University of Nottingham
UK | CHINA | MALAYSIA

Neural Network Methods for PDEs and Perron-Frobenius Operator Problems:

Neumann Series, Invariant Densities, and Test-Space Adaptivity

Tanakorn Udomworarat

Supervised by

Prof. Kristoffer van der Zee

Dr. Martin Richter

Dr. Ignacio Brevis

Dr. Sergio Rojas

Thesis submitted to the University of Nottingham for the degree of Doctor of Philosophy

Abstract

Deep learning has become a popular area of research due to the high expressivity and scalability of neural networks, which make them powerful tools for solving many types of problems. Alongside these developments, problems related to Perron-Frobenius operators (or transfer operators) have been extensively studied and applied across various fields. Developing neural network methods for these operators would offer significant advantages, particularly in solving complex or high-dimensional problems.

This thesis aims to develop a deep-learning framework for solving problems related to these operators and partial differential equations (PDEs). Specifically, we develop neural network methods to approximate solutions to Neumann series and invariant density problems involving the Perron-Frobenius operator. The core methodologies investigated in this study include physics-informed neural networks (PINNs), variational PINNs (VPINNs), and robust VPINNs (RVPINNs). We further aim to design an adaptive version of the RVPINN framework for PDEs to improve the convergence rate by adaptively refining the test-space mesh.

The first major contribution of this study involves the Neumann series of non-expansive Perron-Frobenius operators. We use PINNs and RVPINNs to approximate solutions in their strong and variational forms, respectively. We provide a priori error estimates for quasi-minimizers of the associated loss functions. We present some numerical results for 1D, 2D, and high-dimensional examples to show the performance of our methods. We also demonstrate the applicability of our methods by approximating interior densities in a two-cavity system.

In the context of invariant density problems, we introduce three loss functions for approximating Perron-Frobenius invariant densities. These loss functions are specifically designed to enforce the constraints from the maximum entropy method (MEM). Furthermore, we derive the analytical gradients when using a two-layer rectified linear

unit (ReLU) neural network to improve computational speed. We demonstrate the efficacy of our method through 1D examples, highlighting its ability to approximate invariant densities where traditional discretization techniques might be inefficient.

Finally, we establish theoretical and practical adaptive strategies for the RVPINN framework. We present upper bounds for approximation errors based on the RVPINN loss function and prove the equivalence between these bounds. We then develop adaptive algorithms guided by a Riesz discrepancy indicator and provide their convergence results. Furthermore, we propose a computable refinement indicator and prove that, under the saturation assumption, it serves as a reliable and efficient error estimator for the discrepancy between the discrete and continuous Riesz representatives. We demonstrate the effectiveness of the proposed methodology by solving three problems: a Poisson equation on a square domain with a smooth solution, an elliptic interface problem with a discontinuous coefficient, and a Poisson equation on an L-shaped domain with a singular solution.

Acknowledgements

This doctoral thesis would not have been possible without the guidance and support of many people. I would like to express my sincere thanks to those who have played a vital role in my PhD journey.

First, I would like to express my deepest gratitude to my principal supervisor, Kristoffer van der Zee, for his unwavering support throughout the years. his scientific sharpness and dedication provided the foundation for this thesis. Working with him has been a genuine pleasure, and I am truly grateful for the wealth of knowledge he shared with me.

Next, I would like to thank Martin Richter for your insightful feedback and stimulating discussions. These conversations were instrumental in improving this thesis. I would also like to thank Ignacio Brevis for useful feedback and meaningful discussion on both academic and non-academic topics. I am also thankful to Sergio Rojas for kind discussions, advice, and collaboration.

I was supported by a PhD scholarship from the Royal Thai Government Scholarship (Ministry of Higher Education, Science, Research and Innovation), and would like to thank the staff at the Office of Educational Affairs, the Royal Thai Embassy in London, for their administrative assistance. In addition, I wish to thank the staff at the School of Mathematical Sciences, University of Nottingham, for their support during my time here.

To my colleagues and office-mates—thank you for creating such a pleasant and vibrant atmosphere. Our informal chats provided a much-needed balance to challenges of research.

Beyond the academic world, I am thankful to my friends for the joyful times that kept my mind away from equations and code. Whether we were playing sports, playing board games, or simply sharing a meal, your friendship kept me grounded.

Finally, my family deserves special recognition for their unconditional love. I thank my parents for their support and for encouraging me to follow my dreams. Your belief in me has been my greatest strength.

Contents

Chapter 1

Introduction

In the era of high-performance computing, the application of artificial neural networks to scientific computing is one of the significant areas of interest within deep learning research. Developing neural network frameworks for PDEs and problems related to transfer operators offers an alternative to traditional numerical methods. This chapter provides the essential concepts, outlines the objectives of this thesis, and points out our contributions.

1.1 Background

Deep learning has been a research subject for a long time. One of the most popular algorithms in this field is the *artificial neural network* or *neural network*. The main idea of a neural network is to create complex structures by connecting many simple functions together. Because neural networks have high expressivity [50] and scalability [81], they are a powerful tool for solving many types of problems. Moreover, the neural network approach allows us to solve mathematical problems by using the advantages of data, which is known as a *data-driven method*. Several neural network frameworks have gained popularity for solving integro-differential equations [7, 17, 58, 80, 99, 107], most notably physics-informed neural networks (PINNs) [84], variational PINNs (VPINNs) [58], and robust VPINNs (RVPINNs) [87]. Beyond these methods, operator-learning frameworks such as deep operator networks (DeepONets) [72] and Fourier neural operators [66] have been proposed to learn mappings between infinite-dimensional function spaces.

The statistical properties of dynamical systems play an important role in providing information about a system's behavior. These properties have received significant interest over recent decades [40, 46], particularly in systems exhibiting chaotic behavior [25, 64]. One possible approach to studying the statistical behavior of these systems is by analyzing the evolution of densities using the *transfer operator* known as the *Perron-Frobenius operator*. This operator has been applied to various fields of research, including engineering [51, 104], earth sciences [44], and atmospheric sciences [97, 98].

One interesting problem involving the Perron-Frobenius operator is finding the *steady state*, which represents the accumulated density function after infinitely many iterations. This density can be expressed as the *Neumann series* of the operator [86, 94]. Another important problem is finding an invariant density of the operator. It describes the long-term statistical distribution of orbits in a dynamical system, capturing how trajectories are distributed in phase space regardless of their initial conditions [64]. These two problems strongly motivate our study of the Perron-Frobenius operator.

1.2 Aims and scope

The objective of this research is to develop a deep-learning framework for solving problems related to the Perron-Frobenius operator and PDEs. Specifically, we aim to develop neural network methods to approximate solutions to Neumann series problems and to approximate invariant densities involving the Perron-Frobenius operator. The core methodologies investigated in this study include PINNs, VPINNs, and RVPINNs. We further aim to design an adaptive version of the RVPINN framework for PDEs to improve the convergence rate by adaptively refining the test-space mesh.

1.3 Main contributions

Our main contributions are as follows:

Neural network methods for Perron-Frobenius Neumann series: We establish an abstract setting for non-expansive Perron-Frobenius operators mapping

from L^p spaces to themselves. We state the strong form and variational form for the Neumann series problems. We ensure the well-posedness of the variational formulation in L^2 and L^p spaces by verifying the assumptions of Lax-Milgram Theorem and Banach-Nečas-Babŭska Theorem, respectively. We apply PINNs and RVPINNs for solving these problems in the strong form and variational form, respectively. We present a priori error estimates of our methods in terms of quasi-minimizers of the loss function. We use natural modification of the local Fortin's condition to analyze the error for RVPINNs. We show the performance of our methods using several one and two dimensional dynamical systems, including the tent map, the boundary map in a circular domain, and the standard map. We compare our methods against a fixed-grid-based method and the truncated sum of Neumann series. Moreover, we show an application in approximating interior densities in a two-cavity system. Finally, we show some numerical results for high-dimensional problems with N-coupled standard maps.

Neural network methods for Perron-Frobenius invariant densities: We define the loss function based on the optimization problem of the maximum entropy method (MEM). To link our method with a data-driven approach, we approximate the integration in our loss function using the Gauss-Kronrod integration. We train the neural network to minimize this loss function, calling the method the deep maximum entropy method (DMEM). Given that our method relies on constraints, we analyze the impact of the number of constraints on the accuracy of our algorithm. We also propose two new loss functions for invariant density problems to overcome the limitations of the previous approach: one replaces the entropy term with a quadratic term, and the other omits the entropy term altogether. Since a loss function relies on integration approximations, we improve the training process by computing the exact integration. Without the entropy term, we can compute the exact gradient for shallow neural networks with ReLU activation functions. We show that training the neural network without gradient approximation improves the method's performance in terms of training time.

Test-space adaptive RVPINNs: We derive upper bounds for the approximation error in terms of the RVPINN loss function and the discrepancy between two Riesz representatives. We then establish convergence results by considering key factors such

as the approximation capacity of neural networks, the efficiency of nonlinear solvers, and the quality of the discrete test spaces. Furthermore, we develop a test-space adaptive RVPINN algorithm for PDEs and introduce a computable refinement indicator based on the Riesz representatives of two nested test spaces. Under the saturation assumption, we demonstrate that this indicator serves as a reliable and efficient error estimator for the discrepancy between discrete and continuous representations. To validate our approach, we apply the method to Poisson equations and interface problems with smooth, kinked, and singular solutions. Finally, we evaluate the performance of our adaptive RVPINN approach by comparing it to the standard finite element method (FEM) using equivalent meshes and test spaces.

1.4 Thesis overview

The structure of this thesis is as follows. In Chapter 2 we review fundamental topics in deep learning, including neural network architectures and the optimization methods used for training. We discuss their applications to PDEs, which serve as the inspiration for our approaches. We also cover relevant error estimation theory. In Chapter 3, we provide the motivation for studying Perron-Frobenius operators and describe their application in ray-tracing methods. We discuss the abstract setting of these operators and their adjoints, the *Koopman operators*. Two key problems are introduced: the Neumann series problem and the invariant density problem, along with illustrative examples. Furthermore, we review classical numerical methods for solving them, such as Ulam's method, the Galerkin and Petrov-Galerkin methods, and the maximum entropy method. Chapter 4 focuses on neural network methods for Neumann series problems. The content in this chapter is based on our work [100] published in the Computer Methods in Applied Mechanics and Engineering (CMAME) journal. Chapter 5 presents neural network methods for solving invariant density problems, followed by numerical results and discussion. Chapter 6 introduces the test-space adaptive RVPINNs. Finally, Chapter 7 summarizes conclusions and gives possible directions for extensions.

Chapter 2

Neural Networks and Applications in Scientific Computing

2.1 Introduction

Artificial neural networks or neural networks are mathematical models inspired by the learning process of the human biological brain. Structurally, a neural network consists of multiple layers, where the inputs are passed sequentially from one layer to the next layer. Each layer is composed of neurons, which act as processing units that transform input data through nonlinear functions [52].

Neural networks have been successfully applied to many tasks, including classification [62], pattern recognition [53], and natural language processing [31]. In recent decades, they have also become a tool for solving partial differential equations (PDEs) [20,58,59,84,87,93,107]. Due to the universal approximation property [54,65] (see Page 7), high expressivity[1] [50], and scalability [81] of these models, neural networks have gained popularity across many scientific research fields. Generally, the parameters of a neural network are optimized to minimize a loss function specific to the problem at hand. This is achieved by applying suitable optimization methods to identify the optimal weight and bias parameters.

In this chapter, we describe the mathematical structure of neural networks and the common activation functions used in their construction. We also discuss popular

[1]The expressivity of a neural network is its ability to represent complex, nonlinear functions. It is determined by architectural parameters such as depth, width, and activation functions.

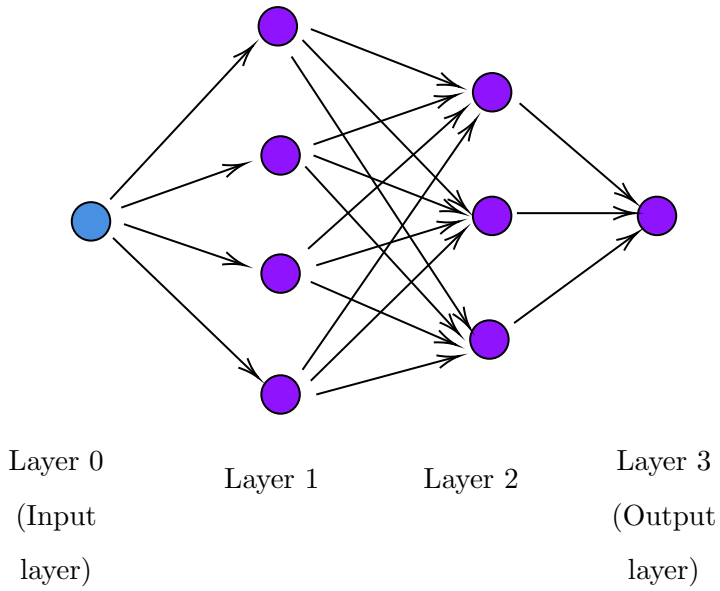


Figure 2.1: Diagram of a three-layer neural network.

optimization algorithms used for training. Furthermore, we review the use of neural networks for solving PDEs, specifically focusing on physics-informed neural networks (PINNs), Variational PINNs (VPINNs), and Robust VPINNs (RVPINNs). Finally, we present theoretical results regarding the analysis of loss functions and error estimation.

2.2 Neural networks

Neural networks are constructed from the composition of an input layer, several hidden layers, and an output layer. Each hidden layer consists of a linear transformation followed by a nonlinear activation function. Let W^i and $\mathbf{b}^i$ denote the weight matrix and the bias vector of the $i-$th layer, respectively. Furthermore, let $\sigma^i : \mathbb{R} \to \mathbb{R}$ be the activation function associated with the $i-$th layer. The component-wise extension $\boldsymbol{\sigma}^i : \mathbb{R}^n \to \mathbb{R}^n$ is defined by:

$$\boldsymbol{\sigma}^i(\mathbf{x}) = \boldsymbol{\sigma}^i(x_1, \ldots, x_n) = (\sigma^i(x_1), \ldots, \sigma^i(x_n)).$$

The $i-$th layer of the neural network can be expressed as:

$$\phi^i(\mathbf{x}) := \boldsymbol{\sigma}^i(W^i\mathbf{x} + \mathbf{b}^i). \tag{2.1}$$

The number of neurons in the i-th layer corresponds to the dimension of the output of that layer. In the output layer, we may use the same structure as the hidden layers or omit the activation function, depending on the specific problem.

An $L-$layer neural network consists of one input layer, $L-1$ hidden layers, and an output layer. So an $L-$layer neural network is written as the composition:

$$u_\theta(\mathbf{x}) = \phi^L \circ \phi^{L-1} \circ \cdots \circ \phi^1(\mathbf{x}) \tag{2.2}$$

$$= \boldsymbol{\sigma}^L(W^L \ldots \boldsymbol{\sigma}^2(W^2\boldsymbol{\sigma}^1(W^1\mathbf{x} + \mathbf{b}^1) + \mathbf{b}^2) \ldots + \mathbf{b}^L), \tag{2.3}$$

where $\theta = \{W^1, \mathbf{b}^1, \ldots, W^L, \mathbf{b}^L\}$ represents the set of all trainable parameters. We assume that the dimension of the matrices and vectors in Equation (2.3) are compatible for matrix multiplications and vector addition. As an example, a three-layer neural network is shown in Figure 2.1 and is given by:

$$u_\theta(\mathbf{x}) = \boldsymbol{\sigma}^3(W^3\boldsymbol{\sigma}^2(W^2\boldsymbol{\sigma}^1(W^1\mathbf{x} + \mathbf{b}^1) + \mathbf{b}^2) + \mathbf{b}^3). \tag{2.4}$$

Neural networks are effective as approximators because they can represent any continuous function on a compact subset of $\mathbb{R}^n$, provided appropriate activation functions are used. This result is known as the universal approximation theorem [54, 65]. For a more comprehensive discussion on the foundation of deep learning, we refer the reader to [52].

2.3 Activation functions

As discussed in the previous section, activation functions are essential components of neural networks that introduce nonlinearity into the model. Without these nonlinear mappings, a multi-layer neural network would simply reduce to a single linear transformation. In this section, we provide examples of widely used activation functions [49], including the ReLU, sigmoid, and softplus functions.

A primary choice in modern deep learning is the rectified linear unit (ReLU). The ReLU function $\mathrm{ReLU} : \mathbb{R} \to \mathbb{R}$ is defined as:

$$\mathrm{ReLU}(x) := \max(0, x) = \begin{cases} 0 & \text{if } x \leq 0, \\ x & \text{if } x > 0. \end{cases}$$

More generally, we can define the power-ReLU function ReLU^n for $n = 0, 1, 2, \ldots$ as:

$$\mathrm{ReLU}^n(x) := \begin{cases} 0 & \text{if } x \leq 0, \\ x^n & \text{if } x > 0. \end{cases}$$

Note that ReLU^0 corresponds to a Heaviside step function. A smooth version of this step function is the sigmoid function $\sigma : \mathbb{R} \to \mathbb{R}$ defined by:

$$\sigma(x) := \frac{1}{1 + e^{-x}}.$$

The sigmoid function is differentiable everywhere, with the derivative given by

$$\sigma'(x) = \sigma(x)(1 - \sigma(x)).$$

Another important smooth activation function is the softplus function, defined as:

$$\mathrm{Softplus}(x) := \ln(1 + e^x).$$

Notably, the derivative of the softplus function is exactly the sigmoid function. Other commonly used activation functions in scientific computing include the hyperbolic tangent (tanh) and trigonometric functions.

2.4 Optimization methods

To determine the optimal parameters for a neural network, we must solve an optimization problem. This involves finding a set of parameters θ that minimize a specific objective function, often referred to as a cost function or loss function.

Suppose we have a training set of N points $X = \{x^{(i)}\}_{i=1}^N \subset \mathbb{R}^d$, with corresponding target values $Y = \{y(x^{(i)})\}_{i=1}^N$. A standard choice for the loss function is the mean squared error (MSE), which measures the average squared difference between the network's prediction $u_\theta(x^{(i)})$ and the exact value $y(x^{(i)})$:

$$\mathcal{L}(u_\theta; X) = \frac{1}{N} \sum_{i=1}^{N} \left| u_\theta(x^{(i)}) - y(x^{(i)}) \right|^2. \tag{2.5}$$

Other commonly used loss functions include the mean absolute error and cross-entropy loss.

These optimization problems are typically solved using gradient-based iterative methods. The required gradients are efficiently computed via automatic differentiation [6], finite difference method [89], and other related techniques. Below, we briefly review the optimization algorithms used in this work.

2.4.1 Gradient descent method

The gradient descent (GD) method [24] is an iterative procedure that updates the parameters θ in the direction of the steepest descent of the loss function $\mathcal{L}$. The update rule is given by

$$\theta_{k+1} = \theta_k - \eta \nabla_\theta \mathcal{L}(u_{\theta_k}; X),$$

where $\eta > 0$ denotes the learning rate (or step size). The process continues until the gradient vanishes, converging to a local minimum of $\mathcal{L}$.

2.4.2 Stochastic gradient descent method

Stochastic gradient descent (SGD) [52] is a widely used extension of gradient descent, particularly suitable for large datasets. Instead of using the entire dataset $X = \{x^{(i)}\}_{i=1}^N$ to compute the gradient, SGD randomly selects a small subset (a mini-batch) $\tilde{X} = \{x^{(i_k)}\}_{k=1}^m \subset X$ of size $m \ll N$ at each iteration. The parameters are then updated as:

$$\theta_{k+1} = \theta_k - \eta \nabla_\theta \mathcal{L}(u_{\theta_k}; \tilde{X}),$$

where $\eta > 0$ is the learning rate. This reduces the computational cost per iteration to $O(m)$, making it faster for deep learning applications.

2.4.3 Adam

The adaptive moment estimation (Adam) optimizer [60] is an extension of SGD that adaptively adjusts learning rates for each parameter. It utilizes estimates of both the first moment and the second moment of the gradients. Due to advantages of momentum and root mean square propagation techniques, Adam typically converges faster and exhibits more stability than standard SGD, making it one of the most popular choices for training neural networks.

2.4.4 BFGS method

The Broyden-Fletcher-Goldfarb-Shanno (BFGS) method [106] is an iterative algorithm used for solving unconstrained nonlinear optimization problems. Unlike first-order

methods (GD and SGD), BFGS approximates the inverse Hessian matrix to incorporate second-order curvature information without the heavy cost of computing the true Hessian or its inverse, leading to a computational complexity of $O(n^2)$. In practice, the limited-memory BFGS (L-BFGS) variant is also applied for neural networks as it reduces memory requirements. These methods often achieve superlinear convergence [106], making them efficient for optimization problems.

2.5 Neural networks in scientific computing

The application of neural networks to scientific computing has gained significant traction, with numerous studies exploring their potential to solve mathematical problems. To provide context of our work, we review key methodologies for solving PDEs using neural networks, as these provide the foundation for our operator-based approaches.

2.5.1 PINNs

A prominent example of applying neural networks into PDEs is the physics-informed neural network (PINN) [84]. This is typically used to solve PDEs of the form:

$$\mathcal{G}u(x) = f(x), \quad x \in \Omega \subset \mathbb{R}^n \tag{2.6}$$

$$u(x) = h(x), \quad x \in \partial\Omega \tag{2.7}$$

where $\mathcal{G}$ is a term containing differential operators and u is the unknown solution. In this framework, the solution u is approximated by a neural network u_θ. The parameters θ are learned by minimizing a loss function that penalizes the residual of the PDE and the boundary conditions:

$$\mathcal{L}_{\text{PINNs}}(u_\theta) := \|\mathcal{G}u_\theta - f\|^2_{H(\Omega)} + \lambda \int_{\partial\Omega} (u_\theta - h)^2 ds,$$

where $\lambda > 0$ is a penalty parameter used to balance the two terms, and $\mathcal{G}u_\theta - f$ is in some Hilbert space $H(\Omega)$. The first term is usually evaluated at a set of collocation points within the domain Ω, while the second term ensures the boundary conditions are satisfied.

2.5.2 VPINNs

An alternative approach is the variational physics-informed neural network (VPINN) [58]. Instead of minimizing the strong-form residual, this method solves PDEs in their variational (weak) form. This is achieved by multiplying Equation (2.6) by test functions v_k, for $k = 1, 2, \ldots, K$, and integrating over the domain. The discrete variational problem is stated as:

$$\begin{aligned} (\mathcal{G}u, v_k)_{L^2(\Omega)} &= (f, v_k)_{L^2(\Omega)}, \quad k = 1, 2, \ldots, K \\ u(x) &= h(x), \quad x \in \partial\Omega \end{aligned}$$

where $(\cdot, \cdot)_{L^2(\Omega)}$ denotes the L^2 inner product. The loss function of VPINNs is then defined based on the variational residual:

$$\mathcal{L}_{\text{VPINNs}}(u_\theta) := \frac{1}{K} \sum_{k=1}^{K} | \left(\mathcal{G}u_\theta, v_k\right)_{L^2(\Omega)} - \left(f, v_k\right)_{L^2(\Omega)} |^2 + \lambda \int_{\partial\Omega} (u_\theta - h)^2 ds.$$

The approximate solution u_{θ^*} is obtained by identifying the parameter set θ^* that best satisfies the variational constraints. This is formulated as:

$$\theta^* = \underset{\theta}{\operatorname{argmin}}\, \mathcal{L}_{\text{VPINNs}}(\theta).$$

2.5.3 RVPINNs

The robust variational physics-informed neural network (RVPINN) [87] is a framework designed to overcome the stability issues sometimes found in standard VPINNs. It is formulated to solve operator equations, such as the Poisson problem or transport equations, by treating the loss function as a residual-based error estimator.

In this approach, we seek to approximate the solution u with a neural network $u_{\boldsymbol{\theta}}$ by solving the variational problem:

$$r(u_\theta, v) = l(v) - b(u_\theta, v) = 0, \quad \forall v \in V,$$

where $b(\cdot, \cdot)$ is a bilinear form and $l(\cdot)$ is a linear functional. The robustness comes from the definition of the loss function, which is defined as the dual norm of the residual:

$$\mathcal{L}_{\text{RVPINNs}}(u_\theta) := \|r(u_\theta, \cdot)\|_{V'}^2 = \left(\sup_{0 \neq v \in V} \frac{r(u_\theta, v)}{\|v\|_V} \right)^2.$$

In practice, this supremum is computed over a discrete test space $V_k \subset V$. By using the Riesz representation theorem, the loss can be calculated as

$$\mathcal{L}_{\text{RVPINNs}}(u_\theta) = \|\phi_k(u_\theta)\|_V^2,$$

where $\phi_k(u_\theta)$ is the discrete Riesz representative of the residual in the test space. Details on deriving this formulation, along with the loss function including neural network constraints (see Equation (6.12)), are presented in Section 6.2.2. This structure ensures that the loss function is equivalent to the true error, up to the remainder term, meaning the training process is naturally guided toward the exact solution with high stability.

2.5.4 Deep Ritz

The last example we will explain in detail here is the Deep Ritz Method [107], which is an algorithm for solving PDEs in a variational form. As an example, the variational form of the problem (2.6)-(2.7) with $\mathcal{G}u = \Delta u$ and $h = 0$ is the following:

$$\min_{u \in H} \int_\Omega \left(\frac{1}{2} |\nabla u(x)|^2 - f(x)u(x) \right) dx$$
$$u(x) = 0, \quad x \in \partial\Omega,$$

where H is the set of trial functions. To solve this problem, the Deep Ritz Method approximates a function u by a neural network u_θ with the ReLU^3 activation function. The loss function is defined by

$$\mathcal{L}_{\text{Ritz}}(u_\theta) := \int_\Omega \left(\frac{1}{2} |\nabla u_\theta(x)|^2 - f(x)u_\theta(x) \right) dx + \lambda \int_{\partial\Omega} (u_\theta)^2 ds$$

and the SGD is used to find the optimal parameter θ^* that minimizes $\mathcal{L}_{\text{Ritz}}(u_\theta)$.

There are other applications of neural networks such as Integrable Deep Neural Networks used for predicting the free energy from derivative data [99], the r-adaptive neural network method to solve PDEs [80], a neural network used as a control variable to solve PDEs in the weak form [17], neural networks for inverse problems for PDEs [7], etc.

2.6 Analysis of loss functions

We denote $\mathcal{M}_n$ as the set of real-valued neural networks with n parameters, based on the architecture described in (2.3):

$$\mathcal{M}_n := \{u_\theta : \mathbb{R}^d \to \mathbb{R} \mid \theta \in \mathbb{R}^n\}. \tag{2.8}$$

It is known that $\mathcal{M}_n$ is neither a linear, closed, nor convex subset of $L^2(\Omega)$ [83]. Although we have an infimum of a loss function $\mathcal{L}$ on $\mathcal{M}_n$, the minimizer in $\mathcal{M}_n$ may not exist [17]. To study numerical analysis related to neural networks, it is worth mentioning a relaxed definition of minimizer, called a quasi-minimizer (cf. [17]).

Definition 2.6.1. Let $\mathcal{L} : U \to \mathbb{R}$ be a loss function. Let $\delta_n > 0$ and $\mathcal{M}_n \subset U$. A function $u_{\theta^*} \in \mathcal{M}_n$ is said to be a *quasi-minimizer* of $\mathcal{L}$ if

$$\mathcal{L}(u_{\theta^*}) \leq \inf_{u_\theta \in \mathcal{M}_n} \mathcal{L}(u_\theta) + \delta_n. \tag{2.9}$$

Note that a quasi-minimizer always exists when the infimum exists, e.g., when the loss function is bounded from below. The following theorem (cf. [17, Theorem 2.A]) provides the error estimate for some loss functions. Before that, let us give a definition of a Gâteaux derivative.

Definition 2.6.2. Let X and Y be Banach spaces and let $j : X \to Y$. A function j is said to be *Gâteaux differentiable* if there is a bounded linear operator $j'_u : X \to Y$ such that

$$j'_u(v) = \lim_{s \to 0} \frac{j(u + sv) - j(u)}{s}.$$

The operator j'_u is called the *Gâteaux derivative* of j at u.

Theorem 2.6.1. *Let $\mathcal{L} : X \to \mathbb{R}$ be a loss function. Assume that $\mathcal{L}$ is Gâteaux differentiable with the derivative $\mathcal{L}' : X \to X^*$ being Lipschitz continuous, i.e., there is a constant $C > 0$ such that*

$$\|\mathcal{L}'_u - \mathcal{L}'_v\|_{X^*} \leq C\|u - v\|_X, \quad \forall u, v \in X.$$

Furthermore, assume that $\mathcal{L}$ is strongly convex, i.e., there is a constant $\gamma > 0$ such that

$$\langle \mathcal{L}'_u - \mathcal{L}'_v, u - v \rangle_{X^*,X} \geq \gamma \|u - v\|_X^2, \quad \forall u, v \in X.$$

Then, the following hold true:

1. *$\mathcal{L}$ has a unique minimizer $\bar{u} \in X$, which satisfies: $\mathcal{L}'_{\bar{u}} = 0$ in X^**

2. *For any subset $\mathcal{M}_n \subset X$, $\mathcal{L}$ has a quasi-minimizer $\bar{u}_n \in \mathcal{M}_n$ that satisfies (2.9).*

3. *Any quasi-minimizer $\bar{u}_n$ in $\mathcal{M}_n$ satisfies the following quasi-optimal error estimate:*

$$\|\bar{u} - \bar{u}_n\|_X \leq \left(\frac{C}{\gamma} \inf_{u_n \in \mathcal{M}_n} \|\bar{u} - u_n\|_X^2 + \frac{2\delta_n}{\gamma} \right)^{1/2}.$$

Remark 2.6.1. Theorem 2.6.1 holds (cf. [17]) for the loss functions of the PINNs and Deep Ritz method.

Chapter 3

Perron-Frobenius Operators and Related Problems

3.1 Introduction

Perron-Frobenius operators, also known as transfer operators, play an important role in the study of dynamical systems, impacting not only pure mathematical problems but also real-world applications. A prominent example is the engineering challenge of tracking vibrational energy within a car body [86]. In this context, ray-tracing methods are employed to model energy propagation through complex domains. The energy movement is represented by rays, and the evolution of their densities is described by the Perron-Frobenius operator.

Numerical methods are required to solve equations involving Perron-Frobenius operators, as these problems are often too challenging to solve analytically. A classical approach is *Ulam's method* [101], which approximates the Perron-Frobenius operator using piecewise-constant functions. Specifically, Ulam's method is a *Galerkin method* that projects the operator onto a subspace spanned by characteristic functions on a partition of the domain. Another numerical approach, proposed by Ding [37], is the *maximum entropy method* (MEM). This is an optimization method based on the concept of entropy. This method was originally studied to approximate an invariant density for one dimensional spaces. Then it has recently been developed for higher-dimensional systems [56]. Many studies have focused on improving the accuracy of MEM and addressing other operator-related problems (see [11, 38, 39]).

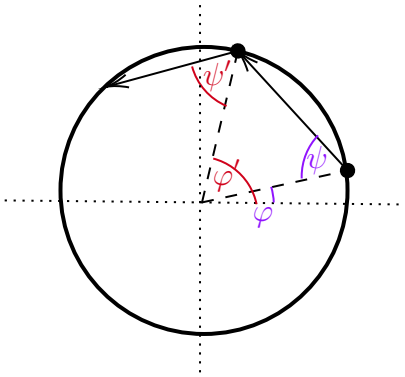


Figure 3.1: Ray trajectory in circular domain.

In this chapter, we first illustrate how ray-tracing in a circular domain leads to the Perron-Frobenius operator. We then establish an abstract setting for these operators and define two problems that we aim to solve: the Neumann series problem and the invariant density problem. We present examples of one-dimensional (1D) and two-dimensional (2D) dynamical systems, along with their steady states and invariant densities. These examples will serve as test cases for the numerical experiments for the methods developed in later chapters. Finally, we review classical numerical methods for solving these problems, including Ulam's method, Galerkin and Petrov-Galerkin methods, the maximum entropy method, and the minimum energy method.

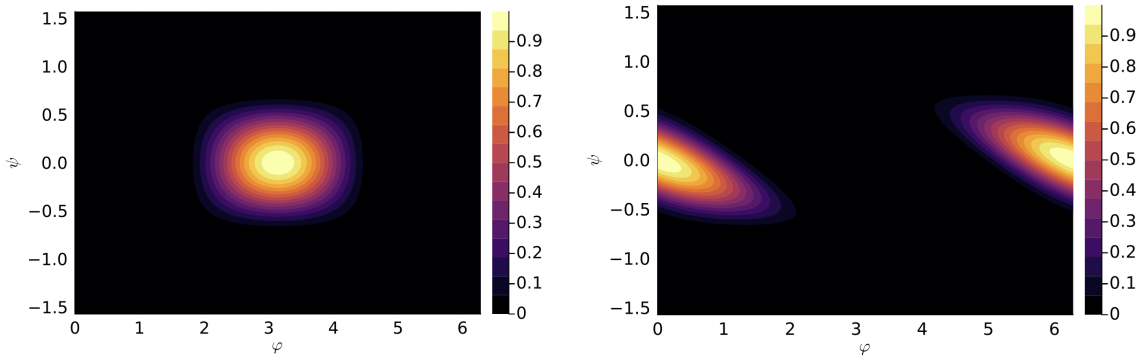
3.2 From ray tracing to Perron-Frobenius operator

To give an example of Perron-Frobenius operators, we consider a ray trajectory in a circular domain. The positions of rays can be described by the coordinate (φ, ψ), as shown in Figure 3.1, where

- $\varphi \in [0, 2\pi)$ is the polar angle, describing the origin point of the ray on the boundary,
- $\psi \in (-\frac{\pi}{2}, \frac{\pi}{2})$ is the angle between the normal line of the circle and the ray, describing the ray direction.

The phase space is given by $\Omega := [0, 2\pi) \times (-\frac{\pi}{2}, \frac{\pi}{2})$ and the phase map $S : \Omega \to \Omega$ describing the receiving coordinates from a ray trajectory leaving (φ, ψ) is given by

$$S(\varphi, \psi) = (\varphi + \pi - 2\psi \mod 2\pi, \psi) =: (S_1(\varphi, \psi), S_2(\varphi, \psi)).$$



(a) The initial density f_0

(b) The density $\mathcal{P}f_0$.

Figure 3.2: Evolution of a ray density in a circular domain.

This map includes the ray reflection under the specular law of reflection, the angle of reflection is equal to the angle of incidence.

Given $(\varphi_0, \psi_0) \in \Omega$, a sequence $(\varphi_0, \psi_0), S(\varphi_0, \psi_0), S^2(\varphi_0, \psi_0), \ldots$ is called a *ray trajectory* starting at (φ_0, ψ_0). To study several ray trajectories at the same time, it is convenient to study them via their distribution, called a density. A *ray density* is a function $f : \Omega \to [0, \infty]$, typically belonging to the space $L^1(\Omega)$, which describes the distribution of trajectories over the domain Ω.

The density f_0 for a single ray at (φ_0, ψ_0) is defined by

$$f_0(\varphi, \psi) = \delta[\varphi - \varphi_0] \cdot \delta[\psi - \psi_0],$$

where δ is the Dirac delta distribution. To study the evolution of the density f_0 under the map S, we can see that after applying the map S, the distribution f_0 evolves to

$$\begin{aligned} f_1(\varphi, \psi) &= \delta[\varphi - S_1(\varphi_0, \psi_0)] \cdot \delta[\psi - S_2(\varphi_0, \psi_0)] \\ &= \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \int_0^{2\pi} \delta[\varphi - S_1(\varphi', \psi')] \cdot \delta[\psi - S_2(\varphi', \psi')] \cdot \delta[\varphi' - \varphi_0] \cdot \delta[\psi' - \psi_0] d\varphi' d\psi'. \end{aligned}$$

In general, if f_n is the density at the n^{th} step of applying S, the density at the $(n+1)^{th}$ step is given by

$$f_{n+1}(\varphi, \psi) = \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \int_0^{2\pi} \delta[\varphi - S_1(\varphi', \psi')] \cdot \delta[\psi - S_2(\varphi', \psi')] \cdot f_n(\varphi', \psi') d\varphi' d\psi'.$$

The operator $\mathcal{P}$ mapping $f_n \mapsto f_{n+1}$ given by the above equation is the so-called *Perron-Frobenius operator*, i.e.,

$$(\mathcal{P}f)(\varphi, \psi) = \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \int_0^{2\pi} \delta[\varphi - S_1(\varphi', \psi')] \cdot \delta[\psi - S_2(\varphi', \psi')] \cdot f(\varphi', \psi') d\varphi' d\psi'. \quad (3.1)$$

Figure 3.2 shows an example of the initial density

$$f_0(\varphi,\psi)=\begin{cases}\cos^2(\varphi)\cos^2(2\psi), & \pi/2<\varphi<3\pi/2 \text{ and } -\pi/4<\psi<\pi/4,\\ 0, & \text{elsewhere,}\end{cases} \tag{3.2}$$

and the density after mapping by the operator $\mathcal{P}$. Note that Perron-Frobenius operators can be defined in a similar manner in other domains [96].

3.3 Perron-Frobenius operators

After introducing the Perron-Frobenius operator for the boundary map on a circular domain, we are now ready to provide a general definition of Perron-Frobenius operators. This generalization requires some background in measure theory. For the necessary results concerning measure theory and Lebesgue integration, the reader is referred to Appendix A.1 and A.2.

Let Ω be a bounded subset of $\mathbb{R}^d, d\geq 1$. Let $(\Omega,\mathfrak{B},\mu)$ denote a measure space where $\mathfrak{B}$ is a Borel sigma-algebra and μ is the Lebesgue measure. For any $f\in L^1(\Omega)$, we define a measure μ_f with respect to μ by

$$\mu_f(A)=\int_A f d\mu,\quad A\in\mathfrak{B}.$$

Let $S:\Omega\to\Omega$ be a map describing a discrete-time dynamical system:

$$x_{n+1}=S(x_n).$$

Suppose that S is nonsingular with respect to μ, i.e., $\mu(S^{-1}(A))=0$ for all $A\in\mathfrak{B}$ with $\mu(A)=0$. We also define a measure ν_f satisfying

$$\nu_f(A)=\int_{S^{-1}(A)} f d\mu,\quad A\in\mathfrak{B}.$$

Since S is nonsingular with respect to μ, $\mu(A)=0$ implies $\mu(S^{-1}(A))=0$, and hence $\nu_f(A)=0$. So ν_f is absolutely continuous with respect to μ. By Radon-Nikodym theorem, there exists a unique function $g\in L^1(\Omega)$ such that

$$\nu_f(A)=\int_A g d\mu,\quad A\in\mathfrak{B}.$$

By extending the mapping $f\mapsto g$, the Perron-Frobenius operator $\mathcal{P}$ is defined by the operator $\mathcal{P}:L^1(\Omega)\to L^1(\Omega)$ such that

$$\int_A \mathcal{P}f d\mu=\int_{S^{-1}(A)} f d\mu,\quad A\in\mathfrak{B}. \tag{3.3}$$

It is well-known that the Perron-Frobenius operator $\mathcal{P}$ is linear[1], positive[2] and non-expansive[3].

In the one-dimensional case, where Ω is the closed interval $[a, b]$ and $A = [a, x]$ for $a \leq x \leq b$, Equation (3.3) becomes

$$\int_a^x \mathcal{P}f(s)ds = \int_{S^{-1}([a,x])} f(s)ds. \tag{3.4}$$

Differentiating both sides we obtain an explicit form of the Perron-Frobenius operator:

$$\mathcal{P}f(x) = \frac{d}{dx}\int_{S^{-1}([a,x])} f(s)ds. \tag{3.5}$$

In general, if S is invertible and S^{-1} is nonsingular, then we have

$$\mathcal{P}f(x) = f(S^{-1}(x))|J_{S^{-1}}|, \tag{3.6}$$

where $|J_{S^{-1}}|$ is the determinant of the Jacobian matrix of S^{-1} [64].

We note that the Perron-Frobenius operator in Equation (3.1) satisfies Equation (3.3). Indeed, through the change of variable $(z_1, z_2) := S(\varphi', \psi')$, we obtain

$$\begin{aligned}
\int_A (\mathcal{P}f)(\varphi, \psi)d\mu &= \int_A \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \int_0^{2\pi} \delta[\varphi - S_1(\varphi', \psi')]\delta[\psi - S_2(\varphi', \psi')]f(\varphi', \psi')d\varphi' d\psi' d\mu \\
&= \int_A \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \int_0^{2\pi} \delta[\varphi - z_1]\delta[\psi - z_2]f(S^{-1}(z_1, z_2))|J_{S^{-1}}(z_1, z_2)|dz_1 dz_2 d\mu \\
&= \int_{S^{-1}(A)} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \int_0^{2\pi} \delta[\varphi - z_1]\delta[\psi - z_2]f(z_1, z_2)dz_1 dz_2 d\mu \\
&= \int_{S^{-1}(A)} f(\varphi, \psi)d\mu.
\end{aligned}$$

Before discussing the Neumann series and invariant density problems in the following sections, we introduce the Koopman operator, as its properties are essential to our methods. For a given map S, the Koopman operator $\mathcal{K} : L^\infty(\Omega) \to L^\infty(\Omega)$ is defined by:

$$\mathcal{K}g := g \circ S.$$

The Koopman operator $\mathcal{K}$ is the adjoint of the Perron-Frobenius operator $\mathcal{P}$ in the following sense:

$$\int_\Omega (\mathcal{P}f) \cdot g d\mu = \int_\Omega f \cdot g(S)d\mu \tag{3.7}$$

[1] $\mathcal{P}(\lambda_1 f_1 + \lambda_2 f_2) = \lambda_1 \mathcal{P}f_1 + \lambda_2 \mathcal{P}f_2$ for $\lambda_1, \lambda_2 \in \mathbb{R}$ and $f_1, f_2 \in L^1(\Omega)$.

[2] $\mathcal{P}f \geq 0$ if $f \geq 0$.

[3] $\|\mathcal{P}f\|_{L^1} \leq \|f\|_{L^1}$.

for any $f \in L^1(\Omega)$ and $g \in L^\infty(\Omega)$. While $\mathcal{P}$ describes the evolution of densities, $\mathcal{K}$ describes the evolution of observable functions. For more detailed properties of Perron-Frobenius and Koopman operators, we refer the reader to [61, 64, 90].

3.4 Neumann series problems

Since the evolution of densities can be described by Perron-Frobenius operators, problems related to ray densities can be formulated as equations involving these operators. Assume that an initial density f_0 represents the proportion of energy emitted from a source on the boundary of the domain. If the source emits energy continuously and the energy is reduced by a factor $\alpha \in (0, 1)$ after each collision, the accumulated densities will converge to a steady state. This steady state can be expressed as a series involving the Perron-Frobenius operator:

$$u := f_0 + \alpha\mathcal{P}f_0 + \alpha^2\mathcal{P}^2 f_0 + \dots \tag{3.8}$$

This is known as the *Neumann series* of Perron-Frobenius operator. Finding the total density u in (3.8) is equivalent to solving the following equation:

$$u - \alpha\mathcal{P}u = f_0.$$

In practice, we can approximate u by using a truncated sum:

$$u_N := f_0 + \alpha\mathcal{P}f_0 + \alpha^2\mathcal{P}^2 f_0 + \dots + \alpha^N\mathcal{P}^N f_0,$$

where N is chosen to be sufficiently large.

Remark 3.4.1. To incorporate more realistic energy losses, the scalar damping parameter α may be replaced by a weight function $\alpha : \Omega \to [0, \infty)$ that varies along individual trajectories.

In the following, we present several examples of 1D and 2D dynamical systems along with their corresponding Perron-Frobenius operators. Examples of 1D systems include the map derived from cuboid cavities and the tent map. For 2D systems, we include the boundary map in a circular domain discussed previously, as well as the standard map, which is a well-known dynamical system with coexisting chaotic and

integrable dynamics. We will use these examples in Chapter 4, where we develop neural network methods to solve Neumann series problems.

Example 3.1: Cuboid cavities

Consider a 3D cuboid cavity of unit length. Let ρ_0 be the source density emitted perpendicularly from the left wall at $x = 0$. The ray density propagating from the left boundary ($x = 0$) toward the right boundary ($x = 1$) is given by:

$$f_+^n(x) = e^{-\beta(2n+x)}\rho_0, \quad n = 0, 1, 2, \ldots. \tag{3.9}$$

After reflecting from the right wall, the density traveling back towards $x = 0$ is described by:

$$f_-^n(x) = e^{-\beta(2n-x)}\rho_0, \quad n = 1, 2, \ldots, \tag{3.10}$$

where n represents the number of reflections at the opposite sides and $\beta > 0$ denotes the dissipation rate [5]. For a single round-trip through the cavity, the resulting spatial density is:

$$f_0(x) = (e^{-\beta x} + e^{-\beta(2-x)})\rho_0, \quad 0 < x < 1.$$

Let ρ_n denote the ray density at the left wall after n reflections at the right wall. The discrete dynamical system describing ray densities at the left wall is governed by $\rho_{n+1} = e^{-2\beta}\rho_n$. Thus, the Perron-Frobenius operator for this system is defined as $\mathcal{P}\rho_n = \rho_n$ and $\alpha = e^{-2\beta}$. This leads to a spatial density at the n^{th} time step:

$$f_n(x) = (e^{-\beta x} + e^{-\beta(2-x)})\rho_n, \quad 0 < x < 1.$$

The steady state at the left wall can be directly evaluated using a geometric series:

$$\rho_\infty = \sum_{n=0}^{\infty} \rho_n = \rho_0 + e^{-2\beta}\rho_0 + e^{-4\beta}\rho_0 + \ldots = \frac{\rho_0}{1 - e^{-2\beta}}.$$

Therefore, the final spatial steady state is given by:

$$f_\infty(x) = \frac{\rho_0}{1 - e^{-2\beta}}(e^{-\beta x} + e^{-\beta(2-x)}). \tag{3.11}$$

Example 3.2: Tent map

The tent map [64] is a classic example of a continuous, piecewise-linear function that exhibits chaotic dynamics. The tent map $S : [0, 1] \to [0, 1]$ is defined as:

$$S(x) = \begin{cases} 2x, & x \in [0, \frac{1}{2}), \\ 2 - 2x, & x \in [\frac{1}{2}, 1]. \end{cases}$$

By applying the general formulation of the Perron-Frobenius operator on closed interval (3.5), the operator associated with the tent map can be derived as:

$$\mathcal{P}f(x) = \frac{1}{2}f\left(\frac{x}{2}\right) + \frac{1}{2}f\left(1 - \frac{x}{2}\right). \tag{3.12}$$

Example 3.3: Circular domains

Let the phase space be denoted by $\Omega := [0, 2\pi) \times (-\frac{\pi}{2}, \frac{\pi}{2})$. Recall that the phase map for a circular domain is the function $S : \Omega \to \Omega$ defined by:

$$S(\varphi, \psi) = (\varphi + \pi - 2\psi, \psi),$$

with its inverse map given by:

$$S^{-1}(\varphi, \psi) = (\varphi - \pi + 2\psi, \psi),$$

where the first component of both maps is computed modulo 2π. Note that the determinant of the Jacobian matrix of the inverse map is $|J_{S^{-1}}(\varphi, \psi)| = 1$. According to Equation (3.6), the Perron-Frobenius operator is defined as:

$$\mathcal{P}f(\varphi, \psi) = f(\varphi - \pi + 2\psi, \psi),$$

where the first component is understood to be modulo 2π.

