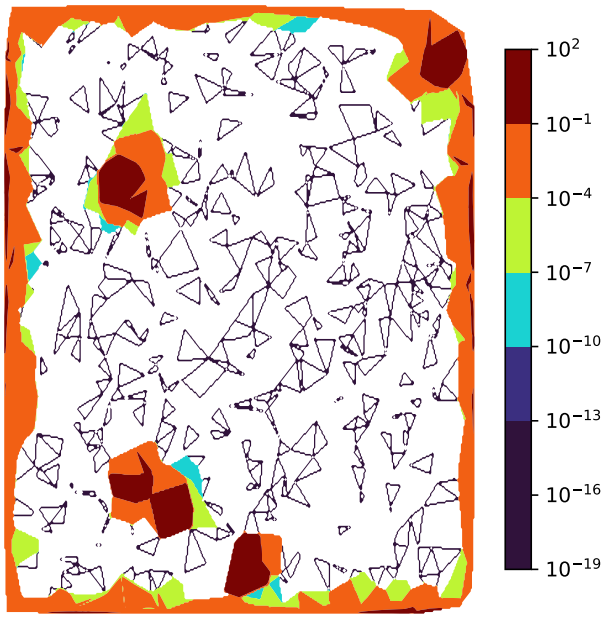
AFMO is a novel neural operator architecture that synergizes the mathematical groundness of AFD theory with the efficiency of structured SSMs in the frenquency domain Gu & Dao (2023); Parnichkun et al. (2024). Different from LaMO Tiwari et al. (2025), which



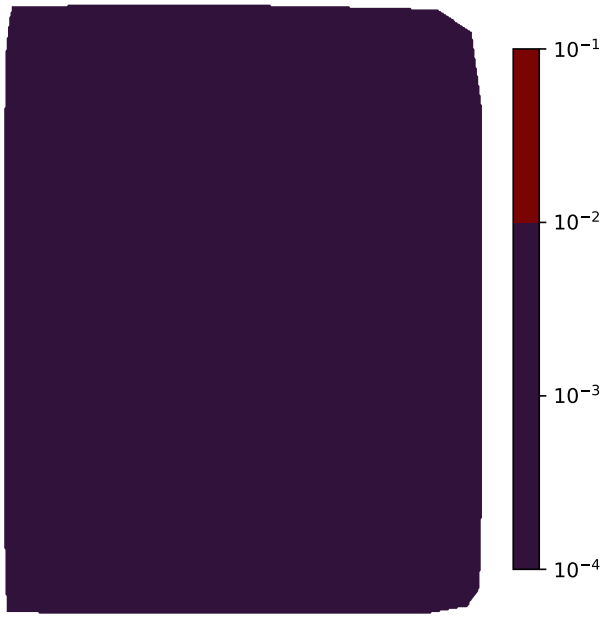
(a) Ground truth u



(b) Predicted by LaMO



(c) Ground truth $\log|\nabla u|$



(d) Predicted by LaMO

Figure 23: The predicted results produced by LaMO compared to the ground truth.

compresses the physical tokens into a fixed-size latent representation, AFMO utilizes a multi-layer fully-connected feedforward neural network (MLP) to first map the encoded tokens to their counterparts on the reproducing kernel Hilbert space (RKHS), and then iteratively refine them by a series of processing blocks. Each block uniquely integrates two components: (i) a TM layer containing global spectral transform via data-dependent TM bases, and (ii) a bidirectional SSM Gu et al. (2021); Gu & Dao (2023) parameterized by transfer functions in the frequency domain Parnichkun et al. (2024) to efficiently capture long-range dependencies within the RKHS.

Neural architecture. Given the parameter function (input) a, the output of AFMO, denoted as $\hat{u}_{N,\theta}$, is:

$$\hat{u}_{N,\theta} = \mathcal{G}_\theta(a) = \left(\mathcal{Q} \circ \mathcal{S}^N \circ \mathcal{L}^N \circ \cdots \circ \mathcal{S}^1 \circ \mathcal{L}^1 \circ \mathcal{R} \circ \mathcal{P}\right)(a), \tag{4.4.1}$$

where $\circ$ is the function composition, N is the number of processing blocks, $\mathcal{P}$ is the lifting operator which encodes into a lower-dimensional space (maps the input to the first latent representation $\mathbf{z}_0$) Tiwari et al. (2025); Li et al. (2020b), $\mathcal{Q}$ is the corresponding projection operator mapping the lower-dimensional space back to the original space (maps the final latent representation $\mathbf{z}_{N+1}$ to the output) Tiwari et al. (2025); Li et al. (2020b), $\mathcal{R}$ is a multi-layer neural network mapping the physical token to an RKHS, $\mathcal{L}^i = \mathrm{SSM}^i \circ \mathrm{TM}^i$ ($i = 1, \ldots, N$) is the processing block of AFMO (which consists of a TM layer and a bidirectional SSM), and $\mathcal{S}^i$ ($i = 1, \ldots, N$) are aggregation layers with skip connections. These aggregation layers not only receive the final output from the layer sequence but also have access to the intermediate outputs from each of the preceding layers.

The lifting operator, $\mathcal{P}$, projects the N_s physical token inputs into a compressed set of M encoded tokens, where $M \ll N_s$. This projection is achieved via a cross-attention mechanism. A learnable query array, $\mathbf{L} \in \mathbb{R}^{M \times D_{\text{embed}}}$, acts as the query. The key and value pairs are constructed by combining a linear projection of the input features $\mathbf{x}_{\text{phys}}$ with a

positional embedding of their coordinates $\mathbf{g}_{\text{phys}}$ generated by a positional encoding network PEN. Here, $\mathbf{x}_{\text{phys}} \in \mathbb{R}^{N_s \times D_{\text{in}}}$ stacks the feature vectors $\{\mathbf{x}_i\}_{i=1}^{N_s}$ and $\mathbf{g}_{\text{phys}} \in \mathbb{R}^{N_s \times d}$ stacks the coordinates $\{\mathbf{g}_i\}_{i=1}^{N_s}$, and the physical token is essentially pair $(\mathbf{g}_i, \mathbf{x}_i)$. The process for generating the initial representation $\mathbf{z}_0$ is formally defined as:

$$\begin{aligned} \mathbf{kv} &= \text{Linear}(\mathbf{x}_{\text{phys}}) + \text{PEN}(\mathbf{g}_{\text{phys}}), \\ \mathbf{z}_0' &= \text{CrossAttn}(\text{query} = \mathbf{L}, \text{key} = \mathbf{kv}, \text{value} = \mathbf{kv}), \\ \mathbf{z}_0 &= \mathbf{z}_0' + \text{FFN}(\mathbf{z}_0'), \end{aligned} \tag{4.4.2}$$

where the output of the cross-attention module is processed through a residual connection and a standard feed-forward network FFN.

The mapping operator, denoted by $\mathcal{R}$, acts on the encoded representation produced by the lifting operator $\mathcal{P}$, which transforms this discrete encoded tokens into a representation within a continuous function space. Let $\mathbf{z}_0 \in \mathbb{R}^{M \times D_{\text{embed}}}$ be the set of encoded tokens generated by $\mathcal{P}$, the operator $\mathcal{R} : \mathbb{R}^{M \times D_{\text{embed}}} \to \mathcal{H}$ maps this representation to its counterpart in an RKHS $\mathcal{H}$. This mapping is typically implemented as a multi-layer fully-connected feedforward network MLP, which processes each token independently as:

$$\mathbf{z}_1 = \mathcal{R}(\mathbf{z}_0) = \text{MLP}(\mathbf{z}_0), \tag{4.4.3}$$

where $\mathbf{z}_1$ denotes the projected tokens in the RKHS. We remark that, the mapping operator $\mathcal{R}$ maps the encoded tokens $\mathbf{z}_0$ to the new tokens $\mathbf{z}_1$ in $\mathcal{H}$ without knowing the physical information $\mathbf{x}_{\text{phys}}$ and $\mathbf{g}_{\text{phys}}$.

The TM layer, denoted by TM^i $(i = 1, \ldots, N)$, performs a global convolution via a spectral transform, where the reproducing kernels and TM bases are constructed from data-dependent poles. To define the reproducing kernels, we parameterize a small MLP to predict a set of i complex values called "poles" $\{a_k\}_{k=1}^{i}$ (denoted as $a_{1:i}$) located in the unit disk $\mathbb{D} = \{z \in \mathbb{C} : |z| < 1\}$ from tokens $\mathbf{z}_i$. Once we have the set of poles, we can explicitly define

the reproducing kernel $K_a(z)$ as:

$$K_a(z) = \frac{1}{1 - \overline{a}z}, \tag{4.4.4}$$

where $z \in \mathcal{H}$ and a is a single pole satisfying $|a| < 1$. Intuitively, we remark that each pole can be viewed as a "tuning knob" that selects a particular spatial pattern in the solution, with its location in the complex plane controlling how localized that pattern is. Adaptive poles allow AFMO to survey more heavily in regions where the parameters change rapidly, while using fewer poles in smooth regions. Across layers, the poles evolve from broad, coarse patterns in early layers to more refined, problem-specific patterns in deeper layers.

To generalize on irregular geometries, the kernels in Equation 4.4.4 need to be modified to become orthonormal. These modified kernels are also known as the TM bases due to their deep connection to TM systems. The first basis, denoted as $\mathscr{B}_1$, is simply the normalized kernel of Equation 4.4.4 with pole a_1 as $\mathscr{B}_1(z; a_1) = \frac{\sqrt{1-|a_1|^2}}{1-\overline{a_1}z}$. Then, we start with $\frac{\sqrt{1-|a_2|^2}}{1-\overline{a_2}z}$, but it is not orthogonal to $\mathscr{B}_1$. We reach the orthogonality by subtracting its projection onto $\mathscr{B}_1$, and we get $\mathscr{B}_2(z; a_{1:2}) = \frac{\sqrt{1-|a_2|^2}}{1-\overline{a_2}z}\left(\frac{z-a_1}{1-\overline{a_1}z}\right)$ after normalization. This way, the bases $\mathscr{B}_i$ are finally formulated as:

$$\mathscr{B}_i(z; a_{1:i}) = \frac{\sqrt{1 - |a_i|^2}}{1 - \overline{a_i}z} \prod_{j=1}^{i-1} \frac{z - a_j}{1 - \overline{a_j}z}, \tag{4.4.5}$$

where $z \in \mathcal{H}$ and $a_{1:i}$ are poles learned by the small MLP satisfying $|a_k| < 1$ for $k = 1, \ldots, i$. Overall, the i-th TM layer TM^i applies a small MLP $\mathbf{z_i} \mapsto a_{1:i}$, and then construct the TM bases $\mathscr{B}_i$ according to 4.4.5. We remark that, the tokens $\mathbf{z}_i$ will be kept as the input of SSM^i along with the TM bases $\mathscr{B}_i$.

Bidirectional SSM block is effective in solving PDEs on irregular geometries Tiwari et al. (2025) and employs inherent kernel integrals. However, this inherent kernel does not contain information in the frequency domain, thereby falling short in capturing high-frequency and singular features. To address this limitation, we utilize the transfer function in training SSMs in the frequency domain Parnichkun et al. (2024). The SSM block SSM^i generates the

spectrum of output in the frequency domain $Y_i(e^{i\omega})$ as the product of the spectrum of input $Z(e^{i\omega})$ and the transfer function $H_i(e^{i\omega})$, i.e., $Z(e^{i\omega})H_i(e^{i\omega})$. We point out that the output is essentially the coefficient of discrete AFD operation with the form $\langle \mathbf{z_i}, \mathscr{B}_i \rangle$ Qian (2010); Qian et al. (2011), where the inner product is defined as $\langle x, f \rangle = \frac{1}{\tilde{N}} \sum_{n=0}^{\tilde{N}-1} x[n] \overline{f(e^{i2\pi n/\tilde{N}})}$. Here, $\tilde{N}$ denotes the length of signal $x = \{x[n]\}_{n=0}^{\tilde{N}-1}$.

Let us consider the impulse response h_i of SSM block SSM^i (linear time-invariant system) as:

$$h_i[n] = \frac{1}{2\pi} \int_0^{2\pi} \overline{\mathscr{B}_i\left(e^{i\omega}; a_{1:i}\right)} e^{i\omega n} \, d\omega. \tag{4.4.6}$$

Then, the corresponding transfer function H_i can be obtained as:

$$H_i(e^{i\omega}) = \overline{\mathscr{B}_i\left(e^{i\omega}; a_{1:i}\right)}. \tag{4.4.7}$$

By setting the transfer function of SSM to be Equation 4.4.7, the SSM block computes a correlation of the input $\mathbf{z}_i$ and $\mathscr{B}_i$:

$$Y_i(e^{i\omega}) = H_i(e^{i\omega}) X(e^{i\omega}) = \overline{\mathscr{B}_i(e^{i\omega}; a_{1:i})} \, X(e^{i\omega}) \tag{4.4.8}$$

in the frequency domain. In the time domain, Equation 4.4.8 leads to the update of $\mathbf{z_i}$:

$$\hat{\mathbf{z}}_{\mathbf{i+1}}[\ell] = (h_i * \mathbf{z_i})[\ell] = \sum_{n=0}^{M-1} \mathbf{z_i}[n] \overline{\mathscr{B}_i\left(e^{i2\pi(n-\ell)/M}; a_{1:i}\right)}, \tag{4.4.9}$$

where ℓ denotes the time shift in the correlation operations. The zero-lag sample gives the final output:

$$\hat{\mathbf{z}}_{\mathbf{i+1}}[0] = (h_i * \mathbf{z_i})[0] = \sum_{n=0}^{M-1} \mathbf{z_i}[n] \overline{\mathscr{B}_i\left(e^{i2\pi n/M}; a_{1:i}\right)} = \langle \mathbf{z_i}, \mathscr{B}_i \rangle. \tag{4.4.10}$$

Aggregation layers $\mathcal{S}^i$ has N neural layers and combines the skip connection $\mathbf{z_i}$ with the intermediate outputs $\hat{\mathbf{z}}_{\mathbf{i+1}}[0] = \mathcal{L}^i(\mathbf{z_i})$ and $\mathscr{B}_i = \text{TM}^i(\mathbf{z_i})$:

$$\begin{aligned} \mathbf{z_2} &= \mathcal{S}^i(\mathbf{z_1}, \hat{\mathbf{z}}_{\mathbf{2}}[0], \mathscr{B}_1) = \hat{\mathbf{z}}_{\mathbf{2}}[0] \odot \mathscr{B}_1 \quad \text{for } i = 1, \\ \mathbf{z_{i+1}} &= \mathcal{S}^i(\mathbf{z_i}, \hat{\mathbf{z}}_{\mathbf{i+1}}[0], \mathscr{B}_i) = \mathbf{z_i} + (\hat{\mathbf{z}}_{\mathbf{i+1}}[0] \odot \mathscr{B}_i) \quad \text{for } i > 1, \end{aligned} \tag{4.4.11}$$

where $\odot$ denotes the element-wise (Hadamard) product.

Output. Finally, the output of $\hat{u}_{N,\theta}$ is the projection of $\mathbf{z_{N+1}}$ by the local transformation $\mathcal{Q}$ as Li et al. (2020b):

$$\hat{u}_{N,\theta} = \mathcal{Q}\left(\sum_{i=1}^{N+1}\left(\sum_{n=0}^{M-1}\mathbf{z_i}[n]\overline{\mathscr{B}_i\left(e^{i2\pi n/M};a_{1:i}\right)}\right)\odot\mathscr{B}_i\right). \tag{4.4.12}$$

4.5 Properties of AFMO

Connections to AFD theory. Adaptive Fourier decomposition (AFD) is a novel signal decomposition technique that leverages the Takenaka-Malmquist system and adaptive orthogonal bases Qian (2010); Qian et al. (2012). It admits a proved convergence of any signal $s \in \mathcal{H}$ such that $s = \sum_{i=1}^{\infty}\langle s, \mathscr{B}_i\rangle\mathscr{B}_i$ Qian et al. (2011); Wang et al. (2022) for the chosen orthonormal bases $\mathscr{B}_i$ Saitoh et al. (2016). Thus, the output of Equation 4.4.11 $\mathbf{z_{i+1}}$, is equivalent to the AFD operation, i.e., $\mathbf{z_{i+1}} = \sum_{k=1}^{i}\langle \mathbf{z_k}, \mathscr{B}_k\rangle\mathscr{B}_k$. Furthermore, the output in Equation 4.4.12 can be approximated as $\hat{u}_{N,\theta} = \mathcal{Q}\left(\sum_{i=1}^{N+1}\langle \mathbf{z_i}, \mathscr{B}_i\rangle\mathscr{B}_i\right) \approx \sum_{i=1}^{N+1}\langle \hat{u}_{i-1,\theta}, \mathscr{B}_i\rangle\mathscr{B}_i$, where $\hat{u}_{i-1,\theta} = \mathcal{Q}(\mathbf{z_i})$. This is also equivalent to the AFD operation. Thus, several theoretical properties of AFMO, including convergence and error bound (see theorems and proofs in Section 4.6), can be guaranteed with efficiently large layers, thanks to AFMO's deep connections with AFD theory.

Connections to Parnichkun et al. (2024). Parnichkun et al. (2024) proposed a state-free inference of SSMs by learning the coefficients of the rational transfer function H instead of the traditional state-space matrices A, B, and C Gu & Dao (2023), which is called rational transfer function (RTF) approach. Specifically, the RTF learns H as:

$$H(z) = h_0 + \frac{b_1 z^{-1} + b_2 z^{-2} + \cdots + b_n z^{-n}}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_n z^{-n}}, \tag{4.5.1}$$

where a_i, b_i, and h_0 are denominator coefficients, numerator coefficients, and feedthrough term, respectively. When it comes to AFMO, we push the formulation of transfer function in Equation 4.4.7 and learn the rational transfer function by learning the poles $a_{1:n}$ (for n

terms). Next, we show that our way of learning poles leads to a similar form of Equation 4.5.1 with n learned parameters (poles) as opposed to learning $2n+1$ parameters in RTF.

We consider a (finite) Blaschke product

$$H(z) \;=\; \prod_{j=1}^{n} \frac{1-p_j z}{z-p_j}, \qquad |p_j|<1, \tag{4.5.2}$$

and convert it into a single ratio of polynomials whose coefficients match the parameterization used to train SSMs. Denote numerator and denominator polynomials

$$B_{\text{poly}}(z)=\prod_{j=1}^{n}(z-p_j), \qquad A_{\text{poly}}(z)=\prod_{j=1}^{n}(1-p_j z), \tag{4.5.3}$$

so that $H(z)=\frac{A_{\text{poly}}(z)}{B_{\text{poly}}(z)}$. Let $d=\deg B_{\text{poly}}=\deg A_{\text{poly}}=n$. To obtain the form with a unit constant term in the denominator, divide numerator and denominator by z^d and then normalize:

$$\widetilde{H}(z) \;=\; \frac{\sum_{k=0}^{d}\alpha_k z^{-k}}{\sum_{k=0}^{d}\beta_k z^{-k}} \;\xrightarrow{\text{normalize}}\; h_0 \;+\; \sum_{k=1}^{d}\frac{b_k}{1}\,z^{-k} \;\Big/\; \Big(1 \;+\; \sum_{k=1}^{d} a_k z^{-k}\Big). \tag{4.5.4}$$

The SSM coefficients are then reduced as:

$$h_0=\frac{\alpha_0}{\beta_0}, \qquad b_k=\frac{\alpha_k}{\beta_0}, \qquad a_k=\frac{\beta_k}{\beta_0}, \quad k=1,\dots,d.$$

Example ($n=2$). With $p_1,p_2\in\mathbb{C}$, expand

$$B_{\text{poly}}(z)=(z-p_1)(z-p_2)=z^2-(p_1{+}p_2)z+p_1p_2,$$

$$A_{\text{poly}}(z)=(1-p_1z)(1-p_2z)=1-(p_1{+}p_2)z+(p_1p_2)z^2.$$

Divide by z^2 to get polynomials in z^{-1} and normalize by the denominator's constant term ($\beta_0=p_1p_2$), yielding

$$H(z) \;=\; \frac{1-(p_1{+}p_2)z^{-1}+(p_1p_2)z^{-2}}{p_1p_2-(p_1{+}p_2)z^{-1}+z^{-2}} \;=\; \frac{h_0+b_1z^{-1}+b_2z^{-2}}{1+a_1z^{-1}+a_2z^{-2}},$$

with

$$h_0=\frac{1}{p_1p_2}, \quad b_1=-\frac{p_1+p_2}{p_1p_2}, \quad b_2=1, \qquad a_1=-\frac{p_1+p_2}{p_1p_2}, \quad a_2=\frac{1}{p_1p_2}.$$

Efficient computation for large n. Direct symbolic expansion scales poorly. Instead, we multiply degree-1 polynomials using FFT-based convolution. Represent each factor by its coefficient vector:

$$(z - p_j) \leftrightarrow [1, -p_j], \qquad (1 - p_j z) \leftrightarrow [1, -p_j],$$

and iteratively convolve to form B_{poly} and A_{poly}. By the convolution theorem, polynomial multiplication is element-wise in the frequency domain, giving $\mathcal{O}(d \log d)$ complexity. After both polynomials are assembled, convert to z^{-1} by dividing by z^d, then normalize by the denominator's constant term to obtain $(h_0, \{a_k\}, \{b_k\})$ as in 4.5.4.

Computational complexity. In terms of computational complexity, AFMO has an overall computational complexity of $\mathcal{O}\big(N(M \log M + MD)\big) + \mathcal{O}(N_s M D)$. The former is from the processing block, whereas the latter comes from $\mathcal{P}$ and $\mathcal{Q}$. When M is treated as a constant with $M \ll N_s$ and a local decoder is used, the dominant cost reduces to $\mathcal{O}(N_s D) + \mathcal{O}(N\, M \log M)$. Consequently, the complexity grows linearly with the number of mesh points N_s. With mesh size fixed, it is approximately linear in the number of latent tokens M and the number of blocks N.

4.6 Theoretical Results of AFMO

Basic settings. Let $\mathbb{D} = \{z \in \mathbb{C} : |z| < 1\}$. Consider a reproducing kernel Hilbert space (RKHS) $(\mathcal{H}, \langle \cdot, \cdot \rangle_{\mathcal{H}})$ of complex-valued functions on $\mathbb{D}$ with the following properties.

Assumption 4.6.1 *There is a family of normalized reproducing kernels $\{e_a : a \in \mathbb{D}\} \subset \mathcal{H}$ such that*

$$e_a(z) = \frac{\sqrt{1 - |a|^2}}{1 - \overline{a} z} \in \mathcal{H}, \qquad \langle f, e_a \rangle_{\mathcal{H}} = f(a)\sqrt{1 - |a|^2} \quad \forall\, f \in \mathcal{H},\, a \in \mathbb{D}. \tag{4.6.1}$$

Given a pole sequence $a_{1:\infty} = (a_1, a_2, \dots) \subset \mathbb{D}$, define the Takenaka–Malmquist (TM) system by

$$\mathscr{B}_1(z) = e_{a_1}(z), \qquad \mathscr{B}_i(z) = e_{a_i}(z) \prod_{j=1}^{i-1} \frac{z - a_j}{1 - \overline{a_j} z} \quad (i \geq 2). \tag{4.6.2}$$

Assume $\{\mathscr{B}_i\}_{i\geq 1}$ is an orthonormal system in $\mathcal{H}$, and its closed linear span equals the model space

$$K_B := \overline{\operatorname{span}}\{\mathscr{B}_i : i \geq 1\} \subseteq \mathcal{H}, \tag{4.6.3}$$

where B is the Blaschke product with zeros $\{a_i\}$.

AFMO notation. Let $s \in \mathcal{H}$ be the latent target representation and $u^\star = \mathcal{Q}(s)$, where $\mathcal{Q} : \mathcal{H} \to \mathcal{U}$ is a Lipschitz decoder with constant $L_{\mathcal{Q}}$. Define the ideal TM coefficients and partial sums

$$c_i^\star := \langle s, \mathscr{B}_i \rangle_{\mathcal{H}}, \qquad s_N := \sum_{i=1}^{N} c_i^\star \, \mathscr{B}_i. \tag{4.6.4}$$

AFMO learns estimates $\widehat{c}_i$ of $c_i^\star$ (via an SSM in the frequency domain) and aggregates them through the skip connection:

$$z_{i+1} := z_i + \widehat{c}_i \, \mathscr{B}_i, \qquad z_1 := 0. \tag{4.6.5}$$

4.6.1 Aggregation identity and frequency-domain coefficient extraction

Lemma 4.6.1 *Under 4.6.5, one has, for every $N \in \mathbb{N}$,*

$$z_{N+1} = \sum_{i=1}^{N} \widehat{c}_i \, \mathscr{B}_i. \tag{4.6.6}$$

Proof. The proof is by induction. For $N = 1$, $z_2 = z_1 + \widehat{c}_1 \mathscr{B}_1 = \widehat{c}_1 \mathscr{B}_1$, so 4.6.6 holds. Assume 4.6.6 holds for N, i.e., $z_{N+1} = \sum_{i=1}^{N} \widehat{c}_i \, \mathscr{B}_i$. Then

$$z_{N+2} = z_{N+1} + \widehat{c}_{N+1} \mathscr{B}_{N+1} = \sum_{i=1}^{N+1} \widehat{c}_i \, \mathscr{B}_i,$$

which establishes the claim for $N + 1$. ∎

Lemma 4.6.2 *Suppose the i-th SSM has transfer function*

$$H_i(e^{i\omega}) = \overline{\mathscr{B}_i(e^{i\omega})}, \tag{4.6.7}$$

so that the block multiplies the input spectrum by $\overline{\mathscr{B}_i}$ and outputs the zero-lag correlation. If the discrete inner product used by AFMO is a consistent quadrature for $\langle\cdot,\cdot\rangle_{\mathcal{H}}$ on the class $\{s\}\cup\{\mathscr{B}_i\}$, then

$$\widehat{c}_i \to \langle s, \mathscr{B}_i\rangle_{\mathcal{H}} = c_i^\star \quad \textit{as the quadrature is refined.} \tag{4.6.8}$$

Proof. By 4.6.7, the block forms (pointwise on the grid) $Y_i = \overline{\mathscr{B}_i}\cdot s$ in the transform domain; the zero-lag correlation is the discretized inner product $\langle s, \mathscr{B}_i\rangle_{\text{disc}}$. Consistency of the quadrature implies $\langle s, \mathscr{B}_i\rangle_{\text{disc}} \to \langle s, \mathscr{B}_i\rangle_{\mathcal{H}}$ as the grid is refined. Hence $\widehat{c}_i \to c_i^\star$. ■

4.6.2 Convergence in the model space and projection error

Theorem 4.6.1 *Under Assumption 4.6.1, if AFMO recovers the exact coefficients $c_i^\star = \langle s, \mathscr{B}_i\rangle_{\mathcal{H}}$, then*

$$s_N := \sum_{i=1}^{N} c_i^\star \mathscr{B}_i \xrightarrow[N\to\infty]{\mathcal{H}} \Pi_{K_B} s, \tag{4.6.9}$$

the orthogonal projection of s onto K_B. Consequently,

$$\|u^\star - \mathcal{Q}(s_N)\| \le L_{\mathcal{Q}}\,\|s - \Pi_{K_B}s\|_{\mathcal{H}} + L_{\mathcal{Q}}\,\|\Pi_{K_B}s - s_N\|_{\mathcal{H}} \xrightarrow[N\to\infty]{} L_{\mathcal{Q}}\,\operatorname{dist}(s, K_B). \tag{4.6.10}$$

Proof. Because $\{\mathscr{B}_i\}$ is an orthonormal basis (ONB) of K_B, the Fourier expansion of $\Pi_{K_B}s$ in this ONB has coefficients $\langle s, \mathscr{B}_i\rangle_{\mathcal{H}}$, and the N-th partial sum equals s_N. Convergence in norm to the projection is standard for orthogonal series in a Hilbert space, giving 4.6.9. The bound 4.6.10 follows from Lipschitz continuity of $\mathcal{Q}$:

$$\|u^\star - \mathcal{Q}(s_N)\| = \|\mathcal{Q}(s) - \mathcal{Q}(s_N)\| \le L_{\mathcal{Q}}\|s - s_N\| \le L_{\mathcal{Q}}\big(\|s - \Pi_{K_B}s\| + \|\Pi_{K_B}s - s_N\|\big).$$

■

Remark 4.6.1 *No greedy or maximal selection is used. The MLP-generated poles determine K_B; AFMO converges to $\Pi_{K_B}s$, and to s whenever $s \in K_B$.*

4.6.3 Best N-term error and rates without greedy selection

Definition 4.6.1 *Let $\mathcal{D} := \{\mathscr{B}_i(\cdot; a_{1:i}) : a_{1:i} \in \mathbb{D}^i,\, i \in \mathbb{N}\}$ be the TM dictionary. Define the best N-term error*

$$E_N(s) \;:=\; \inf_{a_{1:N},\, c_{1:N}} \Big\| s - \sum_{i=1}^{N} c_i\, \mathscr{B}_i(\cdot; a_{1:i}) \Big\|_{\mathcal{H}}. \tag{4.6.11}$$

Theorem 4.6.2 *Let $\tilde{a}_{1:N}$ be the poles output by the MLP and set $c_i^\star = \langle s, \mathscr{B}_i(\cdot; \tilde{a}_{1:i})\rangle_{\mathcal{H}}$. If AFMO learns $\widehat{c}_i$, then*

$$\Big\| s - \sum_{i=1}^{N} \widehat{c}_i\, \mathscr{B}_i(\cdot; \tilde{a}_{1:i}) \Big\|_{\mathcal{H}} \;\le\; E_N(s) \;+\; \Delta_{pole}(N) \;+\; \Big(\sum_{i=1}^{N} |\widehat{c}_i - c_i^\star|^2 \Big)^{\frac{1}{2}}, \tag{4.6.12}$$

where

$$\Delta_{pole}(N) := \inf_{c_{1:N}} \Big\| s - \sum_{i=1}^{N} c_i\, \mathscr{B}_i(\cdot; \tilde{a}_{1:i}) \Big\|_{\mathcal{H}} - E_N(s) \;\ge 0. \tag{4.6.13}$$

Proof. Choose $a_{1:N}^{\text{best}}, c_{1:N}^{\text{best}}$ that attain (or ε-attain) $E_N(s)$ and denote

$$s_N^{\text{best}} := \sum_{i=1}^{N} c_i^{\text{best}} \mathscr{B}_i(\cdot; a_{1:i}^{\text{best}}).$$

Then

$$\begin{aligned}
\Big\| s - \sum_{i=1}^{N} \widehat{c}_i \mathscr{B}_i(\cdot; \tilde{a}_{1:i}) \Big\| &\le \| s - s_N^{\text{best}} \| + \Big\| s_N^{\text{best}} - \sum_{i=1}^{N} c_i^\star \mathscr{B}_i(\cdot; \tilde{a}_{1:i}) \Big\| + \Big\| \sum_{i=1}^{N} (c_i^\star - \widehat{c}_i) \mathscr{B}_i(\cdot; \tilde{a}_{1:i}) \Big\| \\
&\le E_N(s) + \Delta_{\text{pole}}(N) + \Big(\sum_{i=1}^{N} |c_i^\star - \widehat{c}_i|^2 \Big)^{1/2}.
\end{aligned}$$

The last inequality uses the definition of $\Delta_{\text{pole}}(N)$ and orthonormality of $\{\mathscr{B}_i(\cdot; \tilde{a}_{1:i})\}_{i=1}^N$. ∎

Corollary 4.6.1 *Assume for the fixed MLP-produced poles $\tilde{a}_{1:i}$ that the exact TM coefficients satisfy the weak-ℓ^p decay*

$$|c_i^\star|^* \le C\, i^{-1/p}, \qquad 0 < p < 2,$$

where $(|c_i^\star|^*)$ is the nonincreasing rearrangement. Then

$$\inf_{c_{1:N}} \Big\| s - \sum_{i=1}^{N} c_i\, \mathscr{B}_i(\cdot; \tilde{a}_{1:i}) \Big\|_{\mathcal{H}} = \mathcal{O}\big(N^{\frac{1}{2} - \frac{1}{p}}\big). \tag{4.6.14}$$

If, in addition, $\Delta_{pole}(N) = o(1)$ and $\big(\sum_{i=1}^{N} |\widehat{c}_i - c_i^\star|^2 \big)^{1/2} = o(1)$, then the AFMO error in 4.6.12 is $\mathcal{O}\big(N^{\frac{1}{2} - \frac{1}{p}}\big)$.