Example 3.4: Standard map

The standard map [82] is a mathematical model used to study chaotic behavior in dynamical systems. It is a discrete-time dynamical system that exhibits a transition from regular to chaotic motion, making it a popular example in the theory of Hamiltonian chaos. The standard map S is a mapping of the phase space $[0, 2\pi) \times [0, 2\pi)$ onto itself, defined by:

$$S(\theta, p) = (\theta + p + K\sin(\theta), p + K\sin(\theta)),$$

and its inverse map is given by:

$$S^{-1}(\theta, p) = (\theta - p, p - K\sin(\theta - p)),$$

where each component in both maps is computed modulo 2π. The parameter K is the parameter that controls the amount of chaos in the system.

Note that the determinant of the Jacobian matrix satisfies $|J_{S^{-1}}(\theta, p)| = 1$. According to Equation (3.6), the Perron-Frobenius operator associated with the standard map is defined by:

$$\mathcal{P}f(\theta, p) = f(\theta - p, p - K\sin(\theta - p)),$$

where each component is computed modulo 2π.

Figures 3.3 and 3.4 show the evolution of accumulated densities for the circular boundary map and the standard map (with $K = 2.4$), respectively. The initial densities for the boundary map is defined in (3.2), while for the standard map, it is given by:

$$f_0(\theta, p) = \begin{cases} \cos^2(2\theta)\cos^2(2p), & 3\pi/4 < \theta < 5\pi/4 \text{ and } 3\pi/4 < p < 5\pi/4, \\ 0, & \text{elsewhere.} \end{cases}$$

Note that the initial densities in both cases belong to $L^2(\Omega)$ and the L^2-norm of the truncation error, $\|u - u_N\|_{L^2}$, is bounded by $\alpha^{N+1}\|f_0\|_{L^2}/(1-\alpha)$. As the dissipation factor α approaches 1, the steady state u (approximated here by the truncated sum u_{1000}), becomes more complex. This behavior is illustrated in Figure 3.5 for varying values of α.

3.5 Invariant density problems

While the Neumann series describes the accumulation of density, we are often interested in a distribution that remains unchanged as the system evolves over time. This leads us to the concept of an invariant density, which represents a fixed point of the Perron-Frobenius operator.

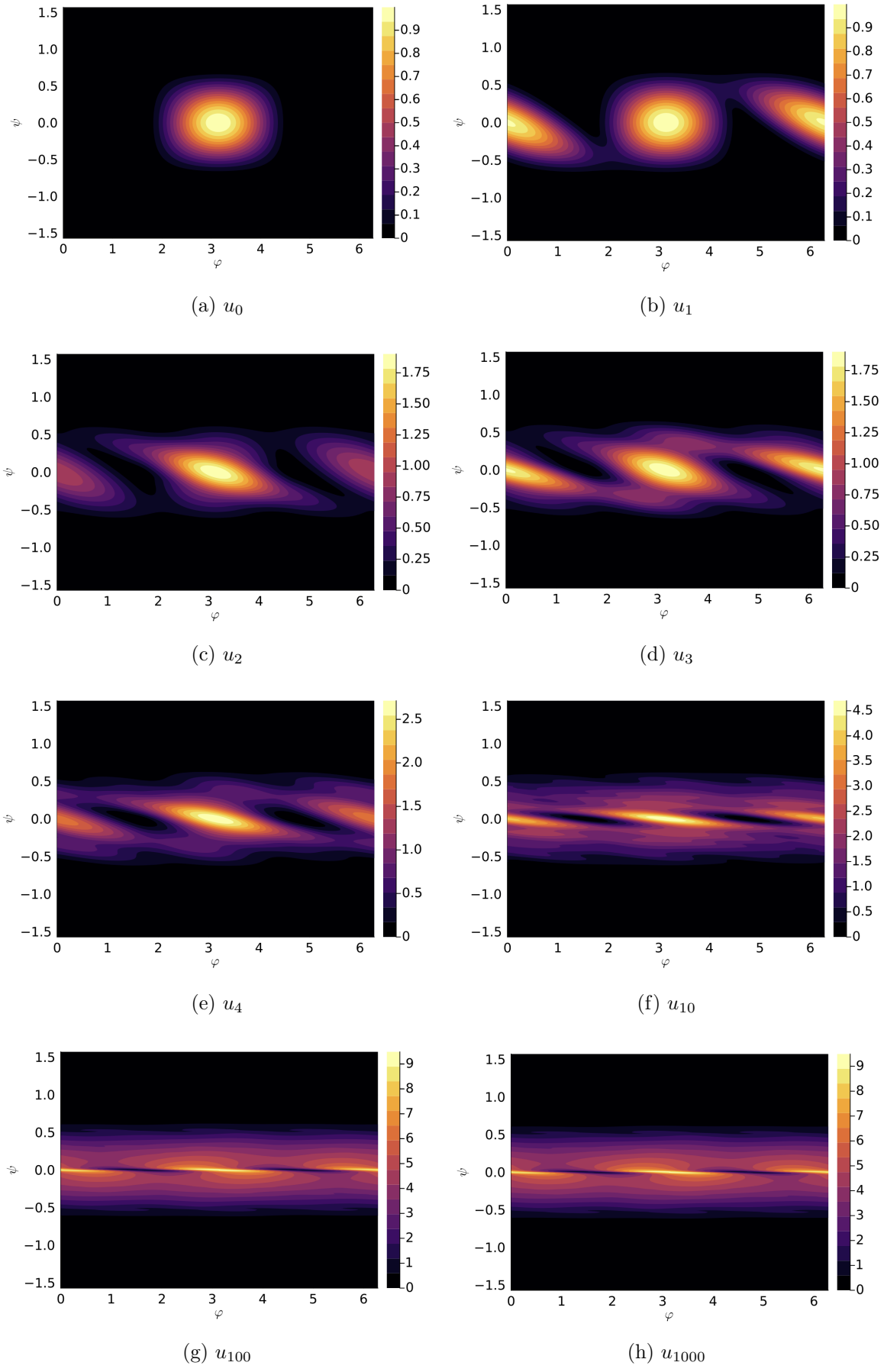
We define the set of all possible density functions as:

$$D := \{f \in L^1(\Omega) : f \geq 0 \text{ and } \|f\|_{L^1} = 1\}. \tag{3.13}$$

A function $f \in D$ is called an *invariant density* of the Perron-Frobenius operator $\mathcal{P}$ if it satisfies the fixed-point equation:

$$\mathcal{P}f = f.$$

Note that an invariant density is an eigenfunction of $\mathcal{P}$ corresponding to the eigenvalue of 1. The existence of such a density f implies the existence of an associated



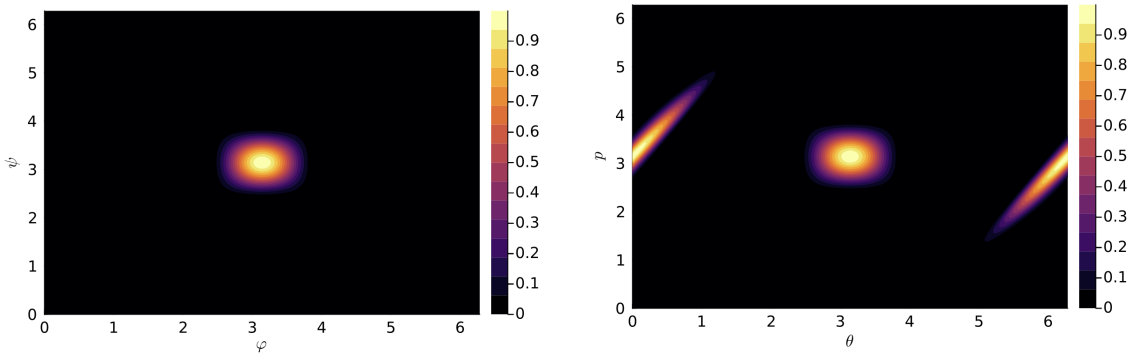
(a) u_0 (b) u_1

(c) u_2 (d) u_3

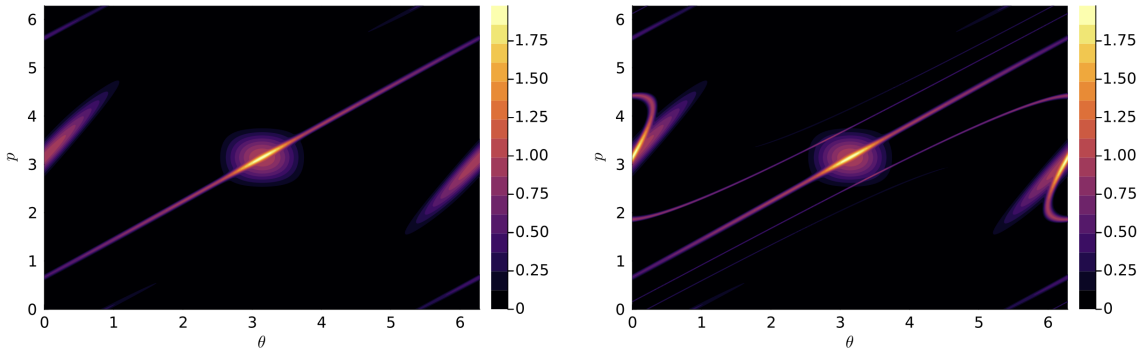
(e) u_4 (f) u_{10}

(g) u_{100} (h) u_{1000}

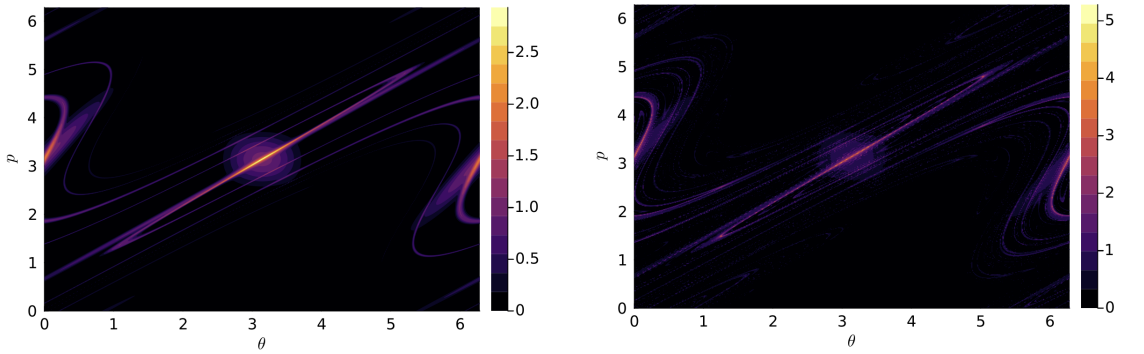
Figure 3.3: Accumulated densities in a circular domain with $\alpha = 0.95$.



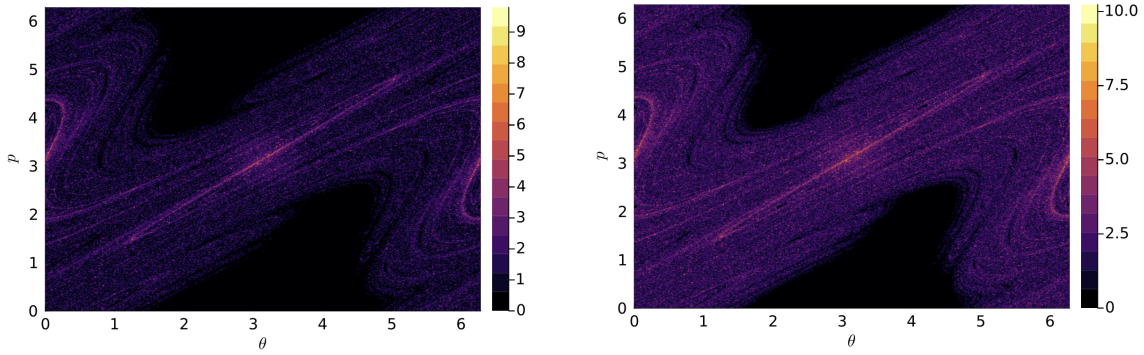
(a) u_0 (b) u_1



(c) u_2 (d) u_3

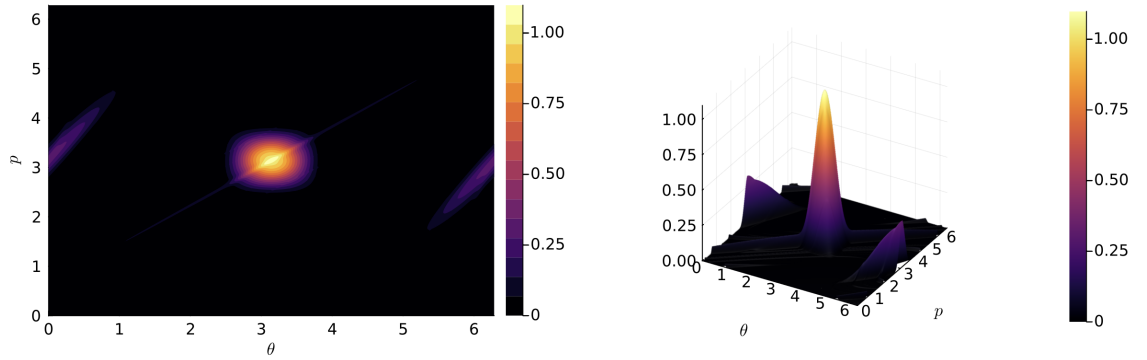


(e) u_4 (f) u_{10}

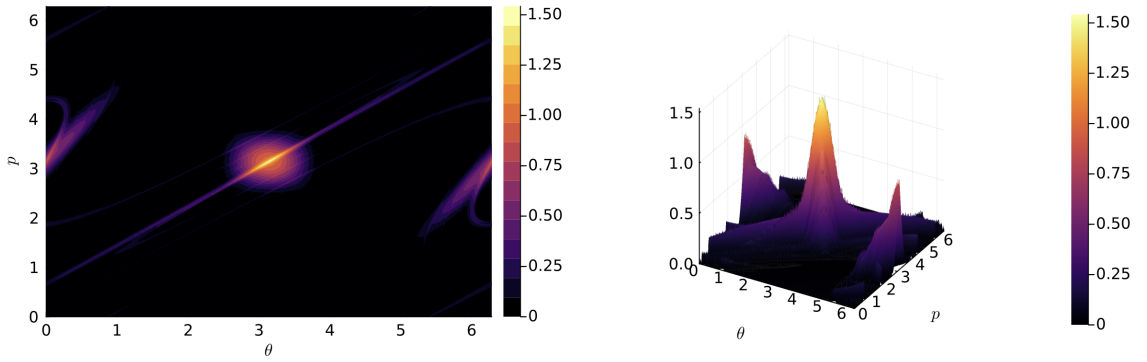


(g) u_{100} (h) u_{1000}

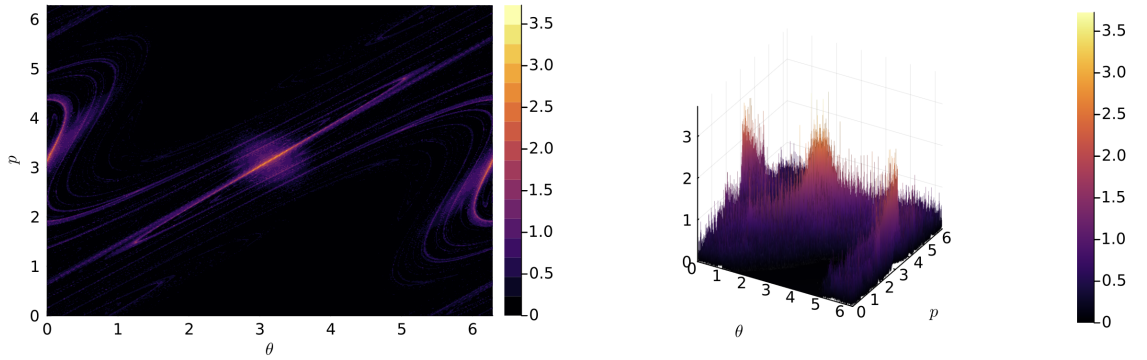
Figure 3.4: Accumulated densities of the standard map with $K = 2.4$ and $\alpha = 0.99$.



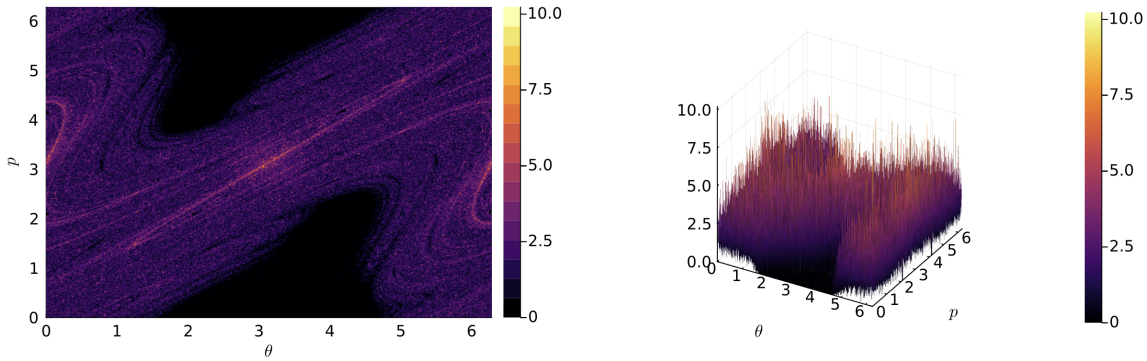
(a) $\alpha = 0.3$ (b) $\alpha = 0.3$



(c) $\alpha = 0.6$ (d) $\alpha = 0.6$



(e) $\alpha = 0.9$ (f) $\alpha = 0.9$



(g) $\alpha = 0.99$ (h) $\alpha = 0.99$

Figure 3.5: 2D (left) and 3D (right) steady states for the standard map with $K = 2.4$ varying values of α.

invariant measure μ_f defined by:

$$\mu_f(A) = \int_A f d\mu, \quad A \in \mathfrak{B}.$$

This measure is invariant under the map S in the sense that $\mu_f(S^{-1}(A)) = \mu(A)$ for all $A \in \mathfrak{B}$ [63, Theorem 4.1.1]. The concept of an invariant measure has applications in many fields of sciences (see [40, Chapter 9] for examples).

The existence of an invariant density $f \in D$ for a given dynamical system is fundamentally linked to the expansion and regularity properties of the transformation S. For a class of one-dimensional systems known as piecewise expanding maps, existence is established by the Lasota-Yorke Theorem [40, Theorem 5.2.1].

Theorem 3.5.1 (Lasota-Yorke Theorem)**.** *If a map $S : [0,1] \to [0,1]$ satisfies the following conditions:*

1. *There is a partition $0 = a_0 < a_1 < \ldots < a_r = 1$ of the interval $[0,1]$ such that for each $i = 1, 2, \ldots, r$ the restriction of S to the interval (a_{i-1}, a_i) is a C^2-function.*

2. *There is a constant $\lambda > 1$ such that $|S'(x)| \geq \lambda$ for $x \neq a_i, i = 0, 1, \ldots, r$.*

3. *There is a constant c such that $S''(x)/[S'(x)]^2 \leq c$ for $x \neq a_i, i = 0, 1, \ldots, r$.*

Then the corresponding Perron-Frobenius operator has at least one invariant density.

While the Lasota-Yorke Theorem guarantees existence, it does not ensure that the invariant density is unique. Uniqueness is a consequence of the ergodicity of the system. A map S is said to be *ergodic* if every set $A \in \mathfrak{B}$ such that $S^{-1}(A) = A$ has the property that either $\mu(A) = 0$ or $\mu(\Omega \backslash A) = 0$. If the map S is ergodic, the associated Perron-Frobenius operator has at most one invariant density [64, Theorem 4.2.2].

A stronger condition, *exactness*, also guarantees uniqueness. A piecewise expanding map is considered *exact* if it is topologically transitive, meaning that any subset of Ω with positive measure eventually expands to cover the entire domain under repeated iterations of S. In such cases, the system is indecomposable; it cannot be partitioned into multiple independent invariant subsystems. Consequently, the Perron-Frobenius operator $\mathcal{P}$ possesses a unique invariant density. The following theorem provides a uniqueness of invariant density [64, Theorem 6.2.2].

Examples	Phase maps	Invariant densities
1.1	$S_1(x) = \begin{cases} \frac{2x}{1-x^2}, & 0 \le x \le \sqrt{2}-1 \\ \frac{1-x^2}{2x}, & \sqrt{2}-1 \le x \le 1 \end{cases}$	$f_1(x) = \frac{4}{\pi(1+x^2)}$
1.2	$S_2(x) = \begin{cases} \frac{2x}{1-x}, & 0 \le x \le \frac{1}{3} \\ \frac{1-x}{2x}, & \frac{1}{3} \le x \le 1 \end{cases}$	$f_2(x) = \frac{2}{(1+x)^2}$
1.3	$S_3(x) = 4x(1-x)$	$f_3(x) = \frac{1}{\pi\sqrt{x(1-x)}}$

Table 3.1: Phase maps and their invariant densities (1D).

Theorem 3.5.2. *Let $S : [0,1] \to [0,1]$ satisfy conditions of Theorem 3.5.1 and $S(a_{i-1}, a_i) = (0,1)$ for all $i = 1, 2, \ldots, r$. Then the associated Perron-Frobenius operator $\mathcal{P}$ has a unique invariant density $f^* \in D$ such that $\mathcal{P}f^* = f^*$ and*

$$\lim_{n\to\infty} \|\mathcal{P}^n f - f^*\|_{L^1} = 0 \quad \text{for all } f \in D.$$

In addition, if a map T is exact, the uniqueness of the invariant density is preserved for any topologically conjugate map S. This is ensured by the following theorem [64, Theorem 6.5.2].

Theorem 3.5.3. *Let $T : [0,1] \to [0,1]$ be a measurable, nonsingular transformation and let $f^* \in D$ be a given density. Let $S : [0,1] \to [0,1]$ be given by $S = g^{-1} \circ T \circ g$, where*

$$g(x) = \int_0^x f^*(y)dy, \quad 0 < x < 1.$$

Then T is exact if and only if f^* is a unique invariant density of the Perron-Frobenius operator $\mathcal{P}$ with respect to S and

$$\lim_{n\to\infty} \|\mathcal{P}^n f - f^*\|_{L^1} = 0 \quad \text{for all } f \in D.$$

Several examples of 1D and 2D dynamical systems, along with their invariant densities [37, 56], are presented in Table 3.1 and 3.2, respectively. We can verify that the maps $S_1(x)$ and $S_2(x)$ satisfy conditions of Theorem 3.5.2. Furthermore, the dyadic transformation $V_1(y)$ and the tent map $U_2(x)$ also satisfy these conditions. Thus, the existence and uniqueness of their invariant densities are guaranteed. For the logistic map $S_3(x)$, the uniqueness of the invariant density is established via Theorem 3.5.3 using T as the tent map (cf. [64, p. 167]). The invariant densities for the two-dimensional systems in Table 3.2 follow from the theory of product transformations.

Examples	**Phase maps** $S_i(x,y) = (U_i(x), V_i(y))$	**Invariant densities**
2.1	$U_1(x) = 4x(1-x)$ $V_1(y) = 2y \mod 1$	$f_1(x,y) = \frac{1}{\pi\sqrt{x(1-x)}}$
2.2	$U_2(x) = \begin{cases} 2x, & 0 \le x \le \frac{1}{2} \\ 2(1-x), & \frac{1}{2} \le x \le 1 \end{cases}$ $V_2(y) = \begin{cases} \frac{2y}{1-y}, & 0 \le y \le \frac{1}{3} \\ \frac{1-y}{2y}, & \frac{1}{3} \le y \le 1 \end{cases}$	$f_2(x,y) = \frac{2}{(1+y)^2}$
2.3	$U_3(x) = \left(\frac{1}{8} - 2\left\lvert x - \frac{1}{2}\right\rvert^3\right)^{1/3} + \frac{1}{2}$ $V_3(y) = \begin{cases} \frac{2y}{1-y^2}, & 0 \le y \le \sqrt{2}-1 \\ \frac{1-y^2}{2y}, & \sqrt{2}-1 \le y \le 1 \end{cases}$	$f_3(x,y) = \frac{48(x-1/2)^2}{\pi(1+y^2)}$

Table 3.2: Phase maps and their invariant densities (2D).

Since each 2D map $S_i(x,y) = (U_i(x), V_i(y))$ is composed of two independent one-dimensional maps, the unique joint invariant density is simply the product of their invariant densities. As an example, a detailed verification showing that $f_3(x)$ is the invariant density for the logistic map $S_3(x)$ is provided in Appendix B.

3.6 Existing methods

In this section, we review several numerical methods for approximating the solutions of invariant density and Neumann series problems. These include Ulam's method, the Galerkin and Petrov-Galerkin methods, the maximum entropy method, and the minimum energy method.

3.6.1 Ulam's method

Ulam's method [61] is a technique for discretizing the Perron-Frobenius operator based on a Galerkin projection onto a subspace of step functions. The approach begins by partitioning the state space Ω into k disjoint subsets (often called boxes or cells), denoted by $\mathbb{B}_1, \ldots, \mathbb{B}_k$. For each subset $\mathbb{B}_i$, we define the characteristic function $\chi_{\mathbb{B}_i}$. The collection of these functions $\{\chi_{\mathbb{B}_i}\}_{i=1}^k$ serves as a basis for a finite-dimensional

subspace of $L^1(\Omega)$. The discretized Perron-Frobenius operator is represented by a matrix $P_k = [p_{ij}]_{k\times k}$ where

$$p_{ij} = \frac{\mu(S^{-1}(\mathbb{B}_j) \cap \mathbb{B}_i)}{\mu(\mathbb{B}_i)}.$$

Here, p_{ij} represents the fraction of the measure of box $\mathbb{B}_i$ that is mapped into box $\mathbb{B}_j$ under the transformation S. In practice, these coefficients can be estimated by sampling n test points $\{x_i^{(l)}\}_{l=1}^n$ uniformly within each box $\mathbb{B}_i$. The entry p_{ij} is then approximated by the proportion of points whose images under S fall into $\mathbb{B}_j$:

$$p_{ij} \approx \frac{1}{n}\sum_{l=1}^{n} \chi_{\mathbb{B}_j}(S(x_i^{(l)})). \tag{3.14}$$

Using this discretization, an invariant density is approximated by the function $f := \sum_{i=1}^k f_i \chi_{\mathbb{B}_i}$ where the vector of coefficients $\mathbf{f} = (f_1, f_2, \ldots, f_k)^T$ is a fixed point of the matrix, satisfying $P_k\mathbf{f} = \mathbf{f}$. Detailed discussions on the convergence conditions and error rates of Ulam's method for certain classes of mappings can be found in [78] and [40, Chapter 7]. Furthermore, Neumann series problems can be solved via the matrix equation $(I_k - \alpha P_k)\mathbf{f}_\infty = \mathbf{f}_0$, where I_k is the $k\times k$ identity matrix and $\alpha \in (0,1)$ is the damping parameter [27].

3.6.2 Galerkin and Petrov-Galerkin methods

The *Galerkin method* is a technique to approximate the solution of an operator equation as a linear combination of basis functions. While it is commonly used for solving PDEs in Hilbert spaces, it is also applied to discretize Perron-Frobenius operators [61].

To apply this method, we consider the solution u, representing the Neumann series or the invariant density, to be an element of the Hilbert space $V := L^2(\Omega)$. We then define a finite-dimensional subspace $V_M \subset V$ spanned by a basis $\{g_1, \ldots, g_M\}$. Let $b : V \times V \to \mathbb{R}$ be a bilinear form and $l : V \to \mathbb{R}$ be a bounded linear functional. The original variational problem is formulated as follows:

$$\text{Find } u \in V \text{ such that } b(u,v) = l(v) \quad \forall v \in V.$$

We then approximate the solution $u \in V$ by a function $u_M \in V_M$ that satisfies the Galerkin condition:

$$\text{Find } u_M \in V_M \text{ such that } b(u_M, v_M) = l(v_M) \quad \forall v_M \in V_M.$$

The specific definitions of b and l depend on the problem being addressed:

- Neumann series problems:

$$b(u,v) := \int_\Omega (u - \alpha \mathcal{P}u) v d\mu \quad \text{and} \quad l(v) := \int_\Omega f_0 v d\mu.$$

- Invariant density problems:

$$b(u,v) := \int_\Omega (\mathcal{P}u - u) v d\mu \quad \text{and} \quad l(v) := 0.$$

By substituting $u_M = \sum_{j=1}^{M} c_j g_j$ into the Galerkin condition, the problem reduces to solving a system of linear equations for the coefficients c_j.

The well-posedness of these variational problems is grounded on classical results from functional analysis. In particular, the *Lax–Milgram theorem* [42, Section 6.2] guarantees the existence and uniqueness of a solution whenever the bilinear form and linear functional satisfy certain boundedness and coercivity conditions:

Theorem 3.6.1 (Lax-Milgram Theorem)**.** *Let V be a Hilbert space with the norm $\|\cdot\|_V$. Let $b : V \times V \to \mathbb{R}$ be a bilinear form and $l : V \to \mathbb{R}$ be a linear functional. Suppose the following conditions hold:*

1. *Boundedness of b: There exists a constant $c_b > 0$ such that*

$$|b(u,v)| \leq c_b \|u\|_V \|v\|_V \quad \forall u, v \in V.$$

2. *Boundedness of l: There exists a constant $c_l > 0$ such that*

$$|l(v)| \leq c_l \|v\|_V \quad \forall v \in V.$$

3. *Coercivity of b: There exists a constant $c > 0$ such that*

$$b(u,v) \geq c \|u\|_V^2 \quad \forall u \in V.$$

Then, there exists a unique solution $u \in V$ to the variational problem:

$$\text{Find } u \in V \text{ such that } b(u,v) = l(v) \quad \forall v \in V.$$

When the solution is in a Banach space $U := L^p(\Omega)$, for $1 < p < \infty$, the test space V is typically chosen as the dual space $L^q(\Omega)$, where $\frac{1}{p} + \frac{1}{q} = 1$. In this setting, we define a variational formulation by choosing a finite-dimensional trial space $U_M \subset U$ and a test space $V_M \subset V$. To solve the original variational problem:

$$\text{Find } u \in U \text{ such that } b(u, v) = l(v) \quad \forall v \in V, \tag{3.15}$$

we approximate u with a function $u_M \in U_M$ that satisfies the discrete condition:

$$\text{Find } u_M \in U_M \text{ such that } b(u_M, v_M) = l(v_M) \quad \forall v_M \in V_M.$$

This approach is known as the *Petrov-Galerkin method*. The well-posedness of this variational formulation is ensured by the *Banach-Nečas-Babŭska Theorem* [91, p. 15], which generalizes the Lax-Milgram theorem to settings where the trial and test spaces differ.

Theorem 3.6.2 (Banach-Nečas-Babŭska Theorem)**.** *Let U be a Banach space and V be a reflexive Banach space with norms $\|\cdot\|_U$ and $\|\cdot\|_V$, respectively. Suppose $b : U \times V \to \mathbb{R}$ is a bilinear form and $l : V \to \mathbb{R}$ is a linear functional. If the following conditions hold:*

1. *Boundedness of b: There exists $c_b > 0$ such that*

$$|b(u, v)| \leq c_b \|u\|_U \|v\|_V \quad \forall u \in U, v \in V.$$

2. *Boundedness of l: There exists $c_l > 0$ such that*

$$|l(v)| \leq c_l \|v\|_V \quad \forall v \in V.$$

3. *Inf-sup stability: There exists $c > 0$ such that*

$$\sup_{0 \neq v \in V} \frac{b(u, v)}{\|v\|_V} \geq c\|u\|_U \quad \forall u \in U.$$

4. *Adjoint injectivity: For any $v \in V$, if $b(u, v) = 0$ for all $u \in U$, then $v = 0$.*

Then, there exists a unique solution $u \in U$ to the variational problem:

$$\textit{Find } u \in U \textit{ such that } b(u, v) = l(v) \quad \forall v \in V.$$

3.6.3 Maximum entropy method

In this section, we describe the *maximum entropy method* (MEM) for approximating invariant densities. To illustrate the method in one dimension, we restrict the domain to $\Omega = [0, 1]$. For a nonnegative density function $f \in D$ (see (3.13)), the *entropy* is defined as:

$$H(f) := -\int_0^1 f(x) \ln f(x) dx.$$

The fundamental principle of MEM is to maximize this entropy subject to constraints derived from the physical system. Following the approach in [37], we apply this method to find the invariant density of the Perron-Frobenius operator.

Let $S : [0, 1] \to [0, 1]$ be a nonsingular transformation such that the corresponding Perron-Frobenius operator $\mathcal{P}$ has an invariant density f^*, satisfying

$$\mathcal{P} f^* = f^*. \tag{3.16}$$

To obtain the necessary constraints, we multiply (3.16) by test functions x^n, for $n = 1, 2, \ldots$, and integrate over the domain:

$$\int_0^1 \mathcal{P} f^*(x) \cdot x^n dx = \int_0^1 f^*(x) \cdot x^n dx, \quad n = 1, 2, \ldots \tag{3.17}$$

Using the duality property from Equation (3.7), we obtain:

$$\int_0^1 f^*(x) S(x)^n dx = \int_0^1 f^*(x) x^n dx, \quad n = 1, 2, \ldots$$

This provides a set of constraints for the invariant density:

$$\int_0^1 f^*(x)[x^n - S(x)^n] dx = 0, \quad n = 1, 2, \ldots \tag{3.18}$$

Conversely, if f^* solves Equation (3.18), then Equation (3.17) also holds true. In addition, Equation (3.17) is always true for $n = 0$. Since for any $\psi \in L^1([0, 1])$, $\int_0^1 \psi(x) \cdot x^n dx = 0 \quad \forall n = 0, 1, 2, \ldots$ implies $\psi = 0$, we obtain that f^* is the invariant density.

In practice, we choose a finite number of constraints N_c and solve the following optimization problem:

$$\text{maximize :} \quad -\int_0^1 f(x) \ln f(x) dx \tag{3.19}$$

$$\text{subject to :} \quad c_0(f) := \int_0^1 f(x) dx - 1 = 0 \tag{3.20}$$

$$c_i(f) := \int_0^1 f(x)[x^i - S(x)^i] dx = 0, \quad i = 1, 2, \ldots, N_c. \tag{3.21}$$

The solution f_{N_c} to this problem, which serves as an approximation of the invariant density f^*, is in the form:

$$f_{N_c}(x) = \frac{\exp\left(\sum_{n=1}^{N_c} a_n[x^n - S(x)^n]\right)}{\int_0^1 \exp\left(\sum_{n=1}^{N_c} a_n[x^n - S(x)^n]\right) dx}, \tag{3.22}$$

where the coefficients $a_n \in \mathbb{R}$ are determined by the constraints (3.20)-(3.21). See [64] for the details of the proof for the general maximum entropy method.

If the entropy of the invariant density $H(f^*) > -\infty$, then f_{N_c} converges to f^* as $N_c \to \infty$. Moreover, the approximation error [37] is bounded by:

$$\|f_{N_c} - f^*\|_{L^1} \leq E_{N_c} e^{E_{N_c}/2},$$

where

$$E_{N_c} = \inf\left\{ \left\| 1 + \ln f^* - \sum_{n=1}^{N_c} a_n[x^n - S(x)^n] \right\|_{L^\infty} : (a_1 \dots a_{N_c})^T \in \mathbb{R}^{N_c} \right\}.$$

This means that the approximation error depends on the L^∞ distance between the function $1+\ln f^*$ and the subspace spanned by $\{x-S(x), x^2-S(x)^2, \dots, x^{N_c}-S(x)^{N_c}\}$.

To improve numerical stability, we can replace the monomials x^n by orthogonal polynomials p_n, such as Chebyshev polynomials [39]. This leads to the modified constraints:

$$\text{maximize :} \quad -\int_0^1 f(x) \ln f(x) dx \tag{3.23}$$

$$\text{subject to :} \quad c_0(f) := \int_0^1 f(x) dx - 1 = 0 \tag{3.24}$$

$$c_i(f) := \int_0^1 f(x)[p_i(x) - p_i(S(x))] dx = 0, \quad i = 1, 2, \dots, N_c. \tag{3.25}$$

The solution f_{N_c} to this problem is in the form

$$f_{N_c}(x) = \frac{\exp\left(\sum_{n=1}^{N_c} a_n[p_n(x) - p_n(S(x))]\right)}{\int_0^1 \exp\left(\sum_{n=1}^{N_c} a_n[p_n(x) - p_n(S(x))]\right) dx}, \tag{3.26}$$

where $a_n \in \mathbb{R}$ satisfying the constraints (3.24)-(3.25).

3.6.4 Minimum energy method

The entropy functional $H(f)$ can be replaced by a quadratic energy functional $E(f) := \frac{1}{2}\int_0^1 [f(x)]^2 dx$ [15]. This alternative approach was studied in [14], where characteristic

functions over a partitioned domain were used as generating functions. Instead of maximizing entropy, we can consider the problem of minimizing the energy functional. We prove the following theorem, which provides a result for this quadratic case that is analogous to the MEM result.

Theorem 3.6.3. *Let $f : [0,1] \to \mathbb{R}$ be a density function and let $\{g_i\}_{i=1}^{N_c}$ be a set of test functions on $[0,1]$. Consider the following optimization problem:*

$$\textit{minimize}: \quad \int_0^1 [f(x)]^2 dx \tag{3.27}$$

$$\textit{subject to}: \quad c_0(f) := \int_0^1 f(x)dx - 1 = 0 \tag{3.28}$$

$$c_i(f) := \int_0^1 f(x)g_i(x)dx = 0, \quad i = 1, 2, \ldots, N_c. \tag{3.29}$$

The solution to this problem is given by a function f_{N_c} of the form:

$$f_{N_c}(x) = a_0 + \sum_{i=1}^{N_c} a_i g_i(x), \tag{3.30}$$

where the coefficients a_i are chosen such that f_{N_c} satisfies the constraints (3.28)-(3.29).

Proof. To prove that f_{N_c} is the minimizer, we show that for any function f satisfying the constraints (3.28)-(3.29), the following inequality holds:

$$\int_0^1 [f(x)]^2 dx \geq \int_0^1 [f_{N_c}(x)]^2 dx.$$

First, we multiply Equation (3.30) by $f_{N_c}(x)$ and integrate over the domain:

$$\int_0^1 [f_{N_c}(x)]^2 dx = a_0 + \sum_{i=1}^{N_c} a_i \int_0^1 f_{N_c}(x) g_i(x) dx.$$

Applying the constraints $c_i(f_{N_c}) = 0, i = 1, \ldots, N_c$, we obtain

$$a_0 = \int_0^1 [f_{N_c}(x)]^2 dx.$$

Now, let f be any function satisfying the constraints (3.28)-(3.29). Using the fact that

$u^2 \geq 2uv - v^2$, and setting $u = f(x)$ and $v = f_{N_c}(x)$, we integrate over the domain:

$$\begin{aligned}
\int_0^1 [f(x)]^2 dx &\geq 2\int_0^1 f(x) f_{N_c}(x) dx - \int_0^1 [f_{N_c}(x)]^2 dx \\
&= 2\int_0^1 f(x) \left[a_0 + \sum_{i=1}^{N_c} a_i g_i(x) \right] dx - \int_0^1 [f_{N_c}(x)]^2 dx \\
&= 2a_0 \int_0^1 f(x) dx + 2\sum_{i=1}^{N_c} a_i \int_0^1 f(x) g_i(x) dx - \int_0^1 [f_{N_c}(x)]^2 dx \\
&= 2a_0 - \int_0^1 [f_{N_c}(x)]^2 dx \\
&= 2\int_0^1 [f_{N_c}(x)]^2 dx - \int_0^1 [f_{N_c}(x)]^2 dx \\
&= \int_0^1 [f_{N_c}(x)]^2 dx.
\end{aligned}$$

This completes the proof that f_{N_c} is the minimizer of the energy functional. $\square$

Chapter 4

Neural Network Methods for Perron-Frobenius Neumann Series

4.1 Introduction

Perron-Frobenius operators, also known as transfer operators, play an important role in capturing behaviors of dynamical systems. These operators have been applied to various fields of research, including engineering [51,104], earth sciences [44], and atmospheric sciences [97, 98]. Another viewpoint to describe system behaviors is studying observables using Koopman operators, which are the adjoints of Perron-Frobenius operators. While Koopman operators track the evolution of functions on the state space, Perron-Frobenius operators describe statistical behavior through the evolution of densities [61]. These operators provide linear representations of (nonlinear) dynamical systems, which are important for estimating the long-term system behavior and prediction of future states [57].

One particularly interesting problem associated with Perron-Frobenius operators is finding their Neumann series when applied to a given initial density. The solution to this problem corresponds to the accumulated density in the long-time limit, converging toward the stationary density. This problem is closely related to the dynamical energy analysis approach [96] used to determine wave energy distributions in the high-frequency regime, which has a direct application in finding equilibrium energy distributions [86,94]. A classical approach for approximating solutions to problems in-

This chapter is based on our work originally published in [100]

volving such operators is Ulam's method [101], which approximates Perron-Frobenius operators by projecting them onto a subspace spanned by characteristic functions (a space of piecewise constant functions). However, without additional effort, fixed-grid-based methods, including Ulam's method, are known to have limitations, particularly in handling irregularities, and the convergence rate is relatively low [13].

In recent years, neural networks have been applied more and more to improve the approximate solution of various mathematical problems (see [20, 58, 59, 84, 87, 93, 107] for examples), including problems in dynamical systems (see [3, 43, 68, 79, 88, 108] for examples). Among these methods, Physics-Informed Neural Networks (PINNs) [84] and various versions of Variational PINNs (VPINNs) [58, 59, 87] have gained popularity for solving equations involving operators, such as partial differential equations (PDEs). PINNs were developed to generate neural network functions to approximate solutions by minimizing a loss function based on the PDE residual. Similar methods can be found in [17, 21]. VPINNs were introduced in [58], aiming to solve PDEs in variational forms, with further studies presented in [8, 9]. Robust VPINNs (RVPINNs) were developed to give a stable version of VPINNs by minimizing a loss function based on the discrete dual norm of the residual of the variational equation. Throughout this work, we use the terms PINNs and RVPINNs to refer to neural networks that encode equations describing systems by minimizing the residual of the equations in their strong and variational forms, respectively.

Neural networks offer several advantages for approximating solutions, including high expressivity [50], handling complex domains [32], and scalability to high-dimensional problems [81]. Several papers have used neural networks to learn operators in dynamical systems. For example, the authors of the seminal work [73] have proposed a deep learning framework to learn a finite-dimensional approximation of Koopman operators. However, few papers have been dedicated to studying neural networks for problems involving Perron-Frobenius operators. Applying neural networks to solve problems related to Perron-Frobenius operators, in particular Neumann series problems, would be beneficial for approximating irregular solutions or addressing high-dimensional spaces.

In this chapter, we propose and analyze neural network methods for approximating solutions to Neumann series problems for Perron-Frobenius operators with an initial

density in L^p, where $1 < p < \infty$, considering both their strong and variational forms. Our primary focus is on Perron-Frobenius operators that are non-expansive under the L^p-norm with a constant damping parameter in $(0, 1)$. For consistency and simplicity, we present most results for solutions in L^2 but also provide details for solutions in L^p.

The main contributions of this work are as follows. We define an abstract setting for non-expansive Perron-Frobenius operators from L^p to L^p. We establish the well-posedness of the variational formulation in L^2 and L^p settings. We propose neural network methods for solving Neumann series problems based on PINNs and RVPINNs. A key advantage of the RVPINNs approach is that it does not require the inverse map of the underlying dynamical system, as highlighted in Remark 4.3.1. To derive a priori error estimates for the proposed methods, we recall the concepts of neural network function manifolds and quasi-minimizers presented in [17]. The local Fortin's condition modified from [87] is required to analyze the RVPINNs method. We implement the proposed methods for approximating the Neumann series of Perron-Frobenius operators associated with the tent map, the boundary map in a circular domain, and the standard map to show the performance of our methods. We also demonstrate the applicability of our methods by approximating interior densities in a two-cavity system. We compare our methods against a fixed-grid-based approach and the truncated sum of Neumann series.

The chapter is organized as follows. In section 4.2, we introduce Perron-Frobenius operators, state our assumptions, present Neumann series problems in both strong and variational forms, and provide the well-posedness of the variational problems when solutions are in L^2. We also review fixed-grid-based methods, including Ulam's method, and provide an associated error estimate. Section 4.3 describes neural network frameworks, including our proposed methods for approximating solutions, focusing on PINNs and RVPINNs. Section 4.4 is devoted to the analysis of these methods. Section 4.5 presents numerical examples. Section 4.6 shows conclusions and possible extensions. Appendix C.1 provides the well-posedness of the variational problem when solutions are in L^p. Appendix C.2 shows the equivalent form of the RVPINNs loss function. Appendix C.3 develops error estimate proofs for PINNs and RVPINNs.

4.2 Abstract framework

Let Ω be a bounded subset of $\mathbb{R}^d, d \geq 1$, and $S : \Omega \to \Omega$ be a map describing a discrete dynamical system. Given a measure space $(\Omega, \mathcal{B}, \mu)$ where $\mathcal{B}$ is a Borel sigma-algebra and μ is the Lebesgue measure, the Perron-Frobenius operator associated with S is defined as the linear operator $\mathcal{P} : L^1(\Omega) \to L^1(\Omega)$ such that

$$\int_A \mathcal{P} f d\mu = \int_{S^{-1}(A)} f d\mu, \quad \forall A \in \mathcal{B}, \forall f \in L^1(\Omega). \tag{4.1}$$

In the special case where $\Omega = [a, b]$ is a closed interval, the Perron-Frobenius operator $\mathcal{P}$ can be expressed explicitly (see Equation (3.5)) as:

$$\mathcal{P} f(x) = \frac{d}{dx} \int_{S^{-1}([a,x])} f(s) ds. \tag{4.2}$$

It is well-known that the Perron-Frobenius operator $\mathcal{P}$ is non-expansive under the L^1-norm, i.e., $\|\mathcal{P} f\|_{L^1} \leq \|f\|_{L^1}$ for all $f \in L^1(\Omega)$. For simplicity[1], we consider densities f in $L^p(\Omega)$, where $1 < p < \infty$, which is a subspace of $L^1(\Omega)$. So, we need additional assumptions on the Perron-Frobenius operator for defining our problem in L^p-setting. In this chapter, we make the following assumptions:

(A1) Density functions f are nonnegative functions in $U := L^p(\Omega)$ with $1 < p < \infty$,

(A2) $\mathcal{P}$ is non-expansive under the $U-$norm, i.e., for all $f \in U$,

$$\|\mathcal{P} f\|_U \leq \|f\|_U.$$

Remark 4.2.1. The assumption (A2) is equivalent to stating that $\mathcal{P} : U \to U$ is a bounded operator with operator norm $\|\mathcal{P}\| \leq 1$.