Proof. For an orthonormal system, the best N-term error equals the ℓ^2 tail of the rearranged coefficients. With $|c_i^\star|^* \le Ci^{-1/p}$ and $p < 2$,

$$\sum_{i>N} (|c_i^\star|^*)^2 \le C^2 \sum_{i>N} i^{-2/p} = \mathcal{O}\big(N^{1-\frac{2}{p}}\big),$$

hence the norm error (square root) is $\mathcal{O}(N^{\frac{1}{2}-\frac{1}{p}})$. ■

4.6.4 Learning and discretization errors

Assumption 4.6.2 *Each $\widehat{c}_i$ is obtained by ERM over m i.i.d. frequency samples using a hypothesis class with effective capacity d_{eff} under sub-Gaussian noise, so that*

$$\mathbb{E}\big[|\widehat{c}_i - c_i^\star|\big] = \mathcal{O}\Big(\sqrt{\tfrac{d_{\text{eff}}}{m}}\Big). \tag{4.6.15}$$

Lemma 4.6.3 *Let $\langle\cdot,\cdot\rangle_{\tilde{N}}$ be a discrete inner product (e.g., uniform frequency grid) that is a consistent quadrature for $\langle\cdot,\cdot\rangle_{\mathcal{H}}$ on the class generated by $\{s\} \cup \{\mathscr{B}_i\}$. Then there exists $\varepsilon_{\text{disc}}(\tilde{N}) \downarrow 0$ such that*

$$\big|\langle f, g\rangle_{\mathcal{H}} - \langle f, g\rangle_{\tilde{N}}\big| \le \varepsilon_{\text{disc}}(\tilde{N}) \qquad \text{for all } f \in \{s\},\ g \in \{\mathscr{B}_i\}_{i\ge 1}. \tag{4.6.16}$$

Proof. Since point evaluations are continuous linear functionals in an RKHS and the involved functions are continuous on compact subsets, standard quadrature consistency yields 4.6.16. (If f, g are analytic in an annulus around the unit circle, one gets exponential rates; under Sobolev regularity, algebraic rates.) ■

Theorem 4.6.3 *Under Assumptions 4.6.1 and 4.6.2 and Lemma 4.6.3, the AFMO output after N blocks and $\tilde{N}$ grid points satisfies*

$$\|u^\star - \hat{u}_{N,\theta}\| \ \le\ L_{\mathcal{Q}}\Big(E_N(s) + \Delta_{pole}(N) + \Big(\sum_{i=1}^{N} |\widehat{c}_i - c_i^\star|^2\Big)^{1/2}\Big) \ +\ \varepsilon_{\text{disc}}(\tilde{N}), \tag{4.6.17}$$

with $\mathbb{E}[|\widehat{c}_i - c_i^\star|] = \mathcal{O}(\sqrt{d_{\text{eff}}/m})$ and $\varepsilon_{\text{disc}}(\tilde{N}) \to 0$ as $\tilde{N} \to \infty$.

Proof. Apply Theorem 4.6.2 to bound the latent $\mathcal{H}$-error. Then use Lipschitz continuity of $\mathcal{Q}$ to transfer the bound to the output space. The discretization error adds $\varepsilon_{\text{disc}}(\tilde{N})$ due to 4.6.16. ■

4.6.5 Stability to pole perturbations

Lemma 4.6.4 *For $a, b \in \mathbb{D}$ and $z \in \mathbb{D}$,*

$$\Big|\frac{1}{1-\overline{a}z} - \frac{1}{1-\overline{b}z}\Big| \le \frac{|a-b|}{(1-|a|)(1-|b|)}, \tag{4.6.18}$$

$$\Big|\sqrt{1-|a|^2} - \sqrt{1-|b|^2}\Big| \le \frac{|a-b|}{\sqrt{1-\max\{|a|,|b|\}^2}}, \tag{4.6.19}$$

and for $F(z;a) = \dfrac{z-a}{1-\overline{a}z}$,

$$|F(z;a) - F(z;b)| \le \frac{4\,|a-b|}{(1-|a|)(1-|b|)}, \qquad |F(z;a)| \le 1. \tag{4.6.20}$$

Proof. For 4.6.18,

$$\frac{1}{1-\overline{a}z} - \frac{1}{1-\overline{b}z} = \frac{(\overline{a}-\overline{b})z}{(1-\overline{a}z)(1-\overline{b}z)},$$

and $|1-\overline{a}z| \ge 1-|a||z| \ge 1-|a|$, $|z| \le 1$, yielding the bound. For 4.6.19, use the mean-value theorem on $x \mapsto \sqrt{1-x}$ with $x = |a|^2, |b|^2$ and $||a|^2 - |b|^2| \le |a-b|(|a|+|b|) \le 2|a-b|$. For 4.6.20, expand

$$F(z;a) - F(z;b) = \frac{(b-a) + (\overline{a}-\overline{b})z^2 + (a\overline{b} - b\overline{a})z}{(1-\overline{a}z)(1-\overline{b}z)},$$

and bound the numerator by $C|a-b|$ for $|z| \le 1$, while the denominator is bounded below by $(1-|a|)(1-|b|)$. ∎

Theorem 4.6.4 *Let $a_{1:i}, \tilde{a}_{1:i} \in \mathbb{D}$ with $|\tilde{a}_j - a_j| \le \delta_j$. Then there exist constants $C_i > 0$ (depending on $a_{1:i}$) such that*

$$\|\mathscr{B}_i(\cdot;\tilde{a}_{1:i}) - \mathscr{B}_i(\cdot;a_{1:i})\|_{\mathcal{H}} \;\le\; C_i \sum_{j=1}^{i} \frac{\delta_j}{1-|a_j|}. \tag{4.6.21}$$

Consequently, for any coefficients $\widehat{c}_i$,

$$\Big\| \sum_{i=1}^{N} \widehat{c}_i\, \mathscr{B}_i(\cdot;\tilde{a}_{1:i}) - \sum_{i=1}^{N} \widehat{c}_i\, \mathscr{B}_i(\cdot;a_{1:i}) \Big\|_{\mathcal{H}} \;\le\; \Big(\sum_{i=1}^{N} |\widehat{c}_i|\, C_i \Big) \Big(\sum_{j=1}^{N} \frac{\delta_j}{1-|a_j|} \Big). \tag{4.6.22}$$

Proof. Write

$$\mathscr{B}_i(\cdot;a_{1:i}) = e_{a_i} \prod_{j=1}^{i-1} F(\cdot;a_j), \qquad \mathscr{B}_i(\cdot;\tilde{a}_{1:i}) = e_{\tilde{a}_i} \prod_{j=1}^{i-1} F(\cdot;\tilde{a}_j).$$

Use the product telescoping identity

$$\prod_{k=1}^{i} P_k - \prod_{k=1}^{i} Q_k = \sum_{k=1}^{i} \Big(\prod_{j<k} P_j\Big)(P_k - Q_k)\Big(\prod_{j>k} Q_j\Big),$$

with $P_1 = e_{\tilde{a}_i}$, $Q_1 = e_{a_i}$, and $P_k = F(\cdot; \tilde{a}_{k-1})$, $Q_k = F(\cdot; a_{k-1})$ for $k \geq 2$. Taking sup-norms on $\mathbb{D}$ and using $|F(\cdot; a)| \leq 1$,

$$\|\mathscr{B}_i(\cdot; \tilde{a}_{1:i}) - \mathscr{B}_i(\cdot; a_{1:i})\|_\infty \;\leq\; \|e_{\tilde{a}_i} - e_{a_i}\|_\infty + \sum_{j=1}^{i-1} \|F(\cdot; \tilde{a}_j) - F(\cdot; a_j)\|_\infty.$$

Apply Lemma 4.6.4 to bound each term by a constant times $\delta_j/(1 - |a_j|)$. Since evaluation functionals are continuous and the kernel is bounded on compact subsets, there exists an embedding constant C_{emb} with $\|f\|_{\mathcal{H}} \leq C_{\text{emb}}\|f\|_\infty$ on the set considered; thus 4.6.21 follows with C_i absorbing all constants. Finally,

$$\Big\| \sum_{i=1}^{N} \widehat{c}_i\big(\mathscr{B}_i(\cdot; \tilde{a}_{1:i}) - \mathscr{B}_i(\cdot; a_{1:i})\big)\Big\|_{\mathcal{H}} \leq \sum_{i=1}^{N} |\widehat{c}_i|\, \|\mathscr{B}_i(\cdot; \tilde{a}_{1:i}) - \mathscr{B}_i(\cdot; a_{1:i})\|_{\mathcal{H}},$$

giving 4.6.22. ∎

4.6.6 End-to-end convergence without greedy selection

Theorem 4.6.5 *Assume:*

1. $s \in K_B$;
2. $\sum_{i=1}^{\infty} \mathbb{E}[|\widehat{c}_i - c_i^\star|^2]^{1/2} < \infty$ *(as sample size $m \to \infty$ and model capacity increase);*
3. $\varepsilon_{\text{disc}}(\tilde{N}) \to 0$ *as* $\tilde{N} \to \infty$.

Then

$$\lim_{N\to\infty} \|u^\star - \hat{u}_{N,\theta}\| = 0.$$

Proof. Since $s \in K_B$ and $\{\mathscr{B}_i\}$ is an ONB of K_B, Theorem 4.6.1 gives $s_N \to s$ in $\mathcal{H}$. In 4.6.17, for this fixed pole sequence one has $E_N(s) = \Delta_{\text{pole}}(N) = 0$. Using (2) and (3), we obtain $\|u^\star - \hat{u}_{N,\theta}\| \to 0$. ∎

4.6.7 Connection of SSM to correlation and AFMO output

Proposition 4.6.1 *With $H_i(e^{i\omega}) = \overline{\mathscr{B}_i(e^{i\omega})}$, the i-th SSM block computes $\widehat{c}_i \approx \langle z_i, \mathscr{B}_i \rangle_{\mathcal{H}}$. Hence, by Lemma 4.6.1, after N blocks*

$$z_{N+1} = \sum_{i=1}^{N} \widehat{c}_i \, \mathscr{B}_i, \qquad \hat{u}_{N,\theta} = \mathcal{Q}(z_{N+1}). \tag{4.6.23}$$

Proof. The coefficient claim follows from Lemma 4.6.2 applied to z_i in place of s. The aggregation identity is Lemma 4.6.1. The last equality is the definition of $\mathcal{Q}$. ■

Corollary 4.6.2 *All latent-space error bounds transfer to the PDE output space via*

$$\|u^\star - \hat{u}_{N,\theta}\| \leq L_{\mathcal{Q}} \left\| s - \sum_{i=1}^{N} \widehat{c}_i \mathscr{B}_i \right\| + \varepsilon_{\text{disc}}(\tilde{N}).$$

4.7 Numerical Experiments

To illustrate the effectiveness of AFMO, we conduct numerical experiments with multiple baseline neural operators on diverse datasets including three categories: (i) regular grids: 2-D Darcy flow equation and 2-D Navier-Stokes equation Li et al. (2020b), (ii) irregular geometries: plasticity, airfoil, pipe, and elasticity Li et al. (2023b), (iii) PDEs with singularities: European option pricing under the Black-Scholes equation, and 3-D Brusselator (reaction-diffusion) equation from Cao et al. (2024).

Metric. In the training and evaluation stage, we utilize relative L^2 error as the metric for accuracy for all problems:

$$\text{Rel-}L^2 = \frac{1}{\mathcal{N}} \sum_{i=1}^{\mathcal{N}} \frac{||\mathcal{G}_\theta(a_i) - \mathcal{G}(a_i)||_{L^2}}{||\mathcal{G}(a_i)|_{L^2}}, \tag{4.7.1}$$

where $\mathcal{N}$ denotes the number of samples. We also consider training time, the number of parameters, and/or GPU memory usage as metrics for computational efficiency.

Implementation details. For baselines, we follow the implementation settings of their works. Note that the architecture of FNO Li et al. (2020b) has been updated after publication, we evaluate FNO using the newest architecture. For AFMO, we train 500 epochs on all datasets. We use AdamW optimizer with decoupled weight decay 1×10^{-5}, base learning rate 2×10^{-4}, and a cosine decay schedule Loshchilov & Hutter (2017) with a linear warm-up over the first 10% of total steps. The nonlinearity is GELU inside the processing blocks. We clip global grad-norm at 0.5 each step. Unless stated otherwise, we use batch size 16, latent width 128, 64 latent tokens, 32 adaptive poles, and 4 processing blocks with SSM state size 16, depthwise 1-D convolution (per channel) of kernel size 4, channel expansion ratio 2. Experiments are conducted on a Linux workstation running Ubuntu (kernel 6.14, glibc 2.39) with Python 3.13.5 (Anaconda), PyTorch 2.8.0+cu129 (CUDA 12.9), an AMD Ryzen 9 9950X (16-core) processor, and a single NVIDIA GeForce RTX 4090 (48 GB) GPU. CUDA is enabled.

4.7.1 Numerical results of benchmark datasets

Table 16 shows the comprehensive comparison with various baselines on the six benchmark problems. Among those problems, N-S and Darcy flow datasets apply regular grids, elasticity dataset uses point clouds, whereas others are generated under structured meshes Li et al. (2020b, 2023b). AFMO consistently outperforms existing SOTA models by an average improvement of 28.42%. In particular, for airfoil, Darcy, and N-S datasets, the relative L^2 error decreased more than 30% compared to the existing SOTA models, demonstrating the superior performance of AFMO compared to existing frequency-, transformer-, and Mamba-based models when solving complex dynamics and handling irregular geometries. To solve the complex dynamics, Tiwari et al. (2025) incorporates latent representations and SSMs, which can be considered as integral kernels without orthogonality. Meanwhile, ONO Xiao et al. (2023b) uses an orthogonal attention to ensure orthogonality. Numerical results on irregular geometries, including elasticity ($0.0050 \rightarrow 0.0043$), plasticity ($0.0007 \rightarrow 0.0006$),

airfoil (0.0041 → 0.0020), and pipe (0.0026 → 0.0023), show that the systematic integration of orthonormal kernels and SSMs leads to an exact AFD approximation and in turn improves PDE solution accuracy in irregular geometries.

Table 16: Relative L^2 error comparisons of AFMO with baselines across six benchmark datasets. Lower relative L^2 error is better. We quantify the improvement as the gain of AFMO relative to the L^2 error of the second best model. **Bold** means the best model, underline means the second best model, red means the third best model, and blue means the fourth best model.

Models	**Elasticity**	**Plasticity**	**Airfoil**	**Pipe**	**N-S**	**Darcy**
FNO Li et al. (2020b)	0.0229	0.0074	0.0138	0.0067	0.0417	0.0052
U-FNO Wen et al. (2022)	0.0239	0.0039	0.0269	0.0056	0.2231	0.0183
F-FNO Tran et al. (2021)	0.0263	0.0047	0.0078	0.0070	0.2322	0.0077
LNO Wang & Wang (2024)	0.0052	0.0029	0.0051	0.0026	0.0845	0.0049
ONO Xiao et al. (2023b)	0.0118	0.0048	0.0061	0.0052	0.1195	0.0076
WMT Gupta et al. (2021)	0.0359	0.0076	0.0075	0.0077	0.1541	0.0082
Galerkin Cao (2021)	0.0240	0.0120	0.0118	0.0098	0.1401	0.0084
LSM Wu et al. (2023)	0.0218	0.0025	0.0059	0.0050	0.1535	0.0065
OFormer Li et al. (2022b)	0.0183	0.0017	0.0183	0.0168	0.1705	0.0124
Transolver Wu et al. (2024)	0.0062	0.0013	0.0053	0.0047	0.0879	0.0059
Transolver++ Luo et al. (2025)	0.0064	0.0014	0.0051	0.0027	0.1010	0.0089
LAMO Tiwari et al. (2025)	0.0050	0.0007	0.0041	0.0038	0.0460	0.0039
AFMO (ours)	**0.0043**	**0.0006**	**0.0020**	**0.0023**	**0.0278**	**0.0021**
Improvement	14.0%	14.3%	51.2%	11.5%	33.3%	46.2%

Computational Efficiency. To explore the computational efficiency of AFMO, we focus on Darcy and airfoil problems. On average, AFMO reaches 46.2% and 51.2% reduction in training time over SOTA models in these two problems, as shown in Figure 24. With light

architectures and small GPU memory, AFMO achieves the best training speed. Compared to the SOTA neural operator, LaMO Tiwari et al. (2025), AFMO is $\sim 1.2\times$ faster and $\sim 2.5\times$ lighter with similar GPU memory. Instead of using orthogonal attention as in ONO Xiao et al. (2023b), AFMO employs bases in the orthogonal form (Equation 4.4.5), which does not require an orthogonalization process, thereby saving $\sim 2.7\times$ in training time and $\sim 3\times$ in GPU memory compared to ONO.

Scalability. We examine the computational scalability of AFMO on 2-D Darcy flow problem. From Table 17, we observe that, as the grid dimension changes from 64 to 128 (N_s becomes 4 times larger), both training and inference times increase approximately linearly (by about 4 times), which aligns with the computational complexity result mentioned earlier. The memory usage remains relatively constant with only a slight increase. This reflects the architectural characteristics of AFMO, where the main computations (SSM blocks) are performed on M latent tokens rather than on N_s physical points, and thus the memory footprint is largely decoupled from the input resolution N_s.

Table 17: AFMO is computationally scalable with respect to input resolution N_s.

Grid dimensions	**Grid size N_s**	**Training time (sec/epoch)**	**Inference time (sec/epoch)**	**GPU memory (GB)**
64×64	4096	14.0	0.007	2.3
128×128	16384	52.5	0.28	2.4
256×256	65536	205.0	1.12	2.7

Learned pole distributions across layers. To understand how the adaptive poles are selected and evolved, Figures 26 and 27 showcase the distributions per layer for 2-D Darcy flow and 3-D Brusselator equations. The learned poles of AFMO on Darcy flow problem tend to approach to the boundary of the unit disk, while those on the Brusselator problem tend to be in the interior of the unit disk. The reason is that, the challenging characteristics and singularities of the Darcy flow problem are located at the boundaries, and then more

adaptive poles would be put there. Meanwhile, the complexity of the Brusselator problem does not come from the boundaries. It comes from the local, non-linear reaction that happens at every single point inside the domain. Therefore, most of the learned poles should be put inside the unit disk.

4.7.2 European Options Pricing

To demonstrate the versatility of AFMO in solving different PDEs in different contexts, we consider the European calls/puts problem modeled using the Black–Scholes equation with continuous dividend yield q. For contract/market parameters $(r, \sigma, q, K, T, \texttt{is\_call})$, the price $V(S,t)$ satisfies the Black–Scholes equation Barles & Soner (1998):

$$\partial_t V + \tfrac{1}{2}\sigma^2 S^2\, \partial_{SS} V + (r-q) S\, \partial_S V - rV = 0, \quad S \in [S_{\min}, S_{\max}],\ t \in [0,T], \tag{4.7.2}$$

with terminal payoff $V(S,T) = \max(\pm(S-K), 0)$ (+ sign for calls, − for puts) and the linear boundary conditions $V(0,t) = 0$ for calls, $V(0,t) = Ke^{-r(T-t)}$ for puts, and controlled growth as $S \to \infty$. This problem setting leads to two singular features: (i) the terminal payoff kink at $S = K$ (jump in $\partial_S V$, concentration in $\partial_{SS} V$) as $t_{\text{norm}} \uparrow 1$; and (ii) degeneracy near small S as a result of the $S^2 \partial_{SS} V$ diffusion term. Our goal is to learn the operator that maps the parameters $(r, \sigma, q, K, T, \texttt{is\_call})$ to the price $V(S,t)$. By comparing AFMO with a set of top-performing solvers, we observe from Table 18 that average improvements of 25%, 4.1%, and 52.7% have been achieved by AFMO in terms of relative L^2 error, training time, and parameter counts, respectively. This indicates that AFMO can accurately and efficiently solve PDE problems with singular features.

4.7.3 Ablation studies

Adaptive kernels vs. static kernels. We now consider the need and benefits of using adaptive kernels. A kernel is adaptive when its parameterization (e.g., coefficients) varies with the input. In this work, the formulation of Equation 4.4.5 varies with the learned poles

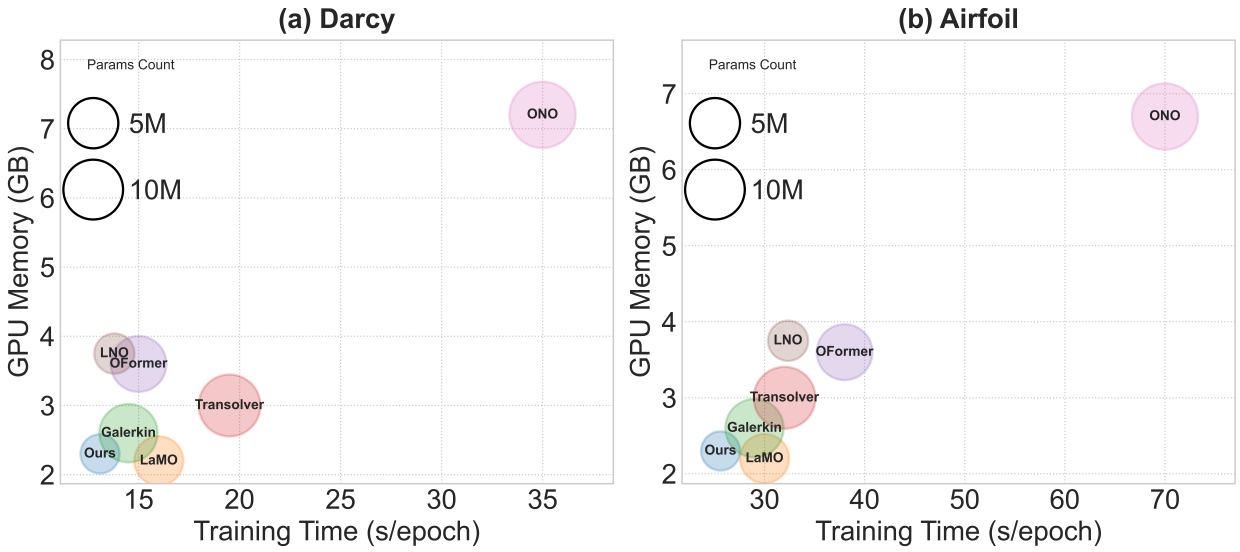


Figure 24: Comparisons of training time per epoch, number of parameters, and GPU memory among existing SOTA models on (a) Darcy and (b) airfoil, where AFMO exhibits the strongest incremental gains.

$a_{1:i}$ and thus is an adaptive kernel. We also randomly fix the value of $a_{1:i}$ for static kernels for comparison. Furthermore, although a total of i poles are needed for i-th processing block, one can still identify more poles and select the best i poles for implementation. Table 19 shows the relative L^2 error results across six benchmark datasets and the European options (EO) dataset. We find that, using adaptive kernels, the relative L^2 errors reduce significantly compared to using static poles for all benchmark problems considered. In fact, the relative L^2 errors when selecting only 4 poles are lower than those when selecting 32 static poles.

Need for ensuring orthogonality. To understand how orthogonal kernels affect AFMO performance, we conduct another ablation study by using non-orthogonal kernels (i.e., Equation 4.4.4) in the AFMO framework. In this case, the transfer functions used in SSMs are $H_i(e^{i\omega}) = \overline{(1-|a_i|^2)\sum_{n=0}^{\infty}(\overline{a_i})^n e^{in\omega}}$ to match the output of AFD operation. Without orthogonality, AFMO experiences higher relative L^2 error, especially for problems with irregular geometries (e.g., airfoil $0.0020 \to 0.0083$ and elasticity $0.0043 \to 0.0094$). At the same time, the training time also increases by $\sim 50.3\%$ per epoch on average across all six benchmark

Table 18: European option pricing: relative L^2 error and resource profile. Lower is better for error, GPU memory, and training time. Parameter counts shown in millions. **Bold** = best, underline = second best, and red = third best.

Models	**Rel. L^2 (↓)**	**Training Time (sec/epoch, ↓)**	**Params (M, ↓)**
FNO Li et al. (2020b)	0.0016	25.1	3.78
LNO Wang & Wang (2024)	0.0010	21.7	2.56
Transolver Wu et al. (2024)	0.0012	22.3	5.91
LAMO Tiwari et al. (2025)	0.0008	22.5	3.52
AFMO (ours)	**0.0006**	**20.8**	**1.21**

Table 19: Relative L^2 error comparisons for **Static** vs. **Adaptive** kernels across seven benchmarks. Lower is better.

Models	**Number of poles**	**Elasticity**	**Plasticity**	**Airfoil**	**Pipe**	**N-S**	**Darcy**	**EO**
AFMO (static)	32	0.0097	0.0021	0.0067	0.0072	0.1103	0.0174	0.0035
AFMO (adaptive)	4	0.0056	0.0012	0.0033	0.0029	0.0311	0.0057	0.0014
	6	0.0051	0.0010	0.0031	0.0027	0.0298	0.0047	0.0010
	8	0.0049	0.0008	0.0027	0.0025	0.0281	0.0036	0.0009
	16	0.0046	0.0008	0.0023	0.0028	0.0290	0.0029	0.0008
	32	**0.0043**	**0.0006**	**0.0020**	**0.0023**	**0.0278**	**0.0021**	**0.0006**
	64	0.0048	0.0007	0.0036	0.0031	0.0372	0.0046	0.0009

datasets. This shows that the use of orthogonal kernels (i.e., TM systems) helps improve both accuracy and computational efficiency of AFMO solver.

Choice of SSMs. Finally, we evaluate the choice of bidirectional SSMs in AFMO compared to unidirectional SSMs and multidirectional SSMs. Results in Figure 25 indicate that the choice of bidirectional SSMs in AFMO consistently outperforms other two SSMs in all datasets.

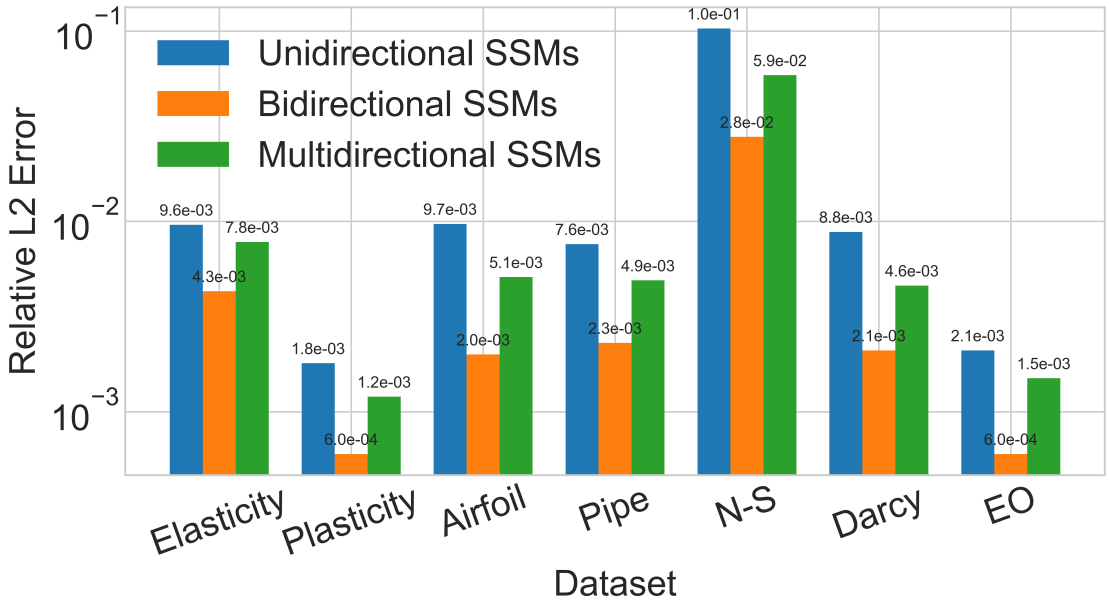


Figure 25: Contribution of three SSMs across seven benchmark datasets. Note that we do not apply weights shared for all experiments. Lower is better.

4.7.4 Experiment using real-world noisy dataset

To validate AFMO's performance on noisy real-world datasets, we perform experiments using the latex glove DIC (Digital Image Correlation) original dataset You et al. (2022). The goal is to learn the mechanical response of a nitrile glove sample directly from experimental data, without assuming a known constitutive law. The goal is to predict the displacement field at the current loading step. The input includes the spatial coordinates, the displacement field from the previous step, and the current boundary displacement. We compare

the performance of AFMO to the current SOTA of this dataset, IFNO, as well as FNO as follows. To ensure fair comparison, we conduct experiments using the same settings as IFNO with the number of hidden layers ranging from 3 to 12.

Table 20: Relative L^2 error of AFMO and other baselines using the latex glove DIC (Digital Image Correlation) original dataset.

Number of hidden layers	**AFMO**	**IFNO**	**FNO**
3	2.87E-02 ± 4.29E-04	3.43E-02 ± 4.96E-04	3.40E-02 ± 4.09E-04
6	2.50E-02 ± 3.28E-04	3.34E-02 ± 4.53E-04	3.84E-02 ± 4.21E-04
12	2.32E-02 ± 4.20E-04	3.32E-02 ± 4.41E-04	4.66E-02 ± 1.47E-03

In addition, You et al. (2022) also reported the results of generalized Mooney-Rivlin (GMR) model in two settings. The relative L^2 errors of GMR model fitting and GMR inverse analysis are 3.30E-01 and 2.91E-01, respectively. We can observe that our AFMO consistently outperforms other models in every L. Finally, the best reported result of IFNO is 3.30E-02 ± 4.63E-04 when $L = 24$ You et al. (2022). Although we do not conduct the experiment $L = 24$, our AFMO still performs better than the best result of IFNO.

3-D Brusselator problem. We introduce a new 3-D Brusselator (diffusion-reaction equation) problem using the dataset from Laplace neural operator (LNO) Cao et al. (2024). The Brusselator problem is formulated as:

$$D\frac{\partial^2 y}{\partial x^2} + ky^2 - \frac{\partial y}{\partial t} = f(x,t), \tag{4.7.3}$$

where $y(x,t)$ represents the concentration of chemical substances or particles at location x and time t, $f(x,t)$ is the source term and A is the amplitude of the source term. In this problem, the diffusion coefficient, $D = 0.01$, and the reaction rate, $k = 0.01$.

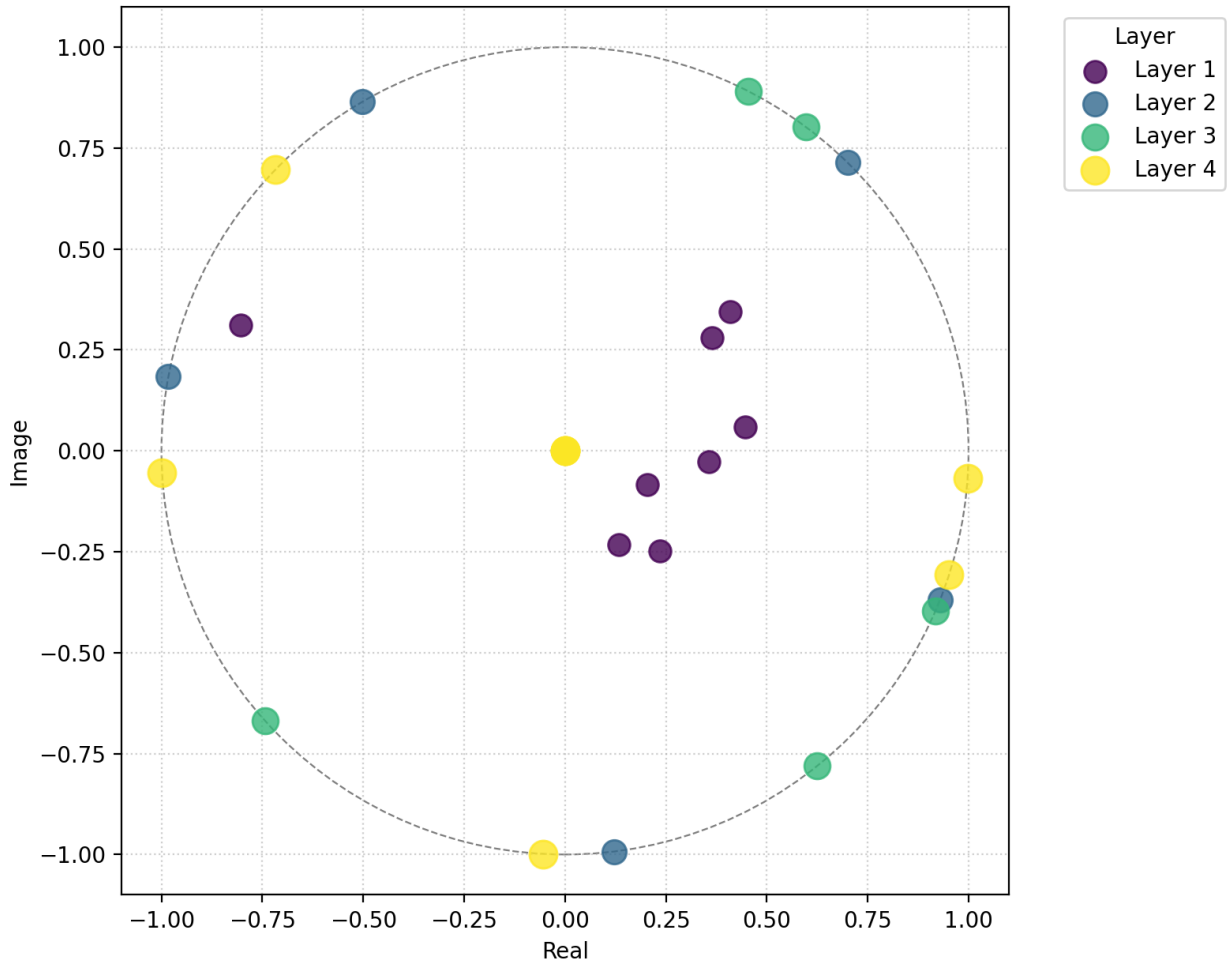


Figure 26: Learned poles distribution for the 2-D Darcy flow equation.

4.8 Distribution of selected poles reflects problem characteristics

To understand how AFMO's pole selection process is adaptive to the characteristics and nature of the problem, we illustrate the learned pole distributions for the 2-D Darcy flow problem and 3-D Brusselator problem in Figures 26. To clarify, here we give a brief overview of the visualization results: The distribution of selected poles for the 2-D Darcy flow problem is shown in Figures 26 and 27, respectively.