Remark 4.2.2. The assumption (A2) holds in many dynamical systems. For instance, if S is an invertible nonsingular transformation, the Perron-Frobenius operator $\mathcal{P}$ associated with S is given by

$$\mathcal{P} f(x) = f(S^{-1}(x)) \left| J_{S^{-1}} \right|, \quad \text{for } x \in \Omega,$$

[1] $L^p(\Omega)$ (with $1 < p < \infty$) has convenient properties for analysis such as reflexivity and strict convexity, but these properties do not hold in $L^1(\Omega)$ [77].

where $|J_{S^{-1}}|$ denotes the Jacobian determinant of the inverse map S^{-1} (cf. [64, Corollary 3.2.1.]). Therefore, if $|J_{S^{-1}}| \leq 1$, $\mathcal{P}$ satisfies the assumption (A2). Indeed, by applying the change of variable, we obtain

$$\begin{aligned}\|\mathcal{P}f\|_U &= \left(\int_\Omega \left(f(S^{-1})\right)^p |J_{S^{-1}}|^p \, d\mu\right)^{1/p} \\ &\leq \left(\int_\Omega \left(f(S^{-1})\right)^p |J_{S^{-1}}| \, d\mu\right)^{1/p} = \left(\int_{S^{-1}(\Omega)} f^p d\mu\right)^{1/p} = \|f\|_U.\end{aligned}$$

4.2.1 Neumann series problem

We are interested in approximating the solution of the following problem: Given a nonnegative function $f_0 \in U$, find $u \in U$ satisfying

$$u - \alpha\mathcal{P}u = f_0, \tag{4.3}$$

where $0 < \alpha < 1$ is a damping parameter. This is equivalent to finding the Neumann series expressed as

$$u = f_0 + \alpha\mathcal{P}f_0 + \alpha^2\mathcal{P}^2 f_0 + \cdots . \tag{4.4}$$

Remark 4.2.3. In applications, the Neumann series (4.4) represents the equilibrium energy distribution resulting from a continuous injection of the initial energy distribution f_0 into the system, with energy reduced by a factor α each time it reaches the boundary. In a more general setting, the damping parameter α can be replaced by a suitable weight function $\alpha : \Omega \to [0, \infty)$. This would mean more realistic energy losses for individual trajectories.

4.2.2 Variational problem

The variational version of problem (4.3) is the following:

$$\text{Find } u \in U \text{ such that } b(u, v) = l(v) \quad \forall v \in V, \tag{4.5}$$

where V is the dual space of U, $b(\cdot, \cdot)$ and $l(\cdot)$ are a bilinear form on $U \times V$ and a linear functional on V, respectively, defined by

$$b(u, v) := \int_\Omega (u - \alpha\mathcal{P}u)v d\mu \quad \text{and} \quad l(v) := \int_\Omega f_0 v d\mu. \tag{4.6}$$

Remark 4.2.4. The bilinear form $b(\cdot,\cdot)$ defined in (4.6) can be written in terms of the associated Koopman operator[2] $\mathcal{K}$, by applying the duality identity[3], as follows:

$$b(u,v) := \int_\Omega u(v - \alpha\mathcal{K}v)d\mu. \tag{4.7}$$

This form will be used to reformulate the RVPINNs loss function in Remark 4.3.1.

When $U = V = L^2(\Omega)$, the variational problem (4.5)-(4.6) satisfies the conditions of the Lax-Milgram Theorem, ensuring the existence and uniqueness of the solution. The details regarding the well-posedness for $U = L^p(\Omega)$ and $V = L^q(\Omega)$, where $1 < p, q < \infty$ with $\frac{1}{p} + \frac{1}{q} = 1$, are provided in C.1.

Theorem 4.2.1 (Well-posedness of variational problem in L^2)**.** *Given $U = V = L^2(\Omega)$, the variational problem (4.5) satisfies the following.*

1. *Boundedness of b:* $|b(u,v)| \leq (1+\alpha)\|u\|_{L^2}\|v\|_{L^2}, \quad \forall u, v \in L^2(\Omega)$,
2. *Boundedness of l:* $|l(v)| \leq \|f_0\|_{L^2}\|v\|_{L^2}, \quad \forall v \in L^2(\Omega)$,
3. *Coercivity:* $b(u,u) \geq (1-\alpha)\|u\|_{L^2}^2, \quad \forall u \in L^2(\Omega)$.

Hence, it has a unique solution.

Proof. 1. Boundedness of b: For any $u, v \in L^2(\Omega)$, by using the Cauchy-Schwarz inequality, the triangle inequality, and the assumption (A2), we have

$$\begin{aligned} |b(u,v)| = \left|\int_\Omega (u - \alpha\mathcal{P}u)v d\mu\right| &\leq \int_\Omega |(u - \alpha\mathcal{P}u)v| d\mu \leq \|u - \alpha\mathcal{P}u\|_{L^2}\|v\|_{L^2} \\ &\leq (\|u\|_{L^2} + \alpha\|\mathcal{P}u\|_{L^2})\,\|v\|_{L^2} \leq (1+\alpha)\|u\|_{L^2}\|v\|_{L^2}. \end{aligned}$$

2. Boundedness of l: Since $f_0, v \in L^2(\Omega)$, by using the Cauchy-Schwarz inequality,

$$|l(v)| = \left|\int_\Omega f_0 v d\mu\right| \leq \|f_0\|_{L^2}\|v\|_{L^2}.$$

3. Coercivity: For any $u \in L^2(\Omega)$, by using the Cauchy-Schwarz inequality and the assumption (A2), we have

$$\begin{aligned} b(u,u) = \int_\Omega (u - \alpha\mathcal{P}u)u d\mu &= \int_\Omega u^2 d\mu - \alpha\int_\Omega (\mathcal{P}u)u d\mu \\ &\geq \|u\|_{L^2}^2 - \alpha\|\mathcal{P}u\|_{L^2}\|u\|_{L^2} \geq \|u\|_{L^2}^2 - \alpha\|u\|_{L^2}^2 = (1-\alpha)\|u\|_{L^2}^2. \end{aligned}$$

□

[2]The Koopman operator associated with the map S is defined by $\mathcal{K}v = v \circ S$.

[3]Let $1 < p, q < \infty$ with $\frac{1}{p} + \frac{1}{q} = 1$. If $\mathcal{P} : L^p(\Omega) \to L^p(\Omega)$ is the Perron-Frobenius operator and $\mathcal{K} : L^q(\Omega) \to L^q(\Omega)$ is the associated Koopman operator, then $\int_\Omega (\mathcal{P}u)v d\mu = \int_\Omega u(\mathcal{K}v)d\mu$.

4.2.3 Fixed-grid-based method

Let $U = V = L^2(\Omega)$. A classical way for solving problem (4.5) is to discretize the problem based on Galerkin projections. To do that, we first consider a finite-dimensional subspace $V_M \subset V$ with a basis $\{g_1, \ldots, g_M\}$. Next, we seek $u_M \in V_M$ such that

$$b(u_M, g_m) = l(g_m) \quad \forall m \in \{1, 2, \ldots, M\}, \tag{4.8}$$

where $b(\cdot, \cdot)$ and $l(\cdot)$ are defined in (4.6). Then, as $u_M \in V_M$, we write $u_M = \sum_{k=1}^{M} c_k g_k$ where c_k is a coefficient associated to g_k. Finally, we solve the system of equations:

$$\sum_{k=1}^{M} c_k \int_\Omega \left(g_k - \alpha(\mathcal{P} g_k)\right) g_m d\mu = \int_\Omega f_0 g_m d\mu, \quad m = 1, 2, \ldots, M.$$

Solving this system is equivalent to solving the matrix equation $A\mathbf{c} = \mathbf{b}$, where the matrix entries of A are given by $a_{mk} = \int_\Omega \left(g_k - \alpha(\mathcal{P} g_k)\right) g_m d\mu$ and the components of the vector $\mathbf{b}$ are $b_m = \int_\Omega f_0 g_m d\mu$.

Remark 4.2.5. If the domain Ω is partitioned into M subdomains $\omega_1, \ldots, \omega_M$ and g_m is chosen to be the normalized characteristic function on a subdomain ω_m, which forms an orthonormal basis of V_M, the fixed-grid-based method is equivalent to the so-called Ulam's method [101]. The matrix equation is equivalent to $(I_M - \alpha P_M)\mathbf{c} = \mathbf{b}$, where I_M is the $M \times M$ identity matrix and P_M is the matrix representation of $\mathcal{P}$ given by $P_M = [p_{mk}]_{M \times M}$ with

$$p_{mk} = \frac{\mu(S^{-1}(\omega_m) \cap \omega_k)}{\mu(\omega_k)}.$$

The following theorem shows an error estimate for the fixed-grid-based method.

Theorem 4.2.2. *Let $u_M \in V_M$ be the solution of problem (4.8) and u be the solution of problem (4.5) for $U = V = L^2(\Omega)$. We have*

$$\|u - u_M\|_{L^2} \leq \frac{2}{1 - \alpha} \inf_{v_m \in V_M} \|u - v_m\|_{L^2}.$$

Proof. For any $v_m \in V_M$, by using coercivity, Galerkin orthogonality, and the bounded-

ness of b, we have

$$\begin{aligned}\|v_m - u_M\|_{L^2}^2 &\leq \frac{1}{1-\alpha} b(v_m - u_M, v_m - u_M) \\ &= \frac{1}{1-\alpha} \left[b(v_m - u, v_m - u_M) + b(u - u_M, v_m - u_M)\right] \\ &= \frac{1}{1-\alpha} b(v_m - u, v_m - u_M) \\ &\leq \frac{1+\alpha}{1-\alpha} \|v_m - u\|_{L^2} \|v_m - u_M\|_{L^2}.\end{aligned}$$

Dividing both sides by $\|v_m - u_M\|_{L^2}$ yields

$$\|v_m - u_M\|_{L^2} \leq \frac{1+\alpha}{1-\alpha} \|v_m - u\|_{L^2}.$$

Applying the triangle inequality, we have

$$\begin{aligned}\|u - u_M\|_{L^2} &\leq \|u - v_m\|_{L^2} + \|v_m - u_M\|_{L^2} \\ &\leq \|u - v_m\|_{L^2} + \frac{1+\alpha}{1-\alpha} \|v_m - u\|_{L^2} \\ &= \frac{2}{1-\alpha} \|u - v_m\|_{L^2}.\end{aligned}$$

Taking the infimum over all possible $v_m \in V_M$ yields the theorem. □

Remark 4.2.6. In the case where V_M is the space of piecewise-linear finite elements defined on a mesh of size h, the error bound for a solution u in the Sobolev space H^2 is of order $O(h^2)$ (cf. [1, Theorem 1.5]).

4.3 Neural network framework

To approximate the solution of (4.3) or (4.5), we use a neural network u_θ of the form given in (2.3). Here, the output layer uses the identity map as its activation function. Recall that $\mathcal{M}_n$ is the set of neural network functions with n parameters (see (2.8)), and assume that $\emptyset \neq \mathcal{M}_n \subset U$. We aim to find $u_\theta \in \mathcal{M}_n$ such that $u_\theta \approx u$ where u is the solution of (4.3) or (4.5). To do that, we train the neural network to minimize a loss function using some optimization methods, such as the BFGS method or Adam (see Section 2.4).

4.3.1 PINNs

To approximate the solution of problem (4.3) for $U = L^p(\Omega)$ with $1 < p < \infty$, we follow the PINNs method by using a loss function defined in terms of the residual of

the equation. The loss function is given by:

$$\mathcal{L}_{\text{PINNs}}(u_\theta) := \|f_0 - u_\theta + \alpha \mathcal{P} u_\theta\|_U. \tag{4.9}$$

The approximate solution of (4.3) is obtained by solving the optimization problem:

$$\text{Find } u_{\theta^*} \in \mathcal{M}_n \text{ such that } \theta^* = \underset{\theta \in \mathbb{R}^s}{\operatorname{argmin}} \mathcal{L}_{\text{PINNs}}(u_\theta). \tag{4.10}$$

4.3.2 RVPINNs

For a given discrete space $V_M \subset V$, we consider the following Petrov-Galerkin type discretization of (4.5):

$$\text{Find } u_\theta \in \mathcal{M}_n \text{ such that } b(u_\theta, v_M) = l(v_M) \quad \forall v_M \in V_M. \tag{4.11}$$

We follow the RVPINNs method to approximate the solution of problem (4.11) for $U = L^p(\Omega)$ with $1 < p < \infty$ by defining a loss function based on the discrete dual norm of the variational equation residual. The loss function is given by:

$$\mathcal{L}_{\text{RVPINNs}}(u_\theta) := \sup_{0 \neq v_M \in V_M} \frac{l(v_M) - b(u_\theta, v_M)}{\|v_M\|_V}. \tag{4.12}$$

The approximate solution of (4.11) is obtained by solving the optimization problem:

$$\text{Find } u_{\theta^*} \in \mathcal{M}_n \text{ such that } \theta^* = \underset{\theta \in \mathbb{R}^s}{\operatorname{argmin}} \mathcal{L}_{\text{RVPINNs}}(u_\theta). \tag{4.13}$$

In practical settings, where $U = V = L^2(\Omega)$ and V_M is a finite-dimensional subspace of $L^2(\Omega)$ with an orthonormal basis $\{g_m\}_{m=1}^M$, the RVPINNs loss function can be expressed in the following equivalent form:

$$\mathcal{L}_{\text{RVPINNs}}(u_\theta) = \sqrt{\sum_{m=1}^{M} \left(\int_\Omega (f_0 - u_\theta + \alpha \mathcal{P} u_\theta) g_m d\mu \right)^2}. \tag{4.14}$$

The equivalence between these two forms is shown in C.2. It is worth noting that computing these loss functions requires numerical integration methods.

Remark 4.3.1. Using the bilinear form (4.7), we can reformulate the loss function (4.14) to eliminate the composition between the Perron-Frobenius operator $\mathcal{P}$ and the neural network u_θ. This reformulation avoids the need for the inverse map S^{-1} associated with the operator $\mathcal{P}$, and instead only requires the forward map S for

evaluating the Koopman operator $\mathcal{K}$ applied to the predefined test functions g_m. The resulting RVPINNs loss function is given by:

$$\mathcal{L}_{\text{RVPINNs}}(u_\theta) = \sqrt{\sum_{m=1}^{M} \left(\int_\Omega f_0 g_m - u_\theta (g_m - \alpha \mathcal{K} g_m) d\mu \right)^2}.$$

Remark 4.3.2. The loss functions in (4.9) and (4.12) are slightly different from the original PINNs and RVPINNs formulations (see Section 2.5), respectively. In particular, we do not square the residual norm in order to ensure consistency with the theoretical results presented in Section 4.4, which are based on the norm, instead of the square of the norm. While one can train based on these loss functions, one can improve training convergence by considering the square of the norm and implement a Least Squares system in the last layer of neural networks [103].

The following section presents a priori error estimates for PINNs and RVPINNs.

4.4 Analysis of methods

For the error analysis of the neural network approximations in this section, we use the concept of a quasi-minimizer (see Definition 2.6.1).

4.4.1 Error estimate for PINNs

Theorem 4.4.1. *Let $\delta_n > 0$ and $u_{\theta^*} \in \mathcal{M}_n$ be a quasi-minimizer of (4.9) satisfying (2.9). If u is the solution of (4.3), we have*

$$\|u - u_{\theta^*}\|_U \leq \left(\frac{1+\alpha}{1-\alpha} \right) \inf_{u_\theta \in \mathcal{M}_n} \|u - u_\theta\|_U + \frac{\delta_n}{1-\alpha}.$$

Proof. See details of the proof in C.3. □

4.4.2 Error estimate for RVPINNs

To obtain a reliable error bound for the RVPINNs method, we need the following assumption.

Local Fortin's condition[4]: Let $\delta_n > 0$ and $u_{\theta^*} \in \mathcal{M}_n$ be a quasi-minimizer

[4]Item 1 of this condition has been modified from Assumption 1 in [87], as we will derive an a priori error estimate for RVPINNs in terms of the infimum over $\mathcal{M}_n^{\theta^*,R}$, which is a more natural form compared to the infimum over span($\mathcal{M}_n^{\theta^*,R}$) used in [87].

of $\mathcal{L}_{\text{RVPINNs}}$ satisfying (2.9). There exists $R > 0$ such that for all $\theta \in B(\theta^*, R)$, an operator $\Pi_\theta : V \to V_M$ and θ-independent constant $C_\Pi > 0$, satisfying:

1. $b(u_{\theta^*} - u_\theta, v - \Pi_\theta v) = 0, \quad \forall v \in V,$

2. $\|\Pi_\theta v\|_V \leq C_\Pi \|v\|_V, \quad \forall v \in V,$

where $B(\theta^*, R)$ is an open ball of center θ^* and radius R, with respect to a given norm of $\mathbb{R}^n$. We denote $\mathcal{M}_n^{\theta^*,R} := \{u_\theta \in \mathcal{M}_n : \theta \in B(\theta^*, R)\}$.

Theorem 4.4.2. *Let $\delta_n > 0$ and $u_{\theta^*} \in \mathcal{M}_n$ be a quasi-minimizer of (4.12) satisfying (2.9). If the local Fortin's condition is satisfied, we have*

$$\|u - u_{\theta^*}\|_U \leq \left(1 + \frac{2(1+\alpha)C_\Pi}{1-\alpha}\right) \inf_{u_\theta \in \mathcal{M}_n^{\theta^*,R}} \|u - u_\theta\|_U + \left(\frac{C_\Pi}{1-\alpha}\right) \delta_n.$$

Proof. See details of the proof in C.3. □

Remark 4.4.1. The set of two-layer (one-hidden-layer) neural networks $\mathcal{M}_n$ with ReLU activation function is equivalent to the set of continuous piecewise-linear functions with free knots. Therefore, for smooth solutions in H^2, both fixed-grid and neural network approximations can achieve a convergence rate of $O(n^{-2})$, however for non-smooth solutions in H^s, fixed-grid-based methods have a limited rate of convergence $O(n^{-s})$ while free-knot splines can achieve the optimal rate $O(n^{-2})$, see e.g., [36].

Remark 4.4.2. Theorem 4.4.1 and 4.4.2 propose the approximation error of the neural network when a quasi-minimizer is attained. However, achieving such a quasi-minimizer in practice may be challenging: It requires solving non-convex optimization problems up to a desired tolerance, and would need nontrivial reliable stopping criteria for the training algorithm. This area of research has been largely unexplored.

4.5 Numerical examples

In this section, we show the performance of PINNs and RVPINNs for Neumann series problems by applying them to various 1D and 2D dynamical systems. We focus on systems where Perron-Frobenius operators are non-expansive under the L^2-norm. Throughout this work, we use the ReLU function ($\text{ReLU}(x) = \max(x, 0)$) as an activation function of neural networks, and hat functions as basis functions in the fixed-grid-based method, so both approaches generate piecewise-linear approximations.

4.5.1 1D dynamical systems

We consider the dynamical system described by the tent map, which is a simple, continuous, piecewise-linear function exhibiting chaotic behavior. The tent map $S : [0,1] \to [0,1]$ is defined as:

$$S(x) = \begin{cases} 2x, & x \in [0, \frac{1}{2}), \\ 2 - 2x, & x \in [\frac{1}{2}, 1]. \end{cases}$$

By applying equation (4.2), the Perron-Frobenius operator associated with the tent map can be derived in the form of

$$\mathcal{P}f(x) = \frac{1}{2} f\left(\frac{x}{2}\right) + \frac{1}{2} f\left(1 - \frac{x}{2}\right), \tag{4.15}$$

and we can verify that it is non-expansive under the L^2-norm. Indeed,

$$\begin{aligned} \|\mathcal{P}f\|_{L^2}^2 &= \int_0^1 \left(\frac{1}{2} f\left(\frac{x}{2}\right) + \frac{1}{2} f\left(1 - \frac{x}{2}\right)\right)^2 dx \\ &\leq \frac{1}{2} \int_0^1 f^2\left(\frac{x}{2}\right) + f^2\left(1 - \frac{x}{2}\right) dx = \int_0^1 f^2(x) dx = \|f\|_{L^2}^2. \end{aligned}$$

Because it is challenging to obtain analytical solutions of (4.3), we design our examples in reverse order: We specify a desired solution and then determine the corresponding initial density. This approach allows us to create well-defined test cases. For the Perron-Frobenius operator $\mathcal{P}$ given in (4.15), we implement PINNs and RVPINNs to approximate two types of solutions:

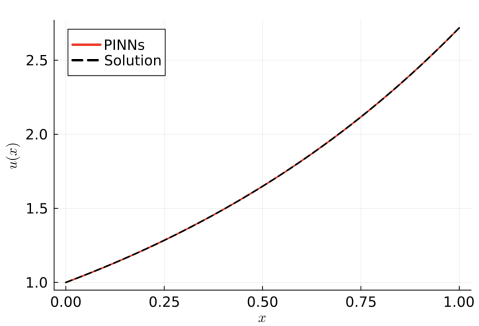
1. A smooth solution $u^{[1]}(x) = \exp(x)$,
2. A singular solution $u^{[2]}(x) = 1 + x^{-1/3}$.

To construct the initial densities $f_0^{[i]}$ corresponding to these solutions, we substitute the desired solution $u^{[i]}$ into (4.3) and derive the following initial densities:

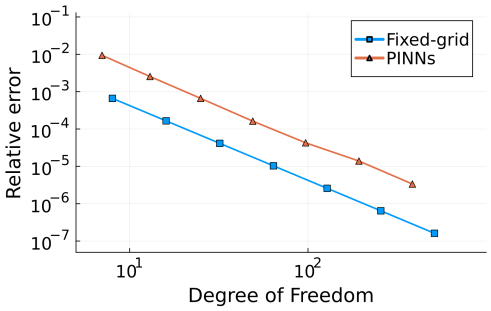
1. $f_0^{[1]}(x) = \exp(x) - \frac{\alpha}{2} \exp\left(\frac{x}{2}\right) - \frac{\alpha}{2} \exp\left(1 - \frac{x}{2}\right)$,
2. $f_0^{[2]}(x) = (1 - \alpha) + x^{-1/3} - \frac{\alpha}{2}\left(\frac{x}{2}\right)^{-1/3} - \frac{\alpha}{2}\left(1 - \frac{x}{2}\right)^{-1/3}$.

We note that $f_0^{[1]}$ and $f_0^{[2]}$ are nonnegative when $\alpha \leq 2/(1+e)$ and $\alpha \leq 2/(1+\sqrt[3]{2})$, respectively.

To show the performance of the PINNs approach, we apply it to approximate the solution $u^{[1]}$. In the loss function (4.9), we use the corresponding initial density $f_0^{[1]}$,



(a) PINNs approximation and solution



(b) Relative L^2-error v.s. degree of freedom

Figure 4.1: PINNs approximation for $u^{[1]}$ with $\alpha = 0.5$.

set the damping parameter to $\alpha = 0.5$, and use the Perron-Frobenius operator $\mathcal{P}$ as defined in (4.15). The neural network architecture consists of a single hidden layer with n neurons and the ReLU activation function, i.e.,

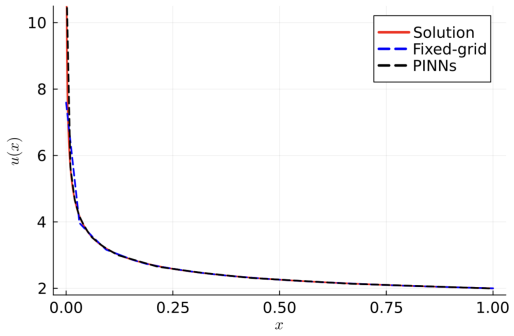
$$u_\theta(x) := \sum_{j=1}^{n} c_j \mathrm{ReLU}(w_j x + b_j) + c_0, \tag{4.16}$$

where $c_j, w_j, b_j \in \mathbb{R}$ are trainable parameters. The neural network parameters are not initialized randomly, but deterministically following [69, 70]. The initialized neural network represents a piecewise-linear approximation over a uniform partition of $[0, 1]$ into n subintervals, with the outer parameters c_j chosen to minimize the PINNs loss function (by solving the linear Galerkin system, see [69, Eq. (7.1)]). To approximate the integral in the loss function, we use a 101-point Gauss-Kronrod quadrature rule[5]. We train the neural network using the BFGS algorithm[6] to minimize the loss function.

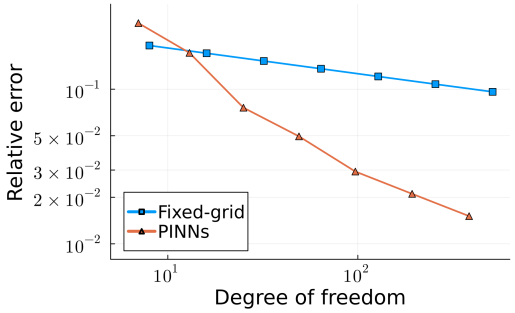
We note that the choice of optimization algorithm plays a crucial role in the training process. In particular, for the 1D examples considered here, a fixed-grid initialization is available, allowing the neural network parameters to be initialized near a minimizer. Therefore, we employ the BFGS method to achieve faster and more reliable convergence than ADAM.

[5]A 101-point Gauss–Kronrod quadrature rule is employed because it provides high-order accuracy for the integral. Numerical experiments showed that increasing the number of quadrature points beyond 101 produced negligible changes in the loss value. Thus, the choice of 101 points represents a compromise between quadrature accuracy and training cost.

[6]The BFGS solver is paired with the line search algorithm and the optimization terminates when the gradient infinity norm drops below 10^{-8}.



(a) PINNs approximation, fixed-grid approximation and solution.



(b) Relative L^2-error comparison. The corresponding experimental orders of convergence are reported in Table 4.1.

Figure 4.2: PINNs approximation for $u^{[2]}$ with $\alpha = 0.5$.

We compare PINNs with the fixed-grid-based method. In the latter approach, the domain is partitioned into n equal-length subintervals, and hat functions defined over these intervals are used as basis functions.

Figure 4.1 (a) shows the PINNs approximation using the neural network with $n = 32$ neurons, which presents a good approximation to the solution. Figure 4.1 (b) demonstrates the relative L^2-errors, $\|u - u_\theta\|_{L^2}/\|u\|_{L^2}$, for both PINNs and the fixed-grid-based method, plotted against the number of degrees of freedom ($3n + 1$ for a neural network with n neurons and n for the fixed-grid-based method with n basis functions). The slopes of the two curves are approximately -2, confirming that the convergence rate is $O(n^{-2})$, as expected for continuous piecewise-linear approximations (see Remark 4.2.6).

In addition, we consider the singular solution $u^{[2]}$. In the PINNs approach, we use an n-neuron neural network as defined in (4.16). The inner parameters w_j and b_j are initialized to generate breakpoints at $(k/n)^6$, for $k = 1, 2, \ldots, n$ while the outer parameters c_j are chosen to minimize the PINNs loss function for the given inner parameters. Since the target function is singular, we use a finer quadrature rule with 1001 points to improve integration accuracy.

As an example, Figure 4.2 (a) shows PINNs with 8 neurons (25 degrees of freedom) and the fixed-grid-based approximation using 32 intervals. Furthermore, Figure 4.2 (b) illustrates that the relative L^2-errors for PINNs are smaller than that of the fixed-grid-based method once the number of degrees of freedom is greater than 10,

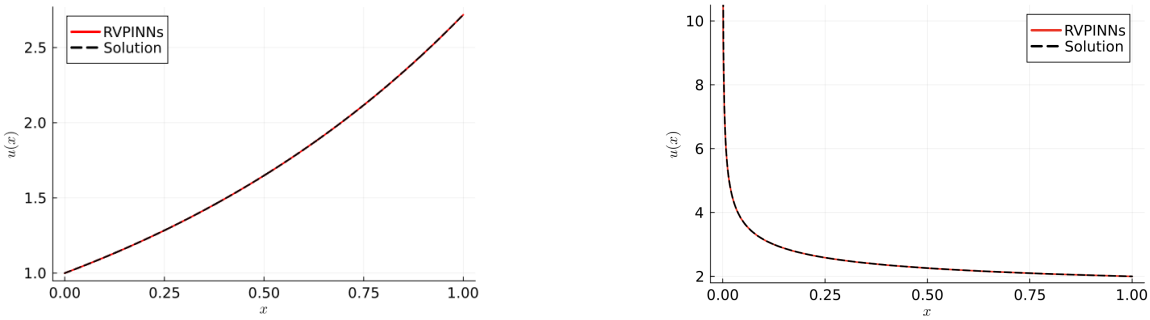
Degree of freedom	Fixed-grid	PINNs
$7-13$	0.164	0.715
$13-25$	0.166	1.249
$25-49$	0.167	0.634
$49-97$	0.167	0.767
$97-193$	0.167	0.483
$193-385$	0.167	0.480

Table 4.1: Experimental orders of convergence of the fixed-grid-based method and PINNs for different numbers of degrees of freedom.

highlighting their superior approximation capability. Table 4.1 reports the experimental order of convergence (EOC) corresponding to Figure 4.2 (b). The EOC for the fixed-grid-based method is approximately 1/6, consistent with the theoretical convergence rate $O(n^{-1/6})$ for a uniform grid, based on a Sobolev embedding argument as discussed in [76, Example 5.8]. Additionally, as expected for a neural network method, the EOC for PINNs shows a higher convergence rate.

In an idealized setting with perfect training, one would expect a convergence rate of order 2, which is optimal for free-knot piecewise-linear approximations of $u^{[2]}$ [36, p. 104]. In practice, however, achieving this rate requires the breakpoints to cluster extremely close to the singularity, which is challenging since numerical round-off errors can make the neural network unstable. As a result, while standard training procedures can yield improved convergence rates compared with the fixed-grid-based method, they are typically insufficient to achieve the optimal $O(n^{-2})$ rate in this example. We refer to the recent work [22] where special training algorithms are proposed involving adaptive neuron strategies and which can further increase the rate towards the optimal one.

The RVPINNs method provides an alternative approach for approximating solutions to Neumann series problems. We use the same neural network structure as the PINNs examples. The domain is partitioned into eight equal-length intervals, with the test functions chosen as normalized characteristic functions over these intervals. Figures 4.3 (a) and 4.3 (b) illustrate that the trained 32-neuron neural networks, using the RVPINNs method, generate accurate approximations for both examples.



(a) RVPINNs approximation for $u^{[1]}$

(b) RVPINNs approximation for $u^{[2]}$

Figure 4.3: RVPINNs approximation for the tent map example.

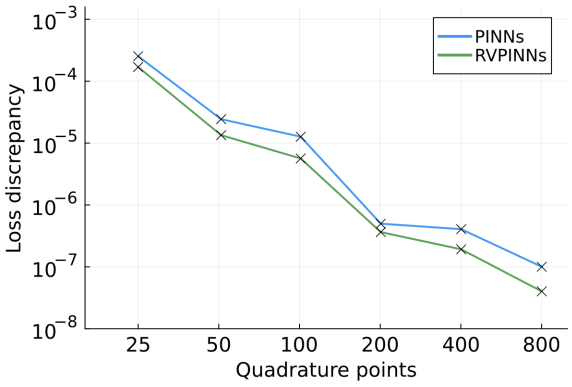


Figure 4.4: The discrepancy between the loss function computed using different numbers of quadrature points and the true value of the loss function for $u^{[1]}$.

Since our implementation relies on numerical integration using the Gauss-Kronrod quadrature rule, we examine the discrepancy between the loss function computed with q quadrature points and that obtained using the adaptive Gauss-Kronrod quadrature rule, which we treat as the exact value (with an error tolerance below 10^{-8}). As shown in Figure 4.4, the approximation error in the loss function decreases as the number of quadrature points increases.

Remark 4.5.1. A comparison between neural network methods and fixed-grid-based methods using higher-order polynomials would require a different activation function (e.g., ReLU^2) to produce higher-degree piecewise polynomial approximations, and is beyond the scope of this work.

Remark 4.5.2. The training of PINNs and RVPINNs involves solving a nonlinear optimization problem, whose computational cost is typically higher than that of the fixed-grid-based approach. In practice, a trade-off between accuracy and computa-

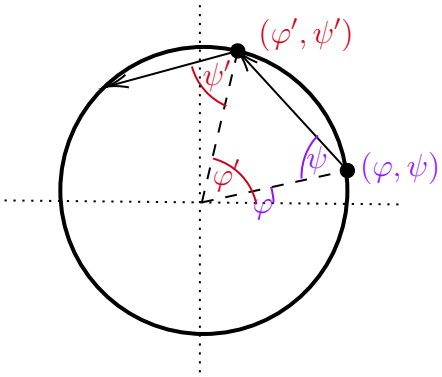


Figure 4.5: Diagram of a ray trajectory in a circular domain. The boundary map S_1 sends (φ, ψ) to the new ray coordinates (φ', ψ').

tional time should be considered when applying the proposed methods. Nevertheless, a comparison of computational times is out of the scope of this work, as neither approach has been optimized for computational efficiency.

4.5.2 2D dynamical systems

In this section, we show the performance of PINNs and RVPINNs on 2D examples. We consider the boundary map in a circular domain and the standard map. We compare the numerical results with the N-term truncated sum, used as the "exact" Neumann series u in (4.4). The L^2-error of the truncated sum is bounded by $\alpha^{N+1}\|f_0\|_{L^2}/(1-\alpha)$.

In a circular domain, the position and direction of rays can be described using coordinates (φ, ψ), where $\varphi \in [0, 2\pi)$ represents the position on the boundary, given by the polar angle, and $\psi \in (-\frac{\pi}{2}, \frac{\pi}{2})$ denotes the direction of the reflected ray, measured as the angle between the inward-pointing normal vector and the outgoing ray (see Figure 4.5). Define the phase space as $\Omega := [0, 2\pi) \times (-\frac{\pi}{2}, \frac{\pi}{2})$. The boundary map $S_1 : \Omega \to \Omega$ giving the coordinates of the next intersection is defined by

$$S_1(\varphi, \psi) = (\varphi + \pi - 2\psi \mod 2\pi, \psi),$$

and its inverse map is

$$S_1^{-1}(\varphi, \psi) = (\varphi - \pi + 2\psi \mod 2\pi, \psi).$$

The standard map is a mathematical model used to study chaotic behavior in dynamical systems. It is a discrete-time dynamical system that exhibits a transition

from regular to chaotic motion, making it a popular example in the study of chaos theory. The standard map S_2 is a mapping from $[0, 2\pi) \times [0, 2\pi)$ to itself defined by

$$S_2(\theta, p) = (\theta + p + K\sin(\theta) \mod 2\pi, p + K\sin(\theta) \mod 2\pi),$$

and its inverse map is

$$S_2^{-1}(\theta, p) = (\theta - p \mod 2\pi, p - K\sin(\theta - p) \mod 2\pi),$$

where K is a parameter that controls the amount of chaos in the system. In this work, we use $K = 2.4$.

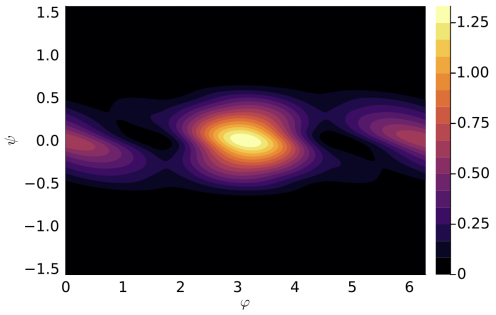
Note that the boundary map and the standard map satisfy Remark 4.2.2 with $\left|J_{S_i^{-1}}\right| = 1$, so the Perron-Frobenius operators associated with S_i are in the form $\mathcal{P}f = f \circ S_i^{-1}$.

We implement PINNs and RVPINNs to approximate the solution of (4.3) and (4.5), respectively, for the Perron-Frobenius operator associated with the boundary map. We set the damping parameter $\alpha = 0.5$. The initial density is given by $f_0(\varphi, \psi) = \cos^2(\varphi)\cos^2(2\psi)$ for $\pi/2 < \varphi < 3\pi/2$ and $-\pi/4 < \psi < \pi/4$, and 0 elsewhere. Since f_0 is bounded and compactly supported, we have $f_0 \in L^2(\Omega)$. For RVPINNs, we partition the domain into 8×8 equal-sized cells, denoted as $\omega_1, \ldots, \omega_{64}$, and define g_m as the normalized characteristic function on ω_m. We use a two-layer (one-hidden-layer) neural network, 32 neurons, and the ReLU activation function, i.e.,

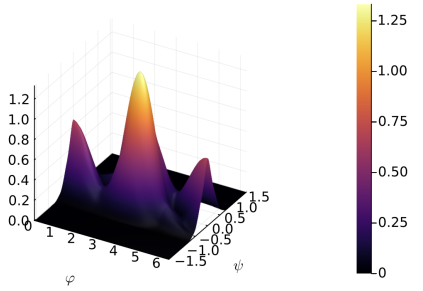
$$u_\theta(x, y) := \sum_{j=1}^{32} c_j \mathrm{ReLU}(w_j^{(1)} x + w_j^{(2)} y + b_j) + c_0,$$

where $c_j, w_j^{(1)}, w_j^{(2)}, b_j \in \mathbb{R}$ are trainable parameters. We approximate double integrals in PINNs and RVPINNs loss functions using a 101-point Gauss-Kronrod quadrature rule for each variable. After training the neural network using Adam optimizer[7], we obtain approximations that closely match the truncated sum of (4.4) with 1,000 terms ("exact" solution), as shown in Figures 4.6 and 4.7. Despite using a relatively small neural network, it achieves a good approximation. As shown in Figure 4.8, the absolute errors remain below 0.2 for PINNs and 0.25 for RVPINNs throughout the domain.

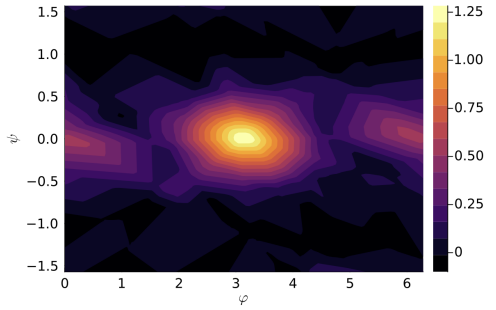
[7]The Adam optimizer is configured with an initial learning rate of $\eta = 0.001$, and exponential decay rates of $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Training is run for up to 10,000 and 50,000 epochs for PINNs and RVPINNs, respectively.



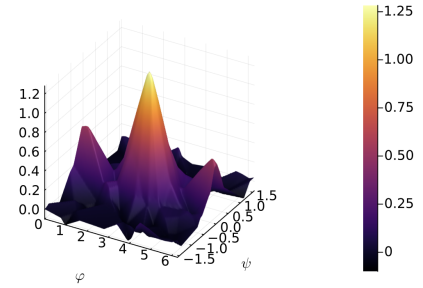
(a) 2D plot of the truncated sum with 1,000 terms



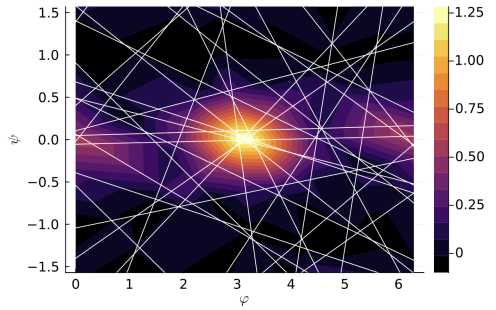
(b) 3D plot of the truncated sum with 1,000 terms



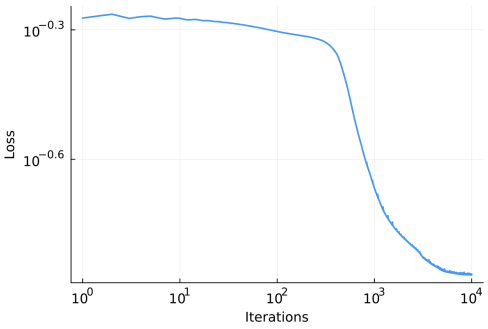
(c) 2D plot of PINNs approximation



(d) 3D plot of PINNs approximation

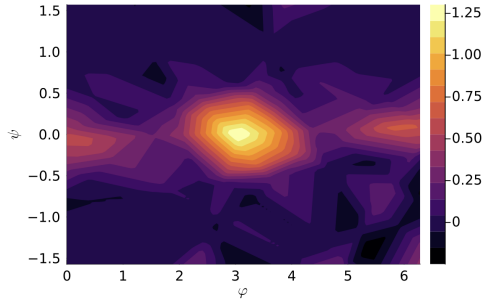


(e) 2D plot of PINNs approximation with straight lines representing the affine transformations in the hidden layer

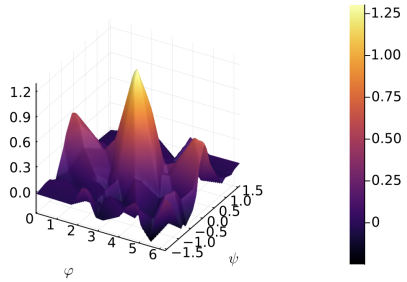


(f) Loss plot

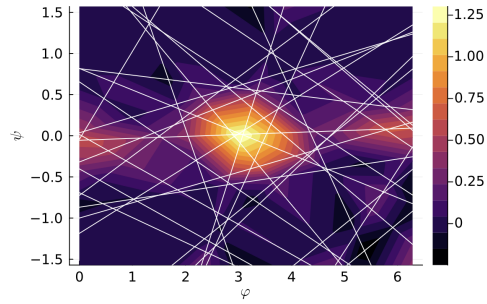
Figure 4.6: PINNs approximation for the boundary map example with $\alpha = 0.5$.



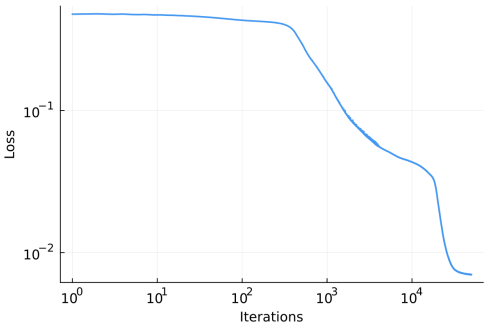
(a) 2D plot of RVPINNs approximation



(b) 3D plot of RVPINNs approximation

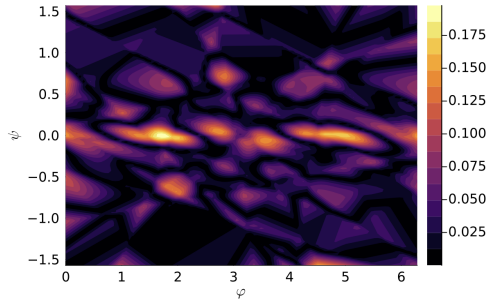


(c) 2D plot of RVPINNs approximation with straight lines representing the affine transformations in the hidden layer

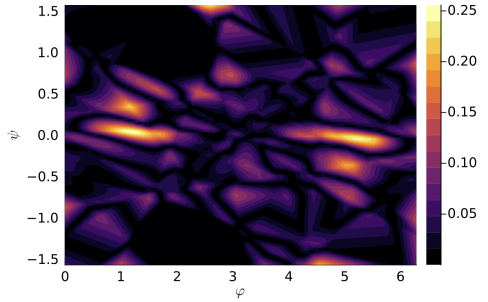


(d) Loss plot

Figure 4.7: RVPINNs approximation for the boundary map example.



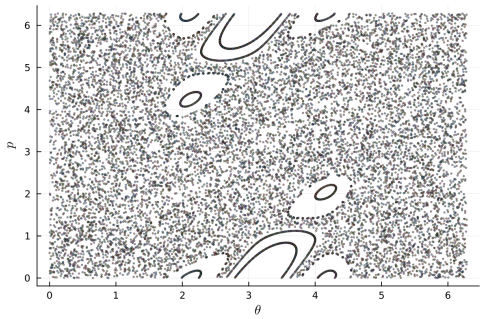
(a) Pointwise absolute error of PINNs



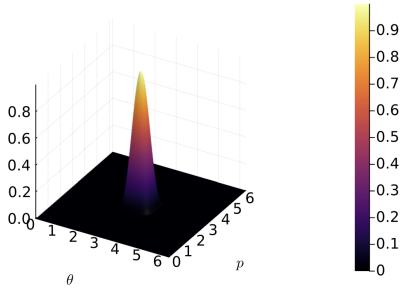
(b) Pointwise absolute error of RVPINNs

Figure 4.8: Pointwise absolute errors for the boundary map example.

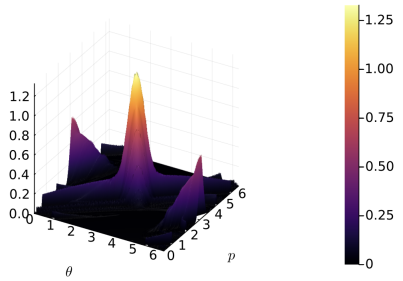
In the 2D setting and higher-dimensional examples that follow, we adopt the Adam optimizer due to its memory efficiency when handling a large number of parameters. In addition, its adaptive momentum helps avoiding entrapment in local minima. For better results, one may consider more advanced optimization techniques or hybrid approaches that combine multiple optimizers (see [102] for example).



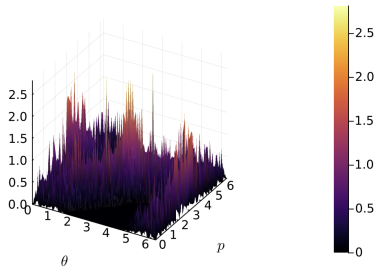
(a) Classical phase space



(b) Initial density



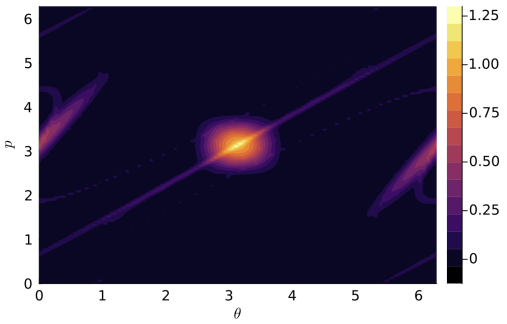
(c) $\alpha = 0.5$



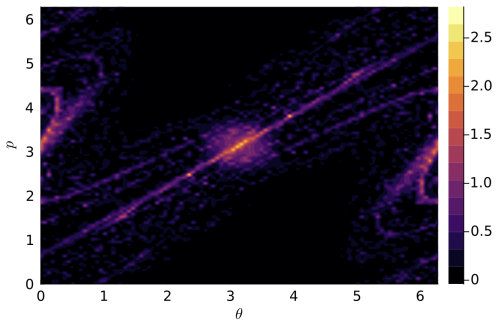
(d) $\alpha = 0.9$

Figure 4.9: (a) A classical phase space showing the distribution of trajectories of the standard map. (b) Initial density. (c) - (d) The truncated sums with 1,000 terms using $\alpha = 0.5$ and $\alpha = 0.9$, respectively.

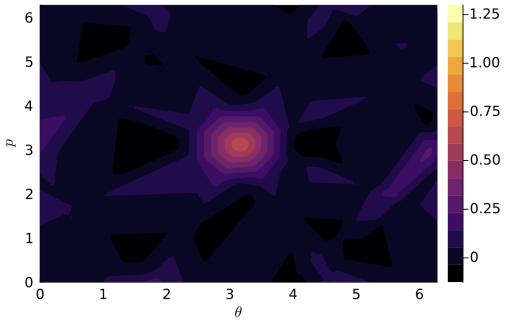
We also apply PINNs and RVPINNs to the standard map example, where the initial density is given by $f_0(\theta, p) = \cos^2(2\theta)\cos^2(2p)$ for $3\pi/4 < \theta < 5\pi/4$ and $3\pi/4 < p < 5\pi/4$, and 0 elsewhere. Because f_0 is bounded with compact support, we have $f_0 \in L^2(\Omega)$. As α increases, Neumann series for the standard map example become more complex, as shown in Figure 4.9. To make neural networks converge to complex solutions faster, we employ a continuation technique: training the neural network to approximate solutions for small α, then using the resulting model as an initialization for training with larger α. We train neural networks following the same approach described above. For the three-layer architecture, each hidden layer contains 32 neurons with the ReLU activation function. Figure 4.10 shows the approximate solutions obtained with PINNs for different values of α. Similar results were observed for RVPINNs, which are therefore omitted for brevity. These results demonstrate that PINNs and RVPINNs can capture the key features of solutions. Moreover, deeper neural networks yield better results due to their greater expressivity.



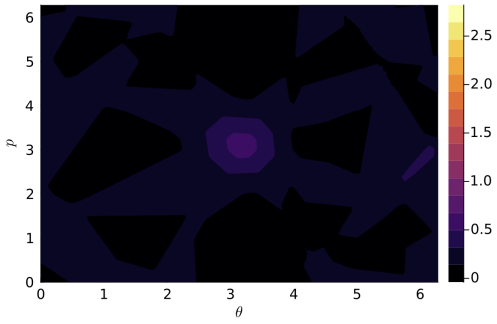
(a) Truncated sum



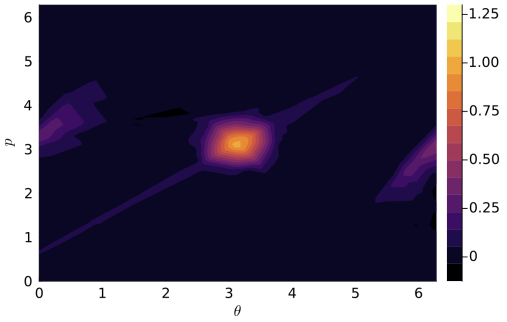
(b) Truncated sum



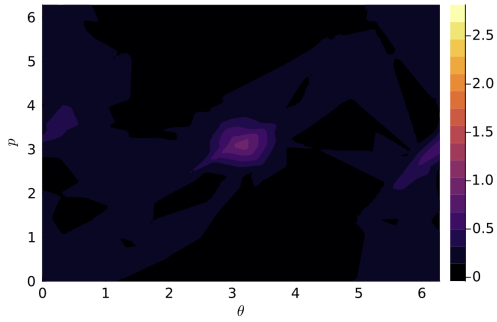
(c) Two-layer neural network



(d) Two-layer neural network



(e) Three-layer neural network



(f) Three-layer neural network

Figure 4.10: Solutions for the standard map example with $\alpha = 0.5$ (left) and $\alpha = 0.9$ (right) computed by the truncated sum with 1,000 terms and PINNs approximations.

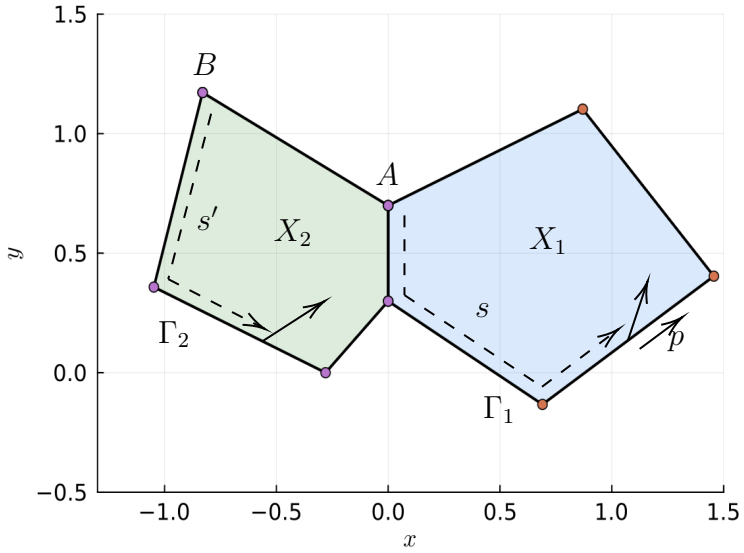


Figure 4.11: Two-cavity system: Γ_1 and Γ_2 are the boundaries of pentagons X_1 and X_2, respectively. When a ray lies in X_1, its arclength s is measured anticlockwise from point A along the boundary Γ_1. When a ray lies in X_2, its arclength s is defined as $s = s' + |\Gamma_1|$, where s' is the anticlockwise distance from point B along the boundary Γ_2 (cf. [26]).

4.5.3 Application: Interior densities in a two-cavity system

Our proposed methods generate approximate functions for stationary boundary densities when S is a boundary map. These boundary densities can then be used to compute the corresponding interior densities. In this section, we demonstrate applications of PINNs for approximating a stationary density within a complex domain. Specifically, we consider a two-cavity system as considered in [4, 26].

The phase-space coordinates on the boundary are denoted by (s, p), where s represents the arclength along the boundary and p is the tangential component of the momentum vector (See Figure 4.11). Let $\Omega = (\Gamma_1 \cup \Gamma_2) \times (-1, 1)$ be the phase space, and define the associated boundary map as a billiard map $S : \Omega \to \Omega$. It is well known that the billiard map is invertible almost everywhere, except at corner points where the reflection is not defined. Moreover, it is a Hamiltonian system [33], which implies that it preserves phase-space volume; equivalently, its Jacobian determinant is equal to one. Using Remark 4.2.2, the Perron-Frobenius operator can be written as $\mathcal{P}f = f \circ S^{-1}$.

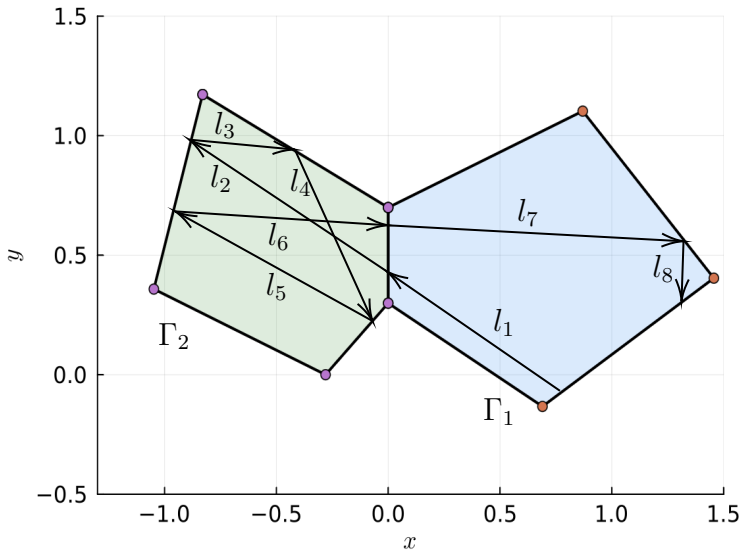


Figure 4.12: An example of a ray trajectory in the two-cavity system. Its trajectory length is $l = l_1 + l_2 + \ldots + l_8$.

Assume that the ray source is located on the third boundary segment of Γ_1. The corresponding initial boundary density is defined as $f_0(s,p) = (1-p^2)\sin(\pi(s - 1.214)/0.936)$ for $1.214 < s < 2.150$ and $-1 < p < 1$, and 0 elsewhere. The damping along each trajectory is modeled by the function $\alpha(l) = \exp(-0.5l)$, where l denotes the trajectory length (see Figure 4.12).

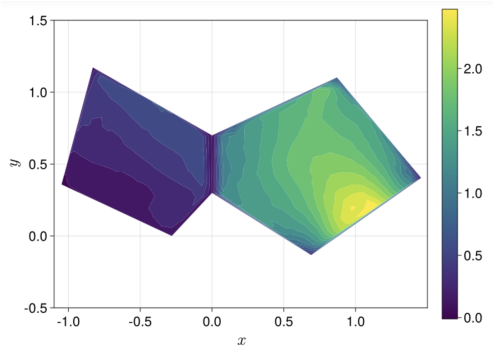
Once the stationary boundary density $u(s,p)$ is obtained, the corresponding interior density can be computed by projecting u onto the interior domain X_i via the mapping[8]

$$u_{X_i}(r) = \int_{\Gamma_i} u(s,p_s)\frac{\cos(\nu(r_s,r))}{|r-r_s|}ds, \tag{4.17}$$

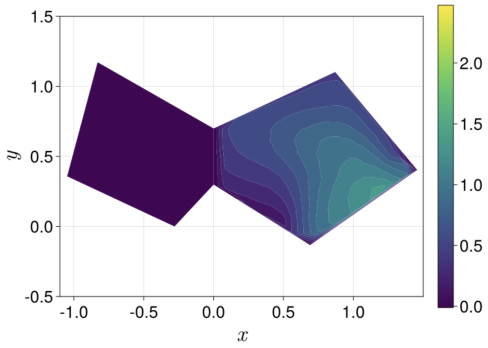
where $r \in X_i$ denotes a point inside the cavity (in Cartesian coordinates), r_s is the Cartesian coordinate corresponding to the boundary point $s \in \Gamma_i$, and $\nu(r_s,r)$ is the angle between the inward-pointing normal vector at s and the direction vector from r_s to r.

For the numerical results, Figure 4.13 shows the approximate stationary interior densities computed using (4.17), where u denotes the stationary boundary densities obtained from the truncated sum ("exact" solution), PINNs with two-layer and three-layer architectures, and the fixed-grid-based method. For PINNs, each hidden layer of

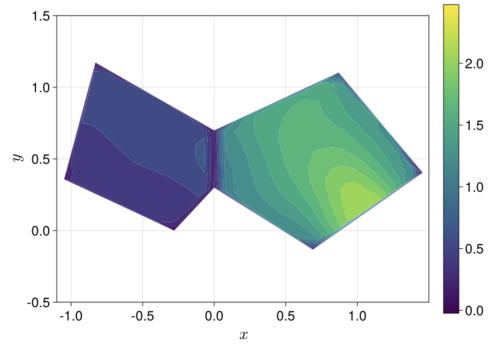
[8]In this formulation, damping is not included in the equation, as the projection is intended only for visualization.



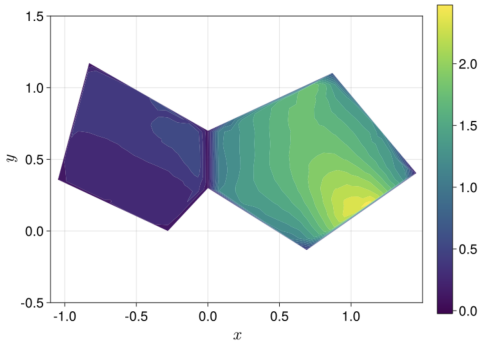
(a) Truncated sum



(b) Fixed-grid-based method



(c) PINNs with two-layer neural network



(d) PINNs with three-layer neural network

Figure 4.13: Approximate stationary interior densities of the two-cavity system.

the neural networks contains 64 neurons with the ReLU activation function. For the fixed-grid-based method, we use piecewise linear "hat" functions defined on an 16×16 uniform grid as basis functions. This is an intended choice to compare this method with the two-layer (one-hidden-layer) neural network in terms of nearly identical degrees of freedom (256 and 257 parameters, respectively). In both approaches, we generate continuous, piecewise linear approximations.

As shown in Figure 4.13, the fixed-grid-based method cannot give fine details in the density distribution within the left pentagon, and the densities in the right pentagon exhibit noticeable deviations from the solution. In contrast, the two-layer neural network captures the main feature of the solution compared to the truncated sum. The three-layer neural network provides a more accurate approximation than the two-layer architecture. Overall, PINNs outperform the fixed-grid-based method.

4.5.4 High-dimensional problems

A primary advantage of neural network methods is their applicability to solve high-dimensional problems, where traditional Galerkin-type methods become impractical. To demonstrate this, we consider a system of N-coupled standard maps. The forward dynamics for $n = 1, \ldots, N$ (with $x_{N+1} = x_1$) are defined by [71],

$$p'_n = p_n + \frac{K}{2\pi}\sin 2\pi x_n + \frac{\beta}{2\pi}[\sin 2\pi(x_{n+1} - x_n) + \sin 2\pi(x_{n-1} - x_n)] \qquad \text{mod } 1$$
$$x'_n = x_n + p'_n \qquad \text{mod } 1$$

The corresponding inverse map, $S_N^{-1} : [0,1]^{2N} \to [0,1]^{2N}$, is defined by

$$S_N^{-1}(x'_1, p'_1, \ldots, x'_N, p'_N) := (x_1, p_1, \ldots, x_N, p_N),$$

where

$$x_n = x'_n - p'_n \qquad \text{mod } 1$$
$$p_n = p'_n - \frac{K}{2\pi}\sin 2\pi x_n - \frac{\beta}{2\pi}[\sin 2\pi(x_{n+1} - x_n) + \sin 2\pi(x_{n-1} - x_n)] \qquad \text{mod } 1.$$

The N-coupled standard map belongs to the class of symplectic coupled map lattices. As a consequence of its symplectic structure, the map is volume-preserving, and the determinant of its Jacobian matrix is 1 [34]. Consequently, by Remark 4.2.2, the Perron-Frobenius operator is in the form $Pf = f \circ S_N^{-1}$.

In our numerical experiments, we fix the parameters at $K = 2.4$ and $\beta = 0.1$. To validate the method across varying dimensions ($N = 1, 2, 3, 4, 5, 10$), we set a target solution

$$u(x_1, p_1, \ldots, x_N, p_N) = 1 + \prod_{n=1}^{N} \sin(\pi x_n)\sin(\pi p_n).$$

The corresponding initial density is derived as $f_0 = u - \alpha u(S_N^{-1})$.

We employ PINNs approach to solve (4.3) with $\alpha = 0.5$. The neural networks consist of fully connected layers with 128 neurons per layer and use the ReLU activation function. The integral appearing in the loss function is approximated via 10,000-point Monte Carlo integration[9], and the training process is carried out using the Adam optimizer[10] for 200,000 epochs.

[9] In high dimensions, the computational cost of Gauss–Kronrod quadrature grows exponentially, whereas Monte Carlo's convergence rate is dimension-independent. Therefore, Monte Carlo provides a better accuracy-to-cost trade-off for high-dimensional PINNs.

[10] The Adam optimizer is initialized with exponential decay rates of $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The learning rate is set to $\eta = 0.001$ and reduced by a factor of 0.95 every 2,000 epochs.

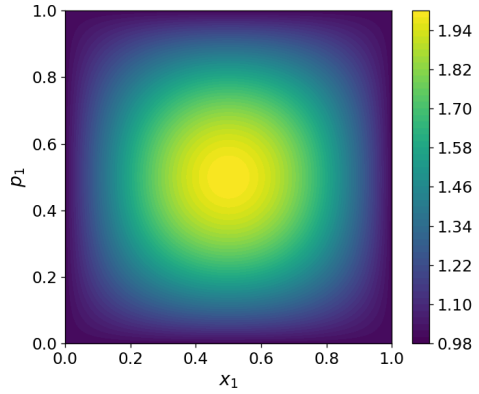
Dimension ($2N$)	# Layers	Degree of freedom	Relative L^2-error
2	5	50,049	4.728×10^{-4}
4	6	66,817	1.765×10^{-3}
6	7	83,585	1.457×10^{-3}
8	8	100,353	1.226×10^{-3}
10	9	117,121	8.307×10^{-4}
20	22	333,057	1.382×10^{-4}

Table 4.2: Relative L^2-error for high-dimensional problems.

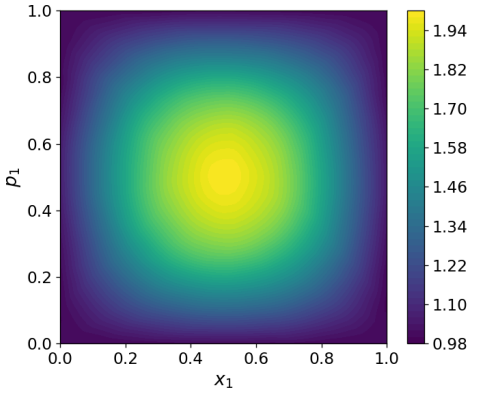
Table 4.2 reports the relative L^2-error, $\|u - u_\theta\|_{L^2}/\|u\|_{L^2}$, evaluated using a Monte Carlo quadrature with 1,000,000 samples, together with details on the number of neural network layers and degrees of freedom. The results indicate that neural networks achieve approximation error below 1% for high-dimensional problems. For visualization, Figure 4.14 shows the first two components of the neural network approximation for the 5-coupled standard map and compares them with the exact solution, along with the associated absolute error.

4.6 Conclusions

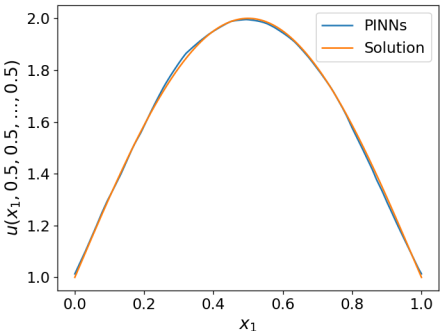
In this chapter, we investigated Neumann series problems associated with non-expansive Perron-Frobenius operators under the L^p-norm, with a focus on the L^2-setting. We established the well-posedness of the corresponding variational formulations in L^2 by showing that they satisfy the conditions of the Lax-Milgram theorem, thereby ensuring the existence and uniqueness of the solution. We proposed two neural network methods for approximating solutions of Neumann series problems. PINNs were employed to approximate solutions in their strong form, while RVPINNs were used to approximate solutions in the corresponding variational problems. Additionally, we provided error estimates for both methods in terms of quasi-minimizers. Numerical experiments demonstrated that both methods are effective for solving Neumann series problems and outperform the standard fixed-grid-based method in terms of approximation error they can reached under the same number of degrees of freedom. Furthermore, we applied PINNs to approximate solutions of Neumann series problems in high dimensional settings.



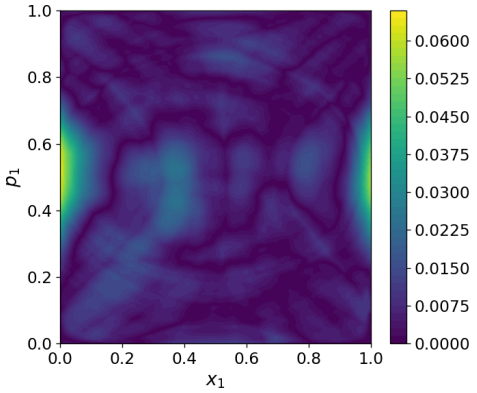
(a) $u(x_1, p_1, 0.5, \ldots, 0.5)$



(b) $u_\theta(x_1, p_1, 0.5, \ldots, 0.5)$



(c) $u_\theta(x_1, 0.5, 0.5, \ldots, 0.5)$ and $u(x_1, 0.5, 0.5, \ldots, 0.5)$



(d) Absolute error

Figure 4.14: Exact solution u, PINNs approximation u_θ, and absolute error $|u - u_\theta|$ for the 10-dimensional example.

Further extensions of this work include comparisons between our methods, using different activation functions, and alternative approaches such as Galerkin methods with quadratic polynomials or spectral methods based on Chebyshev series. In addition, the theoretical results in this chapter, such as a priori error estimates based on the local Fortin's condition, could be applied to establish reliable error bounds for solving PDEs with neural networks.

Chapter 5

Neural Network Methods for Perron-Frobenius Invariant Densities

5.1 Introduction

The study of dynamical systems often focuses on the evolution of probability distributions rather than individual trajectories. Fundamental to this statistical perspective is the Perron-Frobenius operator, a linear operator that describes how an initial density is pushed forward by the system's map. The invariant density (the fixed point of this operator) is an important concept, as it characterizes the long-term, steady-state behavior of the system and allows for the calculation of physical observables through ergodic averages [92].

In Chapter 3, we explored several classical numerical methods for approximating these invariant densities, such as Ulam's method [61] and the maximum entropy method (MEM) [37,39]. While these techniques are effective for simple systems, they often rely on a fixed discretization of the state space or fixed solution form based on the optimization constraints. These methods may require a very fine mesh to capture sharp peaks or singularity in the density.

To address these limitations, neural networks offer a highly flexible alternative [50] for function approximation. Even in one dimension, the mesh-free nature of neural networks allows them to adaptively capture the geometry of the invariant density.

As universal approximators [65], they can represent complex, non-linear features with sufficient parameters. Their differentiability also allows us to incorporate the operator equations or constraints into the learning process, moving toward a physics-informed approximation.

The main contributions of this chapter are as follows. First, we introduce three frameworks that utilize neural network architectures to approximate one-dimensional Perron-Frobenius invariant densities. We provide a detailed construction of specific loss functions designed to enforce the constraints from the MEM. Furthermore, we derive the analytical gradients for a two-layer neural network with ReLU activation functions to enhance computational efficiency. Finally, we demonstrate the efficacy of our method through a series of 1D benchmarks, highlighting its ability to approximate invariant densities where traditional discretization techniques might be inefficient.

The remainder of this chapter is organized as follows. In Section 5.2, we introduce three neural network-based approaches for approximating invariant densities: the first is based on the MEM, the second replaces the entropy term with a quadratic term, and the third omits the entropy term entirely. Within this section, we derive the analytical gradients for the resulting loss functions and discuss a strategy for initializing weights and biases in two-layer neural networks using ReLU activation functions. In Section 5.3, we apply the proposed methodologies to several 1D dynamical systems. We investigate the sensitivity of the model's accuracy to the number of constraints. In addition, we compare the performance of our methods against the MEM and Ulam's method. Furthermore, we demonstrate the computational advantages of using analytical gradients (exact integration) by comparing training durations against those required using Gauss-Kronrod integration. Finally, Section 5.4 summarizes our findings and outlines possible directions for future research.

5.2 Methods for finding invariant densities

In this section, we propose neural network methods based on three different loss functions to approximate invariant densities. Throughout this chapter, we restrict our study to one-dimensional problems on $\Omega = [0, 1]$, where we can exploit analytical gradients of the loss functions to improve accuracy and training speed. Extending this

framework to higher-dimensional problems presents a promising direction for further research.

5.2.1 Approach 1: deep maximum entropy method (DMEM)

To develop a deep learning methodology for finding invariant densities, we represent the density using a neural network u_θ, assuming that $u_\theta(x) > 0$ for all x in the domain. We then transform the constrained optimization problem (3.23) - (3.25) into the minimization of the following loss function:

$$\mathcal{L}(u_\theta) := \left(\int_0^1 u_\theta(x)dx - 1\right)^2 + \sum_{k=1}^{N_c}\left(\int_0^1 [p_k(x) - p_k(S(x))]u_\theta(x)dx\right)^2 + \lambda \int_0^1 u_\theta(x)\ln u_\theta(x)dx, \tag{5.1}$$

where $\lambda > 0$ is a regularization parameter and p_k denotes the Chebyshev polynomial of degree k. In this formulation, the first term ensures that the neural network satisfies the unit norm condition (3.24), the second term incorporates the N_c constraints of the MEM (3.25), and the final term seeks to minimize the negative entropy, which is equivalent to maximizing the entropy in (3.23).

Note that the loss function defined in Equation (5.1) has some practical challenges:

1. The output of the neural network u_θ must remain positive across the entire domain throughout training, as the entropy term involves a logarithm. This restricts the choice of neural network architectures and activation functions.

2. Calculating the exact gradient of the loss function is difficult, even for shallow neural networks, due to the presence of the entropy term.

To address these challenges, we propose two alternative strategies:

1. Replacing the entropy functional with a quadratic energy term.

2. Omitting the entropy term to simplify the loss function.

The details of these approaches are discussed in the following sections.

5.2.2 Approach 2: quadratic energy minimization

To address the challenges posed by the logarithmic term in the entropy function, we approximate the negative entropy with a quadratic integral. This approach is justified by the Taylor expansion of the entropy integrand. Let $g(f) := f \ln f$. We compute its first two derivatives:

$$g'(f) = 1 + \ln f \quad \text{and} \quad g''(f) = \frac{1}{f}.$$

We then construct the second-order Taylor polynomials for $g(f)$ around $f = 1$:

$$g(f) \approx g(1) + g'(1)(f-1) + \frac{1}{2}g''(1)(f-1)^2 = 0 + (f-1) + \frac{1}{2}(f-1)^2 = \frac{1}{2}f^2 - \frac{1}{2}.$$

This shows that minimizing the entropy functional $\int_0^1 f(x) \ln f(x) dx$ is approximately equivalent to minimizing the quadratic energy functional $\int_0^1 [f(x)]^2 dx$, provided that $f(x)$ is close to the uniform distribution (the constant function $f = 1$). However, if the density f deviates significantly from this constant function, the quadratic term cannot effectively replace the entropy.

In this approach, we replace the entropy term in the loss function (5.1) with a quadratic energy term. The modified loss function is defined as:

$$\begin{aligned}\mathcal{L}(u_\theta) := &\left(\int_0^1 u_\theta(x)dx - 1\right)^2 + \sum_{k=1}^{N_c}\left(\int_0^1 [p_k(x) - p_k(S(x))]u_\theta(x)dx\right)^2 \\ &+ \lambda \int_0^1 (u_\theta(x))^2 \, dx \qquad (5.2)\\ =: &\, \mathcal{L}_1(u_\theta) + \sum_{k=1}^{N_c} \mathcal{L}_{2,k}(u_\theta) + \lambda \mathcal{L}_3(u_\theta). \qquad (5.3)\end{aligned}$$

A significant advantage of this approach is that it removes the positive constraint required by the logarithm, allowing for flexibility in the neural network architecture.

Analytical gradients

Since a two-layer (single-hidden-layer) neural network with a non-polynomial activation function can approximate any continuous function [65], we employ a shallow neural network with N_n neurons and the ReLU activation function:

$$u_\theta(x) := \sum_{j=1}^{N_n} c^j \mathrm{ReLU}(w^j x + b^j) =: \sum_{j=1}^{N_n} c^j \phi^j(x), \qquad (5.4)$$

where $\theta := \{c^j, w^j, b^j : j = 1, 2, \ldots, N_n\}$ denotes the set of trainable parameters. Assuming that $p_k \circ S$ is integrable for $k = 1, 2, \ldots, N_c$, the piecewise linear nature of the ReLU function allows us to compute the exact gradient of the loss function.

To compute the integral of the neural network u_θ, we first consider the individual hidden units $\phi^i(x) = \text{ReLU}(w^i x + b^i)$. This unit can be written explicitly as:

$$\text{ReLU}(w^i x + b^i) = \begin{cases} w^i x + b^i, & w^i x + b^i > 0 \\ 0, & w^i x + b^i \leq 0. \end{cases} \tag{5.5}$$

The integral of (5.5) and its gradients with respect to the weights and biases are analytically tractable:

$$\begin{aligned} \int_0^1 \phi^i(x)dx &= \int_p^q w^i x + b^i dx = \left[\frac{w^i x^2}{2} + b^i x\right]_p^q = \frac{w^i}{2}(q^2 - p^2) + b^i(q - p) \\ \frac{\partial}{\partial w^i}\int_0^1 \phi^i(x)dx &= \frac{q^2 - p^2}{2} \\ \frac{\partial}{\partial b^i}\int_0^1 \phi^i(x)dx &= q - p, \end{aligned}$$

where the integration limits p and q represent the intersection of the support of the ReLU unit with the unit interval $[0, 1]$:

$$p = \begin{cases} \min[\max(-b^i/w^i, 0), 1], & w^i > 0 \\ 0, & w^i \leq 0 \end{cases} \tag{5.6}$$

$$q = \begin{cases} 1, & w^i \geq 0 \\ \max[\min(-b^i/w^i, 1), 0], & w^i < 0. \end{cases} \tag{5.7}$$

Given the linear structure of u_θ in the form of (5.4), we can compute its integral

$$\int_0^1 u_\theta(x)dx = \sum_{j=1}^{N_n} c^j \int_0^1 \phi^j(x)dx.$$

We then derive the exact expressions for the gradients of each component of the loss function defined in Equation (5.3).

First term

The gradients of the first loss term $\mathcal{L}_1$ are calculated via the chain rule:

$$\begin{aligned}
\frac{\partial}{\partial c^i}\mathcal{L}_1 &= 2\left(\int_0^1 u_\theta(x)dx - 1\right)\int_0^1 \frac{\partial}{\partial c^i}u_\theta(x)dx \\
&= 2\left(\int_0^1 u_\theta(x)dx - 1\right)\int_0^1 \phi^i(x)dx \\
\frac{\partial}{\partial w^i}\mathcal{L}_1 &= 2\left(\int_0^1 u_\theta(x)dx - 1\right)\frac{\partial}{\partial w^i}\int_0^1 u_\theta(x)dx \\
&= 2\left(\int_0^1 u_\theta(x)dx - 1\right)c^i\frac{\partial}{\partial w^i}\int_0^1 \phi^i(x)dx \\
\frac{\partial}{\partial b^i}\mathcal{L}_1 &= 2\left(\int_0^1 u_\theta(x)dx - 1\right)\frac{\partial}{\partial b^i}\int_0^1 u_\theta(x)dx \\
&= 2\left(\int_0^1 u_\theta(x)dx - 1\right)c^i\frac{\partial}{\partial b^i}\int_0^1 \phi^i(x)dx.
\end{aligned}$$

Second term

Consider the second term $\mathcal{L}_{2,k}$ of the loss function (5.2), using the expansion of u_θ in (5.4), we have:

$$\begin{aligned}
\mathcal{L}_{2,k}(u_\theta) &= \left(\int_0^1 [p_k(x) - p_k(S(x))]u_\theta(x)dx\right)^2 \\
&= \left(\sum_{i=1}^{N_n} c^i \int_0^1 [p_k(x) - p_k(S(x))]\phi^i(x)dx\right)^2.
\end{aligned}$$

Note that

$$\int_0^1 [p_k(x) - p_k(S(x))]\phi^i(x)dx = \int_p^q [p_k(x) - p_k(S(x))](w^i x + b^i)dx,$$

where p and q are given in Equations (5.6) and (5.7). The gradients of $\mathcal{L}_{2,k}$ are:

$$\begin{aligned}
\frac{\partial}{\partial c^i}\mathcal{L}_{2,k} &= 2\left(\sum_{j=1}^{N_n} c^j \int_0^1 [p_k(x) - p_k(S(x))]\phi^j(x)dx\right)\left(\int_0^1 [p_k(x) - p_k(S(x))]\phi^i(x)dx\right) \\
\frac{\partial}{\partial w^i}\mathcal{L}_{2,k} &= 2\left(\sum_{j=1}^{N_n} c^j \int_0^1 [p_k(x) - p_k(S(x))]\phi^j(x)dx\right)c^i\int_p^q [p_k(x) - p_k(S(x))]x\ dx \\
\frac{\partial}{\partial b^i}\mathcal{L}_{2,k} &= 2\left(\sum_{j=1}^{N_n} c^j \int_0^1 [p_k(x) - p_k(S(x))]\phi^j(x)dx\right)c^i\int_p^q [p_k(x) - p_k(S(x))]dx.
\end{aligned}$$

Third term

The energy term $\mathcal{L}_3$ involves the interaction between pairs of neurons ϕ^i and ϕ^j. The integrals of their product are:

$$\begin{aligned}
\int_0^1 \phi^j(x)\phi^i(x)dx &= \int_p^q (w^j x + b^j)(w^i x + b^i)dx \\
&= \int_p^q \left(w^j w^i x^2 + (w^j b^i + w^i b^j)x + b^j b^i\right) dx \\
&= \frac{w^j w^i}{3}(q^3 - p^3) + \frac{w^j b^i + w^i b^j}{2}(q^2 - p^2) + b^j b^i (q - p) \\
\int_0^1 \phi^j(x)\frac{\partial}{\partial w^i}\phi^i(x)dx &= \int_p^q (w^j x + b^j)x dx \\
&= \left[\frac{w^j x^3}{3} + \frac{b^j x^2}{2}\right]_p^q \\
&= \frac{w^j}{3}(q^3 - p^3) + \frac{b^j}{2}(q^2 - p^2) \\
\int_0^1 \phi^j(x)\frac{\partial}{\partial b^i}\phi^i(x)dx &= \int_p^q w^j x + b^j dx \\
&= \left[\frac{w^j x^2}{2} + b^j x\right]_p^q \\
&= \frac{w^j}{2}(q^2 - p^2) + b^j(q - p).
\end{aligned}$$

Here, p and q denote the boundaries of intersection of the supports for ϕ^i and ϕ^j:

$$\begin{aligned}
&p = \max\{\min[\max(-b^i/w^i, 0), 1], \min[\max(-b^j/w^j, 0), 1]\}, q = 1, && \text{if } w^i > 0, w^j > 0 \\
&p = \min[\max(-b^i/w^i, 0), 1], q = \max[\min(-b^j/w^j, 1), 0], && \text{if } w^i > 0, w^j < 0, p < q \\
&p = \min[\max(-b^i/w^i, 0), 1], q = \max[\min(-b^j/w^j, 1), 0], && \text{if } w^i < 0, w^j > 0, p < q \\
&p = 0, q = \min\{\max[\min(-b^i/w^i, 1), 0], \max[\min(-b^j/w^j, 1), 0]\}, && \text{if } w^i < 0, w^j < 0.
\end{aligned}$$

If none of these conditions are met, the integrals vanish to zero. Note that

$$(u_\theta(x))^2 = \left(\sum_{i=1}^{N_n} c^i \phi^i(x)\right)^2 = \sum_{j=1}^{N_n}\sum_{i=1}^{N_n} c^j c^i \phi^j(x)\phi^i(x).$$

Thus, the term $\mathcal{L}_3$ can be written in the form:

$$\mathcal{L}_3(u_\theta) = \int_0^1 (u_\theta(x))^2 dx = \sum_{j=1}^{N_n}\sum_{i=1}^{N_n} c^j c^i \int_0^1 \phi^j(x)\phi^i(x)dx.$$

The gradients of $\mathcal{L}_3$ are then given by:

$$\begin{aligned}
\frac{\partial}{\partial c^i}\mathcal{L}_3 &= \int_0^1 2u_\theta(x) \cdot \frac{\partial}{\partial c^i} u_\theta(x) dx \\
&= 2\int_0^1 u_\theta(x) \cdot \phi^i(x) dx \\
&= 2\sum_{j=1}^{N_n} c^j \int_0^1 \phi^j(x)\phi^i(x) dx \\
\frac{\partial}{\partial w^i}\mathcal{L}_3 &= \int_0^1 2u_\theta(x) \cdot \frac{\partial}{\partial w^i} u_\theta(x) dx \\
&= \int_0^1 2u_\theta(x) \cdot c^i \frac{\partial}{\partial w^i}\phi^i(x) dx \\
&= 2c^i \sum_{j=1}^{N_n} c^j \int_0^1 \phi^j(x) \frac{\partial}{\partial w^i}\phi^i(x) dx \\
\frac{\partial}{\partial b^i}\mathcal{L}_3 &= \int_0^1 2u_\theta(x) \cdot \frac{\partial}{\partial b^i} u_\theta(x) dx \\
&= \int_0^1 2u_\theta(x) \cdot c^i \frac{\partial}{\partial b^i}\phi^i(x) dx \\
&= 2c^i \sum_{j=1}^{N_n} c^j \int_0^1 \phi^j(x) \frac{\partial}{\partial b_i}\phi^i(x) dx.
\end{aligned}$$

Finally, the exact gradient of $\mathcal{L}$ in (5.2) with respect to c^i, w^i and b^i is obtained by summing the gradients of the individual components using the linearity of partial derivatives and the gradients of each term shown above.

5.2.3 Approach 3: VPINNs

We observe that the constraints in the MEM are conceptually similar to the variational formulations used in solving PDEs. By focusing strictly on these constraints, we can construct a VPINN framework (cf. Section 2.5.2) specifically for invariant density problems. The resulting loss function omits the regularization term and is defined as:

$$\mathcal{L}(u_\theta) := \left(\int_0^1 u_\theta(x)dx - 1\right)^2 + \sum_{k=1}^{N_c}\left(\int_0^1 [p_k(x) - p_k(S(x))]u_\theta(x)dx\right)^2. \tag{5.8}$$

A key advantage of this approach is that the exact gradients of the loss function can be computed analytically using the derivations presented in the previous section. This formulation also avoids the need for the quadratic term, which may not reliably approximate the entropy. In this context, the set of Chebyshev polynomials serves as

a discrete test space for the variational formulation of the invariant density problem. Defining the terms $b(u,v) := \int_0^1 [v(x) - v(S(x))]u(x)dx$ and $l(v) := 0$ allows the problem to be solved without requiring the entropy maximization condition, as seen in (3.19). Although further study into how different test spaces affect the neural network approximation could be pursued, this work focuses on Chebyshev polynomials to facilitate a direct comparison with the MEM and other established approaches.

Initialization of weights and biases

Proper initialization of weights and biases is important for accelerating the convergence of the training process. For the neural network architecture defined in (5.4), we initialize the hidden layer parameters by letting $w^j = 1$, and $b^j = -j/(N_c + 1)$. To initialize the output coefficients c^i, we solve the optimality condition $\frac{\partial \mathcal{L}}{\partial c^i} = 0$, where $\mathcal{L}$ is the loss function defined in (5.8). Expanding this derivative, we obtain:

$$2\left(\int_0^1 u_\theta(x)dx - 1\right)\left(\int_0^1 \frac{\partial}{\partial c^i}u_\theta(x)dx\right) + 2\sum_{k=1}^{N_c}\left(\int_0^1 [p_k(x) - p_k(S(x))]u_\theta(x)dx\right)\left(\int_0^1 \frac{\partial}{\partial c^i}[p_k(x) - p_k(S(x))]u_\theta(x)dx\right) = 0.$$

Substituting u_θ from Equation (5.4), this condition becomes:

$$\sum_{j=1}^{N_n} c^j \left(\int_0^1 \phi^j(x)dx\right)\left(\int_0^1 \phi^i(x)dx\right) + \sum_{j=1}^{N_n} c^j \sum_{k=1}^{N_c}\left(\int_0^1 [p_k(x) - p_k(S(x))]\phi^j(x)dx\right)\left(\int_0^1 [p_k(x) - p_k(S(x))]\phi^i(x)dx\right) = \int_0^1 \phi^i(x)dx.$$

This system is equivalent to the matrix equation $A\mathbf{c} = \mathbf{d}$ where the components of the matrix A and vector $\mathbf{d}$ are given by:

$$A_{ij} = \left(\int_0^1 \phi^j(x)dx\right)\left(\int_0^1 \phi^i(x)dx\right) + \sum_{k=1}^{N_c}\left(\int_0^1 [p_k(x) - p_k(S(x))]\phi^j(x)dx\right)\left(\int_0^1 [p_k(x) - p_k(S(x))]\phi^i(x)dx\right)$$

$$d_i = \int_0^1 \phi^i(x)dx.$$

Remark 5.2.1. Minimizing these loss functions requires solving a nonlinear optimization problem, which generally incurs a higher computational cost than Ulam's method

or the maximum entropy approach. Consequently, practical applications demand a balance between precision and runtime. However, evaluating computational speed falls outside the scope of this study, as none of the implementations were optimized for performance.

5.3 Numerical results

In this section, we present the numerical results for our deep learning methodologies in approximating invariant densities. We analyze the effect of varying the number of constraints on the accuracy of the approximation. We also compare the performance of our method with Ulam's method and the MEM. Additionally, we show the performance of the exact integration approach. To evaluate the accuracy of the neural network approximation u_θ, we compute the pointwise absolute error as:

$$e(x) := |u_\theta(x) - u(x)|,$$

where u represents the exact invariant density. Additionally, we compute the mean absolute error (MAE), defined as:

$$\text{MAE} := \int_0^1 |u_\theta(x) - u(x)| dx.$$

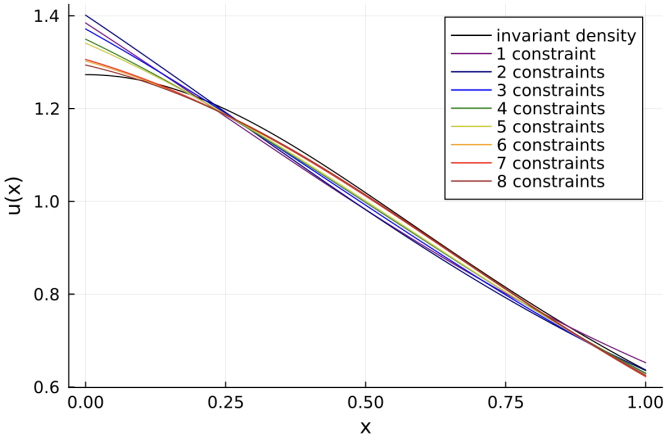
In our experiments, the MAE is approximated using the Gauss-Kronrod quadrature rule with 1,001 nodes.

5.3.1 Effect of the number of constraints in DMEM

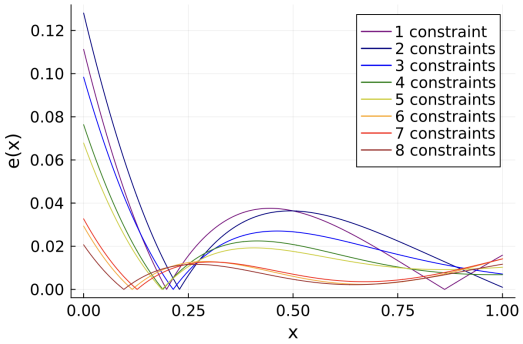
According to the methods proposed in Section 5.2, we need to choose the number of constraints in our loss function before training the neural network. Consistent with the theory of the MEM, the model's accuracy is expected to improve as more constraints are incorporated. We investigate this effect using Approach 1 in Section 5.2.1.

The model is implemented as a two-layer neural network with 10 neurons per hidden layer. We employ a combination of tanh and softplus activation functions as follows:

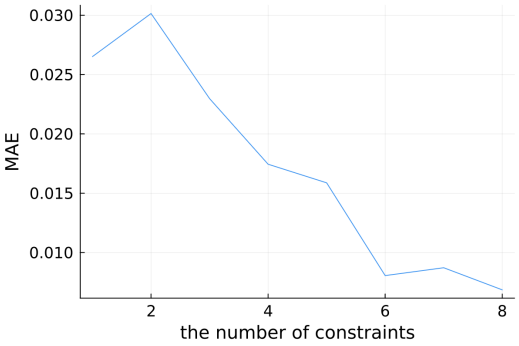
$$u_\theta(x) = \text{Softplus}(W^2 \tanh(W^1 x + \boldsymbol{b}^1) + \boldsymbol{b}^2). \tag{5.9}$$



(a) Exact invariant density and neural network approximations using different numbers of constraints N_c.



(b) Pointwise absolute error

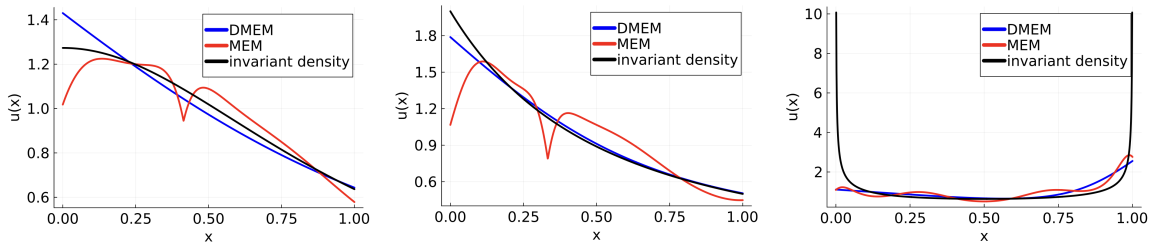


(c) Mean absolute error

Figure 5.1: DMEM results for Example 1.1 in Table 3.1 from Section 3.5.

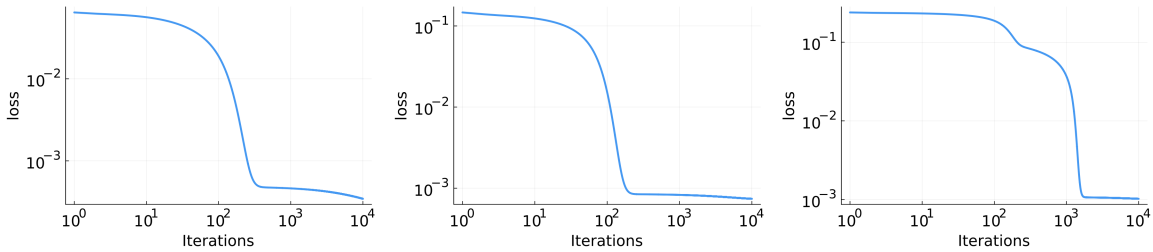
The parameters of the neural network are initialized randomly, and the integral terms in the loss function (5.1) are computed using the 101-point Gauss-Kronrod quadrature rule. We set the regularization parameter to $\lambda = 0.01$. We train the neural network using the gradient descent method with a learning rate $\eta = 0.01$ to minimize the loss function for $N_c \in \{1, 2, \ldots, 8\}$. Automatic differentiation [6] is used to compute gradients of the loss functions.