We observe that, across the layers, the learned poles of AFMO on Darcy flow problem tend to approach to the boundary of the unit disk, while those on the Brusselator problem tend to be in the interior of the unit disk. The reason is that, Darcy flow problem is an

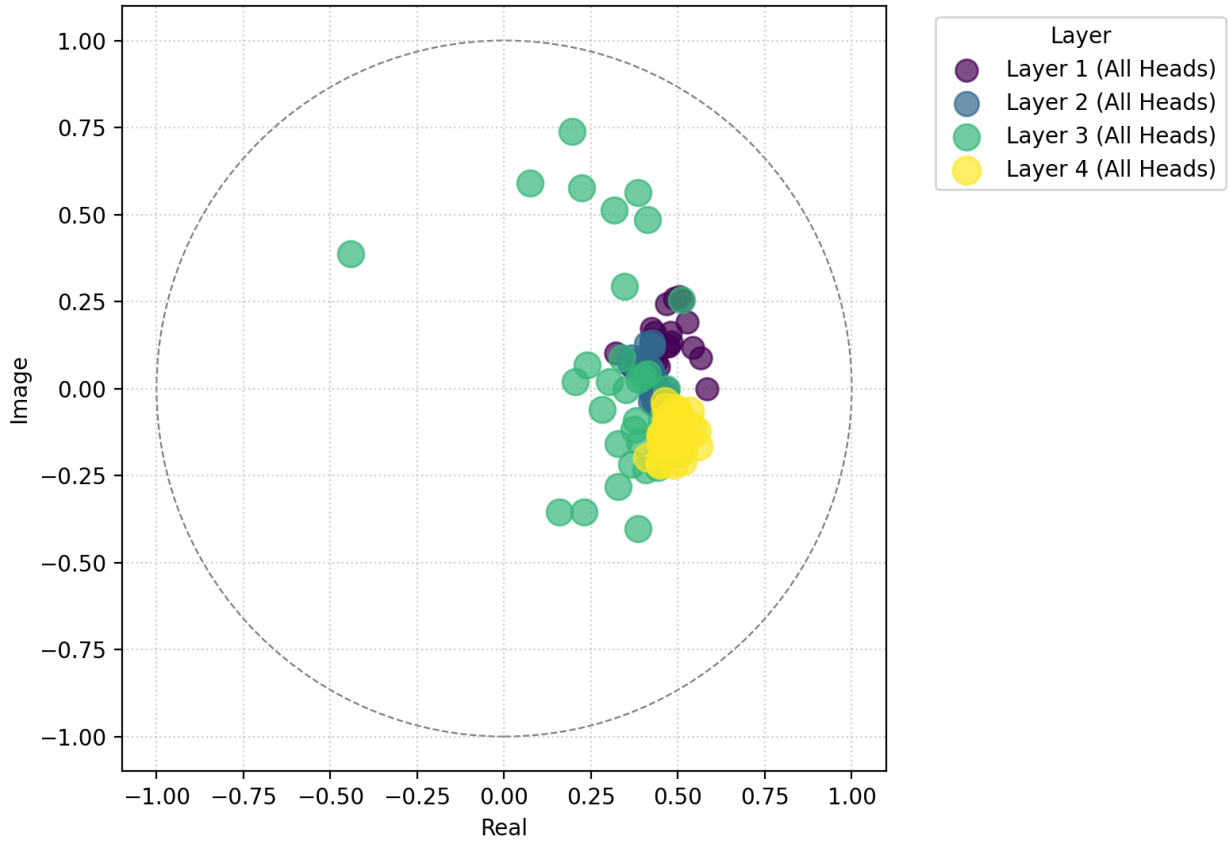


Figure 27: Learned poles distribution for the 3-D Brusselator equation.

elliptic equation, which is a smoothing operator. Thus, even though the input coefficient (the permeability) is very rough and discontinuous, the solution inside the domain will be well-behaved. Therefore, the challenging characteristics and singularities of the Darcy flow problem are located at the boundaries, and then more adaptive poles would be put there. Meanwhile, the complexity of the Brusselator problem does not come from the boundaries. It comes from the local, non-linear reaction that happens at every single point inside the domain. Therefore, most of the learned poles should be put inside the unit disk.

CHAPTER V

INVERSE PROBLEMS IN BANACH SPACE

5.1 Preliminaries

5.1.1 Inverse problem in Hilbert vs. Banach spaces

A Hilbert space is a complete inner product space (i.e., a vector space equipped with an inner product that induces a norm), and every Cauchy sequence in the space converges with respect to this norm. The interpretability given by the inner product has enabled rigorous convergence analysis and comprehensive application regularization techniques to be applied for solving inverse problems in Hilbert spaces over the last decades. However, for numerous inverse problems in PDEs, the reasons for using a Hilbert space setting seem to be based on conventions rather than an appropriate and realistic model choice. In fact, it has been shown that the nature of Hilbert spaces cannot accurately capture the structures of parameter space for many PDEs, and often a Banach space setting would be closer to reality Schuster et al. (2012). As a generalization of the Hilbert space, a Banach space is a complete normed vector space (i.e., a vector space equipped with a norm such that every Cauchy sequence in the space converges with respect to this norm), and the main difference between Hilbert and Banach spaces is the existence of an inner product and thus orthogonality. Banach spaces are more suitable for solving inverse problems involving sparsity, discontinuities, or measure-valued representations.

5.1.2 Adaptive Fourier decomposition (AFD)

Adaptive Fourier decomposition (AFD) is a novel signal decomposition technique, which is essentially established as a new approximation theorem in a reproducing kernel Hilbert space (RKHS) sparsely in a given domain Ω as $\sum_{i=1}^{\infty}\langle s, \mathscr{B}_i\rangle\mathscr{B}_i$ for the chosen orthonormal bases $\mathscr{B}_i$ Qian (2010); Qian et al. (2012); Saitoh et al. (2016). Compared to conventional signal decomposition approaches, AFD achieves higher accuracy and significant computational speedup. AFD was first proposed for the Hardy space Qian (2010); Qian et al. (2011, 2012), then extended to the Bergman space Wu et al. (2022), random signals Qian (2022), and manifolds Song & Sun (2022).

For classic AFD in RKHS, the sparse bases $\{\mathscr{B}_i\}_i$ are made orthonormal to each other by applying the Gram-Schmidt orthogonalization process to the normalized reproducing kernels associated with different "**poles**", which are a set of complex numbers $\{a_i\}_i$ that are adaptively selected. For instance, in classic AFD in Hardy space H^2 (a specific type of Hilbert space consisting of holomorphic functions defined on the unit disk), a common choice of reproducing kernel is the normalized Szegö kernel, defined as $e_a(z) = \frac{\sqrt{1-|a|^2}}{1-\bar{a}z}$, where a belongs to the unit disk. Then, to construct the orthonormal bases $\mathscr{B}_i$, one selects a sequence of distinct poles $\{a_i\}_i$, substitutes them into the normalized Szegö kernel expression, and applies the Gram-Schmidt orthogonalization process on $e_{a_i}(z)$.

To adaptively select the sequence of poles such that convergence of AFD approximation is ensured, one shall follow the so-called "**maximal selection principle**" Song & Sun (2022), such that the resulting $|\langle s, \mathscr{B}_i\rangle|$ is as large as possible. This is similar to a greedy search algorithm. Specifically, to select the next pole a_i given $i-1$ already selected poles, $a_1, \ldots, a_{i-1}$ (hence bases $\mathscr{B}_1, \ldots, \mathscr{B}_{i-1}$), the corresponding orthonormal basis $\mathscr{B}_i$ needs to satisfy:

$$|\langle s, \mathscr{B}_i\rangle| \geq \rho_i \sup\left\{\langle s, \mathscr{B}_i^{b_i}\rangle \ | b_i \in \Omega \backslash \{a_1, \ldots, a_{i-1}\}\right\}, \tag{5.1.1}$$

where $0 < \rho_0 \leq \rho_i < 1$, $\mathscr{B}_1^{b_1} = \frac{k_{b_1}}{\|k_{b_1}\|_{H(\Omega)}}$ and $\mathscr{B}_i^{b_i} = \frac{k_{b_i} - \sum_{j=1}^{i-1}\langle k_{b_i}, \mathscr{B}_j\rangle\mathscr{B}_j}{||k_{b_i} - \sum_{j=1}^{i-1}\langle k_{b_i}, \mathscr{B}_j\rangle\mathscr{B}_j||_{H(\Omega)}}$. Here, k_{b_i} is

the reproducing kernel at b_i. Under the under maximal selection principle, the convergence of AFD approximation has been proven in Song & Sun (2022). In actual implementation, however, we often want to strengthen Equation (5.1.1) by introducing an extra **bias term** denoted as $\gamma_i > 0$ to enhance convergence. Essentially, this ensures that $|\langle s, \mathscr{B}_i \rangle|$ is always greater than the RHS of Equation (5.1.1) by at least γ_i. And the resulting updated maximal selection principle formulation becomes:

$$\gamma_i \leq |\langle s, \mathscr{B}_i \rangle| - \rho_i \sup \left\{ \langle s, \mathscr{B}_i^{b_i} \rangle \ | b_i \in \Omega \backslash \{a_1, \ldots, a_{i-1}\} \right\}. \tag{5.1.2}$$

5.2 AFD in reproducing kernel Banach space (RKBS)

The classic AFD, which operates on RKHS, leverages orthonormal bases. On the other hand, one cannot properly define orthogonality and inner product in a Banach space. Instead, we extend the inner product definition by adopting the concept of "**dual pairing**", denoted as $\langle \cdot, \cdot \rangle_{\mathcal{B},\mathcal{B}^*}$ for the primal RKBS $\mathcal{B}$ and its dual $\mathcal{B}^*$. Essentially, a dual pairing is a non-degenerate bilinear map between two vector spaces that produces a scalar Brezis & Brézis (2011). With this, we will adaptively identify poles following a similar maximal selection principle, such that $|\langle r_{i-1}, J(\mathscr{B}_i) \rangle_{\mathcal{B},\mathcal{B}^*}|$ is as large as possible. Here, $r_{i-1} = s - s_{i-1}$ is the **residual** between the true signal s and its $(i-1)$-th decomposed component, s_{i-1}. And J is called **duality map**, which satisfies $J(\mathscr{B}_i) \in \mathcal{B}^*$ and $\langle \mathscr{B}_i, J(\mathscr{B}_i) \rangle_{\mathcal{B},\mathcal{B}^*} = ||\mathscr{B}_i||_{\mathcal{B}}^2$. Specifically, to select the next pole a_i, the corresponding basis $\mathscr{B}_i$ needs to satisfy:

$$\gamma_i \leq |\langle r_{i-1}, J(\mathscr{B}_i) \rangle_{\mathcal{B},\mathcal{B}^*}| - \rho_i \sup_{\mathscr{B}_i \in \mathcal{D}} \{ |\langle r_{i-1}, J(\mathscr{B}_i) \rangle_{\mathcal{B},\mathcal{B}^*}| \}, \tag{5.2.1}$$

which is analogous to Equation (5.1.2). In actual implementation, we find that setting γ_i to a fixed value of 0.5 works well for most problem settings. Here, we define set $\mathcal{D} = \{\mathscr{B}_i | a_i \in \mathcal{B}\}$ (note that we no longer need to exclude already selected poles as the concept of orthogonality does not hold in RKBS anymore), γ_i is the bias term, and $0 < \rho_0 \leq \rho_i < 1$. With this, the AFD operations in RKBS give $s = \sum_{i=1}^{\infty} \langle r_{i-1}, J(\mathscr{B}_i) \rangle_{\mathcal{B},\mathcal{B}^*} \mathscr{B}_i$. We remark that, to the best

of our knowledge, such generalization of the AFD theory to Banach spaces has not been proposed before.

Mathematical Properties of AFD in RKBS

.

The AFD operations in RKBS are conceptually illustrated in Algorithm 1. Note that here, we do not yet consider the bias term γ_i.

Algorithm 1 AFD in RKBS

Require: Target function α_{true}, dictionary $\mathcal{D}$, duality map J, parameter ρ_i, tolerance ε, maximum levels of decomposition N

Initialize: $\alpha_0 \leftarrow 0$, $r_0 \leftarrow \alpha_{\text{true}}$

for $i = 1$ **to** N **do**

  Kernel selection:

  Choose $\mathscr{B}_i \in \mathcal{D}$ such that

$$|\langle r_{i-1}, \mathscr{B}_i \rangle_{\mathcal{B},\mathcal{B}^*}| \geq \rho_i \sup_{\mathscr{B} \in \mathcal{D}} |\langle r_{i-1}, \mathscr{B} \rangle_{\mathcal{B},\mathcal{B}^*}|$$

  Coefficient selection:

  Compute $a_i \leftarrow \langle r_{i-1}, J(\mathscr{B}_i) \rangle_{\mathcal{B},\mathcal{B}^*}$

  Update:

  $\alpha_i \leftarrow \alpha_{n-1} + a_i \mathscr{B}_i$

  $r_i \leftarrow \alpha_{\text{true}} - \alpha_i$

  if $\|r_i\| < \varepsilon$ **then**

    break

  end if

end for

Here, we assume the following assumption holds. Especially, Assumption 3 is the property (Γ) from Ganichev & Kalton (2009).

1. $\mathcal{B}$ is a uniformly smooth and uniformly convex Banach space of functions on a domain Ω, with dual space $\mathcal{B}^*$.

2. $\mathcal{D}$ is defined as $\{\mathscr{B}_{a_i} | a_i \in \mathcal{B}\}$, where $\mathscr{B}_{a_i}(\cdot) = \frac{K_z(\cdot, a_i)}{||K_z(\cdot, a_i)||_{\mathcal{B}}}$ with poles a_i. The linear span of $\mathcal{D}$ is dense in $\mathcal{B}$.

3. There exists $\tilde{C} > 0$ such that for all $x \in \mathcal{B}$ with $\|x\|_{\mathcal{B}} = 1$ and $y \in \mathcal{B}$ with $\langle y, J(x)\rangle_{\mathcal{B},\mathcal{B}^*} = 0$, we have $\langle y, J(x+y)\rangle_{\mathcal{B},\mathcal{B}^*} \leq \tilde{C}(\|x+y\|_{\mathcal{B}} - 1)$.

With these assumptions, we first show that the residual r_i is decreasing with respect to i:

Theorem 5.2.1 *At each iteration $i \geq 1$, it holds that:*

$$\|r_i\|_{\mathcal{B}} < \|r_{i-1}\|_{\mathcal{B}},$$

unless $\langle J(r_{i-1}), \mathscr{B}_i\rangle_{\mathcal{B},\mathcal{B}^*} = 0$.

Proof. Define $\tilde{\phi}(a) = \|r_{i-1} - a\mathscr{B}_i\|_{\mathcal{B}}$. From Assumption 1, the uniform convexity of $\mathcal{B}$ ensures $\tilde{\phi}$ is strictly convex with a unique minimizer. Moreover, $\tilde{\phi}$ is Gâteaux differentiable in terms of its norm due to the uniform smoothness of $\mathcal{B}$. Its directional derivative is $\frac{d}{dt}\|r_{i-1} + th\|_{\mathcal{B}}\Big|_{t=0} = \frac{\Re\langle J(r_{i-1}), h\rangle_{\mathcal{B},\mathcal{B}^*}}{\|r_{i-1}\|_{\mathcal{B}}}$. Now, we aim to find a direction h based on $\mathscr{B}_i$ that decreases the norm, i.e., $\Re\langle J(r_{i-1}), h\rangle_{\mathcal{B},\mathcal{B}^*} < 0$. From , we choose $h = -\frac{\langle J(r_{i-1}), \mathscr{B}_i\rangle_{\mathcal{B},\mathcal{B}^*}}{|\langle J(r_{i-1}), \mathscr{B}_i\rangle_{\mathcal{B},\mathcal{B}^*}|}$, then

$$\begin{aligned}
\Re\langle J(r_{i-1}), h\rangle_{\mathcal{B},\mathcal{B}^*} &= \Re\langle J(r_{i-1}), -\frac{\langle J(r_{i-1}), \mathscr{B}_i\rangle_{\mathcal{B},\mathcal{B}^*}}{|\langle J(r_{i-1}), \mathscr{B}_i\rangle_{\mathcal{B},\mathcal{B}^*}|}\mathscr{B}_i\rangle_{\mathcal{B},\mathcal{B}^*} \\
&= -\Re(\frac{\langle J(r_{i-1}), \mathscr{B}_i\rangle_{\mathcal{B},\mathcal{B}^*}}{|\langle J(r_{i-1}), \mathscr{B}_i\rangle_{\mathcal{B},\mathcal{B}^*}|}\langle J(r_{i-1}), \mathscr{B}_i\rangle_{\mathcal{B},\mathcal{B}^*}) \qquad (5.2.2) \\
&= -|\langle J(r_{i-1}), \mathscr{B}_i\rangle_{\mathcal{B},\mathcal{B}^*}|.
\end{aligned}$$

If $\langle J(r_{i-1}), \mathscr{B}_i\rangle_{\mathcal{B},\mathcal{B}^*} \neq 0$, the results of Equation (5.2.2) is negative. So, $\tilde{\phi}(t) < \tilde{\phi}(0)$ for $t > 0$. Strict convexity then implies $\tilde{\phi}(a_i) < \tilde{\phi}(0)$ if $a_i \neq 0$, hence $\|r_i\|_{\mathcal{B}} < \|r_{i-1}\|_{\mathcal{B}}$. ■

Then, the convergence of AFD in RKBS can be shown as:

Theorem 5.2.2 *The sequence $\{\alpha_i\}_{i=1}^N$ generated by Algorithm 1 satisfies:*

$$\lim_{i\to\infty} \|\alpha_{true} - \alpha_i\|_{\mathcal{B}} = 0.$$

Proof. From Theorem 5.2.1, the sequence $\{\|r_i\|_{\mathcal{B}}\}$ is nonincreasing and its lower bound is 0. Thus, the sequence $\{\|r_i\|_{\mathcal{B}}\}$ converges. Assume its limit $l > 0$ for contradiction. Uniform convexity from Assumption 1 implies the bounded $\{r_i\}$ has weak limit points r' with $\|r'\|_{\mathcal{B}} \leq l$ by weak lower semicontinuity. By properties of the duality map J, we have:

$$\sup_{\mathscr{B}\in\mathcal{D}} |\langle r_{i-1}, J(\mathscr{B})\rangle_{\mathcal{B},\mathcal{B}^*}| = \|r_{i-1}\|_{\mathcal{B}}. \tag{5.2.3}$$

Combining Equation (5.2.3) and $|\langle r_{i-1}, \mathscr{B}_i\rangle_{\mathcal{B},\mathcal{B}^*}| \geq \rho_i \sup_{\mathscr{B}\in\mathcal{D}} |\langle r_{i-1}, \mathscr{B}\rangle_{\mathcal{B},\mathcal{B}^*}|$ leads to:

$$|\langle r_{i-1}, J(\mathscr{B}_i)\rangle_{\mathcal{B},\mathcal{B}^*}| \geq \rho_i \|r_{i-1}\|_{\mathcal{B}}. \tag{5.2.4}$$

From Xu & Roach (1991), we have the following inequality by setting $p = 2$:

$$\|x+y\|_{\mathcal{B}}^2 \leq \|x\|_{\mathcal{B}}^2 + 2\langle J(x), y\rangle_{\mathcal{B},\mathcal{B}^*} + 2\|x\|_{\mathcal{B}}^2 \rho\left(\frac{\|y\|_{\mathcal{B}}}{\|x\|_{\mathcal{B}}}\right), \tag{5.2.5}$$

where ρ is the modulus of smoothness. By setting $x = r_{i-1}$ and $y = -a_i\mathscr{B}_i$ and assuming that a_i is chosen such that $\langle J(r_{i-1}), a_i\mathscr{B}_i\rangle_{\mathcal{B},\mathcal{B}^*} > 0$, the term $2\langle J(x), y\rangle_{\mathcal{B},\mathcal{B}^*} = -2\langle J(r_{i-1}), a_i\mathscr{B}_i\rangle_{\mathcal{B},\mathcal{B}^*} = -2|\langle J(r_{i-1}), a_i\mathscr{B}_i\rangle_{\mathcal{B},\mathcal{B}^*}|$. Then, the Equation (5.2.5) becomes:

$$\begin{aligned}\|r_i\|_{\mathcal{B}}^2 \leq& \|r_{i-1}\|_{\mathcal{B}}^2 - 2|\langle J(r_{i-1}), a_i\mathscr{B}_i\rangle_{\mathcal{B},\mathcal{B}^*}| \\ &+ 2\rho\left(\frac{|a_i|}{\|r_{i-1}\|_{\mathcal{B}}}\right)\|r_{i-1}\|_{\mathcal{B}}^2.\end{aligned} \tag{5.2.6}$$

Assume, for contradiction, that $\sup_{\mathscr{B}\in\mathcal{D}} |\langle r_i, J(\mathscr{B})\rangle_{\mathcal{B},\mathcal{B}^*}| \not\to 0$. Then there exists $\delta > 0$ such that for infinitely many i, $\sup_{\mathscr{B}\in\mathcal{D}} |\langle r_i, J(\mathscr{B})\rangle_{\mathcal{B},\mathcal{B}^*}| \geq \delta$. For such i, the Equation (5.2.4) yields $|\langle r_{i-1}, J(\mathscr{B}_i)\rangle_{\mathcal{B},\mathcal{B}^*}| \geq \rho_i\delta$. Then, we normalize $u = \frac{r_{i-1}}{\|r_{i-1}\|_{\mathcal{B}}}$ and $b_i = \frac{a_i}{\|r_{i-1}\|_{\mathcal{B}}}$, so $\frac{r_i}{\|r_{i-1}\|_{\mathcal{B}}} = u - b_i\mathscr{B}_i$. Then, the normalized version of Equation (5.2.4) is $|\langle u, J(\mathscr{B}_i)\rangle_{\mathcal{B},\mathcal{B}^*}| \geq \frac{\rho_i\delta}{\|r_{i-1}\|_{\mathcal{B}}} \geq \frac{\rho_i\delta}{l}$. Next, we utilize the decomposition technique as $y = -b_i\mathscr{B}_i = \alpha J(u) + z$, where $\alpha \in \mathbb{C}$ and $\langle z, J(u)\rangle_{\mathcal{B},\mathcal{B}^*} = 0$. Since $\langle y, J(u)\rangle = \alpha$, we have $|\langle u, J(y)\rangle_{\mathcal{B},\mathcal{B}^*}| = |\alpha|$. And $|\alpha| \geq \rho_i\delta' b_i$ with $\delta' \geq \delta/l$. The optimality condition is $\langle J(u+y), y\rangle_{\mathcal{B},\mathcal{B}^*} = 0$, leading to $\alpha\langle J(u+y), J(u)\rangle_{\mathcal{B},\mathcal{B}^*} = -\langle J(u+y), z\rangle_{\mathcal{B},\mathcal{B}^*}$. If $b_i < k\rho_i$ for a small $k > 0$, then y small implies that for $\varepsilon > 0$, $\|J(u+y) - J(u)\|_{\mathcal{B}} < \varepsilon$. Therefore, we also have $\|\alpha - \langle J(u+y), z\rangle_{\mathcal{B},\mathcal{B}^*}\|_{\mathcal{B}} < \varepsilon$. From Assumption 3, since $\langle z, J(u)\rangle_{\mathcal{B},\mathcal{B}^*} = 0$, $|\langle z, J(u+y)\rangle_{\mathcal{B},\mathcal{B}^*}| \leq \tilde{C}(\|u+y\|_{\mathcal{B}} - 1)$. With $b_i \to 0$,

by smoothness, we obtain $\|u+y\|_{\mathcal{B}}-1=O(b_i)$, so $|\alpha|\to 0$, contradicting $|\alpha|\geq \rho_i\delta' b_i$. Thus, it holds that $b_i\geq k\rho_i$ for $k>0$ depending on $\tilde{C}$ and the smoothness modulus. Substitute them into Equation (5.2.5) leads to the term $-2|\langle J(r_{i-1}), a_i\mathscr{B}_i\rangle_{\mathcal{B},\mathcal{B}^*}|$ is at least $c'\rho_i^2\|r_{i-1}\|_{\mathcal{B}}^2$ for the constant c', and the term $2\|r_{i-1}\|_{\mathcal{B}}^2\rho(b_i)$ is $o(\rho_i^2\|r_{i-1}\|_{\mathcal{B}}^2)$ since $\rho(\cdot)=o(\cdot)$, yielding

$$\|r_i\|_{\mathcal{B}}^2\leq \|r_{i-1}\|_{\mathcal{B}}^2-c\rho_i^2\|r_{i-1}\|_{\mathcal{B}}^2, \tag{5.2.7}$$

for $c>0$. Assume that there exists $\delta>0$ such that for infinitely many indices i_k (with $\|r_{i_k}\|_{\mathcal{B}}^2\to l^2$), the decrease satisfies

$$\|r_{i_k}\|_{\mathcal{B}}^2-\|r_{i_k+1}\|_{\mathcal{B}}^2\geq c\rho_{i_k}^2\|r_{i_k}\|_{\mathcal{B}}^2\geq c\rho^2 l^2/2=:d>0, \tag{5.2.8}$$

where the last inequality holds for sufficiently large k since $\|r_{i_k}\|_{\mathcal{B}}^2\to l^2$ and $\rho_{i_k}\geq\rho>0$. Let $S=\sum_{k=1}^{\infty}(\|r_{i_k}\|_{\mathcal{B}}^2-\|r_{i_k+1}\|_{\mathcal{B}}^2)\geq\sum_{k=1}^{\infty}d=\infty$. Since the overall sequence decreases by at most $\|r_1\|_{\mathcal{B}}^2-l^2<\infty$, but includes $S=\infty$, which leads to a contradiction. Therefore, for large M, the remaining decrease after i_M is at least $\sum_{k>M}d=\infty$, but it is upper bounded by $\|r_{i_M}\|_{\mathcal{B}}^2-l^2<\infty$. Between i_k and i_{k+1}, the decrease is nonnegative. Thus, after m such steps,

$$\|r_{i_{k+m}}\|_{\mathcal{B}}^2\leq\|r_{i_k}\|_{\mathcal{B}}^2-md\to-\infty\quad(m\to\infty), \tag{5.2.9}$$

which is impossible for norms. The exponential form follows recursively. If decrease $\geq c\rho^2\|r_i\|_{\mathcal{B}}^2$ at each of m steps, finally it leads to

$$\|r_{i+m}\|_{\mathcal{B}}^2\leq(1-c\rho^2)^m\|r_i\|_{\mathcal{B}}^2\to 0, \tag{5.2.10}$$

contradicting convergence to $l^2>0$.

Thus, $\sup_{\mathscr{B}\in\mathcal{D}}|\langle r_i, J(\mathscr{B})\rangle_{\mathcal{B},\mathcal{B}^*}|\to 0$. For weak limit r' of $\{r_{i_k}\}$, $\langle r_{i_k}, J(\mathscr{B})\rangle\to\langle r', J(\mathscr{B})\rangle=0$ for all $\mathscr{B}\in\mathcal{D}$. Bijectivity of J and density of $\operatorname{span}\mathcal{D}$ imply $\{J(\mathscr{B})\}$ generates dense functionals, forcing $r'=0$. All weak limits are 0, but $\|r_i\|_{\mathcal{B}}\to l>0=\|0\|_{\mathcal{B}}$, contradicting weak lower semicontinuity unless $l=0$. Hence, $\lim_{i\to\infty}\|r_i\|_{\mathcal{B}}=0$, so $\lim_{i\to\infty}\|\alpha_{\text{true}}-\alpha_i\|_{\mathcal{B}}=0$. ■

5.3 AFD-guided Neural Operator Design

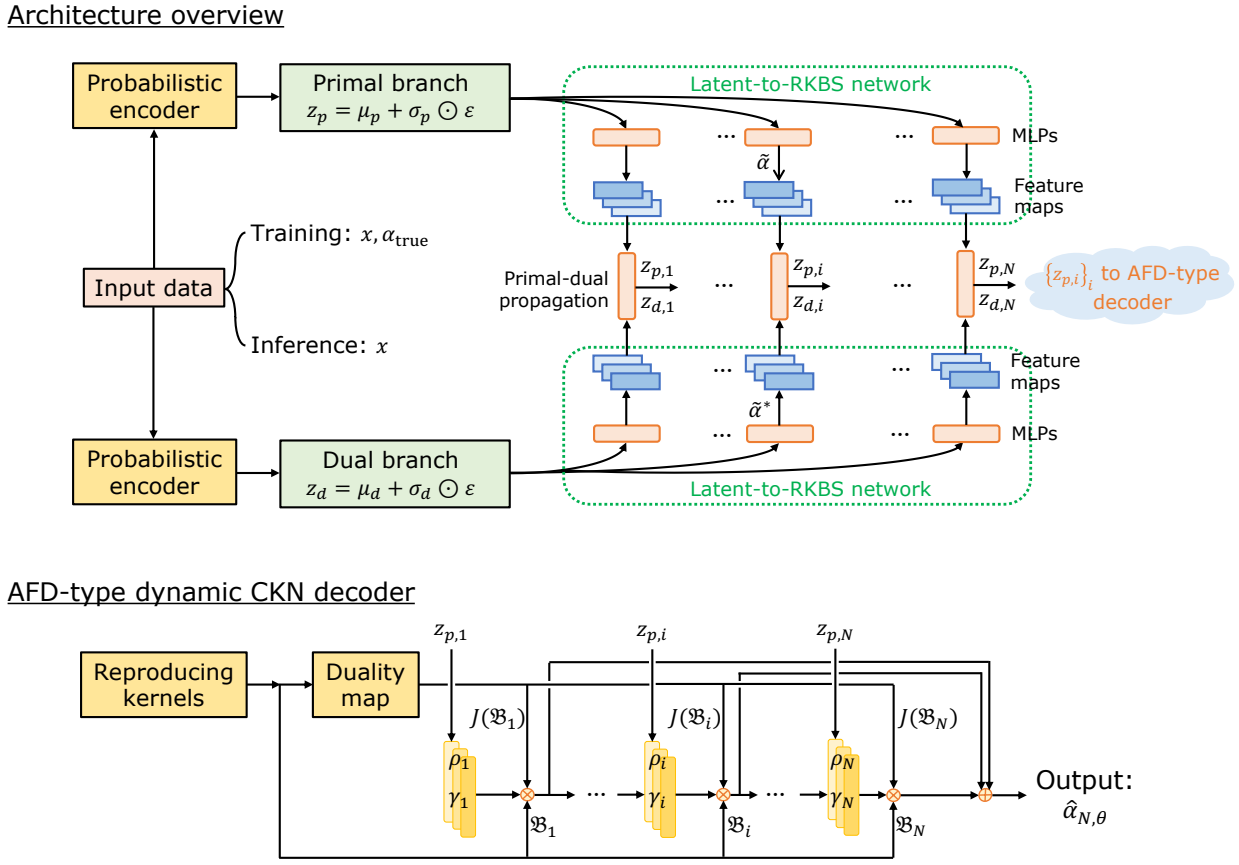


Figure 28: Our proposed AFDONet-inv framework, whose design is guided by the AFD theory and operation, for solving inverse PDE problems in Banach space. Note that the elements in the figure are static representations of the corresponding theoretical component, whereas the actual computation follows the dynamic, recursive update defined by Equation (5.3.9).

Once we establish the theoretical framework for AFD in Banach space, we design a tailored neural operator architecture, which we name as AFDONet-inv, that reproduces and realizes this theoretical framework. AFDONet-inv is an AFD-based VAE architecture (see Figure 28) to solve inverse PDE problems in Banach space. After the encoder, AFDONet-inv identifies the closest RKBS where the latent variables reside using a latent-to-RKBS network. Subsequently, AFDONet-inv reconstructs the PDE parameters by adaptively selecting the

poles in a specially designed decoder network, thereby resembling the AFD operation.

5.3.1 Neural architecture

The encoder network. The encoder network maps the input $x \in \mathbb{R}^d$ (e.g., PDE solutions in training dataset during training or test dataset during inference stage) to latent variables in both the primal and dual latent spaces, which correspond to a Banach space B and its dual space B^*, respectively. Note that during training, since the PDE parameters α_{true} only appear in the loss function, they are part of training dataset but not part of the input x to the encoder. Each encoding branch follows the standard VAE framework. That is, for the primal branch:

$$\left(\mu_p(x),\, \log \sigma_p^2(x)\right) = W_{p,2}\left(\phi\left(W_{p,1}x\right)\right), \tag{5.3.1}$$

$$z_p = \mu_p(x) + \sigma_p(x) \odot \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, I), \tag{5.3.2}$$

where $W_{p,1} \in \mathbb{R}^{w\times d}$, $W_{p,2} \in \mathbb{R}^{2r\times w}$ are the weight matrices, $\phi(\cdot)$ is the activation function, and $z_p \in \mathbb{R}^r$ is the latent variable in the primal space.

Similarly, for the dual branch:

$$\left(\mu_d(x),\, \log \sigma_d^2(x)\right) = W_{d,2}\left(\phi\left(W_{d,1}x\right)\right), \tag{5.3.3}$$

$$z_d = \mu_d(x) + \sigma_d(x) \odot \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, I), \tag{5.3.4}$$

where $W_{d,1} \in \mathbb{R}^{W_e\times d}$ and $W_{d,2} \in \mathbb{R}^{2r\times W_e}$ are the dual encoder weight matrices, and $z_d \in \mathbb{R}^r$ is the dual latent variable in B^*.