We apply DMEM to approximate the invariant density for Example 1.1 in Table 3.1. After 50,000 training iterations, the resulting approximations are shown in Figure 5.1 (a). As we can see, increasing the number of constraints leads to a more accurate approximation of the final results. In addition, the corresponding pointwise absolute error and mean absolute error, shown in Figures 5.1 (b)-(c), indicate that the error tends to decrease as the number of constraints increases. These results suggest that adding more constraints improves the accuracy of our model.



(a) Example 1.1 (b) Example 1.2 (c) Example 1.3

Figure 5.2: Approximations of the invariant densities using DMEM and MEM.



(a) Example 1.1 (b) Example 1.2 (c) Example 1.3

Figure 5.3: Loss plots for DMEM.

5.3.2 DMEM vs. MEM

To demonstrate the performance of our proposed method, we compare the DMEM with the results of the MEM. The MEM approximation functions, as presented in [37], for the three examples listed in Table 3.1 using four constraints are given by:

$$\begin{aligned}
\tilde{f}_1(x) =& 1.018e^{-3.6989(x-S_1(x))+8.6626(x^2-S_1(x)^2)-10.3795(x^3-S_1(x)^3)+4.8509(x^4-S_1(x)^4)} \\
\tilde{f}_2(x) =& 1.067e^{-7.9865(x-S_2(x))+20.2337(x^2-S_2(x)^2)-24.1849(x^3-S_2(x)^3)+11.0712(x^4-S_2(x)^4)} \\
\tilde{f}_3(x) =& 1.093e^{-3.9716(x-S_3(x))+23.3166(x^2-S_3(x)^2)-40.4357(x^3-S_3(x)^3)+22.0177(x^4-S_3(x)^4)}.
\end{aligned}$$

For the DMEM, we use a three-layer neural network defined as:

$$u_\theta(x) = \text{softplus}(W^3\text{softplus}(W^2\tanh(W^1x + \boldsymbol{b}^1) + \boldsymbol{b}^2) + \boldsymbol{b}^3). \tag{5.10}$$

We train the neural network using the procedure described in Section 5.3.1 for 10,000 iterations. The mean absolute errors for both methods are shown in Table 5.1. The results indicate that the DMEM outperforms the traditional MEM across all three examples. This demonstrates the effectiveness of our deep learning approach in approximating invariant densities with higher accuracy compared to the MEM. This

Example	DMEM	MEM
1.1	0.0301	0.0447
1.2	0.0225	0.1270
1.3	0.2734	0.3176

Table 5.1: Mean absolute errors of DMEM and MEM for three examples in Table 3.1.

improvement comes from the high expressivity of neural networks compared to the fixed form of the MEM solution. The corresponding plots of the approximations for both methods, along with their exact invariant densities, are shown in Figures 5.2. These visualizations further confirm the improved performance of DMEM, as the approximations are closer to the exact invariant densities. The loss plots in Figure 5.3 illustrate the convergence of our training process.

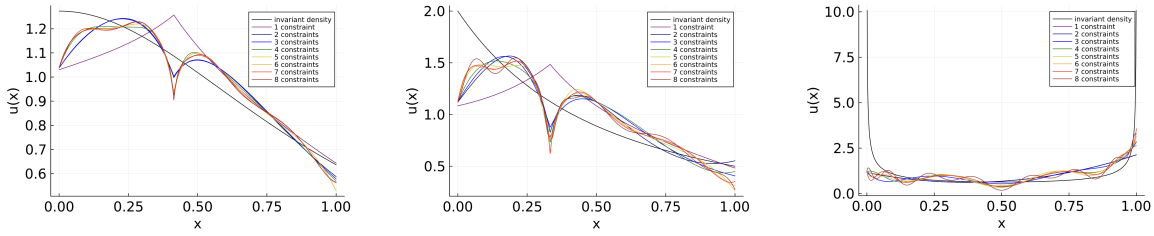
5.3.3 Numerical results and discussion for Approach 2

In this section, we examine the results for Approach 2, where we replace the entropy term by the quadratic energy term. Note that the analytical solution to the optimization problem (3.27)-(3.29) is a linear combination of the form (3.30) where $g_i(x) = p_i(x) - p_i(S(x))$. For the three examples in Table 3.1, we present these analytical invariant densities for varying numbers of constraints in Figure 5.4.

For the numerical implementation, we use a two-layer neural network with ReLU activation functions, as the piecewise-linear nature of ReLU is well-suited for computing exact gradients of the loss function given by Equation (5.2). The hidden layer size is set to 20 neurons for Examples 1.1 and 1.2, and increased to 30 neurons for Example 1.3 to accommodate the higher complexity of the solution. The training process of the neural network is conducted using the BFGS method[1]. This second-order optimizer is chosen for its better convergence properties in shallow neural network optimization compared to first-order methods.

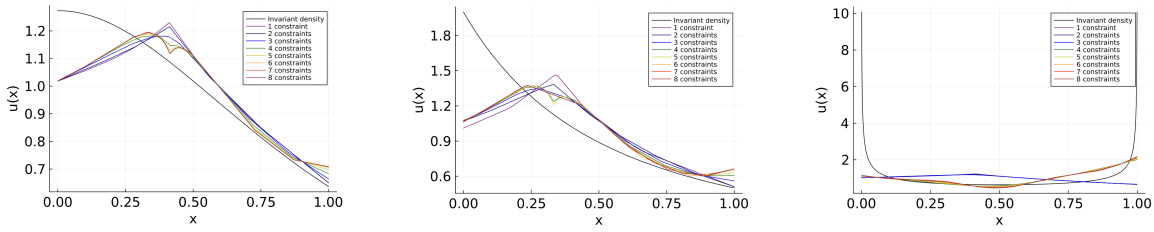
In the training process, we employ exact gradients as derived in Section 5.2.2, and terminate the training when the gradient norm falls below 10^{-8}. As illustrated in Figure 5.5, the neural networks generate piecewise-linear approximations of the

[1]The BFGS solver is paired with the line search algorithm.



(a) Example 1.1 (b) Example 1.2 (c) Example 1.3

Figure 5.4: Analytical solutions of the optimization problem (3.27)-(3.29).



(a) Example 1.1 (b) Example 1.2 (c) Example 1.3

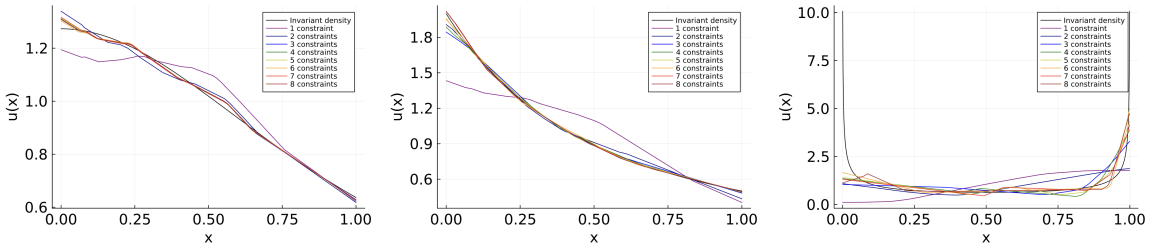
Figure 5.5: Neural networks approximations using Approach 2.

analytical solutions. However, the results indicate that this approach fails to provide a good approximation of the true invariant density. This issue arises because the quadratic approximation of the entropy functional only works well for density functions that are nearly uniform (constant). Since the true invariant densities in these examples are non-uniform, the quadratic term cannot serve as a robust substitute for the entropy term. Consequently, we turn to the other approach, which yields a better approximation.

5.3.4 Effect of the number of constraints for Approach 3

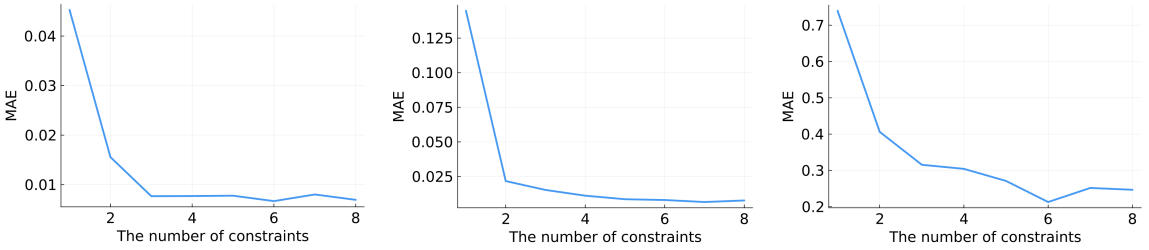
In this section, we present numerical results for Approach 3, which utilize a VPINN framework. In this approach, we use the same neural network architecture and optimization procedure detailed in Section 5.3.3. The loss function used is defined in Equation (5.8).

Figures 5.6 shows the approximated invariant densities for different numbers of constraints N_c. Figures 5.7 illustrates the mean absolute errors against different numbers of constraints. These results indicate that the VPINN approach achieves higher accuracy as the number of constraints (test functions) increases.



(a) Example 1.1 (b) Example 1.2 (c) Example 1.3

Figure 5.6: Approximation of invariant densities using Approach 3 (VPINNs).



(a) Example 1.1 (b) Example 1.2 (c) Example 1.3

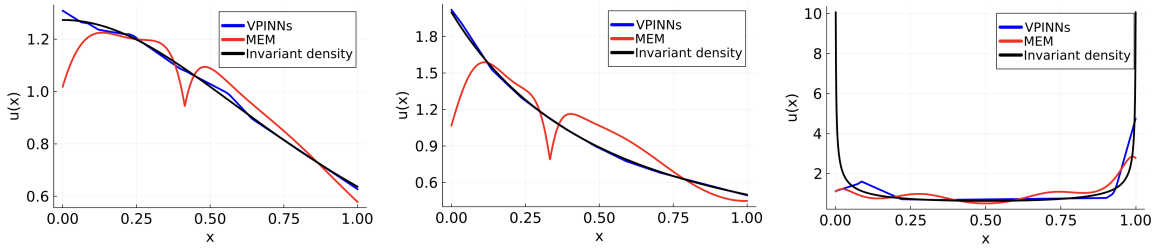
Figure 5.7: Mean absolute errors of neural network approximations of invariant densities.

Example	VPINNs	MEM
1.1	0.0077	0.0447
1.2	0.0112	0.1270
1.3	0.3043	0.3176

Table 5.2: Mean absolute errors for VPINNs and MEM.

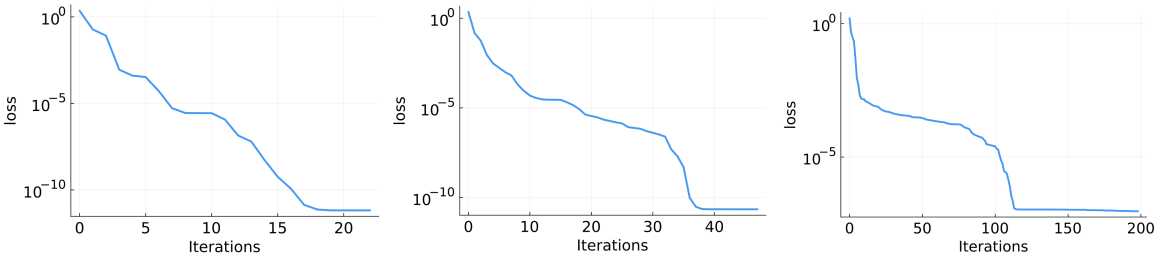
5.3.5 VPINNs vs. MEM

We compare the proposed VPINN method with the MEM using $N_c = 4$ constraints for both approaches. The VPINN implementation uses the same setting detailed in Section 5.3.4. As summarized in Table 5.2, the mean absolute errors produced by VPINNs are lower than those of the MEM across all three examples. These results indicate that the VPINN framework provides a more accurate approximation of the invariant densities, outperforming the MEM.



(a) Example 1.1 (b) Example 1.2 (c) Example 1.3

Figure 5.8: Approximation of the invariant densities using VPINNs v.s. MEM.



(a) Example 1.1 (b) Example 1.2 (c) Example 1.3

Figure 5.9: Loss plot of VPINNs method with $N_c = 4$.

The approximation plots comparing both methods against the exact invariant densities are presented in Figures 5.8. Additionally, the loss plots during the training process, shown in Figure 5.9, illustrate the convergence of optimization process.

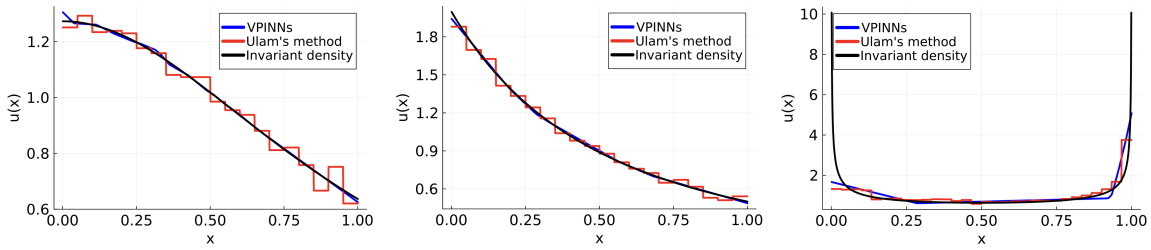
5.3.6 VPINNs vs. Ulam's method

To further validate our approach, we compare the VPINN method against Ulam's method. In the VPINN framework, we use $N_c = 6$ constraints in the loss function (5.8). To balance a comparison between the number of breakpoints in shallow neural networks and the discretization resolution of Ulam's method, we set the number of sub-intervals in Ulam's partition to match the number of neurons in our hidden layers. Specifically, we use 20 neurons (and 20 sub-intervals) for Examples 1.1 and 1.2, and 30 neurons (and 30 sub-intervals) for Example 1.3. Other settings are the same as described in Section 5.3.4.

The transition matrix entries for Ulam's method are computed using (3.14) with $n = 1{,}000$. As shown in Table 5.3, the mean absolute errors for VPINN method are lower than those obtained via Ulam's method across all three examples, indicating

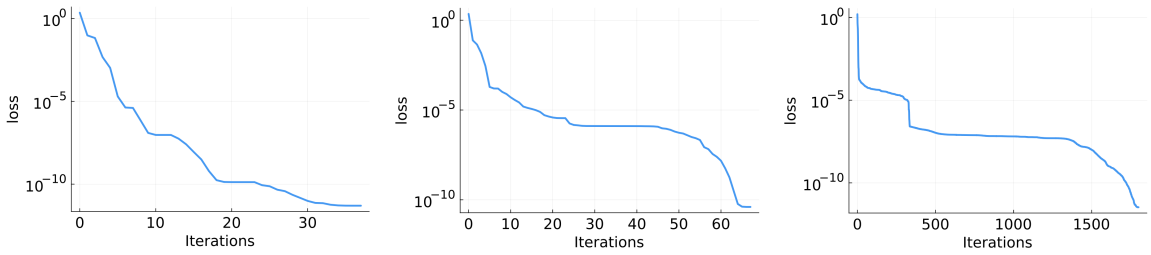
Example	VPINNs	Ulam
1.1	0.0067	0.0276
1.2	0.0080	0.0362
1.3	0.2130	0.2178

Table 5.3: Mean absolute errors of VPINNs and Ulam's method.



(a) Example 1.1 (b) Example 1.2 (c) Example 1.3

Figure 5.10: Invariant density approximations using VPINNs and Ulam's method.



(a) Example 1.1 (b) Example 1.2 (c) Example 1.3

Figure 5.11: Loss plots of VPINNs method with $N_c = 6$.

that the VPINN approach is more effective to capture the invariant densities than a piecewise-constant discretization of Ulam's method. The approximation plots for both methods and the loss plots for the neural network training are shown in Figures 5.10 and 5.11, respectively.

5.3.7 VPINNs: Gauss-Kronrod vs. exact integration

Finally, we conduct a performance comparison between the VPINNs method using 101-point Gauss-Kronrod integration versus the exact integration approach. The neural network architecture and other settings remain consistent with those detailed in in Section 5.3.6.

Examples	Training Time (seconds)		MAE	
	Gauss-Kronrod	Exact	Gauss-Kronrod	Exact
1.1	25	1	0.0151	0.0067
1.2	28	1	0.0134	0.0080
1.3	117	4	0.2892	0.2164

Table 5.4: Training time and MAE for VPINNs method.

Table 5.4 shows that the training time for the exact integration approach is significantly lower than those for the Gauss-Kronrod integration. This result highlights the efficiency of utilizing analytical gradients. In addition, the mean absolute errors for using exact integration are smaller than those using Gauss-Kronrod quadrature rule.

5.4 Conclusions

In this chapter, we developed and analyzed three loss functions for approximating the invariant densities of Perron-Frobenius operators, based on the maximum entropy method, minimum energy method, and VPINN approach.

We found that the mean absolute error decreases as more constraints are added to the loss function. Furthermore, we compared our DMEM with the MEM using four constraints and found that our DMEM method outperforms the MEM. This confirms that the high expressivity of neural networks allows for a more flexible and accurate fit than the fixed functional form required by the classical MEM.

Replacing the logarithmic entropy term with a quadratic energy term, or omitting it entirely, simplifies the loss function and facilitates the use of exact gradients. These approaches allow us to freely adjust the form of neural networks and derive analytical gradients for two-layer neural networks with ReLU activation functions.

However, the quadratic approach (Approach 2) was less effective. We found that the quadratic approximation is only reliable for densities that remain close to a uniform distribution. In contrast, the VPINN-based formulation (Approach 3) was successful. By focusing on the variational constraints, this approach effectively treats the invariant density problem within the modern VPINN framework. Notably, the use

of exact integration within the VPINN framework resulted in much shorter training times compared to the Gauss-Kronrod integration, without losing any approximation quality.

There are several directions for possible extensions from this chapter. First, applying these methods, especially the VPINN approach, to two dimensional or higher-dimensional dynamical systems is a natural next step. Second, a more detailed investigation into the optimal selection of basis functions for VPINN approach could help balance computational cost and accuracy. Finally, it would be valuable to conduct a comparative study between our neural network approach and classical Galerkin methods using piecewise-linear basis functions to further highlight the advantages of mesh-free deep learning techniques.

Chapter 6

Test-Space Adaptive RVPINNs

6.1 Introduction

Deep learning methodologies have been widely applied to various areas of scientific computing over the past decade (see, for example, [2,19,67,93,100,107]). One popular application in this area is the use of neural networks to approximate solutions of partial differential equations (PDEs). This line of research gained significant attention with the introduction of physics-informed neural networks (PINNs) [84] in 2019. The idea of PINNs is to generate data for the training by incorporating physical laws, expressed as PDEs. The neural network is trained using this data, which is on a mesh-free set of collocation points sampled from the physical domain and its boundary. PINNs have been successfully applied to a broad range of problems, including fluid dynamics [20,74], wave propagation [85], and inverse problems [28]. However, a caveat of this method is that it employs a strong-form residual to approximate the solution, whereas for many PDEs, the solution is only meaningful in a variational sense[1]. In addition, the mesh-free nature of PINNs is not always an advantage, as approximating solutions in complex scenarios, such as singular solutions or with steep gradients, can lead to convergence issues due to an improper balance between loss terms [45], making training too expensive or impractical.

As an alternative to the strong-form residual, variational PINNs (VPINNs) [58] and *hp*-VPINNs [59] were introduced to solve PDEs in their weak forms. While these

[1]Examples include the Poisson equation with a Dirac delta source term, Poisson problems with rough data, and elliptic equations with discontinuous coefficients (cf. [42]).

approaches align with weak solutions, they depend on the choice of basis functions and, consequently, their loss functions may not be robust with respect to the true error. Subsequently, robust VPINNs (RVPINNs) [87] were proposed to overcome these limitations. In this framework, the loss function is defined through a discrete Riesz representative of the residual. This construction ensures that the loss function acts as a reliable and efficient estimator for the true error up to a remainder term, which depends on the neural network approximation and the discretization of the test space. When standard FEM functions are used as test functions, the underlying mesh provides a natural framework for adaptive refinement that improves approximation quality.

An analysis of the quadrature rules and piecewise-polynomial degrees of the test space for VPINNs was conducted in [9], highlighting the critical role of test functions and numerical integration on the error decay rate. Several adaptive strategies for VPINNs have been developed, incorporating error estimators to guide test-space refinements. To address the dependence on the choice of basis functions, Berrone et al. [10] proposed a mesh-free approach in which the test-space mesh is no longer required. Instead, an a posteriori error estimator is employed to enrich the test space by incorporating new test functions. Additionally, Hu et al. [55] introduced an error estimator for stationary Navier-Stokes equations to guide adaptive mesh refinement within the VPINN framework. A rigorous a posteriori analysis was carried out in [45], with applications to adaptive refinement for neural network approximations. Beyond test space refinement, adaptive sampling of collocation points based on PDE residuals [47, 48, 95, 109] has been studied to enhance the numerical accuracy of PINNs. These adaptive strategies aim to improve accuracy and convergence rates for neural network-based PDE solvers, particularly when approximating solutions with singularities and sharp gradients.

In RVPINNs, robustness is granted only if the remainder is kept under control. In practice, this means that for a fixed mesh, continued training drives the neural network toward a solution determined by the discrete test space, rather than the true solution of the continuous variational formulation. If the mesh is too coarse, it might not be enough to control the true error during the full training. A fully robust control can be obtained by incorporating a computable upper bound for the remainder into

the loss [45]. However, this implies a major training cost as a natural consequence of including additional terms to the loss function. Another way to ensure full control is to use a sufficiently fine mesh; however, this becomes impractical in complex scenarios involving singularities or steep gradients.

We aim to accelerate convergence for RVPINNs in these complex problems. We extend the RVPINN framework with an adaptive test-space refinement strategy. This approach ensures robust training throughout this process without incorporating extra control terms into the loss function. The loss function is formulated in terms of the discrete Riesz representative of the variational residual, maintaining consistency with the original RVPINN methodology. The boundary term, when necessary, is defined by the trace norm of the boundary residual, following the approach in [45]. To ensure that the discrete Riesz representative accurately approximates its continuous counterpart for a given neural network approximation, we monitor the discrepancy between the discrete and continuous Riesz representatives during training and refine the mesh whenever this discrepancy exceeds a prescribed tolerance. The refinement indicator is based on the local difference between the discrete Riesz representatives computed in the original test space and in an enriched test space with a higher polynomial degree.

The main contributions of this chapter are as follows. We present upper bounds for approximation errors based on the RVPINN loss function and prove the equivalence between these error bounds. Moreover, we develop theoretical adaptive algorithms for RVPINNs and provide their convergence results. We propose a computable refinement indicator derived from two discrete Riesz representatives associated with two nested test spaces. We show that, under the saturation assumption [1, Section 5.2], this indicator is a reliable and efficient error estimator for the discrepancy between the discrete and continuous Riesz representatives. Furthermore, we introduce a practical adaptive RVPINN algorithm driven by this refinement indicator. We demonstrate the effectiveness of the proposed method by solving a Poisson problem on a square domain with a smooth solution, an elliptic interface problem with a discontinuous coefficient, and a Poisson equation on an L-shaped domain with a singular solution. To evaluate the performance of our adaptive approach, we compare our adaptive RVPINNs to the standard finite element method (FEM) on identical mesh and test space settings.

The remainder of this chapter is structured as follows. In Section 6.2, we describe the variational problem of PDEs and the neural network framework used throughout this work. We review the standard RVPINN methodology and discuss the loss function for the problems where the boundary conditions are not strongly imposed on the test space. Section 6.3 provides two types of error estimates and prove their equivalence. In Section 6.4, we introduce a theoretical adaptive algorithm and derive its error estimates. These results guarantee convergence of the neural network training with RVPINNs using the proposed adaptive strategy under certain conditions. In Section 6.5, we present the refinement indicator proposed for the adaptive framework. In Section 6.6, we develop a practical algorithm for the test-space adaptive RVPINNs. Section 6.7 introduces the model problems used for numerical results. Section 6.8 details and discusses the numerical experiments validating the proposed methodology. Section 6.9 summarizes the conclusions obtained from this work and outlines possible directions for future research.

6.2 RVPINNs for PDEs

6.2.1 Abstract framework

Let U and V be Hilbert spaces equipped with norms $\|\cdot\|_U$ and $\|\cdot\|_V$, respectively. Let V' denote the dual space of V. We consider a PDE problem that admits the following variational formulation:

$$\text{Find } u \in U \text{ such that } b(u,v) = l(v), \quad \forall v \in V, \tag{6.1}$$

where $b(\cdot,\cdot)$ is a bilinear form in $U \times V$ and $l(\cdot) \in V'$ is a bounded linear functional satisfying the following assumptions:

1. Boundedness of $b(\cdot,\cdot)$: there is a constant $\mu > 0$ such that

$$b(w,v) \leq \mu \|w\|_U \|v\|_V \quad \forall w \in U, v \in V, \tag{6.2}$$

2. Inf-sup stability: there is a constant $\alpha > 0$ such that

$$\sup_{0 \neq v \in V} \frac{b(w,v)}{\|v\|_V} \geq \alpha \|w\|_U \quad \forall w \in U, \tag{6.3}$$

3. Adjoint injectivity:

$$b(w, v) = 0 \quad \forall w \in U \Longrightarrow v = 0. \tag{6.4}$$

Under these assumptions, Problem (6.1) admits a unique solution $u \in U$ by the Banach–Nečas–Babŭska Theorem. Furthermore, the conditions (6.3) and (6.4) are well-known to imply the adjoint inf-sup condition (see, e.g., [35, Theorem 1]):

$$\sup_{0 \neq w \in U} \frac{b(w, v)}{\|w\|_U} \geq \alpha \|v\|_V, \quad \forall\, v \in V. \tag{6.5}$$

Let $(\cdot, \cdot)_V$ be the inner product inducing the norm $\|\cdot\|_V$ and define the dual norm of V' by

$$\|l(\cdot)\|_{V'} := \sup_{0 \neq v \in V} \frac{l(v)}{\|v\|_V}. \tag{6.6}$$

For any $w \in U$, we define $r(w, \cdot) := l(\cdot) - b(w, \cdot) \in V'$ as the variational residual with respect to w. According to the Riesz-Fréchet representation theorem (see [18, Theorem 5.5]), there exists a unique Riesz representative $\phi(w) \in V$ such that:

$$(\phi(w), v)_V = r(w, v), \quad \forall\, v \in V, \quad \text{and} \quad \|\phi(w)\|_V = \|r(w, \cdot)\|_{V'}. \tag{6.7}$$

As a consequence of the boundedness of $b(\cdot, \cdot)$ and the inf-sup stability, the following inequality can be proven:

$$\frac{1}{\mu} \|\phi(w)\|_V \leq \|u - w\|_U \leq \frac{1}{\alpha} \|\phi(w)\|_V. \tag{6.8}$$

Inequality (6.8) implies that $\|\phi(w)\|_V$ is a robust estimator of the true error $\|u - w\|_U$.

6.2.2 Neural Network framework and loss function

To approximate the solution to Problem (6.1), we employ a deep neural network u_θ of the form given in (2.3). Here, $\theta \in \mathbb{R}^n$ denotes the vector of trainable weights and biases, and the output layer uses an identity activation function. The optimal parameters θ are identified by minimizing a suitable loss functional through training via iterative optimization. We recall the set of all functions representable by this neural network architecture as $\mathcal{M}_n$ (see (2.8)), which we assume to be a subset of the trial space U.

Let $V_k = \text{span}\{\varphi_n\}_{n=1}^{N_k}$ be a finite-dimensional subspace of V. The discrete Riesz representative of the residual $r(u_\theta, \cdot)$ with respect to V_k is the function $\phi_k(u_\theta) \in V_k$ satisfying

$$(\phi_k(u_\theta), v_k)_V = r(u_\theta, v_k), \quad \forall\, v_k \in V_k. \tag{6.9}$$

It follows immediately that the discrete dual norm of the residual is equal to the V-norm of its discrete Riesz representative. Indeed,

$$\|r(u_\theta, \cdot)\|_{V_k'} = \sup_{0 \neq v_k \in V_k} \frac{r(u_\theta, v_k)}{\|v_k\|_V} = \sup_{0 \neq v_k \in V_k} \frac{(\phi_k(u_\theta), v_k)_V}{\|v_k\|_V} = \|\phi_k(u_\theta)\|_V. \tag{6.10}$$

Moreover, a straightforward computation shows that the norm of the discrete Riesz representative provides a lower bound for the approximation error. That is, one has

$$\frac{1}{\mu}\|\phi_k(u_\theta)\|_V \leq \|u - u_\theta\|_U. \tag{6.11}$$

The main idea behind RVPINNs is to formulate the loss function using the norm of the discrete Riesz representative (6.10) of the residual corresponding to u_θ. Following [87], the loss is defined as:

$$\mathcal{L}_k\left(u_\theta\right) := \|\phi_k(u_\theta)\|_V^2 + C(u_\theta), \tag{6.12}$$

where $C(\cdot)$ imposes constraints on the neural network (e.g., data interpolation). As a consequence of the linearity of the residual $r(u_\theta, \cdot)$, and of the inner product $(\cdot, \cdot)_V$, the discrete Riesz representative $\phi_k(u_\theta)$ can be written as

$$\phi_k(u_\theta) := \sum_{n=1}^{N_k} \eta_\theta(n)\varphi_n, \tag{6.13}$$

with coefficients $\eta_\theta(n) \in \mathbb{R}$. Substituting (6.13) into (6.9) yields the linear system

$$G_k \boldsymbol{\eta}_\theta = \boldsymbol{r}_{\theta,k}, \tag{6.14}$$

where $\boldsymbol{\eta}_\theta$ is the vector of coefficients $\eta_\theta(n)$, G_k the (θ-independent) symmetric and positive definite Gram matrix $G_k(n,m) = (\varphi_m, \varphi_n)_V$, and $\boldsymbol{r}_{\theta,k}$ is the vector with entries $\boldsymbol{r}_{\theta,k}(n) = r(u_\theta, \varphi_n)$, $n = 1, \ldots, N_k$. From (6.13) and the definitions of G_k and $\boldsymbol{r}_{\theta,k}$, the norm of the discrete Riesz representative satisfies

$$\|\phi_k(u_\theta)\|_V^2 = (\phi_k(u_\theta), \phi_k(u_\theta))_V = \boldsymbol{r}_{\theta,k}^T G_k^{-1} \boldsymbol{r}_{\theta,k}. \tag{6.15}$$

Hence, the loss functional (6.12) can be equivalently written as

$$\mathcal{L}_k(u_\theta) = \boldsymbol{r}_{\theta,k}^T G_k^{-1} \boldsymbol{r}_{\theta,k} + C(u_\theta). \tag{6.16}$$

For simplicity, throughout this chapter, we assume that the loss function $\mathcal{L}_k(u_\theta)$ has $C(u_\theta) = 0$.

6.2.3 Two residuals

In the previous section, we considered RVPINNs with a single variational formulation. However, for many PDEs where boundary conditions cannot be strongly imposed within the trial space, we may need to treat the boundary conditions as a separate variational form and include them directly into the loss function. In this section, we consider a weak formulation in which the PDE is defined by two variational forms: one acting on the domain Ω and another on its boundary $\Gamma := \partial\Omega$.

Let V_Ω and V_Γ be two vector spaces, and define the product space $V := V_\Omega \times V_\Gamma$. We consider the following variational problem: Find $u \in U$ such that:

$$\begin{aligned} r_\Omega(u, v_\Omega) &:= l_\Omega(v_\Omega) - b_\Omega(u, v_\Omega) = 0 \quad \forall v_\Omega \in V_\Omega, \\ r_\Gamma(u, v_\Gamma) &:= l_\Gamma(v_\Gamma) - b_\Gamma(u, v_\Gamma) = 0 \quad \forall v_\Gamma \in V_\Gamma, \end{aligned}$$

where $b_\Omega(\cdot,\cdot)$ and $b_\Gamma(\cdot,\cdot)$ are bounded bilinear forms, and $l_\Omega(\cdot)$ and $l_\Gamma(\cdot)$ are bounded linear functionals. We can combine these two forms into the following variational formulation:

$$\text{Find } u \in U \text{ such that } r(u, v) := r_\Omega(u, v_\Omega) + r_\Gamma(u, v_\Gamma) = 0 \quad \forall v = (v_\Omega, v_\Gamma) \in V.$$

To apply the RVPINN framework, we let $V_k^\Omega := \text{span}\{\varphi_n^\Omega\}_{n=1}^{N_k}$ and $V_k^\Gamma := \text{span}\{\varphi_m^\Gamma\}_{m=1}^{M_k}$ be discrete subspaces of V_Ω and V_Γ, respectively. The loss function is then defined as:

$$\mathcal{L}_k\left(u_\theta\right) := \|\phi_k^\Omega(u_\theta)\|_{V_\Omega}^2 + \|\phi_k^\Gamma(u_\theta)\|_{V_\Gamma}^2, \tag{6.17}$$

where $\phi_k^\Omega(u_\theta)$ and $\phi_k^\Gamma(u_\theta)$ are discrete Riesz representatives of the residuals $r_\Omega(u_\theta, \cdot)$ and $r_\Gamma(u_\theta, \cdot)$, respectively (see (6.9)). This formulation is computationally equivalent to:

$$\mathcal{L}_k(u_\theta) = (\boldsymbol{r}_{\theta,k}^\Omega)^T (G_k^\Omega)^{-1} \boldsymbol{r}_{\theta,k}^\Omega + (\boldsymbol{r}_{\theta,k}^\Gamma)^T (G_k^\Gamma)^{-1} \boldsymbol{r}_{\theta,k}^\Gamma, \tag{6.18}$$

where $\boldsymbol{r}_{\theta,k}^\Omega$ and $\boldsymbol{r}_{\theta,k}^\Gamma$ are vectors with entries $\boldsymbol{r}_{\theta,k}^\Omega(n) = r_\Omega(u_\theta, \varphi_n^\Omega)$, and $\boldsymbol{r}_{\theta,k}^\Gamma(m) = r_\Gamma(u_\theta, \varphi_m^\Gamma)$. Here, G_k^Ω and G_k^Γ denote the corresponding Gram matrices for the domain and boundary test spaces.

6.3 Error estimates

For a given neural network approximation u_θ, we let $\phi(u_\theta) \in V$ denote the (continuous) Riesz representative (see (6.7)) associated with u_θ defined by

$$(\phi(u_\theta), v)_V = r(u_\theta, v) \quad \forall v \in V. \tag{6.19}$$

In addition, we recall the discrete Riesz representative $\phi_k(u_\theta)$ with respect to $V_k \subset V$ (see (6.9)). In this section, we discuss two types of error estimates: the first is based on a residual remainder term, while the second relies on the discretization error of the Riesz representative.

6.3.1 Two estimates

As shown in [45], suppose there exists an operator $\Pi_k : V \to V_k$ and a constant $C_{\Pi_k} > 0$ such that $\|\Pi_k v\|_V \leq C_{\Pi_k}\|v\|_V$ for all $v \in V$. Then,

$$\|\phi(u_\theta)\|_V \leq C_{\Pi_k}\|\phi_k(u_\theta)\|_V + \mathcal{R}_k(u - u_\theta),$$

where $\mathcal{R}_k(u - u_\theta) := \sup_{0 \neq v \in V} \dfrac{b(u - u_\theta, v - \Pi_k v)}{\|v\|_V}$ is a residual remainder associated with the discrete space V_k. Applying (6.8), we obtain the following upper bound:

$$\|u - u_\theta\|_U \leq \frac{1}{\alpha}\left(C_{\Pi_k}\|\phi_k(u_\theta)\|_V + \mathcal{R}_k(u - u_\theta)\right). \tag{6.20}$$

Alternatively, by using Inequality (6.8) and the triangle inequality, we derive an upper bound for the true error:

$$\|u - u_\theta\|_U \leq \frac{1}{\alpha}\|\phi(u_\theta)\|_V \leq \frac{1}{\alpha}\left(\|\phi_k(u_\theta)\|_V + \|\phi(u_\theta) - \phi_k(u_\theta)\|_V\right). \tag{6.21}$$

Inequality (6.21) highlights that bounding the error between the trained neural network u_θ and the exact solution u depends on two ingredients:

1. Optimization convergence: The training process must successfully minimize the discrete dual norm, ensuring that $\|\phi_k(u_\theta)\|_V$ is small. This reflects the quality of the nonlinear solver (e.g. Adam [60]) in finding a near-optimal discrete minimizer.

2. Accuracy of the discrete test space: The discrete test space V_k must be rich enough to ensure that the discrete Riesz representative $\phi_k(u_\theta)$ is a faithful approximation of the continuous one, $\phi(u_\theta)$.

6.3.2 Equivalence of two error estimates

In this section, we establish the equivalence between the error estimate based on the remainder term, $\|\phi_k(u_\theta)\|_V + \mathcal{R}_k(u - u_\theta)$, and the estimate based on the Riesz representative error, $\|\phi_k(u_\theta)\|_V + \|\phi(u_\theta) - \phi_k(u_\theta)\|_V$.

Lemma 6.3.1. *Let u be the exact solution to (6.1) and u_θ be a neural network approximation. Let $\phi(u_\theta)$ and $\phi_k(u_\theta)$ be the continuous and discrete Riesz representatives of the residual, as defined in (6.19) and (6.9), respectively. For any $v \in V$ and a given operator $\Pi_k : V \to V_k$, the following identity holds:*

$$(\phi(u_\theta) - \phi_k(u_\theta), v)_V = b(u - u_\theta, v - \Pi_k v) - (\phi_k(u_\theta), v - \Pi_k v)_V. \tag{6.22}$$

Proof. By decomposing the inner product, using the definition of Riesz representatives $\phi(u_\theta)$ and $\phi_k(u_\theta)$, utilizing the definition of the residual $r(u_\theta, \cdot)$, and applying the bilinearity of $b(\cdot, \cdot)$, we obtain

$$\begin{aligned}
(\phi(u_\theta) - \phi_k(u_\theta), v)_V &= (\phi(u_\theta), v)_V - (\phi_k(u_\theta), v - \Pi_k v)_V - (\phi_k(u_\theta), \Pi_k v)_V \\
&= r(u_\theta, v) - (\phi_k(u_\theta), v - \Pi_k v)_V - r(u_\theta, \Pi_k v) \\
&= b(u - u_\theta, v) - (\phi_k(u_\theta), v - \Pi_k v)_V - b(u - u_\theta, \Pi_k v) \\
&= b(u - u_\theta, v - \Pi_k v) - (\phi_k(u_\theta), v - \Pi_k v)_V. \qquad \square
\end{aligned}$$

Proposition 6.3.1. Let u be the exact solution to (6.1) and u_θ be a neural network approximation. Let $\phi(u_\theta)$ and $\phi_k(u_\theta)$ be the continuous and discrete Riesz representatives of the residual, as defined in (6.19) and (6.9), respectively. For a given operator $\Pi_k : V \to V_k$, let $D_{\Pi_k} := \|I - \Pi_k\|$. Then, the following inequalities hold:

$$\|\phi(u_\theta) - \phi_k(u_\theta)\|_V + \|\phi_k(u_\theta)\|_V \leq \mathcal{R}_k(u - u_\theta) + (1 + D_{\Pi_k})\|\phi_k(u_\theta)\|_V, \tag{6.23}$$

and

$$\mathcal{R}_k(u - u_\theta) + \|\phi_k(u_\theta)\|_V \leq \|\phi(u_\theta) - \phi_k(u_\theta)\|_V + (1 + D_{\Pi_k})\|\phi_k(u_\theta)\|_V. \tag{6.24}$$

Proof. By the Cauchy-Schwarz inequality and the property of the operator norm for $I - \Pi_k$, we have

$$(\phi_k(u_\theta), v - \Pi_k v)_V \leq \|\phi_k(u_\theta)\|_V \|v - \Pi_k v\|_V \leq D_{\Pi_k} \|\phi_k(u_\theta)\|_V \|v\|_V. \tag{6.25}$$

Applying Lemma 6.3.1, Equation (6.25), and the definition of $\mathcal{R}_k(u-u_\theta)$, we obtain

$$\begin{aligned}\|\phi(u_\theta)-\phi_k(u_\theta)\|_V &= \sup_{0\neq v\in V}\frac{(\phi(u_\theta)-\phi_k(u_\theta),v)_V}{\|v\|_V}\\ &\leq \sup_{0\neq v\in V}\frac{b(u-u_\theta,v-\Pi_k v)}{\|v\|_V}+\sup_{0\neq v\in V}\frac{(\phi_k(u_\theta),v-\Pi_k v)_V}{\|v\|_V}\\ &\leq \mathcal{R}_k(u-u_\theta)+D_{\Pi_k}\|\phi_k(u_\theta)\|_V.\end{aligned}$$

Adding $\|\phi_k(u_\theta)\|_V$ to both sides yields (6.23).

Conversely, starting from the definition of $\mathcal{R}_k(u-u_\theta)$ and applying Lemma 6.3.1 and Equation (6.25), we have

$$\begin{aligned}\mathcal{R}_k(u-u_\theta) &= \sup_{0\neq v\in V}\frac{b(u-u_\theta,v-\Pi_k v)}{\|v\|_V}\\ &\leq \sup_{0\neq v\in V}\frac{(\phi(u_\theta)-\phi_k(u_\theta),v)_V}{\|v\|_V}+\sup_{0\neq v\in V}\frac{(\phi_k(u_\theta),v-\Pi_k v)_V}{\|v\|_V}\\ &\leq \|\phi(u_\theta)-\phi_k(u_\theta)\|_V+D_{\Pi_k}\|\phi_k(u_\theta)\|_V.\end{aligned}$$

Adding $\|\phi_k(u_\theta)\|_V$ to both sides yields (6.24). □

Although the V-norm of the (continuous) Riesz representative provides a robust estimate of the true error in the U-norm (see (6.8)), replacing $\phi(u_\theta)$ with the discrete Riesz representative $\phi_k(u_\theta)$ does not, in general, yield an analogous result. The error estimates in (6.20) and (6.21) show that minimizing $\|\phi_k(u_\theta)\|_V$ does not necessarily minimize the true error when the remainder term dominates. If the mesh underlying the discrete test space V_k is too coarse, a reduction in the loss functional $\mathcal{L}_k(u_\theta)$ does not necessarily guarantee a decrease in the true error. In practice, this issue can be observed after some training iterations: the loss decreases while the true residual norm–and consequently the true approximation error–stagnates (see [87, Fig. 3]). On the other hand, using excessively fine test-space mesh from the beginning is computationally expensive, as it requires a large number of quadrature points and slows down the training process.

To address these issues, the neural network should be trained initially using a coarse test-space mesh, which is computationally inexpensive, and then refined adaptively whenever the loss becomes sufficiently small. The mesh refinement ensures that the remainder term decreases as the test space is enriched, allowing continued training of the neural network under the updated loss function.

6.4 Theoretical adaptivity strategy

In this section, we analyze a theoretical adaptivity strategy designed to guarantee the convergence of neural network approximations within the RVPINN framework. We first discuss an idealized setting where the approximation error can be bounded within a prescribed tolerance. We then consider an adaptivity strategy based on the assumption that the trained neural network does not increase the Riesz discrepancy.

6.4.1 Idealized adaptivity strategy

Inequality (6.21) shows that the true error can be upper bounded by two quantities: the discretization error of the test space and the minimized loss over that space. Therefore, convergence to the solution is ensured if the Riesz discrepancy $\eta_k(u_\theta) := \|\phi(u_\theta) - \phi_k(u_\theta)\|_V$ is kept under control via test-space refinement, while the discrete loss $\sqrt{\mathcal{L}_k(u_\theta)} = \|\phi_k(u_\theta)\|_V$ is minimized through training.

Given an initial mesh $\mathcal{T}_0$, $\epsilon_0 > 0$, and $0 < \delta < 1$. Let u_{θ^0} be an initial neural network. We initialize $\mathcal{T}_1^0 := \mathcal{T}_0$ and $u_{\theta^1}^0 := u_{\theta^0}$. An idealized algorithm sets $k := 1$ and iterates:

$$
\begin{aligned}
&\epsilon_k = \delta \epsilon_{k-1} \\
&\text{For } i = 1, 2, \ldots, N \\
&\quad \mathcal{T}_k^i = \text{ADAPT}(\mathcal{T}_k^0, u_{\theta^k}^{i-1}, \epsilon_k) \\
&\quad u_{\theta^k}^i = \text{LEARN}(\mathcal{T}_k^i, u_{\theta^k}^0, \epsilon_k) \\
&\text{End for} \\
&\mathcal{T}_{k+1}^0 := \mathcal{T}_k^N, u_{\theta^{k+1}}^0 := u_{\theta^k}^N \\
&k \leftarrow k + 1.
\end{aligned}
$$

Here, the ADAPT process uses the trained neural network $u_{\theta^k}^{i-1}$ as prior knowledge to refine the initial mesh $\mathcal{T}_k^0$ and obtain $\mathcal{T}_k^i$ such that

$$
\|\phi(u_{\theta^k}^{i-1}) - \phi_{\mathcal{T}_k^i}(u_{\theta^k}^{i-1})\|_V \leq \epsilon_k, \tag{6.26}
$$

where $\phi_{\mathcal{T}_k^i}(u_{\theta^k}^{i-1})$ is the discrete Riesz representative of $r(u_{\theta^k}^{i-1}, \cdot)$ with respect to $\mathcal{T}_k^i$. The LEARN process uses the refined mesh $\mathcal{T}_k^i$ as an input and trains the neural

network $u^0_{\theta^k}$ over this mesh to obtain $u^i_{\theta^k}$ such that

$$\|\phi_{\mathcal{T}_k^i}(u^i_{\theta^k})\|_V \leq \epsilon_k. \tag{6.27}$$

Proposition 6.4.1 (Error bound for idealized strategy)**.** Let u be the solution to (6.1) and let $N = \infty$. Assuming that $u^i_{\theta^k} \to u_{\theta^k}$ as $i \to \infty$, we obtain:

$$\|u - u_{\theta^k}\|_U \leq \frac{2}{\alpha}\delta^k \epsilon_0.$$

Proof. Applying Equation (6.21), we obtain:

$$\|u - u_{\theta^k}\|_U \leq \frac{1}{\alpha}\left(\|\phi_{\mathcal{T}_k^N}(u_{\theta^k})\|_V + \|\phi(u_{\theta^k}) - \phi_{\mathcal{T}_k^N}(u_{\theta^k})\|_V\right). \tag{6.28}$$

By assumption, both $u^{i-1}_{\theta^k}$ and $u^i_{\theta^k}$ converge to u_{θ^k} as $i \to \infty$. Taking the limit in Equations (6.26) and (6.27) and substituting the results into Equation (6.28), we obtain

$$\|u - u_{\theta^k}\|_U \leq \frac{2}{\alpha}\epsilon_k.$$

Finally, using the relation $\epsilon_k = \delta\epsilon_{k-1}$ recursively yields $\epsilon_k = \delta^k\epsilon_0$, which completes the proof. □

Although this proposition guarantees convergence to the solution when $N = \infty$, the development of a practical algorithm to ensure such convergence requires further study. Alternatively, we propose a sequentially adaptive strategy, where $N = 1$, under the assumption that the Riesz discrepancy decreases after training, a behavior that is observed empirically in later sections. The following section details this adaptive algorithm and its corresponding theoretical results.

6.4.2 Sequential adaptivity strategy

Given an initial mesh $\mathcal{T}_0$, parameters $\epsilon_0 > 0$ and $0 < \delta < 1$, and an initial neural network u_{θ^0} satisfying $\sqrt{\mathcal{L}_0(u_{\theta^0})} \leq \epsilon_0$, a sequential adaptivity algorithm sets $k := 1$ and iterates:

$$\begin{aligned}
\epsilon_k &= \delta\epsilon_{k-1} \\
\mathcal{T}_k &= \text{ADAPT}(\mathcal{T}_{k-1}, u_{\theta^{k-1}}, \epsilon_k) \\
u_{\theta^k} &= \text{LEARN}(\mathcal{T}_k, u_{\theta^{k-1}}, \epsilon_k) \\
k &\leftarrow k + 1.
\end{aligned}$$

Here, the ADAPT process refines the current mesh $\mathcal{T}_{k-1}$ to produce the refined mesh $\mathcal{T}_k$ such that the Riesz discrepancy satisfies $\eta_k(u_{\theta^{k-1}}) \leq \epsilon_k$. The LEARN process trains the neural network $u_{\theta^{k-1}}$ on the mesh $\mathcal{T}_k$ by minimizing the loss function $\mathcal{L}_k$ to obtain the neural network u_{θ^k} such that $\sqrt{\mathcal{L}_k(u_{\theta^k})} \leq \epsilon_k$.

The following proposition establishes the optimality of the algorithm, assuming that the neural network u_{θ^k} obtained after training does not increase the Riesz discrepancy compared to the pre-trained state $u_{\theta^{k-1}}$, i.e., $\eta_k(u_{\theta^k}) \leq \eta_k(u_{\theta^{k-1}})$.

Proposition 6.4.2 (Error bound for sequential strategy)**.** Let u be the solution to (6.1) and $u_{\theta^k} \in U_{NN}$ be the neural network obtained at the kth iteration of the sequential algorithm. If $\eta_k(u_{\theta^k}) \leq \eta_k(u_{\theta^{k-1}})$, then the following bound holds:

$$\|u - u_{\theta^k}\|_U \leq \frac{2}{\alpha}\delta^k \epsilon_0,$$

where α is the constant in Equation (6.3).

Proof. By the assumption of this proposition and the ADAPT condition, we obtain

$$\|\phi(u_{\theta^k}) - \phi_k(u_{\theta^k})\|_V = \eta_k(u_{\theta^k}) \leq \eta_k(u_{\theta^{k-1}}) \leq \epsilon_k.$$

Furthermore, the LEARN condition ensures that

$$\|\phi_k(u_{\theta^k})\|_V \leq \epsilon_k.$$

Applying these two inequalities to the error bound in Equation (6.21) with $u_\theta = u_{\theta^k}$ yields

$$\|u - u_{\theta^k}\|_U \leq \frac{1}{\alpha}(\epsilon_k + \epsilon_k) = \frac{2}{\alpha}\delta^k \epsilon_0,$$

which completes the proof. □

In practice, training a neural network using a sufficiently fine mesh drives the approximation toward a minimizer of the variational residual rather than the exact solution u. Consequently, we analyze the error of the trained neural network by considering the best-in-class approximation, which minimizes the continuous residual norm. Since the set of neural networks $\mathcal{M}_n$ is neither a linear, closed, nor convex subset of U, a unique minimizer of $\|\phi(u_\theta)\|_V$ may not exist within $\mathcal{M}_n$ [83]. To account for this, we evaluate the approximation error by considering a quasi-minimizer of $\sqrt{\mathcal{L}_k(\cdot)} = \|\phi(\cdot)\|_V$ (cf. Definition 2.6.1).

Theorem 6.4.1. *Let $u_{\tilde{\theta}}$ be a quasi-minimizer of $\|\phi(\cdot)\|_V$ defined in Definition* (2.6.1)*, and let u be the exact solution to Problem* (6.1)*. Assume that $u_{\theta^k} \in \mathcal{M}_n$ is the neural network obtained at the kth iteration of the sequential algorithm and $\eta_k(u_{\theta^k}) \leq \eta_k(u_{\theta^{k-1}})$. Then, we have*

$$\|u_{\tilde{\theta}} - u_{\theta^k}\|_U \leq \frac{1}{\alpha}\left(\mu \inf_{\theta\in\mathbb{R}^s} \|u - u_\theta\|_U + \delta_n + 2\delta^k\epsilon_0\right),$$

where μ and α are the constants from Equations (6.2) *and* (6.3)*, respectively.*

Proof. Starting from the adjoint inf-sup condition in Equation (6.5), utilizing the linearity of the bilinear form $b(\cdot,\cdot)$, and using the definition of the Riesz representatives, we have:

$$\begin{aligned}\|u_{\tilde{\theta}} - u_{\theta^k}\|_U &\leq \frac{1}{\alpha}\sup_{0\neq v\in V}\frac{b(u_{\tilde{\theta}} - u_{\theta^k}, v)}{\|v\|_V} = \frac{1}{\alpha}\sup_{0\neq v\in V}\frac{r(u_{\tilde{\theta}}, v) - r(u_{\theta^k}, v)}{\|v\|_V}\\ &= \frac{1}{\alpha}\sup_{0\neq v\in V}\frac{(\phi(u_{\tilde{\theta}}) - \phi(u_{\theta^k}), v)_V}{\|v\|_V} = \frac{1}{\alpha}\|\phi(u_{\tilde{\theta}}) - \phi(u_{\theta^k})\|_V.\end{aligned}$$

Applying the triangle inequality, we obtain:

$$\|u_{\tilde{\theta}} - u_{\theta^k}\|_U \leq \frac{1}{\alpha}\left(\|\phi(u_{\tilde{\theta}})\|_V + \|\phi(u_{\theta^k}) - \phi_k(u_{\theta^k})\|_V + \|\phi_k(u_{\theta^k})\|_V\right).$$

Using the definition of a quasi-minimizer of $\|\phi(\cdot)\|_V$ and substituting the LEARN and ADAPT conditions, we obtain:

$$\|u_{\tilde{\theta}} - u_{\theta^k}\|_U \leq \frac{1}{\alpha}\left(\inf_{\theta\in\mathbb{R}^s}\|\phi(u_\theta)\|_V + \delta_n + 2\epsilon_k\right).$$

Finally, applying Equation (6.8) completes the proof. □

Remark 6.4.1. Theorem 6.4.1 demonstrates that the adaptive algorithm can guide the neural network toward a quasi-minimizer when the neural network architecture is sufficiently deep or wide to ensure that the error of the best approximation is very small.

6.5 Refinement indicator

We note that standard RVPINNs might reach a performance plateau because, after some training iterations, the discrete Riesz representative in the finite-dimensional test space V_k fails to adequately approximate the exact Riesz representative in the infinite-dimensional test space V. To make the discrepancy $\eta_k(u_\theta) = \|\phi(u_\theta) - \phi_k(u_\theta)\|_V$

small, the discrete test space V_k must be sufficiently rich. However, training a neural network using an excessively refined test space from the beginning is computationally expensive. While one could use uniform refinement once the loss reaches a target level, a more efficient approach is adaptive refinement. By employing an appropriate refinement indicator, we can enrich the test space only where necessary, maintaining accuracy without incurring unnecessary computational costs.

We define a computable refinement indicator that acts as a surrogate for the discrepancy between the continuous and discrete Riesz representatives. Specifically, we introduce a two-level refinement criterion that compares Riesz representatives from two discrete test spaces. Let $V_k \subset \hat{V}_k \subset V$ be two nested discrete spaces, where V_k is the original test space (based on some polynomial order) and $\hat{V}_k$ is an enriched space (for example, by using higher-order polynomials or a finer subgrid). We denote the discrete Riesz representatives associated with V_k and $\hat{V}_k$ as $\phi_k(u_\theta)$ and $\hat{\phi}_k(u_\theta)$, respectively. Let $\mathcal{T}_k$ be the (triangular) mesh associated with these spaces, and let $T \in \mathcal{T}_k$ denote an element (triangle) in the mesh. For each element T, we compute the local discrepancy indicator:

$$\iota_k^2(T) := \|\hat{\phi}_k(u_\theta) - \phi_k(u_\theta)\|_{V(T)}^2. \tag{6.29}$$

This indicator measures the local difference between the two discrete Riesz representatives and identifies regions where the test space V_k needs further enrichment.

The following theorem shows that the global estimator defined by:

$$\iota_k(\mathcal{T}_k) := \|\hat{\phi}_k(u_\theta) - \phi_k(u_\theta)\|_V = \sqrt{\sum_{T\in\mathcal{T}_k} \iota_k^2(T)} \tag{6.30}$$

is equivalent to the true discrepancy $\|\phi(u_\theta) - \phi_k(u_\theta)\|_V$. This holds under a standard saturation assumption [1, Section 5.2], which assumes that the enriched Riesz representative $\hat{\phi}_k(u_\theta)$ is a better approximation of the continuous representative $\phi(u_\theta)$ than $\phi_k(u_\theta)$, This equivalence establishes the reliability and efficiency of ι_k as a refinement indicator for the adaptive process.

Theorem 6.5.1 (Equivalence of the two-level indicator)**.** *Suppose the saturation assumption holds: there exists a constant $\beta \in [0, 1)$ such that*

$$\|\phi(u_\theta) - \hat{\phi}_k(u_\theta)\|_V \leq \beta\|\phi(u_\theta) - \phi_k(u_\theta)\|_V.$$

Then, the global refinement indicator satisfies the following two-sided bound:

$$\|\hat{\phi}_k(u_\theta) - \phi_k(u_\theta)\|_V \leq \|\phi(u_\theta) - \phi_k(u_\theta)\|_V \leq \frac{1}{\sqrt{1-\beta^2}}\|\hat{\phi}_k(u_\theta) - \phi_k(u_\theta)\|_V.$$

Proof. Lower bound: Using Galerkin orthogonality $(\hat{\phi}_k(u_\theta) - \phi(u_\theta), v)_V = 0$ for all $v \in \hat{V}_k$, and by the Cauchy-Schwarz inequality, we have

$$\begin{aligned}
\|\hat{\phi}_k(u_\theta) - \phi_k(u_\theta)\|_V^2 &= (\hat{\phi}_k(u_\theta) - \phi_k(u_\theta), \hat{\phi}_k(u_\theta) - \phi_k(u_\theta))_V \\
&= (\hat{\phi}_k(u_\theta) - \phi(u_\theta), \hat{\phi}_k(u_\theta) - \phi_k(u_\theta))_V \\
&\quad + (\phi(u_\theta) - \phi_k(u_\theta), \hat{\phi}_k(u_\theta) - \phi_k(u_\theta))_V \\
&\leq 0 + \|\phi(u_\theta) - \phi_k(u_\theta)\|_V \|\hat{\phi}_k(u_\theta) - \phi_k(u_\theta)\|_V.
\end{aligned}$$

By canceling, we obtain the desired lower bound.

Upper bound: Expanding and using Galerkin orthogonality, we have

$$\begin{aligned}
\|\phi(u_\theta) - \phi_k(u_\theta)\|_V^2 &= \|\phi(u_\theta) - \hat{\phi}_k(u_\theta)\|_V^2 + 2(\phi(u_\theta) - \hat{\phi}_k(u_\theta), \hat{\phi}_k(u_\theta) - \phi_k(u_\theta))_V \\
&\quad + \|\hat{\phi}_k(u_\theta) - \phi_k(u_\theta)\|_V^2 \\
&= \|\phi(u_\theta) - \hat{\phi}_k(u_\theta)\|_V^2 + \|\hat{\phi}_k(u_\theta) - \phi_k(u_\theta)\|_V^2.
\end{aligned}$$

Applying the saturation assumption gives

$$\|\phi(u_\theta) - \phi_k(u_\theta)\|_V^2 \leq \beta^2\|\phi(u_\theta) - \phi_k(u_\theta)\|_V^2 + \|\hat{\phi}_k(u_\theta) - \phi_k(u_\theta)\|_V^2.$$

Rearranging yields $\sqrt{1-\beta^2}\|\phi(u_\theta) - \phi_k(u_\theta)\|_V \leq \|\hat{\phi}_k(u_\theta) - \phi_k(u_\theta)\|_V$ which completes the proof. □

Remark 6.5.1. One could use an a posteriori error estimator for the remainder term $\mathcal{R}_k(u - u_\theta)$ as a refinement indicator. In [45], this estimator is an additional term in the RVPINN loss function.

In case of using two residuals (see Section 6.2.3), we compute two local discrepancy indicators to account for domain and boundary errors. For an element T, these indicators are defined as:

$$\iota_{k,\Omega}^2(T) := \|\hat{\phi}_k^\Omega(u_\theta) - \phi_k^\Omega(u_\theta)\|_{V(T)}^2, \tag{6.31}$$

$$\iota_{k,\Gamma}^2(T) := \|\hat{\phi}_k^\Gamma(u_\theta) - \phi_k^\Gamma(u_\theta)\|_{V(T)}^2, \tag{6.32}$$

where $\hat{\phi}_k^{\Omega}(u_\theta)$ and $\hat{\phi}_k^{\Gamma}(u_\theta)$ are the discrete Riesz representatives with respect to the enriched test spaces $\hat{V}_k^{\Omega} \supset V_k^{\Omega}$ and $\hat{V}_k^{\Gamma} \supset V_k^{\Gamma}$, respectively. Finally, the global estimator is defined by:

$$\iota_k(\mathcal{T}_k) := \sqrt{\sum_{T \in \mathcal{T}_k} \left(\iota_{k,\Omega}^2(T) + \iota_{k,\Gamma}^2(T)\right)}. \tag{6.33}$$

6.6 Practical adaptivity strategy

In this section, we provide a practical algorithm for the adaptivity strategy in Section 6.4.2. The adaptive procedure is based on the standard - SOLVE-ESTIMATE-MARK-REFINE structure in the classical adaptive finite element method (AFEM) loop [29]. We integrate these processes into the sequential adaptivity strategy in Section 6.4.2. We use $\iota_k(\mathcal{T}_k)$ as a computable error estimator for $\eta_k(u_\theta)$. We assume that the neural network architecture possesses sufficient expressive power to the solution and the nonlinear optimizer (e.g. Adam) effectively minimizes the loss function.

Let $\mathcal{T}_0$ be an initial mesh of the domain, and let $\mathcal{T}_k$ denote the mesh after k refinement steps. We define V_k as the discrete test space associated with $\mathcal{T}_k$, typically constructed using low-order polynomials. Let $\mathcal{L}_k$ be the corresponding loss function defined in (6.18), and let u_{θ^k} be the neural network approximation associated with $\mathcal{L}_k$.

Given an initial mesh $\mathcal{T}_0$, a starting tolerance $\epsilon_0 \in (0, \infty)$, scaling factors $\delta_L, \delta_A \in (0, 1)$, and marking parameter $\gamma \in (0, 1]$, we initially train the neural network to obtain u_{θ^0} such that $\sqrt{\mathcal{L}_0(u_{\theta^0})} \leq \epsilon_0$. The adaptive loop sets $k := 1$ and proceeds as follows:

1. $\mathcal{T}_k = \mathrm{ADAPT}(\mathcal{T}_{k-1}, u_{\theta^{k-1}}, \delta_A^k \epsilon_0)$

   $\{\iota_{k-1}(T)\}_{T \in \mathcal{T}_{k-1}} = \mathrm{ESTIMATE}(\mathcal{T}_{k-1}, u_{\theta^{k-1}})$

   $\mathcal{M}_{k-1} = \mathrm{MARK}(\{\iota_{k-1}(T)\}_{T \in \mathcal{T}_{k-1}}, \gamma)$

   $\mathcal{T}_{k-1} = \mathrm{REFINE}(\mathcal{T}_{k-1}, \mathcal{M}_{k-1})$

2. $u_{\theta^k} = \mathrm{LEARN}(\mathcal{T}_k, u_{\theta^{k-1}}, \delta_L^k \epsilon_0)$

   $u_{\theta^{k-1}} = \mathrm{SOLVE}(\mathcal{T}_k, u_{\theta^{k-1}})$

3. $k \leftarrow k + 1$.

The adaptive algorithm proceeds through four main steps during iteration (k):

ESTIMATE: For a predefined test space V_{k-1} (based on some polynomial order) on the current mesh $\mathcal{T}_{k-1}$, we construct an enriched test space $\hat{V}_{k-1}$ (typically by increasing the polynomial order) such that $V_{k-1} \subset \hat{V}_{k-1} \subset V$. We then compute the discrete Riesz representatives $\phi_{k-1}(u_{\theta^{k-1}})$ and $\hat{\phi}_{k-1}(u_{\theta^{k-1}})$ with respect to these spaces. For each element $T \in \mathcal{T}_{k-1}$, we calculate the local discrepancy indicator $\iota^2_{k-1}(T)$ as defined in (6.29). In cases involving two variational forms, we evaluate the domain indicator $\iota^2_{k-1,\Omega}(T)$ and the boundary indicator $\iota^2_{k-1,\Gamma}(T)$ according to (6.31) and (6.32).

MARK: We apply the Dörfler marking criterion [41] to identify the elements that account for the largest portion of the discrepancy. In particular, for a given marking parameter $\gamma \in (0, 1]$, we find a smallest subset of elements $\mathcal{M}_{k-1} \subset \mathcal{T}_{k-1}$ such that:

$$\sum_{T \in \mathcal{M}_{k-1}} \iota^2_{k-1}(T) \geq \gamma \sum_{T \in \mathcal{T}_{k-1}} \iota^2_{k-1}(T). \tag{6.34}$$

In cases involving two variational forms, we follow a separate marking strategy for the domain Ω and the boundary Γ. Let $\mathcal{M}^{\Omega}_{k-1}$ and $\mathcal{M}^{\Gamma}_{k-1}$ be the smallest subsets of marked elements using indicators $\iota^2_{k-1,\Omega}(T)$ and $\iota^2_{k-1,\Gamma}(T)$ from (6.31) and (6.32), respectively. To maintain a balance between the two regions, we define $m = \min(|\mathcal{M}^{\Omega}_{k-1}|, |\mathcal{M}^{\Gamma}_{k-1}|)$ and select the m elements from each subset with the largest local discrepancies. These selected elements are then combined to form the final set of marked elements $\mathcal{M}_{k-1}$.

REFINE: We refine the marked elements in $\mathcal{M}_{k-1}$ using a conforming refinement scheme to generate a finer mesh $\mathcal{T}_{k-1}$. We then compute the global estimator $\iota_{k-1}(\mathcal{T}_{k-1})$ in (6.30) or (6.33). If $\iota_{k-1}(\mathcal{T}_{k-1}) > \delta^k_A \epsilon_0$, we repeat ESTIMATE and MARK procedures to further enrich the test space. Otherwise, this mesh defines an enriched test space mesh $\mathcal{T}_k$ for the next level.

SOLVE: We train the neural network on the current mesh $\mathcal{T}_k$ by minimizing the loss function $\mathcal{L}_k$ using Adam optimizer [60]. This process is repeated until we obtain the neural network u_{θ^k} such that $\sqrt{\mathcal{L}_k(u_{\theta^k})} \leq \delta^k_L \epsilon_0$.

The algorithm for test-space adaptive RVPINNs is described in Algorithm 1.

Algorithm 1 Adaptive training algorithm.

Require: u_{θ^0}, $\mathcal{T}_0$,V_0, $\mathcal{L}_0$, ϵ_0, $\delta_L, \delta_A, \gamma$, $K \in \mathbb{N}$

for k = 1 **to** K **do**
  while $\iota_{k-1}(\mathcal{T}_{k-1}) > \delta_A^k \epsilon_0$ **do**
    Compute $\{\iota_{k-1}(T)\}_{T\in\mathcal{T}_{k-1}}$.
    Identify marked elements $\mathcal{M}_{k-1}$.
    Refine $\mathcal{T}_{k-1}$ and update V_{k-1}.
  end while
  Set $\mathcal{T}_k$ as the refined mesh and V_k as the associated test space.
  while $\sqrt{\mathcal{L}_k(u_{\theta^{k-1}})} > \delta_L^k \epsilon_0$ **do**
    Update θ^{k-1} (one step of the nonlinear solver).
  end while
  Set u_{θ^k} as the optimized neural network.
end for
return Optimized neural network u_{θ^K}.

Remark 6.6.1. The SOLVE step of the proposed methods involves training the neural network, making it computationally more expensive than the SOLVE step in the standard AFEM. Applying these techniques in practice requires balancing accuracy against runtime. However, runtime comparisons are omitted here, as none of the methods were optimized for computational efficiency.

6.7 Model problems

In this section, we consider three two-dimensional PDEs with different degrees of regularity: a Poisson problem with a Dirichlet boundary condition featuring a smooth solution, an elliptic interface problem with a discontinuous diffusion coefficient exhibiting a kink solution, and a Poisson problem on an L-shaped domain possessing a corner singularity. The notation in Appendix A will be used throughout the subsequent sections.

6.7.1 Poisson problem with a smooth solution

Let $\Omega = (0,1)^2$ be a square domain. We consider the following Poisson problem: find u such that

$$\begin{cases} -\Delta u = f & \text{in } \Omega, \\ u = 0 & \text{on } \Gamma, \end{cases} \tag{6.35}$$

where the function $f \in L^2(\Omega)$ is defined as $f(x,y) = 2\pi^2 \sin(\pi x)\sin(\pi y)$. This problem admits a smooth solution $u(x,y) = \sin(\pi x)\sin(\pi y)$. The variational formulation of (6.35) is to find $u \in H_0^1(\Omega)$ such that

$$r(u,v) := (f,v)_{L^2(\Omega)} - (\nabla u, \nabla v)_{L^2(\Omega;\mathbb{R}^2)} = 0 \quad \forall v \in H_0^1(\Omega). \tag{6.36}$$

The well-posedness of this problem is guaranteed by the Lax-Milgram theorem (cf. [42, Section 6.2]).

6.7.2 Elliptic interface problem with a kink solution

Let $\Omega = (-1,1)^2$ be a square domain. We consider an elliptic interface problem: find u such that

$$\begin{cases} -\nabla \cdot (a \nabla u) = f & \text{in } \Omega, \\ u = g & \text{on } \Gamma, \end{cases} \tag{6.37}$$

where a is a piecewise-constant diffusion coefficient. In polar coordinates (ρ, ψ), a is defined as

$$a(\rho,\psi) = \begin{cases} a_1, & \rho < \rho_0 \\ a_2, & \rho \geq \rho_0, \end{cases}$$

with $\rho = \sqrt{x^2+y^2}$ and $\rho_0 = \pi/6.28$. The functions f and g are given in polar coordinates by $f(\rho,\psi) = -9\rho$ and

$$g(\rho,\psi) = \frac{\rho^3}{a_2} + \left(\frac{1}{a_1} - \frac{1}{a_2}\right)\rho_0^3. \tag{6.38}$$

The exact solution to this problem [105] is

$$u(\rho,\psi) = \begin{cases} \frac{\rho^3}{a_1}, & \rho < \rho_0 \\ \frac{\rho^3}{a_2} + \left(\frac{1}{a_1} - \frac{1}{a_2}\right)\rho_0^3, & \rho \geq \rho_0. \end{cases} \tag{6.39}$$

In our experiment, we set $a_1 = 1$ and $a_2 = 5$. The variational formulation of (6.37) is to find $u \in H^1(\Omega)$ such that $u = g$ on Γ and

$$r(u, v) := (f, v)_{L^2(\Omega)} - (a\nabla u, \nabla v)_{L^2(\Omega;\mathbb{R}^2)} = 0 \quad \forall v \in H_0^1(\Omega). \tag{6.40}$$

Given a lifting $w \in H^1(\Omega)$ such that $w|_\Gamma = g$, the existence and uniqueness of $\tilde{u} \in H_0^1(\Omega)$ such that $r(\tilde{u} + w, v) = 0$, or equivalently $r(\tilde{u}, v) = -r(w, v)$ for all $v \in H_0^1(\Omega)$ follow from the Lax-Milgram theorem. By defining $u := \tilde{u} + w \in H^1(\Omega)$, we establish the well-posedness of the variational problem (6.40).

6.7.3 Poisson problem with a singular solution

Let $\Omega := (-1, 1)^2 \backslash (-1, 0]^2$ be an L-shaped domain. We consider the Laplace equation (Poisson problem with zero source term): find u such that

$$\begin{cases} -\Delta u = 0 & \text{in } \Omega, \\ \quad u = g & \text{on } \Gamma, \end{cases} \tag{6.41}$$

where g is defined in polar coordinates (ρ, ψ) as $g(\rho, \psi) = \rho^{2/3} \sin(\frac{2}{3}(\pi - \psi))$. The exact solution to this problem is $u(\rho, \psi) = g(\rho, \psi)$, which can be expressed in Cartesian coordinates as:

$$u(x, y) = \sqrt[3]{x^2 + y^2} \sin\left(\frac{2}{3}(\pi - \operatorname{atan2}(y, x))\right), \tag{6.42}$$

where atan2 denotes the four-quadrant inverse tangent. Note that the solution exhibits a corner singularity at the origin $(0, 0)$, where the gradient becomes unbounded as $(x, y) \to (0, 0)$, and belongs to the fractional Sobolev space $H^{5/3-\epsilon}(\Omega)$ for every $\epsilon > 0$ [75].

The variational formulation of (6.41) is stated as follows: Find $u \in H^1(\Omega)$ such that

$$\begin{aligned} r_\Omega(u, v) &:= (\nabla u, \nabla v)_{L^2(\Omega;\mathbb{R}^2)} = 0 \qquad && \forall v \in H_0^1(\Omega), \\ r_\Gamma(u, \mathbf{v}) &:= \langle (g - u)|_\Gamma, \mathbf{v} \rangle_\Gamma = 0 \qquad && \forall \mathbf{v} \in \mathbf{H}(\mathrm{div}; \Omega). \end{aligned} \tag{6.43}$$

We note that the condition $r_\Gamma(u, \mathbf{v}) = 0$ for all $\mathbf{v} \in \mathbf{H}(\mathrm{div}; \Omega)$ enforces the boundary condition $u|_\Gamma = g$ in the sense of $H^{1/2}(\Gamma)$. Given a lifting $w \in H^1(\Omega)$ such that $w|_\Gamma = g$, the existence and uniqueness of $\tilde{u} \in H_0^1(\Omega)$ such that $r_\Omega(\tilde{u} + w, v) = 0$, or equivalently $r_\Omega(\tilde{u}, v) = -r_\Omega(w, v)$ for all $v \in H_0^1(\Omega)$ follow from the Lax-Milgram theorem. By defining $u := \tilde{u} + w \in H^1(\Omega)$, we establish the existence and uniqueness of the solution $u \in H^1(\Omega)$ to (6.43).

6.8 Numerical results

In this section, we demonstrate the performance of our adaptive strategy using model problems in Section 6.7. We analyze the convergence rates achieved by our approach and compare them against FEM using the same test spaces and meshes.

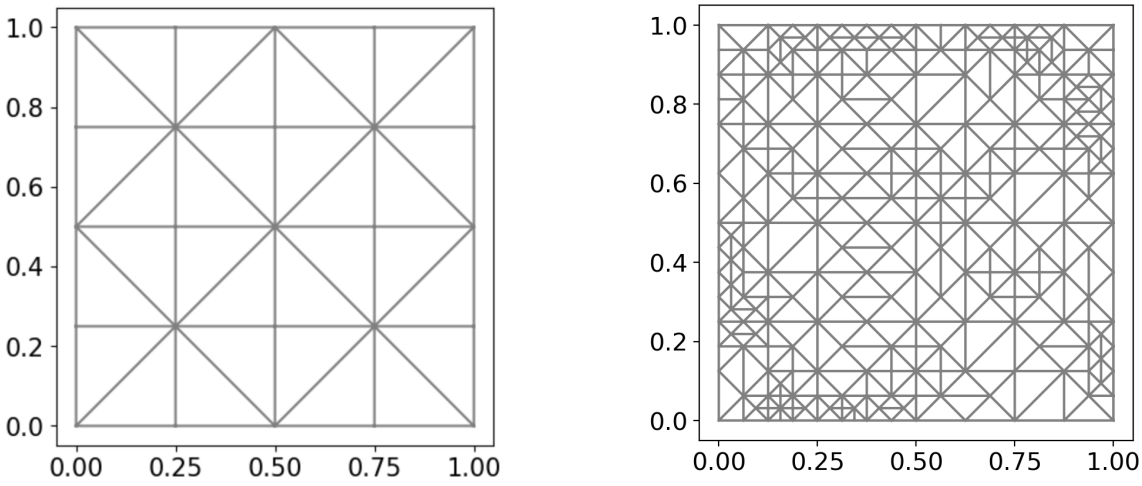
6.8.1 Smooth solution

We consider the variational problem (6.36) and approximate the solution u using a neural network $u_\theta(x,y) = x(1-x)y(1-y)\bar{u}_\theta(x,y)$, where $\bar{u}_\theta$ is a six-layer, fully connected architecture with hyperbolic tangent activation functions, where each hidden layer contains 64 neurons. This ensures that the neural network vanishes on the boundary. The domain Ω is initially partitioned into 32 uniform triangular elements to form the starting mesh $\mathcal{T}_0$ as shown in Figure 6.1 (a). For the test space, we choose a space $V_k := \mathbb{P}_0^1(\mathcal{T}_k) \subset H_0^1(\Omega)$. The corresponding loss function is defined as in (6.16).

To implement Algorithm 1, we set $\epsilon_0 = 0.5, \delta_L = \delta_A = 0.95$, and $K = 120$. We compute integrals using a four-point quadrature rule per element. For training, we employ the Adam optimizer[2] as a nonlinear solver with an adaptive learning rate, initialized at 0.0005 and decayed by a factor of 0.9 every 1,000 iterations. We employ the refinement indicator defined in (6.29) and choose the enriched test space as $\hat{V}_k := \mathbb{P}_0^2(\mathcal{T}_k)$. We refine all marked elements by the Dörfler marking criterion with $\gamma = 0.2$.

Figure 6.1 (b) illustrates the refined mesh, consisting of 448 elements. Figure 6.2 (a) demonstrates that the H^1 error, $\|u - u_\theta\|_{H^1(\Omega)}$, and the square root of the loss coincide throughout the training process. This alignment confirms that controlling the discrepancy between two Riesz representatives enhances the robustness of the training. Furthermore, we evaluate the performance of the adaptive RVPINN algorithm by comparing it to a FEM implementation utilizing the same discrete test space and mesh configurations. This comparison is based on the H^1-norm of the true error. In Figure 6.2 (b), the convergence rate of the neural network approximation for this example is approximately 3, significantly outperforming the optimal adaptive FEM rate of 0.5 (cf. [23]). This superior performance is attributed to the high-expressivity of the neural network, which effectively approximates the smooth solution.

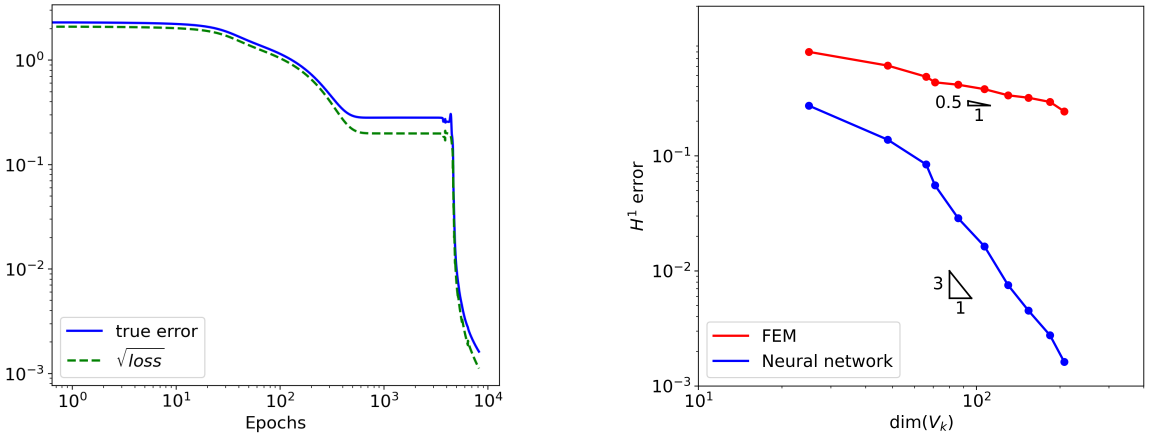
[2]The Adam optimizer is configured with exponential decay rates of $\beta_1 = 0.9$ and $\beta_2 = 0.999$.



(a) Initial mesh (32 elements)

(b) Refined mesh (448 elements)

Figure 6.1: Test-space meshes for a smooth solution.



(a) H^1-error and square root of the loss vs. epochs

(b) H^1-error vs. dimension of test space

Figure 6.2: Results for the Poisson problem on a square domain.

Figure 6.3 illustrates the relationship between the square root of the loss function and the refinement indicator as they reach the specified tolerances $\delta_L^k \epsilon_0$ and $\delta_A^k \epsilon_0$, respectively. While mesh refinement initially increases the loss due to the expansion of the discrete test space, it simultaneously reduces the refinement indicator. Conversely, training the neural network minimizes the loss and generally tends to reduce the refinement indicator, although this reduction is not strictly guaranteed. Empirically, training the neural network using Algorithm 1 ensures a consistent reduction in the true error.

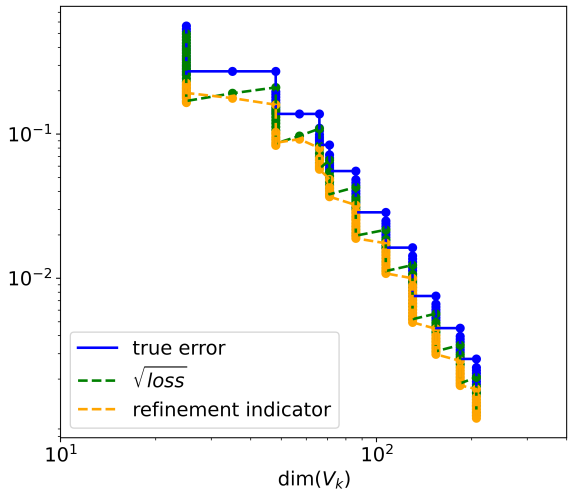
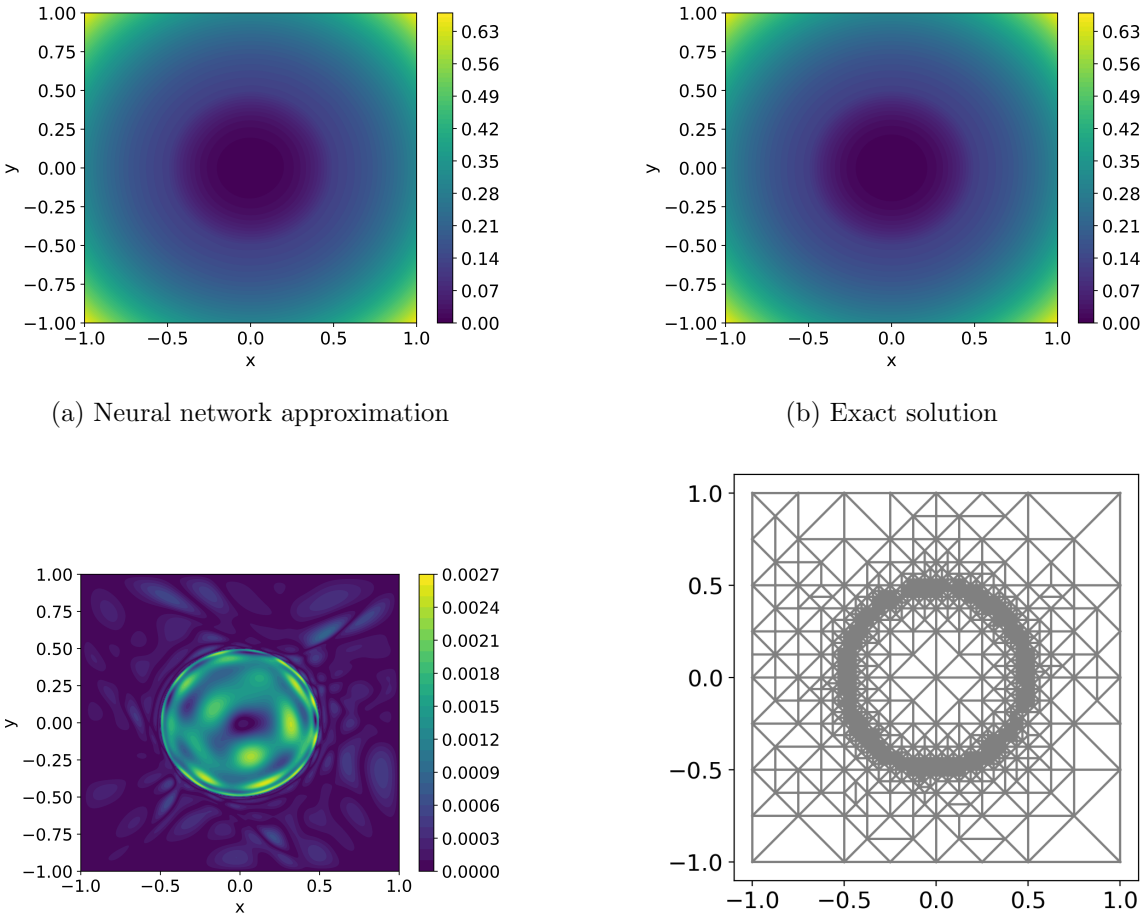


Figure 6.3: H^1-error, square root of the loss, and refinement indicator vs. dimension of test space for smooth solution.

6.8.2 Kink solution

We consider the variational problem (6.40) and approximate the solution u using a neural network $u_\theta(x, y) = (1 - x^2)(1 - y^2)\bar{u}_\theta(x, y) + g(x, y)$, where $\bar{u}_\theta$ is a six-layer, fully connected architecture with hyperbolic tangent activation functions, where each hidden layer contains 64 neurons, and $g(x, y)$ is the Cartesian-coordinate function of the polar-coordinate function in (6.38). This ensures that the boundary condition is strongly imposed into the neural network. The domain Ω is initially partitioned into 32 uniform triangular elements to form the starting mesh $\mathcal{T}_0$. For the test space, we choose a discrete space $V_k := \mathbb{P}_0^1(\mathcal{T}_k) \subset H_0^1(\Omega)$. The corresponding loss function is defined as in (6.16). We set $K = 50$ and utilize the training process and adaptive criteria as described in the previous section.

Figure 6.4 (a) displays the neural network approximation, which is compared to the reference solution (6.39) provided in Figure 6.4 (b). The pointwise absolute errors are shown in Figure 6.4 (c). It is evident that the neural network approximates the solution with high precision, maintaining absolute errors below 3×10^{-3} through out the entire domain. Figure 6.4 (d) illustrates the final mesh, which consists of 4750 elements. As expected, the adaptive refinement is concentrated along the circular interface where the diffusion coefficient is discontinuous and the solution lacks smoothness.



(a) Neural network approximation

(b) Exact solution

(c) Pointwise absolute error

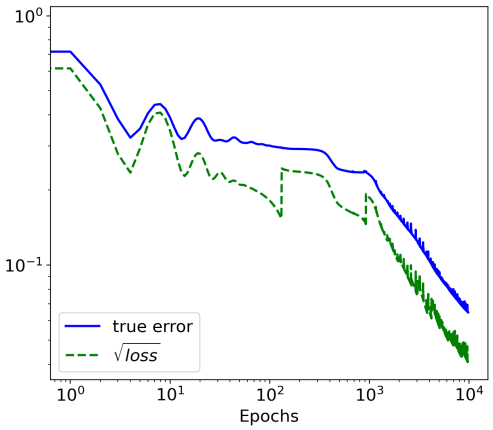
(d) Refined mesh (4750 elements)

Figure 6.4: Results for the interface problem.

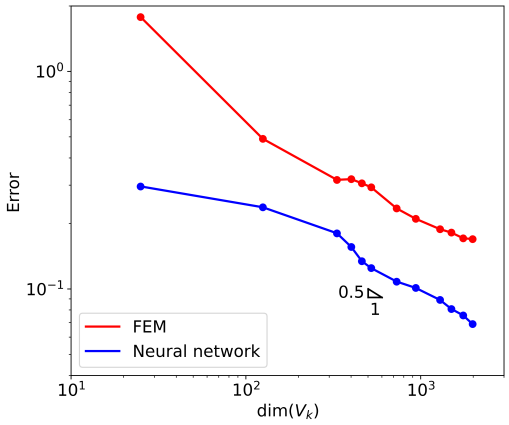
Figure 6.5 (a) demonstrates that the error in the energy norm, $\|a^{1/2}\nabla(u-u_\theta)\|_{L^2(\Omega)}$, and the square root of the loss function remain generally parallel throughout the training process. As shown in Figure 6.5 (b), the observed convergence rate is approximately 0.5, aligning with the optimal rate typically achieved by adaptive FEM. It highlights the performance of the proposed adaptive algorithm, which empirically achieves convergence rates that are competitive with, or superior to, those of standard adaptive FEM.

6.8.3 Singular solution

We consider the variational problem (6.43) and approximate the solution u using a neural network u_θ. The neural network u_θ is a six-layer, fully connected architecture with hyperbolic tangent activation functions, where each hidden layer contains 64

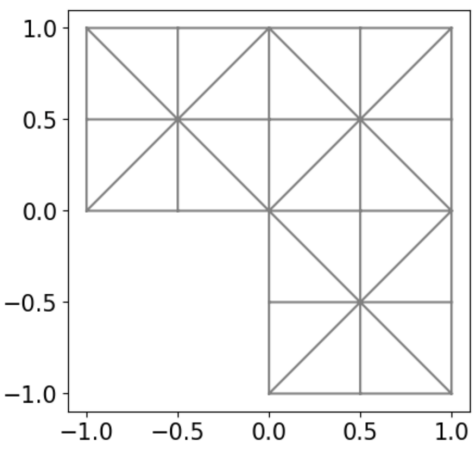


(a) Error in energy norm and square root of the loss vs. epochs

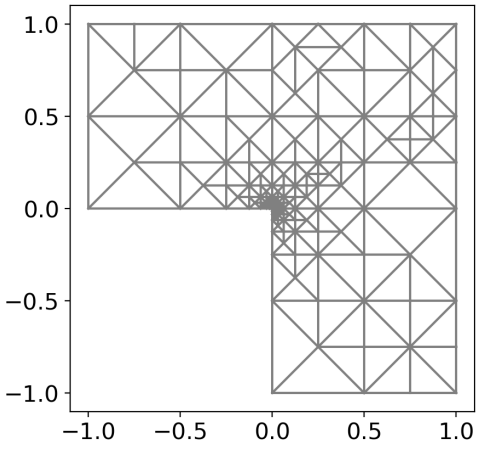


(b) Error in energy norm vs. dimension of test space

Figure 6.5: Results for kink solution.



(a) Initial mesh (24 elements)

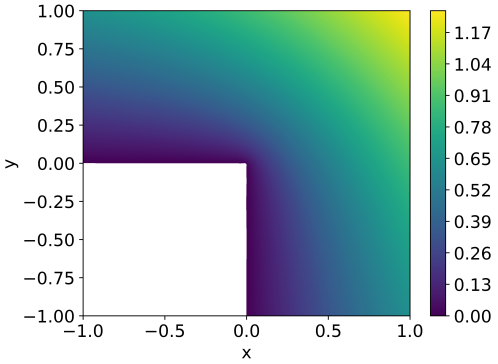


(b) Refined mesh (324 elements)

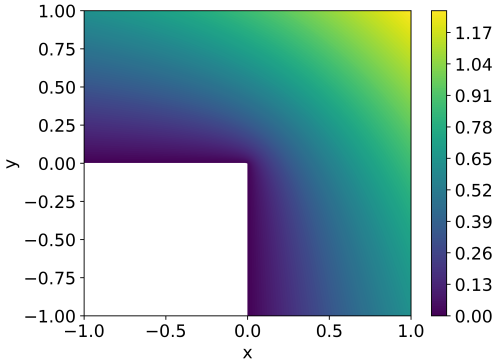
Figure 6.6: Test-space meshes for a singular solution.

neurons. The domain Ω is initially partitioned into 24 uniform triangular elements to form the starting mesh $\mathcal{T}_0$ as shown in Figure 6.6 (a). For the volume test space, we choose a discrete space $V_k^\Omega := \mathbb{P}_0^1(\mathcal{T}_k) \subset H_0^1(\Omega)$. For the boundary test space, we utilize the Raviart-Thomas space of order 1, $V_k^\Gamma := \mathbf{RT}^1(\mathcal{T}_k)$, which is a finite-dimensional subspace of $\mathbf{H}(\mathrm{div};\Omega)$. The loss function is defined in (6.18).

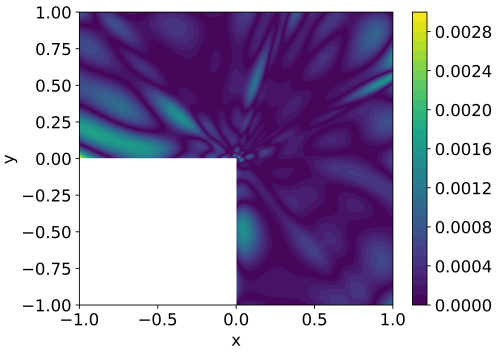
We set $K = 70$ and utilize the training process as described in the previous section. We employ the refinement indicators defined in (6.31) and (6.32). The enriched test spaces are chosen as $\hat{V}_k^\Omega := \mathbb{P}_0^2(\mathcal{T}_k)$ and $\hat{V}_k^\Gamma := \mathbf{RT}^2(\mathcal{T}_k)$. At each level, all marked elements are refined by the Dörfler marking criterion with $\gamma = 0.2$.