The latent-to-RKBS network. Given the latent variables z_p and z_d, our goal is to determine their values in the parameter space, which lies in a RKBS. To do this, we extend the latent-to-kernel idea from Lu et al. (2020a) and design a latent-to-RKBS network to project the latent variable z_p to its nearest RKBS where the kernel is constructed. And then, its dual space in which the latent variable z_d lie is obtained via the duality map $J(v) = \|v\|_{\mathfrak{p}}^{\mathfrak{p}-2}\cdot|v|^{\mathfrak{p}-2}\cdot v$ in the form of $L^{\mathfrak{p}}$ norm. First, z_p and z_d are respectively mapped to the

parameter space $\tilde{\alpha} \in B$ and $\tilde{\alpha}^* \in B^*$ through two identical multilayer perceptron networks (MLPs). Then, feature maps $\mathrm{FM}(\cdot)$ will project $\tilde{\alpha}$ and $\tilde{\alpha}^*$ onto their corresponding RKBS $\mathcal{B}$ and its dual $\mathcal{B}^*$, respectively. In other words, the feature map network is designed to map the latent vector $z_p \in \mathbb{R}^r$ to N scalar coefficients in $\mathbb{R}$ needed for the kernel decomposition (i.e., $\mathrm{FM} : \mathbb{R}^r \to \mathbb{R}^N$). The latent-to-RKBS network learns the feature maps from a Banach space B to its nearest RKBS $\mathcal{B}$, where the reproducing kernel K_z for any pair of encoder inputs (i.e., PDE solutions in training or test dataset) is given by:

$$K_z(x_1, x_2) = \sum_{i=1}^{N} \mathrm{FM}_i(z_p) \cdot k_i(x_1, x_2). \tag{5.3.5}$$

In Equation 5.3.5, N is the total number of basis kernels, $\mathrm{FM}_i(z_p)$ is the i-th feature map coefficient for latent variable z_p, and $\{k_i(x_1, x_2)\}_i$ are the learned basis kernels are based on a Fourier spectral kernel formulation. Here, $k_i(x_1, x_2)$ contains a series of learned parameters in the spectral domain. Since these basis kernels are in RKBS, which is closed under finite linear combinations, K_z lies in the RKBS as well.

Primal-dual propagation. The concept of dual pairing is realized in a primal-dual propagation network. Consisting of a primal net and a dual net, it propagates and refines the feature representations $\mathrm{FM}(\tilde{\alpha}) \in \mathcal{B}$ and $J\left(\mathrm{FM}(\tilde{\alpha})\right) \in \mathcal{B}^*$, which correspond to the latent variables z_p and z_d, respectively. Here, the primal net performs the spectral convolution $\mathcal{S}(f)(x)$, which transforms a given function f into the frequency domain via a 2D Fourier transform Li et al. (2020b):

$$\mathcal{S}(f)(x) = \mathcal{F}^{-1}\left[\chi(\xi) \cdot \mathcal{F}\left(f\right)(\xi) \cdot W(\xi)\right](x), \tag{5.3.6}$$

where $\mathcal{F}[\cdot](\xi)$ is the Fourier transform at ξ, $\mathcal{F}^{-1}[\cdot](x)$ is its inverse Fourier transform at x, $\chi(\xi)$ denotes the mode selector, and $W(\xi)$ refers to the learnable weights in the frequency domain. From an RKBS perspective, the spectral convolution $\mathcal{S}(f)(x)$ is equivalent to a nonlocal kernel operator $\sum_\xi w_\xi \cdot \langle f, \varphi_\xi \rangle \cdot \psi_\xi(x)$, where w_ξ are learnable weights in the frequency domain, $\varphi_\xi \in \mathcal{B}^*$ is the basis in the dual space, and $\psi_\xi \in \mathcal{B}$ is the biorthogonal

primal basis which satisfies $\langle \psi_\xi, \varphi_\eta \rangle = \delta_{\xi\eta} = 1$ if $\xi = \eta$ and $= 0$ otherwise Zhang et al. (2009). On the other hand, the dual net performs a point-wise convolution $\mathcal{C}(f)(x)$ as Hua et al. (2018):

$$\mathcal{C}(f)(x) = \sum_{c'=1}^{C} W_{cc'} f^{(c')}(x), \tag{5.3.7}$$

where C is the number of input channels of f, $f^{(c')}(x)$ refers to the value of f at x and channel c', and $W_{cc'}$ denotes the learned map from channel c' to channel c. This point-wise convolution resembles a Dirac-type kernel integral on its domain Ω as:

$$\mathcal{C}(f)(x) = \int_\Omega \delta(x-y) \cdot W \cdot f(y) \mathrm{d}y. \tag{5.3.8}$$

Finally, the residuals in the primal space and its dual space after each layer are updated and propagated to the corresponding latent variables as:

$$\begin{aligned} z_{p,i} &= \mathrm{GELU} \circ \mathrm{BN}\left(z_{p,i-1} + \mathcal{S}(z_{p,i-1}) + \mathcal{C}(z_{p,i-1})\right), \\ z_{d,i} &= z_{d,i-1} + \mathcal{S}(z_{d,i-1}) + \mathcal{C}(z_{d,i-1}), \end{aligned} \tag{5.3.9}$$

where $z_{p,i}, z_{d,i}$ are the residuals of i-th layer in the primal space and its dual space with $z_{p,0} = \mathrm{FM}(\tilde{\alpha})$ and $z_{d,0} = J\left(\mathrm{FM}(\tilde{\alpha})\right)$, respectively. The initial latent vector, z_p, is used to define $\tilde{\alpha}$, and $\tilde{\alpha}$ then defines the initial decoder input, $z_{p,0}$. Here, GELU denotes the GELU activation function and BN is the batch normalizing transform Ioffe & Szegedy (2015).

It can be shown that z_p and z_d in Equation (5.3.9) actually correspond to the residual r_i and its dual $J(r_i)$ defined in the AFD theory as:

Theorem 5.3.1 *Let $\mathcal{B}$ be a uniformly smooth and uniformly convex Banach space, $r_i \in \mathcal{B}$ be the residual at step i in the AFD process, z_p and z_d are defined in Equation* (5.3.9)*. Then, for any $\varepsilon > 0$, there exists a choice of parameters $W(\xi)$, $W_{cc'}$, and* BN *scalars such that:*

$$\|z_{p,i} - r_i\|_{\mathcal{B}} < \varepsilon, \text{ and } \|z_{d,i} - J(r_i)\|_{\mathcal{B}^*} < \varepsilon.$$

Note that, even though the variables $z_{p,i}$ are computationally represented as tensors, Theorem 5.3.1 shows that the learned network outputs behave as if they were the true theoretical Banach space residuals r_i.

Without loss of generality, we consider the case $\mathcal{B} = L^p([0, L]^d)$ for $1 < p < \infty$. Before giving the complete proof of Theorem 5.3.1, we introduce some lemmas as follows.

Lemma 5.3.1 *Let $\mathcal{B} = L^p([0, L]^d)$ for $1 < p < \infty$, a uniformly smooth Banach space admitting a Fourier transform $\mathcal{F}$. Let $(\psi_\xi)_{\xi \in \mathbb{Z}^d}$ be the normalized Fourier basis $\psi_\xi(x) = e^{2\pi i \xi \cdot x/L}$, with Fourier coefficients $\hat{f}(\xi) = \mathcal{F}(f)(\xi)$. Define the partial sum operators $S_n : \mathcal{B} \to \mathcal{B}$ by $S_n f = \sum_{|\xi|_\infty \le n} \hat{f}(\xi)\psi_\xi$, where $|\xi|_\infty = \max_j |\xi_j|$. Then, there exists $C'_p > 0$ (depending on p and d, but independent of n) such that $\|S_n f\|_\mathcal{B} \le (C'_p)^d \|f\|_\mathcal{B}$ for all $f \in \mathcal{B}$ and all $n \in \mathbb{N}$.*

Proof. First, we prove this lemma for $d = 1$ (torus $\mathbb{T} \cong [0, L]$) and then extend it to higher dimensions. For $d = 1$, we have:

$$\begin{aligned} S_n f(x) &= \sum_{|k| \le n} \hat{f}(k) e^{2\pi i k x/L} \\ &= f * D_n \\ &:= f * \left(\sum_{|k| \le n} e^{2\pi i k x/L} \right). \end{aligned} \tag{5.3.10}$$

Define $\tilde{f}(x) = -i \sum_{k \in \mathbb{Z}} \operatorname{sgn}(k) \hat{f}(k) e^{2\pi i k x/L}$, $\|\tilde{f}\|_\mathcal{B} \le C_p \|f\|_\mathcal{B}$ indicates $\sup_n \|S_n\|_\mathcal{B} < \infty$ holds. Consider the Riesz projection $P_+(f) = \sum_{k \ge 0} \hat{f}(k) e^{2\pi i k x/L}$. Then, $P_+(f) = \frac{1}{2}(f + i\tilde{f}) + \frac{1}{2}\hat{f}(0)$, so boundedness of P_+ on L^p is equivalent to the boundedness of $\tilde{\cdot}$. Since

$$\begin{aligned} S_n f &= \hat{f}(0) + \sum_{k=1}^{n} \left(\hat{f}(k) e^{2\pi i k x/L} + \hat{f}(-k) e^{-2\pi i k x/L} \right) \\ &= 2 \operatorname{Re}(P_+^{(n)}) - \hat{f}(0), \end{aligned} \tag{5.3.11}$$

where $P_+^{(n)}(f) = \sum_{k=0}^{n} \hat{f}(k) e^{ikx}$ and

$$\|P_+^{(n)}(f) - f\|_\mathcal{B} \le (M + 1)\varepsilon \tag{5.3.12}$$

for a constant $M > 0$ and any $\varepsilon > 0$ from Miao (2014). From the generalized M. Riesz Theorem Berkson & Gillespie (1985), it follows $\|\tilde{f}\|_\mathcal{B} \le C_p \|f\|_\mathcal{B}$, which by the identity

$P_+(f) = \frac{1}{2}(f + i\tilde{f}) + \frac{1}{2}\hat{f}(0)$, implies that the Riesz projection P_+ is also bounded on $\mathcal{B}$, with

$$\|P_+ f\|_{\mathcal{B}} \leq C_p' \|f\|_{\mathcal{B}}. \tag{5.3.13}$$

Since $P_+^{(n)} f \to P_+ f$ in $\mathcal{B}$ norm as $n \to \infty$, it follows that $\{P_+^{(n)}\}_{n\in\mathbb{N}}$ is uniformly bounded on $\mathcal{B}$, i.e.,

$$\sup_n \|P_+^{(n)}\|_{\mathcal{B}} < \infty. \tag{5.3.14}$$

Therefore, one can obtain

$$\begin{aligned} \|S_n f\|_{\mathcal{B}} &\leq 2\|P_+^{(n)} f\|_{\mathcal{B}} + |\hat{f}(0)| \cdot \|1\|_{\mathcal{B}} \\ &\leq 2C_p \|f\|_{\mathcal{B}} + C\|f\|_{\mathcal{B}} = C_p' \|f\|_{\mathcal{B}} \end{aligned} \tag{5.3.15}$$

for a constant $C > 0$ using the triangle inequality.

For d-dimensional case, we define the d-dimensional kernel $D_n^d(\mathbf{x}) = \prod_{j=1}^d D_n(x_j)$ and the d-dimensional operator $S_n^d = S_n \otimes \cdots \otimes S_n$. From Equation (5.3.15), we have:

$$\|S_n^d f\|_{\mathcal{B}} \leq (C_p')^d \|f\|_{\mathcal{B}}. \tag{5.3.16}$$

■

Lemma 5.3.2 *Let $\mathcal{P}$ denote the space of trigonometric polynomials, i.e., finite linear combinations of plane waves $\psi_\xi(x) = e^{2\pi i \xi \cdot x/L}$ for $\xi \in \mathbb{Z}^d$. Then, $\mathcal{P}$ is dense in $\mathcal{B}$, meaning for any $f \in \mathcal{B}$ and $\epsilon > 0$, there exists $p \in \mathcal{P}$ such that $\|f - p\|_{\mathcal{B}} < \epsilon$. For any $g \in \mathcal{P}$ of degree at most n, it holds that $S_n g = g$, where $S_n g = \sum_{|\xi|\leq n} \hat{g}(\xi)\psi_\xi$.*

Proof. Since $\mathbb{T}^d$ is compact with finite measure, continuous functions $C(\mathbb{T}^d)$ are dense in $\mathcal{B}(\mathbb{T}^d)$ for any $1 \leq p < \infty$. For $f \in \mathcal{B}$, by Lusin's theorem (every measurable function is nearly continuous), for any $\epsilon > 0$, there exists a compact $K \subset \mathbb{T}^d$ with $\mu(\mathbb{T}^d \setminus K) < \epsilon$ and $f|_K$ continuous, hence can be approximated by continuous g with $\|f - g\|_{\mathcal{B}} < \epsilon$.

Next, we prove that $\mathcal{P}$ is dense in $C(\mathbb{T}^d)$. The Stone-Weierstrass theorem states that if $\mathcal{A}$ is a subalgebra of $C(X)$ that separates points (for any distinct $x, y \in X$, there exists $f \in \mathcal{A}$ with $f(x) \neq f(y)$) and contains constants, then $\mathcal{A}$ is dense in $C(X)$ under the sup-norm.

Following this, considering the algebra $\mathcal{A} = \mathcal{P}_{\mathbb{R}}$ of real parts of $\mathcal{P}$, $\mathcal{A}$ contains constants and separates points: for distinct $x, y \in \mathbb{T}^d$, choose $\xi \in \mathbb{Z}^d$ such that $\xi \cdot (x - y) \not\equiv 0 \pmod 1$, then $\cos(2\pi\xi \cdot x/L) \neq \cos(2\pi\xi \cdot y/L)$. Moreover, $\mathcal{A}$ is closed under multiplication. Thus, $\mathcal{A}$ is dense in $C(\mathbb{T}^d; \mathbb{R})$. Next, for $C(\mathbb{T}^d; \mathbb{C})$, density follows by approximating real and imaginary parts separately, since $\mathcal{P}$ includes both cosines and sines. Therefore, given $\mathcal{P}$ dense in $C(\mathbb{T}^d)$ and $C(\mathbb{T}^d)$ dense in $\mathcal{B}$, implies $\mathcal{P}$ dense in $\mathcal{B}$ from transitivity of density.

Then, for $g \in \mathcal{P}$ with degree at most n, we have $\hat{g}(\xi) = 0$ for all $|\xi| > n$. Therefore, it follows:

$$S_n g = \sum_{|\xi| \leq n} \hat{g}(\xi)\psi_\xi = \sum_{\xi \in \mathbb{Z}^d} \hat{g}(\xi)\psi_\xi = g, \tag{5.3.17}$$

the last equality follows

$$g = \sum_{\xi \in \mathbb{Z}^d} \langle g, \psi_\xi \rangle \psi_\xi, \tag{5.3.18}$$

where $\langle g, \psi_\xi \rangle = \hat{g}(\xi)$. ■

Lemma 5.3.3 *With the Assumption 1, by assuming that $\mathcal{B}$ admits a Fourier transform $\mathcal{F}$ that is well-defined and invertible on a periodic domain $[0, L]^d$), for the Fourier basis of plane waves $\psi_\xi(x) = e^{2\pi i \xi \cdot x/L}$ denoted as $(\psi_\xi)_{\xi \in \mathbb{Z}^d}$, and the biorthogonal dual functionals $(\varphi_\xi)_{\xi \in \mathbb{Z}^d} \subset \mathcal{B}^*$ satisfying $\hat{f}(\xi) = \mathcal{F}(f)(\xi) = \langle f, \varphi_\xi \rangle_{\mathcal{B}, \mathcal{B}^*}$ for any $f \in \mathcal{B}$, then the inverse Fourier transform satisfies $\mathcal{F}^{-1}(\hat{f}(\xi)) = \sum_{\xi \in \mathbb{Z}^d} \hat{f}(\xi)\psi_\xi(x)$ in the sense that the series converges to f in the $\mathcal{B}$-norm, i.e., $\lim_{n \to \infty} \left\| f - \sum_{|\xi| \leq n} \hat{f}(\xi)\psi_\xi \right\|_{\mathcal{B}} = 0$.*

Proof. Following Lin et al. (2022), one can show that the plane waves $\psi_\xi(x) = e^{2\pi i \xi \cdot x/L}$ span a dense subspace of trigonometric polynomials in $\mathcal{B}$. The corresponding dual functionals $\varphi_\xi \in \mathcal{B}^*$ extract the Fourier coefficients via the pairing $\hat{f}(\xi) = \langle f, \varphi_\xi \rangle_{\mathcal{B}, \mathcal{B}^*}$. Biorthogonality, given by $\langle \psi_\xi, \varphi_\eta \rangle_{\mathcal{B}, \mathcal{B}^*} = \delta_{\xi\eta}$, is ensured by the RKBS structure and follows from the spectral theorem applied to the associated kernel integral operator. Then, we define the operator $S_n : \mathcal{B} \to \mathcal{B}$ via $S_n f = \sum_{|\xi| \leq n} \hat{f}(\xi)\psi_\xi$. From Lemma 5.3.1, $\|S_n\|_{\mathcal{B} \to \mathcal{B}}$ is bounded by a constant $(C_p')^d$ independent of n.

From Lemma 5.3.2 and uniform boundedness principle, $S_n f \to f$ in $\mathcal{B}$-norm for all $f \in \mathcal{B}$. ■

Remark 5.3.1 *An important consequence is the Hausdorff–Young inequality:*

$$\left(\sum_{\xi} |\hat{f}(\xi)|^q \right)^{1/q} \leq C \|f\|_{\mathcal{B}}, \quad \text{where } \frac{1}{p} + \frac{1}{q} = 1, \tag{5.3.19}$$

which provides a bound on the Fourier coefficients in ℓ^q, which facilitates convergence of the Fourier series. For finite modes $\Lambda \subset \mathbb{Z}^d$, the sum is finite and exact, and the infinite case follows by taking limits as $|\Lambda| \to \infty$. Thus, the reconstruction defines the inverse Fourier transform $\mathcal{F}^{-1}(\hat{f}(\xi)) = \sum_{\xi} \hat{f}(\xi)\psi_{\xi}(x)$ in $\mathcal{B}$.

Lemma 5.3.4 *It holds that*

$$\mathcal{S}(f)(x) = \sum_{\xi \in \Lambda} w_{\xi} \langle f, \varphi_{\xi} \rangle_{\mathcal{B},\mathcal{B}^*} \psi_{\xi}(x),$$

where $w_{\xi} = W(\xi)$, $\varphi_{\xi} \in \mathcal{B}^*$ are dual bases, and $\psi_{\xi} \in \mathcal{B}$ are biorthogonal primal bases with $\langle \psi_{\xi}, \varphi_{\eta} \rangle_{\mathcal{B},\mathcal{B}^*} = \delta_{\xi\eta}$.

Proof. In spectral methods for RKBS, one can show that the Fourier transform diagonalizes convolution operators Kovachki et al. (2021). Specifically, we assume a biorthogonal Fourier basis $(\psi_{\xi}, \varphi_{\xi})_{\xi \in \mathbb{Z}^d}$, where $\psi_{\xi}(x) = e^{2\pi i \xi \cdot x/L}$ for domain $[0, L]^d$, and φ_{ξ} are dual functionals satisfying the biorthogonality from the reproducing property: $\langle f, \varphi_{\xi} \rangle = \hat{f}(\xi) = \mathcal{F}(f)(\xi)$. Then, the spectral convolution applies pointwise multiplication in frequency space:

$$\widehat{\mathcal{S}(f)}(\xi) = \chi(\xi) W(\xi) \hat{f}(\xi). \tag{5.3.20}$$

Taking the inverse Fourier transform on Equation (5.3.20) leads to:

$$\mathcal{S}(f)(x) = \sum_{\xi \in \Lambda} W(\xi) \hat{f}(\xi) \psi_{\xi}(x), \tag{5.3.21}$$

where $\chi(\xi) = 1$ for $\xi \in \Lambda$ and 0 otherwise. Substituting $\hat{f}(\xi) = \langle f, \varphi_{\xi} \rangle_{\mathcal{B},\mathcal{B}^*}$ into Equation (5.3.21) leads to

$$\mathcal{S}(f)(x) = \sum_{\xi \in \Lambda} W(\xi) \langle f, \varphi_{\xi} \rangle_{\mathcal{B},\mathcal{B}^*} \psi_{\xi}(x). \tag{5.3.22}$$

Biorthogonality $\langle \psi_\xi, \varphi_\eta \rangle = \delta_{\xi\eta}$ follows from the Fourier basis orthogonality in the dual pairing, ensured by the RKBS structure Li et al. (2022a). ■

Remark 5.3.2 *Equation* (5.3.22) *is nonlocal because the kernel involves global frequency modes rather than localized supports.*

Lemma 5.3.5 *For any continuous operator $T : \mathcal{B} \to \mathcal{B}$ and $\delta > 0$, there exists a set of parameters of $W(\xi)$ in Equation* (5.3.6) *such that $\|\mathcal{S}(g) - T(g)\|_{\mathcal{B}} < \delta$ for all g in a bounded subset of $\mathcal{B}$.*

Proof. First, we define integral operators $T(f)(x) = \int K_T(x, y) f(y) dy$, with kernel K_T continuous. Then, fixing a bounded set $G \subset \mathcal{B}$, we say G is weakly compact since $\mathcal{B}$ is uniformly smooth and reflexive. By Arzelà-Ascoli theorem, continuous kernels $K_T(x, y)$ and thus operator $T(f)$ can be approximated uniformly on G. We denote the Fourier operator $T_M(f) = \mathcal{F}^{-1}[\sum_{|\xi| \leq M} \hat{K}_T(\xi) \mathcal{F}(f)(\xi)]$, where $\hat{K}_T(\xi)$ is the Fourier transform of the kernel $K_T(x, y)$. From universal approximation theorem Kovachki et al. (2021), we have:

$$\|T - T_M\|_{\mathcal{B}} \to 0, \text{ as } M \to \infty \tag{5.3.23}$$

Next, by setting $\Lambda = \{\xi : |\xi| \leq M\}$, $\chi(\xi) = 1_\Lambda(\xi)$, and $W(\xi) = \hat{K}_T(\xi)$, it follows

$$\|\mathcal{S} - T_M\|_{\mathcal{B}} \to 0, \text{ as } M \to \infty. \tag{5.3.24}$$

Finally, substituting Equation (5.3.23) to Equation (5.3.24) leads to:

$$\begin{aligned} \|\mathcal{S}(g) - T(g)\|_{\mathcal{B}} &\leq \|\mathcal{S}(g) - T_M(g) + T_M(g) - T(g)\|_{\mathcal{B}} \\ &\leq \|\mathcal{S}(g) - T_M(g)\|_{\mathcal{B}} + \|T(g) - T_M(g)\|_{\mathcal{B}} \\ &< \frac{\delta}{2} + \frac{\delta}{2} = \delta, \end{aligned} \tag{5.3.25}$$

which completes the proof. ■

Lemma 5.3.6 *Define $\mathcal{C}(f)(x) = \int_\Omega \delta(x - y) \cdot W \cdot f(y)\, dy$, where δ is the Dirac delta distribution. The integral is well-defined in the distributional sense.*

Proof. First, we consider the single channel ($C = 1, W = 1$). In this case, we have $\mathcal{C}(f)(x) = f(x) = \int_\Omega \delta(x-y)f(y)\,dy$, where the integral is the pairing $\langle \delta_x, f\rangle_{\mathcal{B},\mathcal{B}^*}$ in the dual space since $\delta_x \in \mathcal{B}^*$ for spaces admitting point evaluations. Next, for multi-channel $f = (f^{(1)}, \ldots, f^{(C)})$, we extend $\mathcal{C}(f)^{(c)}(x)$ to $\sum_{c'=1}^{C} W_{cc'} f^{(c')}(x)$. Then, it follows

$$\begin{aligned} \mathcal{C}(f)^{(c)}(x) &= \sum_{c'=1}^{C} W_{cc'} f^{(c')}(x) \\ &= \sum_{c'=1}^{C} W_{cc'} \int_\Omega \delta(x-y) f^{(c')}(y)\,dy \\ &= \int_\Omega \delta(x-y)(Wf(y))^{(c)}\,dy, \end{aligned} \tag{5.3.26}$$

where $Wf(y)$ applies the matrix pointwise. Equation (5.3.26) becomes to

$$\mathcal{C}(f)(x) = \int_\Omega \delta(x-y)\cdot W \cdot f(y)\,dy, \tag{5.3.27}$$

for $f(y) \in \mathbb{R}^C$. ■

Remark 5.3.3 *$\mathcal{C}(f)(x)$ is a local projection because the kernel $\delta(x-y)W$ has support only at $y = x$ which projects onto the span of channels without spatial smearing. In practice, for a discrete domain, it reduces to matrix multiplication at pixels. For a continuous domain, it is actually the distributional convolution with a point mass.*

Lemma 5.3.7 *Finite-rank operators are dense in the space of compact operators.*

Proof. It is equivalent to show for any $\delta > 0$, there exists a finite-rank operator K_m such that $\|T - K_m\|_{\mathcal{B}} < \delta$. Assume the operator T is compact, then its image on a unit ball B has compact closure. Therefore, for any $x \in \mathcal{B}_B$, there exists $v_i \in \mathcal{B}$, such that $\|T(x) - v_i\|_{\mathcal{B}} < \delta$ for any $\delta > 0$. Now, we define a projection $\pi_V : \text{range}(T) \to V$, which maps each $T(x)$ to its best approximation in V. It is well-defined because V is finite-dimensional and $\mathcal{B}$ is a Banach space. Then, one can also define $K_m := \pi_V \circ T$, which is a linear and bounded operator. And the range of K_m lies in the span of v_i, implying that K_m is finite-rank. This way, one can verify $\|T(f) - K_m(f)\|_{\mathcal{B}} = \|T(f) - \pi_V(T(f))\|_{\mathcal{B}} \leq \delta$ for any $\delta > 0$ and $f \in \mathcal{B}$.

Since $K_m(f) \in \text{span}\{v_1, \ldots, v_m\}$, one can formulate it as $K_m(f) = \sum_{i=1}^{m} a_i(f)\, v_i$, for linear functionals a_i. ■

Lemma 5.3.8 *$\mathcal{C}$ is finite-rank. For any operator $T : \mathcal{B} \to \mathcal{B}$ following $T = I - P$, where P is a projection, and $\delta > 0$, there exists $W_{cc'}$ such that $\|\mathcal{C}(g) - T(g)\|_{\mathcal{B}} < \delta$ for all g in a bounded subset $G \subset \mathcal{B}$, with the error decaying as the matrix rank (channel dimension) C increases.*

Proof. First, we show that $\mathcal{C}$ is finite-rank. Since $\mathcal{B}$ is an RKBS, the evaluation functional is continuous, for $f \in \mathcal{B}$, $f(x) = \langle f, K(\cdot, x)\rangle_{\mathcal{B},\mathcal{B}^*}$ holds, and $K(\cdot, x) \in \mathcal{B}^*$ satisfies $\|K(\cdot, x)\|_{\mathcal{B}^*} < \infty$. Let $\mathcal{B}^C = \mathcal{B} \otimes \mathbb{R}^C$, $f = (f^{(1)}, \ldots, f^{(C)})$ has evaluations $f(x) = (f^{(1)}(x), \ldots, f^{(C)}(x)) \in \mathbb{R}^C$, with

$$f^{(c)}(x) = \langle f^{(c)}, K(\cdot, x)\rangle_{\mathcal{B},\mathcal{B}^*}. \tag{5.3.28}$$

For each channel $c = 1, \ldots, C$, we have

$$\mathcal{C}(f)^{(c)}(x) = \sum_{c'=1}^{C} W_{cc'} f^{(c')}(x). \tag{5.3.29}$$

The reproducing property follows

$$f^{(c')}(x) = \langle f^{(c')}, K(\cdot, x)\rangle_{\mathcal{B},\mathcal{B}^*} = \langle f^{(c')}, \delta_x\rangle_{\mathcal{B},\mathcal{B}^*} \tag{5.3.30}$$

in the distributional sense as long as $\mathcal{B}$ embeds into a space where δ_x is defined. Then, it holds that

$$\langle f^{(c')}, \delta_x\rangle_{\mathcal{B},\mathcal{B}^*} = \int_{\Omega} \delta(x - y) f^{(c')}(y)\, dy \tag{5.3.31}$$

in the weak sense. Therefore, we have:

$$\begin{aligned} \mathcal{C}(f)^{(c)}(x) &= \sum_{c'=1}^{C} W_{cc'} \langle f^{(c')}, \delta_x\rangle_{\mathcal{B},\mathcal{B}^*} \\ &= \sum_{c'=1}^{C} W_{cc'} \int_{\Omega} \delta(x - y) f^{(c')}(y)\, dy. \end{aligned} \tag{5.3.32}$$

Next, Equation (5.3.32) becomes:

$$\mathcal{C}(f) = \sum_{c=1}^{C} \sum_{c'=1}^{C} W_{cc'} \langle f, \mathscr{B}_{c'}\rangle_{\mathcal{B}^C} \mathscr{B}_c, \tag{5.3.33}$$

where $\langle f, \mathscr{B}_{c'} \rangle_{\mathcal{B}^C} = \langle f^{(c')}, \mathscr{B}_c \rangle_{\mathcal{B},\mathcal{B}^*}$ for the basis $\mathscr{B}_c$ of $\mathcal{B}$. The rank of $\mathcal{C}(f)$ is at most $c \times c' \leq C^2$, so $\mathcal{C}(f)$ is finite-rank. We note that if an operator is finite-rank, it is bounded and compact in Banach space $\mathcal{B}$.

Moreover, we consider the operator $T = I - P$, where P is a projection operator. Assume that P is finite-rank and thus compact, and then the operator T is also compact. Since T is compact, for any $\delta > 0$, there exists finite-rank approximation $K_m = \sum_{i=1}^{m} \sigma_i \langle \cdot, u_i \rangle_{\mathcal{B}^C} v_i$, where $u_i, v_i \in \mathcal{B}^C$, such that

$$\|T(g) - K_m(g)\|_{\mathcal{B}} < \frac{\delta}{\sup_{g \in G} \|g\|_{\mathcal{B}^C}} \tag{5.3.34}$$

for $g \in G$ from Lemma 5.3.7.

Then, by denoting $A = I - P$, from Equation (5.3.34), we have:

$$\begin{aligned} \|\mathcal{C}(g) - T(g)\|_{\mathcal{B}^C} &= \left\| \int_\Omega \delta(x - y)(W - A(y))g(y)\, dy \right\|_{\mathcal{B}^C} \\ &= \left(\int_\Omega \|(W - A(y))g(y)\|_{\mathbb{R}^C}^2\, dy \right)^{1/2} \\ &\leq \sup_y \|W - A(y)\|_{\mathcal{B}^C} \left(\int_\Omega \|g(y)\|_{\mathbb{R}^C}^2\, dy \right)^{1/2} \\ &\leq \epsilon \|g\|_{\mathcal{B}^C} < \delta, \end{aligned} \tag{5.3.35}$$

by choosing $\epsilon < \delta / (2 \sup \|g\|_{\mathcal{B}^C})$. ■

Now, we give the full proof of Theorem 5.3.1 assuming GELU is globally Lipschitz with constant L_{GELU}.

Proof. We start with the residual:

$$\begin{aligned} r_i &= \alpha_{\text{true}} - \alpha_i = (\alpha_{\text{true}} - \alpha_{i-1}) - (\alpha_i - \alpha_{i-1}) \\ &= r_{i-1} - \langle r_{i-1}, J(\mathscr{B}_i) \rangle_{\mathcal{B},\mathcal{B}^*} \mathscr{B}_i. \end{aligned} \tag{5.3.36}$$

Then, let $T(g) = \langle g, J(\mathscr{B}_i) \rangle_{\mathcal{B},\mathcal{B}^*} \mathscr{B}_i$, from Lemma 5.3.5, one can obtain:

$$\|\mathcal{S}(z_{p,i-1}) - \langle r_{i-1}, J(\mathscr{B}_i) \rangle_{\mathcal{B},\mathcal{B}^*} \mathscr{B}_i\|_{\mathcal{B}} < \frac{\varepsilon}{4 L_{\text{GELU}}}. \tag{5.3.37}$$

Next, from Lemma 5.3.8, we can also get:

$$\begin{aligned}\|\mathcal{C}(z_{p,i-1}) - (r_{i-1} - z_{p,i-1}) - \langle r_{i-1}, J(\mathscr{B}_i)\rangle_{\mathcal{B},\mathcal{B}^*}\mathscr{B}_i)\|_{\mathcal{B}} \\ < \frac{\varepsilon}{4L_{\text{GELU}}}.\end{aligned} \tag{5.3.38}$$

By adjusting the parameters in GELU (i.e., γ, β), one can show:

$$\begin{aligned}\|\text{BN}(z_{p,i-1} + \mathcal{S}(z_{p,i-1}) + \mathcal{C}(z_{p,i-1})) - (r_{i-1} \\ - \langle r_{i-1}, J(\mathscr{B}_i)\rangle_{\mathcal{B},\mathcal{B}^*}\mathscr{B}_i)\|_{\mathcal{B}} < \frac{\varepsilon}{2L_{\text{GELU}}}.\end{aligned} \tag{5.3.39}$$

By Lipschitz continuity of GELU, we have:

$$\|z_{p,i} - r_i\|_{\mathcal{B}} < L_{\text{GELU}} \cdot \left(\frac{\varepsilon}{4L_{\text{GELU}}} + \frac{\varepsilon}{4L_{\text{GELU}}} + \frac{\varepsilon}{2L_{\text{GELU}}}\right) < \varepsilon. \tag{5.3.40}$$

Since J is continuous, one can show $\|z_{d,i} - J(r_i)\|_{\mathcal{B}}$ at the same way. ■

The AFD-type decoder network. Once the RKBS and its reproducing kernel K_z are constructed, we design a decoder network based on the AFD operation to reconstruct parameters α from $z_{p,i}$. First, we normalize the reproducing kernel K_z in Equation (5.3.5) as $\mathscr{B}_i(\cdot) = \frac{K_z(\cdot,a_i)}{||K_z(\cdot,a_i)||_{\mathcal{B}}}$, each associated with a pole a_i. The set of these normalized reproducing kernels is denoted as $\mathcal{D} = \{\mathscr{B}_i | a_i \in \mathcal{B}\}$. The decoder then adopts a dynamic convolutional kernel network (CKN) Mairal et al. (2014); Chen et al. (2020b), in which, for each convolutional layer i, (i) performs dual pairing between $z_{p,i}$ and the normalized reproducing kernel $\mathscr{B}_i$, (ii) assigns a multiplier $0 < \rho_0 \leq \rho_i < 1$ to the output of each convolutional layer, and (iii) incorporates skip connections for each convolutional layer. Finally, the output of the dynamic CKN containing N convolutional layers is:

$$\hat{\alpha}_{N,\theta} = \sum_{i=1}^{N} \langle z_{p,i}, J(\mathscr{B}_i)\rangle_{\mathcal{B},\mathcal{B}^*}\mathscr{B}_i. \tag{5.3.41}$$

Guided by the AFD theory, the selection of poles and their reproducing kernels follows a similar maximal selection principle as in the AFD theory. Here, starting from Equation (5.2.1), we can write an analogous condition for selecting poles as:

$$\gamma_i \leq |\langle z_{p,i-1}, J(\mathscr{B}_i)\rangle_{\mathcal{B},\mathcal{B}^*}| - \rho_i \sup_{\mathscr{B}_i \in \mathcal{D}} \{|\langle z_{p,i-1}, J(\mathscr{B}_i)\rangle_{\mathcal{B},\mathcal{B}^*}|\}. \tag{5.3.42}$$

With this, and leveraging the convergence behavior of AFD, we can show that our decoder in AFDONet-inv converges as $N \to \infty$ by the following theorem:

Theorem 5.3.2 *Let $\mathcal{B}$ be a uniformly smooth and uniformly convex Banach space, and let $\hat{\alpha}_{N,\theta}$ be the output of the dynamic CKN decoder with N layers. By selecting poles and bias terms following the modified maximal selection principle of Equation* (5.3.42)*, as $N \to \infty$:*

$$\|\hat{\alpha}_{N,\theta} - \alpha_{true}\|_{\mathcal{B}} \leq \hat{C} \prod_{i=1}^{N} \rho_i \cdot \|r_0\|_{\mathcal{B}},$$

where $\hat{C} > 0$ is a constant and r_0 is the initial residual.