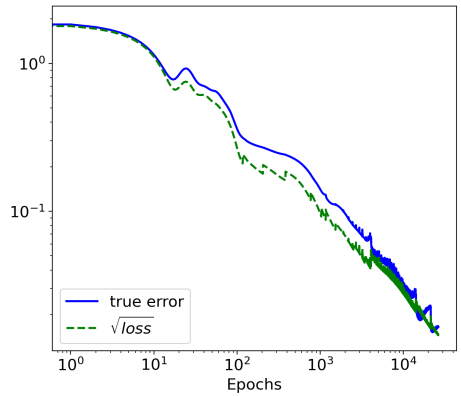
(a) Neural network approximation



(b) Exact solution



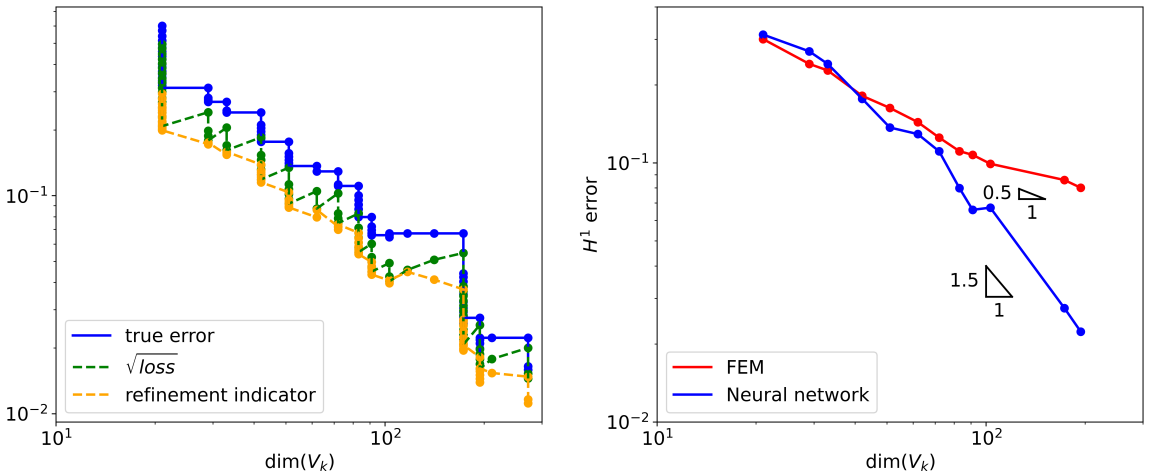
(c) Pointwise absolute error



(d) H^1-error and square root of the loss vs. epochs

Figure 6.7: Results for the Poisson problem on an L-shaped domain.

Figure 6.6 (b) illustrates the refined mesh, consisting of 324 elements. As expected, the adaptive refinement is concentrated around the singularity at the origin $(0, 0)$. Figure 6.7 (a) displays the neural network approximation, while Figure 6.7 (b) provides the reference solution (6.42) for comparison. The pointwise absolute errors are shown in Figure 6.7 (c). It is evident that the neural network approximates the solution with high precision, maintaining absolute errors below 3×10^{-3} across the domain. Figure 6.7 (d) shows the relationship between the square root of the loss and the H^1-norm error. The two curves remain nearly parallel throughout the training process, demonstrating the robustness of the loss. However, after about 10,000 epochs, they begin to diverge slightly. This discrepancy arises from integration error, since refinement is applied only to the test space mesh. One possible approach to overcome this issue is to use separate meshes: one for the trial space, ensuring more accurate integration, and another for the test space, dedicated to adaptive refinement.



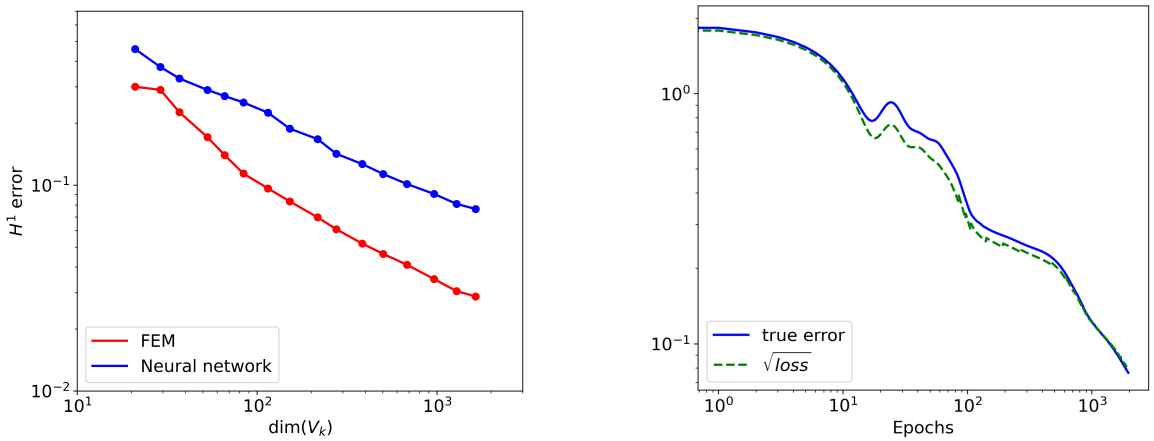
(a) H^1-error, square root of the loss, and refinement indicator vs. dimension of test space

(b) H^1-error for FEM and adaptive RVPINNs vs. dimension of test space.

Figure 6.8: Results for a singular solution.

Figure 6.8 (a) shows the behavior of the square root of the loss function and the refinement indicator as they approach the specified tolerances $\delta_L^k \epsilon_0$ and $\delta_A^k \epsilon_0$, respectively. Consistent with the observations for the smooth solution (Figure 6.3), the mesh refinement initially increases the loss value, while simultaneously decreases the refinement indicator. In contrast, neural network training minimizes the loss and generally leads to a decrease in the indicator. Empirical results confirm that utilizing Algorithm 1 yields a reliable reduction in the true error.

Figure 6.8 (b) displays the H^1-norm error for both adaptive RVPINN method and the FEM against the dimension of the discrete test space. The results demonstrate that adaptive RVPINNs significantly enhance the accuracy of the approximation. Notably, when $\dim(V_k) > 40$, the H^1-norm error of the adaptive RVPINNs is smaller than that of the FEM. Moreover, the adaptive RVPINNs exhibits a convergence rate of approximately 1.5, which outperforms 0.5 rate for the adaptive FEM using continuous piecewise linear test functions. This superior performance is due to the high expressivity of the neural network architecture and the effectiveness of the refinement indicator. However, the exact convergence rate requires further theoretical analysis. It is likely to depend on the neural network architecture, including factors such as depth, width, and activation functions.



(a) H^1-error vs. dimension of test space

(b) H^1-error and loss vs. the number of epochs

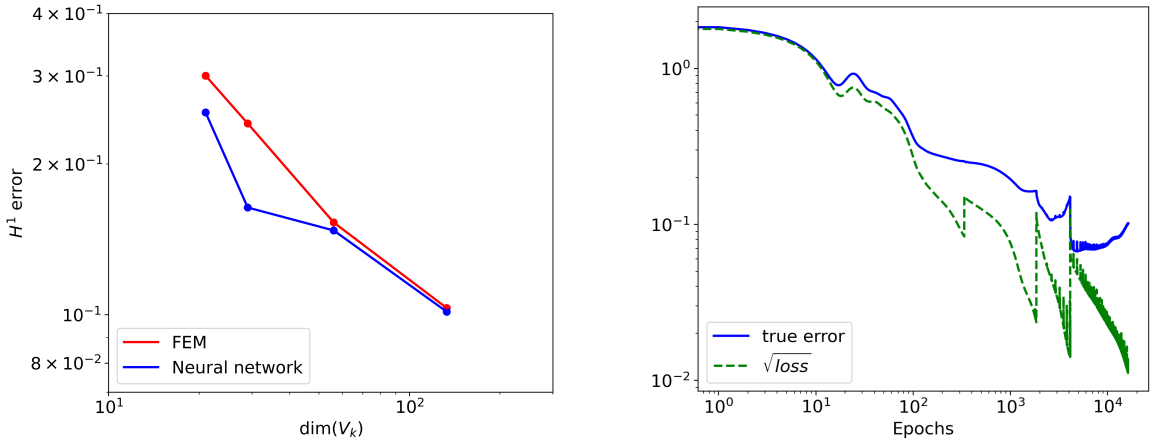
Figure 6.9: Results for $\delta_L = 0.95$ and $\delta_A = 0.9$.

6.8.4 Balance of training and refining

To investigate the importance of balancing between optimization and discretization, we study the interaction between the RVPINN loss and the refinement indicator in the singular solution example. Specifically, we compare two scenarios: one prioritizing mesh refinement over training ($\delta_L > \delta_A$) and another prioritizing excessive training over refinement ($\delta_L < \delta_A$).

Figure 6.9 illustrates the case where $\delta_L = 0.95$ and $\delta_A = 0.9$. In this setting, the refinement indicator is reduced more aggressively than the loss. While this ensures the robustness of the test space, the short training duration on each mesh leads to suboptimal convergence rate. Although the convergence rate remains approximately 0.5, matching the theoretical rate for adaptive FEM, however, the H^1 error may not beat the adaptive FEM due to the insufficient minimization of the neural network residual.

Conversely, Figure 6.10 shows the results for $\delta_L = 0.9$ and $\delta_A = 0.95$. Here, the neural network is trained excessively on relatively coarse mesh. This results in computational waste, as further training does not improve the approximation accuracy once the test space discretization error becomes the dominant factor. Furthermore, if the refinement indicator is not sufficiently small, the true error cannot be effectively controlled leading to instability in convergence.



(a) H^1-error vs. dimension of test space

(b) H^1-error and loss vs. the number of epochs

Figure 6.10: Results for $\delta_L = 0.9$ and $\delta_A = 0.95$.

Our numerical evidence suggests that a balanced approach, specifically setting $\delta_L = \delta_A$, provides a stable and efficient trade-off between minimizing the variational loss and enriching the discrete test space.

6.9 Conclusions

In this chapter, we established an upper bound for neural network approximation errors based on Riesz discrepancy and proved its equivalence to the bound based on the residual remainder term. We identified the error between the discrete and continuous Riesz representatives as a factor to ensure robustness during training with the RVPINN loss function. Furthermore, we introduced idealized and sequential adaptivity strategies. We established their convergence results and proved an error estimate for a quasi-minimizer of the residual norm. Our analysis demonstrated that convergence depends on three primary pillars: the Riesz discrepancy, the approximability of the neural network, and the efficiency of the nonlinear optimizer in minimizing the loss function. We introduced a refinement indicator to minimize the Riesz discrepancy. This led to the development of the test-space adaptive RVPINN framework, which refines the test-space mesh to enhance the accuracy of the discrete Riesz representative in approximating its exact counterpart associated with the variational residual.

We validated the effectiveness of this framework using Poisson problems with smooth and singular solutions and an elliptic interface problem with a kink solution. Numerical results demonstrated that the proposed adaptive approach significantly outperforms the adaptive FEM in terms of the true error. Moreover, we obtained a better convergence rate compared to the adaptive FEM. Furthermore, the results indicated that the proposed method maintains a correlation between the loss function and the true error throughout the training process, demonstrating the robustness of the adaptive strategy.

Potential extensions of this work include developing adaptive strategies that extend the idealized setting by using, for instance, a truncated inner loop ($N < \infty$) to guarantee convergence. In addition, since numerical examples demonstrate that the adaptive process can yield convergence rates superior to those of adaptive FEM, further investigation into the optimal rates for neural network approximations would provide a more comprehensive understanding of the method.

Chapter 7

Conclusions and Future Directions

In this thesis, we have developed, analyzed, and validated a suite of deep-learning-based methodologies for solving Perron-Frobenius operator equations and partial differential equations (PDEs), specifically focusing on Neumann series, invariant densities, and test-space adaptivity. The research progressed from fundamental neural network applications in PDEs and the theory of Perron-Frobenius operators to the development of neural network frameworks for Neumann series and invariant densities of Perron-Frobenius operators and extensions of RVPINNs for PDEs with adaptivity. These methodologies highlight the advantages of neural networks in approximating singular and high-dimensional solutions.

For Neumann series problems, we investigated the approximation of Neumann series associated with non-expansive Perron-Frobenius operators. By establishing the well-posedness of the variational formulations via the Lax-Milgram theorem, we provided a rigorous foundation for our numerical implementations. Our findings demonstrated that both PINNs (for strong forms) and RVPINNs (for variational forms) are effective in solving Neumann series problems. Numerical experiments confirmed that neural networks outperform standard fixed-grid methods for problems exhibiting singular solutions. They asymptotically achieve lower approximation errors with the same number of degrees of freedom. Furthermore, we successfully demonstrated the effectiveness of PINNs to overcome the high-dimensional Neumann series problems. These results have been published in the CMAME journal [100].

The second major contribution of this work involved the approximation of invariant densities. We benchmarked the deep maximum entropy method (DMEM) and

VPINN formulations against classical approaches like the maximum entropy method (MEM) and Ulam's method. Key insights included: Neural networks offer superior flexibility compared to the rigid functional forms of traditional MEM, leading to higher accuracy. Moreover, the derivation of analytical gradients and the use of exact integration within the VPINN framework reduced training times compared to using Gauss-Kronrod integration. While quadratic energy functionals simplify the loss landscape, they are not effective for non-uniform densities. The VPINN approach remains the most robust choice for invariant density problems.

Finally, we addressed the challenges of training neural networks using the RVPINN approach to approximate singular and non-smooth solutions of PDEs. We established theoretical upper bounds for approximation errors and identified the discrepancy between discrete and continuous Riesz representatives as an important factor for robustness. We developed a test-space adaptive framework that refines the test-space mesh to control this discrepancy. When applied to Poisson problems with smooth and singular solutions, our adaptive RVPINN not only outperformed adaptive FEM in the H^1-error but also exhibited a superior convergence rate and maintained a strong correlation between the loss function and actual error. Similarly, for the interface problem, the energy norm error for adaptive RVPINNs was lower than that of adaptive FEM, while achieving a comparable convergence rate.

While this thesis has addressed several key challenges, it also opens several promising avenues for future research. The methodologies developed here, particularly the VPINN framework for invariant densities would offer significant potential for multi-dimensional dynamical systems. Applying this approach to high dimension densities would further highlight the mesh-free advantages of deep learning in systems where traditional discretization becomes computationally challenging. Future studies could benefit from a broader comparative analysis. This includes evaluating the impact of various activation functions and benchmarking against alternative high-order methods, such as spectral methods based on Chebyshev series or Galerkin methods utilizing higher-order polynomials. The a priori error estimates established in this work, particularly those based on local Fortin's conditions require further study to fully benefit the theoretical analysis. Further study of these results could lead to more reliable deep learning solvers for engineering and physics applications.

Appendix A

Fundamental Knowledge

A.1 Measure theory

Definition A.1.1. A collection $\mathfrak{B}$ of subsets of a set $\mathbb{X}$ is a σ-algebra on $\mathbb{X}$ if

1. $\mathbb{X} \in \mathfrak{B}$,
2. for each set A that belongs to $\mathfrak{B}$, the set $\mathbb{X} \backslash A$ belongs to $\mathfrak{B}$,
3. for each finite or infinite sequence $\{A_i\} \subset \mathfrak{B}$, the union $\cup_i A_i$ belongs to $\mathfrak{B}$.

Definition A.1.2. A function $\mu : \mathfrak{B} \to [0, \infty]$ is called a measure if

1. $\mu(\emptyset) = 0$
2. if $\{A_i\} \subset \mathfrak{B}$ is a finite or infinite sequence of pairwise disjoint sets, then $\mu(\cup_i A_i) = \sum_i \mu(A_i)$.

Definition A.1.3. If $\mathfrak{B}$ is a σ-algebra of subsets of $\mathbb{X}$ and μ is a measure on $\mathfrak{B}$, then $(\mathbb{X}, \mathfrak{B}, \mu)$ is called a measure space. Each set in $\mathfrak{B}$ is called a measurable set.

Definition A.1.4. A measure space $(\mathbb{X}, \mathfrak{B}, \mu)$ is called σ-finite if there is a sequence $\{A_k\} \subset \mathfrak{B}$ such that $\mathbb{X} = \cup_{k=1}^{\infty} A_k$ and $\mu(A_k) < \infty$ for all k.

Definition A.1.5. A measure space $(\mathbb{X}, \mathfrak{B}, \mu)$ is called finite if $\mu(\mathbb{X}) < \infty$. In particular, if $\mu(\mathbb{X}) = 1$, it is called a probability space.

The most natural σ-algebra for $\mathbb{X} \subset \mathbb{R}$ is the Borel σ-algebra.

Definition A.1.6. A Borel σ-algebra is the smallest σ-algebra containing intervals. A unique measure μ of Borel σ-algebra is called the Borel measure such that $\mu([a,b]) = b-a$.

Definition A.1.7. Let $(\mathbb{X}, \mathfrak{B})$ be a measurable space, and let μ and ν be positive measures on $(\mathbb{X}, \mathfrak{B})$. Then ν is absolutely continuous with respect to μ if each set A that belongs to $\mathfrak{B}$ and satisfies $\mu(A) = 0$ also satisfies $\nu(A) = 0$. One sometimes writes $\nu \ll \mu$ to indicate that ν is absolutely continuous with respect to μ.

A.2 Lebesgue integration

In this section, we let $(\mathbb{X}, \mathfrak{B}, \mu)$ be a measure space, and denote that $f^+(x) = \max(0, f(x))$ and $f^-(x) = \max(0, -f(x))$. A property is true almost everywhere (a.e.) if it is true except for a subset having measure zero.

Definition A.2.1. A function $f : \mathbb{X} \to \mathbb{R}$ is measurable if $f^{-1}(A) \in \mathfrak{B}$ for every interval $A \subset \mathbb{R}$.

Definition A.2.2. The characteristic function of a set A is χ_A defined by

$$\chi_A(x) := \begin{cases} 1, & x \in A \\ 0, & x \notin A. \end{cases}$$

Definition A.2.3. Let $\{A_i\} \subset \mathfrak{B}$ be a set of disjoint subsets of $\mathbb{X}$. The Lebesgue integral of a simple function Ψ of the form

$$\Psi(x) := \sum_{i=1}^{n} a_i \chi_{A_i}(x).$$

is defined as

$$\int_{\mathbb{X}} \Psi d\mu := \sum_{i=1}^{n} a_i \mu(A_i)$$

Definition A.2.4. Let $f : \mathbb{X} \to \mathbb{R}$ be a nonnegative bounded measurable function, and $\{g_n\}$ be a sequence of simple functions converging uniformly to f. The Lebesgue integral of f is defined by

$$\int_{\mathbb{X}} f d\mu := \lim_{n\to\infty} \int_{\mathbb{X}} g_n d\mu$$

Definition A.2.5. Let $f : \mathbb{X} \to \mathbb{R}$ be a nonnegative unbounded measurable function. Define

$$f_M(x) := \begin{cases} f(x), & \text{if } 0 \le f(x) \le M \\ M, & \text{if } M < f(x). \end{cases}$$

Then the Lebesgue integral of f is defined by

$$\int_{\mathbb{X}} f(x)d\mu := \lim_{M\to\infty} \int_{\mathbb{X}} f_M(x)d\mu.$$

Definition A.2.6. Let $f : \mathbb{X} \to \mathbb{R}$ be a measurable function. The Lebesgue integral of f is defined as

$$\int_{\mathbb{X}} f d\mu := \int_{\mathbb{X}} f^+ d\mu - \int_{\mathbb{X}} f^- d\mu$$

if at least one of the terms $\int_{\mathbb{X}} f^+ d\mu, \int_{\mathbb{X}} f^- d\mu$ is finite.

Definition A.2.7. A function f is called integrable if both $\int_{\mathbb{X}} f^+ d\mu$ and $\int_{\mathbb{X}} f^- d\mu$ are finite.

Definition A.2.8. Let $A \in \mathfrak{B}$. The Lebesgue integral of f over A is defined by

$$\int_A f d\mu := \int_{\mathbb{X}} f \cdot \chi_A d\mu.$$

Remark A.2.1. If f is a bounded function on a closed interval $[a, b]$ and μ is a Borel measure, the Lebesgue integral and the Riemann integral are the same, i.e.,

$$\int_{[a,b]} f d\mu = \int_a^b f(x)dx.$$

Theorem A.2.1 (Change of variables)**.** *Let $\Omega \subset \mathbb{R}^d$ be an open set and $S : \Omega \to \mathbb{R}^d$ be a one-to-one differentiable mapping with non-vanishing Jacobian J_S. Then*

$$\int_{S(\Omega)} u d\mu = \int_\Omega (u \circ S)|J_S| d\mu$$

for any continuous function $u : S(\Omega) \to \mathbb{R}^d$ such that their support $\overline{\{x \in S(\Omega) : u(x) \neq 0\}}$ is a compact subset of $S(\Omega)$.

Theorem A.2.2 (Lebesgue dominated convergence theorem [64])**.** *Let $f, g : \mathbb{X} \to \mathbb{R}$ be measurable functions and $f_n : \mathbb{X} \to \mathbb{R}$ be measurable functions such that $|f_n(x)| \le g(x)$ and $\{f_n(x)\}$ converges to $f(x)$ almost everywhere. If g is integrable, then f and f_n are also integrable and*

$$\lim_{n\to\infty} \int_{\mathbb{X}} f_n d\mu = \int_{\mathbb{X}} f d\mu.$$

Corollary A.2.1 ([64]). *For every integrable function f, there is a sequence of simple functions*

$$f_n(x) = \sum_i \lambda_{i,n} \chi_{A_{i,n}}(x)$$

such that

$$\lim_{n\to\infty} f_n(x) = f(x) \quad a.e. \qquad and \qquad |f_n(x)| \le |f(x)|.$$

Thus, by the Lebesgue dominated convergence theorem, we have

$$\lim_{n\to\infty} \int_{\mathbb{X}} f_n d\mu = \int_{\mathbb{X}} f d\mu.$$

Theorem A.2.3 (Radon-Nikodym Theorem [30]). *Let $(\mathbb{X}, \mathfrak{B})$ be a measurable space, and let μ and ν be $\sigma-$finite measures on $(\mathbb{X}, \mathfrak{B})$. If ν is absolutely continuous with respect to μ, then there is $\mathfrak{B}$-measurable function $g : \mathbb{X} \to [0, \infty)$ such that $\nu(A) = \int_A g d\mu$ hold for each $A \in \mathfrak{B}$. The function g is unique up to $\mu-$almost everywhere equality.*

Definition A.2.9. Let p be a real number such that $1 \le p < \infty$. The $L^p(\mathbb{X})$ space is a family of all measurable functions f satisfying

$$\int_{\mathbb{X}} |f(x)|^p d\mu < \infty.$$

The norm of f is defined as

$$\|f\|_{L^p} := \left[\int_{\mathbb{X}} |f(x)|^p d\mu \right]^{1/p}.$$

Theorem A.2.4 (Riesz-Fisher). *The space L^p for $1 \le p < \infty$ is a Banach space.*

Definition A.2.10. The $L^\infty(\mathbb{X})$ space is a family of all bounded a.e. measurable functions f, i.e., there is some $M \ge 0$, called an essential upper bound, for which

$$|f(x)| \le M \qquad \text{for almost all } x \in \mathbb{X}.$$

The norm of f, $\|f\|_{L^\infty}$, is defined to be the infimum of the essential upper bounds for f.

Theorem A.2.5 (Hölder's Inequality). *Let $1 < p < \infty$ and q be the conjugate of p, i.e., $\frac{1}{p} + \frac{1}{q} = 1$. If $f \in L^p(\mathbb{X})$ and $g \in L^q(\mathbb{X})$, then their product $f \cdot g$ is integrable over $\mathbb{X}$ and*

$$\int_{\mathbb{X}} |f \cdot g| d\mu \le \|f\|_{L^p} \|g\|_{L^q}.$$

The special case of Holder's Inequality when $p = q = 2$ is called Cauchy-Schwarz Inequality.

Proposition A.2.1. Let $f, g \in L^p(\mathbb{X})$ and $\alpha \in \mathbb{R}$. Then

1. $\|\alpha f\|_{L^p} = |\alpha| \cdot \|f\|_{L^p}$
2. $\|f + g\|_{L^p} \leq \|f\|_{L^p} + \|g\|_{L^p}$.

Proposition A.2.2. If $(\mathbb{X}, \mathfrak{B}, \mu)$ is a finite measure space and $1 \leq p_1 < p_2 \leq \infty$, then

$$\|f\|_{L^{p_1}} \leq c\|f\|_{L^{p_2}} \quad \text{for every } f \in L^{p_2}$$

where c is a constant depends on $\mu(\mathbb{X})$. This implies $L^{p_2} \subset L^{p_1}$.

A.3 Product space

Let $(\mathbb{X}_1, \mathfrak{B}_1, \mu_1), \ldots, (\mathbb{X}_d, \mathfrak{B}_d, \mu_d)$ be measure spaces. Define

$$\mathbb{X} = \mathbb{X}_1 \times \cdots \times \mathbb{X}_d, \tag{A.1}$$

$\mathfrak{B}$ to be the smallest σ-algebra containing all sets of the form

$$A_1 \times \cdots \times A_d, \quad A_i \in \mathfrak{B}_i, i = 1, \ldots, d \tag{A.2}$$

and

$$\mu(A_1 \times \cdots \times A_d) = \mu_1(A_1) \cdots \mu_d(A_d). \tag{A.3}$$

Theorem A.3.1. *For $\mathbb{X}, \mathfrak{B}$ and μ defined by Equations (A.1) - (A.3), there exists a unique extension of μ to a measure defined on $\mathfrak{B}$.*

Definition A.3.1. The measure space guaranteed by Theorem A.3.1 is called the product space. The measure μ is called the product measure.

Theorem A.3.2 (Fubini's Theorem)**.** *Let $(\mathbb{X}, \mathfrak{B}, \mu)$ be the product space formed by $(\mathbb{X}_1, \mathfrak{B}_1, \mu_1)$ and $(\mathbb{X}_2, \mathfrak{B}_2, \mu_2)$. Assume $f : \mathbb{X} \to \mathbb{R}$ be a μ integrable function. Then*

$$\iint_{\mathbb{X}} f d\mu = \int_{\mathbb{X}_1} \left(\int_{\mathbb{X}_2} f d\mu_2 \right) d\mu_1.$$

A.4 Vector spaces and Hilbert spaces

This section reviews the basic concepts of vector spaces, Hilbert spaces, and the Riesz representation theorem. For further details, we refer the reader to [16, Chapter 2].

Definition A.4.1. A *linear form* (also known as *linear functional*) on a vector space V is a mapping $l : V \to \mathbb{R}$ such that

1. $l(v + w) = l(v) + l(w) \quad \forall v, w \in V$
2. $l(\alpha v) = \alpha l(v) \quad \forall \alpha \in \mathbb{R}, v \in V$.

Definition A.4.2. A *bilinear form* on a vector space V is a mapping $b : V \times V \to \mathbb{R}$ such that each of the maps $v \mapsto b(v, w)$ and $w \mapsto b(v, w)$ is a linear form on V.

Definition A.4.3. Let V be a vector space over $\mathbb{R}$. An *inner product* on V is a bilinear form $(\cdot, \cdot)_V : V \times V \to \mathbb{R}$ such that

1. $(v, w)_V = (w, v)_V \quad \forall v, w \in V$ (symmetric)
2. $(v, v)_V \geq 0 \quad \forall v \in V$ (positive)
3. $(v, v)_V \neq 0 \quad \forall v \neq 0$ (definite).

The inner product induces a natural norm $\|u\|_V = \sqrt{(u, u)_V}$.

Definition A.4.4. A vector space V together with an inner product $(\cdot, \cdot)_V$ defined on it is called an *inner product space*.

Definition A.4.5. Given a vector space V, a *norm* $\|\cdot\|_V$ is a function from V to the non-negative real numbers such that

1. $\|v\|_V = 0$ if and only if $v = 0$
2. $\|\alpha v\|_V = |\alpha| \|v\|_V \quad \forall \alpha \in \mathbb{R}, v \in V$
3. $\|v + w\|_V \leq \|v\|_V + \|w\|_V \quad \forall v, w \in V$ (the triangle inequality).

A norm induces a distance, or metric, $d(v, w) = \|v - w\|_V$ for $v, w \in V$.

Definition A.4.6. A *normed space* is a vector space V endowed with the metric $d(\cdot, \cdot)$.

Definition A.4.7. For a normed space V, a *Cauchy sequence* is a sequence $\{v_j\}$ of elements of V such that $\|v_j - v_k\|_V \to 0$ as $j, k \to \infty$.

Definition A.4.8. A normed space V is *complete* if every Cauchy sequence $\{v_j\}$ has a limit $v \in V$, i.e., $\|v - v_j\|_V \to 0$ as $j \to \infty$.

Definition A.4.9. A *Banach space* is a complete normed space.

Definition A.4.10. A *Hilbert space* H is an inner product space that is complete with respect to the metric induced by its norm $\|\cdot\|_H$.

Definition A.4.11. A linear functional l is bounded if there is a finite constant C such that $|l(v)| \leq C\|v\|_V \quad \forall v \in V$.

Definition A.4.12. Let V be a vector space. The *dual space* V' is the space of bounded linear functionals on V.

Proposition A.4.1. For a bounded linear functional l on a Banach space B, the following *dual norm* is always finite:

$$\|l\|_{B'} := \sup_{0 \neq v \in B} \frac{l(v)}{\|v\|_B}.$$

Theorem A.4.1 (Riesz Representation Theorem)**.** *Let H be a real Hilbert space and let $l \in H'$. Then there exists a unique element $\phi \in H$ such that*

$$l(v) = (\phi, v)_H \quad \forall v \in H. \tag{A.4}$$

Furthermore, we have $\|\phi\|_H = \|l\|_{H'}$.

The Riesz representation theorem guarantees that every bounded linear functional can be uniquely identified with an element within the space itself.

A.5 Sobolev spaces

This section reviews the foundational concept of Sobolev spaces. For further details, we refer the reader to [16, Chapter 1] and [12, Chapter 2].

To formalize the concept of weak or variational solutions to partial differential equations (PDEs), classical derivatives must be generalized using weak derivatives.

A.5.1 Weak derivatives

Let $\Omega \subset \mathbb{R}^d$ be an open domain, and let $C^\infty(\Omega)$ denote the space of infinitely differentiable functions in Ω. Let $\alpha := (\alpha_1, \ldots, \alpha_d)$ be a multi-index of order $|\alpha| = \sum_{i=1}^d \alpha_i$.

Definition A.5.1. For $\phi \in C^\infty(\Omega)$, the usual partial derivative

$$\left(\frac{\partial}{\partial x_1}\right)^{\alpha_1} \cdots \left(\frac{\partial}{\partial x_d}\right)^{\alpha_d} \phi$$

is denoted by $D^\alpha \phi$.

Definition A.5.2. Let $C_c^\infty(\Omega)$ denote the space of infinitely differentiable functions with compact support in Ω.

Definition A.5.3. The set of *locally integrable* functions is denoted by

$$L^1_{\text{loc}}(\Omega) := \{f : f \in L^1(K) \quad \forall \text{ compact } K \subset \Omega\}.$$

Definition A.5.4. Let $u, v \in L^1_{\text{loc}}(\Omega)$. We say that v is the α-th *weak derivative* of u, denoted as $D^\alpha u = v$, if

$$\int_\Omega v\, \phi \, dx = (-1)^{|\alpha|} \int_\Omega u\, D^\alpha \phi \, dx \quad \forall \phi \in C_c^\infty(\Omega). \tag{A.5}$$

A.5.2 Sobolev spaces and norms

Definition A.5.5. For an integer $k \geq 0$ and $1 \leq p \leq \infty$, the *Sobolev space* $W^{k,p}(\Omega)$ consists of all functions $u \in L^p(\Omega)$ whose weak derivatives up to order k also belong to $L^p(\Omega)$. That is

$$W^{k,p}(\Omega) := \{u \in L^p(\Omega) : D^\alpha u \in L^p(\Omega) \text{ for all } |\alpha| \leq k\}.$$

When $p = 2$, the Sobolev space forms a Hilbert space, conventionally denoted as $H^k(\Omega) := W^{k,2}(\Omega)$. The inner product and corresponding norm on $H^k(\Omega)$ are defined as:

$$(u, v)_{H^k(\Omega)} = \sum_{|\alpha| \leq k} \int_\Omega D^\alpha u\, D^\alpha v \, dx, \quad \|v\|_{H^k(\Omega)} = \sqrt{(v, v)_{H^k(\Omega)}}. \tag{A.6}$$

In particular, $H^1(\Omega) := \{v \in L^2(\Omega) : \nabla v \in L^2(\Omega; \mathbb{R}^d)\}$ is the Sobolev space of functions with square-integrable first-order weak derivatives, equipped with the norm

$$\|v\|_{H^1(\Omega)} := \sqrt{\|v\|^2_{L^2(\Omega)} + \|\nabla v\|^2_{L^2(\Omega;\mathbb{R}^d)}} \quad \forall v \in H^1(\Omega).$$

A.5.3 Vector-valued function spaces

Let $\Omega \subset \mathbb{R}^d$ for $d \geq 1$. Let $L^2(\Omega; \mathbb{R}^d)$ denote the space of vector-valued functions whose components are in $L^2(\Omega)$, equipped with the inner product:

$$(\mathbf{v}, \mathbf{w})_{L^2(\Omega;\mathbb{R}^d)} := \int_\Omega \mathbf{v} \cdot \mathbf{w} d\mu \quad \forall \mathbf{v}, \mathbf{w} \in L^2(\Omega; \mathbb{R}^d)$$

and the induced norm:

$$\|\mathbf{v}\|_{L^2(\Omega;\mathbb{R}^d)} := \left(\int_\Omega \|\mathbf{v}\|^2 d\mu \right)^{1/2} \quad \forall \mathbf{v} \in L^2(\Omega; \mathbb{R}^d).$$

We also define the space:

$$\mathbf{H}(\mathrm{div}; \Omega) := \{\mathbf{v} \in L^2(\Omega; \mathbb{R}^d) : \nabla \cdot \mathbf{v} \in L^2(\Omega)\},$$

equipped with the inner product:

$$(\mathbf{v}, \mathbf{w})_{\mathbf{H}(\mathrm{div};\Omega)} := (\mathbf{v}, \mathbf{w})_{L^2(\Omega;\mathbb{R}^d)} + (\nabla \cdot \mathbf{v}, \nabla \cdot \mathbf{w})_{L^2(\Omega)} \quad \forall \mathbf{v}, \mathbf{w} \in \mathbf{H}(\mathrm{div}; \Omega),$$

and the induced norm:

$$\|\mathbf{v}\|_{\mathbf{H}(\mathrm{div};\Omega)} := \left(\|\mathbf{v}\|^2_{L^2(\Omega;\mathbb{R}^d)} + \|\nabla \cdot \mathbf{v}\|^2_{L^2(\Omega)} \right)^{1/2} \quad \forall \mathbf{v} \in \mathbf{H}(\mathrm{div}; \Omega).$$

For $u \in H^1(\Omega)$, the trace $u|_\Gamma$ is defined in the dual space $(\mathbf{H}(\mathrm{div}; \Omega))'$ via the generalized Green's identity:

$$\langle u|_\Gamma, \mathbf{v} \rangle_\Gamma := (\nabla \cdot \mathbf{v}, u)_{L^2(\Omega)} + (\mathbf{v}, \nabla u)_{L^2(\Omega;\mathbb{R}^d)} \quad \forall \mathbf{v} \in \mathbf{H}(\mathrm{div}; \Omega),$$

where $\langle \cdot, \cdot \rangle_\Gamma$ is the dual paring between $(\mathbf{H}(\mathrm{div}; \Omega))'$ and $\mathbf{H}(\mathrm{div}; \Omega)$. The corresponding trace space is defined as $H^{1/2}(\Gamma) := \{u|_\Gamma : u \in H^1(\Omega)\}$, which is equipped with the trace norm:

$$\|\hat{u}\|_{1/2,\Gamma} = \sup_{0 \neq \mathbf{v} \in \mathbf{H}(\mathrm{div};\Omega)} \frac{\langle \hat{u}, \mathbf{v} \rangle_\Gamma}{\|\mathbf{v}\|_{\mathbf{H}(\mathrm{div};\Omega)}} \quad \forall \hat{u} \in H^{1/2}(\Gamma).$$

For brevity, we denote $\|u\|_{1/2,\Gamma}$ in place of $\|u|_\Gamma\|_{1/2,\Gamma}$ for any $u \in H^1(\Omega)$. Furthermore, let

$$H^1_0(\Omega) := \{v \in H^1(\Omega) : v|_\Gamma = 0\}$$

be the subspace of functions with vanishing trace, equipped with the inner product $(v, w)_{H^1_0} := (\nabla v, \nabla w)_{L^2(\Omega;\mathbb{R}^d)}$ and the associated energy norm $\|v\|_{H^1_0} := \|\nabla v\|_{L^2(\Omega;\mathbb{R}^d)}$. We denote by $C^0(\bar{\Omega})$ the space of functions that are continuous over the closed domain $\bar{\Omega}$.

A.5.4 Raviart–Thomas spaces

Given a partition of the domain Ω into polygonal elements, a conforming approximation of $H^1(\Omega)$ is a space of continuous functions defined by a finite number of parameters (or degrees of freedom). The last condition is usually met by using a space of piecewise polynomial functions.

For triangular elements, it is usual and convenient to use piecewise polynomial functions on T. We denote $\mathbb{P}^p(T)$ the space of polynomials of total degree at most p. The space of continuous piecewise polynomials of degree p defined over $\mathcal{T}_k$ that vanish on the boundary Γ is given by:

$$\mathbb{P}_0^p(\mathcal{T}_k) := \{v \in C^0(\bar{\Omega}) : v|_T \in \mathbb{P}^p(T) \quad \forall T \in \mathcal{T}_k, v|_\Gamma = 0\}. \tag{A.7}$$

We also define polynomial spaces on the edges of the elements

$$\mathbb{E}^p(\partial T) := \{v : v \in L^2(\partial T), v|_{e_i} \in \mathbb{P}^p(e_i) \quad \forall \text{ edges } e_i \text{ of } T\}.$$

On a triangular element $T \subset \mathbb{R}^d$, the *Raviart-Thomas space* of order p (cf. [12, Section 2.3.1]) is defined by

$$\mathbf{RT}^p(T) := \{\mathbf{p} \in ((\mathbb{P}^p(T))^d + \mathbf{x}\mathbb{P}^p(T)) : \mathbf{p} \cdot \mathbf{n} \in \mathbb{E}^p(\partial T)\},$$

where $\mathbf{x} \in T$ and $\mathbf{n}$ is the normal direction to ∂T. The global Raviart–Thomas space over the mesh $\mathcal{T}_k$ is constructed by

$$\mathbf{RT}^p(\mathcal{T}_k) = \{\mathbf{q} \in H(\mathrm{div}, \Omega) : \mathbf{q}|_T \in \mathbf{RT}^p(T) \quad \forall T \in \mathcal{T}_k\}. \tag{A.8}$$

A.6 Adaptive finite element method

A finite element approximation of a PDE in a variational form is to find $u_k \in V_k \subset V$ such that

$$b(u_k, v_k) = l(v_k) \quad \forall v_k \in V_k,$$

where $b(\cdot, \cdot)$ is a bilinear form on $V \times V$ and $l(\cdot) \in V'$ is the bounded linear functional corresponding to the underlying PDE. To resolve localized features, singularities, or sharp gradients without unnecessarily increasing the computational cost, an adaptive finite element method (AFEM) [29] is employed.

Let V_k be the continuous piecewise polynomial finite element space associated with the triangulation $\mathcal{T}_k$. Starting from an initial triangulation $\mathcal{T}_0$, the adaptive loop sets $k := 0$ and iterates the following procedures:

$$\text{SOLVE} \longrightarrow \text{ESTIMATE} \longrightarrow \text{MARK} \longrightarrow \text{REFINE}.$$

The SOLVE step computes the discrete finite element solution u_k on the mesh $\mathcal{T}_k$. Next, the ESTIMATE step computes the local error indicators $\eta(u_k, T)$ for each $T \in \mathcal{T}_k$, derived from the global a posteriori error estimator $\eta(u_k, \mathcal{T}_k) := \left(\sum_{T\in\mathcal{T}_k} \eta^2(u_k, T)\right)^{1/2}$. The MARK step then selects the elements exhibiting the largest error indicators. A common approach is *Dörfler marking*, which identifies a minimal subset $\mathcal{M}_k \subset \mathcal{T}_k$ satisfying

$$\sum_{T\in\mathcal{M}_k} \eta^2(u_k, T) \geq \gamma \sum_{T\in\mathcal{T}_k} \eta^2(u_k, T) \quad \text{for a chosen threshold } \gamma \in (0, 1]. \tag{A.9}$$

Finally, the REFINE step subdivides the elements in $\mathcal{M}_k$ (along with necessary neighboring elements to maintain mesh conformity, e.g., via newest vertex bisection) to generate a finer mesh $\mathcal{T}_{k+1}$. The index k is then increased, and the adaptive loop repeats until a specified error tolerance or the maximum iteration count is reached.

A.7 Automatic differentiation

In neural network training, derivative evaluations must be computationally efficient. Automatic Differentiation (AD) [6] achieves this by evaluating derivatives up to machine precision without the truncation errors of numerical differentiation or the expression explosion of symbolic differentiation.

AD decomposes mathematical functions into a computational graph composed of intermediate variables and elementary operations $(+, -, \times, \div, \sin, \exp, \text{ etc.})$, and systematically applies the chain rule. A computational graph is defined by a directed acyclic graph (DAG) $G = (V, E)$, where V is the set of intermediate variables and E is the set of directed edges representing operational dependencies. AD operates primarily in forward and reverse modes.

A.7.1 Forward mode

Forward mode computes the derivative of all intermediate variables with respect to a single independent input variable simultaneously with the evaluation of the function. For a function $f : \mathbb{R}^n \to \mathbb{R}^m$, every intermediate variable v_i is paired with a tangent (dual) variable defined as: $\dot{v}_i = \frac{\partial v_i}{\partial x_j}$ where x_j is the specific input variable of interest. Forward mode is computationally efficient for functions where $n \ll m$, as a single forward pass evaluates derivatives with respect to one input across all outputs.

A.7.2 Reverse mode

Reverse mode (often called backpropagation in deep learning contexts) evaluates derivatives in reverse order of the function's execution graph. It proceeds in two phases: a forward pass that computes and stores all intermediate node values, followed by a backward pass that computes gradients. For a scalar function $f : \mathbb{R}^n \to \mathbb{R}$, reverse mode defines an *adjoint* variable $\bar{v}_i$ for each intermediate variable v_i as: $\bar{v}_i = \frac{\partial f}{\partial v_i}$.

We define the computational graph as a DAG where directed edges (j, i) represent operational dependencies, i.e., variable v_i directly depends on v_j. Let $\mathcal{S}(j) = \{i \,|\, (j, i) \in E\}$ denote the set of immediate successor nodes that directly depend on v_j. By applying the multivariable chain rule backward through the graph, the adjoint of node j is accumulated via:

$$\bar{v}_j = \sum_{i \in \mathcal{S}(j)} \bar{v}_i \frac{\partial v_i}{\partial v_j}.$$

Reverse mode evaluates the gradient of a scalar output with respect to all n parameters in a single backward pass, making it the standard choice for functions $f : \mathbb{R}^n \to \mathbb{R}$ where $n \gg 1$, such as neural network loss functions.

Appendix B

Invariant Density of Logistic Map

The logistic map S_r is given by $S_r(x) := rx(1-x)$. We will show that

$$f_4(x) = \frac{1}{\pi}\frac{1}{\sqrt{x(1-x)}}$$

is the invariant density of the dynamical system with the logistic map at $r = 4$ [90]. It is straightforward to show that f_4 satisfies the equation

$$\mathcal{P}f_4(x) = f_4(x),$$

where $\mathcal{P}$ is the Perron-Frobenius of this dynamical system defined by

$$\mathcal{P}f(x) := \frac{d}{dx}\int_{S_4^{-1}([0,x])} f(s)ds.$$

Indeed,

$$\int_{S_4^{-1}([0,x])} f(s)ds = \int_0^{\frac{1}{2}(1-\sqrt{1-x})} f(s)ds + \int_{\frac{1}{2}(1+\sqrt{1-x})}^1 f(s)ds.$$

So

$$\begin{aligned}
\mathcal{P}f_4(x) &= f_4\left(\frac{1}{2}(1-\sqrt{1-x})\right)\frac{d}{dx}\left(\frac{1}{2}(1-\sqrt{1-x})\right) \\
&\quad - f_4\left(\frac{1}{2}(1+\sqrt{1-x})\right)\frac{d}{dx}\left(\frac{1}{2}(1+\sqrt{1-x})\right) \\
&= \frac{1}{4\sqrt{1-x}}\left(f_4\left(\frac{1}{2}(1-\sqrt{1-x})\right) + f_4\left(\frac{1}{2}(1+\sqrt{1-x})\right)\right) \\
&= \frac{1}{4\sqrt{1-x}}\left(\frac{2}{\pi\sqrt{x}} + \frac{2}{\pi\sqrt{x}}\right) \\
&= \frac{1}{\pi}\frac{1}{\sqrt{x(1-x)}}.
\end{aligned}$$

Thus, f_4 is the invariant density.

Appendix C

Proofs for Chapter 4

C.1 Well-posedness of variational problem

In the following, we show that the variational problem (4.5) satisfies the assumptions of Banach-Nečas-Babŭska Theorem, so the existence and uniqueness of the solution are verified.

Theorem C.1.1 (Well-posedness of variational problem in $L^p - L^q$ spaces)**.** *If $U = L^p(\Omega)$ and $V = L^q(\Omega)$, where $1 < p, q < \infty$ and $\frac{1}{p} + \frac{1}{q} = 1$. The variational problem (4.5) satisfies the following:*

1. *Boundedness of b:* $|b(u,v)| \leq (1+\alpha)\|u\|_{L^p}\|v\|_{L^q}, \quad \forall u \in L^p(\Omega), v \in L^q(\Omega),$
2. *Boundedness of l:* $|l(v)| \leq \|f_0\|_{L^p}\|v\|_{L^q}, \quad \forall v \in L^q(\Omega),$
3. *Inf-sup stability:* $\sup_{0 \neq v \in L^q(\Omega)} \frac{b(u,v)}{\|v\|_{L^q}} \geq (1-\alpha)\|u\|_{L^p}, \quad \forall u \in L^p(\Omega),$
4. *Adjoint injectivity:* $b(u,v) = 0 \quad \forall u \in L^p(\Omega) \Longrightarrow v = 0.$

Hence, the variational problem (4.5) has a unique solution.