Before providing the full proof of Theorem 5.3.2, we give several definitions and lemmas first.

Definition 5.3.1 *A Banach space $\mathcal{B}$ is uniformly convex if there exists a function $\delta : [0, 2] \to [0, 1]$, the modulus of convexity, such that for all $u, v \in \mathcal{B}$ with $\|u\|_{\mathcal{B}} = \|v\|_{\mathcal{B}} = 1$ and $\|u - v\|_{\mathcal{B}} \geq \tau$,*

$$\left\| \frac{u+v}{2} \right\|_{\mathcal{B}} \leq 1 - \delta(\tau), \quad \delta(\tau) > 0 \text{ for } \tau > 0.$$

$\mathcal{B}$ is uniformly smooth if the modulus of smoothness $\rho(\tau) = \sup\{\frac{\|u+\tau v\|_{\mathcal{B}} + \|u-\tau v\|_{\mathcal{B}}}{2} - 1 : \|u\|_{\mathcal{B}} = \|v\|_{\mathcal{B}} = 1\}$ satisfies $\rho(\tau) = o(\tau)$ as $\tau \to 0$.

Lemma 5.3.9 *For any $\eta > 0$, there exist parameters in the primal-dual propagation such that $\|z_{p,i} - r_i\|_{\mathcal{B}} < \eta$ and $\|z_{d,i} - J(r_i)\|_{\mathcal{B}^*} < \eta$ for each i. Therefore, it holds that*

$$|\langle z_{p,i}, J(\mathscr{B}_{i+1})\rangle_{\mathcal{B},\mathcal{B}^*} - \langle r_i, J(\mathscr{B}_{i+1})\rangle_{\mathcal{B},\mathcal{B}^*}| < \eta.$$

Proof. This lemma follows directly from Theorem 5.3.1. ■

Lemma 5.3.10 *In the AFD operations, the residuals satisfy*

$$\|r_i\|_{\mathcal{B}}^2 \leq \rho_i^2 \|r_{i-1}\|_{\mathcal{B}}^2,$$

where $\rho_i^2 = 1 - \delta\left(2\frac{|\langle r_{i-1}, J(\mathscr{B}_i)\rangle_{\mathcal{B},\mathcal{B}^*}|}{\|r_{i-1}\|_{\mathcal{B}}}\right) < 1$, and $\mathscr{B}_i = \max_{\mathscr{B}_j \in \mathcal{D}} |\langle r_{i-1}, J(\mathscr{B}_j)\rangle_{\mathcal{B},\mathcal{B}^*}|$.

Proof. For $u = \frac{r_{i-1}}{\|r_{i-1}\|_{\mathcal{B}}}$ and $v = \frac{\langle r_{i-1}, J(\mathscr{B}_i)\rangle_{\mathcal{B},\mathcal{B}^*}\mathscr{B}_i}{\|r_{i-1}\|_{\mathcal{B}}}$, it follows

$$\left\| \frac{r_i}{\|r_{i-1}\|_{\mathcal{B}}} \right\|_{\mathcal{B}} = \left\| u - \frac{\langle u, J(\mathscr{B}_i)\rangle_{\mathcal{B},\mathcal{B}^*}\mathscr{B}_i}{\|r_{i-1}\|_{\mathcal{B}}} \right\|_{\mathcal{B}} = \|u - v\|_{\mathcal{B}}. \tag{5.3.43}$$

Since $\|u - v\|_{\mathcal{B}}^2 \leq 1 - \delta\left(2\|u\|_{\mathcal{B}}\|v\|_{\mathcal{B}}\right)$, we have:

$$\begin{aligned} \|u - v\|_{\mathcal{B}}^2 &\leq 1 - \delta\left(2\|v\|_{\mathcal{B}}\right) \\ &\leq 1 - 2\delta\left(\frac{|\langle r_{i-1}, J(\mathscr{B}_{a_i})\rangle_{\mathcal{B},\mathcal{B}^*}|}{\|r_{i-1}\|_{\mathcal{B}}}\right) := 1 - \rho_i. \end{aligned} \tag{5.3.44}$$

Equation (5.3.44) becomes to:

$$\|r_i\|_{\mathcal{B}}^2 \leq \rho_i^2 \|r_{i-1}\|_{\mathcal{B}}^2. \tag{5.3.45}$$

Iterating Equation (5.3.45) leads to:

$$\|r_N\|_{\mathcal{B}} \leq \left(\prod_{i=1}^{N} \rho_i\right) \|r_0\|_{\mathcal{B}}. \tag{5.3.46}$$

■

Lemma 5.3.11 *The bias terms satisfy*

$$\sum_{i=1}^{N} |\gamma_i| \leq \left(1 - \prod_{i=1}^{N} \rho_i\right) \|r_0\|_{\mathcal{B}} + N\varepsilon,$$

where the bias term satisfying $\gamma_i \leq |\langle z_{p,i-1}, J(\mathscr{B}_i)\rangle| - \rho_i \sup_{\mathscr{B}_j \in \mathcal{D}} |\langle z_{p,i-1}, J(\mathscr{B}_j)\rangle|$, *and* $\|z_{p,i-1} - r_{i-1}\|_{\mathcal{B}} < \varepsilon$.

Proof. Assume $\mathcal{D}$ is rich in $\mathcal{B}$, we have:

$$\begin{aligned} |\gamma_i| &= |\langle z_{p,i-1}, J(\mathscr{B}_i)\rangle| - \rho_i \sup_{\mathscr{B}_j} |\langle z_{p,i-1}, J(\mathscr{B}_j)\rangle| \\ &\leq (1 - \rho_i)\|z_{p,i-1}\|_{\mathcal{B}}. \end{aligned} \tag{5.3.47}$$

Now, from Theorem 5.3.1, one can have:

$$\|z_{p,i-1}\|_{\mathcal{B}} \leq \|r_{i-1}\|_{\mathcal{B}} + \|z_{p,i-1} - r_{i-1}\|_{\mathcal{B}} \leq \|r_{i-1}\|_{\mathcal{B}} + \varepsilon \tag{5.3.48}$$

for $\varepsilon > 0$.

From Lemma 5.3.10, Equation (5.3.48) becomes to:

$$\|z_{p,i-1}\|_{\mathcal{B}} \leq \prod_{j=1}^{i-1} \rho_j \|r_0\|_{\mathcal{B}} + \varepsilon. \tag{5.3.49}$$

Substituting Equation (5.3.47) into Equation (5.3.49) leads to:

$$|\gamma_i| \leq (1-\rho_i) \left(\prod_{j=1}^{i-1} \rho_j \|r_0\|_{\mathcal{B}} + \varepsilon \right). \tag{5.3.50}$$

Then, we sum both sides of Equation (5.3.47) over $i = 1$ to N and get:

$$\begin{aligned} \sum_{i=1}^{N} |\gamma_i| &\leq \sum_{i=1}^{N} (1-\rho_i) \left(\prod_{j=1}^{i-1} \rho_j \|r_0\|_{\mathcal{B}} + \varepsilon \right) \\ &= \|r_0\|_{\mathcal{B}} \sum_{i=1}^{N} (1-\rho_i) \prod_{j=1}^{i-1} \rho_j + \varepsilon \sum_{i=1}^{N} (1-\rho_i) \\ &\leq \|r_0\|_{\mathcal{B}} \sum_{i=1}^{N} (1-\rho_i) \prod_{j=1}^{i-1} \rho_j + \varepsilon N. \end{aligned} \tag{5.3.51}$$

For the first term in Equation (5.3.51), it follows

$$\begin{aligned} \sum_{i=1}^{N} (1-\rho_i) \prod_{j=1}^{i-1} \rho_j &= \sum_{i=1}^{N} \left(\prod_{j=1}^{i-1} \rho_j - \prod_{j=1}^{i} \rho_j \right) \\ &= (1-\rho_1) + (\rho_1 - \rho_1\rho_2) + (\rho_1\rho_2 - \rho_1\rho_2\rho_3) \\ &\quad + \cdots + (\prod_{j=1}^{N-1} \rho_j - \prod_{j=1}^{N} \rho_j) \\ &= 1 - \prod_{j=1}^{N} \rho_j. \end{aligned} \tag{5.3.52}$$

Putting everything together, we arrive:

$$\sum_{i=1}^{N} |\gamma_i| \leq \left(1 - \prod_{i=1}^{N} \rho_i \right) \|r_0\|_{\mathcal{B}} + N\varepsilon, \tag{5.3.53}$$

which completes the proof. ■

Now, we are safe to give the full proof of Theorem 5.3.2.

Proof. To start with, we consider

$$\begin{aligned}\|\hat{\alpha}_{N,\theta} - \alpha_{\text{true}}\|_{\mathcal{B}} = \Big\| \sum_{i=1}^{N} \langle z_{p,i}, J(\mathscr{B}_i)\rangle_{\mathcal{B},\mathcal{B}^*} \mathscr{B}_i + \sum_{i=1}^{N} \gamma_i - \\ \sum_{i=1}^{\infty} \langle r_{i-1}, J(\mathscr{B}_i)\rangle_{\mathcal{B},\mathcal{B}^*} \mathscr{B}_i \Big\|_{\mathcal{B}} \\ \leq \sum_{i=1}^{N} |\langle z_{p,i} - r_{i-1}, J(\mathscr{B}_i)\rangle_{\mathcal{B},\mathcal{B}^*}| \cdot \|\mathscr{B}_i\|_{\mathcal{B}} \\ + \sum_{i=1}^{N} |\gamma_i| + \left\| \sum_{i=N+1}^{\infty} \langle r_{i-1}, J(\mathscr{B}_i)\rangle_{\mathcal{B},\mathcal{B}^*} \mathscr{B}_i \right\|_{\mathcal{B}}.\end{aligned} \tag{5.3.54}$$

From Lemma 5.3.9, one can have:

$$|\langle z_{p,i} - r_{i-1}, J(\mathscr{B}_i)\rangle| < \eta, \tag{5.3.55}$$

for any $\eta > 0$, which implies

$$\sum_{i=1}^{N} |\langle z_{p,i} - r_{i-1}, J(\mathscr{B}_i)\rangle_{\mathcal{B},\mathcal{B}^*}| < N\eta = \frac{\varepsilon}{2} \tag{5.3.56}$$

by setting $\eta = \frac{\varepsilon}{2N}$. Then, we have:

$$\alpha_{\text{true}} = \sum_{i=1}^{\infty} \langle r_{i-1}, J(\mathscr{B}_i)\rangle_{\mathcal{B},\mathcal{B}^*} \mathscr{B}_i, \tag{5.3.57}$$

so the tail is

$$\left\| \sum_{i=N+1}^{\infty} \langle r_{i-1}, J(\mathscr{B}_i)\rangle_{\mathcal{B},\mathcal{B}^*} \mathscr{B}_i \right\|_{\mathcal{B}} = \|r_N\|_{\mathcal{B}} \leq \prod_{i=1}^{N} \rho_i \|r_0\|_{\mathcal{B}} \tag{5.3.58}$$

from Lemma 5.3.10. Next, from Lemma 5.3.11, the bias term γ_i satisfying:

$$\begin{aligned}\sum_{i=1}^{N} |\gamma_i| &\leq \left(1 - \prod_{i=1}^{N} \rho_i\right) \|r_0\|_{\mathcal{B}} + N\eta \\ &= \left(1 - \prod_{i=1}^{N} \rho_i\right) \|r_0\|_{\mathcal{B}} + \frac{\varepsilon}{2}.\end{aligned} \tag{5.3.59}$$

By choosing $\gamma_i \leq (1 - \rho_i) \prod_{j=1}^{i-1} \rho_j \|r_0\|_{\mathcal{B}}$, we have:

$$\begin{aligned}\|\hat{\alpha}_{N,\theta} - \alpha_{\text{true}}\|_{\mathcal{B}} &\leq \frac{\varepsilon}{2} + \prod_{i=1}^{N} \rho_i \|r_0\|_{\mathcal{B}} + \frac{\varepsilon}{2} \\ &= \varepsilon + \prod_{i=1}^{N} \rho_i \|r_0\|_{\mathcal{B}},\end{aligned} \tag{5.3.60}$$

which completes the proof. ■

5.3.2 Training

Overall, our AFDONet-inv model is trained end-to-end by minimizing the following loss function:

$$\begin{aligned}\mathcal{L}(\theta) = & \underbrace{\left|\left|\hat{\alpha}_{N,\theta} - \alpha_{\text{true}}\right|\right|_{\mathfrak{p}'}^{\mathfrak{p}'}}_{\text{reconstruction loss in } L^{\mathfrak{p}'}} + \underbrace{\omega_p \mathcal{D}_{\text{KL}}\left(\mathcal{N}(\mu_p, \sigma_p^2) \| \, \mathcal{N}(0, I)\right)}_{\text{latent regularization loss in primal space}} \\ & + \underbrace{\omega_d \mathcal{D}_{\text{KL}}\left(\mathcal{N}(\mu_d, \sigma_d^2) \| \, \mathcal{N}(0, I)\right)}_{\text{latent regularization loss in dual space}}.\end{aligned} \tag{5.3.61}$$

The training dataset consists of various sets of PDE parameters and their corresponding PDE solutions. The Adam optimizer with a learning rate of 5×10^{-4} is used to train our AFDONet-inv model. In actual implementation of the model, we use $\mathfrak{p}' = 1$ (corresponding to L^1 loss) and $\omega_p = \omega_d = 1 \times 10^{-3}$ in the loss function of Equation (5.3.61).

5.3.3 Connections to the AFD theory

In AFDONet-inv, the encoder network first maps the input to its latent space, followed by a latent-to-RKBS network which finds the corresponding nearest RKBS using a feature map FM. During training, when minimizing the loss function of Equation (5.3.61) over different sets of PDE solutions $\{x_j\}_{j=1}^m$, we find that the optimal feature map FM* admits a finite representation $\text{FM}^*(\tilde{\alpha})(x) = \sum_{j=1}^m c_j K_z(x, x_j)$ with coefficients $c_j \in \mathbb{R}$ (see Section 5.3.4 for details). After that, the primal-dual propagation refines the value of latent variable in the Banach space and its dual, and produces the input to each layer of the dynamic CKN. This input is essentially the residual r_i of AFD operation at each step i, as illustrated in Theorem 5.3.1. In this regard, the AFD-type decoder basically replicates the AFD operations. Formally, let $P_N(\alpha) = \sum_{i=1}^N \langle r_{i-1}, J(\mathscr{B}_i)\rangle_{\mathcal{B},\mathcal{B}^*} \mathscr{B}_i$ be the N-term partial sum in the AFD decomposition. Then, the decoder approximation satisfies:

Theorem 5.3.3 *Let $\mathcal{B}$ be a uniformly smooth and uniformly convex Banach space, and let $\hat{\alpha}_{N,\theta}$ be the output of the AFD-type decoder after N layers. Then, for any $\varepsilon > 0$, there exists*

parameters in the primal net and dual net, such that:

$$\|\hat{\alpha}_{N,\theta} - P_N(\alpha)\|_{\mathcal{B}} \leq \varepsilon + O\left(\frac{1}{\sqrt{N}}\right)\|\alpha\|_{\mathcal{B}}.$$

Before proving Theorem 5.3.3, we give a lemma first.

Lemma 5.3.12 *Let ℓ be a convex and L-Lipschitz loss function on a compact convex subset $\mathcal{B}_0 \subseteq \mathcal{B}$, and let Φ be a μ-strongly convex mirror map on $\mathcal{B}_0$. Then, the mirror descent algorithm (MDA) with projected updates achieves a convergence rate of $O(1/\sqrt{N})$ for N iterations:*

$$\ell(f_N) - \ell(f^*) \leq O\left(\frac{1}{\sqrt{N}}\right),$$

where f^* is the minimizer, and the constant depends on L, μ, and the diameter of $\mathcal{B}_0$.

Proof. It follows from Kumar et al. (2024) by setting $g_N = g_{N-1} - \eta\partial_{f_{N-1}}\mathcal{L}$ and $f_N = \Pi^{\Phi}_{\mathcal{B}_0}((\partial\Phi)^{-1}(g_N))$, where Π is the Bregman projection. ■

Remark 5.3.4 *We remark that, AFD can be interpreted as a greedy variant of mirror descent in RKBS, where each layer corresponds to a descent step with duality pairing approximating subgradients, and the normalized kernels $\mathscr{B}_i$ are utilized to select the directions. In AFD, the greedy selection maximizes the projection $|\langle r_{i-1}, J(\mathscr{B}_a)\rangle|$, which is equivalent to a subgradient descent step in the dual space, with the mirror map $\Phi(f) = \frac{1}{2}\|f\|^2_{\mathcal{B}}$. Furthermore, $r_i = r_{i-1} - \langle r_{i-1}, J\rangle_{\mathcal{B},\mathcal{B}^*}\mathscr{B}_i$ is a projected mirror descent step.*

With this, now we prove Theorem 5.3.3.

Proof. Define the loss function $\ell(f) = \|\alpha - f\|_{\mathcal{B}}$ on $\mathcal{B}_0 = \{f \in \text{span}(\mathcal{D}) : \|f\|_{\mathcal{B}} \leq \|\alpha\|_{\mathcal{B}}\}$. Since $\mathcal{B}$ is uniformly convex, $\mathcal{B}_0$ is compact. The function $\ell(f)$ is convex and Lipschitz:

$$|\ell(f) - \ell(g)| = |\|\alpha - f\|_{\mathcal{B}} - \|\alpha - g\|_{\mathcal{B}}| \leq \|f - g\|_{\mathcal{B}}. \tag{5.3.62}$$

One can know that the minimizer is $f^* = \alpha$ and $\ell(f^*) = 0$ holds in this case. In AFD, we have

$$r_i = r_{i-1} - \langle r_{i-1}, J(\mathscr{B}_i)\rangle_{\mathcal{B},\mathcal{B}^*}\mathscr{B}_i \tag{5.3.63}$$

and

$$P_i(\alpha) = P_{i-1}(\alpha) + \langle r_{i-1}, J(\mathscr{B}_i)\rangle_{\mathcal{B},\mathcal{B}^*}\mathscr{B}_i, \tag{5.3.64}$$

where $\mathscr{B}_i$ maximizes $|\langle r_{i-1}, J(\mathscr{B}_j)\rangle_{\mathcal{B},\mathcal{B}^*}|$. This way, AFD corresponds to a greedy mirror descent step, where the subgradient direction is approximated by $J(\mathscr{B}_i)$. Denote $f_i = P_i(\alpha)$, so $r_i = \alpha - f_i$. Furthermore, Equation (5.3.64) is equivalent to:

$$f_i = f_{i-1} + \eta_i J^*(\partial\ell(f_{i-1})), \tag{5.3.65}$$

where $\eta_i = \langle r_{i-1}, J(\mathscr{B}_i)\rangle$, and $J^* : \mathcal{B}^* \to \mathcal{B}$ is the inverse duality map since $\mathcal{B}$ is reflexive.

From Lemma 5.3.12, considering $\ell(f) = \|\alpha - f\|_{\mathcal{B}}$ on $\mathcal{B}_0$, we have

$$\ell(f_N) - \ell(f^*) = \|\alpha - f_N\|_{\mathcal{B}_0} \leq O\left(\frac{1}{\sqrt{N}}\right), \tag{5.3.66}$$

where the diameter $\operatorname{diam}(\mathcal{B}_0) \leq 2\|\alpha\|_{\mathcal{B}}$. Since $f_N = P_N(\alpha)$, and $\ell(f^*) = 0$,

$$\|P_N(\alpha) - \alpha\|_{\mathcal{B}_0} = \|\alpha - f_N\|_{\mathcal{B}_0} \leq O\left(\frac{1}{\sqrt{N}}\right). \tag{5.3.67}$$

Then, we generalize it to $\|\alpha\|_{\mathcal{B}}$ by scaling diameter:

$$\|P_N(\alpha) - \alpha\|_{\mathcal{B}} \leq O\left(\frac{1}{\sqrt{N}}\right)\|\alpha\|_{\mathcal{B}}. \tag{5.3.68}$$

Finally, from Theorem 5.3.2, we obtain:

$$\begin{aligned}\|\hat{\alpha}_{N,\theta} - P_N(\alpha)\|_{\mathcal{B}} &\leq \|\hat{\alpha}_{N,\theta} - \alpha\|_{\mathcal{B}} + \|\alpha - P_N(\alpha)\|_{\mathcal{B}} \\ &\leq \hat{C}\prod_{i=1}^{N}\rho_i \cdot \|r_0\|_{\mathcal{B}} + O\left(\frac{1}{\sqrt{N}}\right)\|\alpha\|_{\mathcal{B}} \\ &< \varepsilon + O\left(\frac{1}{\sqrt{N}}\right)\|\alpha\|_{\mathcal{B}},\end{aligned} \tag{5.3.69}$$

which completes the proof. ■

5.3.4 The Optimal Feature Map

We denote the training data points here as (x_i, y_i). The empirical risk minimization in the RKBS $\mathcal{B}$ is utilized in the form $\min_{f\in\mathcal{B}}\sum_{j=1}^{m}\ell(\mathrm{FM}(x_j), y_j) + \lambda\|\mathrm{FM}\|_{\mathcal{B}}$, where FM refers to

the possible feature maps, ℓ is the loss function which is assumed to be convex, continuous, and Lipschitz, and $\lambda \geq 0$ is the regularization parameter. Since $\mathcal{B}$ is reflexive, we know that the optimal feature map FM^* as a minimizer exists. Then, we calculate it by solving $\partial(\sum \ell(\mathrm{FM}(x_j), y_j)) + \lambda \partial \|\mathrm{FM}\|_{\mathcal{B}}) = 0$ following:

$$\begin{aligned} \partial(\sum \ell(\mathrm{FM}(x_j), y_j)) &= \sum_j \partial_u \ell(u, y_j)|_{u=\mathrm{FM}(x_j)} \cdot \partial \mathrm{FM}(x_j) \\ &= \sum_j \partial_u \ell(u, y_j)|_{u=\mathrm{FM}(x_j)} \cdot K_z^*(x_j, \cdot), \end{aligned} \tag{5.3.70}$$

where $\partial \mathrm{FM}(x_j) = K_z^*(x_j, \cdot) \in \mathcal{B}^*$ from evaluation functional. Thus, we have $\partial \ell(\mathrm{FM}) \subset \mathrm{span}\{K_z^*(x_j, \cdot)\}_{j=1}^m$ in dual.

On the other hand, we have $\partial \|\mathrm{FM}\|_{\mathcal{B}} = \frac{J(\mathrm{FM})}{\|\mathrm{FM}\|_{\mathcal{B}}}$ if $\mathrm{FM} \neq 0$. We admit FM^* follows the form $\sum_{j=1}^m c_j K_z(x, x_j)$, where c_j needs to be determined. Substituting it into the gradient condition leads to:

$$\sum_k c_k K_z(x_k, x_j) + \lambda \langle J(\mathrm{FM}^*), K_z(\cdot, x_j) \rangle_{\mathcal{B}, \mathcal{B}^*} = -\partial \ell(\mathrm{FM}^*(x_j), y_j), \tag{5.3.71}$$

which is a linear system can be solved accordingly.

5.4 Experiments

In this section, we evaluate the performance of our AFDONet-inv on two commonly used benchmark inverse problems by conducting extensive ablation studies and comparing the solution accuracy and run time with state-of-the-art neural solvers, including NAO Yu et al. (2024), NIPS Liu & Yu (2025), LNO Wang & Wang (2024), and MWT Gupta et al. (2021). Each solver is trained for 1000 epochs for both benchmark problems. All experiments are performed on a B760M GAMING WIFI PLUS desktop equipped with an Intel Core i5-14600KF CPU and an NVIDIA GeForce RTX 4090 GPU (with 48GB GDDR6 memory).

5.4.1 Problem settings and datasets

2-D Darcy flow. The first inverse problem we consider is the 2-D Darcy flow problem introduced by Li et al. (2020c) and Yu et al. (2024). It takes the following form:

$$\begin{aligned} \nabla \cdot \big(a(x)\nabla u(x)\big) &= f(x), \quad x \in [0,1]^2, \\ u(x) &= 0, \qquad\qquad\quad x \in \partial[0,1]^2, \end{aligned} \tag{5.4.1}$$

where $a(x)$ denotes the permeability field, and $f(x)$ is the source term. Given the solution $u(x)$ and the source term $f(x)$, here we aim to reconstruct the permeability field $a(x)$.

Nonlinear magnetic Schrödinger equation. The second benchmark inverse problem involves solving the magnetic Schrödinger equation on a complex manifold $\mathcal{M}$:

$$\begin{aligned} \big(\Delta_A + q(|u(z)|^2)\big)\, u(z) &= 0, \quad z \in \mathcal{M}, \\ u(z) &= f, \qquad\qquad\quad z \in \partial\mathcal{M}, \end{aligned} \tag{5.4.2}$$

where $\mathcal{M} = \{z = (z_1, z_2) \in \mathbb{C}^2 : |z_1|^2 + |z_2|^2 \leq 1\}$ with boundary $\partial\mathcal{M} = S^3$, $\Delta_A = (d + iA)^*(d + iA)$ is the magnetic Laplacian (with d the exterior derivative and $*$ the Hodge star with respect to the Kähler metric), q is a nonlinear function, and f is the boundary term. In this problem, we aim to recover the potentials A and q from boundary conditions in the Dirichlet-to-Neumann (DN) map $\Lambda_{A,q} : f \mapsto \partial_\nu u|_{\partial\mathcal{M}}$, where ν is the outward normal vector, and u is the solution of Equation (5.4.2). Note that this inverse problem is more challenging to solve than the one directly given solution u, as the DN map only retains partial information of u.

Two datasets used in this work include Darcy flow (public dataset from Li et al. (2020c)) and nonlinear magnetic Schrödinger. For Darcy flow dataset, the coefficients a are generated following a measure μ defined as $\mu = \psi\left(\mathcal{N}\left(0, (-\Delta + 9I)^{-2}\right)\right)$, where the operator $(-\Delta + 9I)^{-2}$ utilizes a Neumann boundary condition. The field a is constructed to be piecewise constant with random geometry and a fixed contrast of 4, determined by the mapping $\psi(x) = 12$ for $x > 0$ and $\psi(x) = 3$ for $x \leq 0$. Solutions u are generated using a second-order finite difference scheme on a high-resolution 241×241 grid.

For nonlinear magnetic Schrödinger dataset, we just need the Dirichlet-to-Neumann (DN) map without needing to generate PDE solutions. That is, the DN map data serves as the observation, rather than a solution field u. This data is generated by specifying the functional class of the potentials A (magnetic) and q (scalar) and the boundary term f on the complex manifold $\mathcal{M}$. The potentials A and q represent the unknown parameters, and the DN map $\Lambda_A, q : f \mapsto \partial_\nu u|_{\partial\mathcal{M}}$ is computed by numerically solving the highly nonlinear Schrödinger equation for various input boundary terms f and then calculating the resulting normal derivative $\partial_\nu u$ at the boundary 2. The complete dataset consists of pairs of the unknown potentials (A, q) and their corresponding simulated DN maps.

5.4.2 Ablation studies

We conduct the following set of ablation studies to illustrate the need for different components in AFDONet-inv. In Scneario 1, we consider the AFDONet-inv architecture without primal-dual propagation. In Scenario 2, we investigate the impact of considering the dual space by removing the dual branch in the encoder network and the duality map J. Finally, in the third study, we remove both primal-dual propagation and the dual space. Results in Tables 21 and 22 indicate that incorporating primal-dual propagation and dual space is necessary for improving the overall accuracy of AFDONet-inv in terms of reducing MAE and relative L^2 error. To explain this, we observe that, without the dual branch, the primal-dual propagation only gives $z_{p,i}$, which is the approximation of the residual r_i in the AFD theory according to Theorem 5.3.1. In this case, AFDONet-inv essentially approximates $\sum_{i=1}^{N}\langle r_i, \mathscr{B}_i\rangle\mathscr{B}_i$, which asymptotically converges under the pairing $\langle\cdot,\cdot\rangle$ with an error $O(\frac{1}{\sqrt{N}})$. Meanwhile, when we remove primal-dual propagation, the input to the AFD-type decoder is simply $\mathrm{FM}(\tilde{\alpha})$. This way, AFDONet-inv essentially performs the operation $\sum_{i=1}^{N}\langle \mathrm{FM}(\tilde{\alpha}), J(\mathscr{B}_i)\rangle_{\mathcal{B},\mathcal{B}^*}\mathscr{B}_i$ in the Banach space, which converges with an error $O(\frac{1}{N^{k'}})$ if the PDE parameters lie in $C^{k'}$. Finally, when both primal-dual propagation and the dual branch are removed, AFDONet-inv performs the operation $\sum_{i=1}^{N}\langle \mathrm{FM}(\tilde{\alpha}), \mathscr{B}_i\rangle\mathscr{B}_i$, which may not even converge since the kernels

$\{\mathscr{B}_i\}_i$ are not necessarily orthogonal to each other.

Specifically, for the 2-D Darcy flow problem, the permeability field $a(x)$ on a regular domain $[0,1]^2$ is typically considered as a L^∞ (or $C^{s'}$ for $s' < 0.5$ Teng et al. (2024)) function from a statistical or computational perspective due to its irregularity. Thus, in Scenario 1, AFDONet-inv converges with an error $O(\frac{1}{N^{s'}})$ while it converges with an error $O(\frac{1}{N^{0.5}})$ in Scenario 2. This is consistent with the results shown in Table 21, in which the removal of primal-dual propagation has more significant impact on solution accuracy compared to the removal of the dual branch in the encoder. When both primal-dual propagation and dual branch are eliminated from the AFDONet-inv framework, we observe the highest MAE and relative L^2 error values.

Models	MAE	Relative L^2 error
Full	1.82E-01 ± 6.43E-02	6.64E-02 ± 1.38E-03
w/o prop.	3.18E-01 ± 1.06E-01	7.05E-01 ± 2.19E-02
w/o dual	2.39E-01 ± 5.32E-02	1.07E-01 ± 4.45E-02
w/o p.d.	3.56E-01 ± 7.01E-02	1.93E-01 ± 3.74E-02

Table 21: Comparison of MAE and relative L^2 error in permeability field $a(x)$ on Darcy flow equation. Here and hereinafter, "Full" stands for the full AFDONet-inv model, "w/o prop." means "without primal-dual propagation" or Scenario 1 of the ablation studies, "w/o dual" means "without dual branch" or Scenario 2 of the ablation studies, and "w/o p.d." means "without both primal-dual propagation and dual branch".

For the magnetic Schrödinger equation problem on complex manifolds, the deterministic results of Krupchyk et al. (2024) indicate that both A and q can be relaxed to C^∞. This implies a super-algebraic convergence behavior for AFDONet-inv when primal-dual propagation is removed, which explains why both MAE and relative L^2 error values are slightly smaller for Scenario 1 compared to Scenario 2. Last but not least, removing both primal-dual propagation and dual branch leads to the highest MAE and relative L^2 error due to the

worst convergence behavior (or even divergence) for the resulting AFDONet-inv framework.

Models	MAE	Relative L^2 error
Full	1.54E-02 ± 2.78E-03	1.50E-05 ± 6.23E-07
w/o prop.	7.39E-02 ± 1.81E-02	3.06E-04 ± 1.75E-04
w/o dual	7.83E-02 ± 2.20E-05	3.47E-04 ± 1.10E-05
w/o p.d.	8.01E-02 ± 6.75E-03	3.58E-04 ± 3.51E-05

Table 22: Comparison of MAE and relative L^2 error in potentials A and q on magnetic Schrödinger equation.