Proof. 1. Boundedness of b: For $u \in L^p(\Omega)$ and $v \in L^q(\Omega)$, by using Hölder's inequality, the triangle inequality, and the assumption (A2), we have

$$\begin{aligned} |b(u,v)| = \left| \int_\Omega (u - \alpha\mathcal{P}u) v d\mu \right| &\leq \int_\Omega |(u - \alpha\mathcal{P}u)v| d\mu \leq \|u - \alpha\mathcal{P}u\|_{L^p}\|v\|_{L^q} \\ &\leq \left(\|u\|_{L^p} + \alpha\|\mathcal{P}u\|_{L^p}\right)\|v\|_{L^q} \leq (1+\alpha)\|u\|_{L^p}\|v\|_{L^q}. \end{aligned}$$

2. Boundedness of l: Since $f_0 \in L^p(\Omega), v \in L^q(\Omega)$, by using Hölder's inequality, we have

$$|l(v)| = \left| \int_\Omega f_0 v d\mu \right| \leq \|f_0\|_{L^p} \|v\|_{L^q}.$$

3. Inf-sup stability: For $u \in L^p(\Omega)$, by substituting $v = (u - \alpha \mathcal{P}u)^{p-1} \in L^q(\Omega)$, and using the reverse triangle inequality and assumption (A2), we have

$$\begin{aligned}
\sup_{0 \neq v \in L^q(\Omega)} \frac{b(u,v)}{\|v\|_{L^q}} &= \sup_{0 \neq v \in L^q(\Omega)} \frac{\int_\Omega (u - \alpha \mathcal{P}u) v d\mu}{\|v\|_{L^q}} \\
&\geq \frac{\int_\Omega (u - \alpha \mathcal{P}u)^p d\mu}{\|(u - \alpha \mathcal{P}u)^{p-1}\|_{L^q}} = \frac{\|u - \alpha \mathcal{P}u\|_{L^p}^p}{\|u - \alpha \mathcal{P}u\|_{L^p}^{p/q}} \\
&= \|u - \alpha \mathcal{P}u\|_{L^p} \geq \|u\|_{L^p} - \alpha \|\mathcal{P}u\|_{L^p} \geq (1 - \alpha)\|u\|_{L^p}.
\end{aligned}$$

4. Adjoint injectivity: Let $v \in L^q(\Omega)$ and assume that $b(u, v) = 0 \quad \forall u \in L^p(\Omega)$. By choosing $u = v^{q-1} \in L^p(\Omega)$, and using Hölder's inequality and the assumption (A2), we have

$$\begin{aligned}
0 &= \int_\Omega (v^{q-1} - \alpha \mathcal{P}v^{q-1}) v d\mu = \int_\Omega v^q d\mu - \alpha \int_\Omega (\mathcal{P}v^{q-1}) v d\mu \\
&\geq \|v\|_{L^q}^q - \alpha \|\mathcal{P}v^{q-1}\|_{L^p} \|v\|_{L^q} \geq \|v\|_{L^q}^q - \alpha \|v^{q-1}\|_{L^p} \|v\|_{L^q} \\
&= \|v\|_{L^q}^q - \alpha \|v\|_{L^q}^{q/p} \|v\|_{L^q} = (1 - \alpha)\|v\|_{L^q}^q.
\end{aligned}$$

Hence, $\|v\|_{L^q} = 0$, i.e., $v = 0$. □

C.2 Equivalent form of RVPINNs loss function

For $u_\theta \in \mathcal{M}_n \subset L^p(\Omega)$ and $v_M \in V_M \subset L^q(\Omega)$, we define

$$r(u_\theta, v_M) := l(v_M) - b(u_\theta, v_M).$$

Suppose that $\tilde{g}_\theta \in V_M$ is the solution of the following problem:

$$\langle J_M(\tilde{g}_\theta), v_M \rangle_{L^p, L^q} = r(u_\theta, v_M), \quad \text{for all } v_M \in V_M, \tag{C.1}$$

where $\langle \cdot, \cdot \rangle_{L^p, L^q}$ denotes the duality pairing and J_M is the duality map on a subspace V_M defined by $J_M(g) = \|g\|_{L^q}^{2-q} |g|^{q-1} \text{sign}(g)$ (see [77]). The following expressions are equivalent:

1. $\sup_{0\neq v_M\in V_M} \dfrac{l(v_M)-b(u_\theta,v_M)}{\|v_M\|_{L^q}}$,
2. $\sup_{0\neq v_M\in V_M} \dfrac{b(u-u_\theta,v_M)}{\|v_M\|_{L^q}}$, where u is the solution of problem (4.5),
3. $\|J_M(\tilde{g}_\theta)\|_{L^p}$.

Indeed, the equivalence of 1. and 2. follows from $b(u,v_M)=l(v_M)\ \forall v_M\in V_M$ and the linearity of b. The equivalence of 1. and 3. follows from

$$\begin{aligned}\sup_{0\neq v_M\in V_M} \frac{l(v_M)-b(u_\theta,v_M)}{\|v_M\|_{L^q}} &= \sup_{0\neq v_M\in V_M} \frac{r(u_\theta,v_M)}{\|v_M\|_{L^q}}\\ &= \sup_{0\neq v_M\in V_M} \frac{\langle J_M(\tilde{g}_\theta),v_M\rangle_{L^p,L^q}}{\|v_M\|_{L^q}} = \|J_M(\tilde{g}_\theta)\|_{L^p},\end{aligned}$$

where the second equality follows from (C.1) and the third one from the fact that $v_M=J_M(\tilde{g}_\theta)$ is the supremizer.

Thus, the RVPINNs loss function can be written in three different forms:

1. $\mathcal{L}_{\text{RVPINNs}}(u_\theta)=\sup_{0\neq v_M\in V_M} \dfrac{l(v_M)-b(u_\theta,v_M)}{\|v_M\|_{L^q}}$,
2. $\mathcal{L}_{\text{RVPINNs}}(u_\theta)=\sup_{0\neq v_M\in V_M} \dfrac{b(u-u_\theta,v_M)}{\|v_M\|_{L^q}}$,
3. $\mathcal{L}_{\text{RVPINNs}}(u_\theta)=\|J_M(\tilde{g}_\theta)\|_{L^p}$.

Now, consider the case where u is the solution of the variational problem (4.5) with $U=V=L^2(\Omega)$, and $u_\theta\in\mathcal{M}_n\subset L^2(\Omega)$. Let V_M be a finite-dimensional subspace of $L^2(\Omega)$ with an orthonormal basis $\{g_1,\dots,g_M\}$. In this setting, the duality pairing in (C.1) corresponds to the L^2-inner product, and the duality map $J_M(\tilde{g}_\theta)$ is the Riesz representation in V_M of $r(u_\theta,\cdot)$. Suppose that $g_\theta\in V_M$ is the solution of the following Galerkin problem:

$$\langle g_\theta,v_M\rangle_{L^2}=r(u_\theta,v_M),\quad \text{for all } v_M\in V_M.$$

Since $g_\theta\in V_M$, it can be written as

$$g_\theta=\sum_{m=1}^{M}\alpha_m g_m,\ \text{where } \alpha_m\in\mathbb{R}, m=1,2,\dots,M. \tag{C.2}$$

For each $m\in\{1,2,\dots,M\}$, we compute the coefficients α_m by

$$\alpha_m=\langle g_\theta,g_m\rangle_{L^2}=r(u_\theta,g_m).$$

Substituting these into (C.2), we obtain

$$g_\theta = \sum_{m=1}^{M} r(u_\theta, g_m) g_m.$$

Taking the L^2-norm on both sides, we have

$$\|g_\theta\|_{L^2} = \sqrt{\sum_{m=1}^{M} r^2(u_\theta, g_m)}.$$

Thus, the RVPINNs loss function for solutions in $L^2(\Omega)$ is

$$\mathcal{L}_{\text{RVPINNs}}(u_\theta) = \sqrt{\sum_{m=1}^{M} \left(\int_\Omega (f_0 - u_\theta + \alpha \mathcal{P} u_\theta) g_m d\mu \right)^2}.$$

C.3 Error estimate proofs for PINNs and RVPINNs

Proof of Theorem 4.4.1

For arbitrary $u_\theta \in \mathcal{M}_n$, by using the assumption (A2), the reverse triangle inequality, the definition of a quasi-minimizer, and the triangle inequality, we have

$$\begin{aligned}
(1-\alpha)\|u - u_{\theta^*}\|_{L^p} &\leq \|u - u_{\theta^*}\|_{L^p} - \alpha \|\mathcal{P}(u - u_{\theta^*})\|_{L^p} \\
&\leq \|(u - u_{\theta^*}) - \alpha \mathcal{P}(u - u_{\theta^*})\|_{L^p} \\
&= \|f_0 - u_{\theta^*} + \alpha \mathcal{P} u_{\theta^*}\|_{L^p} \\
&\leq \|f_0 - u_\theta + \alpha \mathcal{P} u_\theta\|_{L^p} + \delta_n \\
&= \|(u - u_\theta) - \alpha \mathcal{P}(u - u_\theta)\|_{L^p} + \delta_n \\
&\leq \|u - u_{\theta^*}\|_{L^p} + \alpha \|\mathcal{P}(u - u_{\theta^*})\|_{L^p} + \delta_n \\
&\leq (1+\alpha)\|u - u_\theta\|_{L^p} + \delta_n.
\end{aligned}$$

Dividing by $1 - \alpha$ and taking the infimum over all possible $u_\theta \in \mathcal{M}_n$ yields the theorem.

In RVPINNs, the local Fortin's condition provides the following result, which is necessary for proving Theorem 4.4.2.

Lemma C.3.1. *Let $\delta_n > 0$ and $u_{\theta^*} \in \mathcal{M}_n$ be a quasi-minimizer of (4.12) satisfying (2.9). If the local Fortin's condition is satisfied, it holds:*

$$\|u_{\theta^*} - u_\theta\|_{L^p} \leq \frac{C_\Pi}{1-\alpha} \left(\sup_{0 \neq v_M \in V_M} \frac{b(u_{\theta^*} - u_\theta, v_M)}{\|v_M\|_{L^q}} \right), \quad \forall u_\theta \in \mathcal{M}_n^{\theta^*, R}.$$

Proof. By using the local Fortin's condition and the inf-sup stability in Theorem C.1.1, we have

$$\begin{aligned}\sup_{0\neq v_M\in V_M}\frac{b(u_{\theta^*}-u_\theta,v_M)}{\|v_M\|_{L^q}} &\geq \sup_{0\neq v\in L^q(\Omega)}\frac{b(u_{\theta^*}-u_\theta,\Pi_\theta v)}{\|\Pi_\theta v\|_{L^q}}\\ &= \sup_{0\neq v\in L^q(\Omega)}\frac{b(u_{\theta^*}-u_\theta,v)}{\|\Pi_\theta v\|_{L^q}}\\ &\geq \frac{1}{C_\Pi}\sup_{0\neq v\in L^q(\Omega)}\frac{b(u_{\theta^*}-u_\theta,v)}{\|v\|_{L^q}}\\ &\geq \frac{1-\alpha}{C_\Pi}\|u_{\theta^*}-u_\theta\|_{L^p}.\end{aligned}$$

Multiplying both sides by $\dfrac{C_\Pi}{1-\alpha}$ yields the lemma. □

Proof of Theorem 4.4.2

For any $u_\theta\in\mathcal{M}_n^{\theta^*,R}$, by using the triangle inequality, Lemma C.3.1, the definition of a quasi-minimizer, and the boundedness of $b(\cdot,\cdot)$, we have

$$\begin{aligned}\|u-u_{\theta^*}\|_{L^p} &\leq \|u_{\theta^*}-u_\theta\|_{L^p}+\|u-u_\theta\|_{L^p}\\ &\leq \frac{C_\Pi}{1-\alpha}\left(\sup_{0\neq v_M\in V_M}\frac{b(u_{\theta^*}-u_\theta,v_M)}{\|v_M\|_{L^q}}\right)+\|u-u_\theta\|_{L^p}\\ &\leq \frac{C_\Pi}{1-\alpha}\left(\sup_{0\neq v_M\in V_M}\frac{b(u-u_\theta,v_M)}{\|v_M\|_{L^q}}+\sup_{0\neq v_M\in V_M}\frac{b(u-u_{\theta^*},v_M)}{\|v_M\|_{L^q}}\right)\\ &\quad+\|u-u_\theta\|_{L^p}\\ &\leq \frac{C_\Pi}{1-\alpha}\left(2\sup_{0\neq v_M\in V_M}\frac{b(u-u_\theta,v_M)}{\|v_M\|_{L^q}}+\delta_n\right)+\|u-u_\theta\|_{L^p}\\ &\leq \frac{2C_\Pi}{1-\alpha}(1+\alpha)\|u-u_\theta\|_{L^p}+\left(\frac{C_\Pi}{1-\alpha}\right)\delta_n+\|u-u_\theta\|_{L^p}\\ &= \left(1+\frac{2C_\Pi}{1-\alpha}(1+\alpha)\right)\|u-u_\theta\|_{L^p}+\left(\frac{C_\Pi}{1-\alpha}\right)\delta_n.\end{aligned}$$

Taking the infimum over all possible $u_\theta\in\mathcal{M}_n^{\theta^*,R}$ yields the theorem.

Appendix D

Additional Results for Neumann Series Problems

To approximate the solution $u \in L^2(\Omega)$ of (4.3) using the neural network u_θ, we can consider the PINNs loss function in the form of the L^2-norm squared of the residual:

$$\mathcal{L}(u_\theta) := \int_\Omega \left(u_\theta - \alpha \mathcal{P} u_\theta - f_0\right)^2 d\mu. \tag{D.1}$$

Then we initialize weights and biases of a two-layer neural network with ReLU activation function before the training process.

D.1 Initializing weights and biases

For one-dimensional problems, we start by considering a two-layer neural network, N_n neurons, and the ReLU activation function, i.e.,

$$u_{\boldsymbol{\theta}}(x) = \sum_{j=1}^{N_n} c^j \mathrm{ReLU}(w^j x + b^j) =: \sum_{j=1}^{N_n} c^j \phi^j(x). \tag{D.2}$$

We initialize the weights and biases of the hidden layer as follows: $w^j = 1$, and $b^j = -j/(N+1)$, for $j = 1, 2, \ldots, N_n$. To initialize c^j, we solve the equation $\frac{\partial \mathcal{L}}{\partial c^j} = 0$,

which leads to the following steps:

$$2\int_\Omega (u_{\boldsymbol{\theta}} - \alpha\mathcal{P}u_\theta - f_0)\left(\frac{\partial}{\partial c^i}u_{\boldsymbol{\theta}} - \alpha\frac{\partial}{\partial c^i}\mathcal{P}u_{\boldsymbol{\theta}}\right)d\mu = 0$$

$$\int_\Omega \left(\sum_{j=1}^{N_n}(c^j\phi^j - c^j\alpha\mathcal{P}\phi^j) - f_0\right)(\phi^i - \alpha\mathcal{P}\phi^i)d\mu = 0$$

$$\sum_{j=1}^{N_n} c^j\int_\Omega(\phi^j - \alpha\mathcal{P}\phi^j)(\phi^i - \alpha\mathcal{P}\phi^i)d\mu = \int_\Omega(\phi^i - \alpha\mathcal{P}\phi^i)f_0 d\mu.$$

This is equivalent to the matrix equation $A\mathbf{c} = \mathbf{d}$ where,

$$A_{ij} = \int_\Omega(\phi^j - \alpha\mathcal{P}\phi^j)(\phi^i - \alpha\mathcal{P}\phi^i)d\mu$$

$$d_i = \int_\Omega(\phi^i - \alpha\mathcal{P}\phi^i)f_0 d\mu.$$

For two-dimensional problems, we consider a two-layer neural network, $N_n = 2N$ neurons, and ReLU activation function, given by:

$$u_{\boldsymbol{\theta}}(x,y) = \sum_{j=1}^{2N} c^j\mathrm{ReLU}(w_1^j x + w_2^j y + b^j) =: \sum_{j=1}^{2N} c^j\phi^j(x,y). \tag{D.3}$$

Consider a rectangular domain $x_{\min} < x < x_{\max}$, $y_{\min} < y < y_{\max}$, with lengths $L_x = x_{\max} - x_{\min}$ and $L_y = y_{\max} - y_{\min}$. We initialize the weights and biases of the hidden layer to generate a mesh on the domain. For $i = 1, 2, \ldots, N$, we set $w_1^i = \frac{L_y}{L_x}, w_2^i = -1$, and $b^i = y_{\max} - \frac{2L_y \cdot i}{N+1}$. For $i = N+1, N+2, \ldots, 2N$, we set $w_1^i = \frac{L_y}{L_x}, w_2^i = 1$, and $b^i = -y_{\min} - \frac{2L_y \cdot (i-N)}{N+1}$. To initialize c^i, we solve the equation $\frac{\partial\mathcal{L}}{\partial c^i} = 0$ in the same manner as described above.

D.2 Analysis

We now proceed with the analysis of the loss function (D.1). We will verify that it is Lipschitz continuous and strongly convex. This allows us to establish an error estimate for a quasi-minimizer of this loss function.

Theorem D.2.1. *Let $\mathcal{M}_n \subset L^2(\Omega)$ be the set of neural networks of n parameters. Let $\mathcal{P}$ be the Perron-Frobenius operator associated with a map $S : \Omega \to \Omega$ such that S is invertible and $\|\mathcal{P}f\|_{L^2} \leq \|f\|_{L^2} \quad \forall f \in L^2(\Omega)$, and let $f_0 \in L^2(\Omega)$ be an initial density. Assume that $0 < \alpha < 1$. The loss function $\mathcal{L} : L^2(\Omega) \to \mathbb{R}$ is defined by:*

$$\mathcal{L}(u_\theta) := \int_\Omega (u_\theta - \alpha\mathcal{P}u_\theta - f_0)^2\, d\mu.$$

Then the assumptions of Theorem 2.6.1(Lipschitz continuity and strong convexity) hold true.

Proof. The Gâteaux derivative of $\mathcal{L}$ at u is computed by

$$\begin{aligned}
\mathcal{L}'_u(v) &= \lim_{s\to 0} \frac{\mathcal{L}(u+sv)-\mathcal{L}(u)}{s} \\
&= \lim_{s\to 0} \frac{\int_\Omega \left(u+sv-\alpha\mathcal{P}(u+sv)-f_0\right)^2 d\mu - \int_\Omega \left(u-\alpha\mathcal{P}u-f_0\right)^2 d\mu}{s} \\
&= \lim_{s\to 0} \frac{\int_\Omega \left(2u+sv-2\alpha\mathcal{P}u-\alpha s\mathcal{P}v-2f_0\right)\left(sv-\alpha s\mathcal{P}v\right) d\mu}{s} \\
&= 2\int_\Omega \left(u-\alpha\mathcal{P}u-f_0\right)\left(v-\alpha\mathcal{P}v\right) d\mu.
\end{aligned}$$

Lipschitz continuity: For all $u_1, u_2, v \in L^2(\Omega)$, using Cauchy-Schwarz inequality and Proposition A.2.1, we obtain

$$\begin{aligned}
&\left|\int_\Omega \left((u_1-u_2)-\alpha\mathcal{P}(u_1-u_2)\right)\left(v-\alpha\mathcal{P}v\right) d\mu\right| \\
&\quad\le \|(u_1-u_2)-\alpha\mathcal{P}(u_1-u_2)\|_{L^2}\|v-\alpha\mathcal{P}v\|_{L^2} \\
&\quad\le \left(\|u_1-u_2\|_{L^2}+\alpha\|\mathcal{P}(u_1-u_2)\|_{L^2}\right)\left(\|v\|_{L^2}+\alpha\|\mathcal{P}v\|_{L^2}\right) \\
&\quad\le (1+\alpha)^2\|u_1-u_2\|_{L^2}\|v\|_{L^2}.
\end{aligned}$$

So,

$$\begin{aligned}
\|\mathcal{L}'_{u_1}-\mathcal{L}'_{u_2}\|_{(L^2)^*} &= \sup_{\|v\|_{L^2}=1}\left|2\int_\Omega \left((u_1-u_2)-\alpha\mathcal{P}(u_1-u_2)\right)\left(v-\alpha\mathcal{P}v\right) d\mu\right| \\
&\le 2(1+\alpha)^2\|u_1-u_2\|_{L^2}.
\end{aligned}$$

Strong convexity: For all $u_1, u_2 \in L^2(\Omega)$, using the fact that $\|u-v\|_{L^2} \ge \|u\|_{L^2}-\|v\|_{L^2}$, we obtain

$$\begin{aligned}
\left\langle \mathcal{L}'_{u_1}-\mathcal{L}'_{u_2}, u_1-u_2\right\rangle_{(L^2)^*,L^2} &= 2\int_\Omega \left((u_1-u_2)-\alpha\mathcal{P}(u_1-u_2)\right)\left((u_1-u_2)-\alpha\mathcal{P}(u_1-u_2)\right) d\mu \\
&= 2\|(u_1-u_2)-\alpha\mathcal{P}(u_1-u_2)\|_{L^2}^2 \\
&\ge 2(1-\alpha)^2\|(u_1-u_2)\|_{L^2}^2.
\end{aligned}$$

Thus, the loss function $\mathcal{L}$ is Lipschitz continuous and strongly convex. □

Corollary D.2.1. *Under the conditions of Theorem D.2.1, statements 1-3 of Theorem 2.6.1 hold true with* $C = 2(1+\alpha)^2$ *and* $\gamma = 2(1-\alpha)^2$.

Proof. The results of Theorem D.2.1 are the assumptions of Theorem 2.6.1. □

Bibliography

[1] M. Ainsworth and J. T. Oden. *A Posteriori Error Estimation in Finite Element Analysis*. John Wiley & Sons, 2000. URL: https://doi.org/10.1002/9781118032824. 44, 86, 98

[2] Z. Aldirany, R. Cottereau, M. Laforest, and S. Prudhomme. Multi-level neural networks for accurate solutions of boundary-value problems. *Computer Methods in Applied Mechanics and Engineering*, 419:116666, 2024. URL: https://doi.org/10.1016/j.cma.2023.116666. 84

[3] E. A. Antonelo, E. Camponogara, L. O. Seman, J. P. Jordanou, E. R. de Souza, and J. F. Hübner. Physics-informed neural nets for control of dynamical systems. *Neurocomputing*, 579:127419, 2024. URL: https://doi.org/10.1016/j.neucom.2024.127419. 38

[4] J. Bajars, D. J. Chappell, T. Hartmann, and G. Tanner. Improved approximation of phase-space densities on triangulated domains using discrete flow mapping with p-refinement. *Journal of Scientific Computing*, 72(3):1290–1312, 2017. URL: https://doi.org/10.1007/s10915-017-0397-8. 59

[5] J. Bajars, D. J. Chappell, N. Søndergaard, and G. Tanner. Transport of phase space densities through tetrahedral meshes using discrete flow mapping. *Journal of Computational Physics*, 328:95–108, 2017. URL: https://doi.org/10.1016/j.jcp.2016.10.019. 21

[6] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind. Automatic differentiation in machine learning: a survey. *Journal of Marchine Learning Research*, 18:1–43, 2018. URL: https://www.jmlr.org/papers/volume18/17-468/17-468.pdf. 8, 75, 127

[7] J. Berg and K. Nyström. Neural networks as smooth priors for inverse problems for PDEs. *Journal of Computational Mathematics and Data Science*, 1:100008, 2021. URL: https://doi.org/10.1016/j.jcmds.2021.100008. 1, 12

[8] S. Berrone, C. Canuto, and M. Pintore. Solving PDEs by variational physics-informed neural networks: an a posteriori error analysis. *ANNALI DELL'UNIVERSITA'DI FERRARA*, 68(2):575–595, 2022. URL: https://doi.org/10.1007/s11565-022-00441-6. 38

[9] S. Berrone, C. Canuto, and M. Pintore. Variational physics informed neural networks: the role of quadratures and test functions. *Journal of Scientific Computing*, 92(3):100, 2022. URL: https://doi.org/10.1007/s10915-022-01950-4. 38, 85

[10] S. Berrone and M. Pintore. Meshfree variational-physics-informed neural networks (mf-vpinn): An adaptive training strategy. *Algorithms*, 17(9):415, 2024. URL: https://doi.org/10.3390/a17090415. 85

[11] P. Biswas, H. Shimoyama, and L. R. Mead. Lyapunov exponents and the natural invariant density determination of chaotic maps: an iterative maximum entropy ansatz. *Journal of Physics A: Mathematical and Theoretical*, 43(12):125103, 2010. URL: https://doi.org/10.1088/1751-8113/43/12/125103. 15

[12] D. Boffi, F. Brezzi, M. Fortin, et al. *Mixed finite element methods and applications*, volume 44. Springer, 2013. 123, 126

[13] C. Bose and R. Murray. The exact rate of approximation in Ulam's method. *Discrete and continuous dynamical systems*, 7(1):219–235, 2000. URL: https://doi.org/10.3934/dcds.2001.7.219. 38

[14] C. Bose and R. Murray. Minimum 'energy' approximations of invariant measures for nonsingular transformations. *Discrete and Continuous Dynamical Systems*, 14(3):597–615, 2005. URL: https://doi.org/10.3934/dcds.2006.14.597. 34

[15] C. J. Bose and R. Murray. Duality and the computation of approximate invariant densities for nonsingular transformations. *SIAM Journal on Optimization*, 18(2):691–709, 2007. URL: https://doi.org/10.1137/060658163. 34

[16] S. C. Brenner and L. R. Scott. *The mathematical theory of finite element methods*. Springer, 2008. 122, 123

[17] I. Brevis, I. Muga, and K. G. van der Zee. Neural control of discrete weak formulations: Galerkin, least squares & minimal-residual methods with quasi-optimal weights. *Computer Methods in Applied Mechanics and Engineering*, 402:115716, 2022. URL: https://doi.org/https://doi.org/10.1016/j.cma.2022.115716. 1, 12, 13, 14, 38, 39

[18] H. Brezis and H. Brézis. *Functional analysis, Sobolev spaces and partial differential equations*, volume 2. Springer, 2011. URL: https://doi.org/10.1007/978-0-387-70914-7. 88

[19] S. L. Brunton, B. R. Noack, and P. Koumoutsakos. Machine learning for fluid mechanics. *Annual review of fluid mechanics*, 52(1):477–508, 2020. URL: https://doi.org/10.1146/annurev-fluid-010719-060214. 84

[20] S. Cai, Z. Mao, Z. Wang, M. Yin, and G. E. Karniadakis. Physics-informed neural networks (PINNs) for fluid mechanics: A review. *Acta Mechanica Sinica*, 37(12):1727–1738, 2021. URL: https://doi.org/10.1007/s10409-021-01148-1. 5, 38, 84

[21] Z. Cai, J. Chen, M. Liu, and X. Liu. Deep least-squares methods: An unsupervised learning-based numerical method for solving elliptic PDEs. *Journal of Computational Physics*, 420:109707, 2020. URL: https://doi.org/10.1016/j.jcp.2020.109707. 38

[22] Z. Cai, A. Doktorova, R. D. Falgout, and C. Herrera. Fast iterative solver for neural network method: I. 1d diffusion problems. *arXiv preprint arXiv:2404.17750*, 2024. URL: https://arxiv.org/pdf/2404.17750v1. 51

[23] J. M. Cascon, C. Kreuzer, R. H. Nochetto, and K. G. Siebert. Quasi-optimal convergence rate for an adaptive finite element method. *SIAM Journal on Numerical Analysis*, 46(5):2524–2550, 2008. URL: https://doi.org/10.1137/07069047X. 105

[24] A. Cauchy et al. Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, 25(1847):536–538, 1847. URL: https://cs.uwaterloo.ca/~y328yu/classics/cauchy-en.pdf. 9

[25] K.-S. Chan and H. Tong. *Chaos: a statistical perspective*. Springer Science & Business Media, 2013. 2

[26] D. Chappell, S. Giani, and G. Tanner. Dynamical energy analysis for built-up acoustic systems at high frequencies. *The Journal of the Acoustical Society of America*, 130(3):1420–1429, 2011. URL: https://doi.org/10.1121/1.3621041. 59

[27] D. Chappell, M. Richter, G. Tanner, O. Bandtlow, W. Just, and J. Slipantschuk. Ray-tracing the Ulam way. In *International Conference on Integral Methods in Science and Engineering*, pages 95–101. Springer, 2022. URL: https://doi.org/10.1007/978-3-031-34099-4_8. 30

[28] Y. Chen, L. Lu, G. E. Karniadakis, and L. Dal Negro. Physics-informed neural networks for inverse problems in nano-optics and metamaterials. *Optics express*, 28(8):11618–11633, 2020. URL: https://doi.org/10.1364/OE.384875. 84

[29] A. Cohen, R. DeVore, and R. H. Nochetto. Convergence rates of AFEM with H-1 data. *Foundations of Computational Mathematics*, 12(5):671–718, 2012. URL: https://doi.org/10.1007/s10208-012-9120-1. 100, 126

[30] D. L. Cohn. *Measure theory*, volume 1. Springer, 2013. URL: https://doi.org/10.1007/978-1-4614-6956-8. 120

[31] R. Collobert and J. Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167, 2008. URL: https://doi.org/10.1145/1390156.1390177. 5

[32] S. Cuomo, V. S. Di Cola, F. Giampaolo, G. Rozza, M. Raissi, and F. Piccialli. Scientific machine learning through physics–informed neural networks: Where we are and what's next. *Journal of Scientific Computing*, 92(3):88, 2022. URL: https://doi.org/10.1007/s10915-022-01939-z. 38

[33] P. Cvitanovic, R. Artuso, R. Mainieri, G. Tanner, and G. Vattay. *Chaos: classical and quantum*. 2025. URL: https://chaosbook.org/. 59

[34] L. C. de Souza, M. R. Sales, J. D. Szezech Jr, R. L. Viana, I. L. Caldas, and M. da Silva Baptista. Pattern formation in symplectic coupled map lattices. *Chaos, Solitons & Fractals*, 200:117057, 2025. URL: https://doi.org/10.1016/j.chaos.2025.117057. 62

[35] L. Demkowicz. Babuška $\Longleftrightarrow$ Brezzi. *ICES Report*, pages 06–08, 2006. URL: https://www.oden.utexas.edu/media/reports/2006/0608.pdf. 88

[36] R. A. DeVore. Nonlinear approximation. *Acta numerica*, 7:51–150, 1998. URL: https://doi.org/10.1017/S0962492900002816. 47, 51

[37] J. Ding. A maximum entropy method for solving Frobenius-Perron operator equations. *Applied mathematics and computation*, 93(2-3):155–168, 1998. URL: https://doi.org/10.1016/S0096-3003(97)10061-3. 15, 28, 33, 34, 65, 76

[38] J. Ding, C. Jin, N. H. Rhee, and A. Zhou. A maximum entropy method based on piecewise linear functions for the recovery of a stationary density of interval mappings. *Journal of Statistical Physics*, 145:1620–1639, 2011. URL: https://doi.org/10.1007/s10955-011-0366-9. 15

[39] J. Ding and N. H. Rhee. A maximum entropy method based on orthogonal polynomials for Frobenius-Perron operators. *Advances in Applied Mathematics and Mechanics*, 3(2):204–218, 2011. URL: https://doi.org/10.4208/aamm.10-m1022. 15, 34, 65

[40] J. Ding and A. Zhou. *Statistical properties of deterministic systems*. Springer Science & Business Media, 2009. URL: https://doi.org/10.1007/978-3-540-85367-1. 2, 27, 30

[41] W. Dörfler. A convergent adaptive algorithm for Poisson's equation. *SIAM Journal on Numerical Analysis*, 33(3):1106–1124, 1996. URL: https://doi.org/10.1137/0733054. 101

[42] L. C. Evans. *Partial differential equations*, volume 19. American mathematical society, 2010. 31, 84, 103

[43] R. R. Faria, B. Capron, A. R. Secchi, and M. B. de Souza Jr. A data-driven tracking control framework using physics-informed neural networks and deep reinforcement learning for dynamical systems. *Engineering Applications of Artificial Intelligence*, 127:107256, 2024. URL: https://doi.org/10.1016/j.engappai.2023.107256. 38

[44] G. Froyland, R. M. Stuart, and E. van Sebille. How well-connected is the surface of the global ocean? *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 24(3), 2014. URL: https://doi.org/10.1063/1.4892530. 2, 37

[45] T. Führer and S. Rojas. A posteriori analysis of neural network approximations. *arXiv preprint arXiv:2507.06017*, 2025. URL: https://doi.org/10.48550/arXiv.2507.06017. 84, 85, 86, 91, 99

[46] S. Galatolo, M. Hoyrup, and C. Rojas. Statistical properties of dynamical systems–simulation and abstract computation. *Chaos, Solitons & Fractals*, 45(1):1–14, 2012. URL: https://doi-org.nottingham.idm.oclc.org/10.1016/j.chaos.2011.09.011. 2

[47] Z. Gao, T. Tang, L. Yan, and T. Zhou. Failure-informed adaptive sampling for PINNs, part ii: combining with re-sampling and subset simulation. *Communications on Applied Mathematics and Computation*, 6(3):1720–1741, 2024. URL: https://doi.org/10.1007/s42967-023-00312-7. 85

[48] Z. Gao, L. Yan, and T. Zhou. Failure-informed adaptive sampling for PINNs. *SIAM Journal on Scientific Computing*, 45(4):A1971–A1994, 2023. URL: https://doi.org/10.1137/22M1527763. 85

[49] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. URL: http://www.deeplearningbook.org. 7

[50] I. Gühring, M. Raslan, and G. Kutyniok. *Expressivity of Deep Neural Networks*, page 149–199. Cambridge University Press, 2022. URL: https://doi.org/10.1017/9781009025096.004. 1, 5, 38, 65

[51] T. Hartmann, S. Morita, G. Tanner, and D. J. Chappell. High-frequency structure-and air-borne sound transmission for a tractor model using dynamical

energy analysis. *Wave Motion*, 87:132–150, 2019. URL: https://doi.org/10.1016/j.wavemoti.2018.09.012. 2, 37

[52] C. F. Higham and D. J. Higham. Deep learning: An introduction for applied mathematicians. *Siam review*, 61(4):860–891, 2019. URL: https://doi.org/10.1137/18M1165748. 5, 7, 9

[53] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012. URL: https://doi.org/10.1109/MSP.2012.2205597. 5

[54] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989. URL: https://doi.org/10.1016/0893-6080(89)90020-8. 5, 7

[55] X. Hu, G. Li, J. Li, R. C. Sau, and Z. Zhou. An adaptive error estimator for stationary navier-stokes equations using variational physics informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 455:118876, 2026. URL: https://doi.org/10.1016/j.cma.2026.118876. 85

[56] M. S. Islam and A. Smith. A linear spline maximum entropy method for Frobenius-Perron operators of multi-dimensional transformations. *International Journal of Applied and Computational Mathematics*, 8(4):180, 2022. URL: https://doi.org/10.1007/s40819-022-01386-2. 15, 28

[57] E. Kaiser, J. N. Kutz, and S. L. Brunton. Data-driven approximations of dynamical systems operators for control. *The Koopman operator in systems and control: concepts, methodologies, and applications*, pages 197–234, 2020. URL: https://doi.org/10.1007/978-3-030-35713-9_8. 37

[58] E. Kharazmi, Z. Zhang, and G. E. Karniadakis. Variational physics-informed neural networks for solving partial differential equations. *arXiv preprint arXiv:1912.00873*, 2019. URL: https://doi.org/10.48550/arXiv.1912.00873. 1, 5, 11, 38, 84

[59] E. Kharazmi, Z. Zhang, and G. E. Karniadakis. hp-VPINNs: Variational physics-informed neural networks with domain decomposition. *Computer Methods in Applied Mechanics and Engineering*, 374:113547, 2021. URL: https://doi.org/10.1016/j.cma.2020.113547. 5, 38, 84

[60] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. URL: https://arxiv.org/abs/1412.6980. 9, 91, 101

[61] S. Klus, P. Koltai, and C. Schütte. On the numerical approximation of the Perron-Frobenius and Koopman operator. *Journal of Computational Dynamics*, 3(1):51–79, 2016. URL: https://doi.org/10.3934/jcd.2016003. 20, 29, 30, 37, 65

[62] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012. URL: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf. 5

[63] A. Lasota and M. C. Mackey. *Probabilistic properties of deterministic systems*. Cambridge university press, 1985. 27

[64] A. Lasota and M. C. Mackey. *Chaos, fractals, and noise: stochastic aspects of dynamics*, volume 97. Springer Science & Business Media, 1994. URL: https://doi.org/10.1007/978-1-4612-4286-4. 2, 19, 20, 21, 27, 28, 34, 41, 119, 120

[65] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867, 1993. URL: https://doi.org/10.1016/S0893-6080(05)80131-5. 5, 7, 66, 68

[66] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2021. URL: https://arxiv.org/pdf/2010.08895v3. 1

[67] M. W. Libbrecht and W. S. Noble. Machine learning applications in genetics and genomics. *Nature Reviews Genetics*, 16(6):321–332, 2015. URL: https://doi.org/10.1038/nrg3920. 84

[68] K. Linka, A. Schäfer, X. Meng, Z. Zou, G. E. Karniadakis, and E. Kuhl. Bayesian physics informed neural networks for real-world nonlinear dynamical systems. *Computer Methods in Applied Mechanics and Engineering*, 402:115346, 2022. URL: https://doi.org/10.1016/j.cma.2022.115346. 38

[69] M. Liu and Z. Cai. Adaptive two-layer ReLU neural network: II. Ritz approximation to elliptic pdes. *Computers & Mathematics with Applications*, 113:103–116, 2022. URL: https://doi.org/10.1016/j.camwa.2022.03.010. 49

[70] M. Liu, Z. Cai, and J. Chen. Adaptive two-layer ReLU neural network: I. best least-squares approximation. *Computers & Mathematics with Applications*, 113:34–44, 2022. URL: https://doi.org/10.1016/j.camwa.2022.03.005. 49

[71] G. Livadiotis. Invariant spectra in n-coupled standard maps. *International Journal of Bifurcation and Chaos*, 26(05):1650084, 2016. URL: https://doi.org/10.1142/S021812741650084X. 62

[72] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021. URL: https://doi.org/10.1038/s42256-021-00302-5. 1

[73] B. Lusch, J. N. Kutz, and S. L. Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature communications*, 9(1):4950, 2018. URL: https://doi.org/10.1038/s41467-018-07210-0. 38

[74] Z. Mao, A. D. Jagtap, and G. E. Karniadakis. Physics-informed neural networks for high-speed flows. *Computer Methods in Applied Mechanics and Engineering*, 360:112789, 2020. URL: https://doi.org/10.1016/j.cma.2019.112789. 84

[75] I. Muga, S. Rojas, and P. Vega. An adaptive superconvergent mixed finite element method based on local residual minimization. *SIAM Journal on Nu-*

merical Analysis, 61(5):2084–2105, 2023. URL: https://doi.org/10.1137/22M1526307. 104

[76] I. Muga, M. J. Tyler, and K. G. van der Zee. The discrete-dual minimal-residual method (DDMRes) for weak advection-reaction problems in Banach spaces. *Computational Methods in Applied Mathematics*, 19(3):557–579, 2019. URL: https://doi.org/10.1515/cmam-2018-0199. 51

[77] I. Muga and K. G. van der Zee. Discretization of linear problems in Banach spaces: Residual minimization, nonlinear Petrov–Galerkin, and monotone mixed methods. *SIAM Journal on Numerical Analysis*, 58(6):3406–3426, 2020. URL: https://doi.org/10.1137/20M1324338. 40, 131

[78] R. Murray. Ulam's method for some non-uniformly expanding maps. *Discrete. Contin. Dyn. Syst*, 26(3):1007–1018, 2010. URL: https://doi.org/10.3934/dcds.2010.26.1007. 30

[79] E. O. Oluwasakin and A. Q. Khaliq. Optimizing physics-informed neural network in dynamic system simulation and learning of parameters. *Algorithms*, 16(12):547, 2023. URL: https://doi.org/10.3390/a16120547. 38

[80] Á. J. Omella and D. Pardo. r-adaptive deep learning method for solving partial differential equations. *Computers & Mathematics with Applications*, 153:33–42, 2024. URL: https://doi.org/10.1016/j.camwa.2023.11.005. 1, 12

[81] D. Onken, L. Nurbekyan, X. Li, S. W. Fung, S. Osher, and L. Ruthotto. A neural network approach for high-dimensional optimal control applied to multiagent path finding. *IEEE Transactions on Control Systems Technology*, 31(1):235–251, 2022. URL: http://doi.org/10.1109/TCST.2022.3172872. 1, 5, 38

[82] E. Ott. *Chaos in dynamical systems*. Cambridge university press, 2002. 22

[83] P. Petersen, M. Raslan, and F. Voigtlaender. Topological properties of the set of functions generated by neural networks of fixed size. *Foundations of computational mathematics*, 21:375–444, 2021. URL: https://doi.org/10.1007/s10208-020-09461-0. 13, 96

[84] M. Raissi, P. Perdikaris, and G. 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. URL: https://doi.org/10.1016/j.jcp.2018.10.045. 1, 5, 10, 38, 84

[85] M. Rasht-Behesht, C. Huber, K. Shukla, and G. E. Karniadakis. Physics-informed neural networks (PINNs) for wave propagation and full waveform inversions. *Journal of Geophysical Research: Solid Earth*, 127(5):e2021JB023120, 2022. URL: https://doi.org/10.1029/2021JB023120. 84

[86] M. Richter, D. J. Chappell, and G. Tanner. Convergence of ray-based methods using transfer operators in different bases. In *Forum acousticum*, 2020. URL: http://doi.org/10.48465/fa.2020.0738. 2, 15, 37

[87] S. Rojas, P. Maczuga, J. Muñoz-Matute, D. Pardo, and M. Paszyński. Robust variational physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 425:116904, 2024. URL: https://doi.org/10.1016/j.cma.2024.116904. 1, 5, 11, 38, 39, 46, 85, 89, 93

[88] S. Saqlain, W. Zhu, E. G. Charalampidis, and P. G. Kevrekidis. Discovering governing equations in discrete systems using PINNs. *Communications in Nonlinear Science and Numerical Simulation*, 126:107498, 2023. URL: https://doi.org/10.1016/j.cnsns.2023.107498. 38

[89] K. Scheinberg. Finite difference gradient approximation: To randomize or not? *INFORMS Journal on Computing*, 34(5):2384–2388, 2022. URL: https://doi.org/10.1287/ijoc.2022.1218. 8

[90] H. G. Schuster and W. Just. *Deterministic chaos: an introduction*. John Wiley & Sons, 2006. 20, 129

[91] C. Schwab. *p and hp Finite Element Methods*. Oxford Science Publications, 1998. 32

[92] Y. G. Sinai. *Dynamical systems II: Ergodic theory with applications to dynamical systems and statistical mechanics*. Springer, 1989. 65

[93] J. Sirignano and K. Spiliopoulos. DGM: A deep learning algorithm for solving partial differential equations. *Journal of computational physics*, 375:1339–1364, 2018. URL: https://doi.org/10.1016/j.jcp.2018.08.029. 5, 38, 84

[94] J. Slipantschuk, M. Richter, D. J. Chappell, G. Tanner, W. Just, and O. F. Bandtlow. Transfer operator approach to ray-tracing in circular domains. *Nonlinearity*, 33(11):5773, 2020. URL: https://doi.org/10.1088/1361-6544/ab9dca. 2, 37

[95] K. Tang, X. Wan, and C. Yang. Das-pinns: A deep adaptive sampling method for solving high-dimensional partial differential equations. *Journal of Computational Physics*, 476:111868, 2023. URL: https://doi.org/10.1016/j.jcp.2022.111868. 85

[96] G. Tanner. Dynamical energy analysis—determining wave energy distributions in vibro-acoustical structures in the high-frequency regime. *Journal of Sound and Vibration*, 320(4-5):1023–1038, 2009. URL: https://doi.org/10.1016/j.jsv.2008.08.032. 18, 37

[97] A. Tantet, V. Lucarini, F. Lunkeit, and H. A. Dijkstra. Crisis of the chaotic attractor of a climate model: a transfer operator approach. *Nonlinearity*, 31(5):2221, 2018. URL: https://doi.org/10.1088/1361-6544/aaaf42. 2, 37

[98] A. Tantet, F. R. van der Burgt, and H. A. Dijkstra. An early warning indicator for atmospheric blocking events using transfer operators. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 25(3), 2015. URL: https://doi.org/10.1063/1.4908174. 2, 37

[99] G. H. Teichert, A. R. Natarajan, A. van der Ven, and K. Garikipati. Machine learning materials physics: Integrable deep neural networks enable scale bridging by learning free energy functions. *Computer Methods in Applied Mechanics and Engineering*, 353:201–216, 2019. URL: https://doi.org/10.1016/j.cma.2019.05.019. 1, 12

[100] T. Udomworarat, I. Brevis, M. Richter, S. Rojas, and K. G. van der Zee. Neural network methods for Neumann series problems of Perron-Frobenius operators.

Computer Methods in Applied Mechanics and Engineering, 454:118873, 2026. URL: https://doi.org/10.1016/j.cma.2026.118873. 4, 37, 84, 115

[101] S. M. Ulam. *A Collection of Mathematical problems*. Interscience Tracts in Pure and Applied Mathematics, 1960. 15, 38, 43

[102] J. F. Urbán, P. Stefanou, and J. A. Pons. Unveiling the optimization process of physics informed neural networks: How accurate and competitive can PINNs be? *Journal of Computational Physics*, 523:113656, 2025. URL: https://doi.org/10.1016/j.jcp.2024.113656. 56

[103] C. Uriarte, M. Bastidas, D. Pardo, J. M. Taylor, and S. Rojas. Optimizing variational physics-informed neural networks using least squares. *Computers & Mathematics with Applications*, 185:76–93, 2025. URL: https://doi.org/10.1016/j.camwa.2025.02.022. 46

[104] U. Vaidya, P. G. Mehta, and U. V. Shanbhag. Nonlinear stabilization via control Lyapunov measure. *IEEE Transactions on Automatic Control*, 55(6):1314–1328, 2010. URL: https://doi.org/10.1109/CDC.2007.4434956. 2, 37

[105] Z. Wang and Z. Zhang. A mesh-free method for interface problems using the deep learning approach. *Journal of Computational Physics*, 400:108963, 2020. URL: https://doi.org/10.1016/j.jcp.2019.108963. 103

[106] S. J. Wright. Numerical optimization, 2006. 9, 10

[107] B. Yu et al. The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6(1):1–12, 2018. URL: https://doi.org/10.1007/s40304-018-0127-z. 1, 5, 12, 38, 84

[108] R. Yu and R. Wang. Learning dynamical systems from data: An introduction to physics-guided deep learning. *Proceedings of the National Academy of Sciences*, 121(27):e2311808121, 2024. URL: https://doi.org/10.1073/pnas.2311808121. 38

[109] Z. Zhang, J. Li, and B. Liu. Annealed adaptive importance sampling method in PINNs for solving high dimensional partial differential equations. *Journal of*

Computational Physics, 521:113561, 2025. URL: https://doi.org/10.1016/j.jcp.2024.113561. 85