5.4.3 Comparison with benchmark solvers

Tables 23 shows the MAE, relative L^2 error, and computational efficiency of AFDONet-inv compared to other benchmark solvers for the 2-D Darcy flow problem. In our experiments, the architecture size and training conditions are the same for all methods, and the number of parameters are different (due to the different structures present in different methods) but are in the same order of magnitude. We observe that AFDONet-inv is the second best performing solver in terms of MAE and outperforms all benchmark solvers in terms of relative L^2 error. Since the Darcy flow equation typically lies in L^∞ space, which is a Banach space, our AFDONet-inv incorporating RKBS into our model outperforms other models. Furthermore, we also realize that for this problem, Hilbert space suffices because the permeability field $a(x)$ is often modeled with smoother priors, where the sparsity from Banach space may not be the most significant. This explains the reason that models such as NIPS also performs reasonably well.

Meanwhile, for the nonlinear magnetic Schrödinger equation problem, we see from Table 24 that AFDONet-inv achieves remarkable performance, as it has up to one order of magnitude lower MAE and two to four orders of magnitude lower relative L^2 error compared to other solvers. Since the magnetic Schrödinger equation is highly nonlinear, its inverse

Models	MAE	Relative L^2 error	Training time
Ours	1.82E-01 ± 6.43E-02	6.64E-02 ± 1.38E-03	2.69
NAO	1.11 ± 2.10E-01	7.71E-02 ± 2.09E-03	3.40
NIPS	1.05E-01 ± 4.71E-02	1.56E-01 ± 1.03E-01	0.96
LNO	2.78E-01 ± 3.07E-02	1.00 ± 3.48E-05	2.92
MWT	45.95± 2.48	9.73E-01 ± 6.72E-02	1.65

Table 23: Comparison of MAE, relative L^2 error, training time (seconds per epoch) among different models on Darcy flow equation.

problem is ill-posed, and non-smooth regularization such as L^1 penalty terms can greatly help promote sparsity when reconstructing potentials A and q. Note that sparsity is naturally represented in an L^1 (Banach) space, not an L^2 (Hilbert) space. In this regard, our proposed AFDONet-inv solver, grounded in a novel Banach space representer theorem Parhi & Nowak (2021), can better capture irregular, sparse features on complex manifolds when solving ill-posed inverse problems.

Finally, in terms of computational efficiency, results in Tables 23 suggest that AFDONet-inv is competitive among all state-of-the-art benchmark solvers.

Models	MAE	Relative L^2 error	Training time
Ours	1.54E-02 ± 2.78E-03	1.50E-05 ± 6.23E-07	0.52
NAO	4.65E-01 ± 5.32E-02	6.29E-01 ± 1.96E-01	2.54
NIPS	2.19E-01 ± 9.73E-02	1.32E-01 ± 8.40E-02	0.30
LNO	1.86E-01 ± 6.15E-02	2.89E-03 ± 3.71E-04	0.90
MWT	3.18E-01 ± 1.56E-01	7.05E-01 ± 2.46E-02	1.86

Table 24: Comparison of MAE, relative L^2 error, and training time (seconds per epoch) among different models on magnetic Schrödinger equation.

From the nonlinear magnetic Schrödinger equation results in Table 24, we can quantify

the limitation of Hilbert-space assumptions. This problem is highly ill-posed, and its solution on a complex manifold benefits from sparsity-promoting regularization, which naturally matches a Banach space setting. From Table 24, it is clear that existing state-of-the-art models including NIPS, NAO, and LNO, which are implicitly or explicitly grounded in Hilbert-space frameworks, perform poorly. In contrast, our AFDONet-inv, designed for Banach spaces, achieves a relative error that is two to four orders of magnitude lower than these benchmarks. This significant performance gap indicates how the Hilbert-space assumption in existing models limits their performance in practice.

5.5 Additional Experiments

Here, we conduct an additional experiment considering the magnetic Schrödinger equation problem on a regular rectangular domain $[0, 1] \times [0, 1]$. The results are shown in Table 25.

Models	MAE	Relative L^2 error
Ours	3.20E-02	5.30E-05
NAO	8.09E-01	1.01
NIPS	2.07E-01	8.05E-02
LNO	4.64E-01	1.86E-01
MWT	2.82E-01	1.00

Table 25: Comparison of MAE and relative L^2 error among different models on magnetic Schrödinger equation on $[0, 1] \times [0, 1]$.

Furthermore, we also explore the effect of data augmentation. Given that the data augmentation process is achieved by random permutations. Here, we implement 100 random permutations on top of the training data containing 6000 solution samples. We compare our model to NAO and NIPS, whose performance heavily relies on data augmentation.

Models	MAE	Relative L^2 error	Training time
Ours	9.94E-03	1.20E-05	2.52
NAO	5.98E-02	6.11E-03	2.54
NIPS	4.81E-02	4.29E-03	3.26

Table 26: Comparison of MAE, relative L^2 error and training time (seconds per epoch) among different models on magnetic Schrödinger equation on $[0, 1] \times [0, 1]$ under 100 random permutations.

CHAPTER VI

AUTOMATING THE DESIGN OF NEURAL OPERATORS VIA LARGE LANGUAGE MODELS

Neural network (NN)–based solvers have shown great promise for efficiently solving nonlinear partial differential equations (PDEs) (Li et al., 2020a; Um et al., 2020; Xu & Darve, 2020; Song & Jiang, 2023a; Smith et al., 2020). Although NN-based approaches can produce fast and accurate PDE solutions, a key limitation is that they are typically trained at a specific resolution, which leads to poor generalization to problems at other resolutions. While Bar-Sinai et al. (2019) proposed an NN-based method that learns discretizations of a given PDE from fine to coarse resolutions, it cannot be directly extended to PDEs with different forms or coefficients. Overall, most NN-based approaches must be retrained to handle various resolutions. This motivates the development of resolution-free variants of NN solvers. Noting that standard NN-based methods often rely on prior knowledge (e.g., PDE forms, discretization schemes, coefficients, and boundary/initial conditions), operator learning has been proposed to train neural operators that learn mappings from parameter/function spaces to solution spaces with minimal prior knowledge of the PDE (Lu et al., 2019; Li et al., 2020b; Tripura & Chakraborty, 2023; Gupta et al., 2021; Wang & Wang, 2024).

Among these neural operators, we observe that the top performers differ across PDE problem types. For example, Wang & Wang (2024) reported that their Latent Neural Operator (LNO) exhibits 1.08× and 2.42× relative ℓ^2 error compared to transolver (Wu et al., 2024) on the airfoil and plasticity datasets (Li et al., 2023b), respectively. Furthermore, another key observation is that the performance of neural operators can improve or degrade even with minor architectural changes. To date, the design of neural-operator architectures,

often guided by intuition, expert experience, and trial-and-error, has been "more of an art than a science" (Sanderse et al., 2025). Although neural operators such as the Fourier Neural Operator (FNO) (Li et al., 2020b) and DeepONet (Lu et al., 2019) are grounded in theoretical insights, creating novel components that align with these insights still relies heavily on human expertise. Therefore, theory-driven design of neural operators requires human engagement and remains far from fully automated.

Large language model (LLM) agents have shown great promise in automating processes via human interactions, including mobile tasks (Wen et al., 2024; Guan et al., 2024), hardware design (Xu et al., 2024b), scientific discovery (Zimmermann et al., 2025; Filimonov, 2024; Aamer et al., 2025), code generation (Koziolek et al., 2024; Xu et al., 2024a), hyperparameter tuning (Zhang et al., 2023), and mathematical problem solving (Bian et al., 2025). For solving PDE problems, hybrid approaches (Zhou et al., 2025; Lorsung & Farimani, 2024) incorporate LLMs as components within neural architectures to improve performance. These approaches do not involve process automation, and their architectures are not designed by LLMs. In prior literature, fully automated LLM agents for PDEs include CodePDE (Li et al., 2025) and PINNsAgent (Wuwu et al., 2025). Li et al. (2025) proposed LLM agents that generate and evaluate the code of traditional PDE solvers, whereas PINNsAgent generates code on top of the physics-informed neural network (PINN) architecture. Both methods design the architectures of traditional and PINN-based solvers primarily based on numerical performance rather than theoretical insights. To the best of our knowledge, no prior work has focused on designing neural operators end-to-end with LLM agents guided by theoretical insights.

To bridge the research gaps, we ask the following question question:

Can LLMs design accurate and efficient neural operators driven by theoretical insights?

6.1 Related Work

Neural operators. Operator learning targets mappings between infinite-dimensional function spaces, enabling resolution- and mesh-independent surrogates for PDE families. Two foundational approaches are DeepONet (Lu et al., 2019), which learns nonlinear operators via a branch-trunk factorization, and the Fourier Neural Operator (FNO) (Li et al., 2020b), which applies spectral convolutions to achieve strong resolution transfer on canonical elliptic/parabolic problems. Subsequent variants improve accuracy, efficiency, or inductive bias: U-FNO couples Fourier mixing with U-Net refinements (Wen et al., 2022); F-FNO factorizes spectral weight tensors to lower complexity (Tran et al., 2021); multiwavelet formulations provide multiresolution locality and sparsity (Gupta et al., 2021, 2022; Tripura & Chakraborty, 2023); ONO augments operator learning with orthogonalized kernels and stability enhancements (Xiao et al., 2023b); Galerkin operators embed variational structure (Cao, 2021); LSM exploits learned spectral methods (Wu et al., 2023); and transformer-style operators (OFormer, Transolver) leverage attention for long-range coupling (Li et al., 2022b; Wu et al., 2024). Latent designs (LNO, LaMO) compress fields into compact representations to balance accuracy and cost (Wang & Wang, 2024; Tiwari et al., 2025). Beyond periodic grids, irregular geometries and structured meshes (e.g., airfoil, plasticity, pipe, elasticity) stress resolution transfer and generalization and have motivated the architectural choices and benchmarks used in this work (Li et al., 2023b). Our study differs by asking whether an LLM can *select or synthesize* such operator ingredients end-to-end, guided by mathematical analysis rather than manual trial-and-error.

LLMs in scientific machine learning. LLMs have been used to automate elements of scientific workflows: program synthesis and code repair (Koziolek et al., 2024; Xu et al., 2024a), robotics/mobile task automation (Wen et al., 2024; Guan et al., 2024), hardware and systems design (Xu et al., 2024b), scientific discovery pipelines (Zimmermann et al., 2025; Filimonov, 2024; Aamer et al., 2025), hyperparameter tuning (Zhang et al., 2023),

and mathematical problem solving (Bian et al., 2025). For PDEs, hybrid methods insert LLMs as components within neural architectures or to provide natural-language rationales, but stop short of automating the full design loop (Zhou et al., 2025; Lorsung & Farimani, 2024). Closer to our goal are fully automated agents that generate solvers: CodePDE synthesizes and evaluates traditional PDE codes (Li et al., 2025), and PINNsAgent targets PINN implementations (Wuwu et al., 2025). However, these systems optimize numerical performance without enforcing operator-theoretic design principles. In contrast, we position the LLM as a theory-aware designer that proposes and justifies operator choices (e.g., spectral vs. multiresolution vs. latent), then compiles them into executable PyTorch implementations subject to physics and numerical checks.

Automated agent systems. Role specialization and multi-agent coordination have been shown to improve complex code generation and iterative refinement via planning, self-critique, and division of labor (Carlander et al., 2024; Dong et al., 2024; Takagi et al., 2025). Recent agentic SciML systems instantiate plan-execute-review loops for PDE tasks but largely emphasize execution or empirical tuning (Li et al., 2025; Wuwu et al., 2025). Our pipeline adopts a four-role decomposition, *Theorist* (formal derivation and architectural justification), *Programmer* (faithful implementation), *Critic* (adversarial numerical/software review), and *Refiner* (targeted fixes), explicitly coupling mathematical validation with software iteration. This theory-aware agent design aims to reduce hallucinations, improve stability under discretization changes, and systematize operator selection, compared with prior agent frameworks that lack principled operator-level guidance.

6.2 The Proposed LLM Agent Framework

6.2.1 Neural Operators

Neural operators is a mesh- and resolution- independent neural architectures that learn the mapping from the parameter space to the solution space of the PDE problems. In general,

consider a PDE defined on a spatial domain $\Omega \subset \mathbb{R}^d$ and a time interval $(0, T]$:

$$\mathcal{L}_a[u(x,t)] = f(x,t), \quad \forall (x,t) \in D \times (0,T], \tag{6.2.1}$$

which is subject to a set of initial and boundary conditions. Here, the parameter function $a \in \mathcal{A}$ specifies the coefficients and initial and boundary conditions of Equation 6.2.1. In operator learning, our goal is to construct an accurate approximation for $\mathcal{G} : \mathcal{A} \to \mathcal{F}(D \times [0,T])$, which maps the parameter function a to the corresponding solution function $u(x,t) \in \mathcal{F}$, via a parametric mapping $\mathcal{G}_\theta$. The aim is to learn θ such that $\mathcal{G}_\theta \approx \mathcal{G}$ from a set of training data $\{(a_j, u_j)\}_j$.

6.2.2 Framework Overview

Designing neural operators from a scientific perspective requires several core steps: (i) select or propose a strong neural-operator backbone such as the FNO (Li et al., 2020b) or DeepONet (Lu et al., 2019); (ii) select or propose an appropriate mathematical theory that guarantees desirable properties (e.g., convergence, approximation error, function spaces, etc.); (iii) update the backbone architecture to align with the chosen theory; and (iv) implement and debug the code. As illustrated in Figure, our framework employs a four-agent pipeline to automate this workflow.

System Prompt. Prior studies show that role-playing instructions enable LLMs to collaborate under distinct roles, improving performance, particularly in code generation (Carlander et al., 2024; Dong et al., 2024; Takagi et al., 2025). Building on this idea, we assign the following roles via the system prompt: Theorist, a world-class research mathematician specializing in scientific machine learning; Programmer, an expert PyTorch developer in scientific machine learning; Critic, a skeptical but fair adversarial reviewer for a top AI conference; and Refiner, an expert PyTorch developer focused on debugging and refining complex models.

Step 1: Problem Specification. For a given PDE problem, we first translate the mathematical formulation (6.2.1) into natural language that LLMs can readily understand. This natural-language specification includes the problem name, equation, spatial domain, time interval, initial conditions, boundary conditions, and source terms. Instead of presenting this information in a paragraph, we use a concise, structured natural-language format, which is effective in our framework. For example, we represent 1-D Burgers' equation as:

Problem Statement

PDE Specification

---

name: `1D Burgers' Equation`

equation_latex:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu \frac{\partial^2 u}{\partial x^2}$$

domain: $x \in (0, 1),\ t \in (0, 1)$

initial_condition: $u(x, 0) = u_0(x)$.

boundary_conditions: `Periodic`

viscosity: `0.01`

Step 2: Propose Mathematical Theory (Theorist). The Theorist's ultimate task is to provide rigorous theoretical results to improve the performance of the selected backbone. First, we provide the Theorist with a factory of existing neural operators, including FNO (Li

et al., 2020b), DeepONet (Lu et al., 2019), Transolver (Wu et al., 2024), and LNO (Wang & Wang, 2024), as well as classical architectures such as the variational autoencoder (Kingma et al., 2013) and the Transformer (Li et al., 2022b). We also allow the Theorist to utilize other backbones of its choosing. We then prompt the Theorist to develop clear, rigorous, and efficient mathematical formulations that improve the selected backbone architecture for the specific problem. In this way, the Theorist offers a complete formulation and provides a tailored design of an updated neural architecture in natural language and mathematical form. Finally, we prompt the Theorist to justify its choices and to implement self-correctness checks as determined by the Theorist. The output of this step is a script that derives the theoretical results and descriptions of the proposed neural operator.

Step 3: Code Generation (Programmer). After the Theorist provides the detailed formulation and instructions, we prompt the Programmer to translate them into code for the proposed neural operator, along with any necessary helper functions and package dependencies. We instruct the Programmer to generate the code in PyTorch due to its wide use in the machine learning community.

Step 4: Review Theoretical Results & Implementation (Critic). Motivated by the peer-review mechanism of AI conferences, we prompt the Critic to (i) critically analyze the mathematical derivation provided by the Theorist and the corresponding PyTorch code from the Programmer, and (ii) identify potential inconsistencies in the derivation, edge cases, numerical instabilities, and inefficiencies in the implementation. Finally, we instruct the Critic to provide a structured list of potential issues and concrete suggestions to improve the proposed neural operator.

Step 5: Refine Code (Refiner). The Refiner updates the implementation of the proposed neural operator to address all issues and suggestions identified by the Critic.

Step 6: Code Execution. After the updated Python code is executed, any bugs are recorded and reported back to the Critic, who identifies issues and provides suggestions. The Refiner then revises the code accordingly, and this process repeats until no bugs remain.

6.3 Numerical Experiments

Datasets. We evaluate the performance of neural operators designed by our proposed LLM-agent framework across six benchmark datasets:

1. **Darcy Flow** (Li et al., 2020b): Represents 2D flow through porous media. The PDE is discretized on a 421×421 grid and downsampled to 85×85. Inputs are coefficient fields $a(x)$, and outputs are solutions $u(x, t)$. The dataset contains 1,000 training and 200 testing samples with varying medium structures.

2. **Navier-Stokes** (Li et al., 2020b): Models the 2D incompressible Navier–Stokes equation in vorticity form on the unit torus. Each sample is a 64×64 spatiotemporal field with 20 frames, where the first 10 frames are used to predict the next 10. The dataset consists of 1,000 training and 200 testing samples.

3. **Elasticity** (Li et al., 2023b): Predicts the internal stress of an elastic material discretized into 972 points. Each input is a 972×2 tensor of point positions, and the output is a 972×1 tensor of stresses. The dataset contains 1,000 training and 200 testing samples.

4. **Plasticity** (Li et al., 2023b): Focuses on predicting the deformation of a plastic material under a die of arbitrary shape. The input is a structured mesh of size 101×31, and the output is the deformation over 20 timesteps, recorded as a $20 \times 101 \times 31 \times 4$ tensor. The dataset includes 900 training and 80 testing samples.

5. **Pipe** (Li et al., 2023b): Estimates horizontal fluid velocity within pipes represented as a structured mesh of size 129×129. The input tensor $(129 \times 129 \times 2)$ encodes positions,

while the output tensor ($129 \times 129 \times 1$) gives velocity values. The dataset has 1,000 training and 200 testing samples.

6. **Airfoil** (Li et al., 2023b): Concerns transonic flow over airfoils governed by the Euler equations. Inputs are structured meshes of size 221×51, and outputs are Mach number fields. The dataset includes 1,000 training and 200 testing samples derived from various airfoil designs.

Metrics. We train and evaluate the designed neural operators using the relative ℓ^2 error:

$$\text{relative } \ell^2 \text{ error } = \frac{1}{N} \sum_{i=1}^{N} \frac{\|\mathcal{G}(a_i) - G(a_i)\|_{L^2}}{\|G(a_i)\|_{L^2}}, \tag{6.3.1}$$

where N denotes the number of samples.

Furthermore, we evaluate the correctness and rigor of the mathematical formulations produced by the Theorist through human expert review. The review was conducted by three independent PhD candidates specializing in computational mathematics and neural operators (who are not authors of this paper). The rubric was a binary "Yes/No" judgment based on two criteria: 1) "Is the Theorist's mathematical formulation (e.g., the derivation) correct and sound?" and 2) "Is the connection between the chosen theory and the target PDE justified and logical?"

Baselines. We compare LLM-designed neural operators to 10+ strong baselines and state-of-the-art (SOTA) models designed by humans, including FNO (Li et al., 2020b), U-FNO (Wen et al., 2022), F-FNO (Tran et al., 2021), LNO (Wang & Wang, 2024), ONO (Xiao et al., 2023b), WMT (Gupta et al., 2022), Galerkin (Cao, 2021), LSM (Wu et al., 2023), OFormer (Li et al., 2022b), Transolver (Wu et al., 2024), and LaMO (Tiwari et al., 2025).

Experimental Settings. We evaluate several LLMs in our framework, such as gpt-5, gpt-5-mini, and the reasoning models o1 and o3. All experiments are conducted on a Linux

workstation running Ubuntu (kernel 6.14, glibc 2.39) with Python 3.13.5 (Anaconda), PyTorch 2.8.0+cu129 (CUDA 12.9), an AMD Ryzen 9 9950X (16-core) processor, and a single NVIDIA GeForce RTX 4090 (48 GB) GPU.

6.4 Results and analysis

6.4.1 Can LLMs design neural operators?

We evaluate the capacity of LLM-designed neural operators across six benchmark datasets and find that they outperform existing SOTA models on five of the six datasets (Table 27). Notably, the LLM generates diverse neural architectures tailored to different datasets, underscoring its adaptability across a wide range of tasks. In terms of accuracy, LLM-designed neural operators decrease the relative ℓ^2 error by approximately $\sim 6\%$, depending on the specific problem. Moreover, they demonstrate superior efficiency, achieving a 30-50% reduction in computational time and requiring two to three orders of magnitude fewer parameters (Figure 29). These results suggest that LLMs are capable of designing neural operators that are both efficient and accurate, grounded in theoretical principles. As a side note, while LLM-generated neural operators do not achieve the highest performance on the Darcy dataset, this trade-off in accuracy is made in favor of improved efficiency (see Figure 29(a)).

6.4.2 Does theory-aware design provide benefits?

To further explore the contribution of the theoretical insights provided by the Theorist in the design process, we conduct ablation studies comparing our LLM-agent framework to a variant without the Theorist. In the latter, without theoretical guidance, the LLM agents generate neural-operator code directly (as in (Wuwu et al., 2025; Li et al., 2025)), and the Critic reviews only the numerical aspects. In this way, neural operators are designed in an art-driven rather than theory-aware paradigm. To evaluate the effectiveness of theory-aware design, we measure the performance of both frameworks in terms of accuracy and

Table 27: Relative ℓ^2 error comparisons of LLM-designed neural operators with baselines across six benchmark datasets. Lower relative ℓ^2 error is better.

Models	**Elasticity**	**Plasticity**	**Airfoil**	**Pipe**	**N-S**	**Darcy**
FNO (Li et al., 2020b)	0.0229	0.0074	0.0138	0.0067	0.0417	0.0052
U-FNO (Wen et al., 2022)	0.0239	0.0039	0.0269	0.0056	0.2231	0.0183
F-FNO (Tran et al., 2021)	0.0263	0.0047	0.0078	0.0070	0.2322	0.0077
LNO (Wang & Wang, 2024)	0.0052	0.0029	0.0051	0.0026	0.0845	0.0049
ONO (Xiao et al., 2023b)	0.0118	0.0048	0.0061	0.0052	0.1195	0.0076
WMT (Gupta et al., 2021)	0.0359	0.0076	0.0075	0.0077	0.1541	0.0082
Galerkin (Cao, 2021)	0.0240	0.0120	0.0118	0.0098	0.1401	0.0084
LSM (Wu et al., 2023)	0.0218	0.0025	0.0059	0.0050	0.1535	0.0065
OFormer (Li et al., 2022b)	0.0183	0.0017	0.0183	0.0168	0.1705	0.0124
Transolver (Wu et al., 2024)	0.0062	0.0013	0.0053	0.0047	0.0879	0.0059
LAMO (Tiwari et al., 2025)	0.0050	0.0007	0.0041	0.0038	0.0460	0.0039
LLM (gpt-5)	0.0049	0.0018	0.0043	0.0030	0.0387	0.0132
LLM (gpt-5-mini)	0.0051	0.0023	0.0052	0.0032	0.0420	0.0134
LLM (o1)	0.0047	0.0007	0.0041	0.0023	0.0389	0.0068
LLM (o3)	0.0049	0.0007	0.0038	0.0022	0.0512	0.0064

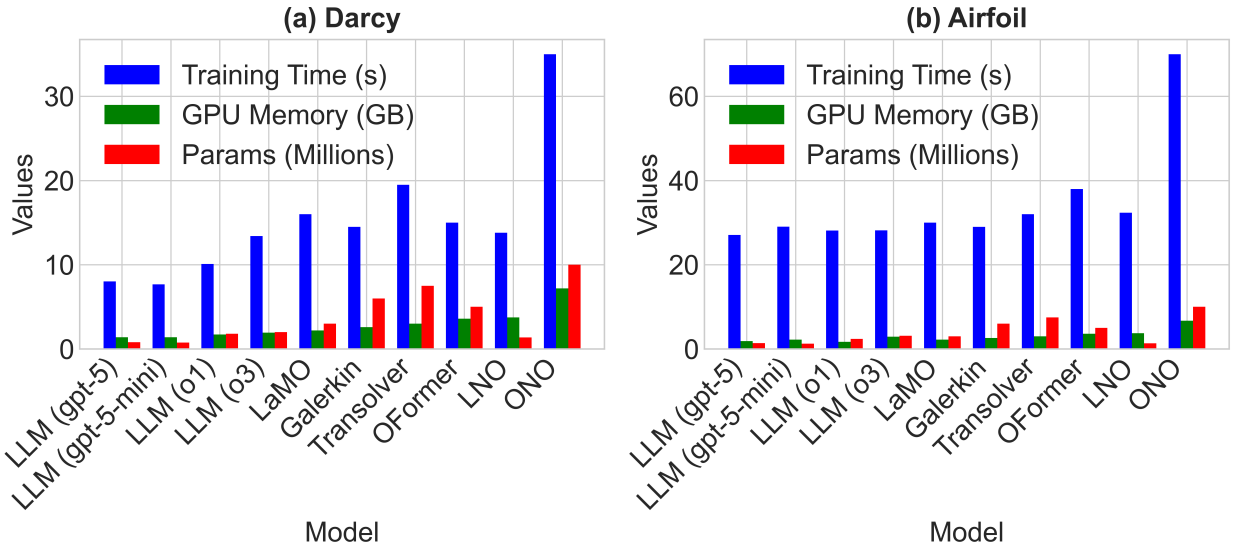


Figure 29: Comparison of computational efficiency including training time (sec per epoch), GPU memory (GB), and parameters count (M) on (a) Darcy and (b) Airfoil datasets.

generalization.

Model	Elasticity	Plasticity	Airfoil	Pipe	N-S	Darcy
With Theorist						
LLM (gpt-5)	0.0049	0.0018	0.0043	0.0030	0.0387	0.0132
LLM (gpt-5-mini)	0.0051	0.0023	0.0052	0.0032	0.0420	0.0134
LLM (o1)	0.0047	0.0007	0.0041	0.0023	0.0389	0.0068
LLM (o3)	0.0049	0.0007	0.0038	0.0022	0.0512	0.0064
Without Theorist						
LLM (gpt-5)	0.0082	0.0047	0.0134	0.0094	0.1630	0.0188
LLM (gpt-5-mini)	0.0086	0.0055	0.0134	0.0116	0.1635	0.0192
LLM (o1)	0.0079	0.0026	0.0098	0.0102	0.1630	0.0106
LLM (o3)	0.0077	0.0031	0.0104	0.0103	0.1639	0.0117

Table 28: Relative ℓ^2 error comparisons of neural operators designed by LLM frameworks with and without Theorist across six benchmark datasets. Lower is better.

Table 28 reports the relative ℓ^2 errors of neural operators obtained with and without the Theorist. In general, neural operators designed by the framework with the Theorist exhibit lower errors, demonstrating the effectiveness of incorporating theoretical insights into the design process. Moreover, the improvements are particularly evident on more complex benchmark datasets, such as *Airfoil* and *Pipe*, where the error differences between the two frameworks range from approximately $3\times$ to $5\times$.

To further examine the generalization of theory-aware neural-operator design via LLMs, we follow the experimental setting in Wang & Wang (2024) and downsample the Darcy dataset from a resolution of 421×421 to 241×241, 211×211, 141×141, 85×85, 61×61, and 43×43. Neural operators are trained on the 43×43 dataset and tested on the others. Figure 30 shows that theory-aware neural operators consistently outperform those without theoretical insights across all resolutions for not only structured grids (e.g., the Navier-Stokes example) but also irregular geometries (e.g., the Elasticity, Airfoil, Pipe, and Plasticity examples). This implies that the LLM design, which incorporates theoretical insights, guarantees that the neural operators exhibit strong generalization capability with respect to the number of sampling points.

6.4.3 Can Critic and Refiner produce better results?

In our framework, collaboration between the Critic and Refiner to identify drawbacks and potential issues in both the theory and the code implementation is a key step toward improving mathematical soundness, performance, and generalization. The Refiner step has been demonstrated to possess strong debugging capability (Li et al., 2025). To analyze the contributions of the Critic and Refiner steps, we compare our full framework to a variant that includes only the Refiner step, which revises code according to reported bugs. Without the Critic step, we hypothesize the following:

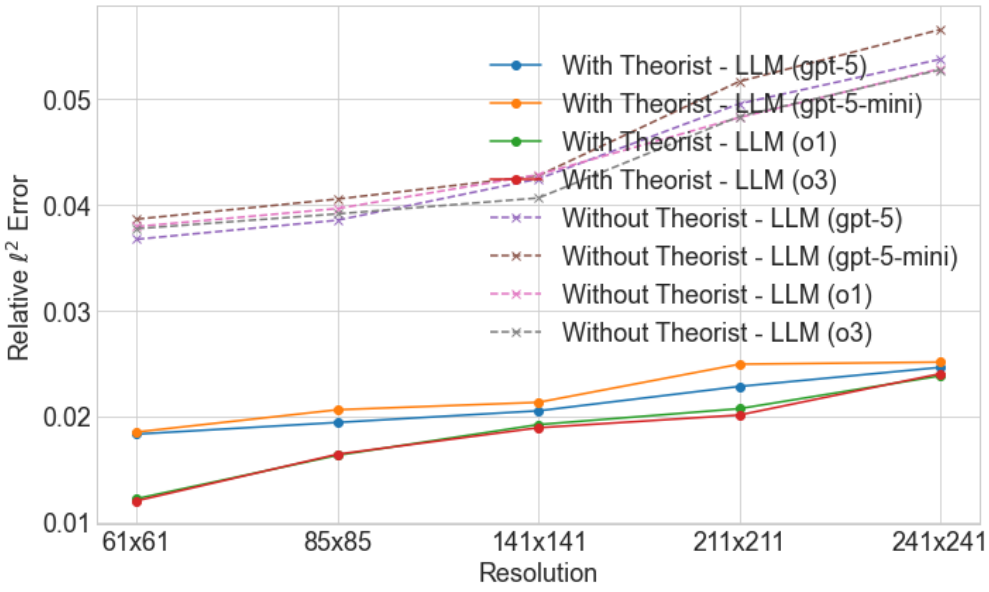


Figure 30: Relative ℓ^2 error comparisons of neural operators designed by LLM frameworks with and without Theorist on different resolutions.

Hypothesis

The quality of the output of Theorist directly decides the performance of the LLM-designed neural operators.

We then conduct experiments on the Darcy dataset and evaluate the relative ℓ^2 error across different resolutions. Moreover, we introduce another LLM (Gemini 2.0 Thinking) as a judge, assigning a score (with a full mark of 5) to quantify the quality of the Theorist's feedback.

Table 29 reports the relative ℓ^2 errors of neural operators designed without the Critic step. For `gpt-5`, `o1`, and `o3`, the Theorist provides high-quality feedback, resulting in only a slight decrease in performance. However, for `gpt-5-mini`, the performance drops by more than 30%.

6.4.4 Can LLMs design neural operators using obscure math?

One key observation during the design process is stated as follows:

Model	61×61	85×85	141×141	211×211	241×241	**Score**
With Critic						
LLM (gpt-5)	0.0183	0.0194	0.0205	0.0228	0.0246	4.5
LLM (gpt-5-mini)	0.0185	0.0206	0.0213	0.0249	0.0251	4.4
LLM (o1)	0.0122	0.0163	0.0192	0.0207	0.0238	4.6
LLM (o3)	0.0120	0.0164	0.0189	0.0201	0.0240	4.6
Without Critic						
LLM (gpt-5)	0.0191	0.0216	0.0224	0.0231	0.0259	4.1
LLM (gpt-5-mini)	0.0305	0.0341	0.0355	0.0368	0.0384	3.5
LLM (o1)	0.0160	0.0177	0.0201	0.0216	0.0249	4.4
LLM (o3)	0.0162	0.0176	0.0204	0.0218	0.0257	4.4

Table 29: Relative ℓ^2 error and score comparisons of neural operators designed by LLM frameworks with and without Critic across six benchmark datasets.

Observation

Theorist tends to generalize the well-established theoretical insights and incorporate them into the selected backbone.

Although the LLM-designed neural operators are novel, we further analyze whether LLMs can design neural operators based on obscure mathematical results, thereby extending the boundary of neural operators from combinations of existing components and incremental contributions to existing architectures to fully new neural operators first proposed in the literature. Obscure mathematical results refer to theories that have attracted attention from only a small fraction of researchers worldwide. Given the lack of training data on obscure mathematics, LLMs may struggle with these topics. Therefore, we conduct a new ablation study that instructs the Theorist to design a neural operator utilizing adaptive Fourier decomposition (AFD) (Qian et al., 2011) on the 1D Burgers' equation dataset (Li

et al., 2020b). We find that the relative ℓ^2 errors of neural operators guided by AFD are consistently one to two orders of magnitude lower than those of neural operators designed using mainstream theories across all four LLM models. For example, with GPT-5, the LLM-designed AFD neural operator attains a relative ℓ^2 error of 0.013, whereas the LLM-designed neural operator based on FNO achieves 0.0006.

In contrast, we also design a neural operator based on AFD ourselves, which reaches a relative ℓ^2 error of 0.0009. That said, AFD has the potential to be integrated into neural operators to improve their performance. However, LLM-designed AFD neural operators perform worse than the human-designed version. Ultimately, we find that the reason is that the Theorist does not fully understand AFD. To illustrate this, we showcase the *Choice and justification* section of the Theorist's feedback, where red indicates the parts of AFD understood by the Theorist:

Choice and justification

- Choose an adaptive Fourier decomposition-based neural operator (AFD-NO): a Fourier neural operator in space with an adaptive, data-dependent selection/mixing of Fourier modes, wrapped inside a stable exponential time-differencing (ETD-1) integrator for the viscous term.

- Why it is best here:

1) Periodic boundary conditions and low-frequency initial conditions make the Fourier basis the natural eigenbasis; Burgers' nonlinearity redistributes energy across modes, so learning in the spectral domain is efficient.

2) Adaptivity is key: noisy initial data and the incipient formation of steep gradients require selectively attending to and mixing the subset of active modes. AFD layers learn soft, sample-dependent spectral masks and frequency-wise linear maps, improving data efficiency and robustness versus fixed-mode FNOs.

3) Stability and inductive bias: we treat the viscous linear part exactly in Fourier (ETD-1), leaving the network to learn only the convective nonlinearity. This aligns with the PDE's semigroup structure and improves numerical stability for $\mu = 0.01$.

With domain knowledge of AFD, this statement has been evaluated by several senior researchers in the field. It was found that LLMs do not grasp the essence of AFD, which is the rational approximation via pole selection in a reproducing kernel Hilbert space. Instead, LLMs conflate Fourier decomposition with the Fourier transform and interpret poles as active modes. Additionally, they ignore the requirements on the basis and the function space needed to implement AFD. Overall, LLMs produce misleading and incorrect content due to hallucination. Moreover, LLMs tend to overfit to similar terms involving adaptivity in Fourier transform theory, overlooking their differences.

We point out that our prompting strategy is highly structured to constrain the LLM's reasoning space, rather than relying on it to invent mathematics from scratch. Specifically,

the prompt first explicitly provides the Theorist with a *factory of existing neural operators* (including FNO, DeepONet, LNO, etc.) as an in-context "toolbox". It then instructs it to first select the most appropriate backbone architecture from this toolbox. Next, it guides it to propose a specific mathematical modification to align that backbone with the specific theoretical properties of the given PDE (such as stiffness, boundary conditions, or conservation laws). Meanwhile, the AFD failure case validates this strategy: when the LLM was forced to use an "obscure" theory not in its pre-trained knowledge base, it began to "hallucinate" and conflate concepts. Therefore, our framework's success relies on guiding the LLM to apply theories it already understands well and are validated, not on its ability to perform novel or obscure mathematical derivations.

Furthermore, it is worth noting that the ablation study presented in Section 6.4.2 only shows the impact of the Theorist component, not its initial correctness. The mechanism we used for the independent validation of the Theorist's output was the human expert review. To clarify, this review was not conducted after the Critic or Refiner intervened. Instead, it was performed specifically on the initial mathematical formulation and architecture description generated by the Theorist, before they were passed to the Critic. For 5 of the 6 benchmark problems we tested, the Theorist's initial theoretical proposal was judged "Yes" (i.e., theoretically correct and logically sound) by the human experts. This indicates that the Theorist provided a solid theoretical foundation in the majority of cases. Subsequently, the Critic's role was focused on identifying implementation-level issues (such as numerical instability, code inefficiency, or edge cases) rather than correcting fundamental theoretical errors. The only exception was the AFD case, where the initial theory was indeed flawed, and this was accurately identified by our human expert review. This confirms that the Theorist's output is, by and large, theoretically reliable before entering the Critic's review loop.

6.4.5 How much time does it take for LLM to design a neural operator?

In terms of design time cost, for a typical benchmark problem (e.g., 2D Navier-Stokes),

the average wall-clock time for our four-agent framework to go from receiving the problem specification to generating a validated, bug-free final operator code is about 35-45 minutes. This process, running on our experimental workstation (equipped with a single NVIDIA RTX 4090), requires an average of 7-9 full agent iterations (i.e., the Theorist $\rightarrow$ Programmer $\rightarrow$ Critic $\rightarrow$ Refiner loop).

While the "human expert effort" baseline is difficult to quantify precisely, it is known that neural operator design is still considered more of an art than a science (Sanderse et al., 2025), typically involving deep intuition, expert experience, numerous iterations, and trial-and-error experimentation. Thus, our proposed automated process represents a significant compression in time cost compared to the days or even weeks required for a human expert to design, implement, and debug a novel, competitive neural operator architecture. Furthermore, it is worth mentioning that the time it takes for a non-expert in neural operators to develop a working neural operator solver for PDEs will be much longer.

CHAPTER VII

CONCLUSIONS AND FUTURE DIRECTIONS

7.1 Summary of Contributions

This dissertation presents comprehensive advances in numerical modeling and computation for solving partial differential equations, with applications spanning from soil moisture monitoring to general PDE problems on complex geometries. The work bridges traditional discretization-based numerical methods with modern neural operator learning, establishing a comprehensive framework for accurate and efficient PDE solution. The research contributions span four major areas: hybrid data-driven numerical methods for nonlinear PDEs, theory-guided neural operator architectures for problems on manifolds, advanced architectures for irregular geometries and inverse problems, and automated neural operator design using large language models.

7.1.1 Hybrid Numerical Methods for the Richards Equation

We introduced the Message Passing Finite Volume Method (MP-FVM), a novel solution algorithm that holistically integrates adaptive fixed-point iteration scheme, encoder-decoder neural network architecture, Sobolev training, and message passing mechanism in a finite volume discretization framework. The MP-FVM algorithm addresses the fundamental challenge of solving the highly nonlinear Richards equation, which governs water flow in unsaturated soils and is critical for precision agriculture and soil moisture monitoring applications. Unlike conventional finite volume methods that convert the discretized equation into large, stiff matrix equations, the MP-FVM algorithm adopts an adaptive fixed-point iteration scheme

where the linearization parameter adjusts dynamically with respect to space, time, and iteration count. This adaptive approach ensures the numerical scheme remains well-posed and achieves convergence within specified iteration limits.

A key innovation of the MP-FVM algorithm is its integration of encoder-decoder neural network architecture with the message passing mechanism. The encoder-decoder architecture learns complex nonlinear relationships between pressure head solutions obtained from different numerical solvers, capturing both the sensitivity to different parameter choices and the distinct topological features of solution spaces. Through persistent homology analysis, we demonstrated that solutions from different sources exhibit fundamentally different topological structures, motivating the use of encoder networks to map between these topological spaces. The message passing mechanism, implemented within the latent space through iteratively solved latent variables, enhances convergence and numerical stability while preserving physical consistency and mass conservation.

We incorporated Sobolev training in the loss functions for both encoder and decoder networks, adding regularization terms that enforce consistency not only at the function value level but also across derivatives. This ensures compatibility and stability across the solution space, preventing small perturbations in solutions at initial conditions or previous time steps from leading to slow convergence or inaccurate solutions. We rigorously proved convergence of the MP-FVM algorithm by showing that the iterative scheme is contractive, with error decreasing geometrically at each iteration. Through comprehensive case studies spanning one-dimensional to three-dimensional problems, including benchmark problems with analytical solutions, layered soil scenarios with discontinuous properties, and realistic irrigation applications with actual center-pivot systems, we demonstrated that MP-FVM achieves superior accuracy compared to state-of-the-art solvers including finite difference methods, physics-informed neural networks, and commercial HYDRUS software. The algorithm also better preserves mass conservation and underlying physical relationships, with mass balance measures consistently exceeding ninety-five percent and often approaching or exceeding one

hundred percent.

7.1.2 Theory-Guided Neural Operators for PDEs on Manifolds

We introduced AFDONet, the first neural PDE solver whose architectural and component design is fully guided by adaptive Fourier decomposition theory. This work presents a new paradigm for designing neural operator frameworks, transforming neural architecture design from an art requiring rare interdisciplinary expertise into a systematic, science-based process grounded in rigorous mathematical theory. AFD is a signal decomposition technique that leverages the Takenaka-Malmquist system and adaptive orthogonal bases to sparsely represent functions in reproducing kernel Hilbert spaces. Unlike classical Fourier methods that use fixed global basis functions, AFD adaptively selects poles that parameterize rational orthogonal bases according to a maximal selection principle, enabling accurate representation of functions with localized features, sharp gradients, or non-periodic structures.

The AFDONet architecture consists of three main components designed following AFD principles. The encoder, based on a variational autoencoder framework, maps PDE inputs to a latent space, exploiting the observation that many PDE solution fields lie on low-dimensional manifolds in high-dimensional function space. The latent-to-RKHS network projects latent representations to their nearest reproducing kernel Hilbert space where AFD operations are defined, explicitly constraining the functional space through feature maps that perform orthogonal projection. This ensures the reproducing property is satisfied and enables rigorous theoretical analysis. The AFD-type dynamic convolutional kernel network decoder reconstructs solutions through adaptive basis selection, replicating AFD operations by performing cross-correlation between mapped representations and orthogonal reproducing kernels, assigning multipliers to each convolutional layer output, and incorporating skip connections.

We proved three main theoretical results establishing the mathematical groundness of AFDONet. First, we bounded the generalization error in terms of the number of training

samples, network depth and width, and the smoothness of the target function, showing that with appropriate network scaling, the expected error decays polynomially in the number of samples. Second, we proved the existence of the RKHS constructed by the latent-to-RKHS network by extending results from approximation theory, showing that for any function in a Hilbert space and any tolerance, there exists a neural network that maps the function to an RKHS with controlled approximation error. Third, we proved convergence of the dynamic convolutional kernel network decoder by leveraging the convergence mechanism of AFD, establishing conditions on layer width, depth, and kernel complexity that ensure reconstructed solutions converge to true solutions.

Through extensive experimental validation on benchmark problems including the Helmholtz equation on planar manifolds with perfectly matched layers, incompressible Navier-Stokes equations on tori, and Poisson equations on quarter-cylindrical surfaces, we demonstrated that AFDONet significantly outperforms existing neural operators such as Fourier Neural Operator, Wavelet Neural Operator, Decomposed Fourier Neural Operator, and DeepONet. The superior performance stems from AFDONet's ability to adapt its basis functions to specific geometry and solution characteristics of each problem. While methods relying on fast Fourier transforms struggle with non-periodic boundaries and curved geometries, AFDONet uses adaptive rational bases parameterized by learned poles that locally adapt to sharp gradients, discontinuities, and complex geometries. Comprehensive ablation studies demonstrated the necessity of each component, showing that the latent-to-RKHS network consistently outperforms latent-to-kernel approaches by at least an order of magnitude for most problems, and that the AFD-type dynamic convolutional kernel network decoder achieves significantly better performance than multi-layer perceptron, propagation, or static convolutional neural network decoders.

7.1.3 Extension to Inverse Problems in Banach Spaces

We extended AFDONet to inverse problems in Banach spaces, introducing AFDONet-inv to address the challenge that most existing operator learning frameworks assume parameters lie in Hilbert spaces, while many real-world inverse problems involve parameters with sparse or discontinuous structures that are better modeled in Banach spaces, particularly L^1 or bounded variation spaces. Inverse problems for PDEs aim to identify unknown parameters from observations of system outputs and are typically ill-posed, requiring careful regularization. By developing an operator learning framework that explicitly accounts for Banach space structures rather than restricting to Hilbert spaces, AFDONet-inv can handle inverse problems where parameters naturally exhibit sparse or discontinuous characteristics, such as identifying piecewise constant material properties or localized sources.

AFDONet-inv extends the AFD framework from reproducing kernel Hilbert spaces to reproducing kernel Banach spaces by constructing appropriate reproducing kernels for Banach spaces and modifying the orthogonalization procedure to account for the duality structure of Banach spaces. The architecture explicitly represents the mapping from operator spaces to parameter spaces, enabling solution of inverse problems where both inputs and outputs are functional objects. The framework incorporates sparsity-promoting regularization through appropriate choice of Banach space norms, naturally enforcing sparse or structured solutions without requiring explicit penalty terms. We derived convergence and stability results for AFDONet-inv, showing that the learned inverse operators are robust to noise in observations through careful analysis of the interplay between approximation error, sampling error, and regularization.

Through benchmark inverse problems including coefficient identification for elliptic PDEs with sparse coefficient fields, source identification problems, and initial condition reconstruction, we demonstrated that AFDONet-inv achieves superior accuracy and stability compared to Hilbert space approaches. The Banach space formulation provides more faithful representation of the true parameter structure, leading to reconstructions that better capture

sharp interfaces and sparse features. Comparisons with traditional variational methods, Bayesian inversion techniques, and Hilbert space operator learning approaches confirmed the advantages of the Banach space formulation for problems with inherently sparse structures, with AFDONet-inv consistently producing more accurate parameter estimates and exhibiting better noise robustness.

7.1.4 Advanced Neural Operator Architectures

We developed Adaptive Fourier Mamba Operators (AFMO), which integrate reproducing kernels for state-space models with Takenaka-Malmquist systems, enabling accurate solutions on diverse geometries and meshes. Frequency-based neural operators are attractive for their ability to capture global dependencies through spectral representations, but they face significant challenges when dealing with irregular geometries and non-uniform meshes. Traditional Fourier transforms require regular grids and periodic boundary conditions, limiting their applicability to complex real-world domains. AFMO addresses these limitations by building upon recent advances in state-space models, particularly the Mamba architecture, which has shown remarkable efficiency in sequence modeling tasks through selective state-space representations.

The key innovation in AFMO is the integration of reproducing kernel theory with state-space models to create a neural operator that can handle irregular geometries while maintaining the computational efficiency of frequency-based approaches. We constructed reproducing kernels that are compatible with the state-space model's hidden state dynamics, allowing the network to learn representations that respect the geometry of the problem domain. The Takenaka-Malmquist system provides the theoretical foundation for adaptively selecting basis functions that can accurately represent solutions on irregular domains. By parameterizing the state-space model's matrices using these adaptive bases, AFMO can selectively focus on important spatial and temporal features while efficiently propagating information across the domain. Unlike methods requiring interpolation or padding to handle irregular domains,

potentially introducing artifacts and reducing accuracy, AFMO operates directly on point clouds or unstructured meshes, with the state-space formulation enabling linear-time complexity in sequence length.

We also developed pole optimization techniques for AFD-based neural layers, addressing the challenge of selecting optimal poles that lie on Riemannian manifolds. In the AFD framework, pole selection is crucial for achieving accurate and efficient decompositions. The classical maximal selection principle provides a greedy algorithm for pole selection, but this approach can be computationally expensive and may not yield globally optimal results. When working with PDEs defined on Riemannian manifolds, poles must lie on the manifold itself, introducing geometric constraints that complicate the optimization problem. Our pole optimization approach formulates the selection of poles as a constrained optimization problem on the manifold, where the objective is to maximize projection magnitudes while respecting manifold geometry.

We developed gradient-based optimization algorithms that operate in the tangent spaces of the manifold, using Riemannian optimization techniques to navigate the curved geometry. The key challenge is computing gradients of projection magnitudes with respect to pole locations while maintaining numerical stability, especially when orthogonalization produces nearly singular systems. We addressed this through a combination of techniques including regularized Gram-Schmidt orthogonalization, manifold-aware learning rate scheduling, and careful initialization strategies that leverage spectral properties of the reproducing kernels. By jointly optimizing pole locations and network weights during training, rather than using fixed or greedily selected poles, we achieve better adaptation to problem-specific features and more compact representations. Experiments on various PDE benchmarks showed that optimized pole placement leads to faster convergence during training, better generalization to unseen parameters, and improved handling of sharp features and discontinuities.

7.1.5 Automated Neural Operator Design

We proposed a four-agent Large Language Model pipeline consisting of specialized agents (Theorist, Programmer, Critic, Refiner) that designs mathematically grounded neural operators end-to-end. The design of neural operators for specific PDE problems currently requires substantial expertise in both the mathematical properties of the equations and the architectural patterns of neural networks. Domain experts must understand the structure of the PDE, identify appropriate functional spaces, select suitable basis representations, and translate these insights into implementable neural architectures through extensive trial and error. This process is time-consuming, requires rare interdisciplinary expertise, and often results in suboptimal designs due to the vast space of possible architectural choices.

Our LLM-assisted framework automates this design process while maintaining mathematical rigor and grounding. The Theorist agent takes as input a description of the PDE problem and relevant mathematical theories, then reasons about the key mathematical structures that should be reflected in the neural architecture. Drawing on its broad knowledge of mathematical theories, approximation methods, and operator theory, the Theorist identifies suitable function spaces, proposes appropriate basis representations, and outlines the mathematical framework that should guide the architecture design. The Programmer agent translates the Theorist's mathematical blueprint into executable code, making specific choices about network layers, activation functions, training procedures, and implementation details while remaining faithful to the mathematical principles identified by the Theorist.

The Critic agent evaluates the designed architecture both theoretically and empirically. It checks whether the implementation correctly reflects the intended mathematical structures, identifies potential issues such as numerical instabilities or violations of physical constraints, and suggests improvements based on mathematical analysis. The Critic performs both static analysis of the code and dynamic analysis of training behavior, checking for issues like gradient pathologies, inappropriate initialization, or insufficient expressiveness. The Refiner agent iteratively improves the architecture based on feedback from the Critic, making adjustments

to address identified issues while preserving the core mathematical framework. This refinement process continues until the architecture meets specified quality criteria in terms of both mathematical soundness and empirical performance.

This LLM-assisted framework consistently outperforms human-designed baselines across diverse PDE benchmarks spanning different equation types, domain geometries, and physical phenomena. We evaluated the framework on benchmark problems including advection-diffusion equations, Burgers' equation, Navier-Stokes equations, and various elliptic and parabolic PDEs. The automatically designed architectures achieve comparable or superior accuracy to carefully hand-crafted baseline methods while requiring significantly less human effort. The framework demonstrates good generalization, producing effective architectures for problems that differ from those seen during the development of the pipeline, suggesting that the LLMs have learned general principles of neural operator design rather than memorizing specific patterns. We demonstrated that the framework is reliable across most mathematical theories commonly used in PDE analysis, including spectral methods, finite element methods, kernel methods, and operator splitting techniques, showing that it can effectively leverage diverse mathematical tools to create specialized neural architectures.

7.2 Limitations and Discussion

While this dissertation presents significant advances in numerical PDE solution methods, several limitations deserve honest assessment and provide opportunities for future improvement. Understanding these limitations is essential for appropriate application of the developed methods and for identifying productive directions for future research.

7.2.1 Limitations of Hybrid Numerical Methods

The MP-FVM algorithm, while achieving superior accuracy and mass conservation compared to conventional methods, has certain practical limitations. First, since the encoder and decoder networks only approximate the true mappings between solution spaces, small but

visible discrepancies may be introduced near domain boundaries even when the finite volume based fixed-point iteration scheme by itself matches ground truth solutions perfectly. This boundary effect arises from the finite capacity of neural networks to represent arbitrarily complex mappings and from the training data distribution, which may not adequately sample the boundary regions. For problems where boundary accuracy is critical, this limitation requires careful consideration, potentially through hybrid approaches that switch to direct adaptive fixed-point iteration near boundaries.

Second, the sensitivity of solution quality to the Sobolev regularization parameter in the loss functions presents a practical challenge. While smaller regularization parameters generally improve accuracy, the optimal value varies across different problem settings and initial conditions. When using pre-trained models for transfer learning, this sensitivity is significantly reduced, but for problems requiring training from scratch, careful tuning of the regularization parameter is necessary. The computational cost of neural network training, while amortized across multiple solves, remains non-negligible. Although we demonstrated substantial reductions in training time through transfer learning and the use of pre-trained models, the initial training phase for a new class of problems still requires more time than conventional direct solvers for single-instance problems.

7.2.2 Limitations of Neural Operator Frameworks

AFDONet, despite its superior performance on manifold-based PDE problems, has certain limitations related to its theoretical assumptions and practical implementation. The framework assumes that PDE solution fields lie on low-dimensional manifolds in high-dimensional function space, which is true for many physical problems but may not hold universally, particularly for highly turbulent or chaotic systems where the effective dimensionality of the solution manifold may be large. For such problems, the encoder may require significantly higher latent space dimensionality, potentially reducing the computational advantages of the approach.

The construction of the latent-to-RKHS network, while theoretically justified, requires careful selection of the reproducing kernel and the number of retained frequency modes. Different PDE problems may have optimal solutions in different reproducing kernel Hilbert spaces, and automatically identifying the most appropriate RKHS for a given problem remains an open question. The current implementation uses Fourier basis kernels, which work well for many problems but may not be optimal for all situations. The adaptive pole selection mechanism, while more flexible than fixed basis approaches, introduces additional hyperparameters related to the maximal selection principle, including the threshold parameters that determine when a pole is considered sufficiently informative.

For AFDONet-inv addressing inverse problems in Banach spaces, the choice of appropriate Banach space norm and the strength of sparsity-promoting regularization significantly impact reconstruction quality. While we demonstrated superior performance with L^1 and bounded variation norms for problems with known sparse or piecewise constant structure, the optimal choice for problems with unknown parameter characteristics is less clear. The framework currently requires some prior knowledge about the expected structure of parameters, whether sparse, smooth, or piecewise constant, to select appropriate functional spaces and regularization schemes. Developing adaptive methods that can automatically identify the most suitable Banach space and regularization strategy from observed data remains an important open problem.

7.2.3 Limitations of Advanced Architectures and Automated Design

AFMO, while effective for irregular geometries and unstructured meshes, requires careful tuning of the state-space model parameters, including the state dimensionality and the selective attention mechanism weights. The optimal configuration of these parameters depends on the specific geometry and PDE characteristics, and currently requires some trial and error or hyperparameter optimization. The computational cost of AFMO, while asymptotically linear in sequence length, can still be substantial for very large-scale problems with mil-

lions of discretization points, as the state-space model must maintain and propagate state information across the entire domain.

The LLM-assisted neural operator design framework, while powerful and demonstrating good generalization, has limitations related to the current capabilities of large language models. First, the framework's reliability depends on the LLM's training data, which may not adequately cover highly specialized or recently developed mathematical theories. For cutting-edge PDE problems involving novel mathematical frameworks, the Theorist agent may produce incomplete or incorrect mathematical guidance. Second, the Programmer agent, while generally effective at translating mathematical concepts to code, can occasionally introduce implementation bugs or make suboptimal architectural choices that are difficult for the Critic to detect automatically.

The iterative refinement process between Critic and Refiner agents, while improving design quality, can sometimes fail to converge for particularly challenging problems, resulting in designs that meet basic functionality requirements but do not achieve optimal performance. The framework currently lacks sophisticated mechanisms for exploring truly novel architectural patterns that deviate significantly from established neural operator paradigms. While the agents can combine existing components in creative ways, generating fundamentally new types of neural network layers or completely novel training procedures remains beyond the current framework's capabilities. Finally, the computational cost of running multiple LLM agents iteratively, while modest compared to manual design effort, is non-negligible and may limit the framework's applicability for rapid prototyping scenarios.

7.3 Future Research Directions

The limitations and open questions identified in this work suggest numerous promising directions for future research that could significantly extend the impact and applicability of the developed methods.

7.3.1 Extensions of Hybrid Numerical Methods

Several promising directions exist for extending the MP-FVM algorithm. First, developing hybrid switch-solve approaches that seamlessly transition between MP-FVM in the domain interior and direct adaptive fixed-point iteration near boundaries could address the boundary accuracy limitations while retaining the superior performance in interior regions. This would require developing criteria for determining the boundary region width and smooth transition mechanisms to avoid introducing artificial discontinuities at the interface between the two solution approaches.

Second, implementing staged or homotopy training strategies for the Sobolev regularization parameter could reduce sensitivity to hyperparameter selection. By starting with pure function value matching (zero regularization parameter) during pre-training, then gradually increasing the parameter to introduce derivative matching, we could achieve better convergence and avoid over-smoothing while still capturing the benefits of Sobolev training. Adaptive regularization schemes that automatically adjust the parameter based on training dynamics and validation performance could further reduce the need for manual tuning.

Third, extending the MP-FVM framework to coupled systems of PDEs, such as simultaneous solution of the Richards equation with solute transport equations for modeling contaminant movement in unsaturated soils, presents both challenges and opportunities. The message passing mechanism would need to operate across multiple latent spaces corresponding to different physical variables, potentially with cross-variable attention mechanisms to capture coupling between equations. Such extensions would enable more comprehensive modeling of real-world agricultural and environmental scenarios.

7.3.2 Extensions of Neural Operator Frameworks

For AFDONet, several theoretical and practical extensions could broaden its applicability. Developing methods for automatically identifying the most appropriate reproducing kernel Hilbert space for a given problem would enhance the framework's autonomy. This could

involve meta-learning approaches that train over a distribution of PDE problems to learn which RKHS characteristics correlate with good performance for different equation types, or Bayesian optimization methods that efficiently explore the space of possible kernel functions and hyperparameters.

Extending AFDONet to time-dependent PDEs on evolving manifolds, where the domain geometry itself changes over time, presents significant theoretical and computational challenges. This would require developing dynamic versions of the latent-to-RKHS network that can adapt to changing manifold structure, possibly through tracking the evolution of reproducing kernels as the manifold deforms. Applications include biological growth processes, fluid-structure interaction problems, and shape optimization.

For inverse problems, developing uncertainty quantification frameworks for AFDONet-inv would provide crucial information about the reliability of parameter estimates. Bayesian neural operator approaches that maintain distributions over possible inverse operators rather than point estimates could quantify both epistemic uncertainty from limited training data and aleatoric uncertainty from noisy observations. This would enable principled decision-making in applications where parameter estimates inform critical interventions.

Extending the Banach space framework to more exotic function spaces such as Besov spaces or spaces with mixed regularity could address inverse problems with parameters exhibiting directional smoothness or anisotropic features. Developing adaptive methods that can identify the appropriate Banach space structure from observed data, perhaps through sparse coding or dictionary learning approaches, would reduce the need for prior knowledge about parameter characteristics.

7.3.3 Extensions of Advanced Architectures

For AFMO, investigating hybrid approaches that combine state-space models with attention mechanisms could capture both long-range dependencies efficiently handled by state-space models and complex local interactions better represented by attention. Developing theoret-

ical understanding of what types of PDE operators and geometries are most amenable to state-space representation versus other approaches would guide appropriate application of AFMO.

Extending AFMO to fully adaptive mesh refinement scenarios, where the mesh structure changes during solution to resolve regions with high gradients or errors, requires developing mechanisms for updating the state-space model structure dynamically. This could involve neural network architectures that can gracefully handle varying input dimensions and connectivity patterns as the mesh is refined or coarsened.

For pole optimization techniques, developing multi-fidelity approaches that use cheap low-fidelity PDE solves to guide pole placement before expensive high-fidelity training could reduce optimization costs. Transfer learning strategies that leverage optimal pole configurations from related problems could provide good initialization, reducing the number of optimization iterations required. Theoretical analysis of the optimization landscape for pole selection on different manifold types would help identify problem characteristics that lead to easy or difficult optimization, guiding development of problem-specific optimization strategies.

7.3.4 Extensions of Automated Design

The LLM-assisted neural operator design framework could be extended in several valuable directions. First, developing mechanisms for the framework to propose and evaluate truly novel neural network components, rather than only combining existing layers in new ways, would enable discovery of fundamentally new architectural patterns. This might involve having the LLM agents interact with symbolic mathematics systems to formally derive neural network structures from first principles, or implementing evolutionary approaches where variants of proposed architectures are systematically explored.

Second, integrating the design framework with automated experiment management systems would enable closed-loop optimization where the framework automatically trains and

evaluates proposed designs, uses performance results to inform refined proposals, and iterates until satisfactory performance is achieved. This would require developing robust error handling to manage training failures and numerical instabilities, and implementing intelligent experiment scheduling to efficiently explore the design space.

Third, extending the framework to handle multi-physics problems and coupled systems would broaden its applicability to real-world engineering applications. The Theorist agent would need access to domain-specific knowledge about coupling mechanisms and interface conditions, and the framework would need to reason about how to represent interactions between different physical phenomena in the neural architecture.

Developing specialized versions of the framework for particular application domains, with domain-specific knowledge built into the agents, could improve reliability and performance. For example, a version focused on fluid dynamics could incorporate deep knowledge of conservation laws, boundary layer phenomena, and turbulence modeling, enabling it to design more effective neural operators for these problems than the general-purpose framework.

7.3.5 Broader Research Directions

Beyond extensions of specific methods developed in this dissertation, several broader research directions could significantly advance the field of neural PDE solution. First, developing rigorous theoretical frameworks for understanding when and why neural operators succeed or fail would enable principled design choices and more reliable application. This includes characterizing the complexity of PDE solution mappings in terms of neural network expressivity requirements, understanding the sample complexity of learning PDE operators as a function of equation characteristics, and deriving generalization bounds that account for the functional nature of inputs and outputs.

Second, integrating neural operators with traditional adaptive numerical methods could create powerful hybrid approaches that leverage the strengths of both paradigms. For example, using neural operators as preconditioners for iterative solvers could accelerate con-

vergence, while adaptive refinement based on neural operator uncertainty estimates could improve efficiency. Developing theoretical frameworks for analyzing these hybrid methods, including convergence guarantees and error bounds, would establish rigorous foundations for their use.

Third, exploring the use of neural operators for multi-scale modeling, where problems span vastly different spatial or temporal scales, represents an important challenge. Hierarchical neural operator architectures that explicitly represent different scales, or attention mechanisms that can dynamically focus on relevant scales depending on input characteristics, could enable efficient solution of multi-scale problems. Applications include modeling molecular dynamics coupled to continuum mechanics, simulating atmospheric processes spanning molecular diffusion to global circulation, and analyzing biological systems across molecular to tissue scales.

Fourth, developing physics-informed neural operators that explicitly incorporate conservation laws, symmetries, and other physical constraints in their architecture or training would improve reliability and generalization. While current approaches either embed physics in loss functions or design architectures guided by mathematical theory, tighter integration of physical knowledge throughout the learning process could yield more robust and trustworthy solutions.

Finally, establishing comprehensive benchmarking frameworks and open datasets for neural PDE solvers would accelerate progress in the field. Standardized benchmark problems spanning diverse equation types, geometries, and difficulty levels, along with reference solutions and evaluation metrics, would enable fair comparison of different methods and identification of remaining challenges. Community-driven efforts to curate such benchmarks and maintain repositories of neural operator implementations would significantly benefit the research community.

7.4 Closing Remarks

This dissertation has presented comprehensive advances in numerical methods for partial differential equations, bridging traditional numerical analysis with modern machine learning to create hybrid algorithms, theory-guided neural operators, and automated design frameworks. From addressing the practical challenges of soil moisture monitoring through the Message Passing Finite Volume Method, to developing mathematically grounded neural operators for problems on arbitrary manifolds and inverse problems in Banach spaces, to creating automated systems for neural operator design using large language models, this work demonstrates that the synergistic combination of classical numerical methods and data-driven approaches can achieve superior performance compared to either paradigm alone.

The key insight underlying this work is that mathematical theory should guide, not merely justify, the design of neural architectures for PDE solution. By systematically translating established mathematical frameworks such as adaptive Fourier decomposition into neural network components, we can create solvers that inherit desirable theoretical properties including convergence guarantees, approximation error bounds, and stability under perturbations. At the same time, the flexibility and learning capabilities of neural networks enable these methods to adapt to problem-specific characteristics and achieve accuracy levels difficult to attain with purely classical approaches.

The methods developed in this dissertation have immediate practical applications in precision agriculture, soil moisture monitoring, groundwater modeling, and numerous other areas where accurate PDE solution is critical but computationally challenging. More broadly, the methodological contributions, particularly the theory-guided design paradigm and the automated neural operator design framework, provide tools and principles that extend far beyond the specific problems studied here. As the scientific and engineering communities increasingly adopt machine learning for computational modeling, the approaches developed in this work offer a path toward creating data-driven methods that are not only powerful

but also mathematically principled, physically consistent, and theoretically grounded.

The future of computational science lies in thoughtful integration of traditional mathematical frameworks with modern data-driven techniques. This dissertation represents steps along that path, demonstrating that rigorous mathematical foundations and flexible machine learning capabilities can be synergistically combined to advance the state of the art in numerical PDE solution. The limitations and open questions identified suggest numerous opportunities for continued progress, and we look forward to future developments that build upon this foundation to enable accurate, efficient, and reliable solution of increasingly complex partial differential equations arising in science and engineering.

REFERENCES

Naafey Aamer, Muhammad Nabeel Asim, Shan Munir, and Andreas Dengel. Automating ai discovery for biomedicine through knowledge graphs and LLM agents. *bioRxiv*, pp. 2025–05, 2025.

Nahla Abdellatif, Christine Bernardi, Moncef Touihri, and Driss Yakoubi. A priori error analysis of the implicit Euler, spectral discretization of a nonlinear equation for a flow in a partially saturated porous media. *Advances in Pure and Applied Mathematics*, 9(1):1–27, 2018.

Robert A Adams and John JF Fournier. *Sobolev spaces*, volume 140. Elsevier, 2003.

Bernard T. Agyeman, Song Bo, Soumya R. Sahoo, Xunyuan Yin, Jinfeng Liu, and Sirish L. Shah. Soil moisture map construction using microwave remote sensors and sequential data assimilation, 2020. URL `https://arxiv.org/abs/2010.07037`.

Bacim Alali and Nathan Albin. Fourier spectral methods for nonlocal models. *Journal of Peridynamics and Nonlocal Modeling*, 2:317–335, 2020.

Mario Amrein. Adaptive fixed point iterations for semilinear elliptic partial differential equations. *Calcolo*, 56(3):30, 2019.

Nakhlé H Asmar. *Partial differential equations with Fourier series and boundary value problems*. Courier Dover Publications, 2016.

Sheldon Axler, Paul Bourdon, and Wade Ramey. *Harmonic Function Theory*, volume 137 of *Graduate Texts in Mathematics*. Springer, 2nd edition, 2001.

Toshiyuki Bandai and Teamrat A. Ghezzehei. Physics-informed neural networks with monotonicity constraints for Richardson-Richards equation: Estimation of constitutive relationships and soil water flux density from volumetric water content measurements. *Water Resources Research*, 57(2):e2020WR027642, 2021.

Yohai Bar-Sinai, Stephan Hoyer, Jason Hickey, and Michael P Brenner. Learning data-driven discretizations for partial differential equations. *Proceedings of the National Academy of Sciences*, 116(31):15344–15349, 2019.

Guy Barles and Halil Mete Soner. Option pricing with transaction costs and a nonlinear Black-Scholes equation. *Finance and Stochastics*, 2(4):369–397, 1998.

Marco Berardi, Fabio Difonzo, Michele Vurro, and Luciano Lopez. The 1D Richards' equation in two layered soils: a filippov approach to treat discontinuities. *Advances in Water Resources*, 115:264–272, 2018.

Luca Bergamaschi and Mario Putti. Mixed finite elements and Newton-type linearizations for the solution of Richards' equation. *International Journal for Numerical Methods in Engineering*, 45(8):1025–1046, 1999.

Earl Berkson and Thomas Gillespie. The generalized M. Riesz theorem and transference. *Pacific Journal of Mathematics*, 120(2):279–288, 1985.

Julius Berner, Dennis Elbrächter, Philipp Grohs, and Arnulf Jentzen. Towards a regularity theory for ReLU networks – chain rule and global error estimates. In *2019 13th International conference on Sampling Theory and Applications (SampTA)*, pp. 1–5, 2019. doi: 10.1109/SampTA45681.2019.9031005.

Rong Bian, Yu Geng, Zijian Yang, and Bing Cheng. Automathkg: The automated mathematical knowledge graph based on llm and vector database. *Computational Intelligence*, 41(4):e70096, 2025.

Alberto Bietti. Approximation and learning with deep convolutional models: A kernel perspective. In *International Conference on Learning Representations*, 2022.

Boris Bonev, Thorsten Kurth, Christian Hundt, Jaideep Pathak, Maximilian Baust, Karthik Kashinath, and Anima Anandkumar. Spherical Fourier neural operators: Learning stable dynamics on the sphere. In *International Conference on Machine Learning*, pp. 2806–2823. PMLR, 2023.

Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. *Concentration Inequalities: A Nonasymptotic Theory of Independence*. Oxford University Press, 2012.

Olivier Bousquet. A Bennett concentration inequality and its application to suprema of empirical processes. *Comptes Rendus Mathematique*, 334(6):495–500, 2002.

Johannes Brandstetter, Daniel E. Worrall, and Max Welling. Message passing neural PDE solvers. *CoRR*, abs/2202.03376, 2022. URL `https://arxiv.org/abs/2202.03376`.

R. Brecht, L. Bakels, A. Bihlo, and A. Stohl. Improving trajectory calculations by flexpart 10.4+ using single-image super-resolution. *Geoscientific Model Development*, 16(8):2181–2192, 2023.

Haim Brezis and Haim Brézis. *Functional analysis, Sobolev spaces and partial differential equations*. Springer, 1 edition, 2011.

Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond Euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

Andreas Buchberger, Christian Häger, Henry D Pfister, Laurent Schmalen, and Alexandre Graell i Amat. Pruning and quantizing neural belief propagation decoders. *IEEE Journal on Selected Areas in Communications*, 39(7):1957–1966, 2020.

Qianying Cao, Somdatta Goswami, and George Em Karniadakis. Laplace neural operator for solving differential equations. *Nature Machine Intelligence*, 6(6):631–640, 2024.

Shuhao Cao. Choose a transformer: Fourier or galerkin. *Advances in neural information processing systems*, 34:24924–24940, 2021.

Jean-Guy Caputo and Yury A. Stepanyants. Front solutions of Richards' equation. *Transport in Porous Media*, 74(1):1–20, August 2008.

Andrei Caragea, Dae Gwan Lee, Johannes Maly, Götz Pfander, and Felix Voigtlaender. Quantitative approximation results for complex-valued neural networks. *SIAM Journal on Mathematics of Data Science*, 4(2):553–580, 2022.

C Carey, TJ Scanlon, and SM Fraser. SUCCA—an alternative scheme to reduce the effects of multidimensional false diffusion. *Applied Mathematical Modelling*, 17(5):263–270, 1993.

Deborah Carlander, Kiyoshiro Okada, Henrik Engström, and Shuichi Kurabayashi. Controlled chain of thought: Eliciting role-play understanding in llm through prompts. In *2024 IEEE Conference on Games (CoG)*, pp. 1–4. IEEE, 2024.

Robert F Carsel and Rudolph S Parrish. Developing joint probability distributions of soil water retention characteristics. *Water Resources Research*, 24(5):755–769, 1988.

Michael A. Celia and Rebecca Zarba. A comparative study of numerical solutions for unsaturated flow. In S. N. Atluri and G. Yagawa (eds.), *Computational Mechanics '88*, pp. 1659–1662, Berlin, Heidelberg, 1988. Springer Berlin Heidelberg.

Michael A Celia, Efthimios T Bouloutas, and Rebecca L Zarba. A general mass-conservative numerical solution for the unsaturated flow equation. *Water Resources Research*, 26(7): 1483–1496, 1990.

Meng Chen and Leevan Ling. Extrinsic meshless collocation methods for PDEs on manifolds. *SIAM Journal on Numerical Analysis*, 58(2):988–1007, 2020.

Qiuhui Chen, Tao Qian, and Lihui Tan. *A Theory on Non-Constant Frequency Decompositions and Applications*, pp. 1–37. Springer International Publishing, Cham, 2020a.

Yinpeng Chen, Xiyang Dai, Mengchen Liu, Dongdong Chen, Lu Yuan, and Zicheng Liu. Dynamic convolution: Attention over convolution kernels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11030–11039, 2020b.

Chun-Wun Cheng, Jiahao Huang, Yi Zhang, Guang Yang, Carola-Bibiane Schönlieb, and Angelica I Aviles-Rivero. Mamba neural operator: Who wins? transformers vs. state-space models for pdes. *arXiv preprint arXiv:2410.02113*, 2024.

Christian Clason, Stanislav Mazurenko, and Tuomo Valkonen. Primal–dual proximal splitting and generalized conjugation in non-smooth non-convex optimization. *Applied Mathematics & Optimization*, 84(2):1239–1284, 2021.

Wojciech M Czarnecki, Simon Osindero, Max Jaderberg, Grzegorz Swirszcz, and Razvan Pascanu. Sobolev training for neural networks. *Advances in Neural Information Processing Systems*, 30, 2017a.

Wojciech Marian Czarnecki, Simon Osindero, Max Jaderberg, Grzegorz Swirszcz, and Razvan Pascanu. Sobolev training for neural networks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 4281–4290, Red Hook, NY, USA, 2017b.

Icamaan B Viegas Da Silva and Paulo JL Adeodato. PCA and Gaussian noise in MLP neural network training improve generalization in problems with small and unbalanced data sets. In *The 2011 International Joint Conference on Neural Networks*, pp. 2664–2669. IEEE, 2011.

Carlos A De Moura and Carlos S Kubrusly. The Courant–Friedrichs–Lewy (CFL) condition. *AMC*, 10(12):45–90, 2013.

Yihong Dong, Xue Jiang, Zhi Jin, and Ge Li. Self-collaboration code generation via chatgpt. *ACM Transactions on Software Engineering and Methodology*, 33(7):1–38, 2024.

Herbert Edelsbrunner and Dmitriy Morozov. *Persistent homology: theory and practice*. eScholarship, University of California, 2013.

Heinz W Engl. Inverse problems and their regularization. In *Computational Mathematics Driven by Industrial Problems: Lectures given at the 1st Session of the Centro Internazionale Matematico Estivo (CIME) held in Martina Franca, Italy, June 21–27, 1999*, pp. 127–150. Springer, 2007.

Lawrence C Evans. *Partial differential equations*, volume 19. American Mathematical Soc., 2010.

Vladimir Sergeevich Fanaskov and Ivan V Oseledets. Spectral neural operators. In *Doklady Mathematics*, volume 108, pp. S226–S232. Springer, 2023.

Matthew W Farthing and Fred L Ogden. Numerical solution of Richards' equation: A review of advances and challenges. *Soil Science Society of America Journal*, 81(6):1257–1269, 2017.

R.A. Feddes and H. Zaradny. Model for simulating soil-water content considering evapotranspiration — comments. *Journal of Hydrology*, 37(3):393–397, 1978.

V Yu Filimonov. Large language models and their role in modern scientific discoveries. *Philosophical Problems of IT & Cyberspace (PhilIT&C)*, 25(1):42–57, 2024.

Xavier Fontaine, Valentin De Bortoli, and Alain Durmus. Convergence rates and approximation results for SGD and its continuous-time counterpart. In Mikhail Belkin and Samory Kpotufe (eds.), *Proceedings of Thirty Fourth Conference on Learning Theory*, volume 134 of *Proceedings of Machine Learning Research*, pp. 1965–2058. PMLR, 2021.

M Ganichev and NJ Kalton. Convergence of the dual greedy algorithm in banach spaces. *New York Journal of Mathematics*, 15:73–95, 2009.

WR Gardner. Some steady-state solutions of the unsaturated moisture flow equation with application to evaporation from a water table. *Soil Science*, 85(4):228–232, 1958.

Dariusz Gasiorowski and Tomasz Kolerski. Numerical solution of the two-dimensional Richards equation using alternate splitting methods for dimensional decomposition. *Water*, 12(6):1780, 2020.

Alex Glyn-Davies, Connor Duffin, O Deniz Akyildiz, and Mark Girolami. ϕ-DVAE: Physics-informed dynamical variational autoencoders for unstructured data assimilation. *Journal of Computational Physics*, 515:113293, 2024.

Somdatta Goswami, Minglang Yin, Yue Yu, and George Em Karniadakis. A physics-informed variational DeepONet for predicting crack path in quasi-brittle materials. *Computer Methods in Applied Mechanics and Engineering*, 391:114587, 2022.

David Gottlieb and Chi-Wang Shu. On the Gibbs phenomenon and its resolution. *SIAM Review*, 39(4):644–668, 1997.

Markus Grasmair, Otmar Scherzer, and Markus Haltmeier. Necessary and sufficient conditions for linear convergence of l1-regularization. *Communications on Pure and Applied Mathematics*, 64(2):161–182, 2011.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

Yanchu Guan, Dong Wang, Zhixuan Chu, Shiyu Wang, Feiyue Ni, Ruihua Song, and Chenyi Zhuang. Intelligent agents with llm-based process automation. In *Proceedings of the 30th*

ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pp. 5018–5027, 2024.

John Guibas, Morteza Mardani, Zongyi Li, Andrew Tao, Anima Anandkumar, and Bryan Catanzaro. Adaptive Fourier neural operators: Efficient token mixers for transformers. *arXiv preprint arXiv:2111.13587*, 2021.

Gaurav Gupta, Xiongye Xiao, and Paul Bogdan. Multiwavelet-based operator learning for differential equations. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 24048–24062. Curran Associates, Inc., 2021.

Gaurav Gupta, Xiongye Xiao, Radu Balan, and Paul Bogdan. Non-linear operator approximations for initial value problems. In *International Conference on Learning Representations (ICLR)*, 2022.

Philip Hartman and Calvin Wilcox. On solutions of the Helmholtz equation in exterior domains. *Mathematische Zeitschrift*, 75(1):228–255, 1961.

Roland Haverkamp, Michel Vauclin, Jaoudat Touma, PJ Wierenga, and Georges Vachaud. A comparison of numerical simulation models for one-dimensional infiltration. *Soil Science Society of America Journal*, 41(2):285–294, 1977.

Junyan He, Seid Koric, Shashank Kushwaha, Jaewan Park, Diab Abueidda, and Iwona Jasiuk. Novel DeepONet architecture to predict stresses in elastoplastic structures with variable complex geometries and loads. *Computer Methods in Applied Mechanics and Engineering*, 415:116277, 2023.

Junyan He, Seid Koric, Diab Abueidda, Ali Najafi, and Iwona Jasiuk. Geom-DeepONet: A point-cloud-based deep operator network for field predictions on 3D parameterized geometries. *Computer Methods in Applied Mechanics and Engineering*, 429:117130, 2024.

R. G. Hills, I. Porro, D. B. Hudson, and P. J. Wierenga. Modeling one-dimensional infiltration into very dry soils: 1. model development and evaluation. *Water Resources Research*, 25 (6):1259–1269, 1989.

Jeffrey AF Hittinger and Jeffrey W Banks. Block-structured adaptive mesh refinement algorithms for Vlasov simulation. *Journal of Computational Physics*, 241:118–140, 2013.

Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.

Zheyuan Hu, Nazanin Ahmadi Daryakenari, Qianli Shen, Kenji Kawaguchi, and George Em Karniadakis. State-space models are accurate and efficient neural operators for dynamical systems. *arXiv preprint arXiv:2409.03231*, 2024.

Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. Pointwise convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 984–993, 2018.

James M Hyman and Mikhail Shashkov. Natural discretizations for the divergence, gradient, and curl on logically rectangular grids. *Computers & Mathematics with Applications*, 33 (4):81–104, 1997.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. pmlr, 2015.

Andrew M Ireson, Raymond J Spiteri, Martyn P Clark, and Simon A Mathias. A simple, efficient, mass-conservative approach to solving Richards' equation (openRE, v1. 0). *Geoscientific Model Development*, 16(2):659–677, 2023.

Dimitrios Kamilis. Numerical methods for the PDEs on curves and surfaces. Master's thesis, Umeå University, Umeå, Sweden, 2013.

Diederik P Kingma, Max Welling, et al. Auto-encoding variational Bayes, 2013.

Nikola Kovachki, Samuel Lanthaler, and Siddhartha Mishra. On universal approximation and error bounds for fourier neural operators. *Journal of Machine Learning Research*, 22 (290):1–76, 2021.

Heiko Koziolek, Sten Grüner, Rhaban Hark, Virendra Ashiwal, Sofia Linsbauer, and Nafise Eskandani. LLM-based and retrieval-augmented control code generation. In *Proceedings of the 1st International Workshop on Large Language Models for Code*, pp. 22–29, 2024.

Katya Krupchyk, Gunther Uhlmann, and Lili Yan. A remark on inverse problems for nonlinear magnetic schrödinger equations on complex manifolds. *Proceedings of the American Mathematical Society*, 152(06):2413–2422, 2024.

Akash Kumar, Mikhail Belkin, and Parthe Pandit. Mirror descent on reproducing kernel banach spaces. *arXiv preprint arXiv:2411.11242*, 2024.

lavenderses. Nssimulation: Simulations of navier-stokes equation in 2d and 3d. `https://github.com/lavenderses/NSsimulation`, 2021.

Chenhao Li, Elijah Stanger-Jones, Steve Heim, and Sang bae Kim. FLD: Fourier latent dynamics for structured motion representation and learning. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=xsd2llWYSA`.

Housen Li, Johannes Schwab, Stephan Antholzer, and Markus Haltmeier. Nett: solving inverse problems with deep neural networks. *Inverse Problems*, 36(6):065005, 2020a.

Jian Li, Yong Liu, and Weiping Wang. Convolutional spectral kernel learning with generalization guarantees. *Artificial Intelligence*, 313:103803, 2022a.

Kangjie Li and Wenjing Ye. D-FNO: A decomposed Fourier neural operator for large-scale

parametric partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 436:117732, 2025.

Shanda Li, Tanya Marwah, Junhong Shen, Weiwei Sun, Andrej Risteski, Yiming Yang, and Ameet Talwalkar. CodePDE: An inference framework for llm-driven PDE solver generation. *arXiv preprint arXiv:2505.08783*, 2025.

Wei Li, Martin Z Bazant, and Juner Zhu. Phase-Field DeepONet: Physics-informed deep operator neural network for fast simulations of pattern formation governed by gradient flows of free-energy functionals. *Computer Methods in Applied Mechanics and Engineering*, 416:116299, 2023a.

Zijie Li, Kazem Meidani, and Amir Barati Farimani. Transformer for partial differential equations' operator learning. *arXiv preprint arXiv:2205.13671*, 2022b.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020b.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020c.

Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. Fourier neural operator with learned deformations for PDEs on general geometries. *Journal of Machine Learning Research*, 24(388):1–26, 2023b.

Senwei Liang, Shixiao W Jiang, John Harlim, and Haizhao Yang. Solving PDEs on unknown manifolds with machine learning. *Applied and Computational Harmonic Analysis*, 71: 101652, 2024.

Rong Rong Lin, Hai Zhang Zhang, and Jun Zhang. On reproducing kernel Banach spaces: Generic definitions and unified framework of constructions. *Acta Mathematica Sinica, English Series*, 38(8):1459–1483, 2022.

Yevgeny Liokumovich, Fernando C Marques, and André Neves. Weyl law for the volume spectrum. *Annals of Mathematics*, 187(3):933–961, 2018.

Ning Liu and Yue Yu. Neural interpretable PDEs: Harmonizing fourier insights with attention for scalable and interpretable physics discovery. In *Forty-second International Conference on Machine Learning*, 2025. URL `https://openreview.net/forum?id=JvRoF9FRga`.

Ning Liu, Siavash Jafarzadeh, and Yue Yu. Domain agnostic Fourier neural operators. *Advances in Neural Information Processing Systems*, 36:47438–47450, 2023.

Cooper Lorsung and Amir Barati Farimani. Explain like i'm five: Using LLMs to improve PDE surrogate models with text. *arXiv preprint arXiv:2410.01137*, 2024.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Lu Lu, Pengzhan Jin, and George Em Karniadakis. DeepONet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*, 2019.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021.

Peter Y Lu, Samuel Kim, and Marin Soljačić. Extracting interpretable physical parameters from spatiotemporal systems using unsupervised learning. *Physical Review X*, 10(3): 031056, 2020a.

Peter Y. Lu, Samuel Kim, and Marin Soljačić. Extracting interpretable physical parameters from spatiotemporal systems using unsupervised learning. *Phys. Rev. X*, 10:031056, 2020b. URL https://link.aps.org/doi/10.1103/PhysRevX.10.031056.

Lok Ming Lui, Yalin Wang, and Tony F Chan. Solving PDEs on manifolds with global conformal parametriazation. In *International Workshop on Variational, Geometric, and Level Set Methods in Computer Vision*, pp. 307–319. Springer, 2005.

Huakun Luo, Haixu Wu, Hang Zhou, Lanxiang Xing, Yichen Di, Jianmin Wang, and Mingsheng Long. Transolver++: An accurate neural solver for pdes on million-scale geometries, 2025. URL https://arxiv.org/abs/2502.02414.

Julien Mairal, Piotr Koniusz, Zaid Harchaoui, and Cordelia Schmid. Convolutional kernel networks. *Advances in Neural Information Processing Systems*, 27, 2014.

Pierre Marchand. helmhurts-python. https://github.com/pierre-24/helmhurts-python, 2023.

Revanth Mattey and Susanta Ghosh. A novel sequential method to train physics informed neural networks for Allen Cahn and Cahn Hilliard equations. *Computer Methods in Applied Mechanics and Engineering*, 390:114474, February 2022. ISSN 0045-7825. doi: 10.1016/j.cma.2021.114474. URL http://dx.doi.org/10.1016/j.cma.2021.114474.

Gregory Matviyenko. On the evaluation of Bessel functions. *Applied and Computational Harmonic Analysis*, 1(1):116–135, 1993.

Mark M. Meerschaert and Charles Tadjeran. Finite difference approximations for fractional advection–dispersion flow equations. *Journal of Computational and Applied Mathematics*, 172(1):65–77, 2004.

W. Merz and P. Rybka. Strong solutions to the Richards equation in the unsaturated zone. *Journal of Mathematical Analysis and Applications*, 371(2):741–749, 2010.

Jing Miao. Convergence of Fourier series in l_p space. *Math. uchicago. edu/may/Rev*, 2014.

Cass T Miller, Glenn A Williams, Carl Tim Kelley, and Michael D Tocci. Robust solution of richards' equation for nonuniform porous media. *Water Resources Research*, 34(10): 2599–2610, 1998.

Oleksandr Misiats and Konstantin Lipnikov. Second-order accurate monotone finite volume scheme for Richards' equation. *Journal of Computational Physics*, 239:123–137, 2013.

Y Mualem. A new model for predicting the hydraulic conductivity of unsaturated porous media. *Water Resources Research*, 12(3):513–522, 1976. ISSN 0043-1397.

Naol Tufa Negero. Fourier transform methods for partial differential equations. *International Journal of Partial Differential Equations and Applications*, 2(3):44–57, 2014.

Abdou Njifenjou. Discrete maximum principle honored by conventional finite volume schemes for diffusion-convection-reaction problems: Proof with geometrical arguments. *London Journal of Research In Science: Natural and Formal*, 25(9):43–57, 2025.

Erfan Orouskhani, Soumya Sahoo, Bernard Agyeman, Song Bo, and Jinfeng Liu. Impact of sensor placement in soil water estimation: a real-case study. *Irrigation Science*, 41(3): 395–411, May 2023.

Rahul Parhi and Robert D Nowak. Banach space representer theorems for neural networks and ridge splines. *Journal of Machine Learning Research*, 22(43):1–40, 2021.

Rom N Parnichkun, Stefano Massaroli, Alessandro Moro, Jimmy TH Smith, Ramin Hasani, Mathias Lechner, Qi An, Christopher Ré, Hajime Asama, Stefano Ermon, et al. State-free inference of state-space models: The transfer function approach. *arXiv preprint arXiv:2405.06147*, 2024.

Tiemo Pedergnana, David Oettinger, Gabriel P Langlois, and George Haller. Explicit unsteady Navier–Stokes solutions and their analysis via local vortex criteria. *Physics of Fluids*, 32(4), 2020.

Salvador Pérez-Esteva and Salvador Valenzuela-Díaz. Reproducing kernel for the herglotz functions in r^ n r n and solutions of the helmholtz equation. *Journal of Fourier Analysis and Applications*, 23:834–862, 2017.

Allan Pinkus. Approximation theory of the MLP model in neural networks. *Acta Numerica*, 8:143–195, 1999. doi: 10.1017/S0962492900002919.

Tao Qian. Intrinsic mono-component decomposition of functions: an advance of Fourier theory. *Mathematical Methods in the Applied Sciences*, 33(7):880–891, 2010.

Tao Qian. Sparse representations of random signals. *Mathematical Methods in the Applied Sciences*, 45(8):4210–4230, 2022.

Tao Qian, Liming Zhang, and Zhixiong Li. Algorithm of adaptive Fourier decomposition. *IEEE Transactions on Signal Processing*, 59(12):5899–5906, 2011.

Tao Qian, Wolfgang Sprößig, and Jinxun Wang. Adaptive fourier decomposition of functions in quaternionic Hardy spaces. *Mathematical Methods in the Applied Sciences*, 35(1):43–64, 2012.

Khalid Rafiq, Wenjing Liao, and Aditya G. Nair. Single-shot prediction of parametric partial differential equations, 2025. URL `https://arxiv.org/abs/2505.09063`.

Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017.

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.

L. A. Richards. Capillary conduction of liquids through porous mediums. *Physics*, 1(5): 318–333, 1931.

Saburou Saitoh, Yoshihiro Sawano, et al. *Theory of reproducing kernels and applications*, volume 44. Springer, 2016.

Benjamin Sanderse, Panos Stinis, Romit Maulik, and Shady E. Ahmed. Scientific machine learning for closure models in multiscale problems: A review. *Foundations of Data Science*, 7(1):298–337, 2025.

Thomas Schuster, Bernd Hofmann, and Barbara Kaltenbacher. Tackling inverse problems in a Banach space environment: from theory to applications. *Inverse Problems*, 28(10): 100201, 2012.

Jiří Šimůnek, Martinus Th Van Genuchten, and Miroslav Šejna. Recent developments and applications of the HYDRUS computer software packages. *Vadose Zone Journal*, 15(7), 2016.

Valery S Sizikov et al. *Well-posed, ill-posed, and intermediate problems with applications*. De Gruyter, 2011.

Jonathan D Smith, Kamyar Azizzadenesheli, and Zachary E Ross. Eikonet: Solving the eikonal equation with deep neural networks. *IEEE Transactions on Geoscience and Remote Sensing*, 59(12):10685–10696, 2020.

Roger E. Smith, Keith Smettem, Philip Broadbridge, and D.A. Woolhiser. *Infiltration Theory for Hydrologic Applications*. American Geophysical Union, 2002.

Emir Sokic, Samim Konjicija, Melita Ahic-Djokic, and Almir Salihbegovic. Stability issues

in discretization of wave equation. In *2011 18th International Conference on Systems, Signals and Image Processing*, pp. 1–4. IEEE, 2011.

Zeyuan Song and Zheyu Jiang. A data-driven modeling approach for water flow dynamics in soil. In *Computer Aided Chemical Engineering*, volume 52, pp. 819–824. Elsevier, 2023a.

Zeyuan Song and Zheyu Jiang. A data-driven modeling approach for water flow dynamics in soil. In Antonios C. Kokossis, Michael C. Georgiadis, and Efstratios Pistikopoulos (eds.), *33rd European Symposium on Computer Aided Process Engineering*, volume 52 of *Computer Aided Chemical Engineering*, pp. 819–824. Elsevier, 2023b.

Zeyuan Song and Zuoren Sun. Representing functions in H^2 on the Kepler manifold via WPOAFD based on the rational approximation of holomorphic functions. *Mathematics*, 10(15):2729, 2022.

Andrew M Stuart. Inverse problems: a bayesian perspective. *Acta numerica*, 19:451–559, 2010.

Daniel J Tait and Theodoros Damoulas. Variational autoencoding of PDE inverse problems. *arXiv preprint arXiv:2006.15641*, 2020.

Hirohane Takagi, Shoji Moriya, Takuma Sato, Manabu Nagao, and Keita Higuchi. A framework for efficient development and debugging of role-playing agents with large language models. In *Proceedings of the 30th International Conference on Intelligent User Interfaces*, pp. 70–88, 2025.

Naoya Takeishi and Alexandros Kalousis. Physics-integrated variational autoencoders for robust and interpretable generative modeling. *Advances in Neural Information Processing Systems*, 34:14809–14821, 2021.

Yuntian Teng, Zihao Li, and Cheng Chen. A comprehensive review of pre-darcy flows in low-permeability porous media. *arXiv preprint arXiv:2401.04930*, 2024.

Karn Tiwari, Niladri Dutta, NM Krishnan, et al. Latent mamba operator for partial differential equations. *International Conference on Machine Learning*, 2025.

Fred T Tracy. Clean two-and three-dimensional analytical solutions of Richards' equation for testing numerical solvers. *Water Resources Research*, 42(8), 2006.

Alasdair Tran, Alexander Mathews, Lexing Xie, and Cheng Soon Ong. Factorized fourier neural operators. *arXiv preprint arXiv:2111.13802*, 2021.

Tapas Tripura and Souvik Chakraborty. Wavelet neural operator for solving parametric partial differential equations in computational mechanics problems. *Computer Methods in Applied Mechanics and Engineering*, 404:115783, 2023.

Kiwon Um, Robert Brand, Yun Raymond Fei, Philipp Holl, and Nils Thuerey. Solver-in-the-loop: Learning from differentiable physics to interact with iterative pde-solvers. *Advances in neural information processing systems*, 33:6111–6122, 2020.

M Th Van Genuchten. A closed-form equation for predicting the hydraulic conductivity of unsaturated soils. *Soil Science Society of America Journal*, 44(5):892–898, 1980.

Martin J Wainwright. *High-dimensional statistics: A non-asymptotic viewpoint*, volume 48. Cambridge university press, 2019.

Tian Wang and Chuang Wang. Latent neural operator for solving forward and inverse pde problems. *Advances in Neural Information Processing Systems*, 37:33085–33107, 2024.

Ze Wang, Chi Man Wong, Agostinho Rosa, Tao Qian, and Feng Wan. Adaptive fourier decomposition for multi-channel signal analysis. *IEEE Transactions on Signal Processing*, 70:903–918, 2022.

George Neville Watson. *A treatise on the theory of Bessel functions*, volume 3. The University Press, 1922.

Gege Wen, Zongyi Li, Kamyar Azizzadenesheli, Anima Anandkumar, and Sally M Benson. U-fno—an enhanced fourier neural operator-based deep-learning model for multiphase flow. *Advances in Water Resources*, 163:104180, 2022.

Hao Wen, Yuanchun Li, Guohong Liu, Shanhui Zhao, Tao Yu, Toby Jia-Jun Li, Shiqi Jiang, Yunhao Liu, Yaqin Zhang, and Yunxin Liu. Autodroid: Llm-powered task automation in android. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking*, pp. 543–557, 2024.

Haixu Wu, Tengge Hu, Huakun Luo, Jianmin Wang, and Mingsheng Long. Solving high-dimensional PDEs with latent spectral models. *arXiv preprint arXiv:2301.12664*, 2023.

Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. Transolver: A fast transformer solver for pdes on general geometries. *arXiv preprint arXiv:2402.02366*, 2024.

Hio Tong Wu, Ieng Tak Leong, and Tao Qian. Adaptive rational approximation in bergman space on bounded symmetric domain. *Journal of Mathematical Analysis and Applications*, 506(1):125591, 2022.

Qingpo Wuwu, Chonghan Gao, Tianyu Chen, Yihang Huang, Yuekai Zhang, Jianing Wang, Jianxin Li, Haoyi Zhou, and Shanghang Zhang. PINNsAgent: Automated PDE surrogation with large language models. *arXiv preprint arXiv:2501.12053*, 2025.

Xiongye Xiao, Defu Cao, Ruochen Yang, Gaurav Gupta, Gengshuo Liu, Chenzhong Yin, Radu Balan, and Paul Bogdan. Coupled multiwavelet neural operator learning for coupled partial differential equations. *arXiv preprint arXiv:2303.02304*, 2023a.

Xiongye Xiao, Defu Cao, Ruochen Yang, Gaurav Gupta, Gengshuo Liu, Chenzhong Yin, Radu Balan, and Paul Bogdan. Coupled multiwavelet neural operator learning for coupled partial differential equations, 2025. URL `https://arxiv.org/abs/2303.02304`.

Zipeng Xiao, Zhongkai Hao, Bokai Lin, Zhijie Deng, and Hang Su. Improved operator learning by orthogonal attention. *arXiv preprint arXiv:2310.12487*, 2023b.

Jia Xu, Weilin Du, Xiao Liu, and Xuejun Li. Llm4workflow: An llm-based automated workflow model generation tool. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, pp. 2394–2398, 2024a.

Kailai Xu and Eric Darve. Adcme: Learning spatially-varying physical fields using deep neural networks. *arXiv preprint arXiv:2011.11955*, 2020.

Kangwei Xu, Ruidi Qiu, Zhuorui Zhao, Grace Li Zhang, Ulf Schlichtmann, and Bing Li. Llm-aided efficient hardware design automation. *arXiv preprint arXiv:2410.18582*, 2024b.

Yuan Xu. Funk–hecke formula for orthogonal polynomials on spheres and on balls. *Bulletin of the London Mathematical Society*, 32(4):447–457, 2000.

Zong-Ben Xu and Gary F Roach. Characteristic inequalities of uniformly convex and uniformly smooth banach spaces. *Journal of Mathematical Analysis and Applications*, 157 (1):189–210, 1991.

Qile Yan, Shixiao Willing Jiang, and John Harlim. Spectral methods for solving elliptic PDEs on unknown manifolds. *Journal of Computational Physics*, 486:112132, 2023.

Dmitry Yarotsky. Error bounds for approximations with deep ReLU networks. *Neural networks*, 94:103–114, 2017.

Huaiqian You, Quinn Zhang, Colton J. Ross, Chung-Hao Lee, and Yue Yu. Learning deep implicit fourier neural operators (ifnos) with applications to heterogeneous material modeling. *Computer Methods in Applied Mechanics and Engineering*, 398:115296, 2022.

Yue Yu, Ning Liu, Fei Lu, Tian Gao, Siavash Jafarzadeh, and Stewart A Silling. Nonlocal attention operator: Materializing hidden knowledge towards interpretable physics discov-

ery. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=uSKzEaj9zJ.

Rebecca Zarba. *A Numerical Investigation of Unsaturated Flow*. Massachusetts Institute of Technology, Department of Civil Engineering, Cambridge, MA, 1988.

Eberhard Zeidler. *Nonlinear functional analysis and its applications. I. Fixed-Point Theorems*. Springer Science & Business Media, 1986.

Haizhang Zhang, Yuesheng Xu, and Jun Zhang. Reproducing kernel banach spaces for machine learning. *Journal of Machine Learning Research*, 10(12), 2009.

Michael R Zhang, Nishkrit Desai, Juhan Bae, Jonathan Lorraine, and Jimmy Ba. Using large language models for hyperparameter optimization. *arXiv preprint arXiv:2312.04528*, 2023.

Jianwei Zheng, Wei Li, Ni Xu, Junwei Zhu, and Xiaoqin Zhang. Alias-free mamba neural operator. *Advances in Neural Information Processing Systems*, 37:52962–52995, 2024.

Weiheng Zhong and Hadi Meidani. Pi-VAE: Physics-informed variational auto-encoder for stochastic differential equations. *Computer Methods in Applied Mechanics and Engineering*, 403:115664, 2023.

Hang Zhou, Yuezhou Ma, Haixu Wu, Haowen Wang, and Mingsheng Long. Unisolver: Pde-conditional transformers towards universal neural pde solvers. In *Forty-second International Conference on Machine Learning*, 2025.

SR Zhu, LZ Wu, ZH Shen, and RQ Huang. An improved iteration method for the numerical solution of groundwater flow in unsaturated soils. *Computers and Geotechnics*, 114:103113, 2019.

Yoel Zimmermann, Adib Bazgir, Alexander Al-Feghali, Mehrad Ansari, Joshua Bocarsly, L Catherine Brinson, Yuan Chiang, Defne Circi, Min-Hsueh Chiu, Nathan Daelman, et al.

32 examples of LLM applications in materials science and chemistry: towards automation, assistants, agents, and accelerated scientific discovery. *Machine Learning: Science and Technology*, 2025.